Analysis of OregonDecode program behaviour

1 Test environment

The test environment has been set up to find out the reasons of strange and wrong behaviour of the program for decoding of Manchester coded data.

1.1 Signal to be decoded

Signals are Manchester encoded OOK data from a wind and a temperature sensor.
The data are simulated to get defined and known values.
The data are sent by a 433 MHz transmitter and received by a receiver with digital output.
The receiver output is connected to a comparator input of the Teensy 3.2.
Data information is contained in pulse widths, measured in usec, short pulses between PW1 and PW2, long pulses between PW2 and PW3.

```
#define PW1 350
#define PW2 720
#define PW3 1150
```

Pulses below PW1 and above PW3 are invalid pulses.
Pulse widths are measured within an interrupt routine, triggered by rising and falling edges.
A data set (either wind data or temperature data) consists of a sequence of at least 16 consecutive short pulses, preamble, (corresponds to 32 edges, see #define FLIP 32), followed by valid pulses (short or long) until 80 data bits have been decoded. A data set with 80 bits is a complete data set.

The data simulator continually generates data sets. Every 7 sec wind data are sent. After every 4th wind data sets one temperature data set follows 2 sec after wind data. The values of wind speed vary in steps of +-4 between 6 and 42. The values of temperature vary in steps of +-3 between -110 and 150.

1.2 Principle of decoding

The ful code of the interrupt routine, cmp1_isr() and the class OOK_OregonDecodeV3_2 is shown in SpracheOregondec_Forum.zip.
The interrupt routine, cmp1_isr(), measures pulse widths and puts them to the decoder which is implemented as a state machine in the class OOK_OregonDecodeV3_2.
There are 4 states, UNKNOWN, T0, OK, DONE, handeled within decode().
State UNKNOWN is active during the preamble pulses. During the state UNKNOWN the variable flip counts the number of edges. With the first long pulse data bits are beginning. Now the variable flip represents the bit values 0 and 1. They are decoded in the states T0 and OK. When 80 data bits have been decoded a data set is complete.
When an invalid pulse is detected resetDecoder() is called which starts decoding from the very beginning.

1.3 Measurement of program execution

The interrupt routine fills 4 data buffers, data[N_DATASETS][L_DATASET], one after the other. When a data set is complete, the variable res_pos (res_pos=0 .. N_DATASETS-1) switches to the next buffer. When a buffer is filled a flag, res_valid[N_DATASETS], is set, and the content can be evaluated.
To observe the running program pins on the T3.2 board are activated by digitalWrite() statements. A logic analyzer records the signals on these pins with a sampling rate of 5 kHz. This guarantees that all valid pulses are sampled and a reasonable long time interval of about 51 sec is recorded which covers a period of time when all four buffers are filled. As mentioned before every invalid pulse shall start decoding from the very beginning.

The pins on the T3.2 board are used as follows:

Pin	Channel of LA	Meaning
2	0	res_pos=0
3	1	res_pos=1
4	2	res_pos=2
5	3	res_pos=3
6	4	state=UNKNOWN
7	5	state=T0
8	6	state=OK
9	7	state=DONE
A2	8	flip>0
A3	9	total_bits>0
A4	A	Case 2, interpretation of data set is neither wind data nor temperature data
A5	B	Pulse, in the interrupt routine a rising edge of a pulse input sets pin A5 to high, a falling edge to low

The digitalWrite() statements for the states of the state machine are contained in the class OOK_OregonDecodeV3_2, the other ones in the interrupt routine cmp1_isr() and in the evaluation part of the program (switch statement).

2 Test results

In the following sometimes code is referenced. It can be found in SpracheOregondec_Forum.zip.

2.1 Missing call to resetDecoder()

During test runs the received data sets are evaluated and printed on the Serial Monitor. Data sets for wind and temperature both have a specific ID at the beginning. If a data set neither contains the wind ID nor the temperature ID then “unbekanntes Signal” is printed on the SM. This happened quite often.

2.1.1 Detection of the error

First analysis showed that calls of resetDecoder() within the decoder class did not happen in contradiction to the code, see the following figure 2-1. Only the calls of resetDecoder() from cmp1_isr() are executed.

The screen shot shows a sequence of pulses (lower track) where obviously calls of resetDecoder() are missing.

The upper track (resp_pos=0) shows that buffer-0 is being filled with data bits.
The left part of the screen shows a very long low pulse which of course is invalid. At the end of this pulse resetDecoder() should have been called, setting flip>0 and total_bits>0 to low and state to UNKNOWN. Later there are some more invalid pulses, one example is marked with two magenta cursor lines of 3.2 ms. They all should result in a call to resetDecoder(). But it does not happen in contradiction to the code.
At the end of the pulse sequence sufficient data bits had been gathered and resetDecoder() was called from within the interrupt routine cmp1_isr() resulting in the expected behaviour of flip>0, total_bits>0 and state.

In this example invalid pulses have been mixed up with valid ones and put together to a complete data set. Of course these data are wrong and evaluation results in “unbekanntes Signal” which means “unknown data”.

During this recording lines 128 to 130 in the interrupt routine had been commented.
Therefore this recording of the LA is a proof that resetDecoder() has not been executed from within the decoder.

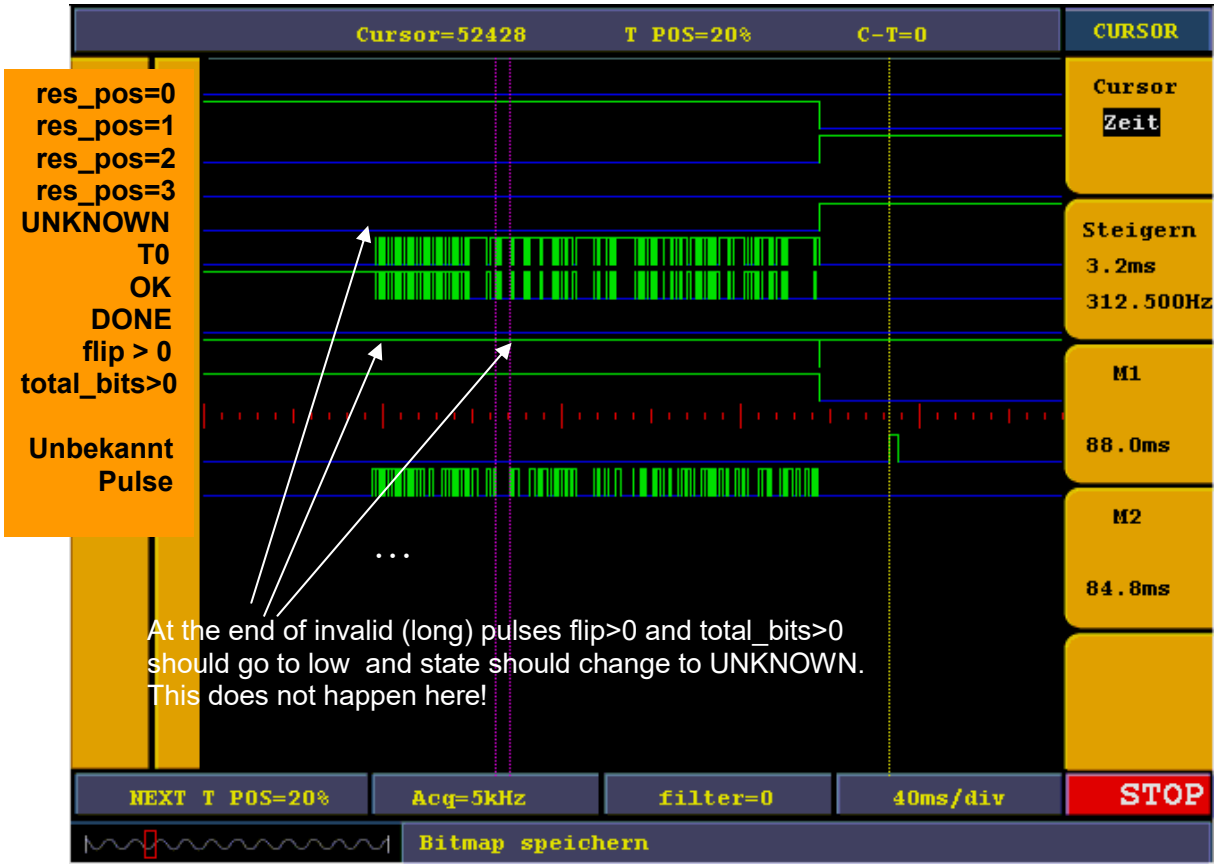


Fig. 2-1: missing call to resetDecoder() (DATA15.BMP)

2.1.2 Countermeasure

To cope with the problem lines 128 to 130 in the interrupt routine cmp1_isr() have been activated. Now the most important condition for a call to resetDecoder(), which according to the code should take place within the decoder, is repeated in the interrupt routine after each call of the decoder.
The result of this countermeasure can be seen in the following figure 2-2.

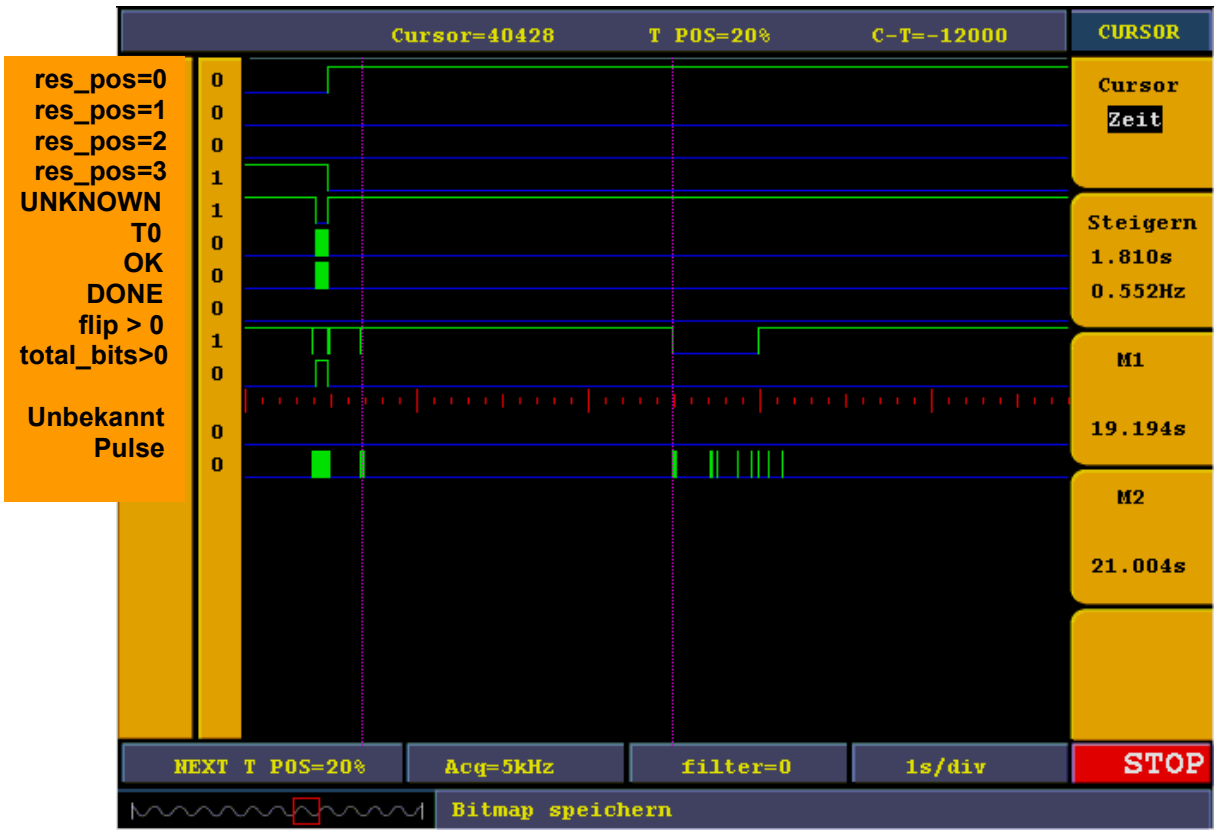


Fig. 2-2: correct behaviour (DATA13.BMP)

At the left of the screen shot we see a completed data set with buffer switching from 3 to 0. After a gap of several ms (position of left cursor) some wrong pulses came up and flip>0 goes to low and again after a very long gap until right cursor. The gaps are invalid pulses, and decoding has to be reseted which is done in line 129.

This countermeasure improved decoding of wind and temperature data dramatically.

2.2 Inhibited buffer switching

This error depends on compiler optimization. If LTO is selected (e.g. DEBUG with LTO) then this error does not occur. But without LTO the error happens.

The following figure shows the expected behaviour which is achieved with optimization setting “DEBUG with LTO”.

The first 4 tracks of the LA show that after every complete data set the buffers are switched to the next one. The variable res_pos is the buffer index.
On the bottom track we also see that some wrong pulse sequences don’t disturb decoding which is the result of the countermeasure described in chapter 2.1.2.

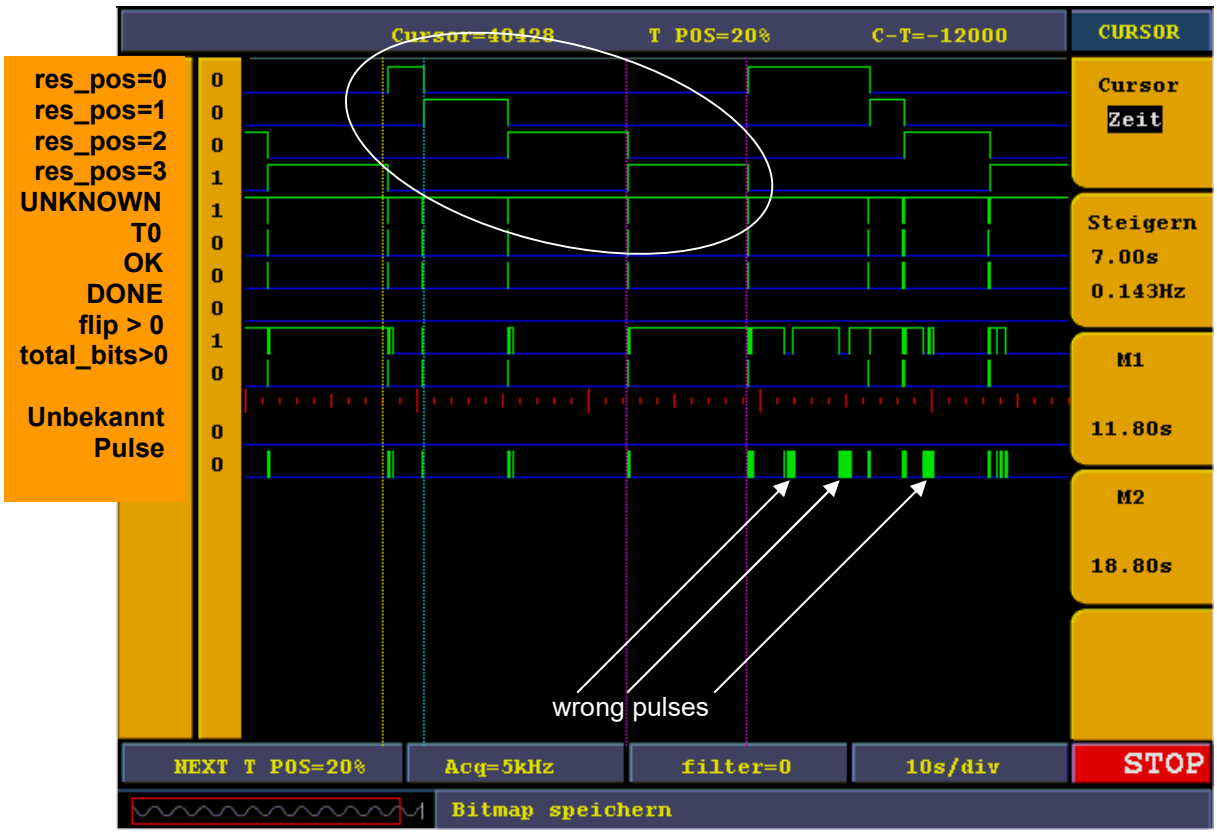


Fig. 2-3: correct buffer switching (DATA12.BMP)

If the same code as before is compiled with optimization set to DEBUG then buffers are not switched any more, see the following figure.

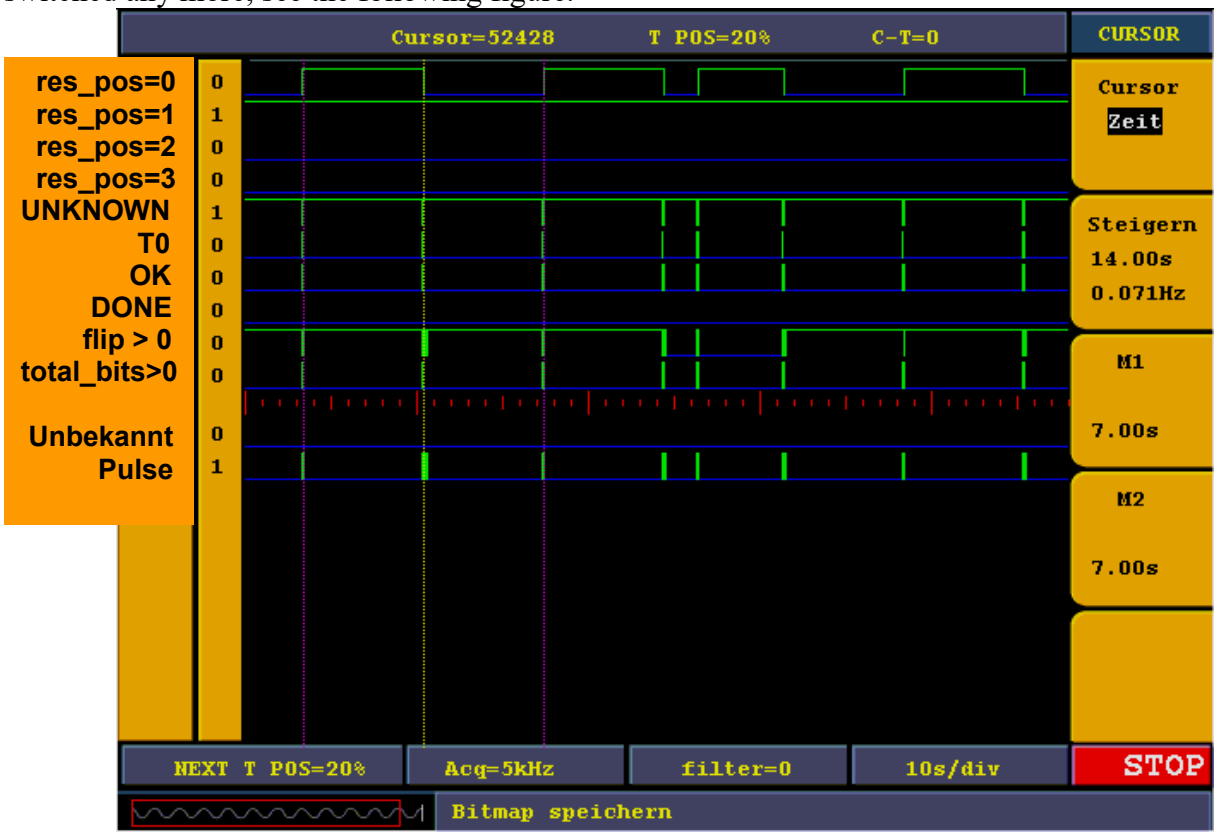


Fig. 2-4: inhibited buffer switching (DATA14.BMP)

The first 4 tracks show the buffer switching. At the left buffer 0 is selected and after a complete data set buffer 1 is selected. But the track shows that buffer 1 is selected all the time

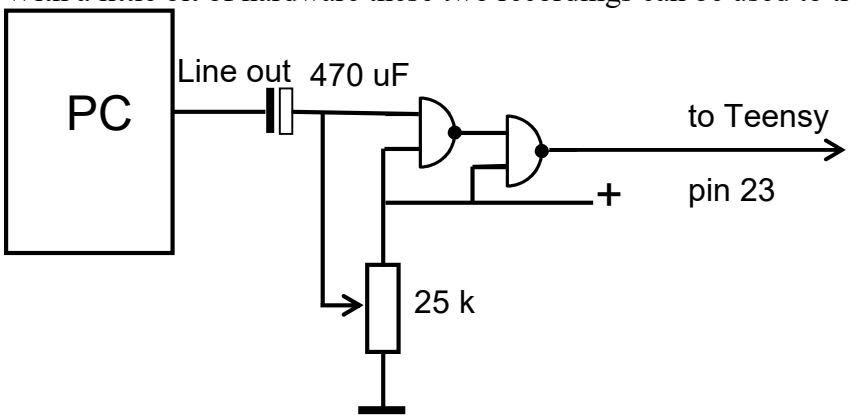
which really is not possible, but can be explained by the code. Lines 108 to 111 of the interrupt routine cmp1_isr() are switching the pins on the T3.2 board according to the values of res_pos, but only if the value of res_pos has changed.
The first two tracks show that for some reasons the switching does not happen. Therefore lines 108 to 111 are not executed. **The only difference between fig. 2-3 and fig. 2-4 was the different optimization setting for compilation**.

Another effect of inhibited buffer switching can be seen on the print outs on the Serial Monitor. For every decoded data set the number of the buffer which contained the data is printed. There are only results from buffer 0 printed. And most time with only very few exceptions every second data set is missing which exactly matches what we see on the first two tracks.

3 Reconstruction of the effects

The effects/errors described above have been measured with a hardware setup in the lab containing simulation of wind and temperature data, transmission via a 433 MHz transmitter, outputs from a 433 MHz receiver linked to pin 23 of the Teensy 3.2.
To enable a reconstruction of these effects without all the hardware around the Teensy 3.2 two sequences of 433 MHz receiver outputs have been recorded with the share ware program audacity.exe.

With a little bit of hardware these two recordings can be used to trigger the Teensy 3.2.



The PC is playing the wav-files via audacity.exe. The signal level on line out of the audio board is +-1V.
The potentiometer shall be set to about 1V to get a clear binary output from the NAND gates which should be checked by an oscilloscope.

With this setup the behaviour of the program could be reconstructed exactly as with the original hardware. The recordings also contain some signals from any other sensors sending 433 MHz signals.

The same PC can play the wav-files and display the window of the Serial Monitor.
For every valid data set the output on the Serial Monitor contains 3 lines:

```
Messzeit: 6  unk/tot: 4/87
print 24us |6848: n=3 dbcnt=10 W014R09B0
 Interrupt-Dauer von 2 bis 9
```

Relevant for evaluation of the behaviour are

- unk/tot: 4/87, two counters, the first one counts “unbekanntes Signal”, the second one counts the total number of data sets
- n=3, n is the number of the buffer which contained the result
- W014R09B0, result of a valid data set, here wind speed 1.4 m/s; in case of temperature data e.g. T-094H54B0

The values of wind speed vary in steps of +-4 between 6 and 42. The values of temperature vary in steps of +-3 between -110 and 150.

Results from OregonSignal_1.wav

1. Compilation of the code with
   - Lines 128 to 130 in SpracheOregondec.ino commented
   - CPU speed: 72 MHz
   - Optimize: DEBUG with LTO

   The Serial Monitor for the 7th data set shows "unbekanntes Signal .."

2. Compilation of the code with
   - Lines 128 to 130 in SpracheOregondec.ino active
   - CPU speed: 72 MHz
   - Optimize: DEBUG with LTO

   The Serial Monitor for the 7th message shows "W030R…"

3. Compilation of the code with
   - Lines 128 to 130 in SpracheOregondec.ino active
   - CPU speed: 72 MHz
   - Optimize: DEBUG

   The sequence of data sets compared to the test before:

DEBUG with LTO	DEBUG
n=0, W010R..	n=0, W010R..,
n=1, W014R..	**missing**
n=2, T-103H..	**n=0**, T-103H..
n=3, W018R..	**missing**
n=0, W022R..	n=0, W022R..
n=1, W026R..	**missing**
n=2, W030R..	**n=0**, W030R..
n=3, W034R..	**n=0**, W034R..
n=0, W038R..	**missing**
n=1, W042R..	**n=0**, W042R..
n=2, W038R..	**missing**
…	**…**

Results from OregonSignal_2.wav

Compilation 1 outputs "unbekanntes Signal …" for the 10th and 11th data set.
Compilation 2 outputs correct results for the 10th and 11th data set.
Compilation 3 also shows missing data sets and always n=0.