```
REM BBC BASIC FOR WINDOWS (BB4W) program to generate a standard Windows bitmap file
REM (.BMP) file from a raw data file capture of serial digital video, 8-bit YUV

REM (c) Alan Roberts 2010

SYS "SetWindowText", @hwnd%, "(1) Generate BMP file from RAW YUV file"

raw%     = 0  : REM input file handle
rawfile$ = "" : REM input file name
bmp%     = 0  : REM output file handle
bmpfile$ = "" : REM output file name

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

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% = "RAW image file " + CHR$0 + "*.raw;*.yuv" + CHR$0 + "Any file (*.*)" + CHR$0 + "*.*" + CHR$0 + CHR$0
SYS "GetOpenFileName", of% TO raw%
IF raw% THEN
  rawfile$ = FNnulterm$(fn%)
  bmpfile$ = rawfile$+".bmp"
  PRINT'" RAW file = " rawfile$
  PRINT " BMP file = " bmpfile$
  PRINT
ELSE
  PRINT " Programme aborted at GetOpen, press any key to exit."
  IF GET QUIT : REM we're not doing any more, so close the window
ENDIF

raw%= OPENIN rawfile$
size% = EXT# raw%      : REM size of the raw file in bytes
PRINT'" The RAW file is assumed to be organised as 4:2:2 subsampled, with UYVY data groups, so the file"
PRINT " size should be double the image width times height, plus any offset to ignore a header block."
PRINT " Enter a zero to abort the process."
PRINT'" This software was written for and tested on video files captured using a DVC ClipRecorder XTreme."

REM Other file formats are possible, e.g. 4:4:4, and 4:2:2 with other data groupings, but I have
REM yet to come across them in practice. Also, it is feasible that FAW files could have some form
REM of header block, so so the image data would start after that, but, again, I have not yet seen
REM any such file.

PRINT
PRINT'" RAW file size = ";size%
REPEAT
  PRINTTAB(5,12) SPC 80
  PRINTTAB(5,13) SPC 80
  PRINTTAB(5,14) SPC 80
  INPUTTAB(5,12) "Enter RAW image width, pixels = " wide%
  INPUTTAB(5,13) "Enter RAW image height, lines = " high%
  INPUTTAB(5,14) "Enter offset to start of image data ";offset%
  IF wide% * high% * 2 + offset% <> size% THEN
    PRINTTAB(5,16) "Dimensions don't match file size (width * heigtht = ";wide% * high% * 2 + offset%" )"
  ENDIF
UNTIL wide% * high% * 2 +offset% = size% OR wide% * high% = 0
IF wide% * high% = 0 THEN
  PRINT " Program aborted at image size entry, press any key to exit."
  IF GET QUIT : REM we're not doing any more, so close the window
ENDIF
PRINTTAB(5,16) SPC 80

REM Now we can get on with it ...

REM Open the output file and insert the header block. Can't do it all in one pass ....

bmp% = OPENOUT bmpfile$                  : REM this generates the output file for writing to.
BPUT# bmp%, ASC("B") : BPUT# bmp%, ASC("M") : REM identify it as BMP
PROCput4(bmp%, 0)                        : REM file size, come back to put this in
PROCput2(bmp%, 0) : PROCput2(bmp%, 0)    : REM reserved, don't know what for
PROCput4(bmp%, 0)                        : REM offset to start of image data, come back to put this in
PROCput4(bmp%, 40)                       : REM size of header info block
PROCput4(bmp%, wide%)                    : REM image width
PROCput4(bmp%, high%)                    : REM image height
PROCput2(bmp%, 1)                        : REM number of planes
PROCput2(bmp%, 24)                       : REM bits per pixel, 3 bytes/pixel
PROCput4(bmp%, 0)                        : REM 0 means no compression
PROCput4(bmp%, 3*wide%*high%)            : REM image size, can also be zero
PROCput4(bmp%, 2834)                     : REM h pixels/m, defaults to 72dpi
PROCput4(bmp%, 2834)                     : REM v pixels/m, defaults to 72dpi
PROCput4(bmp%, 0)                        : REM colours in bitmap, calculated from biBitCount
PROCput4(bmp%, 0)                        : REM number of important colours, 0 means they all are
start%    = PTR# bmp%                    : REM image bytes start here

REM now go back and finish off the header stuff.

PTR# bmp% = 2  : PROCput4(bmp%, start% + 3 * wide% * high%) : REM store the file size
PTR# bmp% = 10 : PROCput4(bmp%, start%)                     : REM size of the header block, normally 54

REM now we can do the actual conversion.
REM The RAW file is assumed to be organised as 4:2:2, in UYVY groups

PTR# raw%=offset% : REM ignore any specified header block in the RAW file

x%=0 : REM horizontal pixel position in BMP file
FOR y% = 1 TO high%
  PTR# bmp% = (high% - y%) * ((3 * wide% + 3) DIV 4 * 4) +3 * (x% - 1) + start%
  FOR x% = 1 TO wide%
    IF x% MOD 2 = 1 THEN
      U% = BGET# raw% : Y% = BGET# raw% :REM UYVY
      BPUT# bmp%, 128 : BPUT# bmp%, Y% : BPUT# bmp%, 128
    ELSE :REM odd pixels above, even below
      V% = BGET# raw% : Y% = BGET# raw% :REM UYVY
      BPUT# bmp%, U%  : BPUT# bmp%, Y% : BPUT# bmp%, V%
    ENDIF
  NEXT
  PRINT TAB(0,18) " Processing line = ";y%
  IF INKEY(0)=27 EXIT FOR
NEXT
CLOSE# raw% : REM finished with the RAW file
CLOSE# bmp% : REM finished with the BMP file as well

PRINTTAB(0,20) " The output BMP file is organised such that Cb is in the B plane, Y in the G plane,"
PRINTTAB(0,21) " and Cr in the R plane. Alternate samples in the chroma planes are mid-grey. Run"
PRINTTAB(0,22) " program 2 to fill in the gaps in chroma and then decode it to RGB."

PRINTTAB(0,24) " Do you want to load the file for viewing now? (N = exit) "

IF INSTR("Nn", GET$) : QUIT : REM we're not loading it, so close the window

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)?", "Load BMP File", 32+1 TO r%
```

```
  scale = FNmax(wide% / wscreen%, high% / (hscreen% - 65))
  IF r%=1 THEN
    SYS "SetWindowText", @hwnd%, FNname(bmpfile$)
  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 """ + bmpfile$ + """ 0,0," + STR$(INT(high% * 2 / scale * aspect)) + "," + STR$(INT(high% * 2 / scale))

END  : REM program ends here, but the window stays open


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$(?A%) : A% += 1 : REM strip off characters to find the first non-null
ENDWHILE
=s$

DEF PROCput4(A%, B%) : REM put 4 byte number B% into file with handle A%
PROCput2(A%, B% MOD (256^2)) : PROCput2(A%, B% DIV (256^2))
ENDPROC

DEF PROCput2(A%, B%) : REM put a 2 byte number B% into file with handfle A%
BPUT# A%, B% MOD 256 : BPUT# A%, B% DIV 256
ENDPROC
```