RF frame format and correspondence between raw and actual data

Format of the frame sent by the emitter (32 bytes)

Function that fills the RF buffer: Fill_RFBUFFER() in main.c source file.

```
(UINT8)(0x55);                    // Preamble
(UINT8)(0x55);                    // Preamble
(UINT8)(0x55);                    // Preamble
(UINT8)(0x01);                    // Sync word
(UINT8)(0x01);                    // Sync word
(UINT8)(0x01);                    // Sync word
(UINT8)(0x01);                    // Sync word
(UINT8)(Payload_Length);          // Payload length, 0x18=24 bytes
(UINT8)(0xF0);                    // MKW01 receiver address (NODE_ADDRESS 0xF0 or BROADCAST_ADDRESS 0xFF)
(UINT8)(Tire_ID >> 24u);          // Tire ID
(UINT8)(Tire_ID >> 16u);          // Tire ID
(UINT8)(Tire_ID >> 8u);           // Tire ID
(UINT8)(Tire_ID);                 // Tire ID
(UINT8)(u16CompPressure >> 8u);   // Pressure
(UINT8)(u16CompPressure);
(UINT8)(u16CompAccelZ >> 8u);     // Z-axis acceleration
(UINT8)(u16CompAccelZ);
(UINT8)(u16CompAccelX >> 8u);     // X-axis acceleration
(UINT8)(u16CompAccelX);
(UINT8)(gu8CompVolt);             // Voltage
(UINT8)(gu8CompTemp);             // Temperature
(UINT8)(u8StatusAcq);             // Status Acquisition
(UINT8)(FrameID >> 8);            // Frame ID: keep alive counter
(UINT8)(FrameID);
(UINT8)(Verification_Type);       // Verification Type: MKW01 CRC, FXTH CRC, checksum or no verification
(UINT8)(Frame_Display);           // Frame display: hyperterminal, Sensor GUI or selection to be done on the MKW01 side
(UINT8)(0xC1);                    // Fixed data => can be modified by the user
(UINT8)(0xC2);                    // Fixed data => can be modified by the user
(UINT8)(0xC3);                    // Fixed data => can be modified by the user
(UINT8)(0xC4);                    // Fixed data => can be modified by the user
(UINT8)(Verification_Value>>8);  // CRC, checksum or fixed data
(UINT8)(Verification_Value);
```

Format of the frames sent by the MKW01 to the hyperterminal

Refer to file hyperterminal.h in IAR MKW01 project.

TireID[4 bytes] Pressure[2 bytes] AccelZ[2 bytes] AccelX[2 bytes] Voltage[1 byte] Temperature[1 byte] StatusAcquisition[1 byte] FrameResultVerification[1 byte] Counter[2 bytes]

Example: BB02BB020001010800FDAD5301331DC7

- **Tire ID**

TireID is either 0xAA01AA01, 0xBB02BB02, 0xCC03CC03 or 0xDD04DD04. Only one of these IDs is accepted, other IDs are discarded.

Correspondence between data contained in the frames and actual data (pressure, acceleration, voltage and temperature):

- **Pressure:**

To know the correspondence between the pressure value given by the module and the actual pressure, c.f to the product specification. An example is given below.

Example:

The example was done with a FXTH8715xx device in the range 100 – 1500kPa, using the FXTH871x6 datasheet, but it is similar for other ranges of pressure and other families of devices, only sensor characteristics indicated in the datasheet have to be changed accordingly.
Pressure value received: Pressure = 0x0001 = 1 (decimal)
The actual pressure can be calculated with the Eqn. 1 on page 80: $P = \Delta P_{1500} * P_{CODE} + (100 - \Delta P_{1500})$
We have PCODE = 1, it is the value given by the module. Then in the table on page 155 of the datasheet we find that ΔP_{1500} = 2.750 kPa/count.
So we have P = 2.750*1 + (100 – 2.750) = **100kPa**

AccelZ and **AccelX:**
To know the correspondence between the acceleration value sent by the module and the actual acceleration, c.f to the product specification.
Two examples are given below. They were done with a FXTH870911 device, using the FXTH870xD datasheet, but it is similar for the other families of devices: only sensor characteristics indicated in the datasheet have to be changed accordingly.

Example 1:
AccelZ = 0x0108 = $(264)_{10}$

- In order to find the actual value of the Z-axis acceleration, we need to know the offset step of the acceleration measure. As the Z-axis acceleration goes from -210g to 300g (FXTH870x11 family) the range has been divided into 16 'windows' or steps, and each step has been divided into 510 counts. So we always get an acceleration value between 1 and 510 counts, but depending on the offset step, that does not correspond to the same actual acceleration. Firmware routines can give the offset step.
  In our case the offset step is step 6 so acceleration is between -30g and 30g (information given in the datasheet).
- Then we also need to know some acceleration measurement characteristics of the device that are given in the datasheet (in this example in part 17.10.2 on page 157):
  - Z-axis Average Accel Sensitivity (1 to 510 counts) = 0.118g/count (here we use the average sensitivity but it can be calculated more precisely for each step, c.f. to the datasheet).
  - $A_{Z\text{-}6}$ @ A_{ZCODE1} = -30g

- Now we can apply the Eqn. 17 on page 156 : $A_Z = \Delta A_{Z6} * A_{ZCODE} + (A_{Z6} @ A_{ZCODE}1 - \Delta A_{Z6})$
  We have $\Delta A_{Z6} = 0.118$g/count, $A_{ZCODE} = 264$ (the acceleration value given by the module) and $A_{Z6} @ A_{ZCODE}1 = -30$g
  So we get $A_Z = 0.118*264 + (-30 – 0.118) = 1.034$g

Example 2:
AccelX = 0x010C = $(268)_{10}$

- In order to find the actual value of the X-axis acceleration, we need to know the offset step of the acceleration measure. As the X-axis acceleration goes from -80g to 90g (FXTH870x11 family) the range has been divided into 16 'windows' or steps, and each step has been divided into 510 counts. So we always get an acceleration value between 1 and 510 counts, but depending on the offset step, that does not correspond to the same actual acceleration. Firmware routines can give the offset step.
  In our case the offset step is step 7 so acceleration is between -10g and 10g (information given in the datasheet).
- Then we also need to know some acceleration measurement characteristics of the device that are given in the datasheet (in this example in part 17.10.2 on page 157):
  - X-axis Average Accel Sensitivity (1 to 510 counts) = 0.039g/count (here we use the average sensitivity but it can be calculated more precisely for each step, c.f. to the datasheet).
  - $A_{X-7} @ A_{XCODE1} = -10$g
- Now we can apply the Eqn. 17 on page 156 for the X-axis:
  $A_X = \Delta A_{X7} * A_{XCODE} + (A_{X7} @ A_{XCODE}1 - \Delta A_{X7})$
  We have $\Delta A_{X7} = 0.039$g/count, $A_{ZCODE} = 268$ (the acceleration value given by the module) and $A_{X7} @ A_{XCODE}1 = -10$g
  So we get $A_X = 0.039*268 + (-10 – 0.039) = 0.413$g

- **Volt:**

To get the value in volt first convert the hexa value in decimal then add 122 and then divide by 100.
Example: Volt = 0xAD = 173 (decimal) => V = 2.95V

- **Temp:**

Temperature is given with an offset of 55 (refer to the product specification).

Example: Temp = 0x53 = 83 (decimal) => T = 83 - 55 = 28°C

- **Note about StatusAcquisition:** Here the status can be set to 1 because the pressure is under 350kPa (measure done at atmospheric pressure). For more details confer to the TPMS emitter project source code.

- **Frame Verification Result:**

- 0x00 => verification ok
- 0x11 => checksum verification failed
- 0x22 => FXTH CRC verification failed
- 0x33 => MKW01 CRC verification failed
- 0x44 => no verification selected (selection to be done on the TPMS emitter side)