```
      REM BBC BASIC FOR WINDOWS (BB4W) program to extract line and column signal
      REM levels from a Windows RGB BMP file, into a text file.

      REM (c) Alan Roberts 2010

      SYS "SetWindowText", @hwnd%, "(7) Line and column signal values extraction."

      REM routine to get the input BMP file name for processing.

      in%=0           : REM this is going to be the input file handle
      infile$=""      : REM and this will be the file name
      hout% = 0       : REM and this is the line scan text output file handle
      hfile$ = ""     : REM and it's name,
      vout% = 0       : REM column scan file handle
      vfile$ = ""     : REM and file name

      DIM of% 75,ff% 255,fn% 255 : REM byte arrays needed for Windows OpenFile routine
      !of%=76    : of%!4=@hwnd%  : of%!12=ff% : of%!28=fn%
      of%!32=256 : of%!52=6      : REM BB4W stuff for Windows GetOpenFile routine
      $fn% = CHR$(0)             : REM this is going to be the file name
      $ff% = "YUV 422 image file (*.bmp)" + CHR$0 + "*.bmp" + CHR$0 + CHR$0
      SYS "GetOpenFileName", of% TO in%
      IF in% THEN
        infile$ = FNnulterm$(fn%)
        hfile$ = infile$ + ".h.txt"
        vfile$ = infile$ + ".v.txt"
        PRINTTAB(0,3) " Input BMP file    = " infile$
        PRINTTAB(0,4) " Output text files = " hfile$
        PRINTTAB(0,5) "               and = " vfile$
      ELSE
        PRINTTAB(0,3) " Programme aborted at GetOpen, press any key to exit."
        IF GET QUIT                : REM we're not doing any more, so close the window
      ENDIF

      PRINTTAB(0,7) " The bitmap file will now be displayed, scaled down if it's too big to fit the screen."
      PRINT'" A measurement cross will be superimposed on it. You can move the cross around (use the"
      PRINT " cursor keys, with Shift and Control to set the step size). The signal values across the line"
      PRINT " and down the column will be exported to text files."
      PRINT'" When you are happy with the cross size and position, press Enter to start the measurement process."
      PRINT'" Press any key to clear this screen and load the bitmap file."

      IF GET

      REM Now we can get on with it ...

      in% = OPENUP infile$                  : REM open RGB bitmap file for reading

      IF CHR$(BGET# in%) + CHR$(BGET# in%) <> "BM" THEN
        PRINTTAB(0,17) " This isn't a Windows bitmap file, press any key to exit."
        IF GET QUIT                         : REM we're not doing any more, so close the window
      ENDIF

      PTR# in% = 10 : start% = FNget4(in%)  : REM size of the header block, where image data starts
      PTR# in% = 18 : wide%  = FNget4(in%)  : REM image width in pixels
      PTR# in% = 22 : high%  = FNget4(in%)  : REM image height in lines
      CLOSE# in%                            : REM must close the file in order to load it for display
      aspect = wide% / high%                : REM image aspect ratio

      scale=1 : REM scale factor for loaded bitmap file
      r% = 1  : REM flag for "OK"
      SYS "GetSystemMetrics", 0 TO wscreen% : REM get the screen width for the actual computer display
      SYS "GetSystemMetrics", 1 TO hscreen% : REM and height

      IF wide%>wscreen% OR high%>hscreen%-65 THEN
        SYS "MessageBox", @hwnd%, "File too big for the display, scale and load it anyway (colours may not be wrong but analysis will be correct) ?',
"Load BMP File", 32+1 TO r%
        scale = FNmax(wide% / wscreen%, high% / (hscreen% - 65))
        IF r%=1 THEN
          SYS "SetWindowText", @hwnd%, FNname(infile$) + "scaled"
        ELSE
          PRINT " Process aborted at file loading stage. Press any key to exit."
          IF GET QUIT                       : REM we're not doing any more, so close the window
        ENDIF
      ENDIF

      REM set a screen mode to accommodate the inage file, this is Windows stuff

      DIM rc% 15                                        : REM data block for screen window size
      VDU 23, 22, high% / scale * aspect; high% / scale; 8, 16, 16, 0 : REM don't ask, just don't ask :-)
      SYS "PatBlt", @memhdc%, 0, 0, 1600, 1200, &FF0062
      SYS "GetSysColor", 5 TO f%                        : REM look up system colours
      COLOUR 15, f%, f%>>8, f%>>16                      : REM define colour 15 in RGB
      SYS "GetClientRect", @hwnd%, rc%                  : REM get the display screen size
      wwindow% = rc%!8 : hwindow% = rc%!12 + 2          : REM size of window after status bar added
      COLOUR 128 + 15 : CLS : REM set white as background colour and clear to it
      COLOUR 0                                          : REM black for printing
      SYS "GetWindowLong", @hwnd%, -16 TO f%            : REM get window dimensions
      SYS "SetWindowLong", @hwnd%, -16, f% OR &40000    : REM don't lock them
      SYS "GetClientRect", @hwnd%, rc%                  : REM get window size
      VDU 26, 28, 1, hwindow% / 16 - 2, wwindow% / 8 - 2, 1 : REM now set the actual display window for the image
      IF scale > 1 SYS "SetStretchBltMode", @memhdc%, 3

      REM now we  can load and display the file

      OSCLI "Display """ + infile$ + """ 0,0," + STR$(INT(high% * 2 / scale * aspect)) + "," + STR$(INT(high% * 2 / scale))

      REM next, define the measurement area

      xc% = wide% / 2 / scale : yc% = high% / 2 / scale           : REM Measurement cross position,
      GCOL 3, 7                                                   : REM Set graphic colour to invert what's there.
      LINE 2 * xc%, 0, 2 * xc%, 2 * high%/scale                   : REM draw vertical line
      LINE 0, 2 * yc%, 2 * wide%/scale, 2 * yc%                   : REM and the horizontal line, BB4W uses scaled graphics
      REPEAT
        WAIT 5                                                    : REM relax for a bit (1/20 second, not critical)
        h% = 0                                                    : REM horizontal movement increment
        v% = 0                                                    : REM vertical movement increment
        s% = 0                                                    : REM size increment
        IF INKEY(-26)  : h% = -2 : IF INKEY(-1) : h% = -10        : REM cursor left : and shifted
        IF INKEY(-122) : h% = 2  : IF INKEY(-1) : h% = 10         : REM cursor right : and shifted
        IF INKEY(-58)  : v% = 2  : IF INKEY(-1) : v% = -2         : REM cursor up : and shifted
        IF INKEY(-42)  : v% = -2 : IF INKEY(-1) : v% = -10        : REM cursor down : and shifted
        IF INKEY(-64)  : v% = 40                                  : REM page up
        IF INKEY(-79)  : v% = -40                                 : REM page down
        IF INKEY(-2) THEN
          IF INKEY(-26)  : h% = -40                               : REM cursor left and ctrl
          IF INKEY(-122) : h% = 40                                : REM cursor right and ctrl
          IF INKEY(-58)  : v% = 40                                : REM cursor up and ctrl
          IF INKEY(-42)  : v% = -40                               : REM cursor down and ctrl
        ENDIF
        LINE 2 * xc%, 0, 2 * xc%, 2 * high%/scale                 : REM delete the lines
        LINE 0, 2 * yc%, 2 * wide%/scale, 2 * yc%
        xc% += h% : yc% += v%                                     : REM move the position
        LINE 2 * xc%, 0, 2 * xc%, 2 * high%/scale                 : REM redraw the lines
        LINE 0, 2 * yc%, 2 * wide%/scale, 2 * yc%
      UNTIL INKEY(-74)                                            : REM until the Enter key is pressed
      REPEAT UNTIL INKEY(0)=-1                                    : REM This empties the keyboard buffer, just to be safe

      REM now we can do the actual measurements.
```

```
DIM Rh%(255), Gh%(255), Bh%(255)                              : REM generate arrays to hold histograms
DIM Rv%(255), Gv%(255), Bv%(255)

in% = OPENUP infile$     : REM re-open the RGB bitmap file for measurement

hout% = OPENOUT hfile$   : REM create the horizontal scan text file for writing

PRINT# hout%, "Analysis of file "+infile$                     : REM identify the file we're anaylsing
BPUT# hout%, 13 : BPUT# hout%, 10                             : REM send carriage return/line feed
BPUT# hout%, 13 : BPUT# hout%, 10                             : REM empty line
PRINT# hout%, "Line scan values for line "+STR$(high% - yc% + 1)         : REM announce what the file contains
BPUT# hout%, 13 : BPUT# hout%, 10
PRINT# hout%, "Pixel position, Rd, Gd, Bd, Ra, Ga, Ba"
BPUT# hout%, 13 : BPUT# hout%, 10
BPUT# hout%, 13 : BPUT# hout%, 10                             : REM empty line
str$ = ""                                                              : REM this will be the string printed to the text file
xc% *= scale : yc% *= scale                                            : REM allow for screen scaling
PTR# in% = FNptr(0, yc%, wide%, high%, 3) + start%
FOR x% = 1 TO wide%
  B = BGET# in% : G = BGET# in% : R = BGET# in%                        : REM get pixel values
  Bh%(B) += 1 : Gh%(G) += 1: Rh%(R) +=1                                : REM accumulate for the histogram
  @% = &A                                                              : REM default print format
  str$ = STR$(x%) + ", "                                               : REM the pixel position
  str$ += STR$(R) + ", " + STR$(G) + ", " + STR$(B) + ", "             : REM add the digital values
  @% = &1030A                                                          : REM 3 decimal places
  str$ += STR$(FNdac(R)) + ", " + STR$(FNdac(G)) + ", " + STR$(FNdac(B)) : REM and the analogue values
  PRINT# hout%, str$
  BPUT# hout%, 13 : BPUT# hout%, 10
NEXT
BPUT# hout%, 13 : BPUT# hout%, 10                             : REM empty line
BPUT# hout%, 13 : BPUT# hout%, 10                             : REM empty line
PRINT# hout%, "Histograms of data values"                     : REM announce what the file contains
BPUT# hout%, 13 : BPUT# hout%, 10
PRINT# hout%, "Rd, Gd, Bd"
BPUT# hout%, 13 : BPUT# hout%, 10
BPUT# hout%, 13 : BPUT# hout%, 10                             : REM empty line
FOR x% = 0 TO 255
  @% = &A                                                              : REM default print format
  str$ = STR$(x%) + ", "                                               : REM the pixel position
  str$ += STR$(Rh%(x%)) + ", " + STR$(Gh%(x%)) + ", " + STR$(Bh%(x%))  : REM this is the histogram
  PRINT# hout%, str$
  BPUT# hout%, 13 : BPUT# hout%, 10
NEXT
CLOSE # hout%                                                 : REM done with this file, close it

vout% = OPENOUT vfile$   : REM create the vertical scan text file for writing

PRINT# vout%, "Analysis of file "+infile$                     : REM identify the file we're anaylsing
BPUT# vout%, 13 : BPUT# vout%, 10                             : REM send carriage return/line feed
BPUT# vout%, 13 : BPUT# vout%, 10                             : REM empty line
PRINT# vout%, "Column scan values for pixel "+STR$(xc%)       : REM announce what the file contains
BPUT# vout%, 13 : BPUT# vout%, 10
PRINT# vout%, "Pixel position, Rd, Gd, Bd, Ra, Ga, Ba"
BPUT# vout%, 13 : BPUT# vout%, 10
BPUT# vout%, 13 : BPUT# vout%, 10                             : REM empty line
str$ = ""                                                              : REM this will be the string printed to the text file
FOR y% = 1 TO high%
  PTR# in% = FNptr(xc%, y%, wide%, high%, 3) + start%
  B = BGET# in% : G = BGET# in% : R = BGET# in%                        : REM get pixel values
  Bv%(B) += 1 : Gv%(G) += 1: Rv%(R) +=1                                : REM accumulate for the histogram
  @% = &A                                                              : REM default print format
  str$ = STR$(x%) + ", "                                               : REM the pixel position
  str$ += STR$(R) + ", " + STR$(G) + ", " + STR$(B) + ", "             : REM add the digital values
  @% = &1030A                                                          : REM 3 decimal places
  str$ += STR$(FNdac(R)) + ", " + STR$(FNdac(G)) + ", " + STR$(FNdac(B)) : REM and the analogue values
  PRINT# vout%, str$
  BPUT# vout%, 13 : BPUT# vout%, 10
NEXT
BPUT# vout%, 13 : BPUT# vout%, 10                             : REM empty line
BPUT# vout%, 13 : BPUT# vout%, 10                             : REM empty line
PRINT# vout%, "Histograms of data values"                     : REM announce what the file contains
BPUT# vout%, 13 : BPUT# vout%, 10
PRINT# vout%, "Rd, Gd, Bd"
BPUT# vout%, 13 : BPUT# vout%, 10
BPUT# vout%, 13 : BPUT# vout%, 10                             : REM empty line
FOR y% = 0 TO 255
  @% = &A                                                              : REM default print format
  str$ = STR$(y%) + ", "                                               : REM the pixel position
  str$ += STR$(Rv%(y%)) + ", " + STR$(Gv%(y%)) + ", " + STR$(Bv%(y%))  : REM this is the histogram
  PRINT# vout%, str$
  BPUT# vout%, 13 : BPUT# vout%, 10
NEXT
CLOSE # vout%                                                 : REM done with this file, close it
CLOSE # in%                                                   : REM close the source file

PRINT'" Processing completed. Press any key to exit. "

@%=&A                                 : REM default print format

IF GET

QUIT   : REM all done, close the window


REM These are standard routines

DEF FNnulterm$(A%) : REM return BB4W string from Windows string (terminated by null)
LOCAL s$
WHILE ?A% <> 0
  s$ += CHR$(?A%) : A% += 1 : REM strip off characters until the first null
ENDWHILE
= s$

DEF FNinput(A%) : REM read a line of text from the file, throw away non-printing characters
LOCAL l$
INPUT# A%, l$
IF ASC(l$) <= 32 : l$ = MID$(l$,2)
IF ASC(RIGHT$(l$, 1)) <=32 : l$ = LEFT$(l$, LEN(l$) -1)
=l$

DEF FNname(A$) : REM drop path from filename
LOCAL n$, p%
p% = LEN(A$)
WHILE MID$(A$, p%, 1) <> "\" AND p% > 0
  n$ = MID$(A$, p%, 1) + n$
  p% -= 1
ENDWHILE
= n$

DEF FNget4(A%) : REM get 4 byte number from file
= FNget2(A%) + 256 * 256 * FNget2(A%)

DEF FNget2(A%) : REM get a 2 byte number from file
= (BGET# A%) + 256 * (BGET# A%)

DEF FNptr(A%,B%,C%,D%,E%) : REM point to pixel at a%,b%, image c%xd%, e% planes
= (D% - B%) * ((E% * C% + E%) DIV 4 * 4) + E% * (A% - 1)
```

```
DEF FNdac(A) : REM undo coder YRGB digitising, return analogue
= (A - 16) / 219

DEF FNmax(A, B) : REM return lerger of A and B
IF A > B : = A
= B
```