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

      REM (c) Alan Roberts 2010

      SYS "SetWindowText", @hwnd%, "(8) Measure signal levels in a BMP file."

      REM Start with the coding eqtaions.

      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$=""    : REM and this will be the file name
      out% = 0      : REM and this is the handle for a text output file
      outfile$ = "" : 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$(fn%)
        PRINTTAB(0,3) " Input  file = " infile$
      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'" Move the cursor around with the mouse. The RGB values at the pointer position will be shown in"
      PRINT " the window title bar."
      PRINT'" Initially, you are measuring exactly one pixel. You can change the size of the measurement area"
      PRINT " using the cursor keys (up/down to change the height, left/right for the width). If either the"
      PRINT " width or height of the measurement area is greater than unity, then you will see a box marking"
      PRINT " it by inverting the image colour, it will move with the cursor."
      PRINT'" Press any key to clear this screen and load the bitmap file."
      PRINT'" Measurements will continue automatically until you press any mouse button to exit the program"

      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
        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))
        PRINT " Scale ";scale
        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))

      in% = OPENUP infile$                              : REM open the bitmap file for reading
      w% = 0 : h% = 0 : n% = (w% + 1) * (h% + 1)        : REM measurement patch size
      MOUSE xm%, ym%, b%                                : REM get pixel coordinates of the mouse cursor
      xp% = xm%/2 - w%/2: yp% = ym%/2 - h%/2
      GCOL 3, 7                                                 : REM invert the image colour
```

```
IF w% > 0 AND h% > 0 : RECTANGLE 2*xp%, 2*yp%, 2*w%, 2*h%      : REM mark the cursor
IF w% > 0 AND h% = 0 : LINE 2*xp%, 2*yp%, 2*xp% + 2*w%, 2*yp%
IF w% = 0 AND h% > 0 : LINE 2*xp%, 2*yp%, 2*xp%, 2*yp% + 2*h%

REPEAT
  IF xm% >= 0 AND xm% < wide%*2 AND ym% >= 0 AND ym% < high%*2 THEN
    R% = 0 : G% = 0 : B% = 0                            : REM reset variables
    N%=0
    FOR y% = yp% TO yp% + h%
      FOR x% = xp% TO xp% + w%                          : REM accumulate values from the patch
        PTR# in% = FNptr(FNmin(FNmax(x%, 1), wide%), high% - FNmin(FNmax(y%, 1), high%), wide%, high%, 3) + start%
        B% += BGET# in% : G% += BGET# in% : R% += BGET# in%
      NEXT
    NEXT
    R = FNdac(R% / n%) : G = FNdac(G% / n%) : B = FNdac(B% / n%)
    Y = Yr * R + Yg * G + Yb * B
    R% /= n% : G% /= n% : B% /= n%
    @% = &A
    str$ = "At " + STR$(x% + 1) + "," + STR$(y% + 1) + " ("+STR$(w% + 1) + "x" + STR$(h% + 1) + ") : "
    @% = &102000A
    str$ += "R' = " + STR$(R%)
    @% = &102030A
    str$ +=  " (" + STR$(R) + "),"
    @% = &102000A
    str$ += " G' = " + STR$(G%)
    @% = &102030A
    str$ +=  " (" + STR$(G) + "),"
    @% = &102000A
    str$ += " B' = " + STR$(B%)
    @% = &102030A
    str$ +=  " (" + STR$(B) + "), Y' = " + STR$(Y)
    SYS "SetWindowText", @hwnd%, str$
    WAIT 5
    IF w% > 0 AND h% > 0 : RECTANGLE 2*xp%, 2*yp%, 2*w%, 2*h% : REM delete the cursor marker
    IF w% > 0 AND h% = 0 : LINE 2*xp%, 2*yp%, 2*xp% + 2*w%, 2*yp%
    IF w% = 0 AND h% > 0 : LINE 2*xp%, 2*yp%, 2*xp%, 2*yp% + 2*h%
    MOUSE xm%, ym%, b%                                  : REM get pixel coordinates of the mouse cursor
    xp% = xm%/2 - w%/2: yp% = ym%/2 - h%/2
    IF INKEY(-122) : w% += 1                            : REM cursor right
    IF INKEY(-26)  : w% -= 1                            : REM cursor left
    IF INKEY(-58)  : h% += 1                            : REM cursor up
    IF INKEY(-42)  : h% -= 1                            : REM cursor down
    w% = FNmin(FNmax(w%, 0), 99) : h% = FNmin(FNmax(h%, 0), 99)
    IF w% > 0 AND h% > 0 : RECTANGLE 2*xp%, 2*yp%, 2*w%, 2*h% : REM re-mark the cursor
    IF w% > 0 AND h% = 0 : LINE 2*xp%, 2*yp%, 2*xp% + 2*w%, 2*yp%
    IF w% = 0 AND h% > 0 : LINE 2*xp%, 2*yp%, 2*xp%, 2*yp% + 2*h%
    n% = (w% * scale + 1) * (h% * scale + 1)            : REM number of pixels measured
    REPEAT
    UNTIL INKEY(0) = -1                                 : REM empty the keyboad buffer
  ENDIF
UNTIL b%

CLOSE# in%  : REM finished with input file

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)

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%)

DE FFNmax(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
```