```
      REM BBC BASIC FOR WINDOWS (BB4W) program to measure signal levels and
      REM noise levels in a Windows RGB BMP file.

      REM (c) Alan Roberts 2010

      SYS "SetWindowText", @hwnd%, "(4) Measurement of signal levels and noise levels, full screen."

      REM Start with the coding equations.

      eqn% = OPENIN "Coding equations.txt"

      IF eqn% = 0 THEN
        PRINT " Can't find coding equations file (Coding equations.txt). Press any key to exit."
        IF GET QUIT                         : REM we're not doing any more, so close the window
      ENDIF

      line$ = FNinput(eqn%)                 : REM read the first line from the file
      IF line$ <> "Coding equations" THEN
        PRINT " File 'Coding equations.txt' is not correct. Press any key to exit."
        IF GET QUIT                         : REM we're not doing any more, so close the window
      ENDIF

      REPEAT                                : REM scan the file, ignoring comments (lines starting with //), looking for the equations
        line$ = FNinput(eqn%)
      UNTIL INSTR(line$, "Coder-") = 1 OR EOF# eqn%
      IF EOF# eqn% THEN
        PRINT " File error, no equations defined. Press any key to exit."
        IF GET QUIT                         : REM we're not doing any more, so close the window
      ENDIF
      PRINTTAB(0,1) "Luma coder : ";line$   : REM this is the filter title
      Yr = VAL(FNinput(eqn%))               : REM coding equation coefficients
      Yg = VAL(FNinput(eqn%))
      Yb = VAL(FNinput(eqn%))
      CLOSE# eqn%                           : REM done with the equations file

      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

      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$(fn%)
        PRINTTAB(0,3) " Input BMP file = " infile$
      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(0,6) " The bitmap file will now be displayed, scaled down if it's too big to fit the screen."
      PRINT'" A measurement box will be superimposed on it. You can move the box around (use the"
      PRINT " cursor keys, with Shift and Control to set the step size) and change it's size (use < and >"
      PRINT " or - and +)."
      PRINT'" When you are happy with the box 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(0,10) " 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 be wrong, but analysis will be correct)?", "Lo
ad BMP File", 32+1 TO r%
        scale = FNmax(wide% / wscreen%, high% / (hscreen% - 65))
        IF r% > 1 THEN
          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

      SYS "SetWindowText", @hwnd%, "(4) Analyse file - " + FNname(infile$)

      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

      PRINTwwindow%,hwindow%:IFGET

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

      REM next, define the measurement area

      xl% = 40 : xr% = wwindow% - 40                            : REM Measurement box horizontal limits,
      yt% = 40 : yb% = hwindow% - 40                            : REM and vertical limits.
      GCOL 3, 7                                                 : REM Set graphic colour to invert what's there.
      RECTANGLE 2 * xl%, 2 * yt%, 2 * (xr% - xl%), 2 * (yb% - yt%) : REM Draw the measurement box, BB4W uses scaled graphics
      REPEAT
        WAIT 5                                                  : REM relax for a bit (1/20 second, not critical)
        h% = 0                                                  : REM horizontal mnovement 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(-103) : s% = -2 : IF INKEY(-1) : s% = -10        : REM < or , key : and shifted
  IF INKEY(-104) : s% = 2  : IF INKEY(-1) : s% = 10         : REM > or . key : and shifted
  IF INKEY(-24)  : s% = -2 : IF INKEY(-1) : s% = -10        : REM _ or - key, : and shifted
  IF INKEY(-94)  : s% = 2  : IF INKEY(-1) : s% = 10         : REM + or = key : and shifted
  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
  RECTANGLE 2 * xl%, 2 * yt%, 2 * (xr% - xl%), 2 * (yb% - yt%)      : REM delete the box
  xl% += h% - s% : xr% += h% + s% : yt% += v% - s% : yb% += v% + s% : REM move the edges
  RECTANGLE 2 * xl%, 2 * yt%, 2 * (xr% - xl%), 2 * (yb% - yt%)      : REM redraw the moved box
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.

left = 0 : right = 0                  : REM These will be the mean luma values at the left and right edges of the box,
top = 0  : bottom = 0                 : REM and these across the top and bottom edges,
middle = 0                            : REM and at the middle, all to see if there's any shading in the file.
Rm = 0 : Gm = 0 : Bm = 0 : Ym = 0     : REM These will be the mean signal levels
Rn = 0 : Gn = 0 : Bn = 0 : Yn = 0     : REM and these the noise levels

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

@%=&2030A                             : REM fixed 3 decimal places, 10 digit columns

xl% *= scale : xr% *= scale : yt% *= scale : yb% *= scale     : REM allow for screen scaling

FOR y% = yt% TO yb%
  PTR# in% = FNptr(xl%, y%, wide%, high%, 3) + start%
  left += FNdac(BGET# in%)
  PTR# in% = FNptr(xr%, y%, wide%, high%, 3) + start%
  right += FNdac(BGET# in%)
NEXT
PRINT'" Mean level, left edge   = " ;left / (yb% - yt% + 1) " "
PRINT " Mean level, right edge  = " ;right / (yb% - yt% + 1) " "

PTR# in% = FNptr(xl%, yt%, wide%, high%, 3) + start%
FOR x% = xl% TO xr%
  top += FNdac(BGET# in%)
NEXT
PRINT " Mean level, top edge    = " ;top / (xr% - xl% + 1) " "

PTR# in% = FNptr(xl%, yb%, wide%, high%, 3) + start%
FOR x% = xl% TO xr%
  bottom += FNdac(BGET# in%)
NEXT
PRINT " Mean level, bottom edge = " ;bottom / (xr% - xl% + 1) " "

FOR y% = high% / 2 - 20 TO high%  / 2 + 20
  PTR# in% = FNptr(wide% / 2 -20, y%, wide%, high%, 3) + start%
  FOR x% = wide% / 2 -20 TO wide% / 2 + 20
    middle += FNdac(BGET# in%)
  NEXT
NEXT
PRINT " Mean level, middle      = " ;middle / (41 * 41) " "

@%=&A                                 : REM default print format

PRINTTAB(0,7) " Measuring overall mean level "
PRINT " Line "
FOR y% = yt% TO yb%
  PTR# in% = FNptr(xl%, y%, wide%, high%, 3) + start%
  PRINTTAB(6,8) ;y% " "
  FOR x% = xl% TO xr%
    Bm += FNdac(BGET# in%) : Gm += FNdac(BGET# in%) : Rm += FNdac(BGET# in%) : REM accumulate pixel values
  NEXT
NEXT
Bm /= ((xr% - xl% + 1) * (yb% - yt% + 1))       : REM divide by the number of measured samples
Gm /= ((xr% - xl% + 1) * (yb% - yt% + 1))
Rm /= ((xr% - xl% + 1) * (yb% - yt% + 1))
Ym = Yr * Rm + Yg * Gm + Yb * Bm                : REM generate the luma value using the coding equation

@%=&2030A                             : REM fixed 3 decimal places, 10 digit columns

PRINTTAB(0,7) " Mean levels of " ;(xr% - xl% + 1) * (yb% - yt% + 1) " pixels "
PRINT " Mean level, Red plane   = " ;Rm " "
PRINT " Mean level, Green plane = " ;Gm " "
PRINT " Mean level, Blue plane  = " ;Bm " "
PRINT " Mean level, luma (Y)    = " ;Ym " "

@%=&A                                 : REM default print format

PRINT'" Measuring noise levels "
PRINT " Line "
FOR y% = yt% TO yb%
  PTR# in% = FNptr(xl%, y%, wide%, high%, 3) + start%
  PRINTTAB(6,13) ;y% " "
  FOR x% = xl% TO xr%
    Bn += (FNdac(BGET# in%) - Bm) ^ 2           : REM accummulate squares of differences from the mean
    Gn += (FNdac(BGET# in%) - Gm) ^ 2
    Rn += (FNdac(BGET# in%) - Rm) ^ 2
  NEXT
NEXT
CLOSE# in%                                      : REM finished with the bitmap file
Bn /= ((xr% - xl% + 1) * (yb% - yt% + 1))       : REM divide by the number of measured samples
Gn /= ((xr% - xl% + 1) * (yb% - yt% + 1))
Rn /= ((xr% - xl% + 1) * (yb% - yt% + 1))
Yn = Yr * Rn + Yg * Gn + Yb * Bn                : REM luma noise is the luma-weighted sum of the square values.
Bn = 20 * LOG(SQR(Bn)) : Gn = 20 * LOG(SQR(Gn)) : REM Noise levels in dB, take square root of accumulated value first
Rn = 20 * LOG(SQR(Rn)) : Yn = 20 * LOG(SQR(Yn)) : REM the log to get the dB value relative to white at level 1.

@%=&2020A                             : REM fixed 2 decimal places, 10 digit columns

PRINT' " PSNR, Red plane       = " ;Rn " dB "
PRINT " PSNR, Green plane     = " ;Gn " dB "
PRINT " PSNR, Blue plane      = " ;Bn " dB "
PRINT " PSNR, Luma (Y') plane = " ;Yn " dB "

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 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

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