Apply NCK2912 and FXTH87xx to kit TPMS RX/TX Solution

JASON CHIANG

10th , 05,2016

SECURE CONNECTIONS FOR A SMARTER WORLD

Outline

1. Background
2. NCK2912 parameters description
3. Required H/W and S/W Components
4. FXTH87xx H/W Environment Setup
5. NCK2912 H/W Environment Setup
6. Total TX/RX H/W Setup
7. FXTH87XX RF data format configuration
8. NCK2912 RX test results output to IREC Tool
9. Summary

Background

NCK2912 is a fully integrated single-chip receiver for use in an automotive environment. The device incorporated several commonly used building blocks including a crystal stabilized oscillator, a fractional-N based Phase Locked Loop(PLL) for a accurate frequency selection, Low Noise Amplifier(LNA), attenuator for Automatic Gain Control(AGC), I/Q down-mixer and two high resolution Analog to Digital Converters(ADC). By transforming signals in the digital domain in an early phase, one highly configurable RX channel is available including channel filter, ASK/FSK demodulator, clock-data recovery, bit processor and a micro-controller memory interface(DMA) allowing the micro-controller to complete the data handling and handshaking. NCK2912 has an embedded RISC micro-controller optimized for high performance and low power as well as an EROM for customer application.

The FXTH87XXX is a sensor for use in applications that monitor tire pressure and temperature. It contains the pressure and temperature sensors, an X-axis and a Z-axis accelerometer, a microcontroller, an LF receiver and an RF transmitter all within a single package.

Background(Cont')

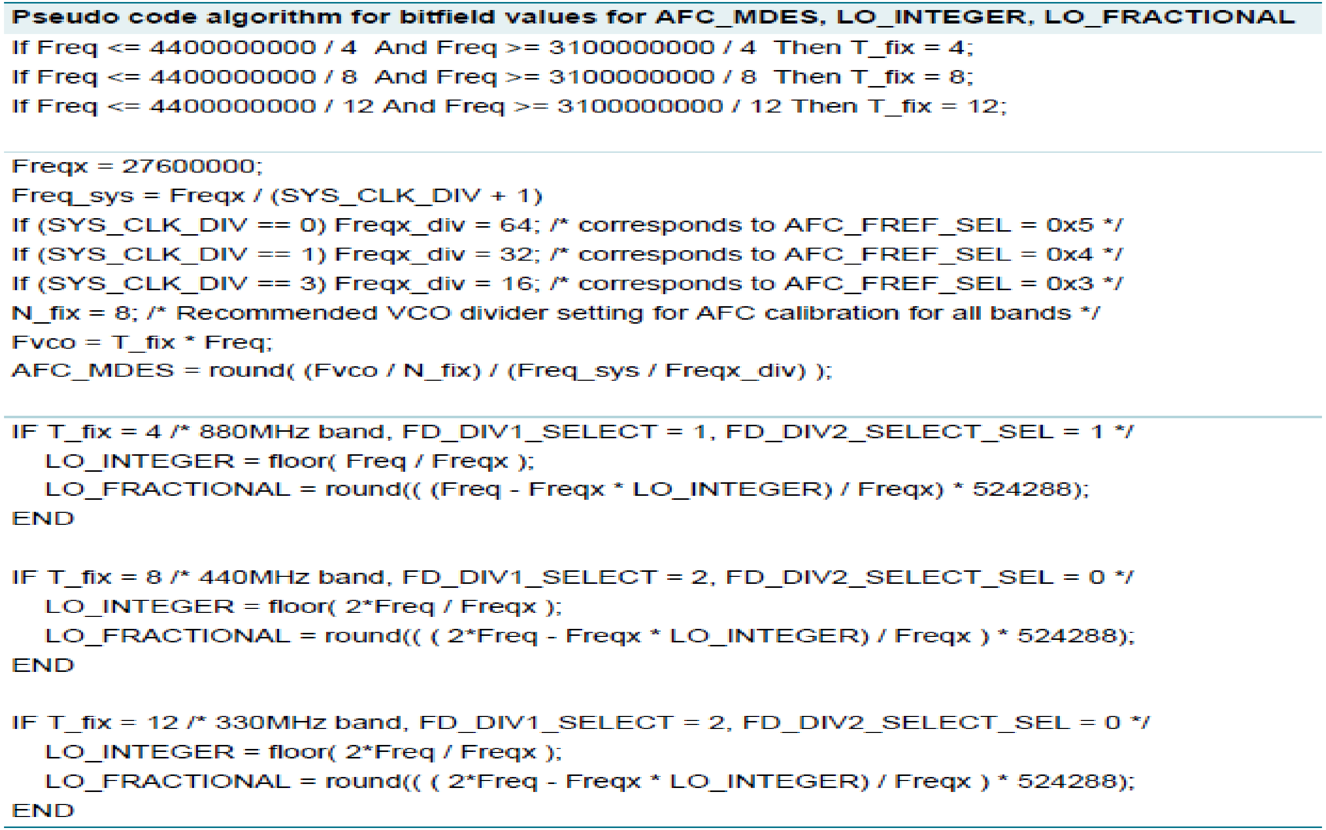
Recently we got feedback from sales and customer, E-Lead, Oringe, Oro-technology. They are interested in FXTH87xx and NCK29xx for solution. This TPMS solution offer customers to quickly evaluate TPMS RX/TX solution by utilizing NXP's FXTH87XX as transmitter and NCK2912 as receiver to kit 315MHz, 433.92MHz TPMS transmitter and receiver solution. Customer can easily integrate NXP's FXTH87XX , NCK2912 to design their time-to-market TPMS product quickly.

NCK2912 Parameters

Set frequency

The frequency is set using three bit fields according to the following algorithm. A windows executable is available to assist in this calculation. AFC_MDES is required as input value for the FAC to calibrate the coarse frequency setting for the VCO calibration; LO_INTEGER and LO_FRATIONAL configure the fraction divider.

NCK2912 Parameters(Cont’)



Pseudo code algorithm for bitfield values for AFC_MDES, LO_INTEGER, LO_FRACTIONAL

```
If Freq <= 4400000000 / 4  And Freq >= 3100000000 / 4  Then T_fix = 4;
If Freq <= 4400000000 / 8  And Freq >= 3100000000 / 8  Then T_fix = 8;
If Freq <= 4400000000 / 12 And Freq >= 3100000000 / 12 Then T_fix = 12;
```

```
Freqx = 27600000;
Freq_sys = Freqx / (SYS_CLK_DIV + 1)
If (SYS_CLK_DIV == 0) Freqx_div = 64; /* corresponds to AFC_FREF_SEL = 0x5 */
If (SYS_CLK_DIV == 1) Freqx_div = 32; /* corresponds to AFC_FREF_SEL = 0x4 */
If (SYS_CLK_DIV == 3) Freqx_div = 16; /* corresponds to AFC_FREF_SEL = 0x3 */
N_fix = 8; /* Recommended VCO divider setting for AFC calibration for all bands */
Fvco = T_fix * Freq;
AFC_MDES = round( (Fvco / N_fix) / (Freq_sys / Freqx_div) );
```

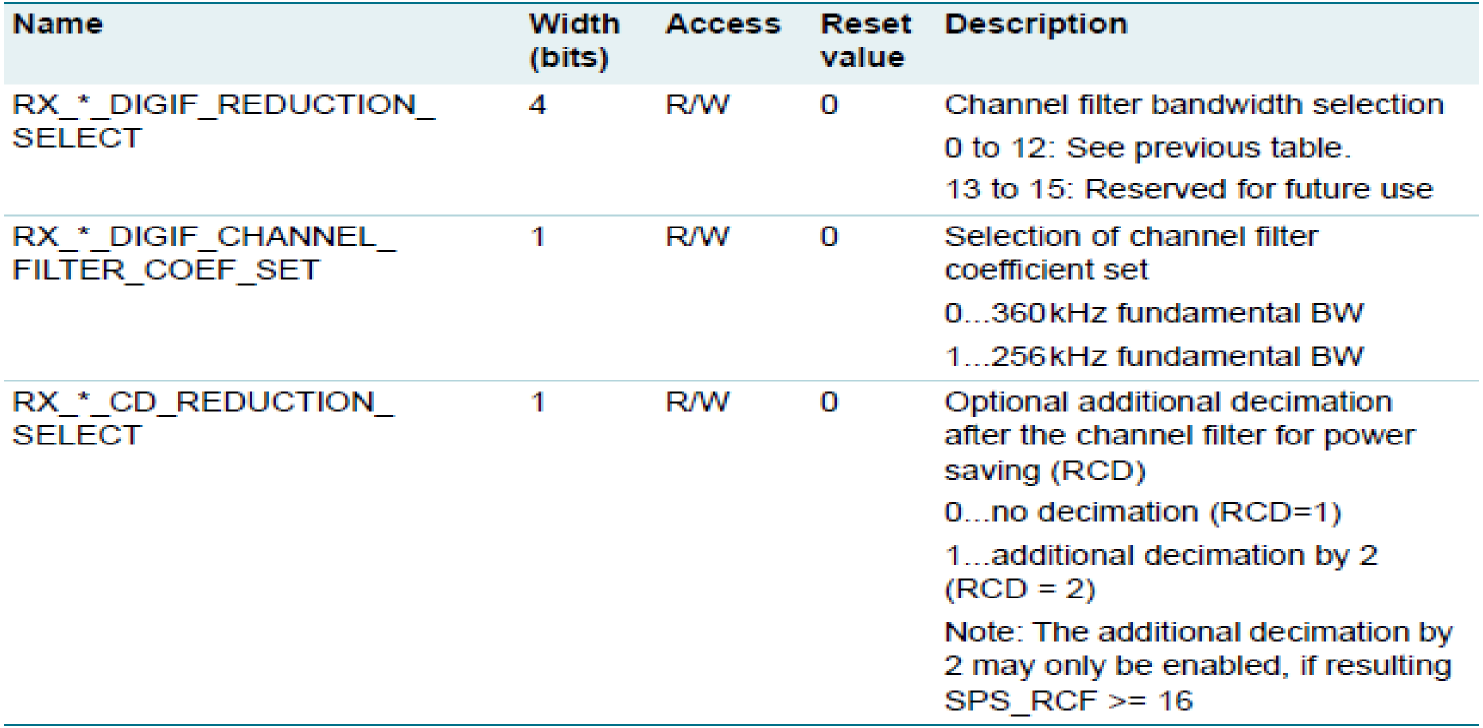
```
IF T_fix = 4 /* 880MHz band, FD_DIV1_SELECT = 1, FD_DIV2_SELECT_SEL = 1 */
    LO_INTEGER = floor( Freq / Freqx );
    LO_FRACTIONAL = round(( (Freq - Freqx * LO_INTEGER) / Freqx) * 524288);
END

IF T_fix = 8 /* 440MHz band, FD_DIV1_SELECT = 2, FD_DIV2_SELECT_SEL = 0 */
    LO_INTEGER = floor( 2*Freq / Freqx );
    LO_FRACTIONAL = round(( ( 2*Freq - Freqx * LO_INTEGER) / Freqx ) * 524288);
END

IF T_fix = 12 /* 330MHz band, FD_DIV1_SELECT = 2, FD_DIV2_SELECT_SEL = 0 */
    LO_INTEGER = floor( 2*Freq / Freqx );
    LO_FRACTIONAL = round(( ( 2*Freq - Freqx * LO_INTEGER) / Freqx ) * 524288);
END
```

NCK2912 Parameters (Cont’)

Channel Filter

The channel filter performs the selection of the desired band of interest out of the wide Band IF signal. The filter cut-off frequency is selected by a configurable sample rate conversion Stage. For correct baseband operation, the maximum chip rate of the wanted signal must not Exceed a certain value.

NCK2912 Parameters (Cont’)



Name	Width (bits)	Access	Reset value	Description
RX_*_DIGIF_REDUCTION_ SELECT	4	R/W	0	Channel filter bandwidth selection 0 to 12: See previous table. 13 to 15: Reserved for future use
RX_*_DIGIF_CHANNEL_ FILTER_COEF_SET	1	R/W	0	Selection of channel filter coefficient set 0...360kHz fundamental BW 1...256kHz fundamental BW
RX_*_CD_REDUCTION_ SELECT	1	R/W	0	Optional additional decimation after the channel filter for power saving (RCD) 0...no decimation (RCD=1) 1...additional decimation by 2 (RCD = 2) Note: The additional decimation by 2 may only be enabled, if resulting SPS_RCF >= 16

Required H/W and S/W Components

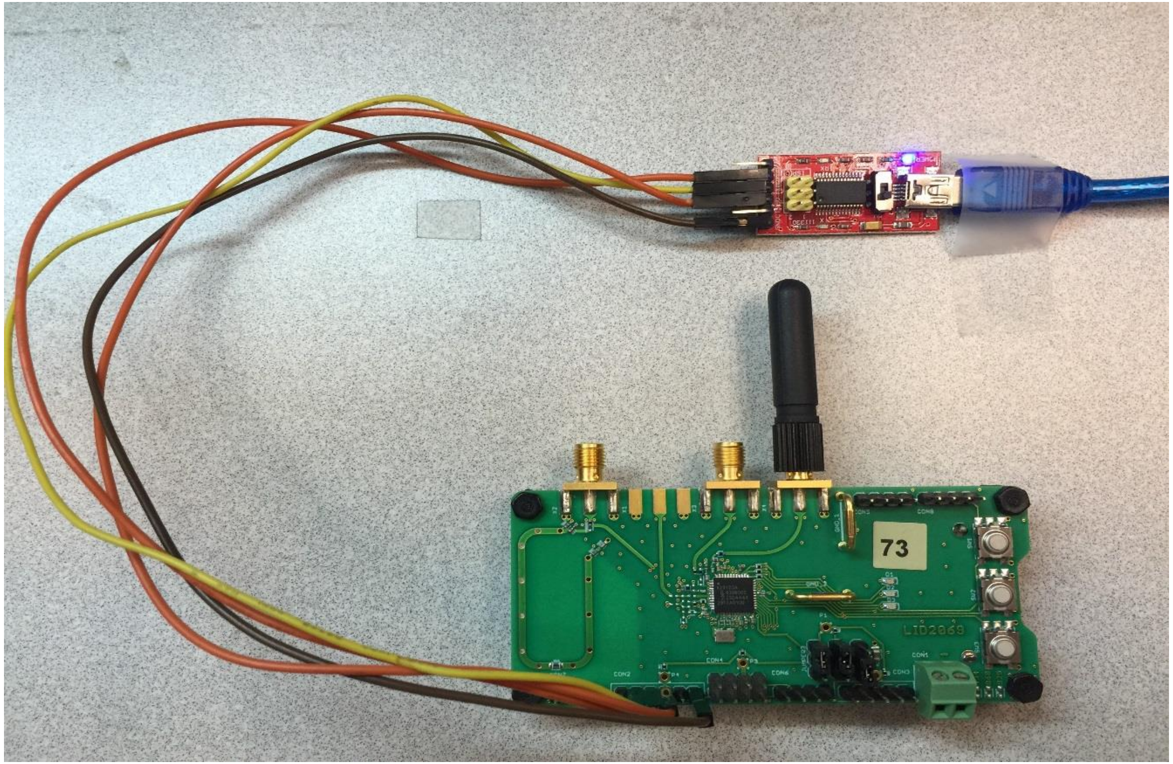
H/W :

a. FXTH87xx - TPMS transmitter
b. NCK2912 EVB - TPMS receiver
c. FTDI USB/UART board - Interface translation
d. Multilink-Universal

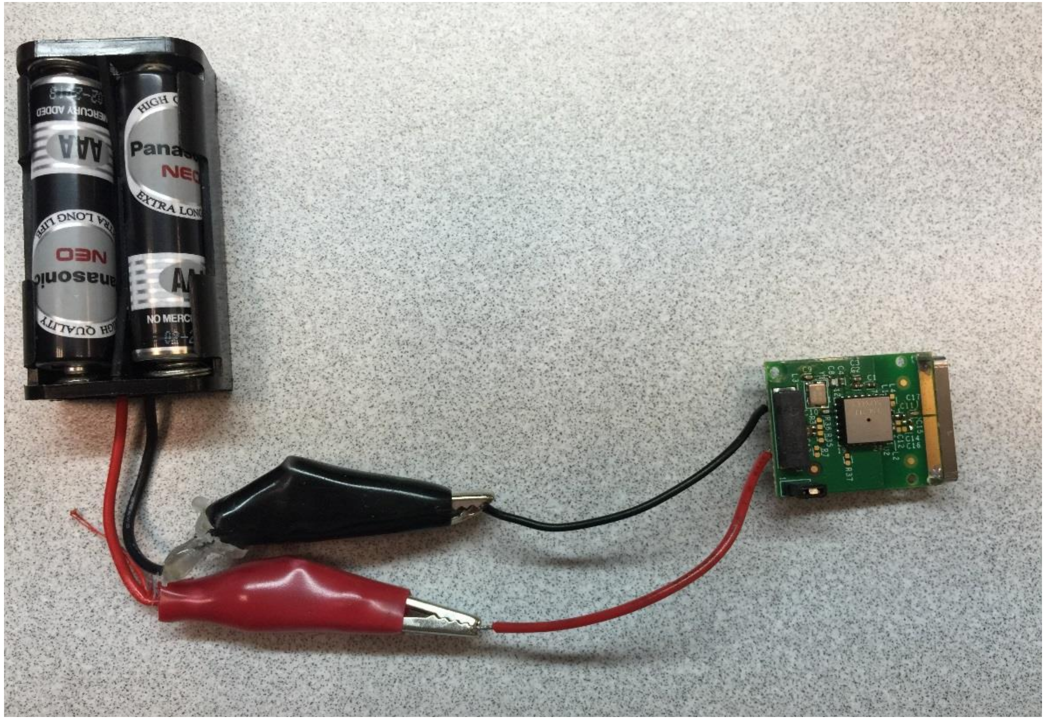
S/W :

a. IREC 5.0.0 – NCK2912 PC Tool
b. CodeWarrior 10.6 – F/W code debugging and downloading

RX H/W Environment Setup – NCK2912

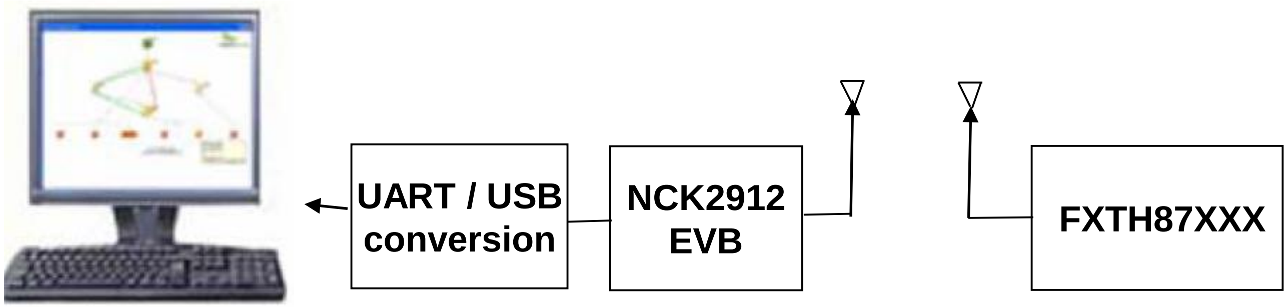


TX H/W Environment Setup – MPXY87xx



Total H/W Environment Setup

1. Prepare FXTH87XX EVB as Transmitter. FXTH87XX ’s RF data buffer sends data packet every 40 mS.
2. Prepare one NCK2912 EVB as Receiver, and the Received data output to IREC through UART to USB interface.

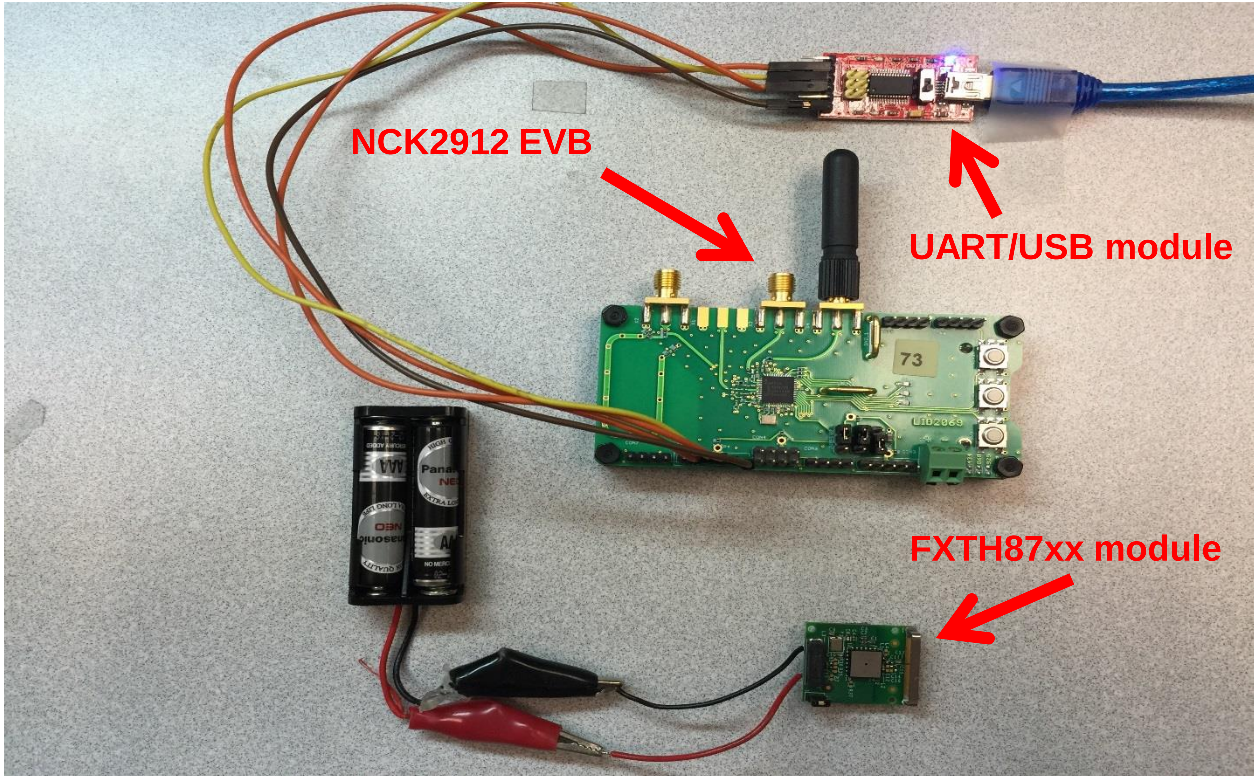


UART / USB conversion

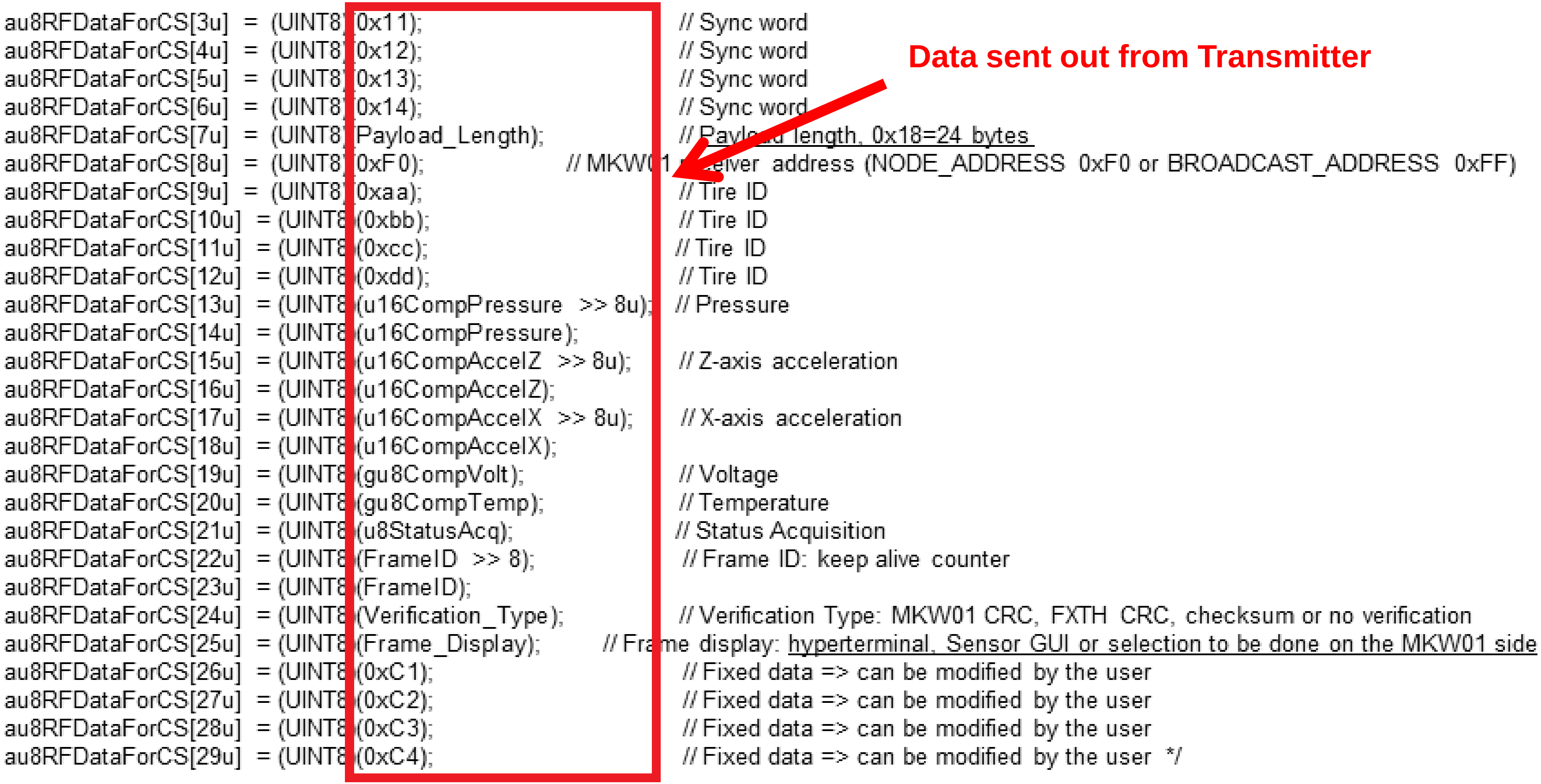
NCK2912 EVB

FXTH87XXX

Total H/W Environment Setup



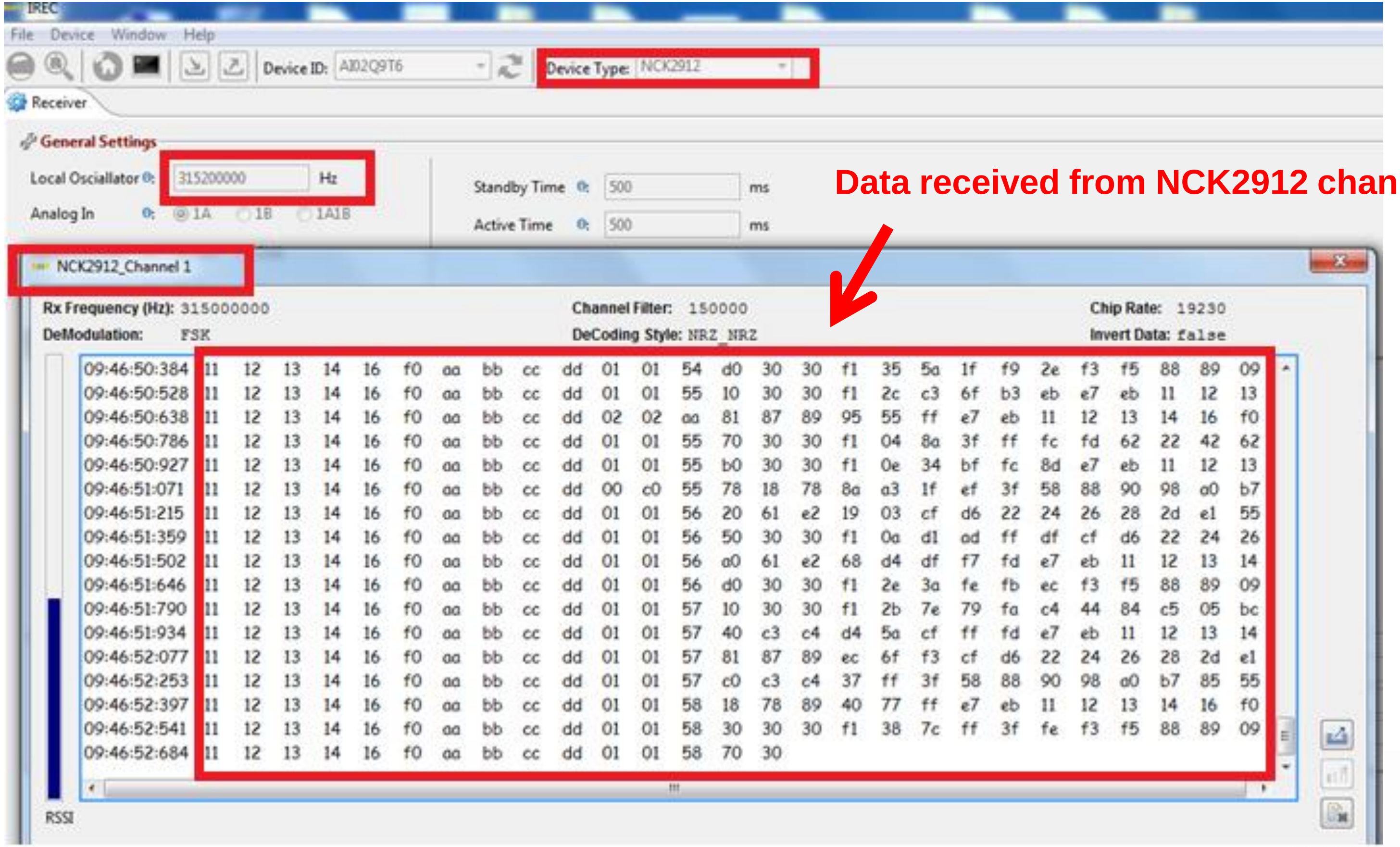
FXTH87XX RF data format configuration



```
au8RFDataForCS[3u]  = (UINT8)(0x11);                      // Sync word
au8RFDataForCS[4u]  = (UINT8)(0x12);                      // Sync word
au8RFDataForCS[5u]  = (UINT8)(0x13);                      // Sync word
au8RFDataForCS[6u]  = (UINT8)(0x14);                      // Sync word
au8RFDataForCS[7u]  = (UINT8)(Payload_Length);            // Payload length, 0x18=24 bytes
au8RFDataForCS[8u]  = (UINT8)(0xF0);             // MKW01 receiver address (NODE_ADDRESS 0xF0 or BROADCAST_ADDRESS 0xFF)
au8RFDataForCS[9u]  = (UINT8)(0xaa);                      // Tire ID
au8RFDataForCS[10u]  = (UINT8)(0xbb);                     // Tire ID
au8RFDataForCS[11u]  = (UINT8)(0xcc);                     // Tire ID
au8RFDataForCS[12u]  = (UINT8)(0xdd);                     // Tire ID
au8RFDataForCS[13u]  = (UINT8)(u16CompPressure  >> 8u);   // Pressure
au8RFDataForCS[14u]  = (UINT8)(u16CompPressure);
au8RFDataForCS[15u]  = (UINT8)(u16CompAccelZ  >> 8u);     // Z-axis acceleration
au8RFDataForCS[16u]  = (UINT8)(u16CompAccelZ);
au8RFDataForCS[17u]  = (UINT8)(u16CompAccelX  >> 8u);     // X-axis acceleration
au8RFDataForCS[18u]  = (UINT8)(u16CompAccelX);
au8RFDataForCS[19u]  = (UINT8)(gu8CompVolt);              // Voltage
au8RFDataForCS[20u]  = (UINT8)(gu8CompTemp);              // Temperature
au8RFDataForCS[21u]  = (UINT8)(u8StatusAcq);              // Status Acquisition
au8RFDataForCS[22u]  = (UINT8)(FrameID  >> 8);            // Frame ID: keep alive counter
au8RFDataForCS[23u]  = (UINT8)(FrameID);
au8RFDataForCS[24u]  = (UINT8)(Verification_Type);        // Verification Type: MKW01 CRC, FXTH CRC, checksum or no verification
au8RFDataForCS[25u]  = (UINT8)(Frame_Display);     // Frame display: hyperterminal, Sensor GUI or selection to be done on the MKW01 side
au8RFDataForCS[26u]  = (UINT8)(0xC1);                     // Fixed data => can be modified by the user
au8RFDataForCS[27u]  = (UINT8)(0xC2);                     // Fixed data => can be modified by the user
au8RFDataForCS[28u]  = (UINT8)(0xC3);                     // Fixed data => can be modified by the user
au8RFDataForCS[29u]  = (UINT8)(0xC4);                     // Fixed data => can be modified by the user  */
```

Data sent out from Transmitter

Output Results at IREC tool (NCK2912 Receiver)



Data received from NCK2912 channel 1

Device ID: AI02Q9T6

Device Type: NCK2912

Local Oscillator: 315200000 Hz

Analog In: ● 1A ○ 1B ○ 1A1B

Standby Time: 500 ms

Active Time: 500 ms

NCK2912_Channel 1

Rx Frequency (Hz): 315000000

DeModulation: FSK

Channel Filter: 150000

DeCoding Style: NRZ_NRZ

Chip Rate: 19230

Invert Data: false

```
09:46:50:384  11  12  13  14  16  f0  aa  bb  cc  dd  01  01  54  d0  30  30  f1  35  5a  1f  f9  2e  f3  f5  88  89  09
09:46:50:528  11  12  13  14  16  f0  aa  bb  cc  dd  01  01  55  10  30  30  f1  2c  c3  6f  b3  eb  e7  eb  11  12  13
09:46:50:638  11  12  13  14  16  f0  aa  bb  cc  dd  02  02  aa  81  87  89  95  55  ff  e7  eb  11  12  13  14  16  f0
09:46:50:786  11  12  13  14  16  f0  aa  bb  cc  dd  01  01  55  70  30  30  f1  04  8a  3f  ff  fc  fd  62  22  42  62
09:46:50:927  11  12  13  14  16  f0  aa  bb  cc  dd  01  01  55  b0  30  30  f1  0e  34  bf  fc  8d  e7  eb  11  12  13
09:46:51:071  11  12  13  14  16  f0  aa  bb  cc  dd  00  c0  55  78  18  78  8a  a3  1f  ef  3f  58  88  90  98  a0  b7
09:46:51:215  11  12  13  14  16  f0  aa  bb  cc  dd  01  01  56  20  61  e2  19  03  cf  d6  22  24  26  28  2d  e1  55
09:46:51:359  11  12  13  14  16  f0  aa  bb  cc  dd  01  01  56  50  30  30  f1  0a  d1  ad  ff  df  cf  d6  22  24  26
09:46:51:502  11  12  13  14  16  f0  aa  bb  cc  dd  01  01  56  a0  61  e2  68  d4  df  f7  fd  e7  eb  11  12  13  14
09:46:51:646  11  12  13  14  16  f0  aa  bb  cc  dd  01  01  56  d0  30  30  f1  2e  3a  fe  fb  ec  f3  f5  88  89  09
09:46:51:790  11  12  13  14  16  f0  aa  bb  cc  dd  01  01  57  10  30  30  f1  2b  7e  79  fa  c4  44  84  c5  05  bc
09:46:51:934  11  12  13  14  16  f0  aa  bb  cc  dd  01  01  57  40  c3  c4  d4  5a  cf  ff  fd  e7  eb  11  12  13  14
09:46:52:077  11  12  13  14  16  f0  aa  bb  cc  dd  01  01  57  81  87  89  ec  6f  f3  cf  d6  22  24  26  28  2d  e1
09:46:52:253  11  12  13  14  16  f0  aa  bb  cc  dd  01  01  57  c0  c3  c4  37  ff  3f  58  88  90  98  a0  b7  85  55
09:46:52:397  11  12  13  14  16  f0  aa  bb  cc  dd  01  01  58  18  78  89  40  77  ff  e7  eb  11  12  13  14  16  f0
09:46:52:541  11  12  13  14  16  f0  aa  bb  cc  dd  01  01  58  30  30  30  f1  38  7c  ff  3f  fe  f3  f5  88  89  09
09:46:52:684  11  12  13  14  16  f0  aa  bb  cc  dd  01  01  58  70  30
```

Summary

From NCK2912 IREC Tool results, we saw NCK2912 can receive TPMS data sending from FXTH87xx, TPMS transmitter after parameters configuration at transmitter and receiver side are proper setup. Parameters need to be fine-tuned are summarized below.

FXTH87xx TX:

a. TX_Speed_Fast timing
b. Baud rate
c. Transmit Frame size
d. Transmit data buffer
e. Operating Frequency
f. Frequency Deviation
g. Coding style format
h. Modulation type

NCK2912 RX :

a. Chip rate
b. Decoding style format
c. Demodulation type
d. Decoding format
e. Decoding packet size
f. Frequency Deviation
g. Decoding Frequency

NXP

SECURE CONNECTIONS
FOR A SMARTER WORLD