NASA Technical Memorandum 87646

IPLIB (IMAGE PROCESSING LIBRARY) USER'S MANUAL

FOR REFERENCE

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

DECEMBER 1985

NASA
National Aeronautics and Space Administration
Langley Research Center
Hampton, Virginia 23665
INTRODUCTION

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

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

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

The manual is divided into six sections:

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

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

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

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

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

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

The following references are suggested for programmers working with IPLIB:


**SUBPROGRAM HANDLING**

**LOADING SUBPROGRAMS**

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

```
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) I<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

3
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).
BUFDS : transfers a monochrome image to COMTAL from disk (p. 19).
CLFDS : transfers a color image to COMTAL from disk (p. 22).
CLRGR : clears a COMTAL graphics plane (p. 26).
CLRIM : clears a monochrome COMTAL image (p. 28).
CMND2 : sends a COMTAL command given as a constant string from HP to COMTAL (p. 38).
CMND : sends a COMTAL command given as constant string and length parameter from HP to COMTAL (p. 39).
COUNT : counts 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).
ICOPY : copies one monochrome COMTAL image to another (using CMND) (p. 57).
ICPY2 : copies one monochrome COMTAL image to another (using RDILN & WRILN) (p. 60).
MERGE : merges two bytes into one byte (all arguments are INTEGER) (p. 63).
NORM : finds lowest pixel value in an image, then subtracts that value from all pixels in that image; used to get light table variations (p. 65).
NOTE2 : writes a line of characters into a graphics plane with a given color and size at a location; takes a constant string argument (p. 67).
NOTES : writes a line of characters into a graphics plane with a given color and size at a location; takes a character array and length (p. 72).
PAINT : interactive "painting" of square patches on COMTAL image (p. 76).
PROFF : gives HP access to the COMTAL profiling capabilities (p. 80).
RANGE : logical function that determines if 1st argument is within 2nd & 3rd (p. 83).
RDGLH : 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).
RDILH : reads one horizontal line of COMTAL pixels; 2 pixels/integer returned (p. 92).
RDIP : reads one pixel from a COMTAL monochrome image (p. 94).
RDIRC : reads a rectangle of pixels from a COMTAL monochrome image (p. 99).
RDLUT : reads the contents of a COMTAL look-up table (p. 101).
RDPST : reads the contents of a COMTAL pseudo-color table (p. 104).
RDTAB : reads the COMTAL Image/Graphics Table (p. 106).
RDTR : reads the COMTAL cursor location (p. 109).
SETV : sets a COMTAL image 5 - 9 to the video camera (p. 111).
SPLIT : splits an integer into two bytes, both bytes stored in new integers (p. 114).
SPRED : finds low and high pixel values in an image, and does a linear stretch on all pixel values to expand the range to 0 - 255 (p. 116).
SUB12 : subtracts two images with an offset of 12; 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).
WIPGR : 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).
WRIL2 : 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).
WRIPT : writes one pixel value to a COMTAL image memory (p. 150).
WIRIC : writes an array of integers to a rectangle of a COMTAL image memory; one pixel value / integer in the buffer (p. 153).
URLUT : 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 ADDS (p. 163).
TADDI : program that tests ADDIM (p. 164).
TCLR : program that tests DSPCL, "Display Color" (p. 165).
TCLRG : program that tests CLRGR, "Clear Graphics" (p. 166).
TCLRI : program that tests CLRIM, "Clear Image" (p. 167).
TCMM2 : program that sends all possible single bytes to COMTAL 1 at a time (p. 168).
TCMPSN : program that sends COMTAL commands via the HP keyboard; tests CMDND (p. 169).
TCTH : program that tests subroutine COUNT (p. 170).
TCONS : 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 DSPBW, "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 PROF, "PROFiling" (p. 179).
TRANG : program that tests the function "RANGE" (p. 180).
TRDATA : program that tests RDTAR, "Read TARget" (p. 181).
TSETV : program that tests SETV, "SET Video camera" (p. 182).
TSPRD : program that tests SPRED, "SPREAd pixel values" (p. 183).
TSUBI : program that tests SUBIM and SUB12 (p. 184).
TTIORS : program that tests THRS, "THRSholding" (p. 185).
TTSTI : program that tests TSTII, "Test Image I" (p. 186).
TV2C : program that tests TV2CM and TV2C4, "TV to CoMpat" transfers (p. 187).
TWAIT : program that tests WAIT (p. 188).
TUPE : 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 WRLIN 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 WIRLN and RDILN, "Write Image Line" and "Read Image Line" (p. 193).
TXIPT : program that tests WRIPT and RDIPT, "Write Image Point" and "Read Image Point" (p. 194).
TXIRC : program that tests WIRIC and RDIIC, "Write Image ReCTangle" and "Read Image ReCTangle" (p. 195).
TXLUT : program that tests WRLUT and RDILUT, "Write Look Up Table" and "Read Look Up Table" (p. 196).
TXPSU : program that tests WRPSU and RDPSU, "Write PSeUdo-color table" and "Read PSeUdo-color table" (p. 197).
TXTAR : program that tests WRTAR and RDTAR, "Write TARget" and "Read TARget" (p. 199).

CROSS-REFERENCE FILE

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

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

ADDI2 calls:

DIGIT COMMAND RANGE

ADDIM calls:
CMND DIGIT RANGE  calls:
BUFDS        calls:
  CMND DIGIT OPEN RANGE READF WRILN
CLDS         calls:
  CMND DIGIT OPEN RANGE READF WRILN
CLRGR        calls:
  CMND DIGIT RANGE
CLRIM        calls:
  CMND DIGIT RANGE
CMNN2        calls:
  **LEN
CMNN          calls no other procedures.
COUNT         calls:
  RANGE RDIL2
DELAY         calls no other procedures.
DIGIT         calls:
  RANGE
DSPBW         calls:
  RANGE
DSPCL         calls:
  CMND DIGIT RANGE
DSPGR         calls:
  CMNN2 DIGIT RANGE
DSPVD         calls:
  CMND DIGIT
HILO          calls:
  RDIL2
HISTO         calls:
  CMNN2 DELAY DIGIT RANGE
ICOPY         calls:
  CMNN2 DIGIT RANGE
ICPY2         calls:
  CMND DIGIT RANGE
MERGE         calls no other procedures.
NORML         calls:
  HILO RDIL2 WRIL2
NOTE2         calls:
  ADDGR CMNN2 CMND DELAY DIGIT *LEN RANGE
NOTES         calls:
  CMNN2 CMND DELAY DIGIT DSPGR RANGE
PAINT         calls:
  CMND RDTAR WAIT WRIRC
PROFL         calls:
  CHAR CMND DIGIT RANGE WAIT
RANGE         calls no other procedures.
RDGLN         calls:
RA
NE
R
calls:
BTEST RANGE
RDIL2 calls:
*ICHAR RANGE
RDILN calls:
RANGE
RDIPT calls:
*ICHAR RANGE
RDIRC calls:
*ICHAR RANGE RDILN
RDLUT calls:
RANGE
RDPSU calls no other procedures.
RDTAB calls:
BTEST RANGE
RDTAR calls no other procedures.
SETV calls:
CMDND DIGIT RANGE
SPLIT calls no other procedures.
SPRED calls:
FLOAT HILO IFIX RDIL2 WRIL2
SSORT calls no other procedures.
SUB12 calls:
CMDND DIGIT RANGE
SUBIM calls:
CMDND DIGIT RANGE
THRSH calls:
RANGE RDIL2 WRIL2
TSTI1 calls:
WRILN
TV2C4 calls:
ADDI2 RANGE TV2CM
TV2CM calls:
CMDND DIGIT DSPBW RANGE
WAIT calls no other procedures.
UIPGR calls:
CMDN2 DIGIT RANGE
URGLN calls:
RANGE
URGPT calls:
*IBCLR *IBSET RANGE
WRIL2 calls:
CHAR RANGE
WRILN 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
cLLR calls:
  CLRGR CMND DIGIT RANGE
cLLRI calls:
  CLRIM CMND DIGIT RANGE
cMMN2 calls:
  CMMDND WAIT
cMMN calls:
  CMMD
TCNT calls:
  COUNT RANGE RDIL2
TCONS calls no other procedures.
TCOPY calls:
  CMMDND DIGIT ICOPY RANGE
TDELA calls:
  DELAY CMMD2 WAIT
TDIGI calls:
  DIGIT RANGE
TDSP calls:
  CMMDND DSPBU DSPCL RANGE WAIT
TDSPV calls:
  DSPVD CMMDND DIGIT WAIT DSPBU RANGE
THIST calls:
  CMMDND DIGIT HISTO RANGE
TNORM calls:
  HILO NORML RANGE RDIL2 WRIL2
TNOTE calls:
  CMMDND DELAY DIGIT DSPGR NOTE2 NOTES RANGE WRTAR
TPNT calls:
CMNND PAINT RANGE RDILN RDTAR WRILN WRIRC

TPROF
  calls:

CMNND DIGIT PROFL RANGE WAIT

TRANG
  calls:

RANGE

TRDTA
  calls:

RANGE RDTAB

TRSTT
  calls no other procedures.

TSETV
  calls:

SETV RANGE DIGIT CMNND DSPBU

TSPRD
  calls:

RANGE RDIL2 SPRED WRIL2

TSSRT
  calls no other procedures.

TSUBI
  calls:

CMNND DIGIT RANGE SUBI2 SUBIM WAIT

TTHRS
  calls:

RANGE RDIL2 THSH WRIL2

TTSTI
  calls no other procedures.

TTV2C
  calls:

ADDI2 CMNND DIGIT DSPBU DSPVD RANGE TV2C4 TV2CM WAIT

TWAIT
  calls:

WAIT

TWIPE
  calls:

CMNND DIGIT DSPGR RANGE WAIT WIPGR

TXFDS
  calls:

BUFDS CLFDS CMNND DIGIT RANGE WRILM

TXGLN
  calls:

RANGE RDGLN URGLN

TXGPT
  calls:

RANGE RDGPT RDTAR WRGPT

TXILN
  calls:

RANGE RDILN WRILN

TXIPT
  calls:

RANGE RDIPT WAIT WRIPT

TXIRC
  calls:

RANGE RDILN RDIRC WRILN WRIRC

TXLUT
  calls:

RANGE RDLUT WRLOT

TXPSU
  calls:

RDPSU WRPSU RANGE WAIT

TXTAR
  calls:

RDTAR WRTAR RANGE
&ADDIM T=00004 IS ON CR00021 USING 00024 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC...
SUBPROGRAMS CALLED:

<table>
<thead>
<tr>
<th>name</th>
<th>source</th>
<th>load</th>
<th>remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMIND</td>
<td>&amp;CMIND</td>
<td>&amp;CMIND</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>&amp;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>&amp;DIGIT</td>
<td>character*1 function which returns '0'-'9' according to integer input 0-9.</td>
</tr>
</tbody>
</table>

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

REVISION HISTORY:

LOCAL VARIABLES:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEGER</td>
<td>IBUF(128)</td>
</tr>
<tr>
<td>CHARACTER*255</td>
<td>CBUF</td>
</tr>
<tr>
<td>EQUIVALENCE</td>
<td>(IBUF,CBUF)</td>
</tr>
<tr>
<td>LOGICAL</td>
<td>RANGE</td>
</tr>
<tr>
<td>INTEGER</td>
<td>IMLO, IMHI</td>
</tr>
<tr>
<td>INTEGER</td>
<td>TERM</td>
</tr>
<tr>
<td>CHARACTER*1</td>
<td>DIGIT</td>
</tr>
</tbody>
</table>

INITIALIZATIONS:

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

PROCESSING:

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>IF (.NOT.(RANGE(A,IMLO,IMHI))) GOTO 8801</td>
</tr>
<tr>
<td>IF (.NOT.(RANGE(B,IMLO,IMHI))) GOTO 8101</td>
</tr>
<tr>
<td>IF (.NOT.(RANGE(C,IMLO,IMHI))) GOTO 8201</td>
</tr>
</tbody>
</table>

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

In this notation, letters in caps were entered into CBUF, and
lower case letters are the full commands filled in by the COMTAL
NOTE: this code assumes that the digit 0 is NOT a legal value for
the parameters A, B, and C.
The "$" separates COMTAL commands.

CBUF =

1 'UN I B $'/
UNassign Image B ! just in case B is already assigned.

2 'AS T B $'/DIGIT(C)/' $'/DIGIT(A)/' $'/DIGIT(B)/' $'/
ASsign Truecolor image B red #C blue #A green #B

3 'D I B $'/
Display Image B

4 'SE COM G + B / 1 $'/
sets 0 offset by default, and
/' 1' sets no scaling.

5 'A COM $'/
Add COMbine

6 'I $'/DIGIT(C)/' D R $'/
Image #C = Displayed Image Red ! Red is arbitrary, since difference
of images is monochrome

7 'D I $'/DIGIT(C)/' $'/
Display Image #C

8 'SU COM '
SUBtract COMbine.

CALL COMND(IBUF, 7B)
RETURN

C**ERROR RETURNS

E001 WRITE(TERM, E203) A
E003 FORMAT( 'THE 2ND IMAGE PARAMETER,' , I3, ' IS OUT OF RANGE.' )
GOTO 8900

E010 WRITE(TERM, E103) B
E103 FORMAT( 'THE 3RD IMAGE PARAMETER,' , I3, ' IS OUT OF RANGE.' )
GOTO 8900

E201 WRITE(TERM, E203) C
E203 FORMAT( 'THE 1ST IMAGE PARAMETER,' , I3, ' IS OUT OF RANGE.' )
GOTO 8900
0137  8900 WRITE(TERM, 8901)
0138  8901 FORMAT(' ADDIM RETURNS WITHOUT FURTHER PROCESSING.')
0139      RETURN
0140  C
0141      END
0142      END
**ADDI2**

```fortran
&ADDI2 T=00004 IS ON CR00021 USING 00024 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE ADDI2(C, A, B, SCALE)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
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 (C = A + B)
0009 INTEGER A, B ! the images whose sum is taken (C = A + B)
0010 INTEGER SCALE ! each pixel sum divided by this number.
0011 C
0012 C***INTRODUCTION:
0013 C
0014 C The subroutine "ADD Images" takes a pixel by pixel sum
0015 C of images A and B, divides each sum by SCALE, and places the results
0016 C into image C. SCALE must be between 1 and 9 inclusive.
0017 C The truecolor image B is used in the processing of ADDI2, and will
0018 C be left as the combination of C, A, and B for red, green, and blue
0019 C respectively.
0020 C
0021 C***LANGUAGE:
0022 C
0023 C FORTRAN 77, the HP 1000 version for RTE-G/VIM.
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.

**SUBPROGRAMS CALLED:**

<table>
<thead>
<tr>
<th>name</th>
<th>source</th>
<th>load</th>
<th>remarks</th>
</tr>
</thead>
</table>

CMMND &CMMND XCMMND Sends a command to the COMTAL as if the command were sent from the keyboard.

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

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

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

**REVISION HISTORY:**

**LOCAL VARIABLES:**

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

**INITIALIZATIONS:**

DATA IMLO/1/, IMHI/4/

DATA TERM/1/

**PROCESSING**

IF (.NOT.(RANGE(A,IMLO,IMHI))) GOTO 8001 ! error return
IF (.NOT.(RANGE(B,IMLO,IMHI))) GOTO 8101 ! error return
IF (.NOT.(RANGE(C,IMLO,IMHI))) GOTO 8201 ! error return

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

In this notation, letters in caps were entered into CBUF, and
lower case letters are the full commands filled in by the COMTAL
NOTE: this code assumes that the digit 0 is NOT a legal value for
the parameters A, B, and C.
The "$" separates COMTAL commands.

CBUF =

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

CALL CMDND(IBMUF, 77)
RETURN

ERROR RETURNS

8001 WRITE(TERM, 8003) A
8003 FORMAT(' THE 2ND IMAGE PARAMETER.' I3, ', ', IS OUT OF RANGE.' )
GOTO 8500

8101 WRITE(TERM, 8103) B
8103 FORMAT(' THE 3RD IMAGE PARAMETER.' I3, ', ', IS OUT OF RANGE.' )
GOTO 8500

8201 WRITE(TERM, 8203) C
8203 FORMAT(' THE 1ST IMAGE PARAMETER.' I3, ', ', IS OUT OF RANGE.' )
GOTO 8500

10
0137  B900 WRITE(TERM. B901)
0138  B901 FORMAT( "ADDI2 RETURNS WITHOUT FURTHER PROCESSING." )
0139    RETURN
0140  C
0141    END
0142
&BLFDS T=00004 IS ON CR02021 USING 00018 BLKS R=0000

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

**Revision History:**

**Local Variables:**

- **INTEGER DBLOCK(144)**: A data control block used in file ID.
- **INTEGER IERR**: Holds HP 10 return code.
- **INTEGER IBUF(256)**: Buffers one horizontal row of COMTAL pixels.
- **INTEGER ROW**: Loop indexing which COMTAL row.
- **LOGICAL RANGE**: Logical function that determines if 1st parameter is between 2nd and 3rd, inclusive.
- **INTEGER IMLO, IMHI**: Limits on COMTAL image numbers.
- **INTEGER TERM**: Logical unit for terminal output.
- **CHARACTER*1 DIGIT**: Function that returns '0', '1', ..., '9' according to a 0,1,...,9 integer input.
- **CHARACTER*255 CBUF**: Overlays IBUF.
- **EQUIVALENCE (CBUF, IBUF)**

**Initializations:**

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

**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.
- **CBUF = 'D I '/DIGIT(IMAGE)**!: Display the image to be filled.
- **CALL CMND(IBUF,5)**
- **DO 8000 ROW = 0, 511**
- **CALL READF(DBLOCK, IERR, IBUF)**
- **CALL WRLN(IMAGE, ROW, IBUF)**
- **1000 CONTINUE**
0091       RETURN
0092  C
0093 *ERROR RETURNS
0094  C
0095  8001 WRITE(TERM, 8003) IMAGE, IMLO, IMHI
0096  8003 FORMAT(' IMAGE NUMBER,'.I3,' IS OUT OF RANGE:''.2I4,'."
0097          GOTO 8900
0098  C
0099  8201 WRITE(TERM, 8203) IERR
0100  8203 FORMAT(' ERROR OCCURED DURING IMAGE FILE OPENING:''.14,'')
0101          GOTO 8900
0102  C
0103  8301 WRITE(TERM, 8303) IERR
0104  8303 FORMAT(' ERROR OCCURED DURING IMAGE FILE READ:''.14,'')
0105          GOTO 8900
0106  C
0107  8900 WRITE(TERM, 8901)
0108  8901 FORMAT(' BWFDS FAILS. NO TRANSFER TAKES PLACE.')
0109          RETURN
0110       END
**INTRODUCTION:**

The subroutine "Color image From Disk" transfers three b&w images from the HP to the COMTAL, and then assigns these to a truecolor image on the COMTAL. The filename names a single file holding all three monochrome images. The COLOR image number is automatically unassigned and assigned by CLFDS, and after all three component images are transferred, the color image is displayed.

**LANGUAGE:**

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

**LIMITATIONS:**

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

**SUBPROGRAMS CALLED:**

name    source    load    remarks
--------- ------- ------- ------------------------------------------
RANGE    &RANGE   %RANGE logical function that determines if the 1st argument is within the 2nd and 3rd inclusive.
OPEN     ------ ------- HP FORTRAN77 intrinsic subroutine; opens a file
READF    ------ ------- HP FORTRAN77 intrinsic subroutine; reads a record from a file; requires an OPENed data block.
WRILN    &WRILN   %WRILN writes a COMTAL Image horizontal Line; 2 pixels
per integer in the buffer.

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

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

C*WRITTEN BY:

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

C*REVISION HISTORY:

C*LOCAL VARIABLES:

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

C*INITIALIZATIONS:

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

C*PROCESSING

IF (.NOT.(RANGE(RED ,IMLO,IMHI))) GOTO 8001 ! error return
IF (.NOT.(RANGE(GREEN ,IMLO,IMHI))) GOTO 8101 ! error return
IF (.NOT.(RANGE(BLUE ,IMLO,IMHI))) GOTO 8201 ! error return
IF ((RED .EQ. GREEN) .OR. (GREEN .EQ. BLUE)
0091 1 .OR. (RED .EQ. BLUE)) GOTO 8301 ! error return
0092 C
0093 IF (.NOT.(RANGE(COLOR, TRLO, TRHI)))) GOTO 8401 ! error return
0094 C
0095 CALL OPEN,DBLOCK, IERR, FLNAME)
0096 IF (IERR .LT. 0) GOTO 8501 ! error return, open failed
0097 C
0098 CBUF = *D I */DIGIT(RED) ! Display the RED Image as it is filled.
0099 CALL CMDND(IBUF,5)
0100 DO 1000 ROW = 0.511
0101 CALL READF(DBLOCK, IERR, IBUF)
0102 IF (IERR .LT. 0) GOTO 8601 ! error return, file read failed
0103 CALL WRILN(RED, ROW, IBUF)
0104 1000 CONTINUE
0105 C
0106 CBUF = *D I */DIGIT(GREEN) ! Display the GREEN Image as it is filled.
0107 CALL CMDND(IBUF,5)
0108 DO 2000 ROW = 0.511
0109 CALL READF(DBLOCK, IERR, IBUF)
0110 IF (IERR .LT. 0) GOTO 8701 ! error return, file read failed
0111 CALL WRILN(GREEN, ROW, IBUF)
0112 2000 CONTINUE
0113 C
0114 CBUF = *D I */DIGIT(BLUE) ! Display the BLUE Image as it is filled.
0115 CALL CMDND(IBUF,5)
0116 DO 3000 ROW = 0.511
0117 CALL READF(DBLOCK, IERR, IBUF)
0118 IF (IERR .LT. 0) GOTO 8801 ! error return, file read failed
0119 CALL WRILN(BLUE, ROW, IBUF)
0120 3000 CONTINUE
0121 C
0122 C Let #C, #R, #G, #B be the DIGIT associated with COLOR, RED,
0123 C GREEN, and BLUE respectively; then the following CMDND calls
0124 C read as follows: UNassign Image #C; ASsign Truecolor #C
0125 C red #R green #G blue #B; Display Image #C
0126 C
0127 CBUF = 'UN I */DIGIT(COLOR)
0128 CALL CMDND(IBUF,6)
0129 CBUF = 'AS T */DIGIT(COLOR)//' */DIGIT(RED)//' */
0130 I */DIGIT(GREEN)//' */DIGIT(BLUE)
0131 CALL CMDND(IBUF,12)
0132 CBUF = 'D I */DIGIT(COLOR)
0133 CALL CMDND(IBUF,5)
0134 RETURN
0135 C
0136 C**ERROR RETURNS
0137 C
0138 8001 WRITE(TERM, 8003) RED, IMLO, IMHI
0139 8003 FORMAT(' RED IMAGE NUMBER,'E3,' IS OUT OF RANGE:'E14,'. ')
0140 GOTO 8900
0141 C
0142 8101 WRITE(TERM, 8103) GREEN, IMLO, IMHI
0143 8103 FORMAT(' GREEN IMAGE NUMBER,'E3,' IS OUT OF RANGE:'E14,'. ')
0144 GOTO 8900
0145 C
0146 8201 WRITE(TERM, 8203) BLUE, IMLO, IMHI
0147 8203 FORMAT(' BLUE IMAGE NUMBER,'E3,' IS OUT OF RANGE:'E14,'. ')
0148 GOTO 8900
0149 C
0150 8301 WRITE(TERM, 8303) RED, GREEN, BLUE
0151 8303 FORMAT(' 3 MONOCROME IMAGES MUST BE DISTINCT. YOURS:'E14)
0152 GOTO 8900
0153 C
0154 8401 WRITE(TERM, 8403) COLOR, CLLO, CLHI
0155 8403 FORMAT(' YOUR TRUECOLOR IMAGE,'E14,' IS OUT OF RANGE:'E14)
0156 GOTO 8900
0157 C
0158 8501 WRITE(TERM, 8503) IERR
0159 8503 FORMAT(' ERROR WHILE OPENING IMAGE FILE:'E, IS)
0160 GOTO 8900
0161 C
0162 8601 WRITE(TERM, 8603) IERR
0163 8603 FORMAT(' ERROR WHILE READING IN THE RED MONOCROME IMAGE:'E, IS)
0164 GOTO 8900
0165 C
0166 8701 WRITE(TERM, 8703) IERR
0167 8703 FORMAT(' ERROR WHILE READING IN THE GREEN MONOCROME IMAGE:'E, IS)
0168 GOTO 8900
0169 C
0170 8801 WRITE(TERM, 8803) IERR
0171 8803 FORMAT(' ERROR WHILE READING IN THE BLUE MONOCROME IMAGE:'E, IS)
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 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
INTEGER TERM ! 1st parameter lies within last 2 parameters.
INTEGER IBUF(128) ! COMTAL command buffer
CHARACTER*255 CBUF ! overlays the IBUF command buffer.
EQUIVALENCE (IBUF, CBUF)

C

C***INITIALIZATIONS:
DATA GRLO/1/, GRHI/4/
DATA TERM/1/

C

C***PROCESSING
IF (.NOT.(RANGE(GRAPH, GRLO, GRHI))) GOTO 8001 ! error return
C ELSE...clear the GRAPH
C
C
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
C***ERROR RETURN
C
WRITE(TERM, 8003) GRAPH, GRLO, GRHI
8003 FORMAT(' GRAPH NUMBER, ', I4, ' IS OUT OF RANGE; ', 2I3, '.')
WRITE(TERM, 8005)
8005 FORMAT(' CLRGR FAILS. NO COMMAND SENT TO COMTAL.')
RETURN
END
SUBROUTINE CLRIM( IMAGE )

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

INTRODUCTION:
CLRIM Image clears the image designated by IMAGE. IMAGE should be within the range 1-4 for the present IRD COMTAL system.
If IMAGE is out of range, an error message is printed and no COMTAL transfer takes place.

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

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

SUBPROGRAMS CALLED:
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 subprogram is based was written by
HETTIE D. FAULCON, July, 1983. This modification is by
KEITH MILLER, June, 1984.
0045  INTEGER IMLO, IMHI  ! limits on COMTAL image numbers
0046  LOGICAL  RANGE    ! logical function that determines if its
0047                          ! 1st parameter lies within last 2 parameters.
0048  INTEGER  TERM      ! logical unit number of the terminal output
0049  INTEGER  IBUF(128) ! COMTAL command buffer
0050  CHARACTER*1 DIGIT  ! function that returns the ASCII character
0051  C                             ! associated with integer input, 0,...,9.
0052  CHARACTER*255 CBUF          ! overlays the IBUF command buffer
0053  EQUIVALENCE (IBUF, CBUF)
0054  C
0055  C**INITIALIZATIONS:
0056  C
0057  DATA IMLO/1/, IMHI/4/
0058  DATA TERM/1/
0059  C
0060  C**PROCESSING
0061  C
0062  IF (.NOT.(RANGE(IMAGE, IMLO, IMHI))) GOTO 8001 ! error return
0063  C
0064  CBUF = 'CLEAR IMAGE' // DIGIT(IMAGE) ! since CBUF overlays IBUF,
0065                          ! this statement loads IBUF with
0066                          ! the COMTAL command
0067  CALL CMMND(IBUF, 13)
0068  RETURN
0069  C
0070  C**ERROR RETURN
0071  C
0072  WRITE(TERM, 8003) IMAGE, IMLO, IMHI
0073  8003 FORMAT(' IMAGE NUMBER, ', 14, ' IS OUT OF RANGE; ', 213, '.')
0074  WRITE(TERM, 8005)
0075  8005 FORMAT(' CLRRIM FAILS. NO COMMAND SENT TO COMTAL.')
0076  RETURN
0077  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.
CMN2 is designed to be sent constant strings.

The major differences are that the INBUF command string may include
multiple commands, each separated by the character "$".
A character array buffer is used in equivalence with an integer array
in this subroutine to illustrate the utility of the characters and
still allow obvious compatibility.

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

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

In order to make it easier to send single COMTAL commands to CMMND,
the final blank, $, and the required zero byte are added automatically
to the INBUF string.
If the last character isn't a blank, CMMND adds one.
However, the caller should NOT add the final $ or zero byte to the
string. Note that each $ in the string should be preceded by a blank.

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

name source load remarks


```fortran
0045 C ----- ----- ----- -------------------------------------------------------
0046 C LEN ----- ----- integer function returns length of character string
0047 C
0048 C***WRITTEN BY:
0049 C
0050 C The original code upon which this subroutine is based was written
0051 C by HETTIE D. FAULCON, JULY, 1983.
0052 C
0053 C***REVISION HISTORY:
0054 C
0055 C Modified by Keith Miller, 6/18/84.
0056 C
0057 C***LOCAL VARIABLES:
0058 C
0059 CHARACTER*1 CBUF(256) ! character buffer
0060 INTEGER IBUF(128) ! the character buffer overlayed as integers
0061 INTEGER IZERO ! constant value 0 for making a 0 byte (ZERO)
0062 INTEGER WORDS ! counts number of words
0063 INTEGER NUMCHR ! counts number of bytes
0064 INTEGER LEN ! intrinsic HP FORTRAN?77 function that returns
0065 C ! the length of a character string.
0066 CHARACTER*1 BLANK, DOLLAR ! special ASCII characters
0067 CHARACTER*1 ZERO ! zero is 00000000 binary.
0068 EQUIVALENCE (CBUF,IBUF), (ZERO, IZERO)
0069 C
0070 C***INITIALIZATIONS:
0071 C
0072 DATA BLANK/' '/, DOLLAR/'$'/, IZERO/0/
0073 C
0074 NUMCHR = LEN(INBUF)
0075 C
0076 C***PROCESSING:
0077 C
0078 DO 10 INDEX = 1, NUMCHR
0079 CBUF(INDEX) = INBUF(INDEX:INDEX)
0080 10 CONTINUE
0081 C
0082 IF (CBUF(NUMCHR) .EQ. BLANK) GOTO 30
0083 C... ELSE...
0084 NUMCHR = NUMCHR + 1
0085 CBUF(NUMCHR) = BLANK
0086 C
0087 C... ADD ENDING CHARACTERS TO COMMAND
0088 C
0089 30 CBUF(NUMCHR+1) = DOLLAR
0090 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
SUBROUTINE CMMND (INBUF, INCNT)

INTEGER INBUF(129) ! the characters of a COMTAL command string
INTEGER INCNT    ! the number of characters in command string

This subroutine "Command" allows a character string command to be
sent to the COMTAL much as if the command were typed at the keyboard.
The major differences are that the INBUF command string may include
multiple commands, each separated by the character "$". Notice
that NUMCHR is in terms of characters (bytes), not words. INBUF
is an integer array to be compatible with previously written software,
but an F77 character array would probably be more appropriate.
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.

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

In order to make it easier to send single COMTAL commands to CMMND,
the final blank, $, and the required zero byte are added automatically
to the INBUF string.
If the last character isn't a blank, CMMND adds one.
However, the caller should NOT add the final $ or zero byte to the
string. Note that each $ in the string should be preceded by a blank.

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

C**SUBPROGRAMS CALLED: NONE.

C**WRITTEN BY:
The original code upon which this subroutine is based was written by NETTIE D. FAULCON, JULY, 1983.

**REVISION HISTORY:**

Modified by Keith Miller, 6/18/84.

**LOCAL VARIABLES:**

- **CHARACTER**: 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**: BLANK, DOLLAR ! special ASCII characters
- **INTEGER**: ZERO ! zero is 00000000 binary.
- **EQUIVALENCE**: (CBUF,IBUF), (ZERO, IZERO)

**INITIALIZATIONS:**

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

NUMCHR = INCNT ! protects the input parameter, since NUMCHR is reassigned in the subroutine.

**PROCESSING:**

WORDS = (NUMCHR+1) / 2
DO 10 INDEX = 1, WORDS
   IBUF(INDEX) = INBUF(INDEX)
10   CONTINUE

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

**ELSE...**

NUMCHR = NUMCHR + 1
CBUF(NUMCHR) = BLANK

**ADD ENDING CHARACTERS TO COMMAND**

CBUF(NUMCHR+1) = DOLLAR
CBUF(NUMCHR+2) = ZERO
CBUF(NUMCHR+3) = ZERO ! safety precaution
NUMCHR = NUMCHR + 2

**CHANGE BYTE COUNT TO WORD COUNT**
```c
0091   WORDS = (NUMCHR+1) / 2 ! if N is even, intentional truncation
0092 C
0093 C Programming note:
0094 C The EXEC command parameters are discussed in the HP RTE-6/VM
0096 C are discussed in section 5.2.4 of the COMTAL User's Manual.
0097 C
0098 C The first parameter to EXEC identifies it as a write command.
0099 C The second parameter identifies the resident HP driver (36B)
0100 C and gives a code for the operation required by this call (500B).
0101 C The third parameter is the command string, and the fourth gives
0102 C the length in words of the buffer that is to be used. The fifth
0103 C parameter is a code for the COMTAL interface that directs the
0104 C command transfer.
0105 C
0106 CALL EXEC(2, 36B + 500B, IBUF, WORDS, 24001B)
0107 RETURN
0108 C
0109 END
```
Subroutine COUNT examines each pixel value in the COMTAL image associated with the number IMAGE, and compiles a count of how many pixels hold the values 0-255. These 256 counts are returned in the INTEGER*4 array COUNTS.

**Language:**

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

**Limitations:**

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

**Subroutines Called:**

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.

**Written By:**

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

**Revision History:**
LOCAL VARIABLES:

LOGICAL RANGE ! logical function that determines if its 1st parameter is within the 2nd and 3rd parameters.
INTEGER INDEX ! loop index variable
INTEGER LBUF(512) ! Line BUFFER holds a row of pixels.
INTEGER WHICH ! changes from pixel value to count array index
INTEGER IMLO, IMHI ! limits on COMTAL image numbers
INTEGER ROW, COL ! loop indices
INTEGER TERM ! logical unit for terminal output

INITIALIZATIONS:
DATA IMLO/1/, IMHI/4/
DATA TERM/1/

PROCESSING
IF (.NOT.(RANGE(IMAGE, IMLO, IMHI))) GOTO 8001 ! error return
DO 1000 INDEX = 1, 256
COUNTS(INDEX) = 0
CONTINUE
1000 COUNTS(INDEX) = 0

DO 2000 ROW = 0, 511
CALL RDIL2(LBUF, IMAGE, ROW)
DO 3800 COL = 1, 512
WHICH = LBUF(COL) + 1 ! "+1" required because pixel values are 0-255, COUNTS array is indexed 1-256.
COUNTS(WHICH) = COUNTS(WHICH) + 1
CONTINUE
2000 CONTINUE
3000 CONTINUE
RETURN

ERROR RETURN
8001 WRITE(TERM, 8003) IMAGE, IMLO, IMHI
8003 FORMAT(IMAGE NUMBER, 14, OUT OF RANGE:, 214)
8901 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 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE DELAY(SECOND)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
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(&) ! the milliseconds, seconds, minutes, hours,
! and Julian day when procedure entered.
! the INTIME in units of tens of milliseconds 
! from 0.0,0.0,0.
! the milliseconds, seconds, minutes, hours,
! and Julian day of the latest EXEC call that 
! NOW in units of tens of milliseconds from 
! 0.0,0.0,0.
! determines the time.
! the EXEC number for a time request.
! logical unit for terminal output.

C

C*INITIALIZATIONS:

DATA TIMREQ/11/
DATA TERM/1/

C

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) + 36000*365*INTIME(5)

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

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

RETURN

C

C*ERROR RETURN:

8001 WRITE(TERM,8003) SECOND
8003 FORMAT(' THE SECONDS COUNT.', IS, ', IS <= 0.')
8000 8900 WRITE(TERM,8901)
8901 FORMAT(' DELAY FAILS. NO TIMED DELAY OCCURS.')
8902 RETURN
8903 END
&DIGIT T=00004 IS ON CR00021 USING 00012 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 CHARACTER FUNCTION DIGIT(ININTIN)
0003 C
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 RANGEC &RANGEC XRANGEC logical function determines if the first parameter
0029 C lies within (inclusive) the next two parameters.
0030 C
0031 C****WRITTEN BY:
0032 C
0034 C
0035 C****REVISION HISTORY:
0036 C
0037 C
0038 C****LOCAL VARIABLES:
0039 C
0040 C CHARACTER 1DGTARA(10)! DIGIT ARRAY holds the digits '0'-'9'
0041 C LOGICAL RANGE ! function that determines if its first parameter
0042 C INTEGER TERM ! logical unit number for terminal output.
**INITIALIZATIONS:**

DATA \( \text{DGTAR} = \{'0','1','2','3','4','5','6','7','8','9'\} \)

DATA \( \text{TERM} = 1 \)

**PROCESSING**

IF (.NOT.(RANGE(INIT, 0, 9))) GOTO 8001 ! error return

ELSE... convert to digit and return

DIGIT = DGTAR(INIT + 1)

RETURN

ERROR RETURN

WRITE(TERM, 8003) INIT

FORMAT(\"THE INPUT TO DIGIT\.\". I4,

1 \". IS NOT A SINGLE DIGIT. DIGIT RETURNS A BLANK.\")

DIGIT = ''

RETURN

END
**SUBROUTINE DSPBW(IMAGE)**

**PARAMETER DECLARATIONS:**

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

**INTRODUCTION:**

"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.

**LIMITATIONS:**

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

**SUBPROGRAMS CALLED:**

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

**WRITTEN BY:**

Keith Miller, June, 1984.

**REVISION HISTORY:**

The code on which this subroutine is based was written by

Keith Miller, June, 1984.

**LOCAL VARIABLES:**

- LOGICAL RANGE ! a function for determining if an integer is
C within a certain range
INTEGER TERM ! the logical unit number for the terminal
INTEGER IDUMMY! fills the place of an unused EXEC parameter
C
C***INITIALIZATIONS:
C
DATA TERM/1/
C
C***PROCESSING
C
IF (RANGE(IMAGE,1,4)) GOTO 2000 ! legal image number
ELSE... illegal image number
WRITE(TERM, 1001) IMAGE
1001 FORMAT( 'The image number ', I3, ' is out of range.' )
WRITE(TERM, 1002)
1002 FORMAT( 'DSDBU fails. No action taken on command.' )
RETURN
C
2000 CONTINUE ! send a display command to COMTAL
C
In the following call, the first parameter indicates a write
operation. The second parameter is a combination of two codes:
000B + 36B. 36B indicates the proper resident driver, and 000B
informs the driver (DRV41) that we require a display operation.
The third and fourth parameters are ignored. The final parameter
indicates the image to be displayed. The subtraction in that
final parameter is necessary because the COMTAL images are numbered
0 to 3; the multiplication is necessary to push the image number
into the proper bits in the command word sent to the COMTAL.
CALL EXEC(2, 000B + 36B, IDUMMY, 0, (IMAGE-1) * 2)
RETURN
END
&DSCPL T=00004 IS ON CR00021 USING 00006 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

0002 SUBROUTINE DSPCL (RED, 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 DSPBW to display black and white images.

0019 C

0020 C The color display is accomplished via the CMDIND 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 CMDIND &CMDIND %CMDIND transfers a command string to the COMTAL, which

0044 C accepts it almost as a keyboard command.
C DIGIT &DIGIT %DIGIT character%1 function that returns a single digit
on legal integer inputs 0-9.

C***WRITTEN BY:

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

C***REVISION HISTORY:

C 0856 C
C
C 0858 C***LOCAL VARIABLES:
C 0859 C
C
8860 INTEGER IBUF(256)
8861 LOGICAL RANGE
8862 INTEGER IMLO, IMHI
8863 INTEGER TRLO, TRHI
8864 INTEGER TERM
8865 CHARACTER*255 CBUF
8866 EQUIVALENT (CBUF, IBUF)
8867 CHARACTER*1 DIGIT

C***INITIALIZATIONS:

C 0873 C
C
C 0874 DATA IMLO/1/, IMHI/4/
C 0875 DATA TRLO/5/, TRHI/5/
C 0876 DATA TERM/1/

C***PROCESSING

C 0879 C

0880 IF (.NOT.(RANGE(RED, IMLO, IMHI))) GOTO 8001 ! error return
0881 IF (.NOT.(RANGE(GREEN, IMLO, IMHI))) GOTO 8101 ! error return
0882 IF (.NOT.(RANGE(BLUE, IMLO, IMHI))) GOTO 8201 ! error return

0883 IF ((RED .EQ. GREEN) .OR. (GREEN .EQ. BLUE))
0884 1 .OR. (RED .EQ. BLUE))
0885 GOTO 8301 ! error return

0886 IF (.NOT.(RANGE(TCLR, TRLO, TRHI))) GOTO 8401 ! error return

0888 C DISPLAY THE COLOR IMAGE
0890 C The following commands are abbreviations of the following
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 CMDN(IBUF.6)

CBUF = 'AS T'  //DIGIT(TCLR)  //DIGIT(RED)/""  //DIGIT(GREEN)/""  //DIGIT(BLUE)

CALL CMDN(IBUF.12)

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

CALL CMDN(IBUF.5)

RETURN

ERROR RETURNS

WRITE(TERM, 8003) RED, IMLO, IMHI

FORMAT(" RED IMAGE NUMBER, '.13.' IS OUT OF RANGE: '.214.'")

GOTO 8900

WRITE(TERM, 8103) GREEN, IMLO, IMHI

FORMAT(" GREEN IMAGE NUMBER, '.13.' IS OUT OF RANGE: '.214.'")

GOTO 8900

WRITE(TERM, 8203) BLUE, IMLO, IMHI

FORMAT(" BLUE IMAGE NUMBER, '.13.' IS OUT OF RANGE: '.214.'")

GOTO 8900

WRITE(TERM, 8303) RED, GREEN, BLUE

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

GOTO 8900

WRITE(TERM, 8403) TCLR, CLLO, CLHI

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

GOTO 8900

WRITE(TERM, 8900)

FORMAT(" DSPCL FAILS. NO DISPLAY TAKES PLACE.")

RETURN

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

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE DSPGR(GRNUM)
0003 "GRNUM! a number 1-4 designating a COMTAL graphics plane.
0004 C
0005 PARAMETER DECLARATIONS:
0006 C
0007 INTEGER G RNUM ! a number 1-4 designating a COMTAL graphics plane.
0008 C
0009 INTRODUCTION:
0010 C
0011 "Display Graphics" is a subroutine that allows the caller
0012 to send a display command to the COMTAL from an HP program.
0013 The call can turn on one graphics plane, number 1, 2, 3, 4.
0014 C
0015 LIMITATIONS:
0016 C
0017 This subroutine does not turn off previous graphics planes.
0018 C
0019 That is accomplished using the subroutine WIPGR.
0020 C
0021 It is OK to call DSPGR repeatedly without an intervening WIPGR.
0022 C
0023 The extra calls have no effect, but they don't hang up the COMTAL.
0024 C
0025 SUBPROGRAMS CALLED:
0026 C
0027 name source load remarks
0028 ----- ------ ------ --------------------------------------------------
0029 RANGE &RANGE %RANGE logical function that determines if the
0030 first parameter is within the bounds defined
0031 by the second and third parameter (inclusive).
0032 CMN1 &CMN1 %CMN1 sends a constant string to the COMTAL as if
0033 the string were typed on the COMTAL keyboard.
0034 CMN2 &CMN2 %CMN2 sends a constant string to the COMTAL as if
0035 the string were typed on the COMTAL keyboard.
0036 DIGIT &DIGIT %DIGIT character*1 function that returns '0'..'9'
0037 according to integer input 0..9.
0038 C
0039 WRITTEN BY:
0040 C
0041 The code on which this subroutine is based was written by
0042 NETTIE D. FAULCON. July, 1983. This modification is by
0043 KEITH MILLER. June, 1984.
0044 C
0045 REVISION HISTORY:
0046 C
0047 LOCAL V A R I A B L E S:
C LOGICAL RANGE ! a function for determining if an integer is
C within a certain range.
C INTEGER TERM ! the logical unit number for the terminal.
C INTEGER IDUMMY ! fills the place of an unused EXEC parameter.
C INTEGER GRLD,GRHI ! limits on a graphics plane number.
C CHARACTER*1 DIGIT ! function that returns '0'..'9' for input
C integers 0..9.

C***INITIALIZATIONS:
C
C DATA TERM/1/ 
C DATA GRLD/1/,GRHI/4/ 

C***PROCESSING
C
C IF (.NOT.(RANGE(GRNUM,GRLD,GRHI))) GOTO 8801 ! error return
C
C "ADD Graphics *GRNUM", where *GRNUM stands for the digit
C corresponding to GRNUM value.
C
C CALL CMN2('ADD GR */DIGIT(GRNUM))
C RETURN
C
C***ERROR RETURN:
C
C WRITE(TERM, 8803) GRNUM, GRLD, GRHI
C 8803 FORMAT( ' THE GRAPHICS NUMBER,'"13."' OUT OF RANGE:'"214)
C
C WRITE(TERM, 8901)
C 8901 FORMAT(' DSPGR fails. No action taken on command.' )
C RETURN
C END
SUBROUTINE DSPVD

C_INPUT PARAMETERS:
None.

C_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.

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

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

C_SUBPROGRAMS CALLED:

name            source     load    remarks
-----            -----      -----    ---------------
CMMND &CMMND %CMMND transfers a command string to the COMTAL, which accepts it as a keyboard command.
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:
INTEGER IBUF(256) ! buffers COMTAL command
CHARACTER*255 CBUF ! character overlay for sending COMTAL commands
EQUIVALENCE (CBUF, IBUF)
INTEGER TVIMA6 ! COMTAL image 6 for video camera
CHARACTER*1 DIGIT ! function that returns '0'- '9' according to integer input 0-9.

INITIALIZATIONS:
DATA TVIMA/5/ ! arbitrary choice.

PROCESSING
DISPLAY THE VIDEO IMAGE
CBUF = 'DISPLAY IMAGE '//DIGIT(TVIMA)//'
CALL CMND(IBUF,16)
RETURN
END
&HIL0  T=00004  IS ON CR00021  USING 00004  BLKS R=0000

   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

   SUBROUTINE HILO(HI, LO, IMAGE)

   C

   PARAMETER DECLARATIONS:

   INTEGER HI, LO  ! output parameters, the high and low pixel values
   INTEGER IMAGE  ! found in the designated image.

   INTRODUCTION:

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

   LANGUAGE:

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

   LIMITATIONS:

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

   SUBPROGRAMS CALLED:

   name    source    load    remarks
   -------  -------    -------  ------------------------------
   RANGE    RANGE    XRANGE  logical function that determines if 1st argument
   is within 2nd & 3rd inclusive.
   RDIL2    RDIL2    %RDIL2  reads a horizontal line of pixel values into
                           a 512 integer array.

   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(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**

- `RETURN`
&HISTO T=00004 IS ON CR0021 USING 00005 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE HISTO(IMAGE)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
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 load CMMN2 sends a constant string to the COMTAL, which
0039 C treats it as a command typed on the COMTAL.
0040 C DELAY &DELAY load XDELAY puts the HP into a busy wait for at least the
0041 C given number of seconds.
0042 C DIGIT &DIGIT load XDIGIT character*1 function that returns '0'-'9'
0043 C RANGE &RANGE load XRANGE logical function that determines if its 1st
argument lies within its 2nd and 3rd, inclusive.

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

**REVISION HISTORY:**

**LOCAL VARIABLES:**

**INITIALIZATIONS:**

**PROCESSING:**

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

IMCHAR = DIGIT(IMAGE) ! delay initialization until after IMAGE has been found to be within its limits.

The following COMTAL command expands to:

Function memory @I = Histogram of image where @I is the single digit associated with IMAGE.

Image @I is automatically used for the histogram.

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

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

CALL DELAY(15) ! 14 seconds experimentally determined as the
! time it takes to compile a histogram.
CALL CM**2("D F "/IMCHAR)! "Display Function memory *I"
RETURN

***ERROR RETURNS:

WRITE(TERM,8003) IMAGE ,IMLO,IMHI
8003 FORMAT( ' THE IMAGE NUMBER,'.IS,' IS OUT OF RANGE:'2I5,'')
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  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002  SUBROUTINE ICOPY(OUTIMG, INIMG)
0003  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
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:
The code on which this subroutine is based was written by NETTIE D. FAULCON in July, 1983. This modification is by KEITH MILLER June, 1984.

REVISION HISTORY:

LOCAL VARIABLES:

LOGICAL RANGE function determines if 1st argument is within 2nd and 3rd argument inclusive.

CHARACTER*1 DIGIT function returns '0'-'9' according to integer input 0-9.

CHARACTER*255 CCOMM character buffer for building up a call to the CMND subroutine.

INTEGER IBUF (128) integer overlay of CCOMM

EQUIVALENCE (CCOMM, IBUF)

INTEGER IMLO, IMHI the range of legal COMTAL image numbers

INTEGER TERM terminal logical unit

INITIALIZATION:

DATA IMLO/1/, IMHI/4/

DATA TERM/1/

PROCESSING:

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

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

Legal image numbers, so do the copy

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

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

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

RETURN

ERROR RETURNS:

8801 WRITE(TERM, 8003)OUTIMG, IMLO, IMHI
0091 8003 FORMAT( 'YOUR OUTPUT PARAMETER.', 15, ' IS OUT OF RANGE:' , 214)
0092   GOTO 8900
0093 C
0094 8101 WRITE(TERM, 8103)INIMG, IMLO, IMHI
0095 8103 FORMAT( 'YOUR INPUT PARAMETER.' , 15, ' IS OUT OF RANGE:' , 214)
0096   GOTO 8900
0097 C
0098 8900 WRITE(TERM, 8901)
0099 8901 FORMAT( 'ICOPY FAILS. NO COPYING TAKES PLACE.' )
0100   RETURN
0101   END
**COPY2** T=00004 IS ON CR00021 USING 00018 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE ICPY2(OUTIMG, INIMG)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
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" commands the COMTAL to copy one black and
0013 C white image into another. The OUTIMG destination image
0014 C is, of course, wiped out by this exchange. The input
0015 C image for ICPY2 can be any single digit number; the output
0016 C image must be a COMTAL image memory plane (1-4 currently).
0017 C The companion subroutine ICPY requires that both images
0018 C be COMTAL image memory planes.
0019 C
0020 C***LANGUAGE:
0021 C
0022 C FORTRAN 77, the HP-100 version for RTE-6/VM.
0023 C
0024 C***LIMITATIONS:
0025 C
0026 C ICPY only works for the grey level images of COMTAL, not the
0027 C truecolor images.
0028 C
0029 C***SUBPROGRAMS CALLED:
0030 C
0031 C name source load remarks
0032 C
0033 C CMDND &CMDND %CMDND 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

60
The code on which this subroutine is based was written by NETTIE D. FAULCON in July, 1983. This modification is by KEITH MILLER, June, 1984.

**REVISION HISTORY:**

**LOCAL VARIABLES:**

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

**COMTAL VARIABLES:**

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

**INITIALIZATION:**

**PROCESSING:**

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

Legal image numbers, so do the copy

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

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

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

RETURN

ERROR RETURNS:

6001 WRITE(TERM, 6003) OUTIMG, IMLO, IMHI
6003 FORMAT(' YOUR OUTPUT PARAMETER, ', IS, ', IS OUT OF RANGE: ', 214)
GOTO 8900

C

WRITE(TERM, 8103) IN IMG

FORMAT(' YOUR INPUT PARAMETER ', I5, ' IS OUT OF RANGE: 1, 9')

GOTO 8900

C

WRITE(TERM, 8901)

FORMAT(' ICYP2 FAILS. NO COPYING TAKES PLACE.')

RETURN

END
&MERGE T=00004 IS ON C00021 USING 00005 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE MERGE(OUTURD, BYTE1, BYTE2)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
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 Nettie D. Faulcon, July, 1983. This subprogram was written by
0036 C Keith Miller, July, 1984, with the support of a NASA-ASEE
0037 C summer fellowship.
0038 C
0039 C**REVISION HISTORY:
0040 C
0041 C
0042 C**LOCAL VARIABLES:
0043 C
0044 INTEGER IHOLD ! an INTEGER interpretation of bits
0045 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.

PROCESSING:

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

RETURN

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

0001  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002  SUBROUTINE NORMAL(IMAGE)
0003  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004  C
0005 C***PARAMETER DECLARATIONS:
0006  C
0007  INTEGER IMAGE ! designates a COMTAL image to "normalize" (see below)
0008  C
0009 C***INTRODUCTION:
0010  C
0011  C The subroutine NORMAL searches through an image to find its lowest
0012  C pixel value. Then NORMAL replaces each pixel in the image
0013  C (call that value X) with the value (X - low). This subroutine was
0014  C developed to obtain an image of the variations inherent in the lighting
0015  C table that should give a constant background light, but is instead giving
0016  C a light with a variation of as many as 18 grey scale levels out of 255.
0017  C The "normalized" background image is subtracted from the digitized image
0018  C to simulate a uniform background.
0019  C
0020 C***LANGUAGE:
0021  C
0022  C FORTRAN 77, the HP-1B08 version for RTE-6/VM.
0023  C
0024 C***LIMITATIONS:
0025  C
0026  C This subroutine makes two passes through the image, one to obtain the
0027  C lowest pixel value, and one to write out the new pixel values. In some
0028  C applications, the programmer may want to add new processes during one
0029  C of those passes. Also, the subroutine HILO is used here, even though
0030  C only the lowest value is required. To optimize, create a new subroutine
0031  C which only determines the low value.
0032  C
0033 C***SUBPROGRAMS CALLED:
0034  C
0035  C
0036  name  source  load  remarks
0037  -------- -------- -------- -------------------------------------------
0038  C CRIL2 CRIL2 CRIL2 reads one horizontal COMTAL image line into
0039  C an integer array; one integer/pixel.
0040  C WRIL2 WRIL2 WRIL2 writes one horizontal COMTAL image line from
0041  C an integer array; one integer/pixel.
0042  C HILO GHIL0 %HILO determines the highest and lowest pixel value
0043  C in an image.
0044 C***WRITTEN BY:
E:05 C
E:06 C  The code on which this subprogram is based was written by
E:07 C  NETTIE D. FAULCON, July, 1983. This subprogram was written by
E:08 C  KEITH MILLER, July, 1984, with the support of a NASA-ASEE
E:09 C  summer fellowship.
E:10 C
E:11 C****REVISION HISTORY:
E:12 C
E:13 C
E:14 C
E:15 C****LOCAL VARIABLES:
E:16 C
E:17 C  INTEGER IBUF(512) ! buffer for pixel values; one pixel / integer
E:18 C  INTEGER ARALO, ARAHI! array bounds for a pixel buffer
E:19 C  INTEGER LNLO, LNHI ! limits on COMTAL row numbers
E:20 C  INTEGER HI, LO ! highest and lowest pixel values in IMAGE
E:21 C  INTEGER ROW, COL ! loop indices
E:22 C
E:23 C****INITIALIZATIONS:
E:24 C
E:25 C  DATA ARALO/1/, ARAHI/512/ 
E:26 C  DATA LNLO /0/, LNHI /511/ 
E:27 C
E:28 C****PROCESSING
E:29 C
E:30 C  CALL HILO(HI, LO, IMAGE)
E:31 C
E:32 C
E:33 C  DO 2000 ROW = LNLO, LNHI 
E:34 C    CALL RDIL2(IBUF, IMAGE, ROW)
E:35 C
E:36 C  DO 1000 COL = ARALO, ARAHI 
E:37 C    IBUF(COL) = IBUF(COL) - LO
E:38 C  1000 CONTINUE
E:39 C
E:40 C  CALL WRIL2(IMAGE, ROW, IBUF)
E:41 C
E:42 C  2000 CONTINUE
E:43 C
E:44 C  RETURN
E:45 C
E:46 C  END
&NOTE2 T=00004 IS ON CR00021 USING 00024 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
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>CMTN2</td>
<td>&amp;CMTN2</td>
<td>XCMTN2</td>
<td>sends constant string to COMTAL as if it were typed at the COMTAL keyboard.</td>
</tr>
<tr>
<td>CMTND</td>
<td>&amp;CMTND</td>
<td>XCMTND</td>
<td>sends command string and length parameter to COMTAL as if it were typed at the COMTAL keyboard.</td>
</tr>
<tr>
<td>DELAY</td>
<td>&amp;DELAY</td>
<td>XDELAY</td>
<td>puts the HP in a busy wait; the argument to DELAY gives the number of seconds to DELAY.</td>
</tr>
<tr>
<td>DIGIT</td>
<td>&amp;DIGIT</td>
<td>XDIGIT</td>
<td>character*1 function that returns '0'-9' according to integer input 0-9.</td>
</tr>
<tr>
<td>DSPGR</td>
<td>&amp;DSPGR</td>
<td>XDSPGR</td>
<td>adds a graphics plane to the display.</td>
</tr>
<tr>
<td>LEN</td>
<td>------</td>
<td>------</td>
<td>HP FORTRAN intrinsic integer function that returns the length of a character string.</td>
</tr>
<tr>
<td>RANGE</td>
<td>&amp;RANGE</td>
<td>XRANGE</td>
<td>logical function that determines if 1st parameter is within the 2nd &amp; 3rd inclusive.</td>
</tr>
</tbody>
</table>

**WRITTEN BY:**

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

**REVISION HISTORY:**

**LOCAL VARIABLES:**

- CHARACTER*1 DIGIT | function that returns '0'-9' for integer input 0-9.
- INTEGER LEN | intrinsic HP FORTRAN function that returns the length of a character string.
- LOGICAL RANGE | logical function determines if 1st argument is within 2nd and 3rd arguments, inclusive.
- CHARACTER*1 GRCHAR | the single digit that corresponds to GRNUM.
- INTEGER LONG | length of the character string.
- INTEGER START | the # of first character in CBUF that holds the first character of NOTE.
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)

C***INITIALIZATIONS:
DATA GRLO/1/, GRHI/4/
DATA CMLO/0/, CMHI/511/
DATA FCLO/1/, FCHI/16/
DATA TERM/1/

C***PROCESSING:
IF (.NOT.(RANGE(GRNUM, GRLO, GRHI))) GOTO 8080 ! 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(FACCTOR, 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 8400 ! error return
Color graphics red
1000 CALL CMM2('CO G '//GRCHAR//' RED')
GOTO 2000
Color graphics green
1100 CALL CMM2('CO G '//GRCHAR//' GRN')
GOTO 2000
Color graphics blue
1200 CALL CMM2('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 CBUR(1:8) = ’G ’/GRCHAR/’ L ’/DIGIT(FACTOR)/’ ’
0158 START = 9
0159 LONG = LONG + 8
0160 GOTO 4000
0161 C THEN...
0162 3000 CBUR(1:9) = ’G ’/GRCHAR/’ L 1’/DIGIT(FACTOR-9)/’ ’
0163 START = 10
0164 LONG = LONG + 9
0165 GOTO 4000
0166 C
0167 4000 CBUR(START:LONG) = NOTE
0168 CALL CMMND(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 ERROR RETURNS:
0175 C
0176 8001 WRITE(TERM,8003)GRNUM,GRLO,GRAH
0177 8003 FORMAT(’ THE GRAPHICS NUMBER,’.15,’ IS OUT OF RANGE:’,.214,’.’)
0178 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 GOTO 8900
0183 C
0184 8201 WRITE(TERM,8203) YCOORD,CMLO,CMHI
0185 8203 FORMAT( ' THE Y COORDINATE,'J5,' IS OUT OF RANGE,'214,'')
0186        GOTO 8900
0187 C
0188 8301 WRITE(TERM,8303) FACTOR,FCLO,FCHI
0189 8303 FORMAT( ' THE SCALE FACTOR,'J5,' IS OUT OF RANGE,'214,'')
0190        GOTO 8900
0191 C
0192 8401 WRITE(TERM,8403) COLOR
0193 8403 FORMAT( ' THE COLOR PARAMETER,'A1,' 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 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE NOTES (GRNUM, XCOORD, YCOORD, COLOR, FACTOR, NOTE, LENGTH)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***PARAMETER DECLARATIONS:
0006 C
0007 INTEGER GRNUM ! the number of the graphic to be written into;
0008 C INTEGER XCOORD ! NOTES doesn't clear or display this graphic.
0009 INTEGER YCOORD ! XCOORD and YCOORD define the upper left corner of the
0010 C INTEGER FACTOR ! position of the first character in the NOTE.
0011 C INTEGER COLOR ! controls size of characters plotted; 1-16.
0012 CHARACTER*1 NOTE ! signals if you wish to stay the same color ('S')
0013 CHARACTER*1 COLOR ! or change to red ('R'), green ('G'), or blue ('B').
0014 CHARACTER*255 NOTE ! the message to be printed in graphics is in the
0015 C INTEGER LENGTH ! first LENGTH characters of this string.
0016 C INTEGER LENGTH ! the number of NOTE characters that are to be used.
0017 C
0018 C***INTRODUCTION:
0019 C
0020 C This subroutine, NOTES writes a line of characters into a COMTAL graphics
0021 C plane. The parameter NOTE should be a declared string.
0022 C A very similar subroutine, NOTE2, uses a string constant without a length
0023 C parameter.
0024 C
0025 C***LANGUAGE:
0026 C
0027 C FORTRAN 77, the HP-1000 version for RTE-6/VM.
0028 C
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"), 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:**

<table>
<thead>
<tr>
<th>name</th>
<th>source</th>
<th>load</th>
<th>remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMMN2</td>
<td>&amp;CMMN2</td>
<td>&amp;CMMN2</td>
<td>%CMMN2 sends constant string to COMTAL as if it were typed at the COMTAL keyboard.</td>
</tr>
<tr>
<td>CMMND</td>
<td>&amp;CMMND</td>
<td>&amp;CMMND</td>
<td>%CMMND sends a fixed string command to COMTAL as if it were typed at the COMTAL keyboard.</td>
</tr>
<tr>
<td>DELAY</td>
<td>&amp;DELAY</td>
<td>&amp;DELAY</td>
<td>%DELAY puts the HP in a busy wait for the number of seconds designated in DELAY's parameter.</td>
</tr>
<tr>
<td>DIGIT</td>
<td>&amp;DIGIT</td>
<td>&amp;DIGIT</td>
<td>%DIGIT character function that returns '0'-9 according to integer input 0-9.</td>
</tr>
<tr>
<td>DSPGR</td>
<td>&amp;DSPGR</td>
<td>&amp;DSPGR</td>
<td>%DSPGR adds a graphic plane to the display.</td>
</tr>
<tr>
<td>RANGE</td>
<td>&amp;RANGE</td>
<td>&amp;RANGE</td>
<td>%RANGE logical function that determines if the 1st argument is within the 2nd &amp; 3rd inclusive.</td>
</tr>
</tbody>
</table>

**WITTEN BY:**

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

**REVISION HISTORY:**

<table>
<thead>
<tr>
<th>LOCAL VARIABLES:</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHARACTER*1: DIGIT</td>
</tr>
<tr>
<td>LOGICAL: RANGE</td>
</tr>
<tr>
<td>CHARACTER*1: GRCHAR</td>
</tr>
<tr>
<td>INTEGER: Long</td>
</tr>
<tr>
<td>INTEGER: GRLO, GRHI</td>
</tr>
<tr>
<td>INTEGER: CMLO, CMHI</td>
</tr>
<tr>
<td>INTEGER: FCLO, FCHI</td>
</tr>
<tr>
<td>INTEGER: STLO, STHI</td>
</tr>
<tr>
<td>INTEGER: TERM</td>
</tr>
<tr>
<td>CHARACTER*255: CBUF</td>
</tr>
</tbody>
</table>
INTEGER IBUF(128) ! overlays CBUF
EQUVALENCE (CBUF,IBUF)

C

C***INITIALIZATIONS:

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

C

C***PROCESSING:

IF (.NOT.(RANGE(_RNUM,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
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

CALL CMN2('CO G '//'GRCHAR//' RED')
GOTO 2000

CALL CMN2('CO G '//'GRCHAR//' GRN')
GOTO 2000

CALL CMN2('CO G '//'GRCHAR//' BLU')
GOTO 2000

2000 CALL DSPGR(_RNUM) ! display the chosen graphics
CALL WRTAR(XCOORD, YCOORD) ! position the cursor for writing

IF (FACTOR .GE. 10) GOTO 3000
ELSE...
"NOTE
LONG = LENGTH + 8
GOTO 4000

C THEN...
3000 CBUF = 'G'//GRCHAR// L1//DIGIT(FACTOR-10)// /*NOTE
LONG = LENGTH + 9
GOTO 4000

C
4000 CALL CMMND(IBUF, LONG)

C Delay for a second or two to relieve timing problem between
C the HP and the COMTAL
C
CALL DELAY((FACTOR/4)+1)
RETURN

C
C**ERROR RETURNS:

C
8001 WRITE(TERM,8003)GRNUM,GRLO,GRHI
8003 FORMAT(' THE GRAPHICS NUMBER','I5',' IS OUT OF RANGE:','214','')
GOTO 8900

C
8103 FORMAT(' THE X COORDINATE','I5',' IS OUT OF RANGE:','214','')
GOTO 8900

C
8203 FORMAT(' THE Y COORDINATE','I5',' IS OUT OF RANGE:','214','')
GOTO 8900

C
8303 FORMAT(' THE SCALE FACTOR','I5',' IS OUT OF RANGE:','214','')
GOTO 8900

C
8403 FORMAT(' THE STRING LENGTH','I5',' IS OUT OF RANGE:','214','')
GOTO 8900

C
8503 FORMAT(' THE COLOR PARAMETER','A1',' IS NOT S, R, G, OR B.')
GOTO 8900

C
8901 WRITE(TERM,8901)
8903 FORMAT(' NOTES FAILS. NO LETTERING PLACED INTO GRAPHICS.')
RETURN

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

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 C SUBROUTINE PAINT(IMAGE, BRUSH, SHADE)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C****PARAMETER DECLARATIONS:
0006 C
0007 C INTEGER IMAGE ! number of the COMTAL image to be painted
0008 C INTEGER BRUSH ! size of the square brush area
0009 C INTEGER SHADE ! the pixel value to be brushed on
0010 C
0011 C****INTRODUCTION:
0012 C
0013 C The subroutine PAINT allows the interactive user to paint
0014 C onto a COMTAL image, using the trackball to guide the brush.
0015 C The user uses the HP keyboard to signal when to paint.
0016 C Each time the HP <CR> is pressed, PAINT paints a BRUSH X BRUSH
0017 C square of pixels with the cursor position in the upper left corner
0018 C of the square. The maximum size for a brush has been set (arbitrarily)
0019 C to 64 pixels square.
0020 C
0021 C****LANGUAGE:
0022 C
0023 C FORTRAN 77, the HP-1000 version for RTE-6/VM.
0024 C
0025 C****LIMITATIONS:
0026 C
0027 C The COMTAL keyboard is inaccessible to the HP. Therefore, we
0028 C must use the HP keyboard even though we use the COMTAL trackball.
0029 C Because of the HP keyboard limitations, the program requires
0030 C a pointillistic painting: one square in the image is darkened
0031 C each time the HP <CR> is pressed.
0032 C
0033 C****SUBPROGRAMS CALLED:
0034 C
0035 C name source load remarks
0036 C --------- ------- ------ ---------------
0037 C DSPBW &DSPBW &DSPBW display a monochrome COMTAL image.
0038 C URIRC &URIRC &URIRC writes the contents of an array to a COMTAL image;
0039 C the values are read into a rectangle in the image.
0040 C RDTAR &RDTAR &RDTAR reads the current COMTAL cursor position.
0041 C WAIT &WAIT &WAIT halts HP processing until HP <CR> entered.
0042 C CMIND &CMIND &CMIND sends a character string to COMTAL, which
0043 C treats the string as a COMTAL keyboard command.
0044 C
C***WRITTEN BY:
C
C The code on which this subprogram is based was written by
C NETTIE D. FAULCON, July, 1983. This subprogram was written by
C KEITH MILLER, July, 1984, with the support of a NASA-ASEE
C summer fellowship.
C
C***REVISION HISTORY:
C
C
C***LOCAL VARIABLES:
C
C INTEGER BOX(64*64) ! the square that acts as a paint brush-
C this rectangle placed into IMAGE whenever
C <CR> entered to locate cursor on COMTAL.
C INTEGER XPOS ! loop index for initializing BOX.
C INTEGER UPLFX, UPLFY ! X and Y coordinates of last COMTAL cursor
C INTEGER ,TERM ! logical unit for terminal I/O.
C INTEGER IMLO, IMHI ! limits on COMTAL image number.
C INTEGER PXLO, PXHI ! limits on COMTAL pixel values.
C INTEGER BRLO, BRHI ! limits on size of brush for painting.
C CHARACTER*1 INCHAR ! character buffer for HP Keyboard input.
C LOGICAL RANGE ! function that determines if 1st argument
C is within the 2nd and 3rd, inclusive.
C INTEGER IBUF(128) ! buffer for sending CMND strings
C CHARACTER*255 CBUF ! character overlay for IBUF
C EQUIVALENCE (IBUF,CBUF)
C
C***INITIALIZATIONS:
C
C DATA IMLO/1/, IMHI/4/
C DATA BRLO/1/, BRHI/128/
C DATA PXLO/0/, PXHI/255/
C DATA TERM/1/
C
C***PROCESSING
C
C IF (.NOT.(RANGE(IMAGE, IMLO, IMHI))) GOTO 8001 ! error return
C IF (.NOT.(RANGE(BRUSH, BRLO, BRHI))) GOTO 8101 ! error return
C IF (.NOT.(RANGE(SHADE, PXLO, PXHI))) GOTO 8201 ! error return
C
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, INCHAR)
1001 FORMAT(1A1)
1000 IF (INCHAR .EQ. ' ') GOTO 2000 ! paint another square and continue.
1001 IF (((INCHAR .EQ. 'S') .OR. (INCHAR .EQ. 's'))) GOTO 9000 ! terminate
1002 ELSE...illegal entry
1003 WRITE(TERM, 1003) INCHAR
1004 1003 FORMAT(' THE CHARACTER ENTERED, ', 1A1, ', IS NOT LEGAL.',
       ' FOR PAINTING. ' , ' PLEASE TRY AGAIN. ')
1000 GOTO 1000

Actual painting takes place here:

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

ERROR RETURNS:

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

**PARAMETER DECLARATIONS:**

```
INTEGER GRAPH ! the number of the COMTAL graphics plane in which the profile is to be displayed.
INTEGER IMAGE ! the number of the COMTAL image which is going to be "profiled."
```

**INTRODUCTION:**

The subroutine PROFIL gives HP access to the COMTAL’s interactive profiling facilities. This subroutine initializes the COMTAL for taking profiles of IMAGE and displaying them in the specified GRAPH plane. Note that this subroutine clears the specified graphics plane.

Control passes to COMTAL for profiling, and then a <CR> on the HP terminal restores control to the HP terminal.

**LANGUAGE:**

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

**LIMITATIONS:**

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

**SUBPROGRAMS CALLED:**

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

**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 argument is within 2nd and 3rd argument, inclusive.
- INTEGER TERM ! logical unit for terminal output

**INITIALIZATIONS:**

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

**PROCESSING**

IF (.NOT.(RANGE(GRAPH,GRLO,GRHI))) GOTO 8881
IF (.NOT.(RANGE(IMAGE,IMLO,IMHI))) GOTO 8881
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.

the following COMTAL command reads: Add Target:

Graphics *G = Profile of image *I.

the following COMTAL command reads: Add Target:

Graphics *G = Profile of image *I.

print out instructions for the COMTAL keyboard interaction

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

CALL WAIT
RETURN

***ERROR RETURNS:

8001 WRITE(TERM, 8003)GRAPH, GRLO, GRHI
8003 FORMAT('The GRAPH parameter, ',J4,' is out of range: ',213)
GOTO 8900

8101 WRITE(TERM, 8103)IMAGE, IMLO, IMHI
8103 FORMAT('The IMAGE parameter, ',J4,' is out of range: ',213)
GOTO 8900

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

PARAMETER DECLARATIONS:

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

INTRODUCTION:

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

LIMITATIONS:

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

WRITTEN BY:

Keith Miller, NASA-Langley ASEE fellow, 1984

REVISION HISTORY:

LOCAL VARIABLES:

INTEGER TERM ! logical unit for terminal output

INITIALIZATIONS:

DATA TERM/1/

PROCESSING:

IF (LOW.GT. HIGH) WRITE(TERM, 1001) LOW, HIGH
1001 FORMAT(' LOW.', IS, ',', 'GT. HIGH.', IS, ' RANGE fails.')
RANGE = ((OBJECT .GE. LOW) .AND. (OBJECT .LE. HIGH))
RETURN
END
SUBROUTINE RDGLN (ONOFFS, GRAPH, LINE)

**PARAMETER DECLARATIONS:**

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:

**INTRODUCTION:**

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

**LANGUAGE:**

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

**LIMITATIONS:**

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

**SUBPROGRAMES CALLED:**

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

**WRITTEN BY:**

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

**REVISION HISTORY:**

**LOCAL VARIABLES:**

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

**INITIALIZATIONS:**

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

**PROCESSING**

```c
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

**ERROR RETURNS**

```c
8001 WRITE(TERM, 0003) GRAPH, GRLO, GRHI
8003 FORMAT(’ GRAPHICS NUMBER:’, 13, ’ OUT OF RANGE:’, 212.’’)
GOTO 8900
8101 WRITE(TERM, 8103) LINE, LNLO, LNHI
8103 FORMAT(’ LINE NUMBER:’, 14, ’ OUT OF RANGE:’, 213, ’’)
GOTO 8900
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:

FORTAN 77, the HP-1000 version for RTE-G 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.

SUBPROGRAMES CALLED:

name source load remarks
--------------- -------------- ---------------------------------------------
RANGE &RANGE XRANGE logical function that determines if its first parameter is within the last two parameters.
BTEST ---- ---- HP FORTRAN 77 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.
**Revision History:**

**Local Variables:**

- **Logical Range**: A function that determines if its first parameter is between (inclusive) its last 2 parameters.
- **Logical Btest**: Is a certain bit on or off.
- **Integer Term**: The logical unit for terminal output.
- **Integer GRLO, GRHI**: The limits on COMTAL monochrome graph numbers.
- **Integer LNLO, LNHI**: The limits on COMTAL graph line numbers.
- **Integer BTLO, BTHI**: The limits on COMTAL graphics values.
- **Integer BITS(32)**: A buffer to read & write a COMTAL graphics line.
- **Integer WORD**: Which word of BITS holds the bit selected by XCOORD.
- **Integer BIT**: Which bit in BITS(WORD) holds the bit selected by XCOORD; bits numbered 0-15, right to left.

**Initializations:**

- **DATA TERM/I/**
- **DATA GRLO/I/, GRHI/4/**
- **DATA LNLO/I/, LNHI/511/**

**Processing:**

- **If (.NOT.(RANGE(GRAPH,GRLO,GRHI))) GOTO 8801** | error return
- **If (.NOT.(RANGE(XCOORD,LNLO,LNHI))) GOTO 8101** | error return
- **If (.NOT.(RANGE(YCOORD,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 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(1,368+1088,BITS,32,(GRAPH-1)*2048+512+YCOORD)**

Find the single bit that has been selected:

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

Value = 0  | bit is clear until proven set.

If (BTEST(BITS(WORD),BIT)) VALUE = 1
RETURN
C
***ERROR RETURNS
C
8001 WRITE(TERM, 8003) GRAPH, GRLO, GRHI
8003 FORMAT(' GRAPH NUMBER: ', I3, ' OUT OF RANGE: ', 212, '.')
GOTO 8900
8008 C
8101 WRITE(TERM, 8103)XCOORD, LNL0, LNNHI
8103 FORMAT(' X COORDINATE: ', I4, ' OUT OF RANGE: ', 214, '.')
GOTO 8900
8108 C
8201 WRITE(TERM, 8203)YCOORD, LNL0, LNNHI
8203 FORMAT(' Y COORDINATE: ', I4, ' OUT OF RANGE: ', 214, '.')
GOTO 8900
8208 C
8900 WRITE(TERM, 8901)
8901 FORMAT(' RDGPT FAILS. NO TRANSFER.')
RETURN
END
&RDIL2 T=00004 IS ON CR00021 USING 00010 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE RDIL2(INTS, IMAGE, LINE)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
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 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 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    X-RANGE   X-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)

**INITIALIZATIONS:**

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

**PROCESSING**

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

Programming note:

- The EXEC call is explained in detail in the HP Programmer's Reference Manual for RTE-6/VM, p.2-19ff. This transfer function for the COMTAL is discussed in the COMTAL User’s Manual, Section 5.2.2.1. In the EXEC call that follows, the HP resident driver, DVR41, is called as follows: the first parameter (I) signifies a read; the second parameter is in two parts: 3GB identifies the resident DVR41 driver, and 100B identifies the line transfer operation of that driver; the third parameter (PIXELS) holds the data to be transferred, and the fourth parameter gives PIXELS' length in words (256); and the final parameter is a COMTAL command code for the transfer CALL EXEC(I, 3GB+100B, PIXELS, 256, (IMAGE-I)*2048 + LINE)

- TRANSFORM the 512 8 bit values into 512 16 bit integers

- DO 1000 INDEX = 1, 512

- CONTINUE

RETURN
C***ERROR RETURNS

C 0091 WRITE(TERM, 8003) IMAGE, IMLO, IMHI
0092 8003 FORMAT(' IMAGE NUMBER.', I3, ' OUT OF RANGE: ', 212., ')
0093 GOTO 8900

C 0094 WRITE(TERM, 8103) LINE, LNLD, LNHI
0095 8103 FORMAT(' LINE NUMBER.', I4, ' OUT OF RANGE: ', 213., ')
0096 GOTO 8900

C 0101 WRITE(TERM, 8901)
0102 8901 FORMAT(' RDIL2 FAILS. NO TRANSFER. ')
0103 RETURN
0104 END
**&RDILN T=00004 IS ON CR00021 USING 00018 BLKS R=0000**

**0001** CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

**0002** SUBROUTINE RDILN(PIXELS, IMAGE, LINE)

**0003** CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

**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 KETTIE 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

92
0045 LOGICAL RANGE  ! function that ascertains if its first parameter
0046 C  ! is between (inclusive) its last 2 parameters
0047 INTEGER TERM  ! the logical unit for terminal output
0048 INTEGER IMLO, IMHI  ! the limits on COMTAL monochrome image numbers
0049 INTEGER LNLO, LNHI  ! the limits on COMTAL image line numbers
0050 C
0051 C***INITIALIZATIONS:
0052 C
0053 DATA TERM/1/
0054 DATA IMLO/1/, IMHI/4/
0055 DATA LNLO/0/, LNHI/511/
0056 C
0057 C***PROCESSING
0058 C
0059 IF (.NOT.(RANGE(IMAGE, IMLO, IMHI))) GOTO 8001  ! error return
0060 IF (.NOT.(RANGE(LINE, LNLO, LNHI))) GOTO 8101  ! error return
0061 C
0062 C Programming note:
0063 C The EXEC call is explained in detail in the
0064 C HP Programmer's Reference Manual for RTE-6/VM.p.2-19ff. This
0065 C transfer function for the COMTAL is discussed in the
0066 C COMTAL User's Manual, Section 5.2.2.1. In the EXEC call
0067 C that follows, the HP resident driver, DVR41, is called as
0068 C follows: the first parameter (I) signifies a read; the
0069 C second parameter is in two parts: 3GB identifies the resident
0070 C DVR41 driver, and 100B identifies the line transfer operation
0071 C of that driver; the third parameter (PIXELS) holds the data to be
0072 C transferred, and the fourth parameter gives PIXELS' length in words
0073 C (256); and the final parameter is a COMTAL command code for the transfer
0074 CALL EXEC(1, 3GB+100B, PIXELS, 256, (IMAGE-1)*2048 + LINE)
0075 RETURN
0076 C
0077 C***ERROR RETURNS
0078 C
0079 8001 WRITE(TERM, 8003) IMAGE, IMLO, IMHI
0080 8003 FORMAT( ' IMAGE NUMBER: ', 13, ' OUT OF RANGE: ', 212, '.')
0081 GOTO 8900
0082 C
0083 8101 WRITE(TERM, 8103) LINE, LNLO, LNHI
0084 8103 FORMAT( ' LINE NUMBER: ', 14, ' OUT OF RANGE: ', 213, '.')
0085 GOTO 8900
0086 C
0087 8900 WRITE(TERM, 8901)
0088 8901 FORMAT(' RDILN FAILS. NO TRANSFER.')
0089 RETURN
0090 END
&RDIPT T=00004 IS ON CR00021 USING 00000 BLKS R=0000

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:

<table>
<thead>
<tr>
<th>name</th>
<th>source</th>
<th>load</th>
<th>remarks</th>
</tr>
</thead>
</table>

RANGE &RANGE X RANGE logical function that determines if its
first parameter is within the last two parameters.
ICHAR HP FORTRAN77 intrinsic function: converts a byte
into its integer code.

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

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

INTEGER TERM ! the logical unit for terminal output

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

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

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

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

EQUIVALENCE (PIXELS,BYTES)

**INITIALIZATIONS:

DATA TERM/I/

DATA IMLO/I/, IMHI/4/

DATA LNLO/0/, LNHI/511/

**PROCESSING

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

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

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

Programming note:

The EXEC call below is to the DVR41 driver, and
is identical to the call made in RDILN.
See the RDILN documentation for the details.

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

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

convert the byte that is to be read:

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

which converts a character into its
integer code.

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

RETURN

**ERROR RETURNS
0091  8081 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  8123 FORMAT(' Y COORDINATE.', I4, ' OUT OF RANGE:', 214, '.')
0097        GOTO 8900
0098  C
0099  8201 WRITE(TERM, 8203)XCOORD, LNLO, LNHI
0100  8223 FORMAT(' Y COORDINATE.', I4, ' OUT OF RANGE:', 214, '.')
0101        GOTO 8900
0102  C
0103  8960 WRITE(TERM, 8901)
0104  8981 FORMAT(' RDIPT FAILS. NO TRANSFER.')
0105        RETURN
0106        END
&RDIRC T=C0004 IS ON CR00021 USING 00004 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***PARAMETER DECLARATIONS:
0006 C
0007 C INTEGER XDIM, YDIM          ! dimensions of the output array, OUTARA
0008 C INTEGER OUTARA(XDIM, YDIM) ! the array to be filled
0009 C INTEGER IMAGE               ! the number of the COMTAL image from
0010 C INTEGER UPLFX, UPLFY       ! which OUTARA is to be filled
0011 C INTEGER IMAGE              ! the image coordinates of the upper left
0012 C INTEGER IMAGE              ! corner of the rectangle of pixels that is
0013 C INTEGER IMAGE              ! to be read into OUTARA.
0014 C
0015 C***INTRODUCTION:
0016 C
0017 C The subroutine Read Image Rectangle transfers pixel values from a
designated section of an image to an integer array. Note that although
pixel values are generally stored with 1 byte/pixel, RDIRC places each
numeric value into a 2 byte integer in OUTARA.
XDIM, YDIM, IMAGE, UPLFX, and UPLFY are all checked for possible out
of range errors before any transfer is attempted.

0024 C
0025 C***LANGUAGE:
0026 C
0027 C FORTRAN 77, the HP-1000 version for RTE-6/VAX.

0029 C***LIMITATIONS:
0030 C
0031 C RDIRC does a great deal of error checking before initiating the
transfer. If the programmer can verify that all calls to RDIRC
will be legal, these checks could be commented out to improve
machine efficiency. Also, the call to the subroutine RDILN could
be replaced inline by the RDILN code (which is short).

0035 C
0037 C***SUBPROGRAMS CALLED:
0038 C
0039 C name source load remarks
0040 C ------ ------ ------ -------------------------------
0041 C RDILN &RDILN &RDILN transfers a line of bytes from a COMTAL image
0042 C RANGE &RANGE &RANGE logical function that determines of its first
parameter is within the last two parameters
0043 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 KEITH MILLER, July, 1984.

**REVISION HISTORY:**

This code was written by KEITH MILLER, July, 1984.

**LOCAL VARIABLES:**

- INTEGER IMROW, IMCOL ! a location in the image
- INTEGER ARAROW, ARACOL ! a location in OUTARA
- INTEGER ICHAR ! intrinsic byte to integer conversion
- LOGICAL RANGE ! function that ascertains if the first parameter is within the last two parameters.
- INTEGER IMLO, IMHI ! limits on COMTAL image numbers
- INTEGER LNLO, LNHI ! limits on COMTAL pixel coordinates
- INTEGER LNCNT ! LNHI-LNLO+1, # of pixels in an image line
- INTEGER TERM ! logical unit for terminal output
- CHARACTER*1 CBUF(512) ! buffer to hold COMTAL horizontal line
- EQUIVALENCE (IBUF, CBUF)

**INITIALIZATIONS:**

- DATA TERM/1/
- DATA IMLO/1/, IMHI/4/
- DATA LNLO/0/, LNHI/511/, LNCNT/512/

**PROCESSING**

- IF (.NOT.(RANGE(XDIM, 1, LNCNT)))GO TO 8101 ! error return
- IF (.NOT.(RANGE(YDIM, 1, LNCNT)))GO TO 8201 ! error return
- IF (.NOT.(RANGE(UPLFX, LNLO, LNHI )))GO TO 8301 ! error return
- IF (.NOT.(RANGE(UPLFY, LNLO, LNHI )))GO TO 8401 ! error return
- IF (.NOT.(RANGE(UPLFX+XDIM-1, LNLO, LNHI )))GO TO 8501 ! error return
- IF (.NOT.(RANGE(UPLFY+YDIM-1, LNLO, LNHI )))GO TO 8601 ! error return
- IMROW = UPLFY
DO 2000 ARAROW = 1, YDIM
   CALL RDILH(IBUF, IMAGE, IMROW)
   IMROW = IMROW + 1 ! increment for next pass thru 2000 loop
C
C   IMCOL = UPLFX + 1 ! the "+1" is necessary because COMTAL image
C   coordinates range from 0 to 511 and the
C   FORTRAN array indices range from 1 to 512.
DO 1000 ARACOL = 1, XDIM
   OUTARA(ARAROW, ARACOL) = ICHAR(CBUF(IMCOL))
   IMCOL = IMCOL + 1
1000 CONTINUE
2000 CONTINUE
RETURN ! successful termination
C
C   ERROR RETURNS:
C
C   8661 WRITE(TERM, 8003) IMAGE, IMLO, IMHI
C   8003 FORMAT(' IMAGE NUMBER', I3, ' OUT OF RANGE: ', 214,'.')
   GOTO 8900
C
C   8103 FORMAT(' X DIMENSION', I4,' OUT OF RANGE: ', 215,'.')
   GOTO 8900
C
C   8203 FORMAT(' Y DIMENSION', I4,' OUT OF RANGE: ', 215,'.')
   GOTO 8900
C
C   8303 FORMAT(' X COORDINATE FOR CORNER', I4,' OUT OF RANGE: ', 215,'.')
   GOTO 8900
C
C   8403 FORMAT(' Y COORDINATE FOR CORNER', I4,' OUT OF RANGE: ', 215,'.')
   GOTO 8900
C
   GOTO 8900
C
C   8603 FORMAT(' Y COORDINATE FOR THE CORNER AND THE Y DIMENSION ', 215, '.)
0137    -1  ' OF THE ARRAY', ', ' OVERFLOW IMAGE BOUNDARIES.'.
0138    -2  ' X COORDINATE =', 14, ' X DIMENSION =', 14, '.
0139    -3  ' IMAGE COORDINATE LIMITS ARE ', 215, '.').
0140    GO TO 8900
0141 C
0142  8900 WRITE(TERM, 8903)
0143  8903 FORMAT(' RDIRC FAILS. NO TRANSFER TAKES PLACE.' )
0144 RETURN
0145 END
**SUBROUTINE RDLUT (TABLE, LUTNUM)**

**PARAMETER DECLARATIONS:**

```fortran
INTEGER TABLE(256) ! the values of the specified COMTAL look-up table are read into this array. The 0 entry goes into TABLE(1), .... the 255 entry goes into TABLE(256).
INTEGER LUTNUM ! the number of the COMTAL look-up table (called "function memory" in the COMTAL literature).
```

**INTRODUCTION:**

The subroutine Read Look-Up Table (LUT) reads the COMTAL mapping from the integers 0-255 into the integer array TABLE. This LUT can be used for grey level enhancements in the COMTAL. A similar subroutine called RDPoufl is used to read from a pseudocolor look-up table. This routine is only used for grey scale look-up tables.

**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:**

<table>
<thead>
<tr>
<th>name</th>
<th>source</th>
<th>load</th>
<th>remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>RANGE</td>
<td>&amp;RANGE</td>
<td>RANGE</td>
<td>logical function which determines if its 1st parameter is within its 2nd and 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:

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

INITIALIZATIONS:

DATA TERM/1/

DATA LUTLO/1/, LUTHI/4/

PROCESSING

IF (.NOT.(RANGE(LUTNUM, LUTLO, LUTHI))) GOTO 6001 ! error return

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).

The fourth parameter gives the length of TABLE in words.

The fifth parameter is a COMTAL code that is described bit by bit in the User's Manual. In short, bit 15 signifies write to COMTAL, bit 14 designates function memory instead of pseudocolor, bit 12 signifies standard replacement, and bits 8 & 9 identify the function memory to be used. (Bits are numbered 15 high, 0 low).

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

RETURN

ERROR RETURN

WRITE(TERM, 8003) LUTNUM, LUTLO, LUTHI

9003 FORMAT( ' THE FUNCTION MEMORY NUMBER,.I4,. IS OUT OF RANGE:' )

WRITE(TERM, 8901)

1 214, 1, 36B, 200B, TABLE, 256, ((LUTNUM-1)*256)
0091 8901 FORMAT(' RDLUT FAILS. NO TRANSFER FROM COMTAL. ')
0092 END
**SUBROUTINE RDPSU(TABLE)**

**PARAMETER DECLARATIONS:**

**INTRODUCTION:**

The subroutine reads the Pseudocolor table, which comprises 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:**

**WRITTEN BY:**

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

**SUMMARY:**

The subroutine reads the Pseudocolor table, which comprises 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:**

**WRITTEN BY:**

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

**SUMMARY:**

The subroutine reads the Pseudocolor table, which comprises the pseudocolor table. Note that the values are placed into TABLE in the order RED, GREEN, and BLUE.
**initializations:**

```c
NONE
```

**processing**

```c

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 read 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 8&9 which identify the color to be transferred.

Note that we make three separate calls to EXEC. Each call fills a different section of TABLE with a different color of the COMTAL's pseudocolor table.

```c
CALL EXEC( 1, 36B+300B, TABLE(1), 256, 1*256 ) ! red
CALL EXEC( 1, 36B+300B, TABLE(257), 256, 0*256 ) ! green
CALL EXEC( 1, 36B+300B, TABLE(513), 256, 2*256 ) ! blue
RETURN
END
```
SUBROUTINE RDTAB(TABLE, CNUMB, IMOGR)

CPARAMETER DECLARATIONS:

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

CINTRODUCTION:

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

CPOWER LANGUAGE:

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

CLIMITATIONS:

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

C SUBPROGRAMS CALLED:

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

CWRITTEN BY:

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

```
LOGICAL RANGE  ! function that ascertains if its first parameter
   ! is between (inclusive) its last 2 parameters
LOGICAL BTEST   ! is a certain bit on or off.
INTEGRAL TERM   ! the logical unit for terminal output
INTEGRAL DGLO, DGHI ! the limits on single Digit image/graphics numbers
INTEGRAL BTLO, BTHI ! the limits on COMTAL graphics values
XCOOR; bits numbered 0-15, right to left.
```

**INITIALIZATIONS:**

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

**PROCESSING**

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

**Programming note:**

The EXEC call below is to the DVR41 driver.

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

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

The third argument, "TABLE", is the buffer that will hold the IGT
("Image/Graphics Table") information after the EXEC is completed;
the fourth argument, "16", gives the length of TABLE in words.

The final parameter is a code to the COMTAL which identifies the
mode of the transfer ("?x4086"), mode 7; selects either image or
graphics ("IMOGRx128"); and gives the number of the image/graphics
memory plane ("CNMB-1").

```
call exec(1,368x5005,table,16, 7x4086 + IMOGRx128 + CNMB-1)
```

**ERROR RETURNS**

```
8001 write(term, 8003) cnmb, dglo, dghi
8003 format( 'COMTAL NUMBER:"", 13, ' OUT OF RANGE:"", 212,"" )
8088 goto 8900
8101 write(term, 8103) imogr, btlo, bthi
```
0091 0103 FORMAT(' IMAGE/GRAPHICS ARGUMENT MUST BE 0 OR 1. NOT '.14.'.).
0092  GOTO 0900
0093  C
0094  0900 WRITE(TERM, 0901)
0095  0901 FORMAT(' RDTAB FAILS. NO TRANSFER.'.)
0096  RETURN
0097  END
&RDTAR T=00004 IS ON CR00021 USING 00012 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE RDTAR(XCOORD, YCOORD)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C**PARAMETER DECLARATIONS:
0006 C
0007 INTEGER XCOORD ! X coordinate of the COMTAL target location.
0008 INTEGER YCOORD ! Y coordinate of the COMTAL target location.
0009 C
0010 C**INTRODUCTION:
0011 C
0012 C This subroutine Reads the TARGET (cursor) location from the COMTAL.
0013 C
0014 C**LANGUAGE:
0015 C
0016 C FORTRAN 77, the HP-1000 version for RTE-G/VM.
0017 C
0018 C**LIMITATIONS: NONE
0019 C
0020 C**SUBPROGRAMS CALLED: NONE
0021 C
0022 C**WRITTEN BY:
0023 C
0024 C The code on which this subprogram is based was written by
0025 C NETTIE D. FAULCON, July, 1983. This modification is by
0026 C KEITH MILLER, June, 1984.
0027 C
0028 C**REVISION HISTORY:
0029 C
0030 C
0031 C**LOCAL VARIABLES:
0032 C
0033 INTEGER IBUF(2) ! the buffer to hold the COMTAL data transfer
0034 C
0035 C**INITIALIZATIONS: NONE
0036 C
0037 C**PROCESSING
0038 C
0039 C Programming note:
0040 C The EXEC call is explained in detail in the
0041 C HP Programmer's Reference Manual for RTE-G/VM, p.2-19ff. This
0042 C transfer function for the COMTAL is discussed in the
0043 C COMTAL User's Manual, Section 5.2.4. In the EXEC call
0044 C that follows, the HP resident driver, DVR41, is called as
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,
(00000B).

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 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

0002 SUBROUTINE SETV(IMAGE)

0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

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

0033 C name source load remarks

0034 C

0035 C CMIND &CMIND %CMIND sends character strings to the COMTAL where

0036 C they are much like COMTAL keyboard commands.

0037 C

0038 C DIGIT &DIGIT %DIGIT character*1 function that returns '0'-'9'

0039 C according to integer input 0-9.

0040 C

0041 C RANGE &RANGE %RANGE logical function which determines if its 1st

0042 C argument is within the 2nd and 3rd, inclusive.

0043 C

0044 C

0045 C****WRITTEN BY:

0046 C

0047 C

0048 C

0049 C The code on which this subprogram is based was written by

0050 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:**

LOGICAL RANGE ! function which determines if its 1st argument
is within the 2nd & 3rd arguments, inclusive.

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

INTEGER IBUF(128) ! holds COMTAL keyboard command strings for
use with CMMND.

CHARACTER*255 CBUF ! overlays IBUF.

EQUIVALENCE (IBUF,CBUF)

INTEGER VDLO,VDHI ! the limits on video image number: one digit,
but not a # reserved for image memory.

INTEGER TERM ! logical unit number for terminal input

**INITIALIZATIONS:**

DATA VDLO/5/, VDHI/9/

DATA TERM/1/

**PROCESSING:**

IF (.NOT.(RANGE(IMAGE,VDLO,VDHI))) GOTO 8801 ! error return

the following COMTAL command is expanded to:

Set Video image #I: Display Image #I

where #I is the digit equal to IMAGE.

This command will hang up the COMTAL if the camera is already
set to IMAGE. However, we can't do a RELEASE just to be sure.

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

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

CALL CMMND(IBUF,12)

RETURN

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

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE SPLIT(BYTE1, BYTE2, INBYTE)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
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 FORTRAN 77, the HP-1000 version for RTE-6/VM.
0023 C
0024 C**LIMITATIONS:
0025 C
0026 C   None.
0027 C
0028 C**SUBPROGRAMS CALLED:
0029 C
0030 C   None.
0031 C
0032 C**WRITTEN BY:
0033 C
0034 C   The code on which this subprogram is based was written by
0035 C   NETTIE D. FAULCON, July, 1983. This subprogram was written by
0036 C   KEITH MILLER, July, 1984, with the support of a NASA-ASEE
0037 C   summer fellowship.
0038 C
0039 C**REVISION HISTORY:
0040 C
0041 C
0042 C**LOCAL VARIABLES:
0043 C
0044 INTEGER IHOLD ! an INTEGER interpretation of bits
0045 CHARACTER CHOLD(2) ! a CHARACTER interpretation of bits
0046 C
0047 INTEGER IMERGE ! an INTEGER interpretation of bits
0048 CHARACTER CSPLIT(2) ! a CHARACTER interpretation of bits
0049 C
0050 EQUIVALENCE (IHOLD, CHOLD), (IMERGE, CSPLIT)
0051 C
0052 IMERGE = INBYTE
0053 IHOLD = 0 ! zero out high order bits
0054 CHOLD(2) = CSPLIT(1)
0055 BYTE1 = IHOLD
0056 CHOLD(2) = CSPLIT(2)
0057 BYTE2 = IHOLD
0058 C
0059 RETURN
0060 END
SUBROUTINE SPRED(IMAGE)

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

C***INTRODUCTION:

The subroutine SPREAD takes as input and output a COMTAL image.

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

C***LANGUAGE:

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

C***LIMITATIONS:

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

C***SUBPROGRAMS CALLED:

name source load remarks
----- ----- ----- ---------------------------------------------------------
RDIL2 &RDIL2 &RDIL2 reads a horizontal line of pixels from the
COMTAL, and places the values in INTEGER array.
WRIL2 &WRIL2 &WRIL2 writes a horizontal line of pixels from an
integer array to a COMTAL image.
HILO &HILO &HILO 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
The code on which this subprogram is based was written by NETTE D. FAULKNER 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**

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

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

WRITE(TERM, 3501) HIGH, LOW

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

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

FACTOR = 255.0 / FLOAT(HIGH-LOW)

DO 4000 ROW = 1,512

CALL RDIL2(IBUT, IMAGE, ROW)

DO 4000 COL = 1,512
0091       IBUF(COL) = IFIX(FLOAT(IBUF(COL)-LOW) * FACTOR)
0092 3000    CONTINUE
0093       CALL URIL2(IMAGE,ROW,IBUF)
0094 4000    CONTINUE
0095 C
0096 C
0097 9999    RETURN
0098       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 *CMMND *CMMND Sends a command to the COMTAL as if the
C command were sent to the keyboard
C
C RANGE &RANGE &RANGE logical function that determines if the 1st
C parameter is within the range of the 2nd & 3rd.
C
C DIGIT &DIGIT &DIGIT character*1 function that returns '0'-'9'
C according to integer input 0-9.
C
C
C**WRITTEN BY:
C
C
C KEITH MILLER, July, 1984, with the support of a NASA-ASEE
C summer fellowship.
C
C
C**REVISION HISTORY:
C
C
C
C**LOCAL VARIABLES:
C
C
C INTEGER IBUF(128) ! a buffer for passing commands to COMTAL
C CHARACTER*255 CBUF ! character overlay for IBUF
C EQUIVALENCE (IBUF, CBUF)
C
C LOGICAL RANGE ! function that determines if 1st parameter
C ! is within 2nd and 3rd parameter
C
C INTEGER IMLO, IMHI ! limits on COMTAL image numbers
C INTEGER TERM
C CHARACTER*1 DIGIT ! returns a single digit character '0'
C ! to '9' for integer input 0-9.
C
C
C**INITIALIZATIONS:
C
C
C DATA IMLO/1/, IMHI/4/
C DATA TERM/1/
C
C
C**PROCESSING
C
C
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
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 'S' separates COMTAL commands.

CBUF =

1 'UN 1 8 $'// UNassign Image B ! just in case B is already assigned.
2 'AS T B '/DlGit(C)/' '/DlGit(A)/' '/DlGit(B)/' $'//' ASSign Truecolor image B red #C green #A blue #B
3 'D I 8 $'//' Display Image B
4 'SE COM G - B $'//' SET COMbine <Green - Blue> ! +128 offset and no scaling is the default.
5 'A COM $'//' Add COMbine
6 'I '/DlGit(C)/' D R $'//' Image #C = Displayed Image Red ! Red is arbitrary, since difference of images is monochome
7 'D I '/DlGit(C)/' $'//' Display Image #C
8 'SU COM ' // Subtract COMbine.
9 CALL CMMD(IBUF, 74)

RETURN

C***ERROR RETURNS
C
0124 8001 WRITE(TERM, 8003) A
0127 8003 FORMAT(' THE 2ND IMAGE PARAMETER, ' , I3, ', ', IS OUT OF RANGE. ')
0128 GOTO 8900
0129 C
0130 8101 WRITE(TERM, 8103) B
0131 8103 FORMAT(' THE 3RD IMAGE PARAMETER, ' , I3, ', ', IS OUT OF RANGE. ')
0132 GOTO 8900
0133 C
0134 8201 WRITE(TERM, 8203) C
0135 8203 FORMAT(' THE 1ST IMAGE PARAMETER, ' , I3, ', ', IS OUT OF RANGE. ')
0136 GOTO 8900
0137 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  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
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
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</th>
<th>load</th>
<th>remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMDND</td>
<td>&amp;CMDND</td>
<td>&amp;CMDND</td>
<td>Sends a command to the COMTAL as if the command were sent to the keyboard</td>
</tr>
<tr>
<td>RANGE</td>
<td>&amp;RANGE</td>
<td>&amp;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>&amp;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*255 | CBUF | character overlay for IBUF |
| EQUIVALENC | (IBUF,CBUF) |

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

**INITIALIZATIONS:**

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

**PROCESSING**

IF (.NOT. (RANGE(A,IMLO,IMHI))) GOTO 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...
0091 C digit number associated with the parameter X.
0092 C In this notation, letters in caps were entered into CBUF, and
0093 C lower case letters are the full commands filled in by the COMTAL
0094 C NOTE: this code assumes that the digit 0 is NOT a legal value for
0095 C the parameters A, B, and C.
0096 C The "$" separates COMTAL commands.
0097 C
0098 C
0099 C CBUF =
0100 C 1 'UN I 8 $'/
0101 C 2 'AS T 6 '/DIGIT(C)//' '/DIGIT(A)//' '/DIGIT(B)//' "$'/
0102 C 3 'D I 8 $'/
0103 C 4 'SE COM G - B + 8 $'/
0104 C 5 'A COM $'/
0105 C 6 'I '/DIGIT(C)//' D R "$'/
0106 C 7 'D I '/DIGIT(C)//' "$'/
0107 C 8 'SU COM '
0108 C 9 CALL COMMAND IBUF, 77)
0109 C RETURN
0110 C C***ERROR RETURNS
0111 C 0112 1 8801 WRITE (TERM, 8803) A
0113 C 0114 8803 FORMAT (' THE 2ND IMAGE PARAMETER.', I3, ',', ' IS OUT OF RANGE. ')
0115 C 0116 GOTO 8900
0117 C 0118 C 0119 8101 WRITE (TERM, 8103) B
0120 C 0121 8103 FORMAT (' THE 3RD IMAGE PARAMETER.', I3, ',', ' IS OUT OF RANGE. ')
0122 C 0123 GOTO 8900
0124 C 0125 8201 WRITE (TERM, 8203) C
0126 C 0127 8203 FORMAT (' THE 1ST IMAGE PARAMETER.', I3, ',', ' IS OUT OF RANGE. ')
0128 C 0129 GOTO 8900
0130 C 0131 8900 WRITE (TERM, 8901)
0132 C 0133 8901 FORMAT (' SUBIM RETURNS WITHOUT FURTHER PROCESSING. ')
0134 C 0135 RETURN
0136 C END
&THRES T=E9904 IS ON CR00021 USING 00010 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE THRESH(OUTIMG, INIMG, THRESH)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***PARAMETER DECLARATIONS:
0006 C
0007 INTEGER OUTIMG the COMTAL image number for the thresholded image.
0008 INTEGER INIMG the COMTAL image & for the image to be thresholded.
0009 INTEGER THRESH the threshold pixel value: < threshold -> 0,
0010 >= threshold -> 255.
0011 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.
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
**INITIALIZATIONS:**

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

**PROCESSING:**

- **IF** (.NOT.(RANGE(OUTIMG, IMLO, IMHI))) GOTO 8001 | 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

- **ELSE**...

- **IBUF(PXCNT) = 255**
- **GOTO** 1000

- **THEN**...

- **IBUF(PXCNT) = 0**
0091 1063  CONTINUE
0092   CALL WRIL2(OUTIMG, LINE, IBUF)
0093 2063 CONTINUE
0094   RETURN
0095 C
0096 C*ERROR RETURNS:
0097 C
0098 8001 WRITE(TERM, 8003) OUTIMG, IMLO, IMHI
0099 8003 FORMAT(‘ THE OUTPUT IMAGE NUMBER,’.15,’ IS OUT OF RANGE:’,214)
0100    GOTO 8900
0101 C
0102 8101 WRITE(TERM, 8103) INIMG, IMLO, IMHI
0103 8103 FORMAT(‘ THE INPUT IMAGE NUMBER,’.15,’ IS OUT OF RANGE:’,214)
0104    GOTO 8900
0105 C
0106 8201 WRITE(TERM, 8203) THRESH, PXLO, PXHI
0107 8203 FORMAT(‘ THE THRESHOLD VALUE,’.15,’ IS OUT OF RANGE:’,214)
0108    GOTO 8900
0109 C
0110 8900 WRITE(TERM, 8901)
0111 8901 FORMAT(‘ THRESH FAILS. OUTIMG NOT CHANGED.’)
0112    RETURN
0113    END
&TSTII T=00004 IS ON CR00021 USING 00004 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE TSTII(WHICH)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***PARAMETER DECLARATIONS:
0006 C
0007 INTEGER WHICH ! designates the COMTAL image in which the test
0008 C image is to be generated
0009 C
0010 C***INTRODUCTION:
0011 C
0012 C The subroutine TstII Image I generates a distinctive pattern in one
0013 C of the COMTAL image memories. The pattern is generated with ascending
0014 C pixel values as you move to the right and down in the image. However,
0015 C when 255 is reached in either the x or y directions, the pixel values
0016 C restart at 0. This gives a slash across the screen appearance at the
0017 C discontinuity, and a gradual change in grey scale elsewhere.
0018 C
0019 C***LANGUAGE:
0020 C
0021 C FORTRAN 77, the HP-1000 version for RTE-6/VM.
0022 C
0023 C***LIMITATIONS:
0024 C
0025 C The test pattern developed here is always the same. An enhancement
0026 C might be to vary the repetition of the pixel values (here, each pixel
0027 C is identical to one of its horizontal neighbors and one of its vertical
0028 C neighbors).
0029 C
0030 C***SUBPROGRAMS CALLED:
0031 C
0032 C name  source  load  remarks
0033 C------  ------- ------- -------------------------------
0034 C WRILN &WRILN %WRILN given an integer buffer of at least 512 bytes.
0035 C WRILN writes a horizontal line of pixels to a
0036 C designated COMTAL image.
0037 C
0038 C***WRITTEN BY:
0039 C
0040 C The code on which this subprogram is based was written by
0041 C NETTIE D. FAULCON, July, 1983. This subprogram was written by
0042 C KEITH MILLER, July, 1984, with the support of a NASA-ASEE
0043 C summer fellowship.
C**REVISION HISTORY:
C
C**LOCAL VARIABLES:
C
INTEGER BIGBUF(512) ! 1024 bytes of pixel values
INTEGER I ! pointer into BIGBUF
INTEGER ROW ! designates a COMTAL image row
C
C**INITIALIZATIONS:
C
DO 1000 I = 8,255
BIGBUF(I+1) = I * 257 ! I*256 numbers the high byte
BIGBUF(I+257) = BIGBUF(I+1) ! facilitates the wraparound effect
1000 CONTINUE
C
C**PROCESSING
C
DO 2000 ROW = 8,511
CALL WRILN(WHICH, ROW, BIGBUF((ROW/2)+1))
2000 CONTINUE
C
RETURN
C
END
SUBROUTINE TV2C4

PARAMETER DECLARATIONS:

None.

**INTRODUCTION:**

The subroutine "TV to Comtal image transfer #4" digitizes four copies of the current TV image, and averages them together into a single image stored in COMTAL image #1. COMTAL images 1, 2, and 3 are used for storage. Image 5 must be set to the video camera image before TV2C4 is called.

**LANGUAGE:**

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

**LIMITATIONS:**

Images 1, 2, 3, and 4 are changed by this routine.

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.

**SUBPROGRAMS CALLED:**

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

TV2CM &TV2CM XTV2CM digitizes the camera associated with COMTAL image 5 into the 1st argument, an image memory.

ADD12 &ADD12 XADD12 adds two images, pixel by pixel, and divides the sums by last argument, to produce a new 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:
LOGICAL RANGE  
INTEGER TVIMAG

**INITIALIZATIONS:
DATA TVIMAG/5/  
DATA TERM/1/

**PROCESSING:
CALL DSPVD
CALL TV2CM(1)
CALL TV2CM(2)
CALL ADDI2(3,1,2,2)  
CALL TV2CM(1)
CALL TV2CM(2)
CALL ADDI2(4,1,2,2)  
CALL ADDI2(1,3,4,2)  
CALL DSPBW(1)
RETURN
END
&TV2CM T=0004 IS ON CR00021 USING 00004 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE TV2CM(CMIMAG)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***PARAMETER DECLARATIONS:
0006 C
0007 INTEGER CMIMAG ! "COMTAL IMAGE" number where the digitized image
0008 C is to be stored.
0009 C
0010 C***INTRODUCTION:
0011 C
0012 C The subroutine TV2CM digitizes
0013 C a "snapshot" of the current TV image into the COMTAL image
0014 C number CMIMAG. Previous to the TV2CM call, image 5 must be set
0015 C to the video camera.
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 The video camera must be associated with TVIMAGE before TV2CM is
0024 C called. TVIMAGE is 5 as an arbitrary convention in this lab.
0025 C
0026 C CMIMAG must identify a COMTAL image memory (IMLO to IMHI).
0027 C
0028 C***SUBPROGRAMS CALLED:
0029 C
0030 C name source load remarks
0031 C
0032 C CMIND &CMIND &CMIND sends character strings to COMTAL which
0033 C interprets them as COMTAL keyboard commands.
0034 C DSPBU &DSPBU &DSPBU displays the indicated monochrome COMTAL image.
0035 C RANGE &RANGE &RANGE logical function that determines if 1st
0036 C argument is within the 2nd & 3rd, inclusive.
0037 C DIGIT &DIGIT &DIGIT character*1 function that returns '0'..'9'
0038 C according to integer input 6..9.
0039 C
0040 C***WRITTEN BY:
0041 C
0042 C The code on which this subprogram is based was written by
0043 C Nettie D. Faulcon, July, 1983. This subprogram was written by
0044 C Keith Miller, July, 1984, with the support of a NASA-ASEE
C summer fellowship.

C

C***REVISION HISTORY:

C

C

C***LOCAL VARIABLES:

C

C

INTEGER 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.

INTEGER TVIMAG ! arbitraily set to 5 in this lab.

C***INITIALIZATIONS:

DATA IMLO/1/, IMHI/4/

DATA TVIMAG/5/ ! arbitrary convention for our lab.

DATA TERM/1/

C***PROCESSING:

IF (.NOT. (RANGE(CMIMAG,IMLO,IMHI))) GOTO 8001 ! 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(TVIMAG)/' SI '/DIGIT(CMIMAG)/' D R '

CALL CMMND(IBUF, 15)

CALL DSPBLJ(CMIMAG)

RETURN

C***ERROR RETURNS:

WRITE(TERM,0003) CMIMAG, IMLO, IMHI

8003 FORMAT(* THE COMTAL IMAGE NUMBER,'.I4.', IS OUT OF RANGE:'I24)
0091    GOTO 8900
0092    C
0093    C
0094    8900 WRITE(TERM,8901)
0095    8901 FORMAT(\"TV2CM FAILS. NO DIGITIZING TAKES PLACE.\")
0096    C
0097    RETURN
0098    END
&WAIT T=E0004 IS ON CR00021 USING 03002 BS KS R=0003

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE WAIT
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***PARAMETERS: NONE
0006 C
0007 C
0008 C***INTRODUCTION:
0009 C
0010 C WAIT pauses until <CR> is pressed on the HP keyboard.
0011 C
0012 C***LANGUAGE:
0013 C
0014 C FORTRAN 77, the HP-1000 version for RTE-6/VM.
0015 C
0016 C***SUBPROGRAMS CALLED: NONE
0017 C
0018 C
0019 C***WRITTEN BY:
0020 C
0021 C KEITH MILLER, NASA-Langley ASEE fellow, 1984
0022 C
0023 C***REVISION HISTORY: NONE
0024 C
0025 C
0026 C***LOCAL VARIABLES:
0027 C
0028 INTEGER TERM ! logical unit of the terminal
0029 INTEGER IDUMMY ! facilitates the read that forces a pause
0030 C
0031 C***INITIALIZATIONS:
0032 C
0033 DATA TERM/1/
0034 C
0035 C***PROCESSING:
0036 C
0037 WRITE(TERM, 1001)
0038 1001 FORMAT(' PUSH <CR> TO CONTINUE. ')  
0039 READ(TERM, 2001) IDUMMY
0040 2001 FORMAT(I2)
0041 RETURN
0042 END
SUBROUTINE WIPGR(GRNUM)

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

"WIPe Graphics" is a subroutine that allows the caller
to send an "un-displo_" command to the COMTAL from an HP program.
The call can turn off a graphics plane, number 1, 2, 3, or 4.

It is OK to call WIPGR repeatedly without an intervening DSPGR.
The extra calls have no effect, but they don’t hang up the COMTAL.

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

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

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.

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.

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.
INTEGER TERM ! within a certain range.
INTEGER IDUMMY ! the logical unit number for the terminal.
INTEGER GRLO, GRHI ! fills the place of an unused EXEC parameter.
CHARACTER*1 DIGIT ! limits on a graphics plane number.

CHARACTER*1 DIGIT ! function that returns '0'-'9' for input
INTEGER IDUMMY ! integers 0-9.

***INITIALIZATIONS:
DATA TERM/1/
DATA GRLO/1/, GRHI/4/

***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 CMN2("SUB GR "//DIGIT(GRNUM))
RETURN

***ERROR RETURN:
WRITE(TERM, 8003) GRNUM, GRLO, GRHI
8003 FORMAT( ' THE GRAPHICS NUMBER', '13', ' OUT OF RANGE:', '214)

WRITE(TERM, 8901)
8901 FORMAT( ' DSPGR fails. No action taken on command.' )
RETURN
END
SUBROUTINE WRGLN(GRAPH, LINE, ONOFFS)

INTEGER GRAPH ! CMTAL image graphics line to be written to.
INTEGER LINE ! which horizontal line to be written to.
INTEGER ONOFFS(32) ! 16 bits per integer, 512 bits in a line.

C***INTRODUCTION:

This subroutine WRITES a Graphics Line to the CMTAL. The line of
bits is coded as 1 for on and 0 for off. Each bit in ONOFFS must be
set by the caller of WRGLN.

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
METHIE D. FAULCON, July, 1983. This modification is by
KEITH MILLER, June, 1984.

C***REVISION HISTORY:

C***LOCAL VARIABLES:

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

C***INITIALIZATIONS:

DATA TERM/1/
DATA GRLO/1/, GRHI/4/
DATA LNLO/0/, 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 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 1000 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(2,36B+1000,ONOFFS,32,(GRAPH-1)*2048+LINE+512)
RETURN

C***ERROR RETURNS

WRITE(TERM, 6003) GRAPH, GRLO, GRHI
FORMAT('GRAPHICS NUMBER: ', 13, 'OUT OF RANGE: ', 212, '.')
GOTO 8900

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

WRITE(TERM, 8901)
FORMAT('WRGLN FAILS. NO TRANSFER.')
RETURN

END
SUBROUTINE WRGPT(GRAPH, XCOOR, YCOOR, VALUE)

PARAMETER DECLARATIONS:

INTEGER GRAPH ! COMTAL graph number to be written to.
INTEGER XCOOR, YCOOR ! 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, XCOOR, YCOOR, 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.

ISSET ------ ------ HP FORTRAN77 intrinsic function that takes an
integer argument and returns that integer with
one bit set, according to the second argument.

IBCLR ------ ------ HP FORTRAN77 intrinsic function that takes an
integer argument and returns that integer with
one bit cleared, according to the second argument.

WRITTEN BY:

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

- **LOGICAL RANGE**: Function that ascertains if its first parameter is between (inclusive) its last 2 parameters.
- **INTEGER TERM**: The logical unit for terminal output.
- **INTEGER GRLO, GRHI**: The limits on COMTAL monochrome graph numbers.
- **INTEGER LNLO, LNHI**: The limits on COMTAL graph line numbers.
- **INTEGER BTLO, BTHI**: The limits on COMTAL graphics values.
- **INTEGER BITS(32)**: A buffer to read & write a COMTAL graphics line.
- **INTEGER WORD**: Which word of BITS holds the bit selected by XCOORD.
- **INTEGER BIT**: Which bit in BITS(WORD) holds the bit selected by XCOORD; bits numbered 0-15, right to left.

**INITIALIZATIONS:**

- **DATA**: Term/1/, GRLO/1/, GRHI/4/.
- **DATA**: LNLO/8/, LNHI/511/.
- **DATA**: BTLO/0/, BTHI/1/.

**PROCESSING**

- **IF**: If (.NOT.(RANGE(GRAPH,GRLO,GRHI))) GOTO 8001 ! error return.
- **IF**: If (.NOT.(RANGE(XCOORD,LNLO,LNHI))) GOTO 8101 ! error return.
- **IF**: If (.NOT.(RANGE(YCOORD,LNLO,LNHI))) GOTO 8201 ! error return.
- **IF**: 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 RDLN. The second EXEC call is identical to the one in URILN. See the documentation for those subroutines for details on these calls.

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

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

Change the single bit that has been selected:

- **WORD** = (XCOORD/16) + 1
- **BIT** = (16*WORD) - XCOORD - 1
IF (VALUE .EQ. 0) GO TO 1000
ELSE...

BITS(WORD) = IBSET(BITS(WORD),BIT)
GOTO 2000

THEN...

BITS(WORD) = IBCLR(BITS(WORD),BIT)
GOTO 2000

Write the graph line with one changed graphics to COMTAL

CALL EXEC(2,36B+100B,BITS,32,(GRAPH-1)*2048 + 512 + YCOORD)
RETURN

***ERROR RETURNS

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

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

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

WRITE(TERM, 8301) VALUE, BTLO, BTHI
8303 FORMAT(' BIT VALUE:, I4, ' OUT OF RANGE:', 214, '.')
GOTO 8900

WRITE(TERM, 8990)
8901 FORMAT(' WRGPT FAILS. NO TRANSFER."
RETURN
END
&URIL2 T=00004 is on CR00021 using 00018 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE URIL2(IMAGE, LINE, INTS)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***PARAMETER DECLARATIONS:
0006 C
0007 INTEGER IMAGE ! COMTAL image number to be written to
0008 INTEGER LINE ! which horizontal line to be written to;
0009 INTEGER INTS(512) ! 512 values, one integer per pixel, to
0010 C
0011 C
0012 C
0013 C***INTRODUCTION:
0014 C
0015 C This subroutine, WRITE Image Line #2, writes a line of pixels to the
0016 C COMTAL. The input array INTS has a two byte integer for each pixel,
0017 C but the COMTAL only uses the lower order byte of each integer. Therefore,
0018 C URIL2 strips off the upper byte before sending the pixels to the COMTAL.
0019 C URIL2 is very similar to URILN, which writes out a line of bytes.
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 INTS array must be completely "full". The IMAGE designated
0028 C must be a monochrome image. The LINE parameter must be between
0029 C 0 and 511. If IMAGE or LINE is out of range, an error message is printed
0030 C and no transfer takes place.
0031 C
0032 C***SUBPROGRAMS CALLED:
0033 C
0034 C name source load remarks
0035 ------ ------ ------ -----------------------------------------------
0036 C RANGE CRANGE XRANGE logical function that determines if its
0037 C first parameter is within the last two parameters.
0038 C CHAR ------ ------ intrinsic HP FORTRAN77 function that strips off
0039 C the upper byte of an integer and returns the lower
0040 C byte as a character.
0041 C
0042 C***WRITTEN BY:
0043 C
0044 C The code on which this subprogram is based was written by

144
INTERGER PIXELS(256) ! holds lower order bytes of INTS values
CHARACTER*1 CPIX (512) ! overlays PIXELS
EQUIVALENCE (PIXELS,CPIX)
LOGICAL RANGE ! function that ascertains if its first parameter
is between (inclusive) its last 2 parameters
INTEGER TERM ! the logical unit for terminal output
INTEGER IMLO, IMHI ! the limits on COMTAL monochrome image numbers
INTEGER LNLO, LNHI ! the limits on COMTAL image line numbers
INTEGER INDEX ! indexes into arrays in a loop
DATA TERM/1/
DATA IMLO/1/, IMHI/4/
DATA LNLO/0/, LNHI/511/
DATA RNAGE(IMAGE, IMLO, IMHI)) GOTO 8001 ! error return
IF (.NOT.(RANGE(LINE, LNLO, LNHI))) GOTO 8101 ! error return
strip off the upper bytes of the pixel values in INTS
DO 1000 INDEX=1,512
CPIX(INDEX) = CHAR(INTS(INDEX)) ! CPIX overlays PIXELS
1000 CONTINUE
Programming note:
The EXEC call is explained in detail in the
HP Programmer's Reference Manual for RTE-6/M,p.2-19ff. This
transfer function for the COMTAL is discussed in the
COMTAL User's Manual, Section 5.2.2.1. In the EXEC call
that follows, the HP resident driver called DVR41 is called as
follows: the first parameter (2) signifies a write; the
second parameter is in two parts: 368 identifies the resident
DVR41 driver, and 1000 identifies the line transfer operation
transferred, and the fourth parameter gives PIXELS' length in words
(256); and the final parameter is a COMTAL command code for the transfer.

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

C***ERROR RETURNS
C

WRITE(TERM, 8003) IMAGE, IMLO, IMHI
FORMAT(' IMAGE NUMBER: ', 13, ' OUT OF RANGE: ', 212, '.')
GOTO 8900

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

WRITE(TERM, 8901)
FORMAT(' WRIL2 FAILS. NO TRANSFER.')
RETURN
END
&WRILN T=00004 IS ON CR00021 USING 00004 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE WRILN(IMAGE, LINE, PIXELS)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***PARAMETER DECLARATIONS:
0006 C
0007 INTEGER IMAGE ! COMTAL image number 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 PIXELS(256)! 512 bytes (pixels) to be transferred
0011 C
0012 C***INTRODUCTION:
0013 C
0014 C This subroutine WRites an Image Line to the COMTAL. The line of
0015 C pixels is made up of 8 bit (0-255) grey scale intensities. The
0016 C PIXELS array is assumed to be completely full.
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 The PIXELS array must be completely "full". The IMAGE designated
0025 C must be a monochrome image. The LINE parameter must be between
0026 C 0 and 511. If IMAGE or LINE is out of range, an error message is printed
0027 C and no transfer takes place.
0028 C
0029 C***SUBPROGRAMS CALLED:
0030 C
0031 C name source load remarks
0032 C ------ ------ ------ -------------------------------
0033 C RANGE &RANGE &RANGE logical function that determines if its
0034 C first parameter is within the last two parameters.
0035 C
0036 C***WRITTEN BY:
0037 C
0038 C The code on which this subprogram is based was written by
0039 C NETTIE D. FAULCON, July, 1983. This modification is by
0040 C KEITH MILLER, June, 1984.
0041 C
0042 C***REVISION HISTORY:
0043 C
0044 C

147
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 1 error return
IF (.NOT.(RANGE(LINE, LNLO, LNHI))) GOTO 8181 ! error return

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

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

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

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

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

PARAMETER DECLARATIONS:

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) grey scale intensity. If the image number, coordinates, or value are out of range, an error message is printed and no transfer takes place.

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

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

SUBPROGRAMS CALLED:
RANGE &RANGE X$RANGE logical function that determines if its first parameter is within the last two parameters.

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

REVISION HISTORY:
LOCAL VARIABLES:

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

INITIALIZATIONS:

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

PROCESSING

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

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

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

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

Change the single byte that needs changing:

I HOLD = VALUE ! transfers the pixel VALUE to CHAR$(2).
BYTES(XCOORD+1) = CHAR$(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 6001 WRITE(TERM, 8003) IMAGE, IMLO, IMHI
0099 6003 FORMAT(‘IMAGE NUMBER’, I3, ‘OUT OF RANGE:’, 212, ‘.’)
0100 GOTO 8900
0101 C
0102 8101 WRITE(TERM, 8103)XCOORD, LNLO, LNHI
0103 8103 FORMAT(‘X COORDINATE’, I4, ‘OUT OF RANGE:’, 214, ‘.’)
0104 GOTO 8900
0105 C
0106 8201 WRITE(TERM, 8203)YCOORD, LNLO, LNHI
0107 8203 FORMAT(‘Y COORDINATE’, I4, ‘OUT OF RANGE:’, 214, ‘.’)
0108 GOTO 8900
0109 C
0110 8301 WRITE(TERM, 8303) VALUE, PXLO, PXHI
0111 8303 FORMAT(‘PIXEL VALUE’, I4, ‘OUT OF RANGE:’, 214, ‘.’)
0112 GOTO 8900
0113 C
0114 8900 WRITE(TERM, 8901)
0115 8901 FORMAT(‘WRITE FAILS. NO TRANSFER.’)
0116 RETURN
0117 END
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.

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

```
0030 SUBROUTINE WRIRC(IMAGE, UPLFX, UPLFY, INPARA, XDIM, YDIM)
0031 INTEGER IMAGE
0032 INTEGER UPLFX, UPLFY
0033 INTEGER XDIM, YDIM
0034 INTEGER INPARA(XDIM, YDIM)

0037 SUBROUTINE RDIL2 &RDIL2 XRDL2 transfers a line of bytes from a COMTAL image into an HP array of integers.
0038 SUBROUTINE URIL2 &URIL2 XRRL2 transfers a buffer of integers to a COMTAL image as a line of bytes (1 byte/integer).

0039 name source load remarks
0040 ------- ------ ------ --------------------------------------------------
0041 RDIL2 &RDIL2 XRDL2 transfers a line of bytes from a COMTAL image into an HP array of integers.
0042 URIL2 &URIL2 XRRL2 transfers a buffer of integers to a COMTAL image as a line of bytes (1 byte/integer).

0043 C
0044 C
```
C. RANGE &RANGE xRANGE 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 NETTIE D. FAULCON, July, 1983. This code was written by KEITH MILLER, July, 1984.

C***REVISION HISTORY:

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

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 that ascertains if the first parameter is within the last two parameters.
C INTEGER IMLO, IMHI | limits on COMTAL image numbers
C INTEGER LNO, LNH1 | limits on COMTAL pixel coordinates
C INTEGER DIMLIM, LNIHI | limits 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 LNH1/0/, LNH1/511/
C DATA DIMLIM/64/

C***PROCESSING

C IF (.NOT. (RANGE (IMAGE, IMLO, IMHI )))GOTO 8001 | error return
C IF (.NOT. (RANGE (DIM, IMLO, IMHI )))GOTO 9101 | error return
C IF (.NOT. (RANGE (DIM, IMLO, IMHI )))GOTO 9201 | error return
C IF (.NOT. (RANGE (DIM, IMLO, IMHI )))GOTO 9301 | error return
C IF (.NOT. (RANGE (DIM, IMLO, IMHI )))GOTO 9401 | error return
C IF (.NOT. (RANGE (DIM, IMLO, IMHI )))GOTO 9501 | error return
C IF (.NOT. (RANGE (DIM, IMLO, IMHI )))GOTO 9601 | error return

154
we get to this point if the transfer is to take place

IMROW = UPLFY

DO 2000 ARAROW = 1, YDIM
   CALL RDIL2(IBUF, IMAGE, IMROW)
   IMCOL = UPLFX + 1 ! the next line initializes the column pointer;
   IMCOL = IMCOL + 1 ! coordination range from 0 to 511 and the
   ! FORTRAN array indices range from 1 to 512.
DO 1000 ARACOL = 1, XDIM
   IBUF(IMCOL) = INPARA(ARAROW, ARACOL)
   IMCOL = IMCOL + 1 ! increment for next 1000 loop pass
1000 CONTINUE
   CALL WRIL2(IMAGE, IMROW, IBUF)
   IMROW = IMROW + 1 ! increment for next 2000 loop pass
2000 CONTINUE
RETURN ! successful termination

**ERROR RETURNS:**

6001 WRITE(TERM, 8003) IMAGE, IMLO, IMHI
6003 FORMAT(’ IMAGE NUMBER,’, I3, ’ OUT OF RANGE:’, 214,’’)
GOTO 8900

6103 FORMAT(’ X DIMENSION,’, I4, ’ OUT OF RANGE:’, 215,’’)
GOTO 8900

6203 FORMAT(’ Y DIMENSION,’, I4, ’ OUT OF RANGE:’, 215,’’)
GOTO 8900

6303 FORMAT(’ X COORDINATE FOR CORNER,’, I4,’ OUT OF RANGE:’, 215,’’)
GOTO 8900

6403 FORMAT(’ Y COORDINATE FOR CORNER,’, I4,’ OUT OF RANGE:’, 215,’’)
GOTO 8900

8003 FORMAT(’ X COORDINATE FOR THE CORNER AND THE X DIMENSION ’,
1 OF THE ARRAY’/’ OVERFLOW IMAGE BOUNDARIES’/’.
2 ’ X COORDINATE =’, I4, ’X DIMENSION =’, I4,’/
3 ’ IMAGE COORDINATE LIMITS ARE ’ ,215,’’)
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 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
REVISION HISTORY:

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

INITIALIZATIONS:

DATA TERM/1/

DATA LUTLO/1/, LUTHI/4/

PROCESSING

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

Programming notes:

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

The first parameter to EXEC identifies the EXEC command as a write command. The second parameter identifies the resident HP driver (36B) and gives the code (208) 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, 36B+208B, TABLE, 256, ((LUTNUM-1)*256) )

ERROR RETURN

WRITE(TERM, 8003) LUTNUM, LUTLO, LUTHI

FORMAT( ' THE FUNCTION MEMORY NUMBER', I4, ', IS OUT OF RANGE:',

1, 214, ', ')

WRITE(TERM, 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 PSedoucolor 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. FAULKON, July, 1983. This subprogram was written by KEITH MILLER, July, 1984, with the support of a NASA-ASEE summer fellowship.

REVISION HISTORY:

LOCAL VARIABLES:

NONE.
**C**

**C**\*\*\*INITIALIZATIONS:

**C**

**C** NONE.

**C**

**C**\*\*\*PROCESSING

**C**

**C** Programming notes:

**C** The EXEC command parameters are discussed in the HP RTE-6/VM

**C** Programmer’s Reference Manual, 2-19ff. The COMTAL parameters

**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 write 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**

**C** Note that we make three separate calls to EXEC. Each call fills a

**C** different section of the pseudocolor table from the TABLE array.

**C**

**C** CALL EXEC( 2, 36B+300B, TABLE(1), 256, 1*256 ) ! red

**C** CALL EXEC( 2, 36B+300B, TABLE(257), 256, 0*256 ) ! green

**C** CALL EXEC( 2, 36B+300B, TABLE(513), 256, 2*256 ) ! blue

**C**

**C** RETURN

**C**

**C** END
SUBROUTINE URTAR(XCOORD, YCOORD)

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

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

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

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

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

logical function that determines if its first parameter is within the last two parameters.

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

logical unit for terminal output
0045 INTEGER LHLO, LNHI  ! the limits on COMTAL image line numbers
0046 INTEGER IBUF(2)  ! buffer for passing coordinates to COMTAL
0047 INTEGER IDUMMY  ! an ignored EXEC call parameter

0048 C
0049 C****INITIALIZATIONS:
0050 C
0051 DATA TERM/*
0052 DATA LNLO/*, LNHI/511/
0053 C
0054 C****PROCESSING
0055 C
0056 IF (.NOT.(RANGE(XCOORD, LNLO, LNHI))) GOTO 8001  ! error return
0057 IF (.NOT.(RANGE(YCOORD, LNLO, LNHI))) GOTO 8101  ! error return
0058 C
0059 C Programming note:
0060 C The EXEC call is explained in detail in the
0061 C HP Programmer's Reference Manual for RTE-6/VM, p.2-19ff. This
0062 C transfer function for the COMTAL is discussed in the
0063 C COMTAL User's Manual, Section 5.2.4. In the EXEC call
0064 C that follows, the HP resident driver called DVR41 is called as
0065 C follows: the first parameter (2) signifies a write; the
0066 C second parameter is in two parts: 36B identifies the resident
0067 C DVR41 driver, and 400B identifies the target transfer operation
0068 C of that driver; the third parameter (IBUF) contains the two coordinates
0069 C to be transferred, and the fourth parameter gives the length of the
0070 C buffer in words. The last parameter is ignored.
0071 C
0072 IBUF(1) = XCOORD
0073 IBUF(2) = YCOORD
0074 CALL EXEC(2, 36B+400B, IBUF, 2, IDUMMY)
0075 RETURN
0076 C
0077 C***ERROR RETURNS
0078 C
0079 8001 WRITE(TERM, 8003) XCOORD, LNLO, LNHI
0080 8003 8023 FORMAT(' XCOORD:', 15, ', OUT OF RANGE:', 214, '.')
0081 GOTO 8900
0082 C
0083 8101 WRITE(TERM, 8103) YCOORD, LNLO, LNHI
0084 8103 8123 FORMAT(' YCOORD:', 15, ', OUT OF RANGE:', 214, '.')
0085 GOTO 8900
0086 C
0087 8900 WRITE(TERM, 8901)
0088 8901 FORMAT(' WRITAR FAILS. NO TRANSFER.')
0089 RETURN
0090 END
&TADD2.T=80004 IS ON CR00021 USING 00002 BLKS R=0000

0001 PROGRAM TADD2
0002 C
0003 C TEST THE ADDPROGRAM ADD12 (ADD IMAGES)
0004 C
0005 C KEITH MILLER 7/17/84
0006 C
0007 C LOAD MODULES: XTADD2, XADD12, XMAND, XRANGE, XDIGIT
0008 C
0009 CALL ADD12(0,1,2,1) ! SHOULD GIVE OUT OF RANGE ERROR.
0010 CALL ADD12(1,0,2,1) ! *
0011 CALL ADD12(1,2,0,1) ! *
0012 C
0013 CALL ADD12(3,1,2,1) ! SHOULD WORK.
0014 C
0015 STOP
0016 END
&TADDI T=60004 IS ON CR00021 USING 00002 BLKS R=0000

0001 PROGRAM TADDI
0002 C
0003 C TEST THE ADD PROGRAM ADDIM (ADD IMAGES)
0004 C
0005 C KEITH MILLER 7/17/84
0006 C
0007 C LOAD MODULES: XTADDI, XADDIM, XCMND, XRANGE, XDIGIT
0008 C
0009 CALL ADDIM(0,1,2,1) ! SHOULD GIVE OUT OF RANGE ERROR
0010 CALL ADDIM(1,0,2,1) ! "
0011 CALL ADDIM(1,2,0,1) ! "
0012 C
0013 CALL ADDIM(3,1,2,1)
0014 C
0015 STOP
0016 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, %CMMD, %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 CMMD(IBUF, 10)
0018       CALL CLRGR(5)
0019       C
0020       STOP
0021       END
&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, CMD, 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 CMD(IBUF, 13)
0018    CALL CLRIM(5)
0019    C
0020    STOP
0021    END
0022
0023
0024
0025
0026
&TCMM2 T=00004 IS ON CR00021 USING 00003 BLKS R=0000

0001 PROGRAM TCMM2
0002 C
0003 C TEST CMIND AGAIN
0004 C
0005 C KEITH MILLER, 7/12/84
0006 C
0007 C LOAD MODULES _TCMM2, _CMIND, _WAIT
0008 C
0009 INTEGER INNUM, I, IBUF(128)
0010 CHARACTER*255 CBUF
0011 EQUIVALENCE (IBUF, CBUF)
0012 C
0013 DO 1000 I = 0, 256
0014 WRITE(1,500) I
0015 500 FORMAT('TESTING CHARACTER #', I4)
0016 CALL WAIT
0017 CBUF = 'G 2 PRO 1'
0018 CALL CMIND(IBUF,9)
0019 CBUF = CHAR(I)
0020 CALL CMIND(IBUF,1)
0021 1000 CONTINUE
0022 STOP
0023 END
PROGRAM TCMMN

Keith Miller, 6/16/84

Testing SUBROUTINE CMMND

LOAD MODULES REQUIRED: %TCMMN, %CMMND

INTEGER IBUF(128)

WRITE(1,1)
1 FORMAT(' TO EXIT THIS TEST, ENTER AN EMPTY STRING')

WRITE(1,10)
10 FORMAT('ENTER ASCII STRING')

READ(1,20) IBUF

N = ITLOG() ! ITLOG gives the number of characters typed in

IF (N .EQ. 0) GOTO 999

CALL CMMND(IBUF,N)

GO TO 5

999 END
PROGRAM TCNT

C TEST THE SUBROUTINE COUNT

C KEITH MILLER 7/11/84

C LOAD MODULES: TCNT, COUNT, RANGE, RDL2

INTEGER I4BUF(256)

CALL COUNT(I4BUF, 1)

DO 1000 INDEX = 0, 31

WRITE(1,999) (I4BUF((INDEX*8)+J), J=1,8)

999 FORMAT(8I9)

1000 CONTINUE

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 + 605
STRING = '***' // CHOLD(2) // '***'
WRITE(1,1000) CHOLD(2),STRING
1000 FORMAT( 'LETTER=', A1, '. STRING=', A5, '.')
STOP
END
&TCOPY T=00004 IS ON CR0021 USING 00001 BLKS R=0000

0001 PROGRAM TCOPY
0002 C TEST ICOPY
0003 C LOAD MODULES REQUIRED: XTCOPY, XICOPY, XCMND, XRANGE, XDIGIT
0005 C CALL ICOPY(1,2)
0009 CALL ICOPY(5,1) ! SHOULD GIVE OUT OF RANGE ERROR
0010 CALL ICOPY(2,-3) ! SHOULD GIVE OUT OF RANGE ERROR
0011 STOP
0012 END
PROGRAM TDIGI

TEST DIGIT

KEITH MILLER, 7/2/84

LOAD MODULES: %TDIGI, %DIGIT, %RANGE

INTEGER INT

CHARACTER*1 HCHAR

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

CONTINUE

STOP
END
&TDSP T=03004 IS ON CR00021 USING 00081 BLKS R=0000

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

TEST THE SUBROUTINE HISTO

KEITH MILLER, 7/31/84

LOAD MODULES: %THIST, %HISTO, %DIGIT, %CMNN2, %RANGE

INTEGER TABLE(256)

INTEGER I          ! implicit do loop index

CALL HISTO(1)

STOP

END
PROGRAM THORM

TEST THE PROCEDURE NORML.

KEITH MILLER JULY 6, 1983

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

CALL NORML(2)

STOP

END
0001 PROGRAM TNOTE
0002 C
0003 C TEST THE SUBROUTINES NOTES AND NOTE2
0004 C
0005 C KEITH MILLER, 7/26/84
0006 C
0007 C LOAD MODULES: XTNOTE, XNOTES, XNOTE2, XDSPGR, XKWTAR,
0008 C XCMMD2, XCMND, XDIGIT, XRANGE, XDELAY
0009 C
0010 C CHARACTER*255 CBUF
0011 INTEGER IBUF(128)
0012 EQUIVALENCE (CBUF, IBUF)
0013 C
0014 CALL NOTE2(1,100,100,'B',3,'1.BLUE')
0015 CALL NOTE2(1,100,200,'R',3,'2.RED')
0016 CALL NOTE2(1,100,300,'G',3,'3.GREEN')
0017 CALL NOTE2(1,100,400,'S',3,'4.SAME')
0018 C
0019 CBUF(1:11) = 'USING NOTES'
0020 CALL NOTES(1,10,10,'R',1,CBUF,11)
0021 CALL NOTES(5,10,10,'R',1,CBUF,11)
0022 CALL NOTES(1,512,10,'R',1,CBUF,11)
0023 CALL NOTES(1,-1,10,'R',1,CBUF,11)
0024 CALL NOTES(1,10,512,'R',1,CBUF,11)
0025 CALL NOTES(1,10,-1,'R',1,CBUF,11)
0026 CALL NOTES(1,10,10,'R',1,CBUF,11)
0027 CALL NOTES(1,10,10,'R',1,CBUF,11)
0028 CALL NOTES(1,10,10,'R',0,CBUF,11)
0029 CALL NOTES(1,10,10,'R',17,CBUF,11)
0030 C
0031 CALL NOTE2(2,120,240,'B',3,'USING NOTE2')
0032 CALL NOTE2(0,120,240,'B',3,'USING NOTE2')
0033 CALL NOTE2(5,120,240,'B',3,'USING NOTE2')
0034 CALL NOTE2(2,512,240,'B',3,'USING NOTE2')
0035 CALL NOTE2(2,1,240,'B',3,'USING NOTE2')
0036 CALL NOTE2(2,120,512,'B',3,'USING NOTE2')
0037 CALL NOTE2(2,120,-1,'B',3,'USING NOTE2')
0038 CALL NOTE2(2,120,240,'B',3,'USING NOTE2')
0039 CALL NOTE2(2,120,240,'B',0,'USING NOTE2')
0040 CALL NOTE2(2,120,240,'B',17,'USING NOTE2')
0041 CALL DSPGR(1)
0042 CALL DSPGR(2)
0043 STOP
0044 END
&TPNT  T=CG004  IS ON CR0021 USING 00001 BLKS R=00003

0001 PROGRAM TPNT
0002 C
0003 C TEST THE SUBROUTINE PAINT
0004 C
0005 C KEITH MILLER, 7/16/84
0006 C
0007 C LOAD MODULES: _TPNT, _PAINT, _RANGE, _URIRC, _RDTAR,
0008 C   _RDIL2, _URIL2, _CMMND, _DSFBW
0009 C
0010 CALL PAINT(1,35, 200)
0011 STOP
0012 END
&TPROF T=23004 IS ON CR00021 USING 00002 BLKS R=0000

0001 PROGRAM TPROF
0002 C
0003 C TEST PROF SUBROUTINE
0004 C
0005 C KEITH MILLER, 7/12/84
0006 C
0007 C LOAD MODULES: %TPROF, %PROFL, %RANGE, %CMDND, %WAIT,
0008 C %DIGIT
0009 C
0010 CALL PROFL(1,1)
0011 WRITE(1,1001)
0012 1001 FORMAT(' SUCCESSFULLY RETURNED TO CALLER OF PROFILER')
0013 STOP
0014 END
PROGRAM TRANG

TESTS THE RANGE FUNCTION

LOAD MODULES REQUIRED: XTRANG, XRANGE

LOGICAL RANGE, ANSWER

WRITE(1,1)

1 FORMAT(' SHOULD BE T F ERROR-F')

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

STOP

END
&TRDTA T=00004 IS ON CR00021 USING 02002 BLKS R=0000

0001 PROGRAM TRDTA
0002 C TEST THE SUBROUTINE RDTAB
0003 C KEITH MILLER, 7/19/84
0004 C
0005 C LOAD MODULES: %TRDTA, %RDTAB, %RANGE
0006 C
0007 C INTEGER TABLE(16), INDEX, WHICH
0008 C
0009 C
0010 C
0011 DO 2000 WHICH = 1, 4
0012 CALL RDTAB(TABLE, WHICH, 0)
0013 WRITE(1,999) WHICH, TABLE(1), TABLE(2), TABLE(3), TABLE(4)
0014 999 FORMAT(12, ', ', 409)
0015 2000 CONTINUE
0016 C
0017 STOP
0018 END
PROGRAM TSETV

TEST THE SETV SUBROUTINE

KEITH MILLER, 7/17/84

LOAD MODULES: TSETV, SETV, RANGE, DIGIT, CMD, DSPBW

CALL SETV(2) !SHOULD BE AN ERROR
CALL SETV(11) !SHOULD BE AN ERROR
CALL SETV(5) !SHOULD WORK

STOP

END
PROGRAM TSPRD

TESTS THE PROGRAM TSPRED.

KEITH MILLER, JULY 5, 1984

LOAD MODULES: TSPRD, XSPRED, RDIL2, WRIL2, RANGE

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

WRITE(1, 1001)

C1001 FORMAT(' GIVE THREE SHADES, USING THE FORMAT 314:')

READ (1, 1003) I1, I2, I3

C1003 FORMAT(314)

DO 2000 INDEX=1, 170

IBUF(INDEX) = I1
IBUF(INDEX+170) = I2
IBUF(INDEX+340) = I3

CONTINUE

CALL WRIL2(I1, INDEX, IBUF)

CALL SPRED(1)

STOP

END
PROGRAM TSUBI

TESTS SUBROUTINES SUBIM AND SUBI2

KEITH MILLER 7/23/64

LOAD MODULES: %TSUBI, %SUBIM, %SUBI2, %RANGE, %DIGIT, %WAIT, %COMMAND

CALL SUBIM(1,3,2)!

CALL WAIT

CALL SUBI2(1,3,2)!

STOP

END
&TTHRS T=60004 IS ON CR00021 USING 00003 BLKS 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: X'TTHRS. X'THRSH, X'RANGE, X'RDIL2, X'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
&TTSTI T=0004 IS ON CR00021 USING 00001 BLKS R=0000

0001 PROGRAM TTSTI
0002 C TESTS THE TEST IMAGES
0003 C LOAD MODULES: XTTSTI, XTSTII, XURILN, XRANGE
0005 C CALL TSTII(1)
0008 STOP
0009 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: _TTV2C, _TV2C4, _TV2CM, _CMDND, _DSPBW,
0008 C _RANGE, _DIGIT, _WAIT, _ADD12, _DSPVD
0009 C
0010 WRITE(1,1001)
0011 1001 FORMAT(' MAKE SURE TV CAMERA IS SET TO IMAGE 5.')
0012 CALL WAIT
0013 CALL TV2C4
0014 STOP
0015 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
0017
&TWIPE T=20004 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 WIPER(3)
0017      CALL WAIT
0018      CALL WIPGR(2)
0019      CALL WAIT
0020      CALL WIPGR(1)
0021      CALL WAIT
0022      CALL WIPGR(1)
0023      STOP
0024      END
&TXFDS J=00004 IS ON CR00021 USING 00001 BLKS R=0000

0001 PROGRAM TXFDS
0002 C
0003 C TEST PROGRAM FOR BUFD, CLFDS
0004 C
0005 C KEITH MILLER, 7/12/84
0006 C
0007 C LOAD MODULES: $TXFDS, $BUFD, $CLFDS, $RANGE, $WRILN, $CMDND, $DIGIT
0008 C
0009 INTEGER FLNAME(3)
0010 CHARACTER*6 CNAME
0011 EQUIVALENCE (FLNAME,CNAME)
0012 C
0013 CNAME = 'CFXRAY'
0014 CALL BUFD(1, FLNAME)
0015 C
0016 CNAME = 'CFMND'
0017 CALL CLFDS(1,3,8, FLNAME)
0018 STOP
0019 END
&TXGLN T=6G004 IS ON CR00021 USING 00002 BLKS R=2030

0001 PROGRAM TXGLN
0002 C
0003 C TEST WRGLN AND RDGLN
0004 C
0005 C LOAD MODULES REQUIRED: %TXGLN,%RDGLN,%URGLN,%RANGE
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
0010 C ! note that 16 ONOFF bits fit in one INTEGER
0011 DO 1000 LINE = 1, 512
0012 CALL RDGLN(IBUF, 2, LINE)
0013 C
0014 CALL WRGLN(1, LINE, IBUF)
0015 2000 CONTINUE
0016 END
0017
0018
&TXGPT T=60004 IS ON CR00021 USING 00002 BLKS R=0000

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 CALL WAIT
0017 C
0018 C DO 2000 INDEX = 1, 20
0019 C CALL WRGPT(I, INDEX, INDEX, 0)
0020 C CALL WRGPT(I, INDEX, 21-INDEX, 0)
0021 C2000 CONTINUE
0022 C
0023 DO 3000 INDEX = 1, 5
0024 C CALL WAIT
0025 CALL RDTAR(I, Y)
0026 CALL RDGPT(IVALUE, 1, I, Y)
0027 WRITE(1, 2001) I, Y, IVALUE
0028 2001 FORMAT(' AT POINT ', 214, ' GRAPHICS VALUE= ', I2, '.')
0029 3000 CONTINUE
0030 C
0031 C STOP
0032 C END
PROGRAM TXILN

TEST WRILN AND RDILN

LOAD MODULES REQUIRED: %TXILN.%RDILN.%WRILN.%RANGE

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

! note that 2 pixels fit in one INTEGER

DO 2000 LINE = 0, 511
   CALL RDILN(IBUF, 1, LINE)
   CALL WRILN(2, LINE, IBUF)

2000 CONTINUE
END
&TXIPT T=60004 IS ON CR00021 USING 00002 BLKS R=0000

0001       PROGRAM TXIPT
0002 C     TEST RDIPT AND WR IPT
0003 C     KEITH MILLER, 6/21/84
0004 C
0005 C     LOAD MODULES: %TXIPT, %RDIPT, %WR IPT, %RANGE, %WRIT
0006 C
0007 C     INTEGER XCOORD, YCOORD, VALUE
0008 C
0009 C     CALL RDIPT(VALUE, 1, 200, 100)
0010 C     WRITE(1,1001) VALUE
0011 1001 FORMAT( ' VALUE = ', I4 )
0012 C     CALL WR IPT(1,200,100,005)
0013 C     CALL RDIPT(VALUE, 1, 200, 100)
0014 C     WRITE(1,1001) VALUE
0015 C
0016 C     STOP
0017 C
0018 C
0019 C
0020 C
0021 C
0022 C
0023
PROGRAM TXIRC

TEST RDIRC (READ IMAGE RECTANGLE) & WRIRC (WRITE IMAGE RECTANGLE)

KEITH MILLER, JULY 3, 1984

LOAD MODULES: %TXIRC, %RDIRC, %WRIRC, %RANGE, %RDLN, %WRILN

INTEGER BUFFER(10,10)
INTEGER ROW, COL

CALL RDIRC(BUFFER, 10, 10, 1, 0, 0)
CALL WRIRC(BUFFER, 10, 10)
STOP
END
&TXLUT T=60004 IS ON CR00021 USING 00002 BLKS R=0000

0001 PROGRAM TXLUT
0002 C TEST WRLUT AND RDLUT
0003 C KEITH MILLER, 7/10/84
0004 C
0005 C LOAD MODULES: %TXLUT, %WRLUT, %RDLUT, %RANGE
0006 C
0007 C INTEGER TABLE(256), TABLE2(256)
0008 C INTEGER INDEX
0009 C
0010 DO 1000 INDEX = 1,256
0011 TABLE(INDEX) = 256-INDEX
0012 CONTINUE
0013 C CALL WRLUT(1,TABLE)
0014 C CALL RDLUT(TABLE2,1)
0015 DO 2000 INDEX = 1,256
0016 WRITE (1,1999) TABLE2(INDEX)
0017 CONTINUE
0018 C
0019 STOP
0020 C
0021 C
0022 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(1,1999)

FORMAT(' THE REDS: ')

DO 2000 INDEX=1.256

WRITE(1,1999) TABLE(INDEX)

1999 FORMAT(14)

2000 CONTINUE

CALL WAIT

WRITE(1,2999)

FORMAT(' THE GREENS: ')

DO 4000 INDEX=257.512

WRITE(1,3999) TABLE(INDEX)

3999 FORMAT(14)

4000 CONTINUE

CALL WAIT

WRITE(1,4999)

FORMAT(' THE BLUES: ')

DO 6000 INDEX=513.768

WRITE(1,5999) TABLE(INDEX)

5999 FORMAT(14)

6000 CONTINUE

DO 1000 INDEX = 1.256

TABLE(INDEX) = 256-INDEX

TABLE(INDEX+256) = INDEX

TABLE(INDEX+512) = 122

1000 CONTINUE

C
0045   CALL WRPSU(TABLE)
0046   STOP
0047   END
PROGRAM TXTAR
TEST RDTAR AND WRTAR
KEITH MILLER, 6/21/84
LOAD MODULES: %TXTAR, %RDTAR, %WRTAR, %RANGE
INTEGER XCOOR, YCOOR ! CURSOR COORDINATES
INTEGER INDEX ! LOOP INDEX
CALL URTAR(12,34) ! INITIAL POSITION OF TARGET
DO 1000 INDEX = 1, 20
CALL RDTAR(XCOOR, YCOOR)
WRITE(1,500)XCOOR, YCOOR
500 FORMAT(215)
CALL WRTAR(INDEX, INDEX)
1000 CONTINUE
CALL WRTAR(513,1) ! SHOULD BE AN ERROR ON X
CALL WRTAR(0, 1) ! SHOULD BE AN ERROR ON X
CALL WRTAR(512,-1) ! SHOULD BE AN ERROR ON Y
STOP
END
**HP DRIVER, DVR41, SOURCE CODE**

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

TO INVOKE THIS DRIVER FROM A FORTRAN ROUTINE, MAKE AN EXEC CALL LIKE THE ONE SHOWN BELOW. FOR MORE INFORMATION ON HOW THE EXEC CALL WORKS, SEE CHAPTER 2 IN THE HP MANUAL.

"RTE-6/VM PROGRAMMER'S REFERENCE MANUAL". FOR A LIST OF THE EQUIPMENT TABLE WORDS AND BITS, SEE SECTION L OF THE QUICK REFERENCE GUIDE FOR THE HP-1000.

SAMPLE CALL:

CALL EXEC(ICODE, LU, IFUNC, I_BUFF, I_LEN, I_CMND)

ICODE: THIS PARAMETER IS EITHER A 1, 2, OR 3.

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

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

LU : LOGICAL UNIT NUMBER. THE LOGICAL UNIT NUMBER FOR THE COMTAL IS SET AT SYSTEM GENERATION. AT THE MOMENT IT IS 36B.

IFUNC: THIS PARAMETER SIGNALS THE DRIVER AS TO THE TYPE OF HP <-> COMTAL COMMUNICATION THAT IS BEING REQUESTED.

ALTHOUGH THERE IS SOMETHING OF A RELATIONSHIP BETWEEN THE IFUNC CODES AND THE TRANSFER CODES OF SECTION 5.2 IN THE COMTAL USERS MANUAL, THE RELATIONSHIP IS HARD TO EXPLAIN.

SO WE WON'T TRY. INSTEAD, WE'LL GIVE A CHART THAT GIVES THE IFUNC VALUES AND THE RELEVANT COMTAL USER MANUAL SECTION NUMBERS WHERE FURTHER DETAILS ON THE TRANSFERS...

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 COMTAL
FOR DVR MANUAL REMARKS

= = = = = = = = = = = = = = = = = = = = = =

000B 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
I 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

*************** INITIATION SECTION FOLLOWS ***************

0104 *
0105 *
0106 *
0107 *
0108 *
0109 I.41 NOP
0110 JSB SETIO
0111 CLB
0112 LDA EOT6.1 I GET CONTROL SUBFUNCTION
0113 AND =B3700
0114 STA FUNC I SAVE FUNCTION MEMORY
0115 LDA EOT6.1 I GET CONTROL WORD OF REQUEST
0116 AND =B3 I ISOLATE THE FUNCTION
0117 CPA =B2 I CHECK FOR WRITE (SKIP IF NOT WRITE)
0118 JMP RORW1 I IF REQUEST FUNCTION = 2 (WRITE)
0119 LDB BIT15 I SET READ FLAG FOR ACTUAL BIT USED
0120 CPA =B1 I CHECK FOR READ (SKIP IF NOT READ)
0121 JMP RORW1 I IF REQUEST FUNCTION = 1 (READ)
0122 * ANYTHING ELSE IS CONSIDERED A CONTROL REQUEST
0123 LDA FUNC I GET CONTROL WORD
0124 SZA
0125 JMP REJCT I 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 I SET A=2, FOR ILLEGAL CONTROL REQUEST
0133 JMP I.41,i I RETURN TO IOC
0134 *
0135 *
0136 *
0137 ****** SETIO ROUTINE FOR INITIATION SECTION follows
0138
0139  SETIO NOP
0140  STA B  SAVE SELECT CODE FROM A (STATUS CARD SC)
0141  IOR CLC  OR IN "CLC" INSTRUCTION
0142  STA CLR2
0143  STA RORW  INCREMENT TO HIGHER SELECT CODE (DATA CARD SC)
0144  INA
0145  STA CLR1
0146  STA RORW
0147  LDA LIA  "LIA INSTRUCTION" TO A
0148  IOR B  OR IN SELECT CODE
0149  STA RORW2
014A  LDA STC  "STC INSTRUCTION" TO A
014B  IOR B
014C  STA IO1L3  STORE INSTRUCTION AT LABEL "IO1L3"
014D  LDA OTA
014E  IOR B
014F  STA IO1L2
0150  LDA OTB
0151  INB  INCREMENT SELECT CODE IN B
0152  IOR B
0153  STA IO1L1
0154  JMP SETIO.I  RETURN
0155
0156  ************* READ OR WRITE REQUEST *************
0157  *********** CHECK FOR COMTAL IN REMOTE ***********
0158  *
0159  RORUI STB RFLAG
0160  RORU CLC DAT.C  CLEAR CONTROL AND FLAG
0161  RORU1 CLC CST.C
0162  RORW2 LIA CST  GET STATUS
0163  AND NSBY  FIX FOR COMTAL V0/20 R.W.B.
0164  CPA NSBY  FIX FOR COMTAL V0/20 R.W.B.
0165  RSS  SKIP IF YES
0166  JMP RORW2  FIX FOR COMTAL V0 20 R.W.B.
0167
0168  ****************** DETERMINE WHICH SUBFUNCTION ******************
0169  ****************** NOTE THAT THE DISPLAY COMMAND AND TRACK BALL REQUESTS ARE
0170  * PERFORMED USING DEVICE INTERRUPTS.
0171  * HOWEVER IMAGE LINE, FUNCTION, AND PSEUDO-COLOR TRANSFERS ARE
0172  * PERFORMED USING DEVICE INTERRUPTS TO TRANSMIT THE COMMAND AND
0173  * ENDING SEQUENCE, BUT WITH A DMA TRANSFER IN BETWEEN.
0174  *
0175  LDA FUNC  GET CONTROL WORD
0183 * CPA DSPLV DISPLAY COMMAND REQUEST?
0184 JMP DSPLY
0185 * CPA ILTRV IMAGE LINE TRANSFER?
0186 JMP CHDCP
0187 * CPA FMTRV FUNCTION MEMORY TRANSFER?
0188 JMP CHDCP
0189 * CPA PCMTV PSEUDO-COLOR TRANSFER
0190 JMP CHDCP
0191 * CPA TBALV TRACK BALL REQUEST
0192 JMP TBALL
0193 * CPA SUPTV CHECK FOR TYPE 3 DMA SUPPORT CONTROL
0194 JMP CHDCP
0195 * CPA LINKV CHECK FOR LINK SUBFUNCTION
0196 JMP LINK
0197 * CPA DSPLY DISPLAY COMMAND REQUEST
0198 * CPA LDA =D1 SET STATE CONTROL VARIABLE
0199 STA EOT12.I WHAT TO DO
0200 JMP CHDCP
0201 * CPA LDB EOT9.I GET OPTION
0202 JMP LINK
0203 * CPA CLA CLEAR OUT A
0204 * CPA JMP 101 GO TO START 10 ROUTINE
0205 *??????????????? ILLEGAL SUBFUNCTION HERE ?????????????
0206 *
0207 *
0208 ********** PROCESS DISPLAY COMMAND REQUEST *************
0209 *
0210 DSPLY LDA =D1 SET STATE CONTROL VARIABLE
0211 STA EOT12.I WHAT TO DO
0212 LDB EOT9.I GET OPTION
0213 CLA CLEAR OUT A
0214 JMP 101 GO TO START 10 ROUTINE
0215 *
0216 ********** PROCESS TRACK BALL REQUEST *************
0217 *
0218 *
0219 *
0220 TBALV LDB RFLAG
0221 CPB BIT15 READ FLAG SET? ( SKIP IF WRITE)
0222 JMP RTBAL JUMP IF SET FOR READ
0223 *
0224 **** TRANSMIT WRITE COMMAND AND X-POSITION.
0225 *
0226 WTBAL LDA =D2 SET STATE CONTROL VARIABLE.
0227 STA EOT12.I
0228 LDB EOT7.I
0229  LDA B.I  CONTENTS OF BUFFER TO A
0230  AND =B0777  MASK FOR BITS 0-8
0231  STA B
0232  LDA =B3  TRANSFER CODE OF 3 TO A
0233  JMP 101  GO START I/O
0234  *
0235  **************** READ TRACK BALL ****************
0236  *****  ALSO READS LITTLE RED SWITCHES ON TRACK BALL APPARATUS.
0237  *
0238  RTBAL LDA =D5  SET FLAG TO TELL CONT. SECTION WHAT TO DO
0239  STA EQT12.I
0240  LDA =B3  TRANSFER CODE=3
0241  JMP 101  GO TO START I/O
0242  *
0243  *
0244  ******** CHECK FOR DCPC CHANNEL ASSIGNED ********
0245  *
0246  CHDCP DLD INTBA.I  ACCESS DCPC ASSIGNMENT WORDS FROM THE
0247  *  THE INTERRUPT TABLE WHOSE ADDRESS IS IN THE BASE PAGE
0248  *  COMMUNICATIONS AREA.
0249  CPA EQT1  IS CHANNEL 1 ASSIGNED TO THIS DRIVER
0250  JMP CH1  IF YES
0251  CPB EQT1  IS CHANNEL 2 ASSIGNED TO THIS DRIVER
0252  JMP CH2  IF YES
0253  LDA =B5  SET RETURN CODE TO 5
0254  JMP 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  CHFTN LDA FUNC  CONTROL WORD TO A
0263  CPA FMTRV
0264  JMP FMTR  FUNCTION MEMORY TRANSFER
0265  CPA PCMTV
0266  JMP PCMTR  PSEUDO COLOR TRANSFER
0267  CPA SUPTV
0268  JMP SUPPT  SUPPORT TRANSFER
0269  *
0270  ******** IMAGE LINE TRANSFER ROUTINE ********
0271  *
0272  ILTR LDA =D8  SET STATE CONTROL VARIABLE
0273  STA EQT12.I
0274  LDA EQT9.I  GET INSTRUCTIONS FOR LINE TRANSFER
0275  IOR RFLAG  SET READ OR WRITE AS PER READ FLAG
0276  ILTR2 STA B
0277  LDA =B1  SET TRANSFER CODE TO 1
0278  STA TRAN  SAVE TRANSFER CODE
0279  JMP I01  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  !OR 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 I01
0297  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0298  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0299  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0300  SUPPT LDA =D8  SET DMA TRANSFER RETURN STATUS FOR CONTINUATION SECTION
0301  STA EOT12.1  AND SAVE IT
0302  LDA EQT9.1  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 I01  AND GO DO IT
0308  *
0309  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0310  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0311  LINK LDA =D1  SET NO BLOCK TRANSFER
0312  STA EOT12.1  SAVE THIS
0313  LDB EQT9.1  GET COMMAND OPTION
0314  LDA =B1  SET TRANSFER CODE
0315  JMP I01  GO DO IT
0316  *
0317  *
0318  *
0319  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0320  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0320  *
0321 *
0322 *
0323 I01 NOP
0324 I01L1 OTE DAT PUT TRANSFER COMMAND ON DATA LINES
0325 I0R GOBIT
0326 I01L2 OTA CST PUT TRANSFER CODE ON COMMAND LINES
0327 I01L3 STC CST.C SEND THE GO PULSE, START TRANSFER
0328 CLA NOW RETURN TO IOC WITH
0329 JMP I.41.1 OPERATION INITIATED (A=0=OK)
0330 *
0331 WHEN THE COMMAND IS RECEIVED BY THE COMTAL, AN
0332 INTERRUPT FROM THE COMTAL WILL CALL CIC AND THIS
0333 DRIVERS COMPLETION AND CONT. SECTION WILL EXEC.
0334 *
0335 *
0336 ********** COMPLETION SECTION FOLLOWS **********
0337 *
0338 *
0339 *
0340 C.41 NOP
0341 JSB CSIO
0342 LDA EQT1.I CHECK FOR SPURIOUS INTERRUPT
0343 AND #B???? GET I/O REQUEST LIST POINTER
0344 S2A IS A REQUEST IN PROGRESS
0345 JMP COMP2 IF YES GO PROCESS REQUEST
0346 STA EQT15.I 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 EQT12.I 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 *

207
0367 SCA2 LDA IDSC INTERUPTING DEVICE SELECT CODE TO A
0368 CPA SSC IS INTERRUPTING SELECT CODE THE STATUS CARD?
0369 RSS SKIP IF YES?
0370 JMP SPUR2 JUMP TO SPURIOUS INTERRUPT RETURN IF NOT SET
0371 LDA =D3 SET STATE CONTROL VAR.
0372 STA EOT12,1 ADDRESS OF BUFFER TO B
0373 LDB EOT7,I INCREMENT B (BUFFER ADDRESS)
0374 INB CONTENTS OF BUFFER+1 TO A
0375 LDA B,1 SIGNIFYS X-POSITION
0376 AND =B0777 TRANSFER CODE OF 3
0377 IOR BIT14
0378 STA B
0379 LDA =B3
0380 IOR GOBIT
0381 JMP C101
0382 *
0383 *** TRACK BALL 2
0384 *
0385 SCA3 LDA IDSC INTERUPTING DEVICE SELECT CODE TO A
0386 CPA SSC IS INTERRUPTING SELECT CODE THE STATUS CARD?
0387 RSS SKIP IF YES?
0388 JMP SPUR2 JUMP TO SPURIOUS INTERRUPT IF NOT SET
0389 LDA =D4 SET STATE CONTROL VARIABLE
0390 STA EOT12,1
0391 JMP TBDUN
0392 *
0393 *** TRACK BALL 3
0394 *
0395 SCA5 LDA IDSC INTERUPTING DEVICE SELECT CODE TO A
0396 CPA SSC IS INTERRUPTING SELECT CODE THE STATUS CARD?
0397 RSS SKIP IF YES?
0398 JMP SPUR2 JUMP TO SPURIOUS INTERRUPT IF NOT SET
0399 LDA =D6 SET STATE CONTROL VARIABLE
0400 STA EOT12,1
0401 JMP C102 JUMP TO START A READ
0402 *
0403 *** TRACK BALL 4
0404 *
0405 SCA6 LDA IDSC INTERUPTING DEVICE SELECT CODE TO A
0406 CPA DSC IS INTERRUPTING SELECT CODE THE DSIS CARD?
0407 RSS SKIP IF YES?
0408 JMP SPUR2 JUMP TO SPURIOUS INTERRUPT IF NOT SET
0409 LDA =D7 SET STATE CONTROL VARIABLE
0410 STA EOT12,1
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.I ADDRESS OF BUFFER TO B
0415 STA B.I STORE X-POSITION IN BUFFER
0416 JMP CI02
0417 *
0418 ** TRACKBALL 5
0419 *
0420 SCA7 LDA IDSC INTERRUPTING DEVICE SELECT CODE TO A
0421 CPA DSC IS INTERRUPTING SELECT CODE THE DSCA CARD?
0422 RSS SKIP IF YES?
0423 JMP SPUR2 JUMP TO SPURIOUS INTERRUPT IF NOT
0424 LDA =D4
0425 STA EQT12.I SET STATE CONTROL VARIABLE
0426 GET2 LIA DAT DATA FROM DATA LINES TO A
0427 LDB EQT7.I GET BUFFER ADDRESS
0428 INB POINT TO WORD TWO
0429 STA E.I STORE Y-POSITION IN BUFFER+1
0430 JMP TBDUN
0431 *
0432 TBDUN LDA =B3 THIS ADDED PGER RUB 6/15/81
0433 IOR CLBIT
0434 IOR ENBIT
0435 JMP CI03
0436 *
0437 ** TRACK BALL TRANSMIT END
0438 *
0439 SCA4 LDA IDSC INTERRUPTING DEVICE SELECT CODE TO A
0440 CPA SSC IS INTERRUPTING SELECT CODE THE STATUS CARD?
0441 RSS SKIP IF YES?
0442 JMP SPUR2 JUMP TO SPURIOUS INTERRUPT IF NOT
0443 LDB =B2 SET TRANS. LOG FOR WRITE
0444 LDA EQT6.I GET CONTROL WORD
0445 AND =B3 ISOLATE FUNCTION CODE
0446 CPA =B1 IS IT A READ?
0447 LDB =B4 SET TRANS. LOG TO 4 FOR READ
0448 JMP CEND1
0449 *
0450 *
0451 *
0452 CI01 NOP
0453 OTB DAT
0454 CI03 OTA.CST
0455 STC CST.C
0456 ISZ C.41
0457 JMP C.41.I
0458 *
**DMA TRANSFER 1**

**SCAB LOG ID INTERUPTING DEVICE SELECT CODE TO A**

**CPL SSC IS INTERRUPTING SELECT CODE THE STATUS CARD?**

**RSS SKIP IF YES?**

**JMP SPUR2 JUMP TO SPURIOUS INTERRUPT IF NOT**

**LDA =D SET STATE CONTROL VARIABLE**

**B47T•DSPC INITIALIZATION INSTRUCTIONS TO SPECIFY THE SELECT CODES OF THE ASSIGNED DCPC**

**LDA OTA "OTA INSTRUCTION" TO A**

**IOR B OR IN THE LOWER SELECT CODE**

**STA D3 STORE AT LABEL D3**

**STA D5 STORE AT LABEL D5**

**ADA =B4 ADD 4 TO INSTR. TO CHANGE TO HIGHER SELECT CODE**

**STA D1**

**LDA CLC "CLC INSTR." TO A**

**IOR B OR IN LOWER SELECT CODE**
0505  IOR DSC
0506   STA D7+1
0507 *
0508 *
0509 *
0510 ********** THIS ROUTINE INITIATES THE DCPC DATA TRANSFERS ***********
0511 ** THE INSTRUCTIONS BELOW WITH D LABELS ARE MODIFIED BY THE
0512 * THE ROUTINE "SETIO" TO SPECIFY THE CORRECT SELECT CODES
0513 * FOR THE DCPC CHANNEL ASSIGNED BY THE SYSTEM
0514 *
0515 GOIO  LDA EOT4,I
0516    AND #B77  MASK FOR COMMAND SELECT CODE
0517    INA    INCREMENT FOR DATA SELECT CODE
0518    IOR BIT15  TURN ON HANDSHAKE BIT (STC)
0519   D1  OTA 6B  PUT CONTROL WORD 1 TO DCPC CHANNEL (SELECT 6 OR 7)
0520   D2  CLC 2B  PREPARE MEM. ADDR. REGISTER FOR CW2 (SELECT CODE 2 OR 3)
0521   LDA EOT6,I  GET CONTROL WORD
0522    AND #B3  MASK FOR FUNCTION
0523   STA B   STORE FUNCTION IN B
0524   LDA EOT7,I  ADDRESS OF BUFFER TO A
0525    CPB =B1  IS FUNCTION A READ?
0526    IOR BIT15  TURN ON "IN" BIT FOR A READ, IF YES
0527   D3  OTA 2B  CONTROL WORD 2 TO DCPC CHANNEL (SC 2 OR 3)
0528   D4  STC 2B  PREPARE MEM. ADDR. REGISTER FOR CW3 (SC 2 OR 3)
0529   LDA EOT8,I  GET LENGTH OF TRANSFER
0530    CMA,INA  MAKE TWO'S COMPLEMENT
0531   D5  OTA 2B  CONTROL WORD 3 TO DCPC CHANNEL (SELECT CODE 2 OR 3)
0532    CPB =B1  IS IT A READ?
0533    JMP D7  JUMP IF YES
0534   D6  STC 6B  ACTIVATE DCPC CHANNEL (SELECT CODE 6 OR 7)
0535    CLC DAT  CLEAR DEVICE
0536    STF DAT  ACTIVATE DEVICE?
0537    JMP GOEND
0538   D7  STC 6B  ACTIVATE DCPC CHANNEL (SELECT CODE 6 OR 7)
0539    STC DAT,C  ACTIVATE DEVICE
0540  GOEND ISZ C.41  NOW RETURN TO CIC WITH
0541   JMP C.41.I  CONTINUATION
0542 *
0543 ** DMA TRANSFER 2
0544 *
0545 SCA9  LDA EOT11,I  GET LOW SELECT CODE OF DCPC CHANNEL
0546    ADA =B4  ADD 4 TO CONVERT LOW DCPC SELECT CODE TO HIGH S.C.
0547 CHKF D CPA IDSC  IS INTERRUPTING SELECT CODE THAT OF THE
0548 *  ASSIGNED DCPC CHANNEL?
0549    RSS  SKIP IF YES
0550    JMP SPUR2  JUMP TO SPURIOUS INTERRUPT IF NOT
0551  LDA =D10  SET CONTROL STATE VARIABLE
0552    STA EQT12.1
0553  ******************************************************
0554  ******************************************************
0555  *
0556  *
0557    LDA TRAN
0558    IOR CLBIT  * SET CLEAR BIT
0559    IOR ENBIT  * ADD END BIT
0560    JMP C103   * GO TO IT
0561  *
0562  ** DMA TRANSFER 3
0563  *
0564  ******************************************************
0565  SCA18 LDA IDSC  INTERRUPTING DEVICE SELECT CODE TO A
0566  CPA SSC  IS INTERRUPTING SELECT CODE THE STATUS CARD?
0567    RSS  SKIP IF YES?
0568    JMP SPUR2  IF NOT GO TO SPURIOUS INTERRUPT
0569    LDB EQT11.1 RETRIEVE LOWER SELECT CODE OF ASSIGNED DCPC CHANNEL
0570    LDA CLC  "CLC INSTRUCTION" TO A
0571    IOR B  OR IN THE CHANNELS LOWER SELECT CODE
0572    STA CD1  STORE INSTRUCTION AT LABEL CD1
0573    ADA =B4  ADD 4 TO INSTRUCTION IN A TO ADJUST TO HIGHER SELECT CODE
0574    STA CD2  STORE INSTRUCTION AT LABEL CD2
0575  *
0576    CD1  CLC 2B.C  CLEAR LOW DCPC SELECT CODE
0577    CD2  CLC 6B.C  CLEAR HIGH DCPC SELECT CODE
0578    LDB EQT8.1 LENGTH OF BUFFER TO B (TRANSMISSION LOG)
0579  CEND1 LIA CST  GET STATUS WORD FROM COMTAL
0580    AND =B37  STRIP OFF UNUSED BITS
0581    STA SAVE1  SAVE IN SAVE1 TEMPORARILY
0582    LDA EQT5.1 REMOVE PREVIOUS STATUS
0583    AND =B1774B0 BITS IN EQT WORD 5
0584    IOR SAVE1  OR IN NEW BITS
0585    STA EQT5.1 AND, RESET INTO EQT WORD 5
0586  *
0587  CEND2 CLC DAT.C  CLEAR DEVICE DATA SELECT CODE
0588    CLC CST.C  CLEAR DEVICE COMMAND SELECT CODE
0589  *
0590  *
0591    CLA  SET A = 0 = OK RETURN CODE
0592    IOR =B100000 SET BIT TO RETURN DCPC CHANNEL
0593    JMP C.41.I MAKE COMPLETION RETURN TO CIC
0594  *
0595  *
0596  ****** SETIO SUBROUTINE FOR COMPLETION/CONTINUATION SECTION ******
0597 *
0598 CSIO NOP
0599 STA IDSC
0600 LDA EQT4,1
0601 AND 'B77'
0602 STA SSC
0603 INA
0604 STA DSC
0605 LDA OTA
0606 IOR SSC
0607 STA CIA
0608 LDA STC
0609 IOR SSC
0610 STA CIA+1
0611 INA
0612 *
0613 STA CIA+2
0614 LDA CLI
0615 IOR SSC
0616 STA CEN2+1
0617 INA
0618 STA CEN2
0619 STA CIA+1
0620 LDA LIA
0621 IOR SSC
0622 STA CEN1
0623 INA
0624 STA GET1
0625 STA GET2
0626 LDA OTB
0627 IOR DSC
0628 STA CIA+1
0629 JMP CSIO.1 RETURN
0630 *
0631 *
0632 *
0633 *
0634 *
0635 *
0636 CONSTANT AND STORAGE AREA
0637 *
0638 RFLAG BSS .1
0639 FUNC BSS .1
0640 TRAP BSS .1
0641 SAVE1 BSS .1
0642 SAVE2 BSS .1

SAVE SELECT CODE OF INTERRUPTING DEVICE

MASK FOR STATUS SELECT CODE

SAVE STATUS SELECT CODE AT SSC

ADD 1 TO A FOR DATA SELECT CODE

SAVE DSCA SELECT CODE AT DSC

"OTA INSTRUCTION" TO A

OR IN STATUS SELECT CODE

STORE INSTRUCTION IN A AT LABEL CIA

ADD 1 TO INSTRUCTION IN A TO CHANGE
STATUS SELECT CODE TO DATA SELECT CODE
THE FOLLOWING VALUES (SUFFIX V) DETERMINE WHICH DRIVER SUBFUNCTIONS ARE INVOKED, PARAMETER IFUNC IN SAMPLE CALL ABOVE.

THE FOLLOWING CONSTANTS ARE USED TO READ THE EQUIPMENT TABLE ASSOCIATED WITH THIS DRIVER. SEE THE HP-1000 QUICK REFERENCE
** MANUAL FOR DIAGRAMS AND TABLES ON WHICH THESE CONSTANTS ARE BASED. **

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 CTABLE 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
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.