Charles's Sensor Specification

The data transmission is organized in frames. One frame consists of 4 sub-frames.

Sub-Frame

A sub-frame consists of 13 bits and is defined as follow.

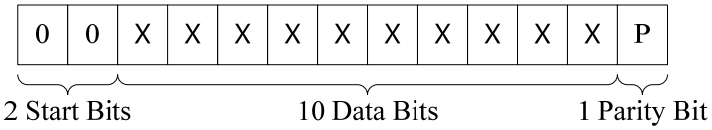

Figure 1 Sub-frame

Both start bits have to be logic “0”, followed by 10 data bits. A sub-frame is completed with one parity bit to check if an error occurred.
Parity bits contain even parity regarding the sub-frame (number of bits with logic “1” within the 13 bits of the sub-frame is even).
The sub-frame is Manchester coded. Logic “1” is represented with a falling edge; a logic “0” is represented with a rising edge in the middle of the bit.
The following figure shows the timing diagram:

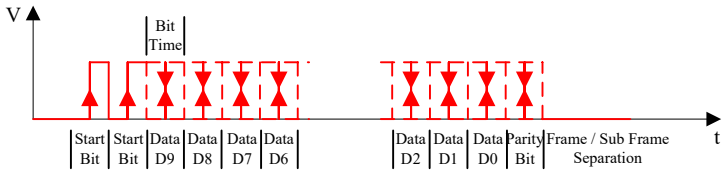

Figure 2 Sub-frame timing - Manchester code

Frame

A complete frame (with 4 sub-frames) is repeated every 5000us.
The timing accuracy depends on the accuracy of the oscillator frequency.

Table 1 Current-Interface Timing

Parameter	Value
Bit Time	40us

Sub-frame time	520us
Separation between two sub-frames	20us
Frame time	2140us
Min. separation between two frames	40us
Frame cycle time	5000us

Table 2 details the sub-frame allocation within a complete frame.

Table 2 Current-Interface frame – general format

Message	**Identifier**		**Data (MSB first)**								**Check**
	D9	**D8**	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**	
Check message	0	0	C7	C6	C5	C4	C3	C2	C1	C0	P
Data message 1	0	1	Data byte 1								P
Data message 2	1	0	Data byte 2								P
Data message 3	1	1	Data byte 3								P

Parity / CRC

Each transmitted message is protected by a parity bit, and all 3 data messages is protected by a Cyclic Redundancy Code (CRC). Any frame with incorrect parity bit or CRC should be discarded.
The parity bit P in every sub-frame is filled with even parity.
An 8 bit CRC is calculated for the data stream consisting of the all 3 data messages (MSB first). The CRC is transmitted in CRC bits as C7, C6, … , C1 and C0.

The 8-bit CRC is specified by the initialization value 0x3C, and the pseudocode is show as following.

```
for (i = Num_Bits - 1; i >= 0; i--)
{
    xor_5 = CRC_4 ^ CRC_7;
    xor_4 = CRC_2 ^ CRC_7;
    xor_3 = CRC_1 ^ CRC_7;
    xor_2 = CRC_0 ^ CRC_7;
    xor_1 = data_in[i] ^ CRC_7;
    CRC_7 = CRC_6;
```

```
        CRC_6 = CRC_5;
        CRC_5 = xor_5;
        CRC_4 = CRC_3;
        CRC_3 = xor_4;
        CRC_2 = xor_3;
        CRC_1 = xor_2;
        CRC_0 = xor_1;
    }
```