***************Draft****************

Openjpeg for Ultibo

changing openjpeg from cmake to script compile_ultibo.sh creates a static library using the arm-none-eabi tools that can be called by Ultibo

04/28/19

***************Draft****************

Compile 22 C files from openjpeg using arm-none-eabi-gcc &
arm-none-eabi-ar to create libopenjp2.a. Then use
arm-none-eabi-gcc & arm-none-eabi-ar to compile dwtlift.c and
link it with libopenjp2.a to create libdwtlift.a. This C library
libdwtlift.a is used by Ultibo written in Pascal to
read the bmp file MyBitmap.bmp and pass it to C library libdwtlift.a.
When j2k jpeg encoded is completed by C library libdwtlift.a
it is written to the micro sd card as file test.j2k.

```
pi@mypi3-1:~/t_ultibo/build/bin $ ./opj_dump -i ~/test.j2k

[INFO] Start to read j2k main header (16370952).
[INFO] Main header has been correctly decoded.
Image info {
	x0=0, y0=0
	x1=2048, y1=2048
	numcomps=3
		component 0 {
          dx=1, dy=1
          prec=8
          sgnd=0
     }
          component 1 {
          dx=1, dy=1
          prec=8
          sgnd=0
     }
          component 2 {
          dx=1, dy=1
          prec=8
          sgnd=0
     }
}
Codestream info from main header: {
      tx0=0, ty0=0
      tdx=2048, tdy=2048
      tw=1, th=1
      default tile {
          csty=0
          prg=0
          numlayers=1
          mct=0
          comp 0 {
               csty=0
               numresolutions=6
               cblkw=2^6
               cblkh=2^6
```

```
			cblksty=0
			qmfbid=1
			preccintsize (w,h)=(15,15) (15,15) (15,15)
(15,15) (15,15) (15,15)
			qntsty=0
			numgbits=2
			stepsizes (m,e)=(0,8) (0,9) (0,9) (0,10) (0,9)
(0,9) (0,10) (0,9) (0,9) (0,10) (0,9) (0,9) (0,10) (0,9) (0,9)
(0,10)
			roishift=0
		}
		comp 1 {
			csty=0
			numresolutions=6
			cblkw=2^6
			cblkh=2^6
			cblksty=0
			qmfbid=1
			preccintsize (w,h)=(15,15) (15,15) (15,15)
(15,15) (15,15) (15,15)
			qntsty=0
			numgbits=2
			stepsizes (m,e)=(0,8) (0,9) (0,9) (0,10) (0,9)
(0,9) (0,10) (0,9) (0,9) (0,10) (0,9) (0,9) (0,10) (0,9) (0,9)
(0,10)
			roishift=0
		}
		comp 2 {
			csty=0
```

```
                        numresolutions=6
                        cblkw=2^6
                        cblkh=2^6
                        cblksty=0
                        qmfbid=1
                        preccintsize (w,h)=(15,15) (15,15) (15,15)
(15,15) (15,15) (15,15)
                        qntsty=0
                        numgbits=2
                        stepsizes (m,e)=(0,8) (0,9) (0,9) (0,10) (0,9)
(0,9) (0,10) (0,9) (0,9) (0,10) (0,9) (0,9) (0,10) (0,9) (0,9)
(0,10)
                        roishift=0
                }
        }
}
Codestream index from main header: {
        Main header start position=0
        Main header end position=125
        Marker list: {
                type=0xff4f, pos=0, len=2
                type=0xff51, pos=2, len=49
                type=0xff52, pos=51, len=14
                type=0xff5c, pos=65, len=21
                type=0xff64, pos=86, len=39
        }
}
```

Below is the decompressed bmp obtained from the test.j2k compressed on Ultibo. This took 16 sec,

Checking that change has been in included in Ultibo
/home/pi/Core
git format-patch --stdout 4c5e8d0bd6 | grep ShareMode
- ShareMode:=0;
+ ShareMode:=FILE_SHARE_READ or

FILE_SHARE_WRITE;

~~Note: The current verison requires a minor change in the file syscalls.pas and the RTL rebuilt for ULTIBO_RELEASE_DATE 30 July 2017 ULTIBO_RELEASE_VERSION 1.3.411~~

~~diff ultibo/core/fpc/source/rtl/ultibo/core/syscalls.pas ultibo/core/fpc/source/rtl/ultibo/core/syscalls.pas.orig~~
~~2321c2321~~
~~< ShareMode:=FILE_SHARE_READ or FILE_SHARE_WRITE;~~
~~---~~
~~> ShareMode:=0;~~
~~7634c7634~~
~~<~~
~~---~~
~~>~~
~~\ No newline at end of file~~

The current testing of openjpeg for Ultibo requires 2 repositories.
This currently takes 15 steps below.
Creating the libopenjp2.a
Fetching the openjpeg repository and switching to the ulitibo branch and jpeg-2000-test repository

git clone https://github.com/develone/jpeg-2000-test.git

commit e821d8ab61bd5073618304d0378b8bffe6db44af
Author: Edward Vidal Jr <develone@sbcglobal.net>
Date: Wed Feb 27 13:18:47 2019 -0700

git clone https://github.com/develone/openjpeg.git t_ultibo/
cd t_ultibo/
opjpeg master
commit 0b4c3ce75d11600ebc6675bd871f78ca3c95bc60
Merge: a35b4891 9f750884
Author: Even Rouault <even.rouault@mines-paris.org>
Date: Fri Aug 11 15:13:35 2017 +0200

This builds ultibo branch apps & static lib. A new header src/lib/openjp2/opj_common.h

git checkout ultibo the ver before 02/27/19 chgs
commit e357de93fb006ec0b84b2a217b6a047bad595eba
Author: Edward Vidal Jr <develone@sbcglobal.net>
Date: Mon Aug 28 20:18:11 2017 +0000

cd build/
cmake ../ cmake_results.txt
make

Testing RaspBian Builds

```
pi@raspberrypi3:~/t_ultibo/build/bin $ ./test_tile_encoder
Using default image ../../lena_rgb_2048.bmp 1
```

pi@raspberrypi3:~/t_ultibo/build/bin $ ***time ./test_tile_encoder***

```
Using default image ../../lena_rgb_2048.bmp 1
testing if bitmap BM bpp = 24
allocating 0x7694b008 0x7654a008 0x76149008
size = 12582912
planes = 4096
colours = 0
height = 2048 width = 2048
bpp = 24
xresolution = 1996034048 yresolution 2129114500
rgb from Matrix to r g b ptrs
splitting data to rgb
0x7694b008 0x7654a008 0x76149008
enc 1 decomp 3 distor 44 filter 0
l_nb_tiles 1 l_data_size 12582912
loading RGB data
TRANSFER 0x7694b008 0x7654a008 0x76149008
before reset 0x7694b008 0x7654a008 0x76149008
In test_tile_encoder
creating J2k
[INFO] tile number 1 / 1
[ERROR] opj_t2_encode_packet(): only 701 bytes remaining in output buffer. 786 needed.
[ERROR] opj_t2_encode_packet(): only 2 bytes remaining in output buffer. 194 needed.
[ERROR] opj_t2_encode_packet(): only 85 bytes remaining in output buffer. 346 needed.
[ERROR] opj_t2_encode_packet(): only 15 bytes remaining in output buffer. 82 needed.
[ERROR] opj_t2_encode_packet(): only 15 bytes remaining in output buffer. 82 needed.
[ERROR] opj_t2_encode_packet(): only 15 bytes remaining in output buffer. 82 needed.
[ERROR] opj_t2_encode_packet(): only 15 bytes remaining in output buffer. 82 needed.
[ERROR] opj_t2_encode_packet(): only 15 bytes remaining in output buffer. 82 needed.
[ERROR] opj_t2_encode_packet(): only 15 bytes remaining in output buffer. 82 needed.
[ERROR] opj_t2_encode_packet(): only 15 bytes remaining in output buffer. 82 needed.
[ERROR] opj_t2_encode_packet(): only 15 bytes remaining in output buffer. 82 needed.
[ERROR] opj_t2_encode_packet(): only 15 bytes remaining in output buffer. 82 needed.
[ERROR] opj_t2_encode_packet(): only 15 bytes remaining in output buffer. 82 needed.
[ERROR] opj_t2_encode_packet(): only 15 bytes remaining in output buffer. 82 needed.
[ERROR] opj_t2_encode_packet(): only 15 bytes remaining in output buffer. 82 needed.
[ERROR] opj_t2_encode_packet(): only 15 bytes remaining in output buffer. 82 needed.
[ERROR] opj_t2_encode_packet(): only 15 bytes remaining in output buffer. 82 needed.
[ERROR] opj_t2_encode_packet(): only 15 bytes remaining in output buffer. 82 needed.
```

```
[ERROR] opj_t2_encode_packet(): only 15 bytes remaining in output buffer. 82 needed.
[ERROR] opj_t2_encode_packet(): only 15 bytes remaining in output buffer. 82 needed.
[ERROR] opj_t2_encode_packet(): only 15 bytes remaining in output buffer. 82 needed.
[ERROR] opj_t2_encode_packet(): only 15 bytes remaining in output buffer. 82 needed.
[ERROR] opj_t2_encode_packet(): only 15 bytes remaining in output buffer. 82 needed.
FREE 0x7694b008 0x7654a008 0x76149008

real    0m12.371s
user    0m10.610s
sys     0m1.710s
```

pi@raspberrypi3:~/t_ultibo/build/bin $ ***time ./test_tile_decoder 0 0 2048 2048 test.j2k***

```
[INFO] Start to read j2k main header (2375928).
[INFO] Main header has been correctly decoded.
[INFO] Setting decoding area to 0,0,2048,2048
[INFO] Header of tile 1 / 1 has been read.

real    0m3.974s
user    0m3.880s
sys     0m0.090s
```

Create the lib to be used by ultibo

```
cd t_ultibo/src/lib/openjp2
pi@raspberrypi3:~/t_ultibo/src/lib/openjp2 $ ls ../../../*.h
../../../opj_config.h  ../../../opj_config_private.h  ../../../opj_includes.h
```

Note : The above headers may require upgrade.

pi@raspberrypi3:~/t_ultibo/src/lib/openjp2 $ ***cp ../../../*.h .***
pi@raspberrypi3:~/t_ultibo/src/lib/openjp2 $ ***./compile_ultibo.sh***

```
pi@raspberrypi3:~/t_ultibo/src/lib/openjp2 $ ls -la libopenjp2.a
-rw-r--r-- 1 pi pi 393428 Feb 27 13:40 libopenjp2.a

cd ~/jpeg-2000-test/bare-metal/openjp/
```

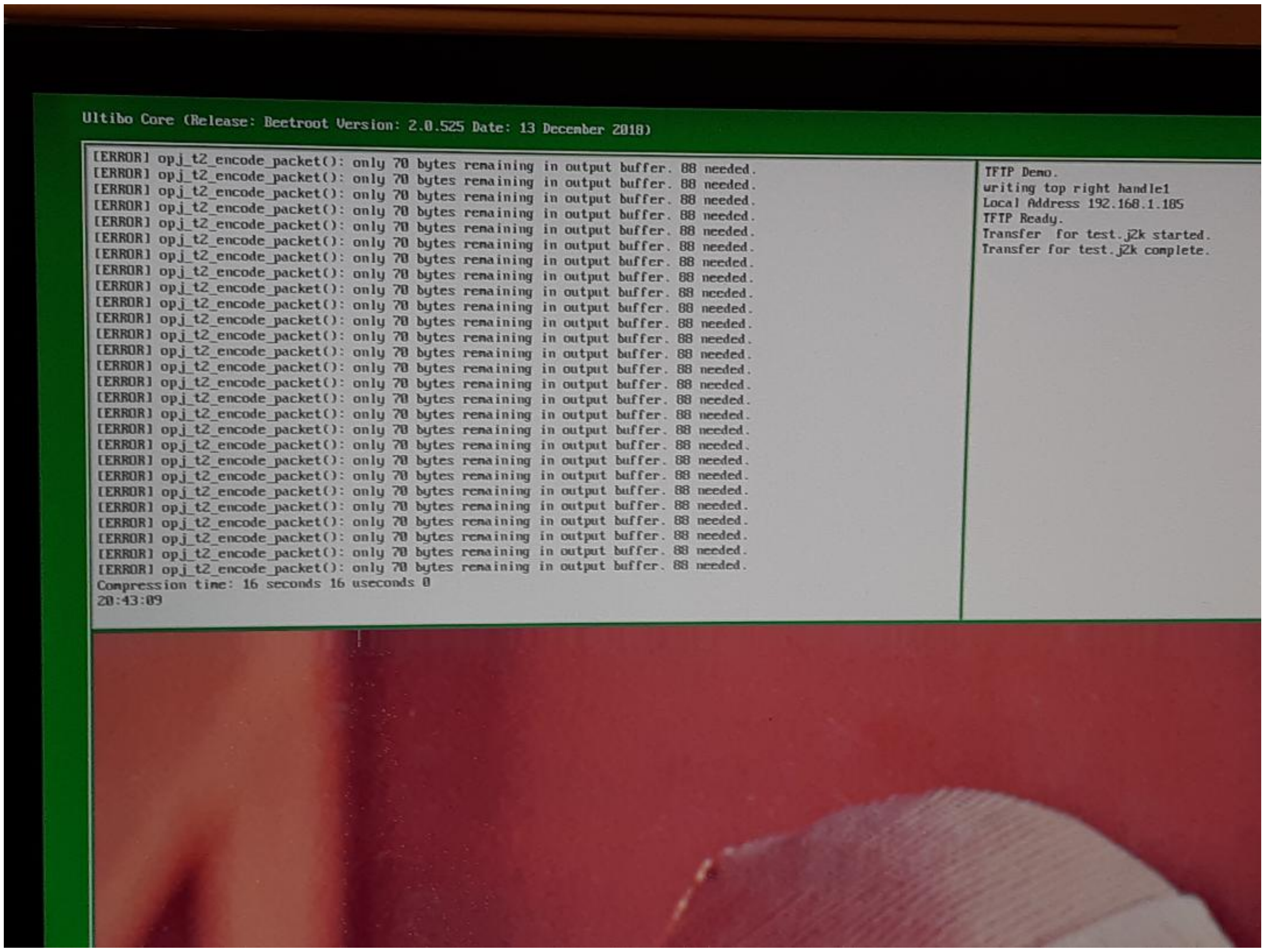
pi@raspberrypi3:~/jpeg-2000-test/bare-metal/openjp $ ***export PATH=/home/pi/ultibo/core/fpc/bin:$PATH***

Note: Needed to chg rpi2.cfg RPI2.CFG in the file
jpeg-2000-test/bare-metal/openjp/compile.sh
on testing 02/27/19

```
pi@mypi3-4:~/jpeg-2000-test/bare-metal/openjp $ ls -la libopenjp2.a
-rw-r--r-- 1 pi pi 415250 Feb 27 13:41 libopenjp2.a
pi@mypi3-4:~/jpeg-2000-test/bare-metal/openjp $ ls -la kernel7.img
-rwxr-xr-x 1 pi pi 2984884 Feb 27 13:41 kernel7.img
```

pi@raspberrypi3:~/jpeg-2000-test/bare-metal/openjp $ ***tftp 192.168.1.185 < cmdstftp***

tftp 192.168.1.185
tftp> binary
tftp> get test.j2k dtest.j2k
Received 125603 bytes in 0.5 seconds

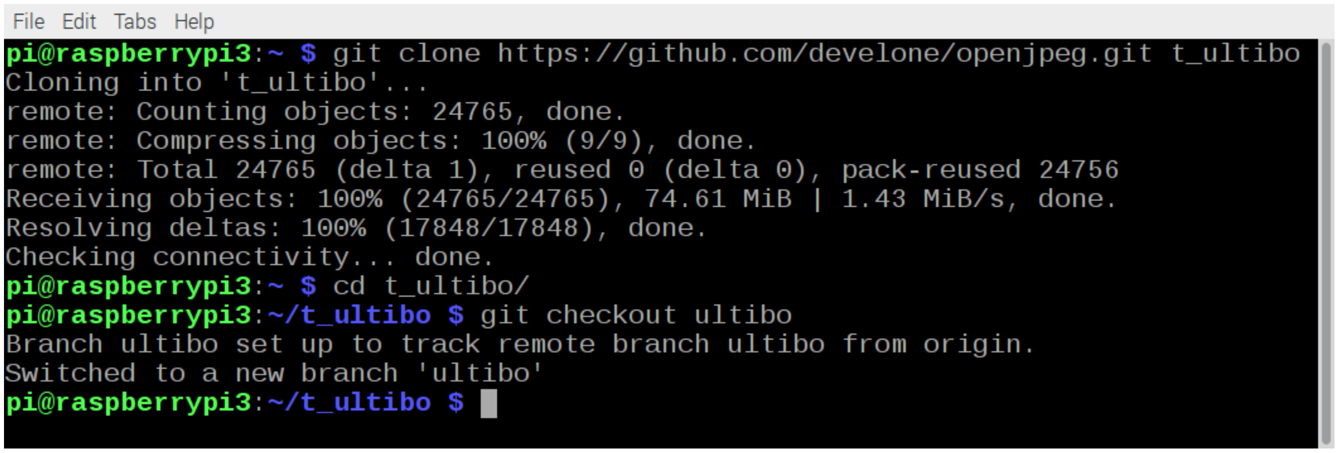


```
Ultibo Core (Release: Beetroot Version: 2.0.525 Date: 13 December 2018)

[ERROR] opj_t2_encode_packet(): only 70 bytes remaining in output buffer. 88 needed.
[ERROR] opj_t2_encode_packet(): only 70 bytes remaining in output buffer. 88 needed.
[ERROR] opj_t2_encode_packet(): only 70 bytes remaining in output buffer. 88 needed.
[ERROR] opj_t2_encode_packet(): only 70 bytes remaining in output buffer. 88 needed.
[ERROR] opj_t2_encode_packet(): only 70 bytes remaining in output buffer. 88 needed.
[ERROR] opj_t2_encode_packet(): only 70 bytes remaining in output buffer. 88 needed.
[ERROR] opj_t2_encode_packet(): only 70 bytes remaining in output buffer. 88 needed.
[ERROR] opj_t2_encode_packet(): only 70 bytes remaining in output buffer. 88 needed.
[ERROR] opj_t2_encode_packet(): only 70 bytes remaining in output buffer. 88 needed.
[ERROR] opj_t2_encode_packet(): only 70 bytes remaining in output buffer. 88 needed.
[ERROR] opj_t2_encode_packet(): only 70 bytes remaining in output buffer. 88 needed.
[ERROR] opj_t2_encode_packet(): only 70 bytes remaining in output buffer. 88 needed.
[ERROR] opj_t2_encode_packet(): only 70 bytes remaining in output buffer. 88 needed.
[ERROR] opj_t2_encode_packet(): only 70 bytes remaining in output buffer. 88 needed.
[ERROR] opj_t2_encode_packet(): only 70 bytes remaining in output buffer. 88 needed.
[ERROR] opj_t2_encode_packet(): only 70 bytes remaining in output buffer. 88 needed.
[ERROR] opj_t2_encode_packet(): only 70 bytes remaining in output buffer. 88 needed.
[ERROR] opj_t2_encode_packet(): only 70 bytes remaining in output buffer. 88 needed.
[ERROR] opj_t2_encode_packet(): only 70 bytes remaining in output buffer. 88 needed.
[ERROR] opj_t2_encode_packet(): only 70 bytes remaining in output buffer. 88 needed.
[ERROR] opj_t2_encode_packet(): only 70 bytes remaining in output buffer. 88 needed.
[ERROR] opj_t2_encode_packet(): only 70 bytes remaining in output buffer. 88 needed.
[ERROR] opj_t2_encode_packet(): only 70 bytes remaining in output buffer. 88 needed.
[ERROR] opj_t2_encode_packet(): only 70 bytes remaining in output buffer. 88 needed.
[ERROR] opj_t2_encode_packet(): only 70 bytes remaining in output buffer. 88 needed.
[ERROR] opj_t2_encode_packet(): only 70 bytes remaining in output buffer. 88 needed.
[ERROR] opj_t2_encode_packet(): only 70 bytes remaining in output buffer. 88 needed.
Compression time: 16 seconds 16 useconds 0
20:43:09
```

```
TFTP Demo.
writing top right handle1
Local Address 192.168.1.185
TFTP Ready.
Transfer  for test.j2k started.
Transfer for test.j2k complete.
```

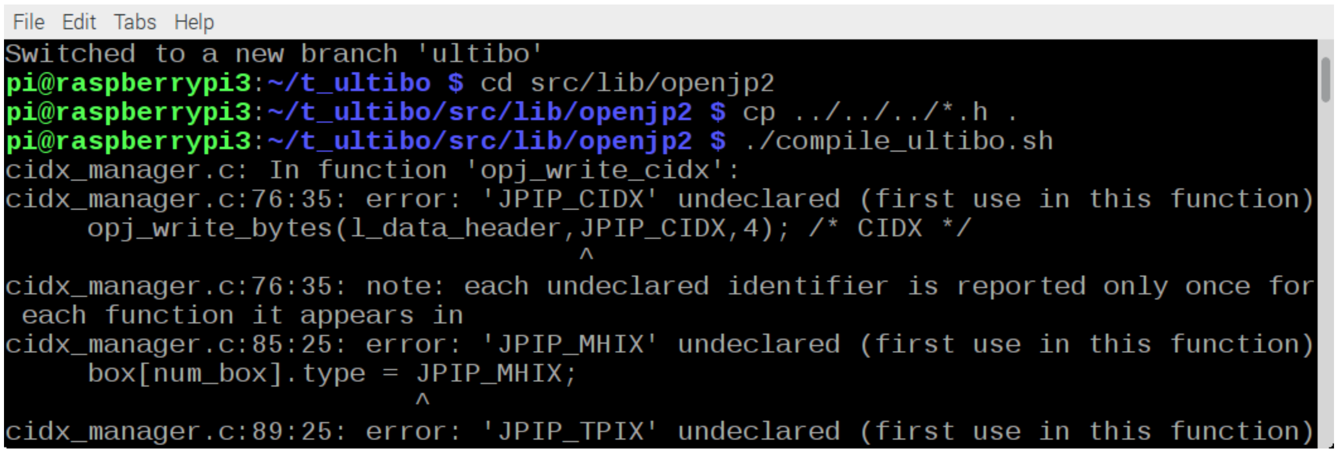
pi@raspberrypi3:~/jpeg-2000-test/bare-metal/openjp $ ***./opj_decompress -i test.j2k -o 100CR.bmp***

[INFO] Start to read j2k main header (17411336).
[INFO] Main header has been correctly decoded.
[INFO] No decoded area parameters, set the decoded area to the whole image
[INFO] Header of tile 1 / 1 has been read.
[INFO] Generated Outfile 100CR.bmp
decode time: 1539 ms



```
pi@raspberrypi3:~ $ git clone https://github.com/develone/openjpeg.git t_ultibo
Cloning into 't_ultibo'...
remote: Counting objects: 24765, done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 24765 (delta 1), reused 0 (delta 0), pack-reused 24756
Receiving objects: 100% (24765/24765), 74.61 MiB | 1.43 MiB/s, done.
Resolving deltas: 100% (17848/17848), done.
Checking connectivity... done.
pi@raspberrypi3:~ $ cd t_ultibo/
pi@raspberrypi3:~/t_ultibo $ git checkout ultibo
Branch ultibo set up to track remote branch ultibo from origin.
Switched to a new branch 'ultibo'
pi@raspberrypi3:~/t_ultibo $ 
```

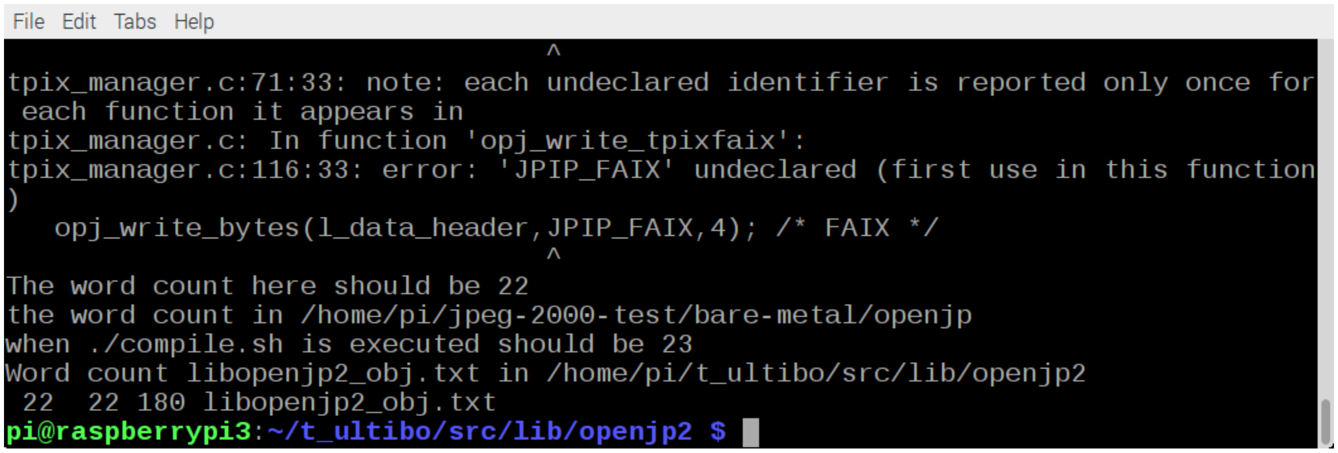
Copying the needed header files and starting the build of libopenjp2.a.



```
Switched to a new branch 'ultibo'
pi@raspberrypi3:~/t_ultibo $ cd src/lib/openjp2
pi@raspberrypi3:~/t_ultibo/src/lib/openjp2 $ cp ../../../*.h .
pi@raspberrypi3:~/t_ultibo/src/lib/openjp2 $ ./compile_ultibo.sh
cidx_manager.c: In function 'opj_write_cidx':
cidx_manager.c:76:35: error: 'JPIP_CIDX' undeclared (first use in this function)
     opj_write_bytes(l_data_header,JPIP_CIDX,4); /* CIDX */
                                   ^
cidx_manager.c:76:35: note: each undeclared identifier is reported only once for
 each function it appears in
cidx_manager.c:85:25: error: 'JPIP_MHIX' undeclared (first use in this function)
     box[num_box].type = JPIP_MHIX;
                         ^
cidx_manager.c:89:25: error: 'JPIP_TPIX' undeclared (first use in this function)
```

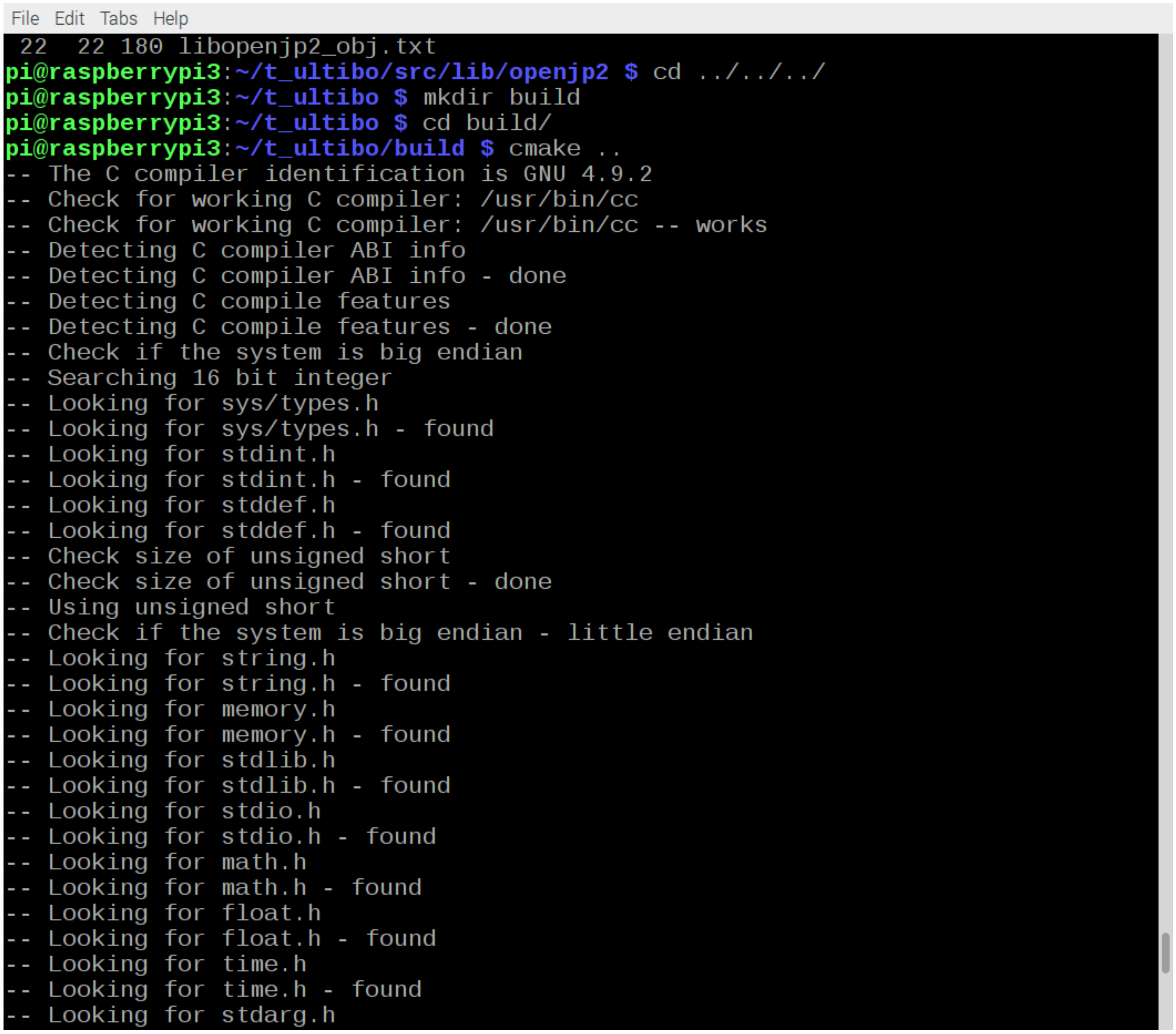
When using Lazarus see pgs 13-15 https://github.com/develone/jpeg-2000-test/blob/master/bare-metal/ultibo-wine.odt.

Library for openjpeg compiled with arm-none-eabi tools is made up of 22 object files.



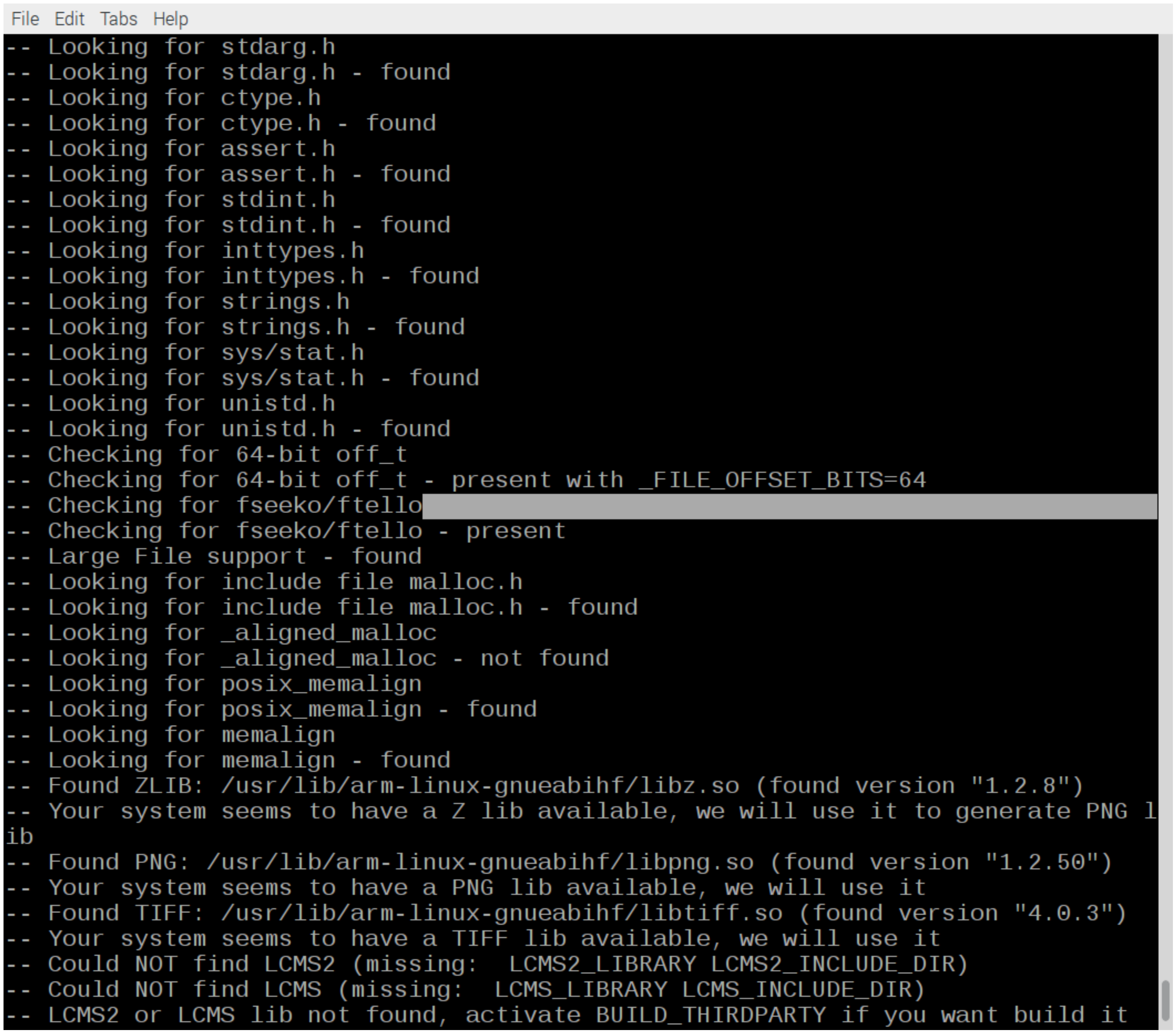
```
                                 ^
tpix_manager.c:71:33: note: each undeclared identifier is reported only once for
 each function it appears in
tpix_manager.c: In function 'opj_write_tpixfaix':
tpix_manager.c:116:33: error: 'JPIP_FAIX' undeclared (first use in this function
)
   opj_write_bytes(l_data_header,JPIP_FAIX,4); /* FAIX */
                                 ^
The word count here should be 22
the word count in /home/pi/jpeg-2000-test/bare-metal/openjp
when ./compile.sh is executed should be 23
Word count libopenjp2_obj.txt in /home/pi/t_ultibo/src/lib/openjp2
 22  22 180 libopenjp2_obj.txt
pi@raspberrypi3:~/t_ultibo/src/lib/openjp2 $ 
```

Building the ***"test_tile_encoder"*** for testing on RaspBian RPi. Using cmake to configure the openjpeg sources.



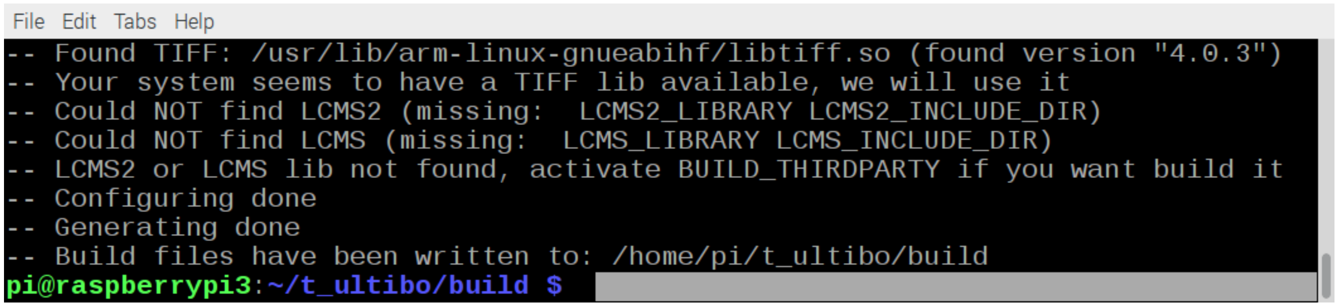
```
 22  22 180 libopenjp2_obj.txt
pi@raspberrypi3:~/t_ultibo/src/lib/openjp2 $ cd ../../../
pi@raspberrypi3:~/t_ultibo $ mkdir build
pi@raspberrypi3:~/t_ultibo $ cd build/
pi@raspberrypi3:~/t_ultibo/build $ cmake ..
-- The C compiler identification is GNU 4.9.2
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check if the system is big endian
-- Searching 16 bit integer
-- Looking for sys/types.h
-- Looking for sys/types.h - found
-- Looking for stdint.h
-- Looking for stdint.h - found
-- Looking for stddef.h
-- Looking for stddef.h - found
-- Check size of unsigned short
-- Check size of unsigned short - done
-- Using unsigned short
-- Check if the system is big endian - little endian
-- Looking for string.h
-- Looking for string.h - found
-- Looking for memory.h
-- Looking for memory.h - found
-- Looking for stdlib.h
-- Looking for stdlib.h - found
-- Looking for stdio.h
-- Looking for stdio.h - found
-- Looking for math.h
-- Looking for math.h - found
-- Looking for float.h
-- Looking for float.h - found
-- Looking for time.h
-- Looking for time.h - found
-- Looking for stdarg.h
```

Building the ***"test_tile_encoder"*** for testing on RaspBian RPi continued. Using cmake to configure the openjpeg sources.



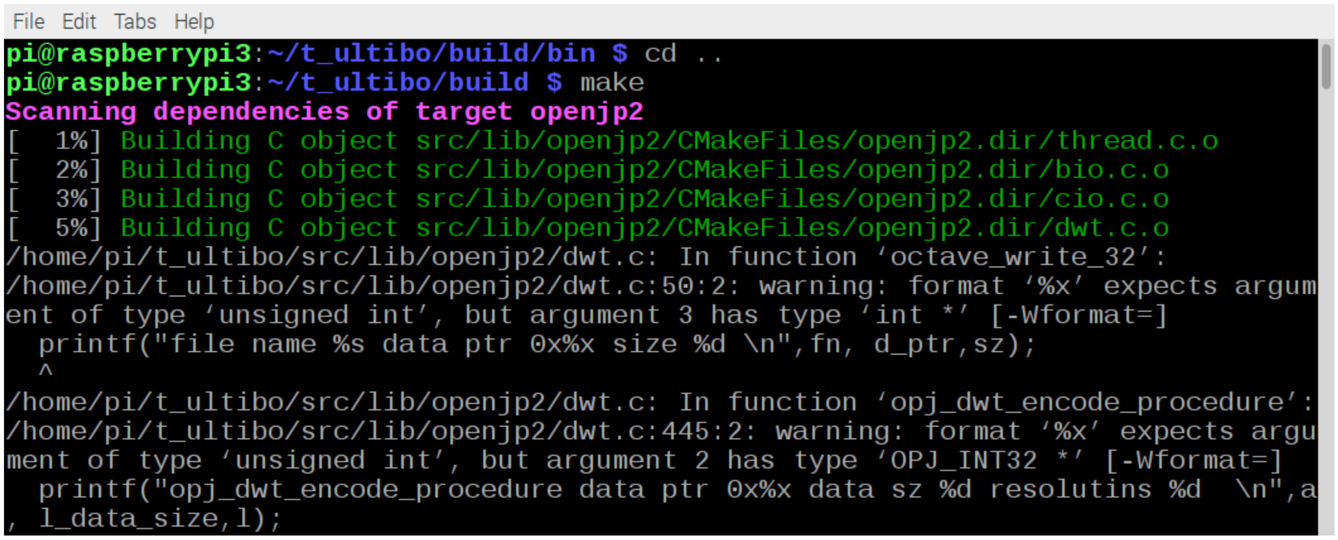
```
-- Looking for stdarg.h
-- Looking for stdarg.h - found
-- Looking for ctype.h
-- Looking for ctype.h - found
-- Looking for assert.h
-- Looking for assert.h - found
-- Looking for stdint.h
-- Looking for stdint.h - found
-- Looking for inttypes.h
-- Looking for inttypes.h - found
-- Looking for strings.h
-- Looking for strings.h - found
-- Looking for sys/stat.h
-- Looking for sys/stat.h - found
-- Looking for unistd.h
-- Looking for unistd.h - found
-- Checking for 64-bit off_t
-- Checking for 64-bit off_t - present with _FILE_OFFSET_BITS=64
-- Checking for fseeko/ftello
-- Checking for fseeko/ftello - present
-- Large File support - found
-- Looking for include file malloc.h
-- Looking for include file malloc.h - found
-- Looking for _aligned_malloc
-- Looking for _aligned_malloc - not found
-- Looking for posix_memalign
-- Looking for posix_memalign - found
-- Looking for memalign
-- Looking for memalign - found
-- Found ZLIB: /usr/lib/arm-linux-gnueabihf/libz.so (found version "1.2.8")
-- Your system seems to have a Z lib available, we will use it to generate PNG l
ib
-- Found PNG: /usr/lib/arm-linux-gnueabihf/libpng.so (found version "1.2.50")
-- Your system seems to have a PNG lib available, we will use it
-- Found TIFF: /usr/lib/arm-linux-gnueabihf/libtiff.so (found version "4.0.3")
-- Your system seems to have a TIFF lib available, we will use it
-- Could NOT find LCMS2 (missing:  LCMS2_LIBRARY LCMS2_INCLUDE_DIR)
-- Could NOT find LCMS (missing:  LCMS_LIBRARY LCMS_INCLUDE_DIR)
-- LCMS2 or LCMS lib not found, activate BUILD_THIRDPARTY if you want build it
```

Building the ***"test_tile_encoder"*** for testing on RaspBian RPi. Using cmake to configure the openjpeg sources.



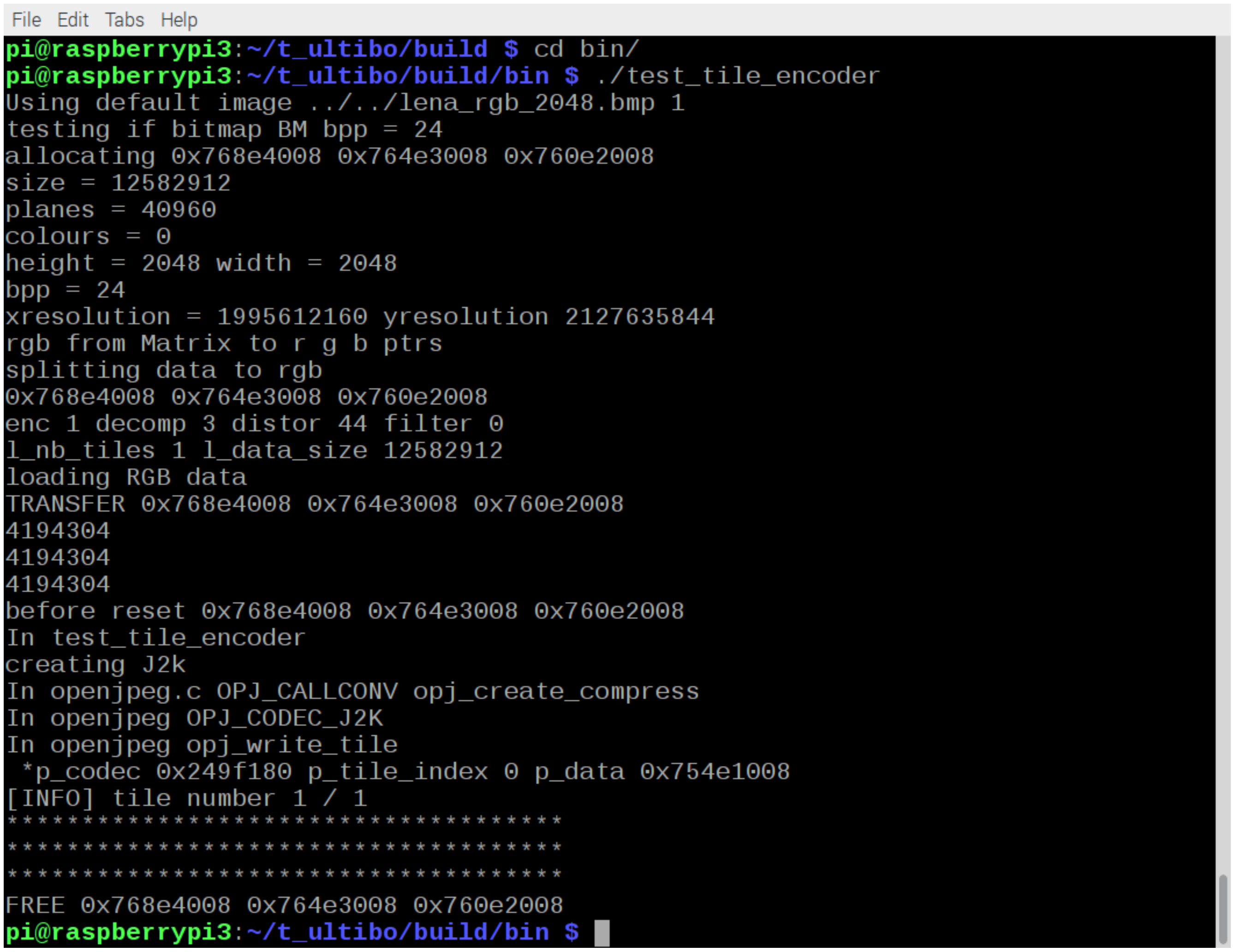
```
-- Found TIFF: /usr/lib/arm-linux-gnueabihf/libtiff.so (found version "4.0.3")
-- Your system seems to have a TIFF lib available, we will use it
-- Could NOT find LCMS2 (missing:  LCMS2_LIBRARY LCMS2_INCLUDE_DIR)
-- Could NOT find LCMS (missing:  LCMS_LIBRARY LCMS_INCLUDE_DIR)
-- LCMS2 or LCMS lib not found, activate BUILD_THIRDPARTY if you want build it
-- Configuring done
-- Generating done
-- Build files have been written to: /home/pi/t_ultibo/build
pi@raspberrypi3:~/t_ultibo/build $
```

Compiling the programs ***"test_tile_encoder", "test_tile_decoder", "opj_compress", " opj_decompress", and "opj_dump"*** for testing on RaspBian RPi.



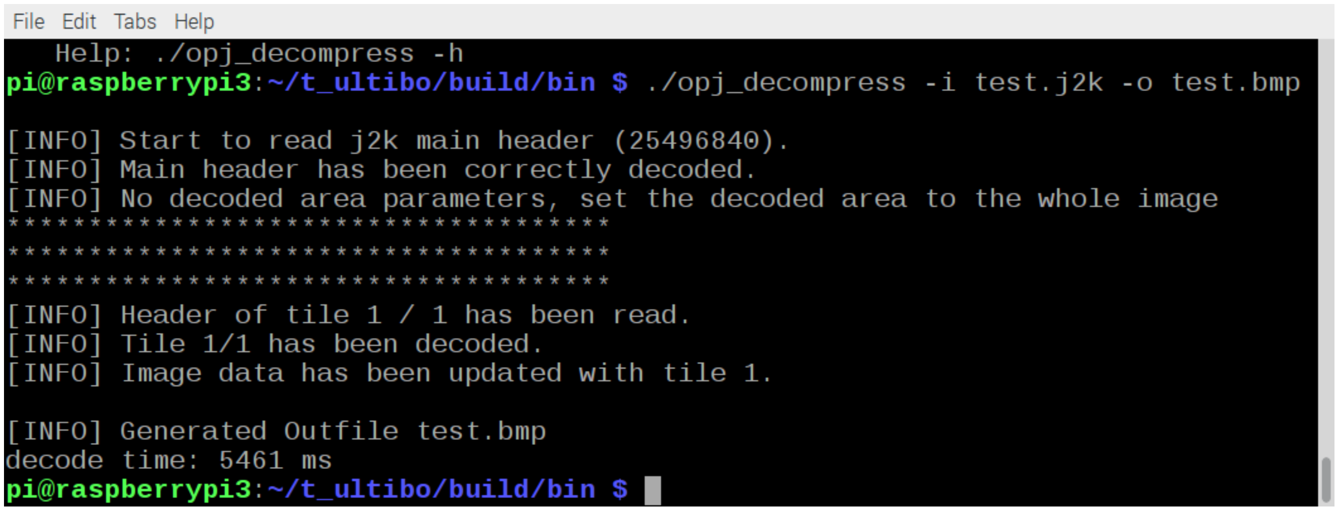
```
pi@raspberrypi3:~/t_ultibo/build/bin $ cd ..
pi@raspberrypi3:~/t_ultibo/build $ make
Scanning dependencies of target openjp2
[  1%] Building C object src/lib/openjp2/CMakeFiles/openjp2.dir/thread.c.o
[  2%] Building C object src/lib/openjp2/CMakeFiles/openjp2.dir/bio.c.o
[  3%] Building C object src/lib/openjp2/CMakeFiles/openjp2.dir/cio.c.o
[  5%] Building C object src/lib/openjp2/CMakeFiles/openjp2.dir/dwt.c.o
/home/pi/t_ultibo/src/lib/openjp2/dwt.c: In function ‘octave_write_32’:
/home/pi/t_ultibo/src/lib/openjp2/dwt.c:50:2: warning: format ‘%x’ expects argum
ent of type ‘unsigned int’, but argument 3 has type ‘int *’ [-Wformat=]
  printf("file name %s data ptr 0x%x size %d \n",fn, d_ptr,sz);
  ^
/home/pi/t_ultibo/src/lib/openjp2/dwt.c: In function ‘opj_dwt_encode_procedure’:
/home/pi/t_ultibo/src/lib/openjp2/dwt.c:445:2: warning: format ‘%x’ expects argu
ment of type ‘unsigned int’, but argument 2 has type ‘OPJ_INT32 *’ [-Wformat=]
  printf("opj_dwt_encode_procedure data ptr 0x%x data sz %d resolutins %d  \n",a
, l_data_size,l);
```

Running test_tile_encoder on RaspBian RPi.



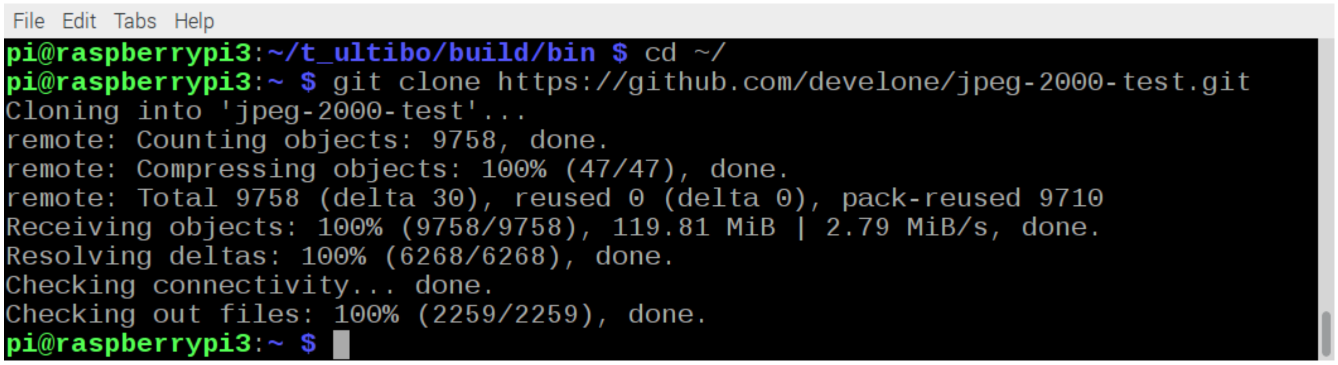
```
pi@raspberrypi3:~/t_ultibo/build $ cd bin/
pi@raspberrypi3:~/t_ultibo/build/bin $ ./test_tile_encoder
Using default image ../../lena_rgb_2048.bmp 1
testing if bitmap BM bpp = 24
allocating 0x768e4008 0x764e3008 0x760e2008
size = 12582912
planes = 40960
colours = 0
height = 2048 width = 2048
bpp = 24
xresolution = 1995612160 yresolution 2127635844
rgb from Matrix to r g b ptrs
splitting data to rgb
0x768e4008 0x764e3008 0x760e2008
enc 1 decomp 3 distor 44 filter 0
l_nb_tiles 1 l_data_size 12582912
loading RGB data
TRANSFER 0x768e4008 0x764e3008 0x760e2008
4194304
4194304
4194304
before reset 0x768e4008 0x764e3008 0x760e2008
In test_tile_encoder
creating J2k
In openjpeg.c OPJ_CALLCONV opj_create_compress
In openjpeg OPJ_CODEC_J2K
In openjpeg opj_write_tile
 *p_codec 0x249f180 p_tile_index 0 p_data 0x754e1008
[INFO] tile number 1 / 1
*************************************
*************************************
*************************************
FREE 0x768e4008 0x764e3008 0x760e2008
pi@raspberrypi3:~/t_ultibo/build/bin $
```

Decompressing the file ***“test.j2k”*** created with ***“test_tile_encoder”.***



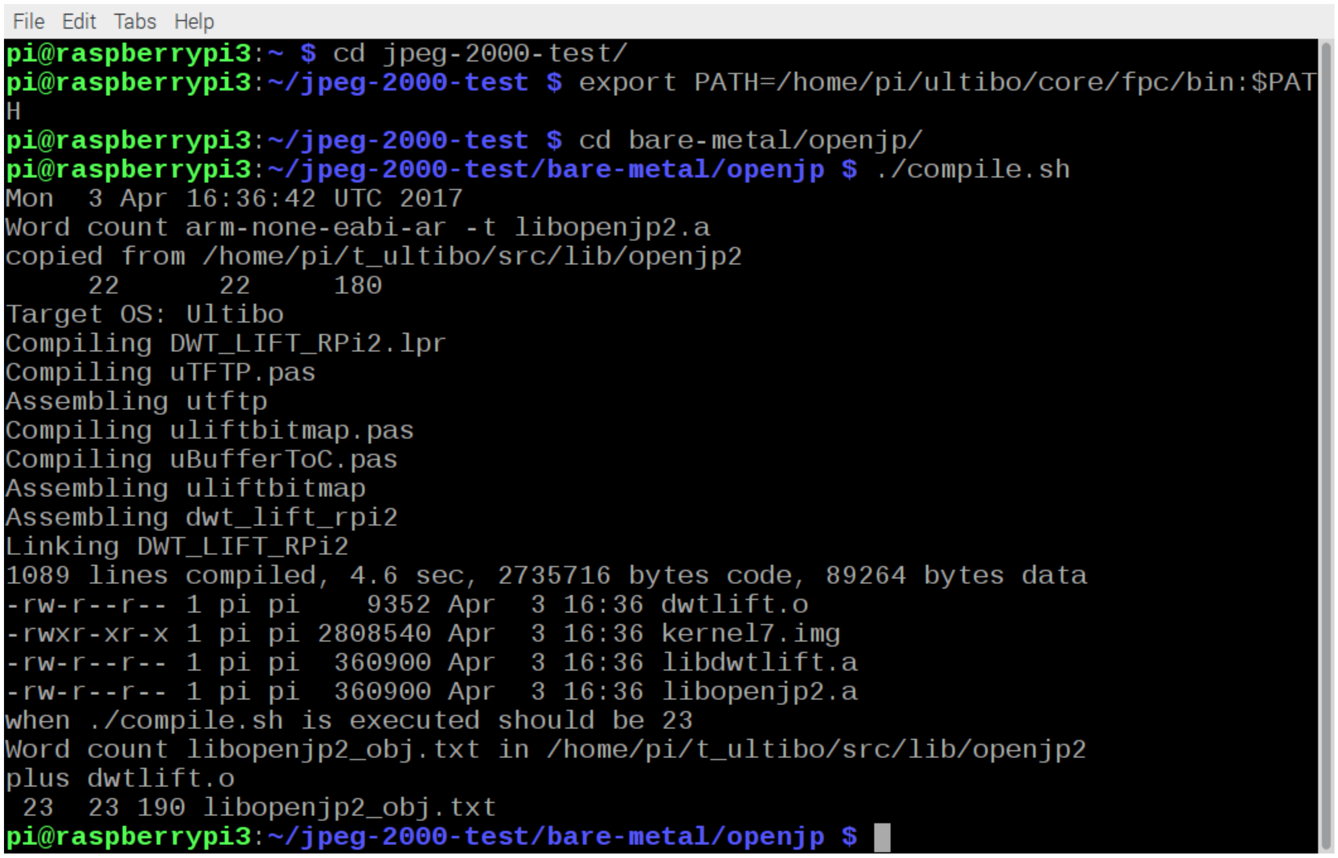
```
   Help: ./opj_decompress -h
pi@raspberrypi3:~/t_ultibo/build/bin $ ./opj_decompress -i test.j2k -o test.bmp

[INFO] Start to read j2k main header (25496840).
[INFO] Main header has been correctly decoded.
[INFO] No decoded area parameters, set the decoded area to the whole image
*************************************
*************************************
*************************************
[INFO] Header of tile 1 / 1 has been read.
[INFO] Tile 1/1 has been decoded.
[INFO] Image data has been updated with tile 1.

[INFO] Generated Outfile test.bmp
decode time: 5461 ms
pi@raspberrypi3:~/t_ultibo/build/bin $ 
```

The next steps in testing of libopenjp2.a is to create a kernel7.img and boot a RPi
Cloneing the jpeg-2000-test repository,



```
pi@raspberrypi3:~/t_ultibo/build/bin $ cd ~/
pi@raspberrypi3:~ $ git clone https://github.com/develone/jpeg-2000-test.git
Cloning into 'jpeg-2000-test'...
remote: Counting objects: 9758, done.
remote: Compressing objects: 100% (47/47), done.
remote: Total 9758 (delta 30), reused 0 (delta 0), pack-reused 9710
Receiving objects: 100% (9758/9758), 119.81 MiB | 2.79 MiB/s, done.
Resolving deltas: 100% (6268/6268), done.
Checking connectivity... done.
Checking out files: 100% (2259/2259), done.
pi@raspberrypi3:~ $ 
```

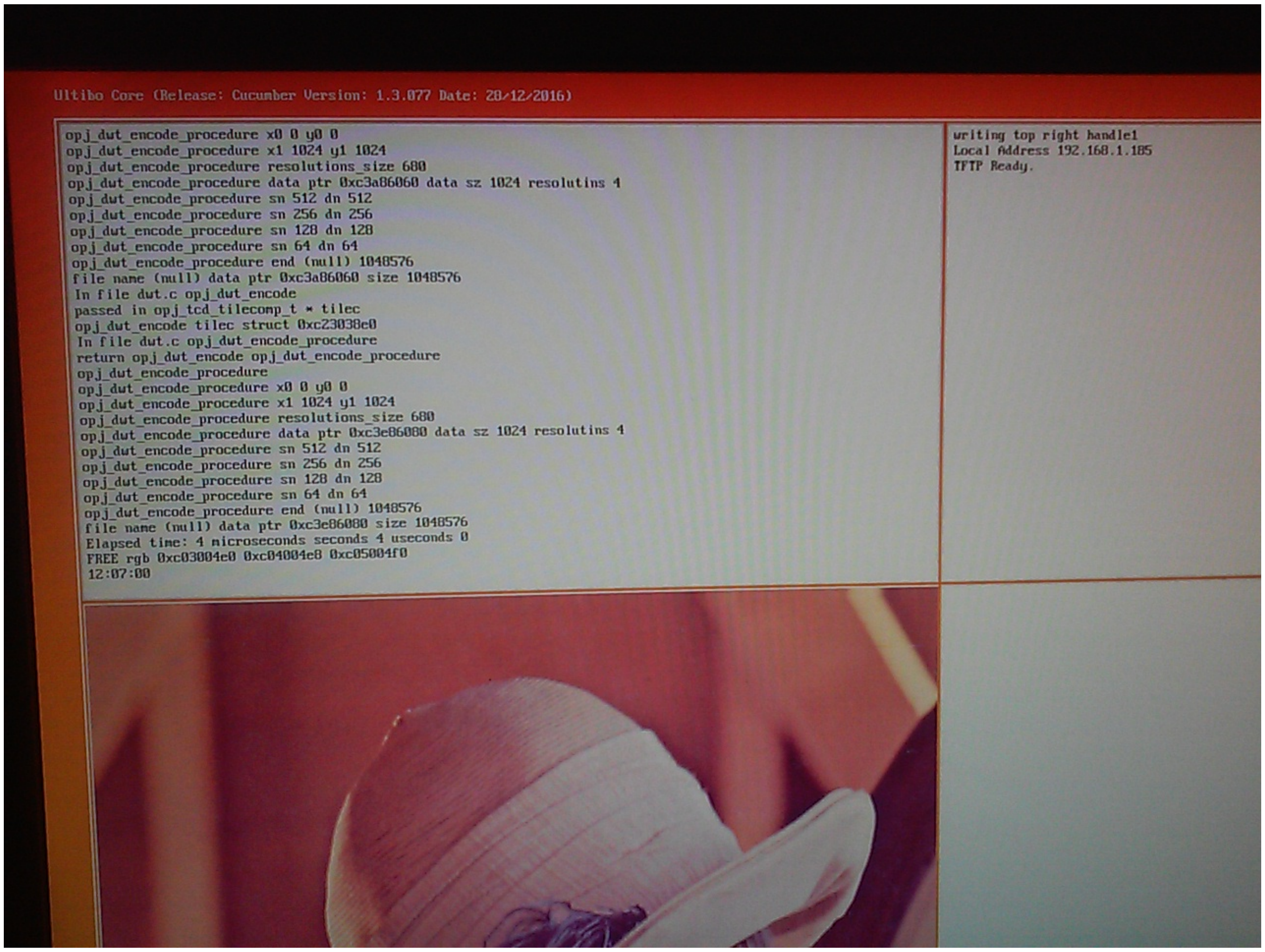
Set the PATH eviornment variable to use FPC and create the ***“kernel7.img”*** consisting of the ***“libopenjp2.a”*** and the ***“dwtlift.o”.***



```
pi@raspberrypi3:~ $ cd jpeg-2000-test/
pi@raspberrypi3:~/jpeg-2000-test $ export PATH=/home/pi/ultibo/core/fpc/bin:$PAT
H
pi@raspberrypi3:~/jpeg-2000-test $ cd bare-metal/openjp/
pi@raspberrypi3:~/jpeg-2000-test/bare-metal/openjp $ ./compile.sh
Mon  3 Apr 16:36:42 UTC 2017
Word count arm-none-eabi-ar -t libopenjp2.a
copied from /home/pi/t_ultibo/src/lib/openjp2
     22      22     180
Target OS: Ultibo
Compiling DWT_LIFT_RPi2.lpr
Compiling uTFTP.pas
Assembling utftp
Compiling uliftbitmap.pas
Compiling uBufferToC.pas
Assembling uliftbitmap
Assembling dwt_lift_rpi2
Linking DWT_LIFT_RPi2
1089 lines compiled, 4.6 sec, 2735716 bytes code, 89264 bytes data
-rw-r--r-- 1 pi pi    9352 Apr  3 16:36 dwtlift.o
-rwxr-xr-x 1 pi pi 2808540 Apr  3 16:36 kernel7.img
-rw-r--r-- 1 pi pi  360900 Apr  3 16:36 libdwtlift.a
-rw-r--r-- 1 pi pi  360900 Apr  3 16:36 libopenjp2.a
when ./compile.sh is executed should be 23
Word count libopenjp2_obj.txt in /home/pi/t_ultibo/src/lib/openjp2
plus dwtlift.o
 23  23 190 libopenjp2_obj.txt
pi@raspberrypi3:~/jpeg-2000-test/bare-metal/openjp $ 
```

The next step will be transfer the files sd-card/
"bootcode.bin", "fixup.dat", "kernel7.img", "MyBitmap.bmp", and "start.elf" and boot the RPi

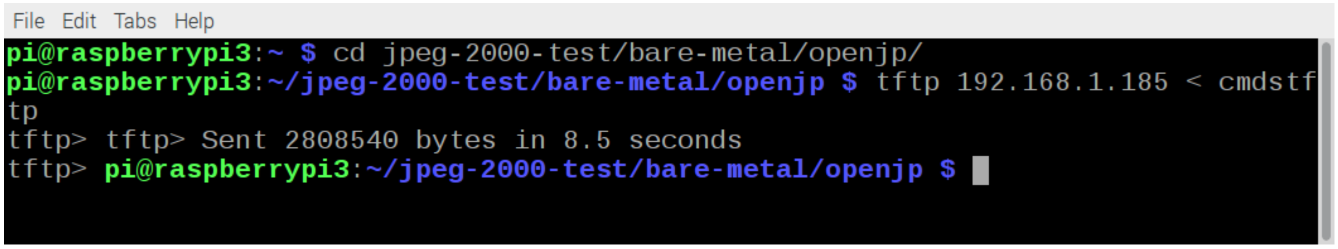
Runnng on Ultibo



```
Ultibo Core (Release: Cucumber Version: 1.3.077 Date: 28/12/2016)
opj_dwt_encode_procedure x0 0 y0 0
opj_dwt_encode_procedure x1 1024 y1 1024
opj_dwt_encode_procedure resolutions_size 680
opj_dwt_encode_procedure data ptr 0xc3a86060 data sz 1024 resolutins 4
opj_dwt_encode_procedure sn 512 dn 512
opj_dwt_encode_procedure sn 256 dn 256
opj_dwt_encode_procedure sn 128 dn 128
opj_dwt_encode_procedure sn 64 dn 64
opj_dwt_encode_procedure end (null) 1048576
file name (null) data ptr 0xc3a86060 size 1048576
In file dwt.c opj_dwt_encode
passed in opj_tcd_tilecomp_t * tilec
opj_dwt_encode tilec struct 0xc23038e0
In file dwt.c opj_dwt_encode_procedure
return opj_dwt_encode opj_dwt_encode_procedure
opj_dwt_encode_procedure
opj_dwt_encode_procedure x0 0 y0 0
opj_dwt_encode_procedure x1 1024 y1 1024
opj_dwt_encode_procedure resolutions_size 680
opj_dwt_encode_procedure data ptr 0xc3e86080 data sz 1024 resolutins 4
opj_dwt_encode_procedure sn 512 dn 512
opj_dwt_encode_procedure sn 256 dn 256
opj_dwt_encode_procedure sn 128 dn 128
opj_dwt_encode_procedure sn 64 dn 64
opj_dwt_encode_procedure end (null) 1048576
file name (null) data ptr 0xc3e86080 size 1048576
Elapsed time: 4 microseconds seconds 4 useconds 0
FREE rgb 0xc03004e0 0xc04004e8 0xc05004f0
12:07:00
```

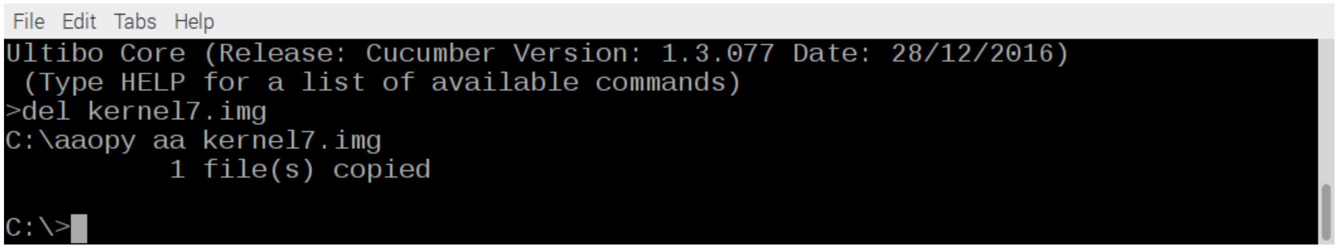
```
writing top right handle1
Local Address 192.168.1.185
TFTP Ready.
```

Now the RPi running Ultibo provides for tftp and telnet.



```
pi@raspberrypi3:~ $ cd jpeg-2000-test/bare-metal/openjp/
pi@raspberrypi3:~/jpeg-2000-test/bare-metal/openjp $ tftp 192.168.1.185 < cmdstf
tp
tftp> tftp> Sent 2808540 bytes in 8.5 seconds
tftp> pi@raspberrypi3:~/jpeg-2000-test/bare-metal/openjp $ 
```

Using a telnet xx.xx.xx.xx. Replace the ***"kernel7.img"*** by first deleting the ***"kernel7.Img" a***nd replacing with the file ***"aa"*** transferred in the step above.



```
Ultibo Core (Release: Cucumber Version: 1.3.077 Date: 28/12/2016)
 (Type HELP for a list of available commands)
>del kernel7.img
C:\aaopy aa kernel7.img
          1 file(s) copied

C:\>
```

line 489 l_param.cp_fixed_quality = 1; to l_param.cp_fixed_quality = 0;
line 489 comment /*l_param.tcp_distoratio[0] = TCP_DISTORATIO;*/
add these which were commented out.
l_param.cp_disto_alloc = 1;
l_param.tcp_rates[0] = 10; or 20 30 40 50 100

```
-rw-r--r-- 1 pi pi 12583034 Apr  3 14:33 ../../lena_rgb_2048.bmp

-rw-r--r-- 1 pi pi 125603 Apr 11 00:21 test.j2k
```

l_param.tcp_rates[0] = 10;

```
-rw-r--r-- 1 pi pi 1257924 Apr 11 00:27 test.j2k
```

l_param.tcp_rates[0] = 20;

```
-rw-r--r-- 1 pi pi 628968 Apr 11 00:34 test.j2k
```

l_param.tcp_rates[0] = 30;

```
-rw-r--r-- 1 pi pi 419384 Apr 11 00:38 test.j2k
```

l_param.tcp_rates[0] = 50;

```
-rw-r--r-- 1 pi pi 251619 Apr 11 00:29 test.j2k
```

l_param.tcp_rates[0] = 100;

```
-rw-r--r-- 1 pi pi 125603 Apr 11 00:21 test.j2k
```

line 489 l_param.cp_fixed_quality = 1; to l_param.cp_fixed_quality = 0;
line 489 comment /*l_param.tcp_distoratio[0] = TCP_DISTORATIO;*/
add these which were commented out.
l_param.cp_disto_alloc = 1;
l_param.tcp_rates[0] = 10; or 20 30 40 50 100

```
-rw-r--r-- 1 pi pi 3145850 Apr  3 16:30 /home/pi/jpeg-2000-test/bare-metal/LibC/lena_rgb_1024.bmp
```

l_param.tcp_rates[0] = 10;

```
-rw-r--r-- 1 pi pi 314281 Apr 11 11:17 test.j2k
```

l_param.tcp_rates[0] = 20;

```
-rw-r--r-- 1 pi pi 157149 Apr 11 11:15 test.j2k
```

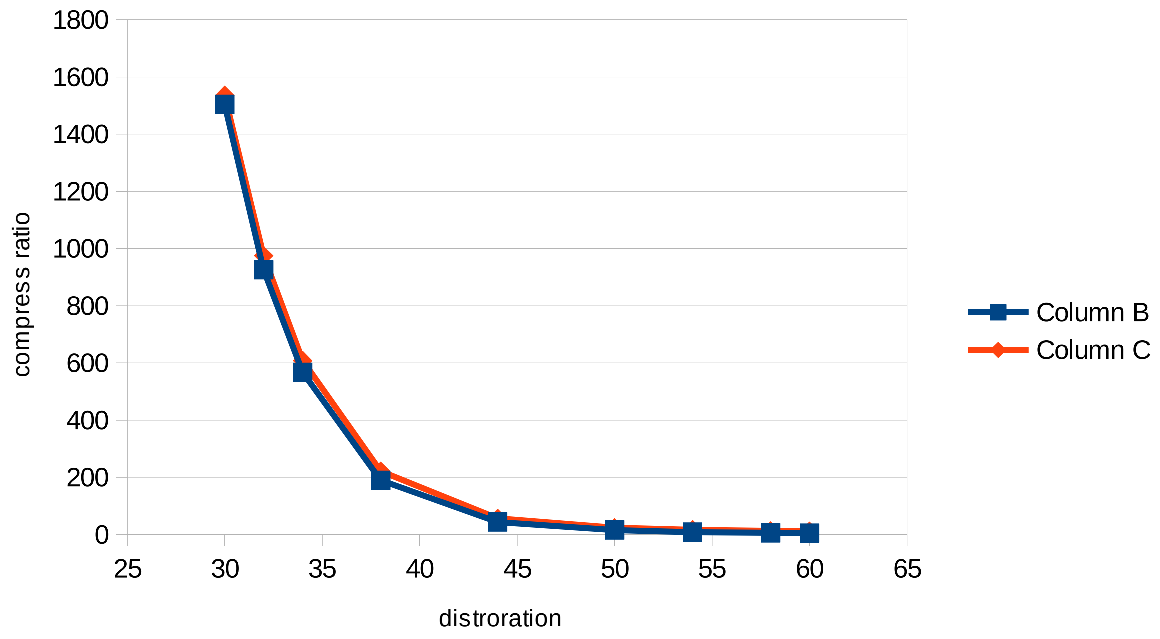
l_param.tcp_rates[0] = 30;

```
-rw-r--r-- 1 pi pi 104852 Apr 11 11:13 test.j2k
l_param.tcp_rates[0] = 50;
-rw-r--r-- 1 pi pi 62781 Apr 11 11:11 test.j2k
l_param.tcp_rates[0] = 100;
-rw-r--r-- 1 pi pi 31339 Apr 11 11:06 test.j2k
```

Testing varying the

	image file	image sz	“5/3 compressed sz	“9/7 compressed sz
distoration	12583034	2048x2048	Irreversible 0	Irreversible 1
	Irreversible 0	Irreversible 1		
30	1503.708652	1535.452593	8368	8195
32	925.42722659	974.89997676	13597	12907
34	566.82886617	607.72924414	22199	20705
38	189.40938991	221.62983708	66433	56775
44	44.058396563	56.360197258	285599	223261
50	15.937796783	24.016677832	789509	523929
54	8.6079098426	16.856649872	1461799	746473
58	6.0368784486	12.753303353	2084361	986649
60	5.5454244973	11.771187004	2269084	1068969

Compression ratio



openjpeg running on Bare-metal Raspberrypi.
It took 5 sec to compress a 1024 x 1024 RGB bitmap.
With only 5 files It provides openjpeg , telnet and tftp support.

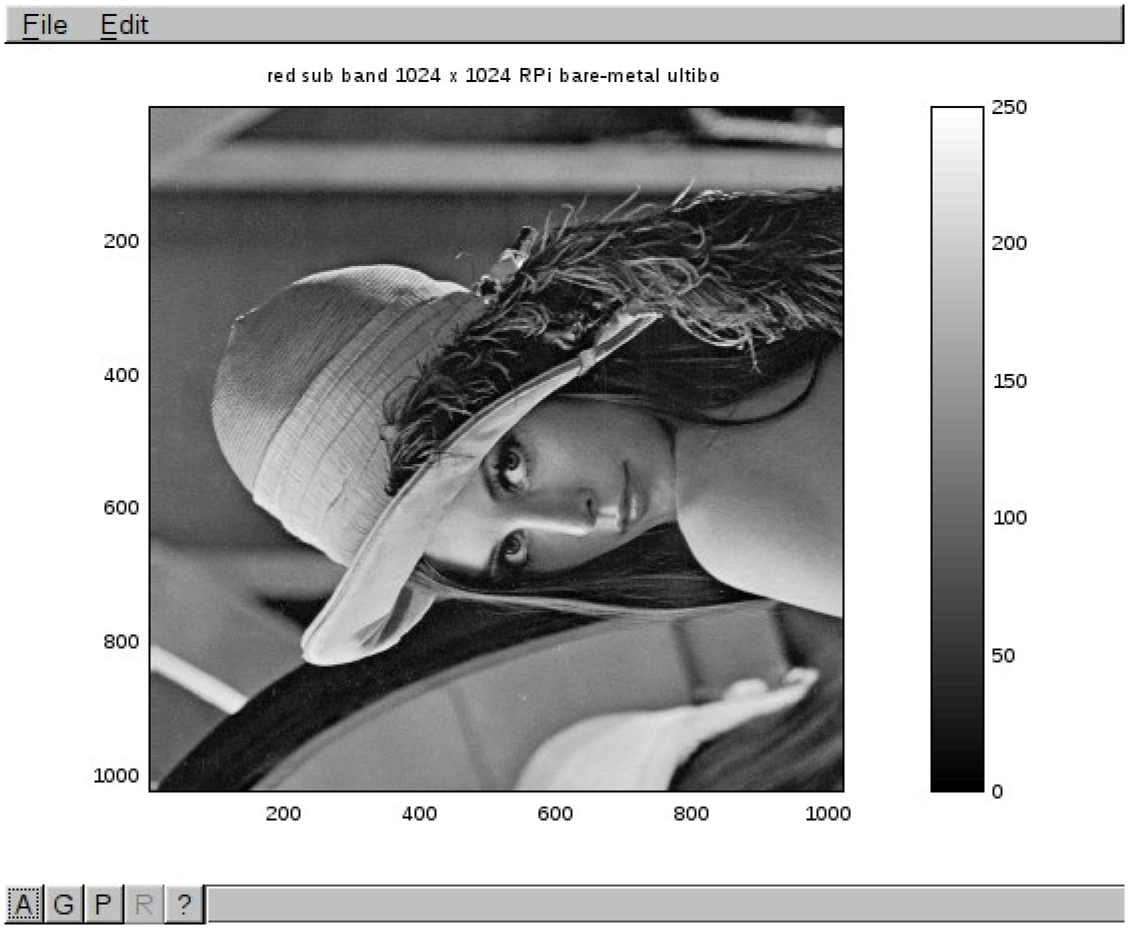
ls -la sd-card/

```
total 8596
drwxr-xr-x 2 pi pi    4096 Mar 22 18:14 .
drwxr-xr-x 3 pi pi    4096 Mar 22 22:43 ..
-rw-r--r-- 1 pi pi  17932 Nov  2 04:14 bootcode.bin
-rw-r--r-- 1 pi pi    6621 Nov  2 04:14 fixup.dat
-rwxr-xr-x 1 pi pi 2796252 Mar 22 18:03 kernel7.img
-rw-r--r-- 1 pi pi 3145850 Feb  7 21:50 MyBitmap.bmp
-rw-r--r-- 1 pi pi 2817796 Nov  2 04:14 start.elf
```

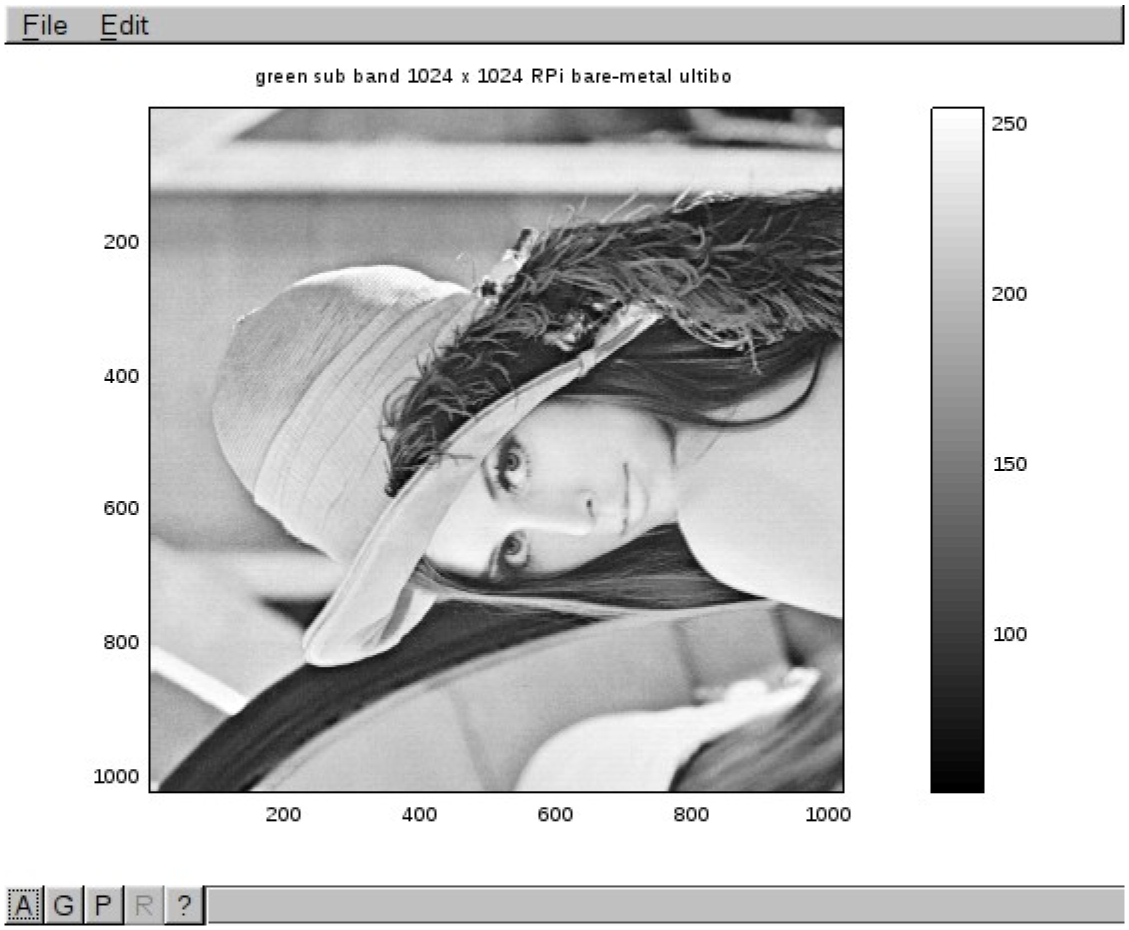
The file MyBitmap.bmp is the lena_rgb_1024.bmp.

This creates the file test.j2k. in jpeg 2000 code stream.

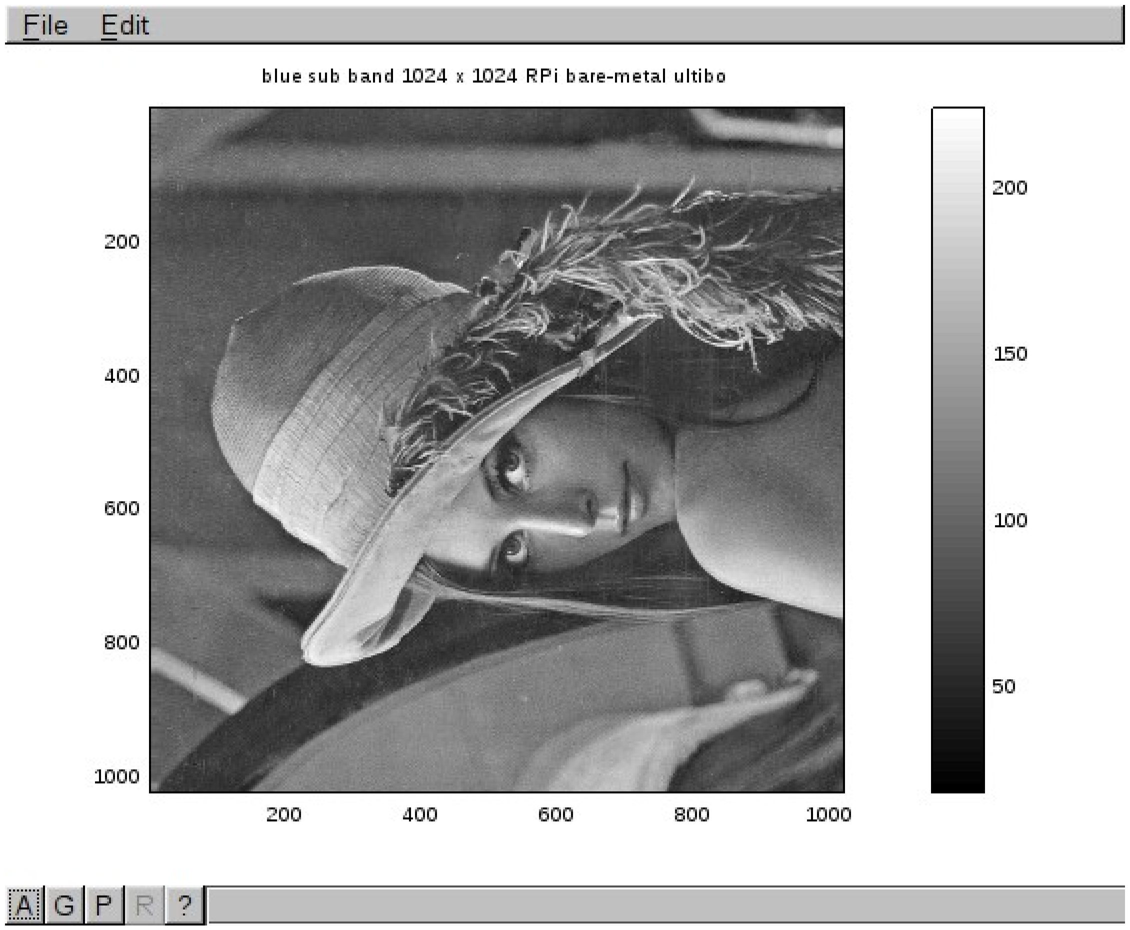
The program reads the bitmap and separates to RGB.
Red subband written as part of reading the bitmap and calling openjpeg



Green subband written as part of reading the bitmap and calling openjpeg



Blue subband written as part of reading the bitmap and calling openjpeg



1024 X 1024 bare-metal RPi3 with armv7-a implemented as RPi2.
4 to 5 sec

1024 x 1024 RPi3B RaspBian
~~real 0m12.830s~~
~~user 0m8.540s~~
~~sys 0m0.550s~~
With openjpeg compiled with -03 default is no optimization

```
real	0m3.774s
user	0m3.260s
sys	0m0.480s
```

1024 x 1024 x86_64 dual core
~~real 0m4.896s~~
~~user 0m3.739s~~
~~sys 0m0.344s~~
With openjpeg compiled with -03 default is no optimization

```
real	0m2.823s
user	0m1.918s
sys	0m0.299s
```

```
git clone https://github.com/develone/openjpeg.git t_ultibo
cd t_ultibo
git checkout ultibo
mkdir build
cd build
cmake ..
make
```

~~*Note: Solution for malloc with align*~~
~~*commented /*#define OPJ_HAVE_POSIX_MEMALIGN bare-metal for ultibo*/*~~
~~*in file opj_config_private.h*~~

07/30/17
Before today, an image greater than 1024 x 1024 was never compressed using Ulitibo and openjpeg.
Addition of POSIX Threads (pthreads) support for standard C library
Latest commits (Ultibo core 1.3.397)
Fix for Raspberry Pi firmware changes (total_mem=1024 and WFE in armstub)
Latest commits (Ultibo core 1.3.411)
compressed an image 2048 x 2048

```
~~ultibo was having issues with~~
~~pi@raspberrypi3:~/jpeg-2000-test/bare-metal/openjp $ ./compile.sh~~
~~dwtlift.c:(.text+0x694): undefined reference to `lifting'~~
~~libdwtlift.a(opj_malloc.o): In function `opj_aligned_malloc':~~
~~opj_malloc.c:(.text+0x40): undefined reference to `posix_memalign'~~
~~libdwtlift.a(opj_malloc.o): In function `opj_aligned_realloc':~~
~~opj_malloc.c:(.text+0x98): undefined reference to `posix_memalign'~~
~~DWT_LIFT_RPi2.lpr(153,98) Error: Error while linking~~
~~DWT_LIFT_RPi2.lpr(153,98) Fatal: There were 1 errors compiling module, stopp~~
~~Fatal: Compilation aborted~~
~~Error: /home/pi/ultibo/core/fpc/bin/ppcarm returned an error exitcode~~
~~ls: cannot access kernel7.img: No such file or directory~~
~~-rw-r--r-- 1 pi pi   6344 Mar 21 18:41 dwtlift.o~~
~~-rw-r--r-- 1 pi pi 330354 Mar 21 18:41 libdwtlift.a~~
~~-rw-r--r-- 1 pi pi 330354 Mar 21 18:41 libopenjp2.a~~

~~00000024 <opj_aligned_malloc>:~~
~~  24:   e2501000        subs    r1, r0, #0~~
~~  28:   012fff1e        bxeq    lr~~
~~  2c:   e3a00010        mov     r0, #16~~
~~  30:   eafffffe        b       0 <memalign>~~

~~00000034 <opj_aligned_realloc>:~~
~~  34:   e92d4070        push    {r4, r5, r6, lr}~~
~~    38:   e2515000        subs    r5, r1, #0~~
~~    3c:   0a000004        beq     54 <opj_aligned_realloc+0x20>~~
~~    40:   ebfffffe        bl      0 <realloc>~~
~~    44:   e310000f        tst     r0, #15~~
      48:   e1a04000        mov     r4, r0
      4c:   1a000002        bne     5c <opj_aligned_realloc+0x28>
      50:   e8bd8070        pop     {r4, r5, r6, pc}

      4c:   1a000002        bne     5c <opj_aligned_realloc+0x28>
      50:   e8bd8070        pop     {r4, r5, r6, pc}
      54:   e1a00005        mov     r0, r5
      58:   e8bd8070        pop     {r4, r5, r6, pc}
      5c:   e3a00010        mov     r0, #16
      60:   e1a01005        mov     r1, r5
      64:   ebfffffe        bl      0 <memalign>
      68:   e2506000        subs    r6, r0, #0
```

```
-- Your system seems to have a Z lib available, we will use it to generate PNG lib
-- Your system seems to have a PNG lib available, we will use it
-- Your system seems to have a TIFF lib available, we will use it
-- Could NOT find LCMS2 (missing:  LCMS2_LIBRARY LCMS2_INCLUDE_DIR)
-- Could NOT find LCMS (missing:  LCMS_LIBRARY LCMS_INCLUDE_DIR)
-- LCMS2 or LCMS lib not found, activate BUILD_THIRDPARTY if you want build it
-- Configuring done
-- Generating done
-- Build files have been written to: /home/pi/t_ultibo/build
```

Note: It appears that the rmoving of memory.h in opj_includes.h does not have an impact on the build on the RPi.

In a new shell build/bin
./test_tile_encoder ../../lena_p b_256.bmp
cp ../../src/lib/openjp2/*.m .
chg line 446 from 3 64 x 64 to 4 32 x 32 or 5 16 x 16 or 6 8 x 8 or
2 128 x 128

cd /home/pi/t_ultibo/src/lib/openjp2
cp ~/t_ultibo/opj_config_private.h .
cp ~/t_ultibo/opj_config.h .
cp ~/t_ultibo/opj_includes.h .

./compile_ultibo.sh
The word count here should be 20
the word count in jpeg-2000-test/bare=metal/dwt
when ./compile.sh is executed should be 21
Word cpunt libopenjp2_obj.txt in src/lib/openjp2
 20 20 163 libopenjp2_obj.txt

Changes to top level CMakeLists.txt to create documentation, and build tests to create static libraries instead of dynamic.

sudo apt-get install doxygen

-option(BUILD_DOC "Build the HTML documentation (with doxygen if available)." O$
+option(BUILD_DOC "Build the HTML documentation (with doxygen if available)." O$

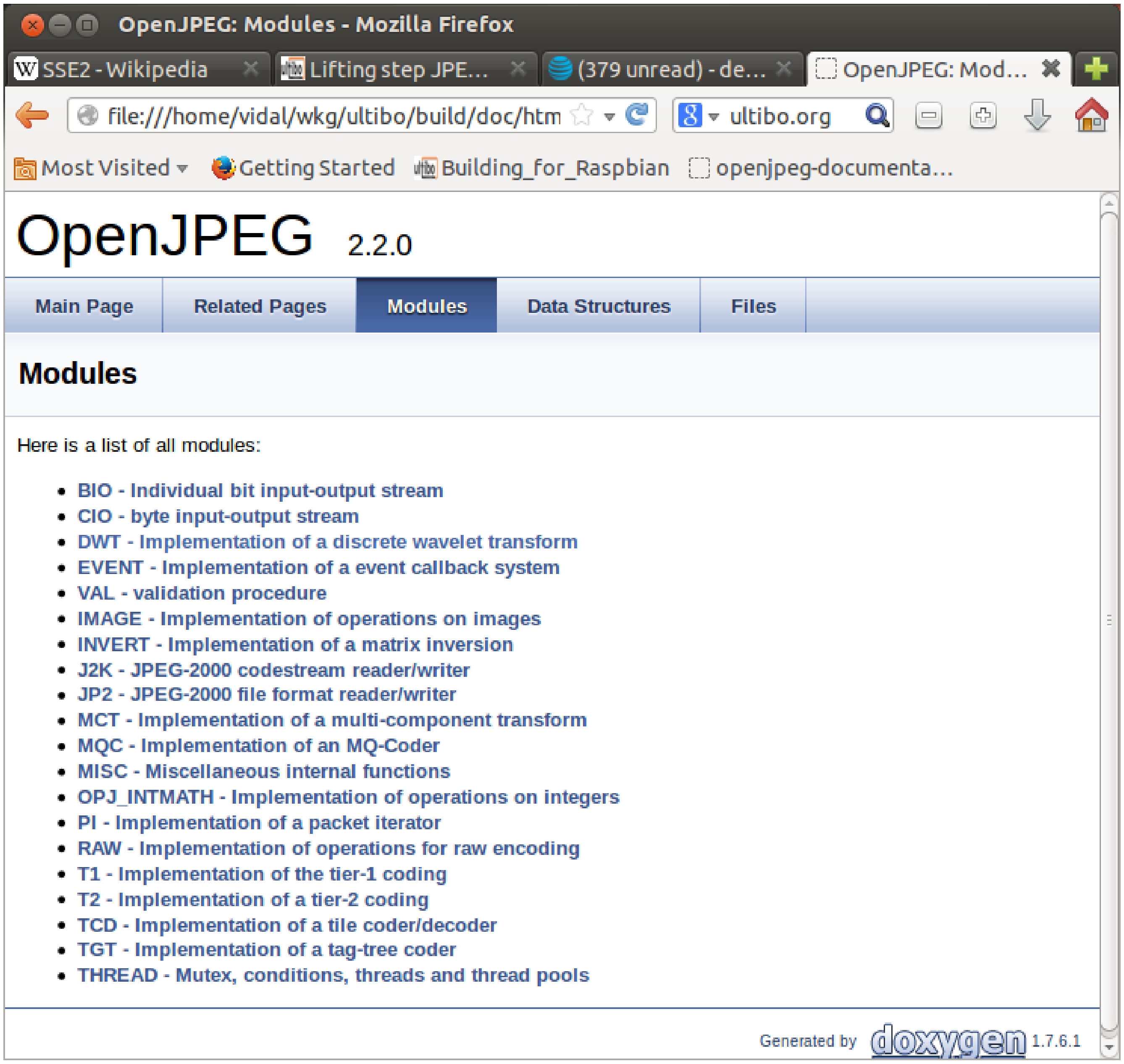
-option(BUILD_SHARED_LIBS "Build OpenJPEG shared library and link executables a$
+option(BUILD_SHARED_LIBS "Build OpenJPEG shared library and link executables a$

-option(BUILD_TESTING "Build the tests." OFF)
+option(BUILD_TESTING "Build the tests." ON)

- # set(CMAKE_C_FLAGS "-Wall -std=c99 ${CMAKE_C_FLAGS}") # FIXME: this setting prevented us from setting a coverage build.
+ # set(CMAKE_C_FLAGS "-g -Wall -std=c99 ${CMAKE_C_FLAGS}") # FIXME: this setting prevented us from setting a coverage build.

wkg/ultibo/build/doc/html

Main documentation page.



OpenJPEG 2.2.0

Modules

Here is a list of all modules:

- BIO - Individual bit input-output stream
- CIO - byte input-output stream
- DWT - Implementation of a discrete wavelet transform
- EVENT - Implementation of a event callback system
- VAL - validation procedure
- IMAGE - Implementation of operations on images
- INVERT - Implementation of a matrix inversion
- J2K - JPEG-2000 codestream reader/writer
- JP2 - JPEG-2000 file format reader/writer
- MCT - Implementation of a multi-component transform
- MQC - Implementation of an MQ-Coder
- MISC - Miscellaneous internal functions
- OPJ_INTMATH - Implementation of operations on integers
- PI - Implementation of a packet iterator
- RAW - Implementation of operations for raw encoding
- T1 - Implementation of the tier-1 coding
- T2 - Implementation of a tier-2 coding
- TCD - Implementation of a tile coder/decoder
- TGT - Implementation of a tag-tree coder
- THREAD - Mutex, conditions, threads and thread pools

Creating a new branch ultibo to the fork of openjpeg

```
git clone git@github.com:develone/openjpeg.git ultibo
cd ultibo/
git branch ultibo master
git checkout ultibo

git clone git@github.com:develone/openjpeg.git
cd openjpeg
git checkout ultibo
mkdir build
cd build/
make
cp ~/wkg/openjpeg_tmp/src/lib/openjp2/mct.c ../src/lib/openjp2
cp ~/wkg/openjpeg_tmp/src/lib/openjp2/dwt.c ../src/lib/openjp2
RPi
git clone https://github.com/develone/openjpeg.git t_ultibo
cd t_ultibo/
git checkout ultibo
mkdir build
cd build
cmake ..
make
cd bin/
./opj_compress -mct 1 -i ../../lena_rgb_256.bmp -o tt.j2k
```

Adding the following lines to opj_mct_encode in the file mct.c writes the YUV compoents of the RGB image.

```
        FILE *ptr_myfile, *ofp;
        ofp = fopen("c0.bin","w");
        fwrite(&c0[0], sizeof(int), len, ofp);
        fclose(ofp);
        ofp = fopen("c1.bin","w");
        fwrite(&c1[0], sizeof(int), len, ofp);
        fclose(ofp);
        ofp = fopen("c2.bin","w");
        fwrite(&c2[0], sizeof(int), len, ofp);
        fclose(ofp);
```

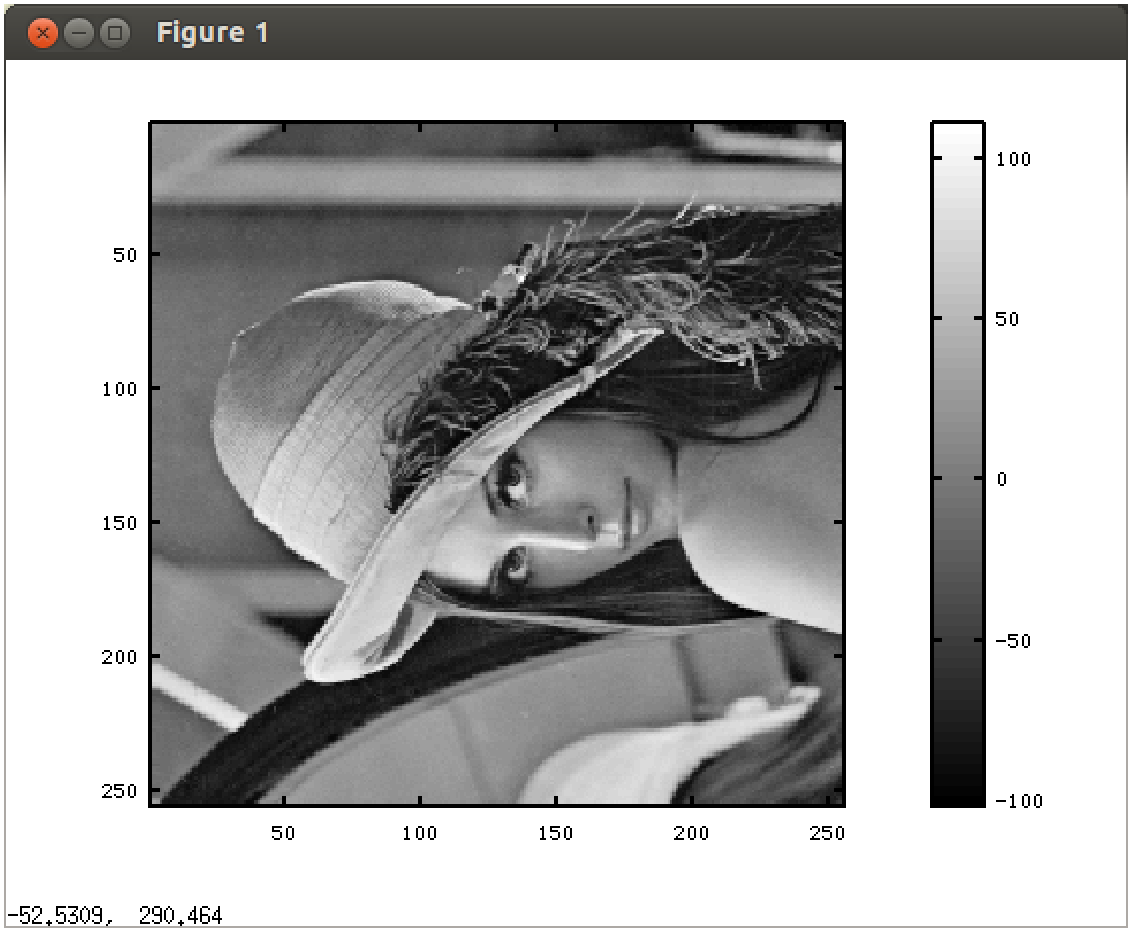
This change was needed in 2 places in the file mct.c. One is lines 116 to 125 and lines 148 to 157. When running on Ubuntu x86_64 and when running on the RPi which does not have support for __SSE2__

https://en.wikipedia.org/wiki/SSE2

SSE2 (**Streaming SIMD Extensions 2**), is one of the Intel SIMD (Single Instruction, Multiple Data) processor supplementary instruction sets first introduced by Intel with the initial version of the Pentium 4 in 2001. It extends the earlier SSE instruction set, and is intended to fully replace MMX. Intel extended SSE2 to create SSE3 in 2004. SSE2 added 144 new instructions to SSE, which has 70 instructions. Competing chip-maker AMD added support for SSE2 with the introduction of their Opteron and Athlon 64 ranges of AMD64 64-bit CPUs in 2003.

Following the loop that computes the YUV values.

```
        for(; i < len; ++i) {
                OPJ_INT32 r = c0[i];
                OPJ_INT32 g = c1[i];
                OPJ_INT32 b = c2[i];
                OPJ_INT32 y = (r + (g * 2) + b) >> 2;
                OPJ_INT32 u = b - g;
                OPJ_INT32 v = r - g;
                c0[i] = y;
                c1[i] = u;
                c2[i] = v;
        }
./opj_compress -mct 1 -i ../../lena_rgb_256.bmp -o tt.j2k

[INFO] tile number 1 / 1
mct.c converts rgb yuv
mct.c 65536
0x21a00e0 0x21a0440 0x219ef40
compno 0
opj_dwt_encode 0x21a0440
0x21a0440 0x219ef40
0x21a0480 0x219f378
compno 1
opj_dwt_encode 0x21a0480
0x21a0480 0x219f378
0x21a04c0 0x219f7b0
compno 2
opj_dwt_encode 0x21a04c0
0x21a04c0 0x219f7b0
0x21a0500 0x219fbe8
[INFO] Generated outfile tt.j2k
encode time: 369 ms
C0
```



C1

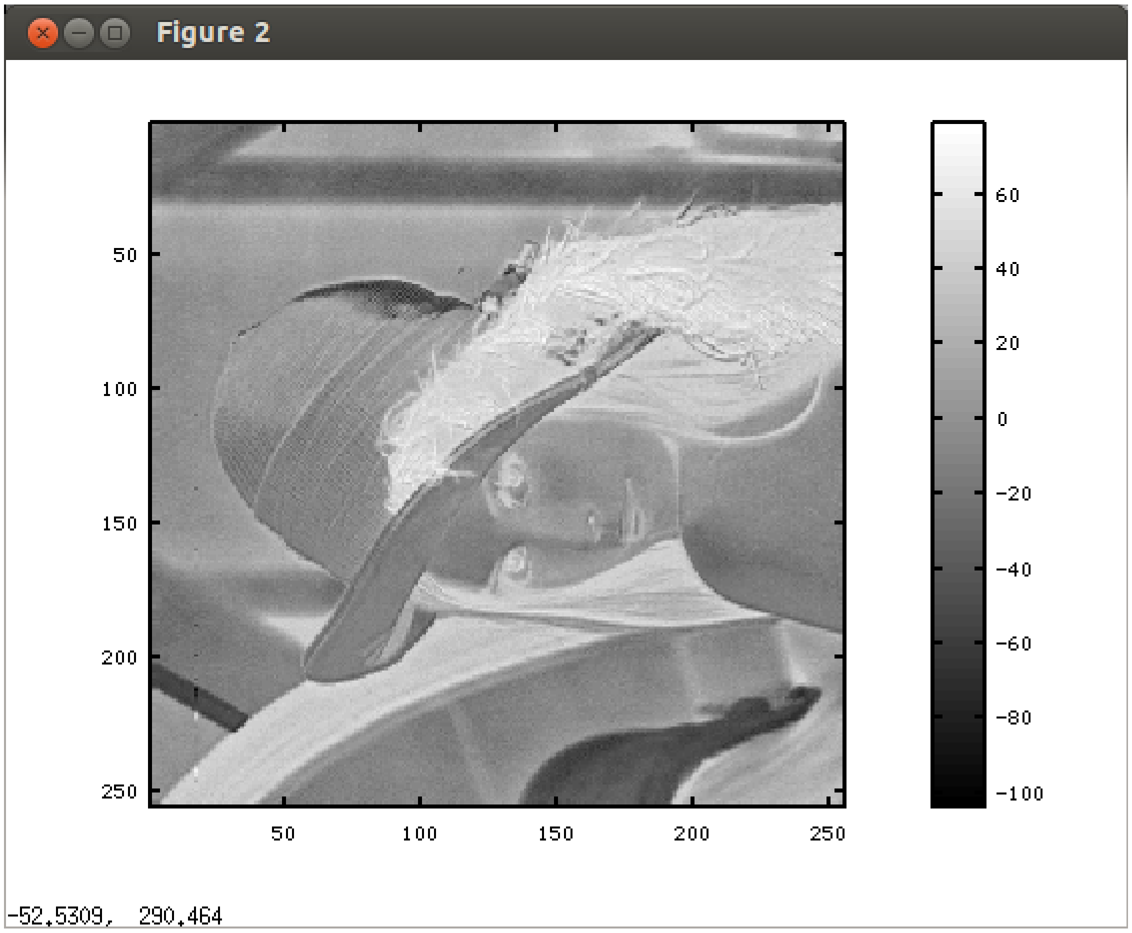
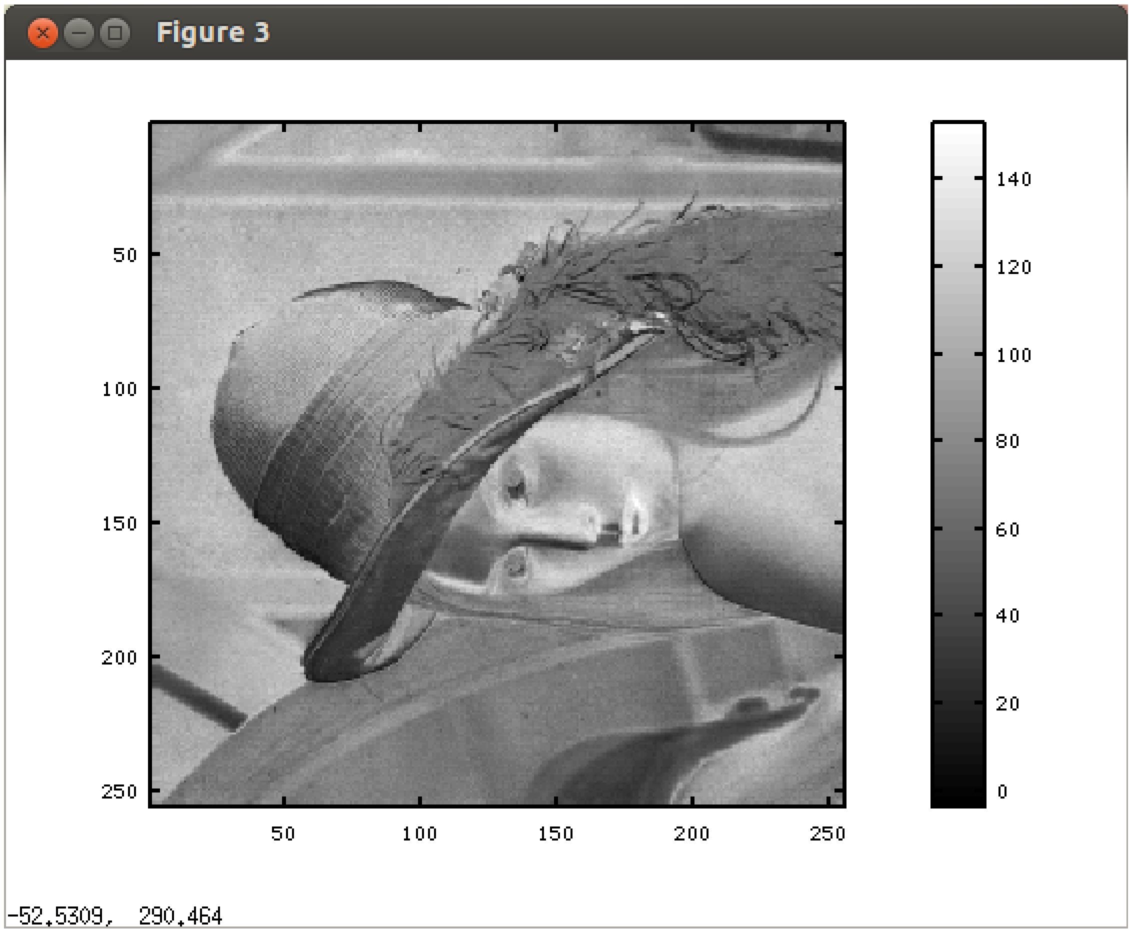


Figure 2
50
100
150
200
250
50
100
150
200
250
60
40
20
0
-20
-40
-60
-80
-100
-52.5309, 290.464

C2



http://www.instesre.org/howto/BW_image/ReadingBitmaps.htm

This document last modified on 01/12/2015 15:19:08.

How to read bitmap (.bmp) files

David R. Brooks

The question of how to manipulate bitmap files came up when my daughter was working on a science fair project involving taking a photo with a digital camera pointed up through a tree canopy. What she needed to know was what percentage of the image was sky and what percentage was tree canopy -- leaves or branches. I'm sure there are photo editing programs that will answer this question, but it seemed as though it shouldn't be too difficult to write my own program specifically for this task. Basically, the plan is to manipulate the colors in the image, perhaps even by creating a two-color black and white image, and then determine how much is clear sky. I often use the freeware IrfanView photo editing software, which allows you to do these kinds of things. By adjusting contrast and brightness of the original image, you can exercise some control over how the software divides the image into "sky" and "not sky."

The first step is to find out how `.bmp` files are organized. An online search will produce many sources of information. Bitmap files are separated into three or four sections, as shown in the table below.

Section	Description
Header	Basic file information, 14 bytes
Image Information Header	Information about the image, 40 bytes
Color Information (optional)	Information about how the image encodes colors, a variable number of bytes if it's present
Image data	The actual image, a variable number of bytes

The next step is to try to interpret a real `.bmp file`. I first created a small file by selecting a small part of my desktop that contains an icon for QuickC, an old Microsoft MS-DOS C programming environment that I have used for many years. Here's the image:

Although this sort of puts the cart before the programming horse, here are the results from running my program, written in QuickC. The third line shows the 14 bytes of the file header. These 14 bytes contain 5 values:

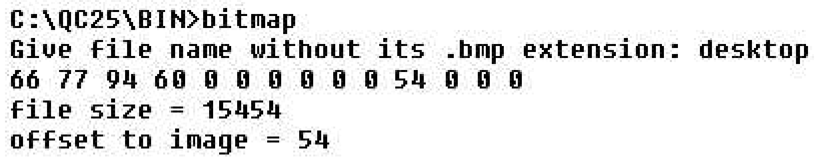
The uppercase characters BM, ASCII codes 66 and 77 expressed as a base-10 integer	2 bytes
File size, bytes	4 bytes
Two "reserved values" that are not needed	2 bytes each
Offset to beginning of image data	4 bytes

In principle, it is possible to read these integer values directly by choosing appropriately declared data types that will automatically select the appropriate number of bytes. However, this may give unpredictable results with different C compilers, so I chose a more simple-minded, although probably more tedious solution: read the header one character at a time (characters occupy a single byte) and then use explicit typecasting to force C to interpret each character as an integer. For this to work, the C declaration must be`unsigned char` rather than `char`. Here's the code:

The two important values in the header are the file size, in bytes, and the offset from the beginning of the file to the start of the image itself, in bytes. The 14 bytes are stored in a character array. The file size starts at element 2 (in C, the first element in an array is 0, not 1) and is stored from low byte to high byte, left to right. I have assumed that even large `.bmp` file sizes will need no more than 3 bytes. I also declared many of the integer values as type `long` because the `int` data type may not be able to represent large file sizes. For the offset value, two characters should always be enough; this value starts at character 10.

Now, let's find out how the image is organized by reading the 40-byte image information header. It contains:

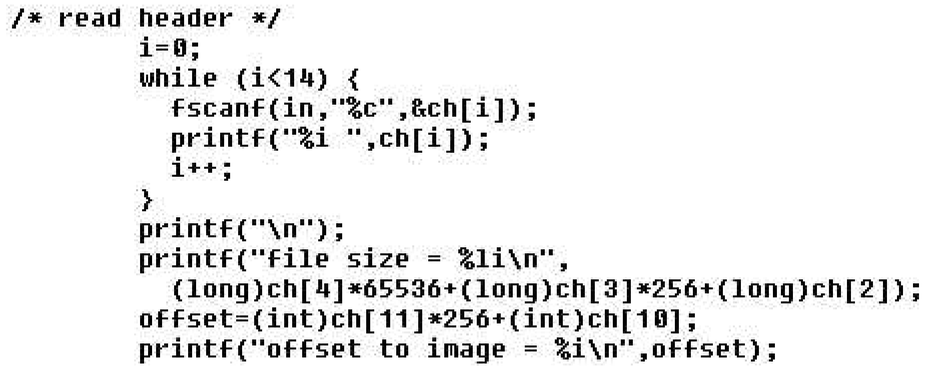
Header size, bytes (should be 40)	4 bytes
Image width, pixels	4 bytes
Image height, pixels	4 bytes
Number of color planes	2 bytes
Bits per pixel, 1 to 24	2 bytes
Compression, bytes (assumed 0)	4 bytes
Image size, bytes	4 bytes
X-resolution and y-resolution, pixels per meter	4 bytes each
Number of colors and "important colors," bytes	4 bytes each



```
C:\QC25\BIN>bitmap
Give file name without its .bmp extension: desktop
66 77 94 60 0 0 0 0 0 0 54 0 0 0
file size = 15454
offset to image = 54
```

Several of the values are simply absent from this file, and I do not know why IrfanView did not put them there when it created the file. I do not know what "important colors" means. In any event, the apparently missing values aren't important for our purposes.

The useful values for this image are the width and height, 73 and 70 pixels, and the image size, 73•70 = 5110 pixels. The first byte in the file is at an offset of 0, so an image offset of 54 means that the image starts at the 55th byte in the file. The image is stored line-by-line. Each pixel requires 3 bytes. Each line has an end-of-line mark. So, the image part of the file equals 5110•3+70=15400. Adding the 54 bytes for the header records gives the file size, 15400+54=15454 byites



```
/* read header */
        i=0;
        while (i<14) {
          fscanf(in,"%c",&ch[i]);
          printf("%i ",ch[i]);
          i++;
        }
        printf("\n");
        printf("file size = %li\n",
          (long)ch[4]*65536+(long)ch[3]*256+(long)ch[2]);
        offset=(int)ch[11]*256+(int)ch[10];
        printf("offset to image = %i\n",offset);
```

The code needs to be compiled with the command “make”

```
pi@raspberrypi3:~/openjpeg/build $ make
[ 28%] Built target openjp2
[ 40%] Built target opj_decompress
[ 52%] Built target opj_compress
[ 64%] Built target opj_dump
[ 68%] Built target compare_dump_files
[ 75%] Built target compare_images
[ 77%] Built target j2k_random_tile_access
[ 81%] Built target compare_raw_files
[ 84%] Built target test_tile_encoder
[ 86%] Built target test_tile_decoder
[ 89%] Built target ppm2rgb3
[ 92%] Built target include_openjpeg
[ 94%] Built target testempty0
[ 97%] Built target testempty2
[100%] Built target testempty1
```

```
pi@raspberrypi3:~/openjpeg/build/bin $ ./test_tile_encoder lena_rgb_256.bmp
```

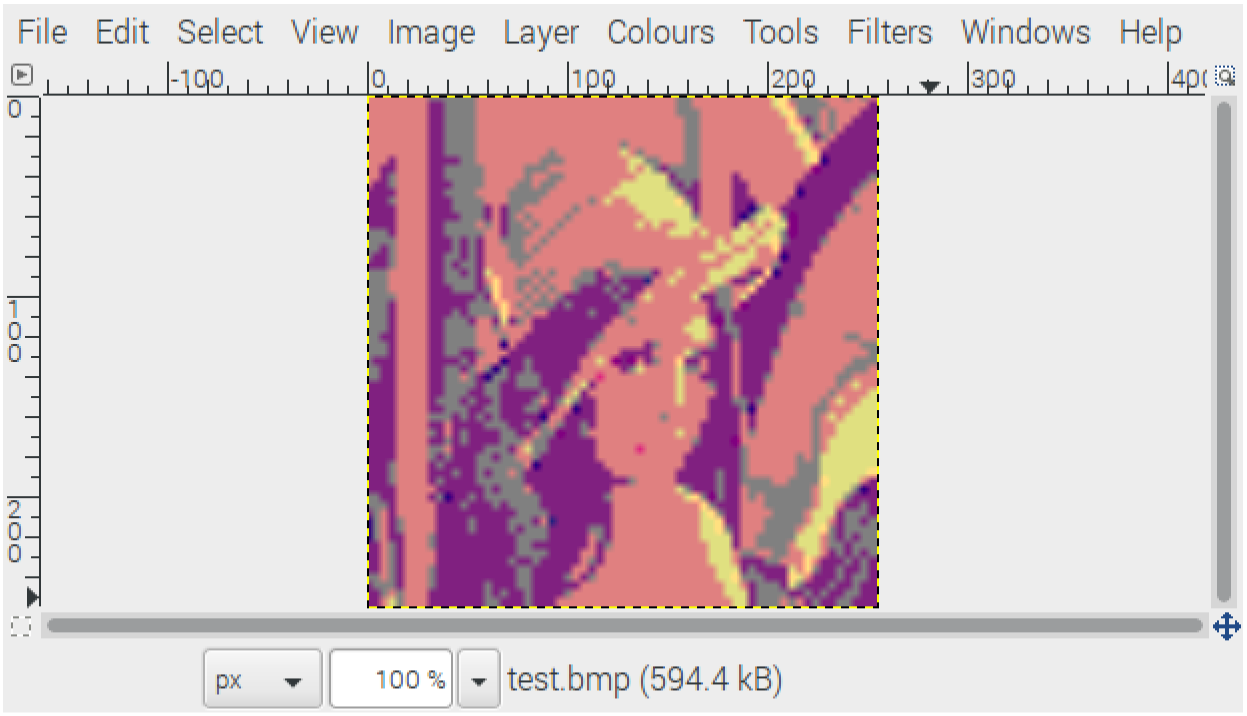
The test_tile_encoder reads the bitmap craate files red-out.32t, grn-out.32t, and blu-out.32t which can be read with octave using "disp_rgb.m". Figures 1-3 are images produces with the commands in upper right shell. In addition the Figure 4 is the DWT of the blue sub band A complete listing if the debug output is found at Appendix A. test_tile_encoder debug A complete listing if the debug output is found at Appendix B. ./opj_compress -mct 0 -n 3 -i lena_rgb_256.bmp -o tt.jp2 debug output. The difference between the Appendix A & Appendix B is in the first 30 lines where the bitmap is being readed.

Note: The results of programs test_tile_tile_encoder & opj_compress are not the same.

```
./test_tile_encoder ../../lena_rgb_256.bmp
./opj_decompress -i test.j2k -o test.bmp
```

As the l_param.tcp_distoratio[0] is changed the quality of the compressed decompressed image apears to improve. In addition the compressed image file test.j2k increases

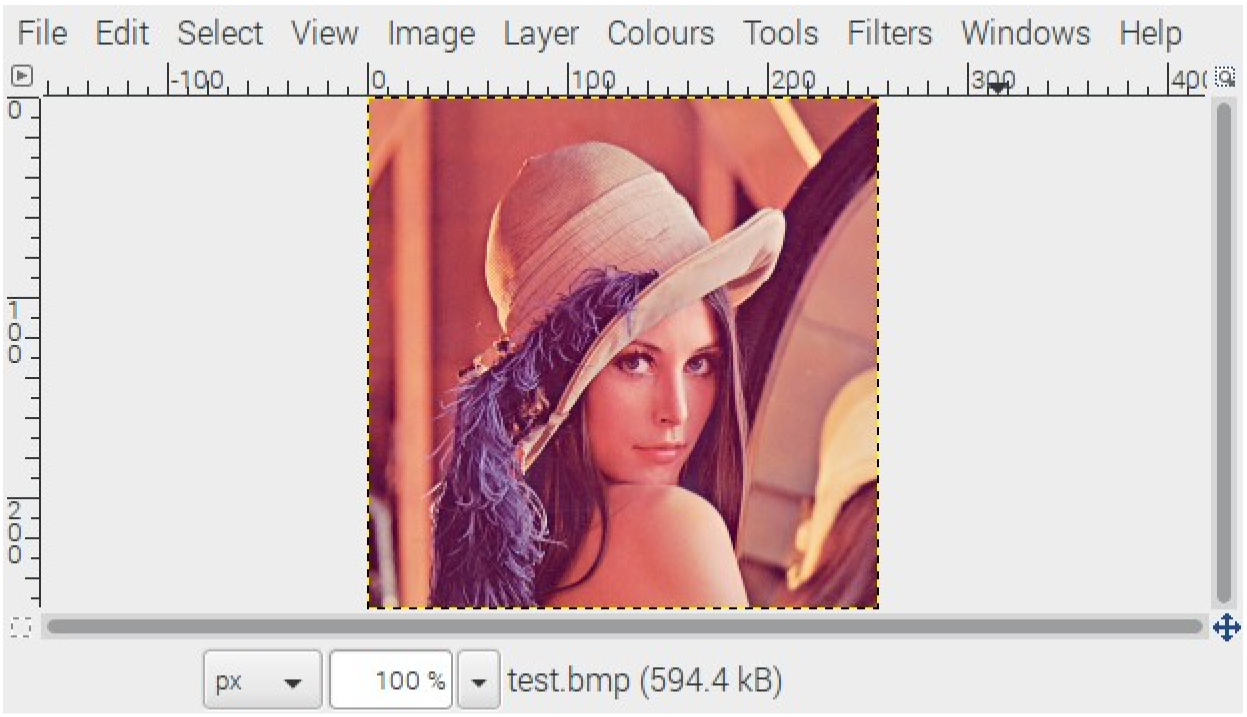
```
-rw-r--r-- 1 pi pi     787 Mar 20 17:32 test.j2k
l_param.tcp_distoratio[0] = 20;
-rw-r--r-- 1 pi pi   41860 Mar 20 17:35 test.j2k
l_param.tcp_distoratio[0] = 40;
-rw-r--r-- 1 pi pi  112303 Mar 20 17:36 test.j2k
l_param.tcp_distoratio[0] = 60;
-rw-r--r-- 1 pi pi  118495 Mar 20 17:38 test.j2k
l_param.tcp_distoratio[0] = 80;
```



Note: completly removed the distroation line.

```
-       l_param.tcp_distoratio[0] = 20;
+       //l_param.tcp_distoratio[0] = 20;
```

```
-rw-r--r-- 1 pi pi 196730 Mar 17 20:31 ../../lena_rgb_256.bmp
pi@raspberrypi3:~/t_ultibo/build/bin $ ./test_tile_encoder ../../lena_rgb_256.bmp
-rw-r--r-- 1 pi pi 119085 Mar 20 19:40 test.j2k
pi@raspberrypi3:~/t_ultibo/build/bin $ ./opj_decompress -i test.j2k -o test.bmp
```



Now, no flip vertically is required.
Original data offset by 122

```
00000070   00 00 00 00  00 00 00 00  00 00 38 15  50 3E 1E 5F  ..........8.P>._
```

Compresed decompresed offset 54 writen by ./opj_decompress -i test.j2k -o test.bmp

```
00000030   00 00 00 00  00 00 38 15  50 3E 1E 5F  3C 19 5F 3D  ......8.P>._<._=
```

1. image is upside down indicating that bitmap reading section of the program test_tite_encoder is not taking in account the most bitmap images are stored upside

2. Colors are not correct.

Note: It should be noted that the majority of the debug output is in the output comes from t1.c. The files t1.c, t2.c and mqc.c are the files that makeup the ***Embedded Block Coding with Optimal Truncation*** *EBCOT*

Appendix D Runs with test_tile_encoder & opj_compress with debug remove from t1.c.

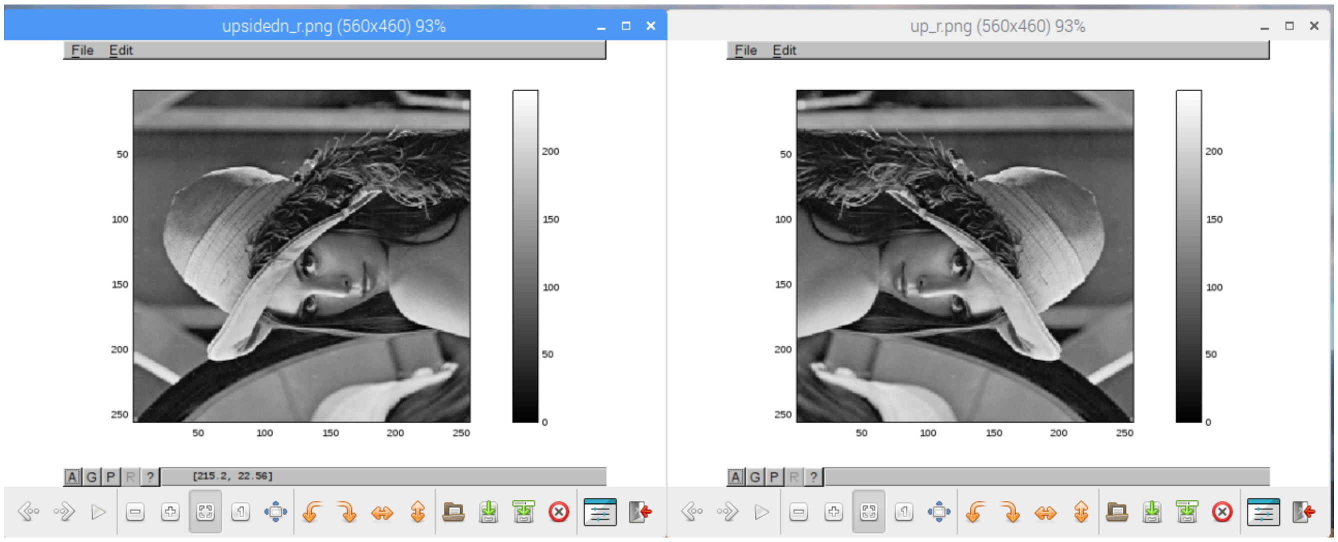
The steps to compile are found at Appendix E steps to compile
The lines of code are found at appendix Appendix F lines of code.

Contents of disp_rgb.m

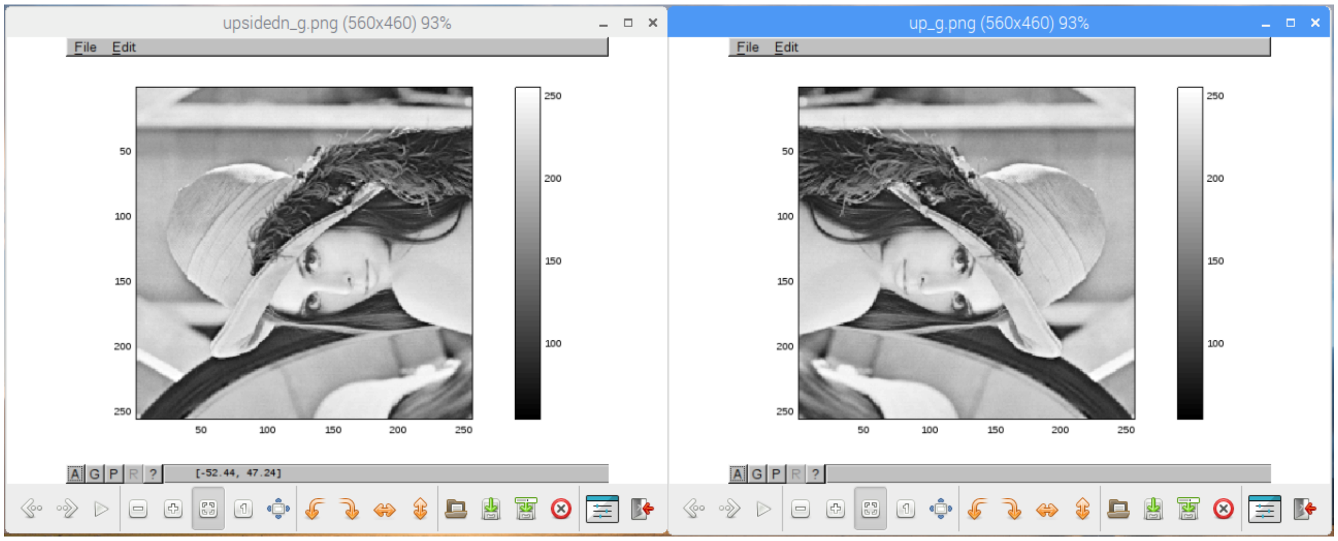
```
fid = fopen('red-out.32t','r'); im4 = fread(fid, [256,inf], 'char'); fclose(fid);
fid = fopen('grn-out.32t','r'); im5 = fread(fid, [256,inf], 'char'); fclose(fid);
fid = fopen('blu-out.32t','r'); im6 = fread(fid, [256,inf], 'char'); fclose(fid);

figure
imagesc(im4)
colorbar
colormap 'gray'
title 'red sub band lena rgb 256.bmp'
figure
imagesc(im5)
colorbar
colormap 'gray'
title 'green sub band lena rgb 256.bmp'
figure
imagesc(im6)
colorbar
colormap 'gray'
title 'blue sub band lena_rgb rgb 256.bmp'
```

Contents of dwt.m

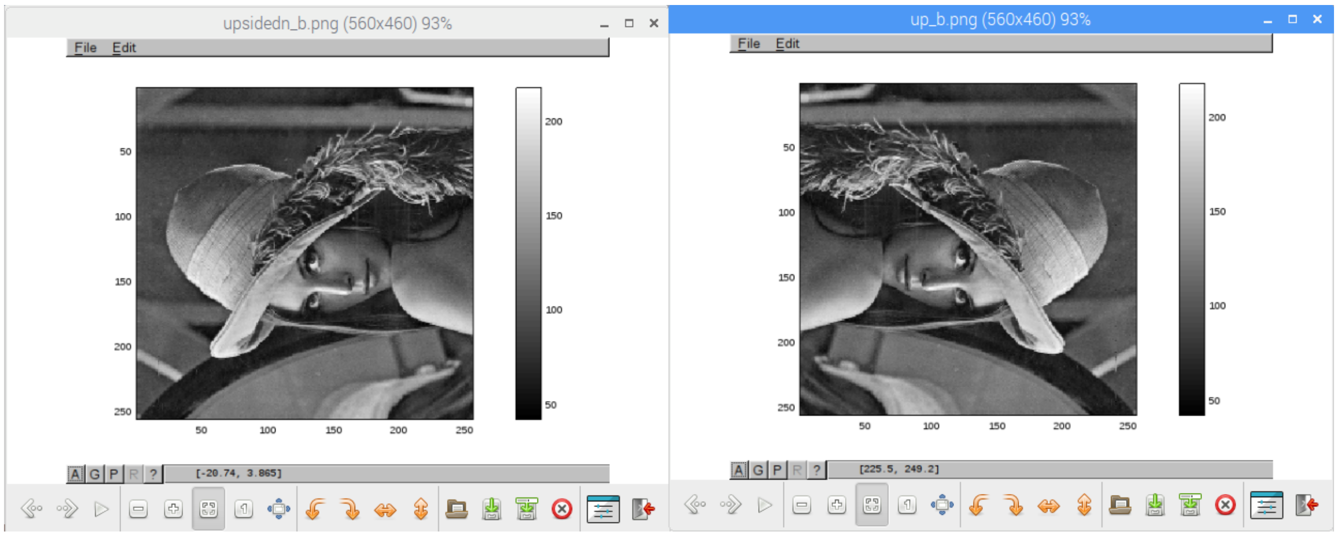
```
fid = fopen('dwt-out.32t','r'); im1 = fread(fid, [256,inf], 'int32'); fclose(fid);
figure; imagesc(im1); colorbar; colormap 'gray'
title 'test tile encoder 3 lvls dwt'
```



Now with the revised method of reading the bitmap. The image on the left is the red upside down that was used.



Now with the revised method of reading the bitmap. The image on the left is the green upside down that was used.



Now with the revised method of reading the bitmap. The image on the left is the blue upside down that was used.
It should be noted the orientation of the images is reversed but the range of the values for each is the same.

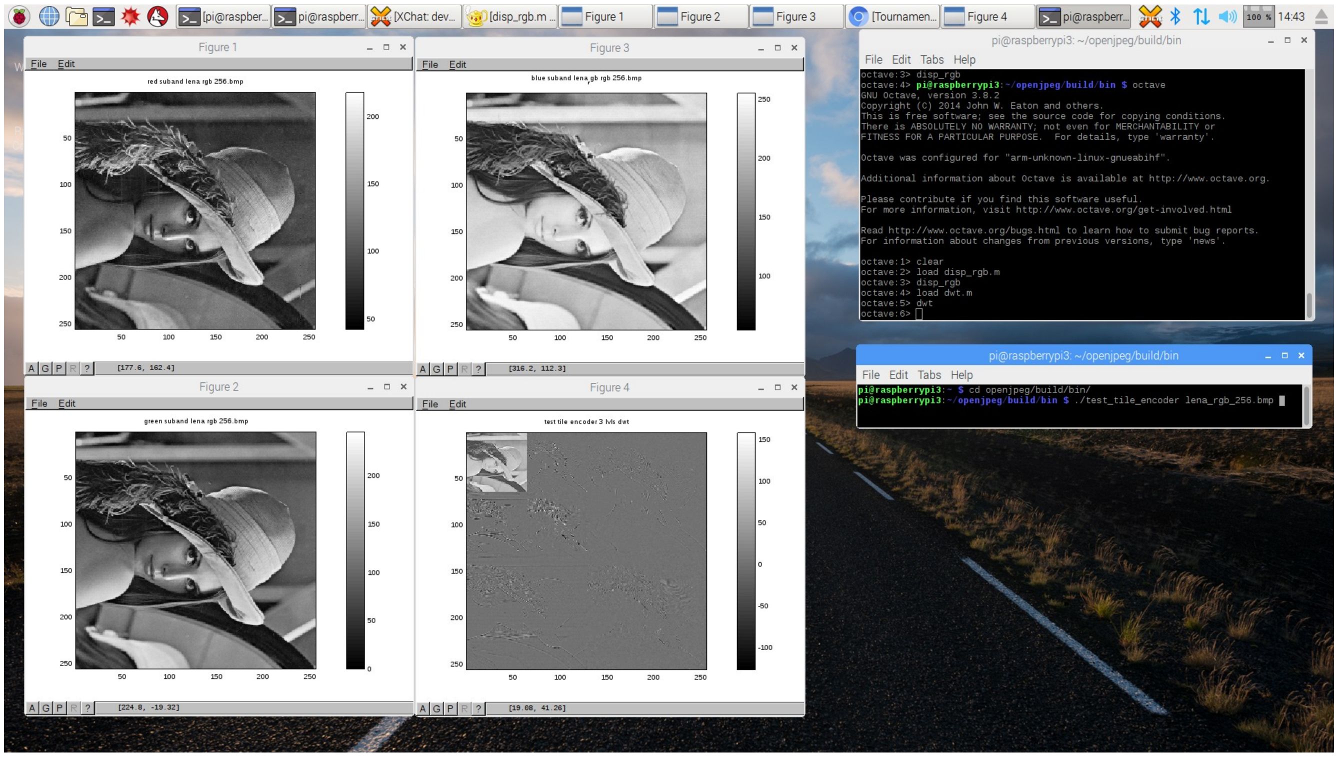


Figure 1
red suband lena rgb 256.bmp
Figure 3
blue suband lena gb rgb 256.bmp
Figure 2
green suband lena rgb 256.bmp
Figure 4
test tile encoder 3 lvls dwt
[177.6, 162.4]
[316.2, 112.3]
[224.8, -19.32]
[19.08, 41.26]

```
octave:3> disp_rgb
octave:4> pi@raspberrypi3:~/openjpeg/build/bin $ octave
GNU Octave, version 3.8.2
Copyright (C) 2014 John W. Eaton and others.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE.  For details, type 'warranty'.

Octave was configured for "arm-unknown-linux-gnueabihf".

Additional information about Octave is available at http://www.octave.org.

Please contribute if you find this software useful.
For more information, visit http://www.octave.org/get-involved.html

Read http://www.octave.org/bugs.html to learn how to submit bug reports.
For information about changes from previous versions, type 'news'.

octave:1> clear
octave:2> load disp_rgb.m
octave:3> disp_rgb
octave:4> load dwt.m
octave:5> dwt
octave:6>
```

```
pi@raspberrypi3:~ $ cd openjpeg/build/bin/
pi@raspberrypi3:~/openjpeg/build/bin $ ./test_tile_encoder lena_rgb_256.bmp
```

The changes to openjpeg
commit 8c33128369816be09968712b50681e743464b93c
Author: Antonin Descampe <antonin@gmail.com>
Date: Fri Mar 3 23:23:39 2017 +0100

Fixed CRLF auto conversion issue in openjpeg-data #655
are found at Appendix C Changes to opemjpeg 8c33128369816.

Steps to compile the file needed for the openjpeg library for use in Ultibo.

```
cd ../../src/lib/openjp2
cp ../../../build_gccultibo.sh .
cp ../../../build/src/lib/openjp2/opj_config_private.h .
cp ../../../build/src/lib/openjp2/opj_config.h .
modify opj_includes.h
#include <memory.h> -> //#include <memory.h>
```

There are 29 C files that make up the library.

bio.c	image.c	openjpeg.c	raw.c	thix_manager.c
cidx_manager.c	invert.c	opj_clock.c	t1.c	thread.c
cio.c	j2k.c	opj_malloc.c	t1_generate_luts.c	tpix_manager.c
dwt.c	jp2.c	phix_manager.c	t2.c	
event.c	mct.c	pi.c	tcd.c	
function_list.c	mqc.c	ppix_manager.c	tgt.c	

```
./compile_ultibo.sh
bio.o    function_list.o  jp2.o       opj_malloc.o       t1.o   thread.o
cio.o    image.o          mct.o       pi.o               t2.o
dwt.o    invert.o         mqc.o       raw.o              tcd.o
event.o  j2k.o            opj_clock.o  t1_generate_luts.o  tgt.o
```

Previous attempts were not providing cio.o which was due to error in compile_ultibo.sh which was compiling the cio.c into cio.c.

21 out of the 29 files compile using the arm-none-eabi-gcc compiler with the Ultibo flags
arm-none-eabi-gcc -O2 -mabi=aapcs -marm -march=armv7-a -mfpu=vfpv3-d16 -mfloat-abi=hard -c

Testing of the mct.c changes on RPi running Ultibo required a few additional chgs.

To get the code to compile I needed to add all of the openjpeg header files.
To add the mct.c which performs the RGB to YUV conversion required using 2 of the files from the openjpeg package. The files that were used opj_malloc.c & mct.c. These at found at

The file opj_malloc.c was needed to compile the mct.c for use with Ultibo. This needed 2 subroutine to be commented out
The opj_includes.h which was including memory.h needed to be modified.
I believe the problem was features.h which is the compiler options.

These 2 features needed to be commented out from opj_malloc.c
void *opj_aligned_malloc(size_t size)
void * opj_aligned_realloc(void *ptr, size_t size)

These features appear to add performance capbilities to openjpeg

The debugging files c0.bin, c1.bin, and c2.bin that were added to mct.c slowed the process from 1 usec to 17 usec.

Appendix A. test_tile_encoder debug

```
Using default image ../../lena_rgb_2048.bmp 1
testing if bitmap BM bpp = 24
allocating 0x7694b008 0x7654a008 0x76149008
size = 12582912
planes = 4096
colours = 0
height = 2048 width = 2048
bpp = 24
xresolution = 1996034048 yresolution 2129114500
```

```
rgb from Matrix to r g b ptrs
splitting data to rgb
0x7694b008 0x7654a008 0x76149008
enc 1 decomp 3 distor 44 filter 0
l_nb_tiles 1 l_data_size 12582912
loading RGB data
TRANSFER 0x7694b008 0x7654a008 0x76149008
before reset 0x7694b008 0x7654a008 0x76149008
In test_tile_encoder
creating J2k
[INFO] tile number 1 / 1
[ERROR] opj_t2_encode_packet(): only 701 bytes remaining in output buffer. 786 needed.
[ERROR] opj_t2_encode_packet(): only 2 bytes remaining in output buffer. 194 needed.
[ERROR] opj_t2_encode_packet(): only 85 bytes remaining in output buffer. 346 needed.
[ERROR] opj_t2_encode_packet(): only 15 bytes remaining in output buffer. 82 needed.
[ERROR] opj_t2_encode_packet(): only 15 bytes remaining in output buffer. 82 needed.
[ERROR] opj_t2_encode_packet(): only 15 bytes remaining in output buffer. 82 needed.
[ERROR] opj_t2_encode_packet(): only 15 bytes remaining in output buffer. 82 needed.
[ERROR] opj_t2_encode_packet(): only 15 bytes remaining in output buffer. 82 needed.
[ERROR] opj_t2_encode_packet(): only 15 bytes remaining in output buffer. 82 needed.
[ERROR] opj_t2_encode_packet(): only 15 bytes remaining in output buffer. 82 needed.
[ERROR] opj_t2_encode_packet(): only 15 bytes remaining in output buffer. 82 needed.
[ERROR] opj_t2_encode_packet(): only 15 bytes remaining in output buffer. 82 needed.
[ERROR] opj_t2_encode_packet(): only 15 bytes remaining in output buffer. 82 needed.
[ERROR] opj_t2_encode_packet(): only 15 bytes remaining in output buffer. 82 needed.
[ERROR] opj_t2_encode_packet(): only 15 bytes remaining in output buffer. 82 needed.
[ERROR] opj_t2_encode_packet(): only 15 bytes remaining in output buffer. 82 needed.
[ERROR] opj_t2_encode_packet(): only 15 bytes remaining in output buffer. 82 needed.
[ERROR] opj_t2_encode_packet(): only 15 bytes remaining in output buffer. 82 needed.
[ERROR] opj_t2_encode_packet(): only 15 bytes remaining in output buffer. 82 needed.
[ERROR] opj_t2_encode_packet(): only 15 bytes remaining in output buffer. 82 needed.
[ERROR] opj_t2_encode_packet(): only 15 bytes remaining in output buffer. 82 needed.
[ERROR] opj_t2_encode_packet(): only 15 bytes remaining in output buffer. 82 needed.
FREE 0x7694b008 0x7654a008 0x76149008
```

Appendix B. ./opj_compress -mct 0 -n 3 -i lena_rgb_256.bmp -o tt.jp2 debug output.

Appendix C Changes to opemjpeg 8c33128369816.

Appendix D Runs with test_tile_encoder & opj_compress with debug remove from t1.c.