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

      REM (c) Alan Roberts 2010

      SYS "SetWindowText", @hwnd%, "(6) Measure signal levels and noise on a Colorchecker chart."

      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

      colours% = OPENIN "Test colours.txt"

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

      line$ = FNinput(colours%)             : REM read the first line from the file
      IF line$ <> "Test colours data" THEN
        PRINT " File 'Test colours.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$, "Colours-") = 1 OR EOF# eqn%
      IF EOF# eqn% THEN
        PRINT " File error, no test colours defined. Press any key to exit."
        IF GET QUIT                   : REM we're not doing any more, so close the window
      ENDIF
      PRINTTAB(0,1) " Test : ";line$  : REM this is the filter title

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

      in%=0         : REM this is going to be the input file handle
      infile$=""    : REM and this will be the file name
      out% = 0      : REM and this is the handle for a text output file
      outfile$ = "" : REM, this it it's 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$(fn%)
        outfile$ = infile$ + ".txt"
        PRINTTAB(0,3) " Input  file = " infile$
        PRINTTAB(0,4) " Output file = " outfile$
      ELSE
        PRINTTAB(0,6) " Programme aborted at GetOpen, press any key to exit."
        IF GET QUIT                 : REM we're not doing any more, so close the window
      ENDIF

      REM Now we can get on with it ...

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

      IF CHR$(BGET# in%) + CHR$(BGET# in%) <> "BM" THEN
        PRINTTAB(0,18) " 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
      PTR# in% = 0                          : REM reset ready to start copying the header block
      CLOSE# in%                            : REM release the file so that I can load it to the screen.
      aspect = wide% / high%                : REM image aspect ratio

      PRINTTAB(0,6) " The bitmap file will now be displayed, scaled down if it's too big to fit the screen."
      PRINT'" A set of measurement boxes will be superimposed on it. You can move the whole array (use cursor"
      PRINT " keys, with Shift and Control to set the step size), change the individual box size (use < and >),"
      PRINT " and the spacing between the boxes (use - and +)."
      PRINT'" The array of boxes should each overlay one of the Colorchecker test patches; it defines the area"
      PRINT " you will be measuring."
      PRINT'" When you are happy with the box sizes and positions, press Enter to start the measurement process."
      PRINT'" Press any key to clear this screen and load the bitmap file."

      IF GET

      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
        scale = FNmax(wide% / wscreen%, high% / (hscreen% - 65))
        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%
        IF r%=1 THEN
          SYS "SetWindowText", @hwnd%, FNname(infile$) + "scaled to fit"
        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

DIM size% 7                                         : REM data block for finding text size on screen

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

d% = high% / 6 / scale                                  : REM set patch dimension
g% = d% / 5                                             : REM and gap between patches
xc% = wide% / 2 / scale : yc% = high% / 2 / scale       : REM starting centre of the pattern
GCOL 3, 7                                               : REM Set graphic colour to invert what's there.
FOR h% = 1 TO 6
  FOR v% = 1 TO 4
    xp% = xc% + (h% - 4) * d% + (h% - 3.5) * g%         : REM coordinates of top-left of each box
    yp% = yc% + (v% - 3) * d% + (v% - 2.5) * g%
    RECTANGLE 2 * xp%, 2 * yp%, 2 * d%, 2 * d%          : REM Draw the measurement box, BB4W uses scaled graphics
  NEXT
NEXT

REPEAT
  WAIT 5                                                : REM relax for a bit (1/20 second, not critical)
  xi% = 0                                               : REM horizontal movement increment
  yi% = 0                                               : REM vertical movement increment
  di% = 0                                               : REM box size increment
  gi% = 0                                               : REM gap size increment
  IF INKEY(-26)  : xi% = -2 : IF INKEY(-1) : xi% = -10  : REM cursor left : and shifted
  IF INKEY(-122) : xi% = 2  : IF INKEY(-1) : xi% = 10   : REM cursor right : and shifted
  IF INKEY(-58)  : yi% = 2  : IF INKEY(-1) : yi% = -2   : REM cursor up : and shifted
  IF INKEY(-42)  : yi% = -2 : IF INKEY(-1) : yi% = -10  : REM cursor down : and shifted
  IF INKEY(-64)  : yi% = 40                             : REM page up
  IF INKEY(-79)  : yi% = -40                            : REM page down
  IF INKEY(-103) : di% = -2 : IF INKEY(-1) : di% = -10  : REM < or , key : and shifted
  IF INKEY(-104) : di% = 2  : IF INKEY(-1) : di% = 10   : REM > or . key : and shifted
  IF INKEY(-24)  : gi% = -2 : IF INKEY(-1) : gi% = -10  : REM _ or - key, : and shifted
  IF INKEY(-94)  : gi% = 2  : IF INKEY(-1) : gi% = 10   : REM + or = key : and shifted
  IF INKEY(-2) THEN
    IF INKEY(-26)  : xi% = -40                          : REM cursor left and ctrl
    IF INKEY(-122) : xi% = 40                           : REM cursor right and ctrl
    IF INKEY(-58)  : yi% = 40                           : REM cursor up and ctrl
    IF INKEY(-42)  : yi% = -40                          : REM cursor down and ctrl
  ENDIF
  FOR h% = 1 TO 6
    FOR v% = 1 TO 4
      xp% = xc% + (h% - 4) * d% + (h% - 3.5) * g%       : REM delete the box array
      yp% = yc% + (v% - 3) * d% + (v% - 2.5) * g%
      RECTANGLE 2 * xp%, 2 * yp%, 2 * d%, 2 * d%
    NEXT
  NEXT
  xc% += xi% : yc% += yi% : d% += di% : g% += gi%       : REM new dimensions for the box array
  FOR h% = 1 TO 6
    FOR v% = 1 TO 4
      xp% = xc% + (h% - 4) * d% + (h% - 3.5) * g%       : REM redraw it in the new position
      yp% = yc% + (v% - 3) * d% + (v% - 2.5) * g%
      RECTANGLE 2 * xp%, 2 * yp%, 2 * d%, 2 * d%
    NEXT
  NEXT
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

in% = OPENUP infile$ :REM open the bitmap file for reading
out% = OPENOUT outfile$

PRINT "Measuring ..."
PRINT# out%, "Analysis of " + infile$                   : REM send a line to the output text file
BPUT# out%, 13 : BPUT# out%, 10                         : REM with a carriage return, line feed

FOR v% = 1 TO 4
  FOR h% = 1 TO 6
    line$ = FNinput(colours%)
    line$ = LEFT$(line$, INSTR(line$, ",") - 1)
    WHILE RIGHT$(line$,1) = " "
      line$ = LEFT$(line$, LEN(line$) - 1)
    ENDWHILE

    xp% = xc% + (h% - 4) * d% + (h% - 3.5) * g%                 : REM top left of each box
    yp% = yc% + (v% - 3) * d% + (v% - 2.5) * g%
    xm% = xp% * scale : ym% = yp% * scale : dm% = d% * scale

    SYS "GetTextExtentPoint32", @memhdc%, " " + line$ + " ",LEN(line$) + 2, size%

    PRINT " " line$ " "                                         : REM identify the colour patch

    BPUT# out%, 13 : BPUT# out%, 10                             : REM send a blank line to the text file
    PRINT# out%, line$ : BPUT# out%, 13 : BPUT# out%, 10        : REM write the colour name to the text file, plus a LF

    Rm = 0 : Gm = 0 : Bm = 0                                    : REM reset mean signal values
    FOR y% = 1 TO dm%
      PTR# in% = FNptr(xm%, ym% + y%, wide%, high%, 3) + start%
      FOR x% = 1 TO dm%
        Bm += FNdac(BGET# in%) : Gm += FNdac(BGET# in%) : Rm += FNdac(BGET# in%)
      NEXT
    NEXT
    Bm /= ((dm% + 1) * (dm% + 1)) : Gm /= ((dm% + 1) * (dm% + 1))
    Rm /= ((dm% + 1) * (dm% + 1))                               : REM these are the mean levels
    Ym = Yr * Rm + Yg * Gm + Yb * Bm                            : REM and this is the mean luma level

    Rn = 0 : Gn = 0 : Bn = 0                                    : REM reset noise values
    xp% = xc% + (h% - 4) * d% + (h% - 3.5) * g%                 : REM top left of each box
    yp% = yc% + (v% - 3) * d% + (v% - 2.5) * g%
    FOR y% = 1 TO dm%
      PTR# in% = FNptr(xm%, ym% + y%, wide%, high%, 3) + start%
      FOR x% = 1 TO dm%
        Bn += ((FNdac(BGET# in%) - Bm) ^ 2)                     : REM accumulate squared differences
        Gn += ((FNdac(BGET# in%) - Gm) ^ 2)
        Rn += ((FNdac(BGET# in%) - Rm) ^ 2)
      NEXT
    NEXT
    Bn /= ((dm% + 1) * (dm% + 1)) : Gn /= ((dm% + 1) * (dm% + 1))
    Rn /= ((dm% + 1) * (dm% + 1))                               : REM these are the squared noise levels
    Yn = Yr * Rn + Yg * Gn + Yb * Bn                            : REM and this is the equivalent squared luma noise level
    Bn = -20 * LOG(SQR(Bn)) : Gn = -20 * LOG(SQR(Gn))           : REM and these are the PSNR values in dB wrt unity
    Rn = -20 * LOG(SQR(Rn)) : Yn = -20 * LOG(SQR(Yn))

    @% = &A                                     : REM default print format
```

```
    PRINT# out%, " Results for blocks of " + STR$((d% + 1) ^ 2) + " pixels" : BPUT# out%, 13 : BPUT# out%, 10

    @% = &102030A                           : REM print 3 decimal places, 10 digit columns

    PRINT# out%, " R mean " + STR$(Rm) : BPUT# out%, 13 : BPUT# out%, 10
    PRINT# out%, " G mean " + STR$(Gm) : BPUT# out%, 13 : BPUT# out%, 10
    PRINT# out%, " B mean " + STR$(Bm) : BPUT# out%, 13 : BPUT# out%, 10

    @% = &102020A                           : REM print 2 decimal places, 10 digit columns

    PRINT# out%, " R PSNR " + STR$(Rn) + "dB" : BPUT# out%, 13 : BPUT# out%, 10
    PRINT# out%, " G PSNR " + STR$(Gn) + "dB" : BPUT# out%, 13 : BPUT# out%, 10
    PRINT# out%, " B PSNR " + STR$(Bn) + "dB" : BPUT# out%, 13 : BPUT# out%, 10
    PRINT# out%, " Y PSNR " + STR$(Yn) + "dB" : BPUT# out%, 13 : BPUT# out%, 10

  NEXT
NEXT
CLOSE# in%  : REM finished with input file
CLOSE# out% : REM finished with the output text file

@% = &A                                 : REM default print format

VDU 4                                   : REM more BB4W voodoo, print at text cursor
PRINTTAB(0,0) " Process completed. The results are in file :-"
PRINT " " outfile$ " "
PRINT'" Press any key to exit."
IF GET

QUIT    : REM all done, so 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 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)

DEFF Nget4(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 FNmax(A, B) : REM return the greater value
IF A > B : = A
= B

DEF FNmin(A, B) : REM return the greater value
IF A < B : = A
= B

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