JAL: Stream Viewer

last modified: okt-2014, Stef Mientki

Introduction

Investigation (hacking) some wireless remote home controllers (mains sockets, lamps, weather stations) I was in the need of a simple logic analyzer. Although it's possible to view these slow and long signals with a scope it's not easy. And above all I wanted to see how well a SI4432 (UHF transceiver, specially designed to perform packet transmission) could both decode and encode the simple OOK (On-Off-Keying) signals from the remote home controllers. This resulted in a simple stream viewer with some nice features to easily detect and decode the OOK signals.

The stream viewer has the following features:

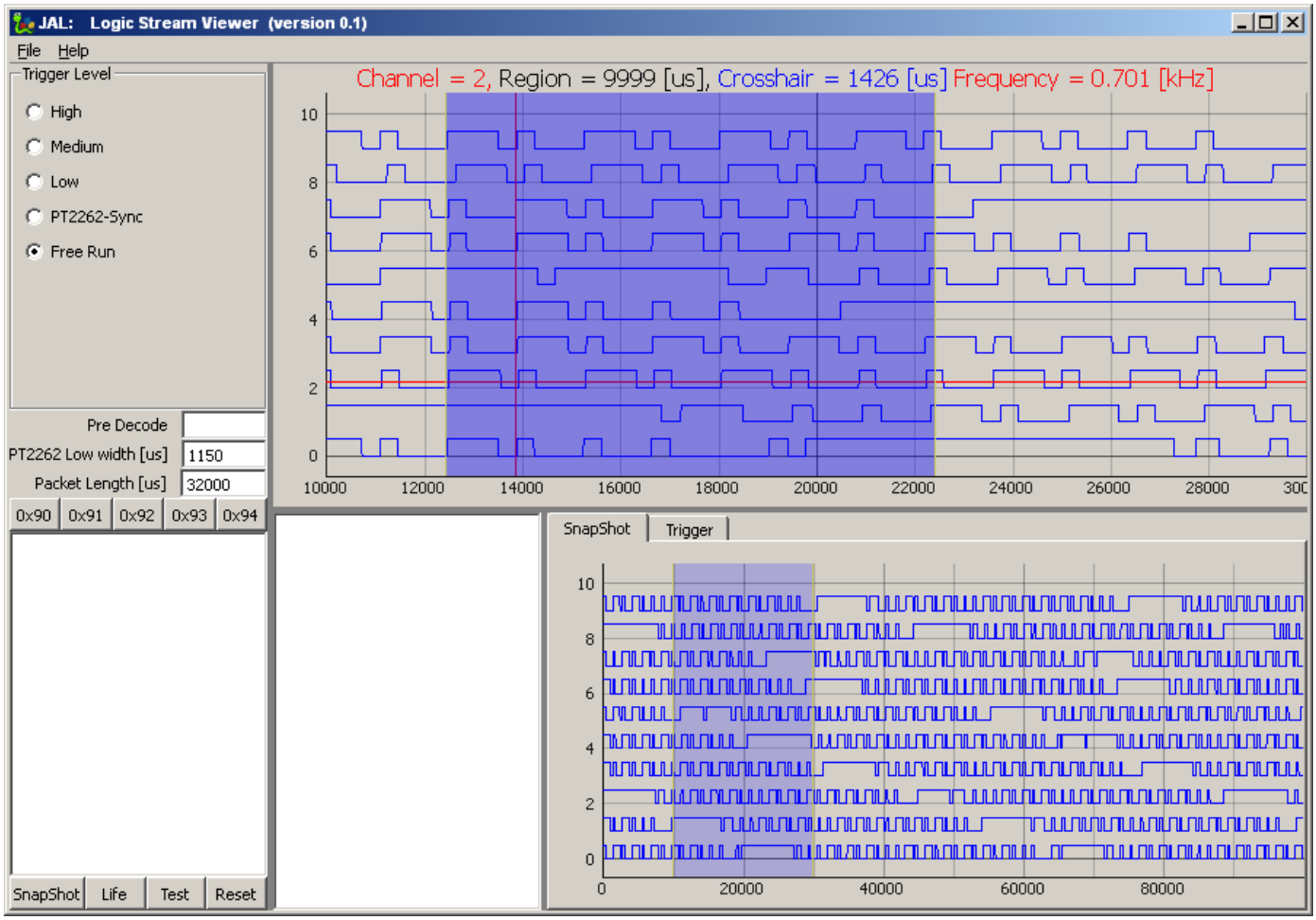
- automatic comm port detection
- all settings directly available through the GUI
- triggering on patterns or special PT2262 Sync-bit
- decoding of PT2262 codes
- visual measurements of timing (10 us resolution) in raw or zoomed recording
- 5 spare buttons for experimental use

This program is a modifed version of the UHF-spectrum analyzer, see
http://mientki.ruhosting.nl/data_www/raspberry/doc/spectrum_analyzer.html

Starting the program

After connecting the PIC to a desktop computer, start the desktop program, which will search for the correct Comm-port. When the program has found the Comm-port, it will start scanning (after a few seconds, because it needs to establish a connection with the PIC) and it will display the sampled signals in 10 traces in a graphical window. F1-key shows this document.

When the program is started the first time, the viewer will start in free running mode, signals are displayed, just as they arrive. Now wait till all 10 traces are written (it continue, overwriting the signals, until you press freeze). After pressing Freeze the program stops sampling. Now also press SnapShot and you should get a picture like this:



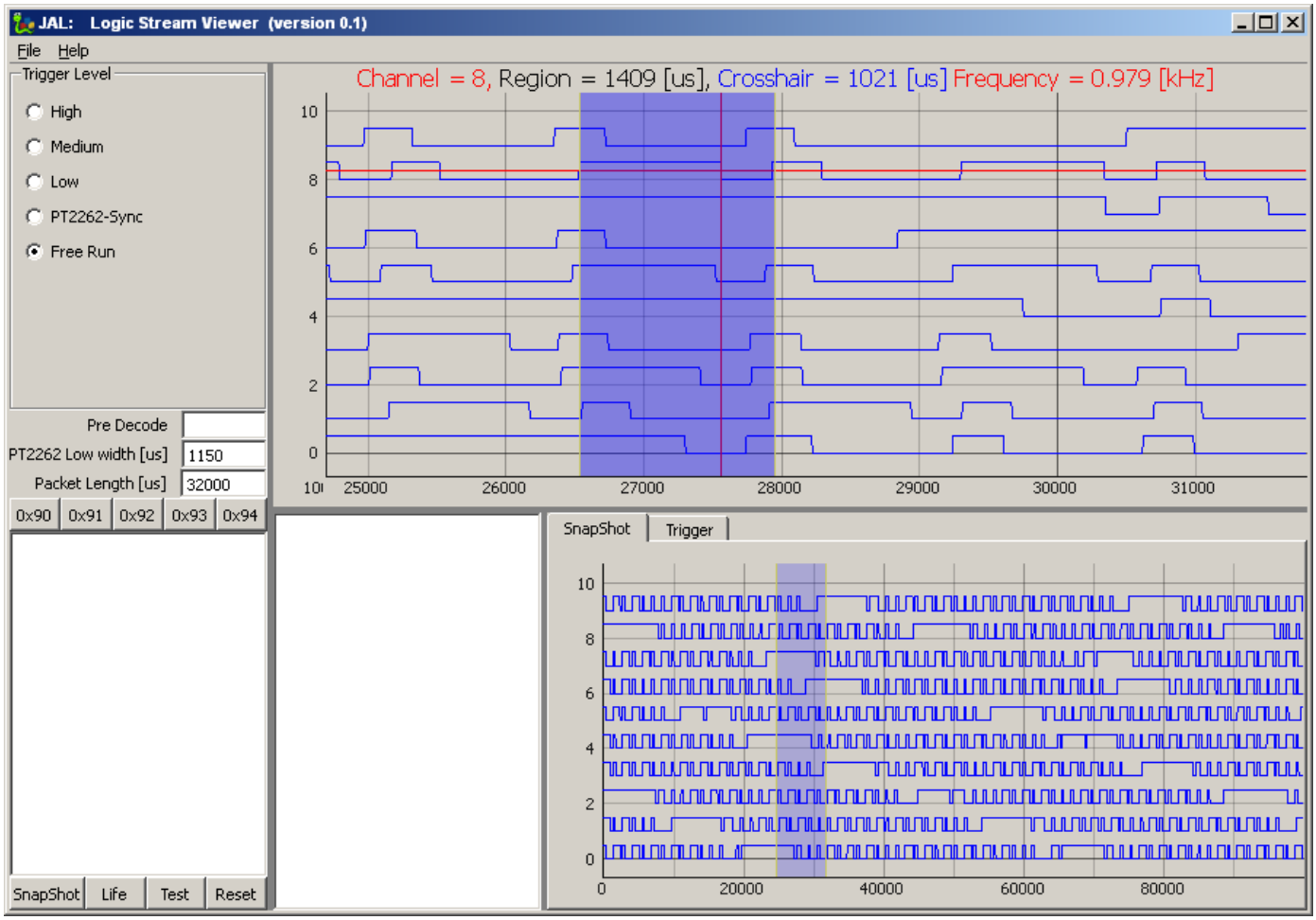
In the SnapShot window the total recording signal is shown. The blue region in the SnapShot window is the part that will be shown in the top window. The blue region in the SnapShot window can be moved (drag and drop) and resized (drag and drop the borders) and you should see the signals in the top-window move synchronuously. You can also drag and drop the top window and the region in the SnapShot window should move synchronuously.

In the Top-Window the blue region can be used to do measurements and define the trigger pattern.

Measurements

Time measurements can be done with an accuracy of 10 usec. By zooming into the signal (by shrinking the blue region in the SnapShot window) the following measurements are done□

- Region is the width of the blue region
- Crosshair is the distance of the crosshair to the left border of the blue block
- Frequency is the frequency based on the Crosshair time
- Channel is the Channel-Number of vertical position of the crosshair (needed to identify the trigger pattern)



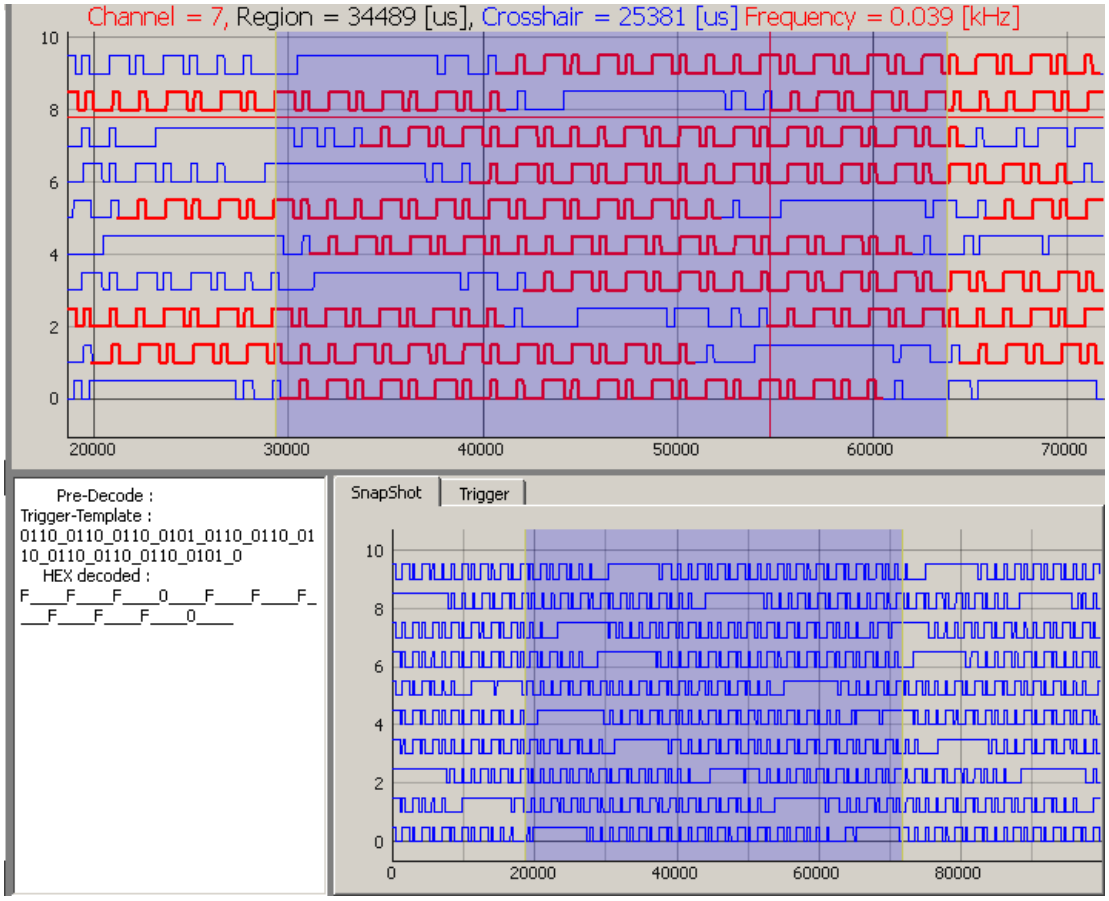
Triggering

The program can synchronize and decode the signal, either by a pattern or or by the PT2262 sync bit.
The first time you want to trigger, you should always trigger on a pattern. The reason is that when a trigger pattern is set, the total signal is analyzed and some statistics over the pulswidths are calculated.

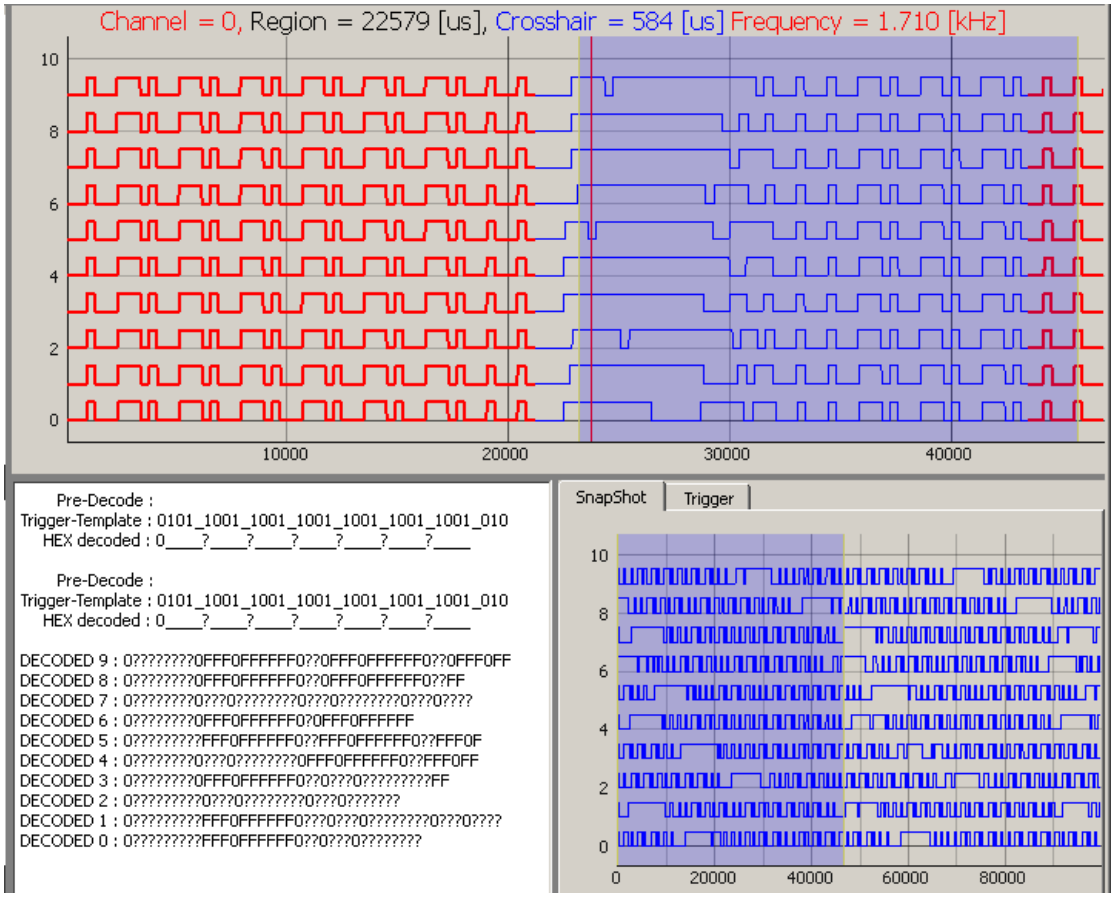
Setting the trigger pattern:

- zoom out enough, so you see a large portion of the recorded signal, at least you should see the repeating pattern in the Top-window
- Move and resize the blue region in the top window so it covers the pattern to be used as trigger/synchronisation. It's important to notice that the first and last time period will not be included in the trigger pattern, because these periods will never be complete. Also the trigger pattern will start at the first positive level. In the eaxmple below, channel 1 is selected (vertical position of the crosshair) to define the trigger pattern. So in the example below, the first half positive puls and the second negative puls will not be included in the trigger pattern.
- position the crosshair on the selected signal and double click

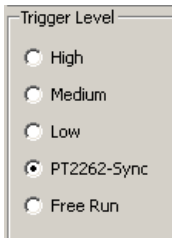
The signals will be analyzed, and with the pattern and the found statistics, all the signals will be rescanned and the found trigger templates will be highlighted. Furthermore the Trigger-Template and it's decoding is shown in the text window. More about that later.

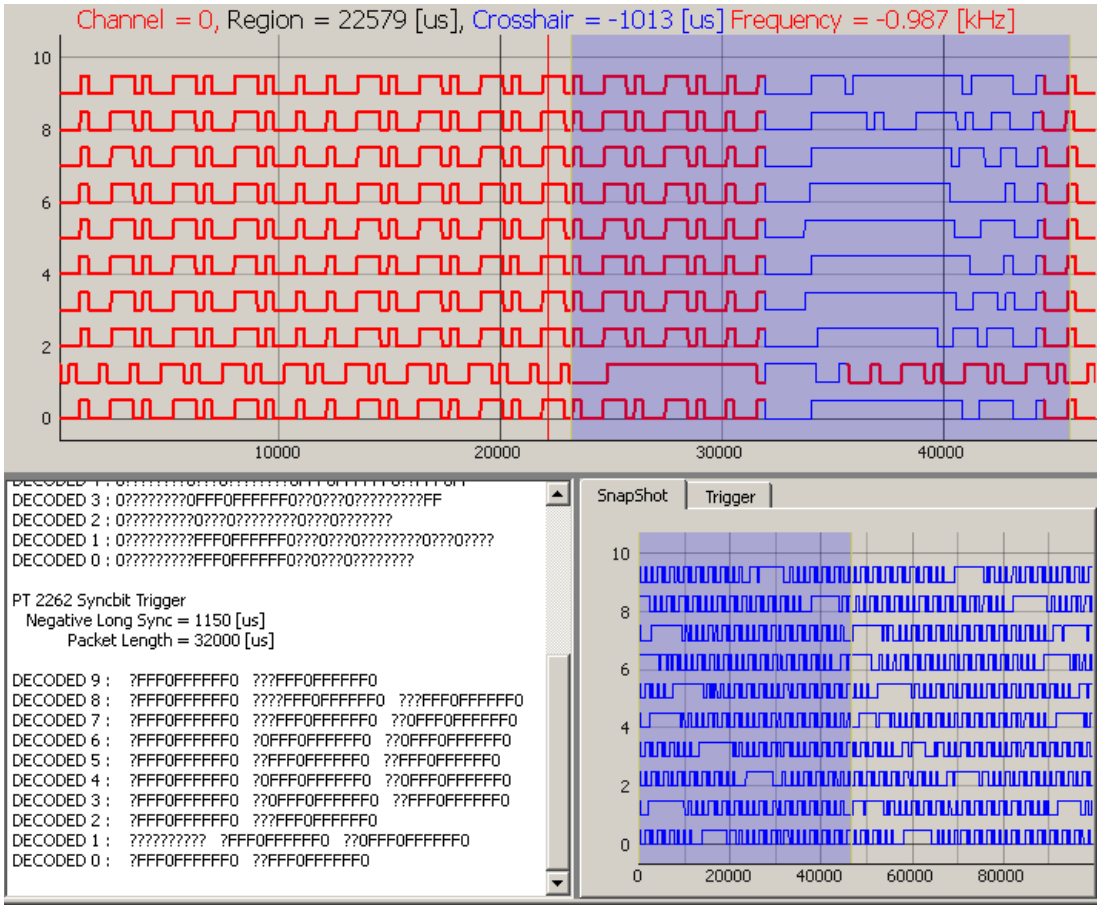


Pressing Life-button, wait till all traces are rescanned and press Freeze-button shows that the trigger template was set to a reasonable value. The incomings signals are very well synchronized, but the decoding (shown in the text window) is not very well yet.
In this case I already knew the signal at it was easy to select the right portion. In cases where this is so obvious, you have retry a few times, or .....



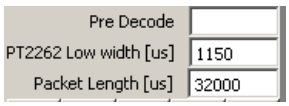
Because this is recording of a PT2262 modulation there's a very simple sync method now (after you've done at least one manual attempt to synchronize), switch to PT2262 sycn, start recording again and press Freeze again.



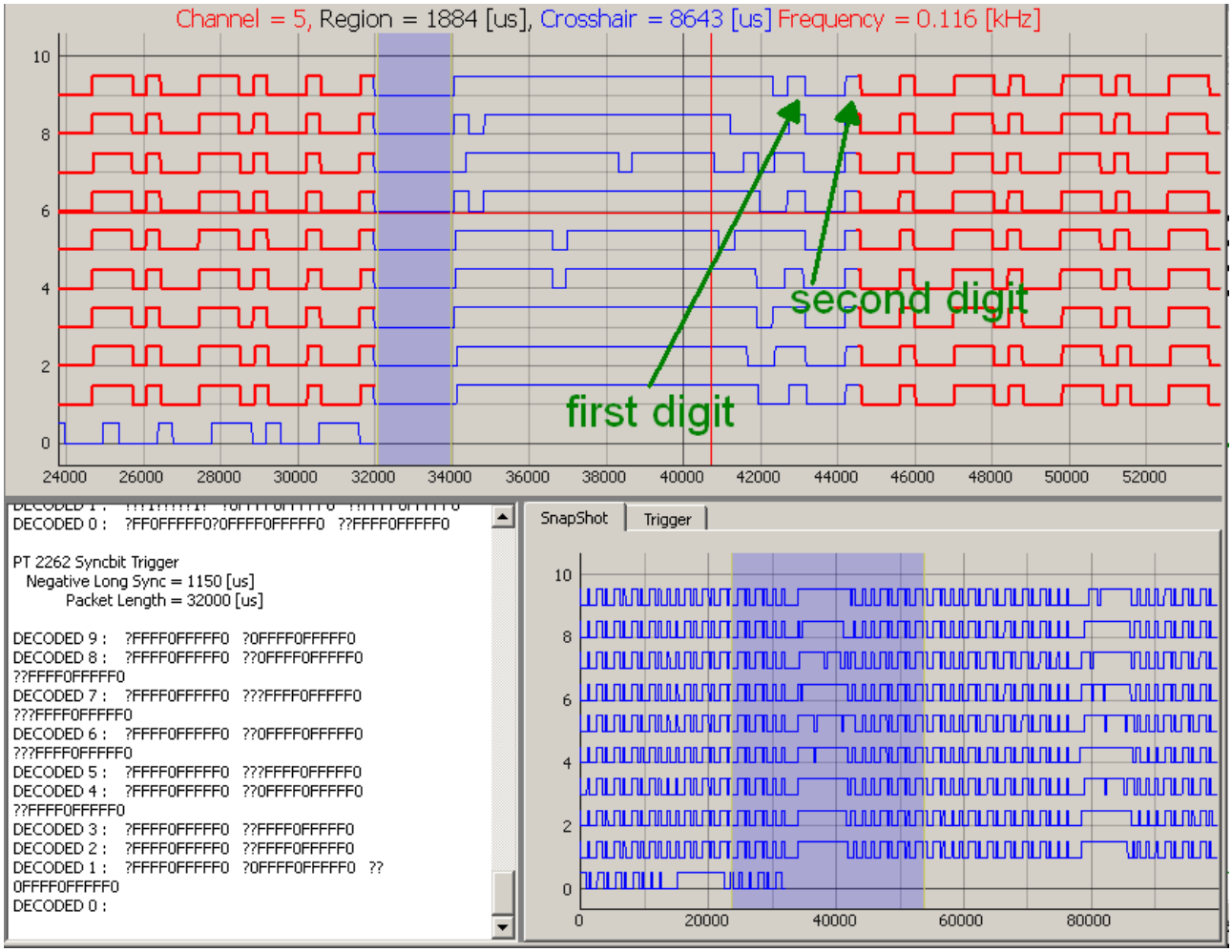


As you can see, sychronisation and decoding are almost perfect.
The transmitted code comes from a GOA handcontroller, EMW200T and button A1-Off is pressed, which sends the code : 0FFF 0FFF FFF0.
First thing to realize is that the code is a trinary code, so each digit has one of the following values: 0, 1, Float. Second thing to realize is that the sync bit (short high value followed by a long low value) is behind the databits (12 bits) and that this long puls will not be correctly detected by the SI4432 chip. This results most of the times in a distortion (and thus wrong detection) of the first bit. In the above we see a good detection of the whole packet in the third packet of signal 1 ("Decoded 1").
Note: in the above picture the SnapShot is from the previous recording (forgot to refresh it).

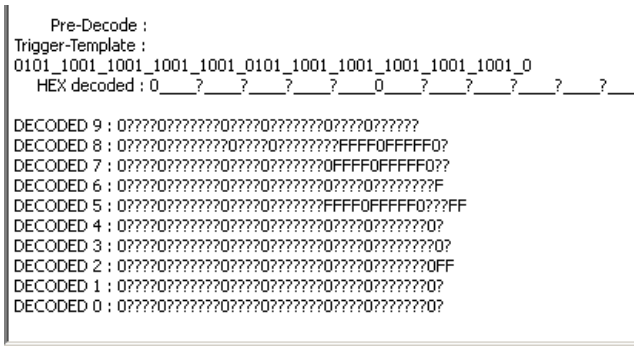
We might be able to finetune the detection, by correcting the default start values



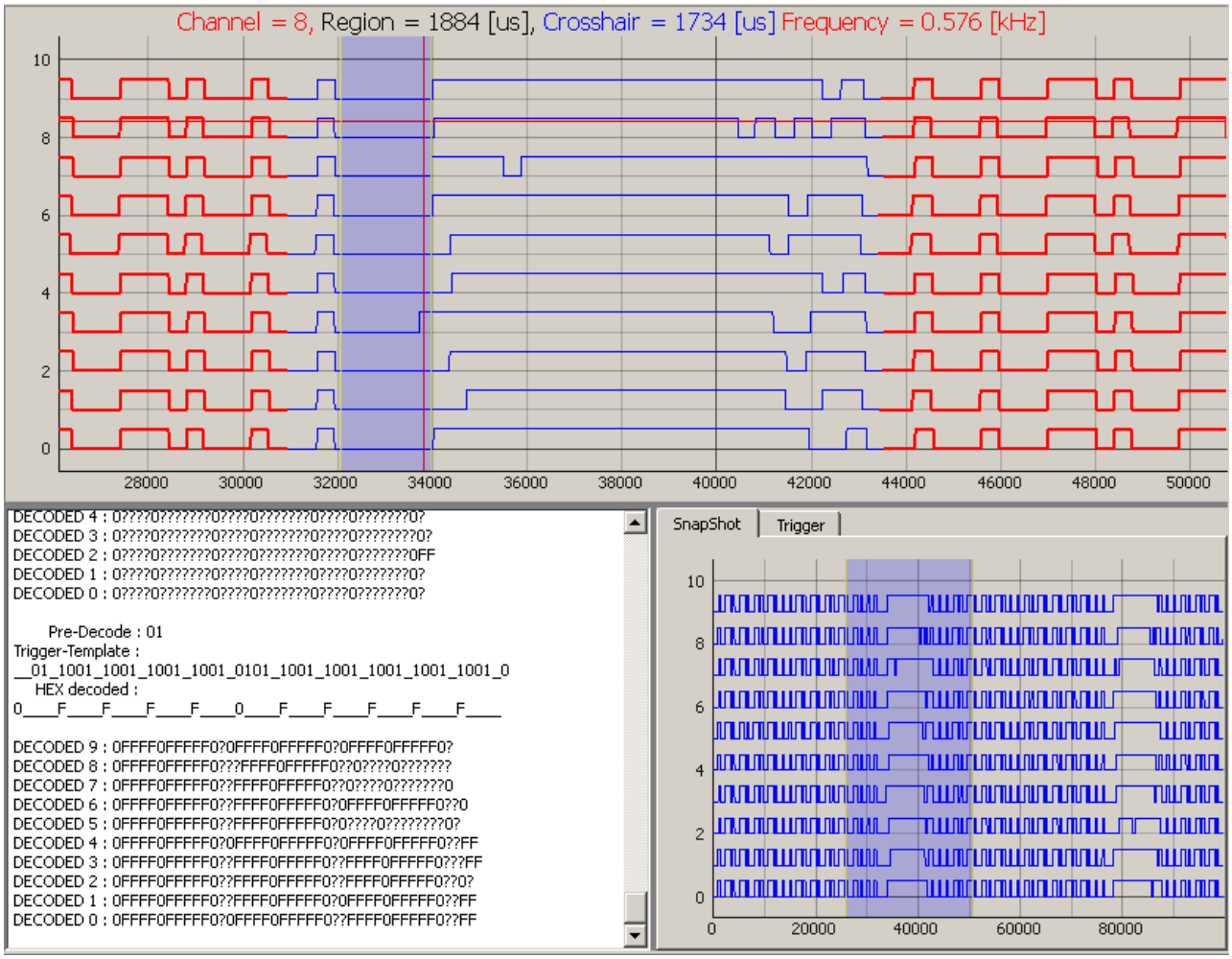
When a low level pulse longer than the PT2262 Low width is detected, this is interpreted as a trigger. So you can now zoom into the signals ans see if there might be better choice. From the picture below you might expect that 1500 would be a better choice. The packet length determines the period before the sync puls (remember with a PT2262 the sync puls is behind the data) that will be seen as signal and thus will be used as an hold off for the next trigger. From the picture below the packet length could be extended to cover the first and second digit, but you can also see that teh first digit is often lost.



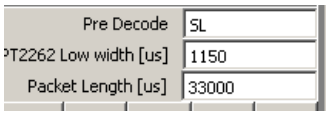
Back to normal triggering with a template. If we switch back we'll see that the decoding is a mess.



Because we mist the first digit, decoding will be out of sync. If we zoom in to the beginning of the signal we can see what we're missing:



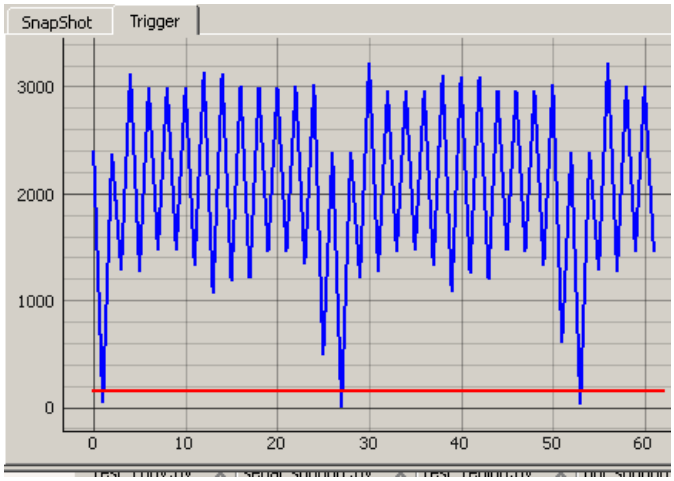
We miss the first digit, which is often distorted, so we can not extend the template but we've to manually add the first digit. In the signals from channel 0,4 and 9 we can see that we're missing a short high and long low puls (the red line here extends a little bit too far to the left). It's good to realize that triggering and decoding always start at a high puls. Now in Pre Decode you can specify the part that will be place in front of the signal before decoding it (you may use Ss0 for a short period and Ll1 for a long period.

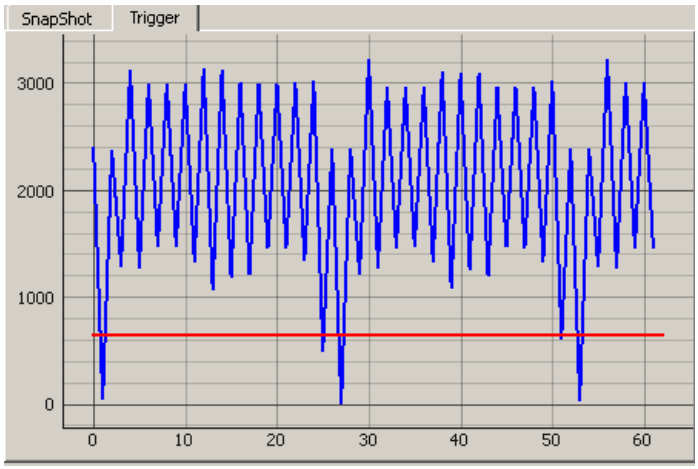


And now we get a much better result:

```
Pre-Decode : 01
Trigger-Template :
__01_1001_1001_1001_1001_0101_1001_1001_1001_1001_1001_0
HEX decoded :
0___F___F___F___F___0___F___F___F___F___F___F___

DECODED 9 : 0FFFF0FFFFF0?0FFFF0FFFFF0?0FFFF0FFFFF0?
DECODED 8 : 0FFFF0FFFFF0???FFFF0FFFFF0??0????0???????
DECODED 7 : 0FFFF0FFFFF0??FFFF0FFFFF0??0????0????????0
DECODED 6 : 0FFFF0FFFFF0??FFFF0FFFFF0?0FFFF0FFFFF0??0
DECODED 5 : 0FFFF0FFFFF0??FFFF0FFFFF0?0????0????????0?
DECODED 4 : 0FFFF0FFFFF0?0FFFF0FFFFF0?0FFFF0FFFFF0??FF
DECODED 3 : 0FFFF0FFFFF0??FFFF0FFFFF0??FFFF0FFFFF0???FF
DECODED 2 : 0FFFF0FFFFF0??FFFF0FFFFF0??FFFF0FFFFF0??0?
DECODED 1 : 0FFFF0FFFFF0??FFFF0FFFFF0?0FFFF0FFFFF0??FF
DECODED 0 : 0FFFF0FFFFF0?0FFFF0FFFFF0??FFFF0FFFFF0??FF
```

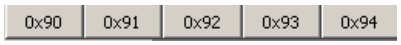
Trigger level High/Medium/Low, in general choose High. The effect can be seen (after selecting a template) in the notebook page "Trigger": in the left picture the "High" trigger level and in the right picture the the "Low". As soon as the calculated cross correlation between Signal and Template (in fact the Summated Absolute Difference is used) comes below the Red line, a trigger is detected. With this signals, a "Low" setting will result in some false triggers.





General purpose buttons

There are 5 general purpose buttons, which sends commands with codes 0x90 .. 0x94 to the PIC.



Here an example to investigate what the setting of the OOK RX Bandwidth will do. The small pulses of the signal we're investigating are about 400 uSec width, which yields a RX-Bandwidth of 2.5 kHz.

Now we modify the JAL code by setting the RX Bandwidth to some different values

```
1070 -- ***********************************
1071 -- ***********************************
1072 procedure SI4432_OOK_Scope_Loop () is
```

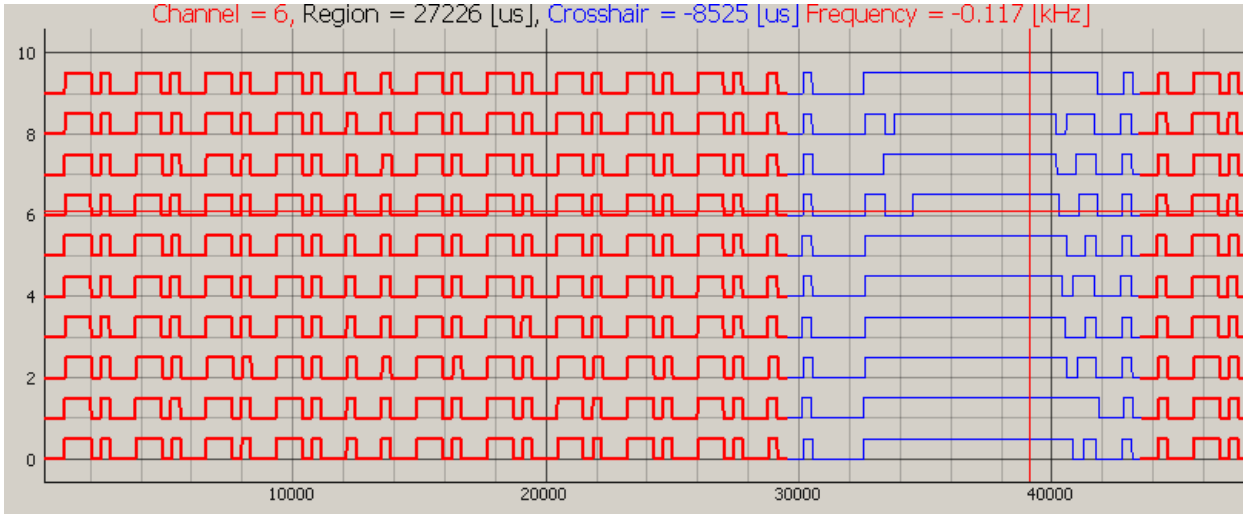
....

```
   -- The desktop program has some extra buttons
   -- X90 .. x94 which can be used for test prurposes
   -- These buttons sends code 0xFA + 0x90 ... 0x94
   if serial_hw_read ( RS232 ) then
     if RS232 == 0xFA then
       -- wait until the real databyte is received
       RS232 = serial_hw_data

       if    RS232 == 0x90 then
         SI4432_OOK_Init_Rx ( 1_900 )

       elsif RS232 == 0x91 then
         SI4432_OOK_Init_Rx ( 2_500 )

       elsif RS232 == 0x92 then
         SI4432_OOK_Init_Rx ( 3_100 )

       elsif RS232 == 0x93 then
         SI4432_OOK_Init_Rx ( 10_000 )

       elsif RS232 == 0x94 then
         SI4432_OOK_Init_Rx ( 1_500 )

       end if
```

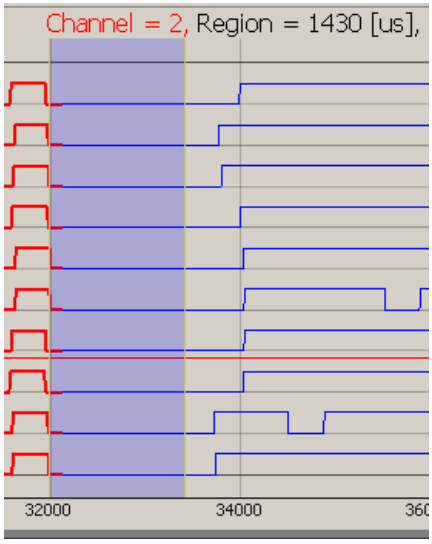
RB Bandwidths are taken as the larger value from the next range:

```
214 var word OOK_bps []  = { 1000, 2000, 3000, 8000, 16000, 32000, 50000, 65000 }
```

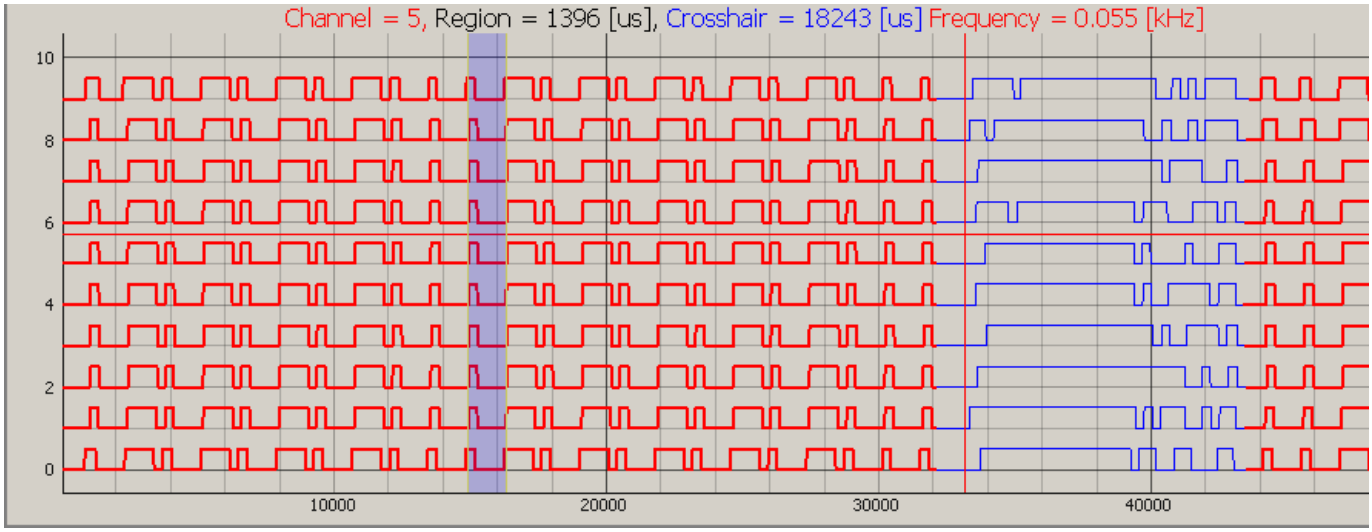
With the standard setting of 2.5 kHz we get the following picture



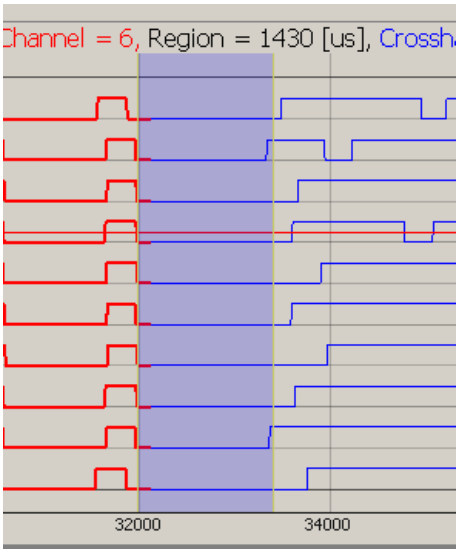
zoomed into the long low period of the sync bit:



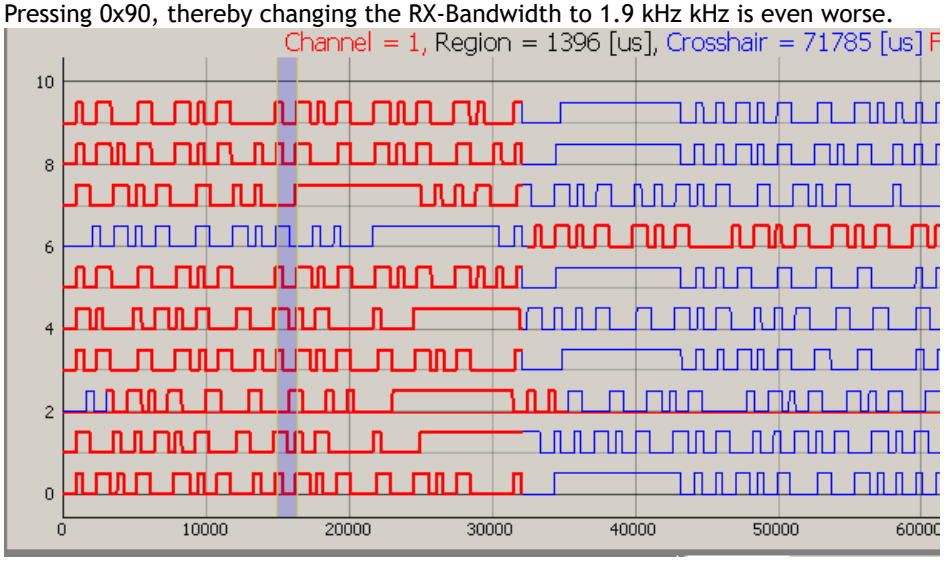
Pressing 0x92, thereby changing the RX-Bandwidth to 3.1 kHz is worse, because the long low of the sync puls is shortened



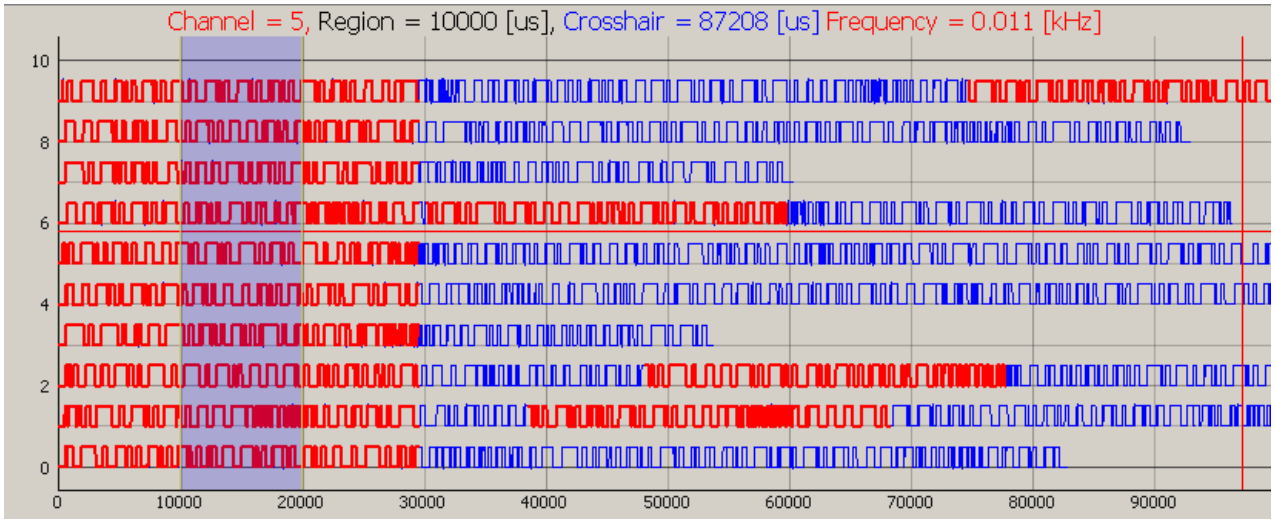
zoomed into the long low period of the sync bit:



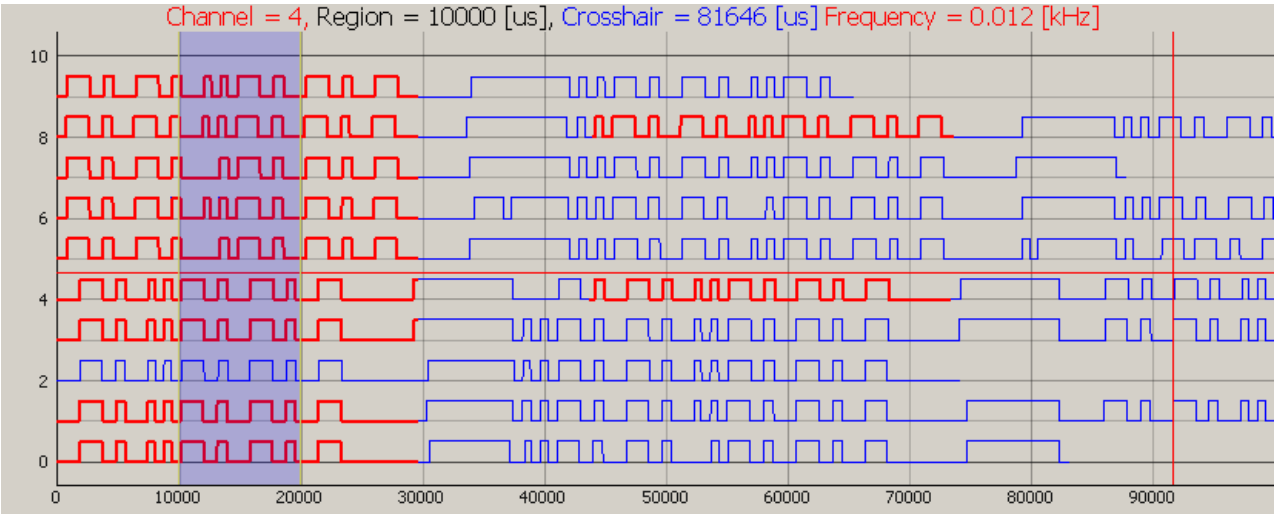
Pressing 0x90, thereby changing the RX-Bandwidth to 1.9 kHz kHz is even worse.



Pressing 0x93, setting RX Bandwidth to 10 kHz gives a terrible picture, just as might be expected: a lot of noise is being detedted as a signal.



Pressing 0x94, yielding a RX-Bandwidth of 1.5 kHz, also looks bad: a lot of small pulses are lost.

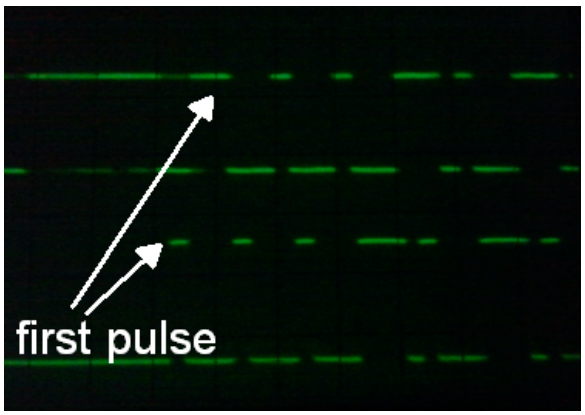


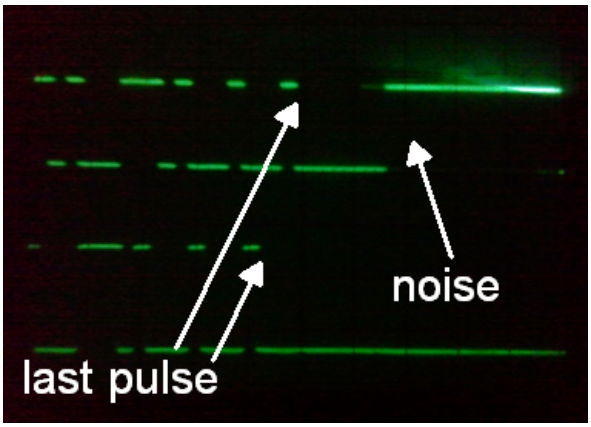
So the conclusion might be, the OOK RX-Bandwidth must be set correctly.

RX Bandwidth is set to 75 kHz (lowest value), but changing it to 600 kHz (maximum value) doesn't show any differences.

Viewing with an oscilloscope

If you've enough experience with the settings of an oscilloscope it's quiet well possible to view the long and slow signal of a remote home controller with an oscilloscope.
The bottom signal is the modulation signal that forms the input signal of the transmitter.
The top signal is the decoded raw signal from the SI4432 used as a OOK (On-Off-Keying) receiver.





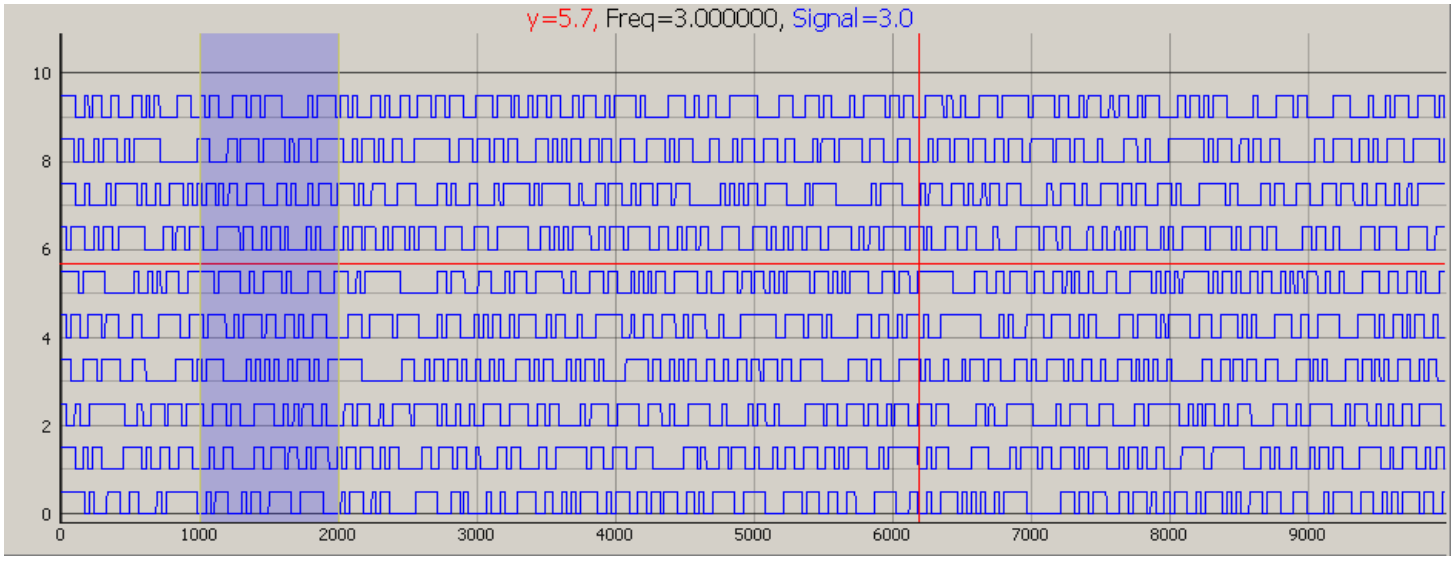
To get the pictures above the following scope settings were used:

- 10 msec/div not calibrated (manual set larger than 10 msec/div)
- x-axis zoomed 10 times
- trigger holdoff set large
- trigger on the modulation signal (the received decoded signal isn't stable enough)

The big trick is to gently tune both the timebase and the holdoff until you get picture that stands still.
If you've got the above picture the next difficult step is to gently move the x-position and read out the complete word, without loosing the count !!

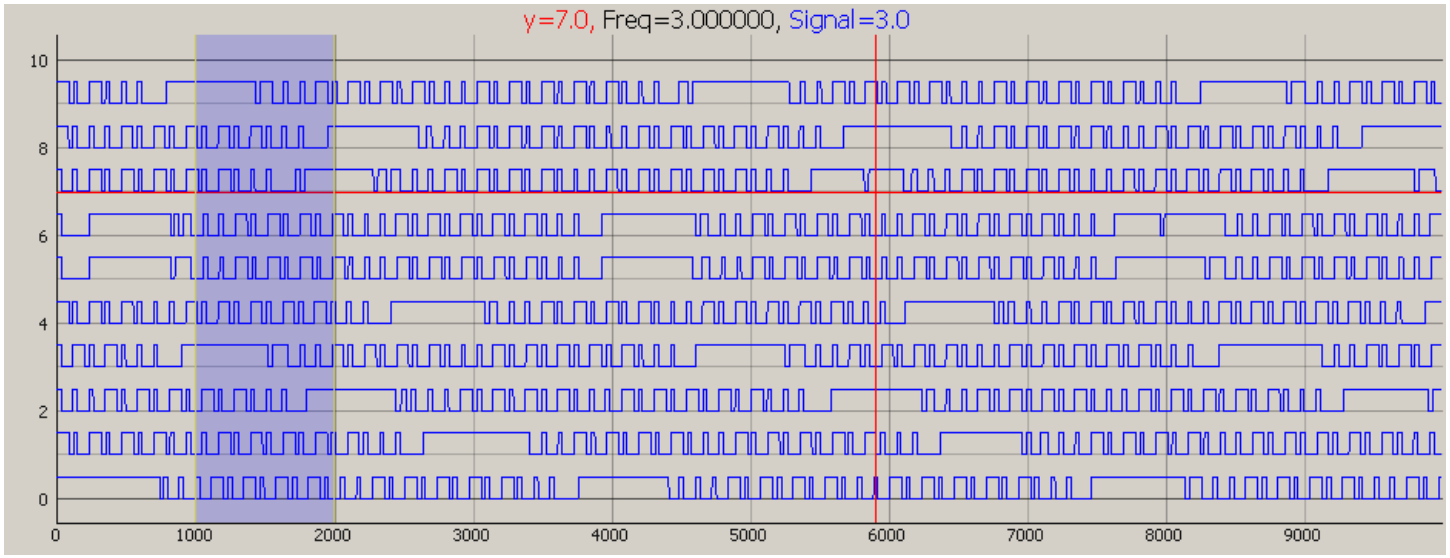
Viewing with Stream Viewer

Leave the program in the free run mode, until a complete set of channels is received, than switch to freeze mode and you should get a picture like this:



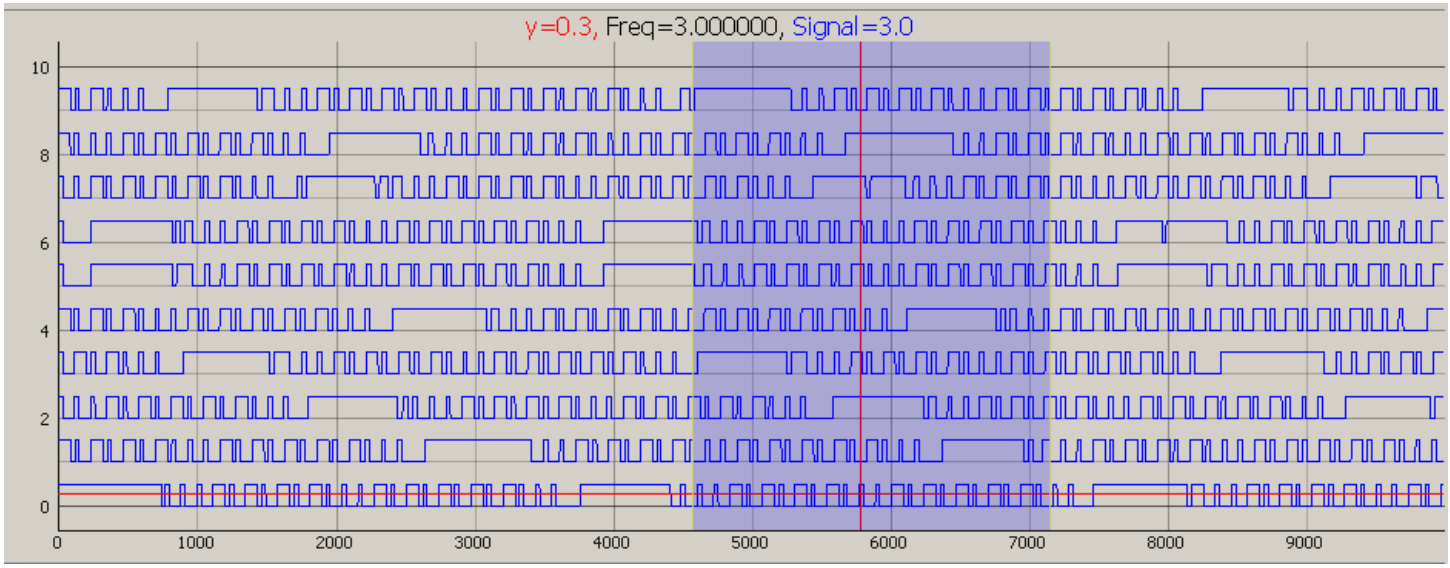
I can tell you, this is noise!
Now turn on your transmitter in continuous mode and let's restart the recording

much better, we can recognize some patterns

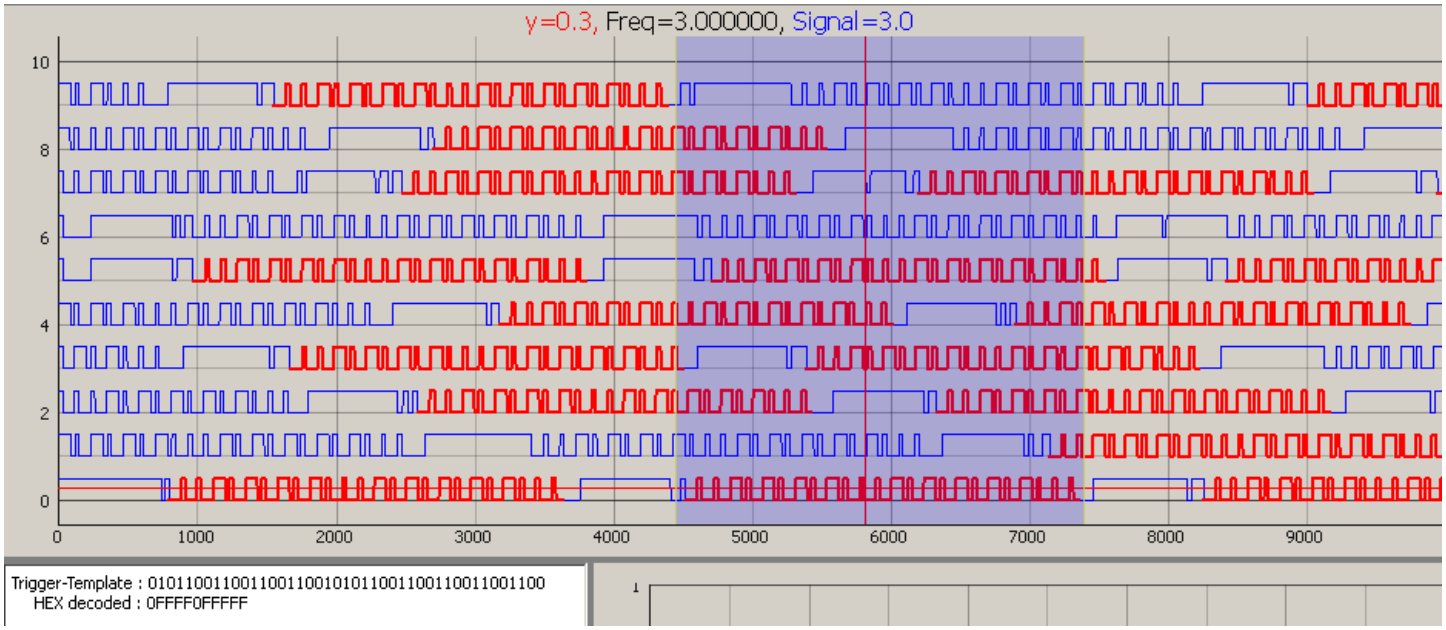


Now we need to select a part of the recorded signal as a trigger. At first sight you might to tend to use the large postive pulse as part of the trigger signal. That's not a good idea. If you look closer to the parts after the big pulse, you'll see many different things. Also you can see in a number of big pulses some dips. The point is that the transmitter sends a long Sync word which is mainly zero, so there's no RF output. The SI4432 tries to detect that non-signal, resulting in the detection of noise. The start puls of a transmission is the small positive pulse before the big pulse. So the best trigger is some part after the big pulse.

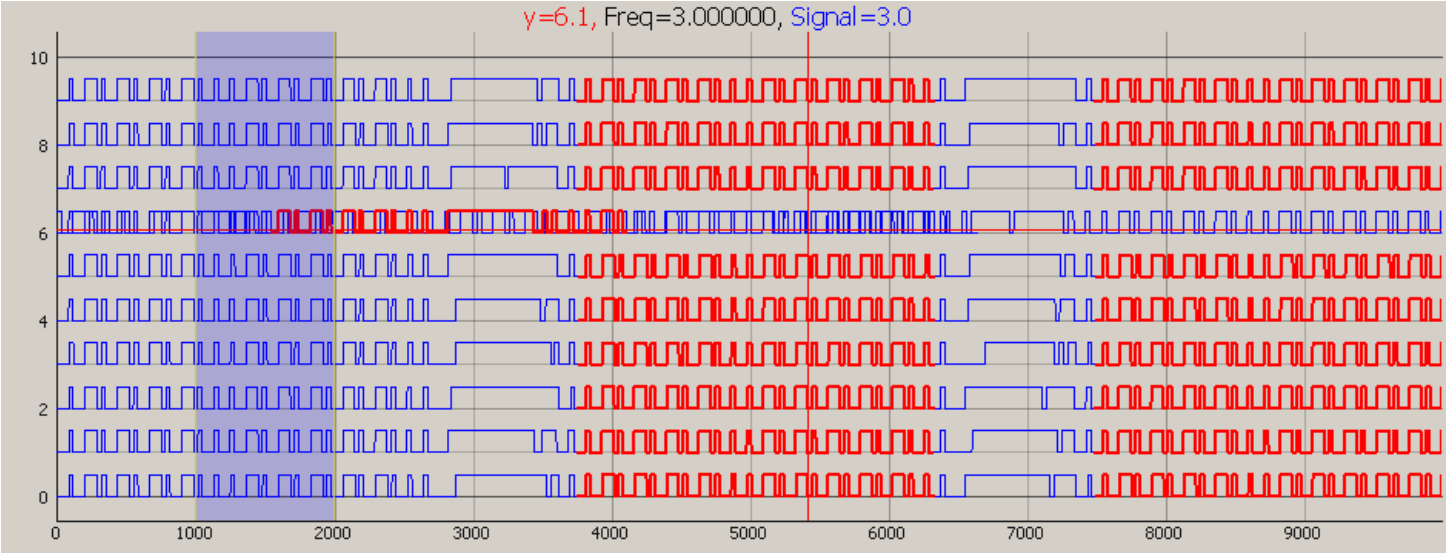
Selecting a trigger puls is done by selecting a part of one of the recorded signals. Look for an interesting part in one of the signals. Drag and resize the selection region on to this interesting part. Move the red crosshair in a vertical way on the selected signal and double-click.



The selected signal part is used as a trigger and all signals are scanned for the occurence of this trigger signal. This results in:



Most of signals are correct detected and indeed the Trigger-Template is correct. On the transmitter A2-Off is pressed which sends 0FFFF0FFFFF0. Note that this is a trinairy code, a digit can 0, 1 or F (Float).
Now the trigger is set well enough, start scanning again an after freeze you'll get

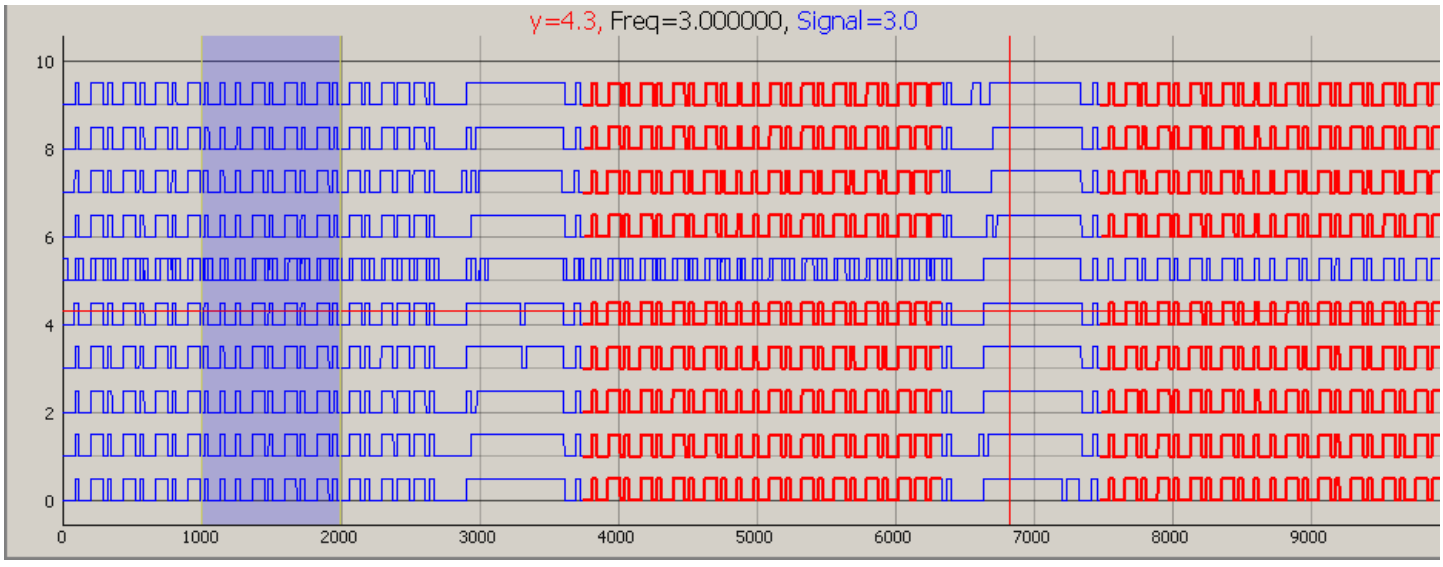


We freezed here in the middle of scanning channel 6, so that's a mess (might be improved later).
The other signals are well in sync and all occurences are well detected (the first packet is not marked, we might improve that later).
Now it's good to see that the big pulse is realy detected as a lot of noise.
On freezing also the signals are decoded:

```
HEX 9 : 0FFFF0FFFFF0??FFFF0FFFFF0?0????0???????0????0??
HEX 8 : 0FFFF0FFFFF0??0????0???????0????0???????0???
HEX 7 : 0FFFF0FFFFF0??FFFF0FFFFF0?0????0???????0????0???????
HEX 6 : 0???0FFFF0FFFFF0?0????0
HEX 5 : 0FFFF0FFFFF0?0????0???????0????0????????0????0
HEX 4 : 0FFFF0FFFFF0??FFFF0FFFFF0??FFFF0FFFFF0?0?
HEX 3 : 0FFFF0FFFFF0?0FFFF0FFFFF0??0????0???????0????0??
HEX 2 : 0FFFF0FFFFF0?0????0???????0????0???????0???
HEX 1 : 0FFFF0FFFFF0??FFFF0FFFFF0??FFFF0FFFFF0?0
HEX 0 : 0FFFF0FFFFF0??FFFF0FFFFF0??FFFF0FFFFF0??0????0??????
```

The first packet is decoded well (except in channel 6, but that was to be expected).
Within a channel the next packet is never detected correctly, this is caused by the big pulse that messes up the synchronisation.

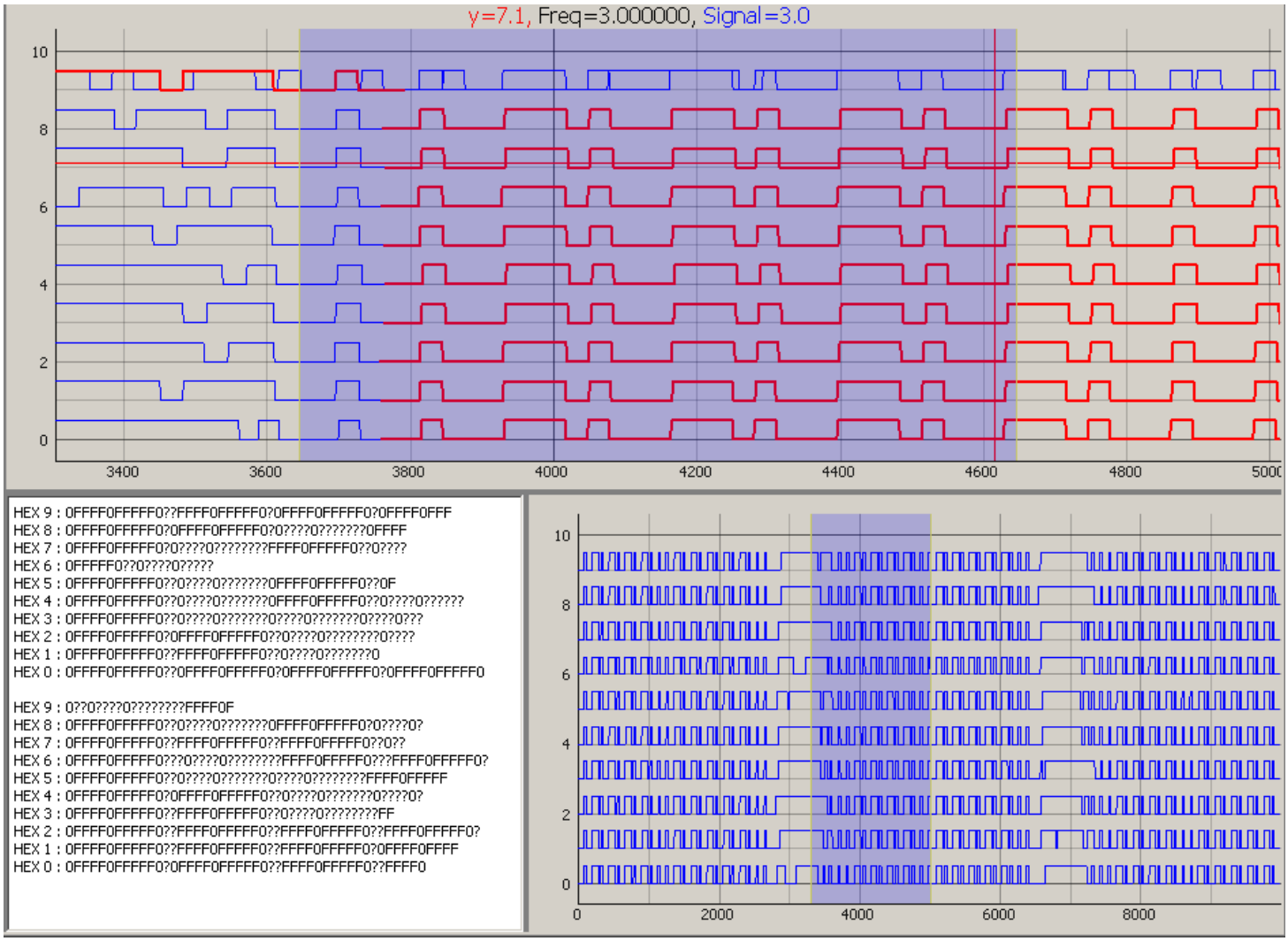
As we selected for the trigger the complete packet minus the last digit we should be able to detect with the same trigger also A2-On button (sending 0FFFF0FFFFF1):



which can be easier interpreted by the Hex code:

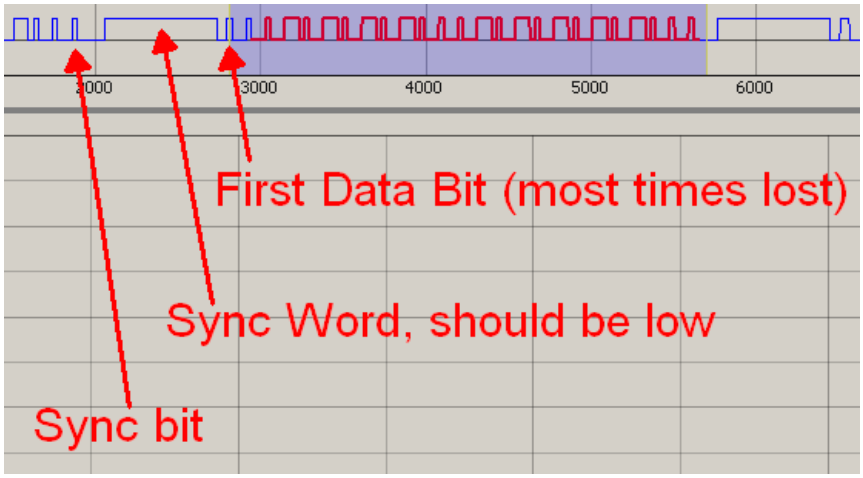
```
HEX 9 : OFFFFOFFFFF1?O????O????1??O????O????1??
HEX 8 : OFFFFOFFFFF1??FFFFOFFFFF1?O????O????1??O
HEX 7 : OFFFFOFFFFF1??O????O????1??FFFFOFFFFF1?O?
HEX 6 : OFFFFOFFFFF1?O????O????1??O????O????1???FF
HEX 5 : OFFFFOFFFFF1??FFFFOFFFFF1
HEX 4 : OFFFFOFFFFF1??FFFFOFFFFF1?O????O????1??O????O????1
HEX 3 : OFFFFOFFFFF1??FFFFOFFFFF1?O????O????1??O????O????1?
HEX 2 : OFFFFOFFFFF1??FFFFOFFFFF1?O????O????1??FFFFOFFF
HEX 1 : OFFFFOFFFFF1?O????O????1??O????O????1??O????O????1??
HEX 0 : OFFFFOFFFFF1?O????O????1??O????O????1??O????O????
```

And if wnat to see more detail, you make a snapshot and zoom into the signals



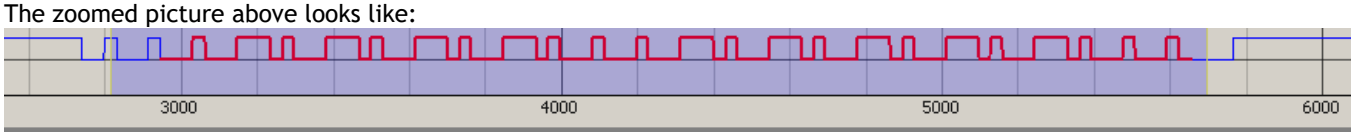
Decoding

here we see an almost perfect decoded signal from the GOA ....
The Sync Word is not well decoded by the SI4432, has the tendecy to stay high.
The first data bit is often lost and eaten by the sync word.



In the above picture the trigger is the selected region (red signal).
The display above has a small defect, the trigger always starts at a High signal level and here the red marking starts just too late.

The zoomed picture above looks like:



In the figure below the decoding of the trigger signal is given:
line 1: Trigger, the width of each period, 0 = Short, 1 = Long
line 2: Time, same as Trigger, but S = Short, L = Long
line 3: High/Low, the level of the signal (Trigger alwqasy starts with a "1" high level)
line 4: Decoded, the SC2262/2272 decoding

```
 1 Trigger  =   01 0110 0110 0110 0110 0101 0110 0110 0110 0110 0110 0101 0
 2 Time     = SLSL SLLS SLLS SLLS SLLS SLSL SLLS SLLS SLLS SLLS SLLS SLSL S
 3 high/low = 1010 1010 1010 1010 1010 1010 1010 1010 1010 1010 1010 1010 1
 4 decoded  =  0    F    F    F    F    0    F    F    F    F    F    0
```

So we'll always miss the first high and low segment.
The very last level of the Trigger puls above is the Sync bit of the next transmission.
If we add in front of the trigger the missing "10" (Short/Long) we can decode the received signal as shown in line 4.
If we compare the received byte with the expected packet (shown in the figure below), we see that we here receive a A2-off signal, which is correct.

```
 6 decoded  =  OFFF FOFF FFF0
 7 A2-Off   =  OFFF FOFF FFF0
 8 A2-On    =  OFFF FOFF FFF1
```

History

october 2014: **Version 0.1**, initial release

ToDo / Future ideas

- keep the region in the topwindow visible while the region in the SnapShot window is moved or resized
- improve the triggering in PT2262 mode by looking at the length of the recorded signal
- RED curve that shows the curve in normal trigger, somewhat shifted to te left
-

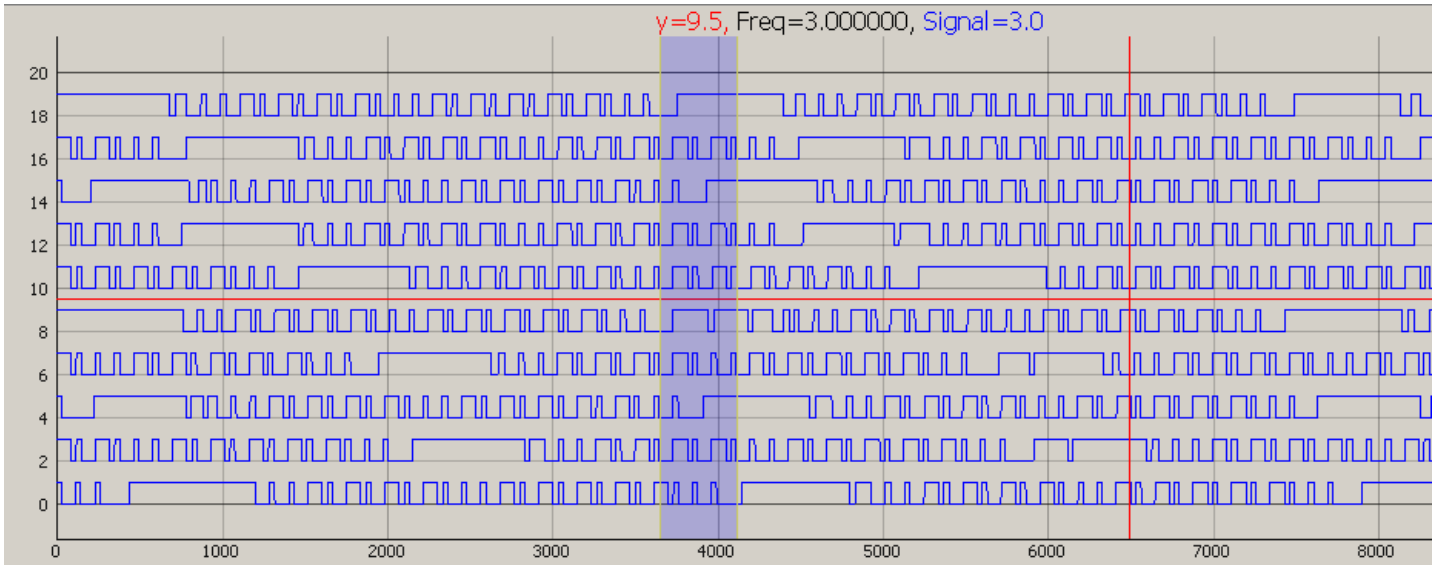
What you need

- PIC, working at 3V3, with SPI and Serial port (at the end of this paragraph the memory requirements for the PIC are given)
- wireless module 470M (RFM22b probably also works)
- JAL-program,...link....
- Python program
- Python compiler
- Computer with a serial port (or USB with virtual Comm Port)

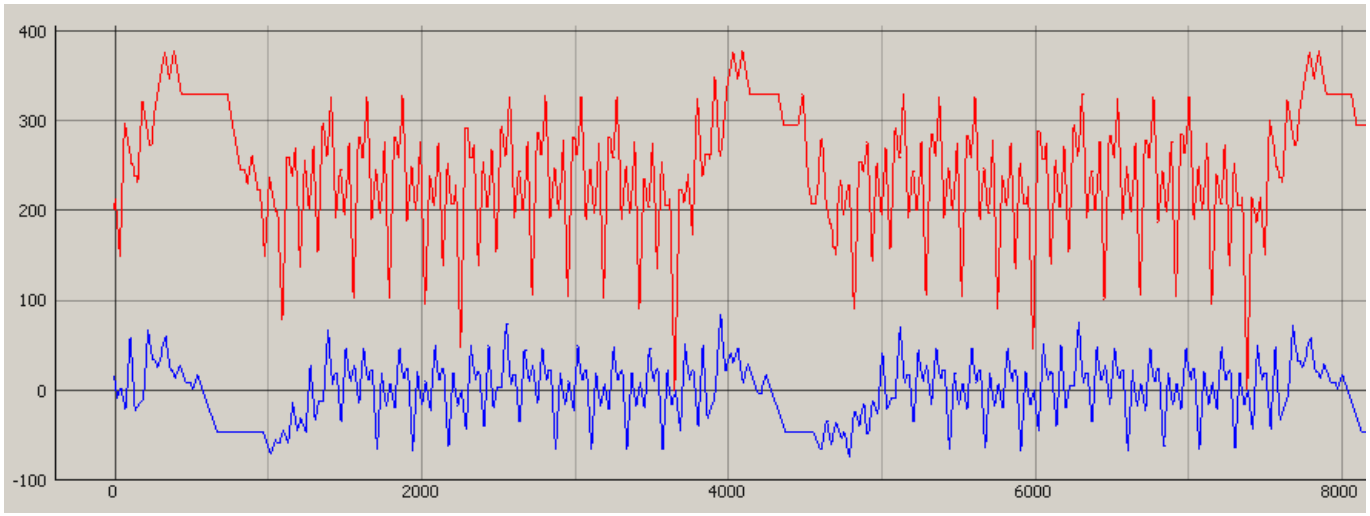
PIC Memory requirements:

```
Code area: 8984 of 65536 used (bytes)
Data area: 736 of 3840 used
Software stack available: 3104 bytes
Hardware stack depth 5 of 31
```

Trigger detection design

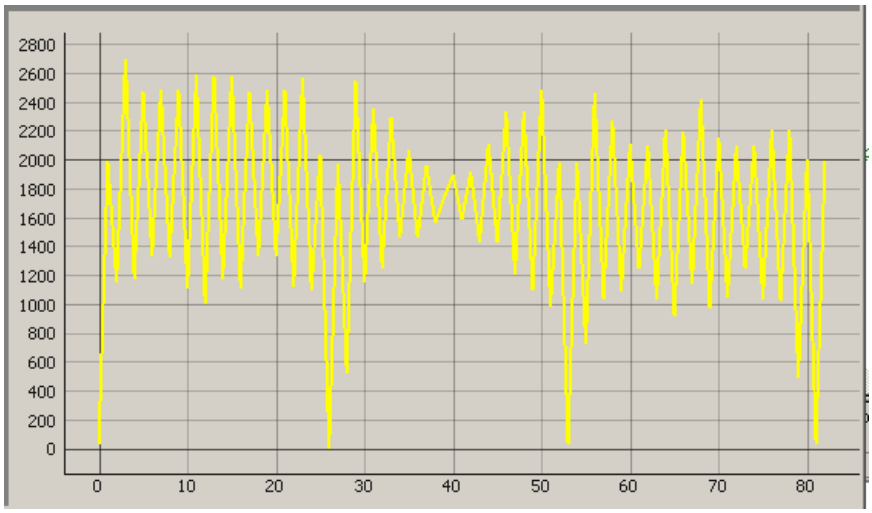


The trigger signal is the selected region, channel 0 is choozen.



The blue curve is the normalized, zero-mean cross correlation between trigger signal and the signal at channel 0.
The red curve is the sum of the absolute difference between signal and trigger.
The sum of the absolute difference is a much better trigger-finder than the correlation. But as there's no fast implementation of this function available, it becomes realy too slow.

Therefor we tried another trick, correlation on the raw signal, which a sequence of times, representing the duration of each bit. zero-mean cross correlation gives about the same result as above. The sum of absolute differences looks great, even better than performed on the final signal itself. The speed is also acceptable (just need to calculate for about 100 point instead of 10,000).



GOA transmit codes

The following table is derived with Stream-View.
Note that the first digit often is distorted.

Switch	On			Off		
A1	0FFF	0FFF	FFF1	0FFF	0FFF	FFF0
A2	0FFF	F0FF	FFF1	0FFF	F0FF	FFF0
A3	0FFF	FF0F	FFF1	0FFF	FF0F	FFF0
B1	F0FF	0FFF	FFF1	F0FF	0FFF	FFF0
B2	F0FF	F0FF	FFF1	F0FF	F0FF	FFF0
B3	F0FF	FF0F	FFF1	F0FF	FF0F	FFF0
C1	FF0F	0FFF	FFF1	FF0F	0FFF	FFF0
C2	FF0F	F0FF	FFF1	FF0F	F0FF	FFF0
C3	FF0F	FF0F	FFF1	FF0F	FF0F	FFF0
D1	FFF0	0FFF	FFF1	FFF0	0FFF	FFF0
D2	FFF0	F0FF	FFF1	FFF0	F0FF	FFF0
D3	FFF0	FF0F	FFF1	FFF0	FF0F	FFF0