MiniMod

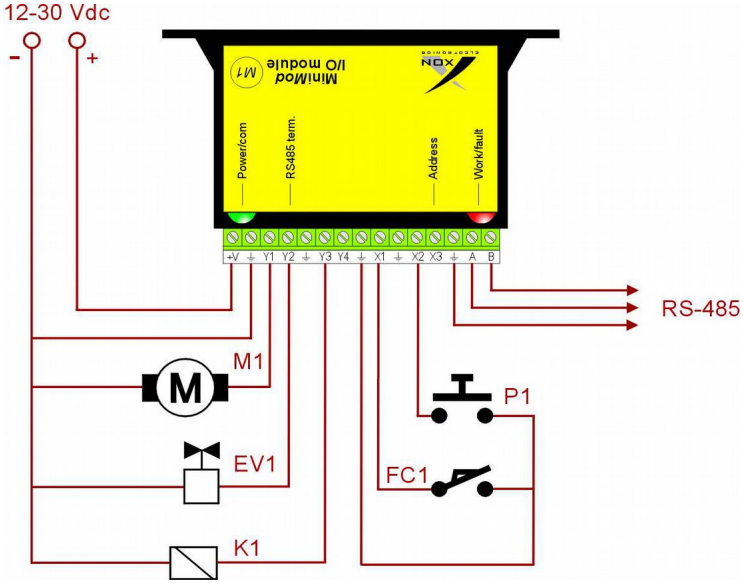
Power I/O module for PC/PLC with Modbus / simple text protocol on RS-485

Index

Introduction

Minimod is an I/O module working at 24 volts, driven by an RS-485 serial interface using either the Modbus protocol or simple text commands. The Minimod features are:

- 3 digital inputs @24 volts (from 5 to 30 volts)
- 4 power PNP outputs, 2A current, also suitable for inductive loads
  - the output voltage is the same as the power supply: from 10 to 30 volts DC
  - the current of all the 4 outputs summed together is 4 maximum
- The outputs are monitored for overload and open (broken) load
- Internal watch-dog, with programmable output configuration
- Advanced functions for outputs: configurable automations, pulses, flashes
- Programmable latency for inputs, and counters
- Dip-switches and rotary selector for quick module configuration
- Internal flash memory to store settings
- Electronic circuit embedded in resin inside a plastic container, with fixing loops



Connection example: switches, motor, relay and electrovalve.

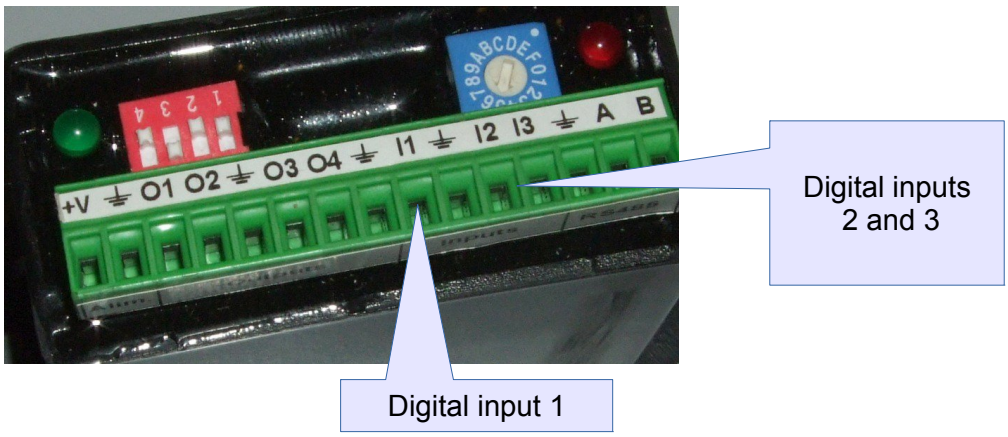
Power supply

The device must be powered with a DC voltage in the range 10-30 volts. The current consumption is 150 mA for the device itself, plus the consumption of the attached loads, up to 4 amperes.

+V is the positive terminal for power supply.

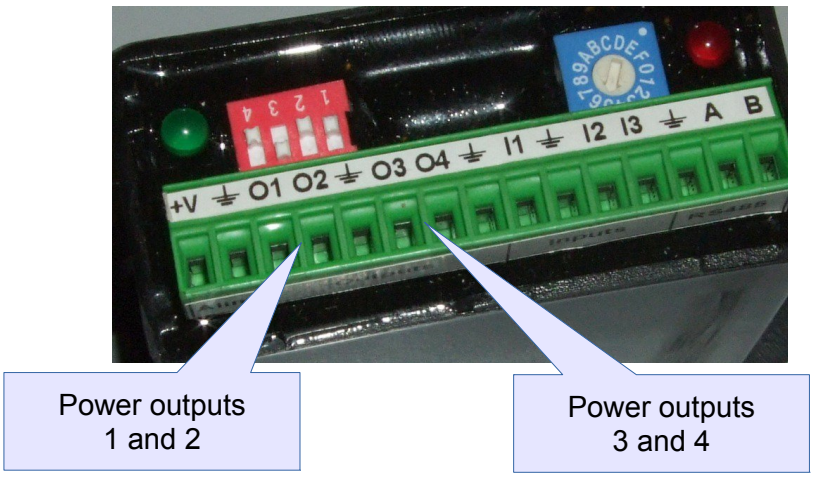
The “ground” symbol is the negative terminal.

Digital (24 volts) inputs



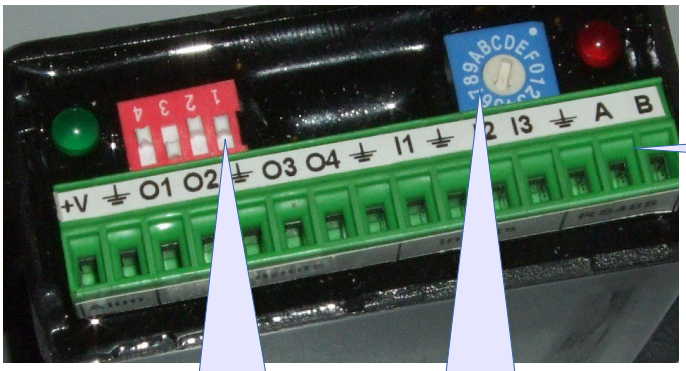
The three digital inputs (I1..I3, also called X1..X3) have a 10 Kohm input impedance and a protection circuit for the internal components. This protection circuit limits the maximum input frequency to about 1 Khz

Digital (24 volts) power outputs



The four power outputs O1..O4 (also called Y1..Y4) can each supply 2 amperes of current, but the total for the module must not exceed 4A. These outputs are protected counter overload and over-temperature, and it is also possible to detect the load absence (broken load). These conditions can be read by the computer or the PLC.

RS-485 serial interface



An RS-485 line is composed by two balanced signals, A and B, and a common reference ground. It is intended to be used on twisted cables; in case of long distances or noisy environments, it is best to use twisted and shielded cables, with the shield connected to protective (earth) ground. At both ends of the line there must be a termination resistor – the two resistors must be of equal value. Minimod already contains two selectable termination resistors, enabled by the dip-switches 1 and 2 as seen in this table:

Switch 2	Switch 1	Termination	Comment
OFF	OFF	*Absent*	Multidrop configuration, when this Minimod IS NOT the last device of the chain.
OFF	ON	680 ohms	Termination for short lines (<50 meters)
ON	OFF	330 ohms	Termination for medium-length lines
ON	OFF	220 ohms	For long distances (hundreds of meters)

The RS-485 standard indicates a termination of 120 ohms, but good results are achievable with higher values, favoring energy saving. When a 120 ohms termination is desired, disable the internal resistors and use an external one.

The rotary address selector assigns the so-called *Slave address* for the Modbus protocol. In Modbus the address is mandatory and must be in the range 1 to 247 (the *Master* always has address 0). If the simple text protocol is used instead of Modbus, the address is optional. The address given to the device is that indicated by the selector, except that positions from “A” to “F” correspond to addresses 10 to 15, and position “0” gives address 16.

On the serial line it is possible to use, even concurrently, the Modbus RTU protocol and the simple text protocol. The communication settings are commonly 19200 bauds, 8-bit data length, “even” parity and 1 stop bit (this is the Modbus recommendation). It is possible to change these settings, along with other options, via text protocol; the settings can then be saved permanently. For convenience, moving dip-switch 3 to ON while the device is on causes the default serial settings to be loaded (19200 baud, even parity and 1 stop bit).

Process control and data acquisition by a PC/PLC

The Minimod module must be driven as a *Slave* by a PC or a PLC acting as a *Master*. Two communication protocols are available, even at the same time, on the same bus or device (but conforming to the correct serial settings, of course).

The Modbus protocol states that the master sends a command to a particular slave, which in turn carries out the command and sends back a reply: the lack of a reply indicates a communication failure. Modbus commands are mainly of two kinds: value-write and value-read. For a value-write command the slave sends back an "operation success" message; for a value-read command the slave replies with the requested values. The simple text protocol is modeled after the same principle: there are two kinds of command for writing values and reading values; the slave replies similarly. Please note that for both protocols the master sends command to a specific slave (using the *slave address*); all the slaves receive the command, but only the selected one works it out and sends a reply. Both the protocols can send a command to all the slaves together (*broadcast*); in this case all the slaves carry out the command, but none of them must send back a reply. Deploying these broadcast is not recommended because it is not possible to be sure that all the slaves have understand (and hence performed) correctly the command sent.

Using a multidrop bus like the RS-485, and a protocol like Modbus, let a master to exchange data with several slaves in a reasonably simple and efficient way; but, when a high number of slaves is present, the cycle time of the master can be slowed down. Every cycle starts by polling several (or all) the slaves, then the data is processed, and finally new values are passed to the slaves. The slave polling phase is normally responsible for the latency of such a system. Using three or four slaves, and also depending on the quantity of data, the cycle time can be about 50 milliseconds; using a larger number of slaves, and many more I/Os, the cycle time can rise up to hundreds of milliseconds and, in some cases, this is simply a too long time. The Minimod module has a few functions which help to mitigate this problem.

Input conditioning

The digital inputs are sampled by Minimod once a millisecond, and the signal variations are reflected inside the module after a programmable time (latency) for both the rising edges and the falling edges. XLATUP is the latency time for rising edges (signal going from 0 to 1), and XLATDN for falling ones (going from 1 to 0). These latencies provide for noise filtering, de-bouncing, or even pulse enlarging. Suppose for example that a normally open button called P1 is connected to the X1 input; when P1 is depressed, voltage is applied to X1 input. When P1 is up, X1 is 0. On a very noisy line, a spike could briefly convey a voltage to X1: by setting a 20ms latency to XLATUP, noises shorter than 20ms are canceled. A 20ms time for the falling edge, XLATDN, can eliminate bounces due to dirty contacts. Finally a much much bigger fall-down latency (larger than the PLC cycle time), like 500ms, can enable the PLC to read pulses shorter than its own cycle time (a pulse is missed if it starts and terminates in-between a polling and the next one).

Pulse generation

If the PLC has a cycle time of, say, 80 milliseconds, it is impossible for it to generate a pulse of 50ms on an output. However the PLC can ask Minimod to perform such a pulse, by setting Y1PULSE to 50; the Y1 output is set high for 50 ms, and then turned off. Minimod has 4 pulse generators (one for each output), each settable to a value in the range 1 to 65000 milliseconds.

Automatic continuous blinking

Minimod can also continuously flash an output, using a 16 bit rotating mask which indicates the on- and off- state with speed of about a tenth of a second. For example, the binary mask “0000000011111111” (255 in decimal) produces a slow continuous flashing, “1010101010101010” produces a fast flashing, and “0000000000000101” (5 in decimal) does continuously a couple of short flashes. Every output has its own mask, named from Y1LAMP to Y4LAMP.

Automatic outputs setting

It is possible to instruct Minimod to automatically turn on or off an output when a certain configuration presents itself to the inputs. For every Y output, it is possible to specify three conditions for turning the output on, and three for turning it off. For example, the text command “Y1AON1=X1X2” indicates that if the inputs X1 and X2 are both high, Y1 must be activated. “Y1AON2=X1-X3” says that Y1 must be activated when X1 is on *and* X3 is off; this condition, combined with the previous one, states that Y1 is activated when X1 is high, and, X2 is high or X3 is low. Adding “Y1AOFF1=-X1” makes Y1 turn off when X1 goes down. This mechanism relieves the PC or PLC from a heavy duty, and is fast and precise. Turn-off conditions have precedence over Turn-on ones.

Modbus function “Collect” (event management)

When many devices have to be polled by the master, the needed time can rise too much; even worse is when the results of many pollings are the same, because there has been no variations. This “collect” function polls a group of devices with a time-slot mechanism which lowers the required time down to 5% of the original time when no variations have happened. The Minimod module stores the input variations in an internal queue, ready to be transmitted to the master, and cleared when the transmission is acknowledged. When the master issues the special Collect command to a group of devices, only the first, of those having variations to send, replies; the other get blocked and must wait for another Collect. See later for details.

Outputs feedback

This module monitors its outputs. When an output is off, it can detect an open circuit (possibly a broken load); when an output is in ON state, Minimod can detect an overload (too much current or thermal warning). When one of these conditions is detected, two feedbacks bit are set, and these bits are also mirrored in two virtual inputs X9 and X10. The outputs Y1 and Y2 share a single feedback bit mirrored in X9; Y3 and Y4 share the other feedback mirrored in X10. By setting the correct register, it is possible to trigger the watch-dog (emergency condition) (see next chapter below).

Internal watch-dog (security and coherency check)

While controlling a process, it is desirable that certain failures are detected and the system is brought to a “safe” state – i.e. the outputs are set in a known and safe state. The internal watch-dog, if enabled, can check a few things and, if a failure is detected, “it fires”: the modules goes in a state where the outputs have a known, programmable configuration which can not be changed without resetting the watch-dog. This emergency state can be triggered by the output feedbacks (see previous paragraph) or by a failure in communication with the PC/PLC: if this check is enabled, a timeout from receiving a valid command from the master is regarded as a communication loss which is a danger. At power up, the watch-dog is completely disabled.

Configuration of Minimod

From the factory, the device is preset with “standard” parameters (especially the serial communication settings, as recommended by Modbus). However it is possible, using text commands, to change some settings; after having modified them, it is possible (and suggested) to save them permanently in the internal non-volatile flash memory.

AFTER HAVING MODIFIED THE SETTINGS, remember to store them in memory.

To change settings please connect the device to a PC via the RS-485 line and use the correct serial parameters. Use then a normal terminal emulator to “talk” to the module.

RS-485 port reset

Factory default parameters for communication are 19200 baud, 8 data bit, even parity and 1 stop bit. If desired it is possible to restore these settings temporarily by moving dip-switch SW3 from off to on position while the device is operating. This will reset the communication parameters, but at the next power up the device will again get the settings from internal stored preferences.

Setting the communication parameters

Send to the device the following command:

>SERLINE=19200E

(greater-than symbol, then SERLINE=19200, then letter “E”), followed by a CR (the Enter key). This way the serial port is set to 19200 baud with even parity and 1 stop bit. Instead of 19200 it is possible to specify 4800, 9600, 38400 and 57600.

Instead of the final letter “E” (which means “even” parity) it is possible to use the letter “O” (which means “odd” parity), “N” for no parity and 2 stop bits, or “X” for no parity and only 1 stop bit.

If no final letter is specified, parity and stop bits remain as before.

If the command is correct, Minimod replies with “OK”. If the syntax or the arguments are not acceptable, the reply is “Error.”; if no reply is received then the communication parameters are wrong or the command did not start with a greater-than “>” symbol. Minimod replies using the same communication parameters used until now: only after having replied with “OK”, the new parameters take effect.

Setting the slave address

The slave address of Minimod depends on the position of the rotary selector and an offset called MBADD stored in the preferences, which is summed to the selector value. This way it is possible to use any address in the range 1 to 255 (Modbus only accepts 1 to 247). The factory default for MBADD is zero, so the slave address is the same as indicated by the selector. By sending the command:

>ADDRESS=180

(“>” symbol, ADDRESS=180, then CR/Enter), the device is assigned 180 as slave address. What really happens is that MBADD is set to 180-selector_rotary_value, so it is still possible to use the selector to change address. It is advisable to move the selector to position 0 before using this command, in order to ease subsequent address changes. Of course, instead of “180” it is possible to use any number from 1 to 247.

Setting the reply delay

Because the RS-485 line is half-duplex, transmissions from two devices must not overlap. Minimod waits 1/100 of a second, before replying, in order to allow the master to deactivate its transmitter; it is possible to augment this delay by sending the following command:

>RDELAYT=nnn

where nnn is a number from 0 to 200, and represents the delay in hundredths of a second.

Setting the Modbus communication timeout

The Modbus protocol states that the end of a message is signaled by a silence on the line, lasting 28 bits or more, and Minimod respects this value. Should the master have some latency problems, inserting unwanted silence between characters, it is possible to enlarge this 28-bit timeout by a number of tenths of milliseconds comprised between 0 and 200. Sending this command:

>RDELAYM=nnn

where nnn is the number of tenths of milliseconds (0.0001) to wait more for a message to finish.

Saving setting permanently

After having done some modifications to the preferences, they must be saved in order for them to be remembered on subsequent power ups. Send the command:

>WCONF=1

to write them in internal flash memory. Minimod replies with “OK”, or “Error 1” if an error has occurred.

Modbus interface

Minimod implements the following Modbus functions for reading and writing bits:

1 Read Coils (0XXXX)

2 Read Discrete Inputs (1XXXX)

5 Write Single Coil

15 Write Multiple Coils

and the following functions for reading and writing 16-bit words:

3 Read Holding Registers (4XXXX)

4 Read Input Registers (3XXXX)

6 Write Single Holding Register

16 Write Multiple Holding Registers

22 Mask Write Register

By reading and writing to relevant addresses, it is possible to access all the functions Minimod implements. All the addresses are counted starting from 0 (the first register has address 0).

Digital inputs, latencies and counters

The inputs can be read as Discrete Inputs from physical addresses 0 to 15. The first 3 bits map to the real inputs X1, X2 and X3; the next 5 are not implemented and always read as 0; the next 2 (X9 and X10) mirror the feedback bits, and the last 6 are non implemented. The 16 inputs can be read as a single Input Register at physical address 0.

The three X1 X2 X3 inputs are cooked, i.e. there is a programmable latency on edge detection, so it is possible to mask noise and bounces out or enlarge a pulse width. The latency times, in milliseconds, reside in the Holding Registers at addresses 19, 20 and 21 for rising edges, and 35, 36 and 37 for falling edges. The power-on setting for these values, which range from 0 to 65535 (16 bits), are 20 milliseconds.

Minimod contains 3 counters for rising edges, one for each input. These counters are readable and writable in the Holding Registers at addresses 11, 12 and 13.

Power outputs, blinkers, pulsers, and feedbacks

The 4 outputs are accessible as Single Coils at addresses 0, 1, 2 and 3. Holding Register 0 contains a bitmask of those bits too.

The blinkers are contained in the Holding Registers 75, 76, 77 and 78; a bitmask different from 0 imposes a flashing to the output and disables the normal usage.

The pulsers generate a positive pulse of the specified duration, in milliseconds, to the relevant output, while counting down to zero. They can be read and written in the Holding Registers at address 83, 84, 85 e 86. These pulsers have higher priority than the blinkers.

As already said, the electronics of Minimod monitor the outputs: when an output is off an open circuit can be detected, which could mean a broken load (a burnt lamp for example). When an output is active

and emits a voltage, an over-current (over 2 amperes) can be detected which could mean a short-circuit. In both cases a feedback bit is set to 1; the outputs Y1 and Y2 set the Discrete Input at address 48; Y3 and Y4 set the Discrete Input at address 49. These two bits are also mirrored in the Discrete Inputs at address 8 and 9 (corresponding to virtual inputs X9 and X10).

The Single Coil at address 112 enables the watch-dog for the first feedback bit (when it turns to 1); the Single Coil at address 113 enables it for the second feedback bit (of Y3/Y4).

Automatic output management

This function can relief the master from the duty of setting outputs basing on the inputs, because the outputs can be turned on and off directly by Minimod when a given pattern presents itself at the inputs. The net result is a very little latency of the system.

The Holding Registers 51, 52, 53 contain the 3 configurations (conditions) for turning on Y1. The next 3 (54,55,56) contain conditions for Y2; for Y3 the addresses are 57, 58, 59 and for Y4 60, 61 and 62.

For each set of conditions, it is sufficient that one of them gets true to perform the associated operation (turn on or off the output). The “turn off” conditions are at addresses 63-64-65 for Y1, 66-67-68 for Y2, 69-70-71 for Y3, and 72-73-74 for Y4.

A single condition is made of 16 bits; the 8 lower order bits (numbered from 0 to 7) indicate what inputs have to be checked, and the 8 higher order bits indicate what inputs must be inverted before evaluation. As Minimod only has 3 inputs, 5 bits in the lower byte and 5 bits in the higher byte are not used.

As an example, the condition binary mask “0000.0000.0000.0111” (7 in decimal) only contains 3 “1” bits in the lower (rightmost) byte – they correspond to inputs X1 X2 and X3. When all the 3 inputs are on (X1=X2=X3=1), the condition is verified; so, if this condition were written into Holding Register 51 (first ON condition for Y1), Y1 would turn on. If this condition were written in a register from 63 to 65, instead, Y1 would turn off. If a condition must check an input to be off, the relevant bit in the upper byte of the mask must be set; for example the condition “X1=1, X2=0” is verified when X1 is on and X2 is off, and can be expressed with the binary mask “0000.0010-0000.0011”. The rightmost 3 bits, “011”, indicate to check X1 and X2; the rightmost 3 bits in the upper half, “010”, indicate that X2 must be inverted.

The Modbus function Collect

The Collect function is a special Modbus message with the following format in byte:

DST, FCN, FIRST, LAST, SLVACK, SLVSEQ

DST indicates the destination slave; very often a broadcast is needed, so it is normally 0.

FCN indicates the modbus function, which is 70.

FIRST and LAST define the address range (or window) of the slaves to be polled; for example, FIRST=1 and LAST=16 authorize to reply only the slaves with addresses from 1 to 16 (a slave replies only if it has some event in its queue). Each slave has 3 ms of time for replying, so a Collect which polls 10 slaves has a nominal timeout of 30 ms (plus 1 for safety).

SLVACK and SLVSEQ will be explained later; if not needed, use 0.

When the master emits the Collect command, the indicated slaves are authorized to reply, in order of ascending address, for 3 ms each. If no slave has events to report, the command goes in time out.

If one of the slaves replies, it does it with a message which has the following format (in bytes):

SLAVE, FCN (=70), SLVSEQ, TYPE (2=inputs, 1=outputs), DATH, DATL, 0

DATH and DATL form the 16-bit mask of the inputs or outputs (see filed TYPE) that was stored as event, after the variation has been detected.

After the master receives the reply, it must send another Collect command to acknowledge the slave that the event has been received. This Collect command must set the fields SLVACK and SLVSEQ equal, respectively, to the just received SLAVE and SLVSEQ. There is no need to direct the command to a specific slave – a broadcast works well too, and in fact this is the normal way. Suppose that the polling cycle starts by querying slaves 1 to 16, and slave number 4 replies. In order to prosecute the polling, slaves 5 to 16 have still to be addressed; so a Collect with window 5-16 must be done, which also indicates that slave number 4 is acknowledged using the lastly received SLVSEQ.

Modbus register map (read-only registers)

Modbus Addr.	Name	Data type	Notes
10001-10003	X1..X3	Bit	Inputs from X1 to X3 (cooked)
10009	FB1	Bit	Feedback (output error monitor) of Y1 + Y2
10010	FB2	Bit	Feedback of Y3 + Y4
10018	EVENTSPRS	Bit	Events for Collect are present
10020	GENERICFAIL	Bit	Indicates a generic hardware error
10023	FLASHFAIL	Bit	Indicates an internal flash memory failure
10025	WDTFB	Bit	The watch-dog has fired because an output feedback (FB1 or FB2) is active
10026	WDTCOMM	Bit	The watch-dog has fired because a connection loss with the computer/PLC
10129	SW3	Bit	Indicates the status of dip-switch 3
10130	SW4	Bit	Dip-switch 4 status
30010	ROTSW	UINT8 (Byte)	Rotary address selector value

The UINT8 *type is an unsigned 8 bit integer, ranging from 0 a 255.*

Modbus register map (read and write registers)

Modbus Addr.	Name	Data type	Notes
00001-00004	Y1..Y4	Bit	Outputs from Y1 to Y4
00081-00083	EVX1..3	Bit	Event recording enable inputs X1..X3
00089-00090	EVFB1..2	Bit	Event recording enable for feedbacks FB1-FB2
00097-00100	EVY1..4	Bit	Event recording enable for outputs Y1..Y4
00113-00114	WDTFB1..2	Bit	Enables the watch-dog for FB1-FB2
00145-00148	WDTY1..4	Bit	Value of Y1..Y4 when the watch-dog fires
00161	WDTFIRED	Bit	Indicates that the watch-dog is firing (active). Must be reset by the master
40001	YWORD	Bitmap4	Outputs Y1..Y4 status
40006	XEVMASK	Bitmap3	Event recording mask for X1..X3
40007	YEVMASK	Bitmap4	Event recording mask for Y1..Y4
40129	WDTTIME	UINT16 (word)	Watch-dog timeout, in 1/100 of seconds, for communication with the master. If set to non-zero, Minimod expects messages from the master at least every WDTTIME time; if no messages are received in that time, the watch-dog fires. Set to 0 to disable this function.
40145	WDTYMASK	Bitmap4	Bitmask for the outputs Y1..Y4 when the watch-dog is firing
40012-40014	XCOUNT1..3	UINT16	Counters for the rising edges of X1..X3
40020-40022	XLATUP1..3	UINT16	Latency (or delay), in ms, for rising edge detection of X1..X3
40036-40038	XLATDN1..3	UINT16	Latency, in ms, for falling edge detection of X1..X3
40052-40054	Y1AON1..3	Bitmap8+8	Three condition masks (in logical OR) for automatic turning on of Y1
40055-40057	Y2AON1..3	Bitmap8+8	ON conditions for Y2
40058-40060	Y3AON1..3	Bitmap8+8	ON conditions for Y3
40061-40063	Y4AON1..3	Bitmap8+8	ON conditions for Y4
40064-40066	Y1AOFF1..3	Bitmap8+8	Three conditions for automatic turning off of Y1
40067-40069	Y2AOFF1..3	Bitmap8+8	OFF conditions for Y2
40070-40072	Y3AOFF1..3	Bitmap8+8	OFF conditions for Y3
40073-40075	Y4AOFF1..3	Bitmap8+8	OFF conditions for Y4
40076-40079	YLAMP1..4	Bitmap16	Automatic blinking mask for Y1..Y4
40084-40087	YPULSE1..4	UINT16	Duration of a positive pulse, in milliseconds, to be executed by output Y1..Y4

The type Bitmap *indicates that the value is expressed as mask of bits.*

The type UINT16 *indicates an unsigned 16-bit number ranging from 0 to 65535.*

Text commands protocol

This protocol is an alternative to Modbus, simpler and effective, suitable for shell scripts or batch files too.

The communication logic is the same: the master (PC or PLC) sends a command, and waits the reply for a few hundredths of a second. Minimod always replies with no delay, apart for the command “>WCONF=1” which is slower and can take up to some tenth of a second.

Every command to Minimod is composed by a string of ASCII characters terminated by a CR (Carriage return, ASCII character number 13 corresponding to the Enter key). If spaces (number 32) or Line Feeds (number 10) are present, they are ignored; Minimod, moreover, pays no attention to case so lowercase and uppercase are the same.

The maximum length of a message is 76 characters – more characters, until the next CR, are discarded. No valid packets can be so long, anyway.

All the packets begin with a question mark "?" character for reading, or a greater-than ">" symbol for writing. The format of a read message is the following:

Example of a command read / interrogate				
?	**X**	**1**	**<CR>**	**(<LF>)**
Command (print)	Argument (X1)		Terminator	Ignored LF if sent

The command argument, in this example **X1**, is an identifier known by Minimod; see later a table with all the known identifiers. If the packet is well formed and the identifier is valid, Minimod replies with the requested value:

Reply to a read command				
X	**1**	**=**	**0**	***<CR>***
Identifier (X1)		Syntactic element	Value	Terminator

After the equal sign, “=”, always come 1, 3 or 5 digits – it depends on identifier type.

For a writing message (set an output for example) the format is the following:

Example of a write command						
>	**Y**	**1**	**=**	**1**	**<CR>**	**(<LF>)**
Command (set)	Argument (Y1)		Syntactic element	Value	Terminator	Ignored LF if sent

The value after the equal sign must be a positive integer without sign; it is not mandatory to use non significant zeros. The value must fall into the acceptable range for the identifier: for a bit, only 0 or 1; for other data types values from 0 to 65535.

After the writing command is received and performed, Minimod sends back a reply:

Reply to successful command		
O	**K**	**<CR>**
Affirmative reply (no error)		Terminator

If the sent message is malformed (does not start with "?" or ">"), no reply is read back. In case of other errors:

Reply to erroneous command					
E	**r**	**r**	**o**	**r**	**<CR>**
Negative reply (error)					Terminator

Slave addressing

The commands shown above were not directed to a specific slave – they correspond to a broadcast and should be used when only a device is connected. When more than a device is connected, it is mandatory to put a slave address before the command. To specify a particular slave, use the “at” symbol “@” and the slave number, then the normal command. For example:

@25?DEVICE

means “to slave 25: print device”. The number after the “@” can not be 0.

Read commands table (text mode)

Each one of the following commands must be preceded by a question mark and followed by CR as explained before. The two characters are not show for clarity. Many of the following identifiers are read-only, but others can also be written to and are also shown in the next section.

Address

(ex. ?ADDRESS): replies with the slave address of the device (ex. ADDRESS=034).

Device

Returns name and version of the device; currently is “MINIMOD 1.1”.

Xword

Replies with a 16 bit mask of the status of the inputs. For example, “XWORD=00003” means that X1 and X2 are on, and X3 is off).

X1, X2, X3, X9, X10

(ex. ?X1): return the status of the single input specified: 1=on, 0=off. X9 and X10 are an alias for the output feedbacks FB1 and FB2.

Fbacks

Returns a bit mask of the two output feedbacks. For example, “FBACKS=00003” indicates that both FB1 (feedback for Y1 and Y2) and FB2 (for Y3 e Y4) are on.

SW3 ed SW4

Shows that status of the dip-switches 3 and 4 respectively.

Yword

Replies with a 16 bit number, as seen for Xword, which is the status of the outputs.

Y1..Y4

Shows the status of the indicated output: 1=on, 0=off.

FBEnable

Replies with the bit mask of the feedbacks enabled to make the watch-dog fire if the feedback bit goes high. For example, “FBENABLE=3” indicates that both feedback bits fire the watch-dog if they go to 1.

WDTtime

Shows the timeout, for communication with the master, which fires the watch-dog. If 0, there is no timeout. If different than 0, then Minimod expects a message from the master at least every WDTTIME hundredths of a second, otherwise a failure of the master is assumed and the watch-dog fires.

WDTmask

Returns the bit mask of the output configuration to be used when the watch-dog fires.

Wflags

Reads a mask of few bits about the internal status of the device. In particular bit 1 (the second bit of weight 2) indicates that the watch-dog has fired.

Xcount1..Xcount3

Replies with the content of the indicated counter (1 to 3) which counts the rising edges of the associated input (X1 to X3).

X1latup..X3latup

Read the latency for rising edges, in milliseconds, for inputs X1 to X3 respectively.

X1latdn..X3latdn

The same as before, but for the falling edge.

Y1lamp..Y4lamp

Returns the 16-bit mask for the automatic blinkers associated with the outputs Y1..Y4.

Y1pulse..Y4pulse

Read the content of the pulse generator of the associated output. The generator starts with the indicated value when written, and counts down every millisecond.

Write commands table (text mode)

To execute the command, the following identifiers must be preceded by a ">", followed by an "=" sign, a value, and the CR character (carriage return, or Enter), as explained in previous chapters. The syntax is not shown here for clarity. When the command is executed, Minimod replies with "OK".

Address, Serline, RdelayT, RdelayM, Wconf

These commands are to set preferences and are explained in the chapter "Configuration of Minimod".

Yword

Sets all the outputs, using the specified number as a bit mask. For example, ">YWORD=1" activates the output Y1 and turns off all the others.

Y1..Y4

Activates or deactivates a single output: ">Y2=1" turns on the second output, Y2, while ">Y2=0" turns it off. After the "=" sign, only a single 0 or 1 digit is valid.

FBEnable

Writes a bitmask to enable the two output feedbacks to fire the watch-dog. ">FBENABLE=0" disables the watch-dog for feedbacks; ">FBENABLE=1" activates it for FB1 (feedback of Y1 and Y2); ">FBENABLE=2" activates only FB2, and ">FBENABLE=3" activates them both.

WDTtime

Sets the time, in 1/100 of a second, for firing the watch-dog on serial communication with the master. When this timeout is greater than 0, Minimod must receive a valid command at least every WDTTIME time, otherwise the watch-dog fires. Valid values are from 0 (no timeout) to 65535 (slightly less than 11 minutes).

WDTmask

Sets the mask for the output configuration when the watch-dog fires. For example, ">WDTMASK=0" states that when the watch-dog fires, all the outputs are turned off.

Wflags

Writes the internal flags of the module. Its main purpose is to reset the watch-dog by issuing a ">WFLAGS=0".

Xcount1..Xcount3

Xcount1, Xcount2 and Xcount3 are the counters associated with the inputs X1..X3. Writing to these identifiers preset their value – commonly they are set to 0 to begin a new count, like ">XCOUNT1=0".

X1latup..X3latup

Set the latency (or delay), in milliseconds, of the rising edge of inputs X1..X3. Valid numbers range from 0 to 65535.

X1latdn..X3latdn

Same as before, for falling edge.

Y1lamp..Y4lamp

Writes the blinker mask of the associated output. For example, ">Y1LAMP=1" corresponds to the 16-bit mask "0000000000000001" and causes a brief, continuously repeated flash of Y1.

Y1pulse..Y4pulse

By writing a value greater than 0 in these identifiers, a positive pulse is generated on the respective output, with the specified duration in milliseconds. For example, ">Y1PULSE=100" will turn on Y1 (if not already on) and, after 100 ms, will turn it off. It is also possible to read back the value, which keeps counting down until 0.

Y1AON1..Y1AON3, Y2AON1..Y2AON3, Y3...

These identifiers set the condition for turning on an output automatically. The value to be specified contain the names of the inputs to check, each perhaps preceded by a minus sign to indicate a negation. For example ">Y1AON1=X1X2" states that Y1 turns on if both X1 and X2 are on; ">Y3AON2=X2-X3" sets the second condition (of the 3 available) and states that Y3 turns on if X2 is on and X3 is off – X2 does not matter in this condition. To clear a condition an empty value is used, i.e. ">Y2=".

Y1AOFF1..Y1AOFF3, Y2AOFF1..Y2AOFF3, Y3...

Same as before, but for turning off an output. These conditions have precedence over those for turning on: if both a turn-on and a turn-off condition are true at the same time, the relevant output is turned off.

XON Electronics Srl
www.xonelectronics.it
info@xonelectronics.it

Please report errors or imprecisions to web@xonelectronics.it