Communication Protocol

CUBE-v2

Communication Protocol

CUBE-v2

Project name	CUBE-PT_v1
Project leader	ChHu
Document code	PCP-CUBE-v2
Document path	\\Projects Documentation\CUBE-PT_v1 -> 04.02.02.03. Communication protocol
Document name	120529_daan_PCP_CUBE-v2_dv2.doc
Release	2012-05-29
Version	dv2
Originated by	PreSens RD - DaAn
Approved by	

PreSens Precision Sensing GmbH
Josef-Engert-Str. 11
93053 Regensburg
Germany

Phone +49 941 94272 100
Fax +49 941 94272 111
info@PreSens.de
www.PreSens.de

Geschäftsführer/CEO: Achim Stangelmayer
Trade Register Ingolstadt HRB 101505
VAT ID No. DE 812 210 724
WEEE Reg. DE 10886810

Raiffeisen-Volksbank Neuburg/Donau eG
Account 60682, Bank Code 721 697 56
SWIFT Code GENO DE F1 ND2
IBAN DE11721697560000060682

INDEX

TABLES

1. Introduction

1.1. Purpose

This document specifies the communication protocol for CUBE_v2 device.

1.2. Document history

Table 1. Revision history

Revision	Date	File Name	Description	Change by
dv1	22.05.12	120522_daan_PCP_CUBE-v2_dv1.doc	First release	DaAn
dv2	29.05.12	120529_daan_PCP_CUBE-v2_dv2.doc	Revision by ChSa and ChHu	DaAn

1.3. Abbreviations and Terminology

The following abbreviations and terminology are used throughout this document.

Table 2. Abbreviations

SA	Slave Address
STT	I2C Start
STP	I2C Stop
RO	Read Only
RW	Read or Write
STR	I2C Restart
ACKS	Acknowledge by Slave
ACKM	Acknowledge by Master
NACK	No Acknowledge

2. Data Registers

The CUBE_v2 have 6 user accessible registers. They are used to control the device and read out measurement values. The device has no internal memory so all registers are set to default after power on.

2.1. Addresses

Table 3. Register addresses and access type

Address [hex]	Name	Size [Bytes]	Access	Reset value
0x00	Control	1	R/W	0x15
0x01	Status	1	RO	0x00
0x10	Sampling rate	1	R/W	0x02
0x11	Sensor phase shift	4	RO	0x00
0x12	Sensor amplitude	4	RO	0x00
0x13	CPU temperature	2	RO	0x00

NOTE: Writing to RO register is ignored.

2.2. Control register

The control register allows for device function and performance control.
Following bits are defined:

Bit0: MODE	- mode 0 – trigger (read by trigger)
	- mode 1 - continuous with predefined sampling rate (default = 1)
Bit1: TRG	- trigger
Bit2: TEMP	- Temperature sensor 0 - off, 1 – on (default = 1)
Bit4: GAIN0	- LED gain Lo (default = 1)
Bit5: GAIN1	- LED gain Hi (default = 0)
Bit6:	- reserved
Bit7:	- reserved

2.3. Status register

The status register shows the current device status.
Following bits are defined:

Bit0: DRDY	- set on new data (phase shift, amplitude, temperature) ready
Bit1: SLEEP	- set when in trigger mode and no measurement
Bit2:	- reserved
Bit3:	- reserved
Bit4:	- reserved
Bit5: ERR0	- error bit 0 = set on amplitude too low
Bit6: ERR1	- error bit 1 = set on amplitude too high
Bit7: ERR2	- error bit 2 = reserved for future

2.4. Sensor phase shift register

The phase shift register contains sensor value that is proportional to measured oxygen concentration. The value (4 bytes) represent float variable according to IEEE754.

NOTE: For more information how to calculate oxygen concentration from phase shift please contact PreSens GmbH directly.

2.5. Sensor amplitude register

The amplitude is the parameter used for sensor and device fitness monitoring. The amplitude value (4 bytes) represent float variable according to IEEE754. The amplitude should be always between 1000 and 20000 units. The lower is the amplitude the higher phase shift noise occurs.

2.6. CPU temperature register

The temperature shows current device temperature. The temperature value (2 bytes) represent integer variable. It must be divided by 10 to get readings in degrees Celsius. Low byte is sent first.

Example.: 0xD7 0x00 = 0xD700 -> 0x00D7 = 215dec / 10 = 21.5°C

3. Serial communication

3.1. General specification

- The protocol is I2C Bus Standard Mode conform
- The CUBE-v2 device is a slave device
- Slave address is 0x90 (0x48 + R/W bit)
- Clock speed up to 100kHz is supported
- Floating point numbers are sent according to ISO 60559:2011 (IEEE 754) standards with binary32 format.

3.2. Data transfer

Data transfers are initiated by a Start condition (Start), followed by a 7-bit device address and a read/write bit. An Acknowledge (ACK) from the slave confirms the reception of each byte. Each access must be terminated by a Stop condition (Stop).
This device does not support sequential register read/write. Each register needs to be addressed using the Register Address.

3.3. I2C write

Following the START condition from the master, the device address (7 bits) and the R/W bit (which is a logic low) are clocked onto the bus by the master transmitter. This indicates to the addressed slave receiver that the address high byte will follow after it has generated an Acknowledge bit during the ninth clock cycle.
The next byte transmitted by the master is the byte of the register address and will be written into the address pointer of CUBE device. After receiving another Acknowledge signal from the slave the master device will transmit the data byte to be written into the selected register. The slave acknowledges again and the master generates a STOP condition.

Message Syntax

S	Byte 1	As	Byte 2	As	Byte 3	As	P
Start	DevAddress +Write		Register Address		Data Byte		Stop

Example: Write control register with value 0x01
STT 0x90 ACKS 0x00 ACKS 0x01 ACKS STP

O - from master
O - from slave

3.4. I2C read

To read a data from slave device a repeated start condition must be used. It allows first sending a command and then reading back an answer right away. Read operations allow the master to access any register in a random manner. To perform read operation, first the register address must be set. This is done by sending the register address to the slave as part of a write operation (R/W bit set to 0). After the address is sent, the master generates a START (repeated start) condition following the acknowledge. This terminates the write operation. Then the master issues the device address byte again but with the R/W bit set to a one. The slave will then issue an acknowledge and transmit the n data bytes. After the master acknowledges the first byte the slave sends next byte. After the last byte is received the master will not acknowledge the transfer but does generate a STOP condition which causes the CUBE to discontinue transmission.

Message Syntax

S	Byte 1	As	Byte 2	As	R	Byte 3	As	Byte 4	Am	Byte N	NAm	P
Start	DevAddress +Write		Register Address		Start	DevAddress +Read		Data from slave		Data from slave		Stop

Example: Read status register
STT 0x90ACKS0x01ACKSSTR 0x91 ACKS 0x01NACK STP

Example: Read phase shift register register
STT0x90ACKS0x11ACKSSTR 0x91 ACKS 0x01ACKM0x02 ACKM0x03ACKM0x04NACK STP

O - from master
O - from slave