MTH02

Temperature & Humidity Sensor Specification

- Relative humidity and temperature sensor
- Pre-calculated temperature and humidity read out, no extra calculation needed
- Dew Point Calculation possible
- Fully Calibrated, Digital Output
- Excellent Long Term Stability
- No Extra Component Required
- Ultra Low Power Consumption
- Fully Interchangeable
- Small Size
- Automatic Power Down

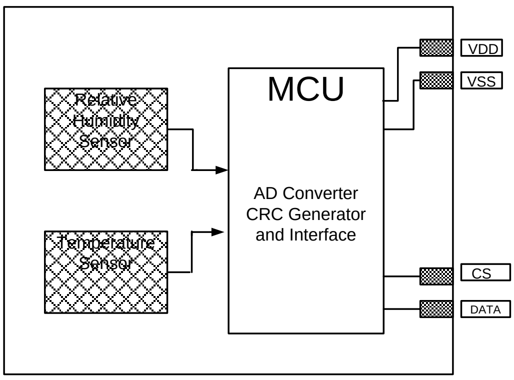
Product Summery

The MTH02 is a MCU based temperature and relative humidity sensor module, comprising one wire interface for direct temperature and humidity read out. The digital output is pre-calculated and no extra calculation is required. The system applied two sensor elements: built in NTC type high precision temperature sensor and resistor type relative humidity sensor from Japan. With a very unique and patented relative humidity calculation algorithm, the system can assure accurate relative humidity output through fine tuned temperature compensation mechanism. Thus very high accurate reading of humidity in the full temperature range (0-50C) can be assured.

Applications

- HVAC
- Consumer Products
- Weather Stations
- Humidifiers
- Dehumidifiers
- Test and Measurements
- Data Logging
- White Goods

Block Diagram



MTH02

Temperature & Humidity Sensor Specification

1. Module Performance Specification

Parameter	Conditions	Min	Typ	Max	Unit
Humidity					
Resolution				1	%
Repeatability			1		%
Accuracy Uncertainty	Temperature at 0C – 50C range	0	3	5	%
Interchangeability		Fully Inter Changeable			
Nonlinearity			1		%
Range	Temperature at 0C – 50C range	18		98	%
Response Time	63% slowly move air		60		Second
Hysterisis	Non-condensate	1		2	%
Long Term Stability	Non-condensate		2		%/yr
Temperature					
Resolution			0.1		˚C
Repeatability			0.1	0.2	˚C
Range		-40		70	˚C
Accuracy	25		+/-0.5	+/-1.0	˚C
Response Time	delta T=1.0		60		S

2. Sensor Interface

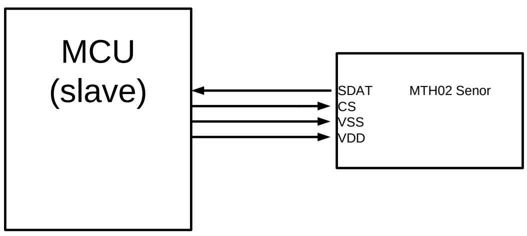


Figure 1: Typical Application

2.1 Power Pins

The MTH02 sensor module requires a voltage supply between 2.2 to 5.5. After power up, the sensor needs 20ms to complete its internal reset process. After reset finished, the sensor will make a measurement automatically and if the CS pin is low, then the measured data will be output automatically.

Power pins should be decoupled through a 10-100nF capacitor. Where in those applications with high power noise environment, it is strongly recommended to use a 10uF tantalum capacitor to protect the sensor from interferences.

2.2 Serial Interface

The serial interface is optimized for convenient reading and reducing IO usage. Application engineer should be kept in mind the characteristics of the IO pins for applications where current consumption is critical.

CS, INPUT pin, has 50k pull up resister connected internally, thus during normal application, the pin should not be tied to low unless read operation is really needed.

DATA, OUTPUT pin in push pull mode. During the power down mode, DATA pin is kept low, and data is modulated through this pin when CS is kept low.

3. Bus Timing

To start a data reading, set the CS line low will wake up the sensor module and start to transfer data through the data line. The bus timing is specified in the following way:

Figure 3: Bus timing

T1: 10ms Nominal
1.5ms - 0
500us - 1

Once CS is set high, sensor module will terminate data transfer process right way, and one extra measurement will start automatically before entering power down mode. The purpose of doing so is that data will be always ready to be transferred once a read command is initiated by pulling low CS pin. This is helpful in reducing waiting time before getting any data back from the sensor module. The imperfect part of this practice is that the data is for previous measured, not up to the current second.

Data output bit stream starts with MSB of temperature (2 byte) data, followed with one byte humidity , two bytes of external sensor temperature and one byte CRC.

T_MSB::T_LSB::RH::EXT_T_MSB::EXT_T_LSB::CRC

3.1Converting output data to temperature and humidity

The temperature value has added an offset value of 40C to avoid negative temperature sign flag problem and multiplied by ten. Thus real temperature can be obtained by deducting 0190(Hex) .
For example:
Data stream: 000000101011001101010001111111111111111110100001
T_offset=0x02B3=691 T_real=691-400=291=29.1C
RH=0x51=81%
EXT_T is not selected through solder option, then 0xFF is output.
CRC=0xA1

4. DC Characteristics

Parameter	Conditions	Min	Typ	Max	Unit
Power Supply DC		2.2	3	5.5	V
Power Supply Current	AD measuring		100	200	uA
	data transfering		0.2		mA
	standby		0.2		uA
Low Level Output Voltage		0		20%	Vdd
High Level Output Voltage		80%		100%	Vdd
Low Level Input Voltage	Negative Edge	0%		30%	Vdd
High Level Input Voltage	Positive Edge	70%		100%	Vdd
Pads Leakage Current			1		uA
Output High Current	80%VDD		10		mA
Output Low Current	20%VDD		20		mA

5. Package Information

The module is in 10 x 20 mm size, with six pins:

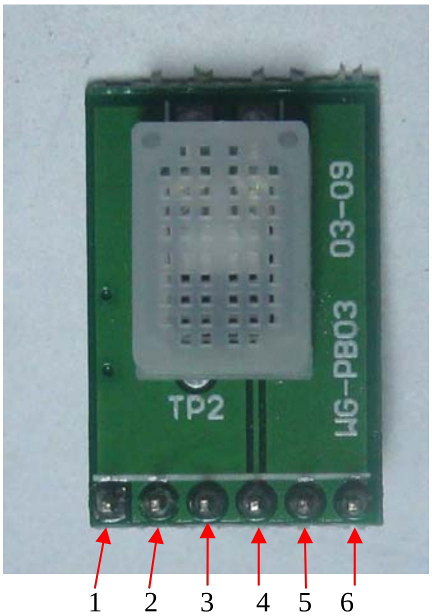


Figure5: outline of the sensor module

5.1 Pin Definition

1. VDD
2. VSS
3. PON
4. DATA
5. SNSA
6. SNSB

6. CRC

CRC stands for Cyclic Redundancy Check. It is one of the most effective error detection schemes and requires a minimal amount of hardware.

The polynomial used in the MTH01 is: $x^8 + x^5 + x^4$. The types of errors that are detectable with this polynomial are:
1. Any odd number of errors anywhere within the transmission.
2. All double-bit errors anywhere within the transmission.
3. Any cluster of errors that can be contained within an 8-bit “window” (1-8 bits incorrect).
4. Most larger clusters of errors.

The receiver can perform the CRC calculation upon the first part of the original message and then compare the result with the received CRC- 8. If a CRC mismatch is detected, the MTH01 should be reset and a new measurement should 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.

6.1 Bitwise

With the bitwise method, the receiver copies the structure of the CRC generator in hard- or software.
An algorithm to calculate this could look like this:

1) Initialize CRC Register to 0
2) Compare each (transmitted and received) bit with bit 7
3) If the same: shift CRC register, bit0='0' else: shift CRC register and then invert bit4 and bit5, bit0='1' (see figure 1)
4) receive new bit and go to 2)
5) After the last byte is treated, the result in CRC register is the calculated.

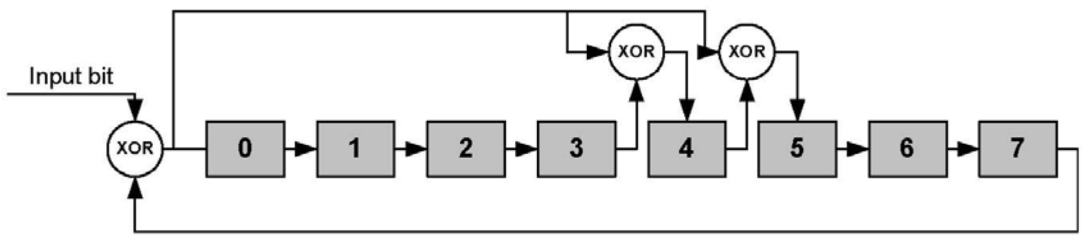


Figure 6 CRC generator state machine

6.1.1 Example for bitwise

Example 1: 0x05-0x09-0x31 CRC calculate example

Input bit's	bit7 … bit0	0x	dec	Comment
	0000'0000			Start value
0	0000'0000	00	0	1st bit of command
0	0000'0000	00	0	2nd bit of command
0	0000'0000	00	0	…
0	0000'0000	00	0	
0	0000'0000	00	0	
1	0011'0001			CRC EXOR 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**
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**

Example 1: 0x40 CRC calculate example

Input bit's	bit7 … bit0	0x	dec	Comment
	0000'0000			Start value see below (1)
0	0000'0000	00	0	1st bit of command
0	0000'0000	00	0	2nd bit of command
0	0000'0000	00	0	…
0	0000'0000	00	0	
0	0000'0000	00	0	
1	0011'0001			CRC EXOR polynom
1	0101'0011			
1	**1001'0111**	**97**	**151**	**CRC after command**
0	0001'1111			1st bit (MSB) of status register
1	0000'1111			
0	0001'1110			
0	0011'1100			
0	0111'1000			
0	1111'0000			
0	1101'0001			

6.2 Bytewise

With this implementation the CRC data is stored in a 256 byte lookup table.
Perform the following operations:

1. Initialize the CRC register with value 0
2. XOR each (transmitted and received) byte with the previous CRC value. The result is the new byte that you need to calculate the CRC value from.
3. Use this value as the index to the table to obtain the new CRC value.
4. Repeat from 2.) until you have passed all bytes through the process.
5. The last byte retrieved from the table is the final CRC value.

6.2.1 256 byte CRC Lookup table

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

6.2.2 Code example for lookup table

The following procedure calculates the CRC-8. The result accumulates in the variable CRC.

```
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;
```

7. Important Notice

Do not use this product as safety or emergency stop devices or in any other applications where failure of the product could result in personal injury. Failure to comply with this instruction could result in death or fatal injury.

7.1 ESD precautions

To prevent ESD related damage and/or degradation, take normal ESD precautions when handling the device.

7.2 Warranty

We make no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor do we assume any liability arising out of the application or use of any product or circuit and specifically disclaims any and all liability, including without limitation consequential or incidental damages.