```
REM BBC BASIC FOR WINDOWS (BB4W) program to convert a 4:2:2 subsampled YUV file to
REM 4:4:4 YUV by interpolating the chroma.

REM (c) Alan Roberts 2010

SYS "SetWindowText", @hwnd%, "(2) Convert bitmap file (4:2:2 YUV) to 4:4:4, interpolating the chroma channels"

REM Start with the resampling filter.

filter% = OPENIN "Chroma filter.txt"

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

line$ = FNinput(filter%)                        : REM read the first line from the file
IF line$ <> "Chroma filter" THEN
  PRINT " File 'Chroma filter.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 filter
  line$ = FNinput(filter%)
UNTIL INSTR(line$, "Filter-") = 1 OR EOF# filter%
IF EOF# filter% THEN
  PRINT " File error, no filter defined. Press any key to exit."
  IF GET QUIT                                   : REM we're not doing any more, so close the window
ENDIF
PRINTTAB(0,1) " Chroma resampling : ";line$     : REM this is the filter title
order% = VAL(FNinput(filter%))                  : REM this should be the filter order
PRINTTAB(5,3) " Filter order = ";order%
DIM term(order%/2+1)                            : REM create an array to hold the coefficients
FOR term%=0 TO order%/2 + 1
  term(term%) = VAL(FNinput(filter%))           : REM read a coefficient at a time
NEXT
CLOSE# filter%                                  : REM done with the filter 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$(fn%)
  PRINTTAB(0,5) " Input/Output YUV (4:2:2) file = " infile$
ELSE
  PRINTTAB(0,5) " 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(0,8) " The input file is assumed to be organised as 4:2:2 chroma subsampled, Cb in B plane,"
PRINTTAB(0,9) " Y in the G plane, Cr in the R plane. Alternate chroma samples are mid-grey."

REM Now we can get on with it ...

in% = OPENUP infile$ :REM open for read/write, to read pixels, and insert missing chroma samples by filtering

IF CHR$(BGET# in%) + CHR$(BGET# in%) <> "BM" THEN
  PRINTTAB(0,11) " This isn't a Windows bitmap file, press any key to exit."
  IF GET QUIT
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

REM now we can do the actual conversion.

u  = 0 : REM Cb value
v  = 0 : REM Cr value
ul = 0 : REM Cb value to the left of centre
ur = 0 : REM Cb value to the right of centre
vl = 0 : REM Cr value to the left of centre
vr = 0 : REM Cr value to the right of centre
Y% = 0 : REM luma value for pixel
z% = 0 : REM dummy value, not used

FOR y% = 1 TO high% : REM beware, BMP images are stored inverted
  FOR x%= 1 TO wide%
    PTR# in% = FNptr(x%, y%, wide%, high%, 3) + start%                                  : REM each pixel in turn
    u = FNUVdac(BGET# in%) * term(0) : Y% = BGET# in% : v = FNUVdac(BGET# in%) * term(0) : REM the chroma values
    FOR term% = 1 TO order%/2 : REM get the chroma pixels either side
      PTR# in%=FNptr(FNmax(x% - term%, 1), y%, wide%, high%, 3) + start%
      ul = FNUVdac(BGET# in%) : z% = BGET# in% :vl = FNUVdac(BGET# in%)
      PTR# in%=FNptr(FNmin(x% + term%, 1), y%, wide%, high%, 3) + start%
      ur = FNUVdac(BGET# in%) : z% = BGET# in% :vr = FNUVdac(BGET# in%)
      u += (ul + ur) * term(term%) : v += (vl + vr) * term(term%)                       : REM this is the filter output
    NEXT
    PTR# in% = FNptr(x%, y%, wide%, high%, 3) + start%                                  : REM back to the original centre pixel
    BPUT# in%, FNUVadc(u/term(0)) : BPUT# in%, Y% : BPUT# in%, FNUVadc(v/term(0))
  NEXT
  PRINTTAB(0,13) " Processing line ";y%
NEXT
CLOSE# in%

PRINTTAB(0,21) " 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(infile$)
  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))

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$(?A%) : A% += 1 : REM strip off characters until the first null
ENDWHILE
=s$

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 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 FNUVadc(A) : REM UV coder quantising, return digits
= 128 + A * 224

DEF FNUVdac(A) : REM undo coder UV digitising, return analogue
= (A - 128) / 224
```