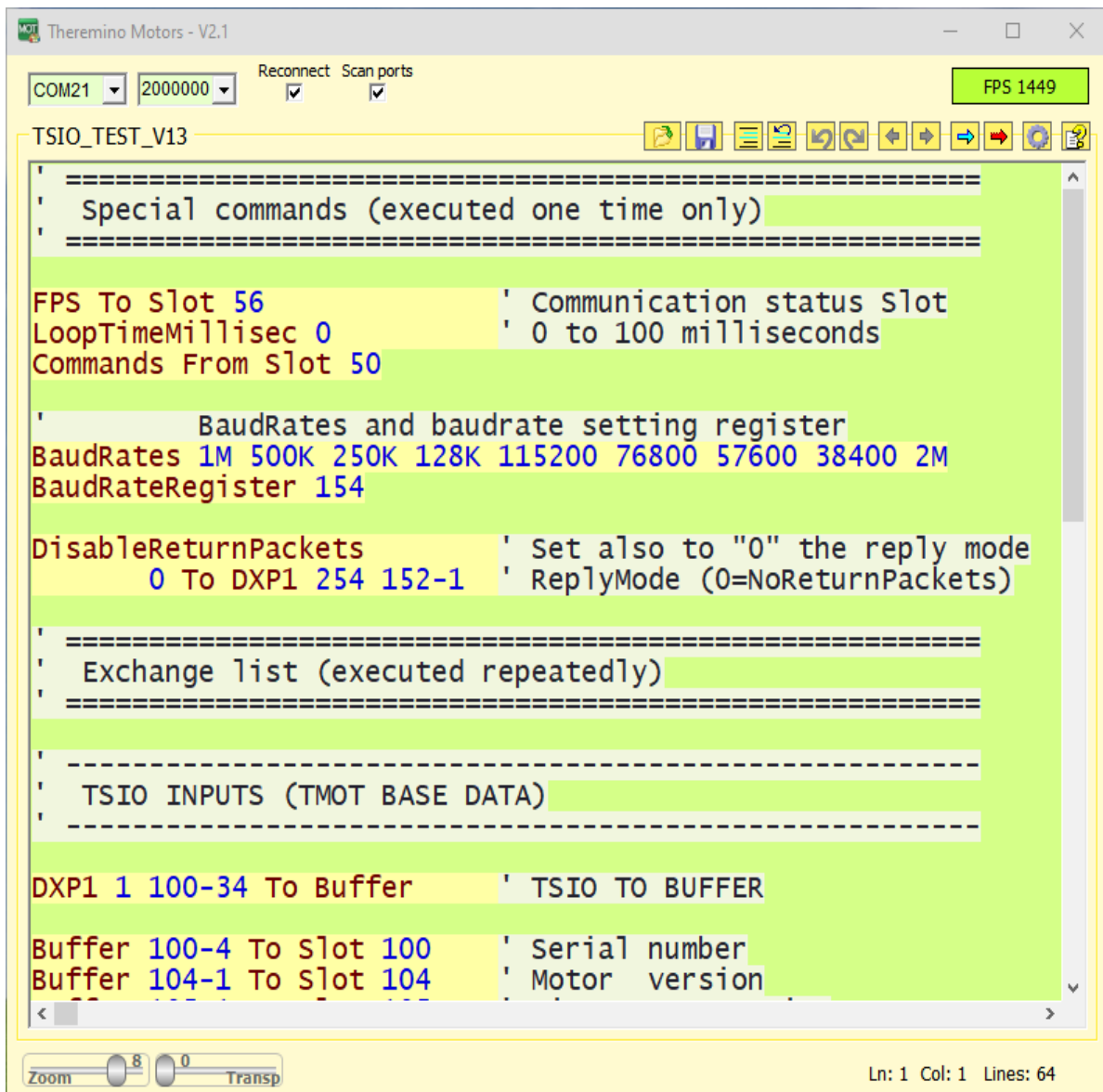


theremino System



Theremino Motors V3.1

The Theremino_Motors application

This application puts the slots of the theremino system in communication with devices connected to a [Half-Duplex](#) serial line (RS485 with two balanced wires, or TTL with a single wire).

The devices are connected in chain by means of a four-wire cable (two signals plus power supply and GND) or even just three wires in the TTL versions.



In this image you can see some [FeeTech](#) and [Dynamixel](#) motors, and a TSIO module opened to show the connections.

The Smart-Motors contain all the control electronics, a 4096-step encoder and a configurable PID algorithm that allows you to control the rotation with a precision of 0.09 degrees (and the TMOT with double encoder up to 0.003 degrees).

Speed, acceleration and torque can also be adjusted, as well as reading the reached position, temperature, current (which is related to torque) and many other parameters.

Devices of different types can be connected on the same communication line, for example three TMOTs in the base, shoulder and elbow to be able to lift 1 Kg at one metre, then three 3215 by FeeTech in the tip, which are small and light, to move pincers and screwdrivers, and finally a TSIO on the tip to control tools.

(Year 2022) The TSIO (Theremino Smart Input Output) modules are already available which provide more than twenty ADC, PWM, DigIN and DigOUT In-Out Pins. The TSIOs can be easily connected on the same data and power line as the Feetech, Dynamixel and TMOT motors.

(Year 2023) TMOT (Theremino Motors) are being prepared, real SmartMotors for industrial controls but at a highly competitive price. TMOTs have better torque control than FeeTech and Dynamixel. Furthermore, it is possible to identify and reprogram the TMOTs without having to open the Robots and disconnect them from the communication chain.

Characteristics of servo motors

On this page we report the characteristics of some servomotors that we found particularly interesting.



We tried the top five on this list and definitely recommend the top three (3032, 3215 and 3046) which, in addition to being very economical, also run smoothly and quietly.

Motor	Voltage	Gears	Speed (RPM)	Torque (kg-cm)	Encoder bits	Size (mm)	Weight (g)	Euro	Processor
STS3215	4V-7.4V	345:1	54	19	12	45*25*35	55	10	GD32F130F8P6TR
STS3032	4V-7.4V	205:1	111	4	12	32*12*27	21	20	GD32F130F8P6TR
STS3046	6-7.4V	378:1	52	40	12	40*20*43	89	25	
SM29BL	12-24V	241:1	110	24	12	40*28*42	102	50	STM32F030K6T6
SM45BL	12-24V	353:1	110	25	12	46*28*34	100	90	STM32F030K6T6
SM120BL	9-25V	232:1	50	120	12	78*43*65	485	420	
eRob70	48V		40	367	19	70*81	1000	550	

The data in this table are approximate, consider them as a rough indication to roughly compare the motors. The millimeters are rounded to the nearest integer and the prices are those found in Europe. Prices in China are 30..40% lower.

All the motors considered here are coreless or brushless and the encoders are magnetic or optical. We have previously discarded all models with brush motors and potentiometer feedback.

The indicated "torque" value is the stall value. Under normal working conditions it must be kept considerably lower.

Communication protocols

The protocol used by this application is the [Dynamixel 1.0](#) (abbreviated to DXP1) which allows communication with all servomotors [Dynamixel](#) is [FeeTech](#).

There is also a protocol [Dynamixel 2.0](#) but we didn't implement it because it would only work with some Dynamixel models (MX and PRO) and with none of the FeeTechs. Furthermore, the 2.0 protocol does not contain any particular advantages. Its only major improvement would be the SyncRead instruction, which is superseded by our implementation of [Buffered transfers](#)

Use DXP1 or Modbus protocols

FeeTech servos can be programmed to use either the DXP1 protocol or the ModBus protocol. The latter is familiar to PLC users, but has lower speed performance and fewer commands.

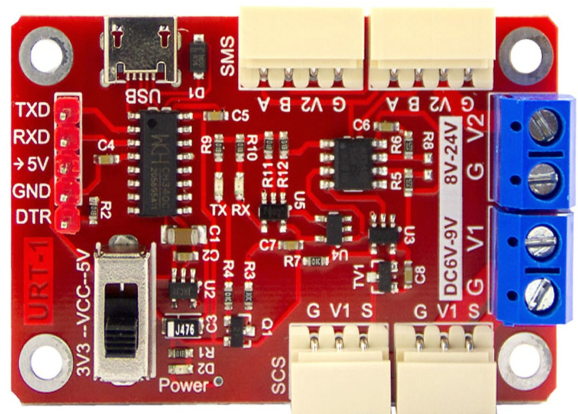
To use Modbus devices we have written an application very similar to this one, but called [Theremino_Modbus](#) In its page you will also find the related documentation files.

Program FeeTech servo motors for DXP1 or Modbus

We have prepared in [This Page](#) a ZIP archive containing everything needed to reprogram the FeeTech servos with the two protocols. You will also find programming instructions in the ZIP file.

To reprogram the FeeTech motors you will need a USB connection module, like the one in the image on the right, a USB cable and a connection cable for the motors.

You will also need a 24 volt power supply which connects with the negative to terminal "G" and with the positive to V2.



The power supply must have a convenient power switch located near the PC keyboard and the Mouse. With the switch you have to give power exactly at the same time you press the programming button on the software. If this action is not done synchronously, the programming will not start.

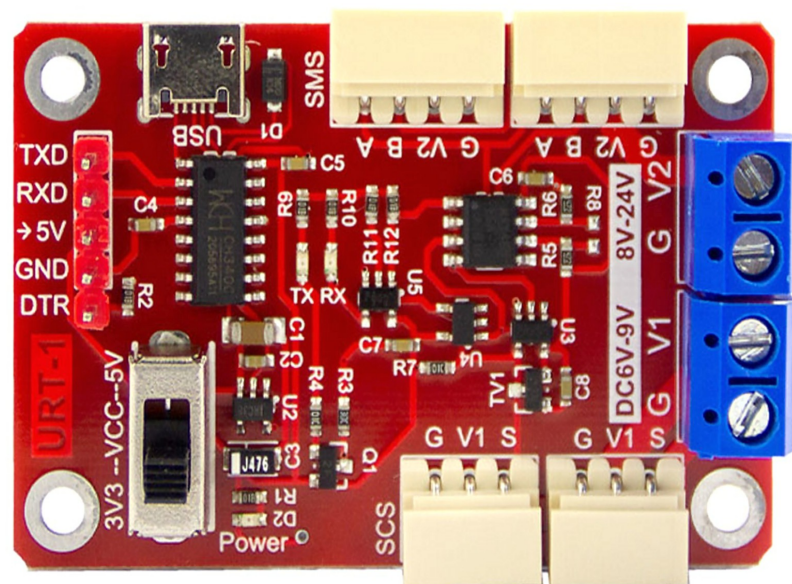
To make the Dynamixels work you could use this same adapter, but not we still tried.

The Feetech URT1 board

With this card you connect the motors to the USB port and power them with the blue connectors on the right.

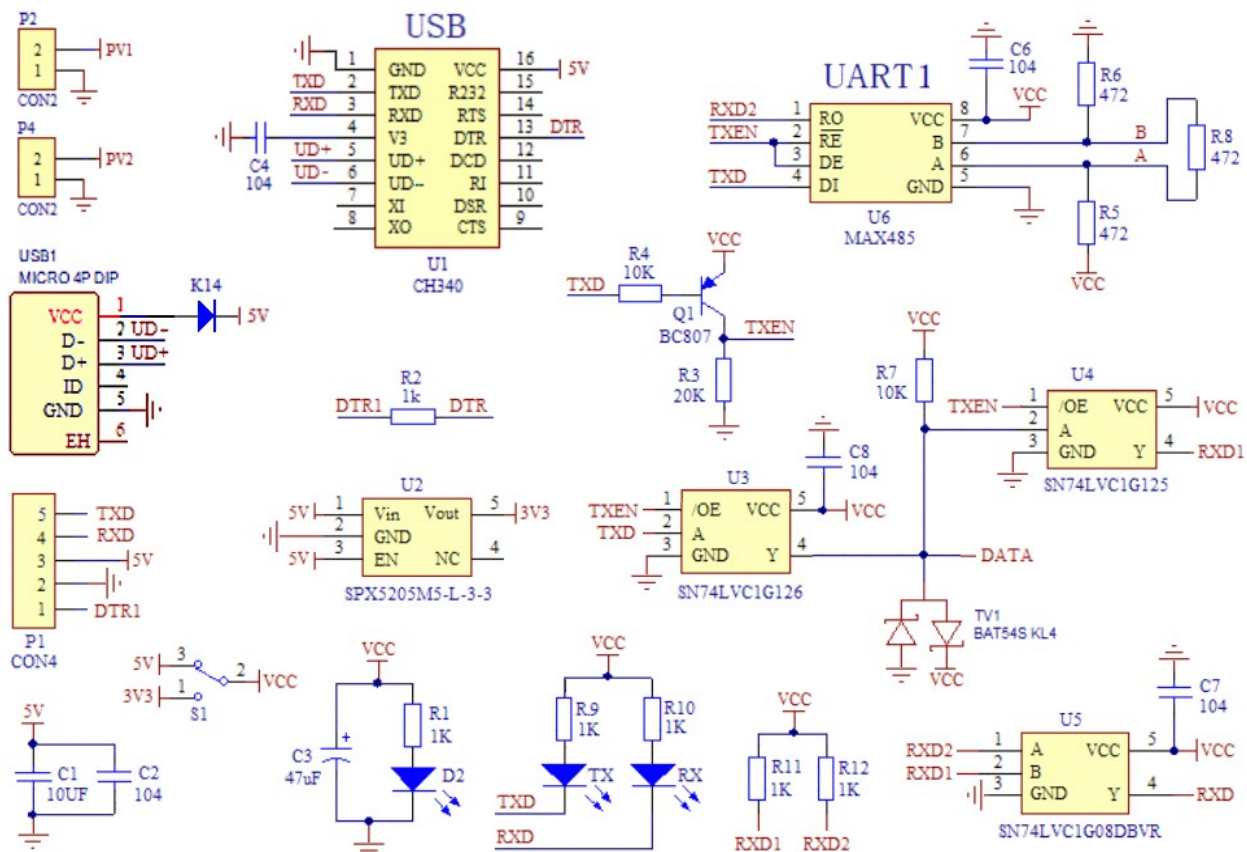
Motors with four wires connect to the top connectors and feed 8 to 24 volts on connectors G and V2

Motors with three wires plug into the connectors below and feed 5 to 7 volts on connectors G and V1



The low voltage motors (we recommend the excellent 3032 and 3215) can be powered by connecting the 5 volts arriving from a second USB port (preferably USB3) to G and V1, or from a 5 or 6 volt power supply and at least 2 or 3 amps.

Wiring diagram



Edit the command list

In the central part of the application there is the list of commands. The colored parts on a yellow background are active and without errors. The parts in light green are commented and do not act.

```
'=====
'=====
'  DEVICE TYPE TSIO  (Theremino Smart InOut module)
'=====
DeviceType TSIO1
0 To DXPI BC 152-1 ' ReplyMode (0=NoReturnPackets)
'-----
'  Exchange list (executed repeatedly)
'-----
'  TSIO INPUTS (TMOT BASE DATA)
'-----
DXPI 1 100-34 To Buffer ' Bytes 100 to 133 >>> Buffer
Buffer 100-4 To Slot 100 ' Serial number
Buffer 104-1 To Slot 104 ' Motor version
Buffer 105-1 To Slot 105 ' Firmware Version
Buffer 106-1 To Slot 106 ' Firmware Subversion
Buffer 107-1 To Slot 107 ' Bootloader Version
'-----
'  TSIO INPUTS (PINS)
'-----
Buffer 110-2 To Slot 110 F03 ' ADC1 PA07
Buffer 112-2 To Slot 112 F03 ' ADC2 PA06
Buffer 114-2 To Slot 114 F03 ' ADC3 PA04
Buffer 116-2 To Slot 116 F03 ' ADC4 PA03
Buffer 118-2 To Slot 118 F03 ' ADC5 PA02
Buffer 120-2 To Slot 120 F03 ' ADC6 PA09
Buffer 122-2 To Slot 122 F03 ' ADC7 PA10
Buffer 124-2 To Slot 124 F03 ' ADC8 PA11
Buffer 126-2 To Slot 126 ' DigTn1 PA25
```

Everything written in this list becomes immediately operational. Every time even a single character is changed, the whole list is checked again and the parts without errors are immediately activated.

For which **be careful when editing** commands not to accidentally create commands that write to registers other than those desired.
Or disable the communication as explained on the next page.

```
'-----
'  Write with option
'-----
Slot 11 To DXPI 0 42-2 E ' Write Destination
```

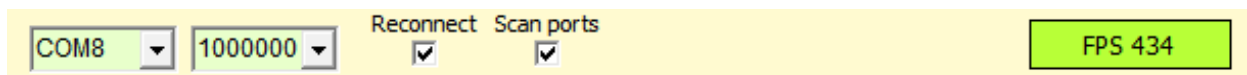
In case of errors the command turns red. Here you see a line containing an error, the "E" option which does not exist.

Disable communication

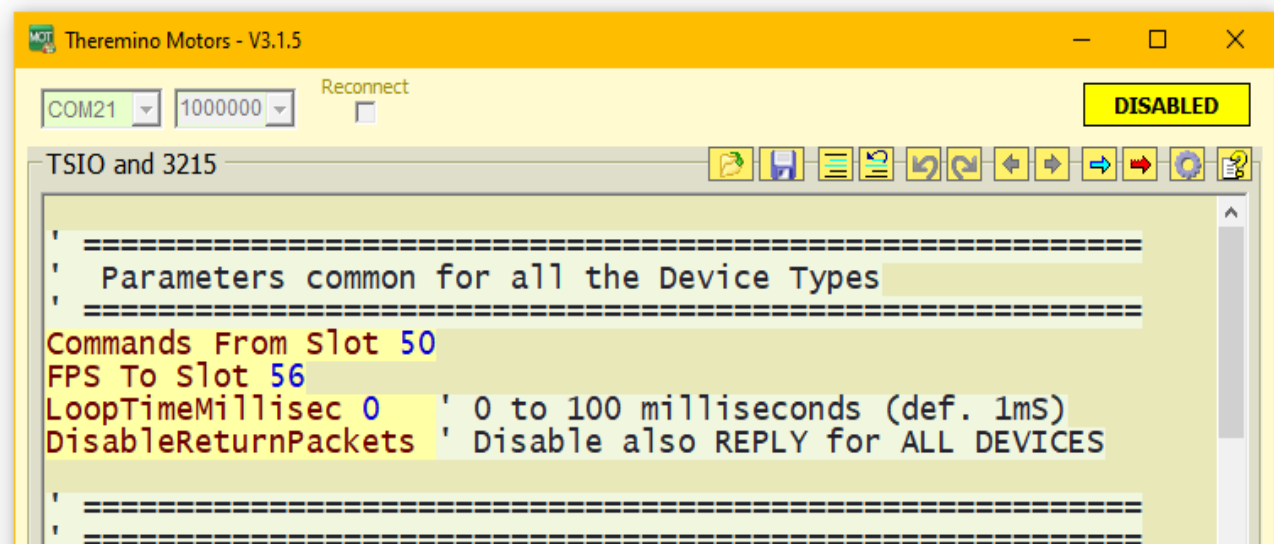
Everything you write in the command list becomes immediately operational. Every time even a single character is changed, the whole list is checked again and the parts without errors are immediately activated.

This is handy for quickly testing the effect of variations, for example when adjusting PID values. But if you are not careful it could happen that you write to other registers without meaning to. For example, if you write to the identifier register, then the motors will no longer work and you will have to disconnect them and reset their identifier one at a time.

Therefore, during substantial changes, it is advisable to disable communication. You could remove power from the motors but it is more convenient to press the button at the top right with the mouse that indicates the communication speed in FPS.



When the button is pressed, the message "DISABLED" appears on a yellow background and the communication port commands are disabled.



The command list also changes color to highlight the disablement.

After finishing the changes, press the button again which will return to mark the speed on a green background.

Command List (DXP1 Protocol)

Initialization commands

Run once, at startup and when editing program text

LoopTimeMillisec nnn	' Delay to slow down the execution loop
SectionSelectorSlot s	' Setting the Sections Slot
nnn To DXP1 d r-b	' "Immediate" number to a register
StepsPerUnit d n.nnn	' See the special commands on the next pages

Device type initialization

It must precede all other commands related to a certain device

DeviceType Feetech	' Device type
---------------------------	---------------

Write commands

Performed continuously, as often as possible

Slot s To DXP1 d r-b	' Read the Slot and write in a register
Slot s To DXP1 d r-b A	' Always write (A = Always)
Slot s To DXP1 BC r-b	' Write to all devices (BC = Broadcast)

Read commands

Performed continuously, as often as possible

DXP1 d rb To Slot s	' Read 1 to 4 bytes and send them to a Slot
DXP1 d rb To Buffer	' Read from 1 to 253 bytes and send them to buffer
Buffer rb To Slot s	' From 1 to 4 bytes from the buffer to a Slot
Buffer rb To Slot s	' 1 to 4 bytes from the buffer to another Slot

Sections setting command, also executed continuously

Section nnn	' Start marker of a section
--------------------	-----------------------------

Meaning of abbreviations

nnn	= Number (eventually with decimal point and decimals too)
s	= Slot (1 to 999)
d	= Device identifier (0 to 199) (BC = "Broadcast")
r	= Device register (0 to 255)
b	= Number of bytes that make up the register (from 1 to 4)

Special commands

These commands are used to initialize the device with fixed values, they are mainly used to adjust the speed of the serial port "Baud Rate", the minimum and maximum movement limits and the PID parameters.

These commands are sent only once when the application is started and are sent again each time any character in the program is changed.

These transfers they do not affect the number of exchanges per second (FPS) because they act only once and with very short times. Therefore, you don't have to worry about using them carefully and limiting their use to the essentials as you should do with all other transfers.

Examples of special commands

DeviceType Feetech	' Device type
LoopTimeMillisec nnn	' Delay to slow down the execution loop
SectionSelector_Slot s	' Setting the Sections Slot
StepsPerUnit d n.nnn	' Unit of measurement setting (see next page)
StepsPerDegree d n.nnn	' Unit of measurement setting (see next page)
StepsPerMM d n.nnn	' Unit of measurement setting (see next page)
Commands From Slot s	' External commands reception Slot
FPS To Slot s	' Slot for the measured FPS value
DisableReturnPackets	' Instruction to disable the replies waiting

If you do not write, or comment, the lines that set the Slots then the related functions will not be performed.

Among the special commands, which are executed only once, there are also the transfers of the "Immediate" numbers, which will be explained in more detail on the next pages.

Example of "Immediate" transfer

nnn To DXP1 d r-b	' "Immediate" number to a register
--------------------------	------------------------------------

Meaning of abbreviations

nnn	= Number
s	= Slot (1 to 999)
d	= Device identifier (0 to 199) (BC = "Broadcast")
r	= Device register (0 to 255)
b	= Number of bytes that make up the register (from 1 to 4)

Measurement units and the "C" option

If the commands on this page are not used then the position data exchanged with the motors are the raw values, measured by the angular encoder located in the motor, i.e. very large integers.

By means of the following commands it is possible to use more convenient and significant units such as degrees or millimeters, depending on whether angular or linear movements are used.

You could also use your preferred units of measurement, for example kilometers, inches, yards, feet, etc. In these cases you will use the generic "StepsPerUnit" command and note in the comments which unit you are working with.

The following three commands have different names to highlight which unit you are using, but they do exactly the same calculations. The actual unit of measurement depends only on the numbers that are set in the numeric field "n.nnn".

StepsPerUnit **d n.nnn**

StepsPerDegree **d n.nnn**

StepsPerMM **d n.nnn**

The first value "d" defines the motor and can go from 1 to 199, in this way each motor can have different reduction ratios.

The second value "n.nnn" can be an integer or with decimals. This number must be calculated, based on the resolution of the motor and the mechanical reductions, in order to obtain the desired unit of measurement.

The "C" option

To use the units of measurement, all the commands that read or write position values (those usually called "Goal position" and "Actual position") must be marked with the letter "C" (unit).

nnn To DXP1 d r-b C ' "Immediate" number to a motor register

Slot s To DXP1 d r-b C ' From a Slot to a motor register

DXP1 d r-b To Slot s C ' From a motor register to a Slot

Buffer r-b To Slot s C ' From the buffer to a Slot

Without the "C" options then the "StepsPerXXX" commands have no effect.

The "C" option is explained in the next page.

The "C" and "M" options

Option "C" (calibrations)

To use the zero calibration (and the units of measure) you must mark with the letter "C" (calibrations) all the commands that read or write position values (those usually called "Destination" and "Actual position").

The "M" (mirror) option

The "M" option (mirror which in Italian means "to mirror") is used to reverse the direction of rotation of a motor.

This option is used when one of the motors is mounted on the opposite side to the others, or when different motor models are used which rotate in the opposite direction to the one desired.

Use the "C" and "M" options only on position commands.

The commands for speed, temperature, acceleration, enabling, adjustments, etc. they must not have the "C", otherwise you would get unpredictable behavior.

The options from "F00" to "F99" (special functions)

The options from "F00" to "F99" (special functions)

The functions from F00 to F07 and F12 are currently defined, all the others up to F99 are free for future needs.

Functions F00, F01 and F02 convert the numerical formats of speed, torque and current of FeeTech motors.

Function F03 converts the number formats of the ADCs of the TSIO modules.

Function F04 converts the numerical format of the temperature of the TSIO modules.

Functions F05 and F06 convert the numerical formats of the PWMs of the TSIO modules.

The F12 function converts the position values of the FeeTech motors into multi-revs which otherwise only go from 0 to 4095 and then restart from zero at each new rev.

The details of the special functions are explained in the next pages.

The command options from F00 to F02

Currently the only function command options are F00, F01, F02, F03, F04, F05, F06 and F12.

The F00 function

This function decodes the Velocity parameter of the FeeTech motors which, despite being two bytes long, does not follow the normal coding with the negative numbers from 32768 to 65535.

Add F00 only to FeeTech engine lines that read the "Velocity" value.

The F01 function

This function decodes the Torque parameter of the FeeTech motors which, despite being two bytes long, does not follow the normal coding with the negative numbers from 32768 to 65535, and makes a further inversion for the numbers from 1024 to 2047.

Add F01 only to FeeTech motor lines that read the "Torque" value.

The F02 function

This function recalculates the Current parameter of the motors and makes it available in Milliampere.

This function works only if preceded in the code by the MilliAmperePerUnit command which is written as in this example:

MilliAmperePerUnit 1 6.5

This example works with FeeTech motors and gets the MilliAmperes consumed by the motor by multiplying the current value of motor 1 by 6.5.

Only add F02 to FeeTech engine lines that read the value "Current".

The command options from F03 onwards

The F03 function

This function decodes the values arriving from the ADCs of the TSIO module. The ADCs are 12 bit and produce integer values from 0 to 4095 but by adding this option the values become the classic floating point values of the Theremino system ranging from 0 to 1000.

Add F03 only to lines of TSIO modules that read ADCs.

The F04 function

This function decodes the temperature value of the TSIO module. The TSIO module sends the temperature as an integer representing degrees centigrade multiplied by one hundred, but adding this option you get a convenient value in degrees centigrade with decimals up to hundredths of a degree.

Add F04 only to the row of TSIO modules that reads the temperature.

The F05 function

This function encodes the classic values from 0 to 1000 of the theremino system towards the PWM of the TSIO module. The TSIO module for PWM requires integer numbers from 0 to 65535, but by adding this option you can use the classic values from 0 to 1000 of the theremino system.

Add F05 only to the PWM lines of TSIO modules.

The F06 function

This function is identical to the previous F05 but performs a logarithmic conversion which allows the LED to turn on more gradually.

Add F06 only to the PWM lines of TSIO modules intended for the gradual switching on of the LEDs

The F12 function

This function decodes the Actual Position parameter of the FeeTech motors, which despite being two bytes long does not return the multi-revolution rotation but is limited to only one revolution from 0 to 4095 and then repeats both for the negatives and for the positives with module 4096 (12 bit).

Add F12 only to lines that read the current position.

Registers tables

Each device has different characteristics and even the control tables are not all the same. Therefore it is always advisable to consult the documentation of the individual devices.

FeeTech Servo Motors

Fortunately FeeTech servo motors have only one table that applies to everyone. The differences in the behavior of the individual motors are of little importance and the registers are always the same.

So we were able to collect all the tables in a convenient PDF file that you download from [This Page](#)

We had to correct the negative values of all the two Byte FeeTech registers, both reading and writing, with the following formula: $n = -32768 - n$

All the examples you will find on the next pages
use the FeeTech motor register scheme.

For Dynamixel servos the examples have to be adapted
to the tables of the single motor, correcting the numbers of the registers.

TMOT Servo Motors and TSIO modules (in preparation in 2023)

The TMOT (theremino motor) and TSIO (Input-Output module) devices have register tables different from the Feetech ones. You can find them on [this page](#).

Dynamixel Servo Motors

Dynamixels are less controllable than FeeTechs and cost a lot more.

We haven't checked them all but Dynamixels from the same group (XL, XC, XM, XH, XW, AX, EX, DX, RX, MX, PH, PM, L, M, H) should have almost identical tables.

Dynamixel documentation is convenient and well specified for each motor, so we have not prepared any documentation on them.

Go to consult the characteristics and registers to use for each motor, you can find them all on this page: <https://emanual.robotis.com/docs/en/dxl>

On the left there is a menu with the categories of the motors, click to open them and inside you will find all the motors with their images.

Then clicking on the single motor opens a page that contains everything about it, technical specifications, registers to use and advice for communication.

Immediate transfers

These commands are used to initialize the device with fixed values, they are mainly used to adjust the "Return-Delay" response time, the minimum and maximum movement limits and the PID parameters.

Immediate transfers are sent only once when the application is started and are sent again whenever any character in the program is changed.

These transfers they do not affect the number of exchanges per second (FPS) because they act only once and with very short times. Therefore, we do not have to worry about using them carefully and limiting their use to the essential, as we must do with all other transfers.

Examples of immediate transfers

0 To DXP1 BC 7-1	'Immediate value to Return Delay
0 To DXP1 BC 9-2	'Immediate value to MinPositionLimit
0 To DXP1 BC 11-2	'Immediate value to MaxPositionLimit
32 To DXP1 BC 21-1	'Immediate value to P
32 To DXP1 BC 21-1	'Immediate value to D
32 To DXP1 BC 21-1	'Immediate value to I
500 To DXP1 BC 48-2	'Immediate value to MaxTorque

Note that in these examples the target device is **BC** ie "Broadcast" which sends the same value to all connected devices. With the "Broadcast" method you save from repeating the instruction for all devices and you get the added benefit of not having to change the program even if you add or remove devices.

- - -

With an immediate transfer you could also control the destination of a single servo motor. This normally doesn't make much sense but it could be useful, in some cases, to make sure that the motor is positioned at a predetermined point when starting.

Example of command that initializes the zero device position

1000 To DXP1 0 42-2	'Immediate value to Destination
----------------------------	---------------------------------

Transfers

These commands transfer numerical values from the Theremino system Slots to the device registers and vice versa.

The following example reads the numerical value of Slot 12 and writes it in four consecutive bytes starting from address 42 of device 3.

Slot 12 To DXP1 3 42-4

- ◆ The first part **Slot 12** defines the Slot (could be from 0 to 999)
- ◆ The word **To** indicates direction (from Slot to DXP1)
- ◆ **DXP1** defines a device that uses the Dynamixel V1 protocol
- ◆ The number after DXP1 defines the device. In this example the device is the **3** but it could be from 0 to 199.
- ◆ The number 200 to 254 cannot be used to indicate the devices because they would specify "broadcast" sending, ie a simultaneous sending to all connected devices.
- ◆ The number 255 cannot be used because it is used by the protocol as a transmission start signal.
- ◆ The last part **42-4** indicates to write 4 consecutive bytes starting from address 42.

Examples of transfers from a Slot to a device

Slot 22 To DXP1 0 22-2 From Slot 22 towards bytes 22 and 23 of device zero.

Slot 22 To DXP1 BC 22-2 From Slot 22 towards "**BC**" ie all connected devices.

Slot 22 To DXP1 0 22-2 A Sending with special option "A" (Always)

Examples of transfers from a device to a Slot

DXP1 2 22-4 To Slot 22 Bytes 22, 23, 24 and 25 of device 2 to Slot 22

DXP1 3 40-2 To Slot 22 Bytes 40 and 41 of device 3 towards Slot 22

Buffered transfers

The transfers that slow down the communication the most are the log readings.

Usually the registers you are interested in are many, for example Position, Speed, Motion Indicator, Temperature, Voltage, Current, etc ... If you read all these registers one by one you have to wait for the answer each time and this can slow down the speed in an unacceptable way. And if there are more than one servos, you can even reach a few exchanges per second and the movements become fragmented, with continuous stops and starts.

When writing registers this problem can be solved by using the SYNC method, which sends data to numerous registers without waiting for responses, but the Dinamixel protocols do not provide efficient methods for reading the registers.

So we've added the ability to read several bytes with a single response packet. The bytes are accumulated in a buffer from which the data of the individual registers can be extracted, at very high speed and without spending time in communication.

To extract the data of a register from the buffer, the address of the register and its size in bytes (1, 2, 3 or 4 bytes) must be indicated

Buffered read example

DXP1 0 56-15 To Buffer 'Read 15 bytes (56 to 70) to buffer

Buffer 56-2 To Slot 100	'Position	bytes 56 and 57 from buffer to Slot 100
Buffer 58-2 To Slot 101	'Velocity	bytes 58 and 59 from buffer to Slot 101
Buffer 60-2 To Slot 102	'Torque	bytes 60 and 61 from buffer to Slot 102
Buffer 62-1 To Slot 103	'Voltage	bytes 62 from buffer to Slot 103
Buffer 63-1 To Slot 104	'Temperatures	bytes 63 from buffer to Slot 104
Buffer 64-1 To Slot 105	'Sync Flag	bytes 64 from buffer to Slot 105
Buffer 65-1 To Slot 106	'Hard.Error	bytes 65 from buffer to Slot 106
Buffer 66-1 To Slot 107	'Moving	bytes 66 from buffer to Slot 107
Buffer 69-2 To Slot 108	'Current	bytes 69 and 70 from buffer to Slot 108

"Broadcast" transmissions

For the Broadcast commands to work, the **DeviceType** command must already be defined. It is therefore good practice to always specify the DeviceType at the beginning of each section relating to a certain type of device.

You can send commands with the same value, to all connected devices, and these commands are called "Broadcast" -

These commands save lines of code as well as transmission time. They are also commands that do not require a response, so they are practically instantaneous.

The "Broadcast" commands are used mainly for initializations, but can also be used during the continuous exchange of information, for example to change the same setting on all devices at the same time.

The "Broadcast" commands are just write commands, because reading from many devices at the same time would cause the data to collide.

To send a "Broadcast" command to all devices, the special number 254 is used instead of the device identifier.

Examples of broadcast transmissions

0 To DXP1 BC 7-1	'Immediate value to Return Delay
0 To DXP1 BC 9-2	'Immediate value to MinPos Limit
0 To DXP1 BC 11-2	'Immediate value to MaxPos Limit
32 To DXP1 BC 21-1	'Immediate value to PID-Proportional
32 To DXP1 BC 22-1	'Immediate value to PID-Derivative
32 To DXP1 BC 23-1	'Immediate value to PID-Integral

With this series of instructions, which is executed only once at start-up, all connected devices are initialized in the same way.

Example of a broadcast transmission that moves all the motors

Slot 100 To DXP1 BC 42-2

This instruction moves the destination of all connected servos. By varying the value of Slot 100 all servos will move together.

COM port settings

COM8	1000000
------	---------

With the two boxes at the top left you can choose the serial port and the communication speed.

COM8	128000
COM1	1000000
COM17	500000
COM18	250000
COM13	128000
COM12	115200
COM8	76800
COM3	57600
	38400

Usually the maximum speed is used but if the cable is very long it can be limited.

To locate the port you disconnect and reconnect the USB cable and each time you close and reopen the box on the left.

COM1	1000000
------	---------

If the port does not work, or is already in use, the two boxes turn red.

```
'=====
'=====
'  DEVICE TYPE FeeTech (STS3215 and other motors)
'=====
'=====
DeviceType FeeTech
```

To send the baud rate to the devices it is essential that the **DeviceType** command has been defined at the beginning of each section relating to a certain type of devices. Always remember to specify the DeviceType at the beginning.

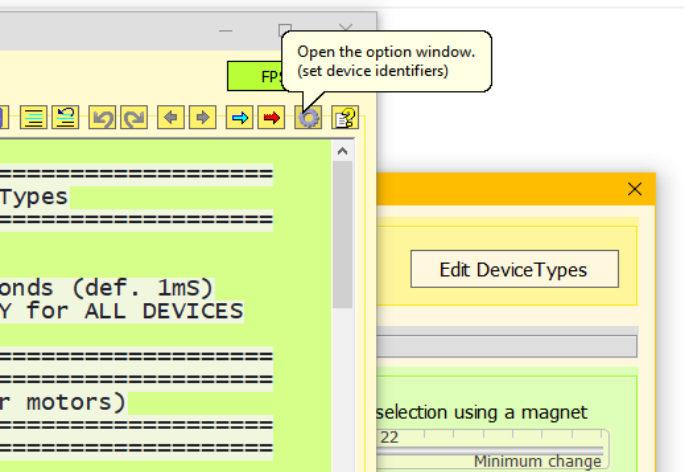
At each start, and each time any character in the program is changed, the communication speed is sent again to all servos. So the box on the right should turn green and indicate good FPS speed.

COM8	1000000	FPS 434
------	---------	---------

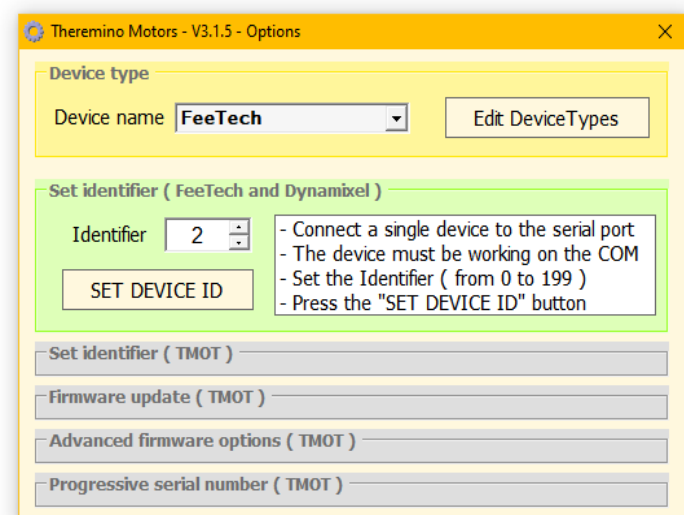
If the box on the right does not turn green, or if it flashes and indicates very low FPS, then you will need to check the communication lines with the servos and perhaps also program the identifiers of the servos, as explained on the following pages.

Note that at least one command that expects a response packet must be used, otherwise the text "Disconnected" will appear instead of the FPS

Options Panel



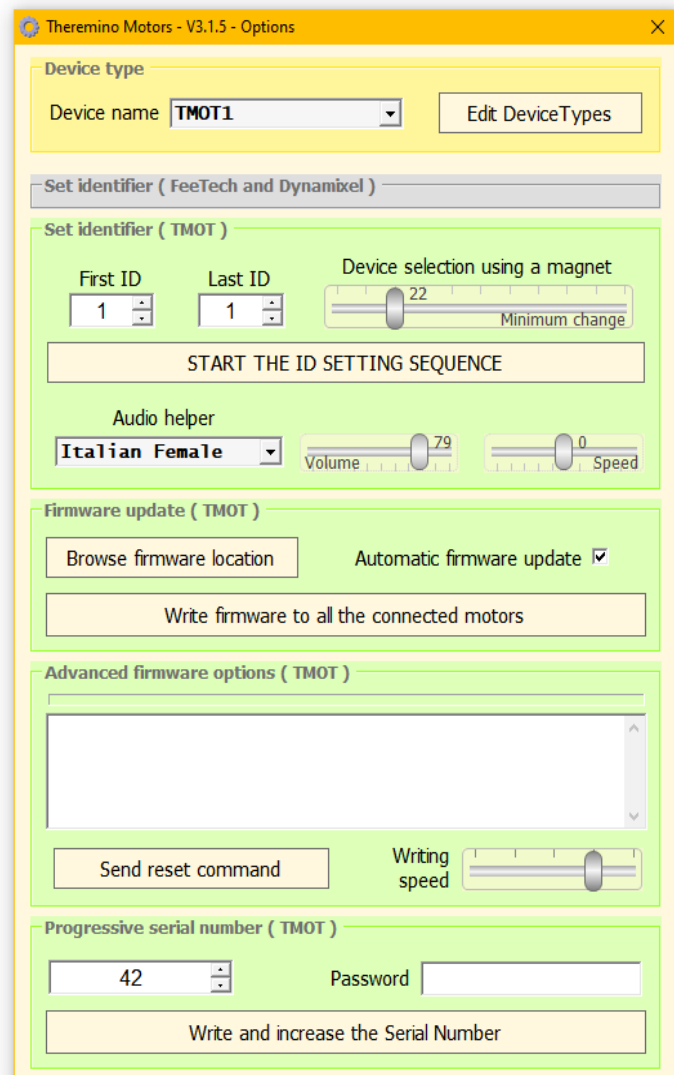
The gear opens the options window to assign the identifier to all devices and also to update the firmware and assign the serial number to theremino system devices (TMOT and TSIO).



The first panel at the top sets the device type.

If you choose a FeeTech or a Dynamixel then only the first panel is enabled to give the identifier to these engines. For this operation you have to disconnect them from the line and connect them one at a time.

If instead you choose a TMOT or a TSIO you will enable four panels with which you assign the identifiers, update the firmware and set the serial number to the devices of the theremino system (TMOT and TSIO). And all this very conveniently, without having to disconnect the devices from the common line.



Locate connected devices

Anyone who assembles or modifies devices made up of several motors often finds themselves in difficulty because they do not know the identifier and the BaudRate of the motors they are connecting. And, sadistically, not knowing them, it is not possible to reprogram them with the appropriate identifier and BaudRate.

Until now the only possible method to obtain this information was to use FeeTech's "FT Servo Debug" application, setting different BaudRates and pressing "Open", "Search", "Stop" and "Close" repeatedly, until randomly identifying the configuration of one or more engines and display them in the list.

Device types

Device type selected

TMOT1

Edit DeviceTypes

Test Devices

Baud rates

ALL

ID range

1 to 9

TEST

1	500K	TSIO
2	500K	STS3215

To facilitate these operations, the latest versions of Motors contain the new "Test devices" panel.

Just press the "TEST" button and in a few seconds all the devices are identified and appear in the list, each with its own ID and BaudRate.

Performing a complete search of all devices for all possible BaudRates could take a few minutes, so we have prepared numerous options to narrow down the search to the most common areas and reduce the search time to seconds.

With "ACTUAL" the IDs of the motors are only searched for with the currently used BaudRate. With "VALID" only the valid BaudRates for the currently configured motors are tested. With "ALL" all possible BaudRates are tested. With "100K..5M" we try those from 100K up and with "1K..100K" we try only those from 100K down. Finally with 2M, 1M, 500K, 250K, 128K, 115200, 76800, 57600, 38400 and 9600 the individual speeds are tested.

Restricting the search to just the first nine devices drastically decreases the search time.

In all common cases a fast and effective search is obtained with: "Baud rates = ALL" and "ID range = 1 to 9"

Test Devices

Baud rates

ALL

ACTUAL

VALID

ALL

100K..5M

1K..100K

2M

1M

500K

250K

128K

115200

76800

57600

38400

9600

ID

22

selection using a magnet

Minimum change

Test Devices

Baud rates

ACTUAL

ID range

1 to 9

1 to 9

1 to 19

1 to 199

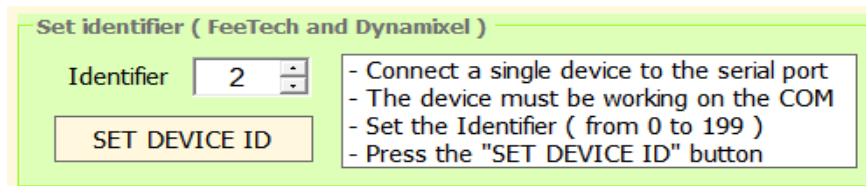
1	500K	TSIO
2	500K	STS3215

Set the device identifier

If you connect two or more devices on the same line and with the same ID, communication errors occur. In these cases, the communication speed may drop significantly and the indicator may cause the word "Disconnected" to flash.

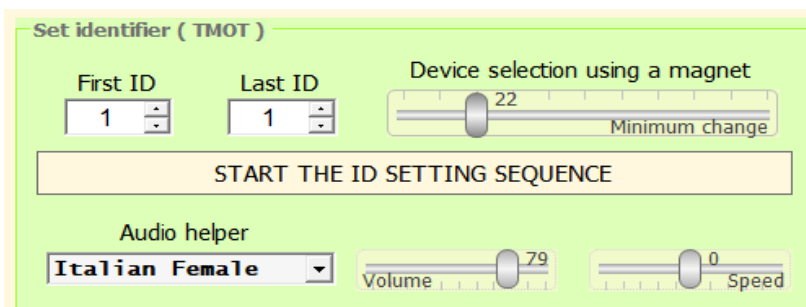
Therefore, to use more than one device it is necessary to prepare them by connecting them one at a time and assigning a different Identifier to each of them.

Feetech and Dynamixel motors



- ◆ Connect a single device and make sure it is communicating.
- ◆ If it doesn't communicate, use the Feetech and Dynamixel software to set it.
- ◆ Choose a different identifier from 0 to 199 for each motor.
- ◆ Press the SET DEVICE ID button.

TMOT and TSIO devices

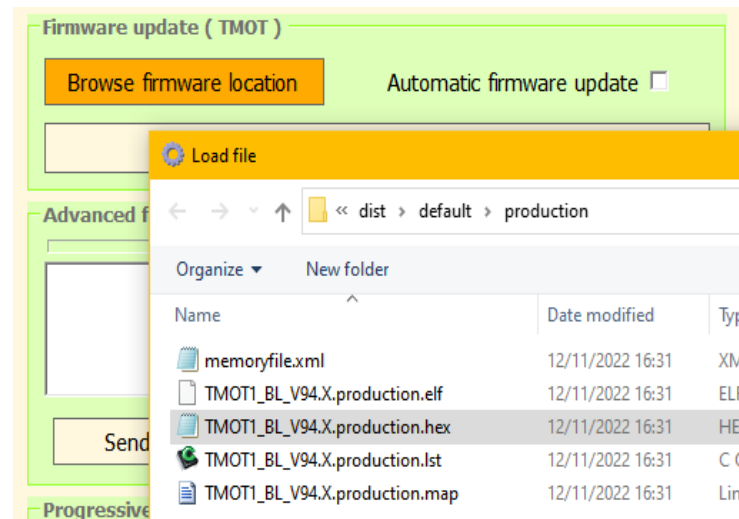


- ◆ All TMOTs and TSIOs can remain connected to the common line.
- ◆ Set FirstID and LastID and then press "Start the id setting sequence".
- ◆ Follow the voice prompts and identify the devices, one after the other when prompted, by holding a magnet close to each one.
- ◆ If the sensitivity is too low, you have to get the magnet too close, while if it is too high, you risk selecting another device by mistake.
- ◆ If you make a mistake you can stop the procedure by pressing the button again and start again.

Update the firmware of the devices

The TSIO modules and the TMOT engine have additional possibilities compared to FeeTech and Dynamixel. One of the most important is the ability to update the firmware through the serial line. And it's very convenient to be able to do it without having to disassemble the robot to connect one device at a time.

- Press "Browse firmware location"
- Choose the "hex" file for the devices to be programmed.
- Press the button "Write firmware.."
- If you keep "Automatic firmware update" enabled then writing starts automatically every time you change the hex file. Thus, programmers can try the changes immediately every time they compile the program.



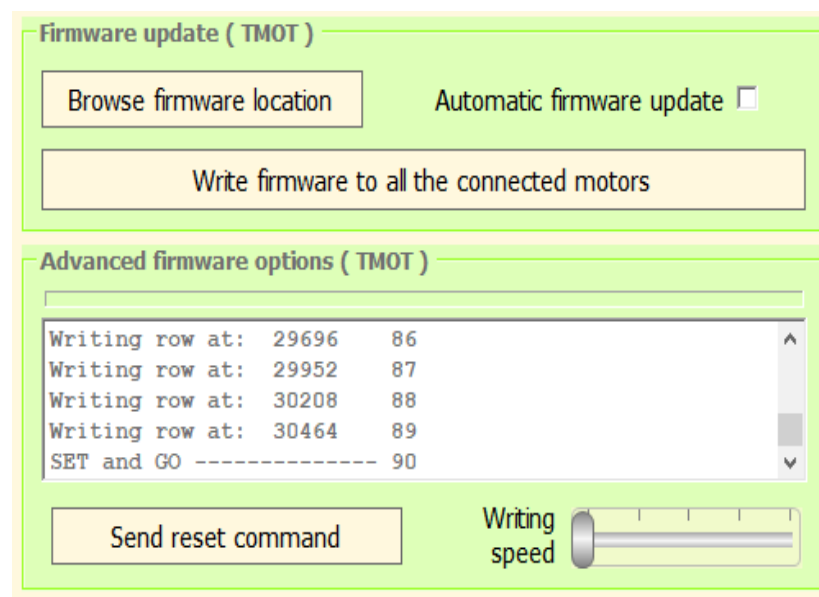
All devices of the same type connected to the data line (RS485 or TTL) will be programmed at the same time. To program others with the same firmware, just connect them and press the button, without having to choose the file again each time.

During programming, the list shows the blocks that are being sent.

We recommend using programming speed "4" because in some cases at speed "5" communication errors could occur.

The "Send reset command" button can be useful for testing.

Remember that the TSIO and TMOT devices restart at 115200 Baud so after the reset you will need to use the BaudRate box or leave the Reconnect option enabled.



Set the serial number

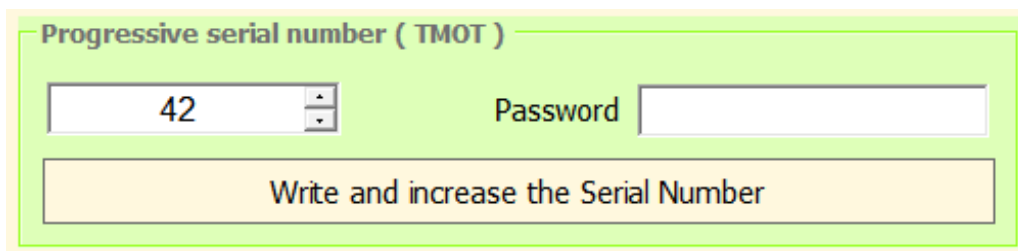
The TSIO modules and the TMOT engine have additional possibilities compared to FeeTech and Dynamixel.

A convenient and useful option is to set the serial number via the serial communication line.

Naturally this setting will not be made by users but in the production phase, so there is also the possibility of setting a password to prevent subsequent changes.

A repairer or the manufacturer could, knowing the password, change the serial number.

If you don't know the password, the only way to change the serial number is to erase all the program memory and start from scratch, writing the Boot-Loader with a programmer.



Progressive serial number (TMOT)

42 Password

Write and increase the Serial Number

The operation of this panel is simple:

- Connect one device at a time.
- Set the serial number.
- If necessary, set also the password.
- Press the button.

The serial number is automatically incremented so every time you just plug in a device and press the button.

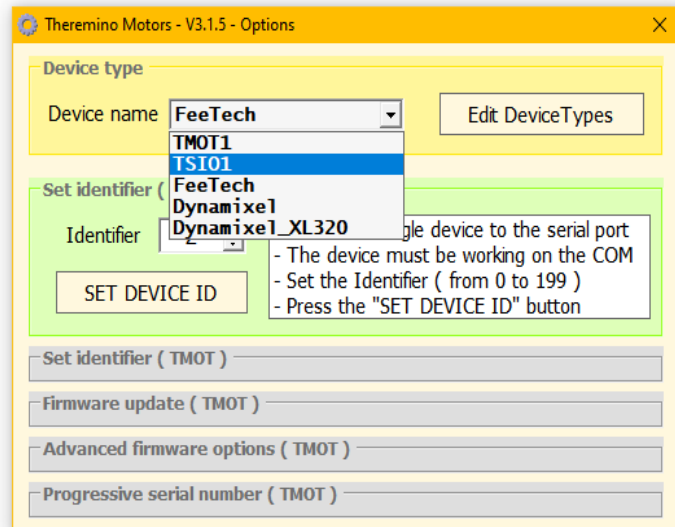
These operations are also useful during production to check that the device works and communicates regularly.

Device configuration

The "Theremino_Motors_CONFIG" file must contain valid values for each type of device that is used.

With the "Edit Device Types" button you can check the values written in the file and possibly modify them or add new devices. If this button is used the file is then automatically reloaded from the Motors application.

The values from the file are used by the options panel commands. Remember to choose the device to send commands to, as seen on the right.



Some commands of the main program (Broadcast and BaudRate) also use the values written in the configuration file. For their functioning it is essential that each program zone starts with the name of the device, as in this image:

```
' =====  
' =====  
' DEVICE TYPE TSIO (Theremino Smart InOut module)  
' =====  
' =====  
DeviceType TSIO1
```

Configuration examples for FeeTech and TSIO devices

```
DeviceName      FeeTech ' 3215 and other motors  
Broadcast       254  
RegisterID      5  
RegisterBaudRate 6  
RegisterWriteEnable 55 ' Enable EEPROM writing  
RegisterWorkingMode --- ' Reset / SetID  
RegisterIDtolerance ---  
RegisterSerialAndPwd ---  
BaudRates 1M 500K 250K 128K 115200 76800 57600 38400
```

```
DeviceName      TSIO1 ' Theremino Smart IO  
Broadcast       250  
RegisterID      55  
RegisterBaudRate 54  
RegisterWriteEnable 50  
RegisterWorkingMode 51 ' Reset / SetID  
RegisterIDtolerance 56  
RegisterSerialAndPwd 80  
BaudRates 1M 500K 250K 128K 115200 76800 57600 38400 2M
```

Enable writing of the EEPROM

Both FeeTech and Dynamixel servos have a memory location in which to write a zero to enable writing to the EEPROM. Normally the value of this location is "1" and the EEPROM is blocked.

If the EEPROM is blocked then when some parameters are changed it only acts in the RAM memory and this setting is then lost as soon as the supply voltage is turned off.

Commands that automatically enable EEPROM

The communication speed setting box (Baud Rate) and the "Identifier setting" located in the options panel, automatically enable the EEPROM and disable it at the end, reading which registers to use from the file "Theremino_Motors_CONFIG.txt"

Commands that DO NOT automatically enable EEPROM

The commands that are written in the main list act only in the temporary memory unless they are preceded by a line that enables the EEPROM.

It is advisable to group all the commands to be written in the EEPROM at the beginning and to precede them with an enabling line.

And it is recommended that this group of commands be followed with an EEPROM disable line.

See the example on the next page.

Enable and disable the EEPROM

In this example we see the first lines that are normally used. These lines set all motors to speed up communication (zero Return Delay and no Return Packets). In addition, in these lines they also ensure that the range of motion is as desired (in this case from 0 to 4095). Other commands could be added between these lines.

0 To DXP1 BC 55-1	' Zero to enable EEPROM writing
0 To DXP1 BC 7-1	' Zero to Return Delay
1 To DXP1 BC 8-1	' ReplyMode (0=NoReturnPackets)
0 To DXP1 BC 9-2	' 0 to MinPos
4095 To DXP1 BC 11-2	' 4095 to MaxPos (0=NoLimit)
1 To DXP1 BC 55-1	' 1 to disable EEPROM writing

DisableReturnPackets ' <- If ReplyMode = 0 add this line !

It is important to note that the first line **0 To DXP1 BC 55-1** enables writing to the EEPROM and that the last line **1 To DXP1 BC 55-1** disables it again.

If you set ReplyMode to zero you must also add a command,
as explained in the next page.

*It is important to disable the EEPROM after the last command that has to write in it.
This reduces the risk of changing important parameters by mistake.
For example if you changed the motor ID then you would have to reset all the IDs.*

- - -

This is an initialization line so you put it between the first lines of the program, indeed it is advisable to place it right at the beginning so that you can see it well and be sure that it is there.

This example is valid for all FeeTech servos that have the lock register in position 55.

For Dynamixel servos, instead, it is necessary to consult the characteristics of the individual devices. In the next few lines we collected the register numbers of some servos. However we recommend that you always check the characteristics table of your motor.

Dynamixel servos using register 24: XL-320, AX-12, AX-18, EX-106 DX-113, DX-116, DX-117, RX-10, RX24F, RX-28, RX-64. MX-12W, MX-28, MX-64, MX-106

Dynamixel servos using register 64: XL-430, XC430, XM430, XM540, XH430, XH540

Finally, it is necessary to pay attention that the Dynamixels use the same memory location both to enable the EEPROM (with value zero) and to enable the moto (with value one). Therefore, after setting the identifier, it is also necessary to set value "1" to make the motor work again.

The Reply Mode

FeeTech servos have the "Replay mode" option which is useful for speeding up communication.

To use it, you must first enable writing to the EEPROM and then send a "ReplyMode = 1" command which disables the sending of confirmation packets.

In the next sequence we see the commands that are normally sent to the servos to initialize them.

0 To DXP1 BC 55-1	' Zero to enable EEPROM writing
0 To DXP1 BC 7-1	' Zero to Return Delay
0 To DXP1 BC 8-1	' ReplyMode (0=NoReturnPackets)
0 To DXP1 BC 9-2	' 0 to MinPos
4095 To DXP1 BC 11-2	' 4095 to MaxPos (0=NoLimit)
1 To DXP1 BC 55-1	' 1 to disable EEPROM writing

DisableReturnPackets ' <- If ReplyMode = 0 add this line !

It is always advisable to leave the writing on the EEPROM commented to avoid accidentally changing the IDs of the devices.

*In fact, it could happen to send by mistake a command with identifier **BC** (therefore broadcast) on the ID register and then write identical IDs on all the servos. In this case you will have to disconnect all the cables and reset the IDs by connecting one motor at a time.*

- - - - -

The line that deletes the return packets is as follows:

1 To DXP1 BC 8-1	' ReplyMode=1
------------------	---------------

DisableReturnPackets ' <- If ReplyMode = 0 add this line !

With ReplyMode = 0 a communication speeds of 130 FPS can be achieved with four servos, and 60 FPS with an entire COBOT arm with six motors plus gripper.

!!! ATTENTION !!!

If you set ReplyMode to zero you must also add a row with the **DisableReturnPackets** command otherwise with every movement communication errors will occur and the FPS box turns red for a few moments.

The options of the **Slot To DXP** command

You can change the behavior of type commands **Slot x To DXPn x xx-x** adding some letters (A, M, C e Fxx) at the end of the command.

The "M" and "C" options, are explained in the page: [The "C" and "M" options](#) and in the page: [Measurement units and the "C" option](#)

Option A ("Always" in English)

The commands that send the value from the Slots to the devices are executed only if the value to be sent changes by at least one unit, but using the A option they are always executed. The comm. speed decreases slightly but a continuous exchange of data is obtained which is useful in some cases, for example with motors that sometimes lose commands. This happened sometimes with the STS3215s that lost the last destination and therefore stopped the execution of the Cobot application.

Example: **Slot 1 To DXP1 1 11-2 A** 'ALWAYS (also if not changed)

The options for **DXP to Slot** and **Buffer to Slot**

You can modify the behavior of commands of type: **DXP1 x xx-x To Slot x** and of type: **Buffer xx-x To Slot x** by adding options (M, C, F00, F01, F02 and F12) at the end of the command.

If desired, they can also be separated with the hyphen (minus) character or the underscore character, as in the following examples:

DXP1 x xx-x To Slot x MCF12
DXP1 x xx-x To Slot x C_F12M
DXP1 x xx-x To Slot x F12-MC
DXP1 x xx-x To Slot x M-F12_C
Buffer xx-x To Slot x CM-F12
Buffer xx-x To Slot x F12-MC

- The M and C options meanings are: Mirror (inversion of rotation) and Calibration (measure units and trimming of the zero position).
- The functions from F00 to F99 are value conversion functions.

Options and functions are explained on the page [The "C" and "M" options](#) and the following pages.

The Sections and the Sections Slot

Using the sections may be necessary when the connected motors are numerous, say maybe a dozen, or even a hundred. In these cases the communication speed may drop so much that it causes obvious problems. If you go below 30 FPS the movements become erratic and you can experience rocking or even uncontrollable oscillations.

The writing commands (from the software to the motors) can be almost instantaneous, so they can always be sent to all the motors at each communication cycle. Furthermore, these commands are sent only if the value changes (check on the previous pages which commands to use to obtain the maximum speed).

On the other hand, the reading commands are very expensive in terms of time. While using the best commands, each motor must respond and it takes at least a few milliseconds to do so. Therefore, if dozens of motors are interrogated, the communication speed is too low.

To solve this problem we have added the possibility to activate only some parts of the program. It is therefore possible to keep the instructions that move the motors always in operation and only occasionally read the data of the motors or, better still, read them one at a time. This method could therefore be useful for occasionally checking the temperature of the motors or the supply voltage.

To use sections, you establish a Control Slot and set it with the statement **SectionSelectorSlot**. If this instruction is missing, or if it is commented out, or if the Slot number is invalid, then all sections of the program are executed and the instructions **Section** they no longer have an effect.

Example of using sections

SectionSelectorSlot 11 ' Slot setting for controlling sections

instructions to always follow

Section 1 ' Section start marker 1

instructions in section 1

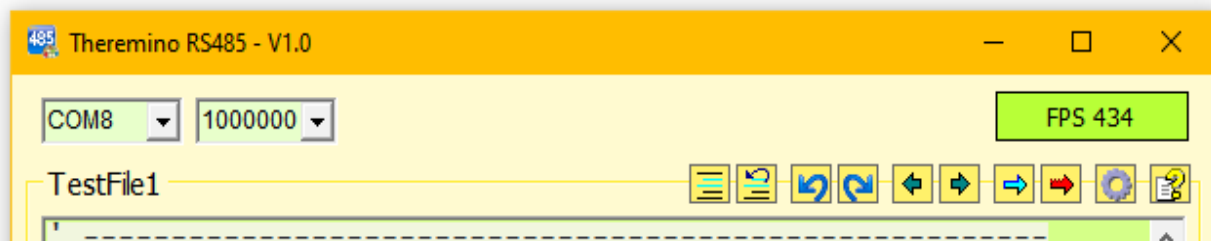
Section 2 ' Section start marker 2

instructions in section 2

Depending on the current numeric value of Slot 11, only the instructions in the corresponding section are executed.

All statements preceding the command **Section 1**, are always performed. If you want you can write a **Section 0** at the beginning but it is not necessary.

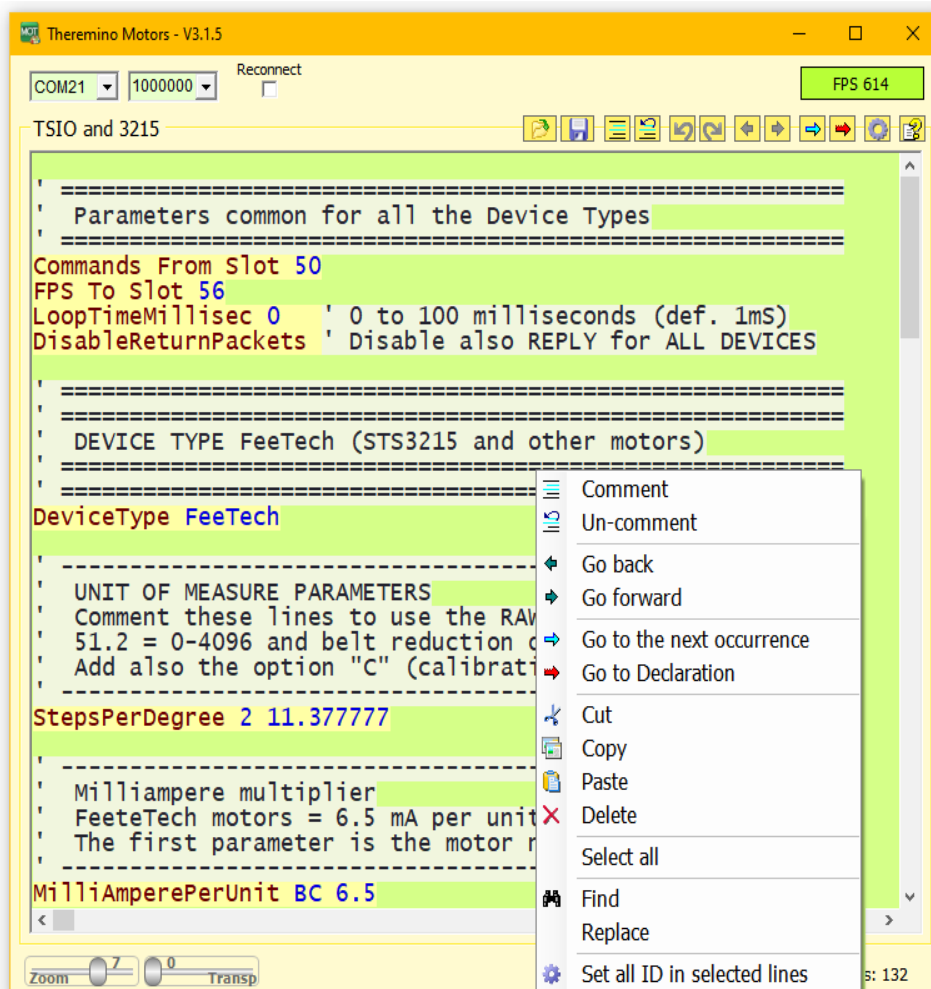
Controls of the application



In the upper area we find the controls for communication and the buttons of the instruments.



In the lower area we find the size and transparency controls and information on the cursor position.



By clicking with the right mouse button on the program area (or by touching the touch screen for two seconds), the menu shown below opens.

These checks will be explained one by one on the following pages.

The controls on the top bar

With the two controls on the left you set the communication port and its speed in Baud (bits per second). With the box on the right you check that no errors occur and you check the communication speed in FPS (Frames per second). "Frame" means sending and receiving all command lines that have been programmed.



With the right settings you should get to write a register and read the response at a speed of 400 ... 500 FPS. In some cases it is possible to read multiple registers towards the Buffer up to 900 FPS and beyond.

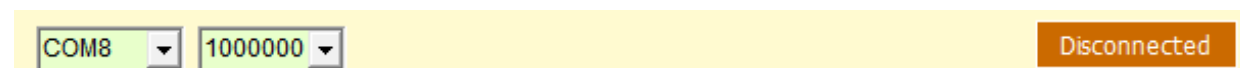
When using multiple motors the speed necessarily drops, but if you use the proper instructions you can check a dozen devices before dropping below 50 FPS and starting to have problems.

Communication errors

Errors are divided into various types:



In this image the two boxes on the left with a red background indicate that the COM1 port is not working.



Here instead we see that the COM8 port is connected but that one or more devices are not responding with valid packets. This can happen if you interrogate devices with wrong ID, or not connected, or without power.

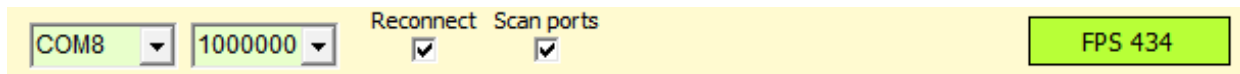


In this case the COM8 port is connected but the word "Inactive" indicates that no return packets are being received.

The "Inactive" condition may also not be due to an error but only to the fact that none of the commands expects a response. This can happen if only Broadcast commands are used and no read commands are used.

The automatic reconnection options

Since version 1.2 we have added the possibility to automatically reconnect the COM port and the devices connected to it.



The "Reconnect" option continuously checks that there is communication with the devices and if they do not respond, it closes and reopens the COM port and re-initializes the connected devices.

The "Scan Ports" option also adds the COM port change. The ports are then tested all in sequence, until one is found working and with the devices responding.

Use automatic reconnection with caution

In some cases the devices may be programmed not to respond and none of the communication commands may expect a response.

In these cases the automatic re-connection may believe that there are communication errors and cause continuous re-connections, thus blocking the communication.

Use port scanning carefully

In some cases the "Scan Ports" option may slow down the system boot and go through all the ports before finding the right one.

So if you know the communication port and it never changes, it might be better not to enable the "Scan Ports" option.

Increase the communication speed

To get smooth movements the number of exchanges per second (FPS) must be at least 20, but if possible it is better to go beyond 50.

Furthermore, if the applications carry out checks and changes in real time, it is good to minimize reaction times and have an exchange rate of at least 100 FPS.

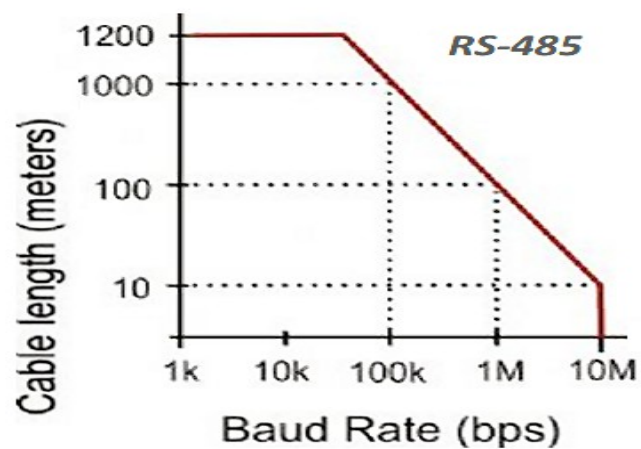
There are no problems when controlling one device, but achieving these speeds with three or more connected devices requires careful programming.

First of all it is better to increase the Baud Rate to the maximum (see previous page "COM Port Settings")

128000
1000000
500000
250000
128000
115200
76800
57600
38400

FeeTech servos work up to 1 mega bit and TMOTs up to 2 mega bit.

The graph on the right indicates to limit the speed only if the cable is over 100 meters long (or over 30 meters in the case of 4 Mb Dynamixel),



Then you have to lower the LoopTime and the Return Delay to zero

Example: **LoopTimeMillisec 0** 'LoopTime = 0

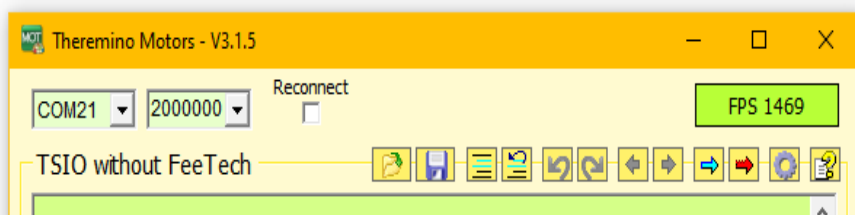
Example: **0 To DXP1 BC 7-1** 'Immediate value to Return Delay

(and also remember to add the **DisableReturnPackets** statement)

Finally, only the communication instructions that allow fast data exchange must be used

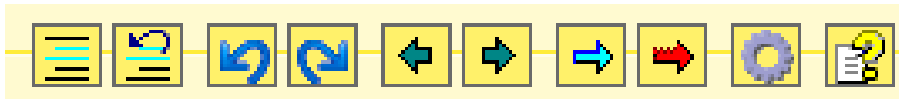
Only instructions that read to the buffer will be used to read. These instructions read an entire block of bytes from a device (even many tens of bytes), using a single return packet.

Example: **DXP1 1 50-8 To Buffer**



With the FeeTech cards of the new version (which have the USB writing to the left of the connector and not below) the maximum data exchange rate is around 1500 FPS.

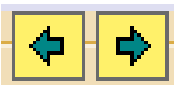
The toolbar



The first two buttons comment and de-comment the selected text.



The two blue arrows they are used to go back in the program changes and to rebuild the deleted changes.



The two dark ARROWS move the cursor, and also the visible page, to the previously visited program sections.

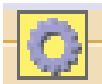


The blue ARROW searches for all occurrences of the selected word, or even just indicated by the text cursor.



The red ARROW only searches for active words (which are not in the commented areas).

The search functions are convenient, just select a word or just place the cursor on it and then press the arrow repeatedly.



The gear opens the options window to assign the identifier to all devices and also to update the firmware and assign the serial number to theremino system devices (TMOT and TSIO).



The question mark opens the instruction file (Help) in the chosen language. For this command to work you must copy the Help file of your preferred language into the "Docs" folder.

The latest Help files are downloaded from [This Page](#).

If the Help file is not found then a message appears suggesting to open the Docs folder and copy the file into it.

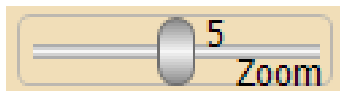
Or you can choose to select a Help file in your preferred language located in the "Docs" folder or any other folder. *To change the selected file click the button with the right mouse button.*

The controls of the lower bar

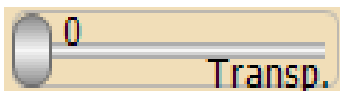


Lines: 83 Line: 1 Col: 0

The cursors are adjustable with the mouse and using the right mouse button return to the default position.



The ZOOM slider determines the size of the text.



This slider adjusts the transparency of the main window and allows you to see below it as well.

Lines: 29 Line: 17 Col: 16

The right part of the bottom bar shows information about the program:

- The total number of lines
- The line where the cursor is (starting from line 1)
- The column where the cursor is (starting from column 1)

The context menu

This menu shows some commands already available with the tool buttons (the buttons at the top right) but integrates them with other useful commands.

By clicking on the program area, with the right mouse button (or by touching the touch screen without removing your finger for two seconds), the menu shown below opens.

Comment is **Uncomment** they are used to comment (add the initial superscript) to entire program areas. Or to delete comments.

Go back is **Go forward** they move the cursor and the visible page to previously visited program areas.

Go to the next occurrence search for other occurrences of the selected word.

Go to declaration search for the selected word but only in the active areas (the parts not commented on).

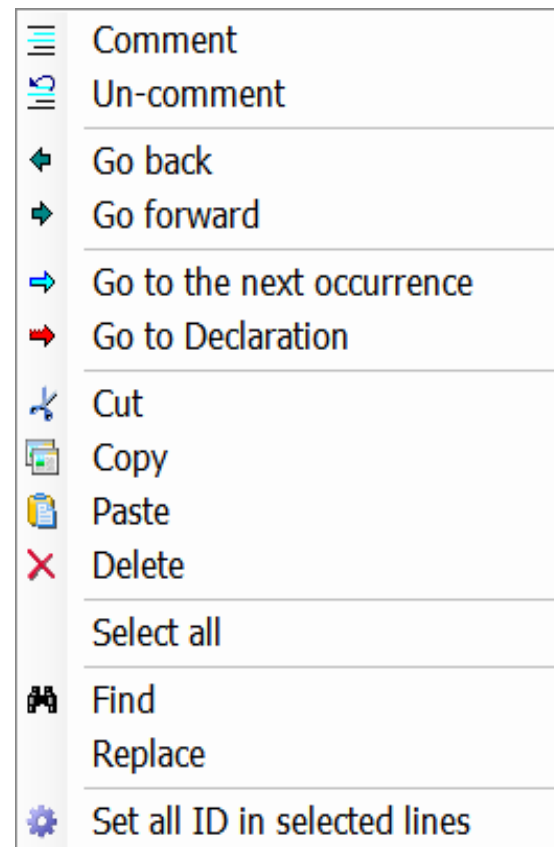
Cut, Copy is Pastes cut, copy and paste selected parts of the text.

Delete deletes the selected part of the text.

Select all select all text.

Find is **Replace**, they open a window to search and replace words and phrases.

Set all ID in selected lines è spiegato nella prossima pagina.



Some of the commands in this menu can also be reached with the keyboard, using the CONTROL key in combination with some letters.

CTRL-X, CTRL-C, CTRL-V for Cut, Copy and Paste

DELETE for Delete

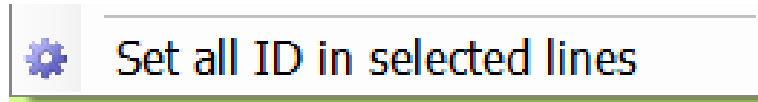
CTRL-A for Select all

CTRL-F for Find

CTRL-R for Replace

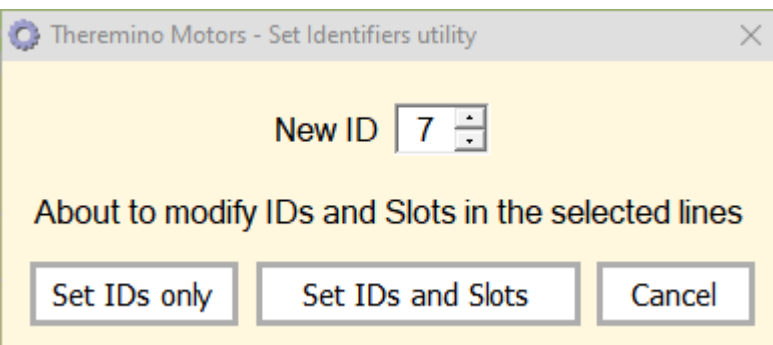
Set all ID in selected lines

This command modifies all the IDs (identifiers of the devices) and all the relative Slots found in the selected text area.



To use the command:

- Select a text area that contains only one device.
- Open the right-click menu and select "Set all ID..."
- Choose the "New ID" from 1 to 9
- Press "Set IDs only" to change only the IDs and not the Slots.
- Press "Set IDs and Slots" to change the IDs and also the Slots.



With just one click you get a result that would have required many minutes of careful manual work.

Here is an example of changing Slots from 2xx a 7xx and IDs from 2 to 7

BEFORE		AFTER	
Slot 201	To DXP1 2 42-2	Slot 701	To DXP1 7 42-2
Slot 220	To DXP1 2 48-2	Slot 720	To DXP1 7 48-2
Slot 221	To DXP1 2 41-1	Slot 721	To DXP1 7 41-1
Slot 222	To DXP1 2 46-2	Slot 722	To DXP1 7 46-2
Slot 223	To DXP1 2 40-1	Slot 723	To DXP1 7 40-1

Depending on how the program is composed, in some cases it will be possible to select many pages that make up the whole device from start to finish, but in other cases you can select only sections of lines and therefore the command will have to be repeated several times with different IDs.

Use this command very carefully.

In case of errors, use the UNDO command to eliminate the changes.

Commands from Automation to Motors

To send commands to the Motors application, the `Slot_CommandsToMotors` variable is set and then used with the commands to write and read in the text slots.

Calibrate Motor x To nn.n

This command makes the current position of motor x correspond to the value nn.n

The nn.n value is a floating point number that takes into account all options:

- Multispeed (option "F12" for FeeTech and other engines with 4096 module)
- Unit of measurement (for example Millimeters or Degrees)
- Mirror (option "M" i.e. reversal of the direction of rotation of the motor)

ResetAllCalibrationValues

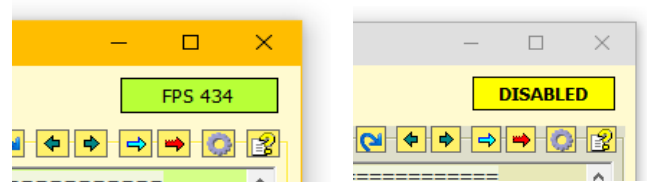
This command acts on all motors by resetting the array containing the calibration values of all motors.

ResetAllMultiturnValues

This command acts on all the motors by resetting the array that takes into account how many revolutions have been accumulated.

EnableConnection

DisableConnection



These commands enable and disable communication as happens when you click the mouse on the top right button, which indicates the communication speed in FPS.