Modbus on CO2 Engine K30

Engineering specification

Table of contents:

1. Revision information

Rev.	Date:	Author	Status:
1.00	June 4, 2006	PZ, JE	Specification based on rev 1.072 of Modbus implementation for P071 firmware based meters.
1.01	Aug 24, 2006	JE	Clarification of application examples
1.02	Aug 24, 2006	JE	New layout
1.03	Sept 1,2006	JE	Correction to background calibration application example
1.04	Sept 14, 2006	PZ, JE	Add Read Device identification specification. Add ABC period as a Hold register to the xls file.
1.05	Sept. 20, 2006	JE	Document name change, remove draft status. Changed error responses for Read Device ID. Added Read Device ID application example. Minor corrections.
1.06	Sept. 25, 2006	JE	Added application example, read/write ABC_PERIOD. Replaced "meter" with "sensor"
1.07	Oct 12, 2006	JE	Minor correction; Changed Maximum packet length from 27 to 28 bytes

2. General

Modbus is a simple, open protocol for both PLC and sensors. Details on Modbus can be found on www.modbus.org.

Present specification is based on specification of Modbus implementation on aSense and eSense families of sensors and aims to support backwards compatibility with them. There are differences between the Modbus specification [1] and the default implementation in the CO2 Engine K30 sensor. The differences are listed in this document.

2.1. General overview of protocol and implementation in our application

Master – slave:
Only master can initiate transaction. The CO2 Engine K30 sensor is a slave and will never initiate communication. The host system initiates transactions to read CO2 value from the corresponding register. The host system shall also check status of the CO2 Engine K30 sensor periodically (say every 2 sec) in order to determine if it is running without faults detected.

Packet identification:
Any message (packet) starts with a silent interval of **3.5** characters. Another silent interval of **3.5** characters marks message end. Silence interval between characters in the message needs to be kept less than 1.5 characters.
Both intervals are from the end of Stop-bit of previous byte to the beginning of the Start-bit of the next byte.

Packet length:
According to the Modbus specification [1], the packet length shall be maximum 255 bytes including address and CRC. We cannot support so large packets. Maximum length of packet (serial line PDU including address byte and 2 bytes CRC) supported by CO2 Engine K30 is 28 bytes. Packets of larger size are rejected without any answer from sensor even if the packet was addressed to the sensor. The number is selected in order to allow reading of Device ID strings of up to 15 bytes in length.

ModBus data model:
There are 4 primary data tables (addressable registers), which may overlay:

- Discrete Input (read only bit).
- Coil (read / write bit).
- Input register (read only 16 bit word, interpretation is up to application).
- Holding register (read / write 16 bit word).

Note: The CO2 Engine K30 does not support bitwise access of registers.

Exception responses:
Slave will send answer to the master only in the case of valid message structure. Nevertheless, it can send exception response because of detection of:

- Invalid function code.
- Invalid data address (requested register doesn’t exist in given device).
- Invalid data.
- Error in execution of requested function.

Modbus diagnostic counters:
T.B.D.

3. Byte transmission.

RTU transmission mode is the only mode supported by CO2 Engine K30 sensor.

3.1. Byte format:

The format for each byte in RTU mode differs between CO2 Engine K30 sensor default configuration and the description on page 12 of MODBUS over serial line specification [2].

	MODBUS over serial line specification [2]	CO2 Engine K30 default configuration
Coding system	8-bit binary	8-bit binary
Bits per byte:	1 start bit	1 start bit
	8 data bits, least significant bit first	8 data bits, least significant bit first
	1 bit for even parity	NO parity
	1 stop bit	1 stop bit

The reason for the difference is compatibility with test and production systems. Standard byte format can be provided on request.

3.2. Baud rate:

Required default baud rate according to MODBUS over serial line specification [2], page 20, is 19200 bps

The CO2 Engine K30 has 9600 bps as default.

Availability of 19200 bps and higher rate configurations will be specified later.

3.3. Physical layer:

The CO2 Engine K30 sensors provide CMOS logical levels RxD and TxD lines for serial transmission. It's up to system integrator to use them for direct communication with master processor or for connection to RS232 or RS485 drivers. In the latter case R/T control line may be added on request.

Communication lines are fed directly to micro controller with serial 56 Ohm protection resistors. Power supply to micro controller is 3.3V and it's a reason why voltages on communication lines are not allowed to exceed 3.5V (minimum voltage of regulator plus internal protection diode voltage drop)

RxD line is configured as digital input.
Input high level is 2.1V min
Input low level is 0.8V max

TxD line is configured as digital output.
Output high level is 2.3V (assuming 3.3V supply) min.
Output low level is 0.75V max

RxD input is pulled up to DVCC = 3.3V by 56 kOhm
TxD output is pulled up to DVCC = 3.3V by 56 kOhm

4. Modbus registers on sensor.

The idea is to define mapping of Modbus registers on memory, both RAM and EEPROM of the sensor. Mapping shall be configurable in memory (program FLASH) and shall be interpreted at command reception.

Presently, the following restrictive decisions are made:
1. Read only and read / write registers are not allowed to overlay.
2. Bit addressable items (i.e. Coils and Discrete inputs) will not be implemented.
3. Only write single register functional codes are implemented. Multiple write functional codes are not planned for implementation.
4. The total number of registers should be limited. Present decision is to limit number of input registers to 32 and number of holding registers to 32.
Note: the limited buffer space of the CO2 Engine K30 puts a limit on how many registers that can be read in one command, currently 8 registers.
5. Larger amount of data should be transferred as file. It is not implemented at the current stage of development.

Maps of registers are summarized in
K30 ModBus map rev1_01.xls [3]

5. Serial line frame and addressing.

5.1. Serial line frame

Modbus over serial line specification [2] distinguishes Modbus Protocol PDU and Modbus serial line PDU in the following way (RTU mode only is under consideration):

Modbus serial line PDU

Address field (1 byte)	Function Code	Data	CRC (Hi and Low)
	Modbus PDU		

5.2. Addressing rules

Addressing rules are summarised in the table:

Address	Modbus over serial line V1.0	CO2 Engine K30
0	Broadcast address	No broadcast commands currently implemented
From 1 to 247	Slave individual address	Slave individual address
From 248 to 253	Reserved	Nothing$^{1)}$
254	Reserved	"Any sensor"
255	Reserved	Nothing$^{1)}$

NOTES:

1. "Nothing" means that sensor doesn't recognise Modbus serial line PDUs with this address as addressed to the sensor. Sensor does not respond.
2. "Any sensor" means that any sensor with any slave individual address will recognise serial line PDUs with address 254 as addressed to them. They will respond. So that this address is for production / test purposes only. It must not be used in the installed network.
   This is a violation against the Modbus specification [1].

5.3. Broadcast address

Modbus specification [1] requires execution of all write commands in the broadcast address mode.

Current status for CO2 Engine K30 sensor:
Only one broadcast command, reset sensor, is planned but not implemented yet.

6. Bus timing.

Parameter	Min	Typ	Max	Units
Response time-out			180	msec
Turnaround delay			???	msec

“Response time-out” is defined to prevent master (host system) from staying in “Waiting for reply” state indefinitely. Refer to page 9 of MODBUS over serial line specification [2].

For slave device “Response time-out” represents maximum time allowed to take by “processing of required action”, “formatting normal reply” and “normal reply sent” alternatively by “formatting error reply” and “error reply sent”, refer to the slave state diagram on page 10 of the document mentioned above.

“Turnaround delay” is defined in MODBUS over serial line specification [2] as delay respected by Master after broadcast command in order to allow any slave to process the current request before sending a new one.

7. Function codes descriptions (PUBLIC).

Description of exception responses.

If the PDU of the received command has wrong format:

No Response PDU, (sensor doesn’t respond)

If Function Code isn’t equal to any implemented function code:

Exception Response PDU,

Function code	1 byte	Function Code + 0x80
Exception code = ***Illegal Function***	1 byte	0x01

If one or more of addressed Registers is not assigned (register is reserved or Quantity of registers is larger than maximum number of supported registers):

Exception Response PDU,

Function code	1 byte	Function Code + 0x80
Exception code = ***Illegal Data Address***	1 byte	0x02

7.1. 01 (0x01) Read Coils (one bit read / write registers).

Not implemented at the moment. Not to be implemented.

7.2. 02 (0x02) Read Discrete Inputs (one bit read only registers).

Not implemented at the moment. Not to be implemented.

7.3. 03 (0x03) Read Holding Registers (16 bits read / write registers).

Refer to Modbus specification [1].

Quantity of Registers is limited to 8.

Address of ModBus Holding Registers for 1-command reading is limited in range 0x0000..0x001F.

Request PDU

Function code	1 byte	0x03
Starting Address Hi	1 byte	Address Hi
Starting Address Lo	1 byte	Address Lo
Quantity of Registers Hi	1 byte	Quantity Hi
Quantity of Registers Lo	1 byte	Quantity Lo

Response PDU

Function code	1 byte	0x03
Byte Count	1 byte	2 x N *
Register Value	N x 2 bytes	

* N = Quantity of Registers

If Address>0x001F or (Address + Quantity)>0x0020:
Exception Response PDU,

Function code	1 byte	0x83
Exception code = ***Illegal Data Address***	1 byte	0x02

If Quantity=0 or Quantity>8:
Exception Response PDU,

Function code	1 byte	0x83
Exception code = ***Illegal Data Value***	1 byte	0x03

7.4. 04 (0x04) Read Input Registers (16 bits read only registers).

Refer to Modbus specification [1].

Quantity of Registers is limited to 8.

Address of ModBus Input Registers for 1-command reading is limited in range 0x0000..0x001F.

Request PDU

Function code	1 byte	0x04
Starting Address Hi	1 byte	Address Hi
Starting Address Lo	1 byte	Address Lo
Quantity of Registers Hi	1 byte	Quantity Hi
Quantity of Registers Lo	1 byte	Quantity Lo

Response PDU

Function code	1 byte	0x04
Byte Count	1 byte	2 x N *
Register Value	N x 2 bytes	

* N = Quantity of Registers

If Address>0x001F or (Address + Quantity)>0x0020:
Exception Response PDU,

Function code	1 byte	0x84
Exception code = ***Illegal Data Address***	1 byte	0x02

If Quantity=0 or Quantity>8:
Exception Response PDU,

Function code	1 byte	0x84
Exception code = ***Illegal Data Value***	1 byte	0x03

7.5. 05 (0x05) Write Single Coil (one bit read / write register).

Not implemented at the moment. Not to be implemented.

7.6. 06 (0x06) Write Single Register (16 bits read / write register).

Refer to Modbus specification [1].

Address of ModBus Holding Registers for 1-command reading/writing is limited in range 0x0000..0x001F.

Request PDU

Function code	1 byte	0x06
Starting Address Hi	1 byte	Address Hi
Starting Address Lo	1 byte	Address Lo
Register Value Hi	1 byte	Value Hi
Register Value Lo	1 byte	Value Lo

Response PDU (is an echo of the Request)

Function code	1 byte	0x06
Starting Address Hi	1 byte	Address Hi
Starting Address Lo	1 byte	Address Lo
Register Value Hi	1 byte	Value Hi
Register Value Lo	1 byte	Value Lo

If Address>0x001F:
Exception Response PDU,

Function code	1 byte	0x86
Exception code = ***Illegal Data Address***	1 byte	0x02

7.7. 15 (0x0F) Write Multiple Coils (one bit read / write registers).

Not implemented, TBD

7.8. 16 (0x10) Write Multiple Registers (16 bits read / write register).

Not implemented, TBD

7.9. 20 (0x14) Read File record.

Not implemented, TBD

7.10. 21 (0x15) Write File record.

Not implemented, TBD

7.11. 22 (0x16) Mask Write Register (16 bits read / write register).

Not implemented, TBD

7.12. 23 (0x17) Read / Write Multiple Registers (16 bits read / write register).

Not implemented, TBD

7.13. 43 / 14 (0x2B / 0x0E) Read Device Identification.

Refer to Modbus specification [1].

The CO2 Engine K30 sensor supports only Read Device ID code 4, individual access.

Objects 0x00..0x02 (basic identification) and 0x80..0x83 (extended identification) are available (see table)

Object ID	Object Name / Description	Type	Modbus status	Category	Implement. status
0x00	Vendor Name	ASCII string*	Mandatory	Basic	Implemented
0x01	ProductCode	ASCII string*	Mandatory	Basic	Implemented
0x02	MajorMinorRevision	ASCII string*	Mandatory	Basic	Implemented
0x03	VendorUrl	ASCII string	Optional	Regular	Not Implemented
0x04	ProductName	ASCII string	Optional	Regular	Not Implemented
0x05	ModelName	ASCII string	Optional	Regular	Not Implemented
0x06	UserApplicationName	ASCII string	Optional	Regular	Not Implemented
0x07.. 0x7F	Reserved				
0x80	Memory map version	1 byte unsigned	Optional	Extended	Implemented
0x81	Firmware revision, consists of: Firmware type, Revision Main, Revision Sub	3 bytes unsigned	Optional	Extended	Implemented
0x82	Sensor serial number (sensor ID)	4 bytes unsigned	Optional	Extended	Implemented
0x83	Sensor type	3 bytes unsigned	Optional	Extended	Implemented

*The ASCII strings are defined as:

Vendor Name	= “SenseAir AB”	(length 11 bytes)
Product Code	= “CO2 Engine K30”	(length 14 bytes)
MajorMinorRevision	= “V1.00”	(length 5 bytes)

Example: Read objects of category "Basic".

Request PDU, Object ID 0x00 to 0x02

Function code	1 byte	0x2B
MEI Type	1 byte	0x0E
Read Device ID code	1 byte	0x04 (individual access only)
Object ID	1 byte	0x00..0x02

Response PDU, Object ID 0x00 to 0x02

Function code	1 byte	0x2B
MEI Type	1 byte	0x0E
Read Device ID code	1 byte	0x04, same as in request
Conformity level	1 byte	0x81, basic identification for individual or stream access
More Follows	1 byte	0x00
Next Object ID	1 byte	0x00
Number of objects	1 byte	0x01
Object ID	1 byte	0x00..0x02
Object length	1 byte	0x0B or 0x0E or 0x05 (see definition of ASCII strings)
Object value	n byte	Object Data

Example: Read objects of category "Extended".

Request PDU, Object ID 0x80 to 0x83

Function code	1 byte	0x2B
MEI Type	1 byte	0x0E
Read Device ID code	1 byte	0x04 (individual access only)
Object ID	1 byte	0x80..0x83

Response PDU, Object ID 0x80 to 0x83

Function code	1 byte	0x2B
MEI Type	1 byte	0x0E
Read Device ID code	1 byte	0x04, same as in request
Conformity level	1 byte	0x83 : extended identification for individual or stream access
More Follows	1 byte	0x00
Next Object ID	1 byte	0x00
Number of objects	1 byte	0x01
Object ID	1 byte	0x80..0x83
Object length	1 byte	0x01 or 0x03 or 0x04
Object value	1 or 3 or 4 byte	Object Data

If wrong MEI Type:
Exception Response PDU,

Function code	1 byte	0xAB
Exception code = ***Illegal Function Code***	1 byte	0x01

If Object ID is not in range 0x00..0x03 or 0x80..0x83:
Exception Response PDU,

Function code	1 byte	0xAB
Exception code = ***Illegal Data Address***	1 byte	0x02

If wrong Device ID:
Exception Response PDU,

Function code	1 byte	0xAB
Exception code = ***Illegal Data Value***	1 byte	0x03

Note: The exception responses for function code 43 is implemented according to the RFC "RFC Non extended Exception code format of   43 Encapsulated Transport .doc" which is in status "Recommended for approval" at time of writing. This is in contrast with the ModBus specification [1] where the exception responses for function code 43 also have a MEI type field.

8. References

- [1] MODBUS Application Protocol Specification V1.1a
- [2] MODBUS over serial line specification and implementation guide V1.01
- [3] K30 ModBus map rev1_01.xls

9.Appendix A: Application examples

Prerequisites for the application examples:

1. A single slave (sensor) is assumed (address "any sensor" is used).
2. Values in <..> are hexadecimal.

CO2 read sequence:

CO2 Engine K30 sensor is addressed as "Any address" (0xFE).
We read CO2 value from IR4 using "Read input registers" (function code 04).
Hence, Starting address will be 0x0003 (register number-1) and Quantity of registers 0x0001. CRC calculated to 0xC5D5 is sent with low byte first.

Sensor replies with CO2 reading 400ppm (400 ppm = 0x190 hexadecimal).

Master Transmit:
<FE> <04> <00> <03> <00> <01> <D5> <C5>

Slave Reply:
<FE> <04> <02> <01> <90> <AC> <D8>

Sensor status read sequence:

CO2 Engine K30 sensor is addressed as "Any address" (0xFE).
We read status from IR1 using "Read input registers" (function code 04). Hence, Starting address will be 0x0000 (register number-1) and Quantity of registers 0x0001. CRC calculated to 0xC525 is sent with low byte first.

Sensor replies with status 0.

Master Transmit:
<FE> <04> <00> <00> <00> <01> <25> <C5>

Slave Reply:
<FE> <04> <02> <00> <00> <AD> <24>

Sensor status and CO2 read sequence:

CO2 Engine K30 sensor is addressed as “Any address” (0xFE).
Here we read both status and CO2 in one command by reading IR 1 to 4 using “Read input registers” (function code 04). Hence, Starting address will be 0x0000 (register number-1) and Quantity of registers 0x0004. CRC calculated to 0xC6E5 is sent with low byte first.

Sensor replies with status=0 and CO2 value 400ppm (0x190 hexadecimal).

Master Transmit:
```
<FE> <04> <00> <00> <00> <04> <E5> <C6>
```

Slave Reply:
```
<FE> <04> <08> <00> <00> <00> <00> <00> <00> <01> <90> <16> <E6>
               |  Status  |                     |  CO2  |
```

Background calibration sequence:

CO2 Engine K30 sensor is addressed as “Any address” (0xFE).

1. Clear acknowledgement register by writing 0 to HR1. Starting address is 0x0000 and Register value 0x0000. CRC calculated as 0xC59D is sent with low byte first.

Master Transmit:
<FE> <06> <00> <00> <00> <00> <9D> <C5>

Slave Reply:
<FE> <06> <00> <00> <00> <00> <9D> <C5>

2. Write command to start background calibration. Parameter for background calibration is 6 and for nitrogen calibration is 7. We write command 0x7C with parameter 0x06 to HR2. Starting address is 0x0001 and Register value 0x7C06. CRC calculated as 0xC76C is sent with low byte first.

Master Transmit:
<FE> <06> <00> <01> <7C> <06> <6C> <C7>

Slave Reply:
<FE> <06> <00> <01> <7C> <06> <6C> <C7>

3. Wait at least 2 seconds for standard sensor with 2 sec lamp cycle.

4. Read acknowledgement register. We use function 3 “Read Holding register” to read HR1. Starting address is 0x0000 and Quantity of registers is 0x0001. CRC calculated as 0x0590 is sent with low byte first.

Master Transmit:
<FE> <03> <00> <00> <00> <01> <90> <05>

Slave Reply:
<FE> <03> <02> <00> <20> <AD> <88>

Check that bit 5 (CI6) is 1. It is an acknowledgement of that the sensor has performed the calibration operation. The sensor may skip calibration; an example of a reason for this could be unstable signal due to changing CO2 concentration at the moment of the calibration request.

Read Device ID, Vendor Name:

CO2 Engine K30 sensor is addressed as "Any address" (0xFE).
We use the Read Device ID to read Vendor Name (object 0, basic access). This object is an ASCII string containing "SenseAir AB".

Function code is 0x2B, MEI Type 0x0E. Read Device ID code must be 0x04 (since the CO2Engine K30 only supports individual access.) Object ID is 0x00. CRC calculated to 0x3367 is sent with low byte first.

Sensor replies with a packet containing the 11-byte string.

Master Transmit:
```
<FE> <2B> <0E> <04> <00> <67> <33>
```

Slave Reply:
```
<FE> <2B> <0E> <04> <81> <00> <00> <01> <00> <0B> <53> <65> <6E>
<73> <65> <41> <69> <72> <20> <41> <42> <BE> <18>
```

In the response we can see:
Address = 0xFE
Function code = 0x2B
MEI Type = 0x0E
Read Device ID code = 0x04
Conformity level = 0x81
More Follows = 0x00
Next Object ID = 0x00
Number of objects = 0x01
Object ID = 0x00
Object Length = 0x0B (11 bytes)
Object Value = 0x53 … 0x42 (11 bytes with ASCII codes for "SenseAir AB")
CRC = 0x18BE sent with low byte first

Read ABC parameter, ABC_PERIOD:

One of the ABC parameters, ABC_PERIOD, is available for modification as it is mapped as a holding register. This example shows how to read ABC_PERIOD by accessing HR32.

CO2 Engine K30 sensor is addressed as "Any address" (0xFE).
Read current setting of ABC_PERIOD by reading HR32. We use function code 03 "Read Holding registers". Starting address is 0x001f and Quantity of Registers 0x0001. CRC calculated as 0xC3A1 is sent with low byte first.

Master Transmit:
<FE> <03> <00> <1F> <00> <01> <A1> <C3>

Slave Reply:
<FE> <03> <02> <00> <B4> <AC> <27>

In the slave reply we can see:

Address = 0xFE	
Function code = 0x03	
Byte count = 0x02	- We read 2 bytes (1 register of 16 bits)
Register value = 0x00B4	- 0xB4 hexadecimal = 180 decimal; 180 hours / 24 equals 7,5 days.
CRC = 0x27AC	- CRC sent with low byte first

Disable ABC function

We can disable the ABC function by setting ABC_PERIOD to 0.

CO2 Engine K30 sensor is addressed as "Any address" (0xFE).

We use function code 06 "Write Single Register" to write to HR32. Register address is 0x001f, register value 0x0000. CRC calculated as 0x03AC is sent with low byte first.

Master transmit:
<FE> <06> <00> <1F> <00> <00> <AC> <03>

Slave reply:
<FE> <06> <00> <1F> <00> <00> <AC> <03>

We can see the reply which is an echo of the transmitted sequence.

Enable ABC function

We can enable the ABC function by setting ABC_PERIOD to some value other than 0. In this example we set it to 7,5 days.

CO2 Engine K30 sensor is addressed as “Any address” (0xFE).

We use function code 06 “Write Single Register” to write to HR32. Register address is 0x001f, register value 0x00B4 (7,5 days * 24 hours = 180; 180 in hexadecimal format is 0xB4). CRC calculated as 0x74AC is sent with low byte first.

Master transmit:
<FE> <06> <00 <1F> <00> <B4> <AC> <74>

Slave reply:
<FE> <06> <00> <1F> <00> <B4> <AC> <74>

We can see the reply which is an echo of the transmitted sequence.