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

Pico Lifting Step
with CRC and head & tail code.

11/17/21

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

Now have all of code to receive data over the serial port checking the data received was correct using a CRC. Once all of the data is received compute the lifting step. And send over the serial port the results of the lifting step.

Sending 151,147,140,134,167,193,195,19 in groups of 10 and computing the CRC
1 5 1 , 1 4 7 , 1 4
CRC = 0x56

0,134,167,
CRC = 0x18

,193,195,19
CRC = 0x5b

head < endofbuf

head < endofbuf

head < endofbuf

head < endofbuf

head < endofbuf

head < endofbuf

head < endofbuf

head < endofbuf

head < endofbuf

head == endofbuf

tail < endofbuf

tail < endofbuf

tail < endofbuf

tail < endofbuf

tail < endofbuf

tail < endofbuf

tail < endofbuf

tail < endofbuf

tail < endofbuf

tail < endofbuf

1 5 1 , 1 4 7 , 1 4

0x20005178 0x20005181 0x20005178

CRC = 0x56

0x20005178 0x20005181 0x20005178

head < endofbuf

head < endofbuf

head < endofbuf

head < endofbuf

head < endofbuf

head < endofbuf

head < endofbuf

head < endofbuf

head < endofbuf

head == endofbuf

tail < endofbuf

tail < endofbuf

tail < endofbuf

tail < endofbuf

tail < endofbuf

tail < endofbuf

tail < endofbuf

tail < endofbuf

tail < endofbuf

tail < endofbuf

tail < endofbuf

0 , 1 3 4 , 1 6 7 ,

0x20005178 0x20005181 0x20005178

CRC = 0x18

0x20005178 0x20005181 0x20005178

head < endofbuf

head < endofbuf

head < endofbuf

head < endofbuf

head < endofbuf

head < endofbuf

head < endofbuf

head < endofbuf

head < endofbuf

head == endofbuf

tail < endofbuf

tail < endofbuf

tail < endofbuf

tail < endofbuf

tail < endofbuf

tail < endofbuf

tail < endofbuf

tail < endofbuf

tail < endofbuf

tail < endofbuf

tail < endofbuf

1 9 3 , 1 9 5 , 1 9

0x20005178 0x20005181 0x20005178

CRC = 0x5b

0x20005178 0x20005181 0x20005178

Added control for sending results. Command (1 = Send or 0 = Wait):

The methods bump_head, bump_tail, dec_head, and dec_tail now require modifcation for receiving single bytes instead of short int.

bump_head, bump_tail, dec_head, and dec_tail code.
For testing head tail bump & dec
sleep_ms(50);
#define DBUG 1
#define DBUG1 0
#define DBUG2 0
#define imgsize 512

#define DBUG 0
#define DBUG1 1
#define DBUG2 0
#define imgsize 4096

sleep_ms(8000);
head = 0x2000114c tail = 0x2000114c end = 0x2000154c top = 0x2000114c
head = 0x2000114c tail = 0x2000114c 0x2000154c 0x2000114c
head < endofbuf
tail < endofbuf
head < endofbuf
tail < endofbuf
head < endofbuf
tail < endofbuf
head = 0x20001152 tail = 0x20001152 0x2000154c 0x2000114c
head < topofbuf
tail < topofbuf
head < topofbuf
tail < topofbuf
head < topofbuf
tail < topofbuf
head = 0x2000114c tail = 0x2000114c 0x2000154c 0x2000114c

Add code to do a CRC.

https://www.pololu.com/docs/0J44/6.7.6
Simple Example

The following example program shows how to compute a CRC byte in the C language. The outer loop processes each byte, and the inner loop processes each bit of those bytes. In the example main() routine, this is applied to generate the CRC byte in the message 0x83, 0x01, that was used in Section 6.5. The getCRC() function will work without modification in both Arduino and Orangutan programs. “pico-lifting/crc/crc-ex.c”
“gcc crc-ex.c -o crc-ex1"

./crc-ex
83 1 17
83 2 45

Advanced Example

The following example program shows a more efficient way to compute a CRC in the C language. The increased efficiency is achieved by pre-computing the CRCs of all 256 possible bytes and storing them in a lookup table, which can be in RAM, flash, or EEPROM. These table values are then XORed together based on the bytes of the message to get the final CRC. In the example main() routine, this is applied to generate the CRC byte in the message 0x83, 0x01, that was used in Section 6.5. “pico-lifting/crc/crc-ex-1.c”

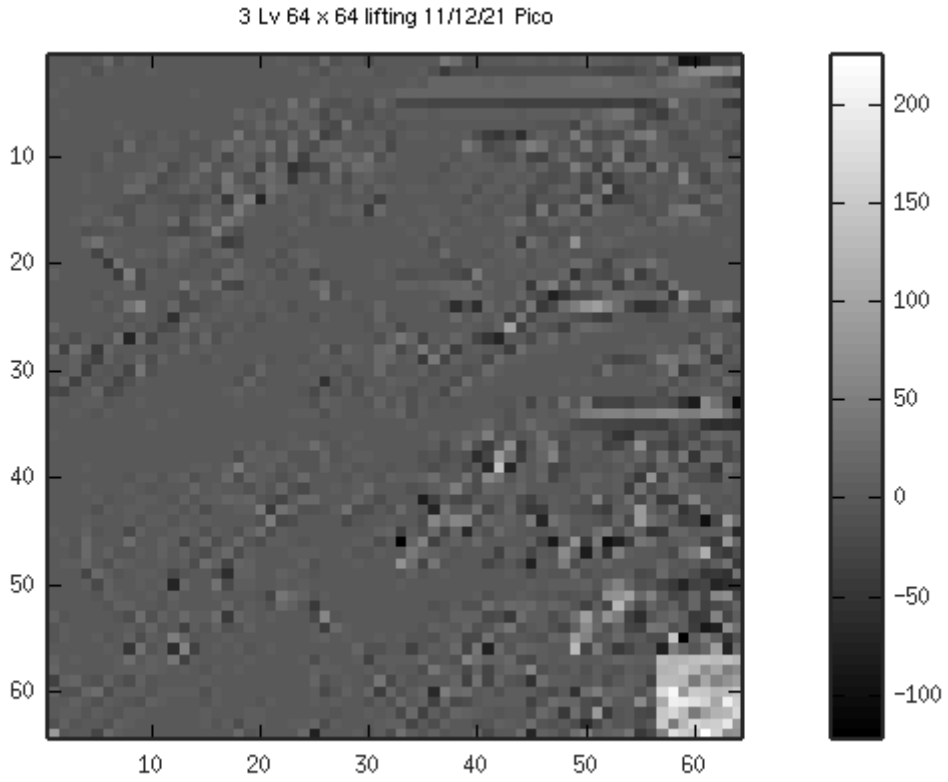
The 2 bytes 0xd3, 0x01 result in CRC8 0x4e

message[3] = {0xd3, 0x01, 0x00};

11001011 10000000 01110100

https://github.com/lammertb/libcrc.git
lena

The pico computes the 64 x 64 lifting step. In the image below



y2= 246.411

Add the C code to perform the lifting step DWT to the hello_usb.c

cp testfiles/64/* .
./buildpi_lift.sh
rm -rf dwt.bin ; ./pi_jpeg 0 1

octave
GNU Octave, version 4.4.1
Copyright (C) 2018 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 https://www.octave.org.

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

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

octave:1> rgb

Need to find how to insert the 4096 values in the program hello_usb.c
Create 4096 values of 8 bit from the file "r-8.bin" using myformat.

```
00000000   A1 9D 9C 9D  9F A2 A2 A6  AC A5 94 75  5D 5E 5E 5E  ...........u]^^^
00000010   64 68 67 66  64 66 67 6B  6D 6E 6E 6C  6B 6A 69 69  dhgfdfgkmnnlkjii
00000020   66 65 69 66  5E 52 50 54  55 52 4D 42  3C 39 35 34  feif^RPTURMB<954
00000030   34 54 83 7F  52 33 2A 27  28 28 27 4D  5B 48 39 34  4T..R3*'(('M[H94
00000040   9F 9D 9E A0  A1 A3 A7 A6  99 82 63 5C  5F 5F 5D 62  ..........c\__]b
00000050   67 67 66 65  63 64 66 67  6A 6B 6A 6A  6A 67 67 6A  ggfecdfgjkjjjggj
00000060   65 62 64 5D  56 4F 4D 4E  50 51 4C 47  43 3C 36 32  ebd]VOMNPQLGC<62
00000070   2F 3B 6C 91  98 95 8C 86  7A 70 6B 6E  7C 69 53 48  /;l.....zpkn|iSH
00000080   9D 9C 9E 9F  A5 A6 9E 89  68 57 5A 5C  5D 5E 5B 60  ........hWZ\]^[`
00000090   65 67 68 66  64 6A 6C 6D  6C 68 66 64  64 63 64 64  eghfdjlmlhfddcdd
000000A0   5F 59 58 52  4C 47 42 3F  45 4A 48 48  4B 4C 5B 55  _YXRLGB?EJHHKL[U
000000B0   48 3F 4E 71  8F 99 9C A0  A1 A1 AA B0  AF A8 B4 BF  H?Nq............
000000C0   9B 9B 9D A4  A8 9F 90 84  7F 7F 80 7F  7D 7E 7B 7A  ............}~{z
000000D0   7C 80 7E 7E  80 82 86 85  84 7F 7A 78  78 76 76 74  |.~~......zxxvvt
000000E0   70 6D 6A 65  62 60 5D 5A  65 75 77 75  78 73 77 79  pmjeb`]Zeuwuxswy
000000F0   70 5B 48 46  46 3E 3F 47  4E 59 65 75  82 86 A2 B5  p[HFF>?GNYeu....
00000100   A0 A3 A6 A6  A5 A0 9F A2  A3 A4 A3 A1  9F 9E 9E 9F  ................
00000110   9F 9F A2 A3  A3 A4 A5 A5  A3 A4 A2 A1  A2 A1 A0 A1  ................
00000120   A0 A2 A1 A0  9F 9E 9D 9D  9D A0 A1 A2  A3 9F 9B 9C  ................
00000130   9D 98 94 95  97 96 96 95  92 8E 8A 88  87 85 85 86  ................
00000140   AA AA A5 A0  9F A1 A3 A5  A6 A7 A8 A7  A6 A5 A4 A7  ................
00000150   A8 AA AE AD  AB AD AE AD  AC AD AD AD  AD AE AF AF  ................
00000160   B1 B1 B3 B3  B2 B0 AD AF  AD AA AB AE  B1 B2 B0 AF  ................
                    .
                    .
                    .
00000FB0   55 42 59 68  62 63 68 63  5E 60 60 57  52 45 3D 3D  UBYhbchc^``WRE==
00000FC0   8E 35 33 37  36 32 32 34  30 67 95 92  A1 A2 9D 9D  .53762240g......
00000FD0   9B 99 9A 9C  9C 99 98 98  98 98 98 98  97 97 8F 85  ................
00000FE0   88 8C B7 BE  C2 CC CC D0  CF D1 D3 D5  D2 CF D1 AC  ................
00000FF0   37 55 59 5E  66 66 60 58  5E 52 40 3E  43 3F 3B 5B  7UY^ff`X^R@>C?;[
```

```
gcc myformat.c -o myformat

./myformat
161,157,156,157,159,162,162,166,172,165,148,117,93,94,94,94,
100,104,103,102,100,102,103,107,109,110,110,108,107,106,105,105,
102,101,105,102,94,82,80,84,85,82,77,66,60,57,53,52,
52,84,131,127,82,51,42,39,40,40,39,77,91,72,57,52,
159,157,158,160,161,163,167,166,153,130,99,92,95,95,93,98,
103,103,102,101,99,100,102,103,106,107,106,106,106,103,103,106,
101,98,100,93,86,79,77,78,80,81,76,71,67,60,54,50,
47,59,108,145,152,149,140,134,122,112,107,110,124,105,83,72,
157,156,158,159,165,166,158,137,104,87,90,92,93,94,91,96,
101,103,104,102,100,106,108,109,108,104,102,100,100,99,100,100,
95,89,88,82,76,71,66,63,69,74,72,72,75,76,91,85,
72,63,78,113,143,153,156,160,161,161,170,176,175,168,180,191,
                    .
                    .
                    .
```

85,66,89,104,98,99,104,99,94,96,96,87,82,69,61,61,
142,53,51,55,54,50,50,52,48,103,149,146,161,162,157,157,
155,153,154,156,156,153,152,152,152,152,152,152,151,151,143,133,
136,140,183,190,194,204,204,208,207,209,211,213,210,207,209,172,
55,85,89,94,102,102,96,88,94,82,64,62,67,63,59,91,

The following was added to "hello_usb.c"

```
struct PTRs {
        /*This is the buffer for inp & output
        2048 x 2048 = 4194304
        256 x 256 = 65536
        */
        short int inpbuf[4096*2];
        short int *inp_buf;
        short int *out_buf;
        short int flag;
        short int w;
        short int h;
        short int *fwd_inv;
        short int fwd;
        short int *red;
} ptrs;

const short int a[] = {161,157,156,157,159,162,162,166,172,165,148,117,93,94,94,94,
                        .
                        .
                        .
142,53,51,55,54,50,50,52,48,103,149,146,161,162,157,157,
155,153,154,156,156,153,152,152,152,152,152,152,151,151,143,133,
136,140,183,190,194,204,204,208,207,209,211,213,210,207,209,172,
55,85,89,94,102,102,96,88,94,82,64,62,67,63,59,91};

for(i = 0; i < 4096;i++) ptrs.inp_buf[i] = a[i];
```

```
cd tmp
git clone git@github.com:develone/pico-lifting.git
cd pico-lifting
cp testfiles/2048/* .
./buildpi_lift.sh
rm -f dwt.bin;./pi_jpeg 0 1
```

Next instead of reading the files and writing the results in dwt.bin The goal is use as serial tx & rx to send the data to the program. Then transmit the result over the serial tx.

The folder testfiles/2048/ has the files to perform a 2048 lifting step.
The folder testfiles/256/ has the files to perform a 256 lifting step.

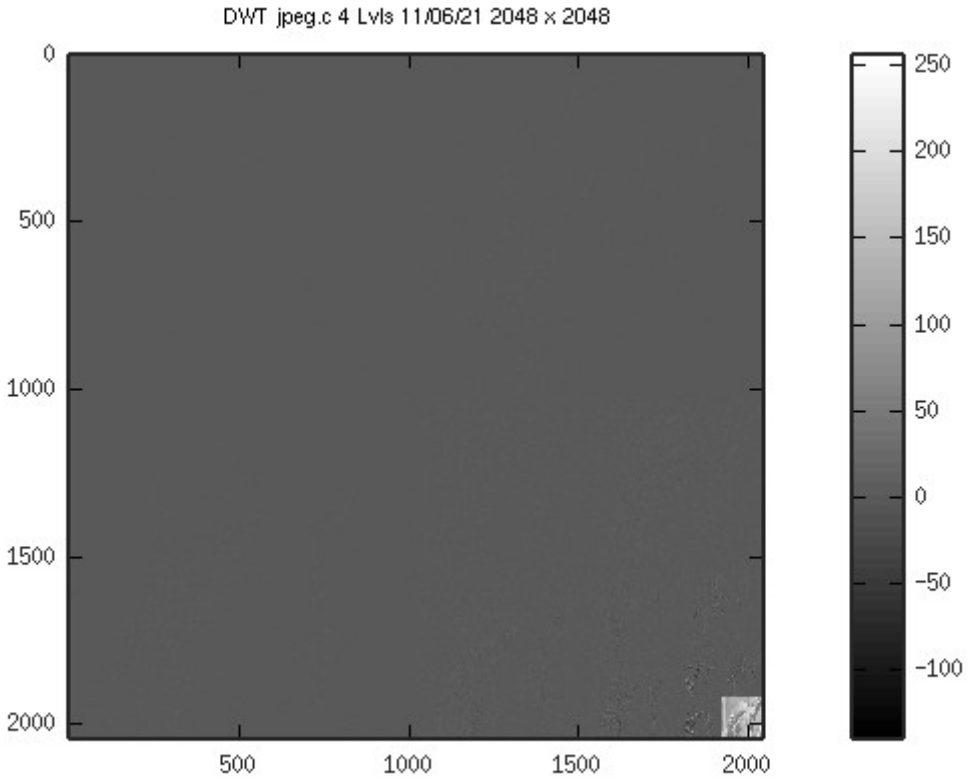
The following command compiles the code ./buildpi_lift.sh

There is a define in pi_jpeg.c that turns off the debug

```
rm -f dwt.bin ; ./pi_jpeg 0 1
0x0 0x22048 0x1022048
ptrs.fwd_inv = 0x2022060
reading r.bin
fwd lifting step only
w = 0x800 ptrs.inp_buf wptr = 0x22048 alt =  0x1022048 ptrs.fwd_inverse =  0x2022060
ptrs.fwd_inverse =  0x1
starting red dwt
finished ted dwt
octave
GNU Octave, version 4.4.1
Copyright (C) 2018 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 https://www.octave.org.

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

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

octave:1> rgb

The input was a pgm file 2048 x 2048
```

y2=-4.11346

4 lvs 2048 x 2048 lifting step.



cp testfiles/256/* .
 ./buildpi_lift.sh
devel@mypi3-20:~/pico-lifting $ rm -f dwt.bin ; ./pi_jpeg 0 1
0x0 0x22048 0x62048
ptrs.fwd_inv = 0xa2060
reading r.bin
fwd lifting step only
w = 0x100 ptrs.inp_buf wptr = 0x22048 alt = 0x62048 ptrs.fwd_inverse = 0xa2060
ptrs.fwd_inverse = 0x1
starting red dwt
finished ted dwt

octave
GNU Octave, version 4.4.1
Copyright (C) 2018 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 https://www.octave.org.

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

Read https://www.octave.org/bugs.html to learn how to submit bug reports.

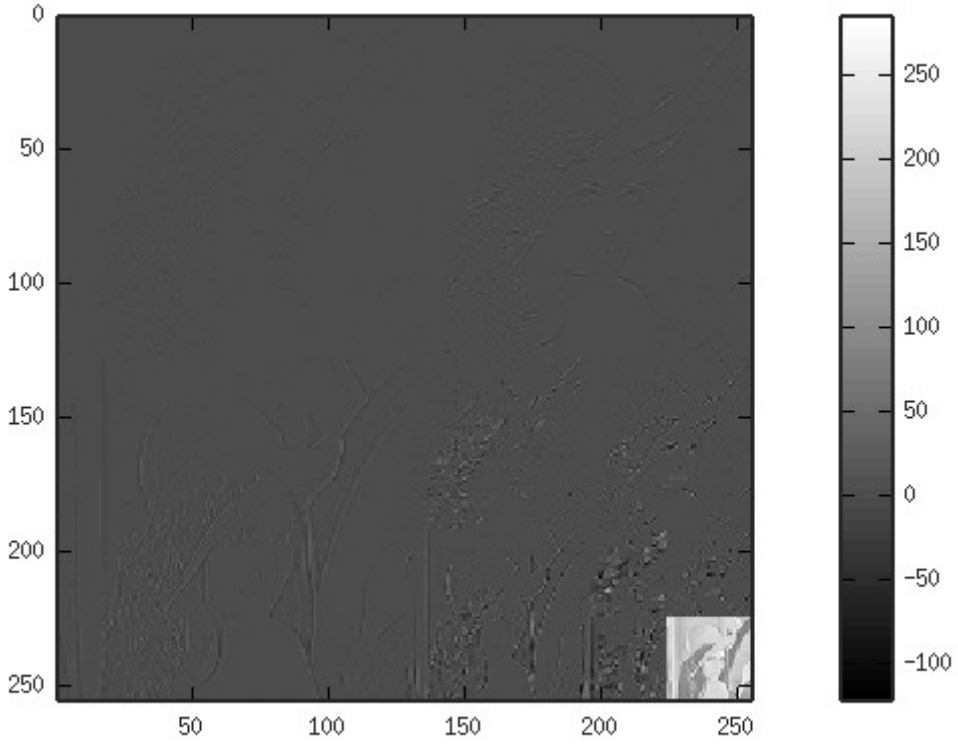
For information about changes from previous versions, type 'news'.

octave:1> rgb

The input was a pgm file 256 x 256

y2= 174.001

3 lvs 256 x 256 lifting step.



y2=-178.095