NASA Technical Memorandum 87646

IPLIB (IMAGE PROCESSING LIBRARY) USER'S MANUAL

FOR REFERENCE

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

DECEMBER 1985
INTRODUCTION

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

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

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

The manual is divided into six sections:

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

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

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

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

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

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

The following references are suggested for programmers working with IPLIB:


SUBPROGRAM HANDLING

LOADING SUBPROGRAMS

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

SEA %IPLIB::21

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

LOADR
RE:*TNOTE
SEA,%IPLIB
SEA
DI
EN

ADDING SUBPROGRAMS

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

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

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

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

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

(:)MERGE
(ENTER DESTINATION NAMR) %IPLIB<CR>
(ENTER COMMAND NAMR) 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
IPLIB CATALOG

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. 30).
CMIND: sends a COMTAL command given as constant string and length parameter from HP to COMTAL (p. 33).
COUNT: compiles the pixel count for each of the possible values, 0 - 255 (p. 36).
DELAY: puts HP in a busy wait for number of seconds designated (p. 39).
DIGIT: function that takes 0 - 9 integer input and returns '0' - '9' (p. 41).
DSPBU: displays a monochrome COMTAL image (p. 43).
DSPCL: displays a color COMTAL image (p. 45).
DSPGR: displays a graphic plane to the COMTAL monitor (p. 48).
DSPVD: displays the COMTAL image (5) set to the video camera (p. 50).
HILO: scans a monochrome image and returns the high and low pixel values (p. 52).
HISTO: displays a scaled histogram of the designated image on the COMTAL monitor (p. 54).
ICOPY: copies one monochrome COMTAL image to another (using CMND) (p. 57).
ICOPY2: 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).
PROFL: 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).
RDIRT: reads one pixel from a COMTAL monochrome image (p. 94).
RDIRC: reads a rectangle of pixels from a COMTAL monochrome image (p. 97).
RDUT: reads the contents of a COMTAL look-up table (p. 101).
RDPUS: 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 128; differences <0 set to 0 (p. 119).
SUBIM : subtracts two images with no offset; differences <0 set to 0 (p. 123).
THRESH : 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).
WRIP2 : writes one pixel value to a COMTAL image memory (p. 150).
WRIPN : writes an array of integers to a rectangle of a COMTAL image memory; one pixel value / integer in the buffer (p. 153).
WRUL : writes a look-up table to the COMTAL (p. 157).
WRPSU : writes a pseudo-color table to the COMTAL (p. 159).
WRTAR : writes a target (cursor) location to the COMTAL (p. 161).

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

TADD2 : program that tests ADDI2 (p. 163).
TADDI : program that tests ADDI3 (p. 164).
TCLR : program that tests DSPCL, "Display Color" (p. 165).
TCLRG : program that tests CLRGR, "Clear Graphics" (p. 166).
TCLR1 : program that tests CLRIM, "Clear Image" (p. 167).
TCMN2 : program that sends all possible single bytes to COMTAL 1 at a time (p. 168).
TCMN3 : program that sends COMTAL commands via the HP keyboard; tests CMDND (p. 169).
TCHT : 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).
TDIG1 : program that tests the function DIGIT (p. 173).
TDSP : program that tests DSPBU, "Display Black & White" and DSPCL, "Display Color" (p. 174).
THIST : program that tests HIST, "HISTogram" (p. 175).
TNORM : program that tests NORM, "Normalize" (p. 176).
TNOTE : program that tests NOTE2 and NOTES (p. 177).
TPNT : program that tests PAINT (p. 178).
TPROF : program that tests PROFIL, "PROFiling" (p. 179).
TRANG : program that tests the function "RANGE" (p. 180).
TRDATA : program that tests RDTAR, "Read TARget" (p. 181).
TSETV : program that tests SETV, "SET Video camera" (p. 182).
TSPRD : program that tests SPRED, "SPREAD pixel values" (p. 183).
TSUB1 : program that tests SUBIM and SUBI2 (p. 184).
TTDRS : program that tests THRSH, "THRESHolding" (p. 185).
TTSTI : program that tests TSTII, "Test Image II" (p. 186).
TTV2C : program that tests TV2CM and TV2C4, "TV to Color" transfers (p. 187).
TWAIT : program that tests WAIT (p. 188).
TWIPE : program that tests WIPGR and DSPGR (p. 189).
TXFD : program that tests BWFDS and CLFDS, "Black & White From Disk" and "Color From Disk" (p. 190).
TXGLN : program that tests LRGGLN and RDLGN, "Write Graphics Line" and "Read Graphics Line" (p. 191).
TXGPT : program that tests LRGPT and RDGPT, "Write Graphics Point" and "Read Graphics Point" (p. 192).
TXILN : program that tests LRLIN and RDLIN, "Write Image Line" and "Read Image Line" (p. 193).
TXIPT : program that tests LRIPT and RDIPT, "Write Image Point" and "Read Image Point" (p. 194).
TXIRC : program that tests LIRIC and RDIRC, "Write Image Rectangle" and "Read Image Rectangle" (p. 195).
TXLUT : program that tests LRLUT and RDLUT, "Write Look Up Table" and "Read Look Up Table" (p. 196).
TXPSU : program that tests WRPSU and RDPSU, "Write Pseudo-color table" and "Read Pseudo-color table" (p. 197).
TXTAR : program that tests WRTAR and RDTAR, "Write TARget" and "Read TARget" (p. 199).

CROSS-REFERENCE FILE

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

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

ADDI2 calls:
  *DIGIT CMDN RANGE

ADDIM calls:
CMMND DIGIT RANGE

BUFS calls:
CMMND DIGIT OPEN RANGE READF WRILN

CLFD calls:
CMMND DIGIT OPEN RANGE READF WRILN

CLGR calls:
CMMND DIGIT RANGE

CLRM calls:
CMMND DIGIT RANGE

CMMN2 calls:
CMMND DIGIT RANGE

LEN calls no other procedures.

COUNT calls:
RANGE RDIL2

DELAY calls no other procedures.

DIGIT calls:
RANGE

DSPBW calls:
RANGE

DSPCL calls:
CMMND DIGIT RANGE

DSPGR calls:
CMMND DIGIT RANGE

DSPVD calls:
CMMND DIGIT

HLO calls:

RDIL2

HISTO calls:
CMMN2 DELAY DIGIT RANGE

ICOPY calls:

ICPY calls:
CMMND DIGIT RANGE

ICPY2 calls:
CMMND DIGIT RANGE

MERGE calls no other procedures.

NORM calls:

HILO RDIL2 WRIL2

NOTE2 calls:
ADDGR CMMN2 CMMND DELAY DIGIT LEN RANGE

NOTES calls:
CMMN2 CMMND DELAY DIGIT DSPGR RANGE

PAINT calls:
CMMND RDTAR WAIT WRIRC

PROFL calls:
CHAR CMMND DIGIT RANGE WAIT

RANGE calls no other procedures.

RDGLN calls:
RANGE
RDGPT  calls:
   BTEST RANGE
RDIL2  calls:
   *ICHAR RANGE
RDILN  calls:
   RANGE
RDRIPT 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:
   CMMND DIGIT RANGE
SPLIT  calls no other procedures.
SPRED  calls:
   FLOAT HILO IFIX RDIL2 WRIL2
SSORT  calls no other procedures.
SUB12  calls:
   CMMND DIGIT RANGE
SUBIM  calls:
   CMMND DIGIT RANGE
THRSH  calls:
   RANGE RDIL2 WRIL2
TSTI1  calls:
   WRILN
   TV2C4  calls:
   ADDI2 RANGE TV2CM
TV2CM  calls:
   CMMND DIGIT DSPBU RANGE
WAIT  calls no other procedures.
UIPGT  calls:
   CMMND2 DIGIT RANGE
URGLN  calls:
   RANGE
WRGPT  calls:
   *ISCLR *IBSET RANGE
WRIL2  calls:
   CHAR RANGE
WRILN  calls:
   RANGE
wr ipt calls:
RANGE
wr irc calls:
CHAR RANGE RDIL2 WRIL2
wr lut calls:
RANGE
wr psu calls: no other procedures.
wr tar calls:
RANGE

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

tadd2 calls:
ADDI2 CMND DIGIT RANGE
taddi calls:
ADDIM CMND DIGIT RANGE
tclr calls:
DSPCL
tclrg calls:
CLRG CMND DIGIT RANGE
tclri calls:
CLRIM CMND DIGIT RANGE
tcmn2 calls:
CMND WAIT
tcmn calls:
CMND
tcnt calls:
COUNT RANGE RDIL2
tcons calls: no other procedures.
tcopy calls:
CMND DIGIT ICOPY RANGE
tdel calls:
DELAY CMNN2 WAIT
tdig calls:
DIGIT RANGE
tdsp calls:
CMND DSPBU DSPCL RANGE WAIT
tdspv calls:
DSPVD CMND DIGIT WAIT DSPBU RANGE
thist calls:
CMNN2 DIGIT HISTO RANGE
tnorm calls:
HILO NORML RANGE RDIL2 WRIL2	note calls:
CMNN2 CMND DELAY DIGIT DSPGR NOTE2 NOTES RANGE WRTAR
tpnt calls:
CMDND PAINT RANGE RDILH RDTAR WRILN WRIRC

TPROF calls:
CMDND DIGIT PROFL RANGE WAIT

TRANG calls:
RANGE

TRDTA calls:
RANGE RDTAB

TRSET calls no other procedures.

TSETV calls:
SETV RANGE DIGIT CMDND DSPBU

TSPRD calls:
RANGE RDIL2 SPRED WRIL2

TSSRT calls no other procedures.

TSUBI calls:
CMDND DIGIT RANGE SUBI2 SUBIM WAIT

TTHRS calls:
RANGE RDIL2 THSH WRIL2

TTSTI calls no other procedures.

TTV2C calls:
ADDI2 CMDND DIGIT DSPBU DSPVD RANGE TV2C4 TV2CM WAIT

TWAIT calls:
WAIT

TWIPE calls:
CMDND2 DIGIT DSPGR RANGE WAIT WIPGR

TXFDS calls:
BUFDS CLFDS CMDND DIGIT RANGE WRILM

TXGLN calls:
RANGE RDGLN WRGLN

TXGPT calls:
RANGE RDGPT RDTAR WRGPT

TXILN calls:
RANGE RDILN WRILN

TXIPT calls:
RANGE RDIPT WAIT WRIPT

TXIRC calls:
RANGE RDILN RDIRC WRILN WRIRC

TXLUT calls:
RANGE RDLUT WRLUT

TXPSU calls:
RDPSU WRPSU RANGE WAIT

TXTAR calls:
RDTAR WRDTAR RANGE
&ADDIM T=00004 IS ON CR00021 USING 0024 BLKS R=0000

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

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

**WRITTEN BY:**

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

**REVISION HISTORY:**

LOCAL VARIABLES:

| INTEGER IBUF(128) | character overlay for IBUF |
| CHARACTER*255 CBUF | character overlay for CBUF |
| 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.

```
0101 C
0102 1 'UN I 8 $' //
0103 C
0104 2 'AS T 8 '/DIGIT(C) '/'DIGIT(A) '/'DIGIT(B) '/' $' //
0105 C
0106 3 'D I 8 $' //
0107 C
0108 4 'SE COM G + B / 1 $' // sets 0 offset by default, and "$ / I" sets no scaling.
0109 C
0110 C
0111 5 'A COM $' //
0112 C
0113 6 'I '/DIGIT(C)' / D R $' //
0114 C
0115 C
0116 7 'D I '/'DIGIT(C) '/' $' //
0117 C
0118 8 'SU COM ' //
0119 C
0120 C
0121 CALL CMDND(IBUF, 7B)
0122 RETURN
0123 C**ERROR RETURNS
0124 C
0125 E001 WRITE(TERM, E203) A
0126 E003 FORMAT( ' THE 2ND IMAGE PARAMETER.', I3, ', ', IS OUT OF RANGE.' )
0127 GOTO 8900
0128 C
0129 E101 WRITE(TERM, E103) B
0130 E103 FORMAT( ' THE 3RD IMAGE PARAMETER.', I3, ', ', IS OUT OF RANGE.' )
0131 GOTO 8900
0132 C
0133 E201 WRITE(TERM, E203) C
0134 E203 FORMAT( ' THE 1ST IMAGE PARAMETER.', I3, ', ', IS OUT OF RANGE.' )
0135 GOTO 8900
0136 C
```
0137  8900 WRITE(TERM, 8901)
0138  8901 FORMAT(' ADDIM RETURNS WITHOUT FURTHER PROCESSING.')
0139         RETURN
0140  C
0141         END
0142
&ADDI2 T=00004 IS ON CR00021 USING 00024 BLKS R=0000

0081 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0082 SUBROUTINE ADDI2(C, A, B, SCALE)
0083 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0084 C
0085 C***PARAMETER DECLARATIONS:
0086 C
0087 INTEGER C  ! the COMTAL image into which the sum of
0088 C          ! image A and image B is placed (C = A + B)
0089 INTEGER A, B  ! the images whose sum is taken (C = A + B)
0090 INTEGER SCALE! each pixel sum divided by this number.
0091 C
0092 C***INTRODUCTION:
0093 C
0094 C The subroutine "ADD IMAGES" takes a pixel by pixel sum
0095 C of images A and B, divides each sum by SCALE, and places the results
0096 C into image C. SCALE must be between 1 and 9 inclusive.
0097 C The truecolor image B is used in the processing of ADDI2, and will
0098 C be left as the combination of C, A, and B for red, green, and blue
0099 C respectively.
0100 C
0101 C***LANGUAGE:
0102 C
0103 C FORTRAN 77, the HP 1000 version for RTE-6/VM.
0104 C
0105 C***LIMITATIONS:
0106 C
0107 C The truecolor B image is destroyed during this operation. C is
0108 C obviously destroyed. This subroutine is accomplished using COMTAL
0109 C commands that exploit the pipeline processors. Because of this, the
0110 C processing steps are obscure. For example, there is no motivation
0111 C outside the COMTAL instructions for making the combination of C, A, and B
0112 C a color image. Readers should be aware of these obscurities before trying to
0113 C understand the code.
0114 C
0115 C The three image numbers must be distinct.
0116 C
0117 C If the sum of any two pixels exceeds 255, the value in C is set to 255.
0118 C
0119 C The SCALE factor must be between 1 and 9 inclusive.
0120 C
0121 C If any of the image numbers are out of range, an error message is printed
0122 C and no further processing takes place.
0123 C
0124 C This subroutine assumes that 0 is not a legal image for the COMTAL
configuration.

**SUBPROGRAMS CALLED:**

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

**WRITTEN BY:**

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

**REVISION HISTORY:**

**LOCAL VARIABLES:**

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

```plaintext
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 COM ';
CALL CMDND(IBUF, 77)
RETURN
```

C***ERROR RETURNS

8001 WRITE(TERM, 8003) A
8003 FORMAT(’THE 2ND IMAGE PARAMETER,’ I3, ’ IS OUT OF RANGE.’)
GOTO 8500
8021 WRITE(TERM, 8023) B
8023 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
0137  B900 WRITE(TERM, 8901)
0138  B901 FORMAT( 'ADD12 RETURNS WITHOUT FURTHER PROCESSING.' )
0139     RETURN
0140  C
0141     END
0142
&BUFDS T=00004 IS ON CR0002: USING 00018 BLKS R=0000

0011 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
002 SUBROUTINE BUFDS(IMAGE, FLNAME)
003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
004 C
005 C***PARAMETER DECLARATIONS:
006 C
007 INTEGER IMAGE ! the number of the COMTAL image plane to be
008 C ! filled from the HP disk file.
009 INTEGER FLNAME(3) ! the HP filename from which an image will be read.
010 C
011 C***INTRODUCTION:
012 C
013 C The subroutine "Black and White From Disk" transfers one b&w image
014 C from the HP to the COMTAL. The image number and the name of the HP
015 C disk are given as parameters.
016 C
017 C***LANGUAGE:
018 C
019 C FORTRAN 77, the HP 1000 version for RTE-6/M.
020 C
021 C***LIMITATIONS:
022 C
023 C BUFDS only works for b&w images. Since a color image requires
024 C three separate b&w images, another subroutine, CLFDS is available
025 C for reading color images from the disk.
026 C
027 C***SUBPROGRAMS CALLED:
028 C
029 C name source load remarks
030 C-------- ------- ---------- ---------------------------------------------------------
031 C RANGE &RANGE XRANGE logical function that determines if the 1st
032 C argument is within the 2nd and 3rd inclusive.
033 C OPEN ------- ------- HP FORTRAN77 intrinsic subroutine: opens a file
034 C and stores data block information in first param.
035 C READF ------- ------- HP FORTRAN77 intrinsic subroutine: reads a record
036 C from a file; requires an OPENed data block.
037 C WRILN &WRILN XWRILN writes a COMTAL Image horizontal Line: 2 pixels
038 C per integer in the buffer.
039 C CMNND &CMNND XCMNND sends commands to the COMTAL as if they were
040 C typed at the COMTAL keyboard.
041 C DIGIT &DIGIT XDIGIT a character*1 function that returns a single
042 C ASCII digit when given an integer 0-9
043 C
044 C***WRITTEN BY:

19
The code on which this subprogram is based was written by NETTE D. FAULCON, July, 1983. This subprogram was written by KEITH MILLER, July, 1984, with the support of a NASA-ASEE summer fellowship.

***REVISION HISTORY:

***LOCAL VARIABLES:

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

***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
CALL CBUF = 'D' I '/'DIGIT(IMAGE) ! Display the image to be filled.
CALL CMDND(IBUF, 5)
DO 1000 ROW = 0, 511
CALL READF(DBLOCK, IERR, IBUF)
IF (IERR .LT. 0) GOTO 8301 ! Error return, bad read
CALL URILN(IMAGE, ROW, IBUF)
1000 CONTINUE
RETURN
C
ERROR RETURNs
C
0095 8001 WRITE(TERM, 8003) IMAGE, IMLO, IMHI
0096 8003 FORMAT(' IMAGE NUMBER.'13.' IS OUT OF RANGE:'.214.'.'
0097         GOTO 8900
0098 C
0099 8201 WRITE(TERM, 8203) IERR
0100 8203 FORMAT(' ERROR OCCURED DURING IMAGE FILE OPENING:'.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
SUBROUTINE CLFDS(RED, GREEN, BLUE, COLOR, FLNAME)

INTEGER RED, GREEN, BLUE ! colors of the COMTAL image planes
INTEGER COLOR ! the truecolor image to be formed and displayed.
INTEGER FLNAME(3) ! the HP filename from which an image will be read.

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.

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

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.

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.

SUBPROGRMS 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 and stores data block information in first param.
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
**LOCAL VARIABLES:**

- `DBLOCK(144)`
- `IERR`
- `IBUF(256)`
- `ROW`
- `RANGE`
- `IMLO`, `IMHI`
- `TRLO`, `TRHI`
- `TERM`
- `CBUF`
- `DIGIT`

**INITIALIZATIONS:**

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

**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, TRIO, 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(RANGE)  ! Display the RED Image as it is filled.
0099  CALL CMMND(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 CMMND(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 CMMND(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, GREEN, and BLUE respectively; then the following CMMND calls read as follows: UNassign Image #C; Assign Truecolor #C red #R green #G blue #B; Display Image #C
0123  C
0124  CBUF = 'UN I' //DIGIT(COLOR)
0125  CALL CMMND(IBUF,6)
0126  CBUF = 'AS T' //DIGIT(COLOR)  //DIGIT(RANGE)  //DIGIT(BLUE)
0127  1  //DIGIT(GREEN)  //DIGIT(BLUE)
0128  CALL CMMND(IBUF,12)
0129  CBUF = 'D I' //DIGIT(COLOR)
0130  CALL CMMND(IBUF,5)
0131  RETURN
0132  C
0133  C***ERROR RETURNS
C
8001 WRITE(TERM, 8003) RED, IMLO, IMHI
8003 FORMAT(‘ RED IMAGE NUMBER,’I3,’ IS OUT OF RANGE:’214,’’
8008 GOTO 8900
8137 C
8138 8101 WRITE(TERM, 8103) GREEN, IMLO, IMHI
8143 8103 FORMAT(‘ GREEN IMAGE NUMBER,’I3,’ IS OUT OF RANGE:’214,’’
8148 GOTO 8900
8149 C
8150 8301 WRITE(TERM, 8303) RED, GREEN, BLUE
8156 8303 FORMAT(‘ 3 MONOCHROME IMAGES MUST BE DISTINCT. YOURS:’I4)
8157 GOTO 8900
8158 C
8159 8501 WRITE(TERM, 8503) IERR
8163 8503 FORMAT(‘ ERROR WHILE OPENING IMAGE FILE:’I5)
8165 GOTO 8900
8167 C
8168 8601 WRITE(TERM, 8603) IERR
8172 8603 FORMAT(‘ ERROR WHILE READING IN THE RED MONOCHROME IMAGE:’I4)
8173 GOTO 8900
8175 C
8176 8701 WRITE(TERM, 8703) IERR
8180 8703 FORMAT(‘ ERROR WHILE READING IN THE GREEN MONOCHROME IMAGE:’I4)
8182 GOTO 8900
8184 C
8185 8801 WRITE(TERM, 8803) IERR
8190 8803 FORMAT(‘ ERROR WHILE READING IN THE BLUE MONOCHROME IMAGE:’I4)
8192 GOTO 8900
8194 C
8195 8901 WRITE(TERM, 8901)
8199 8901 FORMAT(‘ CLFDS FAILS. NO TRANSFER TAKES PLACE.’)
8201 RETURN
8203 END
0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE CLRGR( GRAPH )
0003 C
0004 C***PARAMETER DECLARATIONS:
0005 C
0006 C
0007 INTEGER GRAPH ! the number of the COMTAL GRAPH to be cleared.
0008 C
0009 C***INTRODUCTION:
0010 C
0011 C CLEAR GRAPH clears the graphics designated by GRAPH. GRAPH should
0012 C be within the range 1-4 for the present IRD COMTAL system.
0013 C If GRAPH is out of range, an error message is printed and
0014 C no COMTAL transfer takes place.
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 GRAPH must be within the limits GRLD and GRIH explained below.
0023 C
0024 C***SUBPROGRAMS CALLED:
0025 C
0026 C name     source     load     remarks
0027 C ------     ------     ------     ------------------------------
0028 C RANGE &RANGE   %RANGE logical function that determines if its 1st
0029 C parameter is between (inclusive) its last parameters
0030 C CMND &CMND    %CMND sends a COMTAL command as if it were typed at
0031 C           the COMTAL keyboard.
0032 C DIGIT &DIGIT %DIGIT returns a character (1st parameter) which is the
0033 C           character equivalent of the integer 2nd parameter.
0034 C
0035 C***WRITTEN BY:
0036 C
0037 C The code on which this subroutine is based was written by
0038 C HETTIE 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
INTEGER GRLO,GRHI ! limits on COMTAL GRAPH numbers
CHARACTER*1 WHICH
LOGICAL RANGE ! logical function that determines if its
! 1st parameter lies within last 2 parameters.
INTEGER TERM ! logical unit number of the terminal output
INTEGER IBUF(128) ! COMTAL command buffer
CHARACTER*255 CBUF ! overlays the IBUF command buffer.
EQUIVALENCE (IBUF,CBUF)
C
***INITIALIZATIONS:
C
DATA GRLO/1/, GRHI/4/
DATA TERM/*
C
***PROCESSING
C
IF (.NOT.(RANGE(GRAPH,GRLO,GRHI))) GOTO 8001 ! error return
ELSE...clear the GRAPH
CALL DIGIT(WHICH, GRAPH)
CBUF = 'CLEAR GRAPH ' // WHICH ! since CBUF overlays IBUF,
! this statement loads IBUF with
! the COMTAL command
CALL CMMND(IBUF, 13)
RETURN
C
***ERROR RETURN
C
WRITE(TERM, 8003) GRAPH, GRLO, GRHI
FORMAT(' GRAPH NUMBER., I4, IS OUT OF RANGE:., 213, .')
WRITE(TERM, 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:
CLR 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.

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

SUBPROGRAMS CALLED:
name | source | load | remarks
--- | --- | --- | ---
RANGE | &RANGE | XRANGE | logical function that determines if its 1st parameter is between (inclusive) its last parameter
CMDND | &CMDND | XCMDND | sends a COMTAL command as if it were typed at the COMTAL keyboard.
DIGIT | &DIGIT | XDIGIT | 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 KETTIE D. FAULCON, July, 1983. This modification is by KEITH MILLER, June, 1984.

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

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

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

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

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

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

CMMN2 is designed to be sent constant strings.

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

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

The subroutine 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 preceeded 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

SUBPROGRAMS CALLED:
LEN integer function returns length of character string

**WRITTEN BY:**

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

**REVISION HISTORY:**

Modified by Keith Miller, 6/18/84.

**LOCAL VARIABLES:**

- CHARACTER*1 CBUF(256) | character buffer
- INTEGER IBUF(128) | the character buffer overlayed as integers
- INTEGER IZERO | constant value 0 for making a 0 byte (ZERO)
- INTEGER WORDS | counts number of words
- INTEGER NUMCHR | counts number of bytes
- INTEGER LEN | intrinsic HP FORTRAN77 function that returns the length of a character string.
- CHARACTER*1 BLANK, DOLLAR | special ASCII characters
- CHARACTER*1 ZERO | zero is 00000000 binary.
- EQUIVALENCE (CBUF,IBUF), (ZERO, IZERO)

**INITIALIZATIONS:**

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

NUMCHR = LEN(INBUF)

**PROCESSING:**

DO 10 INDEX = 1, NUMCHR
  CBUF(INDEX) = INBUF(INDEX:INDEX)
  CONTINUE
  IF (CBUF(NUMCHR) .EQ. BLANK) GOTO 30
  NUMCHR = NUMCHR + 1
  CBUF(NUMCHR) = BLANK
  C
  ADD ENDING CHARACTERS TO COMMAND

=30 CBUF(NUMCHR+1) = DOLLAR
  CBUF(NUMCHR+2) = ZERO
CBUF(NUMCHR+3) = ZERO  ! safety precaution

NUMCHR = NUMCHR + 2

CHANGE BYTE COUNT TO WORD COUNT

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

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

The first parameter to EXEC identifies it as a write command. The second parameter identifies the resident HP driver (36B) and gives a code for the operation required by this call (500B).

The third parameter is the command string, and the fourth gives the length in words of the buffer that is to be used. The fifth parameter is a code for the COMTAL interface that directs the command transfer.

CALL EXEC(2, 36B + 500B, IBUF, WORDS, 24001B)

RETURN

END
SUBROUTINE CMNND(INBUF, INCNT)

INTEGER INBUF(128) ! the characters of a COMTAL command string
INTEGER INCNT   ! the number of characters in command string

This subroutine "CMNND" 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 CMNND,
the final blank, $, and the required zero byte are added automatically
to the INBUF string.
If the last character isn't a blank, CMNND 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.

SUBPROGRAMS CALLED: NONE.

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

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

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

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

C***PROCESSING:
WORDS = (NUMCHR+1) / 2
DO 10 INDEX = 1, WORDS
   IBUF(INDEX) = INBUF(INDEX)
   CONTINUE
10   IF (CBUF(NUMCHR) .EQ. BLANK) GOTO 30
   ELSE...
   NUMCHR = NUMCHR + 1
   CBUF(NUMCHR) = BLANK
   ... ADD ENDING CHARACTERS TO COMMAND
   C
   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
`COUNT T=C0004 IS ON CRO0021 USING 00012 BLKS R=0000`

```fortran
SUBROUTINE COUNT(COUNTS, IMAGE)

PARAMETER DECLARATIONS:

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

INTRODUCTION:

The subroutine COUNT examines each pixel value in the COMTAL image associated with the number IMAGE, and compiles a count of how many pixels hold the values 0-255. These 256 counts are returned in the INTEGER*4 array COUNTS.

LANGUAGE:

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

LIMITATIONS:

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

SUBPROGRAMS CALLED:

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

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:

36`
C * LOCAL VARIABLES:

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

C **INITIALIZATIONS:

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

C **PROCESSING

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

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

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

RETURN

C **ERROR RETURN

WRITE(TERM, 8003) IMAGE, IMLO, IMHI
8003 FORMAT(' IMAGE NUMBER=', 14, ', OUT OF RANGE=', 214)
WRITE(TERM, 8901)
0091 8901 FORMAT( 'SUBROUTINE COUNT FAILS. COUNTS ARRAY NOT CHANGED.' )
0092     RETURN
0093     END
&DELAY T=00004 IS ON CR00021 USING 00005 BLKS R=0000

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

C

DATA TIMREQ/11/  
DATA TERM/1/  

C

C***PROCESSING:

C

IF (SECOND .LE. 0) GOTO 8001 ! error return

C

CALL EXEC(TIMREQ, INTIME)

INCNT = INTIME(1) + 100*INTIME(2) + 6000*INTIME(3)

1 + 360000*INTIME(4) + 360000*365*INTIME(5)

C

1000 CALL EXEC(TIMREQ, NOW)

NOWCNT = NOW(1) + 100*NOW(2) + 6000*NOW(3)

1 + 360000*NOW(4) + 360000*365*NOW(5)

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

C

RETURN

C

C***ERROR RETURN:

C

8001 WRITE(TERM, 8003) SECOND

8003 FORMAT(‘ THE SECONDS COUNT,’.IS,’ IS <= 0.’)

8006 8900 WRITE(TERM, 8901)

8901 FORMAT(‘ DELAY FAILS. NO TIMED DELAY OCCURS.’)

C

RETURN

C

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

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 CHARACTER FUNCTION DIGIT( INTIN )
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C**PARAMETER DECLARATIONS:
0006 C
0007 INTEGER INTIN ! INTEGER Input parameter, converted to a digit.
0008 C
0009 C**INTRODUCTION:
0010 C
0011 DIGITS converts an integer between 0 and 9 into a single
0012 character digit. If the INTIN parameter is out of range,
0013 an error message is printed at the terminal and DIGIT is
0014 set to a blank.
0015 C
0016 C**LANGUAGE:
0017 C
0018 FORTRAN 77, the HP-1000 version for RTE-6/VM.
0019 C
0020 C**LIMITATIONS:
0021 C
0022 The INTIN parameter must be between 0 and 9.
0023 C
0024 C**SUBPROGRAMS CALLED:
0025 C
0026 name source load remarks
0027 -------- -------- ---------------
0028 C
0029 C
0030 C
0031 RANGE / &RANGE XRANGE logical function determines if the first parameter
0032 lies within (inclusive) the next two parameters.
0033 C
0034 C
0035 C**WRITTEN BY:
0036 C
0037 C
0039 C
0040 C
0041 C
0042 C**REVISION HISTORY:
0043 C
0044 C
0045 C
0046 C
0047 C**LOCAL VARIABLES:
0048 C
0049 C
0050 CHARACTER*1 DGTARA(10) ! Digit ARRAY holds the digits '0'-'9'
0051 LOGICAL RANGE ! function that determines if its first parameter
0052 C
0053 INTEGER TERM ! logical unit number for terminal output.
0054 C
C**INITIALIZATIONS:

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

DATA TERM/1/

C**PROCESSING

$\text{IF (.NOT. (RANGE(INTH, 0, 9))) GOTO 8001} \text{ ! error return}$

ELSE... convert to digit and return

DIGIT = DGTARA(INTH + 1)
RETURN

C**ERROR RETURN

WRITE (TERM, 8003) INTH

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

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

DIGIT = \"\"
RETURN
END
SUBROUTINE DSPBW(IMAGE)

C***PARAMETER DECLARATIONS:

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

C***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.

C***LIMITATIONS:

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

C***SUBPROGRAMS CALLED:

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

C***WRITTEN BY:

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

KEETIE D. FAULCON, July, 1983. This modification is by

KEITH MILLER, June, 1984.

C***REVISION HISTORY:

C***LOCAL VARIABLES:

LOGICAL RANGE ! a function for determining if an integer is
Alien within a certain range
the logical unit number for the terminal
fills the place of an unused EXEC parameter

**INITIALIZATIONS:**

DATA TERM/1/

**PROCESSING**

IF (RANGE(IMAGE, 1, 4)) GOTO 2000 ! legal image number
ELSE... illegal image number
WRITE(TERM, 1001) IMAGE
FORMAT( 'The image number ', I3, ' is out of range.' )
WRITE(TERM, 1002)
FORMAT( 'DSPBW fails. No action taken on command.' )
RETURN

2000 CONTINUE ! send a display command to COMTAL

In the following call, the first parameter indicates a write
operation. The second parameter is a combination of two codes:
008B + 36B. 36B indicates the proper resident driver, and 008B
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, 008B + 36B, IDUMMY, 0, (IMAGE-1) * 2)
RETURN
END
&DSPCL T=00004 IS ON CR00021 USING 00006 BLKS R=0000

001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
002 SUBROUTINE DSPCL(RED, GREEN, BLUE, TCLR)
003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
004 C
005 C***INPUT PARAMETERS:
006 C
007 INTEGER RED, GREEN, BLUE | COMTAL image numbers for the 3 color
008 | components of the truecolor image to
009 | be displayed.
010 INTEGER TCLR | COMTAL image number for the truecolor
011 | image formed from RED, GREEN, and BLUE.
012 C
013 C***INTRODUCTION:
014 C
015 C "Display CoLor" commands the COMTAL to display one
016 C RGB true color image. Any previous function or
017 C pseudocolor memory commands are nullified. See the
018 C subroutine DSPBW to display black and white images.
019 C
020 C The color display is accomplished via the CMMND subroutine,
021 C which allows COMTAL commands to be sent to the COMTAL as if
022 C they were typed on the COMTAL keyboard.
023 C
024 C***LANGUAGE:
025 C
026 FORTRAN 77, the HP-100 version for RTE-6/VM.
027 C
028 C***LIMITATIONS:
029 C
030 C The three parameters RED, GREEN and BLUE must be distinct and
031 C within the IMLO, IMHI range. TCLR must be within the TRLO, TRHI
032 C range (which is mutually exclusive with IMLO-IMHI). The limits on
033 C truecolor numbers are arbitrary. However, this subroutine enforces
034 C the arbitrary limits. The HP image file must contain all three
035 C monochrome images in the order RED, GREEN, and BLUE.
036 C
037 C***SUBPROGRAMS CALLED:
038 C
039 C name source load remarks
040 ----------------------------------------------
041 RANGE &RANGE XRANGE logical function that determines if the 1st
042 C argument is within the 2nd and 3rd inclusive.
043 CMMD &CMMD XCMMD transfers a command string to the COMTAL, which
044 C accepts it almost as a keyboard command.

45
**DIGIT &DIGIT &DIGIT** character*1 function that returns a single digit on legal integer inputs 0-9.

**WRITTEN BY:**

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

**REVISION HISTORY:**

0858 **LOCAL VARIABLES:**

**INITIALIZATIONS:**

**PROCESSING**

**DISPLAY THE COLOR IMAGE**

The following commands are abbreviations of the following:
COMTAI 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 CMND(IBUF,6)
CBUF = 'AS T' /DIGIT(TCLR)' /DIGIT(RED)' /DIGIT(BLUE)
1 /DIGIT(GREEN)' /DIGIT(BLUE)
CALL CMND(IBUF,12)
CBUF = 'D I' /DIGIT(TCLR)
CALL CMND(IBUF,5)
RETURN

**ERROR RETURNS

8001 WRITE(TERM, 8003) RED, IMLO, IMHI
8003 FORMAT(' RED IMAGE NUMBER','.13', ' IS OUT OF RANGE:'.214').
8010 GOTO 8900
8011 C
8012 8101 WRITE(TERM, 8103) GREEN, IMLO, IMHI
8103 FORMAT(' GREEN IMAGE NUMBER','.13', ' IS OUT OF RANGE:'.214').
8110 GOTO 8900
8111 C
8112 8201 WRITE(TERM, 8203) BLUE, IMLO, IMHI
8203 FORMAT(' BLUE IMAGE NUMBER','.13', ' IS OUT OF RANGE:'.214').
8210 GOTO 8900
8211 C
8301 WRITE(TERM, 8303) RED, GREEN, BLUE
8303 FORMAT(' 3 MONOCROME IMAGES MUST BE DISTINCT. YOURS:',.314)
8310 GOTO 8900
8311 C
8401 WRITE(TERM, 8403) TCLR, CLLO, CLHI
8403 FORMAT(' YOUR TRUECOLOR IMAGE','.14', ' IS OUT OF RANGE:'.214)
8410 GOTO 8900
8411 C
8500 WRITE(TERM, 8500)
8501 FORMAT(' DSPLC FAILS. NO DISPLAY TAKES PLACE. ')
SUBROUTINE DSPGR(GRNUM)

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

"Display Graphics" is a subroutine that allows the caller to send a display command to the COMTAL from an HP program.

The call can turn on one graphics plane, number 1, 2, 3, or 4.

This subroutine does not turn off previous graphics planes.

That is accomplished using the subroutine WIPGR.

It is OK to call DSPGR repeatedly without an intervening WIPGR.

The extra calls have no effect, but they don't hang up the COMTAL.

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

*CMMN2 &CMMN2 *CMMN2 sends a constant string to the COMTAL as if the string were typed on the COMTAL keyboard.

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

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

*LOCAL VARIABLES:
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 IDUMITY  ! 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  DATA TERM/1/
C  DATA GRLD/1/,GRHI/4/

C***PROCESSING

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

C  "ADD Graphics *GRNUM", where *GRNUM stands for the digit
C  corresponding to GRNUM value.

C  CALL CMN2(‘ADD GR */DIGIT(GRNUM))

RETURN

C***ERROR RETURN:

C  WRITE(TERM, 8003) GRNUM, GRLD, GRHI
C  8003 FORMAT( ‘ THE GRAPHICS NUMBER,’.13.’, OUT OF RANGE:’,214)

C  WRITE(TERM, 8901)
C  8901 FORMAT( ‘ DSPGR fails. No action taken on command.’ )

RETURN

END
SUBROUTINE DSPVD

C***INPUT PARAMETERS:
C
C None.
C
C***INTRODUCTION:
C
"Display Video" commands the COMTAL to display the video image
which, by arbitrary convention, is always assigned to COMTAL
image #5. The video must be set to image 5 previous to this
call.

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

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

C***SUBPROGRAMS CALLED:
C
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*R1 function that returns '0'-'9'
according to integer input 0-9.

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

C***LOCAL VARIABLES:
INTEGER IBUF(256) ! buffers COMTAL command
0046 CHARACTER*255 CBUF ! character overlay for sending COMTAL commands
0047 EQUIVALENCE (CBUF, IBUF)
0048 INTEGER TVIMA6 ! COMTAL image _ for video camera
0049 CHARACTER*1 DIGIT ! function that returns '0'-'9' according
0050 C ! to integer input 0-9.
0051 C
0052 C***INITIALIZATIONS:
0053 C
0054 DATA TVIMA5/ ! arbitrary choice.
0055 C
0056 C***PROCESSING
0057 C
0058 C DISPLAY THE VIDEO IMAGE
0059 C
0060 CBUF = 'DISPLAY IMAGE ' //DIGIT(TVIMA5)/' '
0061 CALL CMND(IBUF, 16)
0062 C
0063 RETURN
0064 END
&HILO T=00004 IS ON CR00021 USING 00004 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE HILO(HI, LO, IMAGE)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***PARAMETER DECLARATIONS:
0006 C
0007 INTEGER HI, LO ! output parameters, the high and low pixel values
0008 C          ! found in the designated image.
0009 INTEGER IMAGE ! the COMTAL image number of the image that is to
0010 C          ! be searched for its high and low pixel values.
0011 C
0012 C***INTRODUCTION:
0013 C
0014 C The subroutine "High and Low values #2" reads through an image and
0015 C determines the highest and lowest pixel values, returning the
0016 C values found. HILO scans the entire image.
0017 C
0018 C***LANGUAGE:
0019 C
0020 C FORTRAN 77, the HP-1000 version for RTE-6/VM.
0021 C
0022 C***LIMITATIONS:
0023 C
0024 C As noted above, this subroutine passes through an entire image, pixel
0025 C by pixel. In some applications you may want to combine other processing
0026 C during that pass, but this subroutine won't let you do that.
0027 C
0028 C***SUBPROGRAMS CALLED:
0029 C
0030 C name source load remarks
0031 C ------- ------ --------- ----------------------------------------
0032 C RANGE   RANGE   XRANGE logical function that determines if 1st argument
0033 C          ! is within 2nd & 3rd inclusive.
0034 C RDIL2   RDIL2   %RDIL2 reads a horizontal line of pixel values into
0035 C          ! a 512 integer array.
0036 C
0037 C***WRITTEN BY:
0038 C
0039 C The code on which this subprogram is based was written by
0040 C NETTIE D. FAULCON, July, 1983. This subprogram was written by
0041 C KEITH MILLER, July, 1984, with the support of a NASA-ASEE
0042 C summer fellowship.
0043 C
0044 C***REVISION HISTORY:
C**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.

C**INITIALIZATIONS

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

C**PROCESSING

IF (.NOT.(RANGE(IMAGE, IMLO, IMHI))) GOTO 8801 ! 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)
2000 CONTINUE
2001 RETURN

GOTO 8900

C

C 8801 WRITE(TERM,8803)IMAGE ,IMLO,IMHI
8803 FORMAT(IMAGE NUMBER,'.15, IS OUT OF RANGE:' ,215,'.')
8900 GOTO 8800

C

C 8900 WRITE(TERM,8901)
8901 FORMAT(' HILO FAILS, HI AND LO PARAMETERS UNCHANGED.')
8999 RETURN

C

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

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE HISTO(IMAGE)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
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 XCMMN2 sends a constant string to the COMTAL, which
0039 C treats it as a command typed on the COMTAL.
0040 C DELAY &DELAY XDELAY puts the HP into a busy wait for at least the
0041 C given number of seconds.
0042 C DIGIT &DIGIT XDIGIT character*1 function that returns '0'-'9'
0043 C according to integer input 0-9.
0044 C RANGE &RANGE 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:

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

INITIALIZATIONS:

DATA 'IMLO/,IMHI/'
DATA 'TERM/'

PROCESSING:

IF (.NOT.(RANGE(IMAGE,IMLO,IMHI))) GOTO 8881 ! 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 #1 = Histogram of image
where #1 is the single digit associated with IMAGE.
Image #1 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 CMMN2('D F '//'IMCHAR')! "Display Function memory #I"
RETURN

C

***ERROR RETURNS:
C
C
6001 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  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004  C
0005  C***INPUT PARAMETERS:
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:
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:
0027  C ICOPY only works for the grey level images of COMTAL, not the
0028  C truecolor images.
0029  C
0030  C***SUBPROGRAMS CALLED:
0032  C
0033  C name  source  load  remarks
0034  C CMNDCMND  &CMNDCMND  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 RANGERANGE  &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 DIGITDIGIT  &DIGITDIGIT  character*1 function that returns a single digit
0042  C '0'-'9' according to integer input 0-9.
0043  C
0044  C***WRITTEN BY:
8045 C
8046 C The code on which this subroutine is based was written by
8047 C NETTIE D. FAULKON in July, 1983. This modification is by
8048 C KEITH MILLER June, 1984.
8049 C
8050 C****REVISION HISTORY:
8051 C
8052 C
8053 C****LOCAL VARIABLES:
8054 C
8055 C LOGICAL RANGE function determines if 1st argument is
8056 C within 2nd and 3rd argument inclusive.
8057 C CHARACTER*1 DIGIT function returns '0'-'9' according to
8058 C integer input 0-9.
8059 C CHARACTER*255 CCOMM character buffer for building up a call
8060 C to the CMDND subroutine.
8061 C INTEGER IBUF (128) integer overlay of CCOMM
8062 C EQUIVALENCE (CCOMM, IBUF)
8063 C
8064 C INTEGER IMLO, IMHI the range of legal COMTAL image numbers
8065 C INTEGER TERM terminal logical unit
8066 C
8067 C****INITIALIZATION:
8068 C
8069 C DATA IMLO/1/, IMHI/4/
8070 C DATA TERM/1/
8071 C
8072 C****PROCESSING:
8073 C
8074 C IF (.NOT.(RANGE(OUTIMG, IMLO, IMHI))) GOTO 8001 error return
8075 C IF (.NOT.(RANGE(INIMG, IMLO, IMHI))) GOTO 8101 error return
8076 C
8077 C Legal image numbers, so do the copy
8078 C
8079 C 2000 CCOMM = 'IMAGE '//'DIGIT(OUTIMG)'' = IMAGE '//'DIGIT(INIMG)
8080 C
8081 C PROGRAMMING NOTE: see COMTAL USER'S GUIDE for
8082 C further information on the command string abbreviated
8083 C in the string above.
8084 C
8085 C CALL CMDND( IBUF, 17 ) sends copy command to COMTAL
8086 C RETURN
8087 C
8088 C****ERROR RETURNS:
8089 C
8090 C 8001 WRITE(TERM, 8003)OUTIMG, IMLO, IMHI
0091 803 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
&ICPY2 T=00004 IS ON CRO0021 USING 00018 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE ICPY2(OUTIMG, INIMG)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***INPUT PARAMETERS:
0006 C
0007 INTEGER OUTIMG ! COMTAL image number for the destination
0008 INTEGER INIMG ! COMTAL image number for the source
0009 C
0010 C***INTRODUCTION:
0011 C
0012 C "Image Copy #2" commands the COMTAL to copy one black and
0013 C white image into another. The OUTIMG destination image
0014 C is, of course, wiped out by this exchange. The input
0015 C image for ICPY2 can be any single digit number; the output
0016 C image must be a COMTAL image memory plane (1-4 currently).
0017 C The companion subroutine 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 CMIND &CMIND &CMIND 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. FAULKON 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 `CMD` 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, 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 CMD( IBUF, 17 ) ! sends copy command to COMTAL`

**ERROR RETURNS:**

- `8001 WRITE(TERM, 8003)OUTIMG, IMLO, IMHI`
- `8003 FORMAT(' YOUR OUTPUT PARAMETER,'$, IS,'$, IS OUT OF RANGE:',214)`
0091      GOTO 8900
0092  C
0093  8101 WRITE(TERM, 8103) IN IMG
0094  8103 FORMAT(' YOUR INPUT PARAMETER,’, 15,’ IS OUT OF RANGE: 1, 9’)
0095      GOTO 8900
0096  C
0097  8900 WRITE(TERM, 8901)
0098  8901 FORMAT(' ICPY2 FAILS. NO COPYING TAKES PLACE.’)
0099      RETURN
0100      END
&MERGE T=00004 IS ON CR00021 USING 00005 BLKS R=0000

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
SUBROUTINE MERGE(OUTURD, BYTE1, BYTE2)

1CCCCCCCC
C
C**PARAMETER DECLARATIONS:

C
C
CINTEGER OUTURD ! the output, the 2 input bytes merged into 1 integer
CINTEGER BYTE1 ! the left, high order input byte (in lower byte),
CINTEGER BYTE2 ! the right, low order input byte (in lower byte).

C
C**INTRODUCTION:

C
C  The subroutine MERGE takes two integer inputs which should be non-zero
C  in the lower byte only, and merges these lower order bytes into a single
C  integer output.

C
C**LANGUAGE:

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

C
C**LIMITATIONS:

C
C  Note that MERGE does not check that the upper order byte of the
C  input INTEGERS are zeros. This check could be added, but will slow
C  down MERGE execution. This speed is important, since MERGE was designed
C  to be a very low level routine.

C
C**SUBPROGRAMS CALLED:

C
C  none.

C
C**WRITTEN BY:

C
C  The code on which this subprogram is based was written by
C  NETTIE D. FAULCON, July, 1982. 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 HOLD ! an INTEGER interpretation of bits
C  CHARACTER CHOLD(2) ! a CHARACTER interpretation of bits
INTEGER IMERGE ! an INTEGER interpretation of bits
CHARACTER CSPLIT(2) ! a CHARACTER interpretation of bits

EQUVALENCE (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=004 IS ON CR00021 USING 0004 BLKS R=0000

SUBROUTINE NORMAL (IMAGE)

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

SUBROUTINE HORML (IMAGE)

INTRODUCTION:

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

The "normalized" background image is subtracted from the digitized image to simulate a uniform background.

LANGUAGE:

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

LIMITATIONS:

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

SUBPROGRAMS CALLED:

name source load remarks

CDI2 CDI2 CDI2 reads one horizontal COMTAL image line into an integer array, one integer/pixel.

URIL2 URIL2 URIL2 writes one horizontal COMTAL image line from an integer array, one integer/pixel.

HILO HILO HILO determines the highest and lowest pixel value in an image.
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 pixel values: one pixel / integer
INTEGER ARALO, ARAHI! array bounds for a pixel buffer
INTEGER LNLO, LNHI ! limits on COMTAL row numbers
INTEGER HI, LO ! highest and lowest pixel values in IMAGE
INTEGER ROW, COL ! loop indices
```

**INITIALIZATIONS:**

```
DATA ARALO/1/, ARAHI/512/
DATA LNLO/0/, LNHI/511/
```

**PROCESSING**

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

```
DO 2000 ROW = LNLO, LNHI
   CALL RDIL2(IBUF, IMAGE, ROW)
   DO 1000 COL = ARALO, ARAHI
      IBUF(COL) = IBUF(COL) - LO
   1000 CONTINUE
   CALL WRIL2(IMAGE, ROW, IBUF)
2000 CONTINUE
RETURN
END
```
&NOTE2 T=00004 IS ON CR0021 USING 00024 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE NOTE2(GRNUM, XCOORD, YCOORD, COLOR, FACTOR, NOTE)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***PARAMETER DECLARATIONS:
0006 C
0007 C INTEGER GRNUM | the number of the graphic to be written into;
0008 C INTEGER XCOORD  |
0009 C 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 C CHARACTER*1 NOTE | signals if you wish to stay the same color ('S')
0013 C CHARACTER*1 NOTE | or change to red ('R'), green('G'), blue('B'),
0014 C CHARACTER*1 NOTE | black('K'), white('W'), or yellow('Y').
0015 C CHARACTER*(*NOTE | the message to be printed in graphics; NOTES
0016 C INTEGER FACTOR  | designed for constant strings.
0017 C INTEGER COLOR   |
0018 C
0019 C***INTRODUCTION:
0020 C
0021 C This subroutine, NOTE2 writes a line of characters into a COMTAL graphics
0022 C plane. The parameter NOTE should be a constant string.
0023 C A very similar subroutine, NOTES, uses a string variable and a length
0024 C parameter instead of the constant string.
0025 C
0026 C***LANGUAGE:
0027 C
0028 C FORTRAN 77, the HP-1000 version for RTE-6/VM.
0029 C
0030 C***LIMITATIONS:
0031 C
0032 C GRNUM must be in the range 1-4 inclusive.
0033 C XCOORD and YCOORD must be in the range 0-511.
0034 C COLOR must be one of the following: 'S' (for "Same color"),
0035 C 'R' (for "Red"), 'G' (for "Green"), 'B' (for "Blue"),
0036 C 'K' (for "black"), 'W' (for "White") or 'Y' (for "Yellow").
0037 C Other colors could be easily added. See TABLE 1 in the NOTES
0038 C section for other colors.
0039 C FACTOR must be in the range 1-16 inclusive.
0040 C If a restriction is violated, NOTES fails with an error message.
0041 C
0042 C The graphics plane named here is turned on and all other graphics
0043 C planes are turned off when this subroutine is executed.
0044 C The GRNUM plane is NOT cleared.
The length of time it takes the COMTAL to write a note in graphics causes a timing problem; the COMTAL may ignore the next COMTAL command sent from the HP. Therefore, we DELAY the HP for a number of seconds proportional to the size of the NOTE characters.

### SUBPROGRAMS CALLED:

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

### WRITTEN BY:

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

### LOCAL VARIABLES:

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

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 8081 ! error return
IF (.NOT.(RANGE(XCOORD, CMLO, CMHI))) GOTO 8181 ! error return
IF (.NOT.(RANGE(YCOORD, CMLO, CMHI))) GOTO 8281 ! error return
IF (.NOT.(RANGE(FACTOR, FCLO, FCHI))) GOTO 8381 ! error return

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

IF ((COLOR .EQ. 'S').OR.(COLOR .EQ. 's')) GOTO 1000 ! "Same"
IF ((COLOR .EQ. 'R').OR.(COLOR .EQ. 'r')) GOTO 1000 ! "Red"
IF ((COLOR .EQ. 'G').OR.(COLOR .EQ. 'g')) GOTO 1000 ! "Green"
IF ((COLOR .EQ. 'B').OR.(COLOR .EQ. 'b')) GOTO 1000 ! "Blue"
IF ((COLOR .EQ. 'K').OR.(COLOR .EQ. 'k')) GOTO 1000 ! "black"
IF ((COLOR .EQ. 'W').OR.(COLOR .EQ. 'w')) GOTO 1000 ! "White"
IF ((COLOR .EQ. 'Y').OR.(COLOR .EQ. 'y')) GOTO 1000 ! "Yellow"
ELSE...COLOR an illegal character
GOTO 8481 ! error return

1000 CALL CMMN2('CO G ' //GRCHAR// ' RED')
1000 GOTO 2000

Color graphics green
1000 CALL CMMN2('CO G ' //GRCHAR// ' GRN')
1000 GOTO 2000

Color graphics blue
1200 CALL CMMN2('CO G ' //GRCHAR// ' BLU')
GOTO 2000
0138 C
0139 C Color graphics black
0140 1300 CALL CMNN2('CO G' //GRCHAR//' BLA')
0141 GOTO 2000
0142 C
0143 C Color graphics white
0144 1400 CALL CMNN2('CO G' //GRCHAR//' WHT')
0145 GOTO 2000
0146 C
0147 C Color graphics white
0148 1500 CALL CMNN2('CO G' //GRCHAR//' YEL')
0149 GOTO 2000
0150 C
0151 2000 CALL DSPGR(GRNUM)  //display the chosen graphics
0152 CALL WRTAR(XCOOR, YCOOR)  //position the cursor for writing
0153 C
0154 LONG = LEN(NOTE)
0155 IF (FACTOR .GE. 10) GOTO 3000
0156 C ELSE...
0157 CBUF(1:8) = 'G' //GRCHAR//' L' //DIGIT(FACTOR)//'
0158 START = 9
0159 LONG = LONG + 8
0160 GOTO 4000
0161 C THEN...
0162 3000 CBUF(1:9) = 'G' //GRCHAR//' L 1' //DIGIT(FACTOR-10)//'
0163 START = 10
0164 LONG = LONG + 9
0165 GOTO 4000
0166 C
0167 4000 CBUF(START:LONG) = NOTE
0168 CALL CMNND(IBUF, LONG)
0169 C
0170 C Put the HP in a busy wait while the COMTAL writes the note.
0171 CALL DELAY((FACTOR/4)+1)
0172 RETURN
0173 C
0174 C***ERROR RETURNS:
0175 C
0176 0600 WRITE(TERM, 8003) GRNUM, GRLO, GRAH
0177 8003 FORMAT(' THE GRAPHICS NUMBER','.15',' IS OUT OF RANGE: 214.')
0178 GOTO 8000
0179 C
0180 8100 WRITE(TERM, 8103) XCOOR, CMLO, CMHI
0181 8103 FORMAT(' THE X COORDINATE','.15',' IS OUT OF RANGE: 214.')
0182 GOTO 8000

70
0183 C
0184 8201 WRITE(TERM.8203) YCOORD,CMLO,CMHI
0185 8203 FORMAT( ' THE Y COORDINATE, ', I5, ' IS OUT OF RANGE: ', 214, '.' )
0186       GOTO 8900
0187 C
0188 8301 WRITE(TERM.8303) FACTOR,FCLO,FCHI
0189 8303 FORMAT( ' THE SCALE FACTOR, ', I5, ' 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(什NUM, 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     ! NOTES doesn’t clear or display this graphic.
0009 INTEGER XCOORD !
0010 INTEGER YCOORD ! XCOORD and YCOORD define the upper left corner of the
0011 C     ! position of the first character in the NOTE.
0012 INTEGER FACTOR ! controls size of characters plotted: 1-16.
0013 CHARACTER*1 COLOR ! signals if you wish to stay the same color ("S")
0014 C     ! or change to red ("R"), green ("G"), or blue ("B").
0015 CHARACTER*255 NOTE ! the message to be printed in graphics is in the
0016 C     ! first LENGTH characters of this string.
0017 INTEGER LENGTH ! the number of NOTE characters that are to be used.
0018 C
0019 C***INTRODUCTION:
0020 C
0021 C This subroutine, NOTES writes a line of characters into a COMTAL graphics
0022 C plane. The parameter NOTE should be a declared string.
0023 C A very similar subroutine, NOTE2, uses a string constant without a length
0024 C parameter.
0025 C
0026 C***LANGUAGE:
0027 C
0028 C FORTRAN 77, the HP-1000 version for RTE-6/VM.
0029 C
0030 C***LIMITATIONS:
0031 C
0032 C GRNUM must be in the range 1-4 inclusive.
0033 C 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
CAUSE 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.

**SUBPROGAMS 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>%CMMN2</td>
<td>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>%CMMND</td>
<td>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>%DELAY</td>
<td>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>%DIGIT</td>
<td>CHARACTER FUNCTION THAT RETURNS '0'- '9' ACCORDING TO INTEGER INPUT 0-9.</td>
</tr>
<tr>
<td>DSPGR</td>
<td>&amp;DSPGR</td>
<td>%DSPGR</td>
<td>ADDS A GRAPHIC PLANE TO THE DISPLAY.</td>
</tr>
<tr>
<td>RANGE</td>
<td>&amp;RANGE</td>
<td>%RANGE</td>
<td>LOGICAL FUNCTION THAT DETERMINES IF THE 1ST ARGUMENT IS WITHIN THE 2ND &amp; 3RD INCLUSIVE.</td>
</tr>
</tbody>
</table>

**WRITTEN BY:**

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

**REVISION HISTORY:**

**LOCAL VARIABLES:**

<table>
<thead>
<tr>
<th>Character</th>
<th>DIGIT</th>
<th>FUNCTION THAT RETURNS '0'- '9' FOR INTEGER INPUT 0-9.</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOGICAL</td>
<td>RANGE</td>
<td>FUNCTION THAT DETERMINES IF 1ST ARGUMENT IS WITHIN 2ND AND 3RD ARGUMENTS, INCLUSIVE.</td>
</tr>
<tr>
<td>CHARACTER</td>
<td>GRCHAR</td>
<td>THE SINGLE DIGIT THAT CORRESPONDS TO GRNUM.</td>
</tr>
<tr>
<td>INTEGER</td>
<td>LONG</td>
<td>THE LENGTH OF THE CMMND COMMAND.</td>
</tr>
<tr>
<td>INTEGER</td>
<td>GRLO,GRHI</td>
<td>LIMITS FOR COMTAL GRAPHICS PLANE NUMBER.</td>
</tr>
<tr>
<td>INTEGER</td>
<td>CMLO,CMHI</td>
<td>LIMITS ON COMTAL COORDINATES.</td>
</tr>
<tr>
<td>INTEGER</td>
<td>FCLO,FCHI</td>
<td>LIMITS ON FACTOR.</td>
</tr>
<tr>
<td>INTEGER</td>
<td>STLO,STHI</td>
<td>LIMITS ON STRING LENGTH.</td>
</tr>
<tr>
<td>INTEGER</td>
<td>TERM</td>
<td>LOGICAL UNIT FOR TERMINAL OUTPUT.</td>
</tr>
<tr>
<td>CHARACTER</td>
<td>255 CBUF</td>
<td>BUFFER FOR CMMND COMMANDS</td>
</tr>
</tbody>
</table>
INTEGER IBUF(128) ! overlays CBUF
EQUIVALENCE (CBUF, IBUF)

**C***INITIALIZATIONS:

**DATA**

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

**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(FACTDR, FCLO, FCHI))) GOTO 8301 ! error return**
**IF (.NOT.(RANGE(LENGTH, STLO, STHI))) GOTO 8401 ! error return**

**GRCHAR = DIGIT(_RNUM); ! initialization delayed until _RNUM checked.**

**IF ((COLOR, .EQ. 'S').OR.(COLOR .EQ. 'S')) GOTO 2000 ! "Same"**
**IF ((COLOR, .EQ. 'R').OR.(COLOR .EQ. 'R')) GOTO 1000 ! "Red"**
**IF ((COLOR, .EQ. 'G').OR.(COLOR .EQ. 'G')) GOTO 1100 ! "Green"**
**IF ((COLOR, .EQ. 'B').OR.(COLOR .EQ. 'B')) GOTO 1200 ! "Blue"**
**ELSE...COLOR an illegal character**

**GOTO 6501 ! error return**

**CALL CMMN2('CO G '//GRCHAR// ' RED')**
**GOTO 2000**

**CALL CMMN2('CO G '//GRCHAR// ' GRN')**
**GOTO 2000**

**CALL CMMN2('CO G '//GRCHAR// ' BLU')**
**GOTO 2000**

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

**IF (FACTDR .GE. 10) GOTO 3000**
**ELSE...**

**CBUF = 'G '//GRCHAR// ' L '//DIGIT(FACTDR) // ' NOTE**
0137    LONG = LENGTH + 8
0138    GOTO 4000
0139    C   THEN...
0140    3000    CBUF = 'G'/'GRCHAR//' L 1//'DIGIT(FACTOR-10)//' '///NOTE
0141    LONG = LENGTH + 9
0142    GOTO 4000
0143    C
0144    4000    CALL CMD(INDBUF,LENGTH)
0145    C
0146    C   Delay for a second or two to relieve timing problem between
0147    C   the HP and the COMTAL
0148    CALL DELAY((FACTOR/4)+1)
0149    RETURN
0150    C
0151    C***ERROR RETURNS:
0152    C
0153    8001    WRITE(TERM,8003)GRNUM,GRLO,GRHI
0154    8003    FORMAT(' THE GRAPHICS NUMBER.',I5,' IS OUT OF RANGE:',214,'.')
0155    GOTO 8900
0156    C
0157    8101    WRITE(TERM,8103)XCOORD,CMLO,CMHI
0158    8103    FORMAT(' THE X COORDINATE.',I5,' IS OUT OF RANGE:',214,'.')
0159    GOTO 8900
0160    C
0161    8201    WRITE(TERM,8203)YCOORD,CMLO,CMHI
0162    8203    FORMAT(' THE Y COORDINATE.',I5,' IS OUT OF RANGE:',214,'.')
0163    GOTO 8900
0164    C
0165    8301    WRITE(TERM,8303)FACTOR,FCLO,FCHI
0166    8303    FORMAT(' THE SCALE FACTOR.',I5,' IS OUT OF RANGE:',214,'.')
0167    GOTO 8900
0168    C
0169    8401    WRITE(TERM,8403)LENGTH,STLO,STHI
0170    8403    FORMAT(' THE STRING LENGTH.',I5,' IS OUT OF RANGE:',214,'.')
0171    GOTO 8900
0172    C
0173    8501    WRITE(TERM,8503)COLOR
0174    8503    FORMAT(' THE COLOR PARAMETER.',A1,' IS NOT S, R, G, OR B.')
0175    GOTO 8900
0176    C
0177    8900    WRITE(TERM,8901)
0178    8901    FORMAT(' NOTES FAILS. NO LETTERING PLACED INTO GRAPHICS.')
0179    C
0180    RETURN
0181    END
&PAINT T=00004 IS ON CR00021 USING 00004 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE PAINT(IMAGE, BRUSH, SHADE)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***PARAMETER DECLARATIONS:
0006 C
0007 INTEGER IMAGE ! number of the COMTAL image to be painted
0008 INTEGER BRUSH ! size of the square brush area
0009 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 ------- ------- -------- -------------------------------
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 RDTAR &RDTAR &RDTAR reads the current COMTAL cursor position.
0040 C WAIT &WAIT &WAIT &WAIT halts HP processing until HP <CR> entered.
0041 C CMIND &CMIND &CMIND sends a character string to COMTAL, which
0042 C treats the string as a COMTAL keyboard command.
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 BOX(64*64) ! the square that acts as a paint brush-
  - INTEGER XPOS ! this rectangle placed into IMAGE whenever
  - INTEGER UPLFX, UPLFY ! <CR> entered to locate cursor on COMTAL.
  - INTEGER IMLO, IMHI ! loop index for initializing BOX.
  - INTEGER PXLO, PXHI ! X and Y coordinates of last COMTAL cursor
  - INTEGER BRLO, BRHI ! (target) position recorded via RDTAR;
  - INTEGER ,TERM ! used as upper left corner of rectangle
  - INTEGER IBUF(128) ! to be painted.
  - CHARACTER*1 INCHAR ! logical unit for terminal I/O.
  - INTEGER IMLO, IMHI ! limits on COMTAL image number.
  - INTEGER PXLO, PXHI ! limits on COMTAL pixel values.
  - INTEGER BRLO, BRHI ! limits on size of brush for painting.
  - CHARACTER*1 INCHAR ! character buffer for HP keyboard input.
  - LOGICAL RANGE ! function that determines if 1st argument
  - INTEGER IBUF(128) ! is within the 2nd and 3rd, inclusive.
  - CHARACTER*255 CBUF ! buffer for sending CMNND strings
  - EQUIVALENCE (IBUF,CBUF) ! character overlay for IBUF

**Initialization:**

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

**Processing:**

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

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

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

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

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

add the target and dump the image on the COMTAL

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

Give instructions to user

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

Loop for input/painting starts here:

1000 INCHAR = ' ' 
READ(TERM, 1001) INCHAR
1001 FORMAT(IA1)
1020 IF (INCHAR .EQ. ' ') GOTO 2000 ! paint another square and continue.
1021 IF ((INCHAR .EQ. 'S') .OR. (INCHAR .EQ. 's')) GOTO 9000 ! terminate
1022 ELSE...illegal entry
1023 WRITE(TERM, 1003) INCHAR
1003 FORMAT(' THE CHARACTER ENTERED, ', 1A1, ',', ' IS NOT LEGAL'.
  ' FOR PAINTING. '/, ' PLEASE TRY AGAIN.')
1026 GOTO 1000
1027

Actual painting takes place here:

2000 CALL RDTAR(UPLFX, UPLFY)
2010 CALL WRIRC(IMAGE, UPLFX, UPLFY, BOX, BRUSH, BRUSH)
2012 GOTO 1000
2033

***ERROR RETURNS:

8001 WRITE(TERM, 8003) IMAGE, IMLO, IMHI
0137  0003 FORMAT(' IMAGE NUMBER.', 15, ' OUT OF RANGE:', 214, ' .')
0138              GOTO 8900
0139 C
0140  8101 WRITE(TERM, 8103) BRUSH, BRLO, BRHI
0141  8103 FORMAT(' BRUSH ARGUMENT.', 15, ' OUT OF RANGE:', 214, ' .')
0142              GOTO 8900
0143 C
0144  8201 WRITE(TERM, 8203) SHADE, PXLO, PXSH
0145  8203 FORMAT(' SHADE ARGUMENT.', 15, ' OUT OF RANGE:', 214, ' .')
0146              GOTO 8900
0147 C
0148  8900 WRITE(TERM, 8901)
0149  8901 FORMAT(' PAINT SUBROUTINE FAILS.')
0150 C
0151  9000 CONTINUE
0152              RETURN
0153  9003 FORMAT(' PAINT SUBROUTINE FAILS.')
0154              RETURN
0155 END
0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE PROFL(GRAPH, IMAGE)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***PARAMETER DECLARATIONS:
0006 C
0007 INTEGER GRAPH ! the number of the COMTAL graphics plane in which
0008 C ! the profile is to be displayed.
0009 INTEGER IMAGE ! the number of the COMTAL image which is going to
0010 C ! be "profiled."
0011 C
0012 C***INTRODUCTION:
0013 C
0014 C The subroutine PROFL gives HP access to the COMTAL's interactive
0015 C profiling facilities. This subroutine initializes the COMTAL for
0016 C taking profiles of IMAGE and displaying them in the specified GRAPH
0017 C plane. Note that this subroutine clears the specified graphics plane.
0018 C Control passes to COMTAL for profiling, and then a <CR> on the HP
0019 C terminal restores control to the HP terminal.
0020 C
0021 C***LANGUAGE:
0022 C
0023 C FORTRAN 77, the HP-1B version for RTE-E/VM.
0024 C
0025 C***LIMITATIONS:
0026 C
0027 C GRAPH and IMAGE are checked, and if they are out of bounds, no
0028 C profiling takes place and an error message is printed. The shift
0029 C from the HP terminal to the COMTAL terminal may result in some
0030 C confusion, but this switch exploits the COMTAL circuitry much more
0031 C efficiently than would be possible using the HP terminal alone.
0032 C
0033 C***SUBPROGRAMS CALLED:
0034 C
0035 C name source load remarks
0036 C ------- ------ ------ -----------------------------------------------
0037 C RANGE &RANGE &RANGE logical function that determines if 1st argu-
0038 C ment is within 2nd and 3rd, inclusive.
0039 C CMNND &CMNND &CMNND sends a command to the COMTAL as if it were
0040 C typed at the COMTAL keyboard.
0041 C WAIT &WAIT &WAIT halts HP processing until <CR> is pressed on
0042 C the HP keyboard.
0043 C CHAR ------- ------- intrinsic HP FORTRAN77 function that converts
0044 C integers into characters
C  DIGIT &DIGIT &DIGIT character*1 function that returns '0'..'9'
   according to integer input 0..9.

C**WRITTEN BY:

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

C***REVISION HISTORY:

C

C***LOCAL VARIABLES:

C

C CHARACTER*1 GCHAR ! the character that corresponds to the single
   ! digit argument, GRAPH
C CHARACTER*1 ICHAR ! the character that corresponds to the single
   ! digit argument, IMAGE
C CHARACTER*1 BELL ! the ASCII code for ESCAPE.
C CHARACTER*1 DIGIT ! function that returns '0'..'9' according to
   ! integer argument 0..9.
C CHARACTER CHAR ! intrinsic FORTRAN77 function for integer to
   ! character conversion.
C INTEGER IBUF(128) ! integer buffer for sending COMTAL commands
C CHARACTER*255 CBUF ! overlays IBUF
C EQUIVALENCE (IBUF,CBUF)
C
C INTEGER GRLO, GRHI! limits on numbers of graphics planes in COMTAL
C INTEGER IMLO, IMHI! limits on numbers of image planes in COMTAL
C LOGICAL RANGE ! logical function that determines if 1st argu-
   ! ment is within 2nd and 3rd argument, inclusive.
C INTEGER TERM ! logical unit for terminal output

C**INITIALIZATIONS:

C

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

C**PROCESSING

C

C IF (.NOT. (RANGE(GRAPH,GRLO,GRHI))) GOTO 8801
C IF (.NOT. (RANGE(IMAGE,IMLO,IMHI))) GOTO 8101

C
set up COMTAL display for profiling

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

the following COMTAL command reads as follows (*G stands for
the single digit associated with GRAPH; and *I, with IMAGE):
Display Image *I; Add Graphics *G; Clear Graphics *G;
COLOR Graphics *G RED.

CBUF = 'D I ' ICHAR//' SADD G ' /GCHAR/' $CL G ' /GCHAR//' I'
' SCO G ' /GCHAR//' RED'

the following COMTAL command reads: Add Target;
Graphics *G = PROfile of image *I.

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

print out instructions for the COMTAL Keyboard interaction

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

CALL WAIT
RETURN

ERROR RETURNS:

8001 WRITE(TERM, 8003)GRAPH, GRLO, GRHI

8003 FORMAT(' THE GRAPH PARAMETER','I4,',' IS OUT OF RANGE:','213)
GOTO 8900

8101 WRITE(TERM, 8103)IMAGE, IMLO, IMHI

8103 FORMAT(' THE IMAGE PARAMETER','I4,',' IS OUT OF RANGE:','213)
GOTO 8900

8900 WRITE(TERM, 8901)

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

PARAMETER DECLARATIONS:

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

INTRODUCTION:

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

LIMITATIONS:

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

WRITTEN BY:

Keith Miller, NASA-Langley ASEE fellow, 1984

REVISION HISTORY:

LOCAL VARIABLES:

INTEGER TERM ! logical unit for terminal output

INITIALIZATIONS:

DATA TERM/1/

PROCESSING:

IF (LOW .GT. HIGH) WRITE(TERM, 1001) LOW, HIGH
1001 FORMAT(' LOW, ', IS, ', ', GT, HIGH, ', IS, ', RANGE fails.')
RANGE = ((OBJECT GE, LOW), AND, (OBJECT LE, HIGH))
RETURN
END
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:**

C 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. FRALCON, July, 1983. This modification is by KEITH MILLER, June, 1984.

**REVISION HISTORY:**

**LOCAL VARIABLES:**

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

C***INITIALIZATIONS:

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

C***PROCESSING

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

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

CALL EXEC(1,3GB+1008,ONOFFS,32,(GRAPH-1)*2048 + LINE + 512)

RETURN

C***ERROR RETURNS

C
8001 WRITE(TERM, 8003) GRAPH, GRLO, GRHI
8003 FORMAT(' GRAPHICS NUMBER', ', 13,' 'OUT OF RANGE: ', 212,'.')
GOTO 8900

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

C
8900 WRITE(TERM, 8901)
8901 FORMAT(' RDGLN FAILS. NO TRANSFER.')
RETURN
END
&RDGPT T=60084 IS ON CR0021 USING 00024 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE RDGPT(VALUE, GRAPH, XCOORD, YCOORD)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***PARAMETER DECLARATIONS:
0006 C
0007 INTEGER VALUE ! 0-1 graphics value to be retrieved.
0008 INTEGER GRAPH ! COMM graph number to be read from.
0009 INTEGER XCOORD, YCOORD ! point where value is to be read from.
0010 C
0011 C***INTRODUCTION:
0012 C
0013 C This subroutine Reads a Graph Point from the COMM. The value
0014 C of the graphics is an on/off decision, represented
0015 C by an integer 1 or 0.
0016 C If the graph number or coordinates are out of range, an error
0017 C message is printed and no transfer takes place.
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 If GRAPH, XCOORD, YCOORD, or VALUE are out of range, an error message
0026 C is printed at the terminal and no transfer takes place.
0027 C
0028 C***SUBPROGRAMS CALLED:
0029 C
0030 C name source load remarks
0031 ------ ------ ------ -----------------------------------------------
0032 C RANGE &RANGE XRANGE logical function that determines if its
0033 C first parameter is within the last two parameters.
0034 C BTEST ------ ------ HP FORTRAN77 intrinsic function that takes an
0035 C integer argument and returns .TRUE. or .FALSE.
0036 C according to the bit selected by its second
0037 C argument; bit = 0, FALSE returned.
0038 C
0039 C***WRITTEN BY:
0040 C
0041 C The code on which this subprogram is based was written by
0042 C NETTIE D. FAULCON, July, 1983. This modification is by
0043 C KEITH MILLER, June, 1984.
0044 C

86
0000 LOCAL VARIABLES:
0001
0002 LOGICAL RANGE
0003 "function that ascertains if its first parameter
0004 is between (inclusive) its last 2 parameters
0005 LOGICAL BTEST
0006 "is a certain bit on or off.
0007 INTEGER TERM
0008 "the logical unit for terminal output
0009 INTEGER GRLO, GRHI
0010 "the limits on COMTAL monochrome graph numbers
0011 INTEGER LNLO, LNHI
0012 "the limits on COMTAL graph line numbers
0013 INTEGER BTLO, BTHI
0014 "the limits on COMTAL graphics values
0015 INTEGER BITS(32)
0016 "a buffer to read & write a COMTAL graphics line
0017 INTEGER WORD
0018 "which word of BITS holds the bit selected by XCOORD.
0019 INTEGER BIT
0020 "which bit in BITS(WORD) holds the bit selected by
0021 "XCOORD; bits numbered 0-15, right to left.
0022
0023 C**INITIALIZATIONS:
0024
0025 DATA TERM/1/
0026 DATA GRLO/1/, GRHI/4/
0027 DATA LNLO/0/, LNHI/511/
0028
0029 C**PROCESSING
0030 IF (.NOT.(RANGE(GRAPH,GRLO,GRHI))) GOTO 8801 ! error return
0031 IF (.NOT.(RANGE(XCOORD,LNLO,LNHI))) GOTO 8801 ! error return
0032 IF (.NOT.(RANGE(YCOORD,LNLO,LNHI))) GOTO 8801 ! error return
0033 Programming note:
0034 The EXEC calls below are to the DVR41 driver. The first call
0035 is identical to the call made in RDILN. The second EXEC call
0036 is identical to the one in WRILN. See the documentation for
0037 those subroutines for details on these calls.
0038
0039 Read the COMTAL line (horizontal) that contains the point in question:
0040 CALL EXEC(1,368+1008,BITS.32,(GRAPH-1)*2048+512+YCOORD)
0041 Find the single bit that has been selected:
0042 WORD = (XCOORD/16) + 1
0043 BIT = (16*WORD) - XCOORD - 1
0044 VALUE = 0 ! bit is clear until proven set.
0045 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)XCOOR, LNLO, LNH1
8103 FORMAT(' X COORDINATE,' , I4, ' OUT OF RANGE: ', 214, '.')
GOTO 8900
8102 C
8201 WRITE(TERM, 8203)YCOOR, LNLO, LNH1
8203 FORMAT(' Y COORDINATE,' , I4, ' OUT OF RANGE: ', 214, '.')
GOTO 8900
8106 C
8900 WRITE(TERM, 8901)
8901 FORMAT(' RDGPT FAILS. NO TRANSFER.' )
RETURN
END
&RDIL2 T=00004 IS ON CR00021 USING 00018 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 C SUBROUTINE RDIL2(INTS, IMAGE, LINE)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
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 ! 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 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 ----- ------ ------ -----------------------------------------------
0032 C RANGE %RANGE %RANGE logical function that determines if its
0033 C first parameter is within the last two parameters.
0034 C ICHAR ------ ------ intrinsic HP FORTRAN77 function that converts
0035 C a character (1 byte) into an integer (2 bytes)
0036 C
0037 C***WRITTEN BY:
0038 C
0039 C The code on which this subprogram is based was written by
0040 C NETTIE D. FAULCON, July, 1983. This modification is by
0041 C KEITH MILLER, June, 1984.
0042 C
0043 C***REVISION HISTORY:
0044 C
**LOCAL VARIABLES:**

```
INTEGER  PIXELS(256)  ! 512 bytes, 1 pixel/byte, transferred to COMTAL
CHARACTER*1  CPIX(512)  !  overlays PIXELS
EQUIVALENCE (PIXELS,CPIX)
```

**LOGICAL RANGE**

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

**INTEGER TERM**

The logical unit for terminal output

**INTEGER IMLO, IMHI**

The limits on COMTAL monochrome image numbers

**INTEGER LNLO, LNHI**

The limits on COMTAL image line numbers

**INTEGER INDEX**

Loop index for stepping through the arrays.

**INITIALIZATIONS:**

```
DATA  TERM/'\n/, IMLO/'\n/, IMHI/'\n/
DATA  LNLO/'\n/, LNHI/'\n11/
```

**PROCESSING**

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

**Programming note:**

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

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

Transform the 512 8 bit values into 512 16 bit integers

```
DO 1000  INDEX = 1, 512
  INTS(INDEX)  =  ICHAR(CPIX(INDEX))  !  CPIX overlays PIXELS
```

```
1000  CONTINUE
```

**RETURN**
C***ERROR RETURNS

0094 WRITE(TERM, 8003) IMAGE, IMLO, IMHI
0095 FORMAT(' IMAGE NUMBER.', I3, ' OUT OF RANGE:', 212., ')
0096 GOTO 8988

0101 WRITE(TERM, 8103) LINE, LNLO, LNHI
0102 FORMAT(' LINE NUMBER.', I4, ' OUT OF RANGE:', 213., ')
0103 GOTO 8900

0104 RETURN
0105 END

0901 WRITE(TERM, 8901)
0103 FORMAT(' RDIL2 FAILS. NO TRANSFER.')
**SUBROUTINE RDILN(PIXELS, IMAGE, LINE)***

**PARAMETER DECLARATIONS:**

**INTEGER** PIXELS(256) ! 512 bytes (pixels) to be read

**INTEGER** IMAGE ! COMTAL image number to be read from

**INTEGER** LINE ! which horizontal line to be read from;

**INTEGER** LINE ! lines numbered from 1 (screen top) to 512.

**INTRODUCTION:**

This subroutine reads an image line from the COMTAL. The line of
pixels is made up of 8 bit (0-255) grey scale intensities. The
PIXELS array will be filled to capacity by RDILN.

**LANGUAGE:**

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

**LIMITATIONS:**

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

**SUBPROGRAMS CALLED:**

name source load remarks

**RANGE** CRANGE 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
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 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 1008 identifies the line transfer operation of that driver; the third parameter (PIXELS) holds the data to be transferred, and the fourth parameter gives PIXELS' length in words (256); and the final parameter is a COMTAL command code for the transfer.

CALL EXEC(1, 3GB+1008, PIXELS, 256, (IMAGE-1)*2048 + LINE)
RETURN

C***ERROR RETURNS

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

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

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

END
&RDIP T=00004 IS ON CR00021 USING 0008 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE RDIPT(VALUE, IMAGE, XCOOR, YCOOR)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***PARAMETER DECLARATION:
0006 C
0007 INTEGER VALUE ! receives pixel value, converted from byte
0008 C ! to integer.
0009 INTEGER IMAGE ! COMTAL image number to be read from.
0010 INTEGER XCOOR, YCOOR ! point where value is to be read from.
0011 C
0012 C***INTRODUCTION:
0013 C
0014 C This subroutine Reads an Image Point from the COMTAL. The value
0015 C of the pixel is an 8 bit (0-255) grey scale intensity. If the
0016 C image number or coordinates are out of range, an
0017 C message is printed and no transfer takes place.
0018 C
0019 C***LANGUAGE:
0020 C
0021 C FORTRAN 77, the HP-1000 version for RTE-6/VM.
0022 C
0023 C***LIMITATIONS:
0024 C
0025 C The IMAGE designated
0026 C must be a monochrome image. The YCOOR parameter must be between
0027 C 0 and 511. If IMAGE or YCOOR is out of range, an error message is printed
0028 C and no transfer takes place.
0029 C
0030 C***SUBPROGRAMS CALLED:
0031 C
0032 C name source load remarks
0033 C ----- ----- ------ -------------------------------
0034 C RANGE &RANGE XORANGE logical function that determines if its
0035 C first parameter is within the last two parameters.
0036 C ICHAR ----- ------ HP FORTRAN77 intrinsic function: converts a byte
0037 C into its integer code.
0038 C
0039 C***WRITTEN BY:
0040 C
0041 C The code on which this subprogram is based was written by
0042 C NETTIE D. FAULCON, July, 1983. This modification is by
0043 C KEITH MILLER, June, 1984.
0044 C
**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 PIXELS(256)` a buffer to read a COMTAL line.
- `CHARACTER*1 BYTES(512)` overlay for PIXELS buffer.
- `EQUIVALENCE (PIXELS, BYTES)`

**Initializations:**

- `DATA TERM/'`
- `DATA IMLO/1/, 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 fed 0-511 to the FORTRAN array, 1-512.

**Error Returns:**

- `RETURN`
0091 0081 WRITE(TERM, 8003) IMAGE, IMLO, IMHI
0092 8003 FORMAT(′ IMAGE NUMBER: ′, I3, ′ OUT OF RANGE: ′, 212, ′ ′)
0093     GOTO 8900
0094 C
0095 8101 WRITE(TERM, 8103)YCOORD, LNLO, LNHI
0096 8103 FORMAT(′ Y COORDINATE: ′, I4, ′ OUT OF RANGE: ′, 214, ′ ′)
0097     GOTO 8900
0098 C
0099 8201 WRITE(TERM, 8203)XCOORD, LNLO, LNHI
0100 8203 FORMAT(′ Y COORDINATE: ′, I4, ′ OUT OF RANGE: ′, 214, ′ ′)
0101     GOTO 8900
0102 C
0103 8960 WRITE(TERM, 8901)
0104 8901 FORMAT(′ RDIPT FAILS. NO TRANSFER. ′)
0105     RETURN
0106     END
SUBROUTINE RDIRC(OUTARA, XDIM, YDIM, IMAGE, UPLFX, UPLFY)

INTEGER XDIM, YDIM ! dimensions of the output array, OUTARA
INTEGER OUTARA(XDIM, YDIM) ! the array to be filled
INTEGER IMAGE ! the number of the COMTAL image from
INTEGER UPLFX, UPLFY ! the image coordinates of the upper left

GO TO 043

PARAMETER DECLARATIONS:

INTEGER XDIM, YDIM ! dimensions of the output array, OUTARA
INTEGER OUTARA(XDIM, YDIM) ! the array to be filled
INTEGER IMAGE ! the number of the COMTAL image from
INTEGER UPLFX, UPLFY ! the image coordinates of the upper left

INTRODUCTION:

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

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

LANGUAGE:

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

LIMITATIONS:

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

SUBPROGRAMS CALLED:

name source load remarks

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

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

**Revision History:**

**Local Variables:**

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

**Initializations:**

<table>
<thead>
<tr>
<th>DATA</th>
<th>TERM/1/</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA</td>
<td>IMLO/1/, IMHI/4/</td>
</tr>
<tr>
<td>DATA</td>
<td>LNLO/0/, LNHI/511/, LNCT/512/</td>
</tr>
</tbody>
</table>

**Processing**

| IF (.NOT.(RANGEIMAGE, IMLO, IMHI)))GOTO 8001 ! error return |
| IF (.NOT.(RANGEXDIM, 1, LNCT)))GOTO 8101 ! error return |
| IF (.NOT.(RANGEYDIM, 1, LNCT)))GOTO 8201 ! error return |
| IF (.NOT.(RANGEUPLFX, LNLO, LNHI )))GOTO 8301 ! error return |
| IF (.NOT.(RANGEUPLFY, LNLO, LNHI )))GOTO 8401 ! error return |
| IF (.NOT.(RANGEUPLFX*XDIM-1, LNLO, LNHI )))GOTO 8501 ! error return |
| IF (.NOT.(RANGEUPLFY*YDIM-1, LNLO, LNHI )))GOTO 8601 ! error return |

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

IMROW = UPLFY
DO 2000 ARAROW = 1, YDIM
   CALL RDILH(IBUTF, IMAGE, IMROU)
   IMROU = IMROU + 1 ! increment for next pass thru 2000 loop
   IMCOL = UPLFX + 1 ! the "+1" is necessary because COMTAL image coordinates range from 0 to 511 and the FORTRAN array indices range from 1 to 512.
   IMROW = IMROW + I ! increment for next pass thru 2000 loop
C
   OUTARA(ARAROW, ARACOL) = ICHAR(CBUF(IMCOL))
   IMCOL = IMCOL + 1
DO 1000 ARACOL = 1, XDIM
1000 CONTINUE
2000 CONTINUE
RETURN ! successful termination
C
ERROR RETURNS:
8661 WRITE(TERM, 8003) IMAGE, IMLO, IMHI
8003 FORMAT(' IMAGE NUMBER:, I3, ' OUT OF RANGE:, 214,'."
GOTO 8900
8103 FORMAT(' X DIMENSION:, I4, ' OUT OF RANGE:, 215,'."
GOTO 8900
8203 FORMAT(' Y DIMENSION:, I4, ' OUT OF RANGE:, 215,'."
GOTO 8900
8303 FORMAT(' X COORDINATE FOR CORNER:, I4, ' OUT OF RANGE:, 215,'."
GOTO 8900
8403 FORMAT(' Y COORDINATE FOR CORNER:, I4, ' OUT OF RANGE:, 215,'."
GOTO 8900
8503 FORMAT(' X COORDINATE FOR THE CORNER AND THE X DIMENSION:, I4, ' OUT OF THE ARRAY', ., ' OVERFLOW IMAGE BOUNDARIES.
GOTO 8900
8603 FORMAT(' Y COORDINATE FOR THE CORNER AND THE Y DIMENSION:, I4, ' OUT OF RANGE:, 215,'."
GOTO 8900
8703 FORMAT(' IMAGE COORDINATE LIMITS ARE:, 215,'."
GOTO 8900
99
0137   - 1   'OF THE ARRAY', '/ ', 'OVERFLOW IMAGE BOUNDARIES.'.
0138   - 2   'X COORDINATE =', I4, 'X DIMENSION =', I4, '/ '.
0139   - 3   'IMAGE COORDINATE LIMITS ARE ', 215, '/ '.
0140   GOTO 8900
0141 C
0142 8900 WRITE(TERM, 8903)
0143 8963 FORMAT(' RDIRC FAILS. NO TRANSFER TAKES PLACE.')
0144 RETURN
0145 END
0146
0147
**&RDLUT T=00004 IS ON CR00021 USING 00018 BLKS R=0000**

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 C SUBROUTINE RDLUT(TABLE, LUTNUM)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***PARAMETER DECLARATIONS:
0006 C
0007 INTEGER TABLE(256) ! the values of the specified COMTAL look-up
0008 C ! table are read into this array. The 0 entry
0009 C ! goes into TABLE(1), .... the 255 entry goes
0010 C ! into TABLE(256).
0011 INTEGER LUTNUM ! the number of the COMTAL look-up table (called
0012 C ! "function memory" in the COMTAL literature).
0013 C
0014 C***INTRODUCTION:
0015 C
0016 C The subroutine Read Look-Up Table (LUT) reads the COMTAL mapping from
0017 C the integers 0-255 into the integer array TABLE. This LUT can be used
0018 C for grey level enhancements in the COMTAL. A similar subroutine
0019 C called RDPSU is used to read from a pseudocolor look-up table. This
0020 C routine is only used for grey scale look-up tables.
0021 C
0022 C***LANGUAGE:
0023 C
0024 C FORTRAN 77, the HP-1000 version for RTE-6/VM.
0025 C
0026 C***LIMITATIONS:
0027 C
0028 C Although 8 bits are sufficient for the look up table values, full
0029 C integers are used in TABLE. This format is dictated by the COMTAL
0030 C conventions as given in section 5.2.3.1.
0031 C
0032 C***SUBPROGRAMS CALLED:
0033 C
0034 C name source load remarks
0035 C ------ ------ ------- -------------------------------
0036 C RANGE &RANGE &RANGE logical function which determines if its 1st
0037 C parameter is within its 2nd and 3rd inclusive.
0038 C
0039 C***WRITTEN BY:
0040 C
0041 C The code on which this subprogram is based was written by
0042 C NETTIE D. FAULCON, July, 1983. This subprogram was written by
0043 C KEITH MILLER, July, 1984, with the support of a NASA-ASEE
0044 C summer fellowship.
**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-6/VM Programmer’s Reference Manual, 2-19ff. The COMTAL parameters are discussed in section 5.2.3 of the COMTAL User’s Manual.

- The first parameter to EXEC identifies the EXEC command as a read command. The second parameter identifies the resident HP driver (36B) and gives the code (200B) that identifies this operation, a transfer to a COMTAL function memory (Look-Up Table).

- The third parameter gives the Look-Up Table values (TABLE), and the fourth parameter gives the length of TABLE in words.

- The fifth parameter is a COMTAL code that is described bit by bit in the User’s Manual. In short, bit 15 signifies write to COMTAL, bit 14 designates function memory instead of pseudocolor, bit 12 signifies standard replacement, and bits 8 and 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)

**ERROR RETURN**

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

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

```c
0045 C
0046 C***INITIALIZATIONS:
0047 C
0048 C   NONE
0049 C
0050 C***PROCESSING
0051 C
0052 C   Programming notes:
0053 C   The EXEC command parameters are discussed in the HP RTE-6/VM
0055 C   are discussed in section 5.2.3 of the COMTAL User’s Manual.
0056 C
0057 C   The first parameter to EXEC identifies the EXEC command as
0058 C   a read command. The second parameter identifies the resident
0059 C   HP driver (36B) and gives the code (300B) that identifies this
0060 C   operation, a transfer from the COMTAL pseudocolor table.
0061 C   The third parameter gives the array that will hold the values,
0062 C   and the fourth parameter gives the length of TABLE in words.
0063 C   The fifth parameter is a COMTAL code that is described bit by
0064 C   bit in the User’s Manual. The DVR41 driver takes care of all the
0065 C   bits except 8&9 which identify the color to be transferred.
0066 C
0067 C   Note that we make three separate calls to EXEC. Each call fills a
0068 C   different section of TABLE with a different color of the COMTAL’s
0069 C   pseudocolor table.
0070 C
0071 C   CALL EXEC( 1, 36B+300B, TABLE(1), 256, 1*256 ) ! red
0072 C   CALL EXEC( 1, 36B+300B, TABLE(257), 256, 0*256 ) ! green
0073 C   CALL EXEC( 1, 36B+300B, TABLE(513), 256, 2*256 ) ! blue
0074 C   RETURN
0075 C   END
```
RRR T=00004 IS ON CR00021 USING 00018 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE RDTAB(TABLE, CNUMB, IMOGR)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***PARAMETER DECLARATIONS:
0006 C
0007 INTEGER TABLE(16) ! holds the image/graphics table from COMTAL.
0008 INTEGER CNUMB ! COMTAL image or graph number.
0010 C
0011 C***INTRODUCTION:
0012 C
0013 C This subroutine Reads the Table associated with the image or graphics
0014 C memory plane identified by CNUMB (the COMTAL image or graphics number)
0015 C and IMOGR (which is either a 0, indicating an image, or 1, indicating
0016 C a graphics plane. The 16 word, 32 byte table goes into TABLE.
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 IMOGR must be either a 0 or a 1, and CNUMB must be between 1 and 9
0025 C inclusive. If either number is out of range, no transfer takes place.
0026 C
0027 C***SUBPROGRAMS CALLED:
0028 C
0029 C name source load remarks
0030 C ----- ----- ------- ---------------------------------------------
0031 C RANGE &RANGE X RANGE logical function that determines if its
0032 C first parameter is within the last two parameters.
0033 C BTEST ------- ------ HP FORTRAN?? intrinsic function that takes an
0034 C integer argument and returns .TRUE. or .FALSE.
0035 C according to the bit selected by its second
0036 C argument; bit = 0, FALSE returned.
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 modification is by
0042 C KEITH MILLER, June, 1984.
0043 C
0044 C***REVISION HISTORY:
**LOCAL VARIABLES:**

- **LOGICAL RANGE**: A function that ascertains if its first parameter is between (inclusive) its last 2 parameters.
- **LOGICAL BTEST**: A logical test that determines if a certain bit is on or off.
- **INTEGER TERM**: The logical unit for terminal output.
- **INTEGER DGLO, DGHI**: The limits on single-digit image/graphics numbers.
- **INTEGER BTLO, BTHI**: The limits on COMTAL graphics values.
- **XCOOR**: Bits numbered 0-15, right to left.

**INITIALIZATIONS:**

- **DATA TERM/1/**: The first argument, "1", identifies the operation as a read.
- **DATA DGLO/1/, DGHI/9/**: The second argument has two parts: "368" identifies the DVR41 driver, and "5006" selects a transfer code for operation of that driver.
- **DATA BTLO/0/, BTHI/1/**: The third argument, "TABLE", is the buffer that will hold the IGT (Image/Graphics Table) information after the EXEC is completed.
- **CALL EXEC(1,368X5006,TABLE,16,7X4096+IMOGRX128+CNUMB-1)**: The final parameter is a code to the COMTAL which identifies the mode of the transfer ("7X4096"), mode 7 selects either image or graphics ("IMOGRX128"); and gives the number of the image/graphics memory plane ("CNUMB-1").

**PROCESSING**

- **Programming note:**
- **CALL EXEC(1,368X5006,TABLE,16,7X4096+IMOGRX128+CNUMB-1)**: The first parameter is a code to the DVR41 driver.
- **CALL EXEC(1,368X5006,TABLE,16,7X4096+IMOGRX128+CNUMB-1)**: The second argument has two parts: "368" identifies the DVR41 driver, and "5006" selects a transfer code for operation of that driver.
- **CALL EXEC(1,368X5006,TABLE,16,7X4096+IMOGRX128+CNUMB-1)**: The third argument, "TABLE", is the buffer that will hold the IGT (Image/Graphics Table) information after the EXEC is completed.
- **CALL EXEC(1,368X5006,TABLE,16,7X4096+IMOGRX128+CNUMB-1)**: The fourth argument, "16", gives the length of TABLE in words.
- **CALL EXEC(1,368X5006,TABLE,16,7X4096+IMOGRX128+CNUMB-1)**: The final parameter is a code to the COMTAL which identifies the mode of the transfer ("7X4096"), mode 7 selects either image or graphics ("IMOGRX128"); and gives the number of the image/graphics memory plane ("CNUMB-1").

**ERROR RETURNS**

- **WRITE(TERM, 8003) CNUMB, DGLO, DGHI**: The first argument, "1", identifies the operation as a read.
- **WRITE(TERM, 8003) CNUMB, DGLO, DGHI**: The second argument has two parts: "368" identifies the DVR41 driver, and "5006" selects a transfer code for operation of that driver.
- **WRITE(TERM, 8003) CNUMB, DGLO, DGHI**: The third argument, "TABLE", is the buffer that will hold the IGT (Image/Graphics Table) information after the EXEC is completed.
- **WRITE(TERM, 8003) CNUMB, DGLO, DGHI**: The fourth argument, "16", gives the length of TABLE in words.
- **WRITE(TERM, 8003) CNUMB, DGLO, DGHI**: The final parameter is a code to the COMTAL which identifies the mode of the transfer ("7X4096"), mode 7 selects either image or graphics ("IMOGRX128"); and gives the number of the image/graphics memory plane ("CNUMB-1").
0091     8103 FORMAT(' IMAGE/GRAPHICS ARGUMENT MUST BE 0 OR 1. NOT '.14.'')
0092          GOTO 8900
0093  C
0094     8900 WRITE(TERM, 8901)
0095     8901 FORMAT(' RDTAB FAILS. NO TRANSFER.')
0096          RETURN
0097          END
&RDTAR T=00004 IS ON CR00021 USING 00012 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE RDTAR(XCOORD, YCOORD)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
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 400B 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+400B, IBUF, 2, 00000B)
XCOORD = IBUF(1)
YCOORD = IBUF(2)
RETURN
END
&SETV T=00004 IS ON CR0021 USING 00004 BLKS R=0000

0081 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0082 SUBROUTINE SETV(IMAGE)
0083 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0084 C
0085 C***PARAMETER DECLARATIONS:
0086 C
0087 INTEGER IMAGE ! the COMTAL image number to be assigned to the
0088 C video camera: 5 is traditional.
0089 C
0090 C***INTRODUCTION:
0091 C
0092 C The subroutine "SET Video" establishes the image number that is to be
0093 C associated with the video camera, and displays the camera input.
0094 C
0095 C***LANGUAGE:
0096 C
0097 FORTRAN 77, the HP-1000 version for RTE-6/VM.
0098 C
0099 C***LIMITATIONS:
0100 C
0101 SETV should only be called once during a COMTAL session. If the
0102 camera is already set to IMAGE and SETV is called, the COMTAL
0103 freezes. Manually reset the COMTAL with the SHIFT-to release any
0104 previous SETV command.
0105 C
0106 Note that IMAGE should not be an image number used for COMTAL memory.
0107 For programming convenience, we insist on a single digit. 5 is
0108 traditional in this lab, but 5-9 will do.
0109 C
0110 C***SUBPROGRAMS CALLED:
0111 C
0112 C name source load remarks
0113 C----- ----- ----- --------------------------------------------
0114 C034 C CMND &CMND %CMND sends character strings to the COMTAL where
0115 C they are much like COMTAL keyboard commands.
0116 C035 C DIGIT &DIGIT %DIGIT character*1 function that returns '0'-'9'
0117 C according to integer input 0-9.
0118 C036 C RANGE &RANGE %RANGE logical function which determines if its 1st
0119 C argument is within the 2nd and 3rd, inclusive.
0120 C
0121 C***WRITTEN BY:
0122 C
0123 The code on which this subprogram is based was written by
0124 C
0125 NETTIE D. FAULCON. July, 1983. This subprogram was written by
C KEITH MILLER, July, 1984, with the support of a NASA-ASEE summer fellowship.

C***REVISION HISTORY:
C

C***LOCAL VARIABLES:
C
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 digit reserved for image memory.
INTEGER TERM ! logical unit number for terminal input

C***INITIALIZATIONS:

DATA VDLO/5/, VDHI/9/
DATA TERM/1/

C***PROCESSING:
IF (.NOT.(RANGE(IMAGE,VDLO,VDHI))) GOTO 8801 ! error return

the following COMTAL command is expanded to:
Set Video image #1; 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

C***ERROR RETURNS:
C
0091 WRITE(TERM,8003) IMAGE, VDLO, VDHI
0092 8003 FORMAT(" YOUR IMAGE ARGUMENT,'14,' IS OUT OF RANGE;'-214")
0093 C
0094 WRITE(TERM,8901)
0095 8901 FORMAT(" SETV FAILS. NO ACTION TAKEN.")
0096 C
0097 RETURN
0098 END
0100
SUBROUTINE SPLIT(BYTE1, BYTE2, INBYTE)

PARAMETER DECLARATIONS:

INTEGER BYTE1 ! on output, this integer holds the left (high order)
! byte of INBYTE in its right (low order) byte.
INTEGER BYTE2 ! on output, this integer holds the right (low order)
! byte of INBYTE in its right (low order) byte.
INTEGER INBYTE ! the input integer whose two bytes are to be SPLIT.

INTRODUCTION:
The subroutine SPLIT takes each of the two bytes of INBYTE and
interprets each byte as a separate integer. The two resulting
numbers are assigned to BYTE1 (which gets the high order byte of
INBYTE) and to BYTE2 (which gets the low order byte).

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

LIMITATIONS:
None.

SUBPROGRAMS CALLED:
None.

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

LOCAL VARIABLES:
INTEGER IHOLD ! an INTEGER interpretation of bits
CHARACTER CHOLD(2) ! a CHARACTER interpretation of bits

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

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

IMERGE = INBYTE
IHOLD = 0 ! zero out high order bits
CHOLD(2) = CSPLIT(1)
BYTE1 = IHOLD
CHOLD(2) = CSPLIT(2)
BYTE2 = IHOLD

RETURN
END
SUBROUTINE SPRED(IMAGE)

C***PARAMETER DECLARATIONS:
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. On the
first pass, SPREAD determines the high and low pixel values in the image. On the second pass, SPREAD replaces each pixel value
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 XRDIL2 reads a horizontal line of pixels from the COMTAL, and places the values in INTEGER array.
WRIL2 &WRIL2 XWRIL2 writes a horizontal line of pixels from an integer array to a COMTAL image.
HILO &HILO XHILO passes through a COMTAL image and returns the highest and lowest pixel value found.
IFIX ------ ------- FORTRAN IV intrinsic function: converts from real to integer.
FLOAT ------ ------- FORTRAN IV intrinsic function: converts from
integer to real.

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

REVISION HISTORY:

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

LOCAL VARIABLES:

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

INITIALIZATIONS:

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

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 4888 ROW = 1,512
   CALL RDIL2(IBUF, IMAGE, ROW)
4888 DO 3000 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 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE SUBI2(C, A, B)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C**PARAMETER DECLARATIONS:
0006 C
0007 INTEGER C    ! the COMTAL image into which the difference between
0008 C    ! image A and image B is placed by SUBI2 (C = A - B)
0009 INTEGER A, B ! the images whose difference is taken
0010 C
0011 C**INTRODUCTION:
0012 C
0013 C The subroutine "SUBtract Images #2" takes a pixel by pixel difference
0014 C of images A and B and places the resulting image into image C.
0015 C The truecolor image B is used in the processing of SUBI2, and will
0016 C be left as the combination of C, A, and B for red, green, and blue
0017 C respectively.
0018 C
0019 C Unlike SUBIM, which does no scaling or offsetting, SUBI2 offsets the
0020 C result of the subtraction by adding 128 to each pixel difference.
0021 C Thus, a 128 pixel value in image C after the call means that
0022 C the true value of the difference was 0. This offset can be handy when
0023 C many of the values of the difference are less than 0. After the offset
0024 C is added, any pixel values less than 0 are set to 0.
0025 C
0026 C**LANGUAGE:
0027 C
0028 C FORTRAN 77, the HP-1000 version for RTE-6/VM.
0029 C
0030 C**LIMITATIONS:
0031 C
0032 C The truecolor B image is destroyed during this operation. C is
0033 C obviously destroyed. This subroutine is accomplished using COMTAL
0034 C commands that exploit the pipeline processors. Because of this, the
0035 C processing steps are obscure. For example, there is no motivation
0036 C outside the COMTAL instructions for making the combination of C, A, and B
0037 C a color image. Readers should be aware of these obscurities before trying
0038 C to understand the code.
0039 C
0040 C If any offset difference is less than 0, the pixel value is set to 0.
0041 C
0042 C If any image number is out of range, an error message is printed and
0043 C no further processing takes place.
0044 C This subroutine assumes that 0 is not a legal image for the COMTAL
configuration.

**SUBPROGRAMS CALLED:**

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

**C**

- **CMND** &CMND **CMND**: Sends a command to the COMTAL as if the command were sent to the keyboard.
- **RANGE** &RANGE **RANGE**: logical function that determines if the 1st parameter is within the range of the 2nd & 3rd.
- **DIGIT** &DIGIT **DIGIT**: character*1 function that 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** CBUF(255): character overlay for IBUF
- **EQUIVALENCE (IBUF, CBUF)**
- **LOGICAL** RANGE: function that determines if 1st parameter is within 2nd and 3rd parameter
- **INTEGER** IMLO, IMHI: limits on COMTAL image numbers
- **INTEGER** TERM
- **CHARACTER** DIGIT: returns a single digit character '0' to '9' for integer input 0-9.

**INITIALIZATIONS:**

- **DATA** IMLO/4/, 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.

```c
CBUF =

1 'UN I 8 $' />
2 'AS T $' '/'DIGIT(C)/' '/'DIGIT(A)/' '/'DIGIT(B)/' $' />
3 'D I 8 $' />
4 'SE COM G = B $' />
5 'A COM $' />
6 'I '/'DIGIT(C)/' D R $' />
7 'D I '/'DIGIT(C)/' $' />
8 'SU COM '
9 'SE COM '<Green - Blue> ! +128 offset and no scaling is the default.
10 'A COM $' />
11 'Add COMbine
12 'I '/'DIGIT(A)/' D $' />
13 'Display Image $'
14 'Set COMbine <Green - Blue> ! 128 offset and no scaling is the default.
15 Image $C = Displayed Image Red ! Red is arbitrary, since difference of images is monochrome
16 'D I '/'DIGIT(C)/' $' />
17 'Display Image $'
18 'SU COM '
19 'Subtract COMbine.
20 'CALL CMND(IBUF, 74)
21 CALL CMND(IBUF, 74)
22 RETURN
23 C
24 CERROR RETURNS
25 C
26 8001 WRITE(TERM, 8003) A
27 8003 FORMAT(' THE 2ND IMAGE PARAMETER, ', I3, ', IS OUT OF RANGE, ')
28 GOTO 8900
29 C
30 8101 WRITE(TERM, 8103) B
31 8103 FORMAT(' THE 3RD IMAGE PARAMETER, ', I3, ', IS OUT OF RANGE, ')
32 GOTO 8900
33 C
34 8201 WRITE(TERM, 8203) C
35 8203 FORMAT(' THE 1ST IMAGE PARAMETER, ', I3, ', IS OUT OF RANGE, ')
36 GOTO 8900
```

C
0138  8900 WRITE(TERM, 8901)
0139  8901 FORMAT(' SUB12 RETURNS WITHOUT FURTHER PROCESSING.
0140        RETURN
0141        END
0142
&SUBIM T=00004 IS ON CR0021 USING 00004 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE SUBIM(C, A, B)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***PARAMETER DECLARATIONS:
0006 C
0007 INTEGER C ! the COMTAL image into which the difference between
0008 C ! image A and image B is placed by SUBIM (C = A - B)
0009 INTEGER A, B ! the images whose difference is taken (C = A - B)
0010 C
0011 C***INTRODUCTION:
0012 C
0013 C The subroutine "SUBtract IMages" takes a pixel by pixel difference
0014 C of images A and B and places the resulting image into image C.
0015 C The truecolor image B is used in the processing of SUBIM, and will
0016 C be left as the combination of C, A, and B for red, green, and blue
0017 C respectively.
0018 C
0019 C***LANGUAGE:
0020 C
0021 C FORTRAN 77, the HP-1000 version for RTE-6/VM.
0022 C
0023 C***LIMITATIONS:
0024 C
0025 C The truecolor B image is destroyed during this operation. C is
0026 C obviously destroyed. This subroutine is accomplished using COMTAL
0027 C commands that exploit the pipeline processors. Because of this, the
0028 C processing steps are obscure. For example, there is no motivation
0029 C outside the COMTAL instructions for making the combination of C, A, and B
0030 C a color image. Readers should be aware of these obscurities before trying
0031 C to understand the code.
0032 C
0033 C The 3 images C, A, and B must be distinct.
0034 C
0035 C If any difference is less than 0, the pixel value is set to 0.
0036 C
0037 C SUBIM does no scaling or offsetting. SUB12 does an automatic
0038 C scale and offset.
0039 C
0040 C If any image number is out of range, an error message is printed and
0041 C no further processing takes place.
0042 C This subroutine assumes that 0 is not a legal image for the COMTAL
0043 C configuration.
0044 C
**SUBPROGRAMS CALLED:**

<table>
<thead>
<tr>
<th>name</th>
<th>source</th>
<th>load</th>
<th>remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMND</td>
<td>&amp;CMND</td>
<td>&amp;CMND</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
- 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 ! 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
0100 C
0101 C
0102 C
0103 C
0104 C
0105 C
0106 C
0107 C
0108 C
0109 C
0110 C
0111 C
0112 C
0113 C
0114 C
0115 C
0116 C
0117 C
0118 C
0119 C
0120 C
0121 C
0122 C
0123 C
0124 C
0125 C
0126 C
0127 C
0128 C
0129 C
0130 C
0131 C
0132 C
0133 C
0134 C
0135 C
0136 C

CBUF =

1 'UN I B $'/

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

3 'D I B $'/

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

5 'A COM $'/

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

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

8 'SU COM '

Subtract COMbine.

CALL COMMAND(IBUF, ??)

RETURN

C***ERROR RETURNS

0801 WRITE(TERM, 0803) A

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

GOTO 8900

08101 WRITE(TERM, 08103) B

08103 FORMAT( ' THE 3RD IMAGE PARAMETER', I3, ', IS OUT OF RANGE.' )

GOTO 8900

08201 WRITE(TERM, 08203) C

08203 FORMAT( ' THE 1ST IMAGE PARAMETER', I3, ', IS OUT OF RANGE.' )

GOTO 8900

08900 WRITE(TERM, 08901)

6901 FORMAT( ' SUBIM RETURNS WITHOUT FURTHER PROCESSING.' )

RETURN

END
&THRSH T=E0004 IS ON CR00021 USING 00010 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE THRSH(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 THRShold 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
C* NETTIE D. FAULCON, July, 1983. This subprogram was written by
C* KEITH MILLER, July, 1984, with the support of a NASA-ASEE
C* summer fellowship.

C**REVISION HISTORY:
C
C
C**LOCAL VARIABLES:
C
0054 INTEGER IBUF(512) ! buffer for read/write of horizontal line of
0055 INTEGER LINE ! loop index that keeps track of the line
0056 INTEGER PXCNT ! loop index that counts pixels in a line.
0057 LOGICAL RANGE ! function that determines if its 1st argument
0058 C* lies within its 2nd and 3rd inclusive.
0059 INTEGER IMLO, IMHI ! range of legal COMTAL image numbers.
0060 INTEGER PXLO, PXHI ! range of legal COMTAL pixel values.
0061 INTEGER LNLO, LNHI ! range of legal COMTAL line numbers.
0062 INTEGER ARALO, ARAHI ! range of array holding a line of pixels;
0063 INTEGER TERM ! logical unit for terminal output.

C**INITIALIZATIONS:
C
0069 DATA IMLO/1/, IMHI/4/
0070 DATA PXLO/0/, PXHI/255/
0071 DATA LNLO/0/, LNHI/511/
0072 DATA ARALO/1/, ARAHI/512/
0073 DATA TERM/1/

C**PROCESSING:
C
0079 IF (.NOT.(RANGE(INIMG, IMLO, IMHI))) GOTO 8001 ! error return
0078 IF (.NOT.(RANGE(INIMG, IMLO, IMHI))) GOTO 8101 ! error return
0077 IF (.NOT.(RANGE(THRESH, PXLO, PXHI))) GOTO 8201 ! error return

0080 DO 2000 LINE = LNLO, LNHI
0081 CALL RDIL2(IBUF, INIMG, LINE)
0082 DO 1000 PXCNT = ARALO, ARAHI
0083 IF (IBUF(PXCNT) .LT. THRESH) GOTO 500
0084 ELSE...
0085 IBUF(PXCNT) = 255
0086 1000 GOTO 1000
0087 THEN...
0088 IBUF(PXCNT) = 0
0089 500
1063 CONTINUE
0092 CALL WRIL2(OUTIMG, LINE, IBUF)
0093 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,’ , I5,’ 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,’ , I5,’ IS OUT OF RANGE: ‘, 214)
0104 GOTO 8900
0105 C
0106 8201 WRITE(TERM, 8203) THRESH, PXLO, PXHI
0107 8203 FORMAT(’THE THRESHOLD VALUE,’ , I5,’ IS OUT OF RANGE: ‘, 214)
0108 GOTO 8900
0109 C
0110 8900 WRITE(TERM, 8901)
0111 8901 FORMAT(’THRESH FAILS. OUTIMG NOT CHANGED.’)
0112 RETURN
0113 END
&TSTI1 T=00004 IS ON CR00021 USING 00004 BLKS R=0000

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

**Local Variables:**

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

**Initializations:**

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

**Processing:**

- DO 2000 ROW = 0, 511
  - by starting WRILN at different places in BIGBUF, the desired wrap-around effect is achieved.
- CALL WRILN( WHICH, ROW, BIGBUF((ROW/2)+1) )
- 2000 CONTINUE

**Return**

- RETURN

**End**
&TV2C4 T=60004 IS ON CR00021 USING 00006 ELKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE TV2C4
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***PARAMETER DECLARATIONS:
0006 C
0007 C None.
0008 C
0009 C***INTRODUCTION:
0010 C
0011 C The subroutine "TV to Comtal image transfer #4" digitizes four
0012 C copies of the current TV image, and averages them together into a
0013 C single image stored in COMTAL image #1. COMTAL images 1, 2, and 3
0014 C are used for storage. Image 5 must be set to the video camera image
0015 C before TV2C4 is called.
0016 C
0017 C***LANGUAGE:
0018 C
0019 C FORTRAN 77, the HP-1000 version for RTE-6/VM.
0020 C
0021 C***LIMITATIONS:
0022 C
0023 C Images 1, 2, 3, and 4 are changed by this routine.
0024 C
0025 C This subroutine assumes that image 5 has been set to the video
0026 C camera previous to the call. If 5 is not set to video, the COMTAL
0027 C hangs up.
0028 C
0029 C
0030 C***SUBPROGRAMS CALLED:
0031 C
0032 C name source load remarks
0033 C __________ __________ __________ ________________
0034 C RANGE &RANGE &RANGE logical function that determines if its 1st
0035 C argument lies within 2nd & 3rd argument, inclusive.
0036 C TV2CM &TV2CM &TV2CM digitizes the camera associated with COMTAL image
0037 C 5 into the 1st argument, an image memory.
0038 C ADDI2 &ADDI2 &ADDI2 adds two images, pixel by pixel, and divides the
0039 C sums by last argument, to produce a new image.
0040 C
0041 C***WRITTEN BY:
0042 C
0043 C The code on which this subprogram is based was written by
0044 C NETTIE D. FAULKON, July, 1983. This subprogram was written by

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

**REVISION HISTORY:**

LOCAL VARIABLES:

LOGICAL RANGE

INTEGER TVIMAG

IMPLEMENTATION:

DATA TVIMAG/5/

DATA TERM/1/

PROCESSING:

CALL DSPVD

CALL TV2CM(1)

CALL TV2CM(2)

CALL ADDI2(3,1,2,2)

CALL ADDI2(4,1,2,2)

CALL ADDI2(1,3,4,2)

CALL DSPBW(1)

RETURN

END
SUBROUTINE TV2CM(CMIMAG)

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

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

LANGUAGE:

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

LIMITATIONS:

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

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

SUBPROGRAMS CALLED:

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

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
0045 C summer fellowship.
0047 C
0047 C***REVISION HISTORY:
0048 C
0049 C
0050 C***LOCAL VARIABLES:
0051 C
0052 INTEGER IMLO, IMHI ! limits on COMTAL image memory numbers.
0053 INTEGER TVIMAGE ! the COMTAL image associated with the video camera.
0054 C
0055 LOGICAL RANGE ! function that determines if the 1st argument
0056 C ! is within the 2nd and 3rd arguments, inclusive.
0057 CHARACTER*1 DIGIT ! function that returns '0'..'9' according
0058 C ! to integer input 0..9.
0059 INTEGER TERM ! logical unit for terminal output.
0060 C
0061 INTEGER IBUF(128) ! buffer for CMMND COMTAL command strings.
0062 CHARACTER*255 CBUF ! overlays IBUF.
0063 EQUIVALENCE (IBUF,CBUF)
0064 INTEGER TVIMAG ! COMTAL image associated with video image;
0065 C ! arbitrarily set to 5 in this lab.
0066 C
0067 C***INITIALIZATIONS:
0068 C
0069 DATA IMLO/1/, IMHI/4/ ! arbitrary convention for our lab.
0070 DATA TVIMAGE/5/ !
0071 DATA TERM/1/
0072 C
0073 C***PROCESSING:
0074 C
0075 IF (.NOT. (RANGE(CMIMAG,IMLO,IMHI))) GOTO 8081 ! error return
0076 C
0077 C
0078 C let @C and @V be the digits associated with CMIMAG and TVIMAG;
0079 C then the following CMMND string is expanded by the COMTAL into:
0080 C
0081 C
0082 CBUF = 'D I '//DIGIT(TVIMAGE)':' SI '//DIGIT(CMIMAG)':' D R '
0083 CALL CMMND(IBUF, 15)
0084 CALL DSPBU(CMIMAG)
0085 RETURN
0086 C
0087 C***ERROR RETURNS:
0088 C
0089 8081 WRITE(TERM,8003) CMIMAG, IMLO, IMHI
0090 8003 FORMAT(' THE COMTAL IMAGE NUMBER,' //I4,': IS OUT OF RANGE:',..214)
0091  GOTO 8900
0092  C
0093  C
0094  8900 WRITE(TERM,8901)
0095  8901 FORMAT('TV2CM FAILS. NO DIGITIZING TAKES PLACE.')
0096  C
0097    RETURN
0098  END
***SUBROUTINE WAIT***

***PARAMETERS: NONE***

***INTRODUCTION:***

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

***LANGUAGE:***

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

***SUBPROGRAMS CALLED: NONE***

***WRITTEN BY:***

KEITH MILLER, NASA-Langley ASEE fellow, 1984

***REVISION HISTORY: NONE***

***LOCAL VARIABLES:***

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

***INITIALIZATIONS:***

DATA TERM/1/

***PROCESSING:***

WRITE(TERM, 1001)
1001 FORMAT(' PUSH <CR> TO CONTINUE.'),
READ(TERM, 2001) IDUMMY
2001 FORMAT(I2)
RETURN
END
&WIPGR T=00004 IS ON C00021 USING 00012 BLKS R=0000

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

C***INITIALIZATIONS:

DATA TERM/I/
DATA GRLO/I,GRHI/4/

C***PROCESSING

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

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

CALL CMN2('SUB GR '//'DIGIT(GRNUM))
RETURN

C***ERROR RETURN:

8001 WRITE(TERM, 8003) GRNUM, GRLO, GRHI
8003 FORMAT( ' THE GRAPHICS NUMBER','.13.', OUT OF RANGE:','.214)

8900 WRITE(TERM, 8901)
8901 FORMAT( ' DSPGR fails. No action taken on command.' )
RETURN

END
SUBROUTINE WRGLN(GRAPH, LINE, ONOFFS)

C
C_PARAMETER DECLARATIONS:

INTEGER GRAPH ! COMTAL image graphics 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
C_INSTRUCTION:

This subroutine writes a Graphics Line to the COMTAL. 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
C_LANGUAGE:

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

C
C_LIMITATIONS:

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

C
C_SUBPROGRAMS CALLED:

NAME SOURCE LOAD REMARKS
------ ------ ------ --------------------------
RANGE &RANGE %RANGE logical function that determines if its
first parameter is within the last two parameters.

C
C_WRITTEN BY:

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

C
C_REVISION HISTORY:

C
C_LOCAL VARIABLES:

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

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 B001  ! 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, 8003) GRAPH, GRLO, GRHI
FORMAT(' GRAPHICS NUMBER:', 13, ' OUT OF RANGE:', 212, '.')
GOTO 8900

GOTO 8900

WRITE(TERM, 8901)
FORMAT(' WRGLN FAILS, NO TRANSFER. ')
RETURN
END
SUBROUTINE WRPGRAPH(GRAPH, XCOORD, YCOORD, VALUE)

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

INTRODUCTION:

This subroutine writes a Graph Point to the COMTAL. The value of the graphics is an on/off decision, represented in the call by an integer that must be a 1 or 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 RTE-6/VM.

LIMITATIONS:

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

SUBPROGRAMS CALLED:

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

WRITTEN BY:

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

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

**INITIALIZATIONS:**

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

**PROCESSING**

1. IF (.NOT.(RANGE(GRAPH, GRLO, GRHI))) GOTO 8001 ! Error return
2. IF (.NOT.(RANGE(XCOORD, LNLO, LNHI))) GOTO 8101 ! Error return
3. IF (.NOT.(RANGE(YCOORD, LNLO, LNHI))) GOTO 8201 ! Error return
4. IF (.NOT.(RANGE(VALUE, BTLO, BTHI))) GOTO 8301 ! Error return

**Programming note:**

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

- Read the COMTAL line (horizontal) that contains the point in question:
- Change the single bit that has been selected:
- WORD = (XCOORD/16) + 1
- BIT = (16*WORD) - XCOORD - 1
C
0092 IF (VALUE .EQ. 0) GOTO 1000
0093 ELSE...VALUE .EQ. 1
0094 BITS(WORD) = IBSET(BITS(WORD),BIT)
0095 GOTO 2000
0096 THEN...VALUE .EQ. 0
0097 1000 BITS(WORD) = IBCLR(BITS(WORD),BIT)
0098 GOTO 2000
0099 C
0100 Write the graph line with one changed graphics to COMTAL
0101 C
0102 2000 CALL EXEC(2,368+1008.BITS.32.(GRAPH-1)*2048 + 512 + YCOORD)
0103 RETURN
0104 C
0105 ***ERROR RETURNS
0106 C
0107 8001 WRITE(TERM, 8003) GRAPH, GRLO, GRHI
0108 8003 FORMAT(' GRAPH NUMBER:, I3, ' OUT OF RANGE:', 212, '.')
0109 GOTO 8900
0110 C
0111 8101 WRITE(TERM, 8103)XCOORD, LNLO, LNHI
0112 8103 FORMAT(' X COORDINATE:, I4, ' OUT OF RANGE:', 214, '.')
0113 GOTO 8900
0114 C
0115 8201 WRITE(TERM, 8203)YCOORD, LNLO, LNHI
0116 8203 FORMAT(' Y COORDINATE:, I4, ' OUT OF RANGE:', 214, '.')
0117 GOTO 8900
0118 C
0119 8301 WRITE(TERM, 8303) VALUE, BTLO, BTHI
0120 8303 FORMAT(' BIT VALUE:, I4, ' OUT OF RANGE:', 214, '.')
0121 GOTO 8900
0122 C
0123 8900 WRITE(TERM, 8901)
0124 8901 FORMAT(' WRGPT FAILS. NO TRANSFER.')
0125 RETURN
0126 END
SUBROUTINE URIL2(IMAGE, LINE, INTS)

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

0013 **INTRODUCTION:

This subroutine, Write Image Line 42, writes a line of pixels to the
COMTAL. The input array INTS has a two byte integer for each pixel,
but the COMTAL only uses the lower order byte of each integer. Therefore,
URIL2 strips off the upper byte before sending the pixels to the COMTAL.
URIL2 is very similar to URILN, which writes out a line of bytes.

**LANGUAGE:

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

**LIMITATIONS:

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

**SUBPROGRAMS CALLED:

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

**WRITTEN BY:

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

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

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

- Integer Term
  - the logical unit for terminal output

- Integer IMLO, IMHI
  - the limits on COMTAL monochrome image numbers

- Integer LNLO, LNHI
  - the limits on COMTAL image line numbers

- Integer INDEX
  - indexes into arrays in a loop

**INITIALIZATIONS:**

- DATA TERM/
- DATA IMLO/IMHI/4/
- DATA LNLO/LNHI/511/

**PROCESSING**

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

- strip off the upper bytes of the pixel values in INTS

- DO 1000 INDEX=1,512
- CPIX(INDEX) = CHAR(INTS(INDEX)) ! CPIX overlays PIXELS
- 1000 CONTINUE

**Programming note:**

- The EXEC call is explained in detail in the HP Programmer's Reference Manual for RTE-6/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: 368 identifies the resident DVR41 driver, and 1008 identifies the line transfer operation transferred, and the fourth parameter gives PIXELS' length in words.
C (256); and the final parameter is a COMTAL command code for the transfer.

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

RETURN

C***ERROR RETURNS

WRITE(TERM, 0003) IMAGE, IMLO, IMHI
FORMAT(' IMAGE NUMBER: ', L3, ' OUT OF RANGE: ', 2I2, '.')
GOTO 0000

WRITE(TERM, 0103) LINE, LNLO, LNHI
FORMAT(' LINE NUMBER: ', L4, ' OUT OF RANGE: ', 2I3, '.')
GOTO 0100

WRITE(TERM, 0901)
FORMAT(' WRIL2 FAILS. NO TRANSFER. ')
SUBROUTINE WRILN(IMAGE, LINE, PIXELS)

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

INTEGER PIXELS(256) ! 512 bytes (pixels) to be transferred

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

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

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

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

The code on which this subprogram is based was written by NETTIE D. FAULCON, July, 1983. This modification is by KEITH MILLER, June, 1984.
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/I/, IMHI/4/
DATA LNLO/0/, LNHI/511/

C***PROCESSING

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

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

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

RETURN

C***ERROR RETURNS

WRITE(TERM, 8003) IMAGE, IMLO, IMHI
WRITE(TERM, 8003) IMAGE, IMLO, IMHI
GOTO 8900

WRITE(TERM, 8103) LINE, LNLO, LNHI
WRITE(TERM, 8103) LINE, LNLO, LNHI
GOTO 8900

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

INTEGER IMAGE ! COMTAL image number to be written to.
INTEGER XCOORD, YCOORD! point where new value is to be written to.
INTEGER VALUE ! 0-255 pixel value to be written to image pt.

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

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

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.

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.
**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)` an overlay for PIXELS buffer.
- `EQUIVALENCE (PIXELS, BYTES)`.
- `INTEGER IHOLD` hold an integer for byte conversion.
- `CHARACTER*1 CHAR(2)` an overlay for IHOLD.
- `EQUIVALENCE (IHOLD, CHAR(2))`.

**INITIALIZATIONS:**

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

**PROCESSING**

```fortran
0859 IF (.NOT.(RANGE(IMAGEX, IMLO, IMHI))) GOTO 8081 ! error return
0860 IF (.NOT.(RANGE(IMAGEX, IMLO, IMHI))) GOTO 8081 ! error return
0861 IF (.NOT.(RANGE(YCOORD, LNLO, LNHI))) GOTO 8181 ! error return
0862 IF (.NOT.(RANGE(VALUE, PXLO, PXHI))) GOTO 8201 ! error return

Programming note:

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

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

- `CALL EXEC(1, 36B+101D, PIXELS, 255, (IMAGE-1)*2048 + YCOORD)`.

Change the single byte that needs changing:

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

INTEGER IMAGE
INTEGER UPLFX, UPLFY
INTEGER XDIM, YDIM
INTEGER INPARA(XDIM, YDIM)

**INTRODUCTION:**

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.

**LANGUAGE:**

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

**LIMITATIONS:**

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

**SUBPROGRAMS CALLED:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Source</th>
<th>Load</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDIL2</td>
<td>&amp;RDIL2</td>
<td>XRDIL2 transfers a line of bytes from a COMTAL image into an HP array of integers.</td>
<td></td>
</tr>
<tr>
<td>WRIL2</td>
<td>&amp;WRIL2</td>
<td>XWRIL2 transfers a buffer of integers to a COMTAL image as a line of bytes (1 byte/integer).</td>
<td></td>
</tr>
</tbody>
</table>
C. RANGE &RANGE &RANGE logical function that determines of its first
parameter is within the last two parameters
C. CHAR ---- ---- HP FORTRAN77 intrinsic function: converts a
2 byte integer into a one byte char (removes
high order byte).
C
C***WRITTEN BY:
C
C The code on which this subprogram is based was written by
C NETTIE D. FAULCON. July, 1983. This code was written by
C
C***REVISION HISTORY:
C
C***LOCAL VARIABLES:
C
C INTEGER IMROW, IMCOL | a location in the image
C INTEGER ARAROW, ARACOL | a location in INPARA
C INTEGER ICHAR | intrinsic byte to integer conversion
C LOGICAL RANGE | function that ascertains if the first
C parameter is within the last two parameters.
C INTEGER IMLO, IMHI | limits on COMTAL image numbers
C INTEGER LNLO, LNHI | limits on COMTAL pixel coordinates
C INTEGER DIMLIM,LNHII | limit on the dimensions of INPARA
C INTEGER TERM | logical unit for terminal output
C INTEGER IBUF(512) | buffer to hold COMTAL horizontal line
C
C***INITIALIZATIONS:
C
C DATA TERM/1/
C DATA IMLO/1/, IMHI/4/
C DATA LNLO/6/, LNHI/511/
C DATA DIMLIM/64/
C
C***PROCESSING
C
C IF (.NOT.(RANGE(Image, IMLO, IMHI )))GOTO 6001 | error return
C IF (.NOT.(RANGE(XDIM, 1, DIMLIM )))GOTO 8101 | error return
C IF (.NOT.(RANGE(YDIM, 1, DIMLIM )))GOTO 6201 | error return
C IF (.NOT.(RANGE(UPLFX, LNLO,LNHI )))GOTO 8301 | error return
C IF (.NOT.(RANGE(UPLFY, LNLO,LNHI )))GOTO 8401 | error return
C IF (.NOT.(RANGE(UPLFX+XDIM-1,LNLO,LNHI )))GOTO 6501 | error return
C IF (.NOT.(RANGE(UPLFY+YDIM-1,LNLO,LNHI )))GOTO 8601 | error return

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

```fortran
IMROW = UPLFY
DO 2000 ARAROW = 1, YDIM
   CALL RDIL2(IBUF, IMAGE, IMROW)
   IMCOL = UPLFX + 1 ! the next line initializes the column pointer;
   ! the "+1" is necessary because COMTAL image
   ! coordinates range from 0 to 511 and the
   ! FORTRAN array indices range from 1 to 512.
DO 1000 ARACOL = 1, XDIM
   IBUF(IMCOL) = INPARA(ARAROW, ARACOL)
   IMCOL = IMCOL + 1 ! increment for next 1000 loop pass
1000 CONTINUE
   CALL WRL2(IMAGE, IMROW, IBUF)
   IMROIJ = IMROW + 1 ! increment for next loop pass
2000 CONTINUE
RETURN ! successful termination

***ERROR RETURNS:

0801 WRITE(TERM, 8003) IMAGE, IMLO, IMHI
0803 FORMAT( ' IMAGE NUMBER:', I3, ' OUT OF RANGE:', 214, '.' )
GOTO 8900

0818 WRITE(TERM, 8103) XDIM, 1, DIMLIM
0816 FORMAT( ' X DIMENSION:', I4, ' OUT OF RANGE:', 215, '.' )
GOTO 8900

0828 WRITE(TERM, 8203) YDIM, 1, DIMLIM
0826 FORMAT( ' Y DIMENSION:', I4, ' OUT OF RANGE:', 215, '.' )
GOTO 8900

0838 WRITE(TERM, 8303) UPLFX, LNLLO, LNHI
0836 FORMAT( ' X COORDINATE FOR CORNER:', I4, ' OUT OF RANGE:', 215, '.' )
GOTO 8900

0848 WRITE(TERM, 8403) UPLFX, LNLLO, LNHI
0846 FORMAT( ' Y COORDINATE FOR CORNER:', I4, ' OUT OF RANGE:', 215, '.' )
GOTO 8900

0858 WRITE(TERM, 8503) UPLFX, XDIM, LNLLO, LNHI
0856 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
```

155
C
0138 8661 WRITE(TERM, 8603) UPLFY, YDIM, LNLO, LNHI
0139 8603 FORMAT(5 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  C
&LUT T=00004 IS ON CR00021 USING 00004 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE LUT(LUTNUM, TABLE)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 PARAMETER DECLARATIONS:
0006 C
0007 INTEGER LUTNUM ! the number of the COMTAL look-up table (called
0008 INTEGER TABLE(256)) ! "function memory" in the COMTAL literature).
0009 INTEGER TABLE(256) ! notice that 0 maps to TABLE(1), 1 maps to
0010 ! TABLE(2), ..., and 255 maps to TABLE(256).
0011 C
0012 C
0013 INTRODUCTION:
0014 C
0015 C The subroutine Write Look-Up Table (LUT) establishes a COMTAL mapping
0016 C from the integers 0-255 to the elements in TABLE. This LUT can be
0017 C used for grey level enhancements in the COMTAL. A similar subroutine
0018 C called WRPSU is used to establish a pseudocolor look-up table. This
0019 C routine is only used for grey scale look-up tables.
0020 C
0021 LANGUAGE:
0022 C
0023 C FORTRAN 77, the HP-1000 version for RTE-6/VM.
0024 C
0025 LIMITATIONS:
0026 C
0027 C Although 8 bits are sufficient for the look-up table values, full
0028 C integers are used in TABLE. This format is dictated by the COMTAL
0029 C conventions as given in section 5.2.3.1.
0030 C
0031 SUBPROGRAMS CALLED:
0032 C
0033 C
0034 C
0035 C
0036 C
0037 C
0038 SUBWRITTEN BY:
0039 C
0040 C
0041 C
0042 C
0043 C
0044 C

name | source | load | remarks
----- | ------ | ----- | --------------------------------------------------
RANGE | $RANGE$ | $X RANGE$ | logical function which determines if its 1st
parameter is within its 2nd and 3rd inclusive.

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:

C  LOGICAL RANGE  ! logical function which determines if its 1st parameter is within its 2nd and 3rd inclusive.
C  INTEGER TERM  ! logical unit for terminal output
C  INTEGER LUTLO, LUTHI  ! limits for COMTAL function memories

C**INITIALIZATIONS:

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

C**PROCESSING

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

Programming notes:

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

The first parameter to EXEC identifies the EXEC command as a write command. The second parameter identifies the resident HP driver (368) and gives the code (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), and the fourth parameter gives the length of TABLE in words.

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

CALL EXEC( 2, 368+208, TABLE, 256, ((LUTNUM-1)*256) )

RETURN

C**ERROR RETURN

C  WRITE(TERM, 8803) LUTNUM, LUTLO, LUTHI
C  FORMAT(' THE FUNCTION MEMORY NUMBER is OUT OF RANGE! ', 14, '.', 214, '.')

C  WRITE(TERM, 8901)
C  FORMAT(' LUT FAILS. NO TRANSFER TO COMTAL. ')

END
**SUBROUTINE WRPSU(TABLE)**

**PARAMETER DECLARATIONS:**

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

**INTRODUCTION:**

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

**LANGUAGE:**

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

**LIMITATIONS:**

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

**SUBPROGRAMS CALLED:**

NONE.

**WRITTEN BY:**

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

**REVISION HISTORY:**

NONE.

**LOCAL VARIABLES:**

NONE.
**PROGRAMMING NOTES:**

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

The first parameter to EXEC identifies the EXEC command as a write command. The second parameter identifies the resident HP driver (3GB) and gives the code (300B) that identifies this operation, a transfer from the COMTAL pseudocolor table.

The third parameter gives the array that will hold the values, and the fourth parameter gives the length of TABLE in words.

The fifth parameter is a COMTAL code that is described bit by bit in the User's Manual. The DVR41 driver takes care of all the bits except B&9 which identify the color to be transferred.

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

- CALL EXEC( 2, 36B+300B, TABLE(1), 256, 1*256 ) ! red
- CALL EXEC( 2, 36B+300B, TABLE(257), 256, 0*256 ) ! green
- CALL EXEC( 2, 36B+300B, TABLE(513), 256, 2*256 ) ! blue
- RETURN
- 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.

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

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

Function that ascertains if its first parameter is between (inclusive) its last 2 parameters. The 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/1/
0052 DATA LNLO/0/, LNHI/511/
0053 C
0054 C***PROCESSING
0055 C
0056 IF (.NOT.(RANGE(XCOOR, LNLO, LNHI))) GOTO 8881 ! error return
0057 IF (.NOT.(RANGE(YCOOR, LNLO, LNHI))) GOTO 8810 ! error return
0058 C
0059 Programming note:
0060 C
0061 C The EXEC call is explained in detail in the
0062 C HP Programmer's Reference Manual for RTE-6/VM, p.2-19ff. This
0063 C transfer function for the COMTAL is discussed in the
0064 C COMTAL User's Manual, Section 5.2.4. In the EXEC call
0065 C that follows, the HP resident driver called DVR41 is called as
0066 C follows: the first parameter (2) signifies a write; the
0067 C second parameter is in two parts: 36B identifies the resident
0068 C DVR41 driver, and 400B identifies the target transfer operation
0069 C of that driver; the third parameter (IBUF) contains the two coordinates
0070 C to be transferred, and the fourth parameter gives the length of the
0071 C buffer in words. The last parameter is ignored.
0072 C
0073 IBUF(1) = XCOOR
0074 IBUF(2) = YCOOR
0075 CALL EXEC(2, 36B+400B, IBUF, 2, IDUMMY)
0076 RETURN
0077 C
0078 C***ERROR RETURNS
0079 C
0080 C 8001 WRITE(TERM, 8003) XCOOR, LNLO, LNHI
0081 C 8003 FORMAT(' XCOOR', 15, ' OUT OF RANGE: ', 214,'.')
0082 C
0083 GOTO 8900
0084 C
0085 C 8101 WRITE(TERM, 8103) YCOOR, LNLO, LNHI
0086 C 8103 FORMAT(' YCOOR', 15, ' OUT OF RANGE: ', 214,'.')
0087 C
0088 GOTO 8900
0089 C
0090 C 8900 WRITE(TERM, 8901)
0091 C 8901 FORMAT(' WRITE FAILS. NO TRANSFER.')
0092 C
0093 RETURN
0094 C
0095 END
COMMENT

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

C

TEST THE ADDPROGRAM ADDIM (ADD IMAGES)

C

KEITH MILLER 7/17/84

C

LOAD MODULES: XTADDI, XADDIM, XCMND, XRANGE, XDIGIT

C

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

C

CALL ADDIM(1,0,2,1) ! "

C

CALL ADDIM(2,0,1) ! "

C

CALL ADDIM(3,1,2,1)

C

STOP

C

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
PROGRAM TCLRG

C TEST CLEAR GRAPHICS

C KEITH MILLER, 7/2/84

C LOAD MODULES: %TCLRG, %CLRGR, %RANGE, %CMNND, %DIGIT

INTEGER IBUF(128)

CHARACTER*20 CBUF ! OVERLAYS THE FIRST 10 ELEMENTS OF IBUF

EQUIVALENCE (IBUF,CBUF)

CBUF = 'CLEAR GR 2'

CALL CLRGR(0)

CALL CLRGR(1)

CALL CMNND(IBUF, 10)

CALL CLRGR(5)

STOP

END
PROGRAM TCLRI

TEST CLEAR IMAGE

KEITH MILLER, 7/2/84

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

INTEGER IBUF(128)

CHARACTER*20 CBUF ! OVERLAYS THE FIRST 10 ELEMENTS OF IBUF

EQUIVALENCE (IBUF, CBUF)

CBUF = 'CLEAR IMAGE 2'

CALL CLRIM(0)

CALL CLRIM(1)

CALL CMDND(IBUF, 13)

CALL CLRIM(5)

STOP

END
PROGRAM TCMM2

TEST CMDND AGAIN

KEITH MILLER, 7/12/84

LOAD MODULES _TCMM2, _CMDND, _WAIT

INTEGER INNUM, I, IBUF(128)

CHARACTER*255 CBUF

EQUIVALENCE (IBUF, CBUF)

DO 1000 I = 0, 256

WRITE(1,500) I

500 FORMAT('TESTING CHARACTER ', I4)

CALL WAIT

CBUF = 'G 2 PRO 1'

CALL CMDND(IBUF, 9)

CBUF = CHAR(I)

CALL CMDND(IBUF, 1)

1000 CONTINUE

STOP

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')

INTEGER IBUF

READ(1,10) IBUF

FORMAT('ENTER ASCII STRING')

READ(1,20) IBUF

N = ITLOG() IF (N .EQ. 0) GOTO 999

CALL CMMND(IBUF,N)

GO TO 5

999 END
&TCNT T=03004 IS ON C000021 USING 00002 BLKS R=0000

0001 PROGRAM TCNT
0002 C
0003 C TEST THE SUBROUTINE COUNT
0004 C
0005 C KEITH MILLER 7/11/84
0006 C
0007 C LOAD MODULES: TCNT, COUNT, RANGE, NRDIL2
0008 C
0009 INTEGER 4 I4BUF(256)
0010 INTEGER INDEX
0011 C
0012 CALL COUNT(I4BUF, 1)
0013 DO 1000 INDEX = 0, 31
0014 WRITE(1,999) (I4BUF((INDEX*8)+J), J=1,8)
0015 999 FORMAT(8I9)
0016 1000 CONTINUE
0017 C
0018 STOP
0019 END
&TCONS T=00004 IS ON CR00021 USING 00002 BLKS R=0000

0001 PROGRAM TCONS
0002 C
0003 C TEST CONCATENATION OF CHARACTER STRING, FORTRAN 77
0004 C
0005 INTEGER IHOLD
0006 CHARACTER*1 CHOLD(2)
0007 EQUIVALENCE (IHOLD,CHOLD)
0008 CHARACTER*5 STRING
0009 C
0010 IHOLD = 6 + 608
0011 STRING = '**' // CHOLD(2) // '**'
0012 WRITE(1,1000) CHOLD(2),STRING
0013 1000 FORMAT(' LETTER=', A1, ', STRING=', A5, ',')
0014 STOP
0015 END
&TCOPY T=00004 IS ON CR00021 USING 00001 DLKS R=0000

0001 PROGRAM TCOPY
0002 C TEST ICOPY
0003 C LOAD MODULES REQUIRED: TCOPY, ICOPY, CMND, RANGE,
0004  R, DIGIT
0005 C CALL ICOPY(1,2)
0006 C CALL ICOPY(5,1) ! SHOULD GIVE OUT OF RANGE ERROR
0007 C CALL ICOPY(2,-3) ! SHOULD GIVE OUT OF RANGE ERROR
0008 C STOP
0009 C END

172
PROGRAM TDIGI

C TEST DIGIT
C KEITH MILLER, 7/2/84
C
C LOAD MODULES: TDIGI, DIGIT, RANGE
C
C INTEGER INT
C CHARACTER*1 HCHAR
C
DO 1000 INT = -1, 12
C
CALL DIGIT(HCHAR, INT)
C
WRITE(1,501) INT, HCHAR
C
501 FORMAT(' INTEGER INPUT AND CHARACTER OUTPUT: ',I13,'A1,'')
C
1000 CONTINUE
C
STOP
C
END
PROGRAM TDSP

C TEST DSPBW AND DSPCL

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

CALL DSPBW(1)
CALL WAIT
CALL DSPBW(2)
CALL WAIT
CALL DSPBW(3)
CALL WAIT
CALL DSPBW(0)
CALL WAIT
CALL DSPCL(1, 2, 3, 8)
STOP
END
PROGRAM THIST

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
&THORM T=60004 IS ON CR00021 USING 00002 BLKS R=0000

0001 PROGRAM THORM
0002 C
0003 C TEST THE PROCEDURE NORML.
0004 C
0005 C KEITH MILLER JULY 6, 1983
0006 C
0007 C LOAD MODULES: %THORM, %NORML, %HIL0, %RDIL2, %WRIL2, %RANGE
0008 C
0009 C CALL NORML(2)
0010 STOP
0011 END
PROGRAM TNOTE

TEST THE SUBROUTINES NOTES AND NOTE2

KEITH MILLER, 7/26/84

LOAD MODULES: \texttt{TNOTE, NOTES, NOTE2, DSPGR, XUTAR, CMMD2, CMMD, DIGIT, RANGE, DELAY}

CHARACTER*255 \texttt{CBUF}

INTEGER \texttt{IBUF(128)}

EQUIVALENCE \texttt{(CBUF, IBUF)}

CALL \texttt{NOTE2(I,100,100,'B',3,'1.BLUE')}

CALL \texttt{NOTE2(I,100,200,'R',3,'2.RED')}

CALL \texttt{NOTE2(I,100,300,'G',3,'3.GREEN')}

CALL \texttt{NOTE2(I,100,400,'S',3,'4.SAME')}

\texttt{CBUF(1:11) = 'USING NOTES'}

CALL \texttt{NOTES(I,10,10,'R',1,\texttt{CBUF,11})}

CALL \texttt{NOTES(5,10,10,'R',1,\texttt{CBUF,11})}

CALL \texttt{NOTES(1,512,10,'R',1,\texttt{CBUF,11})}

CALL \texttt{NOTES(1,-1,10,'R',1,\texttt{CBUF,11})}

CALL \texttt{NOTES(1,10,512,'R',1,\texttt{CBUF,11})}

CALL \texttt{NOTES(1,10,-1,'R',1,\texttt{CBUF,11})}

CALL \texttt{NOTES(1,10,10,'R',1,\texttt{CBUF,11})}

CALL \texttt{NOTES(1,10,10,'R',17,\texttt{CBUF,11})}

CALL \texttt{NOTE2(2,120,240,'B',3,'USING NOTE2')}

CALL \texttt{NOTE2(2,120,240,'B',3,'USING NOTE2')}

CALL \texttt{NOTE2(2,512,240,'B',3,'USING NOTE2')}

CALL \texttt{NOTE2(2,-1,240,'B',3,'USING NOTE2')}

CALL \texttt{NOTE2(2,120,512,'B',3,'USING NOTE2')}

CALL \texttt{NOTE2(2,120,-1,'B',3,'USING NOTE2')}

CALL \texttt{NOTE2(2,120,240,'B',3,'USING NOTE2')}

CALL \texttt{NOTE2(2,120,240,'B',0,'USING NOTE2')}

CALL \texttt{NOTE2(2,120,240,'B',17,'USING NOTE2')}

CALL \texttt{DSPGR(1)}

CALL \texttt{DSPGR(2)}

STOP

END
PROGRAM TPNT

C TEST THE SUBROUTINE PAINT

C KEITH MILLER, 7/16/84

C LOAD MODULES: _TPNT, _PAINT, _RANGE, _URIRC, _RDTAR,
  _RDIL2, _URIL2, _CMDND, _DSPBW

CALL PAINT(1,35, 200)

STOP

END
&TPROF T=E0004 IS ON CR00021 USING 00002 BLKS R=0000

0001   PROGRAM TPROF
0002   C
0003   C   TEST PROFL SUBROUTINE
0004   C
0005   C   KEITH MILLER, 7/12/84
0006   C
0007   C   LOAD MODULES: %TPROF, %PROFL, %RANGE, %CMND, %WAIT,
0008   C   %DIGIT
0009   C
0010   C   CALL PROFL(1,1)
0011   C   WRITE(1,1001)
0012   1001 FORMAT(' SUCCESSFULLY RETURNED TO CALLER OF PROFILER')
0013   C   STOP
0014   C   END
PROGRAM TRANG
C TESTS THE RANGE FUNCTION
C LOAD MODULES REQUIRED: TRANG, RANGE
LOGICAL RANGE, ANSWER
WRITE(1,1)
1 FORMAT(' SHOULD BE T F ERROR-F')
WRITE(1,2) RANGE(2,1,3)
2 FORMAT(' ', L1)
WRITE(1,2) RANGE(1,2,3)
ANSWER = RANGE(3,2,1)
WRITE(1,2) ANSWER
STOP
END
&TRDTA T=00004 IS ON CR00021 USING 00002 ELKS R=0000

0001       PROGRAM TRDTA
0002       C       TEST THE SUBROUTINE RDTAB
0003       C       KEITH MILLER, 7/19/84
0004       C       LOAD MODULES: %TRDTA, %RDTAB, %RANGE
0005       C       INTEGER TABLE(16), INDEX, WHICH
0006       C
0007       C       DO 2000 WHICH = 1,4
0008       C       CALL RDTAB(TABLE, WHICH, 0)
0009       C       WRITE(1,999)WHICH, TABLE(1), TABLE(2), TABLE(3), TABLE(4)
0010       C       FORMAT(12, ', ', 409)
0011       C       2000 CONTINUE
0012       C       STOP
0013       C       END
0001        PROGRAM TSETV
0002        C
0003        C  TEST THE SETV SUBROUTINE
0004        C
0005        C  KEITH MILLER, 7/17/84
0006        C
0007        C  LOAD MODULES: _TSETV, _SETV, _RANGE, _DIGIT, _CMMND, _DSPBW
0008        C
0009        CALL SETV(2) !SHOULD BE AN ERROR
0010        CALL SETV(11) !SHOULD BE AN ERROR
0011        CALL SETV(5) !SHOULD WORK
0012        C
0013        STOP
0014        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

2000 CONTINUE

IBUF(511) = I3

IBUF(512) = I3

DO 3000 INDEX=0, 511

CALL WRIL2(1, INDEX, IBUF)

3000 CONTINUE

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, %CMND

CALL SUBIM(1.3,2)!
CALL WAIT
CALL SUBI2(1.3,2)!

STOP
END
PROGRAM TTHRS

TESTS THE SUBROUTINE THRESHOLD

KEITH MILLER 7/25/84

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

INTEGER THRESH! THRESHOLD FROM USER

INTEGER IN, OUT! TWO IMAGES

WRITE(1,1001)

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

READ(1,1003) OUT

1003 FORMAT(I)

WRITE(1,1005)

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

READ(1,1005) IN

WRITE(1,1007)

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

READ(1,1009) THRESH

1009 FORMAT(I)

CALL THRSH(OUT, IN, THRESH)

STOP

END
\*TTSTI T=0004 IS ON CR00021 USING 00001 BLKS R=0000

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

0001      PROGRAM TTV2C
0002      C
0003      C   TEST THE SUBROUTINES TV2CM AND TV2C4
0004      C
0005      C   KEITH MILLER, 7/17/84
0006      C
0007      C   LOAD MODULES: TTV2C, TV2C4, TV2CM, XCMND, XDSPBW,
0008      C   XRANGE, XDIGIT, XWAIT, XADD12, XDSPVD
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
PROGRAM TWAIT

TESTING THE WAIT SUBROUTINE

KEITH MILLER, JUNE 8, 1984

INTEGER IERR, IX, IY

WRITE(1,1000)

1000 FORMAT(' START WAIT TEST')

CALL WAIT(IERR)

CALL KMRTA(IX, IY, IERR)

WRITE(1, 2000)

2000 FORMAT(' END WAIT TEST')

STOP

END
&TWIFE T=C0004 IS ON CR00021 USING 02002, BLKS R=0000

0001 PROGRAM TWIFE
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: %TWIFE, %DSPGR, %WIPGR, %WAIT,
0008 C %RANGE, %DIGIT, %CMMP2
0009 C
0010 CALL DSPGR(1)
0011 CALL DSPGR(2)
0012 CALL DSPGR(3)
0013 C
0014 CALL WIPGR(1)
0015 C
0016 CALL WIPGR(3)
0017 CALL WAIT
0018 CALL WIPGR(2)
0019 CALL WAIT
0020 CALL WIPGR(1)
0021 CALL WAIT
0022 CALL WIPGR(1)
0023 STOP
0024 END
PROGRAM TXFDS

TEST PROGRAM FOR BUFD%. CLFDS

KEITH MILLER, 7/12/84

LOAD MODULES: %TXFDS, %BUFD%, %CLFDS, %RANGE, %WRILN, %CMND, %DIGIT

INTEGER FLNAME(3)

CHARACTER*6 CNAME

EQUIVALENCE (FLNAME,CNAME)

CNAME = 'CFXRAY'

CALL BUFD%(1, FLNAME)

CNAME = 'CFMAND'

CALL CLFDS(1,3,8, FLNAME)

STOP

END
PROGRAM TXGLN

C TEST URGLN AND RDGLN

C LOAD MODULES REQUIRED: %TXGLN,%RDGLN,%URGLN,%RANGE

INTEGER LINE ! number of line in image being processed

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

C ! note that 16 ONOFF bits fit in one INTEGER

DO 2000 LINE = 1, 512
  CALL RDGLN(IBUF, 2, LINE)

  CALL WRGLN(1, LINE, IBUF)

2000 CONTINUE

END
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, 1)
0013      CALL WRGPT(I, INDEX, 21-INDEX, 1)
0014      1000 CONTINUE
0015      C
0016      CALL WAIT
0017      C
0018      DO 2000 INDEX = 1, 20
0019      CALL WRGPT(I, INDEX, INDEX, 0)
0020      CALL WRGPT(I, INDEX, 21-INDEX, 0)
0021      2000 CONTINUE
0022      C
0023      DO 3000 INDEX=1,5
0024      CALL WAIT
0025      CALL RDTAR(I, IY)
0026      CALL RDGPT(IVALUE, I, IX, IY)
0027      WRITE(1, 2001) IX, IY, IVALUE
0028      2001 FORMAT( ' AT POINT ', 214, ' GRAPHICS VALUE=', I2, ' ')
0029      3000 CONTINUE
0030      C
0031      STOP
0032      END
CTXILN T=00004 IS ON CR00024 USING 00004 BLKS R=00000

0001 PROGRAM TXILN
0002 C
0003 C TEST URILN AND RDILN
0004 C
0005 C LOAD MODULES REQUIRED: %TXILN,%RDILN,%URILN,%RANGE
0006 C
0007 INTEGER LINE ! number of line in image being processed
0008 INTEGER IBUF(256) ! buffer to hold pixel values, one line/time
0009 C ! note that 2 pixels fit in one INTEGER
0010 C
0011 DO 2000 LINE = 0, 511
0012 CALL RDILN(IBUF, I, LINE)
0013 C
0014 CALL URILN(2, LINE, IBUF)
0015 2000 CONTINUE
0016 C
0017 C
0018 C

END
&TXIPT TXIPT T-00004 IS ON CR00021 USING 00002 BLKS R=0000

0001      PROGRAM TXIPT
0002      C
0003      C   TEST RDIPT AND WRIPT
0004      C
0005      C   KEITH MILLER, 6/21/84
0006      C
0007      C   LOAD MODULES: %TXIPT, %RDIPT, %WRIPT, %RANGE, %WAIT
0008      C
0009      C   INTEGER XCOORD, YCOORD, VALUE
0010      C
0011      C   CALL RDIPT(VALUE, 1, 200, 100)
0012      C   WRITE(1,1001) VALUE
0013      1001 FORMAT(‘ VALUE ’, I4 )
0014      C   CALL WRIPT(1,200,100,005)
0015      C   CALL RDIPT(VALUE, 1, 200, 100)
0016      C   WRITE(1,1001) VALUE
0017      C
0018      C   STOP
0019      C   END
0020
0021
0022
0023
0001 PROGRAM TXIRC
0002 C
0003 C TEST RDIRC (READ IMAGE RECTANGLE)
0004 C & WRIRC (WRITE IMAGE RECTANGLE)
0005 C
0006 C KEITH MILLER, JULY 3, 1984
0007 C
0008 C LOAD MODULES: %TXIRC, %RDIRC, %WRIRC, %RANGE, %RDILN, %WRILN
0009 C
0010 C INTEGER BUFFER(10,10)
0011 C INTEGER ROW, COL
0012 C
0013 C CALL RDIRC(BUFFER, 10, 10, 1, 0, 0)
0014 C CALL WRIRC(1, 100, 100, BUFFER, 10, 10)
0015 C STOP
0016 C END
0017
0018
PROGRAM TXLUT

TEST WRLUT AND RDLUT

KEITH MILLER, 7/10/84

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

INTEGER TABLE(256), TABLE2(256)

INTEGER INDEX

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

CALL WRLUT(1, TABLE)

CALL RDLUT(TABLE2, 1)

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

1999 FORMAT(14)

STOP

END
&TXPSU T=C0004 IS ON CR00021 USING 00004 BLKS R=0000

0001  PROGRAM TXPSU
0002  C
0003  C  TEST PSEUDO COLOR TABLE TRANSFERS: RDPSU AND URPSU
0004  C
0005  C  LOAD MODULES: %TXPSU, %RDPSU, %URPSU, %RANGE, %WAIT
0006  C
0007  C  KEITH MILLER, 7/10/84
0008  C
0009  C  INTEGER TABLE(768)
0010  INTEGER INDEX
0011  C
0012  CALL RDPSU(TABLE)
0013  C
0014  WRITE(1,999)
0015  999 FORMAT(' THE REDS:')
0016  DO 2000 INDEX=1,1256
0017    WRITE(1,999)TABLE(INDEX)
0018  999 FORMAT(I4)
0019  2000 CONTINUE
0020  C
0021  CALL WAIT
0022  C
0023  WRITE(1,2999)
0024  2999 FORMAT(' THE GREENS:')
0025  DO 4000 INDEX=257,512
0026    WRITE(1,3999)TABLE(INDEX)
0027  3999 FORMAT(I4)
0028  4000 CONTINUE
0029  C
0030  CALL WAIT
0031  C
0032  WRITE(1,4999)
0033  4999 FORMAT(' THE BLUES:')
0034  DO 6000 INDEX=513,768
0035    WRITE(1,5999)TABLE(INDEX)
0036  5999 FORMAT(I4)
0037  6000 CONTINUE
0038  C
0039  DO 1000 INDEX = 1,1256
0040    TABLE(INDEX) = 256-INDEX
0041  TABLE(INDEX+256) = INDEX
0042  TABLE(INDEX+512) = 122
0043  1000 CONTINUE
0044  C
0045    CALL WRPSU(TABLE)
0046    STOP
0047    END
PROGRAM TXTAR

TEST RDTAR AND WRTAR

KEITH MILLER, 6/21/84

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

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

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

DO 1000 INDEX = 1, 20

CALL RDTAR(XCOORD, YCOORD)
WRITE(1,500)XCOORD, YCOORD
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
BP DRIVER, DVR41, SOURCE CODE

ASMB.L

NAM DVR41

WRITTEN BY M. BROUKE COULTER COMPUTER CORPORATION

REWRITTEN BY R.W. BAGDADIAN HUGHES AIRCRAFT COMPANY.

WRITTEN NOV., 1979

REWRITTEN APR. 1980

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

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

SAMPLE CALL:

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

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

1: SIGNIFIES A READ OPERATION. (COMTAL -> HP)

2: SIGNIFIES A WRITE OPERATION. (HP -> COMTAL)

3: SIGNIFIES A CONTROL OPERATION: THIS MODE DOESN’T SEEM TO WORK IN OUR SYSTEM.

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

LU : LOGICAL UNIT NUMBER. THE LOGICAL UNIT NUMBER FOR THE COMTAL IS SET AT SYSTEM GENERATION. AT THE MOMENT IT IS 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...
ARE AVAILABLE. THE EXEC CALL STORES THE IFUNC CODE IN
IN BITS 6-10 OF WORD 6 OF THE EQUIPMENT TABLE (EQT6 IN
THIS DRIVER).

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

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

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

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

NOTES:
1. USES STANDARD INTERRUPTS FOR TRANSFER OF DEVICE COMMANDS AND TERMINATION SEQUENCES.
2. USES D.M.A. FOR TRANSFER OF DATA BLOCKS IF NECESSARY
3. THE TWO INTERFACE CARDS FOR THE COMTAL MUST BE MOUNTED IN CONSECUTIVE I/O SLOTS.
4. THE COMMAND CARD MUST BE THE LOW I/O SLOT OR SELECT CODE.
5. MANY BUGS WERE FOUND IN THE ORIGINAL AND THESE HAVE BEEN REMOVED AS WELL AS MAKING THE CODE A MORE EFFICIENT ROUTINE. R.W.B.

INITIATION SECTION FOLLOWS

141 NOP
110 JSB SETIO
111 CLB
112 LDA EOT6.I GET CONTROL SUBFUNCTION
113 AND =B3700
114 STA FUNC SAVE FUNCTION MEMORY
115 LDA EOT6.I GET CONTROL WORD OF REQUEST
116 AND =B3 ISOLATE THE FUNCTION
117 CPA =B2 CHECK FOR WRITE (SKIP IF NOT WRITE)
118 JMP RORWI IF REQUEST FUNCTION = 2 (WRITE)
119 LDB BIT15 SET READ FLAG FOR ACTUAL BIT USED
120 CPA =B1 CHECK FOR READ (SKIP IF NOT READ)
121 JMP RORWI IF REQUEST FUNCTION = 1 (READ)
122 * ANYTHING ELSE IS CONSIDERED A CONTROL REQUEST
123 LDA FUNC GET CONTROL WORD
124 SZA
125 JMP REJCT FOR NOW ONLY CONTROL REQUEST "0" IS VALID
126 CLR1 CLC DAT.C
127 CLR2 CLC CST.C
128 LDA =B4
129 JMP I.41.I
130 * 131 *
132 REJCT LDA =B2 SET A=2, FOR ILLEGAL CONTROL REQUEST 133 JMP I.41.I RETURN TO IOC
134 *
135 *
136 *
0137 ****** SETIO ROUTINE FOR INITIATION SECTION FOLLOWS
0138 SETIO NOP
0139 STA B  SAVE SELECT CODE FROM A (STATUS CARD SC)
0140 IOR CLC  OR IN "CLC" INSTRUCTION
0141 STA CLR2
0142 STA RORW1
0143 INA  INCREMENT TO HIGHER SELECT CODE (DATA CARD SC)
0144 STA CLR1
0145 STA RORW
0146 LDA LIA  "LIA INSTRUCTION" TO A
0147 IOR B  OR IN SELECT CODE
0148 STA RORW2
0149 LDA STC  "STC INSTRUCTION" TO A
0150 IOR B
0151 STA IOIL3  STORE INSTRUCTION AT LABEL "IOIL3"
0152 LDA OTA
0153 IOR B
0154 STA IOIL2
0155 LDA OTB
0156 INB  INCREMENT SELECT CODE IN B
0157 IOR B
0158 STA IOIL1
0159 JMP SETIO.I  RETURN
0160 ********************************************
0161 ********** READ OR WRITE REQUEST ************
0162 ********************************************
0163 * CHECK FOR COMTAL IN REMOTE
0164 *
0165 RORW1 STB RFLAG
0166 RORW CLC DAT.C  CLEAR CONTROL AND FLAG
0167 RORW1 CLC CST.C
0168 RORW2 LIA CST  GET STATUS
0169 AND NBSY  FIX FOR COMTAL VO/20  R.W.B.
0170 CPA NBSY  FIX FOR COMTAL VO/20  R.W.B.
0171 RSS  SKIP IF YES
0172 JMP RORW2  FIX FOR COMTAL VO 20  R.W.B.
0173 ********************************************
0174 ********** DETERMINE WHICH SUBFUNCTION ***********
0175 ********************************************
0176 *** NOTE THAT THE DISPLAY COMMAND AND TRACK BALL REQUESTS ARE
0177 * PERFORMED USING DEVICE INTERRUPTS.
0178 * HOWEVER IMAGE LINE, FUNCTION, AND PSEUDO-COLOR TRANSFERS ARE
0179 * PERFORMED USING DEVICE INTERRUPTS TO TRANSMIT THE COMMAND AND
0180 * ENDING SEQUENCE, BUT WITH A DMA TRANSFER IN BETWEEN.
0181 *
0182 LDA FUNC  GET CONTROL WORD
0183 * CPA DSPLV DISPLAY COMMAND REQUEST?
0185 JMP DSPLY
0187 * CPA IILTRV IMAGE LINE TRANSFER?
0188 JMP CHDCP
0189 * CPA FMTRV FUNCTION MEMORY TRANSFER?
0191 JMP CHDCP
0192 * CPA PCMTV PSEUDO-COLOR TRANSFER
0193 JMP CHDCP
0195 * CPA TBALV TRACK BALL REQUEST
0196 JMP TBALL
0198 * CPA SUPTV CHECK FOR TYPE 3 DMA SUPPORT CONTROL
0200 JMP CHDCP
0201 * CPA LINKV CHECK FOR LINK SUBFUNCTION
0202 JMP LINK
0204 * CPA LINKV CHECK FOR LINK SUBFUNCTION
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 IO1 GO TO START IO1 ROUTINE
0215 *
0216 ** PROCESS TRACK BALL REQUEST **************
0217 *
0218 *
0219 *
0220 TBAL 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 WTAL LDA =D2 SET STATE CONTROL VARIABLE.
0227 STA EOT12.I
0228 LDB EOT7.I
0229 LDA B.1 CONTENTS OF BUFFER TO A
0230 AND #B0777 MASK FOR BITS 0-8
0231 STA B
0232 LDA =B3 TRANSFER CODE OF 3 TO A
0233 JMP 101 GO START I/O
0234 *
0235 ************ READ TRACK BALL ************
0236 ***** ALSO READS LITTLE RED SWITCHES ON TRACK BALL APPARATUS.
0237 *
0238 RTBAL LDA =D5 SET FLAG TO TELL CONT. SECTION WHAT TO DO
0239 STA EQT12.I
0240 LDA =B3 TRANSFER CODE=3
0241 JMP 101 GO TO START I/O
0242 *
0243 *
0244 ********** CHECK FOR DCPC CHANNEL ASSIGNED **********
0245 *
0246 CHDCP DLD INTBA.I ACCESS DCPC ASSIGNMENT WORDS FROM THE
0247 * THE INTERRUPT TABLE WHOSE ADDRESS IS IN THE BASE PAGE
0248 * COMMUNICATIONS AREA.
0249 CPA EQT1 IS CHANNEL 1 ASSIGNED TO THIS DRIVER
0250 JMP CH1 IF YES
0251 CPB EQT1 IS CHANNEL 2 ASSIGNED TO THIS DRIVER
0252 JMP CH2 IF YES
0253 LDA =B5 SET RETURN CODE TO 5
0254 JMP 1.41.I RETURN TO IOC REQUESTING A DCPC CHANNEL
0255 CH1 LDB =B2 LOW SELECT CODE OF CHANNEL 1 TO B
0256 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 101  GO START I/O
0280                                                                                     
0281 * FUNCTION MEMORY OR PSUEDO COLOR TRANSFER                                                                 
0282                                                                                     
0283 PCMT2 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 IDPTN1 WORD
0291 IOR BIT13  MERGE IN 8000S COMPATIBILITY BIT
0292 IOR RFLAG  MERGE IN READ OR WRITE BIT
0293 FMTR2 STA B  PLACE CONTROL WORD IN B
0294 LDA =B2  SET TRANSFER CODE
0295 STA TRAN  SAVE TRANSFER CODE FOR LATER
0296 JMP 101
0297                                                                                     
0298 * SUPPORT FUNCTION BLOCK TRANSFER ( TR CODE = 3 )                                                                 
0299                                                                                     
0300 SUPPT LDA =D8  SET DMA TRANSFER RETURN STATUS FOR CONTINUATION SECTION
0301 STA EOT12.1 AND SAVE IT
0302 LDA 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 101  AND GO DO IT
0308 *
0309 * SPECIAL PROCESSING FOR TR 1 MODE 6 CALL
0310 *
0311 LINK LDA =D1  SET NO BLOCK TRANSFER
0312 STA EOT12.1  SAVE THIS
0313 LDB EQT9.1  GET COMMAND OPTION
0314 LDA =B1  SET TRANSFER CODE
0315 JMP 101  GO DO IT
0316 *
0317 *
0318 *
0319 * THIS ROUTINE STARTS THE DATA TRANSFER OF A COMMAND                             *
0320 * TO THE COMTAL
0321 *
0322 *
0323 101 NOP
0324 101L1 OTB DAT  PUT TRANSFER COMMAND ON DATA LINES
0325 101L2 OTA CST  PUT TRANSFER CODE ON COMMAND LINES
0326 101L3 STC CST.C SEND THE GO PULSE, START TRANSFER
0327 CLA  NOW RETURN TO IOC WITH
0328 JMP I.41.1 OPERATION INITIATED (A=B=OK)
0329 *  WHEN THE COMMAND IS RECEIVED BY THE COMTAL, AN
0330 *  INTERRUPT FROM THE COMTAL WILL CALL CIC AND THIS
0331 *  DRIVERS COMPLETION AND CONT. SECTION WILL EXEC.
0332 *
0333 *
0334 *
0335 *
0336 COMPLETION SECTION Follows
0337 *
0338 *
0339 *
0340 C.41 NOP
0341 JSB CS10
0342 LDA EQT1.I CHECK FOR SPURIOUS INTERRUPT
0343 AND #B7777 GET I/O REQUEST LIST POINTER
0344 SZA IS A REQUEST IN PROGRESS
0345 JMP COMP2 IF YES GO PROCESS REQUEST
0346 STA EOT15.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 EOT12.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 *
0367 SCA2 LDA IDSC INTERRUPTING DEVICE SELECT CODE TO A
0368 CPA SSC IS INTERRUPTING SELECT CODE THE STATUS CARD?
0369 RSS SKIP IF YES?
0370 JMP SPUR2 JUMP TO SPURIOUS INTERRUPT RETURN IF NOT
0371 LDA =D3 SET STATE CONTROL VAR.
0372 STA EOT12,1 ADDRESS OF BUFFER TO B
0373 LDB EOT7,1 INCREMENT B (BUFFER ADDRESS)
0374 INB CONTENTS OF BUFFER+1 TO A
0375 LDA B,1 SIGNIFYS Y-POSITION
0376 AND =B0777 TRANSFER CODE OF 3
0377 IOR BIT14
0378 STA B
0379 LDA =B3
0380 IOR GOBIT
0381 JMP CI01
0382 *
0383 *** TRACK BALL 2
0384 *
0385 SCA3 LDA IDSC INTERRUPTING DEVICE SELECT CODE TO A
0386 CPA SSC IS INTERRUPTING SELECT CODE THE STATUS CARD?
0387 RSS SKIP IF YES?
0388 JMP SPUR2 JUMP TO SPURIOUS INTERRUPT IF NOT SET
0389 LDA =D4 SET STATE CONTROL VARIABLE
0390 STA EOT12,1
0391 JMP TBDUN
0392 *
0393 *** TRACK BALL 3
0394 *
0395 SCA5 LDA IDSC INTERRUPTING DEVICE SELECT CODE TO A
0396 CPA SSC IS INTERRUPTING SELECT CODE THE STATUS CARD?
0397 RSS SKIP IF YES?
0398 JMP SPUR2 JUMP TO SPURIOUS INTERRUPT IF NOT
0399 LDA =D6 SET STATE CONTROL VARIABLE
0400 STA EOT12,1
0401 JMP CI02 JUMP TO START A READ
0402 *
0403 *** TRACK BALL 4
0404 *
0405 SCA6 LDA IDSC INTERRUPTING DEVICE SELECT CODE TO A
0406 CPA DSC IS INTERRUPTING SELECT CODE THE DSCA CARD?
0407 RSS SKIP IF YES?
0408 JMP SPUR2 JUMP TO SPURIOUS INTERRUPT IF NOT
0409 LDA =D7 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 EOT7.1 ADDRESS OF BUFFER TO B
0415 STA B.1 STORE X-POSITION IN BUFFER
0416 JMP C102

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 EOT12.1 SET STATE CONTROL VARIABLE
0426 GET2 LIA DAT DATA FROM DATA LINES TO A
0427 LDB EOT7.1 GET BUFFER ADDRESS
0428 INB POINT TO WORD TWO
0429 STA B.1 STORE Y-POSITION IN BUFFER+1
0430 JMP TBDUN

0431 *
0432 TBDUN LDA =B3
0433 IOR CLBIT THIS ADDED PÆR RÆB 6/15/81
0434 IOR ENBIT
0435 JMP C103

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 EOT6.1 GET CONTROL WORD
0445 AND *B3 ISOLATE FUNCTION CODE
0446 CPA =B1 IS IT A READ?
0447 LDB =B4 SET TRANS. LOG TO 4 FOR READ
0448 JMP CEND1

0449 *
0450 *
0451 *
0452 C101 NOP
0453 OTB DAT
0454 C103 QTA.CST
0455 STC CST.C
0456 ISZ C.41
0457 JMP C.41.I
0458 *

209
* B45D
* 8460
* B4Gl CI0
* 2 NOP
* 44 CLC DAT,C
* 44 STC DAT,C
* 44 ISZ C.41
* 44 JMP C.41,1
* 44 *
* 4477 ** DMA TRANSFER 1
* 4478 *
* 4479 SCA8 LDA IDSC INTERUPTING DEVICE SELECT CODE TO A
* 4470 CPA SSC IS INTERRUPTING SELECT CODE THE STATUS CARD?
* 4471 RSS SKIP IF YES?
* 4472 JMP SPUR2 JUMP TO SPURIOUS INTERRUPT IF NOT
* 4473 LDA =D9
* 4474 STA EOT12.I SET STATE CONTROL VARIABLE
* 4475 *
* 4476 *
* 4477 *
* 4478 ******** THIS ROUTINE MODIFIES THE DCPC INITIALIZATION INSTRUCTIONS
* 4479 * TO SPECIFY THE SELECT CODES OF THE ASSIGNED DCPC
* 4480 *
* 4481 STDMA LDB EOT11.I GET LOW SELECT CODE FROM EOT
* 4482 LDA OTA "OTA INSTRUCTION" TO A
* 4483 IOR B OR IN THE LOWER SELECT CODE
* 4484 STA D3 STORE AT LABEL D3
* 4485 STA D5 STORE AT LABEL D5
* 4486 ADA =B4 ADD 4 TO INSTR. TO CHANGE TO HIGHER SELECT CODE
* 4487 STA D1
* 4488 *
* 4489 LDA CLC "CLC INSTR." TO A
* 4490 IOR B OR IN LOWER SELECT CODE
* 4491 STA D2
* 4492 LDA STC
* 4493 IOR B
* 4494 STA D4
* 4495 ADA =B4
* 4496 STA D6
* 4497 STA D7
* 4498 LDA CLC
* 4499 IOR DSC
* 4500 STA D6+1
* 4501 LDA STF
* 4502 IOR DSC
* 4503 STA D6+2
* 4504 LDA STC

210
**THIS ROUTINE INITIATES THE DCPC DATA TRANSFERS**

**THE INSTRUCTIONS BELOW WITH D LABELS ARE MODIFIED BY THE ROUTINE SETIO TO SPECIFY THE CORRECT SELECT CODES FOR THE DCPC CHANNEL ASSIGNED BY THE SYSTEM**

- **LDA EQT4.I** - MASK FOR COMMAND SELECT CODE
- **AND =B77** - INCREMENT FOR DATA SELECT CODE
- **IOR BIT15** - TURN ON HANDSHAKE BIT (STC)
- **OTA 6B** - PUT CONTROL WORD 1 TO DCPC CHANNEL (SELECT 6 OR 7)
- **CLC 2B** - PREPARE MEM. ADDR. REGISTER FOR CW2 (SELECT CODE 2 OR 3)
- **LDA EQT6.I** - GET CONTROL WORD
- **AND =B3** - MASK FOR FUNCTION
- **STA B** - STORE FUNCTION IN B
- **LDA EQT7.I** - ADDRESS OF BUFFER TO A
- **CPB =B1** - IS FUNCTION A READ?
- **IOR BIT15** - TURN ON "IN" BIT FOR A READ, IF YES
- **OTA 2B** - CONTROL WORD 2 TO DCPC CHANNEL (SC 2 OR 3)
- **STC 2B** - PREPARE MEM. ADDR. REGISTER FOR CW3 (SC 2 OR 3)
- **LDA EQT8.I** - GET LENGTH OF TRANSFER
- **CMA,INA** - MAKE TWO'S COMPLEMENT
- **OTA 2B** - CONTROL WORD 3 TO DCPC CHANNEL (SELECT CODE 2 OR 3)
- **CPB =B1** - IS IT A READ?
- **JMP D7** - JUMP IF YES
- **STC 6B** - ACTIVATE DCPC CHANNEL (SELECT CODE 6 OR 7)
- **CLC DAT** - CLEAR DEVICE
- **STF DAT** - ACTIVATE DEVICE?
- **JMP GOEND**
- **D7** - ACTIVATE DCPC CHANNEL (SELECT CODE 6 OR 7)
- **STC DAT.C** - ACTIVATE DEVICE
- **GOEND ISZ C.41** - NOW RETURN TO CIC WITH
- **JMP C.41.I** - CONTINUATION
- **SCA9 LDA EQT11.I** - GET LOW SELECT CODE OF DCPC CHANNEL
- **ADA =B4** - ADD 4 TO CONVERT LOW DCPC SELECT CODE TO HIGH S.C.
- **CHKDF CPA IDSC** - IS INTERRUPTING SELECT CODE THAT OF THE ASSIGNED DCPC CHANNEL?
- **RSS** - SKIP IF YES
- **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 SCA10 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 =B177408  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 EGT4.1
0601 AND #B77
0602 STA SSC
0603 INA
0604 STA DSC
0605 LDA OTA
0606 IOR SSC
0607 STA CI03
0608 LDA STC
0609 IOR SSC
0610 STA CI03+1
0611 INA
0612 *
0613 STA CI02+2
0614 LDA CLC
0615 IOR SSC
0616 STA CEND2+1
0617 INA
0618 STA CEND2
0619 STA CI02+1
0620 LDA LIA
0621 IOR SSC
0622 STA CEND1
0623 INA
0624 STA GET1
0625 STA GET2
0626 LDA OTB
0627 IOR DSC
0628 STA CI01+1
0629 JMP CSIO.1 RETURN
0630 *
0631 *
0632 *
0633 *
0634 *
0635 *
0636 ********* CONSTANTS AND STORAGE AREA **********
0637 *
0638 RFLAG BSS .1
0639 FUNC BSS 1
0640 TRAN BSS 1
0641 SAVE1 BSS 1
0642 SAVE2 BSS 1
0643  CW1  BSS 1
0644  CW2  BSS 1
0645  CW3  BSS 1
0646  TMSK  BSS 1
0647  IDSC  BSS 1
0648  SSC  BSS 1
0649  DSC  BSS 1
0650  BIT13  OCT 20000
0651  BIT14  OCT 40000
0652  BIT15  OCT 100000
0653  RMDTE  OCT 2
0654  GOBIT  OCT 4
0655  ENBIT  OCT 10
0656  NBSY  OCT 10
0657  CLBIT  OCT 20
0658  TOUT  DEC -30000
0659  ERNUM  DEC 3
0660  ***
0661  ***  THE  FOLLOWING  VALUES  (SUFFIX  V)  DETERMINE  WHICH  DRIVER
0662  ***  SUBFUNCTIONS  ARE  INVOKED,  PARAMETER  IFUNC  IN  SAMPLE  CALL  ABOVE.
0663  ***
0664  DSPLV  OCT  0
0665  ILTRV  OCT  100
0666  FMTRV  OCT  200
0667  PCMTV  OCT  300
0668  TBALV  OCT  400
0669  SUPTV  OCT  500
0670  LINKV  OCT  600
0671  ***
0672  CST  EQU  24  USED  ONLY  FOR  DOCUMENTATION.  ACTUAL  INSTRUCTION  IS
0673  DAT  EQU  25  SET  UP  BY  A  SETIO  ROUTINE.
0674  A  EQU  0
0675  D  EQU  1
0676  OTA  OTA  0
0677  CLC  CLC  0,C
0678  STC  STC  0,C
0679  LIA  LIA  0
0680  OTB  OTB  0
0681  SFS  SFS  0
0682  STF  STF  0
0683  ******BASE  PAGE  COMMUNICATIONS  AREA  DEFINITIONS
0684  .  EQU  1650B
0685  *
0686  ***
0687  ***  THE  FOLLOWING  CONSTANTS  ARE  USED  TO  READ  THE  EQUIPMENT  TABLE
0688  ***  ASSOCIATED  WITH  THIS  DRIVER.  SEE  THE  HP-1000  QUICK  REFERENCE
** MANUAL FOR DIAGRAMS AND TABLES ON WHICH THESE CONSTANTS ARE BASED.**

0690 **

0691 INTBA EQU .+4
0692 EQT1 EQU .+8
0693 EQT2 EQU .+9
0694 EQT3 EQU .+10
0695 EQT4 EQU .+11
0696 EQT5 EQU .+12
0697 EQT6 EQU .+13
0698 EQT7 EQU .+14
0699 EQT8 EQU .+15
0700 EQT9 EQU .+16
0701 EOT10 EQU .+17
0702 EOT11 EQU .+18
0703 EOT12 EQU .+81
0704 EOT13 EQU .+82
0705 EOT14 EQU .+83
0706 EOT15 EQU .+84
0707 *
0708 *
0709 *

0710 ******** STATE CONTROL ADDRESS TABLE ********

0711 CTABA DEF CTAB
0712 CTAB NOP

0713 JMP SCA1 DISPLAY COMMAND COMPLETION
0714 JMP SCA2 TRACK BALL 1
0715 JMP SCA3 TRACK BALL 2
0716 JMP SCA4 TRACK BALL TRANSMIT END
0717 JMP SCA5 TRACK BALL 3
0718 JMP SCA6 TRACK BALL 4
0719 JMP SCA7 TRACK BALL 5
0720 JMP SCA8 DMA TRANSFER 1
0721 JMP SCA9 DMA TRANSFER 2
0722 JMP SCA10 DMA TRANSFER 3
0723 * BSS 10
0724 END
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.