SI4432 OOK Detection

4 August, 2020
0:11

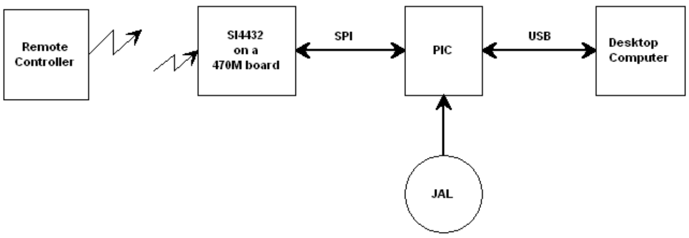
JAL: SI4432 OOK Detection

last updated: oct. 2014, SM

Introduction

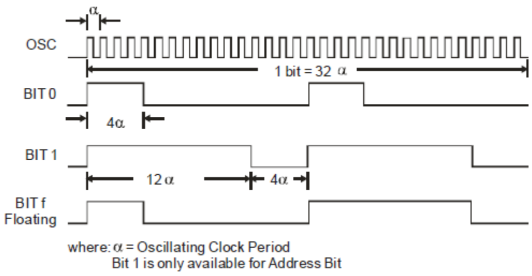
On-Off-Keying (OOK) is a very simple modulation techniek used in infrared control and many (cheap) remote home control systems like lamp controls, door openers etc. Most of the remote control units works with a carrier frequency of 434 MHz or 868 MHz.

The SI443 is a cheap RF transceiver (both sender and receiver) specially meant for packet transmission in the UHF-band (200 MHz ... 900 MHz). This relative low frequency, compared to the popular 2.4 GHz band, has the advantage of a longer range and better passing through obstacles like walls. Although the SI4432 is not well suited for raw OOK transmission, experiments shows that it's quiet well possible to build a reasonable reliable OOK receiver with a PIC and some JAL code. The JAL library "SI4432_support.jal" contains all necessary functions. This package also contains a number of desktop programs to view and analyse the signals.



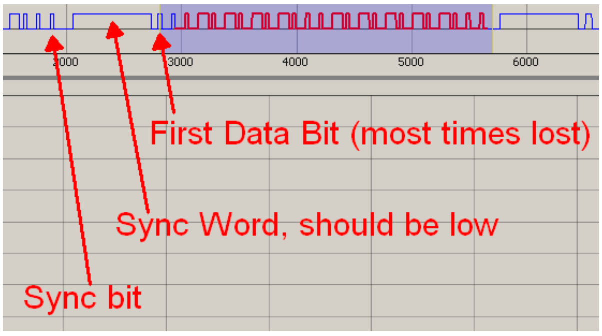
The modulation signal

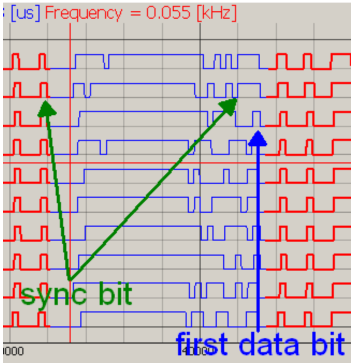
Often a PT2262 (SC2262 is equivalent) is used as the encoder to generate the modulation signal. The PT2262 sends a 12 digit, trinary coded, word followed by a sync bit. Normally a sync is in front of the data but here the sync is after the data !! Trinary code means that each digit can have one out of three values: zero, one and float. The MSB of the 12 digit word contains the address and the LSB contains the data. At least 6 of the digits needs to be the address, and the address can extend to 11 digits. Of course, the more address bits, the less data bits. The difference between address digits and data digits becomes clear when using the standard decoder (PT2272 or SC2272), the address digits are only used as inputs for comparison of the incoming signal, while data digits can be used both as an input or as an output with a logic zero or one.



Detection

The SI4432 is basically a packet transceiver. Therefor it needs a preamble (a long 101010101.....) to get a stable setting of the RF-receiver, demodulator and detector. Accordinging the datasheet of the SI4432, even in raw OOK-mode the preamble is required. Experiments shows that with the right settings, it's reasonable well possible to receive and detect raw OOK-signals.





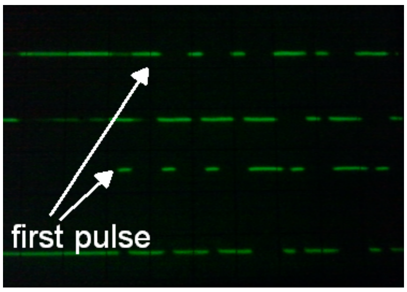
In the above picture the two green arrows identifies the beginning and the end of the sync bit (send after the data word). As can be seen from the above picture the syc bit is not as long low as it should. Moreover the first data-bit (identified by the blue arrow) is often eaten by the sync bit of the previous transmission. So the data is received almost undistorted, expect for the first digit, which is often eaten by the sync bit of the previous transmission.

Getting the Carrier Frequency

First thing to determine is the carrier frequency.
The easiest way is to look into the documentation of the remote control, because the cariere frequency should always be mentioned and should also be mentioned on the controler.
You could also use the spectrum analyzer to measure or verify the carrier frequency.
http://mientki.ruhosting.nl/data_www/raspberry/doc/spectrum_analyzer.html
If you use the spectrum analyzer you should realize that the transmitters of these controlers are not very well designed at that you'll find probably also a response on the double frequency.

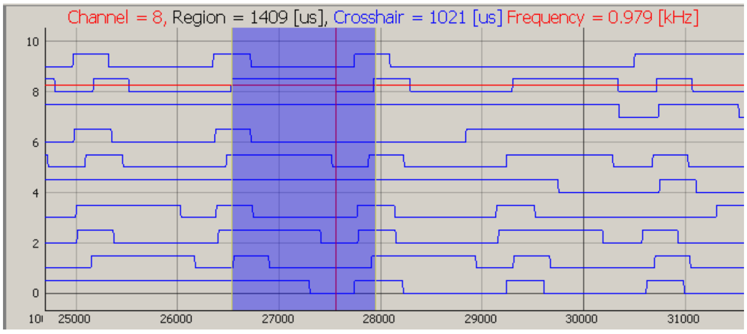
Getting the Baudrate

Probably the Baudrate can not be found in the manual.
You either have to measure the Baudrate with an oscilloscope. In the documentation of the spectrum analyzer you can how it's measured for the EMW200T controler.



or with some trial and error using the stream-viewer (using the 0x90 .. 0x94 buttons to change the Baudrate.
The available baudrates in the SI4432 are:

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



For both oscilloscope measurements of stream-viewer measurements, the Baudrate is the inverse value of the smallest pulse (In stream viewer the distance between left site of blue region and crosshair is automatically transfered to a frequency, red value on the right).

The right settings

When you know both the Carrier Frequency and the Baudrate, you know enough to choose the right register settings for the SI4432. There are several ways to find the right register settings:

- if Carrier Freqeuncy = 434 MHz (433.92 to be exact) and your Baudrate = 2.5 kHz, there's a ready settings procedure in the support library, called "SI4432_Init_OOK_Rx_434_2_5"

```
procedure SI4432_Init_OOK_Rx_434_2_5 () is
```

- There is a special OOK-receive-initialisation procedure that calculates in JAL all the needed register settings. The disadvantage is that this procedure does alot of havey calculations and therefor uses a lot of memory

```
procedure SI4432_OOK_Init_Rx ( dword in Rb_Hz, dword in BW_Hz ) is
```

- Use the SI-443x-Excel-sheet to calculate the values and manual copy past all values to JAL
- Use the SI-443x-Excel-sheet and SI4432-Register-Viewer and let the Register-Viewer generate the JAL code.

The last method is thé prefered method, because it not only sets the internal settings of the SI4432 but also the interrupts and IO pins of the SI4432 and the PIC, see for further information:

http://mientki.ruhosting.nl/data_www/raspberry/doc/si4432_register_viewer.html

The JAL Function

The JAL procedure

```
procedure SI4432_Detect_OOK () is
```

continously polls the raw data pin of the SI443, detects the duration of the low and high periods and stores the measured times of each period. When a sync-bit is detected and at least 11 data digits are received, the result is send to the serial port (USB).

This procedure has some static settings, to switch between test- and operate-mode and to choose the final high-level decoding.

```
const Do_Timing      = False   ; False = normal operating mode
                               ; True = output raw timings
const Decoded_Output = 1       ; 0 = output short(0) / long(1)
                               ; 1 = output readible 0-1-F code
                               ; 2 = output real data digits (ToDo)
```

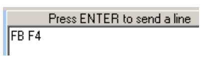
These settings are static, i.e. they are evaluated during compilation, and therefor garantee the least use of program and data memory.

JAL Function in Timing Mode

Set Do_Timing = True

Start any Serial Comm Port Viewer on your desktop.

Send the command "FB F4" to the PIC.



Now the PIC will initialize the settings for OOK-receiving, sends all the register settings to the terminal (surrounded by "AA BB CC"

```
                                    HEX received
AA BB CC
08 06 21 20 12 00 00 07 00 7F 06 1F 1D 14 00 00
00 00 20 00 03 00 01 00 00 01 14 00 BE 40 0A 00
58 40 36 9D 10 39 C4 1E 00 00 00 00 18 BC 2B 08
8D 20 0C 22 08 2A 2D D4 00 00 00 00 00 00 00 00
00 00 00 FF FF FF FF 00 00 00 00 FF 08 08 08 10
00 00 DF 52 20 64 00 01 87 00 09 94 0E E7 83 E7
A0 00 E4 00 00 FE 62 FF 13 73 9D 00 01 18 0A 3D
24 21 20 00 00 53 62 00 19 00 00 03 37 04 37 9F
AA BB CC
```

en starts the OOK-detector

```
0 9 114 26 27 29 85 188 29 54 55 26 28 26 89 47 131 26 27 24 28 103 28 27 115 26 28 26 56 26 29 85
115 29 28 27 28 82 169 55 29 240 56 26 86 27 27 27 85 61 85 27 59 27 29 28 28 27 174 49 109 55 27 27
144 196 56 191 81 26 59 83 28 27 27 52 112 134 58 108 27 58 29 52 28 28 57 26 172 84 28 112 57 55 54
159 136 84 26 84
52 51 30 187 113 78 114 27 28 137 55 28 57 54 29 82 28 81 111 134 197 26 27 82 28 53 59 28 27 164 84
26 57 26 29 78 30 131 113 25 29 81 56 27 30 83 53 137 27 52 87 82 85 26 82 188 113 28 57 26 29 30
30 53 56 26 28 82 55 59 136 54 28 53 28 159 29 53 194 26 27 28 27 26 57 83 55 27 28 139 28 55 59 28
56 136 28 28 30 27
0 11 58 109 82 82 28 29 28 139 135 27 28 136 27 26 57 27 59 56 144 109 56 54 57 81 29 28 28 25 57 26
57 109 118 54 235 51 27 163 114 78 28 81 26 85 26 27 56 28 85 54 28 54 28 26 28 25 58 27 28 54 29 138
140 109 143 136 29 27 27 27 27 27 147 25 28 109 87 112 52 82 27 56 145 53 57 135 28 159 137 80 56 27
145 218 28 81 28 108
0 1 58 57 51 82 26 26 86 82 198 193 30 54 85 26 27 81 113 27 55 28 26 82 26 56 115 166 28 28 28 28
145 26 28 107 26 137 84 78 255 25 115 26 29 26 28 195 85 110 83 79 28 191 115 28 26 106 28 27 56 220
114 54 27 137 82 105 176 82 115 25 27 54 57 27 142 214 57 54 55 82 28 28 167 168 57 26 29 111 27 82
28 136 59 28 27 28 56 27
```

what you see here above is noise.
Now set the remote controler in continous mode

If possible set your serial-terminal on decimal. You should get a image like this on your terminal:

```
0 32 28 25 75 71 27 26 74 24 76 24 255 54 54 132 27 53 77 27 75 25 75 72 28 26 74 71 28 25 75 71 27
26 74 71 28 25 75 24 75 25 76 71 28 26 74 71 28 25 75 71 27 25 75 71 28 25 75 71 27 25 75 24 76 24
255 0 29 28 140 131 75 24 75 24 76 71 29 25 75 71 27 25 75 71 28 25 75 71 27 26 74 24 76 24 76 72 28
25
15 255 240 255 255 240 5 25 255 81 58 27 112 45 75 23 75 25 76 71 28 25 75 71 28 25 75 71 28 25 74 72
27 25 74 25 75 25 75 72 28 25 75 71 27 25 75 71 28 25 74 72 28 25 74 24 76 24 255 0 28
106 110 25 76 25 76 24 76 72 27 26 75 71 27 25 75 71 28 25 75 71 28 25 74 24 76 25 75 72 27 26 75 70
28 25 75 71 28 25 75 71
15 255 240 255 255 240 52 83 85 70 75 24 75 25 74 72 28 26 74 71 27 26 74 71 28 26 74 71 28 25 75 24
75 25 75 72 28 25 75 71 28 25 74 71 28 25 75 71 27 26 74 71 28 25 75 24 75 25 255 54 30 29 81 25 29
49 75 25 75 26 75 71 29 25
75 71 27 25 75 71 28 25 75 71 27 25 75 24 76 25 75 71 29 25 74 72 27 25 75 71 27 26 74 71 28 25
15 255 240 255 255 240
76 52 27 77 76 24 76 25 75 72 28 26 74 71 28 25 75 71 27 25 75 71 28 25 74 24 76 25 76 71 27 26 75 71
27 26 74 71 28 25 75 71 27 26 75 71 27 25 75 23 76 25 255 52 57 28 54 130 75 24 76 25 75 72 28 25 74
72 28 25 74 72 27 26 74 71 27 25 75 24 76 24 76 71 29 25 75 71 27 26 74 72 27 25 74 72 27 25 75 71
15 255 240 255 255 240 0 83 77 25 76 25 75 72 28 25 74 72 28 25 75 71 28 25 74 71 27 26 74 25 75 24
76 72 28 25 75 71 27 26 74 71 28 25 74 72 27 26 74 71 28 25 74 25 75 24 255 0 163 104 58 48 75 25 75
25 75 72 28 26 74 71 28 25 75 71 28 25 74 71 28 25 75 23 76 25 75 72 28 25 75 71 28 25 74 71 28 25 75
71 27 26 74 72 27 25
15 255 240 255 255 240
0 30 77 25 75 25 75 71 29 26 73 72 27 26 74 71 28 25 74 72 27 26 74 24 76 24 76 71 28 26 74 71 28 25
74 72 27 25 75 71 28 25 75 71 27 26 74 25 75 24 255 52 60 28 113 24 76 24 75 25 75 72 28 25 75 71 28
25 74 72 27 26 74 71 28 25 74 24 76 25 75 72 28 25 75 71 27 25 75 72 27 25 75 71 28 25 74 72 27 25
15 255 240 255 255 240 0 31 76 24 76 24 76 72 27 26 75 71 27 25 75 71 28 25 74 71 28 25 75 24 76 24
76 71 28 26 74 72 27 25 75 71 27 26 74 72 27 25 75 71 28 25 74 24 76 24 255 53 28 27 112 102 77 25 75
25 76 71 28 26 74 71 28 25 75 71 28 25 74 71 28 25 75 24 76 24 76 71 28 26 74 71 28 25 75 71 27 25 75
71 27 26 75 71 27 25
```

The PIC sends the received timings in chunck of 100 bytes.
You can now clearly see that most of the data consists of short periods 24..28 and long periods 71 .. 75 and that are the numbers we're looking for.
If you rearange the received bytes on the terminal, you should be able to detect the received code manually.
Look for the sync puls, i.e. a short puls: around 26, followed by a very long puls : 255
The time of the complete loop in the polling routine is about 11..12 usec. So one high and one low period = (25+75)* 11 usec = 1100 usec .. 1200 usec , which was to be expected.
From the sync puls, walk back and take 4 bytes a time (4 bytes is 1 digit)

```
77 76 24 76    rubish
25 75 72 28    F
26 74 71 28    F
25 75 71 27    F
25 75 71 28    F
25 74 24 76    0
25 76 71 27    F
26 75 71 27    F
26 74 71 28    F
25 75 71 27    F
26 75 71 27    F
25 75 23 76    0
25 255         sync bit
```

And indeed exactly what we expected, the first digit is rubish and the following 11 digits are correct.
Now we can adapt the timing borders (if necessary)

```
-- the timing borders
-- between B1 and B2 is a short period
-- between B3 and B4 is a long period
const B1 = 20
const B2 = 35
const B3 = 65
const B4 = 80
```

Timings smaller than B1, in between B2-B3 and larger than B3 are ignored and will reset the detector.
A short pulse folowed by a value > 250 will be seen as a sync bit.

JAL Function in Operate Mode

Set Do_Timing = False, Decoded_Output = 1 , reprogram the PIC and repeat the above procedure.
If we now press A1-On we get the following picture

```
0F FF 0F FF FF F1
0F FF 0F FF FF F1
0F FF 0F FF FF F1
0F FF 0F FF FF F1
0F FF 0F FF FF F1
0F FF 0F FF FF F1
0F FF 0F FF FF F1
0F FF FF FF FF F1
```

The decoding is very well, even the first bit is detected correctly.
The last series is wrong but will ignored by the top-level program, because the address is not correct.
If an invalid time sequence is encounterd it will be decoded as a "5".
If we test all the A-buttons

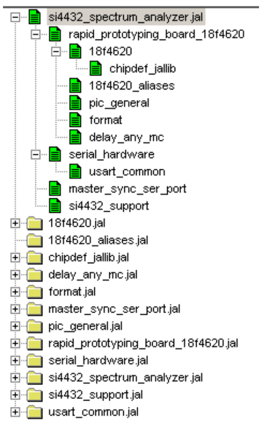
```
0F FF 0F FF FF F1     A1-On
0F FF 0F FF FF F0     A1-Off

0F FF F0 FF FF F1     A2-On
0F FF F0 FF FF F0     A2-Off

0F FF FF 0F FF F1     A3-On
0F FF FF 0F FF F0     A3-Off
```

again even the first bit is correct and all values are equal to the table below

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

The JAL Program

The Jal program consists of the following files:



For this project we only wrote the main program SI4432_spectrum_Analyzer and the SI4432_support library.

A zip file containing all the above JAL files can be found here:
http://mientki.ruhosting.nl/data_www/raspberry/Spectrum_Analyzer/SI4432_v2.zip

The pic definition files should ofcourse match the PIC you use.