Appendix B - BASIC program listing for the Meteosat receiving station

N.B. Comments are preceded by an apostrophe character " ' ".

```
' ****************************************************************
' ** Integrated Meteosat Image Reception Display and Processing **
' ** by John Shardlow 8th August 1990 University College London **
' ** MSc in Spacecraft Technology and Satellite Communications  **
' ****************************************************************
rem $option Y+       ' Open libraries and declare library functions
LIBRARY "dos.library"
LIBRARY "exec.library"
LIBRARY "graphics.library"

DECLARE FUNCTION xOpen&  LIBRARY
DECLARE FUNCTION xRead&  LIBRARY
DECLARE FUNCTION xWrite& LIBRARY
DECLARE FUNCTION AllocMem&() LIBRARY
' Make the default variable type an integer
defint a-z
' Allocate memory for the machine code routine and load it into
' the buffer storing the start address in "progptr&"
dim mem$(500)
bload "dh1:meteosat/getimage",varptr(mem$(0))
progptr&=varptr(mem$(0))+32
```

```
' Create arrays to store pixel kernels for processing routines
' then read the coefficients into the arrays
dim HPF!(3,3)
dim LPF!(3,3)
dim kernel!(3,3)
for dy=0 to 2
for dx=0 to 2
read HPF!(dx,dy):read LPF!(dx,dy)
next dx
next dy
data -1,.1,-1,.1,-1,.1,-1,.1,9,.2,-1,.1,-1,.1,-1,.1,-1,.1
' Open a window for text input/output on the default screen
window 2,"Meteosat Receiving Station (SDUS)",(0,0)-(640,256),276
' Set up the pull-down menus and wait for an item to be selected
gosub MenuSetup
on menu gosub MenuHandler
menu on
waitformenu:
sleep
goto waitformenu:
' Subroutine to initialise menus
MenuSetup:
menu 1,0,1,"Program"
menu 1,1,1,"About"
menu 1,2,1,"Quit"
menu 2,0,1,"Image"
menu 2,1,1,"Get Image"
menu 2,2,1,"Display Top"
```

```
menu 2,3,1,"Display Bottom"
menu 2,4,1,"Display Full"
menu 3,0,1,"Files"
menu 3,1,1,"List Raw Data Files"
menu 3,2,1,"List Picture Files"
menu 3,3,1,"Load Picture"
menu 3,4,1,"Print Picture"
menu 4,0,1,"Information"
menu 4,1,1,"Image Formats"
menu 4,2,1,"Time"
menu 5,0,1,"Process"
menu 5,1,1,"Low Pass Filter"
menu 5,2,1,"High Pass Filter"
return
' Subroutines to deal with menu selections and respond by calling
' the required routine
MenuHandler:
choice=menu(0)
item=menu(1)
if choice=1 then gosub Program
if choice=2 then gosub Image
if choice=3 then gosub FileRoutine
if choice=4 then gosub Information
if choice=5 then gosub Process
return
```

```
Program:
if item=2 then goto Omega
if item=1 then gosub About
return

Image:
if item=1 then gosub GetImage
if item=2 then gosub Top
if item=3 then gosub Bottom
if item=4 then gosub Full
return

FileRoutine:
if item=1 then gosub Raw
if item=2 then gosub Pics
if item=3 then gosub LoadPic
if item=4 then gosub PrintPic
return

Information:
if item=1 then gosub Formats
if item=2 then gosub Time
return

Process:
if item=1 then gosub LowPass
if item=2 then gosub HighPass
return
```

```
' This section is the only exit point from the program
Omega:
window close 2
menu reset
system

' Entry point to Low Pass Filter routine
LowPass:
' Load required kernel into an array
for dy=0 to 2
for dx=0 to 2
kernel!(dx,dy)=LPF!(dx,dy)
next
next
goto ProcessImage

' Entry point to High Pass Filter routine
HighPass:
' Load required kernel into an array
for dy=0 to 2
for dx=0 to 2
kernel!(dx,dy)=HPF!(dx,dy)
next
next
goto ProcessImage
```

```
' This section is shared by both processing routines
ProcessImage:

input "Filename of Picture to be processed";f$
ILBMname$="dh1:iff/"+f$
gosub LoadILBM
' Input image in window 3 - output image in window 4
WINDOW 4,"",(0,0)-(scrWidth%,scrHeight%),16+256+128+32,2
FOR y=0 TO (scrHeight%-1)
FOR x=0 TO (scrWidth%-1)
TotalBV!=0:WINDOW OUTPUT 3
FOR dx=0 TO 2:FOR dy=0 TO 2
bv!=POINT(x-dx-1,y-dy-1)*kernel!(dx,dy)
TotalBV!=TotalBV!+bv!
NEXT:NEXT
WINDOW OUTPUT 4
IF TotalBV!<0 THEN TotalBV!=0
IF TotalBV!>15 THEN TotalBV!=15
PSET (x,y),int(TotalBV!)
NEXT:NEXT
window close 3
window 3,"",(20,200)-(300,250),16,2
color 15,0
input "Filename for picture ";file$
ILBMname$="dh1:iff/"+file$
window close 3
window output 4
if file$<>"" then gosub SaveILBM
```

```
window close 4
screen close 2
window output 2
RETURN

' This routine calls the machine code program to read data from
' the parallel port
GetImage:
print "Press any key when start tone begins"
while inkey$="":wend
print "Press Left Mouse Button to continue after image reception"
menu off
rem $event off
call loc progptr&
rem $event on
menu on
input "Filename for raw image data ";file$
last$=file$
file$="dh1:images/"+file$
name "dh1:images/temp" as file$
return

' List raw image data files from receiver
Raw:
files "dh1:images"
return
```

```
' List picture files produced by the display routines
Pics:
files "dh1:iff"
return

' Print the time from the system clock
Time:
print
print "Current Time is ";time$
print
return

' Print information about the program
About:
print "************************************************"
print "* Meteosat Secondary Data User Station Software *"
print "************************************************"
print
print "Written by John Shardlow"
print
print "Dept. of Physics and Astronomy and"
print "Dept. of Electronic and Electrical Engineering"
print "University College London"
print
print "MSc in Spacecraft Technology and Satellite Communications"
print
return
```

```
' Display the top 500 lines of an image in full resolution
Top:
gosub buffermem
if file$="" then return
bufferpos&=buffer&+buffersize&-1-offset
bufend&=bufferpos&-(int(500*wide!))
label1:
y=int(count&/wide!)
x=count&-(y*wide!)
byte%=peek(bufferpos&)
pset(x,y),(int(byte%/16))
incr count&
decr bufferpos&
if bufferpos&=bufend& then goto finish:
if inkey$<>"Q" then goto label1 else goto finish
finish:
gosub SavePic
window close 3
screen close 2
call FreeMem&(buffer&,buffersize&)
return

' Display the bottom 500 lines of an image in full resolution
Bottom:
gosub buffermem
if file$="" then return
bufferpos&=buffer&+offset
bufend&=bufferpos&+(int(500*wide!))
```

```
label2:
y=int(count&/wide!)
x=count&-(y*wide!)
byte%=peek(bufferpos&)
pset((310-x),(511-y)),(int(byte%/16))
incr count&
incr bufferpos&
if bufferpos&=bufend& then goto finish:
if inkey$<>"Q" then goto label2 else goto finish

' Display an entire image in half of the full resolution
Full:
gosub buffermem
if file$="" then return
bufferpos&=buffer&+buffersize&-1-offset
bufend&=buffer&
label3:
y=int(count&/wide!)
x=count&-(y*wide!)
byte%=peek(bufferpos&)
y2=y/2
if (y mod 2)=0 then pset(x,y2),(int(byte%/16))
incr count&
decr bufferpos&
if bufferpos&=bufend& then goto finish:
if inkey$<>"Q" then goto label3 else goto finish
```

```
' This routine is shared by all three display routines and is used
' to allocate a memory buffer for the image data, load the data in
' and open a graphics screen to plot the image
buffermem:
file$=""
buffersize&=255600
buffer&=0
ClearPublic&=65537
buffer&=AllocMem&(buffersize&,ClearPublic&)
if buffer&=0 then print "Not enough memory":return
input "Filename of image data (RETURN for most recent file) ";f$
if f$<>"" then file$=f$ else file$=last$
last$=file$
file$="dh1:images/"+file$
if not fexists(file$) then
print "This file does not exist"
file$=""
return
end if
bload file$,buffer&
print "Press Q (capital q) to exit before end of frame"

' Prompt for input of values to correct for offset and
' shear in image
input "How many bytes offset ";offset
input "How many pixels per line (299-301) ";wide!
screen 2,320,512,4,3
window 3,"",(0,0)-(320,512),256+128+32+16,2
```

```
' Set the 16 colour palette to a grey scale
for col=0 to 15
clr!=col/15
palette col,clr!,clr!,clr!
next col
color 15,0
count&=0
return

' Prompt for a filename for the picture then call a routine to
' save the picture in a standard graphics file format (ILBM)
SavePic:
window 4,"",(20,200)-(300,250),16,2
color 15,0
input "Filename for picture ";file$
ILBMname$="dh1:iff/"+file$
window close 4
window output 3
if file$<>"" then gosub SaveILBM
window output 2
return

' Load a picture file onto the screen
LoadPic:
input "Filename of picture ";file$
ILBMname$="dh1:iff/"+file$
```

```
if not fexists(ILBMname$) then
print "This file does not exist"
return
end if
gosub LoadILBM
while inkey$="":wend
window close 3
screen close 2
window output 2
return

' Load a picture file and print it on a graphics printer
PrintPic:
input "Filename of picture ";file$
ILBMname$="dh1:iff/"+file$
if not fexists(ILBMname$) then
print "This file does not exist"
return
end if
gosub LoadILBM
pcopy
window close 3
screen close 2
window output 2
return
```

```
' Load the image formats diagram
Formats:
ILBMname$="dh1:meteosat/wefax-formats"
if not fexists(ILBMname$) then
print "Cannot find image formats diagram"
return
end if
gosub LoadILBM
while inkey$="":wend
window close 3
screen close 2
window output 2
return

' This routine saves the graphics image on the current screen as
' a standard graphics file called an InterLeaved BitMap (ILBM)
SaveILBM:
f$ = ILBMname$
fHandle& = 0
mybuf& = 0
filename$ = f$ + CHR$(0)
fHandle& = xOpen&(SADD(filename$),1006)
IF fHandle& = 0 THEN
saveError$ = "Can't open output file"
GOTO Scleanup
END IF
```

```
' Allocate RAM for work buffers
ClearPublic& = 65537&
mybufsize& = 120
mybuf& = AllocMem&(mybufsize&,ClearPublic&)
IF mybuf& = 0 THEN
saveError$ = "Can't alloc buffer"
GOTO Scleanup
END IF
cbuf& = mybuf&
' Get addresses of screen structures
GOSUB GetScrAddrs
zero& = 0
pad%  = 0
aspect% = &HA0B

' Compute chunk sizes
BMHDsize& = 20
CMAPsize& = (2^scrDepth%) * 3
CAMGsize& = 4
CCRTsize& = 14
BODYsize& = (scrWidth%/8) * scrHeight% * scrDepth%
FORMsize& = BMHDsize&+CMAPsize&+CAMGsize&+BODYsize&+36

' Write FORM header
tt$ = "FORM"
wLen& = xWrite&(fHandle&,SADD(tt$),4)
wLen& = xWrite&(fHandle&,VARPTR(FORMsize&),4)
tt$ = "ILBM"
```

```
wLen& = xWrite&(fHandle&,SADD(tt$),4)
IF wLen& <= 0 THEN
saveError$ = "Error writing FORM header"
GOTO Scleanup
END IF

' Write out BMHD chunk
tt$ = "BMHD"
wLen& = xWrite&(fHandle&,SADD(tt$),4)
wLen& = xWrite&(fHandle&,VARPTR(BMHDsize&),4)
wLen& = xWrite&(fHandle&,VARPTR(scrWidth%),2)
wLen& = xWrite&(fHandle&,VARPTR(scrHeight%),2)
wLen& = xWrite&(fHandle&,VARPTR(zero&),4)
temp% = (256 * scrDepth%)
wLen& = xWrite&(fHandle&,VARPTR(temp%),2)
wLen& = xWrite&(fHandle&,VARPTR(zero&),4)
wLen& = xWrite&(fHandle&,VARPTR(aspect%),2)
wLen& = xWrite&(fHandle&,VARPTR(scrWidth%),2)
wLen& = xWrite&(fHandle&,VARPTR(scrHeight%),2)

IF wLen& <= 0 THEN
saveError$ = "Error writing BMHD"
GOTO Scleanup
END IF

' Write CAMG chunk
tt$ = "CAMG"
wLen& = xWrite&(fHandle&,SADD(tt$),4)
```

```
wLen& = xWrite&(fHandle&,VARPTR(CAMGsize&),4)
vpModes& = PEEKW(sViewPort& + 32)
wLen& = xWrite&(fHandle&,VARPTR(vpModes&),4)
IF wLen& <= 0 THEN
saveError$ = "Error writing CAMG"
GOTO Scleanup
END IF

' Write CMAP chunk
tt$ = "CMAP"
wLen& = xWrite&(fHandle&,SADD(tt$),4)
wLen& = xWrite&(fHandle&,VARPTR(CMAPsize&),4)

' Build IFF ColorMap
FOR kk = 0 TO nColors% - 1
regTemp% = PEEKW(colorTab& + (2*kk))
POKE(cbuf&+(kk*3)),(regTemp% AND &HF00) / 16
POKE(cbuf&+(kk*3)+1),(regTemp% AND &HF0)
POKE(cbuf&+(kk*3)+2),(regTemp% AND &HF) * 16
NEXT

wLen& = xWrite&(fHandle&,cbuf&,CMAPsize&)
IF wLen& <= 0 THEN
saveError$ = "Error writing CMAP"
GOTO Scleanup
END IF
```

```
' Write BODY chunk
tt$ = "BODY"
wLen& = xWrite&(fHandle&,SADD(tt$),4)
wLen& = xWrite&(fHandle&,VARPTR(BODYsize&),4)
scrRowBytes% = scrWidth% / 8
FOR rr = 0 TO scrHeight% -1
FOR pp = 0 TO scrDepth% -1
scrRow& = bPlane&(pp)+(rr*scrRowBytes%)
wLen& = xWrite&(fHandle&,scrRow&,scrRowBytes%)
IF wLen& <= 0 THEN
saveError$ = "Error writing BODY"
GOTO Scleanup
END IF
NEXT
NEXT
saveError$ = ""
Scleanup:
IF fHandle& <> 0 THEN CALL xClose&(fHandle&)
IF mybuf& <> 0 THEN CALL FreeMem&(mybuf&,mybufsize&)
RETURN

GetScrAddrs:
' Get addresses of screen structures
sWindow&   = WINDOW(7)
sScreen&   = PEEKL(sWindow& + 46)
sViewPort& = sScreen& + 44
sRastPort& = sScreen& + 84
sColorMap& = PEEKL(sViewPort& + 4)
```

```
colorTab&   = PEEKL(sColorMap& + 4)
sBitMap&    = PEEKL(sRastPort& + 4)

' Get screen parameters
scrWidth%  = PEEKW(sScreen& + 12)
scrHeight% = PEEKW(sScreen& + 14)
scrDepth%  = PEEK(sBitMap& + 5)
nColors%   = 2^scrDepth%

' Get addresses of Bit Planes
FOR kk = 0 TO scrDepth% - 1
bPlane&(kk) = PEEKL(sBitMap&+8+(kk*4))
NEXT
RETURN

' This routine loads an InterLeaved BitMap (ILBM) file onto a screen
' which it opens automatically

LoadILBM:
f$ = ILBMname$
fHandle& = 0
mybuf& = 0
foundBMHD = 0
foundCMAP = 0
foundCamg = 0
foundCCRT = 0
foundBODY = 0
filename$ = f$ + CHR$(0)
```

```
fHandle& = xOpen&(SADD(filename$),1005)
IF fHandle& = 0 THEN
loadError$ = "Can't open/find pic file"
GOTO Lcleanup
END IF

ClearPublic& = 65537&
mybufsize& = 360
mybuf& = AllocMem&(mybufsize&,ClearPublic&)
IF mybuf& = 0 THEN
loadError$ = "Can't alloc buffer"
GOTO Lcleanup
END IF

inbuf& = mybuf&
cbuf& = mybuf& + 120
ctab& = mybuf& + 240

rLen& = xRead&(fHandle&,inbuf&,12)
tt$ = ""
FOR kk = 8 TO 11
tt% = PEEK(inbuf& + kk)
tt$ = tt$ + CHR$(tt%)
NEXT
IF tt$ <> "ILBM" THEN
loadError$ = "Not standard ILBM pic file"
GOTO Lcleanup
END IF
```

```
' - Read ILBM chunks
ChunkLoop:
' - Get Chunk name/length
rLen& = xRead&(fHandle&,inbuf&,8)
icLen& = PEEKL(inbuf& + 4)
tt$ = ""
FOR kk = 0 TO 3
tt% = PEEK(inbuf& + kk)
tt$ = tt$ + CHR$(tt%)
NEXT

IF tt$ = "BMHD" THEN  'BitMap header
foundBMHD = 1
rLen& = xRead&(fHandle&,inbuf&,icLen&)
iWidth%  = PEEKW(inbuf&)
iHeight% = PEEKW(inbuf& + 2)
iDepth%  = PEEK(inbuf& + 8)
iCompr%  = PEEK(inbuf& + 10)
scrWidth%  = PEEKW(inbuf& + 16)
scrHeight% = PEEKW(inbuf& + 18)

iRowBytes% = iWidth% /8
scrRowBytes% = scrWidth% / 8
nColors%  = 2^(iDepth%)

' Enough free ram to display ?
AvailRam& = FRE(-1)
NeededRam& = ((scrWidth%/8)*scrHeight%*(iDepth%+1))+5000
```

```
IF AvailRam& < NeededRam& THEN
loadError$ = "Not enough free ram"
GOTO Lcleanup
END IF

hires& = &H8000
lace&  = &H4
kk = 1
IF foundCamg THEN
IF (camgModes& AND hires&) THEN kk = kk+1
IF (camgModes& AND lace&)  THEN kk = kk+2
ELSE
IF scrWidth% >= 640 THEN kk = kk + 1
IF scrHeight% >= 400 THEN kk = kk + 2
END IF

SCREEN 2,scrWidth%,scrHeight%,iDepth%,kk
WINDOW 3,"",(0,0)-(scrWidth%,scrHeight%),256+128+32+16,2

' Get addresses of structures
GOSUB GetScrAddrs
' Black out screen
CALL LoadRGB4&(sViewPort&,ctab&,nColors%)
ELSEIF tt$ = "CMAP" THEN  'ColorMap
foundCMAP = 1
rLen& = xRead&(fHandle&,cbuf&,icLen&)
```

```
' Build Color Table
FOR kk = 0 TO nColors% - 1
red% = PEEK(cbuf&+(kk*3))
gre% = PEEK(cbuf&+(kk*3)+1)
blu% = PEEK(cbuf&+(kk*3)+2)
regTemp% = (red%*16)+(gre%)+(blu%/16)
POKEW(ctab&+(2*kk)),regTemp%
NEXT

ELSEIF tt$ = "CAMG" THEN  'Amiga ViewPort Modes
foundCamg = 1
rLen& = xRead&(fHandle&,inbuf&,icLen&)
camgModes& = PEEKL(inbuf&)

ELSEIF tt$ = "CCRT" THEN 'Graphicraft color cycle info
foundCCRT = 1
rLen& = xRead&(fHandle&,inbuf&,icLen&)
ccrtDir%     = PEEKW(inbuf&)
ccrtStart%   = PEEK(inbuf& + 2)
ccrtEnd%     = PEEK(inbuf& + 3)
ccrtSecs&    = PEEKL(inbuf& + 4)
ccrtMics&    = PEEKL(inbuf& + 8)

ELSEIF tt$ = "BODY" THEN  'BitMap
foundBODY = 1
IF iCompr% = 0 THEN  'no compression
FOR rr& = 0 TO iHeight% -1
FOR pp& = 0 TO iDepth% -1
```

```
scrRow& = bPlane&(pp&)+(rr&*scrRowBytes%)
rLen& = xRead&(fHandle&,scrRow&,iRowBytes%)
NEXT
NEXT
ELSEIF iCompr% = 1 THEN  'cmpByteRun1
FOR rr& = 0 TO iHeight% -1
FOR pp& = 0 TO iDepth% -1
scrRow& = bPlane&(pp&)+(rr&*scrRowBytes%)
bCnt% = 0
WHILE (bCnt% < iRowBytes%)
rLen& = xRead&(fHandle&,inbuf&,1)
inCode% = PEEK(inbuf&)
IF inCode% < 128 THEN
rLen& = xRead&(fHandle&,scrRow& + bCnt%, inCode%+1)
bCnt% = bCnt% + inCode% + 1
ELSEIF inCode% > 128 THEN
rLen& = xRead&(fHandle&,inbuf&,1)
inByte% = PEEK(inbuf&)
FOR kk = bCnt% TO bCnt% + 257 - inCode%
POKE(scrRow&+kk),inByte%
NEXT
bCnt% = bCnt% + 257 - inCode%
END IF
WEND
NEXT
NEXT
```

```
ELSE
loadError$ = "Unknown compression algorithm"
GOTO Lcleanup
END IF
ELSE
' Reading unknown chunk
FOR kk = 1 TO icLen&
rLen& = xRead&(fHandle&,inbuf&,1)
NEXT
' If odd length, read 1 more byte
IF (icLen& OR 1) = icLen& THEN
rLen& = xRead&(fHandle&,inbuf&,1)
END IF
END IF
' Done if got all chunks
IF foundBMHD AND foundCMAP AND foundBODY THEN
GOTO GoodLoad
END IF
' Good read, get next chunk
IF rLen& > 0 THEN GOTO ChunkLoop
IF rLen& < 0 THEN  'Read error
loadError$ = "Read error"
GOTO Lcleanup
END IF
' - rLen& = 0 means EOF
IF (foundBMHD=0) OR (foundBODY=0) OR (foundCMAP=0) THEN
loadError$ = "Needed ILBM chunks not found"
GOTO Lcleanup
```

```
END IF
GoodLoad:
loadError$ = ""
'   Load proper Colors
IF foundCMAP THEN
CALL LoadRGB4&(sViewPort&,ctab&,nColors%)
END IF

Lcleanup:
IF fHandle& <> 0 THEN CALL xClose&(fHandle&)
IF mybuf& <> 0 THEN CALL FreeMem&(mybuf&,mybufsize&)
RETURN
```