MiniStep

Stepper motor drive for PC/PLC with Modbus / simple text protocol on RS-485

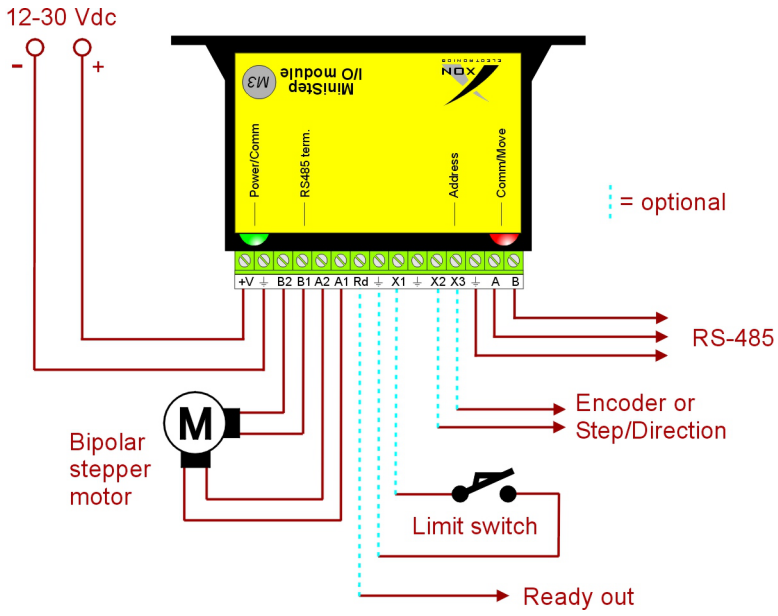
Index

Introduction

Ministep is a stepper motor drive with RS-485 interface using either the Modbus protocol or simple text commands, and I/O pins for easier deployment. Motor movements can be commanded via RS-485, at high level (i.e., positioning), or using Step/Direction signals wired to terminal block.

The Ministep features are:

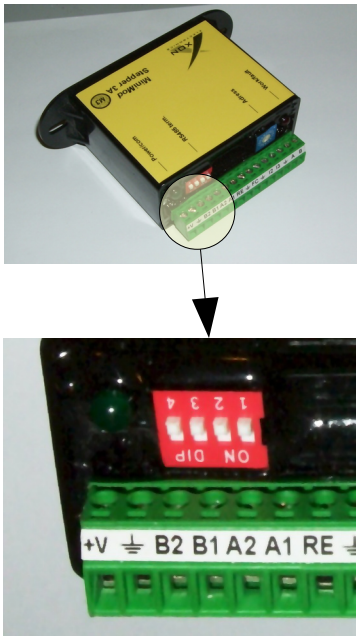
- Power supply from 12 to 34 volts D.C.
- Up to 3A (continuous) output current for bipolar stepper motor (6A peak)
- Full Step, Half step, microstepping up to 1/16
- Configurable currents for hold, run, acceleration and deceleration phases
- Programmable movement profile with speed, acceleration and deceleration
- Commands for absolute/relative positioning and other advanced functions
- 3 digital inputs as GPIO, limit switch or quadrature encoder, 1 digital output (Ready/Busy)
- Configurable via rotary selector, dip switch, and internal flash memory
- Internal watch-dog to ensure reliability and safety
- 2 LEDs for showing the module status
- Electronic circuit embedded in resin inside a plastic container, with fixing loops



Connection example. Dotted lines indicate optional connections.

The RS-485 interface can be used to fully control the motor, so there is no need to use any optional connections – but they can be used to connect additional equipment such switches or push-buttons. Otherwise, the RS-485 interface can be used to set up parameters, store them in non-volatile memory, and from that point the motor can be commanded via Step/Direction without using the RS-485 interface.

Power supply



The device must be powered with a DC voltage in the range 10-34 volts. The current consumption is 150 mA for the device itself, plus the consumption of the attached motor, up to 6 amperes.

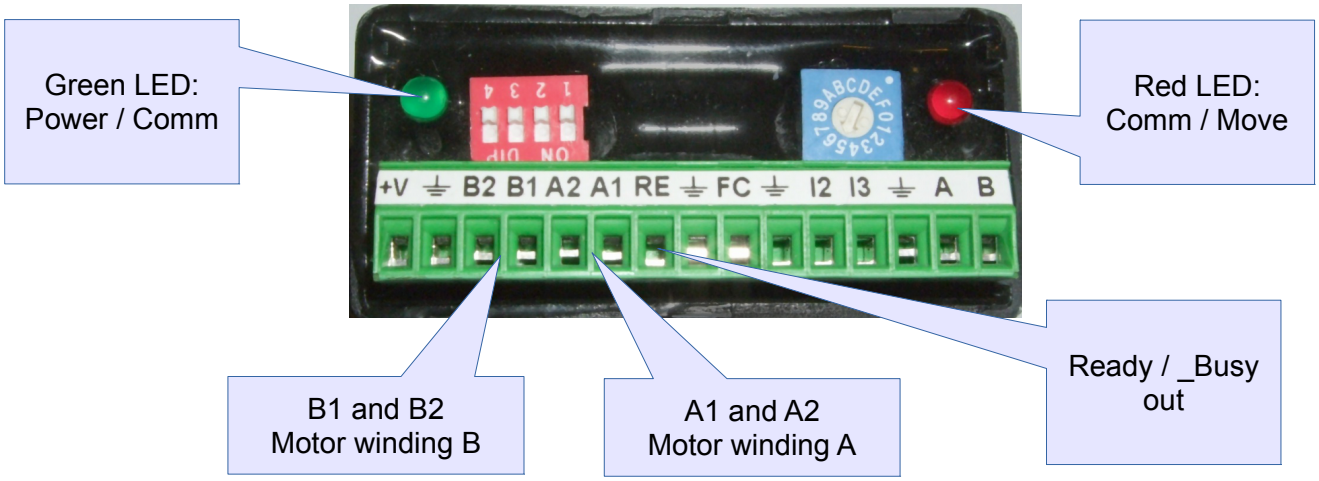
+V is the positive terminal for power supply.

The "ground" symbol is the negative terminal.

Status LED

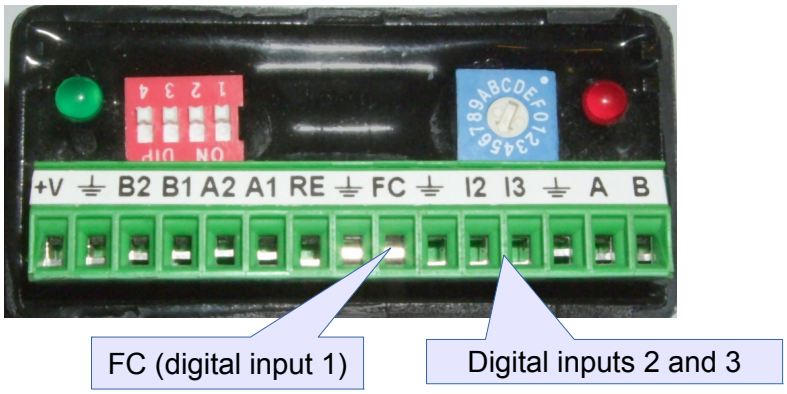
Ministep has two leds (see picture below). The green one on the left stays normally lit to signal power ON state, and blinks briefly when a command is received from the RS-485. It also blinks continuously if the internal watch-dog has fired. The red LED on the right is lit when the motor turns, and it blinks briefly on or off when the module sends a reply to the master.

Motor connection and Digital Ready/BUSY output



The Ready/BUSY pin is an open collector output, with impedance of 100 ohms. It goes low to indicate that the Ministep is busy (for example, during a movement), and it is released when the drive is idle. It can be used to quickly detect the end of a programmed positioning, for example.

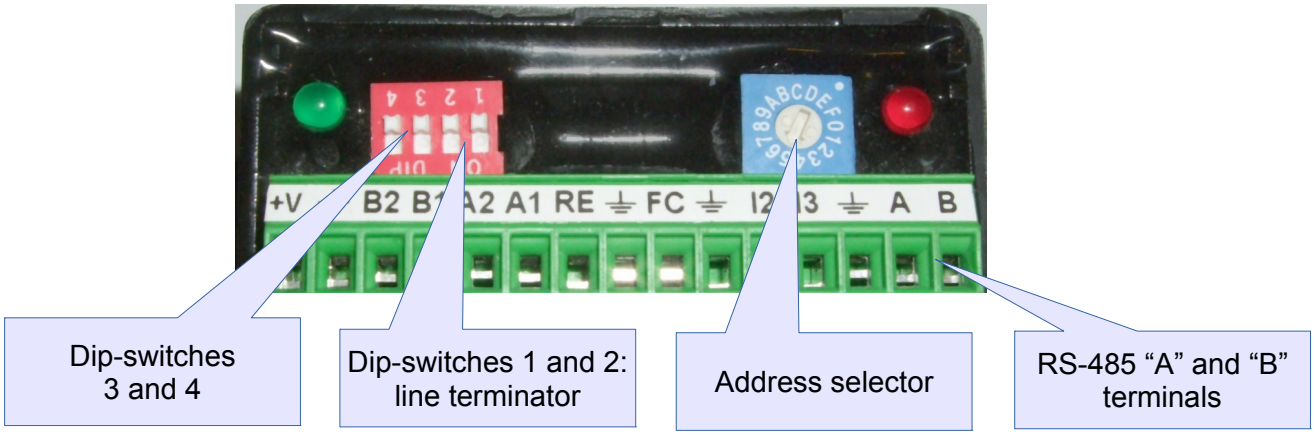
Digital (24 volts) inputs



The three digital inputs (I1..I3, also called X1..X3) have a 5 Kohm input impedance and a protection circuit for the internal components. They must be driven by positive voltage from outside in order to activate (i.e., left open they are in *off* state). The functions of these inputs are as follows:

1. FC (X1) General purpose / F.C. switch (limit switch) / Low current (Step mode)
2. I2 (X2) General purpose / Encoder channel "A" / Step command (Step mode)
3. I3 (X3) General purpose / Encoder channel "B" / Direction (Step mode)

RS-485 serial interface and Dip-switches



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. Ministep already contains two selectable termination resistors, enabled by the dip-switches 1 and 2 as seen in the following table:

Switch 2	Switch 1	Termination	Comment
OFF	OFF	*Absent*	Multidrop configuration, when this Ministep 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	ON	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).

Dip switches 3 and 4

Dip-switch 3 is used to reset the communication parameters, when modified while the module is powered up. It is otherwise ignored.

Dip-switch 4, when set to ON, makes Ministep enter the Step/Direction mode. The mode can be commanded (and canceled) also via RS-485, so the normal use of this dip-switch is to make the Ministep power on in Step/Direction mode.

Configuring Ministep

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. It is possible to recall 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, Ministep 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. Ministep 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 Ministep 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. Ministep 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 Ministep 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. Ministep replies with “OK”, or “Error 1” if an error has occurred.

Other settings via Modbus

Ministep can save to its internal memory further several settings about the motor to drive. These settings comprise:

Parameter	Adr	Description
MAXSPEED	93	Top speed for movements (steps / s) (default 800)
MINSPEED	94	Minimum speed (steps / s) (default 0)
ACC	95	Acceleration rate (steps / s^2) (default 1600)
DEC	96	Deceleration rate (steps / s^2) (default 1600)
CURRHOLD	97	Driving current with motor stopped (mA, default 300)
CURRRUN	98	Driving current in top speed (mA) (default 1000)
CURRACC	99	Driving current during acceleration phase (default 1000)
CURRDEC	100	Driving current in deceleration phase (default 1000)
MODSPD	101	Speed switch point for full step mode (default 1000)
STEPMODE	104	Configuration of various parameters (default 0x2082)

At any time, it is possible to save the current configuration (of parameters above), by writing 1 to the discrete coil 162.

Commanding Ministep from a PC/PLC

Step/Direction mode

In this mode, every falling edge of the signal fed to I2 (Input 2) will produce a *logical step* of the motor, in the direction set by I3 (low level=+, high level=-). This Step/Direction mode is entered via a command by RS-485, or when dip-switch 4 is ON. The type of stepping, as well as other parameters, can be set up at any time via RS-485, or can be programmed once – the Ministep can store the settings in its flash memory. Steps are made using CURRRUN current, but CURRHOLD can be selected by applying voltage to X1 input.

RS-485 mode (always active)

In this mode, it is possible to set up parameters, execute movements with speed profiles (ramps) and electrical currents in different phases of the trip. Movements can be commanded with absolute or relative position, or free run, and there are also some special form of movement like “Go home”, “Go to FC switch”. The module can be read to know for example the position of the motor, or the state of the inputs, or the value of the (optional) connected encoder.

The Ministep 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.

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 an internally diagnosed failure, or by a failure in the communication with the PC/PLC: if this latter 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.

Modbus interface

Ministep 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 Ministep 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 following instead are *virtual* bits used to know some status of the module:

X4	*On* ("1") when the motor is rotating – 0 otherwise.
X5	*On* when the motor is not rotating
X6	On when the motor is rotating in "+" direction (position increments)
X7	On when the motor is rotating in "-" (reverse) direction
X8	On if the motor is in "Hiz" state: the outputs are disconnected and the motor has no torque.
X9	Busy flag; it is On when Ministep is executing a command or a ramp (it is not on if the motor is in free running mode)
X10	Signals that the motor is in Home position (value = 0)
X11	Signals that the motor is in the POSMARK position (a writable register)
X13	Signals that a step loss has been detected
X14	Signals a drop in the power supply
X15	OCD (Over current detected)
X16	Thermal warning: the module is heating too much

The three inputs X1 X2 X3 are cooked, i.e. there is a programmable latency on edge detection, and all 3 inputs have a counter associated with them.

Input conditioning

When used as GPIO (general purpose I/O), the 3 digital inputs are sampled by Ministep 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).

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 3 milliseconds.

Input counters

Ministep 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.

Other discrete inputs

The following table describes a few virtual inputs, readable in order to expose some internal state of the drive:

Parameter	Adr	Description
PENDEVENTS	17	There are events readable via Modbus function ***Collect***
GENERICFAIL	19	A generic failure has been detected and noted
ALWAYS0	20	This bit is always 0 (for read testing purpose)
ALWAYS1	21	This bit is always 1
FLASHERR	22	The flash memory failed to store the configuration
WDTFBACKS	25	Watch dog fired because of feedbacks
WDTCOMMTM	26	Watch dog fired because of communication timeout
COREDRVFAIL	29	Internal error diagnosing the power output section
DIPSW3	128	Value of dip-switch 3
DIPSW4	129	Value of dip-switch 4

Input Registers (read-only)

These registers are 16-bit values, that can be read but not written (as per Modbus specification). The module maps all the discrete inputs in the same address space of these registers, so all the Discrete Inputs can also be read via 16-bit registers. The following table shows the used addresses:

Parameter	Adr	Description
INPUTS	0	The 16 inputs X1..X16 described above
ROTSELECTOR	9	The value of the rotary address selector
VALIM	12	The voltage of the power supply

Single Coil Registers

Ministep uses Single Coils to perform several commands. Refer to the table below:

Parameter	Adr	Description
RUNPLUS	0	Set to 1 to start free running in positive direction, or set to 0, when active, to stop rotation.
RUNMINUS	1	As before, for reverse direction.
STOP	2	Set to 1 to stop rotation when in free-running mode.
GOHOME	3	Set to 1 to bring the motor to HOME (position 0).
GOSTEPDIR	4	Enables the Step/Direction mode through X2/X3.
GOSWITCH	5	Perform a positioning on the switch (X1) input.
RELSWITCH	6	Perform a switch release (exit from X1).
HIZ	7	Detach the output bridge (frees the motor – no torque).
SWRMINUS	9	Sets reverse direction for GOSWITCH/RELSWITCH.
SWRZERO	11	Sets operation for GOSWITCH/RELSWITCH.
STPLOSS	12	Signals a step loss detected; can be reset.
VALIMLOW	13	Signals a power supply brownout; can be reset.
OVCURR	14	Signals an over-current detected; can be reset.
THSHUTDOWN	15	Signals an overheating condition, which shut the power section off; can be reset.

Holding Registers

Like the Input Registers, the Holding Registers are overlapped to the Single Coil Registers. So, the bits of the previous paragraph can be read or set in a single Modbus operation at address 0. In addition, there are other registers used to set up parameters and make movements. The following table lists the relevant items:

Parameter	Adr	Description
YWORD	0	Correspond to the first 16 ***Single Coils***.
XCOUNT1..3	11..13	Rising edge count of the 3 inputs. Can be zeroed.
XLATUP1..3	19..21	Latency for rising edge of the 3 inputs
XLATDN1..3	35..37	Latency for falling edge of the 3 inputs
RELPLUS (Y1PULSE)	81	(16 bits unsigned integer) When written, makes an incremental move in positive direction, of the amount specified
RELMINUS (Y2PULSE)	82	(16 bits unsigned integer) When written, makes an incremental move in negative (reverse) direction.
POSMARK	87-88	(***LONGINT***) Position (coordinate) of MARK position.
POSMOT	89-90	(***LONGINT***) Actual position of the motor (axis). By writing a new position to this parameter, a movement is commanded.
POSPRESET	91-92	(***LONGINT***) Presets the axis. By writing a new value, POSMOT changes accordingly without moving the motor.
MAXSPEED	93	Maximum speed in steps/s for the motor movements.
MINSPEED	94	Minimum speed in steps/s
ACC	95	Acceleration rate (steps / s^2) (default 1600)
DEC	96	Deceleration rate (steps / s^2) (default 1600)
CURRHOLD	97	Driving current with motor stopped (mA, default 300)
CURRRUN	98	Driving current in top speed (mA) (default 1000)
CURRACC	99	Driving current during acceleration phase (default 1000)
CURRDEC	100	Driving current in deceleration phase (default 1000)
FSSPEED	101	Speed switch point for full step mode (default 1000)
STEPMODE	104	Configuration of various parameters (default 0x2082)
ENCODER	105-106	(***LONGINT***) Current value of the optional encoder.

Note: the parameters marked as (***LONGINT***) are 32-bit signed integers in little endian format: the lower 16 bits are stored in the lower (Modbus) address, followed by the higher-order 16 bits in the next address.

Moving the motor

In order to correctly manage the axis (i.e., the motor), the following steps has to be executed.

First, set up parameters

Choose suitable values for currents, speeds, and ramps. This is done by writing to the Holding Registers of the previous chapter. At power on, Ministep loads a set of parameters (either the factory defaults, or those last saved by the user). These parameters are used for every movement, and should be changed only when the motor is not turning. It is very well possible to make different movements (i.e., with different speeds or currents) one after another – simply change the parameters between movements. These parameters never change by themselves, and are used and reused every time a movement is initiated. The relevant parameters are MAXSPEED, MINSPEED, ACC, DEC, CURRHOLD, CURRRUN, CURRACC, and CURRDEC.

Second, assign a position (coordinate) to the axis

Stepper motors are intrinsically *digital* and *discrete*: the drive controls primarily their position, and only marginally their torque. Speed is controlled as a direct consequence of controlling their position. Movements of such motor are better thought as changes in position (coordinate): the axis will respect automatically the assigned destination in an open-loop configuration.

That said, after a power on it is advisable to move slowly the motor to a recognizable physical position, signaled by a limit switch for example, or push softly the axis toward a mechanical stop; once the known physical position is reached, a coordinate must be assigned to the axis by writing the desired value to the POSPRESET parameter. After this preset, the axis can be moved everywhere with precision by stating directly the new desired position.

Third, make the axis move

There are different ways to make a movement using the last active parameters (speed, ramps, currents):

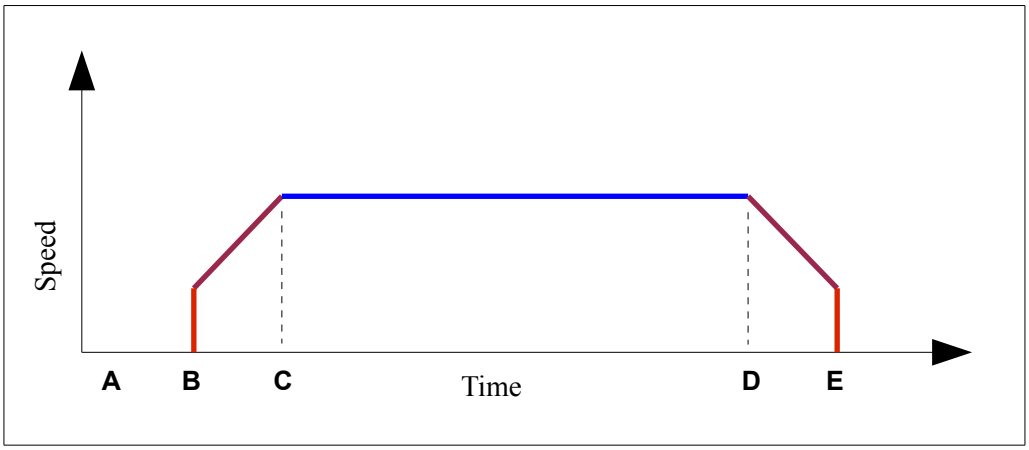
- New destination: simply write the new desired (absolute) position to the POSMOT register. The motor will ramp up, travel at maximum speed for the correct time/space, then ramp down, and finally stop in the new commanded position.
- GOHOME: an equivalent way to indicate 0 (zero) as new destination.
- Incremental move: write the number of steps to move in the RELPLUS register, to make the axis go the destination POSMOT+RELPLUS (in positive direction). Similarly, write the value to the RELMINUS register to make the axis goto POSMOT-RELMINUS new position.
- Free run: the motor can turn freely. In such situation the drive still tracks the real position but, simply, no destination is specified – the motor has to be stopped by another command. Write a 1 in the single coil RUNPLUS to make the motor rotate in positive direction, or write 1 to RUNMINUS coil to make it rotate in reverse. Write 1 to the STOP coil to make it decelerate and stop. It is possible to set RUNPLUS while RUNMINUS is active, or vice-versa: the axis performs a deceleration followed by an acceleration in the commanded direction (i.e., no abrupt change is made).
- Use GOSWITCH command to reach the switch (**if present**) connected to input X1. First set (or reset) SWRMINUS to indicate the direction to go (0=+, 1=-), and SWRZERO (see later). Then write 1 to GOSWITCH. The axis will initiate a normal movement, until the switch is signaled (X1 goes high). At that point, a stop is performed. If SWRZERO was 1, the POSMOT register is

cleared (zero axis); if it was 0, POSMOT is not touched but, instead, copied into the POSMARK register.

- Use RELSWITCH (*release* switch). The motor will use the minimum speed to leave the switch (X1 goes to 0). Again, SWRMINUS tells the direction of the movement, and SWRZERO tells to clear POSMOT or set POSMARK. This function is much more precise than the previous one: a low speed is used and the motor is stopped suddenly (without ramp).

Anatomy of a movement

Stepper motors are also peculiar because they need ramps to change speed, as any other motor, but they also have a relatively low speed, that they can reach *instantaneously*. It is possible to take advantage of this by setting MINSPEED to a value different than 0 (but not too high, also depending on other conditions: for example, say 200 steps/s or less). In this case where MINSPEED is greater than 0, the following is the speed profile used to do a movement:



In **A** time, the motor is still. When a movement is commanded, in **B**, it starts to rotate at MINSPEED steps per second *immediately* and, also, begins to increase the speed until the MAXSPEED is reached in **C**. The top speed is held until it's time to decelerate; in **D** the speed is decreased until MINSPEED is reached, and then the motor stops suddenly. In other words, the motor will never rotate slower than MINSPEED steps per second. Different output currents (i.e. output torques) are used in the different phases. When the motor is still, in **A** and after **E**, CURRHOLD is used. This value should be fairly low, to avoid heating of the motor, but still enough to hold the system in place. During the MAXSPEED phase, from **C** to **D**, CURRRUN is used. This current must be chosen high enough to win the friction of the system, and the gravity (if movement is not horizontal). During acceleration (**B-C**), CURRACC is used; this current can be equal to CURRRUN but, if desired, it can be presumably higher taking in account the inertial mass of the system, which must be accelerated. During deceleration, quite the opposite happens: friction aids, but the inertial mass obstacles. The correct values can be planned ahead of time, but some field test is almost always required.

Too much current imposed to a stepper motor can make it stall (this is a peculiarity of such motors), or simply make them noisy. For this reason, driving currents must be planned and tested with care.

The Modbus function Collect

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 Ministep 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.

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.

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. Ministep 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 Ministep 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; Ministep, 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 Ministep; see later a table with all the known identifiers. If the packet is well formed and the identifier is valid, Ministep 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, Ministep 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.

Device

Returns name and version of the device; currently is “Ministp3 1.2”.

Address

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

Flags

Reads a mask of few bits about the internal status of the device. See Other discrete inputs for the meaning of the single bits.

Valim

Returns the value (in volts) of the power supply, as seen by the internal ADC converter.

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). Refer to Digital inputs for an explanation of the bits.

X1 .. X16

(ex. ?X1): return the status of the single input specified: 1=on, 0=off. Refer to Digital inputs section for an explanation.

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).

Yword

Replies with a 16 bit number, as seen for Xword, which is the status of the outputs. See Single Coil Register for details.

Y1 .. Y16

Shows the status of the indicated output: 1=on, 0=off. In Ministep, all the outputs are virtual – they are not associated to real pins. See Single Coil Register for details.

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 Ministep 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.

X1latup .. X3latup

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

X1latdn .. X3latdn

The same as before, for the falling edge.

Pos, Mark

Read the content of POSMOT and POSMARK register, related to the position (coordinate) or the axis. Refer to Holding Registers for a deep explanation.

Encoder

The optional encoder, connected to inputs X2 and X3, tracks the axis position independently from the module. By reading Encoder, it is possible to verify the *real* mechanical position, in contrast with the internal one, which is calculated in open loop, and hence in particular situations can be inaccurate (because of step losses, for example).

Maxspeed, Minspeed, Accel, Decel

Read the parameters for the speed profile when moving. Refer to Holding Registers for a deep explanation.

Currhold, Currrun, Curracc, Currdec

Read the values, in mA, of the currents used in different moments of a movement. See Anatomy of a movement for details.

FSspeed

Reads the speed over which the full-step mode is used. Under this speed, microstepping can be used if configured.

MotMode

A word indicating several configurations for the drive.

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, Ministep replies with "OK".

Address, Serline, RdelayT, RdelayM, Wconf

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

Yword

Sets all the (virtual) outputs, using the specified number as a bit mask. For example, ">YWORD=1" activates the output Y1 and turns off all the others. In order to correctly use these identifier, refer to Single Coil Register.

Y1 .. Y16

Activates or deactivates a single (virtual) output: ">Y2=1" turns on the second output, making the axis move in negative direction, while ">Y3=1" stops the movement. After the "=" sign, only a single 0 or 1 digit is valid. Refer to Single Coil Register.

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, Ministep 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).

Wflags

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

Xcount1 .. Xcount3

These 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.

Y1pulse, Y2pulse

By writing a value greater than 0 in Y1pulse, a relative, positive movement is made. For example, ">Y1pulse=200" will move the motor in direction + for 200 (logical) steps. Y2pulse does the same, but in reverse (negative) direction.

Mark

Sets a value (position) for the POSMARK register. Normally it is not needed to write this register.

Pos

By writing a value into this register, a movement is commanded to reach the new requested position.

Preset

Assign a position to the axis, without moving it. This operation is commonly called "preset".

Encoder

The optional encoder, connected to inputs X2 and X3, tracks the axis position independently from the module. If this identifier is written to, the effect is to *preset* the register.

Maxspeed, Minspeed, Accel, Decel

These parameter are set to shape the speed profile while making a positioning of the motor (i.e., almost any movement). Refer to Holding Registers for a deep explanation.

Currhold, Currrun, Curracc, Currdec

Set the values, in mA, of the currents used in different moments of a movement. See Anatomy of a movement for details.

FSspeed

Sets the speed over which the full-step mode is used. Under this speed, microstepping can be used if configured.

MotMode

A word indicating several configurations for the drive.

Moving motor with text commands

After having set up the desired parameters (refer to the relevant sections), all the movements can be triggered by **setting one of Y1..Y16** virtual outputs and writing to the POS register; the status of the system can be examined by reading the same outputs and/or the X1 .. X16 inputs. The following tables sum up the operations.

Y1 .. Y16 operations

Operation	Name	Description
RUNPLUS	Y1	Set to 1 to start free running in positive direction. If read, indicates that the motor is running in positive direction.
RUNMINUS	Y2	As before, for reverse direction.
STOP	Y3	Set to 1 to stop rotation when in free-running mode. When read, it is 1 if the motor is stopped.
GOHOME	Y4	Set to 1 to bring the motor to HOME (coordinate 0).
GOSTEPDIR	Y5	Enters the Step/Direction mode.
GOSWITCH	Y6	Perform a positioning on the switch (X1) input.
RELSWITCH	Y7	Perform a switch release (exit from X1).
HIZ	Y8	Detach the output bridge (frees the motor – no torque).
SWRMINUS	Y10	Sets reverse direction for GOSWITCH/RELSWITCH.
SWRZERO	Y12	Sets operation for GOSWITCH/RELSWITCH.
STPLOSS	Y13	Signals a step loss detected; set to 0 to reset.
VALIMLOW	Y14	Signals a power supply brownout; set to 0 to reset.
OVCURR	Y15	Signals an over current; set to 0 to reset.
THSHUTDOWN	Y16	Signals an overheating condition, which shut the power section off; can be reset, after enough time has passed to let the device cool down.

X1 .. X16 status reading

Parameter	Name	Description
X1	X1	Reflects the state of input I1 (X1)
X2	X2	Reflects the state of input I2 (X2)
X3	X3	Reflects the state of input I3 (X3)
RUNNING	X4	On (1) when the axis is moving
STOPPED	X5	On (1) when the motor is stopped
RUNPLUS	X6	On when the motor turns in positive direction
RUNMINUS	X7	On when the motor turns in negative (reverse) direction
HIZ	X8	At 1 (on) when the motor is electrically detached
BUSY	X9	On when Ministep is executing some command. It is not on when the axis is in free running mode.
ATHOME	X10	At 1 (on) when the axis is at position (*coordinate*) 0
ATMARK	X11	On (1) when the motor is in the POSMARK position
STPLOSS	X13	Signals a step loss detected
VALIMLOW	X14	Signals a power supply brownout
OVCURR	X15	Signals an overcurrent
THWARN	X16	Signals an initial overheating condition. This is only a warning; when the temperature rises further too much, the device protects itself by disconnecting the power outputs.

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

Please report errors or imprecisions to web@xonelectronics.it

Buy Ministep online: www.xonelectronics.eu