NASA Technical Memorandum 87646

NASA-TM-87646 19860007473

IPLIB (IMAGE PROCESSING LIBRARY) USER'S MANUAL

FOR REFERENCE

NETTIE D. FAULCON, JAMES H. MONTEITH, AND KEITH MILLER

DECEMBER 1985
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:

SEA %IPLIB::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
SEA,%IPLIB
SEA
DI
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.

%IPLIB 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) %IPLIB<CR>
(ENTER COMMAND NAMR) 1<CR>
(ENTER NAMR) %IPLIB<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 %IPLIB with the modified version, enter the following sequence: (HP prompts given within parenthesis.)

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

**ADD12**: adds 2 images, and requires a scaling factor (p. 11).
**ADDIM**: adds 2 images, no scaling factor (p. 15).
**BFDS**: transfers a monochrome image to COMTAL from disk (p. 19).
**CLFDS**: transfers a color image to COMTAL from disk (p. 22).
**CLGR**: clears a COMTAL graphics plane (p. 26).
**CLRM**: clears a monochrome COMTAL image (p. 28).
**CMND**: sends a COMTAL command given as a constant string from HP to COMTAL (p. 38).
**COUNT**: compiles 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).
**DSPBU**: 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).
**ICPY**: copies one monochrome COMTAL image to another (using CMND) (p. 57).
**ICPY2**: copies one monochrome COMTAL image to another (using RDILN & URILN) (p. 60).
**MERGE**: merges two bytes into one byte (all arguments are INTEGER) (p. 63).
**NORML**: 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).
**RDILN**: reads one horizontal line of a COMTAL graphics plane (p. 84).
**RDGPT**: reads one point from a COMTAL graphics plane (p. 86).
**RDIL2**: reads one horizontal line of COMTAL pixels; 1 pixel/integer returned (p. 89).
**RDLH**: reads one horizontal line of COMTAL pixels; 2 pixels/integer returned (p. 92).
**RDLT**: reads the contents of a COMTAL look-up table (p. 101).
**RDLUT**: reads the contents of a COMTAL monochrome image (p. 97).
**RDPAT**: reads the contents of a COMTAL pseudo-color table (p. 104).
**RDTAB**: reads the COMTAL Image/Graphics Table (p. 106).
RD TAR : 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).
THRSH : 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).
TST11 : 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).
WRGLN : writes a horizontal line of graphics bits to a COMTAL graphics plane (p. 139).
WRGPT : writes one graphics bit to a COMTAL graphics plane (p. 141).
WRILD : 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).
WRIPC : writes one pixel value to a COMTAL image memory (p. 150).
WRIRC : writes an array of integers to a rectangle of a COMTAL image memory;
one pixel value / integer in the buffer (p. 153).
WRLUT : 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).
TADD1 : program that tests ADDIM (p. 164).
TCLR : program that tests DSPCL, "DIsplay CoLor" (p. 165).
TCLRG : program that tests CLRGR, "CLeaR GrAPHics" (p. 166).
TCLR1 : program that tests CLRIM, "CLeaR ImAGe" (p. 167).
TCMM2 : program that sends all possible single bytes to COMTAL 1 at a time (p. 168).
TCMMN : program that sends COMTAL commands via the HP keyboard; tests CMND (p. 169).
TCHT : program that tests subroutine COUNT (p. 170).
TCNS : program that tests the string concatenation facility in HP FORTRAN 77 (p. 171).
TCOPY : program that tests ICOPY, "ImAGe COpy" (p. 172).
TDIGI : program that tests the function DIGIT (p. 173).
TDSP : program that tests DSPBU, "DiSpLAY Black & White" and DSPCL, "DiSpLAY
CoLor" (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 PROFL. "PROFiling" (p. 179).
TRANG : program that tests the function "RANGE" (p. 180).
TRDATA : program that tests RDTAR. "Read TARget" (p. 181).
TSETTV : program that tests SETV, "SET Video camera" (p. 182).
TSPRD : program that tests SPRED, "SPREAD pixel values" (p. 183).
TSUB1 : program that tests SUBIM and SUB12 (p. 184).
TTTHRS : program that tests THRSH. "THRESHolding" (p. 185).
TTSTI : program that tests TSTII, "TEST Image I" (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 *.

ADD12 calls:
* DIGIT CMND RANGE
ADDIM calls:
CMMND DIGIT RANGE

BUFDS calls:

CMMND DIGIT OPEN RANGE READF WRILN

CLFDS calls:

CMMND DIGIT OPEN RANGE READF WRILN

CLGRG calls:

CMMND DIGIT RANGE

CLRM calls:

CMMND DIGIT RANGE

CMMN2 calls:

*LEN

CMMND calls no other procedures.

COUNT calls:

RANGE RDIL2

DELAY calls no other procedures.

DIGIT calls:

RANGE

DSPBW calls:

RANGE

DSPCL calls:

CMMND DIGIT RANGE

DSPGR calls:

CMMN2 DIGIT RANGE

DSPVD calls:

CMMND DIGIT

HILO calls:

RDIL2

HISTO calls:

CMMN2 DELAY DIGIT RANGE

ICOPY calls:

CMMND DIGIT RANGE

ICPY2 calls:

CMMND DIGIT RANGE

MERGE calls no other procedures.

NORM calls:

HILO RDIL2 WRIL2

NOTE2 calls:

ADGR CMMN2 CMMND DELAY DIGIT *LEN RANGE

NOTES calls:

CMMN2 CMMND DELAY DIGIT DSPGR RANGE

PAINT calls:

CMMND RDTAR WAIT WRIRC

PROFL calls:

CHAR CMMND DIGIT RANGE WAIT

RANGE calls no other procedures.

RDGLN calls:
RANGE
RDGPT calls:
BTEST RANGE
RDIL2 calls:
*ICHAR RANGE
RDILN calls:
RANGE
RDIPR calls:
*ICHAR RANGE
RDIRC calls:
*ICHAR RANGE RDILN
RDLUT calls:
RANGE
RDPSU calls no other procedures.
RDTAB calls:
BTEST RANGE
RDTAR 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
TST11 calls:
WRILN
TV2C4 calls:
ADDI2 RANGE TV2CM
TV2CM calls:
CMND DIGIT DSPBU RANGE
WAIT calls no other procedures.
UIFGR calls:
CMND2 DIGIT RANGE
URGLN calls:
RANGE
URGPT calls:
*IBCLR *IBSET RANGE
WRIL2 calls:
CHAR RANGE
WRILN calls:
RANGE
Calls:

WRPT calls:
  RANGE
WRIRC calls:
  CHAR RANGE RDIL2 WRIL2
WRLUT calls:
  RANGE
WRPSU calls: no other procedures.
URTAR calls:
  RANGE

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

TADD2 calls:
  ADDI2 CMND DIGIT RANGE
TADDI calls:
  ADDIM CMND DIGIT RANGE
TCLR calls:
  DSPCL
TCLRG calls:
  CLRGR CMND DIGIT RANGE
TCLRI calls:
  CLRIM CMND DIGIT RANGE
TCMM2 calls:
  CMND WAIT
TCMMN calls:
  CMND
TCNT calls:
  COUNT RANGE RDIL2
TCONS calls: no other procedures.
TCOPY calls:
  CMND DIGIT ICOPY RANGE
TDELA calls:
  DELAY CMN2 WAIT
TDIGI calls:
  DIGIT RANGE
TDSP calls:
  CMND DSPBW DSPCL RANGE WAIT
TDSPV calls:
  DSPVD CMND DIGIT WAIT DSPBW RANGE
THIST calls:
  CMNN2 DIGIT HISTO RANGE
TNORM calls:
  HILO NORML RANGE RDIL2 WRIL2
TNOTE calls:
  CMNN2 CMND DELAY DIGIT DSPGR NOTE2 NOTES RANGE URTAR
TPNT calls:
CMMD PAINT RANGE RDILN RDTAR WRILN WRIRC
TPROF calls:
CMMD DIGIT PROFL RANGE WAIT
TRANG calls:
RANGE
TMDCG calls:
RANGE RDTAB
TR?SET calls no other procedures.
TSED calls:
SETV RANGE DIGIT CMNND DSPBW
TSPRD calls:
RANGE RDIL2 SPRED WRIL2
TSSRT calls no other procedures.
TSUBI calls:
CMMD DIGIT RANGE SUBI2 SUBIM WAIT
TTHRS calls:
RANGE RDIL2 THSH WRIL2
TTSI calls no other procedures.
TTV2C calls:
ADDI2 CMNND DIGIT DSPBW DSPVD RANGE TV2C4 TV2CM WAIT
TWAIT calls:
WAIT
TWIPE calls:
CMNND DIGIT DSPGR RANGE WAIT WIPGR
TXFDS calls:
BLFDS CLFDS CMNND DIGIT RANGE WRILN
TXGLN calls:
RANGE RDGLN WRGLN
TXGPT calls:
RANGE RDGPT RDTAR WRGPT
TXILN calls:
RANGE RDILN WRILN
TXIPT calls:
RANGE RDIPT WAIT WRIPT
TXIRC calls:
RANGE RDILN RDIIRC WRILN WRIRC
TXLUT calls:
RANGE RDLUT WRLUT
TXPSU calls:
RDPSU WRPSU RANGE WAIT
TXTAR calls:
RDTAR WRTAR RANGE
&ADDIM T=00004 IS ON CR00021 USING 00024 BLKS R=0000

0001  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002  SUBROUTINE ADDIM(C, A, B)
0003  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
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
0019  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-6/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.
**C**

**C**SUBPROGRAMS CALLED:

<table>
<thead>
<tr>
<th>name</th>
<th>source</th>
<th>load</th>
<th>remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMDND</td>
<td>&amp;CMDND</td>
<td>%CMDND</td>
<td>Sends a command to the COMTAL as if the command were sent from the keyboard.</td>
</tr>
<tr>
<td>RANGE</td>
<td>&amp;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>&amp;DIGIT</td>
<td>%DIGIT</td>
<td>Character*1 function which returns '0'-'9' according to integer input 0-9.</td>
</tr>
</tbody>
</table>

**C**WRITTEN BY:

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

**C**REVISION HISTORY:

<table>
<thead>
<tr>
<th>C</th>
<th>LOCAL VARIABLES:</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>INTEGER IBUF(128) ! a buffer for passing commands to COMTAL</td>
</tr>
<tr>
<td>C</td>
<td>CHARACTER*255 CBUF ! character overlay for IBUF</td>
</tr>
<tr>
<td>C</td>
<td>EQUIVALENCE (IBUF,CBUF)</td>
</tr>
<tr>
<td>C</td>
<td>LOGICAL RANGE ! function that determines if 1st parameter is within 2nd and 3rd parameter</td>
</tr>
<tr>
<td>C</td>
<td>INTEGER IMLO, IMHI ! limits on COMTAL image numbers</td>
</tr>
<tr>
<td>C</td>
<td>INTEGER TERM</td>
</tr>
<tr>
<td>C</td>
<td>CHARACTER<em>1 DIGIT ! character</em>1 function that returns '0'-'9' according to integer input 0-9.</td>
</tr>
</tbody>
</table>

**C**INITIALIZATIONS:

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

**C**PROCESSING

| C | IF (.NOT.(RANGE(A,IMLO,IMHI))) GOTO 8001 ! error return |
| C | IF (.NOT.(RANGE(B,IMLO,IMHI))) GOTO 8101 ! error return |
| C | 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 B $'/'
   UNassign Image B if 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 $'/'
   Set COMbine <Green - Blue> / 1

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, 78)
RETURN

C***ERROR RETURNS

6001 WRITE(TERM, E283) A
6003 FORMAT(' THE 2ND IMAGE PARAMETER, ', I3, ', IS OUT OF RANGE. ')
GOTO 8900

6101 WRITE(TERM, E103) B
6103 FORMAT(' THE 3RD IMAGE PARAMETER, ', I3, ', IS OUT OF RANGE. ')
GOTO 8900

6201 WRITE(TERM, E283) C
6203 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
**ADD12** T=00004 IS ON CR00021 USING 00024 BLKS R=0000

```fortran
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE ADD12(C, A, B, SCALE)
0003 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 ADD12, 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 The three image numbers must be distinct.
0036 C
0037 C If the sum of any two pixels exceeds 255, the value in C is set to 255.
0038 C
0039 C The SCALE factor must be between 1 and 9 inclusive.
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
0044 C This subroutine assumes that 0 is not a legal image for the COMTAL
```
configuration.

C**SUBPROGRAMS CALLED:

<table>
<thead>
<tr>
<th>name</th>
<th>source</th>
<th>load</th>
<th>remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMDN</td>
<td>CMDN</td>
<td>CMDN</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>

C**WRITTEN BY:

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

C**REVISION HISTORY:

C**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 IMLD, IMHI ! limits on COMTAL image numbers
INTEGER TERM
CHARACTER*1 DIGIT ! character*1 function that returns '0'-'9' according to integer input 0-9.

C**INITIALIZATIONS:

DATA IMLD/1/, IMHI/4/
DATA TERM/1/

C**PROCESSING

IF (.NOT.(RANGE(A,IMLD,IMHI))) GOTO 8001 ! error return
IF (.NOT.(RANGE(B,IMLD,IMHI))) GOTO 8101 ! error return
IF (.NOT.(RANGE(C,IMLD,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 I 8 $'//
0103 C UNassign Image 8 ! just in case 8 is already assigned.
0105 C 'AS T 8 '/'DIGIT(C)//' '/'DIGIT(A)//' '/'DIGIT(B)//' $'//'
0106 C ASSign Truecolor image 8 red #C blue #A green #B
0107 C 'D I 8 $'//'
0108 C Display Image 8
0109 C 'SE COM G + B '/'DIGIT(SCALE)//' $'//'
0110 C SEt COMbine <Green + Blue> / SCALE
0111 C 'A COM $'//'
0112 C Add COMbine
0113 C 'I '/'DIGIT(C)//' D R $'//'
0114 C Image #C = Displayed image Red ! Red is arbitrary, since difference of images is monochrome
0115 C 'D I '/'DIGIT(C)//' $'//'
0116 C Display Image #C
0117 C 'SU COM '
0118 C Subtract COMbine,
0119 C CALL CMMND(IBUT, 77)
0120 C RETURN
0121 C
0122 C ERROR RETURNS
0123 C
0124 C
0125 C 8001 WRITE(TERM, 8003) A
0126 C 8003 FORMAT( 'THE 2ND IMAGE PARAMETER., I3, ', ' IS OUT OF RANGE.' )
0127 C GOTO 8500
0128 C
0129 C 8101 WRITE(TERM, 8103) B
0130 C 8103 FORMAT( 'THE 3RD IMAGE PARAMETER., I3, ', ' IS OUT OF RANGE.' )
0131 C GOTO 8500
0132 C
0133 C 8201 WRITE(TERM, 8203) C
0134 C 8203 FORMAT( 'THE 1ST IMAGE PARAMETER., I3, ', ' IS OUT OF RANGE.' )
0135 C GOTO 8500
0136 C
```
0137  8900 WRITE(TERM, 8901)
0138  8901 FORMAT(‘ ADD12 RETURNS WITHOUT FURTHER PROCESSING.’)
0139   RETURN
0140  C
0141   END
0142
&BUFDS T=00004 IS ON CR0001: USING 00010 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE BUFDS(IMA8E, FLNAME)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C**PARAMETER DECLARATIONS:
0006 C
0007 INTEGER IMAGE ! the number of the COMTAL image plane to be
0008 C filled from the HP disk file.
0009 INTEGER FLNAME(3) ! the HP filename from which an image will be read.
0010 C
0011 C**INTRODUCTION:
0012 C
0013 C The subroutine "Black and White From Disk" transfers one b&w image
0014 C from the HP to the COMTAL. The image number and the name of the HP
0015 C disk are given as parameters.
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 BUFDS only works for b&w images. Since a color image requires
0024 C three separate b&w images, another subroutine, CLFDS is available
0025 C for reading color images from the disk.
0026 C
0027 C**SUBPROGRAMS CALLED:
0028 C
0029 C
0030 C name source load remarks
0031 C ---- ---- ---- -------------------------------
0032 C RANGE &RANGE XRANGE logical function that determines if the 1st
0033 C argument is within the 2nd and 3rd inclusive.
0034 C OPEN ------ ------ HP FORTRAN77 intrinsic subroutine; opens a file
0035 C READF ------ ------ HP FORTRAN77 intrinsic subroutine; reads a record
0036 C WRILN &WRILN WRILN writes a COMTAL Image horizontal LiNe; 2 pixels
0037 C & WRILN per integer in the buffer.
0038 C CMIND &CMIND CMIND sends commands to the COMTAL as if they were
0039 C typed at the COMTAL keyboard.
0040 C DIGIT &DIGIT xDIGIT a character*1 function that returns a single
0041 C ASCII digit when given an integer 0-9
0042 C
0043 C
0044 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.

**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** Digit | function that returns '0','1',... or '9' according to a 0,1,... or 9 integer input.
- **CHARACTER** CBUF | overlays IBUF.
- **EQUIVALENCE** (CBUF,IBUF)

**INITIALIZATIONS:**

- **DATA** IMLO/=, IMHI/=, TERM/=.

**PROCESSING**

- **IF** (.NOT.(RANGE(IMAGE ,IMLO,IMHI))) GOTO 8001 | error return.
- **CALL** OPEN(DBLOCK, IERR, FLNAME).
- **IF** (IERR .LT. 0) GOTO 8201 | error return, open failed.
- **CALL** CMND(IBUF,5).
- **DO** 1000 ROW = 0, 511.
- **CALL** READF(DBLOCK, IERR, IBUF).
- **IF** (IERR .LT. 0) GOTO 8301 | error return, bad read.
- **1000 CONTINUE**
0891       RETURN
0892       C
0893       C**ERROR RETURNS
0894       C
0895       8001 WRITE(TERM, 8003) IMAGE, IMLO, IMHI
0896       8003 FORMAT(' IMAGE NUMBER:".13:" IS OUT OF RANGE:".214:"')
0897       GOTO 8900
0898       C
0899       8201 WRITE(TERM, 8203) IERR
0900       8203 FORMAT(' ERROR OCCURED DURING IMAGE FILE OPENING:".14:"')
0901       GOTO 8900
0902       C
0903       8301 WRITE(TERM, 8303) IERR
0904       8303 FORMAT(' ERROR OCCURED DURING IMAGE FILE READ:".14:"')
0905       GOTO 8900
0906       C
0907       8900 WRITE(TERM, 8901)
0908       8901 FORMAT(' BWFDS FAILS. NO TRANSFER TAKES PLACE."')
0909       RETURN
0910       END
&CLFDS T=00004 IS ON CR00021 USING 00018 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
per integer in the buffer.

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

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

****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:

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:

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

****INITIALIZATIONS:

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

****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)
I .OR. (RED .EQ. BLUE)) GOTO 8301 ! error return
IF (.NOT.(RANGE(COLOR, TRLO, TRHI))) GOTO 8401 ! error return
CALL OPEN(DBLOCK, IERR, FNAME)
IF (IERR .LT. 0) GOTO 8501 ! error return, open failed
CBUF = ' D I' //DIGIT(RED) ! Display the RED Image as it is filled.
CALL CMMND(IBUF,5)
DO 1000 ROW = 0,511
   CALL READF(DBLOCK, IERR, IBUF)
   IF (IERR .LT. 0) GOTO 8701 ! error return, file read failed
   CALL WRILN(RED, ROW, IBUF)
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, IBUF)
   IF (IERR .LT. 0) GOTO 8901 ! error return, file read failed
   CALL WRILN(GREEN, ROW, IBUF)
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, IBUF)
   IF (IERR .LT. 0) GOTO 8801 ! error return, file read failed
   CALL WRILN(BLUE, ROW, IBUF)
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)/' //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,'I3,' IS OUT OF RANGE:'I4,'.')
0140 GOTO 8900
0141 C
0142 8101 WRITE(TERM, 8103) GREEN, IMLO, IMHI
0143 8103 FORMAT(' GREEN IMAGE NUMBER,'I3,' IS OUT OF RANGE:'I4,'.')
0144 GOTO 8900
0145 C
0146 8201 WRITE(TERM, 8203) BLUE, IMLO, IMHI
0147 8203 FORMAT(' BLUE IMAGE NUMBER,'I3,' IS OUT OF RANGE:'I4,'.')
0148 GOTO 8900
0149 C
0150 8301 WRITE(TERM, 8303) RED, GREEN, BLUE
0151 8303 FORMAT(' 3 MONOCHROME IMAGES MUST BE DISTINCT. YOURS:'I4)
0152 GOTO 8900
0153 C
0154 8401 WRITE(TERM, 8403) COLOR, CLLO, CLHI
0155 8403 FORMAT(' YOUR TRUECOLOR IMAGE,'I4,' IS OUT OF RANGE:'I4)
0156 GOTO 8900
0157 C
0158 8501 WRITE(TERM, 8503) IERR
0159 8503 FORMAT(' ERROR WHILE OPENING IMAGE FILE:'I5)
0160 GOTO 8900
0161 C
0162 8601 WRITE(TERM, 8603) IERR
0163 8603 FORMAT(' ERROR WHILE READING IN THE RED MONOCHROME IMAGE:'I4)
0164 GOTO 8900
0165 C
0166 8701 WRITE(TERM, 8703) IERR
0167 8703 FORMAT(' ERROR WHILE READING IN THE GREEN MONOCHROME IMAGE:'I4)
0168 GOTO 8900
0169 C
0170 8801 WRITE(TERM, 8803) IERR
0171 8803 FORMAT(' ERROR WHILE READING IN THE BLUE MONOCHROME IMAGE:'I4)
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.

INTRODUCTION:
CLRGR GRAPH clears the graphics designated by GRAPH. GRAPH should be within the range 1-4 for the present IRD COMTAL system.

If GRAPH 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:
GRAPH must be within the limits GRLO and GRHI explained below.

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

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

REVISION HISTORY:

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.

EQUALENCE (IBUF, CBUF)

C
C***INITIALIZATIONS:
DATA GRLO/-1, GRHI/-4/
DATA TERM/-1/
C
C***PROCESSING
C
IF (.NOT.(RANGE(GRAPH, GRLO, GRHI))) GOTO 8001 ! error return
C 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 CMDND(IBUF, 13)
RETURN
C
C***ERROR RETURN
C
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
C
END
SUBROUTINE CLRIM( IMAGE )

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

INTRODUCTION:
CLRIM 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:
RANGE &RANGE %RANGE logical function that determines if its 1st parameter is between (inclusive) its last parameter
CMDND &CMDND %CMDND sends a COMTAL command as if it were typed at the COMTAL keyboard.
DIGIT &DIGIT %DIGIT a character*1 function which returns the character associated with integer inputs 0,...,9.

WRITTEN BY:
The code on which this subroutine is based was written by HETTIE D. FAULCON, July, 1963. This modification is by KEITH MILLER, June, 1984.

REVISION HISTORY:
LOCAL VARIABLES:
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/1/, IMHI/4/
DATA TERM/1/

C***PROCESSING
IF (.NOT.(RANGE(IMAGE, IMLO, IMHI))) GOTO 8001 ! error return
ELSE...clear the image
CBUF = 'CLEAR IMAGE ' // DIGIT(IMAGE) ! since CBUF overlays IBUF,
! this statement loads IBUF with
! the COMTAL command
CALL CMMND(IBUF, 13)
RETURN

C***ERROR RETURN
WRITE(TERM, 8003) IMAGE, IMLO, IMHI
FORMAT(' IMAGE NUMBER, ', 14, ' IS OUT OF RANGE:', 213, '.')
WRITE(TERM, 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 CMMLD 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 CMMLD, the final blank, $, and the required zero byte are added automatically to the INBUF string.

If the last character isn't a blank, CMMLD 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
**LEN** integer function returns length of character string.

**WRITTEN BY:**

The original code upon which this subroutine is based was written by NETTIE D. FAULKON, JULY, 1983.

**REVISION HISTORY:**

Modified by Keith Miller, 6/18/84.

**LOCAL VARIABLES:**

**INITIALIZATIONS:**

DATA BLANK/ '', DOLLAR/ $', IZERO/ 0/

NUMCHR = LEN(INBUF)

**PROCESSING:**

DO 10 INDEX = 1, NUMCHR

CBUF(INDEX) = INBUF(INDEX:INDEX)

CONTINUE

IF (CBUF(NUMCHR) .EQ. BLANK) GOTO 30

ELSE...

NUMCHR = NUMCHR + 1

CBUF(NUMCHR) = BLANK

ADD ENDING CHARACTERS TO COMMAND

30 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-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 (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
&CMMD T=00004 IS ON CR00021 USING 00012 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE CMMD(INBUF, INCNT)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***PARAMETERS:
0006 C
0007 INTEGER INBUF(128) ! 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 "CMMD" 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-G/VM.
0026 C
0027 C***LIMITATIONS:
0028 C
0029 C In order to make it easier to send single COMTAL commands to CMMD,
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, CMMD 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, 1963.

**REVISION HISTORY:**
Modified by Keith Miller, 6/18/84.

**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
- 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/

**PROCESSING:**
1. WORDS = (NUMCHR+1) / 2
2. DO 10 INDEX = 1, WORDS
   IBUF(INDEX) = INBUF(INDEX)
   10 CONTINUE
3. IF (CBUF(NUMCHR) .EQ. BLANK) GOTO 30
4. ELSE...
5. NUMCHR = NUMCHR + 1
6. CBUF(NUMCHR) = BLANK
7. ADD ENDING CHARACTERS TO COMMAND
8. CBUF(NUMCHR+1) = DOLLAR
9. CBUF(NUMCHR+2) = ZERO
10. CBUF(NUMCHR+3) = ZERO  | safety precaution
11. NUMCHR = NUMCHR + 2
12. CHANGE BYTE COUNT TO WORD COUNT

34
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, 248818)
RETURN
END
SUBROUTINE COUNT(COUNTS, IMAGE)

INTEGER*4 COUNTS(256) ! holds the counts for pixel values 0-255
INTEGER IMAGE ! COMTAL image number of which COUNTgram is
                  ! to be taken

The 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.

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

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.

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

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**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

C**INITIALIZATIONS:

DATA        IMLO/1/, IMHI/4/

DATA        TERM/1/

C**PROCESSING

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

DO 1000 INDEX = 1, 256
    COUNTS(INDEX) = 0
1000 CONTINUE

DO 3800 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
2000 CONTINUE
3800 CONTINUE

RETURN

C**ERROR RETURN

WRITE(TERM, 8003) IMAGE, IMLO, IMHI

WRITE(TERM, 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 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE DELAY(SECOND)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
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/VM.
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. 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 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*4 NOWCNT  ! the milliseconds, seconds, minutes, hours,
                  ! and Julian day of the latest EXEC call that
                  ! determines the time.
INTEGER TIMREQ  ! the EXEC number for a time request.
INTEGER TERM  ! logical unit for terminal output.

C**INITIALIZATIONS:
DATA TIMREQ/11/
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.’)
8000 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 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 CHARACTER FUNCTION DIGIT( INTIN )
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
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 CHARACTER*1 DGTARA(10)  ! DIGIT Array holds the digits '0'- '9'
0041 LOGICAL RANGE           ! function that determines if its first parameter
0042 C            is between (inclusive) its last two parameters.
0043 C INTEGER TERM           ! logical unit number for terminal output.
C***INITIALIZATIONS:

DATA DGTARA/'0', '1', '2', '3', '4', '5', '6', '7', '8', '9'/

DATA TERM/1/

C

C***PROCESSING

IF (.NOT.(RANGE(INTH, 0, 9))) GOTO 8001 ! error return
ELSE... convert to digit and return

DIGIT = DGTARA(INTH + 1)
RETURN

C

C***ERROR RETURN

WRITE(TERM, 8003) INTH
8003 FORMAT('THE INPUT TO DIGIT IS NOT A SINGLE DIGIT. DIGIT RETURNS A BLANK. ')

DIGIT = '
RETURN

END

0066
SUBROUTINE DSPBW (IMAGE)

INTEGER IMAGE ! a number 1-4 designating a COMTAL image

"Display Black and White" is a subroutine that allows the caller to send a display command to the COMTAL from an HP program. The call can turn on one black and white image, number 1, 2, 3, or 4. Any previous pseudocolor or function memory commands are nullified by a DSPBW call.

The resident driver DVR41 is called via an EXEC to accomplish the display.

This subroutine does NOT display true color images. That is accomplished using the subroutine DSPCL.

SUBPROGRAMS CALLED:

RANGE &RANGE XRANGE logical function that determines if the first parameter is within the bounds defined by the second and third parameter (inclusive).

The code on which this subprogram is based was written by KEITH MILLER, June, 1984.

The code on which this subprogram is based was written by KEITH MILLER, June, 1984.

The code on which this subprogram is based was written by KEITH MILLER, June, 1984.
C    ! within a certain range
0046 INTEGER TERM ! the logical unit number for the terminal
0047 INTEGER IDUMMY! fills the place of an unused EXEC parameter
0048 C
0049 C***INITIALIZATIONS:
0050 C
0051 DATA TERM/1/
0052 C
0053 C***PROCESSING
0054 C
0055 IF (RANGE(IMAGE, 1, 4)) GOTO 2000 ! legal image number
0056 C ELSE... illegal image number
0057 WRITE(TERM, 1001) IMAGE
0058 1001 FORMAT( ' The image number ', I3, ' is out of range.', )
0059 WRITE(TERM, 1002)
0060 1002 FORMAT( ' DSPBW fails. No action taken on command.', )
0061 RETURN
0062 C
0063 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 CALL EXEC(2, 000B + 36B, IDUMMY, 0, (IMAGE-1) * 2)
0076 RETURN
0077 END
&DSPCL T=00004 IS ON CR00021 USING 00006 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE DSPCL(REDC, GREEN, BLUE, TCLR)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C**INPUT PARAMETERS:
0006 C
0007 INTEGER RED, GREEN, BLUE | COMTAL image numbers for the 3 color
0008 C | components of the truecolor image to
0009 C | be displayed.
0010 INTEGER TCLR | COMTAL image number for the truecolor
0011 C | image formed from RED, GREEN, and BLUE.
0012 C
0013 C**INTRODUCTION:
0014 C
0015 C "Display CoLor" commands the COMTAL to display one
0016 C RGB true color image. Any previous function or
0017 C pseudocolor memory commands are nullified. See the
0018 C subroutine DSPBU to display black and white images.
0019 C
0020 C The color display is accomplished via the CMMND subroutine,
0021 C which allows COMTAL commands to be sent to the COMTAL as if
0022 C they were typed on the COMTAL keyboard.
0023 C
0024 C**LANGUAGE:
0025 C
0026 C FORTRAN 77, the HP-100 version for RTE-6/VM.
0027 C
0028 C**LIMITATIONS:
0029 C
0030 C The three parameters RED, GREEN and BLUE must be distinct and
0031 C within the IMLO, IMHI range. TCLR must be within the TRLO, TRHI
0032 C range (which is mutually exclusive with IMLO-IMHI). The limits on
0033 C truecolor numbers are arbitrary. However, this subroutine enforces
0034 C the arbitrary limits. The HP image file must contain all three
0035 C monochrome images in the order RED, GREEN, and BLUE.
0036 C
0037 C**SUBPROGRAMS CALLED:
0038 C
0039 C name | source | load | remarks
0040 C ------ | ------ | ------ | -----------------------------------------------
0041 C RANGE | &RANGE | XRANGE | logical function that determines if the 1st
0042 C | | | argument is within the 2nd and 3rd inclusive.
0043 C CMMND | &CMMND | XCMMND | transfers a command string to the COMTAL, which
0044 C | | | accepts it almost as a keyboard command.
**DIGIT** &**DIGIT** *&**DIGIT* character*1* function that returns a single digit on legal integer inputs 0-9.

**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:**

**Initialization:**

**Processing**

The following commands are abbreviations of the following:

```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.
DATA IMLO/1/, IMHI/4/ 
DATA TRLO/5/, TRHI/9/ 
DATA TERM/1/ 

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) ) GOTO 8301 ! error return
IF (.NOT.(RANGE(TCLR, TRLO,TRHI))) GOTO 8401 ! error return
DISPLAY THE COLOR IMAGE
```

46
COMTAL commands, where *R*, *G*, *B*, and *C* stands for the single character digits corresponding to RED, GREEN, BLUE, and TCLR:

UNassign Image *C*

ASSign Truecolor *C* red *R* green *G* blue *B*

Display Image *C*

CBUF = "UN I"//DIGIT(TCLR)

CALL CMDND(IBUF,6)

CBUF = "AS T"//DIGIT(TCLR)""//DIGIT(RED)""""//

1       DIGIT(GREEN)""//DIGIT(BLUE)

CALL CMDND(IBUF,12)

CBUF = "D I"//DIGIT(TCLR)

CALL CMDND(IBUF,5)

RETURN

**ERROR RETURNS**

WRITE(TERM, 8003) RED, IMLO, IMHI

8003 FORMAT(" RED IMAGE NUMBER','.I3,'. IS OUT OF RANGE:'.214,'."")

GOTO 8900

WRITE(TERM, 8103) GREEN, IMLO, IMHI

8103 FORMAT(" GREEN IMAGE NUMBER','.I3,'. IS OUT OF RANGE:'.214,'."")

GOTO 8900

WRITE(TERM, 8203) BLUE, IMLO, IMHI

8203 FORMAT(" BLUE IMAGE NUMBER','.I3,'. IS OUT OF RANGE:'.214,'."")

GOTO 8900

WRITE(TERM, 8303) RED, GREEN, BLUE

8303 FORMAT(" 3 MONOCHROME IMAGES MUST BE DISTINCT. YOURS:'.314")

GOTO 8900

WRITE(TERM, 8403) TCLR, CLLO, CLHI

8403 FORMAT(" YOUR TRUECOLOR IMAGE','.14,'. IS OUT OF RANGE:'.214")

GOTO 8900

WRITE(TERM, 8501)

8901 FORMAT(" DSPLCL FAILS. NO DISPLAY TAKES PLACE.")

RETURN

END
SUBROUTINE DSPGR(GRNUM)

INTEGER GRNUM ! a number 1-4 designing a COMTAL graphics plane.

"Display Graphics" is a subroutine that allows the caller
to send a display command to the COMTAL from an HP program.
The call can turn on one graphics plane, number 1, 2, 3,
or 4.

This subroutine does not turn off previous graphics planes.
That is accomplished using the subroutine WIPGR.
It is OK to call DSPGR repeatedly without an intervening WIPGR.
The extra calls have no effect, but they don't hang up the COMTAL.

name  source  load  remarks
-----  ------  ------  -----------------------------------------------
RANGE  &RANGE  $RANGE  logical function that determines if the
first parameter is within the bounds defined by the second and third parameter (inclusive).
CMNN2  &CMNN2  $CMNN2  sends a constant string to the COMTAL as if
the string were typed on the COMTAL keyboard.
DIGIT  &DIGIT  $DIGIT  character*1 function that returns '0'-'9'

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
NETTIE D. FAULCON, July, 1983. This modification is by
KEITH MILLER, June, 1984.
0045 C  LOGICAL  RANGE  ! a function for determining if an integer is
0046 C  ! within a certain range.
0047 C  INTEGER  TERM  ! the logical unit number for the terminal.
0048 C  INTEGER  IDUMTY  ! fills the place of an unused EXEC parameter.
0049 C  INTEGER  GRLD,GRHI ! limits on a graphics plane number.
0050 C  CHARACTER*1  DIGIT  ! function that returns '0'..'9' for input
0051 C  ! integers 0..9.
0052 C
0053 C
0054 C***INITIALIZATIONS:
0055 C
0056 DATA  TERM/I/
0057 DATA  GRLD/1/,GRHI/4/
0058 C
0059 C***PROCESSING
0060 C
0061 IF (.NOT.(RANGE(GRNUM,GRLD,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 CALL CMMN2('ADD GR'//DIGIT(GRNUM))
0067 RETURN
0068 C
0069 C***ERROR RETURN:
0070 C
0071 8001 WRITE(TERM, 8003) GRNUM, GRLD, 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 RETURN
0077 END
SUBROUTINE DSPVD

**INPUT PARAMETERS:**
None.

**INTRODUCTION:**
"Display Video" command 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.

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

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

**SUBPROGRAMS CALLED:**

<table>
<thead>
<tr>
<th>name</th>
<th>source</th>
<th>load</th>
<th>remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMMND</td>
<td>&amp;CMMND</td>
<td>&amp;CMMND</td>
<td>transfers a command string to the COMTAL, which accepts it as a keyboard command.</td>
</tr>
<tr>
<td>DIGIT</td>
<td>&amp;DIGIT</td>
<td>&amp;DIGIT</td>
<td>character*1 function that returns '0'-'9' according to integer input 0-9.</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:**
INTEGER IBUF(256) ! buffers COMTAL command
CHARACTER*255 CBUF ! character overlay for sending COMTAL commands
EQUIVALENCE (CBUF, IBUF)
INTEGER TVIMA6 ! COMTAL image & for video camera
CHARACTER*1 DIGIT ! function that returns '0'-'9' according
! to integer input 0-9.

**INITIALIZATIONS:**
DATA TVIMA/S/ ! arbitrary choice.

**PROCESSING**
DISPLAY THE VIDEO IMAGE
CBUF = 'DISPLAY IMAGE '/DIGIT(TVIMA)/' '
CALL CMDN(D(IBUF,16)
RETURN
END
&HILO T=00004 IS ON CR00021 USING 00004 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE HILO(HI, LO, IMAGE)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***PARAMETER DECLARATIONS:
0006 C
0007  INTEGER HI, LO ! output parameters, the high and low pixel values
0008  C ! found in the designated image.
0009  INTEGER IMAGE ! the COMTAL image number of the image that is to
0010  C ! be searched for its high and low pixel values.
0011 C
0012 C***INTRODUCTION:
0013 C
0014 C ! The subroutine "High and Low values #2" reads through an image and
0015 C ! determines the highest and lowest pixel values, returning the
0016 C ! values found. HILO scans the entire image.
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 ! As noted above, this subroutine passes through an entire image, pixel
0025 C ! by pixel. In some applications you may want to combine other processing
0026 C ! during that pass, but this subroutine won't let you do that.
0027 C
0028 C***SUBPROGRAMS CALLED:
0029 C
0030 C ! name source load remarks
0031 C ------- ------- ------- ----------------------------------------------
0032 C RANGE RANGE X RANGE logical function that determines if 1st argument
0033 C is within 2nd & 3rd inclusive.
0034 C RDIL2 RDIL2 XRDIL2 reads a horizontal line of pixel values into
0035 C ! a 512 integer array.
0036 C
0037 C***WRITTEN BY:
0038 C
0039 C ! The code on which this subprogram is based was written by
0040 C ! NETTIE D. FAULCON, July, 1983. This subprogram was written by
0041 C ! KEITH MILLER, July, 1984, with the support of a NASA-ASEE
0042 C ! summer fellowship.
0043 C
0044 C***REVISION HISTORY:
**LOCAL VARIABLES:**

- 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 LNLO, LNHI ! limits on COMTAL image line 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**

- DATA IMLO/1/, IMHI/4/
- DATA PXLO/0/, PXHI/255/
- DATA LNLO/0/, LNHI/511/
- DATA ARALO/1/, ARAHI/512/

**PROCESSING**

- IF (.NOT.(RANGE(IMAGE, IMLO, IMHI))) GOTO 8001 ! error return
- HI = PXLO ! artificially low
- LO = PXHI ! artificially high

- DO 2000 ROW = LNLO, LNHI
  - 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

**ERROR HANDLING**

- WRITE(TERM, 8001) IMAGE, IMLO, IMHI
- FORMAT(" IMAGE NUMBER," ,15," IS OUT OF RANGE:" ,215,".")
- GOTO 8900

- WRITE(TERM, 8900)
- FORMAT(" HILO FAILS. HI AND LO PARAMETERS UNCHANGED.")
- RETURN

END
&HISTO T=00004 IS ON CR0021 USING 00005 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE HISTO(IMAGE)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***PARAMETER DECLARATIONS:
0006 C
0007 INTEGER IMAGE ! the COMTAL image number to take the histogram of.
0008 C
0009 C***INTRODUCTION:
0010 C
0011 C The subroutine HISTOgram uses the COMTAL "function memory" (a look-up
0012 C table) to construct a histogram of the given image, which is held
0013 C IN A SCALED VERSION in the function memory associated with IMAGE.
0014 C This histogram is displayed by HISTO.
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 function memory associated with image number IMAGE is destroyed.
0023 C IMAGE must be within the boundaries for COMTAL image memories, or
0024 C an error message is given and HTABLE is unchanged.
0025 C
0026 C Notice that the histogram is scaled so that the largest value reaches
0027 C to the top of the screen when displayed. Thus, no absolute counts can
0028 C be easily deduced from the function memory values. See the subroutine
0029 C COUNT if absolute pixel value counts are desired.
0030 C
0031 C HISTO puts the HP into a busy wait while the COMTAL determines the
0032 C histogram. Otherwise, the COMTAL ignores subsequent CMMN2 commands.
0033 C
0034 C***SUBPROGRAMS CALLED:
0035 C
0036 C name source load remarks
0037 C ------- ------ ------- --------------------------------------------------
0038 C CMMN2 &CMMN2 %CMMN2 sends a constant string to the COMTAL, which
0039 C treats it as a command typed on the COMTAL.
0040 C DELAY &DELAY %DELAY puts the HP into a busy wait for at least the
0041 C given number of seconds.
0042 C DIGIT &DIGIT %DIGIT character*1 function that returns '0'-'9' according to integer input 0-9.
0043 C RANGE &RANGE %RANGE logical function that determines if its 1st
argument lies within its 2nd and 3rd, inclusive.

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**REVISION HISTORY:

C

C

C**LOCAL VARIABLES:

C

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.

C**INITIALIZATIONS:

C

DATA IMLO/1/, IMHI/4/

DATA TERM/1/

C**PROCESSING:

C

IF (.NOT.(RANGE(IMAGE, IMLO, IMHI))) GO TO 8801 ! 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 IM = Histogram of image

where IM is the single digit associated with IMAGE.

Image IM 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
C**ERROR RETURNS:

CALL CM**2(’D F ’/IMCHAR)! "Display Function memory #I"
RETURN

WRITE(TERM, 8003) IMAGE, IMLO, IMHI
8003 FORMAT(’ THE IMAGE NUMBER,’15,’ IS OUT OF RANGE:’215,’’)
8900 WRITE(TERM, 8901)
8901 FORMAT(’ HISTO FAILS. NO HISTOGRAM TABLE ASSIGNMENT.’)
RETURN
END
&ICOPY T=00004 IS ON CR00021 USING 00012 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE ICOPY(OUTIMG, INIMG)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
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" 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 ICOPY and the output image must be associated with
0016 C a COMTAL image memory plane (1-4 currently).
0017 C The companion subroutine ICOPY2 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:
C The code on which this subroutine is based was written by
C NETTIE D. FAULCON in July, 1983. This modification is by
C KEITH MILLER June, 1984.

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

C

C LOGICAL RANGE function determines if 1st argument is
C within 2nd and 3rd argument inclusive.
C
C CHARACTER*1 DIGIT function returns '0'-9' according to
C integer input 0-9.
C
C CHARACTER*255 CCOMM character buffer for building up a call
C to the CMDND subroutine.
C
C INTEGER IBUF (128) integer overlay of CCOMM
C
C EQUIVALENCE (CCOMM, IBUF)

C
C INTEGER IMLO, IMHI the range of legal COMTAL image numbers
C
C INTEGER TERM terminal logical unit

C

C***INITIALIZATION:

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

C

C

C***PROCESSING:

C

C IF (.NOT.(RANGE(OUTIMG, IMLO, IMHI))) GOTO 8001 error return
C
C IF (.NOT.(RANGE(INIMG, IMLO, IMHI))) GOTO 8101 error return
C
C Legal image numbers, so do the copy
C
C CCOMM = 'IMAGE 'DIGIT(OUTIMG)'/ = IMAGE 'DIGIT(INIMG)
C
C PROGRAMMING NOTE: see COMTAL USER'S GUIDE for
C further information on the command string abbreviated
C in the string above.

C CALL CMDND( IBUF, 17 ) sends copy command to COMTAL
C
C RETURN

C

C

C***ERROR RETURNS:

C

C

C 8001 WRITE(TERM, 8003)OUTIMG, IMLO, IMHI
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 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE ICPY2(OUTIMG, INIMG)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C**INPUT PARAMETERS:
0006 C
0007 C INTEGER OUTIMG ! COMTAL image number for the destination
0008 C INTEGER INIMG ! COMTAL image number for the source
0009 C
0010 C**INTRODUCTION:
0011 C
0012 C "Image Copy #2" 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 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 ICOPY 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 CMMND &CMMND %CMMND 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. FAULKON in July, 1983. This modification is by
KEITH MILLER , June, 1984.

**REVISION HISTORY:**

**LOCAL VARIABLES:**

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0054</td>
<td>LOGICAL RANGE ! function determines if 1st argument is within 2nd and 3rd argument inclusive.</td>
</tr>
<tr>
<td>0055</td>
<td>CHARACTER*1 DIGIT ! function returns '0'-'9' according to integer input 0-9.</td>
</tr>
<tr>
<td>0056</td>
<td>CHARACTER*255 CCOMM ! character buffer for building up a call to the CMND subroutine.</td>
</tr>
<tr>
<td>0059</td>
<td>INTEGER IBUF (128)! integer overlay of CCOMM</td>
</tr>
<tr>
<td>0061</td>
<td>EQUIVALENCE (CCOMM, IBUF)</td>
</tr>
<tr>
<td>0062</td>
<td>INTEGER IMLO, IMHI ! the range of legal COMTAL image numbers</td>
</tr>
<tr>
<td>0063</td>
<td>INTEGER TERM ! terminal logical unit</td>
</tr>
</tbody>
</table>

**INITIALIZATION:**

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0068</td>
<td>DATA IMLO/1/, IMHI/4/</td>
</tr>
<tr>
<td>0069</td>
<td>DATA TERM/1/</td>
</tr>
</tbody>
</table>

**PROCESSING:**

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0073</td>
<td>IF (.NOT.(RANGE(OUTIMG,IMLO,IMHI))) GOTO 8001 ! error return</td>
</tr>
<tr>
<td>0074</td>
<td>IF (.NOT.(RANGE(INIMG, 1, 9 ))) GOTO 8101 ! error return</td>
</tr>
<tr>
<td>0076</td>
<td>Legal image numbers, so do the copy</td>
</tr>
<tr>
<td>0078</td>
<td>CCOMM = 'IMAGE '/DIGIT(OUTIMG)'/ = IMAGE '/DIGIT(INIMG)</td>
</tr>
<tr>
<td>0079</td>
<td>PROGRAMMING NOTE: see COMTAL USERS GUIDE for further information on the command string abbreviated in the string above.</td>
</tr>
<tr>
<td>0084</td>
<td>CALL CMND( IBUF, 17 ) ! sends copy command to COMTAL</td>
</tr>
<tr>
<td>0085</td>
<td>RETURN</td>
</tr>
</tbody>
</table>

**ERROR RETURNS:**

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0089</td>
<td>8001 WRITE(TERM, 8003)OUTIMG, IMLO, IMHI</td>
</tr>
<tr>
<td>0090</td>
<td>8003 FORMAT(' YOUR OUTPUT PARAMETER:, IS , IS OUT OF RANGE:',214)</td>
</tr>
</tbody>
</table>
GOTO 8900

C

8101 WRITE(TERM, 8103) IN IMG

8103 FORMAT(' YOUR INPUT PARAMETER, .IS., IS OUT OF RANGE: 1, 9')

GOTO 8900

C

8900 WRITE(TERM, 8901)

8901 FORMAT(' ICPY2 FAILS. NO COPYING TAKES PLACE.')

RETURN

END
&MERGE T=00004 IS ON CR60021 USING 00005 BLKS R=0000

0001  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002  SUBROUTINE MERGE(OUTURD, BYTE1, BYTE2)
0003  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
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
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  METTE D. FAULCON, July, 1982. 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**LOCAL VARIABLES:
0042  C
0043  C  INTEGER IHOLD  ! an INTEGER interpretation of bits
0044  C  CHARACTER CHOLD(2)  ! a CHARACTER interpretation of bits

63
INTEGER IMERGE ! an INTEGER interpretation of bits
CHARACTER CSPLIT(2) ! a CHARACTER interpretation of bits

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

INITIALIZATIONS: none.

PROCEDURE:

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

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

OUTURD = IMERGE

RETURN
END
SUBROUTINE NORML(IMAGE)

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 called 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 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-6/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:**

- **RDIL2** reads one horizontal COMTAL image line into an integer array, one integer/pixel.
- **WRIL2** writes one horizontal COMTAL image line from an integer array; one integer/pixel.
- **HILO** determines the highest and lowest pixel value in an image.

**WRITTEN BY:**
C The code on which this subprogram is based was written by
C 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 INTEGER IBUF(512) ! buffer for pixel values; one pixel / integer
C INTEGER ARALO, ARAHI! array bounds for a pixel buffer
C INTEGER LNLO, LNHI! limits on COMTAL row numbers
C INTEGER HI, LO! highest and lowest pixel values in IMAGE
C INTEGER ROW, COL! loop indices

C***INITIALIZATIONS:

C

C DATA ARALO/=2, ARAHI/=512/
C DATA LNLO /0/, LNHI /511/

C***PROCESSING

C

C CALL HILO(HI, LO, IMAGE)

C

DO 2000 ROW = LNLO, LNHI
  CALL RDIL2(IBUF, IMAGE, ROW)
  DO 1000 COL = ARALO, ARAHI
    IBUF(COL) = IBUF(COL) - LO
    CONTINUE
  CONTINUE
C 2000 CONTINUE

C CALL WRL2(IMAGE, ROW, IBUF)
C
C 2000 CONTINUE

C RETURN
C
C END
&NOTE2  T=00004 IS ON CR0021 USING 00024 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE NOTE2( GRNUM, XCOORD, YCOORD, COLOR, FACTOR, NOTE)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C**PARAMETER DECLARATIONS:
0006 C
0007 INTEGER GRNUM ! the number of the graphic to be written into;
0008 C COLOR doesn't clear or display this graphic.
0009 INTEGER XCOORD !
0010 INTEGER YCOORD ! XCOORD and YCOORD 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 C or change to red ('R'), green('G'), blue('B'),
0015 C black('K'), white('W'), or yellow('Y').
0016 CHARACTER*(*) NOTE ! the message to be printed in graphics; NOTES
0017 C designed for constant strings.
0018 C
0019 C**INTRODUCTION:
0020 C
0021 C This subroutine, NOTE2 writes a line of characters into a COMTAL graphics
0022 C plane. The parameter NOTE should be a constant string.
0023 C A very similar subroutine, NOTE5, uses a string variable and a length
0024 C parameter instead of the constant string.
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 XCOORD and YCOORD 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"), 'B' (for "Blue"),
0036 C 'K' (for "black"), 'W' (for "White") or 'Y' (for "Yellow").
0037 C Other colors could be easily added. See TABLE 1 in the NOTES
0038 C section for other colors.
0039 C FACTOR must be in the range 1-16 inclusive.
0040 C If a restriction is violated, NOTES fails with an error message.
0041 C
0042 C The graphics plane named here is turned on and all other graphics
0043 C planes are turned off when this subroutine is executed.
0044 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.

**SUBPROGRAMS CALLED:**

<table>
<thead>
<tr>
<th>name</th>
<th>source</th>
<th>load</th>
<th>remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMNT2</td>
<td>CMNT2</td>
<td>CMNT2</td>
<td>sends constant string to COMTAL as if it were typed at the COMTAL keyboard.</td>
</tr>
<tr>
<td>CMNTD</td>
<td>CMNTD</td>
<td>CMNTD</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>DELAY</td>
<td>DELAY</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>DIGIT</td>
<td>DIGIT</td>
<td>character*1 function that returns '0'-'9' according to integer input 0-9.</td>
</tr>
<tr>
<td>DSPGR</td>
<td>DSPGR</td>
<td>DSPGR</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>RANGE</td>
<td>RANGE</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:**

**FUNCTIONS:**

<table>
<thead>
<tr>
<th>CHARACTER*1</th>
<th>DIGIT</th>
<th>function that returns '0'-'9' for integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEGER</td>
<td>LEN</td>
<td>intrinsic HP FORTRAN intrinsic integer function that returns the length of a character string.</td>
</tr>
<tr>
<td>LOGICAL</td>
<td>RANGE</td>
<td>logical function determines if 1st argument is within 2nd and 3rd arguments, inclusive.</td>
</tr>
<tr>
<td>CHARACTER*1</td>
<td>GRCHAR</td>
<td>the single digit that corresponds to GRNUM.</td>
</tr>
<tr>
<td>INTEGER</td>
<td>LONG</td>
<td>length of the character string.</td>
</tr>
<tr>
<td>INTEGER</td>
<td>START</td>
<td>the # of first character in CBUF that holds the first character of NOTE.</td>
</tr>
</tbody>
</table>
INTEGER GRLO,GRHI ! limits for COMTAL graphics plane number.
INTEGER CMLO,CMHI ! limits on COMTAL coordinates.
INTEGER FCLO,FCHI ! limits on FACTOR.
INTEGER TERM ! logical unit for terminal output.
INTEGER IBUF(128) ! buffer for sending CMMND commands.
CHARACTER*255 CBUF ! overlays IBUF
EQUIVALENCE (IBUF, CBUF)

***INITIALIZATIONS:

DATA GRLO/1/,GRHI/4/
DATA CMLO/0/,CMHI/511/
DATA FCLO/1/,FCHI/16/
DATA TERM/1/

***PROCESSING:

IF (.NOT.(RANGE(GRNUM, GRLO, GRHI))) GOTO 8001 ! error return
IF (.NOT.(RANGE(XCOORD, CMLO, CMHI))) GOTO 8101 ! error return
IF (.NOT.(RANGE(YCOORD, CMLO, CMHI))) GOTO 8201 ! error return
IF (.NOT.(RANGE(FACTOR, FCLO, FCHI))) GOTO 8301 ! error return

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

IF ((COLOR .EQ. 'S') .OR. (COLOR .EQ. 's')) GOTO 2000 ! "Same"
IF ((COLOR .EQ. 'R') .OR. (COLOR .EQ. 'r')) GOTO 1000 ! "Red"
IF ((COLOR .EQ. 'G') .OR. (COLOR .EQ. 'g')) GOTO 1100 ! "Green"
IF ((COLOR .EQ. 'B') .OR. (COLOR .EQ. 'b')) GOTO 1200 ! "Blue"
IF ((COLOR .EQ. 'K') .OR. (COLOR .EQ. 'k')) GOTO 1300 ! "Black"
IF ((COLOR .EQ. 'W') .OR. (COLOR .EQ. 'w')) GOTO 1400 ! "White"
IF ((COLOR .EQ. 'Y') .OR. (COLOR .EQ. 'y')) GOTO 1500 ! "Yellow"
ELSE...COLOR an illegal character
GOTO 8401 ! error return

Color graphics red
1000 CALL CMMN2(‘CO G ’/’GRCHAR’/’ RED’)
GOTO 2000

Color graphics green
1100 CALL CMMN2(‘CO G ’/’GRCHAR’/’ GRN’)
GOTO 2000

Color graphics blue
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 chosen graphics
0152    CALL WRTAR(XCOORD, YCOORD)  ! position the cursor for writing
0153  C
0154    LONG = LEN(NOTE)
0155  IF (FACTOR .GE. 10) GOTO 3000
0156  C  ELSE...
0157    CBUF(1:8) = 'G 'GRCHAR'/" L '/DIGIT(FACTOR)/" '  
0158      START = 9
0159    LONG = LONG + 8
0160  GOTO 4000
0161  C  THEN...
0162  3000  CBUF(1:9) = 'G 'GRCHAR'/" L 1'/DIGIT(FACTOR-10)/" '  
0163      START = 10
0164    LONG = LONG + 9
0165  GOTO 4000
0166  C
0167  4000  CBUF(ST A R T:LONG) = NOTE
0168    CALL CM N D(IBUF,LONG)
0169  C
0170  C  Put the HP in a busy wait while the COMTAL 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,GRAHI
0177  8003 FORMAT(' THE GRAPHICS NUMBER.'15,' IS OUT OF RANGE:'214,'')
0178  8000  GOTO 8900
0179  C
0180  8101 WRITE(TERM.8103)XCOORD,CMLO,CMHI
0181  8103 FORMAT(' THE X COORDINATE.'15,' IS OUT OF RANGE:'214,'')
0182  8900  GOTO 8900

70
0183 C
0184 8201 WRITE(TERM,8203) YCOOR,CMLO,CMHI
0185 8203 FORMAT(' THE Y COORDINATE, ',15,' IS OUT OF RANGE: ',214,'. ')
0186 GOTO 8900
0187 C
0188 8301 WRITE(TERM,8303) FACTOR,FCL0,FCHI
0189 8303 FORMAT(' THE SCALE FACTOR, ',15,' IS OUT OF RANGE: ',214,'. ')
0190 GOTO 8900
0191 C
0192 8401 WRITE(TERM,8403) COLOR
0193 8403 FORMAT(' THE COLOR PARAMETER, ',15,' IS NOT S, R, G, OR B. ')
0194 GOTO 8900
0195 C
0196 8900 WRITE(TERM,8901)
0197 8901 FORMAT(' NOTE2 FAILS. NO LETTERING PLACED INTO GRAPHICS. ')
0198 C
0199 RETURN
0200 END
&NOTES T=00004 IS ON CR00021 USING 00004 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE NOTES(GRNUM,XCOOR,YCOOR,COLOR,FACTOR,NOTE,LENGTH)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
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 and 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 C 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 C 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 DELAY is used to cause a delay proportional to the size of the characters being printed at the COMTAL.

**SUBPROGRAMS CALLED:**

- **CMNN2** sends constant string to COMTAL as if it were typed at the COMTAL keyboard.
- **CMMND** sends a fixed string command to COMTAL as if it were typed at the COMTAL Keyboard.
- **DELAY** puts the HP in a busy wait for the number of seconds designated in DELAY's parameter.
- **DIGIT** character function that returns '0'-'9' according to integer input 0-9.
- **DSPGR** adds a graphic plane to the display.
- **RANGE** logical function that determines if the 1st argument is within the 2nd & 3rd inclusive.

**WITTEN 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 DIGIT function that returns '0'-'9' for integer input 0-9.
- LOGICAL RANGE function that determines if 1st argument is within 2nd and 3rd arguments, inclusive.
- CHARACTER GRCHAR the single digit that corresponds to GRNUM.
- INTEGER LONG the length of the CMMND command.
- INTEGER GRLO,GRHI limits for COMTAL graphics plane number.
- INTEGER CMLO,CMHI limits on COMTAL coordinates.
- INTEGER FCLO,FCHI limits on FACTOR.
- INTEGER STLO,STHI limits on string LENGTH.
- INTEGER TERM logical unit for terminal output.

**BUFFER:**

- CHARACTER CBUF buffer for CMMND commands
INTEGER IBUF(128) ! overlays CBUF
EQUIVALENCE (CBUF,IBUF)
C
C***INITIALIZATIONS:
DATA GRLO/1/.GRHI/4/
DATA CMLO/0/.CMHI/511/
DATA FClO/1/.FCHI/16/
DATA STLO/0/.STHI/255/
DATA TERM/1/
C
C***PROCESSING:
IF (.NOT.(RANGE(_RNUM,GRLO,GRHI))) GOTO 8001 ! error return
IF (.NOT.(RANGE(XCOOR,CMLO,CMHI))) GOTO 8101 ! error return
IF (.NOT.(RANGE(YCOOR,CMLO,CMHI))) GOTO 8201 ! error return
IF (.NOT.(RANGE(FACTDR,FCLO,FCHI))) GOTO 8301 ! error return
IF (.NOT.(RANGE(LENGTH,STLO,STHI))) GOTO 8401 ! error return
GRCHAR = DIGIT(_RNUM) ! initialization delayed until _RNUM checked.
IF ((COLOR .EQ. 'S').OR.(COLOR .EQ. 's')) GOTO 2000 ! "Same"
IF ((COLOR .EQ. 'R').OR.(COLOR .EQ. 'r')) GOTO 1000 ! "Red"
IF ((COLOR .EQ. 'G').OR.(COLOR .EQ. 'g')) GOTO 1100 ! "Green"
IF ((COLOR .EQ. 'B').OR.(COLOR .EQ. 'b')) GOTO 1200 ! "Blue"
ELSE...COLOR an illegal character
GOTO 6501 ! error return
C
C Color graphics red
CALL CMN2('CO G '//GRCHAR//' RED')
GOTO 2000
C
C Color graphics green
CALL CMN2('CO G '//GRCHAR//' GRN')
GOTO 2000
C
C Color graphics blue
CALL CMN2('CO G '//GRCHAR//' BLU')
GOTO 2000
C
CAL%'DSPCR(_RNUM) ! display the chosen graphics
CALL WRTR(XCOOR, YCOOR) ! position the cursor for writing
C
IF (FACTOR .GE. 10) GOTO 3000
ELSE...
CBUF = 'G '//GRCHAR//' L '//DIGIT(FACTOR) '/'//NOTE
0137    LONG = LENGTH + 8
0138    GOTO 4000
0139 C    THEN...
0140 3000 CBUF = 'G'//GRCHAR//' L 1'//DIGIT(FACTOR-10)//' '/*NOTE
0141    LONG = LENGTH + 9
0142    GOTO 4000
0143 C
0144 4000 CALL CMDND(IBUF, LONG)
0145 C
0146 C    Delay for a second or two to relieve timing problem between
0147 C    the HP and the COMTAL
0148    CALL DELAY((FACTOR/4)+1)
0149    RETURN
0150 C
0151 C**ERROR RETURNS:
0152 C
0153 8001 WRITE(TERM,8003)GRNUM, GRLO, GRHI
0154 8003 FORMAT(' THE GRAPHICS NUMBER,'I5,' IS OUT OF RANGE:'I14,'')
0155    GOTO 8900
0156 C
0157 8101 WRITE(TERM,8103)XCOORD, CMLO, CMHI
0158 8103 FORMAT(' THE X COORDINATE,'I5,' IS OUT OF RANGE:'I14,'')
0159    GOTO 8900
0160 C
0161 8201 WRITE(TERM,8203)YCOORD, CMLO, CMHI
0162 8203 FORMAT(' THE Y COORDINATE,'I5,' IS OUT OF RANGE:'I14,'')
0163    GOTO 8900
0164 C
0165 8301 WRITE(TERM,8303)FACTOR, FCLO, FCHI
0166 8303 FORMAT(' THE SCALE FACTOR,'I5,' IS OUT OF RANGE:'I14,'')
0167    GOTO 8900
0168 C
0169 8401 WRITE(TERM,8403)LENGTH, STLO, STHI
0170 8403 FORMAT(' THE STRING LENGTH,'I5,' IS OUT OF RANGE:'I14,'')
0171    GOTO 8900
0172 C
0173 8501 WRITE(TERM,8503)COLOR
0174 8503 FORMAT(' THE COLOR PARAMETER,'A1,' IS NOT S, R, G, OR B.')
0175    GOTO 8900
0176 C
0177 8900 WRITE(TERM,8901)
0178 8901 FORMAT(' *NOTES FAILS. NO LETTERING PLACED INTO GRAPHICS.*')
0179 C
0180    RETURN
0181 C
0182 C
0183 C
SUBROUTINE PAINT(IMAGE, BRUSH, SHADE)

INTEGER IMAGE ! number of the COMTAL image to be painted
INTEGER BRUSH ! size of the square brush area
INTEGER SHADE ! the pixel value to be brushed on

The subroutine PAINT allows the interactive user to paint
onto a COMTAL image, using the trackball to guide the brush.
The user uses the HP keyboard to signal when to paint.
Each time the HP <CR> is pressed, PAINT paints a BRUSH X BRUSH
square of pixels with the cursor position in the upper left corner
of the square. The maximum size for a brush has been set (arbitrarily)
to 64 pixels square.

The COMTAL keyboard is inaccessible to the HP. Therefore, we
must use the HP keyboard even though we use the COMTAL trackball.
Because of the HP keyboard limitations, the program requires
a pointillistic painting: one square in the image is darkened
each time the HP <CR> is pressed.

name source load remarks

DSPBW &DSPBW XDSPBW display a monochrome COMTAL image.
URIRC &URIRC XURIRC writes the contents of an array to a COMTAL image:
the values are read into a rectangle in the image.
RDTAR &RDTAR XRDTR reads the current COMTAL cursor position.
WAIT &WAIT XWAIT halts HP processing until HP <CR> entered.
CMND &CMND XCMND sends a character string to COMTAL, which
treats the string as a COMTAL keyboard command.
0045 **WRITTEN BY:**
0046 C
0047 C The code on which this subprogram is based was written by
0048 C NETTIE D. FAULCON, July, 1983. This subprogram was written by
0049 C KEITH MILLER, July, 1984, with the support of a NASA-ASEE
0050 C summer fellowship.
0051 C
0052 **REVISION HISTORY:**
0053 C
0054 C
0055 **LOCAL VARIABLES:**
0056 C
0057 INTEGER BOX(64*64) ! the square that acts as a paint brush-
0058 C this rectangle placed into IMAGE whenever
0059 C a <CR> entered to locate cursor on COMTAL.
0060 INTEGER XPOS ! loop index for initializing BOX.
0061 INTEGER UPLFX, UPLFY ! X and Y coordinates of last COMTAL cursor
0062 INTEGER TERM ! logical unit for terminal I/O.
0063 INTEGER IMLO, IMHI ! limits on COMTAL image number.
0064 INTEGER PXLO, PXHI ! limits on COMTAL pixel values.
0065 INTEGER BRLO, BRHI ! limits on size of brush for painting.
0066 INTEGER INCHAR ! character buffer for HP keyboard input.
0067 INTEGER IBUF(128) ! buffer for sending CMDND strings
0068 LOGICAL RANGE ! function that determines if 1st argument
0069 C is within the 2nd and 3rd, inclusive.
0070 CHARACTERS CBUF ! character overlay for IBUF
0071 EQUIVALENCE (IBUF,CBUF)
0072 C
0073 **INITIALIZATIONS:**
0074 C
0075 DATA IMLO/1/, IMHI/4/
0076 DATA BRLO/1/, BRHI/128/
0077 DATA PXLO/0/, PXHI/255/
0078 DATA TERM/1/
0079 C
0080 **PROCESSING**
0081 C
0082 IF (.NOT.(RANGE(IMAGE, IMLO, IMHI))) GOTO 8001 ! error return
0083 IF (.NOT.(RANGE(BRUSH, BRLO, BRHI))) GOTO 8101 ! error return
0084 IF (.NOT.(RANGE(SHADE, PXLO, PXHI))) GOTO 8201 ! error return
0085 C
0086 C 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
400 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.',
     ' ENTER S<CR> TO EXIT PAINTING.').

Loop for input/painting starts here:

1000 INCHAR = '('
READ(TERM, 1001) INCHAR
1001 FORMAT(A1)
IF (INCHAR .EQ. '"') GOTO 2000 ! paint another square and continue.
IF ((INCHAR .EQ. 'S') .OR. (INCHAR .EQ. 's')) GOTO 9000 ! terminate
ELSE...illegal entry
WRITE(TERM, 1003) INCHAR
1003 FORMAT(' THE CHARACTER ENTERED, "', 1A1, '" IS NOT LEGAL',
     ' FOR PAINTING., '/PLEASE TRY AGAIN.')
GOTO 1000

Actual painting takes place here:

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

***ERROR RETURNS:

8001 WRITE(TERM, 8003) IMAGE, IMLO, IMHI
0137  8003 FORMAT(' IMAGE NUMBER.', I5, ', OUT OF RANGE: ', 214, '.')
0138           GOTO 8900
0139  C
0140  8101 WRITE(TERM, 8103) BRUSH, BRLO, BRHI
0141  8103 FORMAT(' BRUSH ARGUMENT.', I5, ', OUT OF RANGE: ', 214, '.')
0142           GOTO 8900
0143  C
0144  8201 WRITE(TERM, 8203) SHADE, PXLO, PXSH
0145  8203 FORMAT(' SHADE ARGUMENT.', I5, ', 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          END
0154
SUBROUTINE PROFIL(GRAPH, IMAGE)

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-6/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:

name       source    load     remarks
---------   ---------  ---------
RANGE      &RANGE    &RANGE   logical function that determines if 1st argu-
                   ment is within 2nd and 3rd, inclusive.
CMNND      &CMNND    &CMNND   sends a command to the COMTAL as if it were
                   typed at the COMTAL keyboard.
WAIT       &WAIT     &WAIT    halts HP processing until <CR> is pressed on
                   the HP keyboard.
CHAR       -------   -------   intrinsic HP FORTRAN77 function that converts
                   integers into characters
C DIGIT &DIGIT %DIGIT character*1 function that returns '0'..'9'
according to integer input 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:

CHARACTER*1 GCHAR ! the character that corresponds to the single
digit argument, GRAPH

CHARACTER*1 ICHAR ! the character that corresponds to the single
digit argument, IMAGE

CHARACTER*1 BELL ! the ASCII code for ESCAPE.

CHARACTER*1 DIGIT ! function that returns '0'..'9' according to
! integer argument 0..9.

CHARACTER CHAR ! intrinsic FORTRAN77 function for integer to
! character conversion.

INTEGER IBUF(128) ! integer buffer for sending COMTAL commands
CHARACTER*255 CBUF ! overlays IBUF

EQUIVALENCE (IBUF,CBUF)

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

C****INITIALIZATIONS:

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

C****PROCESSING

IF (.NOT.(RANGE(GRAPH,GRLO,GRHI))) GOTO 8001
IF (.NOT.(RANGE(IMAGE,IMLO,IMHI))) GOTO 8101
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/' $CL G '/GCHAR/'
1 ' SCO G '/GCHAR/' RED'

CALL CMND(IBUF, 34)

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

CBUF = 'A TA $G '/GCHAR/' PRO '/ICHAR

CALL CMND(IBUF, 15)

print out instructions for the COMTAL Keyboard interaction

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

CALL WAIT
RETURN

ERROR RETURNS:

8001 WRITE(TERM, 8003)GRAPH, GRLO, GRHI
8003 FORMAT(' THE GRAPH PARAMETER, ',I4,', IS OUT OF RANGE:',213)
GOTO 8900

8101 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)

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

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.

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

Keith Miller, NASA-Langley ASEE fellow, 1984

INTEGER TERM ! logical unit for terminal output

DATA TERM /1/

IF (LOW .GT. HIGH) WRITE(TERM, 1001) LOW, HIGH
1001 FORMAT(‘LOW,’ ,15,’ ,’ ,GT. HIGH,’ ,15,’ ,’ ,RANGE fails.’)

RANGE = ((OBJECT .GE. LOW) .AND. (OBJECT .LE. HIGH))
RETURN
END
SUBROUTINE RDGLN (ONOFFS, GRAPH, LINE)

INTEGER ONOFFS(32) ! 16 bits per integer, 512 bits in a line.
INTEGER GRAPH ! COMTAL image graphics to be written to.
INTEGER LINE ! which horizontal line to be written to:
INTEGER LINE ! lines numbered from 1 (screen top) to 512.

C***INTRODUCTION:
This subroutine Reads a Graphics Line from the COMTAL. The line of bits is coded as 1 for on and 0 for off.

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

C***LIMITATIONS:
If GRAPH or LINE is out of range, an error message is printed and no transfer takes place.

C***SUBPROGRAMS CALLED:

C***WRITTEN BY:
The code on which this subprogram is based was written by
NETTIE D. FRULCON, July, 1963. This modification is by
KEITH MILLER, June, 1984.

C***REVISION HISTORY:

C***LOCAL VARIABLES:

LOGICAL RANGE  ! function that ascertains if its first parameter
C INTEGER TERM ! the logical unit for terminal output
C INTEGER GRLO, GRHI ! the limits on COMTAL monochrome image numbers
C INTEGER LNLO, LNHI ! the limits on COMTAL image line numbers

C***INITIALIZATIONS:

DATA TERM/1/
DATA GRLO/1/ , GRHI/4/
DATA LNLO/8/ , LNHI/511/

C***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)

RETURN

C***ERROR RETURNS

WRITE(TERM, 0003) GRAPH, GRLO, GRHI
0003 FORMAT(‘ GRAPHICS NUMBER,’ , 13, ‘ OUT OF RANGE:’, 212,’.‘)
GOTO 8900

C
WRITE(TERM, 8103) LINE, LNLO, LNHI
8103 FORMAT(‘ LINE NUMBER,’ , 14, ‘ OUT OF RANGE:’, 213,’.‘)
GOTO 8900

C
WRITE(TERM, 0900)
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.
- BTEST ------ ------ HP FORTRAN77 intrinsic 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.
LOGICAL RANGE
   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 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 XCOOR.
INTEGER BIT
   which bit in BITS(WORD) holds the bit selected by
      XCOOR: bits numbered 0-15, right to left.

C***INITIALIZATIONS:
DATA TERM/I/
DATA GRLO/I/, GRHI/4/
DATA LNLO/0/, LNHI/511/

C***PROCESSING

IF (.NOT.(RANGE(GRAPH, GRLO, GRHI))) GOTO 8801 ! error return
IF (.NOT.(RANGE(XCOOR, LNLO, LNHI))) GOTO 8101 ! error return
IF (.NOT.(RANGE(YCOOR, LNLO, LNHI))) GOTO 8201 ! 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, 368+1009, BITS, 32, (GRAPH-1)*2048 + 512 + YCOOR)

Find the single bit that has been selected:

WORD = (XCOOR/16) + 1
BIT = (16*WORD) - XCOOR - 1

VALUE = 0 ! bit is clear until proven set.
IF (BTEST(BITS(WORD), BIT)) VALUE = 1
RETURN

C

***ERROR RETURNS

C

C001 WRITE(TERM, 8003) GRAPH, GRLO, GRHI

C003 FORMAT(' GRAPH NUMBER', IS, ' OUT OF RANGE:', 212, '.')

C005 GOTO 8900

C

C101 WRITE(TERM, 8103) XCOORD, LNLO, LNHI

C103 FORMAT(' X COORDINATE', IS, ' OUT OF RANGE:', 214, '.')

C105 GOTO 8900

C

C201 WRITE(TERM, 8203) YCOORD, LNLO, LNHI

C203 FORMAT(' Y COORDINATE', IS, ' OUT OF RANGE:', 214, '.')

C205 GOTO 8900

C

C900 WRITE(TERM, 8901)

C903 FORMAT(' RDGPT FAILS. NO TRANSFER.')

C905 RETURN

C910 END
&RDIL2 T=00004 IS ON CR00021 USING 00010 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE RDIL2(INTS, IMAGE, LINE)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***PARAMETER DECLARATIONS:
0006 C
0007 INTEGER INTS(512) ! 512 pixel values to be read, one integer/pixel
0008 INTEGER IMAGE ! COMTAL image number to be read from
0009 INTEGER LINE ! which horizontal line is to be read from;
0010 C
0011 C
0012 C***INTRODUCTION:
0013 C
0014 C This subroutine Reads on 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 RDIL2.
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 %RANGE %RANGE logical function that determines if its
0033 C first parameter is within the last two parameters.
0034 C ICHAR ------- ------ intrinsic HP FORTRAN77 function that converts
0035 C a character (1 byte) into an integer (2 bytes)
0036 C
0037 C***WRITTEN BY:
0038 C
0039 C The code on which this subprogram is based was written by
0040 C NETTIE D. FAULCON, July, 1983. This modification is by
0041 C KEITH MILLER, June, 1984.
0042 C
0043 C***REVISION HISTORY:
0044 C
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/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

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)*2848 + 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
1000 CONTINUE
RETURN
**ERROR RETURNS**

```plaintext
0091 C
0092 C
0093 C
0094 0081 WRITE(TERM, 0003) IMAGE, IMLO, IMHI
0095 0083 FORMAT(' IMAGE NUMBER, ', I3, ' OUT OF RANGE: ', 212, '. ')
0096     GOTO 0900
0097 C
0098 0101 WRITE(TERM, 0103) LINE, LNLO, LNHI
0099 0103 FORMAT(' LINE NUMBER, ', I4, ' OUT OF RANGE: ', 213, '. ')
0100     GOTO 0900
0101 C
0102 0900 WRITE(TERM, 0901)
0103 0901 FORMAT(' RDIL2 FAILS. NO TRANSFER. ')
0104     RETURN
0105     END
```
&RDILN T=00004 IS ON CR00021 USING 00018 BLKS R=0000

0001 CCCCCCCCCCC
0002 SUBROUTINE RDILN(PIXELS, IMAGE, LINE)
0003 CCCCCCCCCCC
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 ! lines numbered from 1 (screen top) to 512.
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 XRANGE 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
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

**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: 36B identifies the resident
DVR41 driver, and 1008 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(1, 36B+1008, PIXELS, 256, (IMAGE-1)*2048 + LINE)
RETURN

**ERROR RETURNS**
WRITE(TERM, 8003) IMAGE, IMLO, IMHI
8003 FORMAT(’, IMAGE NUMBER:’, I3, ’ OUT OF RANGE:’, 212,’.’)
GOTO 8900

WRITE(TERM, 8103) LINE, LNLO, LNHI
8103 FORMAT(’, LINE NUMBER:’, I4, ’ OUT OF RANGE:’, 213,’.’)
GOTO 8900

WRITE(TERM, 8901)
6901 FORMAT(’ RDILN FAILS. NO TRANSFER.’)
RETURN

END
&RDIP T=0004 IS ON CR00021 USING 0018 BLKS R=0000

0011 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0012 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0013 C
0014 C
0015 C
0016 C
0017 C
0018 C
0019 C
0020 C
0021 C
0022 C
0023 C
0024 C
0025 C
0026 C
0027 C
0028 C
0029 C
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

SUBROUTINE RDIPT(VALUE, IMAGE, XCOORD, YCOORD)

**PARAMETER DECLARATIONS:**

INTEGER VALUE  ! receives pixel value, converted from byte
INTEGER IMAGE  ! COMTAL image number to be read from.
INTEGER XCOORD, YCOORD  ! point where value is to be read from.

**INTRODUCTION:**

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

**LANGUAGE:**

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

**LIMITATIONS:**

The IMAGE designated
must be a monochrome image. The YCOORD parameter must be between
0 and 511. If IMAGE or YCOORD is out of range, an error message is printed
and no transfer takes place.

**SUBPROGRAMS CALLED:**

RANGE &RANGE  xRANGE 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.

**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.
C**REVISION HISTORY:
0045

C**LOCAL VARIABLES:
0048

C

C

C

C

C

C

C

C

C

LOCAL VARIABLES:
0050

LOGICAL RANGE ! function that ascertains if its first parameter
0051

is between (inclusive) its last 2 parameters
0052

INTEGER TERM ! the logical unit for terminal output
0053

INTEGER IMLO, IMHI ! the limits on COMTAL monochrome image numbers
0054

INTEGER LNLO, LNHI ! the limits on COMTAL image line numbers
0055

INTEGER PIXELS(256) ! a buffer to read a COMTAL line
0056

CHARACTER*1 BYTES(512) ! overlay for PIXELS buffer
0057

EQUIVALENCE (PIXELS, BYTES)

C

C**INITIALIZATIONS:
0059

C

DATA TERM/1/
0061

DATA IMLO/1/, IMHI/4/
0062

DATA LNLO/0/, LNHI/511/
0063

C

C**PROCESSING
0065

C

IF (.NOT.(RANGE(IMAGE, IMLO, IMHI))) GOTO 8001 ! error return
0067

IF (.NOT.(RANGE(YCOORD, LNLO, LNHI))) GOTO 8101 ! error return
0068

IF (.NOT.(RANGE(XCOORD, LNLO, LNHI))) GOTO 8201 ! error return
0069

C

Programming note:
0071

C

The EXEC call below is to the DVR41 driver, and
0072

C

is identical to the call made in RDILN.
0073

C

See the RDILN documentation for the details.
0074

C

Read the COMTAL line (horizontal) that contains the point in question:
0076

C

CALL EXEC(1, 36B+100B, PIXELS, 256, (IMAGE-1)*2048 + YCOORD)
0078

C

convert the byte that is to be read:
0080

C

VALUE = ICHAR(BYTES(XCOORD+1)) ! ICHAR is an intrinsic F77 function
0082

C

which converts a character into its
0083

C

integer code.
0084

C

! The '1' changes from pixels, which are
0085

C

! fed 0-511 to the FORTRAN array, 1-512
0086

C

RETURN
0087

C

C**ERROR RETURNS
0089

C
0091 0803 WRITE(TERM, 8003) IMAGE, IMLO, IMHI
0092 8003 FORMAT(‘ IMAGE NUMBER,’ , I3, ’ OUT OF RANGE:’ , 212,’ ’)
0093       GOTO 8900
0094       C
0095 8101 WRITE(TERM, 8103) YCOORD, LNLO, LNHI
0096 8103 FORMAT(‘ Y COORDINATE,’ , I4, ’ OUT OF RANGE:’ , 214,’ ’)
0097       GOTO 8900
0098       C
0099 8201 WRITE(TERM, 8203) XCOORD, LNLO, LNHI
0100 8203 FORMAT(‘ X COORDINATE,’ , I4, ’ OUT OF RANGE:’ , 214,’ ’)
0101       GOTO 8900
0102       C
0103 8900 WRITE(TERM, 8901)
0104 8901 FORMAT(’ RD1PT FAILS. NO TRANSFER.’)
0105       RETURN
0106       END
&RDIRC T=0004 IS ON CR00021 USING 00004 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE RDIRC(OUTARA, XDIM, YDIM, IMAGE, UPLFX, UPLFY)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***PARAMETER DECLARATIONS:
0006 C
0007 INTEGER XDIM, YDIM ! dimensions of the output array, OUTARA
0008 INTEGER OUTARA(XDIM, YDIM) ! the array to be filled
0009 INTEGER IMAGE ! the number of the COMTAL image from
0010 C
0011 INTEGER UPLFX, UPLFY ! the image coordinates of the upper left
0012 C
0013 C
0014 C
0015 C***INTRODUCTION:
0016 C
0017 C
0018 C The subroutine Read Image Rectangle transfers pixel values from a
0019 C designated section of an image to an integer array. Note that although
0020 C pixel values are generally stored with 1 byte/pixel, RDIRC places each
0021 C numeric value into a 2 byte integer in OUTARA.
0022 C XDIM, YDIM, IMAGE, UPLFX, and UPLFY are all checked for possible out
0023 C of range errors before any transfer is attempted.
0024 C
0025 C***LANGUAGE:
0026 C
0027 C FORTRAN 77, the HP-1000 version for RTE-VM.
0028 C
0029 C***LIMITATIONS:
0030 C
0031 C RDIRC does a great deal of error checking before initiating the
0032 C transfer. If the programmer can verify that all calls to RDIRC
0033 C will be legal, these checks could be commented out to improve
0034 C machine efficiency. Also, the call to the subroutine RDILN could
0035 C be replaced inline by the RDILN code (which is short).
0036 C
0037 C***SUBPROGRAMS CALLED:
0038 C
0039 C
0040 name source load remarks
0041 C RDILN &RDILN &RDILN transfers a line of bytes from a COMTAL image
0042 C RANGE &RANGE &RANGE logical function that determines if its first
0043 C parameter is within the last two parameters
0044 C ICHAR ------ ------ HP FORTRAN77 intrinsic function: converts a
byte into its integer code.

The code on which this subprogram is based was written by
NETTIE D. FAULCON, July, 1983. This code was written by

**Revision History:**

Local Variables:

<table>
<thead>
<tr>
<th>INTEGER IMROW, IMCOL</th>
<th>a location in the image</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEGER ARAROW, ARACOL</td>
<td>a location in OUTARA</td>
</tr>
<tr>
<td>INTEGER ICHAR</td>
<td>intrinsic byte to integer conversion</td>
</tr>
<tr>
<td>LOGICAL RANGE</td>
<td>function that ascertains if the first</td>
</tr>
<tr>
<td>INTEGER IMLO, IMHI</td>
<td>parameter is within the last two parameters</td>
</tr>
<tr>
<td>INTEGER LNLO, LNHI</td>
<td>limits on COMTAL image numbers</td>
</tr>
<tr>
<td>INTEGER LCNT</td>
<td>LNHI-LNLO+1, number of pixels in an image line</td>
</tr>
<tr>
<td>INTEGER TERM</td>
<td>logical unit for terminal output</td>
</tr>
<tr>
<td>INTEGER IBUF(256)</td>
<td>buffer to hold COMTAL horizontal line</td>
</tr>
<tr>
<td>CHARACTER CBUF(512)</td>
<td>overlay for IBUF</td>
</tr>
</tbody>
</table>

Equivalence (IBUF, CBUF)

**Initialization:**

| DATA TERM/1/ |
| DATA IMLO/1/, IMHI/4/ |
| DATA LNLO/8/, LNHI/511/, LCNT/512/ |

**Processing:**

- IF (.NOT.(RANGE(IMAGE, IMLO, IMHI)))GOTO 8001 ! error return
- IF (.NOT.(RANGE(XDIM, 1, LCNT)))GOTO 8101 ! error return
- IF (.NOT.(RANGE(YDIM, 1, LCNT)))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+XDIM-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 RDILH(IBUTF, 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 COMTAB image
coordinates range from 0 to 511 and the
! 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
8661 WRITE(TERM, 8603) IMAGE, IMLO, IMHI
8663 FORMAT( ' IMAGE NUMBER: ', I3, ' OUT OF RANGE: ', 214, ' ' )
8665 GOTO 8900
8667 WRITE(TERM, 8603) XDIM, I, LNCNT
8669 FORMAT( ' X DIMENSION, ', I4, ' OUT OF RANGE: ', 215, ' ' )
8671 GOTO 8900
8673 WRITE(TERM, 8603) YDIM, I, LNCNT
8675 FORMAT( ' Y DIMENSION, ', I4, ' OUT OF RANGE: ', 215, ' ' )
8677 GOTO 8900
8679 WRITE(TERM, 8603) UPLFX, LNLO, LNHI
8681 FORMAT( ' X COORDINATE FOR CORNER, ', I4, ' OUT OF RANGE: ', 215, ' ' )
8683 GOTO 8900
8685 WRITE(TERM, 8603) UPLFY, LNLO, LNHI
8687 FORMAT( ' Y COORDINATE FOR CORNER, ', I4, ' OUT OF RANGE: ', 215, ' ' )
8689 GOTO 8900
8691 WRITE(TERM, 8603) UPLFX, XDIM, LNLO, LNHI
8693 FORMAT( ' X COORDINATE FOR THE CORNER AND THE X DIMENSION ',
8695 ' OF THE ARRAY', ', ', ' OVERFLOW IMAGE BOUNDARIES', ').'
8697 WRITE(TERM, 8603) UPLFY, XDIM, LNLO, LNHI
8699 FORMAT( ' Y COORDINATE FOR THE CORNER AND THE Y DIMENSION ',
8701 ' OUT OF RANGE: ', 215, ' ' )
8703 GOTO 8900
8705 WRITE(TERM, 8603) UPLFX, XDIM, LNLO, LNHI
8707 FORMAT( ' X COORDINATE = ', I4, ' X DIMENSION = ', I4, ' ' )
8709 WRITE(TERM, 8603) UPLFY, XDIM, LNLO, LNHI
8711 FORMAT( ' Y COORDINATE LIMITS ARE ', 215, ' ' )
8713 GOTO 8900
8715 WRITE(TERM, 8603) UPLFX, XDIM, LNLO, LNHI
8717 FORMAT( ' X COORDINATE =', I4, ' X DIMENSION =', I4, ' ' )
8719 WRITE(TERM, 8603) UPLFY, XDIM, LNLO, LNHI
8721 FORMAT( ' Y COORDINATE LIMITS ARE', 215, ' ' )
8723 GOTO 8900
8725 WRITE(TERM, 8603) UPLFX, XDIM, LNLO, LNHI
8727 FORMAT( ' X COORDINATE = ', I4, ' X DIMENSION = ', I4, ' ' )
8729 WRITE(TERM, 8603) UPLFY, XDIM, LNLO, LNHI
8731 FORMAT( ' Y COORDINATE LIMITS ARE ', 215, ' ' )
8733 GOTO 8900
8735 WRITE(TERM, 8603) UPLFX, XDIM, LNLO, LNHI
8737 FORMAT( ' X COORDINATE = ', I4, ' X DIMENSION = ', I4, ' ' )
8739 WRITE(TERM, 8603) UPLFY, XDIM, LNLO, LNHI
8741 FORMAT( ' Y COORDINATE LIMITS ARE ', 215, ' ' )
8743 GOTO 8900
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
&RDLUT T=00004 IS ON CR00021 USING 00818 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE RDLUT(TABLE, LUTNUM)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
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.
**C***REVISION HISTORY:

**C***LOCAL VARIABLES:

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0051</td>
<td>LOGICAL RANGE</td>
</tr>
<tr>
<td>0052</td>
<td>INTEGER TERM</td>
</tr>
<tr>
<td>0053</td>
<td>INTEGER LUTLO, LUTHI</td>
</tr>
</tbody>
</table>

**C***INITIALIZATIONS:

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0058</td>
<td>DATA TERM/1/</td>
</tr>
<tr>
<td>0059</td>
<td>DATA LUTLO/1/, LUTHI/4/</td>
</tr>
</tbody>
</table>

**C***PROCESSING:

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0063</td>
<td>IF (.NOT.(RANGE(LUTNUM, LUTLO, LUTHI))) GOTO 8001</td>
</tr>
</tbody>
</table>

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 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).

**C***ERROR RETURN:

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0087</td>
<td>WRITE(TERM, 8003) LUTNUM, LUTLO, LUTHI</td>
</tr>
<tr>
<td>0088</td>
<td>WRITE(TERM, 8003) FORMAT( ' THE FUNCTION MEMORY NUMBER, ', I4, ', IS OUT OF RANGE:' )</td>
</tr>
<tr>
<td>0089</td>
<td>1 214, ',')</td>
</tr>
<tr>
<td>0090</td>
<td>WRITE(TERM, 8901)</td>
</tr>
</tbody>
</table>
0091  8901 FORMAT(' RDLUT FAILS. NO TRANSFER FROM COMTAL.')
0092    END
&RDPSU T=00004 IS ON CR00021 USING 00018 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 C
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
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 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  NONE
C
C***PROCESSING
C
Programming notes:
C
The EXEC command parameters are discussed in the HP RTE-6/VM
C
C
are discussed in section 5.2.3 of the COMTAL User's Manual.
C
C
The first parameter to EXEC identifies the EXEC command as
C
a read command. The second parameter identifies the resident
C
HP driver (36B) and gives the code (300B) that identifies this
C
operation, a transfer from the COMTAL pseudocolor table.
C
The third parameter gives the array that will hold the values,
C
and the fourth parameter gives the length of TABLE in words.
C
The fifth parameter is a COMTAL code that is described bit by
C
bit in the User's Manual. The DVR41 driver takes care of all the
C
bits except 8 & 9 which identify the color to be transferred.
C
Note that we make three separate calls to EXEC. Each call fills a
C
different section of TABLE with a different color of the COMTAL's
C
pseudocolor table.
C
CALL EXEC( 1, 36B+300B, TABLE(1), 256, 1*256 ) ! red
C
CALL EXEC( 1, 36B+300B, TABLE(257), 256, 0*256 ) ! green
C
CALL EXEC( 1, 36B+300B, TABLE(513), 256, 2*256 ) ! blue
C
RETURN
C
END
**&RDTAB T=000004 IS ON CR00021 USING 00016 BLKS R=0000**

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
**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: is the logical unit for terminal output.
- INTEGER DGLO, DGHI: the limits on single Digital image/graphics numbers.
- INTEGER BTLO, BTHI: the limits on COMTAL graphics values.
- XCOORD: bits numbered 0-15, right to left.

**INITIALIZATIONS:**

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

**PROCESSING**

```c
IF (.NOT.(RANGE(CNUMB,DGLO,DGHI))) GOTO 8001 ! error return
IF (.NOT.(RANGE(IMOGR,BTLO,BTHI))) GOTO 8101 ! error return
```

Programming note:

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

The second argument has two parts: "3GB" identifies the DVR41 driver,
and "500B" 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(1,36B*500B,TABLE,16, 7*4096 + IMOGR*128 + CNUMB-1)

**ERROR RETURNS**

```c
8001 WRITE(TERM, 8003) CNUMB, DGLO, DGHI
8003 FORMAT( 'COMTAL NUMBER:"', 13, 'OUT OF RANGE:"', 212,'"')
GOTO 8900
```

```c
8101 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 CR00021 USING 00012 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE RDTAR(XCOORD, YCOORD)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
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 C 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

109
follows: the first parameter (1) signifies a read; the
second parameter is in two parts: 36B identifies the resident
DVR41 driver, and 4008 identifies the target transfer operation
of that driver; the third parameter (IBUF) will hold the COMTAL data
to be transferred, and the fourth parameter gives the length in words
(2); the final parameter is a COMTAL command code for the transfer,

CALL EXEC(1, 36B+4008, IBUF, 2, 00000B)
XCOORD = IBUF(1)
YCOORD = IBUF(2)
RETURN
END
&SETV  T=-00004 IS ON CR00021 USING 00004 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE SETV (IMAGE)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
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 CMNND &CMNND %CMNND sends character strings to the COMTAL where
0035 C they are much like COMTAL keyboard commands.
0036 C DIGIT &DIGIT %DIGIT character*1 function that returns '0'-9'
0037 C according to integer input 0-9.
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:**

**INITIALIZATIONS:**

**PROCESSING:**

```c
IF (.NOT.(RANGE(IMAGE.VDLO,VDHI))) GOTO 8801 | error return
```

```c
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.
```

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

```c
RETURN
```

**ERROR RETURNS:**
C
0091 WRITE(TERM,8003) IMAGE, VDLO, VDHI
0092 8003 FORMAT(' YOUR IMAGE ARGUMENT,', I4,' IS OUT OF RANGE:', 2I4)
0093 C
0094 WRITE(TERM, 8901)
0095 8901 FORMAT(' SETV FAILS. NO ACTION TAKEN.')
0096 C
0097 RETURN
0098 END
0100
&SPLIT T=60004 IS ON CR00021 USING 00005 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE SPLIT(BYTE1, BYTE2, INBYTE)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C****PARAMETER DECLARATIONS:
0006 C
0007 INTEGER BYTE1  ! on output, this integer holds the left (high order)
0008       ! byte of INBYTE in its right (low order) byte.
0009 INTEGER BYTE2  ! on output, this integer holds the right (low order)
0010       ! byte of INBYTE in its right (low order) byte.
0011 INTEGER INBYTE  ! the input integer whose two bytes are to be SPLIT.
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 C 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 HETTIE 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

114
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 SPRED(IMAGE)

C***PARAMETER DECLARATIONS:

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

C***INTRODUCTION:

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

C***LANGUAGE:

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

C***LIMITATIONS:

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

C***SUBPROGRAMS CALLED:

name source load remarks

RDIL2 GRDIL2 XRDIL2 reads a horizontal line of pixels from the
COMTAL, and places the values in INTEGER array.

WRIL2 GRWRL2 XWRIL2 writes a horizontal line of pixels from an
integer array to a COMTAL image.

HILO GHIL0 XHILO passes through a COMTAL image and returns the
highest and lowest pixel value found.

IFIX ------ ------ FORTRAN IV intrinsic function; converts from
real to integer.

FLOAT ------ ------ FORTRAN IV intrinsic function; converts from
C integer to real.

C

C****WRITTEN BY:

C

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

C

NETTIE D. FAULKNER, 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

INTEGEB IBUF(512) ! holds pixel values read/written to/from COMTAL

C

INTEGEB HIGH, LOU ! high and low pixel values in the IMAGE

C

INTEGEB PXHI, PXLO ! limits of pixel values.

C

INTEGEB LNLO, LNHI ! limits on COMTAL line numbers

C

INTEGEB ROW, COL ! indices for COMTAL images and the arrays.

C

INTEGEB TERM ! logical unit for terminal output

C

REAL FACTOR ! the scaling factor for doing the contrast spread

C

C****INITIALIZATIONS:

C

DATA PXLO/0/, PXHI/255/

C

DATA LNLO/0/, LNHI/511/

C

DATA TERM/1/

C

C****PROCESSING

C

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

C

! in the image

C

IF (HIGH .LE. LOW) GOTO 9999 ! no processing required

C

WRITE(TERM, 3501) HIGH, LOW

C

3501 FORMAT(‘SPREAD DIAGNOSTIC. HIGH AND LOW ARE ’, 214, ’’)

C

IF ((HIGH .EQ. PXHI) .AND. (LOW .EQ. PXLO)) GOTO 9999 ! no processing

C

C

FACTOR = 255.0 / FLOAT(HIGH-LOW)

C

C

During the second pass, replace each pixel in the image with a

C new pixel that has been spread linearly according to HIGH and LOW

C

DO 4000 ROW = 1,512

C

CALL RDIL2(IBUF, IMAGE, ROW)

C

DO 3000 COL = 1,512

C

9995 CONTINUE

C

9999 CONTINUE
IBUF(COL) = IFIX(FLOAT(IBUF(COL)-LOW) * FACTOR)

3000 CONTINUE
CALL URIL2(IMAGE,ROW,IBUF)
4000 CONTINUE
C
C
9999 RETURN
END
&SUBI2 T=00004 IS ON CR00021 USING 00024 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE SUBI2(C, A, B)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
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***SUBPROGRAMS CALLED:

C

C name       source   load   remarks

C---------------

C CMMND &CMNND &CMMND Sends a command to the COMTAL as if the
c command were sent to the keyboard

C RANGE &RANGE &RANGE logical function that determines if the 1st
parameter is within the range of the 2nd & 3rd.

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

0058 C***WRITTEN BY:

0059 C

0060 C KEITH MILLER. July, 1984, with the support of a NASA-ASEE

0061 C summer fellowship.

0062 C

0063 C***REVISION HISTORY:

0064 C

0065 C

0066 C***LOCAL VARIABLES:

0067 C

0068 INTEGER IBUF(128) ! a buffer for passing commands to COMTAL

0069 CHARACTER*255 CBUF ! character overlay for IBUF

0070 EQUIVALENCE (IBUF, CBUF)

0071 C

0072 LOGICAL RANGE ! function that determines if 1st parameter
is within 2nd and 3rd parameter

0073 C

0074 C

0075 INTEGER IMLO, IMHI ! limits on COMTAL image numbers

0076 INTEGER TERM

0077 CHARACTER*1 DIGIT ! returns a single digit character '0'

0078 C ! to '9' for integer input 0-9.

0079 C

0080 C***INITIALIZATIONS:

0081 C

0082 DATA IMLO/1/, IMHI/4/

0083 DATA TERM/1/

0084 C

0085 C***PROCESSING

0086 C

0087 IF (.NOT.(RANGE(A, IMLO, IMHI))) GOTO 8001 ! error return

0088 IF (.NOT.(RANGE(B, IMLO, IMHI))) GOTO 8101 ! error return

0089 IF (.NOT.(RANGE(C, IMLO, IMHI))) GOTO 8201 ! error return

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 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 '
10 CALL CMND(IBUF, 74)
11 RETURN
12 C
13 C
14 C
15 C
16 C
17 C
18 C
19 C
20 C
21 C
22 C
23 C
24 C
25 C
26 8001 WRITE(TERM, 8003) A
27 8003 FORMAT(''THE 2ND IMAGE PARAMETER,'', I3, '' IS OUT OF RANGE.'')
28 GOTO 8900
29 C
30 8101 WRITE(TERM, 8103) B
31 8103 FORMAT(''THE 3RD IMAGE PARAMETER,'', I3, '' IS OUT OF RANGE.'')
32 GOTO 8900
33 C
34 8201 WRITE(TERM, 8203) C
35 8203 FORMAT(''THE 1ST IMAGE PARAMETER,'', I3, '' IS OUT OF RANGE.'')
36 GOTO 8900
C
0138  8900 WRITE(TERM, 8901)
0139  8901 FORMAT('' SUB12 RETURNS WITHOUT FURTHER PROCESSING.'')
0140       RETURN
0141       END
0142
SUBIM T=00004 IS ON CR00021 USING 00004 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE SUBIM(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 SUBIM (C = A - B)
0009 INTEGER A, B ! the images whose difference is taken (C = A - B)
0010 C
0011 C***INTRODUCTION:
0012 C
0013 C The subroutine "SUBtract Images" 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 SUBIM, 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***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 truecolor B image is destroyed during this operation. C is
0026 C obviously destroyed. This subroutine is accomplished using COMTAL
0027 C commands that exploit the pipeline processors. Because of this, the
0028 C processing steps are obscure. For example, there is no motivation
0029 C outside the COMTAL instructions for making the combination of C, A, and B
0030 C a color image. Readers should be aware of these obscurities before trying
0031 C to understand the code.
0032 C
0033 C The 3 images C, A, and B must be distinct.
0034 C
0035 C If any difference is less than 0, the pixel value is set to 0.
0036 C
0037 C SUBIM does no scaling or offsetting. SUB12 does an automatic
0038 C scale and offset.
0039 C
0040 C If any image number is out of range, an error message is printed and
0041 C no further processing takes place.
0042 C This subroutine assumes that 0 is not a legal image for the COMTAL
0043 C configuration.
0044 C
SUBPROGRAMS CALLED:

<table>
<thead>
<tr>
<th>name</th>
<th>source load</th>
<th>remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMDND &amp;CMDND</td>
<td>CMDND</td>
<td>Sends a command to the COMTAL as if the command were sent to the keyboard</td>
</tr>
<tr>
<td>RANGE &amp;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 &amp;DIGIT</td>
<td>DIGIT</td>
<td>character*1 function that 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   | 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   | DIGIT     | returns a single digit character '0' to '9' for integer input 0-9. |

INITIALIZATIONS:

<table>
<thead>
<tr>
<th>DATA</th>
<th>IMLO/1, IMHI/4</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA</td>
<td>TERM/1</td>
</tr>
</tbody>
</table>

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 I B $'/

2 'AS T E '/DIGIT(C)/' '/DIGIT(A)/' '/DIGIT(B)/' $'/'

3 'D I B $'/'

4 'SE COM G - B + B $'/'

5 'A COM $'/'

6 'I '/DIGIT(C)/' D R $'/'

7 'D I '/DIGIT(C)/' $'/'

8 'SU COM '

9 CALL CMDND(IBM, 77)

RETURN

C***ERROR RETURNS

0801 WRITE(TERM, 0803) A

0803 FORMAT(' THE 2ND IMAGE PARAMETER.' , I3, ' IS OUT OF RANGE.')

0804 GOTO 0800

1012 C

1021 0801 WRITE(TERM, 0803) B

0803 FORMAT(' THE 3RD IMAGE PARAMETER.' , I3, ' IS OUT OF RANGE.')

0804 GOTO 0800

1025 C

1029 0801 WRITE(TERM, 0803) C

0803 FORMAT(' THE 1ST IMAGE PARAMETER.' , I3, ' IS OUT OF RANGE.')

0804 GOTO 0800

1032 C

1033 0800 WRITE(TERM, 0801)

0801 FORMAT(' SUBIM RETURNS WITHOUT FURTHER PROCESSING.')

1035 RETURN

1036 END
&THRESH T=E0004 IS ON CRO0021 USING 00018 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0035 C***PARAMETER DECLARATIONS:
0036 C
0037 INTEGER OUTIMG! the COMTAL image number for the thresholded image.
0038 INTEGER INIMG! the COMTAL image & for the image to be thresholded.
0039 INTEGER THRESH! the threshold pixel value: < threshold -> 0,
0040 >= threshold -> 255.
0007 C
0012 C***INTRODUCTION:
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***LANGUAGE:
0021 C
0022 C FORTRAN 77, the HP-1000 version for RTE-6/VM.
0023 C
0024 C***LIMITATIONS:
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.
0029 C
0030 C
0031 C***SUBPROGRAMS CALLED:
0032 C
0033 C name source load remarks
0034 C ------- ------ ------ -----------------------------------------------------
0035 C RANGE &RANGE %RANGE logical function that determines if its 1st
0036 C argument is within the 2nd and 3rd inclusive.
0037 C RDIL2 &RDIL2 %RDIL2 reads a horizontal line of COMTAL pixels:
0038 C each pixel value put into its own integer.
0039 C WRIL2 &WRIL2 %WRIL2 writes a horizontal line of pixels to a COMTAL
0040 C image from an integer array: 1 pixel/integer.
0041 C
0042 C***WRITTEN BY:
0043 C
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:**

**LOCAL VARIABLES:**

**INITIALIZATIONS:**

**PROCESSING:**

```
INTEGER IBUF(512) ! buffer for read/write of horizontal line of
                  ! COMTAL pixels.
INTEGER LINE      ! loop index that keeps track of the line
                  ! in INIMG and OUTIMG being processed.
INTEGER PXCNT     ! loop index that counts pixels in a line.
LOGICAL RANGE     ! function that determines if its 1st argument
                  ! lies within its 2nd and 3rd inclusive.
INTEGER IMLO, IMHI ! range of legal COMTAL image numbers.
INTEGER PXLO, PXHI ! range of legal COMTAL pixel values.
INTEGER LNLO, LNHI ! range of legal COMTAL line numbers.
INTEGER ARALO, ARAHI ! range of array holding a line of pixels;
                      ! (+1 of LNLO and LNHI).
INTEGER TERM      ! logical unit for terminal output.

DATA IMLO/1/, IMHI/4/ ! range of legal COMTAL image numbers.
DATA PXLO/0/, PXHI/255/ ! range of legal COMTAL pixel values.
DATA LNLO/0/, LNHI/511/ ! range of legal COMTAL line numbers.
DATA ARALO/1/, ARAHI/512/ ! range of array holding a line of pixels;
DATA TERM/1/ ! logical unit for terminal output.
```

```
IF (.NOT.(RANGE(INIMG,IMLO,IMHI))) GOTO 8101 ! error return
IF (.NOT.(RANGE(INIMG,IMLO,IMHI))) GOTO 8101 ! error return
IF (.NOT.(RANGE(THRESH,PXLO,PXHI))) GOTO 8201 ! error return

DO 2000 LINE = LNLO, LNHI
    CALL RDIL2(IBUF, INIMG, LINE)
    DO 1000 PXCNT = ARALO, ARAHI
        IF (IBUF(PXCNT) .LT. THRESH) GOTO 500
        ELSE...
            IBUF(PXCNT) = 255
        GOTO 1000
    THEN...
    500 IBUF(PXCNT) = 0
```
CONTINUE

CALL WRIL2(OUTIMG, LINE, IBUF)

CONTINUE

RETURN

ERROR RETURNS:

C

WRITE(TERM, 8003) OUTIMG, IMLO, IMHI

FORMAT( 'THE OUTPUT IMAGE NUMBER,' , I5, ' IS OUT OF RANGE: ' , 214)

GOTO 8900

WRITE(TERM, 8103) INIMG, IMLO, IMHI

FORMAT( 'THE INPUT IMAGE NUMBER,' , I5, ' IS OUT OF RANGE: ' , 214)

GOTO 8900

WRITE(TERM, 8203) THRESH, PXLO, PXHI

FORMAT( 'THE THRESHOLD VALUE,' , I5, ' IS OUT OF RANGE: ' , 214)

GOTO 8900

WRITE(TERM, 8901)

FORMAT( 'THRESH FAILS. OUTIMG NOT CHANGED.' )

RETURN

END
SUBROUTINE TSTII(WHICH)

INTEGER WHICH ! designates the COMTAL image in which the test image is to be generated

INTRODUCTION:

The subroutine TeST Image I generates a distinctive pattern in one of the COMTAL image memories. The pattern is generated with ascending pixel values as you move to the right and down in the image. However, when 255 is reached in either the x or y directions, the pixel values restart at 0. This gives a slash across the screen appearance at the discontinuity, and a gradual change in grey scale elsewhere.

LANGUAGE:

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

LIMITATIONS:

The test pattern developed here is always the same. An enhancement might be to vary the repetition of the pixel values (here, each pixel is identical to one of its horizontal neighbors and one of its vertical neighbors).

SUBPROGRAMS CALLED:

name source load remarks
--- ------- ------- ---------------- côtter
WRILN &WRILN %WRILN given an integer buffer of at least 512 bytes,
WRILN writes a horizontal line of pixels to a designated COMTAL image.

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:**

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

**INITIALIZATIONS:**

```
DO 1000 I = 0,255
   BIGBUF(I+1) = I * 257 ! I*256 numbers the high byte
   BIGBUF(I+257) = BIGBUF(I+1) ! facilitates the wraparound effect
1000 CONTINUE
```

**PROCESSING**

```
DO 2000 ROW = 0,511
   CALL WRILN( WHICH, ROW, BIGBUF((ROW/2)+1) )
2000 CONTINUE
```

```
RETURN
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
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
camera previous to the call. If 5 is not set to video, the COMTAL
hangs up.
0026 C
0027 C
0028 C
0029 C
0030 C***SUBPROGRAMS CALLED:
0031 C
0032 C name source load remarks
0033 C RANGE &RANGE XRANGE logical function that determines if its 1st
0034 C argument lies within 2nd & 3rd argument, inclusive.
0035 C TV2CM &TV2CM xTV2CM digitizes the camera associated with COMTAL image
0036 C 5 into the 1st argument, an image memory.
0037 C ADDI2 &ADDI2 xADDI2 adds two images, pixel by pixel, and divides the
0038 C sums by last argument, to produce a new image.
0039 C
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
C**REVISION HISTORY:
0050 C

C**LOCAL VARIABLES:
0051 C
0052 C
0053 C LOGICAL RANGE ! function that determines if its 1st argument
0054 C is within its 2nd & 3rd, inclusive.
0055 C INTEGER TVIMAG ! COMTAL image number associated with the video
0056 C camera.
0057 C

C**INITIALIZATIONS:
0058 C
0059 C
0060 C DATA TVIMAG/5/ ! arbitrary convention used at our lab.
0061 C DATA TERM/1/
0062 C

C**PROCESSING:
0063 C
0064 C
0065 C CALL DSPVD
0066 C CALL TV2CM(1)
0067 C CALL TV2CM(2)
0068 C CALL ADDI2(3,1,2,2) ! put average of images 1&2 into image 1
0069 C
0070 C CALL TV2CM(1)
0071 C CALL TV2CM(2)
0072 C CALL ADDI2(4,1,2,2) ! put average of images 2&3 into image 2
0073 C
0074 C CALL ADDI2(1,3,4,2) ! put average of images 1&2 into image 1
0075 C
0076 C CALL DSPBV(1)
0077 C
0078 C RETURN
0079 C END
SUBROUTINE TV2CM(CMIMAG)

INTEGER CMIMAG ! "Comtal IMAGE" number where the digitized image
   is to be stored.

! Introduction:
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.

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

! Limitations:
The video camera must be associated with TVIMAGE before TV2CM is
called. TVIMAGE is 5 as an arbitrary convention in this lab.

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

! Subprograms Called:

name source load remarks
--------- ------- ------- -----------------------------------------------
CMIND &CMIND XCMIND sends character strings to COMTAL which
interprets them as COMTAL keyboard commands.
DSPBU &DSPBU XDSPBU displays the indicated monochrome COMTAL image.
RANGE &RANGE XRANGE logical function that determines if 1st
argument is within the 2nd & 3rd, inclusive.
DIGIT &DIGIT XDIGIT character&& function that returns '0'..'9'
according to integer input 0..9.

! 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

- INTEGER CMIMAG ! "Comtal IMAGE" number where the digitized image
  is to be stored.

- 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.

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

- The video camera must be associated with TVIMAGE before TV2CM is
called. TVIMAGE is 5 as an arbitrary convention in this lab.

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

- CMIND &CMIND XCMIND sends character strings to COMTAL which
  interprets them as COMTAL keyboard commands.

- DSPBU &DSPBU XDSPBU displays the indicated monochrome COMTAL image.

- RANGE &RANGE XRANGE logical function that determines if 1st
  argument is within the 2nd & 3rd, inclusive.

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

- NETTIE D. FAULCON. July, 1983. This subprogram was written by
  KEITH MILLER, July, 1984, with the support of a NASA-ASEE
C summer fellowship.

C

***REVISION HISTORY:

C

C

C

C

LOCAL VARIABLES:

INTEGER IMLO, IMHI  ! limits on COMTAL image memory numbers.
INTEGER TVIMAGE  ! the COMTAL image associated with the video camera.
LOGICAL RANGE  ! function that determines if the 1st argument is within the 2nd and 3rd arguments, inclusive.
CHARACTER*1 DIGIT  ! function that returns '0'..'9' according to integer input 0..9.
INTEGER TERM  ! logical unit for terminal output.
INTEGER IBUF(128)! buffer for CMMND COMTAL command strings.
CHARACTER*255 CBUF  ! overlays IBUF.
EQUIVALENCE  (IBUF,CBUF)
INTEGER TVIMAG  ! COMTAL image associated with video image.
               ! arbitrarily set to 5 in this lab.
DATA IMLO/1/, IMHI/4/  ! arbitrary convention for our lab.
DATA TVIMAGE/5/  !
DATA TERM/1/  !

***PROCESSING:

IF (.NOT. (RANGE(CMIMAG,IMLO,IMHI))) GOTO 8081 ! error return

let #C and #V be the digits associated with CMIMAG and TVIMAG;
then the following CMMND string is expanded by the COMTAL into:
Display Image #V; Image #C = Displayed image Red

CBUF = 'D I'//DIGIT(TVIMAGE)//' SI'//DIGIT(CMIMAG)//' D R'
CALL CMMND(IBUF,15)
CALL DSPBLJ(CMIMAG)
RETURN

***ERROR RETURNS:

WRITE(TERM,0003) CMIMAG, IMLO, IMHI
0003 FORMAT(' THE COMTAL IMAGE NUMBER,.I4,. IS OUT OF RANGE;.214)
891  GOTO 8900
892  C
893  C
894  8900 WRITE(TERM,8901)
895  8901 FORMAT(' TV2CM FAILS. NO DIGITIZING TAKES PLACE. ')
896  C
897  RETURN
898  END
&WAIT T=2004 IS ON CR00021 USING 03002 BLKS R=0003

0061 C

0062 SUBROUTINE WAIT

0063 C

0064 C

0065 C**PARAMETERS: NONE

0066 C

0067 C

0068 C**INTRODUCTION:

0069 C

0070 C WAIT pauses until <CR> is pressed on the HP keyboard.

0071 C

0072 C**LANGUAGE:

0073 C

0074 C FORTRAN 77, the HP-1000 version for RTE-6/VM.

0075 C

0076 C**SUBPROGRAMS CALLED: NONE

0077 C

0078 C

0079 C**WRITTEN BY:

0080 C

0081 C KEITH MILLER, NASA-Langley ASEE fellow, 1984

0082 C

0083 C**REVISION HISTORY: NONE

0084 C

0085 C

0086 C**LOCAL VARIABLES:

0087 C

0088 INTEGER TERM ! logical unit of the terminal

0089 INTEGER IDUMMY ! facilitates the read that forces a pause

0090 C

0091 C**INITIALIZATIONS:

0092 C

0093 DATA TERM/1/

0094 C

0095 C**PROCESSING:

0096 C

0097 WRITE(TERM, 1001)

0098 1001 FORMAT( '<PUSH<CR> TO CONTINUE.>' )

0099 READ(TERM, 2001) IDUMMY

0100 2001 FORMAT(I2)

0101 RETURN

0102 END

0103
&WIPGR T=00004 IS ON CR00021 USING 00012 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C+++PARAMETER DECLARATIONS:
0006 C
0007 C     INTEGER  GRNUM  !  a number, 1-4, designating a COMTAL graphics plane.
0008 C
0009 C+++INTRODUCTION:
0010 C
0011 C     "WIPe GRaphics" is a subroutine that allows the caller
0012 C     to send an "un-display" command to the COMTAL from an HP program.
0013 C     The call can turn off a graphics plane, number 1, 2, 3,
0014 C     or 4.
0015 C
0016 C+++LIMITATIONS:
0017 C
0018 C     It is OK to call WIPGR repeatedly without an intervening DSPGR.
0019 C     The extra calls have no effect, but they don't hang up the COMTAL.
0020 C
0021 C+++SUBPROGRAMS CALLED:
0022 C
0023 C     name  source  load  remarks
0024 C     ------  ------  -------  -----------------------------
0025 C     RANGE &RANGE &RANGE logical function that determines if the
0026 C     first parameter is within the bounds defined
0027 C     by the second and third parameter (inclusive).
0028 C     CMNN2 &CMNN2 &CMNN2 sends a constant string to the COMTAL as if
0029 C     the string were typed on the COMTAL keyboard.
0030 C     DIGIT &DIGIT &DIGIT character**1 function that returns '0'-"9"
0031 C     according to integer input 0-9.
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. FAULCON, 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  !  a function for determining if an integer is
INTEGER TERM ! the logical unit number for the terminal.
INTEGER IDUMMY ! fills the place of an unused EXEC parameter.
INTEGER GRLO,GRHI ! limits on a graphics plane number.
CHARACTER*1 DIGIT ! function that returns '0'-'9' for input integers 0-9.

C***INITIALIZATIONS:

DATA TERM/1/
DATA GRLO/1/,GRHI/4/

C***PROCESSING

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

"SUBtract Graphics @GRNUM", where @GRNUM stands for the digit corresponding to GRNUM value.

CALL CMM2('SUB GR '/DIGIT(GRNUM))
RETURN

C***ERROR RETURN:

8001 WRITE(TERM, 8003) GRNUM, GRLO, GRHI
8003 FORMAT( 'THE GRAPHICS NUMBER','13',' OUT OF RANGE:','214)
8011 WRITE(TERM, 8901)
8901 FORMAT( 'DSGPR fails. No action taken on command.' )
RETURN
END
0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE WRGLN(GRAPH, LINE, ONOFFS)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***PARAMETER DECLARATIONS:
0006 C
0007 INTEGER GRAPH ! COMTAL image graphics 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 C name source load remarks
0030 C ------- ------- ------- -------------------------------
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
0038 C***REVISION HISTORY:
0039 C
0040 C***LOCAL VARIABLES:
0041 C
0042 LOGICAL RANGE ! function that ascertains if its first parameter
0045 C is between (inclusive) its last 2 parameters
0046 INTEGER TERM the logical unit for terminal output
0047 INTEGER GRLO, GRHI the limits on COMTAL monochrome image numbers
0048 INTEGER LNLO, LNHI the limits on COMTAL image line numbers
0049 C
0050 C***INITIALIZATIONS:
0051 C
0052 DATA TERM/1/
0053 DATA GRLO/1/, GRHI/4/
0054 DATA LNLO/0/, LNHI/511/
0055 C
0056 C***PROCESSING
0057 C
0058 IF (.NOT.(RANGE(GRAPH, GRLO, GRHI))) GOTO 8001 ! error return
0059 IF (.NOT.(RANGE(LINE, LNLO, LNHI))) GOTO 8101 ! error return
0060 C
0061 C Programming note:
0062 C The EXEC call is explained in detail in the
0063 C HP Programmer's Reference Manual for RTE-6/VM.p.2-19ff. This
0064 C transfer function for the COMTAL is discussed in the
0065 C COMTAL User's Manual. Section 5.2.2.1. In the EXEC call
0066 C that follows, the HP resident driver called DVR41 is called as
0067 C follows: the first parameter (2) signifies a write; the
0068 C second parameter is in two parts: 36B identifies the resident
0069 C DVR41 driver, and 1000 identifies the line transfer operation
0070 C of that driver; the third parameter (ONOFFS) holds the data to be
0071 C transferred, and the fourth parameter gives ONOFFS' length in words
0072 C (32); and the final parameter is a COMTAL command code for the transfer.
0073 C
0074 CALL EXEC(2,36B+1000,ONOFFS,32,(GRAPH-71)*2048 + LINE + 512)
0075 RETURN
0076 C
0077 C***ERROR RETURNS
0078 C
0079 8001 WRITE(TERM, 8003) GRAPH, GRLO, GRHI
0080 8003 FORMAT('GRAPHICS 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('WRGLN FAILS. NO TRANSFER.')
0089 RETURN
0090 END
SUBROUTINE WRGPT(GRAPH, XCOORD, YCOORD, VALUE)

PARAMETER DECLARATIONS:

INTEGER GRAPH ! COMTAL graph number to be written to.
INTEGER XCOORD, YCOORD ! point where new value is to be written to.
INTEGER VALUE ! 0-1 graphics value to be written to graph pt.

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 a 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:

name source load remarks
--------------- ------- ------- -------------
RANGE &RANGE XRANGE logical function that determines if its first parameter is within the last two parameters.
IGSET ----- ----- HP FORTRAN?7 intrinsic function that takes an integer argument and returns that integer with one bit set, according to the second argument.
IBCLR ----- ----- HP FORTRAN?7 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**: Logical unit for terminal output.

- **INTEGER GRLO, GRHI**: Limits on COMTAL monochrome graph numbers.

- **INTEGER LNLO, LNHI**: Limits on COMTAL graph line numbers.

- **INTEGER BTLO, BTHI**: Limits on COMTAL graphics values.

- **INTEGER BITS(32)**: Buffer to read & write a COMTAL graphics line.

- **INTEGER WORD**: Which word of BITS holds the bit selected by XCOOR.

- **INTEGER BIT**: Which bit in BITS(WORD) holds the bit selected by XCOOR.

**INITIALIZATIONS:**

- **DATA TERM/I/**: Terminal.

- **DATA GRLO/I/, GRHI/4/**: Limits on graph numbers.

- **DATA LNLO/O/, LNHI/511/**: Limits on graph line numbers.

- **DATA BTLO/O/, BTHI/1/**: Limits on graphics values.

**PROCESSING**

- **IF (.NOT.(RANGE(GRAPH, GRLO, GRHI)))**: GOTO 8001 (error return).

- **IF (.NOT.(RANGE(XCOOR, LNLO, LNHI)))**: GOTO 8101 (error return).

- **IF (.NOT.(RANGE(YCOOR, LNLO, LNHI)))**: GOTO 8201 (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 + YCOOR)**

Change the single bit that has been selected:

- **WORD = (XCOOR/16) + 1**

- **BIT = (16 * WORD) - XCOOR - 1**
0091 C
0092 IF (VALUE .EQ. 0) GOTO 1000
0093 C ELSE...VALUE .EQ. 1
0094 BITS(WORD) = IBSET(BITS(WORD), BIT)
0095 GOTO 2000
0096 C THEN...VALUE .EQ. 0
0097 1000 BITS(WORD) = IBCLR(BITS(WORD), BIT)
0098 GOTO 2000
0099 C
0100 C Write the graph line with one changed graphics to COMTAL
0101 C
0102 2000 CALL EXEC(2,368+1008.BITS,32,(GRAPH-1)*2048 + 512 + YCOORD)
0103 RETURN
0104 C
0105 C**ERROR RETURNS
0106 C
0107 8001 WRITE(TERM, 8003) GRAPH, GRLO, GRHI
0108 8003 FORMAT( 'GRAPH NUMBER, ', I3, ' OUT OF RANGE: ', 212, ' ')
0109 GOTO 8900
0110 C
0111 8101 WRITE(TERM, 8103) XCOORD, LNLO, LNHI
0112 8103 FORMAT( 'X COORDINATE, ', I4, ' OUT OF RANGE: ', 214, ' ')
0113 GOTO 8900
0114 C
0115 8201 WRITE(TERM, 8203) YCOORD, LNLO, LNHI
0116 8203 FORMAT( 'Y COORDINATE, ', I4, ' OUT OF RANGE: ', 214, ' ')
0117 GOTO 8900
0118 C
0119 8301 WRITE(TERM, 8303) VALUE, BTLO, BTHI
0120 8303 FORMAT( 'BIT VALUE, ', I4, ' OUT OF RANGE: ', 214, ' ')
0121 GOTO 8900
0122 C
0123 8900 WRITE(TERM, 8901)
0124 8901 FORMAT( 'WRGPT FAILS. NO TRANSFER.' )
0125 RETURN
0126 END

143
SUBROUTINE WRIL2(IMAGE, LINE, INTS)

PARAMETER DECLARATIONS:

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:

name source load remarks
------ ------ ------- ------------------------------
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
0045 C NETTIE D. FAULCON. July, 1983. This modification is by
0046 C KEITH MILLER. June, 1984.
0047 C
0048 C***REVISION HISTORY:
0049 C
0050 C
0051 C***LOCAL VARIABLES:
0052 C
0053 INTEGER PIXELS(256)! holds lower order bytes of INTS values
0054 CHARACTER*1 CPIX (512)! overlays PIXELS
0055 EQUIVALENCE (PIXELS, CPIX)
0056 C
0057 LOGICAL RANGE ! function that ascertains if its first parameter
0058 C is between (inclusive) its last 2 parameters
0059 INTEGER TERM ! the logical unit for terminal output
0060 INTEGER IMLO, IMHI ! the limits on COMTAL monochrome image numbers
0061 INTEGER LNLO, LNHI ! the limits on COMTAL image line numbers
0062 INTEGER INDEX ! indexes into arrays in a loop
0063 C
0064 C***INITIALIZATIONS:
0065 C
0066 DATA TERM/1/
0067 DATA IMLO/1/, IMHI/4/
0068 DATA LNLO/0/, LNHI/511/
0069 C
0070 C***PROCESSING
0071 C
0072 IF (.NOT.(RANGE(IMAGE, IMLO, IMHI))) GOTO 8001 ! error return
0073 IF (.NOT.(RANGE(LINE, LNLO, LNHI))) GOTO 8101 ! error return
0074 C
0075 C strip off the upper bytes of the pixel values in INTS
0076 C
0077 DO 1000 INDEX=1,512
0078 CPIX(INDEX) = CHAR(INTS(INDEX)) ! CPIX overlays PIXELS
0079 1000 CONTINUE
0080 C
0081 C Programming note:
0082 C The EXEC call is explained in detail in the
0083 C HP Programmer’s Reference Manual for RTE-6/M, p.2-19ff. This
0084 C transfer function for the COMTAL is discussed in the
0085 C COMTAL User’s Manual, Section 5.2.2.1. In the EXEC call
0086 C that follows, the HP resident driver called DVR41 is called as
0087 C follows: the first parameter (2) signifies a write; the
0088 C second parameter is in two parts: 368 identifies the resident
0089 C DVR41 driver, and 1008 identifies the line transfer operation
0090 C 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, 368+100B, PIXELS, 256, (IMAGE-1)*2048 + LINE)
RETURN

C***ERROR RETURNS

WRITE(TERM, 0003) IMAGE, IMLO, IMHI
FORMAT(' IMAGE NUMBER, ', I3, ' OUT OF RANGE: ', 2I2, ' )')
GOTO 8900

WRITE(TERM, 8103) LINE, LNLO, LNHI
FORMAT(' LINE NUMBER, ', I4, ' OUT OF RANGE: ', 2I3, ' )')
GOTO 8900

WRITE(TERM, 8901)
FORMAT(' WRIL2 FAILS, NO TRANSFER. ')
RETURN
END
SUBROUTINE WRILN(IMAGE, LINE, PIXELS)

INTEGER IMAGE ! COMTAL image number to be written to
INTEGER LINE ! which horizontal line to be written to:
INTEGER PIXELS(256) ! 512 bytes (pixels) to be transferred

BBB4 C
BBB5 C_o_KPARAMETER DECLARATIONS:

BBB6 C
BBB7 C
BBB8 C
BBB9 C
BBB10 C

BBB11 C

BBB12 C***INTRODUCTION:

BBB13 C
BBB14 C This subroutine WRites an Image LIne to the COMTAL. The line of
BBB15 C pixels is made up of 8 bit (0-255) grey scale intensities. The
BBB16 C PIXELS array is assumed to be completely full.

BBB17 C

BBB18 C***LANGUAGE:

BBB19 C

BBB20 C FORTRAN 77, the HP-1000 version for RTE-6/VM.

BBB21 C

BBB22 C***LIMITATIONS:

BBB23 C
BBB24 C The PIXELS array must be completely "full". The IMAGE designated
BBB25 C must be a monochrome image. The LINE parameter must be between
BBB26 C 0 and 511. If IMAGE or LINE is out of range, an error message is printed
BBB27 C and no transfer takes place.

BBB28 C

BBB29 C***SUBPROGRAMS CALLED:

BBB30 C

BBB31 C name source load remarks

BBB32 C

BBB33 C RANGE &RANGE &RANGE logical function that determines if its
BBB34 C first parameter is within the last two parameters.

BBB35 C

BBB36 C***WRITTEN BY:

BBB37 C
BBB38 C The code on which this subprogram is based was written by
BBB39 C NETTIE D. FAULCON, July, 1983. This modification is by
BBB40 C KEITH MILLER, June, 1984.

BBB41 C

BBB42 C***REVISION HISTORY:

BBB43 C

BBB44 C
C***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 PIXELE'S 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
8003 FORMAT(' IMAGE NUMBER: ', ' IS OUT OF RANGE: ', 212, '.')

GOTO 8900
8005 C

8101 WRITE(TERM, 8103) LINE, LNLO, LNHI
8103 FORMAT(' LINE NUMBER: ', ' IS OUT OF RANGE: ', 213, '.')

GOTO 8900
8009 C

8900 WRITE(TERM, 8901)
0091  S901 FORMAT(' WRILN FAILS. NO TRANSFER.')
0092       RETURN
0093       END
SUBROUTINE WRIPT(IMAGE, XCOORD, YCOORD, VALUE)

INTEGER IMAGE ! COMTAL image number to be written to.
INTEGER XCOORD, YCOORD ! point where new value is to be written to.
INTEGER 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) gray 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(S(2)) ! overlay for IHOLD

EQUIVALENCE (IHOLD, CHAR(S)

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 8081 ! error return

IF (.NOT. (RANGE(XCOORD, LNLO, LNHI))) GOTO 8181 ! error return

IF (.NOT. (RANGE(YCOORD, LNLO, LNHI))) GOTO 8281 ! error return

IF (.NOT. (RANGE(VALUE, PXLO, PXHI))) GOTO 8381 ! 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, 368+1308, PIXELS, 255, (IMAGE-1)*2048 + YCOORD)

Change the single byte that needs changing:

I HOLD = VALUE ! transfers the pixel VALUE to CHAR(S(2).

BYTES(XCOORD+1) = CHAR(S(2)) ! transfers VALUE to PIXELS in proper position ‘+1’ 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, 36B+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 8002 GO TO 8900
0101 C
0102 8101 WRITE(TERM, 8103)XCOORD, LNLO, LNHI
0103 8103 FORMAT(‘X COORDINATE,’, I4, ‘OUT OF RANGE:’, 214,’.’)
0104 8105 GO TO 8900
0105 C
0106 8201 WRITE(TERM, 8203)YCOORD, LNLO, LNHI
0107 8203 FORMAT(‘Y COORDINATE,’, I4, ‘OUT OF RANGE:’, 214,’.’)
0108 8206 GO TO 8900
0109 C
0110 8301 WRITE(TERM, 8303) VALUE, PXLO, PXHI
0111 8303 FORMAT(‘PIXEL VALUE,’, I4, ‘OUT OF RANGE:’, 214,’.’)
0112 8305 GO TO 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  ! INPARA is to be written
INTEGER XDIM, YDIM  ! the image coordinates of the upper left
! corner of the rectangle of pixels that is
INTEGER XDIM, YDIM  ! dimensions of the input array, INFARA
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 replaced by inline code, which is not long.

RDIL2 &RDIL2 &RDIL2 transfers a line of bytes from a COMTAL image
into an HP array of integers.

WRIL2 &WRIL2 &WRIL2 transfers a buffer of integers to a COMTAL image
as a line of bytes (1 byte/integer).
C. RANGE &RANGE $\text{X_RANGE}$ logical function that determines of its first
parameter is within the last two parameters

C CHAR ------ ------ HP FORTRAN77 intrinsic function; converts a
2 byte integer into a one byte char (removes
high order byte).

C***WRITTEN BY:

C The code on which this subprogram is based was written by
C NETTIE D. FAULKNER, July, 1983. This code was written by

C***REVISION HISTORY:

C LOCAL VARIABLES:

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
C LOGICAL 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 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
C IF (.NOT. (RANGE(XDIM, 1, DIMLIM)))GOTO 8101 | error return
C IF (.NOT. (RANGE(YDIM, 1, DIMLIM)))GOTO 8201 | error return
C IF (.NOT. (RANGE(UPLFX, LNL0, LNHI )))GOTO 8301 | error return
C IF (.NOT. (RANGE(UPLFY, LNL0, LNHI )))GOTO 8401 | error return
C IF (.NOT. (RANGE(UPLFY+YDIM-1, LNL0, LNHI )))GOTO 8501 | error return
C IF (.NOT. (RANGE(UPLFY+YDIM-1, LNL0, LNHI )))GOTO 8601 | error return
0091 C we get to this point if the transfer is to take place
0092 C
0093 IMROW = UPLFY
0094 DO 2000 ARAROW = 1, YDIM
0095 CALL RDIL2(IBUF, IMAGE, IMROW)
0096 C ! the next line initializes the column pointer:
0097 IMCOL = UPLFX + 1 ! the "+1" is necessary because COMTAL image
0098 C ! coordinates range from 0 to 511 and the
0099 C ! FORTRAN array indices range from 1 to 512.
0100 DO 1000 ARACOL = 1, XDIM
0101 IBUF(IMCOL) = INPARA(ARAROW, ARACOL)
0102 IMCOL = IMCOL + 1 ! increment for next 1000 loop pass
0103 1000 CONTINUE
0104 CALL WRL2(IMAGE, IMROW, IBUF)
0105 IMROW = IMROW + 1 ! increment for next 2000 loop pass
0106 2000 CONTINUE
0107 RETURN ! successful termination
0108 C
0109 C***ERROR RETURNS:
0110 C
0111 8061 WRITE(TERM, 8003) IMAGE, IMLO, IMHI
0112 8003 FORMAT(' IMAGE NUMBER:, I3, ' OUT OF RANGE:', 214, '.')
0113 GOTO 8900
0114 C
0115 8181 WRITE(TERM, 8103) XDIM, 1, DIMLIM
0116 8103 FORMAT(' X DIMENSION:, I4, ' OUT OF RANGE:', 215, '.')
0117 GOTO 8900
0118 C
0119 8281 WRITE(TERM, 8203) YDIM, 1, DIMLIM
0120 8203 FORMAT(' Y DIMENSION:, I4, ' OUT OF RANGE:', 215, '.')
0121 GOTO 8900
0122 C
0123 8381 WRITE(TERM, 8303) UPLFX, LNLO, LNHI
0124 8303 FORMAT(' X COORDINATE FOR CORNER:, I4, ' OUT OF RANGE:', 215, '.')
0125 GOTO 8900
0126 C
0127 8481 WRITE(TERM, 8403) UPLFY, LNLO, LNHI
0128 8403 FORMAT(' Y COORDINATE FOR CORNER:, I4, ' OUT OF RANGE:', 215, '.')
0129 GOTO 8900
0130 C
0131 8581 WRITE(TERM, 8503) XDIM, LNLO, LNHI
0132 8503 FORMAT(' X COORDINATE FOR THE CORNER AND THE X DIMENSION ',
0133 1 ' OF THE ARRAY', ' OVERFLOW IMAGE BOUNDARIES', ')
0134 2 ' X COORDINATE =', I4, ' X DIMENSION =', I4, ')
0135 3 IMAGES COORDINATE LIMITS ARE ', 215, ')
0136 GOTO 8900
C
0138 8661 WRITE(TERM, 8603) UPLFY, YDIM, LNLO, LNHI
0139 8663 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
&URLUT T=00004 IS ON CR00021 USING 00004 BLKS R=0000

0001  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002  SUBROUTINE URLUT(LUTNUM, TABLE)
0003  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004  C
0005  C**PARAMETER DECLARATIONS:
0006  C
0007  INTEGER LUTNUM  ! the number of the COMTAL look-up table (called
0008  C       ! "function memory" in the COMTAL literature).
0009  INTEGER TABLE(256) ! the values to be placed in the look-up table:
0010  C       ! notice that 0 maps to TABLE(1), 1 maps to
0011  C       ! 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  name     source     load   remarks
0034  ---------  ---------  --------
0035  RANGE     $RANGE   $RANGE logical function which determines if its 1st
0036  0038  parameter is within its 2nd and 3rd inclusive.
0037  C
0039  C**WRITTEN BY:
0040  C
0041  C       The code on which this subprogram is based was written by
0042  C       NETTIE D. FAULKON. July, 1983. This subprogram was written by
0043  C       KEITH MILLER. July, 1984, with the support of a NASA-ASEE
0044  C
0045  C
C***REVISION HISTORY:

C***LOCAL VARIABLES:

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:

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

C***PROCESSING

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 (368) and gives the code (206B) that identifies this operation, a transfer to a COMTAL function memory (Look-Up Table).
The third parameter gives the Look-Up Table values (TABLE).
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, 368+2068, TABLE, 256, ((LUTNUM-1)*256) )

RETURN

C***ERROR RETURN

8801 WRITE(TERM, 8803) LUTNUM, LUTLO, LUTHI
8823 FORMAT( ' THE FUNCTION MEMORY NUMBER.',I4, ', IS OUT OF RANGE: ', 1, 214, ', ') 
8900 WRITE(TERM, 8901)
8901 FORMAT( ' LUT FAILS. NO TRANSFER TO COMTAL.' )
END
SUBROUTINE WRPSU(TABLE)

PARAMETER DECLARATIONS:
INTEGER TABLE (768) ! the values of the COMTAL look-up table are read from this array. The RED table is in TABLE(1:256); the GREEN, in TABLE(257:512); and the BLUE, in TABLE(513:768).

INTRODUCTION:
The subroutine WRITE the Pseudocolor table writes the 3 COMTAL mappings from 0-255 which comprise the pseudocolor table. Note that the values are placed into TABLE in the order RED, GREEN, and BLUE.

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

LIMITATIONS:
Although 8 bits are sufficient for the look-up table values, full integers are used in TABLE. This format is dictated by the COMTAL conventions as given in section 5.2.3.1.

SUBPROGRAMS CALLED:
NONE.

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:
NONE.

LOCAL VARIABLES:
NONE.
C***INITIALIZATIONS:
C
C
C
C
C
C***PROCESSING
C
C
Programming notes:
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
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 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 B59 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), 256, 0*256 ) ! green
CALL EXEC( 2, 36B+300B, TABLE(513), 256, 2*256 ) ! blue
RETURN
END
**URTAR**

T=63004 IS ON CR00021 USING 000062 BLKS R=6300

```fortran
0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE URTAR(XCOOR, YCOOR)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***PARAMETER DECLARATIONS:
0006 C
0007 INTEGER XCOOR  ! X coordinate of the desired target location.
0008 INTEGER YCOOR  ! Y coordinate of the desired target location.
0009 C
0010 C***INTRODUCTION:
0011 C
0012 C This subroutine WRites a TARget location to the COMTAL, thereby
0013 C "positioning" the COMTAL cursor.
0014 C
0015 C***LANGUAGE:
0016 C
0017 FORTRAN 77, the HP-1000 version for RTE-6/VM.
0018 C
0019 C***LIMITATIONS:
0020 C
0021 C Both the XCOOR and YCOOR must be within the range 0 to 511. If not,
0022 C an error message is printed and no transfer takes place to the COMTAL.
0023 C
0024 C***SUBPROGRAMS CALLED:
0025 C
0026 name source load remarks
0027 ------ ------ ----- -----------------------------------------------
0028 C RANGE &RANGE %RANGE logical function that determines if its
0029 C ! first parameter is within the last two parameters.
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, 1983. This modification is by
0035 C KEITH MILLER. June, 1984.
0036 C
0037 C***REVISION HISTORY:
0038 C
0039 C
0040 C***LOCAL VARIABLES:
0041 C
0042 LOGICAL RANGE ! function that ascertains if its first parameter
0043 C ! is between (inclusive) its last 2 parameters
0044 C INTEGER TERM ! the logical unit for terminal output
```

161
**INTRODUCTION**:

- **INTEGER LHLO, LNHI**: the limits on COMTAL image line numbers.
- **INTEGER IBUF(2)**: buffer for passing coordinates to COMTAL.
- **INTEGER IDUMMY**: an ignored EXEC call parameter.

**INITIALIZATIONS:**

- **DATA TERM/*/**: defining the terminals.
- **DATA LNLO/*/, LNHI/*/***: defining the limits.

**PROCESSING**:

- **IF (.NOT. (RANGE(XCOOR, LHLO, LHNO))) GOTO 8081**: error return.
- **IF (.NOT. (RANGE(YCOOR, LHLO, LHNO))) GOTO 8181**: 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.

**ERROR RETURNS**

- **WRITE(TERM, 'XCOOR, LHLO, LHNO')**: error return.
- **WRITE(TERM, 'YCOOR, LHLO, LHNO')**: error return.

**RETURN**

- **RETURN**
- **END**
PROGRAM TADD2

TEST THE ADDPROGRAM ADDI2 (ADD IMAGES)

KEITH MILLER 7/17/84

LOAD MODULES: XTADD2, XADDI2, XCMDND, XRANGE, XDIGIT

CALL ADDI2(0,1,2) ! SHOULD GIVE OUT OF RANGE ERROR.
CALL ADDI2(1,0,2) ! *
CALL ADDI2(1,2,0) ! *

CALL ADDI2(3,1,2) ! SHOULD WORK.

STOP

END
PROGRAM TADDI

TEST THE ADD PROGRAM ADDIM (ADD IMAGES)

KEITH MILLER 7/17/84

LOAD MODULES: XTADDI, XADDIM, _CMND, X RANGE, X DIGIT

CALL ADDIM(0,1,2,1) ! SHOULD GIVE OUT OF RANGE ERROR
CALL ADDIM(1,0,2,1) ! 
CALL ADDIM(1,2,0,1) ! 
CALL ADDIM(3,1,2,1)

STOP

END
&TCLR  T=00004  IS ON CR00021 USING 00001 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
&TCLRG T-00004 IS ON CR00021 USING 00002 BLKS R-0000

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, %CMNND, %DIGIT
0008  C
0009  INTEGER IBUF(128)
0010  CHARACTER*20 CBUF   ! OVERLAYS THE FIRST 10 ELEMENTS OF IBUF
0011  EQUIVALENCE (IBUF, CBUF)
0012  C
0013  CBUF = 'CLEAR GR 2'
0014  C
0015  CALL CLRGR(0)
0016  CALL CLRGR(1)
0017  CALL CMNND(IBM, 10)
0018  CALL CLRGR(5)
0019  C
0020  STOP
0021  END
0022
0023
0024
0025
0026
&TCLRI T=00004 IS ON CR00021 USING 00002 BLKS R=0000

0001 PROGRAM TCLRI
0002 C
0003 C TEST CLEAR IMAGE
0004 C
0005 C KEITH MILLER, 7/2/84
0006 C
0007 C LOAD MODULES: %TCLRI, %CLRIM, %RANGE, %CMDND, %DIGIT
0008 C
0009 INTEGER IBUF(128)
0010 CHARACTER*20 CBUF ! OVERLAYS THE FIRST 10 ELEMENTS OF IBUF
0011 EQUIVALENCE (IBUF, CBUF)
0012 C
0013 CBUF = 'CLEAR IMAGE 2'
0014 C
0015 CALL CLRIM(0)
0016 CALL CLRIM(1)
0017 CALL CMDND(IBUF, 13)
0018 CALL CLRIM(5)
0019 C
0020 STOP
0021 END
0022
0023
0024
0025
0026
PROGRAM TCMM2
C
TEST CMND AGAIN
C
KEITH MILLER, 7/12/84
C
LOAD MODULES _TCMM2, _CMND, _WAIT
C
INTEGER INNUM, I, IBUF(128)
CHARACTER(255) CBUF
EQUIVALENCE (IBUF,CBUF)
C
DO 1000 I = 0,256
WRITE(1,500) I
500 FORMAT('TESTING CHARACTER #', I4)
CALL WAIT
CBUF = 'G 2 PRO 1'
CALL CMND(IBUF,9)
CBUF = CHAR(I)
CALL CMND(IBUF,1)
1000 CONTINUE
STOP
END
&TCMMN T=00004 IS ON CR00021 USING 00002 BLKS R=0000

0001    PROGRAM  TCMMN
0002    C
0003    C    Keith Miller, 6/16/84
0004    C
0005    C    Testing SUBROUTINE CMDND
0006    C
0007    C    LOAD MODULES REQUIRED: %TCMMN, %CMDND
0008    C
0009    C    INTEGER  IBUF(128)
0010    C
0011    C    WRITE(1,1)
0012    1    FORMAT(' TO EXIT THIS TEST, ENTER AN EMPTY STRING')
0013    C
0014    5    WRITE(1,10)
0015    10   FORMAT('ENTER ASCII STRING')
0016    C
0017    20   FORMAT(128A2)
0018    C    N = ITLOG()  ! ITLOG gives the number of characters typed in
0019    C    IF (N .EQ. 0) GOTO 999
0020    C
0021    CALL CMDND(IBUF,N)
0022    C
0023    999   END
PROGRAM TCNT

C TEST THE SUBROUTINE COUNT

KEITH MILLER 7/11/84

C LOAD MODULES: TCNT, COUNT, RANGE, NRDIL2

INTEGER 4 BUF(256)

CALL COUNT(14BUF, 1)

DO 1000 INDEX = 0, 31

WRITE(1,999) (14BUF((INDEX*8)+J), J=1,8)

999 FORMAT(8I9)

1000 CONTINUE

C

STOP

END
PROGRAM TCONS

TEST CONCATENATION OF CHARACTER STRING, FORTRAN 77

INTEGER IHOLD
CHARACTER*1 CHOLD(2)
EQUIVALENCE (IHOLD, CHOLD)
CHARACTER*5 STRING

IHOLD = 6 + 608
STRING = '***' // CHOLD(2) // '***'
WRITE(1, 1000) CHOLD(2), STRING
1000 FORMAT(' LETTER=', A1, '. STRING=', A5, '.')
STOP
END
&TCOPY T=00004 IS ON CR80021 USING 00001 DLKS R=0000

0001 PROGRAM TCOPY
0002 C TEST ICOPY
0003 C LOAD MODULES REQUIRED: TCOPY,ICOPY,CMMND,RANGE,\n0004 C \n0005 C DIGIT
0006 C \n0007 C \n0008 CALL ICOPY(1,2) \n0009 CALL ICOPY(5,1) ! SHOULD GIVE OUT OF RANGE ERROR \n0010 CALL ICOPY(2,-3) ! SHOULD GIVE OUT OF RANGE ERROR \n0011 STOP \n0012 END
PROGRAM TDIGI

C TEST DIGIT
C KEITH MILLER, 7/2/84

C LOAD MODULES: TDIGI, DIGIT, RANGE

INTEGER INT
CHARACTER*1 HCHAR

DO 1000 INT = -1, 12
   CALL DIGIT(HCHAR, INT)
   WRITE(1,501) INT, HCHAR
   501 FORMAT(' INTEGER INPUT AND CHARACTER OUTPUT: ',113,1A1,' ')!
1000 CONTINUE

STOP
END
PROGRAM TDSP

TEST DSPBW AND DSPCL

LOAD MODULES REQUIRED: %TDSP, %DSPBW, %DSPCL, %CMMND,
%WAIT, %RANGE, %DIGIT

CALL DSPBW(1)

CALL WAIT

CALL DSPBW(2)

CALL WAIT

CALL DSPBW(3)

CALL WAIT

CALL DSPBW(0)

CALL WAIT

CALL DSPCL(1,2,3,8)

STOP

END
PROGRAM THIST

C TEST THE SUBROUTINE HISTO

C KEITH MILLER, 7/31/84

C LOAD MODULES: %THIST, %HISTO, %DIGIT, %CMN2, %RANGE

INTEGER TABLE(256)

INTEGER I  ! implicit do loop index

CALL HISTO(1)

STOP

END
&THORM T=03004 IS ON CR0021 USING 00002 BLKS R=00003

0001        PROGRAM THORM
0002        C
0003        C    TEST THE PROCEDURE NORML.
0004        C
0005        C    KEITH MILLER JULY 6, 1983
0006        C
0007        C    LOAD MODULES: %THORM, %NORML, %HILO, %RDIL2, %WRIL2, %RANGE
0008        C
0009        CALL NORML(2)
0010        STOP
0011        END
PROGRAM TNOTE

TEST THE SUBROUTINES NOTES AND NOTE2

KEITH MILLER, 7/26/84

LOAD MODULES: xNOTE, xNOTES, xNOTE2, xDSPGR, xWRTAR,
xCMMN2, xCMMND, xDIGIT, xRANGE, xDELAY

CHARACTER*255 CBUF
INTEGER Ibuf(128)
EQUIVALENCE (CBUF, Ibuf)

CALL NOTE2(1, 100, 100, 'B', 3, '1.BLUE')
CALL NOTE2(1, 100, 200, 'R', 3, '2.RED')
CALL NOTE2(1, 100, 300, 'G', 3, '3.GREEN')
CALL NOTE2(1, 100, 400, 'S', 3, '4.SAME')

CBUF(1:11) = 'USING NOTES'

CALL NOTES(1, 10, 10, 'R', 1, CBUF, 11)
CALL NOTES(5, 10, 10, 'R', 1, CBUF, 11)
CALL NOTES(1, 512, 10, 'R', 1, CBUF, 11)
CALL NOTES(1, -1, 10, 'R', 1, CBUF, 11)
CALL NOTES(1, 10, 512, 'R', 1, CBUF, 11)
CALL NOTES(1, 10, -1, 'R', 1, CBUF, 11)
CALL NOTES(1, 10, 10, 'R', 17, CBUF, 11)
CALL NOTES(1, 10, 10, 'R', 17, CBUF, 11)
CALL NOTE2(1, 120, 240, 'B', 3, 'USING NOTE2')
CALL NOTE2(0, 120, 240, 'B', 3, 'USING NOTE2')
CALL NOTE2(5, 120, 240, 'B', 3, 'USING NOTE2')
CALL NOTE2(2, 512, 240, 'B', 3, 'USING NOTE2')
CALL NOTE2(2, -1, 240, 'B', 3, 'USING NOTE2')
CALL NOTE2(2, 120, 512, 'B', 3, 'USING NOTE2')
CALL NOTE2(2, 120, -1, 'B', 3, 'USING NOTE2')
CALL NOTE2(2, 120, 240, 'B', 0, 'USING NOTE2')
CALL NOTE2(2, 120, 240, 'B', 17, 'USING NOTE2')
CALL DSPGR(1)
CALL DSPGR(2)
STOP
END
PROGRAM TPNT

TEST THE SUBROUTINE PAINT

KEITH MILLER, 7/16/84

LOAD MODULES: TPNT, PAINT, RANGE, URIRC, RDTAR, RDIL2, URIL2, CMND, 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, %CMMD, %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
PROGRAM TRDTA

TEST THE SUBROUTINE RDTAB

KEITH MILLER, 7/19/84

LOAD MODULES: %TRDTA, %RDTAB, %RANGE

INTEGER TABLE(16), INDEX, WHICH

DO 2000 WHICH = 1, 4
    CALL RDTAB(TABLE, WHICH, 0)
    WRITE(1,999) WHICH, TABLE(1), TABLE(2), TABLE(3), TABLE(4)
    FORMAT(12, ',', 409)
2000 CONTINUE

STOP

END
PROGRAM TSETV

TEST THE SETV SUBROUTINE

KEITH MILLER, 7/17/84

LOAD MODULES: _TSETV, _SETV, _RANGE, _DIGIT, _CMDND, _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, XSPRED, XRDIL2, XWRIL2, XRANGE

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

WRITE(1, 1001)
C
1001 FORMAT(' GIVE THREE SHADES, USING THE FORMAT 314:')

READ (1, 1003) I1, I2, I3

INDEX = I

DO 2000 INDEX = 1, 170

IBUF(INDEX) = I1

IBUF(INDEX + 170) = I2

IBUF(INDEX + 340) = I3

2000 CONTINUE

IBUF(511) = I3

IBUF(512) = I3

DO 3000 INDEX = 0, 511

CALL WRIL2(1, INDEX, IBUF)

3000 CONTINUE

CALL SPRED(1)

CALL SPRED(1)

STOP

END
PROGRAM TSUBI

C TESTS SUBROUTINES SUBIM AND SUBI2

C KEITH MILLER 7/23/84

C LOAD MODULES: %TSUBI, %SUBIM, %SUBI2, %RANGE, %DIGIT.

C \_WAIT, \_CMND

CALL SUBIM(1,3,2) \_ \_off
CALL SUBI2(1,3,2) \_ \_128 \_ \_off

STOP
END
&TTHRS T=00004 IS ON CRO0021 USING 00003 CLKS R=0000

0001 PROGRAM TTHRS
0002 C
0003 C TESTS THE SUBROUTINE THRESHOLD
0004 C
0005 C KEITH MILLER 7/25/84
0006 C
0007 C LOAD MODULES: TTHRS, THRSH, RANGE, RDIL2, WRIL2
0008 C
0009 C INTEGER THRESH! THRESHOLD FROM USER
0010 C INTEGER IN, OUT! TWO IMAGES
0011 C
0012 WRITE(1,1001)
0013 1001 FORMAT(’ GIVE THE OUTPUT IMAGE NUMBER:’)
0014 READ(1,1003)OUT
0015 1003 FORMAT(I)
0016 WRITE(1,1005)
0017 1005 FORMAT(’ GIVE THE INPUT IMAGE NUMBER:’)
0018 READ(1,1003)IN
0019 WRITE(1,1007)
0020 1007 FORMAT(’ GIVE THE THRESHOLD PIXEL VALUE:’)
0021 READ(1,1009)THRESH
0022 1009 FORMAT(I)
0023 CALL THRSH(OUT, IN, THRESH)
0024 STOP
0025 END
PROGRAM TTSTI

TESTS THE TEST IMAGES

LOAD MODULES: XTTSTI, XSTSTL, XURILN, XRANGE

CALL TSTII(1)

STOP

END
&TTV2C T=00004 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: xTTV2C, xTV2C4, xTV2CM, xCMND, xDSPBW,
0008      C   xRANGE, xDIGIT, xWAIT, xADD12, xDSPVD
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
0015      C   CALL TV2C4
0016      C
0017      C   STOP
0018      C
0019      C   END
&TWAIT T=00004 IS ON CR00021 USING 00002 BLKS R=0000

0001       PROGRAM TWAIT
0002       C
0003       C       TESTING THE WAIT SUBROUTINE
0004       C       KEITH MILLER, JUNE 8, 1984
0005       C
0006       INTEGER IERR, IX, IY
0007       C
0008       WRITE(1,1000)
0009       1000 FORMAT( ‘START WAIT TEST’)
0010       CALL WAIT(IERR)
0011       CALL KMRTA( IX,IY,IERR)
0012       WRITE(1, 2000)
0013       2000 FORMAT( ‘END WAIT TEST’)
0014       C
0015       STOP
0016       END
&TWIPE T=0004 IS ON CR00021 USING 02002 BLKS R=0060

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, %COMM2
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

189
&TXFDS J=00004 IS ON CR00021 USING 00001 BLKS R=00000

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, _CMND, _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 = 'CFMAND'
0017      CALL CLFDS(1 ?,3,8, FLNAME)
0018      STOP
0019      END
0001 PROGRAM TXGLN
0002 C
0003 C TEST URGLN AND RDGLN
0004 C
0005 C LOAD MODULES REQUIRED: TXGLN, RDGLN, URGLN, XRANGE
0006 C
0007 INTEGER LINE ! number of line in image being processed
0008 INTEGER IBUF(32) ! buffer to hold ONOFF values, one line/time
0009 C ! note that 16 ONOFF bits fit in one INTEGER
0010 C
0011 DO 2000 LINE = 1, 512
0012 CALL RDGLN(IBUF, 2, LINE)
0013 C
0014 CALL URGLN(1, LINE, IBUF)
0015 2000 CONTINUE
0016 END
0017
0018
0001    PROGRAM TXGPT
0002    C
0003    C  TEST RDGPT AND WRGPT
0004    C
0005    C  KEITH MILLER, 6/22/84
0006    C
0007    C  LOAD MODULES: TXGPT, RDGPT, WRGPT, RANGE, RDTAR
0008    C
0009    C  INTEGER INDEX ! LOOP INDEX
0010    C
0011    DO 1000 INDEX = 1, 20
0012        CALL WRGPT(I, INDEX, INDEX, I)
0013        CALL WRGPT(I, INDEX, 21-INDEX, 1)
0014    1000 CONTINUE
0015    C
0016    C    CALL WAIT
0017    C
0018    C    DO 2000 INDEX = 1, 20
0019        CALL WRGPT(I, INDEX, INDEX, 0)
0020        CALL WRGPT(I, INDEX, 21-INDEX, 0)
0021    2000 CONTINUE
0022    C
0023    C    DO 3000 INDEX=1,5
0024        CALL WAIT
0025        CALL RDTAR(I, I)
0026        CALL RDGPT(IVALUE, I, IX, IY)
0027        WRITE(1, 2001) IX, IY, IVALUE
0028    2001 FORMAT(' AT POINT ', 214, ' GRAPHICS VALUE=', 12, '.')
0029    3000 CONTINUE
0030    C
0031    C    STOP
0032    C    END
PROGRAM TXILN

TEST WRILN AND RDILN

LOAD MODULES REQUIRED: TXILN, XRDILN, XWRILN, XRANGE

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, LINE)

CALL WRILN(2, LINE, IBUF)

2000 CONTINUE

END
&TXIPT T=00004 IS ON CR00021 USING 0002 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(' 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
0001      PROGRAM TXIRC
0002      C
0003      C       TEST RDIRC (READ IMAGE RECTANGLE)
0004      C       & WRIRC (WRITE IMAGE RECTANGLE)
0005      C
0006      C       KEITH MILLER, JULY 3, 1984
0007      C
0008      C       LOAD MODULES: %TXIRC, %RDIRC, %WRIRC, %RANGE, %RDILN, %WRILN
0009      C
0010      INTEGER BUFFER(10,10)
0011      INTEGER ROW, COL
0012      C
0013      CALL RDIRC(BUFFER, 10, 10, 1, 0, 0)
0014      CALL WRIRC(1, 100, 100, BUFFER, 10, 10)
0015      STOP
0016      END
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 (1999) TABLE2(INDEX)

2000 CONTINUE

STOP

END
PROGRAM TXPSU

TEST PSEUDO COLOR TABLE TRANSFERS: RDPSU AND WRPSU

LOAD MODULES: %TXPSU, %RDPSU, %URPSU, %RANGE, %WAIT

KEITH MILLER, 7/10/84

INTEGER TABLE(768)

INTEGER INDEX

CALL RDPSU(TABLE)

WRITE(1,999)

559 FORMAT(' THE REDS: ')

DO 2000 INDEX=1,1256

WRITE(1,1999) TABLE(INDEX)

1999 FORMAT(I4)

CONTINUE

CALL WAIT

WRITE(1,2999)

2999 FORMAT(' THE GREENS: ')

DO 4000 INDEX=257,512

WRITE(1,3999) TABLE(INDEX)

3999 FORMAT(I4)

CONTINUE

CALL WAIT

WRITE(1,4999)

4999 FORMAT(' THE BLUES: ')

DO 6000 INDEX=513,768

WRITE(1,5999) TABLE(INDEX)

5999 FORMAT(I4)

CONTINUE

DO 1000 INDEX = 1,1256

TABLE(INDEX) = 256-INDEX

TABLE(INDEX+256) = INDEX

TABLE(INDEX+512) = 122

1000 CONTINUE

C
0045    CALL WRPSU(TABLE)
0046    STOP
0047    END
&TXTAR T=00004 IS ON CR00021 USING 00003 BLKS R=0000

0001 PROGRAM TXTAR
0002 C
0003 C TEST RDTAR AND WRTAR
0004 C
0005 C KEITH MILLER, 6/21/84
0006 C
0007 C LOAD MODULES: %TXTAR, %RDTAR, %WRTAR, %RANGE
0008 C
0009 C INTEGER XCOOR, YCOOR ! CURSOR COORDINATES
0010 C INTEGER INDEX ! LOOP INDEX
0011 C
0012 C CALL URTAR(12,34) ! INITIAL POSITION OF TARGET
0013 C
0014 C DO 1000 INDEX = 1, 20
0015 C CALL RDTAR(XCOOR, YCOOR)
0016 C WRITE(1,500)XCOOR, YCOOR
0017 C 500 FORMAT(215)
0018 C CALL WRTAR(INDEX, INDEX)
0019 C 1000 CONTINUE
0020 C
0021 C CALL WRTAR(513,1) ! SHOULD BE AN ERROR ON X
0022 C CALL WRTAR(0,1) ! SHOULD BE AN ERROR ON X
0023 C CALL WRTAR(512,-1) ! SHOULD BE AN ERROR ON Y
0024 C STOP
0025 C END
0026 C
0027 C
0028 C
0029
HP DRIVER, DVR41, SOURCE CODE

&DOCU T=00004 IS ON CR00021 USING 00084 BLKS R=0000

0001 ASMB,L
0002 NAME DVR41
0003 **
0004 **
0005 **
0006 ** WITTEN BY M.BROUNE COULTER COMPUTER CORPORATION
0007 ** REWRITTEN BY R.W. Bagdasian HUGHES AIRCRAFT COMPANY.
0008 ** WRITTEN NOV., 1979
0009 ** REWRITTEN APR. 1980
0010 **
0011 ** DOCUMENTATION ADDED AUG. 1984, JIM MONTEITH & KEITH MILLER.
0012 **
0013 ** TO INVOKE THIS DRIVER FROM A FORTRAN ROUTINE, MAKE AN EXEC
0014 ** CALL LIKE THE ONE SHOWN BELOW. FOR MORE INFORMATION ON HOW
0015 ** THE EXEC CALL WORKS, SEE CHAPTER 2 IN THE HP MANUAL
0016 ** "RTE-6/VM PROGRAMMER'S REFERENCE MANUAL". FOR A LIST OF THE
0017 ** EQUIPMENT TABLE WORDS AND BITS, SEE SECTION L OF THE QUICK
0018 ** REFERENCE GUIDE FOR THE HP-1000.
0019 **
0020 ** SAMPLE CALL:
0021 **
0022 **
0023 **
0024 ** CALL EXEC(IDC, LU+IFUNC, IBUFF, ILEN, ICMND)
0025 **
0026 ** ICODE: THIS PARAMETER IS EITHER A 1, 2, OR 3.
0027 ** 1: SIGNIFIES A READ OPERATION. (COMTAL -> HP)
0028 ** 2: SIGNIFIES A WRITE OPERATION. (HP -> COMTAL)
0029 ** 3: SIGNIFIES A CONTROL OPERATION. THIS MODE DOESN'T
0030 ** SEEM TO WORK IN OUR SYSTEM.
0031 ** THE EXEC CALL SETS BITS 0 AND 1 OF WORD 6 OF THE
0032 ** EQUIPMENT TABLE (EOT6 IN THE CODE BELOW) ACCORDING
0033 ** TO THE ICODE VALUE.
0034 **
0035 **
0036 **
0037 **
0038 **
0039 **
0040 **
0041 **
0042 **
0043 **
0044 ** IFUNC: THIS PARAMETER SIGNALS THE DRIVER AS TO THE TYPE OF
0045 ** HP <-> COMTAL COMMUNICATION THAT IS BEING REQUESTED.
0046 ** ALTHOUGH THERE IS SOMETHING OF A RELATIONSHIP BETWEEN THE
0047 ** IFUNC CODES AND THE TRANSFER CODES OF SECTION 5.2 IN THE
0048 ** COMTAL USERS MANUAL, THE RELATIONSHIP IS HARD TO EXPLAIN.
0049 ** SO WE WOULDN'T TRY. INSTEAD, WE'LL GIVE A CHART THAT GIVES
0050 ** THE IFUNC VALUES AND THE RELEVANT COMTAL USER MANUAL
0051 ** SECTION NUMBERS WHERE FURTHER DETAILS ON THE TRANSFERS

200
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).

IFUNC FOR DFR MANAL REMARKS

00B 5.2.1 DISPLAY COMMANDS
100B 5.2.2 IMAGE/GRAPHIC TRANSFERS
200B 5.2.3 FUNCTION MEMORY (LOOK UP TABLE) TRANSFERS
300B 5.2.3 PSEUDO-COLOR TABLE TRANSFERS
400B 5.2.4.1 CURSOR AND TRACKBALL REQUESTS
500B 5.2.4.3 MACRO & COMMAND BLOCK TRANSFERS
5.2.4.4 IMAGE/GRAPHICS PARAMETER BLOCK
5.2.4.6 CODE BLOCK TRANSFERS
600B 5.2.2.3 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.

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

0104 *

0108 *

0109 I.41 NOP

0110 JSB SETIO

0111 CLB

0112 LDA EOT6.I  GET CONTROL SUBFUNCTION

0113 AND =B3700

0114 STA FUNC  SAVE FUNCTION MEMORY

0115 LDA EOT6.I  GET CONTROL WORD OF REQUEST

0116 AND =B3  ISOLATE THE FUNCTION

0117 CPA =B2  CHECK FOR WRITE (SKIP IF NOT WRITE)

0118 JMP RORWI  IF REQUEST FUNCTION = 2 (WRITE)

0119 LDB BIT15  SET READ FLAG FOR ACTUAL BIT USED

0120 CPA =B1  CHECK FOR READ (SKIP IF NOT READ)

0121 JMP RORWI  IF REQUEST FUNCTION = 1 (READ)

0122 * ANYTHING ELSE IS CONSIDERED A CONTROL REQUEST

0123 LDA FUNC  GET CONTROL WORD

0124 SZA

0125 JMP REJCT  FOR NOW ONLY CONTROL REQUEST "0" IS VALID

0126 CLR1 CLC DAT.C

0127 CLR2 CLC CST.C

0128 LDA =B4

0129 JMP I.41,i

0130 *

0131 *

0132 REJCT LDA =B2  SET A=2, FOR ILLEGAL CONTROL REQUEST

0133 JMP I.41,i  RETURN TO IOC

0134 *

0135 *

0136 *
SETIO ROUTINE FOR INITIATION SECTION FOLLOWS

STA B  SAVE SELECT CODE FROM A (STATUS CARD SC)
IOR CLC  OR IN "CLC" INSTRUCTION
STA CLR2
STA RORW
INA  INCREMENT TO HIGHER SELECT CODE (DATA CARD SC)
STA CLR1
STA RORW
LDA LIA  "LIA INSTRUCTION" TO A
IOR B  OR IN SELECT CODE
STA RORW2
LDA STC  "STC INSTRUCTION" TO A
IOR B
STA IO1L3  STORE INSTRUCTION AT LABEL "IO1L3"
LDA OTA
IOR B
STA IO1L2
LDA OTB
INB  INCREMENT SELECT CODE IN B
IOR B
STA IO1L1
JMP SETIO.I  RETURN

READ OR WRITE REQUEST

CHECK FOR COMTAL IN REMOTE

RORW1 STB RFLAG
RORU CLC DAT.C  CLEAR CONTROL AND FLAG
RORU1 CLC CST.C
RORU2 LIA CST  GET STATUS
AND NBSY  FIX FOR COMTAL VO/20  R.U.B.
CPA NBSY  FIX FOR COMTAL VO/20  R.U.B.
RSS  SKIP IF YES
JMP RORU2  FIX FOR COMTAL VO 20  R.U.B.

DETERMINE WHICH SUBFUNCTION

NOTE THAT THE DISPLAY COMMAND AND TRACK BALL REQUESTS ARE
PERFORMED USING DEVICE INTERRUPTS.
HOWEVER IMAGE LINE, FUNCTION, AND PSEUDO-COLOR TRANSFERS ARE
PERFORMED USING DEVICE INTERRUPTS TO TRANSMIT THE COMMAND AND
ENDING SEQUENCE, BUT WITH A DMA TRANSFER IN BETWEEN.

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 LINKY CHECK FOR LINK SUBFUNCTION
0196 JMP LINK
0197 * CPA DMTRV FUNCTION MEMORY TRANSFER?
0198 JMP CHDCP
0199 * CPA TBALV TRACK BALL REQUEST
0200 JMP TBALL
0201 * CPA SUPTV CHECK FOR TYPE 3 DMA SUPPORT CONTROL
0202 JMP CHDCP
0203 * CPA LINKY CHECK FOR LINK SUBFUNCTION
0204 * ILLEGAL SUBFUNCTION HERE ????????????
0205 * ??????????? PROCESS DISPLAY COMMAND REQUEST ******
0206 * DSPLY LDA =D1 SET STATE CONTROL VARIABLE
0207 * STA EOT12,I WHAT TO DO
0208 * LDB EOT9,I GET OPTION
0209 * CLA CLEAR OUT A
0210 * JMP IO1 GO TO START IO ROUTINE
0211 * PROCESS TRACK BALL REQUEST ******
0212 * TBALL LDB RFLAG
0213 * CPB BIT15 READ FLAG SET? (SKIP IF WRITE)
0214 * JMP RTBAL JUMP IF SET FOR READ
0215 * **** TRANSMIT WRITE COMMAND AND X-POSITION.
0216 ******** PROCESS TRACK BALL REQUEST ******
0217 * WTBAL LDA =D2 SET STATE CONTROL VARIABLE.
0218 * STA EOT12,I
0219 * LDB EOT7,I
LDA B.1 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 *
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 I.41.I RETURN TO IOC REQUESTING A DCPC CHANNEL
0255 CH1 LDB ≈B2 LOW SELECT CODE OF CHANNEL 1 TO B
0256 RSS 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 CHFTH LDA FUNC CONTROL WORD TO A
0263 CPA FMTRY
0264 JMP FMTR FUNCTION MEMORY TRANSFER
0265 CPA PCMTV
0266 JMP PCMTV 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
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******************************************************************************
0281 * FUNCTION MEMORY OR PSUEDO COLOR TRANSFER
0282******************************************************************************
0283 PCMTR CLA CLEAR A FOR Psuedo COLOR TRANSFER
0284 JMP PCMT2 CONTINUE TRANSFER
0285 *
0286 FMTR LDA BIT14 SET FUNCTION MEMORY BIT
0287 *
0288 PCMT2 LDB =D8 SET STATE VARIABLE
0289 STB EOT12,1 SAVE STATE FOR CONTINUATION
0290 IOR EOT9,1 MERGE IN IOPTN1 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******************************************************************************
0298 ***** SUPPORT FUNCTION BLOCK TRANSFER ( TR CODE = 3 )
0299******************************************************************************
0300 SUPPT LDA =D8 SET DMA TRANSFER RETURN STATUS FOR CONTINUATION SECTION
0301 STA EOT12,1 AND SAVE IT
0302 LDA EOT9,1 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 EOT12,1 SAVE THIS
0313 LDB EOT9,1 GET COMMAND OPTION
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
0321 *
0322 *
0323 101 NOP
0324 101L1 OTB DAT          PUT TRANSFER COMMAND ON DATA LINES
0325 IOR GOBIT
0326 101L2 OTA CST          PUT TRANSFER CODE ON COMMAND LINES
0327 101L3 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 WILLL EXEC.
0333 *
0334 *
0335 *
0336 ********************* COMPLETION SECTION FOLLOWS  *********************
0337 *
0338 *
0339 *
0340 C.41 NOP
0341 JSB CS10
0342 LDA EQI1.1          CHECK FOR SPURIOUS INTERRUPT
0343 AND B7777           GET I/O REQUEST LIST POINTER
0344 S2A          IS A REQUEST IN PROGRESS
0345 JMP COMP2          IF YES GO PROCESS REQUEST
0346 STA EQI15.1        NO, ITS SPURIOUS SO ZERO TIME-OUT CLOCK TO PREVENT TIME-OUT
0347 SPUR2 ISZ 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 EQI12.1        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 *
0367 SCA2 LDA IDSC INTERRUPTING DEVICE SELECT CODE TO A IS INTERRUPTING SELECT CODE THE STATUS CARD?
0368 CPA SSC SKIP IF YES?
0369 RSS
0370 JMP SPUR2 JUMP TO SPURIOUS INTERRUPT RETURN IF NOT SET
0371 LDA =D3
0372 STA EOT12,1 SET STATE CONTROL VARIABLE
0373 LDB EOT7,1 ADDRESS OF BUFFER TO B
0374 INB INCREMENT B (BUFFER ADDRESS)
0375 LDB Z,1 CONTENTS OF BUFFER+1 TO A
0376 AND =B0777
0377 IOR BIT14
0378 STA B
0379 LDA =B3 TRANSFER CODE OF 3
0380 IOR GOBIT
0381 JMP C101
0382 *
0383 ** TRACK BALL 2
0384 *
0385 SCA3 LDA IDSC INTERRUPTING DEVICE SELECT CODE TO A IS INTERRUPTING SELECT CODE THE STATUS CARD?
0386 CPA SSC SKIP IF YES?
0387 RSS
0388 JMP SPUR2 JUMP TO SPURIOUS INTERRUPT IF NOT SET
0389 LDA =D4
0390 STA EOT12,1 SET STATE CONTROL VARIABLE
0391 JMP TBDUN
0392 *
0393 ** TRACK BALL 3
0394 *
0395 SCA5 LDA IDSC INTERRUPTING DEVICE SELECT CODE TO A IS INTERRUPTING SELECT CODE THE STATUS CARD?
0396 CPA SSC SKIP IF YES?
0397 RSS
0398 JMP SPUR2 JUMP TO SPURIOUS INTERRUPT IF NOT SET
0399 LDA =D6
0400 STA EOT12,1 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 IS INTERRUPTING SELECT CODE THE DSCA CARD?
0406 CPA DSC SKIP IF YES?
0407 RSS
0408 JMP SPUR2 JUMP TO SPURIOUS INTERRUPT IF NOT SET
0409 LDA =D7
0410 STA EOT12,1 SET STATE CONTROL VARIABLE
0411 GET1 LIA DAT DATA FROM DATA LINES TO A (X-POSITION & SWITCH 1)
0412 * STA EOT13,1 SAVE FOR SWITCH 1
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 ** TRACKBALL 5
0418
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 TBDUN LDA =B3 FOR CLBIT THIS ADDED PER RUB 6/15/81
0433 IOR CLBIT
0434 IOR ENBIT
0435 JMP CI03
0436 ** TRACK BALL TRANSMIT END
0437
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 C103 MOP
0453 OTB DAT
0454 C103 OTA.CST
0455 STC CST.C
0456 ISZ C.41
0457 JMP C.41.I
0458 *
0460  * 
0461  CIA2  NOP 
0462  CLC DAT.C 
0463  STC DAT.C 
0464  ISZ C.41 
0465  JMP C.41,1 
0466  * 
0467  ** DMA Transfer 1 
0468  * 
0469  SCAB 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 EOT11.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
0500  IOR DSC
0501  STA D7+1
0502  *
0503  *
0504  THIS ROUTINE INITIATES THE DCPC DATA TRANSFERS
0505  THE INSTRUCTIONS BELOW WITH D LABELS ARE MODIFIED BY THE
0506  ROUTINE "SETIO" TO SPECIFY THE CORRECT SELECT CODES
0507  FOR THE DCPC CHANNEL ASSIGNED BY THE SYSTEM
0508  *
0509  *
0510  GOIO LDA EOT4,I
0511  AND #177  MASK FOR COMMAND SELECT CODE
0512  INA  INCREMENT FOR DATA SELECT CODE
0513  IOR BIT15  TURN ON HANDSHAKE BIT (STC)
0514  OTA 6B  PUT CONTROL WORD 1 TO DCPC CHANNEL (SELECT 6 OR 7)
0515  D2 CLC 2B  PREPARE MEM. ADDR. REGISTER FOR CW2 (SELECT CODE 2 OR 3)
0516  LDA EOT6,I  GET CONTROL WORD
0517  AND #3  MASK FOR FUNCTION
0518  STA B  STORE FUNCTION IN B
0519  LDA EOT7,I  ADDRESS OF BUFFER TO A
0520  CPB =B1  IS FUNCTION A READ?
0521  IOR BIT15  TURN ON "IN" BIT FOR A READ, IF YES
0522  OTA 2B  CONTROL WORD 2 TO DCPC CHANNEL (SC 2 OR 3)
0523  STC 2B  PREPARE MEM. ADDR. REGISTER FOR CW3 (SC 2 OR 3)
0524  LDA EOT8,I  GET LENGTH OF TRANSFER
0525  CMA, INA  MAKE TWO'S COMPLEMENT
0526  D5 OTA 2B  CONTROL WORD 3 TO DCPC CHANNEL (SELECT CODE 2 OR 3)
0527  CPB =B1  IS IT A READ?
0528  JUMP IF YES
0529  D6 STC 6B  ACTIVATE DCPC CHANNEL (SELECT CODE 6 OR 7)
0530  CLC DAT  CLEAR DEVICE
0531  STF DAT  ACTIVATE DEVICE?
0532  JMP GOEND
0533  D7 STC 6B  ACTIVATE DCPC CHANNEL (SELECT CODE 6 OR 7)
0534  STC DAT,C  ACTIVATE DEVICE
0535  GOEND ISZ C.41  NOW RETURN TO CIC WITH
0536  JMP C.41,I  CONTINUATION
0537  *
0538  DMA TRANSFER 2
0539  *
0540  SCA9 LDA EOT11,I  GET LOW SELECT CODE OF DCPC CHANNEL
0541  ADA =B4  ADD 4 TO CONVERT LOW DCPC SELECT CODE TO HIGH S.C.
0542  CHKDF CPA IDSC  IS INTERRUPTING SELECT CODE THAT OF THE
0543  ASSIGNED DCPC CHANNEL?
0544  RSS  SKIP IF YES
0545  JMP SPUR2  JUMP TO SPURIOUS INTERRUPT IF NOT
LDA =D10   SET CONTROL STATE VARIABLE
STA EQT12,1

**** SET CONTROL STATE VARIABLE  

0554

LDA TRAN
IOR CLBIT  * SET CLEAR BIT
IOR ENBIT  * ADD END BIT
JMP CL03   * GO TO IT

0556

** DMA TRANSFER 3

0564

SCA10 LDA IDSC  INTERRUPTING DEVICE SELECT CODE TO A
CPA SSC  IS INTERRUPTING SELECT CODE THE STATUS CARD?
RS  SKIP IF YES?
JMP SPUR2  IF NOT GO TO SPURIOUS INTERRUPT
LDB EQT1,1 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

0575

CD1 CLC 2B.C  CLEAR LOW DCPC SELECT CODE
CD2 CLC 6B.C  CLEAR HIGH DCPC SELECT CODE
LDB EQT8,1 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
STA EQT5.1 REMOVE PREVIOUS STATUS
AND =B177400 BITS IN EQT WORD 5
IOR SAVE1  OR IN NEW BITS
STA EQT5.1 AND RESET INTO EQT WORD 5

0586

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

0592

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

0594

**** SETIO SUBROUTINE FOR COMPLETION/CONTINUATION SECTION ****
0597 *
0598 CSIO NOP
0599 STA IDSC
0600 LDA EOT4,1
0601 AND #077
0602 STA SSC
0603 INA
0604 STA DSC
0605 LDA OTA
0606 IOR SSC
0607 STA 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 CI0.1 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
THE FOLLOWING VALUES (SUFFIX V) DETERMINE WHICH DRIVER SUBFUNCTIONS ARE INVOKED, PARAMETER IFUNC IN SAMPLE CALL ABOVE.

**

DSPLV OCT 0
ILTRV OCT 100
FMTRV OCT 200
PCTMV OCT 300
TBALV OCT 400
SUPTV OCT 500
LINKV OCT 600

**

CST EQU 24 USED ONLY FOR DOCUMENTATION. ACTUAL INSTRUCTION IS SET UP BY A SETIO ROUTINE.
DAT EQU 25
A EQU 0
D EQU 1
OTA OTA 0
CLC CLC 0.C
STC STC 0.C
LIA LIA 0
OTB OTB 0
SFS SFS 0
STF STF 0

BASE PAGE COMMUNICATIONS AREA DEFINITIONS
EQU 1650B

**

THE FOLLOWING CONSTANTS ARE USED TO READ THE EQUIPMENT TABLE ASSOCIATED WITH THIS DRIVER. SEE THE HP-1000 QUICK REFERENCE
0689 ** MANUAL FOR DIAGRAMS AND TABLES ON WHICH THESE CONSTANTS ARE BASED.
0690 **
0691 INTBA EQU .+4
0692 EOT1  EQU .+8
0693 EOT2  EQU .+9
0694 EOT3  EQU .+10
0695 EOT4  EQU .+11
0696 EOT5  EQU .+12
0697 EOT6  EQU .+13
0698 EOT7  EQU .+14
0699 EOT8  EQU .+15
0700 EOT9  EQU .+16
0701 EOT10 EQU .+17
0702 EOT11 EQU .+18
0703 EOT12 EQU .+81
0704 EOT13 EQU .+82
0705 EOT14 EQU .+83
0706 EOT15 EQU .+84
0707 *
0708 *
0709 *
0710 ******** STATE CONTROL ADDRESS TABLE ********
0711 CTABA DEF CTAB
0712 CTAB  NOP
0713 JMP SCA1  DISPLAY COMMAND COMPLETION
0714 JMP SCA2  TRACK BALL 1
0715 JMP SCA3  TRACK BALL 2
0716 JMP SCA4  TRACK BALL TRANSMIT END
0717 JMP SCA5  TRACK BALL 3
0718 JMP SCA6  TRACK BALL 4
0719 JMP SCA7  TRACK BALL 5
0720 JMP SCA8  DMA TRANSFER 1
0721 JMP SCA9  DMA TRANSFER 2
0722 JMP SCA10 DMA TRANSFER 3
0723 *  BSS 10
0724 END
### Abstract

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.