VAISALA

TECHNICAL REFERENCE

I2C Protocol Implementation
HMM105 Humidity Module

M211638EN-C

PUBLISHED BY

Vaisala Oyj
Street address: Vanha Nurmijärventie 21, FI-01670 Vantaa, Finland
Mailing address: P.O. Box 26, FI-00421 Helsinki, Finland
Phone: +358 9 8949 1

Visit our Internet pages at www.vaisala.com.

Table of Contents

CHAPTER 1
GENERAL INFORMATION

About This Document

This document describes the I^2C interface implementation of the Vaisala HUMICAP® Digital Humidity Module HMM105.

Version Information

Table 1 Document Versions

Document Code	Description
M211638EN-C	November 2019. Updated Table 16, Table 21, and Table 32.
M211638EN-B	December 2015. Updated description of checksum calculation.
M211638EN-A	May 2014. First version.

Related Manuals

Table 2 Related Manuals

Document Code	Name
M211637EN	Vaisala HUMICAP® Digital Humidity Module HMM105 Quick Reference Guide

CHAPTER 2

I²C INTERFACE

Overview

HMM105 has an inter-integrated circuit (I^2C) interface for interfacing with the incubator's control computer. HMM105 implements I^2C slave functionality, with the incubator's computer acting as the master. The interface can be used to read measurement values and status information, set operation parameters, and make adjustments.

Physical Interface

The physical interface is a non-isolated 3-wire interface. Wires are SDA, SCL and ground. SDA and SCL lines are buffered. Ground is shared with power supply. There are small pull-up resistors for SCL and SDA.

Maximum cable length should not exceed 5 m and maximum capacitance between communication lines and ground should not exceed 500 pF.

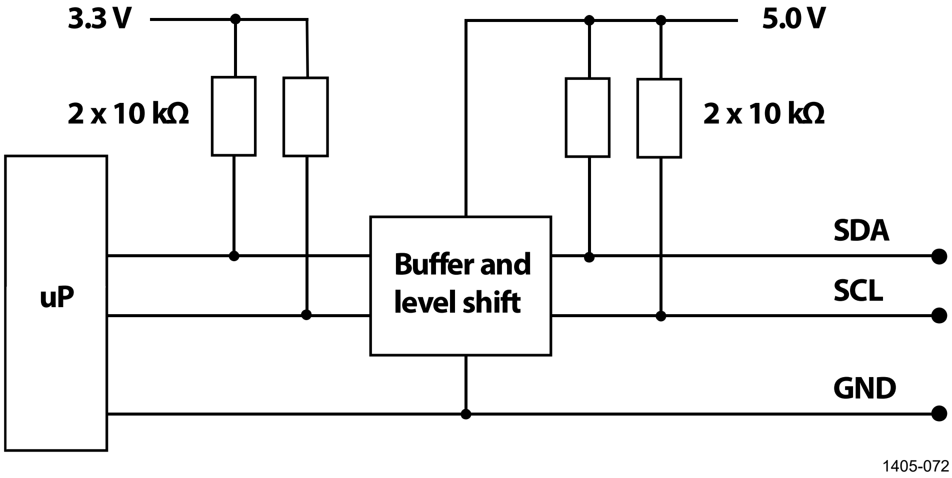
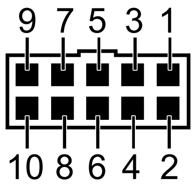


Figure 1 Hardware Schematic

The physical connector is a Molex 87832-1007, 2 mm pitch shrouded pin header with a locking window. It is marked **X6** on the component board. See Table 3 below.

Table 3 HMM105 Signal and Power Connector X6

Connector Pinout	Pin #	Function
	6, 8	Supply voltage input

		10 ... 35 VDC or 24 VAC
	5, 7	Ground
	1, 3	5 V I^2C bus SDA
	2, 4	5 V I^2C bus SCL
	9, 10	Not connected

Communication Parameters

HMM105 supports a maximum clock speed of 50 kHz. Protocol bits are sent most significant bit (MSB) first. Parameter bytes are sent using little endian order.

Addressing

HMM105 uses 7-bit addressing. The address consists of:

- 4-bit device type identifier part (default “0101” for HMM105)
- 3-bit sub address (default “111”)

The full 7-bit default address is “0101111” (2Fh). In I^2C communication, the address should be provided by the master in the standard way after the I^2C start condition, with the read/write bit as the least significant bit (LSB).

Table 4 HMM105 I^2C Address

0	1	0	1	1	1	1	R/W
Device type				Sub-address			Read/write bit (LSB)

The I^2C implementation of the HMM105 also includes the address inside the message frame. The purpose of this is to make the I^2C implementation easier, since the I^2C address can be lost by the I^2C hardware. This address is provided without the read/write bit, with zero as the most significant bit (MSB).

Table 5 HMM105 Device Address

0	0	1	0	1	1	1	1
MSB	Device type				Sub-address		

NOTE	Make sure there are no addressing conflicts if other I^2C devices are put on the same bus. The HMM105 address can be changed, see Table 32 on page 20.

Communication Flow

Basic communication flow always includes I^2C write and read commands. First the master writes a command to the slave device, and then the master reads the results of that command from the slave.

When the master reads data from the slave, there is an ACK/NACK-bit in the status byte that informs the master whether or not communication was success. ACK is defined as zero and NACK is defined as one in the logical level.

NOTE This ACK/NACK refers to message level acknowledge. I^2C protocol includes separate byte level acknowledge.

Command messages sent by the master are called **invokes**. Replies sent by the slave are called **responses**. Note that also responses include the device address.

Table 6 Invoke Message in HMM105 I^2C Interface

Message Segment	Length	Content
Start		I^2C start condition.
I^2C address	1 byte	See Table 4 on page 4.
Command	1 byte	Identifier of command.
Device address	1 byte	See Table 5 on page 4.
Frame length	1 byte	Length of invoke message in bytes (excluding I^2C address, including CRC). Minimum invoke frame length is 5 bytes.
Data	Variable length	Content and length depending on the command. May be left out if command contains no data in the invoke message.
Checksum	2 bytes	See section Checksum on page 9.
Stop		I^2C stop condition

Table 7 Response Message in HMM105 I^2C Interface

Message Segment	Length	Content
Start		I^2C start condition.
I^2C address	1 byte	See Table 4 on page 4.
Status	1 byte	See section Status Byte on page 9.
Command	1 byte	Identifier of the command that was used in the invoke that the slave is responding to. If slave is in idle state (no valid invoke), command code is FFh.
Device address	1 byte	See Table 5 on page 4.
Frame length	1 byte	Length of the response message in bytes (excluding I^2C address, including CRC). The minimum response frame length is 6 bytes.
Data	Variable length	Content and length depending on the command.
Checksum	2 bytes	See section Checksum on page 9.
Stop		I^2C stop condition

HMM105 State Machine

The main and initial state of the HMM105 is Idle, and it will change to WaitResponse state only when a valid invoke is received. In WaitResponse state HMM105 expects to see an I^2C read operation. After seeing a read operation the HMM105 outputs its response with ACK/NACK and status signals.

If HMM105 is in WaitResponse state and the master sends a new invoke, the old response belonging to the old invoke will be lost. When the master sends the next I^2C read command, the HMM105 sends it the response to the latest invoke.

If HMM105 is in Idle state and I^2C read is sent, HMM105 responds with a NACK-signal. This tells the master that the data which the master may have received in same I^2C read is not valid.

If the HMM105 receives an invalid invoke, it goes to the Idle state. An invoke message can be invalid because of an unknown command, erroneous CRC, or invalid message length.

.

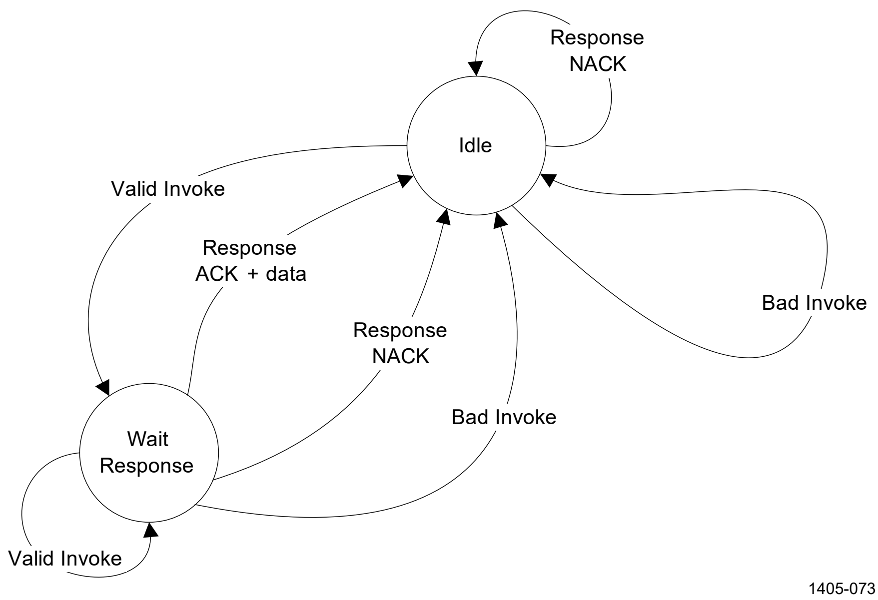


Figure 2 HMM105 State Machine

Examples of Communication Flow

The basic data transfer communication flows are presented in the following figures. Each arrow represents an I^2C read or write operation The examples are simplified presentations. For actual message content, see section Commands on page 11.

For example, Figure 3 below shows the following message sequence:

1. The master invokes the Get_Parameter command to read the RH parameter using an I^2C write operation. This command prepares the slave to wait for a read operation from the master.
2. The master begins an I^2C read operation to read the results of the Get_Parameter RH command. The slave sends the requested RH result and an ACK signal to the master.

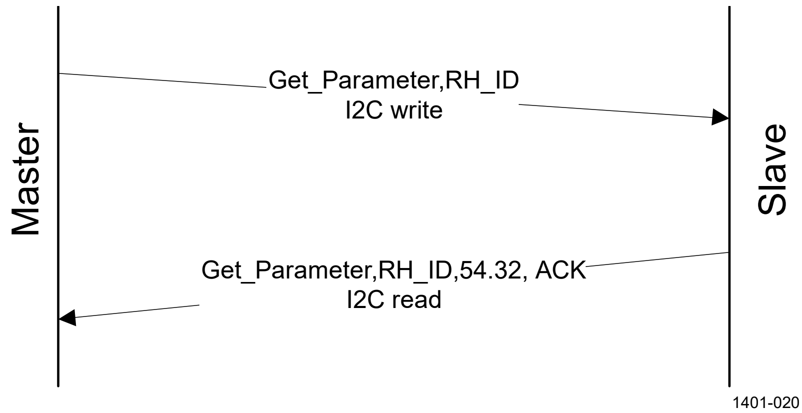


Figure 3 Get_Parameter with ACK

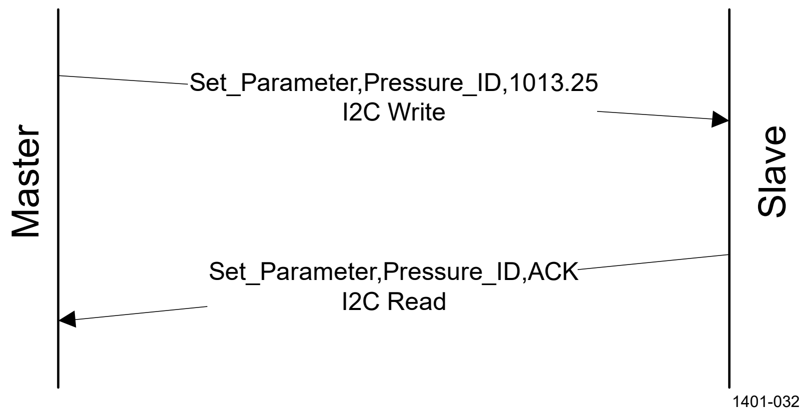


Figure 4 Set_Parameter with ACK

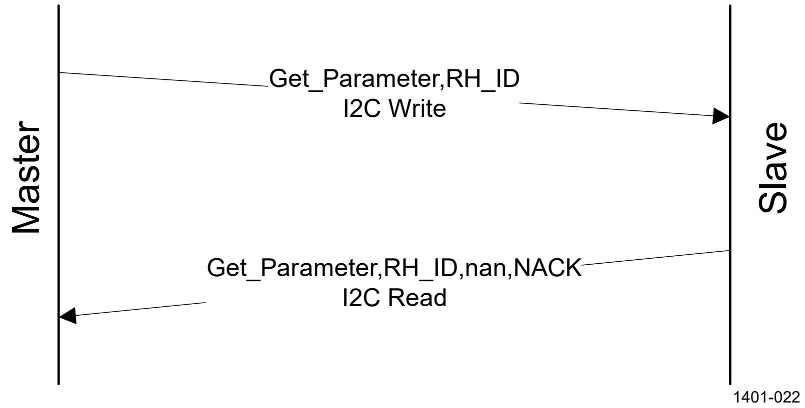


Figure 5 Get_Parameter with NACK

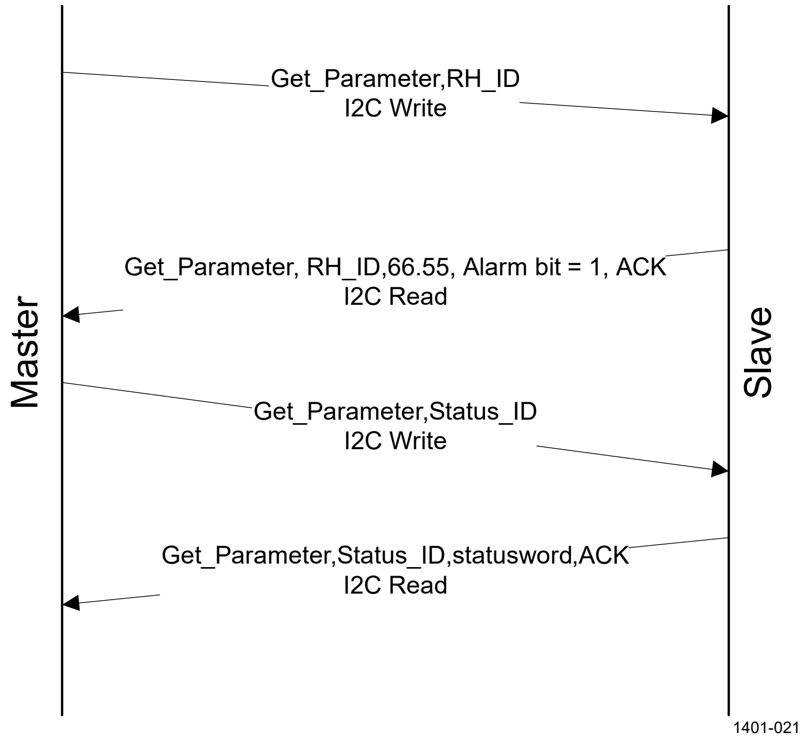


Figure 6 Get_Parameter with Alarm

NOTE	Alarm can be any of the state bits in status byte.

Timing

The minimum time delay that the master must wait between the invoke and response messages depends on the operation. If the operation includes a write to the non-volatile memory, it takes more time than other operations.

Table 8 Timing

Operation	Minimum delay
Normal delay between invoke and response	10 ms
Delay when write to non-volatile memory	300 ms

Status Byte

The status byte gives the master device information about the slave device's state, and information about the communication status with the ACK/NACK bit.

The status byte has bits for signaling Status, Warning, Error and Critical error statuses. These bits are cleared when the status word is read. Bits are set when the corresponding status or error state changes in the slave device.

Table 9 Status Byte

Bit#	Description
0	ACK/NACK
1	Critical error
2	Error
3	Warning
4	Status
5	
6	
7	

Checksum

HMM105 uses a CRC16-CCITT checksum in its protocol frame. CCITT (x.25) polynomial is $X^{16}+X^{12}+X^{5}+1$ = 11021h. Initial value is FFFFh and the final value is XORed with FFFFh. The checksum uses a bit reversed algorithm.

When the invoke message is received, the checksum is calculated using bytes from the command byte to the last byte of the data field. The checksum needs to be valid before data can be accepted. If the checksum is not valid, the message is rejected and the internal state of the GMP231 is set to Idle.

When the response message is transmitted, the checksum is calculated using bytes from the status byte to the last byte of the data field.

Status Word

Status word is used to monitor the status of the HMM105 device. It can be read in the same way as any other slave parameter. Status word is used as 32-bit long bit field. Each bit represents a state of some error or other essential status.

Changes in status word bits can be monitored by reading the status byte. See section Status Byte on page 9.

Table 10 Status Word Content

Bit#	Type	Purpose
0	Critical Error	
1	Critical Error	Parameter memory corrupted
2	Critical Error	Parameter read failed
3	Critical Error	Parameter write failed
4	Error	
5	Error	RH measurement error
6	Error	T measurement error
7	Error	
8	Error	
9	Error	
10	Error	
11	Error	
12	Error	
13	Error	
14	Warning	
15	Warning	
16	Warning	
17	Warning	
18	Warning	
19	Status	
20	Status	
21	Status	
22	Status	
23	Status	
24	Status	
25	Status	
26	Status	
27	Status	
28	Status	
29	Status	
30	Status	
31	Status	

Commands

Get_Interface_Version

The Get_Interface_Version command (ID 80h) reads the interface version information. Recommended for verifying version compatibility before reading or writing parameters.

Table 11 Get_Interface_Version Invoke Message

Message Segment	Length	Content
I^2C address	1 byte	I^2C address (write)
Command ID	1 byte	Always 80h
Device address	1 byte	See Table 5 on page 4
Frame length	1 byte	Always 05h
Checksum	2 bytes	See section Checksum on page 9

Table 12 Get_Interface_Version Response Message

Message Segment	Length	Content
I^2C address	1 byte	I^2C address (read)
Status	1 byte	See section Status Byte on page 9
Command ID	1 byte	Always 80h
Device address	1 byte	See Table 5 on page 4
Frame length	1 byte	Always 0Ah
Data	1 byte	Device version
	1 byte	Protocol frame version
	1 byte	Command set version
	1 byte	Parameter set version
Checksum	2 bytes	See section Checksum on page 9

Get_Parameter

The Get_Parameter command (ID 81h) reads parameter values. You can use it to read both measurement results and operational parameters. If the number of bytes the master tries to read exceeds the message size, the slave sends FFh bytes.

Table 13 Get_Parameter Invoke Message

Message Segment	Length	Content
I²C address	1 byte	I²C address (write)
Command ID	1 byte	Always 81h
Device address	1 byte	See Table 5 on page 4
Frame length	1 byte	Always 06h
Data	1 byte	ID of parameter to be read
Checksum	2 bytes	See section Checksum on page 9

Table 14 Get_Parameter Response Message

Message Segment	Length	Content
I²C address	1 byte	I²C address (read)
Status	1 byte	See section Status Byte on page 9. If requested parameter ID was unknown, NACK bit is set in the status byte.
Command ID	1 byte	Always 81h
Device address	1 byte	See Table 5 on page 4
Frame length	1 byte	Varies according to data length, 07h ... 39h.
Data	1 byte	ID of parameter
	1 ... 50 bytes	Value of parameter. If requested parameter ID is unknown, these data bytes are left out of the response.
Checksum	2 bytes	See section Checksum on page 9

Example: Read RH Measurement Result

Table 15 Get_Parameter Invoke Message Example

Message Bytes (hex)	Content
2F	I^{2}C address
81	Get_Parameter command
2F	Device address
06	Frame length
4F	RH Parameter ID
6A	CRC high
D4	CRC lo

Table 16 Get_Parameter Response Message Example

Message Bytes (hex)	Content
2F	I^{2}C address
00	Status byte (no errors, ACK)
81	Get_Parameter command
2F	Device address
0B	Frame length
4F	RH Parameter ID
D4	RH value 14.43086624 %RH
E4	
66	
41	
85	CRC high
6A	CRC lo

Set_Parameter

The Set_Parameter command (ID 82h) writes parameter values to non-volatile memory. You can use it to set the operational parameters of the HMM105. Depending on the change, the functionality of the HMM105 may not change immediately.

The length of the invoke message varies depending on the data length. The master must write as many bytes as indicated in the frame length of the invoke message. If the master tries to write a longer or shorter message, HMM105 rejects the message.

Table 17 Set_Parameter Invoke Message

Message Segment	Length	Content
I^2C address	1 byte	I^2C address (write)
Command ID	1 byte	Always 82h
Device address	1 byte	See Table 5 on page 4
Frame length	1 byte	Varies according to data length, 07h ... 38h.
Data	1 byte	Parameter ID
	1 ... 50 bytes	Parameter value
Checksum	2 bytes	See section Checksum on page 9

Table 18 Set_Parameter Response Message

Message Segment	Length	Content
I^2C address	1 byte	I^2C address (read)
Status	1 byte	See section Status Byte on page 9.
Command ID	1 byte	Always 82h
Device address	1 byte	See Table 5 on page 4
Frame length	1 byte	Always 08h.
Data	1 byte	Parameter ID
	1 byte	Return code. See Table 19 below.
Checksum	2 bytes	See section Checksum on page 9

Table 19 Set_Parameter Return Codes

Code	Meaning
0	Ok
1	Unknown parameter ID
2	Not writeable parameter
3	Value field too long
4	Value field too short
5	Data value not accepted

Example: Set Compensation Pressure

Table 20 Example Set_Parameter Invoke Message

Message Bytes (hex)	Content
2F	I²C address
82	Set_Parameter command
2F	Device address
0A	Frame length
40	Parameter ID of compensation pressure
00	Pressure 1000 hpa
00	
7A	
44	
D8	CRC high
31	CRC lo

Table 21 Example Set_Parameter Response Message

Message Bytes (hex)	Content
2F	I²C address
00	Status byte (error active, ACK)
82	Set_parameter command
2F	Device address
08	Frame length
40	Parameter ID of compensation pressure
00	Return code: Ok
D6	CRC high
5C	CRC lo

Get_Parameter_Info

The Get_Parameter_Info command (ID 83h) reads the properties of a single parameter from the HMM105. The command is useful for adapting to parameters that have been added in a new software version. If the requested parameter ID is not valid, data type **Unknown Parameter ID** is returned.

Table 22 Get_Parameter_Info Invoke Message

Message Segment	Length	Content
I^{2}C address	1 byte	I^{2}C address (write)
Command ID	1 byte	Always 83h
Device address	1 byte	See Table 5 on page 4
Frame length	1 byte	Always 06h.
Data	1 byte	Parameter ID
Checksum	2 bytes	See section Checksum on page 9

Table 23 Get_Parameter_Info Response Message

Message Segment	Length	Content
I^{2}C address	1 byte	I^{2}C address (read)
Status	1 byte	See section Status Byte on page 9.
Command ID	1 byte	Always 83h
Device address	1 byte	See Table 5 on page 4
Frame length	1 byte	Always 12h.
Data	1 byte	Parameter ID
	1 byte	Parameter datatype. See Table 24 below.
	1 byte	Parameter length in bytes.
	1 byte	Parameter persistence. Describes if the parameter value is volatile (lost at reset) or non-volatile (survives reset). See Table 25 below.
	8 bytes	Parameter name. If the name is shorter than 8 bytes, extra bytes are filled with 00h.
Checksum	2 bytes	See section Checksum on page 9

Table 24 Parameter Data Types

Code	Meaning
0	Unknown parameter ID
1	Byte
2	Integer (16-bit)
3	Unsigned Integer (16–bit)
4	Float (32-bit)
5	String

Table 25 Parameter Persistence

Code	Meaning
0	Void
1	Volatile
2	Non volatile

Adjust

The Adjust command (ID 84h) controls the user adjustment sequence of the HMM105.

Table 26 Adjust Invoke Message

Message Segment	Length	Content
I^2C address	1 byte	I^2C address (write)
Command ID	1 byte	Always 84h
Device address	1 byte	See Table 5 on page 4
Frame length	1 byte	Varies according to data length, 07h ... 0Bh.
Data	1 byte	Adjustment subcommand. See Table 28 below.
	1 byte	Parameter to be adjusted. See Table 29 below.
	4 bytes	Value of parameter. Used only with subcommands 2 and 3.
Checksum	2 bytes	See section Checksum on page 9

Table 27 Adjust Response Message

Message Segment	Length	Content
I^2C address	1 byte	I^2C address (read)
Status	1 byte	See section Status Byte on page 9.
Command ID	1 byte	Always 84h
Device address	1 byte	See Table 5 on page 4
Frame length	1 byte	Always 07h.
Data	1 byte	Return code. See Table 30 on page 18.
Checksum	2 bytes	See section Checksum on page 9

Table 28 Adjustment Subcommands

Code	Adjustment operation
0	Start 1 point adjustment
1	Start 2 point adjustment
2	Record measured point 1, reference value is given in parameter
3	Record measured point 2, reference value is given in parameter
4	Cancel adjustment (previous adjustment is reverted)
5	End (adjustment is saved and new values are taken in use)
6	Revert to factory calibration (clear user adjustment)

Table 29 Adjustment Parameters

Code	Purpose
0	All parameters (valid with revert to factory calibration)
1	Not used
2	T
3	Not used
4	RH

Table 30 Adjustment Return Codes

Code	Meaning
0	Ok
1	Function not supported
2	Sequence error
3	Recorded-Reference difference too large
4	2-point adjustment: Points too close

Adjusting Measurement

HMM105 can be adjusted while it remains installed in an incubator. Create the reference environment inside the chamber, and use the **Adjust** command to perform the adjustment sequence.

Adjustment can fail for a number of reasons. Refer to the list of return codes for the Adjust command in Table 30 above.

One Point Adjustment

To perform a one point adjustment of relative humidity (RH) or temperature (T), you must perform the following sequence:

1. Using the **Adjust** command, start 1 point adjustment for the chosen parameter.
2. Place the sensor in the reference environment and wait until the measurement has stabilized.
3. Using the **Adjust** command, record the measured point and give the reference value.
4. Use the **Adjust** command to end the adjustment.

Two Point Adjustment

To perform a two point adjustment of relative humidity (RH) or temperature (T), you must perform the following sequence:

1. Using the **Adjust** command, start 2 point adjustment.
2. Place the sensor in the low end reference and wait until the measurement has stabilized.
3. Using the **Adjust** command, record the first measured point and give the first reference value.
4. Place the sensor in the high end reference and wait until the measurement has stabilized.
5. Using the **Adjust** command, record the second measured point and give the second reference value.
6. Use the **Adjust** command to end the adjustment.

Data Registers

The register table describes all available parameters and their properties. The properties are: parameter meaning, ID, name, data type, length and persistence. Persistence defines if the parameter is saved in EEPROM (non-volatile memory) or in RAM. Make sure not to write excessively to the EEPROM, as it has a maximum lifetime of approximately 30000 cycles.

Data Formats

Table 31 Data Formats

Data type name	Size (Bytes)	Other
Byte	1	Value range 0...255
Integer	2	Value range -32768...32767
Unsigned integer	2	Value range 0…65535
Long integer	4	Value range −2,147,483,648…2,147,483,647
Unsigned long integer	4	Value range 0…4,294,967,295
Float	4	According to IEEE-754. NaN (7FC00000) is returned if no value is available
String	Max. 50	All characters accepted (00h…FFh)

NOTE Maximum string size refers to maximum communication buffer size. Actual reserved storage space for each parameter can be seen in Table 32 on page 20

NOTE Integers and floating point numbers are sent LSB first.

Register Table

Table 32 HMM105 Register Table

Meaning	Name	ID	Size (bytes)	Type	Read/ Write	Persistent	Other
Factory information							
Device Address	ADDR	0	1	Byte	R	Yes	Normally 0x2F. Only 3 LSBs are changeable. Address range is therefore 1 ... 7.
Device serial number	SNUM	1	12	Str	R	Yes	Example: A1234567
Software version number	VERS	4	20	Str	R	Yes	Example: 1.2.3.4567
Factory calibration date DDMMYYYY	CDATE	6	4	Uint	R	Yes	Example: 19052014
Factory calibration info	CTEXT	7	19	Str	R	Yes	Example: "CAL INFO"
General parameters and variables							
Device status word	STATUS	8	4	Str	R	No	See section Status Word on page 10.
Unit representation selection	UNITS	10	2	Uint	R/W	Yes	0 = Metric units 1 = Nonmetric units
Module batch number	BNUM	11	4	Str	R	Yes	Example: J413
RH result	RH	79	4	Float	R	No	%RH
Temperature result	T	65	4	Float	R	No	Degrees of Celsius/Fahrenheit (see UNITS)
Dew/Frost point Tdf	TDF	88	4	Float	R	No	Degrees of Celsius/Fahrenheit (see UNITS)
Ambient pressure in hPa	P_AMB	64	4	Float	R/W	Yes	Ambient pressure in hPa. Used for RH result.
RH Gain	RH_G	96	4	Float	R/W	Yes	RH gain
RH Offset	RH_O	97	4	Float	R/W	Yes	RH offset
Temperature Gain	T_G	94	4	Float	R/W	Yes	Temperature gain adjustment
Temperature Offset	T_O	95	4	Float	R/W	Yes	Temperature offset adjustment
Temperature reference point 1	T_RP1	90	4	Float	R/W	Yes	Temperature adjustment reference point 1. Write ref 1 here when using the calibration command.
Temperature reference point 2	T_RP2	91	4	Float	R/W	Yes	Temperature adjustment reference point 2. Write ref 2 here when using the calibration command.
RH reference point 1	RH_RP1	92	4	Float	R/W	Yes	RH adjustment reference point 1. Write ref 1 here when using the calibration command.
RH reference point 2	RH_RP2	93	4	Float	R/W	Yes	RH adjustment reference point 2. Write ref 2 here when using the calibration command.

www.vaisala.com

M211638EN