CRC Checksum Calculation

For Safe Communication of SHT1x and SHT7x Sensors

Preface

SHT1x and SHT7x humidity and temperature sensors communicate with the Sensirion specific digital interface protocol. Furthermore, the checksum formalism also is not of standard format. This specific checksum formalism is outlined in this document.

In case data transmission from the sensor to the microcontroller the checksum optionally can be read-out. Data and checksum must be equivalent. If not, there is a transmission error and respective data must be read out again.

Basic Considerations

CRC stands for Cyclic Redundancy Check. It is one of the most effective error detection schemes and requires a minimal amount of hardware. For in-depth information on CRC we recommend the comprehensive “A painless guide to CRC error detection algorithms” which is available at: http://www.repairfaq.org/filipg/LINK/F_crc_v3.html

The polynomial used in the SHT1x/7x sensor generation is the following: $x^8 + x^5 + x^4 + 1$. The types of errors that are detectable with this polynomial are:

- Any odd number of errors anywhere within the transmission
- All double-bit errors anywhere within the transmission
- Any cluster of errors that can be contained within an 8-bit “window” (1-8 bits incorrect)
- Most larger clusters of errors

The CRC register initializes with the value of the lower nibble of the status register (“0000’$s_3s_2s_1s_0$”, default “00000000”). It covers the whole transmission (command and response bytes) without the acknowledge bits.

The receiver can perform the CRC calculation upon the first part of the original message and then compare the result with the received CRC. In case of CRC mismatch the sensor shall be reset (command “00011110”) and the measurement be repeated.

This application note will cover two methods for checking the CRC. The first “Bitwise” is more suited for hardware or low-level implementation while the later “Bytewise” is the preferred method for more powerful microcontroller solutions.

Bit-Wise with Generator Model

With bitwise method, the receiver copies the structure of the CRC generator in hard or software.

An algorithm to calculate this looks like this:

1. Initialize CRC register to low nibble of status register (reversed ($s_0s_1s_2s_3$‘0000))
2. Compare each (transmitted and received) bit with bit 7
3. If the same: shift CRC register, bit0='0'
4. Else: shift CRC register to left and then invert bit4 and bit5 bit0='1' (see Figure 1)
5. Receive new bit and go to 2
6. The CRC value retrieved from the sensor must be reversed (bit 0 = bit 7, bit 1=bit 6 … bit 7 = bit 0) and can then be compared to the final CRC value.

Please note that this is different to other CRC implementations.

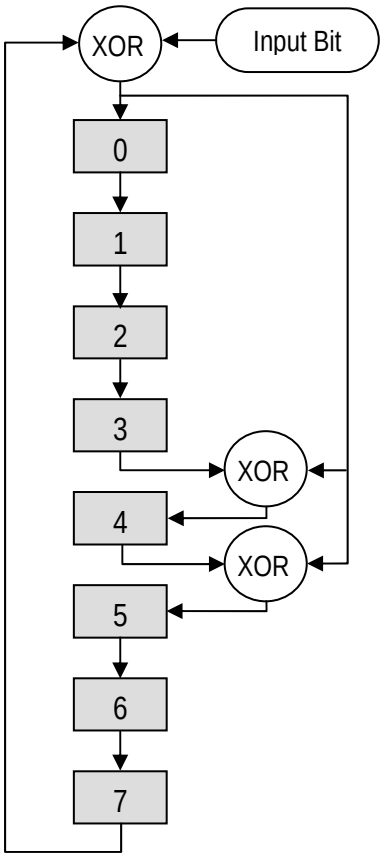

Figure 1 Internal structure of the SHTxx internal CRC generator.

Example 1: RH Measurement

In the following example a measurement of relative humidity is performed. The address plus command for issuing such measurement is '0000'0101', see Datasheet SHT1x, Sect. 3.2. In the example we calculate the CRC checksum with a sensor output '0000'1001''0011'0001', which corresponds to a humidity value (non temperature compensated) of 79.65%RH.

For the start value in the register (bit7 ... bit0) the actual values in the User Register are taken [$s_7s_6s_5s_4$'$s_3s_2s_1s_0$], order inverted and the last 4 bit are set to '0000' → [$s_0s_1s_2s_3$'0000]. In the example all bits of the User Register are default and hence the start value is '0000'0000'.

Input bit's	bit7 ... bit0	hex	dec	Comment
	0000'0000			'$s_0s_1s_2s_3$'0000'
0	0000'0000	00	0	1st bit of address
0	0000'0000			2nd bit of address
0	0000'0000			3rd bit of address
0	0000'0000			1st bit of command
0	0000'0000			...
1	0011'0001			CRC XOR polynom
0	0110'0010			
1	**1111'0101**	F5	245	CRC after command
0	1101'1011			1st byte (MSB) of measurement
0	1000'0111			
0	0011'1111			
0	0111'1110			
1	1100'1101			
0	1010'1011			
0	0110'0111			
1	**1111'1111**	FF	255	CRC value 1st byte
0	1100'1111			2nd byte (LSB) of measurement
0	1010'1111			
1	0101'1110			
1	1000'1101			
0	0010'1011			
0	0101'0110			
0	1010'1100			
1	**0101'1000**	58	88	Final CRC value
	0001'1010	1A	26	Reversed CRC value

If there was not transmission error the CRC checksum output of the sensor is 88. There are two ways for proof the successful transmission:

1. Very simple, compare final value of above calculation with CRC checksum output of the sensor. If the deviation is 0 then the transmission was successful. This method requires enough register on the microcontroller and is the faster method.

2. The above calculation can be continued by inserting the CRC checksum generated by the sensor. The final result must be '0000'0000' for proof of successful transmission. Here the example:

Input bit's	bit7 ... bit0	hex	dec	Comment
	0101'1000	58	88	Final CRC value
0	1011'0000			1st bit of sensor checksum output
1	0110'0000			
0	1100'0000			
1	1000'0000			
1	0000'0000			
0	0000'0000			
0	0000'0000			
0	**0000'0000**	00	0	Final Output

Example 2: Read Out of User Register

In this example we check the transmission of the readout of the User Register with the value 0x01 or '0000'0001' for illustration of the start condition:

Input bit's	bit7 ... bit0	hex	dec	Comment
	1000'0000			Start value see below
0	0011'0001	31	49	1st bit of address
0	0110'0010			2nd bit of address
0	1100'0100			3rd bit of address
0	1011'1001			1st bit of command
0	0100'0011			...
1	1011'0111			
1	0110'1110			
1	**1110'1101**	ED	237	CRC after command
0	1110'1011			1st byte (MSB) of measurement
0	1110'0111			
0	1111'1111			
0	1100'1111			
1	1001'1110			
0	0000'1101			
0	0001'1010			
1	**0000'0101**	5	5	CRC value
	1010'0000	A0	160	Reversed CRC value

Again, we have the two options to check for proper transmission: Either comparing with the actual value of the CRC checksum provided by the sensor or by continuing

the calculation with the CRC checksum output of the sensor:

Input bit's	bit7 ... bit0	hex	dec	Comment
1	**1011'1100**	58	88	Final CRC value
1	0111'1000			1st bit of sensor checksum output
0	1111'0000			
1	1110'0000			
1	1100'0000			
1	1000'0000			
1	0000'0000			
0	0000'0000			
0	**0000'0000**	00	0	Final Output

Byte-wise Calculation

As alternative, the checksum proof may be done by byte-wise XOR division using a look-up table. Please recall that XOR division compares two bytes bitwise and in case the digits differ ('0' – '1' and '1' – '0') the result is '1' while for equal digits ('0' – '0' and '1' – '1') the result is '0'. See the following example:

'0011'1010'
'0011'0101'

'0000'1111'

The algorithm for calculating byte-wise looks like this:

1. Initialize the CRC register with the value of the lower nibble of the value of the status register (reversed '$_0S_1S_2S_3$'0000' (default '00000000' = 0)
2. XOR each transmitted and received byte with the previous CRC value. Address and command first, then MSB, LSB
3. After each XOR division the result is compared with the look-up table and the resulting value in binary format is used as the new starting value for next XOR division
4. Repeat steps 2. and 3. until last transmitted byte is used. The last byte retrieved from the table is the final CRC value.

Respective Look-Up Table (LUT) is given in the following. Bold figures are input figures while plain figures are the output figures to be transformed in binary state and be used as divisor for next XOR division.

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	49	98	83	196	245	166	151	185	136	219	234	125	76	31	46
16	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
67	114	33	16	135	182	229	212	250	203	152	169	62	15	92	109
32	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**
134	183	228	213	66	115	32	17	63	14	93	108	251	202	153	168
48	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**60**	**61**	**62**	**63**
197	244	167	150	1	48	99	82	124	77	30	47	184	137	218	235
64	**65**	**66**	**67**	**68**	**69**	**70**	**71**	**72**	**73**	**74**	**75**	**76**	**77**	**78**	**79**
61	12	95	110	249	200	155	170	132	181	230	215	64	113	34	19
80	**81**	**82**	**83**	**84**	**85**	**86**	**87**	**88**	**89**	**90**	**91**	**92**	**93**	**94**	**95**
126	79	28	45	186	139	216	233	199	246	165	148	3	50	97	80
96	**97**	**98**	**99**	**100**	**101**	**102**	**103**	**104**	**105**	**106**	**107**	**108**	**109**	**100**	**111**
187	138	217	232	127	78	29	44	2	51	96	81	198	247	164	149
112	**113**	**114**	**115**	**116**	**117**	**118**	**119**	**120**	**121**	**122**	**123**	**124**	**125**	**126**	**127**
248	201	154	171	60	13	94	111	65	112	35	18	133	180	231	214
128	**129**	**130**	**131**	**132**	**133**	**134**	**135**	**136**	**137**	**138**	**139**	**140**	**141**	**142**	**143**
122	75	24	41	190	143	220	237	195	242	161	144	7	54	101	84
144	**145**	**146**	**147**	**148**	**149**	**150**	**151**	**152**	**153**	**154**	**155**	**156**	**157**	**158**	**159**
57	8	91	106	253	204	159	174	128	177	226	211	68	117	38	23
160	**161**	**162**	**163**	**164**	**165**	**166**	**167**	**168**	**169**	**170**	**171**	**172**	**173**	**174**	**175**
252	205	158	175	56	9	90	107	69	116	39	22	129	176	227	210
176	**177**	**178**	**179**	**180**	**181**	**182**	**183**	**184**	**185**	**186**	**187**	**188**	**189**	**190**	**191**
191	142	221	236	123	74	25	40	6	55	100	85	194	243	160	145
192	**193**	**194**	**195**	**196**	**197**	**198**	**199**	**200**	**201**	**202**	**203**	**204**	**205**	**206**	**207**
71	118	37	20	131	178	225	208	254	207	156	173	58	11	88	105
208	**209**	**210**	**211**	**212**	**213**	**214**	**215**	**216**	**217**	**218**	**219**	**220**	**221**	**222**	**223**
4	53	102	87	192	241	162	147	189	140	223	238	121	72	27	42
224	**225**	**226**	**227**	**228**	**229**	**230**	**231**	**232**	**233**	**234**	**235**	**236**	**237**	**238**	**239**
193	240	163	146	5	52	103	86	120	73	26	43	188	141	222	239
240	**241**	**242**	**243**	**244**	**245**	**246**	**247**	**248**	**249**	**250**	**251**	**252**	**253**	**254**	**255**
130	179	224	209	70	119	36	21	59	10	89	104	255	206	157	172

Figure 2: Look-up table for byte-wise CRC calculation. Bold figures are input figures and plain figures are output figures.

Example 1: RH Measurement

For illustration we will calculate CRC checksum of example 1 from the previous Section – measurement of relative humidity with sensor output '0000'1001''0011'0001', which corresponds to a humidity value (non temperature compensated) of 79.65%RH.

bit7 ... bit0	dec	LUT	Comment
0000'0000			Register ('$S_0S_1S_2S_3$'0000')
0000'0101			Address & command
0000'0101	5	245	
1111'0101	245		
0000'1001			Data output MSB
1111'1100	252	255	
1111'1111	255		
0011'0001			Data output LSB
1100'1110	206	88	
0101'1000	88		Final CRC value
0001'1010	26		Reversed CRC value

Example 2: Read Out of User Register

Here we repeat the readout of the User Register with the value 0x01 or '0000'0001' for illustration of the start condition:

bit7 ... bit0	dec	LUT	Comment
1000'0000			Register ('$s_0s_1s_2s_3$'0000')
0000'0111			Address & command
1000'0111	135	237	
1110'1101	237		
0000'1001			Data output MSB
1110'0100	228	5	
0000'0101	5		Final CRC value
1010'0000	160		Reversed CRC value

Sample Code for Byte-Wise Calculation

```
Var
CRC : Byte;
Procedure calc_CRC(X: Byte);

Const
CRC_Table: Array[0..255] of Byte = (0, 49,
98, 83, 196, 245, 166, 151, 185, 136, 219,
234, 125, 76, 31, 46, 67, 114, 33, 16, 135,
182, 229, 212, 250, 203, 152, 169, 62, 15,
92, 109, 134, 183, 228, 213, 66, 115, 32,
17, 63, 14, 93,108, 251, 202, 153, 168,
197, 244, 167, 150, 1, 48, 99, 82, 124, 77,
30, 47, 184, 137, 218, 235, 61, 12, 95,
110, 249, 200, 155, 170, 132, 181, 230,
215, 64, 113, 34, 19, 126, 79, 28, 45, 186,
139, 216, 233, 199, 246, 165, 148, 3, 50,
97, 80, 187, 138, 217, 232, 127, 78, 29,
44, 2, 51, 96, 81, 198, 247, 164, 149, 248,
201, 154, 171, 60, 13, 94, 111, 65, 112,
35, 18, 133, 180, 231, 214, 122, 75, 24,
41, 190, 143, 220, 237, 195, 242, 161, 144,
7, 54, 101, 84, 57, 8, 91, 106, 253, 204,
159, 174, 128, 177, 226, 211, 68, 117, 38,
23, 252, 205, 158, 175, 56, 9, 90, 107, 69,
116, 39, 22, 129, 176, 227, 210, 191, 142,
221, 236, 123, 74, 25, 40, 6, 55, 100, 85,
194, 243, 160, 145, 71, 118, 37, 20, 131,
178, 225, 208, 254, 207, 156, 173, 58, 11,
88, 105, 4, 53, 102, 87, 192, 241, 162,
147, 189, 140, 223, 238, 121, 72, 27, 42,
193, 240, 163, 146, 5, 52, 103, 86, 120,
73, 26, 43, 188, 141, 222, 239, 130, 179,
224, 209, 70, 119, 36, 21, 59, 10, 89, 104,
255, 206, 157, 172);

Begin
CRC := CRC_Table[X xor CRC];
End;
```

Revision History

Date	Version	Changes
30 December 2001	0.9 (Preliminary)	Initial revision
5 May 2008	1.08	Improved Generator polynomial and explained more detailed CRC calculation
23 October 2008	1.1	Revised format
5 May 2010	1.2	Adapted to the new Sensirion format, rearrangement of text. Sensirion China added.
8 July 2010	1.21	Figure 2 “LUT for bytewise CRC-calculation” – value byte 8 corrected Version number corrected Descriptive text in example calculation corrected
13 April 2011	1.22 RRU	Review calculation examples Review descriptive text in calculation examples Inversion of final CRC value added to display comparability with sensor CRC output