I2C communication guide

Table of contents:

1. Revision information

Revision	Date	Author	Status
			New document
1.00	2005-04-19	JE	
1.01	2005-06-16	JE	
1.02	2005-08-24	JE	Added Appendix B
1.03	2005-11-07	AP	Adds
1.04	2005-11-08	VP	Note about Write EEPROM command added.
1.05	2006-12-18	JE	New template and new name; Added K30/K50
1.06	2007-05-23	VP+SP	Additions and Answers for questions. Corrected error in Read commands specification. Version for internal discussion and answers.
1.06a	2007-06-25	PZ	Version for distribution

2. Introduction

This document is a guide to how to communicate with CO_2 sensors in the K20/K21/K22/K30/K50 platform series from SenseAir AB. Communication is implemented using I^2C, this standard is described in "THE I^2C-BUS SPECIFICATION" (Philips Semiconductors, 1990).
I^2C is a trademark of Philips Corps

3. Communication protocol

The sensors in the K20/K21/K22/K30/K50 series uses the I^2C Bus for communication with other systems. It acts as a slave device on the I^2C bus, which means that it must be controlled by an I^2C master device. The physical layer of the I^2C Bus is implemented in a dedicated hardware block in the sensor processor. The I^2C hardware block is configured and controlled by sensor firmware.

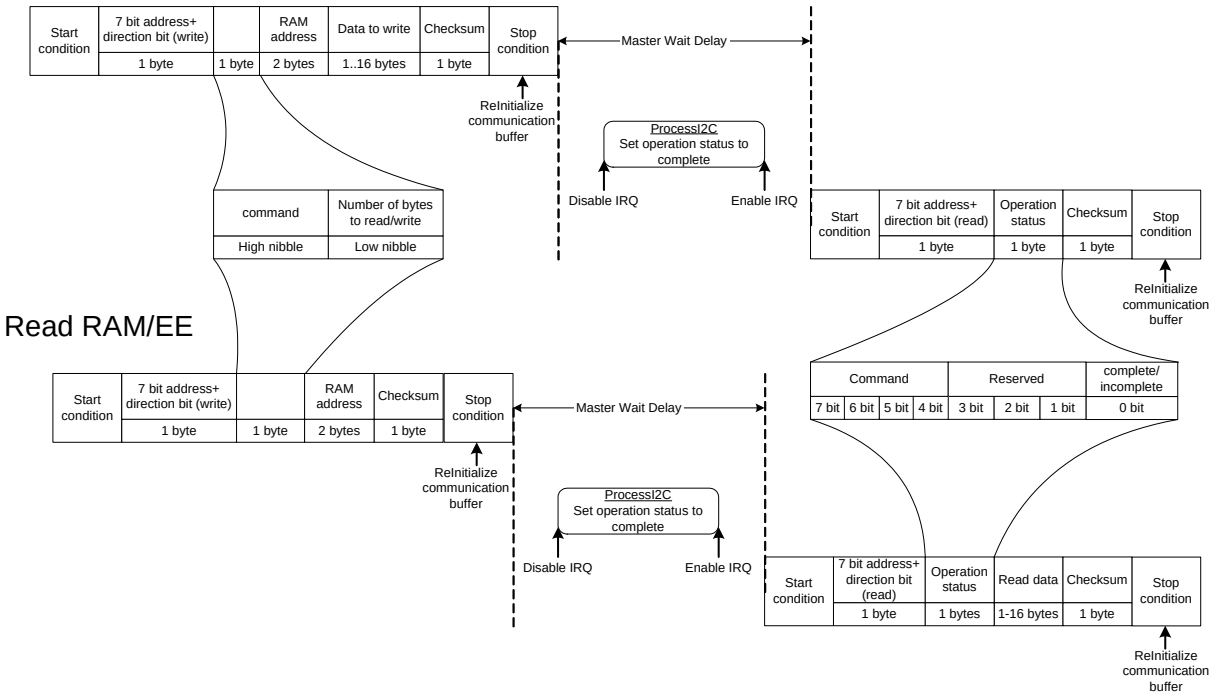
Table 1: I^2C parameters for K20 sensors

Parameter	Value	Comment
Master/slave mode	Slave	Sensor will never initiate communication, it only replies to master
Data rate	100kbits/s	
Addressing mode	7bit	
Address	0x68	Configurable in EEPROM

3.1. Command sequences

The command sequence during I2C communication session is presented on Picture 1.

Picture 1: Command sequence



There are four command sequences supported, "WriteRAM", "ReadRAM", "WriteEE" and "ReadEE". Access to other commands is allowed through use of Special Command Register (see table 3).
Note that Read/Write EE commands are supported only K21/K22/K30/K50 sensors for external EEPROM.

The sequence to perform a "Write RAM/EE" (Write data to sensor RAM/EE) is as follows:

1. Write command code(1), sensor RAM/EE address to write to and the data to be written.
2. Read a status frame from sensor, and check the complete/incomplete bit if the command was successful.

The sequence to perform a "Read RAM/EE" (Read data from sensor RAM/EE) is as follows:

1. Write command code(1) and sensor RAM/EE address to read data from.
2. Read one I^2C frame containing the read data from sensor RAM, and check the complete/incomplete bit if the command was successful.

(1) Command codes are listed in table 2

Step 2 must be repeated until a valid frame with "complete bit" set to one is received or timeout is occurred. Timeout must be defined by master. The complete bit has to be checked to verify that the command was successfully executed.

Table 2: Encoding of fields

Field	Size (bits)	Value	Interpretation
7 bit address + direction bit	8		I^2C address and R/W bit. Sensor Address is 0x68, R/W bit encoding Read = 1, Write = 0. (See "THE I^2C-BUS SPECIFICATION"). E.g. for write sensor RAM this field should be 0xD0 (0x68 shifted 1 step to left and R/W bit = 0)
Command	4	0x1 0x2 0x3 0x4	Write RAM Read RAM Write EE (1) Read EE (1)(2)
Number of bytes to read/write	4		The number of bytes to read/write. Value range from 1 to 15, 0 means 16 bytes. E.g. for write 1 byte to sensor RAM this byte should be 0x11 (Command = WriteRAM, NbrOfBytes = 1)
RAM Address	16		Address in sensor RAM. See "RAM Addresses" below.
Checksum	8		Arithmetic sum of the bytes sent (not including first byte with address and direction bit). See Appendix A for code example.

Notes:
1. The commands ReadEE and WriteEE are legal only for sensors with external EEPROM! (K21K22/K30/K50)
2. External EEPROM has page size 16 bytes. If data in command WriteEE crosses bound of the EEPROM page, the command is ignored and Bridge or Backplane return exception "Slave Failure".

3.2. RAM Addresses

The Read/Write RAM/EE gives access to the whole RAM area of the microprocessor and external EEPROM. The user must take care when writing to not overwrite any RAM/EE location that could compromise the execution of the sensor firmware such as calibration data, stack variables and other local variables.

Maps are product specific. But CO2 value keeps its location. It is located in RAM at address 0x08 (high byte) and 0x09 (low byte). Refer examples in Appendix B

3.3. Commands available through SCR

The Special Command Register (SCR), gives access to commands other than ReadRAM and WriteRAM, these commands are called by writing the appropriate command code to RAM address 0x60 (See memory map)
The user must take care when writing to not overwrite any EEPROM location that could compromise the execution of the sensor firmware such as configuration and calibration data. The EEPROM (virtual EEPROM) accessed by SCR register is a part of sensor Flash and has nothing to do with external EEPROM in K21/K22/K30/K50 sensors.

Table 3: SCR Commands

Command	Code	Function	Comment
ReadEEPROM	0x1	Copies first page of EEPROM contents to RAM so it can be read through a WriteRAM command	
WriteEEPROM	0x2	Copies RAM contents to first page of EEPROM	

3.4. Commands description

4.3.1. Write RAM

Writes up to 16 bytes to Sensor RAM

Request:

I^2C Start condition	7-bit ofI^2C Address	Read/ Write bit	Command high nibble	Number of bytes Low nibble	RAM address	Data to write	check sum	I^2C Stop condition
	Sensor addr.	0 (write)	0x1	0..0xF				
	1 byte		1 byte		2 bytes	1..16 bytes	1 byte	

"Write Complete" Response:

I^2C Start condition	7-bit ofI^2C Address	Read/ Write bit	Command high nibble	Status bit	checksum	I^2C Stop condition
	Sensor addr.	1 (read)	0x11			
	1 byte		1 byte		1 byte	

"Write Incomplete" Response:

I^2C Start condition	7-bit ofI^2C Address	Read/ Write bit	Command high nibble	Status bit	checksum	I^2C Stop condition
	Sensor addr.	1 (read)	0x10			
	1 byte		1 byte		1 byte	

4.3.2. Read RAM

Reads up to 16 bytes from Sensor RAM.

Request:

I^2C Start condition	7-bit ofI^2C Address	Read/ Write bit	Command high nibble	Number of bytes Low nibble	RAM address	checksum	I^2C Stop condition
	Sensor addr.	0 (write)	0x2	0..0xF			
	1 byte		1 byte		2 bytes	1 byte	

"Read Complete" Response:

I^2C Start condition	7-bit ofI^2C Address	Read/ Write bit	Command high nibble	Status bit	Read Data	checksum	I^2C Stop condition
	Sensor addr.	1 (read)	0x21				
	1 byte		1 byte		1..16 bytes	1 byte	

"Read Incomplete" Response:

I^2C Start condition	7-bit ofI^2C Address	Read/ Write bit	Command high nibble	Status bit	All other bytes	I^2C Stop condition
	Sensor addr.	1 (read)	0x20		0x20	
	1 byte		1 byte			

4.3.3. Write EEPROM

Writes up to 16 bytes to Sensor external EEPROM (available only in K21/K30/K50 versions)

Request:

I²C Start condition	7-bit ofI²C Address	Read/ Write bit	Command high nibble	Number of bytes Low nibble	RAM address	Data to write	check sum	I²C Stop condition
	Sensor addr.	0 (write)	0x3	0..0xF				
	1 byte		1 byte		2 bytes	1..16 bytes	1 byte	

"Write Complete" Response:

I²C Start condition	7-bit ofI²C Address	Read/ Write bit	Command high nibble	Status bit	checksum	I²C Stop condition
	Sensor addr.	1 (read)	0x31			
	1 byte		1 byte		1 byte	

"Write Incomplete" Response:

I²C Start condition	7-bit ofI²C Address	Read/ Write bit	Command high nibble	Status bit	checksum	I²C Stop condition
	Sensor addr.	1 (read)	0x30			
	1 byte		1 byte		1 byte	

4.3.4. Read EEPROM

Reads up to 16 bytes from Sensor external EEPROM (available only in K21/K30/K50 versions)

Request:

I²C Start condition	7-bit ofI²C Address	Read/ Write bit	Command high nibble	Number of bytes Low nibble	RAM address	checksum	I²C Stop condition
	Sensor addr.	0 (write)	0x4	0..0xF			
	1 byte		1 byte		2 bytes	1 byte	

"Read Complete" Response:

I²C Start condition	7-bit ofI²C Address	Read/ Write bit	Command high nibble	Status bit	Read Data	checksum	I²C Stop condition
	Sensor addr.	1 (read)	0x41				
	1 byte		1 byte		1..16 bytes	1 byte	

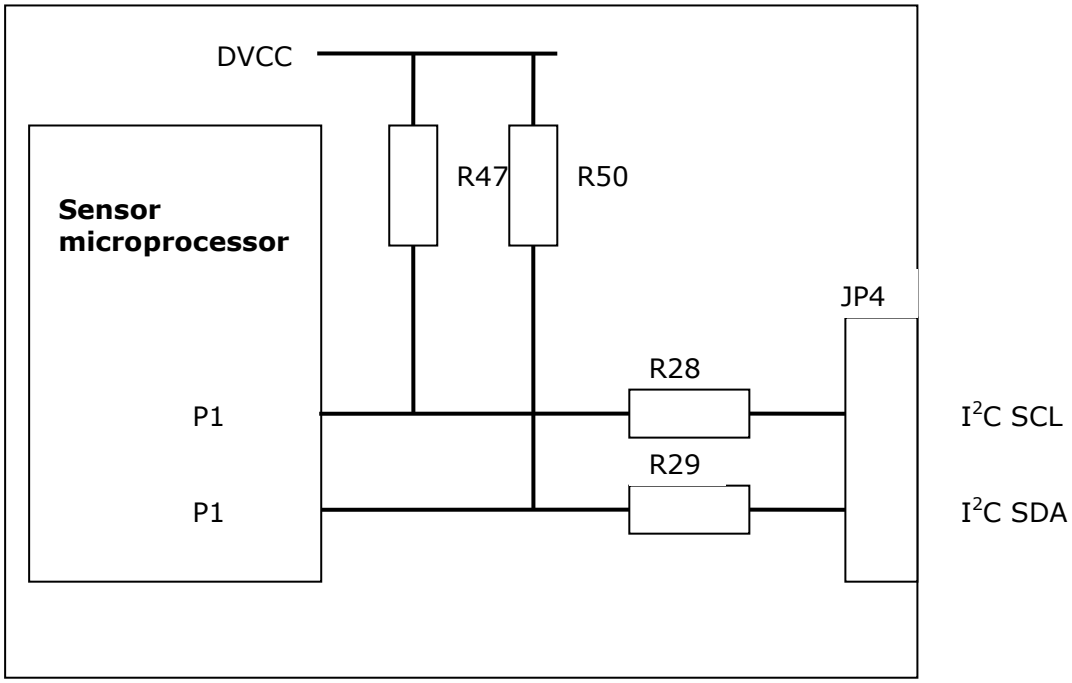
"Read Incomplete" Response:

I²C Start condition	7-bit ofI²C Address	Read/ Write bit	Command high nibble	Status bit	All other bytes	I²C Stop condition
	Sensor addr.	1 (read)	0x40		0x20	
	1 byte		1 byte			

4. Electrical connections

The electrical schematic for the sensors in the K20/K21/K22/K30/K50 series is shown in fig. 1 Both the SCL and SDA lines have pull-ups to DVCC. The pull-ups are the processor internal pull-up which is connected in parallel with an external resistor. There is also a series resistor for each of the lines.
See also Section 16 of "THE I^2C-BUS SPECIFICATION" for more information on electrical connections.

Figure 1: I^2C schematics



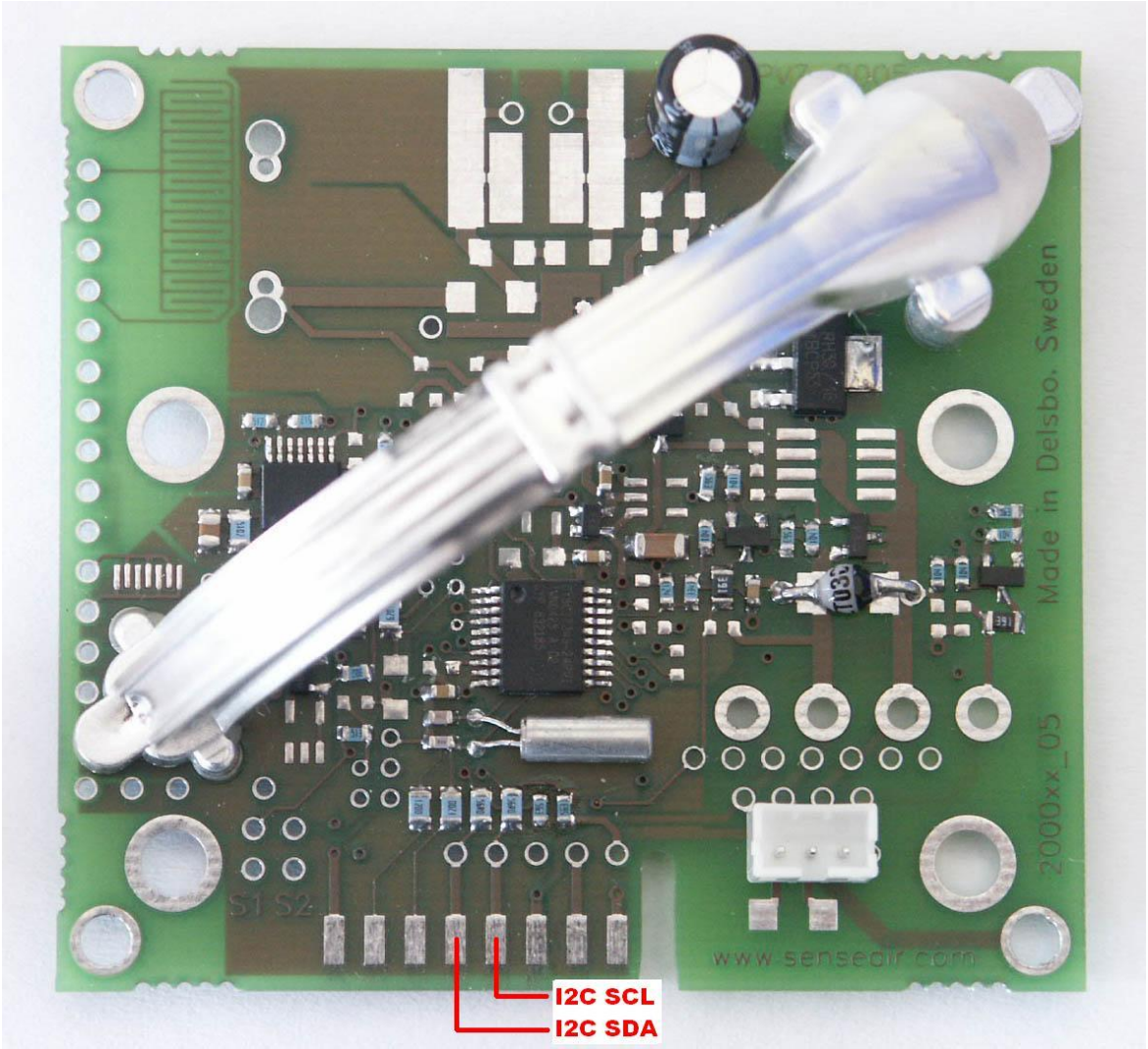
* JP4 is sensor edge connector (See also picture 2)

Table 4: Resistor values

Resistor name	**Value**	**Comment**
R47 (I^2C SCL Pull-up)	56kOhms	
R50 (I^2C SDA Pull-up)	56kOhms	
R28 (I^2C SCL Series resistor)	56Ohms	
R29 (I^2C SDA Series resistor)	56Ohms	
SCL processor internal pull-up	5.6kOhms (typ.)	See processor data sheet
SDA processor internal pull-up	5.6kOhms (typ.)	See processor data sheet

The I^2C Bus pins are available at the JP4 edge connector. Pin 7 and 8 is SCL and pin 9 and 10 is SDA. (See picture 2).

Picture 2: SCL and SDA connections



DVCC voltage is different for different models in the K20 series, as described in table 5. This needs to be considered when interfacing the sensors since it might be necessary to use level conversion circuitry in some cases. Descriptions of such circuitry can be found in Section 18 of "THE I^2C-BUS SPECIFICATION", and other examples are available from the internet (e.g. http://www.maxim-ic.com/appnotes.cfm/appnote_number/1159) Note also that prototypes may have different DVCC voltage than production version of sensor type.

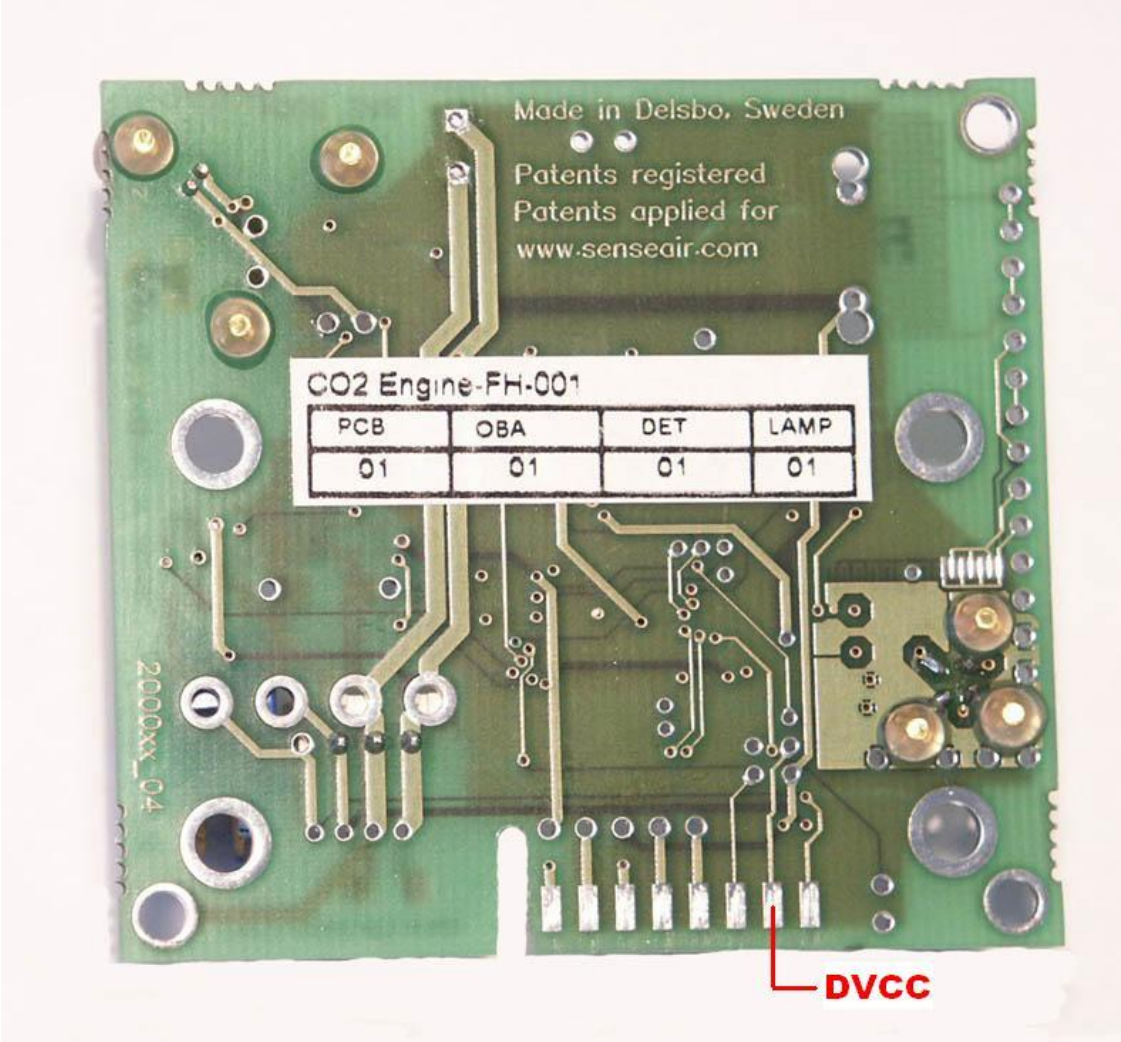
Table 5: DVCC levels for different sensor types:

Sensor type	**DVCC voltage**		**Comment**
FH	5V		
FHP	5V		
PO	3.3V		
POP	3.3V		
4B	3.3V		
4BP	3.3V		

The sensor DVCC voltage is available at pin 14 of the JP4 edge connector. (See picture 3).

This pin is connected directly to the output of the on-board voltage regulator and can deliver a small amount of current, so it can be used to drive external circuitry. (For maximum current rating figure contact SenseAir AB).

Picture 3: DVCC Connection



5. Limitations

Sensor dead time:
The sensor is equipped with only a small microcontroller that handles both measurements and communication with the external world and since the quality of measurements are of highest priority there are some limitation to the way communication is handled. While the sensor is actively measuring CO_2, the communication is switched off (i.e. I^2C interrupts are disabled) not to disturb the measurements. This will have the effect that there will be a short period during which the sensor will not respond to I^2C Bus activities. This period is short and the impact on communication is very limited. However, when designing an I^2C system where the sensor is a component this situation needs to be handled correctly. It may be necessary to implement checks and actions if a command sent to the sensor has failed to execute correctly. The sensor protocol has therefore a mandatory check of status after each command, with 1 bit (complete/not complete) that will tell if the command has been executed by the sensor.

6. Related topics

1. ModBus Specification: \UTVECK\Info Topics\Communication\modbus\March 2003 download\Obsolete Modicon Modbus ref guide 1996.pdf
2. ModBus Implementation \UTVECK\MEMOs\Company's communication protocol\Draft\ModBus Implementation Sensor rev1_07.doc
3. PC-Bridge-Meter Communication diagrams Utveck\PROJEKT\Platforms\P200\SystemCommunication\PC-Bridge-Meter Communication diagram rev1_06.vsd
4. ModBus Implementation Smart Bridge \Utveck\MEMOs\Company's communication protocol\Draft\Bridge to P200\ModBus Implementation SmartBridge rev1_01.doc

7. Appendix A, Code examples

Checksum calculation example:

```
typedef unsigned char BYTE;


BYTE CheckSum(BYTE * buf, BYTE count) {
      BYTE sum=0;
      while (count>0) {
          sum += *buf;
          buf++;
          count--;
      }
      return sum;
}
```

Let buf point to the first byte after the "7-bit address+Direction bit" field. Byte counter count should be set to number of bytes sent excluding checksum byte.

8. Appendix B, I2C bus transaction examples

Example: Reading of CO2 value from sensor

To read the current CO2 concentration from the sensor we need to read memory locations 0x08 (hi byte) and 0x09 (low byte).

To do this we need to send a sequence of two I2C frames: first we send an I2C write frame containing the sensor address, command number and how many bytes to read, RAM address to read from, and a checksum. Then we send an I2C read frame to read the status, data and checksum. See chapter 2 for details.

In our case we want to read 2 bytes starting from address 0x08. This will give us data from address 0x08 and 0x09, which contains current CO2 reading. The sensor address is 0x68 (default factory setting, configurable in EEPROM).

So, the first frame should look like:

Start | 0xD0 | 0x22 | 0x00 | 0x08 | 0x2A | Stop

a. 0xD0 is Sensor address and read/write bit. 0x68 shifted one bit to left and R/W bit is 0 (Write).
b. 0x22 is command number 2 (ReadRAM), and 2 bytes to read
c. Checksum 0x2A is calculated as sum of byte 2, 3 and 4.

The next frame will read the actual data:

Start | 0xD1 | <4 bytes read from sensor> | Stop

d. The 1:st byte from the sensor will contain operation status, where bit 0 tells us if the read command was successfully executed.
e. The 2:nd and 3:rd byte will contain CO2 value hi byte and CO2 value low byte.
f. The 4:th byte contains checksum

NOTEs:

1. It is worth to give 20 msec "Master Wait Delay", refer Picture 1.
2. Please take care about 2 ways the sensor can indicate "busy" or "incomplete" status.
   a. According I2C specification slave (sensor) shall acknowledge every byte it receives by sinking SDA line on the 9th SCL clock pulse. If sensor doesn't acknowledge byte with address and direction bit then it means "busy".
   b. Another case is when sensor acknowledges every byte but complete bit is 0. Master doesn't need to read the rest of the message and can just send stop condition.