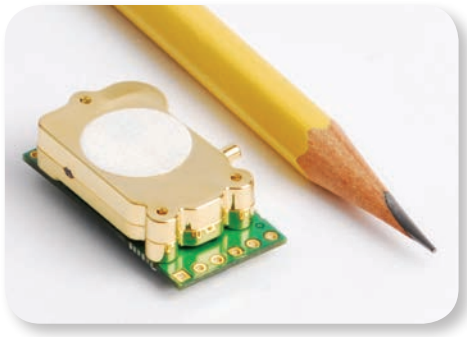
TELAIRE®

Application Note

T67xx CO_2 Sensor Module



OVERVIEW

The purpose of this document is to outline the required interface design and communication protocol for the T67xx CO_2 Sensor modules. The intended audience is any developer who wishes to query the sensor for information utilizing either the I^2C, PWM or UART interfaces.

All information about the performance of the sensor can be found on the Telaire website. Please refer to the spec sheets found at www.telaire.com for the latest specifications.

Amphenol

Advanced Sensors

TABLE OF CONTENTS

T67XX CO_2 SENSOR MODULE

INSTALLATION AND MOUNTING

Design Considerations

To maximize the performance of T67xx module, it is important to plan an appropriate location for the sensor. Airflow and proper exposure to ambient air must be secured for T67xx module to ensure optimal performance. Inadequate airflow will deteriorate the response time of the sensor. Too high airflow may cause extra noise on the sensor output. The preferred installation should have more than 1 mm clearance above the diffusion fitter on top of the sensor and should be in a location not exposed to high airflow or turbulence around the sensor. Similarly, avoid excessive heat, preferably placing the sensor away from high heat dissipating components on the PCB.

The sensor is designed for benign environments. The performance and reliability may be compromised in environments that contain corrosive or caustic gases including but not limited to Ammonia, Chlorine, NOx and Ozone. Care must be taken to ensure that the sensor is not exposed to these compounds under any operating condition.

Installation

The T67xx module is a sensitive electronics assembly, so effort should be made to minimize exposure to excess heat from any type of soldering operation. Excess exposure to heat from installation is known to create small shift in calibration of the sensor. Although the ABC Logic™ algorithm (where applied) will correct these minor fluctuations within the first 24 hours of operation, the user should minimize overexposure to heat when soldering to negate unexpected operation after installation. Typically, an expose of not more than 10 sec. using a soldering iron set to 750°F (400°C) will minimize the influence of heat on the measurement of the T67xx module. The use of low temperature solder is also recommended.

It is recommended to handle the sensor by the edges of the PCBA. Extra stress applied to the PCB or the gold OBA assembly can cause mechanical stress that will create temporary shift in the calibration of the sensor. Although he ABC Logic™ algorithm will correct for these shifts in the first 24 hours of operation, extra care in handling will minimize the risk of variations that can be seen out of the box.

ESD Precautions

Precautions should be taken to observe specified limits and prevent damage from electrostatic discharge or rough handling. Please refer to ANSI/ESD S20. 20-1999 for more information on preventing ESD damage for more information on proper electronic assembly practices.

OPERATION DETAILS

ABC LOGIC™

Automatic Background Logic, or ABC Logic, is a patented self-calibration technique that is designed to be used in applications where concentrations will drop to outside ambient conditions (400 ppm) at least one time (15 minutes) in a 7day period, which is typically seen during unoccupied periods.

*Full accuracy to be achieved utilizing ABC Logic. With ABC Logic enabled, the sensor will typically reach its operational accuracy after 25 hours of continuous operation at a condition that it was exposed to ambient reference levels of air at 400 ppm ±10 ppm CO_2. Sensor will maintain accuracy specifications with ABC Logic enabled, given that it is at least one time per 7 days exposed to the reference value and this reference value is the lowest concentration to which the sensor is exposed.

ABC Logic requires continuous operation of the sensor for increments of at least 4 hours each.

Note: Applies when used in typical residential ambient air. Consult Amphenol Advanced Sensors if other gases or corrosive agents are part of the application environment.

T67XX CO_2 SENSOR MODULE

ENVIRONMENTAL

Operating Temperature: -10°C to 60°C (14°F to 140°F) 0 to 95% RH, non-condensing

Storage Temperature: -30°C to 70°C (-22°F to 158°F)

ABSOLUTE MAXIMUM RATINGS

In order to avoid damage to the sensor, the following parameters should never be exceeded at any time during operation:

Parameter	Min (V)	Max (V)
Supply Voltage (V+, pin 3)	-0.3	7
Ground (V-, pin 4)	-0.3	0.3
TX / SDA, RX / SCL (pin 1,2)	-0.3	7
MD DIR / PWM (pin 5)	-0.3	7
UART/I^2C Select (pin 6)	-0.3	3

Power-On Sequence

The sensor is capable of responding to commands after power-on, but operational accuracy of sensor won't happen until 120 sec have elapsed. The sensor will reach full accuracy / warm up after 10 min. of operation (exception is '–R15' calibration).

Power Supply Requirements

Supply Voltage: +4.5 – 5.5VDC

Average Current*: 25 (20mA typical)

Peak Current*: 200mA (155mA typical)

**Based on values at a nominal 5VDC input*

Evaluation / Demonstration Kits

Evaluation kits are available in order to help customers develop their Air Quality application. The kit includes a sample of the sensor, cable and software that will enable the user to better understand the performance of the sensor in their intended design. Please contact Telaire for further details or visit the website at www.telaire.com.

Safety

Disclaimer

Telaire makes no warranty, representation or guarantee regarding the suitability of this product for any particular application, including safety critical applications. Nor does Telaire assume any liability arising out of the application or use in any product or circuit. Telaire specifically disclaims all liability without limitation consequential or incidental damages. No statuary or fitness for particular purpose shall be implied.

WARNING! Before installing the product, review the product data sheet and this application guide. The product shall be used only within power supply and electrical input and output limits as specified by the datasheet and application guide. Improper use of the product may result in product damage and property loss and/or personal injury.

Safety While In Use

Before installing, handling, using, or servicing this product, please consult the data sheet and application notes. The product shall be used only within power supply and electrical input and output limits as specified by the datasheet and application guide. Improper use of the product may result in product damage and property loss and/or personal injury. In use of the product, the customer has sole

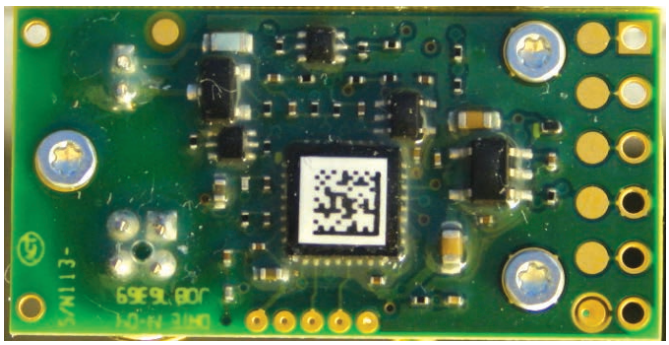
responsibility for designing and implementing a solution which will ensure safe operation (including review of appropriate reliability or required redundancy, mitigation of failure modes, and/or meeting appropriate standards). The customer is responsible for review of any special conditions for use including, but not limited to, environmental conditions, electrical supply, residual risk, etc.). The sensor is designed for benign environments. The performance and reliability may be negatively affected in environments that contain corrosive or caustic gases including but not limited to Ammonia, Chlorine, NOx and Ozone.

MATERIAL CONTENTS

ROHS, REACH and Proposition 65 declaration of conformity are available upon request. Please contact customer service for more details.

INTERFACE CONNECTOR AND OPTIONS

The six pin through-hole connector on the PCB is where power and IO for the sensor are located, see Figure 1 below. A four to six pin, 0.1" (2.54mm) header is typically used to connect the sensor to a controller. There are several different I/O configurations described in the following sections that are supported by the sensor and are determined by the voltage that is measured on pin 6 of the connector at startup.



1. TX / SDA
2. TX / SCL
3. V++
4. GND
5. PWM
6. CTRL / TEST

Figure 1 - T67XX Interface Connector

Note: Precautions should be taken to observe specified limits and prevent damage from electrostatic discharge or rough handling. Please refer to ANSI/ESD S20. 20-1999 for more information on preventing ESD damage and IPC 610 Rev D for more information on proper electronic assembly practices. In addition to this, the sensor does not have internal reverse polarity protection. Care should be taken to connect the sensor to the controller in the correct wiring configuration to avoid damage. Sensor is not designed for hot swapping.

T67XX CO_2 SENSOR MODULE

UART & 1 HZ PWM

Pin 6 is left unconnected by the user and will therefore be left floating. It will be pulled up by an internal 1MΩ resistor. In this condition:

Table 1 – I/O Pin Configuration #1

Pin	Description
1	UART TX (output from sensor)
2	UART Rx (input to sensor)
5	PWM output at approximately 1Hz

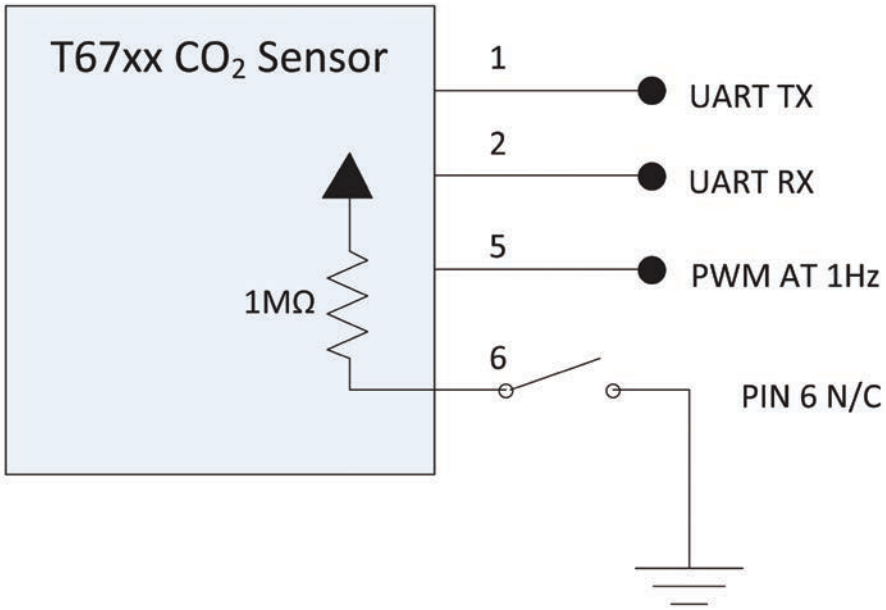


Figure 2 - Simplified Schematic I/O configuration #1

In this condition, the sensor implements a RS-232 serial interface. The default serial conditions are:

- 19200 Baud
- 1 START bit
- 8 DATA bits
- 1 EVEN PARITY bit
- 1 STOP bit

I^2C & 25 KHZ PWM

Pin 6 is grounded by the user. In this condition:

Table 2 - I/O Pin Configuration #2

Pin	Description
1	UART TX (output from sensor)
2	UART Rx (input to sensor)
5	PWM output at approximately 1Hz

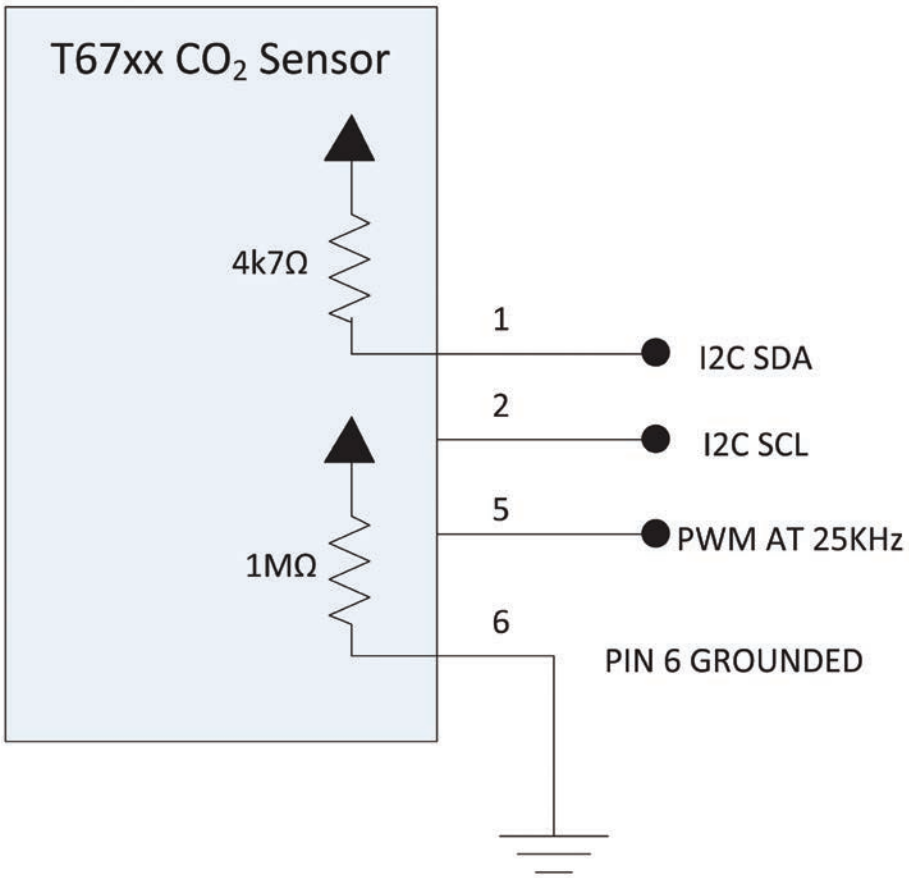


Figure 3 - Simplified Schematic I/O configuration #2

In this condition, the sensor implements an I^2C interface. The default conditions are:

- The sensor only acts as a slave device
- I^2C 7-bit addressing is used. The default slave address is 0x15 (21)
- I^2C 100kbit/s Standard Mode

Note: There is an internal pull up resistor on pin 1 of the I^2C interface. Customer will need to provide an external pull up resistor on pin 2 with a recommended value of 4.7k. I^2C interface can operate at both 3.3V and 5V logic levels.

6.3 UART WITH RS485 TRANSCEVER SUPPORT

Pin 6 is pulled to ground by a resistor between 10kΩ and 100kΩ

Table 3 - I/O Pin Configuration #3

Pin	Description
1	UART TX (output from sensor)
2	UART Rx (input to sensor)
5	Becomes a test input for Telaire and should be left unconnected by the user
6	Becomes an output pin used to control an RS485 transceiver

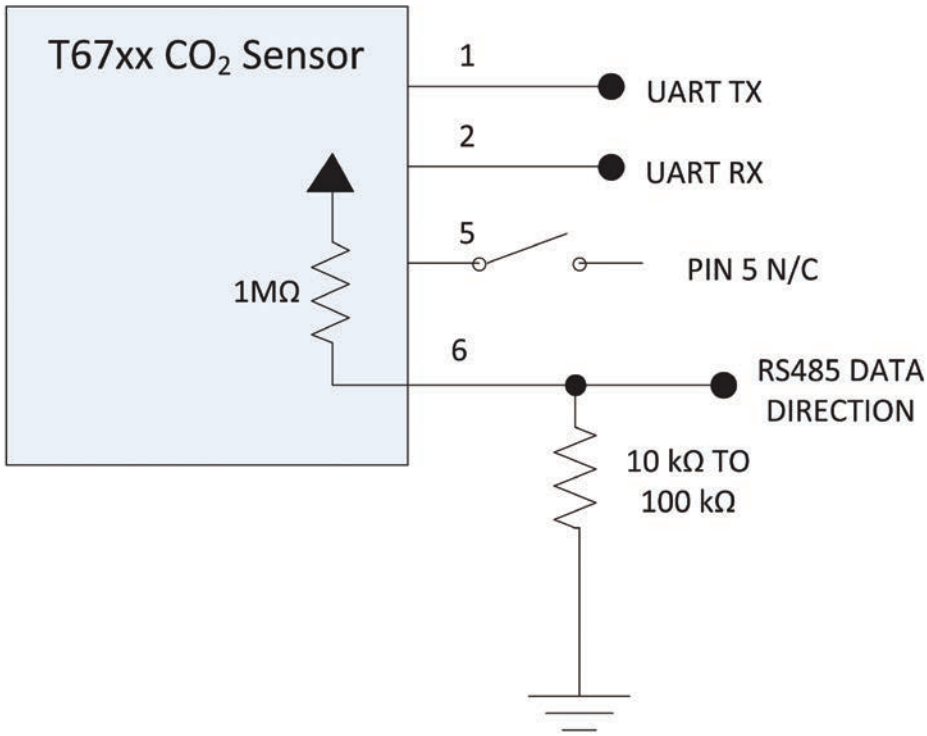


Figure 4 - Simplified Schematic I/O Configuration #3

The condition described by (3) above is useful if the sensor is used in an RS485 dropped node network configuration. This network would be supplied by the user. In this condition pin 6 controls the RS-485 transceiver data-direction logic.

T67XX CO_2 SENSOR MODULE

PWM OUTPUTS

The T67xx sensor has a selectable Pulse Width Modulation (PWM) output feature on Pin 5, based on the state of Pin 6 during power on.

The two types of supported PWM operate at ~1Hz and 25 kHz and are scaled to the range of the sensor as determined by the model. For example, if the model is a T6713-5k, the units PWM output will have 0 to 5000 ppm output range. For the model T6713, the units PWM will have 0 – 2000 ppm output range.

An external pull up resistor to a voltage between 3.3 to 5V is required on the PWM output. Typical value could be anywhere from 4.7 kOhm to 100 kOhm depending on chosen frequency of the PWM output and capacitance of the output load.

The slow ~1Hz PWM output option supports legacy application and was developed to communicate with simple low cost microcontrollers without using complex communication protocols. It allows the user to measure the duration of the pulse, and correlate this to a CO_2 measurements. Each PWM cycle has start sequence including 2 msec of low level followed by 2 msec of high level.

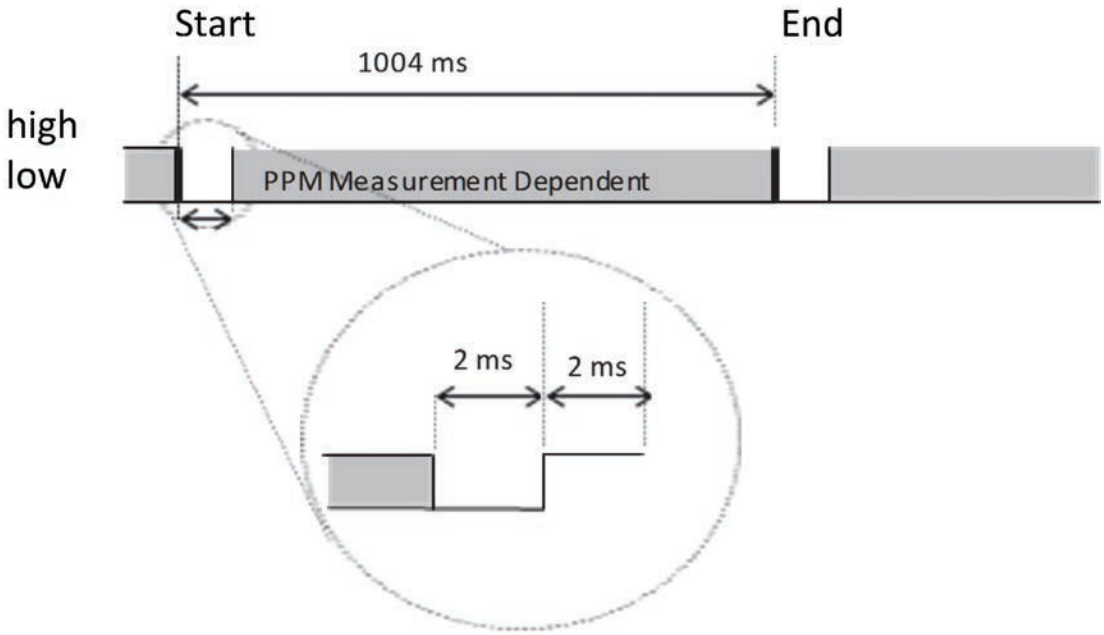


Figure 5 - Details of ~1Hz PWM

In order to convert the pulse to a reading in PPM, the user should use the following equation:

PPM=(t_pulse-2)*2 for 0 – 2000 ppm models

PPM=(t_pulse-2)*5 for 0 – 5000 ppm models

Where:

t_pulse=Measured duration of pulse in msec

PPM=Measured CO_2 Value

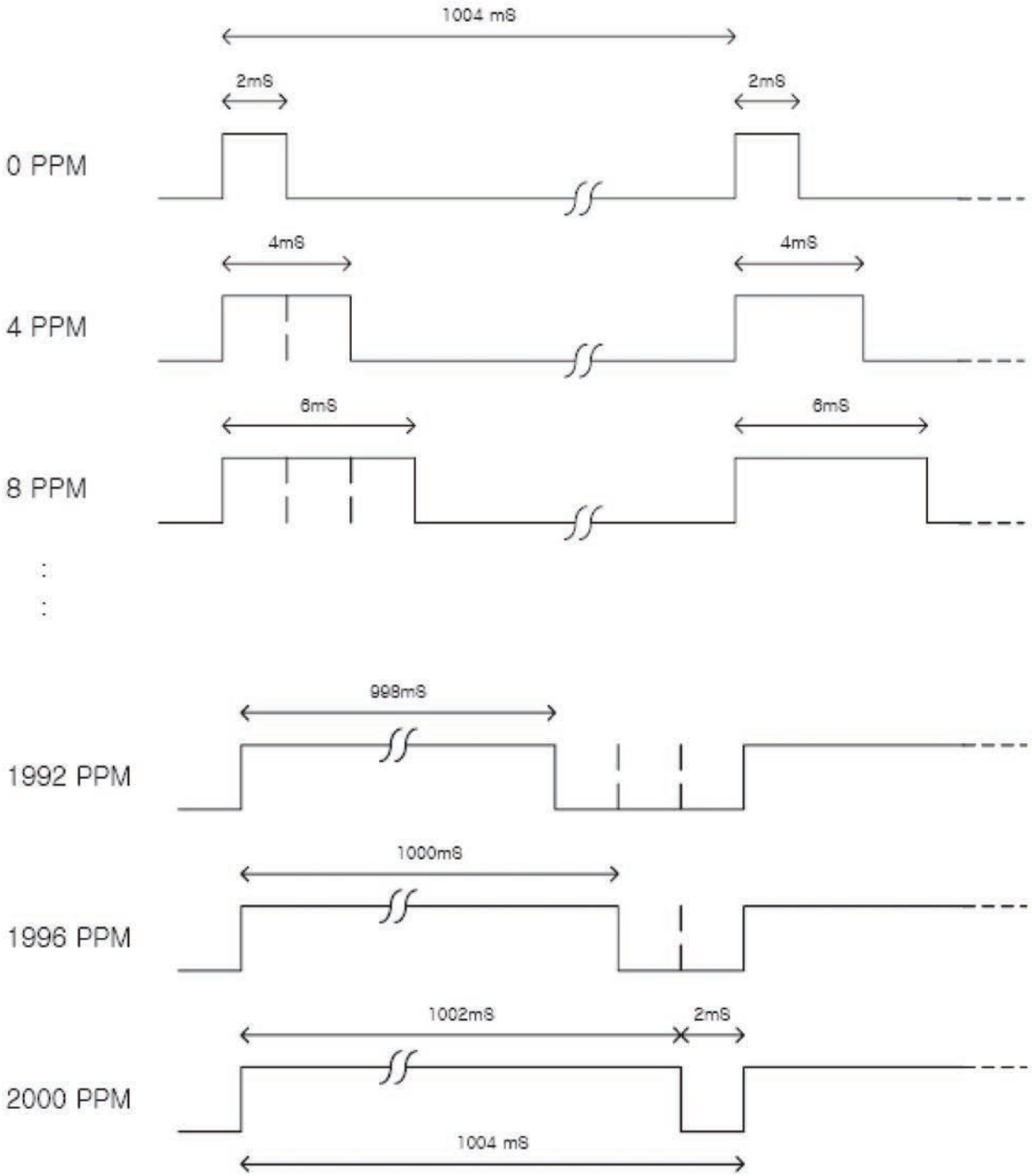


Figure 6 - Example of the PWM for -2K model

T67XX CO_2 SENSOR MODULE

The 25 kHz PWM output is a traditional PWM with 40 micro seconds period. It can be can be filtered to create an analog output. Below is one of possible circuit designs to convert 25 kHz PWM to an analog output, where R3 and R4 will set the gain of the amplifier.

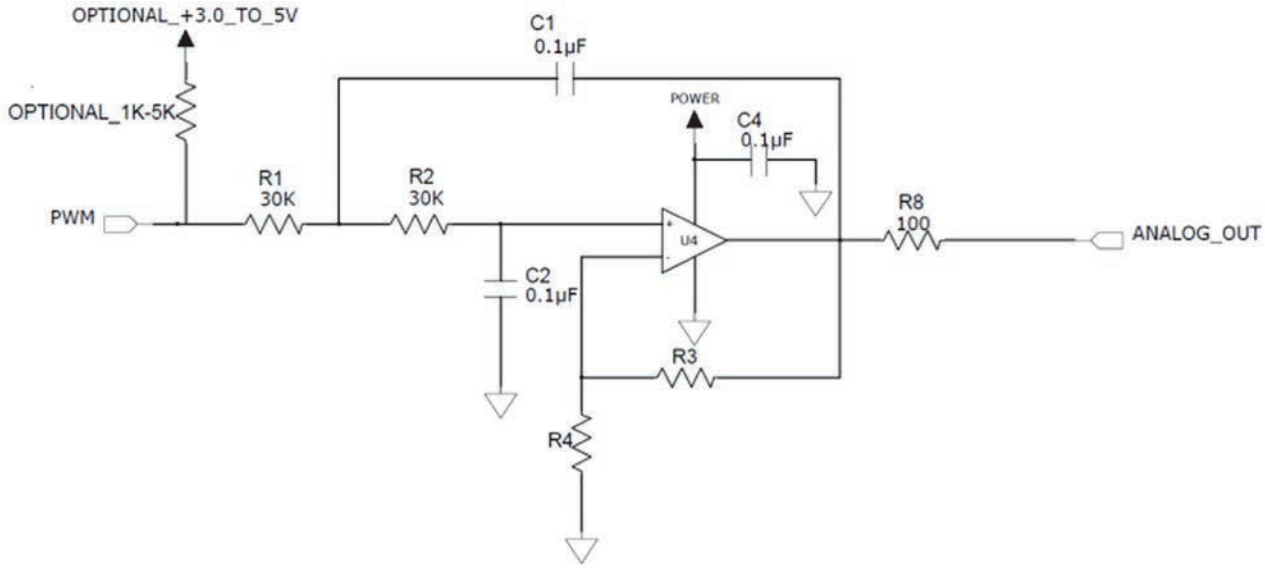


Figure 7 - Circuit design for 25 kHz PWM output conversion to analog voltage

T67XX CO_2 SENSOR MODULE

COMMUNICATION – MODBUS PROTOCOL

The T67xx sensor uses the Modbus protocol for all communications. The documents are freely available on the Modbus WEB site at http://www.modbus.org/specs.php.

UART (RS232/RS485)

For UART communications, reference the recommendations found in the document 'Modbus Serial Line Protocol and Implementation Guide V1.02'. This document includes detailed information on how to calculate the required CRC (Cyclical Redundancy Checking) bytes.

It is important to note that for Modbus over serial lines (i.e., RS-232 and RS-485) the user MUST INCLUDE THE CYCLICAL REDUNDANCY CHECK (CRC) fields at the end of the Modbus request. The CRC calculation is not necessary for communications over the I^2C interface.

I^2C

The I^2C implementation does use the Modbus protocol, and wraps the message in I^2C format, however the CRC is not included in this case. Please reference to the "I^2C specification and user's manual" at the following URL for details on I^2C communication www.nxp.com/documents/user_manual/UM10204.pdf.

The T67xx sensor always operates as a slave. In the examples below the implementer using the I^2C interface will need to communicate with the sensor as either a master-transmitter (for Modbus requests) or a master-receiver (for Modbus responses). The sensor will not initiate communications, i.e., it will not become a master and respond to the request. It is up to the master to read the response from the sensor.

Since I^2C clock stretching is enabled and used by T67xx. The I^2C master must support clock stretching.

I^2C Timing Considerations

Unlike the UART interface, where the sensor's serial port controls the response back to the master device, the I^2C master can ask for the response immediately after sending the request because the I^2C master controls the clock. If the master asks for a response immediately, without giving the sensor time to formulate a response, the master will receive a string of zeros.

It is suggested that the master send the request, wait 5 to 10 milliseconds and then ask for the response. This time does depend on bus loading and board layout but carefully constructed test setups have demonstrated that the sensor can respond within 1 millisecond in controlled conditions with a data rate of 100kbps. The suggested delay of 5 to 10 milliseconds should be adequate for almost all conceivable cases.

T67XX CO_2 SENSOR MODULE

COMMAND SUMMARY

Table 4 – T67xx Modbus Command Summary

Name	Modbus Register	Register Address	Data Type	Description
FIRMWARE_REVISION	Input (RO)	‘1389’H ‘5001’D	uint16_t	Returns the firmware revision from the sensor
STATUS	Input (RO)	‘138A’H ‘5002’D	uint16_t	Returns a status register from the sensor. Additional details are below
GAS PPM	Input (RO)	‘138B’H ‘5003’D	uint16_t	The current gas ppm calculation
RESET DEVICE	Coils (WO)	‘03E8’H ‘1000’D	uint16_t	Reset the sensor over the Modbus network
START SINGLE POINT CAL	Coils (WO)	‘03EC’H ‘1004’D	uint16_t	Start a single-point calibration
SLAVE ADDRESS	Holding (RW)	‘0FA5’H ‘4005’D	uint16_t	Change of sensor address (default address is ‘15’H (‘21’D)
ABC LOGIC™ ENABLE / DISABLE	Coils (RW)	‘03EE’H ‘1006’D	uint16_t	Enable or disable ABC Logic™
MEASURE ON DEMAND	Holding (RW)	‘100B’H ‘4107’D	uint16_t	Enable or disable Measure on Demand

Where RO is Read Only, WO is Write Only and RW is Read or Write.

Examples are given in the following pages.

Firmware Version

This command will return the current firmware revision of the sensor. Use the Modbus Read Input Registers function (4) and read one (1) register at address ‘1389’H (‘5001’D).

Example 1 - Modbus request/response to read the firmware revision (UART)

Modbus Request UART

‘15’H	Slave Address (default is ‘15’H’)
‘04’H	Function Code
‘13’H	Starting Address (MSB)
‘89’H	Starting Address (LSB)
‘00’H	Number of registers to read (MSB)
‘01’H	Number of registers to read (LSB)
xx	CRC (LSB)
xx	CRC (MSB)

Modbus Response UART

‘15’H	Slave Address (default is ‘15’H’)
‘04’H	Function Code
‘02’H	Byte Count
xx	Status (MSB)
xx	Status (LSB)
xx	CRC (LSB)
xx	CRC (MSB)

Example 2 - Modbus request/response to read the firmware revision (I^2C)

The default T67xx I^2C slave address is ‘15’H and not shown.

Modbus Request I^2C *(Master-Transmitter/Slave-Receiver)*

‘04’H	Function Code
‘13’H	Starting Address (MSB)
‘89’H	Starting Address (LSB)
‘00’H	Number of registers to read (MSB)
‘01’H	Number of registers to read (LSB)

Modbus Response I^2C *(Master-Receiver/Slave-Transmitter)*

‘04’H	Function Code
‘02’H	Byte Count
xx	Status (MSB)
xx	Status (LSB)

Status

This command returns a register with the status of various functions on the sensor. The user must verify that no error condition exists in the sensor. Also, user must verify that the sensor has completed the warm-up stage.

Use the Modbus Read Input Registers function (4) and read one (1) register at address 138A'H ('5002'D).

Example 3 – Modbus request/response to read the Status Register (UART)

Modbus Request (UART)

'15'H	Slave Address (default is '15'H')
'04'H	Function Code
'13'H	Starting Address (MSB)
'8A'H	Starting Address (LSB)
'00'H	Input registers to read (MSB)
'01'H	Input registers to read (LSB)
xx	CRC (LSB)
xx	CRC (MSB)

Modbus Response (UART)

'15'H	Slave Address (default is '15'H')
'04'H	Function Code
'02'H	Byte Count
xx	Status (MSB)
xx	Status (LSB)
xx	CRC (LSB)
xx	CRC (MSB)

Example 4 – Modbus request/response to read the Status Register (I^2C)

The default T67xx I^2C slave address is '15'H and not shown.

Modbus Request (I^2C) *(Master-Transmitter/Slave-Receiver)*

'04'H	Function Code
'13'H	Starting Address (MSB)
'8A'H	Starting Address (LSB)
'00'H	Input registers to read (MSB)
'01'H	Input registers to read (LSB)

Modbus Response (I^2C) *(Master-Receiver/Slave-Transmitter)*

'04'H	Function Code
'02'H	Byte Count
xx	Status (MSB)
xx	Status (LSB)

T67XX CO_2 SENSOR MODULE

The **STATUS** register is a bit-vector where each bit represents the status of some function within the sensor. Not all bits are assigned.

Bit position	HEX	Comments
xxxxxxxx, xxxxxxx1	‘0001’H	Error condition
xxxxxxxx, xxxxxx1x	‘0002’H	Flash error
xxxxxxxx, xxxxx1xx	‘0004’H	Calibration error
xxxxxxxx, xxxx1xxx	‘0008’H	NA
xxxxxxxx, xxx1xxxx	‘0010’H	NA
xxxxxxxx, xx1xxxxx	‘0020’H	NA
xxxxxxxx, x1xxxxxx	‘0040’H	NA
xxxxxxxx, 1xxxxxxx	‘0080’H	NA
xxxxxxx1, xxxxxxxx	‘0100’H	NA
xxxxxx1x, xxxxxxxx	‘0200’H	NA
xxxxx1xx, xxxxxxxx	‘0400’H	Reboot
xxxx1xxx, xxxxxxxx	‘0800’H	Warm-up mode
xxx1xxxx, xxxxxxxx	‘1000’H	NA
xx1xxxxx, xxxxxxxx	‘2000’H	NA
x1xxxxxx, xxxxxxxx	‘4000’H	NA
1xxxxxxx, xxxxxxxx	‘8000’H	Single point calibration

For error conditions, a '1' indicates an error; a '0' indicates no error. Flash error is fatal (i.e., there is no recovery). Calibration errors can be cleared by running the calibration procedure again with successful results.

For calibration conditions, a '1' indicates that the calibration cycle is currently in progress. No other calibration cycle can start while one is currently in progress and will result in an error being reported by the Modbus response to the new calibration request.

If the warm-up bit is set the sensor is in a mode where internal registers are being initialized and gas (ppm) data is not necessarily correct.

GAS PPM

This command reports the current gas measurement in ppm. Use the Modbus Read Input Registers function (4) and read one (1) register at address '138B'H ('5003'D).

Example 5 – Modbus request/response to read the Gas PPM Register (UART)

Modbus Request (UART)

'15'H	Slave Address (default is '15'H')
'04'H	Function Code
'13'H	Starting Address (MSB)
'8B'H	Starting Address (LSB)
'00'H	Input registers to read (MSB)
'01'H	Input registers to read (LSB)
xx	CRC (LSB)
xx	CRC (MSB)

Modbus Response (UART)

'15'H	Slave Address (default is '15'H')
'04'H	Function Code
'02'H	Byte Count
xx	MSB of the 16-bit gas ppm data
xx	LSB of the 16-bit gas ppm data
xx	CRC (LSB)
xx	CRC (MSB)

Example 6 – Modbus request/response to read the Gas PPM Register (I^2C)

The default T67xx I^2C slave address is '15'H and not shown.

Modbus Request (I^2C) *(Master-Transmitter/Slave-Receiver)*

'04'H	Function Code
'13'H	Starting Address (MSB)
'8B'H	Starting Address (LSB)
'00'H	Input registers to read (MSB)
'01'H	Input registers to read (LSB)

Modbus Response (I^2C) *(Master-Receiver/Slave-Transmitter)*

'04'H	Function Code
'02'H	Byte Count
xx	MSB of the 16-bit gas ppm data
xx	LSB of the 16-bit gas ppm data

To calculate the gas ppm, do the following: ppm = MSB * 256 + LSB

Example 7 – Calculating Gas PPM

For example, if the Modbus (UART) response was:

'15'H	Slave Address (default is '15'H')
'04'H	Function Code
'02'H	Byte Count
'01'H	MSB of the 16-bit data '01'H = '1'D
'9F'H	LSB of the 16-bit data '9F'H = '159'D
xx	CRC (LSB)
xx	CRC (MSB)

Then the gas concentration will be calculated as:

1*256 + 159 = 415 ppm

RESET

The sensor can be reset using the Modbus command. The reset is immediate and there is no response. The sensor will act as if the power was cycled.

Use the Modbus Write Single Coil function (5) and write '00FF'H to the register at the address '03E8'H ('1000'D) to reset the sensor.

Example 8 - Modbus request to reset the sensor (UART)

Modbus Request (UART)

'15'H	Slave Address (default is '15'H')
'05'H	Function Code
'03'H	Output Address (MSB)
'E8'H	Output Address (LSB)
'FF'H	Output Value (MSB)
'00'H	Output Value (LSB)
xx	CRC (LSB)
xx	CRC (MSB)

Example 9 – Modbus request to reset the sensor (I^2C)

The default T67xx I^2C slave address is '15'H and not shown.

Modbus Request (I^2C) (Master-Transmitter/Slave-Receiver)

'05'H	Function Code
'03'H	Output Address (MSB)
'E8'H	Output Address (LSB)
'FF'H	Output Value (MSB)
'00'H	Output Value (LSB)

Note: The sensor resets immediately. Use the STATUS command to check the sensor status after the RESET.

START SINGLE POINT CALIBRATION

This command starts the single point calibration routine.

The single-point calibration routine is usually done at ambient conditions (~500 ppm, 25 °C) and takes several minutes to complete after being started (~6 minutes). The sensor can be queried during this time for status and current gas ppm readings. The user can check on the status of the calibration by reading the status register and noting if the single-point calibration bit is set. The calibration can be stopped before completing. See examples.

Use the Modbus Write Single Coil function (5) and write '00FF'H to the register at the address 03EC'H ('1004'D) to start the calibration. Write '0000'H during calibration to stop the calibration function.

Example 10 - Modbus request/response to start Single Point Calibration (UART)

Modbus Request (UART)

'15'H	Slave Address (default is '15'H')
'05'H	Function Code
'03'H	Output Address (MSB)
'EC'H	Output Address (LSB)
'FF'H	Output Value (MSB)
'00'H	Output Value (LSB)
xx	CRC (LSB)
xx	CRC (MSB)

Modbus Response (UART)

'15'H	Slave Address (default is '15'H')
'05'H	Function Code
'03'H	Output Address (MSB)
'EC'H	Output Address (LSB)
'FF'H	Output Value (MSB)
'00'H	Output Value (LSB)
xx	CRC (LSB)
xx	CRC (MSB)

Example 11 – Modbus request/response to start Single Point Calibration (I^2C)

The default T67xx I^2C slave address is '15'H and not shown.

Modbus Request (I^2C)

'05'H	Function Code
'03'H	Output Address (MSB)
'EC'H	Output Address (LSB)
'FF'H	Output Value (MSB)
'00'H	Output Value (LSB)

Modbus Response (I^2C)

'05'H	Function Code
'03'H	Output Address (MSB)
'EC'H	Output Address (LSB)
'FF'H	Output Value (MSB)
'00'H	Output Value (LSB)

The calibration command cannot be restarted but can be stopped as detailed in the following example.

Example 12 – Modbus request/response to stop Single Point Calibration (UART)

Modbus Request (UART)

'15'H	Slave Address (default is '15'H')
'05'H	Function Code
'03'H	Output Address (MSB)
'EC'H	Output Address (LSB)
'00'H	Output Value (MSB)
'00'H	Output Value (LSB)
xx	CRC (LSB)
xx	CRC (MSB)

Modbus Response (UART)

'15'H	Slave Address (default is '15'H')
'05'H	Function Code
'03'H	Output Address (MSB)
'EC'H	Output Address (LSB)
'00'H	Output Value (MSB)
'00'H	Output Value (LSB)
xx	CRC (LSB)
xx	CRC (MSB)

Example 13 – Modbus request/response to stop Single Point Calibration (I^2C)

The default T67xx I^2C slave address is '15'H and not shown.

Modbus Request (I^2C)

'05'H	Function Code
'03'H	Output Address (MSB)
'EC'H	Output Address (LSB)
'00'H	Output Value (MSB)
'00'H	Output Value (LSB)

Modbus Response (I^2C)

'05'H	Function Code
'03'H	Output Address (MSB)
'EC'H	Output Address (LSB)
'00'H	Output Value (MSB)
'00'H	Output Value (LSB)

CHANGE SLAVE ADDRESS

It is possible to change the Modbus slave address. The change will only take effect after the sensor has been reset (e.g., over the Modbus RESET command) or power cycled. The change does not take effect immediately.

Use the Modbus Write Single Register function (6) and write the new address to the register at address '4005'D ('0FA5'H).

Example 14 – Changing the default Slave Address (UART)

This example changes the current slave address from default '15'H to '10'H.

Modbus Request (UART)

'15'H	Slave Address (default is '15'H)
'06'H	Function Code
'0F'H	Register Address (MSB)
'A5'H	Register Address (LSB)
'00'H	Register Value (MSB) MSB will always be zero
'10'H	Register Value (LSB) LSB - new slave address
xx	CRC (LSB)
xx	CRC (MSB)

Modbus Response (UART)

'15'H	Slave Address (default is '15'H) still using old slave address
'06'H	Function Code
'0F'H	Register Address (MSB)
'A5'H	Register Address (LSB)
'00'H	Register Value (MSB)
'15'H	Register Value (LSB) still returns old address value because no RESET command issues yet
xx	CRC (LSB)
xx	CRC (MSB)

Example 15 – Changing the default Slave Address (I^2C)

The default T67xx I^2C slave address is ‘15’H and not shown.

This example changes the current slave address from default ‘15’H to ‘10’H.

Modbus Request (I^2C)

‘06’H	Function Code
‘0F’H	Register Address (MSB)
‘A5’H	Register Address (LSB)
‘00’H	Register Value (MSB) MSB will always be zero
‘10’H	Register Value (LSB) LSB - new slave address

Modbus Response (I^2C) still using old ‘15’H address

‘06’H	Function Code
‘0F’H	Register Address (MSB)
‘A5’H	Register Address (LSB)
‘00’H	Register Value (MSB)
‘15’H	Register Value (LSB) still returns old address value because no RESET command issues yet

ABC LOGIC™ ENABLE / DISABLE

ABC Logic™ is manipulated through the Modbus interface by using the Write Single Coil function (5). Coils are viewed as basically switches, relays or discrete (i.e., single bit) inputs and outputs.

The ABC LOGIC CONTROL coil register is ‘03EE’H (‘1006’D). A write of ‘FF00’H (i.e., ON) will enable the ABC Logic™ and a write of ‘0000’H (i.e., OFF) will disable the ABC Logic™ function.

Example 16 – Enable ABC Logic™ (UART)

Modbus Request (UART)

‘15’H	Slave Address (default is ‘15’H’)
‘05’H	Function Code
‘03’H	Register Address (MSB)
‘EE’H	Register Address (LSB)
‘FF’H	Register Value (MSB) Enable ABC Logic™
‘00’H	Register Value (LSB) LSB will always be zero
xx	CRC (LSB)
xx	CRC (MSB)

Modbus Response (UART)

‘15’H	Slave Address (default is ‘15’H’)
‘05’H	Function Code
‘03’H	Register Address (MSB)
‘EE’H	Register Address (LSB)
‘FF’H	Register Value (MSB)
‘00’H	Register Value (LSB)
xx	CRC (LSB)
xx	CRC (MSB)

Example 17 – Disable ABC Logic™ (I^2C)

The default T67xx I^2C slave address is ‘15’H and not shown.

Modbus Request (I^2C)

‘05’H	Function Code
‘03’H	Register Address (MSB)
‘EE’H	Register Address (LSB)
‘00’H	Register Value (MSB) Disable ABC Logic™
‘00’H	Register Value (LSB) LSB will always be zero

MEASURE ON DEMAND (MOD)

T6713 firmware V201 and greater supports the Measure on Demand (MOD) feature. MOD command is available on all T67xx models but can only be used with T67xx-R15 models as measure on demand will affect the accuracy of other sensor types. MOD incorporates a timer that makes it impossible to sample the data in MOD mode faster than every 15 seconds.

When the sensor is not sampling in this mode it will automatically go to sleep. To set T67xx-R15 to MOD, set holding register at ‘100B’H to 0. The command for this is 0x15 06 10 0B 00 00 + CRC. Once the sensor is in MOD mode, it will not return to normal mode until the same register ‘100B’H is set to 1. The command for this is 0x15 06 10 0B 00 01 + CRC.

After that the RAM entry needs to be written to flash memory. The command is 0x15 05 03 ED FF 00 + CRC. The memory location can be read back after updating flash memory. 0x15 05 03 ED FF 00 + CRC.

For the change to take place a power cycle or RESET is needed. RESET command is 0x15 05 03 E8 FF 00 + CRC.

MOD mode is now enabled. The sensor will not measure until it is requested. Request a measure with the command 0x 15 05 03 F3 00 00 + CRC.

The sensor requires 2.25 seconds to acquire the measurement and update the PPM register. Wait 2.25 seconds minimum after the measure request, and request the PPM, 0x 15 04 13 8B 00 01 + CRC.

A 15 second timer will prevent measurement being taken if any request is made within that time. If measurement is requested more frequently than this threshold the sensor will not acknowledge the request.

If the sensor is to be recalibrated, then the MOD mode needs to be disabled. Set MOD holding register ‘100B’H = 1, 0x15 06 10 0b 3f 80, followed by memory write 0x 15 05 03 ED FF 00 and with RESET or power cycle.

T67XX CO_2 SENSOR MODULE

MEASURE ON DEMAND MODE USING UART

Example 18 – Enable Measure on Demand (UART)

Modbus Request (UART) to set RAM value to MOD mode

‘15’H	Slave Address (default is ‘15’H’)
‘06’H	Function Code
‘10’H	Register Address (MSB)
‘0B’H	Register Address (LSB)
‘00’H	Register Value (MSB) set value to zero
‘00’H	Register Value (LSB) set value to zero
xx	CRC (LSB)
xx	CRC (MSB)

Modbus Response (UART)

‘15’H	Slave Address (default is ‘15’H’)
‘06’H	Function Code
‘10’H	Register Address (MSB)
‘0B’H	Register Address (LSB)
‘00’H	Register Value (MSB)
00’H	Register Value (LSB)
xx	CRC (LSB)
xx	CRC (MSB)

Example 19 – Flash Update (UART)

Modbus Request UART)

‘15’H	Slave Address (default is ‘15’H’)
‘05’H	Function Code
‘03’H	Register Address (MSB)
‘ED’H	Register Address (LSB)
‘FF’H	Register Value (MSB) MSB to update RAM to Flash
‘00’H	Register Value (LSB) LSB will always be zero
xx	CRC (LSB)
xx	CRC (MSB)

Modbus Response (UART)

‘15’H	Slave Address (default is ‘15’H’)
‘05’H	Function Code
‘03’H	Register Address (MSB)
‘ED’H	Register Address (LSB)
‘FF’H	Register Value (MSB)
‘00’H	Register Value (LSB)
xx	CRC (LSB)
xx	CRC (MSB)
xx	CRC (MSB)

Example 20 – RESET (UART)

Modbus Request (UART)

'15'H	Slave Address (default is '15'H')
'05'H	Function Code
'03'H	Register Address (MSB)
'E8'H	Register Address (LSB)
'FF'H	Register Value (MSB) MSB to Reboot sensor
'00'H	Register Value (LSB) LSB will always be zero
xx	CRC (LSB)
xx	CRC (MSB)

Modbus Response (UART)

'15'H	Slave Address (default is '15'H')
'05'H	Function Code
'03'H	Register Address (MSB)
'E8'H	Register Address (LSB)
'FF'H	Register Value (MSB)
'00'H	Register Value (LSB)
xx	CRC (LSB)
xx	CRC (MSB)

Example 21 – Sensor Takes Single Measurement (UART)

Modbus Request (UART)

'15'H	Slave Address (default is '15'H')
'05'H	Function Code
'03'H	Register Address (MSB)
'F3'H	Register Address (LSB)
'00'H	Register Value (MSB)
'00'H	Register Value (LSB)
xx	CRC (LSB)
xx	CRC (MSB)

Modbus Response (UART)

'15'H	Slave Address (default is '15'H')
'05'H	Function Code
'03'H	Register Address (MSB)
'F3'H	Register Address (LSB)
'00'H	Register Value (LSB)
'00'H	Register Value (MSB)
xx	CRC (LSB)
xx	CRC (MSB)

Example 22 – Read current gas PPM (UART)

Modbus Request (UART)

‘15’H	Slave Address (default is ‘15’H’)
‘04’H	Function Code
‘13’H	Register Address (MSB)
‘8B’H	Register Address (LSB)
‘00’H	Register Value (MSB)
‘01’H	Register Value (LSB)
xx	CRC (LSB)
xx	CRC (MSB)

Modbus Response (UART)

‘15’H	Slave Address (default is ‘15’H’)
‘04’H	Function Code
‘02’H	Byte Count
‘01’H	MSB of the 16-bit data
‘9F’H	LSB of the 16-bit data
xx	CRC (LSB)
xx	CRC (MSB)

MEASURE ON DEMAND MODE USING I^2C

The default T67xx I^2C slave address is ‘15’H and not shown.

Example 23 – Enable Measure on Demand (I^2C)

Modbus Request (I^2C)

‘06’H	Function Code
‘10’H	Register Address (MSB)
‘0B’H	Register Address (LSB)
‘00’H	Register Value (MSB) set value to zero
‘00’H	Register Value (LSB) set value to zero

Modbus Response (I^2C)

‘06’H	Function Code
‘10’H	Register Address (MSB)
‘0B’H	Register Address (LSB)
‘00’H	Register Value (MSB)
00’H	Register Value (LSB)

Example 24 – Flash Update (I^2C)

Modbus Request (I^2C)

'05'H	Function Code
'03'H	Register Address (MSB)
'ED'H	Register Address (LSB)
'FF'H	Register Value (MSB) MSB to update RAM to Flash
'00'H	Register Value (LSB) LSB will always be zero

Modbus Response (I^2C)

'05'H	Function Code
'03'H	Register Address (MSB)
'ED'H	Register Address (LSB)
'FF'H	Register Value (MSB)
'00'H	Register Value (LSB)

Example 25 – RESET (I^2C)

Modbus Request (I^2C)

'05'H	Function Code
'03'H	Register Address (MSB)
'E8'H	Register Address (LSB)
'FF'H	Register Value (MSB) MSB to Reboot sensor
'00'H	Register Value (LSB) LSB will always be zero

Modbus Response (I^2C)

'05'H	Function Code
'03'H	Register Address (MSB)
'E8'H	Register Address (LSB)
'FF'H	Register Value (MSB)
'00'H	Register Value (LSB)

Example 26 – Sensor takes single measurement (I^2C)

Modbus Request (I^2C)

'05'H	Function Code
'03'H	Register Address (MSB)
'F3'H	Register Address (LSB)
'00'H	Register Value (MSB)
'00'H	Register Value (LSB)

Modbus Response (I^2C)

'05'H	Function Code
'03'H	Register Address (MSB)
'F3'H	Register Address (LSB)
'00'H	Register Value (LSB)
'00'H	Register Value (MSB)

Example 27 – Read current gas PPM (I^2C)

Modbus Request (I^2C)

'04'H	Function Code
'13'H	Register Address (MSB)
'8B'H	Register Address (LSB)
'00'H	Register Value (MSB)
'01'H	Register Value (LSB)

Modbus Response (UART)

'04'H	Function Code
'02'H	Byte Count
'01'H	MSB of the 16-bit data
'9F'H	LSB of the 16-bit data

To calculate the gas ppm, do the following:

ppm = MSB * 256 + LSB

Table summary of Measure on Demand operation command (note - only to be used on '-R15', 15 second timing models)

Step	Byte Sequence (CRC excluded)	Reply (CRC excluded)	Purpose
Read Sensor Mode	0x15 03 10 0B 00 00	15 3f 80 (Normal Mode)	Read current sensor mode. If response is 15 00 00, follow Measure on Demand step.
Enable Measure on Demand	0x15 06 10 0B 00 00	15 06 10 0B 00 00	Sets register '100B'H to 0
Memory update required	0x15 05 03 ED FF 00	15 05 03 ED FF 00	Ram to Flash update
Reboot	0x15 05 03 E8 FF 00	15 05 03 E8 FF 00	Measure on Demand active
Measure on Demand	0x 15 05 03 F3 00 00	15 05 03 F3 00 00	Sensor takes a measurement
PPM Request	0x 15 04 13 8B 00 01	15 04 02 01 9C (Ex. 412 ppm)	Available 2.5 seconds after request

EXAMPLE CODE

Reading Value for CO_2

Reading the T67xx sensor output in an embedded design is fairly easy. This example assumes that the controlling micro-processor has already initialized a UART and the sensor has the default settings from the factory (e.g., the slave address is '15'H).

Because the slave address and CRC will never change the programmer can just define the entire Modbus request as constants in as array.

For example:

```
static const uint8_t Read_CO2_Cmd[8] = {
  0x15,          // assumed slave address
  0x04,          // read input register function code
  0x13,          // register address 5003 (MSB)
  0x8B,          // register address 5003 (LSB)
  0x00,          // number of registers (MSB)
  0x01,          // number of registers (LSB)
  0x46,          // CRC (LSB)
  0x70           // CRC (MSB)
};

static uint16_t Read_CO2_Cmd_Length =
  (uint16_t)(sizeof(Read_CO2_Cmd)/sizeof(uint8_t));
```

The following example sends the above character array out the serial port and then delays 50ms before looking for a response. The response should be 7 bytes long and in this example will fill a data structure (i.e., char array) named input_buf[] with the result.

```
input_buf[0] = 0x15   /* slave address */
input_buf[1] = 0x04   /* function code */
input_buf[2] = 0x02   /* byte count */
input_buf[3] = 0x??   /* CO2 reading, MSB */
input_buf[4] = 0x??   /* CO2 reading, LSB */
input_buf[5] = 0x??   /* CRC, LSB */
input_buf[6] = 0x??   /* CRC, MSB */
```

The CO_2 can then be calculated as

```
input_buf[3] * 256 + input_buf[4]
```

The code snippet follows.

```
uart1_write((uint8_t*)Read_CO2_Cmd, Read_CO2_Cmd_Length);
delay(50);
uart1_read((uint8_t*)input_buf, 8);
if ((input_buf[0]==0x15) && (input_buf[1]==0x04) && (input_buf[2]==0x02))
{
  raw_co2 = (uint16_t)((input_buf[3]<<8) | input_buf[4]);
}
```

Arduino I²C Examples

The I²C needs some delay after the request for a correct response. The following is a snippet of Arduino code that can be used to test the I²C interface.

Following definitions apply to both examples.

```
#define readDelay 5  //delay between I2C write & read requests in mS (10 recommended)
#define measureDelay 1000  //delay between measure and read PPM requests in mS (2250 min recommended)
#define ADDR_6700  0x15 // default I2C slave address is 0x15
int GetCO2PPM()  // example 1, standard I2C get CO2 value.
{
  byte data[6];
  // start I2C

 Wire.beginTransmission(ADDR_6700);
  Wire.write(0x04);
  Wire.write(0x13);
  Wire.write(0x8B);
  Wire.write(0x00);
  Wire.write(0x01);
  // end transmission
  Wire.endTransmission();
  // read report of current gas measurement in ppm after delay!
  delay(readDelay);

 Wire.requestFrom(ADDR_6700, 4);    // request 4 bytes from slave device
  data[0] = Wire.read();
  data[1] = Wire.read();
  data[2] = Wire.read();
  data[3] = Wire.read();
  return ((data[2] & 0x3F ) << 8) | data[3];
}
int GetCO2PPM()    //example 2 returns value for measure on demand variant, demand timing is assumed elsewhere, and there is no test for no response from sensor.
{
  byte data[5];
  Wire.beginTransmission(ADDR_6700);
  Wire.write(0x05); Wire.write(0x03); Wire.write(0xF3); Wire.write(0x00); Wire.write(0x00);
  // end transmission
  Wire.endTransmission();
  delay(readDelay);
  Wire.requestFrom(ADDR_6700, 5);    // request 5 bytes from slave device
  data[0] = Wire.read();
  data[1] = Wire.read();
  data[2] = Wire.read();
  data[3] = Wire.read();
  data[4] = Wire.read();
```

```
  if (data[1] == 0x03 && data[4] == 0x00 ) {  //test to ensure sensor responded to measure command
    delay(measureDelay);

  Wire.beginTransmission(ADDR_6700);
  Wire.write(0x04); Wire.write(0x13); Wire.write(0x8B); Wire.write(0x00); Wire.write(0x01);
  // end transmission
  Wire.endTransmission();
  // read report of current gas measurement in ppm
  delay(readDelay);   //delay in mS
  Wire.requestFrom(ADDR_6700, 4);    // request 4 bytes from slave device
  data[0] = Wire.read();
  data[1] = Wire.read();
  data[2] = Wire.read();
  data[3] = Wire.read();
    return ((data[2] & 0x3F ) << 8) | data[3];

  } else {
    return ((0xC3 & 0x3F ) << 8) | 0x50;   //or whatever value for failure you want
  }
}
```

Other code samples are available at: https://github.com/AmphenolAdvancedSensors/Telaire

Amphenol

Advanced Sensors

www.telaire.com

www.amphenol-sensors.com

AAS-916-142A - 02/2019