MANUAL

Cold Storage Room Alarm

MADE IN EUROPE

Modbus over RS485

INDICE

1 INTRODUCTION

We sincerely appreciate your trust in choosing our products. We are committed to offering you high quality solutions that meet your needs and exceed your expectations.

This manual describes in detail the operation of our alarm system for cold rooms with Modbus communication. The equipment is designed to emit audible and visual alarms in case the door of the chamber remains open, thus ensuring that optimal conditions are maintained.

In addition, the system includes an NTC probe to measure the internal temperature of the chamber and, optionally, you can use an AM2302 probe to monitor the temperature and humidity of the external environment.

The remote communication functionality allows this system to be integrated with a PLC (Programmable Logic Controller) or a computer. This makes it possible to record and store data on the status of the chamber and the alarms generated, facilitating later analysis. In this way, customized reports can be generated that comply with legal requirements and help optimize the control of your facilities.

General characteristics:

Parameter	Specification
Construction	90x90 mm watertight box, compact and robust design, ideal for mounting on the cold room door.
Power supply	12-24V CC.
Visual Alert	High intensity front LED, automatically activated when opening the door.
Sound Alert	105 dB buzzer with intermittent beeps after a programmable time (5 minutes by default).
Integrated Sensors	• Magnetic sensor: Detects the door status (open or closed). • NTC 10K temperature probe: Accurate internal temperature measurement (-40 to 80ºC). • Optional AM2302 sensor: Internal temperature and humidity monitoring (-40 to 80ºC and 0 to 99.9%).
Programmable Configuration	Adjustable alert time to tailor the interval before the buzzer and LED activate.
Communication	Modbus port for remote configuration and monitoring.
Applications	Monitoring doors and temperatures in cold storage rooms to ensure optimal conditions.

It is common for different establishments to have the chamber partially open or for there to be a fault in the machinery that breaks the cold chain.

The equipment has the following applications:

- **Food storage in hotels and restaurants:**

  Temperature monitoring in cold storage rooms in restaurants, hotels, cafes and catering services to ensure proper food preservation.

- **Supermarkets and grocery stores:**

  Control of refrigerated chambers used to store perishable products such as meat, fish, dairy products and frozen foods.

- **Cold chain logistics warehouses:**

  Monitoring distribution centres for food or temperature-sensitive products, avoiding possible failures in the cold chain.

- **Food industries:**

  Monitoring of cold storage rooms in factories that process, package or store food products.

- **Pharmacies and laboratories:**

  Supervision of chambers where medicines, vaccines or biological samples are stored that require specific temperature and humidity conditions.

- **Hospitals and health centres:**

  Use in storage chambers for blood banks, medications or temperature-sensitive medical products.

- **Chemical industry:**

  Monitoring of refrigerated chambers where chemical products that require specific temperature conditions for their conservation are stored.

- **Agricultural sector:**

  Control of cold storage chambers for the preservation of fruits, vegetables and flowers.

- **Meat maturation chambers:**

  Monitoring internal conditions in chambers designed for meat aging, helping to maintain optimal temperatures for the process.

- **Wine cellars and wineries:**

  Temperature and humidity monitoring in chambers for the conservation of wines and other sensitive products.

- **Storage spaces in large areas:**

  Monitoring refrigerated areas in shopping centres and large warehouses to keep perishable products in good condition.

- **Educational centres and residences:**

  Use in kitchens and food storage areas to ensure proper preservation.

- **Research and development areas:**

  Monitoring of experimental chambers for product testing requiring precise temperature and humidity control.

The measured data can be read remotely using an RS485 bus with Modbus RTU protocol. RS485, an inexpensive and simple-to-trace communication bus, allows data transmission at relatively high speeds with guaranteed distances of up to 1.300 meters without the need for repeaters, showing great immunity to interference.

The ModBus RTU protocol, used in this family of devices, is open, standard and universal, supported by numerous development environments, PLCs, microcontrollers and SCADAs. It conforms to the standards published by ModBus.org.

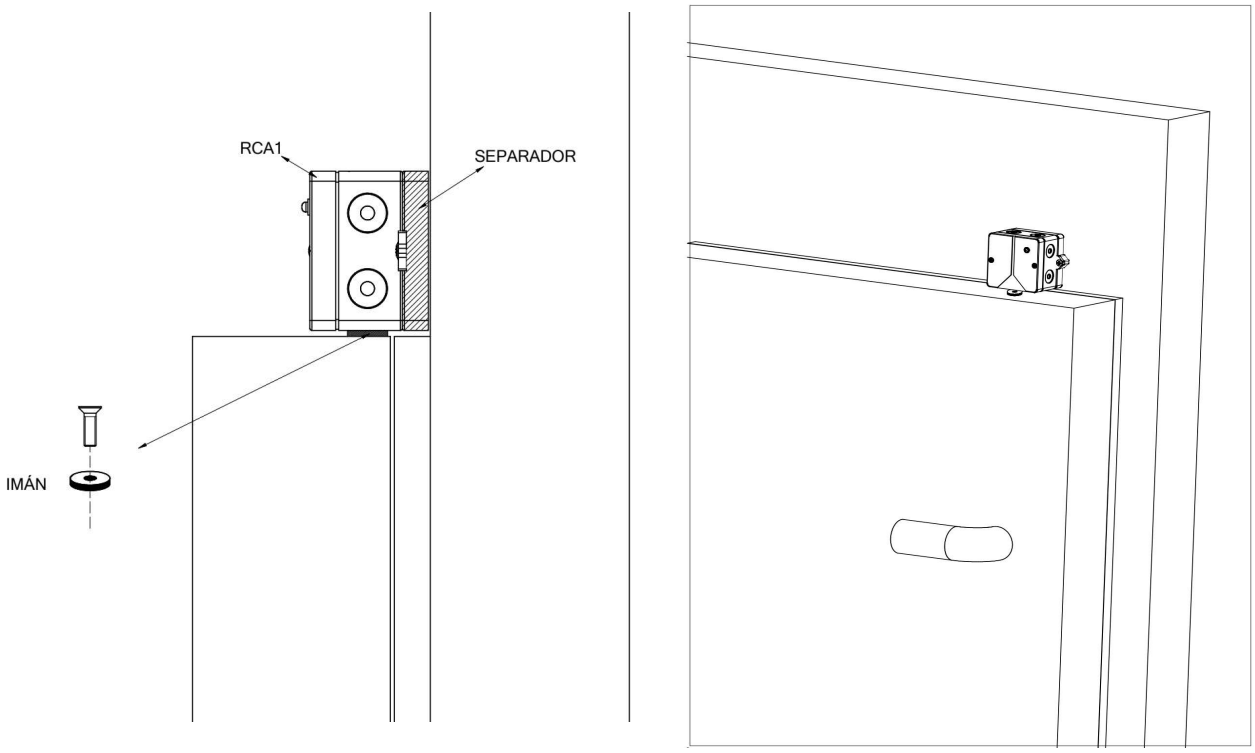
The communication bus is properly protected against overvoltages and static overcurrents by devices such as TBU, TVS and GDT, offering level 4 protection.

	Test	Normas
ESD	15 kV	IEC 61000-4-2 electrostatic discharge.
EFT	2 kV	IEC 61000-4-4 electrical fast transients.
Surge	6 kV	IEC 61000-4-5 surge immunity.

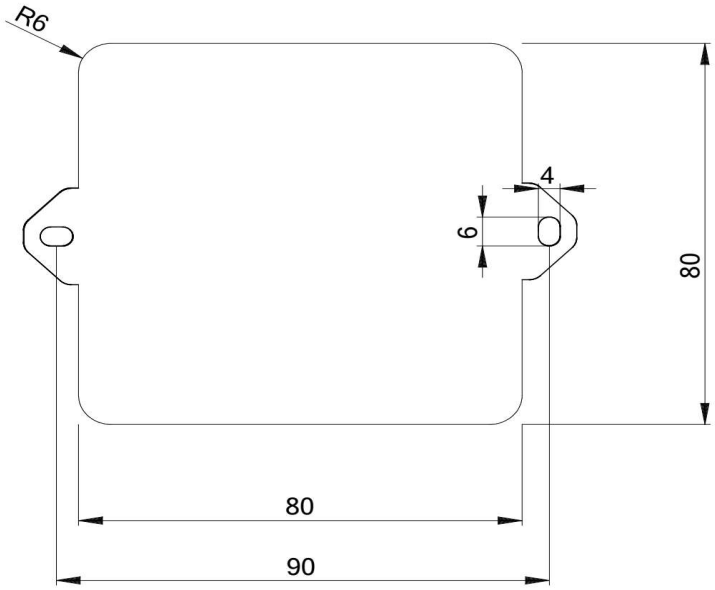
The equipment is integrated into an IP65 box measuring 80x80x50 mm that has ears for screwing to a frame or conduit.

2 ASSEMBLY AND CONNECTION

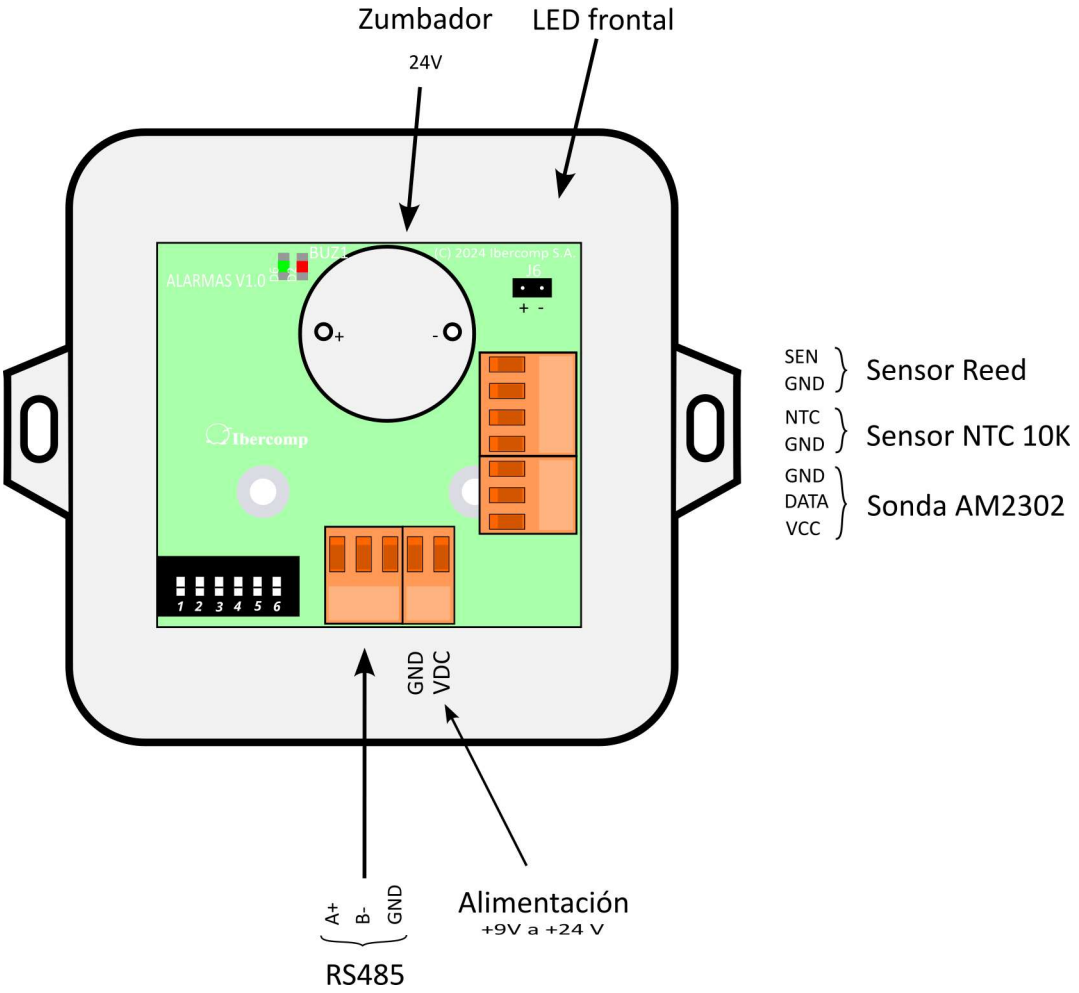
The device can ideally be mounted on the door by screwing a 20 mm neodymium magnet onto it using a self-drilling screw. The magnet does not need to be perfectly centred. If the distance is large, several magnets can be stacked.



In some installations it will be necessary to build a separator with the following measurements:



Opening the cover of the equipment, taking care not to pull the LED and buzzer cables, you will be able to see the following connections inside:



The power supply of about 10mA should be a direct voltage between 10V and 30V, the most common being 24V, which is the one used by all our industrial equipment. Ideally, we recommend our DIN rail power supplies.

The front LED will go in its connector, red wire positive, black wire negative.

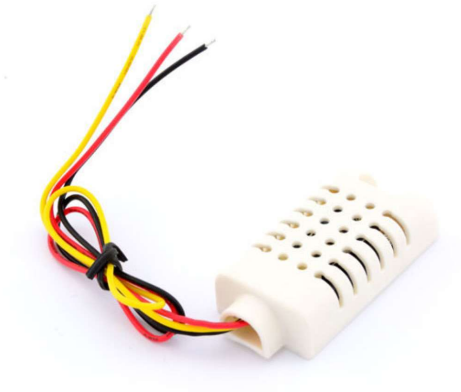
On the BUZ1 connector, you can mount a piezoelectric buzzer, the one we included can give up to 105dB, but we feed it at 5V. If you want to have a higher volume you can cut the red wire and connect it directly to the VDC power supply.

The reed sensor is integrated into the housing, it is simply a contact that is closed when the chamber door is closed. It can be mounted if any other alternative is desired.

The 10K NTC probe to be mounted is the TEWA NTC TT02-10KC3-1D-1000-20-5 Water Proof reference probe. Other 10K NTC probes may add errors to the measurement, errors that can

be corrected by means of a program. The probe should not be more than 10m from the equipment.

Optionally, we can include a complementary temperature/humidity probe type AM2302. This probe can be mounted outside the chamber or inside. Outside it can be used to see the effect that the door has on the outside temperature/humidity. This probe should not be more than 5 meters from the equipment.



The RS485 connector contains the lines necessary to communicate remotely from a PC or from a microcontroller or programmable logic controller. In both cases, the RS45 lines are connected under the following rules:

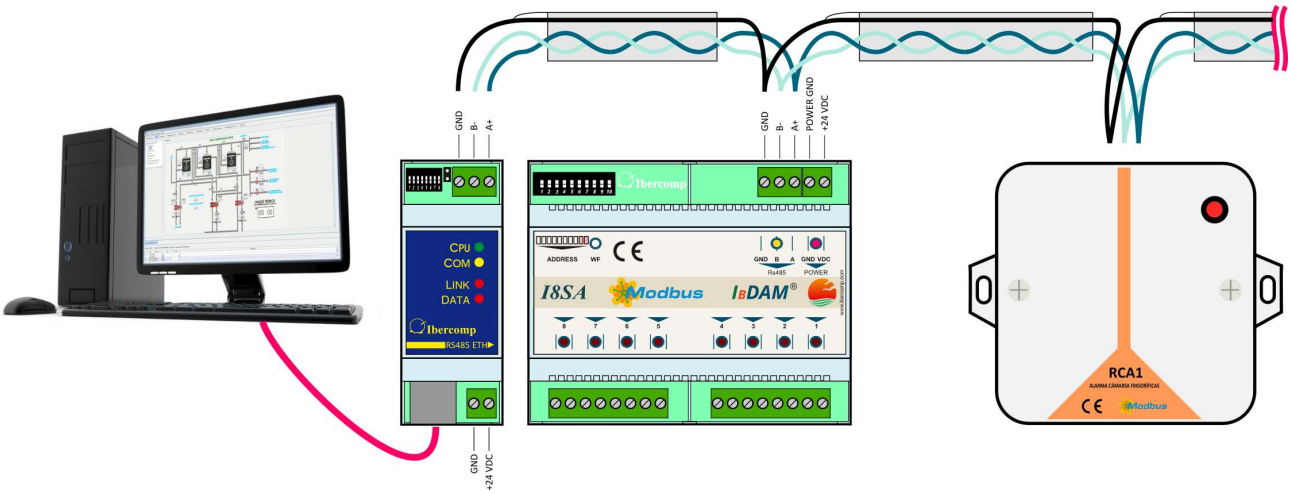
a) All A signals must be connected together, as must all B signals. All grounds must be properly connected (for more information, read the chapter “RS485 installation tips”).

b) RS485 lines should have a maximum length of 1.300 meters between the first and last equipment (approximately 4.000 feet).

c) RS485 lines should ideally not have branches.

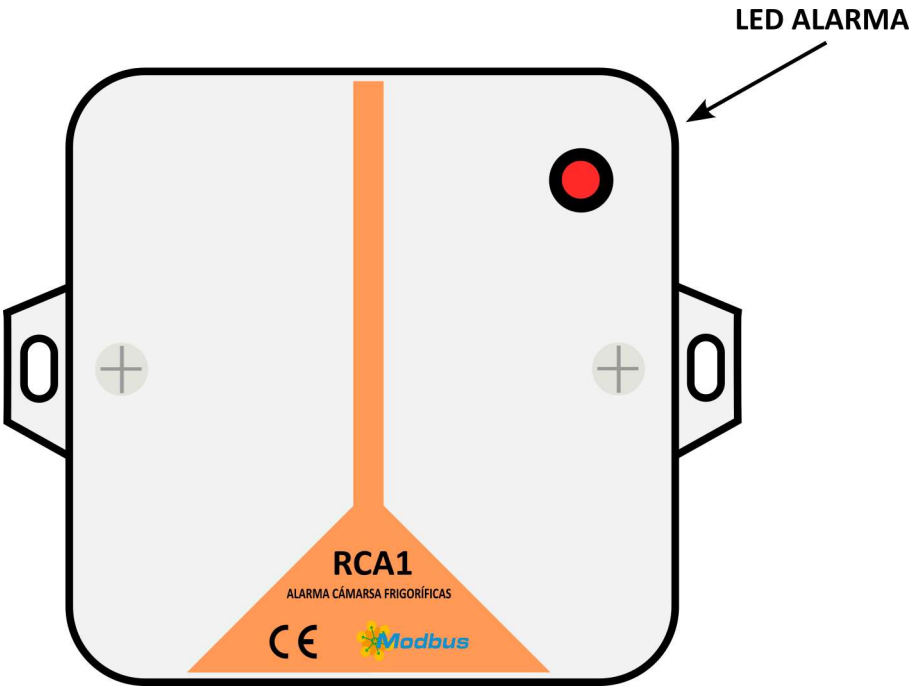
NOTE: Please note that we use the nomenclature used by transceiver manufacturers, i.e. A has the positive signal and B the negative signal. In some PLCs they use the opposite nomenclature. Our A should always receive the positive and B the negative, regardless of how they are called in the PLC.

In the following illustration we show a line where our Ethernet to RS485 converter is used to access several devices among which there is an RCA1 cold room alarm.


GND
B-
A+
CPU
COM
LINK
DATA
Ibercomp
RS485 ETH
GND
+24 VDC
GND
B-
A+
POWER GND
+24 VDC
ADDRESS
WF
GND B A
Rs485
GND VDC
POWER
I8SA
Modbus
IBDAM
RCA1

3 FRONT OF THE EQUIPMENT

The front has a clearly visible LED indicator. They have the following meaning:



ALARM LED	This high-brightness red LED is activated when the door is open, and starts flashing intermittently when the door is open for more than a preset time.

4 DESCRIPTION OF THE COMMUNICATIONS PROTOCOL

On the 485 physical network, it is necessary to implement some communication standards that allow access to certain devices. These standards are called “communication protocol”.

It is difficult to define the best protocol and proof of this is the diversity of protocols that coexist in the market, the best known being ModBus, PROFIBUS, CHIPBUS, etc. All of them have some advantage over their competitors and the choice of one or the other will depend on the requirements of the application.

Within a communication protocol, data is transmitted in information packets, which usually have a header followed by some data and a tail.

Most protocols are complex and often use very large data packets to transmit very little information.

We have chosen to equip this family of modules with the ModBus RTU protocol, adjusting as mentioned above to the specifications given by ModBus.org.

The reasons for choosing this protocol are:

- It has a widely standardized protocol that is used by existing Scada programs, PLCs, etc.
- A protocol that is easy to implement and open, that is, without complications, secrets or “royalties”.
- It can use a RS485 physical network.

ModBus devices have registers inside them that can be similar to variables. A register can be physical, for example the state of a relay, or logical, for example the value of a timer.

Four types of records are defined:

Discrete input

These are 1-bit read-only registers. They are usually associated with digital inputs.

Coil

1-bit output registers. They can be read and written. They are usually associated with digital inputs.

Input Register

16-bit read register. They cannot be written and their function is to serve as support for analog inputs.

Holding Register

These are 16-bit output registers, which can be read and written. They are usually associated with analog inputs, although their meaning is defined by the module manufacturer.

In our case, as we indicated before, the first 16 “holding registers” are common to all devices and are used to identify the card, as well as to define some behaviours.

In our modules the four types of registers have different maps, that is, the value of the “holding register” 7 does not match that of the “input register” 7. They are two different registers.

The ModBus protocol is message-oriented and master-slave. In the RS485 network there is only one device that acts as master and numerous devices that act as slaves.

There cannot be more than one master device at the same time because the protocol knows nothing about resolving collisions. At a logical level, using our ModBus service, a system could be implemented where several SCADA-type master applications could access a module at the same time. That is beyond the scope of this manual.

The master, usually a computer, automaton or microcontroller, maintains an imperative dialogue with the slaves. To do this, it sends them messages and they interpret them, obey them and respond. Each time a slave is asked a question, it responds.

There are two types of messages within ModBus: ASCII messages and binary messages. The latter are also known as RTU and are the ones implemented by our equipment.

Messages have the following form:

Address	Function Code	D A T A	CRC

The master sends a message and the slave returns another message, respecting the same format. The meaning of each of the parts is as follows:

Address This is the address of the slave device to which the message has been addressed. When the slave responds, it indicates its origin in this field.

Please note that ModBus only accepts addresses 1 to 247. Addresses 248 to 255 are reserved and should not be used. All devices respond to address 254, but this should only be used if there is only one device on the network, as all responses will collide.

Function Code This is what the slave module is expected to do. The function code, as will be described later, basically serves to tell the slave that a register is to be read or written to.
In the response this function code may also contain an indication that an error has occurred.

Data It is a number of bytes that contains additional data to the function code, for example, if you want to write about relays, the data must contain the desired state for the relays.

CRC is a value that allows us to know whether or not the content of the message has been altered along the way. It is a 16-bit polynomial function that is shown at the end of this manual.

In order to distinguish the beginning and end of a packet, a period of time is inserted between the packets during which no device is transmitting anything. This period must be at least 3.5 characters, i.e. 35 bits.

If you are working at 9600 baud, this time will be 35/9600 baud, i.e. about 3.6ms. In order to avoid times being excessively short when the speed increases, the minimum time is set at 1.7ms.

To avoid confusion, there should not be more than 1.5 characters between characters in the same message. To avoid very long times, a maximum time of 759us is set for speeds higher than 19200 baud.

In practice, meeting this requirement is simple. The programmer only needs to ensure that the characters in a message are transmitted one after the other, without pauses or interruptions. The solution to this is to load the entire message into a variable or array and then send the whole message to the RS485 port.

On the other hand, if you send the address byte first, then the function code, then the data, and finally calculate the CRC, it is guaranteed not to work. Calculating the CRC may take some time, and among the many instructions the operating system may perform a context switch (assign the CPU to another process/program).

The ModBus protocol has a large number of function codes, not all of which are necessary. We have not implemented all of them. The ones implemented in our cards are the following:

0x01 Read Coils

Allows reading the status of the “coils”.

0x02 Read Discrete Inputs

Allows reading the status of the “discrete inputs”.

0x03 Read Holding Registers

Allows reading of the computer's 16-bit registers.

0x04 Read Input Register

Allows reading of the computer's 16-bit registers.

0x05 Write Single Coil

Allows you to define the status of a “coil”.

0x06 Write Single Holding Register

Allows you to define the status of a holding register.

0x0F Write Multiple Coils

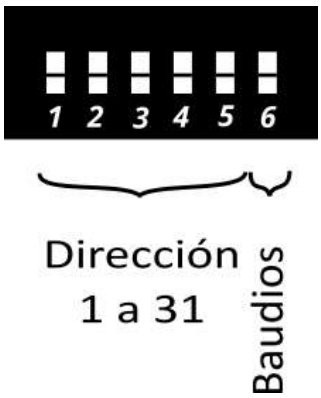
Allows you to write multiple/all relays via a single transaction.

0x10 Write Multiple Holding Register

Allows multiple/all records to be written via a transaction.

5 CONFIGURATION SWITCHES

In the WS1 equipment circuit there is an array of 6 switches that serve to configure the modbus id and the baud rate:



Switches	Description
1 TO 5	Selects the device's address within the network in binary, with devices 1 to 31 being valid.
6	10/9 Baud rate. 0 9600 baud 1 19200 baud

If you set all switches to 1, when you power it up it will acquire the factory values.

When powering it will take a few seconds, and when it is reprogrammed both LEDs on the board will flash rapidly.

Once restarted, you must set the desired address, remove power and reconnect it. If you set all switches to OFF, it will take the values indicated by the software.

In this table a state 1 is synonymous with an ON state and a state 0 is synonymous with an OFF state.

If the switches are set to ON after the board has been started, the red LED will start flashing, which means the board can be accessed using address 247. Using the Modbus ID program you can modify the speed and Modbus user ID.

6 REGISTERS

DISCRETE INPUTS TABLE

This equipment does not have "discrete inputs".

TABLA COILS

nº	Type	Description
0		If set to 1, access to some holding records is granted. If set to 0, writing to the records is unlocked.
1	RO	It is set to 1 to indicate that all switches are ON. In this mode the board will respond to address 247, but at least one switch must be turned down, otherwise if the power is cut and then comes back on it will reset to factory settings and remain locked.
2	RO	It is set to 1 to indicate that there is an error in the temperature probe, that is, it is either missing, short-circuited, or outside the range -30ºC to 140ºC. NOTE: It can measure correctly between -40ºC and -30ºC even if the error is activated.
3	RO	Set to 1 to indicate that it does not detect AM2302 probe.
4	RO	It is the status of the door sensor, 1 indicates that it is open and 0 that it is closed.

REGISTER INPUTS TABLE

This equipment does not have "register inputs".

HOLDING REGISTERS TABLE

The modules have a number of holding registers that allow us to know some of the module's characteristics. Some of these configuration registers are stored in an E2PROM memory.

The first sixteen registers are the "holding registers" common to all boards. They allow you to know what type of module it is, firmware version, number of registers, ...

nº	Type	Description
0	EE	Number of module starts. Each time the module starts or is restarted, the value of this register is increased by one unit. This allows us to determine whether a module is restarting abnormally.

nº	Type	Description
1	RO	Module version. The version is encoded as version*10 + subversion. Version 1.0 is represented as 10.
2	RO	Module code, in this case: 224 RCA1
3	EE	User baud rate, i.e. the speed the equipment has if it is started with all switches set to zero. By default 9600 baud You can set standard values such as 4800, 9600, 19200, …, 57600. For example, if you set 19200 baud, the system will calculate the closest real speed that the processor can have, in this case 19230. A small difference in speed is of no importance. In order to modify it, coil[0] must have a value of 0 and holding register 15 must have a value of 3.
4	EE	R E S E R V E D
5	EE	Modbus address that the device has if it is started with the switches set to zero. By default id=1. In order to modify it, coil[0] must have a value of 0 and holding register 15 must have a value of 3.
6	RO	Number of “discrete input” type registers.
7	RO	Number of “coil” type registers.
8	RO	Number of “input register” type registers.
9	RO	Number of holding registers. Keep in mind that the first 16 will always be common to all modules and that the module's own registers begin at register 16.
10	EE	R E S E R V E D
11	EE	R E S E R V E D
12	RO	Displays the status of the switches. Since all boards respond to address 254, connecting one to the line can check if the switches are working correctly. It can also be used in conjunction with registers 2 and 5 to deduce whether the module is installed in the correct place.
13		R E S E R V E D
14		R E S E R V E D
15	MM	By writing the appropriate values sequentially, the module can be reset remotely. This is equivalent to all intents and purposes to unplugging the power and plugging it back in. Before you begin, you should read register 15 and make sure it has a value of 0. If it does not, write any value other than 0x55 or 0xAA (85 or 170). Once written, check again.

nº	Type	Description
		To start the sequence, you will write the value 0x55 (85), then write a 0xAA (170), and finally a 0xAA (85). Reading the register will give you the machine reset value: 0, 1, and 2. The third write will not be answered by a communications failure. You will know that the computer has been reset when register 0 is incremented by one. If having the register a value of 0 the value 0xA5A5 (42405) is written, this register will take the value 3 and it will be possible to modify registers 3 and 5 (if in addition coil[0]=0).
16	RO	NTC probe temperature in tenths of a degree. It is a 16-bit signed integer with 2's complement.
17	RO	Ohm value measured at the input. This may be useful if a different type of probe is mounted. The equipment that makes the reading in ohms must calculate the actual temperature.
18	RO	Value of the analogue-digital converter. It should be noted that it gives an analogue value obtained from a resistive divider where the upper resistance is 10K 1% and the lower one is the probe. In a slightly more complex but more precise way, the real temperature can be inferred from this value.
19	EE	In some installations, the temperature measured by the probe may not exactly match the actual temperature of the chamber. This may be due, for example, to the location of the probe. An offset can be added that raises or lowers the entire slope, or that changes its inclination. This record contains the offset in tenths of a degree when the temperature is 0ºC. In order to modify this record, coil[0] must have a value of 0.
20	EE	This register contains the offset when the temperature is 100ºC. Like the previous register, it is a 16-bit number with a 2's complement sign. If the same amount is added or subtracted from both offset registers, the entire slope will shift while maintaining its inclination. In order to modify this register, coil[0] must have a value of 0.
21	RO	AM2302 probe temperature in tenths of a degree. 16-bit number with 2's complement sign. The measurement will only be valid if there is no error in the reading.

nº	Type	Description
22	RO	Humidity of the AM2302 probe x 10 in %. The value ranges from 0 to 999. The measurement will only be valid if there is no error in the reading.
23	RO	High part of a 32-bit counter that tells us how many seconds the door has been open.
24	RO	Lower part of a 32-bit counter that tells us how many seconds the door has been open.
25	EE	Time that the door must be open in seconds for the alarm to go off.

LEGEND:

EE Indicates that these are persistent records stored in the E2PROM memory. These records can be rewritten, but you must keep in mind that after 1.000.000 writes their content may not be correct.

MM Indicates that the record is stored in memory. If the power is cut off, it is lost. You can write the record as many times as necessary.

RO Indicates that this is a read-only register. Writes to this register will be ignored.

7 DISASSEMBLY OF THE MODULE

To open the unit, you must remove the screws on the cover and carefully remove it so as not to pull on the front LED and buzzer cables. Once opened, you will have access to the configuration switches described in section 5.

You will also have access to the various connection strips.

8 RS485 INSTALLATION TIPS

Setting up an RS485 network is not a delicate or complicated task, especially when it comes to low-speed networks. In practice, it is not at all demanding, either in terms of the typology or the type of cable.

Now, all RS485 transceivers, including those from competing equipment, are very susceptible to power surges, so if care is not taken at some points the installation will not only not work but will almost certainly break down.

The transceivers used are 65HVD3082, which support 128 nodes and are robust against static discharges. This integrated circuit can be replaced by other compatible ones with other characteristics.

These integrated transceivers are always in a socket.

In a poorly executed installation, a lightning strike a few hundred metres away will be enough to damage it and put it out of service.

It is also necessary to understand the concept of “Half Duplex” transmission and that if two wires are needed, it does not mean that one is for transmission and the other for reception.

Below we describe some ideas or recipes that may be useful to you in making a good installation.

8.1 Flow control using RTS

The RS485 protocol is “half duplex”, which means that the same lines are used for data transmission and reception. So, in addition to transmitting or receiving data, flow control is necessary.

In systems where this is not taken into account or is not managed correctly, the following is typical:

a) Module combinations that are incompatible with each other, so you have to test which ones can be mixed or not.

b) The system works correctly on the workshop or laboratory table but it causes problems in the actual installation.

If we do not manage the RTS line it is likely that by chance all the equipment is in transmission mode, transmitting a signal 1. If any one transmits a signal 0, it is possible that, over

very short distances, the other equipment receives this 0. In any case, it is necessary to manage this line correctly.

Generally, in computers that have RS485 ports (industrial PCs), this flow control is carried out using the RTS line. This line must be set to high level before starting the transmission of information and set to low level again to wait for the reception of data.

This task seems simple but it is not for the following reason: when you send a character from a program to the serial port, it is actually being sent to a buffer controlled by the operating system, so it does not know in advance when the transmission begins or when it ends.

There are several solutions to this problem, the first solution is to provide the operating system with a “driver” that controls the RTS line at a low level or, if the language allows it, enable this transmission protocol. It should not be confused with the RTS/CTS flow control, it has nothing to do with it and is not useful for managing an RS485 NETWORK.

The second solution is to use an RS232c serial port and connect an RS232c ←→ RS485 converter with automatic flow control to it. A second problem can arise here; there are some converters that add a delay to this flow control. If this delay is greater than 3.5 characters we may lose the start of the response packet.

Normally no converter has such long delays, but if repeaters and different RS485-fiber converters are inserted along the way, this problem may arise.

You should keep in mind that unlike a “full duplex” serial port (RS232 or RS422), a “half duplex” port (RS485) always receives an Echo of the transmitted data. So when you ask a peripheral card something, you receive the question followed by the answer.

There are some RS232c ←→ RS485 converters on the market that have an Echo cancellation option, but it is better for the software to use this Echo to check that the line is working correctly. For example, a short between lines A and B would prevent this ECHO from occurring and in a “MultiMaster” system it could be used to detect message collision (although it is not a good idea in ModBus).

8.2 Termination of lines

In some cases it is necessary to add terminating resistors at the ends of the RS485 lines to adapt the line impedance to the impedances of the transceivers. When the impedances do not match, the transmitted signal is not adequately absorbed and part of it is reflected back into the line.

By adding them we eliminate this reflection but we increase the complexity of the installation and the consumption of the transceivers. When we use the RS485 line at low speed, as is the case with our modules, we do not gain any advantage by installing these resistors since the reflection does not affect communications.

If you use an RS232c ←→ RS485 converter that is powered by the serial port itself, you should not use this type of termination. If you are going to use terminating resistors, you should install only one at each end of the line and calculate them so that the line impedance is 120 Ohms.

In the particular case of carrying out an installation with our modules, we do not recommend the installation of said resistors, although from experience you can connect them and they can be used to verify proper operation.

If it does and the system works without errors, then all the equipment is working correctly. Otherwise, some transceiver or the line has a problem. Normally, without the resistors, communications will work.

8.3 Polarization of the lines

When no equipment is transmitting on the RS485 line, it is in high impedance, that is, there is no voltage present. In this state, it is possible for a current to be easily induced, which can be confused with the reception of data.

Although it is almost never necessary, you can add resistors connecting line B to +5V and line A to GND so that there is at least 200 mV between A and B. The value of these resistors depends on the number of devices installed and the installation. This will ensure a defined state during silent periods.

These resistors can be mounted anywhere in the network, or even distributed. If these resistors are too large (or not mounted) noise immunity is reduced. On the other hand, if the resistors are too small, the transceivers have a harder time transmitting (= shorter distance and fewer connected devices).

Some RS232c ←→ RS485 converters do not support this type of resistors.

8.4 Proper use of the cable

In an RS485 installation a minimum of 3 lines are required, 2 for data transmission/reception and one to unify the grounds (GND). It is important to use the appropriate cable. For data transmission a twisted cable should be used, preferably shielded.

Not connecting the grounds of the different devices is a guarantee that problems will arise over time, even in modules or devices in which the RS485 transceiver is optocoupled and the manufacturer indicates that two lines are sufficient.

Although it works, it is not advisable to use the cable shield to connect the grounds. We recommend using a second pair for this purpose and if you wish you can use a third pair for the power supply (+24 DC) of the equipment.

The reason is that the resistivity of the cable shielding is higher than that of the data wires and therefore significant voltage differences may appear between the data and ground lines of two distant devices.

We therefore recommend using a shielded cable with 3 twisted pairs, one data pair, one ground pair and one power pair. The shield must be connected to a good ground at only one point.

If budget is not a problem, it is better to use a cable with polyethylene insulation instead of PVC as it has less signal attenuation.

A suitable cable is the Category 5 STP Ethernet cable, it is cheap, easy to locate and far exceeds the RS485 specifications.

8.5 Proper cable installation

An RS485 installation must be laid out in a line, that is, it must have a start and an end. Equipment can be connected along the line. The piece of cable that connects the equipment to the line must be very short.

It is not correct to carry out an installation in which the line is divided into several (star configuration), since there may be out-of-phase reflections that confuse communication.

The maximum length that the line can have is 1.300 meters from end to end. In practice, although it is not recommended, using low speeds (e.g.: 9600 baud) works with a star topology installation and lengths greater than 1.300 meters.

We have seen an installation with 12km of RS485 cable, 9600 baud, without repeaters and with a topology that does not comply with any recommendation, working without problems. It does not seem correct to us, but in this specific case, unexpectedly it works.

8.6 Overvoltage protection

As we have already indicated, RS485 transceivers are very delicate and sensitive to overvoltages and their deterioration is usually responsible for almost all the breakdowns that occur in RS485 installations.

These overvoltages are caused in most cases by poorly designed installations, such as running the RS485 lines with inadequate cables along with high voltage cables. In other cases, they cannot be avoided due to a budget that prevents separating the lines or they are due to phenomena such as a nearby lightning strike.

Our equipment includes protection varistors, which will provide sufficient protection for installations that are to be carried out inside buildings or that are equipped with a good ground connection.

However, in installations with long cable lengths where there is no ground connection, which connect two distant buildings or which are in the open countryside, if no protection is taken we can be sure that half of the transceivers will break at the first opportunity, leaving the system inoperative.

We can take several measures:

Make sure you have a good ground that is unified throughout the installation, even if this means installing a separate cable with a generous section.

Install a surge protector with several levels of protection in cascade (fuses, transducers, gas dischargers and varistors). This should be installed in the middle of the line, for example in an installation that connects two buildings it should be mounted at the entrance and exit of the buildings.

Install optoisolated repeaters at some midpoints of the network.

Insert 100 mA fast fuses into the lines. It is preferable for these to blow than for the transceivers to blow.

9 WARRANTY CONDITIONS

We have put the utmost care and attention into the design of this new family of RS485 modules and they are designed to operate 24 hours a day without any problems.

Ibercomp SA undertakes to provide a 3-year warranty against any manufacturing defect from the invoice date on the SW1 wind sensor.

Any breakage that has occurred physically, i.e., a direct result of poor installation, incorrect voltage supply or overvoltages induced through the electrical network or data lines, is exempt from warranty.

The warranty includes all the elements that must be replaced and the labour required to carry out the repair. If the equipment cannot be repaired under warranty, it will be replaced with a new identical or superior replacement model.

The warranty does not include transport costs to our workshops.

Ibercomp SA, as the equipment manufacturer, is not responsible for:

1) Any damage or harm that may be caused to the equipment as a result of a malfunction or loss of data.

2) Any errors or defects that may be found in the software or examples associated with it. These are under continuous development and may be updated free of charge whenever a new version is available.

The software or examples we provide are free of charge, although Ibercomp owns the rights to them. You are free to use them, use third-party software, or develop your own software.

Ibercomp S.A. guarantees the acceptance of the return of the equipment up to 30 days after the date of purchase on the condition that the equipment and accessories as well as its packaging are in their original condition.

If the purchase was made through a store or distributor, the return of the package must be made through the same.

10 CRC CALCULATION

C/C++

```
unsigned short CRC16 ( puchMsg, usDataLen ) /* The function returns the
                                               CRC as a unsigned short type */
unsigned char *puchMsg ;                    /* message to calculate CRC upon */
unsigned short usDataLen ;                  /* quantity of bytes in message */
{
  unsigned char uchCRCHi = 0xFF ;           /* high byte of CRC initialized */
  unsigned char uchCRCLo = 0xFF ;           /* low byte of CRC initialized */
  unsigned uIndex ;                         /* will index into CRC lookup table */
  while (usDataLen--)                       /* pass through message buffer */
  {
        uIndex = uchCRCLo ^ *puchMsg++ ;    /* calculate the CRC */
        uchCRCLo = uchCRCHi ^ auchCRCHi[uIndex] ;
        uchCRCHi = auchCRCLo[uIndex] ;
  }
  return (uchCRCHi << 8 | uchCRCLo) ;
}
```

High-Order Byte Table

```
/* Table of CRC values for high–order byte */
static unsigned char auchCRCHi[] = {
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40
};
```

Low-Order Byte Table

```
/* Table of CRC values for low–order byte */
static char auchCRCLo[] = {
0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2,
0xC6, 0x06, 0x07, 0xC7, 0x05, 0xC5, 0xC4, 0x04,
```

```
0xCC, 0x0C, 0x0D, 0xCD, 0x0F, 0xCF, 0xCE, 0x0E,
0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09, 0x08, 0xC8,
0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A,
0x1E, 0xDE, 0xDF, 0x1F, 0xDD, 0x1D, 0x1C, 0xDC,
0x14, 0xD4, 0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6,
0xD2, 0x12, 0x13, 0xD3, 0x11, 0xD1, 0xD0, 0x10,
0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3, 0xF2, 0x32,
0x36, 0xF6, 0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4,
0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE,
0xFA, 0x3A, 0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38,
0x28, 0xE8, 0xE9, 0x29, 0xEB, 0x2B, 0x2A, 0xEA,
0xEE, 0x2E, 0x2F, 0xEF, 0x2D, 0xED, 0xEC, 0x2C,
0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0,
0xA0, 0x60, 0x61, 0xA1, 0x63, 0xA3, 0xA2, 0x62,
0x66, 0xA6, 0xA7, 0x67, 0xA5, 0x65, 0x64, 0xA4,
0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F, 0x6E, 0xAE,
0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68,
0x78, 0xB8, 0xB9, 0x79, 0xBB, 0x7B, 0x7A, 0xBA,
0xBE, 0x7E, 0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C,
0xB4, 0x74, 0x75, 0xB5, 0x77, 0xB7, 0xB6, 0x76,
0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71, 0x70, 0xB0,
0x50, 0x90, 0x91, 0x51, 0x93, 0x53, 0x52, 0x92,
0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54,
0x9C, 0x5C, 0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E,
0x5A, 0x9A, 0x9B, 0x5B, 0x99, 0x59, 0x58, 0x98,
0x88, 0x48, 0x49, 0x89, 0x4B, 0x8B, 0x8A, 0x4A,
0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86,
0x82, 0x42, 0x43, 0x83, 0x41, 0x81, 0x80, 0x40
        };
```

JAVA

```
package com.ibercomp.ModBus;

/**
 * Paquete de ModBus. En la nomenclatura ModBus
 * esta clase representa un mensaje RTU.
 * @author A.Miguel - (C) 2009 Ibercomp SA
 * @version 1.00
 */
public class ModBusPacket {

    private int uchCRCHi = 0xFF ;    //Byte alto del CRC
    private int uchCRCLo = 0xFF ;    //Byte bajo del CRC

    private byte [] trama=null;
    private int lenght=0;

    /**
     * Construye un nuevo paquete.
     */
    public ModBusPacket() {
        trama=new byte[256];    //La longitud máxima de una trama es de 256
    }

    /**
     * Obtiene la trama del paquete.
     * @return El resultado es un byte[] de 256 elementos. <BR/>
     * Los elementos pueden tener valores de -128 a 127, y solo son válidos
           * los indicados por getLenght
     */
    public byte[] getTrama() {
        return trama;
    }
```

```
/**
 * Indica la dirección del esclavo ModBus asociado a este paquete.
 * @param addr
 */
public void setAddress(int addr) {
    trama[0]=toByte(addr);
}

/**
 * Obtiene la dirección del esclavo ModBus asociado a este paquete.
 * @return Entero conteniendo la dirección.
 */
public int getAddress() {
    return toInt(trama[0]);
}

/**
 * Indica el código de función asociado al paquete.
 * @param function
 */
public void setFunction(int function) {
    trama[1]=toByte(function);
}

/**
 * Obtiene el código de función asociado al paquete.
 * So este código es negativo, quiere decir que este paquete
 * es un código de error ModBus.
 * @return Entero con la función ModBus.
 */
public int getFunction() {
    return toInt(trama[1]);
}

/**
 * Permite escribir un elemento de la trama.
 * Sirve para crear el paquete a bajo nivel.
 * @param address
 * @param value
 */
public void setElement(int address, int value) {
    trama[address]=toByte(value);
}

/**
 * Permite leer un elemento de la trama.
 * Sirve para interpretar a bajo nivel la trama.
 * @param address
 * @return Un entero conteniendo una dirección del módulo esclavo.
 */
public int getElement(int address) {
    return toInt(trama[address]);
}

/**
 * Define la longitud del paquete.
 * Esta longitud incluye direccion, funcion, datos y crc.
 * @param l
 */
public void setLenght(int l) {
    lenght=l;
}

/**
 * Obtiene la longitud del paquete.
 * @return Entero con la longitud del paquete.
 */
public int getLenght() {
    return lenght;
}

/**
 * Calcula el CRC del paquete.
 */
```

```
public void calculateCRC() {
    CRC16(trama,lenght-2);
    trama[lenght-2]=toByte(uchCRCLo);
    trama[lenght-1]=toByte(uchCRCHi);
}

/**
 * Nos permite conocer si el CRC del paquete es correcto
 * @return Devuelve true si el paquete es correcto,
 * false en caso contrario.
 */
public boolean isCRCOK() {
    if(CRC16(trama,lenght)==0) {
        return true;
    }
    else {
        return false;
    }
}

/**
 * Obtiene el contenido del CRC
 * @return Devuelve un entero con el CRC
 */
public int getCRC() {
    return uchCRCHi*256+uchCRCLo;
}

/*
* Convierte un entero de 0 a 255 en un byte con signo
* @param i Valor entero a codificar como byte, de 0 a 255
* @return  Devuelve byte codificado de -128 a 127
*/
static private byte toByte(int i) {
    byte resultado;
    if (i<128) {
        resultado=(byte)i;
    }
    else if (i<256){
        resultado=(byte)(i-256);
    }
    else {
        resultado=0;
    }
    return resultado;
}

/**
 * Convierte un byte a un entero.
 * @param b
 * @return
 */
static private int toInt(byte b) {
    int resultado;

    if (b>=0) {
        resultado=b;
    }
    else {
        resultado=256+b;
    }
    return resultado;
}

//Tabla CRC bytes altos

private final static int [] auchCRCHi = {
   0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
   0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
   0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01,
   0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
   0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81,
   0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
   0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01,
   0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
```

```
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
    0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
    0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01,
    0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
    0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
    0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01,
    0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
    0x40
} ;

//Tabla CRC bytes bajos
private final static int [] auchCRCLo = {
    0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06, 0x07, 0xC7, 0x05, 0xC5, 0xC4,
    0x04, 0xCC, 0x0C, 0x0D, 0xCD, 0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09,
    0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A, 0x1E, 0xDE, 0xDF, 0x1F, 0xDD,
    0x1D, 0x1C, 0xDC, 0x14, 0xD4, 0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
    0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3, 0xF2, 0x32, 0x36, 0xF6, 0xF7,
    0x37, 0xF5, 0x35, 0x34, 0xF4, 0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A,
    0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29, 0xEB, 0x2B, 0x2A, 0xEA, 0xEE,
    0x2E, 0x2F, 0xEF, 0x2D, 0xED, 0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
    0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60, 0x61, 0xA1, 0x63, 0xA3, 0xA2,
    0x62, 0x66, 0xA6, 0xA7, 0x67, 0xA5, 0x65, 0x64, 0xA4, 0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F,
    0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68, 0x78, 0xB8, 0xB9, 0x79, 0xBB,
    0x7B, 0x7A, 0xBA, 0xBE, 0x7E, 0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
    0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71, 0x70, 0xB0, 0x50, 0x90, 0x91,
    0x51, 0x93, 0x53, 0x52, 0x92, 0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C,
    0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B, 0x99, 0x59, 0x58, 0x98, 0x88,
    0x48, 0x49, 0x89, 0x4B, 0x8B, 0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
    0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42, 0x43, 0x83, 0x41, 0x81, 0x80,
    0x40
};


/*
 * Inicializa CRC
 */
private void CRCInit() {
    uchCRCHi = 0xFF ;      //Byte alto del CRC
    uchCRCLo = 0xFF ;                                                   //Byte bajo del CRC
}

/*
 * Añade un byte al calculo de crc
 */
private void CRC16l(int c) {
    int uIndex ;                                //El indice en la tabla CRC

     uIndex = uchCRCLo ^ c;                     //Calcula el indice del CRC
     uchCRCLo = uchCRCHi ^ auchCRCHi[uIndex];   //Byte bajo del CRC
     uchCRCHi = auchCRCLo[uIndex] ;
    //Byte alto del CRC
}

/**
 * Obtiene el resultado del CRC
 * @return
 */
private int obtieneCRC() {
    return uchCRCHi * 256 | uchCRCLo ;
}

//
// Calcula el CRC del ModBus
//
//  puchMsg       <- Puntero al mensaje a calcular CRC
//  usDataLen <- Numero de bytes a calcular
//
//  CRC16 -> el crc calculado
//
private int CRC16 ( byte [] puchMsg, int usDataLen ) {
    int x;

    CRCInit();

    //Muestrea todo el mensaje
```

```
        for (x=0;x<usDataLen;x++) {
            if (puchMsg[x]>=0) {
                CRC16l(puchMsg[x]);
            }
            else {
                CRC16l(256+puchMsg[x]);
            }
        }
        return obtieneCRC();
    }
}
```

VB .net

```
Module modulo_crc_ModBus

    Public crc_table() As Byte = _
       {&H0, &HC1, &H81, &H40, &H1, &HC0, &H80, &H41, _
        &H1, &HC0, &H80, &H41, &H0, &HC1, &H81, &H40, _
        &H1, &HC0, &H80, &H41, &H0, &HC1, &H81, &H40, _
        &H0, &HC1, &H81, &H40, &H1, &HC0, &H80, &H41, _
        &H1, &HC0, &H80, &H41, &H0, &HC1, &H81, &H40, _
        &H0, &HC1, &H81, &H40, &H1, &HC0, &H80, &H41, _
        &H0, &HC1, &H81, &H40, &H1, &HC0, &H80, &H41, _
        &H1, &HC0, &H80, &H41, &H0, &HC1, &H81, &H40, _
        &H1, &HC0, &H80, &H41, &H0, &HC1, &H81, &H40, _
        &H0, &HC1, &H81, &H40, &H1, &HC0, &H80, &H41, _
        &H0, &HC1, &H81, &H40, &H1, &HC0, &H80, &H41, _
        &H1, &HC0, &H80, &H41, &H0, &HC1, &H81, &H40, _
        &H0, &HC1, &H81, &H40, &H1, &HC0, &H80, &H41, _
        &H1, &HC0, &H80, &H41, &H0, &HC1, &H81, &H40, _
        &H1, &HC0, &H80, &H41, &H0, &HC1, &H81, &H40, _
        &H0, &HC1, &H81, &H40, &H1, &HC0, &H80, &H41, _
        &H1, &HC0, &H80, &H41, &H0, &HC1, &H81, &H40, _
        &H0, &HC1, &H81, &H40, &H1, &HC0, &H80, &H41, _
        &H0, &HC1, &H81, &H40, &H1, &HC0, &H80, &H41, _
        &H1, &HC0, &H80, &H41, &H0, &HC1, &H81, &H40, _
        &H0, &HC1, &H81, &H40, &H1, &HC0, &H80, &H41, _
        &H1, &HC0, &H80, &H41, &H0, &HC1, &H81, &H40, _
        &H1, &HC0, &H80, &H41, &H0, &HC1, &H81, &H40, _
        &H0, &HC1, &H81, &H40, &H1, &HC0, &H80, &H41, _
        &H0, &HC1, &H81, &H40, &H1, &HC0, &H80, &H41, _
        &H1, &HC0, &H80, &H41, &H0, &HC1, &H81, &H40, _
        &H0, &HC0, &HC1, &H1, &HC3, &H3, &H2, &HC2, _
        &HC6, &H6, &H7, &HC7, &H5, &HC5, &HC4, &H4, _
        &HCC, &HC, &HD, &HCD, &HF, &HCF, &HCE, &HE, _
        &HA, &HCA, &HCB, &HB, &HC9, &H9, &H8, &HC8, _
        &HD8, &H18, &H19, &HD9, &H1B, &HDB, &HDA, &H1A, _
        &H1E, &HDE, &HDF, &H1F, &HDD, &H1D, &H1C, &HDC, _
        &H14, &HD4, &HD5, &H15, &HD7, &H17, &H16, &HD6, _
        &HD2, &H12, &H13, &HD3, &H11, &HD1, &HD0, &H10, _
        &HF0, &H30, &H31, &HF1, &H33, &HF3, &HF2, &H32, _
        &H36, &HF6, &HF7, &H37, &HF5, &H35, &H34, &HF4, _
        &H3C, &HFC, &HFD, &H3D, &HFF, &H3F, &H3E, &HFE, _
        &HFA, &H3A, &H3B, &HFB, &H39, &HF9, &HF8, &H38, _
        &H28, &HE8, &HE9, &H29, &HEB, &H2B, &H2A, &HEA, _
        &HEE, &H2E, &H2F, &HEF, &H2D, &HED, &HEC, &H2C, _
        &HE4, &H24, &H25, &HE5, &H27, &HE7, &HE6, &H26, _
        &H22, &HE2, &HE3, &H23, &HE1, &H21, &H20, &HE0, _
        &HA0, &H60, &H61, &HA1, &H63, &HA3, &HA2, &H62, _
        &H66, &HA6, &HA7, &H67, &HA5, &H65, &H64, &HA4, _
        &H6C, &HAC, &HAD, &H6D, &HAF, &H6F, &H6E, &HAE, _
        &HAA, &H6A, &H6B, &HAB, &H69, &HA9, &HA8, &H68, _
        &H78, &HB8, &HB9, &H79, &HBB, &H7B, &H7A, &HBA, _
        &HBE, &H7E, &H7F, &HBF, &H7D, &HBD, &HBC, &H7C, _
        &HB4, &H74, &H75, &HB5, &H77, &HB7, &HB6, &H76, _
```

```
            &H72, &HB2, &HB3, &H73, &HB1, &H71, &H70, &HB0, _
            &H50, &H90, &H91, &H51, &H93, &H53, &H52, &H92, _
            &H96, &H56, &H57, &H97, &H55, &H95, &H94, &H54, _
            &H9C, &H5C, &H5D, &H9D, &H5F, &H9F, &H9E, &H5E, _
            &H5A, &H9A, &H9B, &H5B, &H99, &H59, &H58, &H98, _
            &H88, &H48, &H49, &H89, &H4B, &H8B, &H8A, &H4A, _
            &H4E, &H8E, &H8F, &H4F, &H8D, &H4D, &H4C, &H8C, _
            &H44, &H84, &H85, &H45, &H87, &H47, &H46, &H86, _
            &H82, &H42, &H43, &H83, &H41, &H81, &H80, &H40}


    Public Function crc_16(ByVal ModBusframe() As Byte, ByVal length As Integer) As Integer

        Dim i As Integer
        Dim index As Integer
        Dim crc_Low As Integer = &HFF
        Dim crc_High As Integer = &HFF

        For i = 0 To length - 1
            index = (crc_High Xor CByte(ModBusframe(i) And 255))
            crc_High = (crc_Low Xor CByte(crc_table(index) And 255))
            crc_Low = CByte(crc_table(index + 256) And 255)
        Next

        Return ((crc_High * 256) + crc_Low)
    End Function

    Public Sub main()
        'paquete a enviar
        Dim d1() As Byte


        'Añadir CRC
        Dim crc As Integer = crc_16(d1, d1.Length)

        ReDim Preserve d1(UBound(d1) + 2)
        d1(UBound(d1) - 1) = crc \ 256
        d1(UBound(d1)) = crc Mod 256
    End Sub

End Module
```