ANGELANTONI Industrie S.P.A.

Serial Communications Protocol Specifications – DISCOVERY

Vers. 2.04 rev. H–June 2011

REVISION	REVISIONED CHAPTERS	DATE	AUTOR/S	Q C. APPROVAL.(date and signature)
A	Emissione	22/09/98	Thecnical Office/pm	11/02/1998
B		08/10/99	Software Dept.	08/10/1999
C		26/10/2007	Software Dept.	26/10/2007
D	AnyVib	05/08/2008	Software Dept.	05/08/2008
E	Wazzle	08/01/2009	Software Dept.	08/01/2009
F	Sunrise	28/05/2009	Software Dept.	28/05/2009
G	Discovery	05/10/2009	Software Dept.	05/10/2009
H	Discovery	10/06/2011	Software Dept.	10/06/2011

INDEX

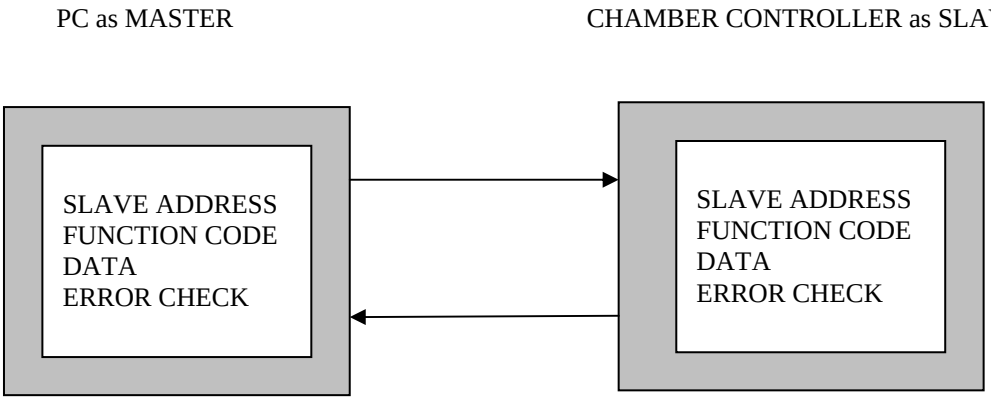
Figures		
Fig. 1	Master and Slave	Pag. 3
Fig. 2:	Communication overview	Pag. 4
Fig. 3:	Noise on the line	Pag. 4
Fig. 4:	Decoding a data packet	Pag. 5
Fig. 5:	Frame identification	Pag. 7
Fig. 6	CRC calculation	Pag. 12
Fig. 7:	System logic layout	Pag. 18
Fig. 8:	Serial cable	Pag. 30

INTRODUCTION

The present document contains information on the Modbus protocol for performing data exchange between a PC and the SAIA chamber controller. You can find here a short description of the query and answer sintax for the Modbus function 3 (read function) and function 16 (write function).
Moreover, a short example of code has been included as a guideline for the development of a complete program.

A communication protocol defines media, data formats and rules which are common to two or more systems wich must exchange informations by means of a phisical communication line.
Moreover is described here how the devices (master and salve) connected to the line interact between them to start and close the communication, and the way to identify error conditons during the data transmission.
Pratically the protocol allows a data exchange between Master device and a Slave device according to the following situation:

Fig. 1 Master and Slave



In this architecture, PC always acts as Master while the chamber controller is always the Slave device.
The Master device is the only one that can start the communication.
The communication consists essentially in a **query** made by the Master device (normally a PC) that is directly followed by an **answer** coming from the Slave device (the SAIA chamber controller).

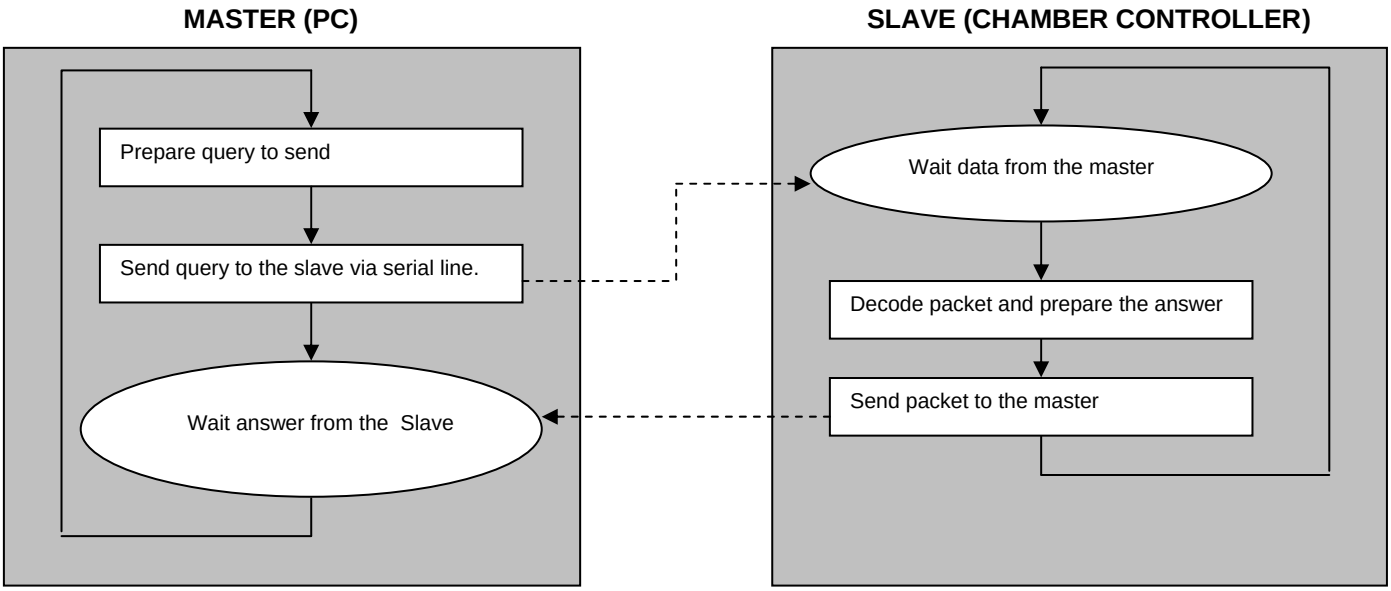
The communication task, either is a query or an answer, consists in preparing a data packet wich includes the identification code of the requested function, the lenght of the query/answer, the data itself and error check information. This data packet is then transmitted to the device on the other end of the serial communication line.

Thus the Modbus protocol defines the structure of the data packet in which the relevant information to be exchanged are encapsulated. The fields consituting the data packet are: destination address, function code of the action to be performed, the relevant data for the operation and an error verification code.

When a Slave device receives a data packet, it decodes the data and execute the operation that the Master device has required (operation that can be reading or writing data in the Slave device memory).
At this point the Slave device create the needed data packet and “returns it to sender”. The information in the response message is the slave device address, the action performed, the data acquired as a result of the action and a mean of checking errors.

The entire task can be described as follows:

Fig. 2 Communication overview



The Modbus protocol implemented in the SAIA chamber controller is the binary type (also known as RTU). In RTU mode data are sent as 8 bits binaries characters. A remote control computer (typically a PC) is always the master device meanwhile the SAIA controller is the slave device.

ERROR DETECTION

As well known, communication errors can occour during data transmission on a serial line. The errors generally come from noise on the line and electrical or hardware problems on the devices used for the communication.
The software that controls the communication (the PC) takes care to identify the errors and manage them with appropriate actions.
Specifically, when an error is detected, the data packet must be discarded (not used) because the data inside are probably corrupted and meaningless.

Only when no error is present, the data packet can be decoded to extract the answer coming from the chamber controller.

The errors can be present in both sides of the serial line (PC side and chamber controller side). If the error is identified by the chamber controller, then no answer will be send to the master.

Fig. 3 Noise on the line

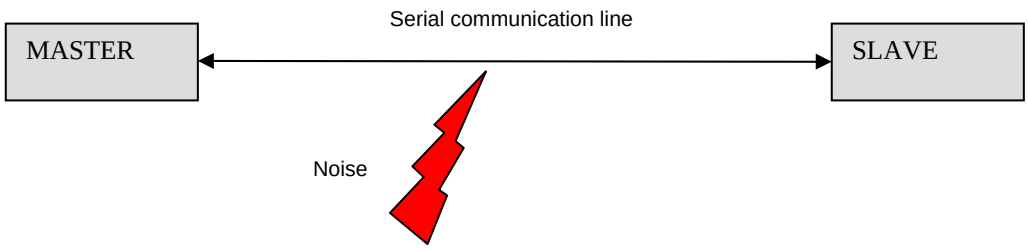
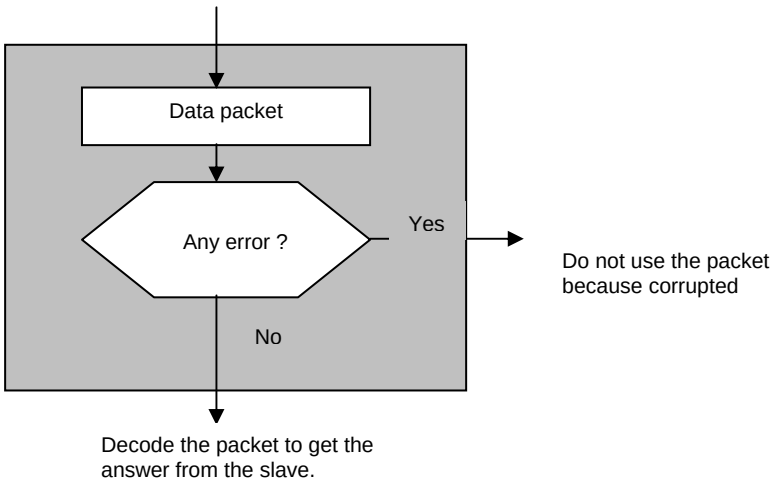


Fig. 4 Decoding a data packet



There are 2 types of errors which may occour: transmission errors and programming or operation errors. The Modbus system has specific methods for dealing with either type of error.

COMMUNICATION ERRORS

This kind of errors happens when a condition of **FRAME** or **PARITY** or **OVERRUN** error is detected (generally due to same problems of noise on the transmission lines) or when an invalid **CRC** (cyclical redundancy check) is found in the received data packet.
When one or more of these errors is detected, the **message (**probably damaged**) is not processed**: the **Slave device (SAIA) will not answer to the message.**

Operation / programming errors
Are those which present incorrect data in a message. In this case the Slave device (SIEMENS S7) produces an **"exception"** depending on the error type; the exception error code is inserted in the answer data packet, so that it allows the master device (PC) to identify the error condition.

To identify this exception, the **most significant bit of the function code of the answer packet (coming from the SAIA to the PC) is set to 1.**

Exception codes

01 Illegal function
02 Illegal data address
03 Illegal data value
04 Failure in associated device
05 Acknowledge: the slave is processing the request
06 Busy: busy device, rejected message
NAK: It is impossible to execute the requested function
07 Memory parity error

For example, let us say that we want to read data from the slave. Master prepares a data packet (query) with modbus function code 3 (function code 3 identifies a READ request as we will see later) and send the packet to the slave.

In preparing the answer, the slave will return the same function code if no exception is present, otherwise it will return the exception code as reported below (note that only the first two bytes of the data packet are showed here: the detailed explanation of the entire frame will follow in the specific paragraph)

QUERY FROM THE MASTER TO THE SLAVE

0	ADDRESS	Chamber controller address
1	03 dec: 00000011 binary	Modbus function code for read operation
2	….	
3	….	
4	….	

ANSWER FROM THE SLAVE, NO EXCEPTION

0	ADDRESS	Chamber controller address
1	03 hex: 00000011 binary	Modbus function code for read operation
2	….	
3	….	
4	….	

ANSWER FROM THE SLAVE, EXCEPTION 02

0	ADDRESS	Chamber controller address
1	83 hex: 10000011 binary	Modbus function code + exception
2	02	Exception number: illegal data address
3	Error check	
4	Error check	

IDENTIFICATION OF A DATA PACKET

Two different ways are available to the programmer to identify a data packet in binary mode:

- FRAME SYNCHRONIZATION
- LENGHT SYNCHRONIZATION.

In the following section a short description of both will be done in order to have an overview on advantages and difficulties of both.

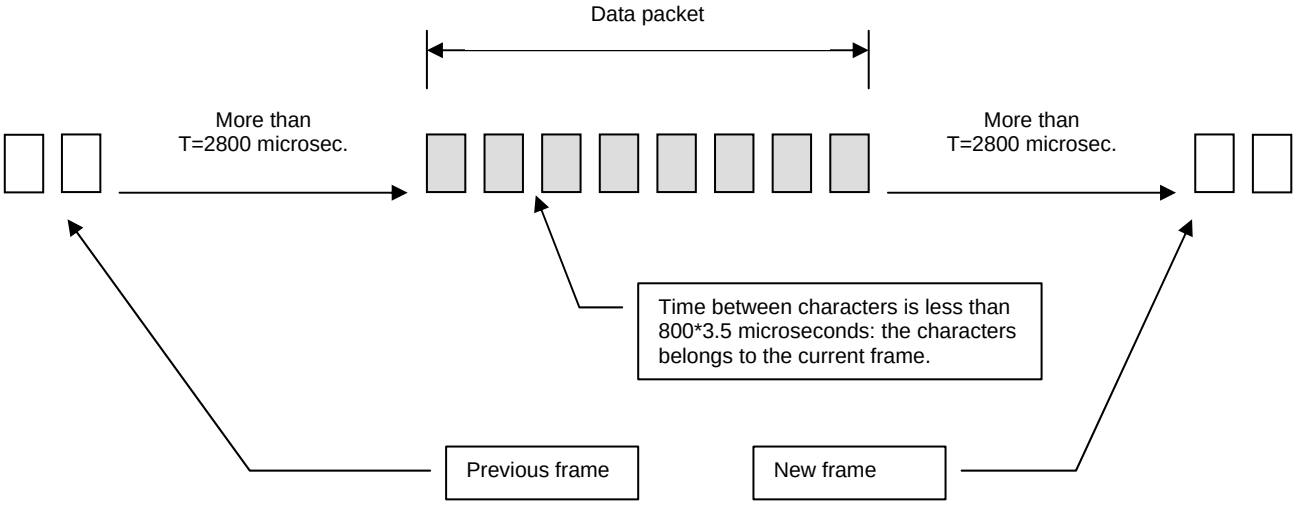
FRAME SYNCHRONIZATION

Frame synchronization can be manteined in binary transmission mode (RTU) only by simulating a synchronous message. The receiving device monitors the elapsed time between receipt of characters.

Modbus specifications requests that this time (time between the reception of a character and the following) is less or equal to **3.5 times the time of character transmission** because the received data can be considered as part of the current data packet. If not, the character will be considered as the first character of a new data packet.
For example, at 9600 bps the time between the transmission of a character and the next one is approximately 800 microsec. If the next character will be received after more than 800 * 3.5 microseconds, the character will be the first of a new data packet.

The following picture shows how a received character belongs to the current data packet (frame) or begins a new frame according to the time elapsed since the previous character was detected.
In this picture the horizontal axis is the time axis and each rectangle identifies a character. A packet consists of all characters with intercharacter time less than time specified by Modbus (T = 800 * 3.5 microseconds at 9600 bps)

Fig. 5 Frame identification



Note that this method requires a very precise timing to get data from serial line. Considering that Windows is not a real-time system (and so it cannot garantee the timing required) it is preferable not to use it under this operating system.

LENGHT SYNCHRONIZATION

Each modbus function has a well defined length, both for queries and answers. Knowing these lengths, it is possible to identify the end of a packet while receiving.
This way of operation is probably the simplest to use, both for development of a commincation software and for the debug operation.
In effect you always know how many character you are waiting for according to the query you sent to the slave device.
This is also true when an error occours during the communication: the lenght of an answer with an exception is however known.

MODBUS FUNCTIONS

All the operations with the chamber controller can be done with only two of the Modbus functions:

- Read Function: function code 3 (decimal)
- Write Function: function code 16 (decimal)

Master can use these functions in order to take actions on the slave (chamber controller).

READ FUNCTION (FUNCTION CODE: 3)

The Modbus read operation (**function code 3**) can be described as follows:

FUNCTION CODE	QUERY LENGTH (without error check field)	ANSWER LENGTH (without error check field)
3 (decimal)	**6 bytes**	**3 bytes + content of the 3rd byte of the answer**

WRITE operation (FUNtION CODE: 16)

The write operation (function code 16) is described as following

FUNCTION CODE	QUERY LENGTH. (without error check field)	LUNGHEZZA RISPOSTA (without error check field)
16 (decimal)	**7 + the content of the 7th byte**	**6 bytes**

In order to get the total data length, add 2 bytes for CRC.

Each function is organized in fields with specific meaning. In the following section a short description of the fields is given, and a detailed explanation of the read and write functions follows.

FIELDS DESCRIPTION

ADDRESS FIELD

This is the first field of the structure and consists of 1 byte. This byte identifies the address of the chamber controller (slave) to work with.
Each slave must have a unique address and only the addressed slave will respond to a query that contains its address.

The chamber controller address is always 17 (decimal)

FUNCTION FIELD

Tells the addressed slave what function to perform. The high order bit (**H**igh **O**rder **HO** in the following) in this field is set by the slave device to indicate that other than a normal response is being transmitted to the Master device (see the discussion regarding the exceptions in the answer). The bit remains 0 if the message is a query or a normal response message.

DATA FIELD

Contains information needed by the slave to perform the specific function or it contains data collected by the slave in response to a query.

ERROR CHECK FIELD

The error check field uses a CRC calculation, which described later in the examples.

MODBUS FUNCTIONS DESCRIPTION

For each function an example is given for the query packet from the PC and the answer packet from the slave. Take into account that the chamber controller has 17 (decimal) as address and that the addresses of the memory locations to read from and to write to are listed in the next section of this manual.
Some conventions are used to describe the data inside the functions:

"**HO**": **H**igh **O**rder or Most Significant (refers to bits or bytes depending on the situation)
"**LO**": **L**ow **O**rder or Least Significant (refers to bits or bytes depending on the situation)
"**Register**": a memory location inside the chamber controller memory area; it consistes of 2 bytes

Master prepares a query packet to read from or to write to the chamber controller, then waits for the answer. When the answer is available a check is done to identify the errors (if any). The decode operation is the last action to be performed when no errors have been encountered in the data packet.

READ MULTIPLE REGISTERS: FUNCTION CODE: 3 (DECIMAL)

Allows the user to obtain the binary contents of holding registers in the addressed slave (the chamber controller)
The addressing allows up to 125 registers to be obtained at each request.
The registers are numbered starting from 0.

The below example reads 104 registers from slave starting at address 0. All values are in decimal format.

Note that the following table is exactly the content of a byte array you can send to the serial device in order to read from the chamber controller.

QUERY

INDEX	MEANING	TX BUFFER
0	PLC ADDRESS	17
1	FUNCTION NUMBER	03
2	DATA START REGISTER, HO	00
3	DATA START REGISTER, LO	00
4	REGISTERS TO BE READ, HO	00
5	REGISTERS TO BE READ, LO	104
6	ERROR CHECK	70
7	ERROR CHECK	180

ANSWER

The addressed slave responds with its address and the function code, followed by the information field. The information field contains 2 bytes describing the quantity of data bytes to be returned. The contents of the requested registers (data) are two bytes each, with the binary content right justified within each pair of characters.
The first byte includes the high order bits and the second the low order bits.

DATA	MEANING
17	ADDRESS
03	FUNCTION NUMBER
208 (&HD0)	BYTE IN THE PACKET (REG. REQUESTED * 2)
	HO OF REG. OF DATA OUTPUT 0
	LO OF REG. OF DATA OUTPUT 0
	HO OF REG. OF DATA OUTOPUT 1
	LO OF REG. OF DATA OUTPUT 1
	HO OF REG. OF DATA OUTPUT 2
	LO OF REG. OF DATA OUTPUT 2

	ERROR CHECK
	ERROR CHECK

Data that must be decoded according to the description of SIEMENS S7 memory

The total length of the data packet expected is of 213 bytes (3 bytes of header + 208 + 2 bytes CRC)

WRITING FUNCTION: FUNCTION CODE: 16 (DECIMAL)

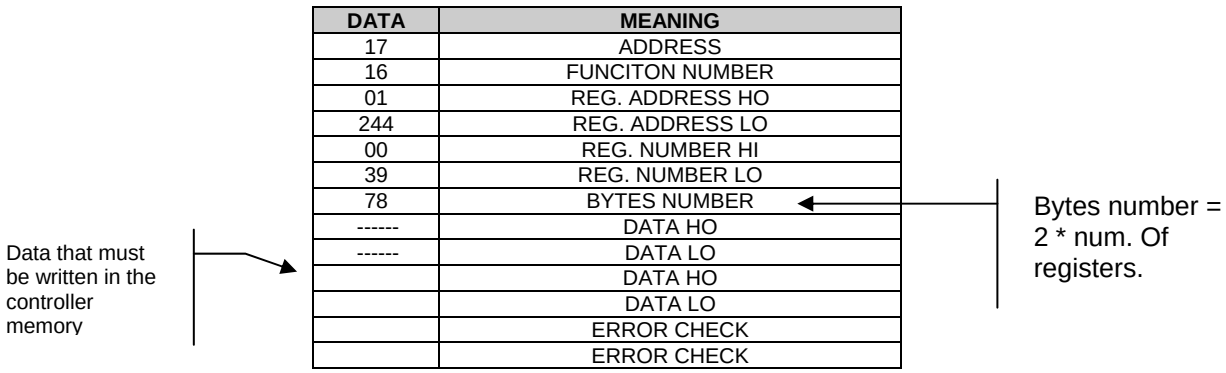
NOTE:
This operation overwrite existing memory values. Be carefull to write only to the right position according to the SAIA memory layout.
Wrong write operation (wrong address) can damage specific memory contents reserved to the functionality of the system.

Registers existings within the controller can have their contents changed by this message (a maximum of 60 registers)

The following example shows the writing of 39 registers (78 bytes) at address 500 (decimal). This address is a controller memory area reserved to write operations for chamber control. You can find the description of this area in the specific section of this manual.

All values are in decimal.

QUERY



DATA	MEANING
17	ADDRESS
16	FUNCITON NUMBER
01	REG. ADDRESS HO
244	REG. ADDRESS LO
00	REG. NUMBER HI
39	REG. NUMBER LO
78	BYTES NUMBER
------	DATA HO
------	DATA LO
	DATA HO
	DATA LO
	ERROR CHECK
	ERROR CHECK

ANSWER

The normal answer to function 16 returns the following data, with a total packet length of 8 bytes.

DATA	MEANING
17	ADDRESS
16	FUNCTION NUMBER
01	ADDRESS HO
244	ADDRESS LO
00	REG. NUM. HO
39	REG. NUM. LO
194	ERROR CHECK
141	ERROR CHECK

COMMUNICATION PARAMETERS

The following parameters must be used to configure the serial port of the PC in order to communicate with the chamber controller.
These parameters cannot be changed.

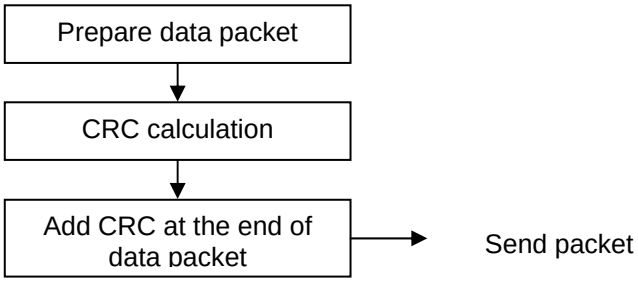
NAME	VALUE
BAUD RATE	9600
PARITY	NONE
STOP BITS	1
DATA LENGHT	8

CODE EXAMPLES

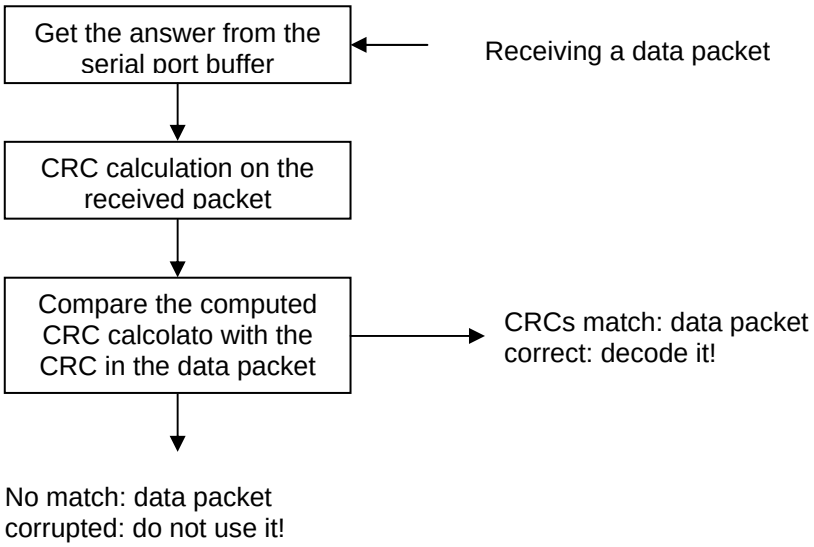
In this section we show how to compute the CRC according to the Modbus protocol requirements. Note that the CRC must be added to the packet to be sent to the controller and also must be computed on the received packet to verify its integrity.
Also some little programs are reported on how to prepare a query packet to talk to the controller using C language and Visual Basic.

Fig. 6 – CRC calculation

QUERY



ANSWER



Warning! All examples are only suggestions and indications of a possible program organization. ANGELANTONI Industrie declaims all responsibility for the use (correct or not correct) of this code.
It is up to the software engineer to modify, complete and test the examples for his own purposes.

C LANGUAGE

The following examples show how to initialize the CRC table and how to compute the CRC bytes to be added at the end of the packet to be sent to the controller.

```
#define CRC16  0xA001
static unsigned char TABLE1[256], TABLE2[256];

//Initialize tables for the calculation of the CRC
static void InitCRC()
{
   unsigned char i;
   unsigned int mask, crc, mem;

   for(mask=0;mask<256;mask++) {
      crc = mask;
      for(i=0;i<8;i++)
      {
         mem = (unsigned int)(crc & 0x0001) ;
         crc /=  2;
         if (mem) crc ^= CRC16 ;
      }
      TABLE2[mask] = (unsigned char)(crc & 0xff); /* lobyte */
      TABLE1[mask] = (unsigned char)(crc >> 8);   /* hibyte */
   }
}
```

```
//Calculate the CRC
static unsigned int CalcCRC(unsigned char *buf, unsigned char size)
{
   unsigned char car,i;
   unsigned char crc0,crc1;
   crc0 = 0xff;
   crc1 = 0xff;

   for(i=0;i<size;i++) {
     car = buf[i];
     car ^= crc0;
     crc0 = (unsigned char)(crc1 ^ TABLE2[car]);
     crc1 = TABLE1[car];
   }
   return (crc1<<8) + crc0;
```

The following code is just an example that shows how to elaborate a data block to be transmitted to the chamber controller in order to read back others data.
As explained previously, you have to specify the starting address of the data block and the number of the registers to be read.

QUERY (function 3)

```
void ModbusReadStatus(void)
{
int r, n, cmd;
unsigned char tx[50];
  //Read 104 words starting at address 0
  r   = 0;                              //starting address
  n   = 104;                            //number of registers to read
  tx[0] = 17;                           //SIEMENS S7 address
  tx[1] = 3;                            //Modbus read (function 3)
  tx[2] = (unsigned char)(r >> 8);      //first register (HO byte)    =     7
  tx[3] = (unsigned char)(r & 0xFF);    //first register (LO byte)    =     208
  tx[4] = (unsigned char)(n >> 8);      //num. of registers (HO byte) =     0
  tx[5] = (unsigned char)(n & 0xFF);    //num. Of registers (LO byte) =     104
  ModbusTxMsg(tx, sizeof(tx[0]) * 6);
}

//The ModbusTxMsg()function is used to reach the CRC field at the end of
//the packet and and send packet to the controller. Below there is just
//a skeleton of a possible way to implement the code.

   memcpy(pData, buffer, size);
  *(unsigned int *)(pData+size) = CalcCRC(pData,size);

//send data
```

In effect, in order to read 104 registers starting from address 0, the complete packet to be transmitted to the controller is as follows:

```
tx[0] = 17;
tx[1] = 3;
tx[2] = 0
tx[3] = 0
tx[4] = 0
tx[5] = 104
tx[6] = 70
tx[7] = 180
```

ANSWER (Function 3)

The PC software waits for the complete packet coming from the controller. As previously seen, a data packet can be detected using TIME SYNCHRONIZATION or FRAME SYNCHRONIZATION.
All data must be converted into a type according to the specifications of controller memory layout.

VISUAL BASIC

We assume you are familiar with VISUAL BASIC 5.0 (or upper) for Windows 95. Only the important sections of code are present in the following description.

CRC CALCULATION

```
Private Const CRC16 As Long = 40961  '0xA001 in exadecimal

Private table1(255) As Byte
Private table2(255) As Byte

Private Function calcCrc(buf() As Byte, size As Byte) As Long
'Calculation of CRC
Dim i As Byte
Dim car As Long
Dim crc0 As Long
Dim crc1 As Long

crc0 = &HFF
crc1 = &HFF

For i = 0 To size - 1
    car = buf(i)
    car = car Xor crc0
    crc0 = (crc1 Xor table2(car)) And &HFF
    crc1 = table1(car)
Next i

calcCrc = (crc1 * 256 + crc0) And &HFFFF

End Function
```

```
Public Sub initCrc()
'Initialize the tables to use by the calcCrc() function
Dim i As Byte
Dim mask As Long
Dim crc As Long
Dim mem As Long

For mask = 0 To 255
    crc = mask

    For i = 0 To 7
        mem = crc And &H1
        crc = crc \ 2
        If mem <> 0 Then crc = crc Xor CRC16
    Next i

    table2(mask) = crc And &HFF
    table1(mask) = crc \ 256

Next mask

End Sub
```

The query packet to be sent to the controller is reported below:

```
address = 0
mbtx(0) = 17                        'address of SAIA
mbtx(1) = 3                         'function number (reading))
mbtx(2) = address \ 256             'hi byte  (start reg.)
mbtx(3) = address And &HFF          'low byte (start reg.)
mbtx(4) = 0                         'hi byte  (num. reg.)
mbtx(5) = 104                       'low byte (num. reg.)
crc = calcCrc(mbtx(), 6)            'calculate the CRC
mbtx(6) = crc And &HFF              'low byte CRC
mbtx(7) = crc \ 256                 'hi byte CRC
```

READING AND WRITING REQUESTS FROM PC TO PLC

Note that the meaning of specific memory locations could be different from those reported here. Always refer to the memory layout of your chamber (as appendix to this document) to have the right information.

READ REQUEST

If we want to read data from the controller, we can use the following data packet:

```
> Query sent:
> 000:   017 0011 plc address
> 001:   003 0003 modbus function number (read function)
> 002:   000 0000 memory address high
> 003:   000 0000 memory address low
> 004:   000 0000 number of registers high
> 005:   087 0057 number of registers low          asks for 87 registers, that is 174 bytes
> 006:   006 0006 error check
> 007:   164 00A4 error check
> End packet.
```

WRITE REQUEST

Suppose we want to set the controller to the following situation:

Temperature: 50 Celsius degree
Rel. Humidity: 70%
Run status: ON
Temperature regulation: ON
Rel. Humidity regulation: ON

The packet to send the controller is reported below. The meaning of each field is described in the memory layout section. The first column is the index of the byte in the data packet

```
> Query sent:
> 000:   017 0011 plc address
> 001:   016 0010 modbus function number
> 002:   001 0001 memory address high
> 003:   244 00F4 memory address low
> 004:   000 0000 number of registers high
> 005:   012 000C number of registers low
> 006:   024 0018 byte count
> 007:   003 0003 high order data      Temperature channel ON, Rel. Humidity channel ON
> 008:   001 0001 low order data       Run bit ON
> 009:   000 0000 high order data
> 010:   000 0000 low order data
> 011:   000 0000 high order data
> 012:   000 0000 low order data
> 013:   000 0000 high order data
> 014:   000 0000 low order data
> 015:   066 0042 high order data      Temperature setpoint: 50 Celsius degree
> 016:   072 0048 low order data
> 017:   000 0000 high order data
> 018:   000 0000 low order data
> 019:   000 0000 high order data
> 020:   000 0000 low order data
> 021:   000 0000 high order data
> 022:   000 0000 low order data
> 023:   066 0042 high order data      Rel. Humidity setpoint: 70%
> 024:   140 008C low order data
> 025:   000 0000 high order data
> 026:   000 0000 low order data
> 027:   000 0000 high order data
> 028:   000 0000 low order data
> 029:   000 0000 high order data
> 030:   000 0000 low order data
> 031:   240 00F0 error check
> 032:   115 0073 error check
> End packet.
```

CALCULATED CRC TABLES

The following tables contain the CRC values computed using C e Visual Basic code previously reported. These tables can be used as riference in case you want to write a program using languages different from those are indicated in the examples.

TABLE 1

```
0    192  193  1    195  3    2    194          (items from 0 to 7)
198  6    7    199  5    197  196  4            (items from 8 to 15)
204  12   13   205  15   207  206  14
10   202  203  11   201  9    8    200
216  24   25   217  27   219  218  26
30   222  223  31   221  29   28   220
20   212  213  21   215  23   22   214
210  18   19   211  17   209  208  16
240  48   49   241  51   243  242  50
54   246  247  55   245  53   52   244
60   252  253  61   255  63   62   254
250  58   59   251  57   249  248  56
40   232  233  41   235  43   42   234
238  46   47   239  45   237  236  44
228  36   37   229  39   231  230  38
34   226  227  35   225  33   32   224
160  96   97   161  99   163  162  98
102  166  167  103  165  101  100  164
108  172  173  109  175  111  110  174
170  106  107  171  105  169  168  104
120  184  185  121  187  123  122  186
190  126  127  191  125  189  188  124
180  116  117  181  119  183  182  118
114  178  179  115  177  113  112  176
80   144  145  81   147  83   82   146
150  86   87   151  85   149  148  84
156  92   93   157  95   159  158  94
90   154  155  91   153  89   88   152
136  72   73   137  75   139  138  74
78   142  143  79   141  77   76   140
68   132  133  69   135  71   70   134
130  66   67   131  65   129  128  64
```

TABLE 2

```
0  193  129  64  1  192  128  65          (items from 0 to 7)
1  192  128  65  0  193  129  64          (items from 8 to 15)
1  192  128  65  0  193  129  64
0  193  129  64  1  192  128  65
1  192  128  65  0  193  129  64
0  193  129  64  1  192  128  65
0  193  129  64  1  192  128  65
1  192  128  65  0  193  129  64
1  192  128  65  0  193  129  64
0  193  129  64  1  192  128  65
0  193  129  64  1  192  128  65
1  192  128  65  0  193  129  64
0  193  129  64  1  192  128  65
1  192  128  65  0  193  129  64
1  192  128  65  0  193  129  64
0  193  129  64  1  192  128  65
1  192  128  65  0  193  129  64
0  193  129  64  1  192  128  65
0  193  129  64  1  192  128  65
1  192  128  65  0  193  129  64
0  193  129  64  1  192  128  65
1  192  128  65  0  193  129  64
1  192  128  65  0  193  129  64
0  193  129  64  1  192  128  65
0  193  129  64  1  192  128  65
1  192  128  65  0  193  129  64
1  192  128  65  0  193  129  64
0  193  129  64  1  192  128  65
1  192  128  65  0  193  129  64
0  193  129  64  1  192  128  65
0  193  129  64  1  192  128  65
1  192  128  65  0  193  129  64
```

SAIA MEMORY LAYOUT VERS. 2.0

NOTE:
It's very important that writing operations in SAIA are correct and right. In effect it is absolutely important that the data written to the SAIA are correct from the format point of view and from the addresses point of view.
Wrong data or wrong addresses for data can be very dangeorus for the functionality of the controller.
It is operator's liability to make sure that all safety conditions are guaranteed, in order to avoid damages to the chamber controller and chamber devicecs or to prevent its right functioning through wrong operations.
ANGELANTONI Industrie doesn't take any liability on itself, neither directly or inderectly, about problems caused by bad functioning due to software development based on this manual.

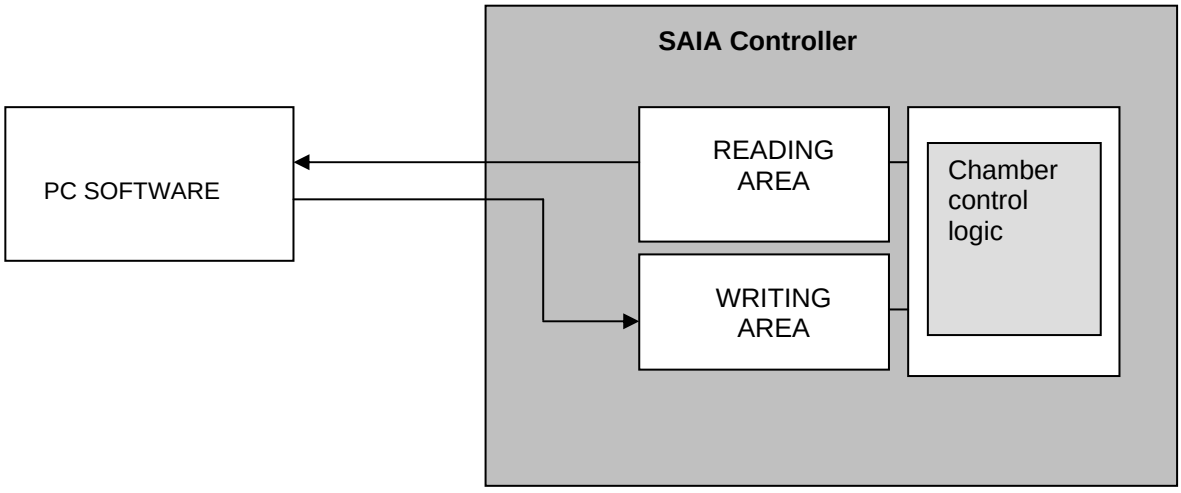
In the following section the memory layout of the chamber controller version 2.00 is shown
In this version the entire area available to the user is divided in two different sections, the first of them dedicated only for reading and the second one for writing.

Please, note that the SAIA controller provides a wide range of facilities (temperature inputs, analog inputs, contacts, regulators and so on) and only a sub-set of these could be really used to control the chamber.
Please, always refers to the memory layout to have the detailed description of what available in your system.

The PC software normally reads the **reading area** in order to know the status (measures, actiovations, alarms) of the chamber.
When a new status for the chamber is required (for example, the user wants to change the temperature set for the chamber from 35 to 75 Celsius degree), the PC software makes a query to the SIEMENS S7, writing data into the **writing area.** When done, the PC software simply turn back to its normal task: reading from the chamber.
According to the data sent by the PC, chamber controller will try to act in order to satisfy the user requests, if possible.

A logic layout is shown below:

Fig. 7 System logic layout



As already stated, according to the type of chamber you have, the meanings of some channels could be different from those reported here. This because the SAIA controller is used for custom chambers and not for standard systems.

CHAMBER CONTROL AT A GLANCE

From logical point of view a chamber can be seen as:

Measure channels

Analogic channels, temperature and humidity channels and so on.
In effect each channel is related to a specific parameter to acquire.

Controlled channels

Temperature and rel. humidity are controlled channels, for example.
The user defines a setpoint and the internal (PLC) PID try to reach (and mantain) the required setpoint.
Note that it is necessary also to enable the channel to work. In other words, you need to give the regulator the setpoint and the “enable” command (a bit in a specific word inside the memory layout, as you will see later).

Activations

Rele' are available to the user

Alarms

ON/OFF signals coming from chambers. they identifies specific conditions or events.

All these information are the matter to work with, even if a very little subset is strictly necessary to make the system working. It depends on the user requirements to decide how many data to acquire.

The entire set of data as described above is available in a specific PLC memory area, as described into the documentation.

RUNNING A TEST

A test is defined by T setpoint and slope, RH setpoint and slope (if RH control is required), activation status (ON or OFF) for the channels, activation status for the contacts.
All these data have to be sent to the Write Area of the mplc, according to the PLC memory layout.
The chamber will work with the sent parameters until the user will change them.

To stop a test, simply make the same operations as before, forcing to zero all the values and the contact status.

POSSIBLE SOFTWARE ORGANIZATION

The system always make the 2 only possible tasks: read or write.
The read operation is always done if no write request is pending. A write request is generated when the user prepares data (i.e. setpoint or other) to send to the mPLC. The software has to complete the current operation then it can perform the write operation.

All the querys have to be controlled to check the CRC and to be sure that no comms error is present.
If errors have been found, discard the packet because it is probably corrupted.If all things look good, decode the packet according to the mPLC memory layout, and assign to the internal software structures.

The following pseudo-code could be called by a timer used to schedule the communications.

```
if (communication_already_in_progress == 0) {

          if (user_write_request)
               prepare_write_query();
          else
               prepare_read_query();

          send_query()
          communication_already_in_progress = 1;
}

else {
        receive_data()
        if (no_error and no_badCRC) decode_data()
         communication_already_in_progress = 0;
}
```

READING AREA

The information in this area allows the user to know the general status of the system.
According to the application to be developed only a part of these information could be necessary.
Gray backgorund (where present) identifies items not used in this system.
A skeleton of a conversion C function is showed for each table. Please note that the type “int” represents 4 bytes while “short int” is 2 bytes.

SYSTEM MEASURES

MEASURE INDEX	MEASURE ADDRESS	MEANING	NOTES
0	0	Dry bulb temperature/PT100 suction	
1	2	Wet bulb/PT100 discharge	
2	4	PT100-User n.0	
3	6	PT100-User n.1	
4	8	PT100-User n.2	
5	10	PT100-User n.3	
6	12	Low stage suction presuure	
7	14	Low stage discharge pressure	
8	16	High stage suction pressure	
9	18	High stage discharge pressure	
10	20	User analog input 0	
11	22	User analog input 1	
12	24	User analog input 2	
13	26	User analog input 3	
14	28	User analog input 4	
15	30	User analog input 5/Capacitive probe	
16	32	Chamber Temperature	
17	34	Relative humidity	
18	36	Absolute Humidity	
19	38	Not used	
20	40	Not used	
21	42	Not used	
22	44	Not used	
23	46	Not used	
24	48	Not used	
25	50	Not used	
26	52	CHannel 0 Measure	
27	54	CHannel 1 Measure	
28	56	Not used	
29	58	Not used	
30	60	Not used	
31	62	Not used	

Use this function to decode floating point value coming from Siemens S7 into C floating point

```
float conv_float_s7_pc(char * plc_real)
{
  char tmp[4];

  tmp[0] = * (plc_real + 3);
  tmp[1] = * (plc_real + 2);
  tmp[2] = * (plc_real + 1);
  tmp[3] = * (plc_real + 0);

  return (* (float *) tmp) ;
}
```

USER AND REAL SETTINGS

It is important to well understand the logic operations in sending a new data configuration (we intend a new group of data the chamber has to use, for example new temperature settings, new RUN/STOP status and so on).
In effect, this operation can be seen as composed of 2 different stages:

1. **the user prepares the new set of data and send it to the controller (user settings)**
2. **the controller acts properly to work with the new set (if possible) (real settings)**

From the above description you can understand that the final result of the entire operation can or cannot be the same as requested by the user depending on the chamber and controller status. For example, suppose that an alarm is present in the system, so the chamber is in alarm condition. If the user send a new set of data in order to change the status of the RUN bit (the user wants the chamber running), no action will take place, because of the alarm.
In other words, we have a “user setting” group of data in which the RUN bit is in ON status, while the same bit in the “real settings” will be in OFF status.

ACTIVATIONS AND LOGICAL STATUS (USER SETTINGS)

INDEX	USER SETTINGS ADDRESS	REAL SETTINGS ADDRESS	MEANING	NOTES
1	69.0	73.0	Run	
2	69.1	73.1	Alarm reset	
3	69.2	73.2	Program (1) – manual (0) mode	
4	69.3	73.3	Local(1) – Remote (0) mode	RESERVED. KEEP AT 0
5	69.4	73.4	Pause	RESERVED. KEEP AT 0
6	69.5	73.5	Freeze	RESERVED. KEEP AT 0
7	69.6	73.6	Not used	RESERVED. KEEP AT 0
8	69.7	73.7	Not used	RESERVED. KEEP AT 0
9	69.8	73.8	Enable channel n. 0	TEMPERATURE
10	69.9	73.9	Enable channel n. 1	HUMIDITY
11	69.10	73.10	Enable channel n. 2	
12	69.11	73.11	Enable channel n. 3	
13	69.12	73.12	Enable channel n. 4	
14	69.13	73.13	Enable channel n. 5	
15	69.14	73.14	Enable channel n. 6	
16	69.15	73.15	Enable channel n. 7	

INDEX	USER SETTINGS ADDRESS	REAL SETTINGS ADDRESS	MEANING	NOTES
17	70.0	74.0	Dedicated contact n. 01	Dehumidification
18	70.1	74.1	Dedicated contact n. 02	Vibration system
19	70.2	74.2	Dedicated contact n. 03	Device under test
20	70.3	74.3	Dedicated contact n. 04	UV Lamp
21	70.4	74.4	Dedicated contact n. 05	Water recharge
22	70.5	74.5	Dedicated contact n. 06	LN2 AUX
23	70.6	74.6	Dedicated contact n. 07	LN2
24	70.7	74.7	Dedicated contact n. 08	Dry air
25	70.8	74.8	Dedicated contact n. 09	Auxiliary relais 1
26	70.9	74.9	Dedicated contact n. 10	Auxiliary relais 2
27	70.10	74.10	Dedicated contact n. 11	Auxiliary relais 3
28	70.11	74.11	Dedicated contact n. 12	Auxiliary relais 4
29	70.12	74.112	Dedicated contact n. 13	Auxiliary relais 5
30	70.13	74.13	Dedicated contact n. 14	Auxiliary relais 6
31	70.14	74.14	Dedicated contact n. 15	Auxiliary relais 7
32	70.15	74.15	Dedicated contact n. 16	Auxiliary relais 8
33	71.0	75.0	Auxiliary conatct n. 01	
34	71.1	75.1	Auxiliary conatct n. 02	
35	71.2	75.2	Auxiliary conatct n. 03	
36	71.3	75.3	Auxiliary conatct n. 04	
37	71.4	75.4	Auxiliary conatct n. 05	
38	71.5	75.5	Auxiliary conatct n. 06	
39	71.6	75.6	Auxiliary conatct n. 07	
40	71.7	75.7	Auxiliary conatct n. 08	
41	71.8	75.8	Auxiliary conatct n. 09	
42	71.9	75.9	Auxiliary conatct n. 10	
43	71.10	75.10	Auxiliary conatct n. 11	
44	71.11	75.11	Auxiliary conatct n. 12	
45	71.12	75.12	Auxiliary conatct n. 13	
46	71.13	75.13	Auxiliary conatct n. 14	
47	71.14	75.14	Auxiliary conatct n. 15	
48	71.15	75.15	Auxiliary conatct n. 16	
			VACUUM CHAMBERS ONLY	
49	72.0	76.0	Vacuum command n. 01	
50	72.1	76.1	Vacuum command n. 02	
51	72.2	76.2	Vacuum command n. 03	
52	72.3	76.3	Vacuum command n. 04	
53	72.4	76.4	Vacuum command n. 05	
54	72.5	76.5	Vacuum command n. 06	
55	72.6	76.6	Vacuum command n. 07	
56	72.7	76.7	Vacuum command n. 08	
57	72.8	76.8	Vacuum command n. 09	
58	72.9	76.9	Vacuum command n. 10	
59	72.10	76.10	Vacuum command n. 11	
60	72.11	76.11	Vacuum command n. 12	
61	72.12	76.12	Vacuum command n. 13	
62	72.13	76.13	Vacuum command n. 14	
63	72.14	76.14	Vacuum command n. 15	
64	72.15	76.15	Vacuum command n. 16	

The following function can be used to decode data coming from Siemens S7

```
short int conv_word_mplc_to_word_pc(unsigned char * buf)
{
  return ((short int) ((* (buf + 0)) << 8) + ((* (buf + 1)) << 0));
}
```

After decoding, each bit of the word will have the meaning previously described.

ALARMS AND MESSAGES

The following messages list is general: ***some of them could not be used in your chamber***. Note that messages are associated to bits, so the term “64.3” means “bit number 3 (counting from 0) at address 64”.

INDEX	ADDRESS	MEANING	NOTES
1	64.0	Safety temperature switch	OR of each alarm device
2	64.1	Overload cutout	
3	64.2	Thermal cutout	
4	64.3	Power fail	
5	64.4	Emergency switch	
6	64.5	Door open	
7	64.6	Max temperature	
8	64.7	Adjustable max temperature	
9	64.8	Adjustable min temperature	
10	64.9	Pt100 dry bulb sensor fail	Sensors
11	64.10	TC dut sensor fail	
12	64.11	Pt100 wet bulb sensor fail	
13	64.12	Salt solution min level	
14	64.13	Water lack	Relative humidity control
15	64.14	Steam generator max temperature	
16	64.15	Low stage compressor min oil pressure	Low stage
17	65.0	Low stage compressor overload cutout	
18	65.1	Low stage compressor thermal cutout	
19	65.2	Low stage fan compressor overload cutout	
20	65.3	Max low stage pressure	
21	65.4	Min low stage pressure	
22	65.5	Max low stage discharge temperature	
23	65.6	Intermediatre fluid min oil pressure	High stage
24	65.7	Intermediate fluid min temperature	
25	65.8	Intermediate fluid max level	
26	65.9	Intermediate fluid min level	
27	65.10	Intermediate fluid max pressure	
28	65.11	Intermediate fluid min pressure	
29	65.12	Max high stage discharge temperature	
30	65.13	Intermediate fluid pump overload cutout	Baysilone, Coolanol, Glicole, Gladen ….
31	65.14	Intermediate fluid pump thermal cutout	
32	65.15	Max intermediate fluid level	
33	66.0	Condenser fan overload cutout	
34	66.1	Intermediate exch. Fan overload cutout	
35	66.2	Min condenser water flow	
36	66.3	Min compressed air pressure	
37	66.4	Min differential pressure	
38	66.5	Min liquid nytrogen pressure	
39	66.6	Smoke detector	
40	66.7	Max CO concentration	
41	66.8	Min oxygen concentration	
42	66.9	Inverter fail	
43	66.10	Vacuum pump overload cutout	
44	66.11	Vacuum pump thermal cutout	
45	66.12	Vacuum pump warmup	
46	66.13	Ventilation lack	
47	66.14	Air filter	
48	66.15	Low stage compressor start-up failure	

INDEX	ADDRESS	MEANING	NOTES
49	67.0	High stage compressor start-up failure	
50	67.1	Basket timeout	
51	67.2	Defrosting in progress	
52	67.3	Defrosting timeout	
53	67.4	Capacitive probe temperature out of range	
54	67.5	Acceleration critical deviation	
55	67.6	Oxygen min. concentration (Non critical)	
56	67.7	Hardware alarm (WinSmartkit)	
57	67.8		
58	67.9		
59	67.10		
60	67.11		
61	67.12		
62	67.13		
63	67.14		
64	67.15		
65	68.0		
66	68.1		
67	68.2		
68	68.3		
69	68.4		
70	68.5		
71	68.6		
72	68.7		
73	68.8		
74	68.9		
75	68.10		
76	68.11		
77	68.12	Remote mode control	
78	68.13	Abort test in program mode	
79	68.14	End test	
80	68.15	Critical alarm	

The decode operation on data coming from Siemens S7 can be accomplished by the following function:

```
short int conv_word_mplc_to_word_pc(unsigned char * buf)
{
  return ((short int) ((* (buf + 0)) << 8) + ((* (buf + 1)) << 0));
}
```

After decoding, each bit of the word will have the meaning previously described.

SETPOINTS READING

The values reported here are the setpoint values currently used by the regulators inside the chamber controller.
Note that the final setpoint equals the current setpoint only in case of maintenances or slopes at max. speed.
They will be different in case of controlled slopes (gradient different from 0).
As previously stated, in the standard chambers channel 0 is the Temperature regulator, while channel 1 is the Rel. Humidity regulator.
Note: the word “gradient” is used in the meaning of “slope”, even if the real phisical meaning is different

Please, note that all the values are floating point, so they require 2 words each.

CHANNEL INDEX	ADDRESS	MEANING	NOTES
0	77-78	User final setpoint	TEMPERATURE
0	79-80	Current user setpoint	TEMPERATURE
0	81-82	Gradient	TEMPERATURE
1	83-84	User final setpoint	RELATIVE HUMIDITY
1	85-86	Current user setpoint	RELATIVE HUMIDITY
1	87-88	Gradient	RELATIVE HUMIDITY
2	89-90	User final setpoint	NOT USED
2	91-92	Current user setpoint	NOT USED
2	93-94	Gradient	NOT USED
3	95-96	User final setpoint	NOT USED
3	97-98	Current user setpoint	NOT USED
3	99-100	Gradient	NOT USED
4	101-102	User final setpoint	NOT USED
4	103-104	Current user setpoint	NOT USED
4	105-106	Gradient	NOT USED
5	107-108	User final setpoint	NOT USED
5	109-110	Current user setpoint	NOT USED
5	111-112	Gradient	NOT USED
6	113-114	User final setpoint	NOT USED
6	115-116	Current user setpoint	NOT USED
6	117-118	Gradient	NOT USED
7	119-120	User final setpoint	NOT USED
7	121-122	Current user setpoint	NOT USED
7	123-124	Gradient	NOT USED

How to decode a floating point value coming from Siemens SIMATIC S7 into C floating point variable:

```
float conv_float_s7_pc(char * plc_real)
{
  char tmp[4];

  tmp[0] = * (plc_real + 3);
  tmp[1] = * (plc_real + 2);
  tmp[2] = * (plc_real + 1);
  tmp[3] = * (plc_real + 0);

  return (* (float *) tmp) ;
}
```

TEST PROGRAM STATUS

All information on the test cycle status currently running are reported in this section.

	ADDRESS	MEANING	NOTES
	125	Total segment number	
	126	First segment	
	127	Current segment	
	128-129	Current segment duration (seconds)	
	130-131	Current segment elapsed time (seconds)	
	132-133	Current segment remaining time (seconds)	
	134-135	Test program elapsed time (seconds)	
	136	RESERVED	
CONTACT NUMBER	ADDRESS	DEDICATED CONTACTS	NOTES
1	137.0	Dedicated contact n. 01	Dehumidification
2	137.1	Dedicated contact n. 02	Vibration system
3	137.2	Dedicated contact n. 03	Device under test
4	137.3	Dedicated contact n. 04	UV Lamp
5	137.4	Dedicated contact n. 05	Water recharge
6	137.5	Dedicated contact n. 06	LN2 AUX
7	137.6	Dedicated contact n. 07	LN2
8	137.7	Dedicated contact n. 08	Dry air
9	137.8	Dedicated contact n. 09	Auxiliary relais 1
10	137.9	Dedicated contact n. 10	Auxiliary relais 2
11	137.10	Dedicated contact n. 11	Auxiliary relais 3
12	137.11	Dedicated contact n. 12	Auxiliary relais 4
13	137.12	Dedicated contact n. 13	Auxiliary relais 5
14	137.13	Dedicated contact n. 14	Auxiliary relais 6
15	137.14	Dedicated contact n. 15	Auxiliary relais 7
16	137.15	Dedicated contact n. 16	Auxiliary relais 8
CONTACT NUMBER	ADDRESS	AUXILIARY CONTACTS	NOTES
1	138.0	Auxiliary conatct n. 01	
2	138.1	Auxiliary conatct n. 02	
3	138.2	Auxiliary conatct n. 03	
4	138.3	Auxiliary conatct n. 04	
5	138.4	Auxiliary conatct n. 05	
6	138.5	Auxiliary conatct n. 06	
7	138.6	Auxiliary conatct n. 07	
8	138.7	Auxiliary conatct n. 08	
9	138.8	Auxiliary conatct n. 09	
10	138.9	Auxiliary conatct n. 10	
11	138.10	Auxiliary conatct n. 11	
12	138.11	Auxiliary conatct n. 12	
13	138.12	Auxiliary conatct n. 13	
14	138.13	Auxiliary conatct n. 14	
15	138.14	Auxiliary conatct n. 15	
16	138.15	Auxiliary conatct n. 16	

The decode operation on data coming from Siemens S7 can be accomplished by the function reported below. After decoding, each bit of the word will have the meaning previously described.

```
short int conv_word_mplc_to_word_pc(unsigned char * buf)
{
  return ((short int) ((* (buf + 0)) << 8) + ((* (buf + 1)) << 0));
}
```

Converting the time representation from Siemens S7 (double word) into PC representation

```
int conv_dword_s5s7_pc(char * plc_dword)
{
  char tmp[4];

  tmp[0] = * (plc_dword + 3);
  tmp[1] = * (plc_dword + 2);
  tmp[2] = * (plc_dword + 1);
  tmp[3] = * (plc_dword + 0);

  return(* (int *) tmp);
}
```

WRITING AREA

This is the “control” area wherein the user must write the new operative configuration he wants for the system. The term “new operative configuration” means the desired status of the relays, setpoints, gradients and channels in order to obtain a certain behaviour from the chamber.

The setpoint and gradient values of the various channels are represented as floating point values. So, for instance, if you want to set the value of 75.0 degrees with gradient of 1.5 Celsius degrees/minute, it will be necessary to write 75 and 1.5 in the correspondent locations.
As already stated, channel 0 refers to the temperature channel.
Gray background identifies items not used for the current system.

ADDRESS	MEANING	NOTES
500.0	Run	
500.1	Alarm reset	
500.2	Program(1) - Manual(0) mode	
500.3	Local(1) - Remote(0) mode	Reserved, keep at 0
500.4	Pause (reserved)	Reserved, keep at 0
500.5	Freeze (reserved)	Reserved, keep at 0
500.6	Not used	Reserved, keep at 0
500.7	Not used	Reserved, keep at 0
500.8	Enable channel n. 0	TEMPERATURE
500.9	Enable channel n. 1	RELATIVE HUMIDITY
500.10	Enable channel n. 2	NOT USED
500.11	Enable channel n. 3	NOT USED
500.12	Enable channel n. 4	NOT USED
500.13	Enable channel n. 5	NOT USED
500.14	Enable channel n. 6	NOT USED
500.15	Enable channel n. 7	NOT USED
501.0	Dedicated contact n. 01	Dehumidification
501.1	Dedicated contact n. 02	Vibration system
501.2	Dedicated contact n. 03	Device under test
501.3	Dedicated contact n. 04	UV Lamp
501.4	Dedicated contact n. 05	Water recharge
501.5	Dedicated contact n. 06	LN2 AUX
501.6	Dedicated contact n. 07	LN2
501.7	Dedicated contact n. 08	Dry air
501.8	Dedicated contact n. 09	Auxiliary relais 1
501.9	Dedicated contact n. 10	Auxiliary relais 2
501.10	Dedicated contact n. 11	Auxiliary relais 3
501.11	Dedicated contact n. 12	Auxiliary relais 4
501.12	Dedicated contact n. 13	Auxiliary relais 5
501.13	Dedicated contact n. 14	Auxiliary relais 6
501.14	Dedicated contact n. 15	Auxiliary relais 7
501.15	Dedicated contact n. 16	Auxiliary relais 8
502.0	Auxiliary contact n. 01	
502.1	Auxiliary contact n. 02	
502.2	Auxiliary contact n. 03	
502.3	Auxiliary contact n. 04	
502.4	Auxiliary contact n. 05	
502.5	Auxiliary contact n. 06	
502.6	Auxiliary contact n. 07	
502.7	Auxiliary contact n. 08	
502.8	Auxiliary contact n. 09	
502.9	Auxiliary contact n. 10	
502.10	Auxiliary contact n. 11	
502.11	Auxiliary contact n. 12	
502.12	Auxiliary contact n. 13	
502.13	Auxiliary contact n. 14	
502.14	Auxiliary contact n. 15	
502.15	Auxiliary contact n. 16	

VACUUM CHAMBERS ONLY		
ADDRESS	**MEANING**	**NOTES**
503.0	**Vacuum command n. 01**	
503.1	**Vacuum command n. 02**	
503.2	**Vacuum command n. 03**	
503.3	**Vacuum command n. 04**	
503.4	**Vacuum command n. 05**	
503.5	**Vacuum command n. 06**	
503.6	**Vacuum command n. 07**	
503.7	**Vacuum command n. 08**	
503.8	**Vacuum command n. 09**	
503.9	**Vacuum command n. 10**	
503.10	**Vacuum command n. 11**	
503.11	**Vacuum command n. 12**	
503.12	**Vacuum command n. 13**	
503.13	**Vacuum command n. 14**	
503.14	**Vacuum command n. 15**	
503.15	**Vacuum command n. 16**	

CHANNEL NUMBER	ADDRESS	MEANING	NOTES
0	**504-505**	**User final setpoinrt**	**TEMPERATURE**
0	**506-507**	**Gradient**	**TEMPERATURE**
1	**508-509**	**User final setpoinrt**	**RELATIVE HUMIDITY**
1	**510-511**	**Gradient**	**RELATIVE HUMIDITY**
2	**512-513**	**User final setpoinrt**	**NOT USED**
2	**514-515**	**Gradient**	**NOT USED**
3	**516-517**	**User final setpoinrt**	**NOT USED**
3	**518-518**	**Gradient**	**NOT USED**
4	**520-521**	**User final setpoinrt**	**NOT USED**
4	**522-523**	**Gradient**	**NOT USED**
5	**524-525**	**User final setpoinrt**	**NOT USED**
5	**526-527**	**Gradient**	**NOT USED**
6	**528-529**	**User final setpoinrt**	**NOT USED**
6	**530-531**	**Gradient**	**NOT USED**
7	**532-533**	**User final setpoinrt**	**NOT USED**
7	**534-535**	**Gradient**	**NOT USED**

The setpoint and slope values can be done via the following functions

```
void conv_float_pc_s7(float pc_real, char * plc_real)
{
  plc_real[0] = * ((char *) &pc_real + 3);
  plc_real[1] = * ((char *) &pc_real + 2);
  plc_real[2] = * ((char *) &pc_real + 1);
  plc_real[3] = * ((char *) &pc_real + 0);
}
```

This function can be used to write words at address 500 up to 502.

```
short int conv_word_mplc_to_word_pc(unsigned char * buf)
{
  return ((short int) ((* (buf + 0)) << 8) + ((* (buf + 1)) << 0));
}
```

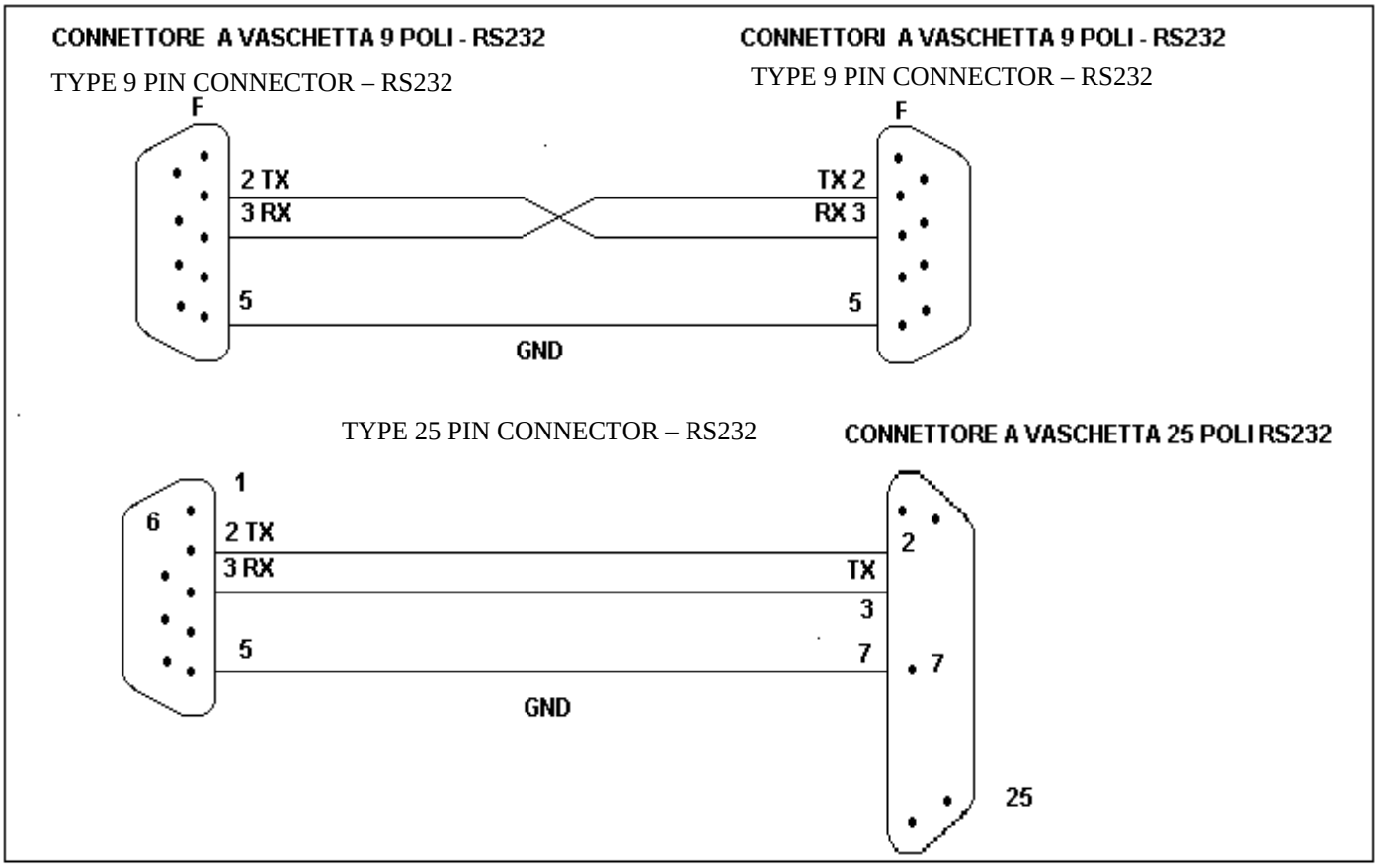
NOTES

- The ALARM RESET operation is made setting to 1 the corresponding bit. The SIEMENS S7 will provide in automatic mode to force it to 0 after few seconds.
- The bit **LOCAL/REMOTE** bit is reserved and not used at the moment. It must be forced to 0. Same for all the bits not used for the specific application.
- In order to work, a regulator needs both the setpoint value and the enable bit in ON status (bit 500.8 and following). For instance, the temperature controller (Channel 0) will be in stand-by and will not execute any regulation in case that the enabling bit (bit 500.8) is equal to 0.

APPENDIX A: CONNECTION OF A SERIAL CABLE

In the following are described the connections required to have a serial connection between chamber and PC.
In the following drawing the female connector on the left must be connected to the RS232/485 connector of the chamber.

Fig. 8 Serial cable



APPENDIX B: COMMUNICATION PARAMETERS

The following parameters must be used to configure the serial port of the PC in order to communicate with the chamber controller. These parameters cannot be changed.

NAME	VALUE
BAUD RATE	9600
PARITY	NONE
STOP BITS	1
DATA LENGHT	8