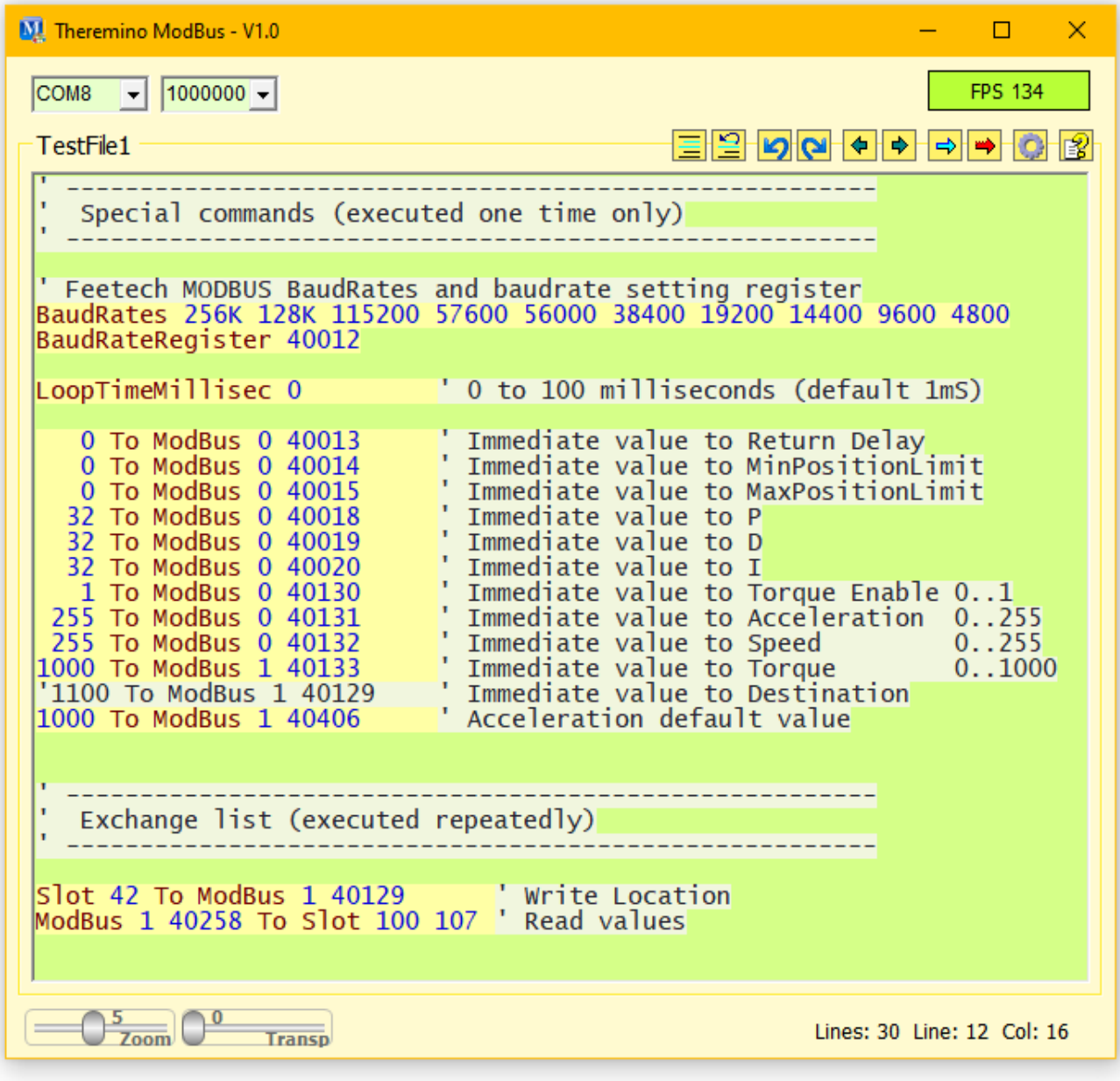
System theremino

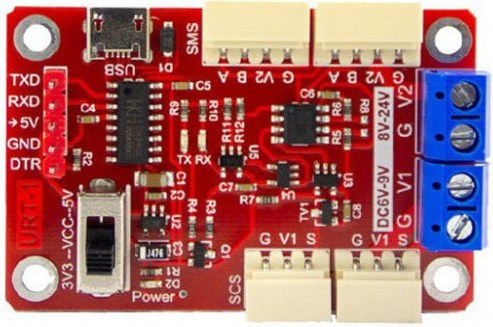


Theremino ModBus V1.2

The Theremino_ModBus application

This application connects the Theremino system Slots with devices connected to a serial line ***Half-Duplex*** (RS485 with two balanced wires, or TTL with a single wire).

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



The connectable devices are only i ***FeeTech*** , while the Dynamixels don't work with the ModBus protocol.

Given the choice, it would always be better to use the Dynamixel protocol, even for FeeTechs, as the ModBus has considerably lower performance.

These servo-motors, also called Robot-Motor or Smart-Motor, contain all the control electronics, a 4096 step encoder and a configurable PID algorithm that allows to control the rotation with 0.09 degrees precision.

You can also adjust the speed, acceleration and torque, as well as read the position reached, the temperature, the current (which is related to the torque) and many other parameters.

Some Dynamixel motors can be controlled in a range of +/- 256 rpm, while currently the FeeTechs ***(Note 1)*** they have a smaller range, of only +/- 7.5 turns.

*(**Note 1**) Two models from FeeTech are in preparation with true torque control and position control in a practically infinite range. True SmartMotors for industrial controls but at a highly competitive price.*

Communication protocols

The protocol used by this application is the ModBus, of which we only use the commands for writing and reading the registers (Write-Holding-Register and Read-Holding-Register).

The commands for writing and reading the registers start with the number 4 and can be composed of a variable number of digits, the important thing is that there is 4 at the beginning and the register number at the end (from 1 to 65536).

For which the following addresses are all the same: 4014, 40014, 400014, 4000014 and all refer to register 14.

Use DXP1 or Modbus protocols

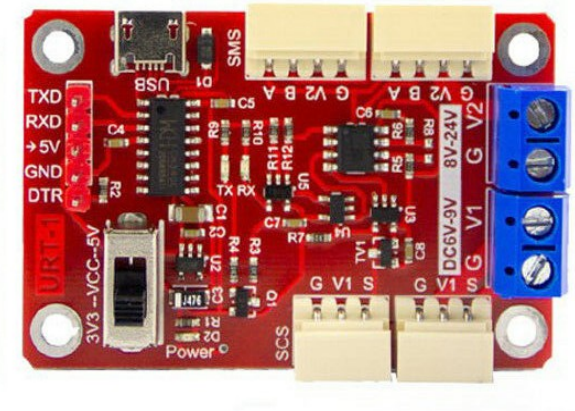
FeeTech servos can be programmed to use either the DXP1 protocol or the ModBus protocol. The latter is familiar to PLC users, but has lower speed performance and fewer commands.

To use DXP1 devices we have written an application very similar to this one, but called ***Theremino_RS485*** 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 page.

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.

You could use this same adapter to make the Dynamixels work, but we haven't tried it yet.

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.

```
' ------------------------------------------------------
'  Special commands (executed one time only)
' ------------------------------------------------------

' Feetech MODBUS BaudRates and baudrate setting register
BaudRates 256K 128K 115200 57600 56000 38400 19200 14400 9600 4800
BaudRateRegister 40012

LoopTimeMillisec 0          ' 0 to 100 milliseconds (default 1mS)

   0 To ModBus 0 40013      ' Immediate value to Return Delay
   0 To ModBus 0 40014      ' Immediate value to MinPositionLimit
   0 To ModBus 0 40015      ' Immediate value to MaxPositionLimit
  32 To ModBus 0 40018      ' Immediate value to P
  32 To ModBus 0 40019      ' Immediate value to D
  32 To ModBus 0 40020      ' Immediate value to I
   1 To ModBus 0 40130      ' Immediate value to Torque Enable 0..1
 255 To ModBus 0 40131      ' Immediate value to Acceleration  0..255
 255 To ModBus 0 40132      ' Immediate value to Speed         0..255
1000 To ModBus 1 40133      ' Immediate value to Torque        0..1000
'1100 To ModBus 1 40129     ' Immediate value to Destination
1000 To ModBus 1 40406      ' Acceleration default value


' ------------------------------------------------------
'  Exchange list (executed repeatedly)
' ------------------------------------------------------

Slot 42 To ModBus 1 40129       ' Write Location
ModBus 1 40258 To Slot 100 107  ' Read values
```

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.

```
' ------------------------------------------------------
'  Exchange list (executed repeatedly)
' ------------------------------------------------------

Slot 42 To Mobbus 1 40129       ' Write Location
ModBus 1 40258 To Slot 100 107  ' Read values
```

In case of errors the command turns red here you see a line that contains an error, the word "Mobbus" which does not exist.

List of ModBus commands

Initialization commands
Run once, at startup and when editing program text

```
BaudRates nnn, nnn, nnn      ' BaudRate List
BaudRateRegister rrr         ' Register to set the BaudRate
LoopTimeMillisec nnn         ' Delay to slow down the execution loop
SectionSelectorSlot s        ' Setting the Sections Slot
nnn To ModBus d rrr          ' "Immediate" number to a register
```

Write commands
Performed continuously, as often as possible

```
Slot s To ModBus d rrr       ' Read the Slot and write in a register
Slot s To ModBus 0 rrr       ' Write to all devices (0 = Broadcast)
```

Read commands
Performed continuously, as often as possible

```
ModBus d rrr To Slot s       ' Read from a register and send to a Slot
ModBus d rrr To Slot s1 s2   ' Read from many registers and send to many Slots
```

Section setting command
Executed at startup and when changing program text,
and also when the value of the "SectionSelectorSlot" is changed.

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

Meaning of abbreviations

nnn = Number
s = Slot (1 to 999)
s1 = Initial slot (1 to 999)
s2 = Final slot (1 to 999)
d = Device identifier (1 to 247) (0 = "Broadcast")
rrr = Device register (4000001 to 4065536)

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

```
BaudRates nnn, nnn, nnn          ' BaudRate List
BaudRateRegister rrr             ' Servo register to set the Baud Rate
LoopTimeMillisec nnn             ' Delay to slow down the execution loop
SectionSelector_Slot s           ' Setting the Sections Slot
```

Among the special commands, which are performed only once, there are also the transfers of the "Immediate" numbers, which will be explained more fully in the following pages.

Example of "Immediate" transfer

```
nnn To DXP1 d rrr                ' "Immediate" number to a register
```

Meaning of abbreviations

nnn = Number
s = Slot (1 to 999)
s1 = Initial slot (1 to 999)
s2 = Final slot (1 to 999)
d = Device identifier (1 to 247) (0 = "Broadcast")
rrr = Device register (4000001 to 4065536)

Log 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***

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

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

Dynamixel Servo Motors

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 engine, so we have not prepared any documentation on them.

Go to consult the characteristics and registers to use for each engine, 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 engines, click to open them and inside you will find all the engines with their images.

Then clicking on the single engine 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 ModBus 0 40013          'Immediate value to Return Delay
0 To ModBus 0 40014          'Immediate value to MinPositionLimit
0 To ModBus 0 40015          'Immediate value to MaxPositionLimit
32 To ModBus 0 40018         'Immediate value to PID Proportional
32 To ModBus 0 40019         'Immediate value to PID Derivative
32 To ModBus 0 40020         'Immediate value to PID Integral
1 To ModBus 0 40130          'Immediate value to Torque Enable
255 To ModBus 0 40131            'Immediate value to Acceleration
255 To ModBus 0 40132            'Immediate value to Speed
1000 To ModBus 0 40133           'Immediate value to Torque
```

Note that in these examples the target device is 0 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.

Command example that initializes the position of device one

```
1000 To DXP1 1 40129         '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 numeric value of Slot 12 and writes it to the register located at address 129 of device 3.

Slot 12 To ModBus 3 40129

- The first part Slot 12 defines the Slot (could be from 0 to 999)
- The word To indicates the direction (from Slot to ModBus)
- ModBus defines a device that uses the ModBus protocol
- The number after ModBus defines the device. In this example the device is the 3 but it could be from 1 to 247.
- The number 0 cannot be used to indicate the devices because it would specify a "broadcast" sending, ie a simultaneous sending to all the connected devices.
- The number 255 cannot be used because it is used by the protocol as a transmission start signal.
- The last part **40129** indicates to write to the device register 129.

Examples of transfers from a Slot to a device

Slot 22 To ModBus 1 40129 From Slot 22 to register 129 of device one.

Slot 22 To ModBus 0 40129 From Slot 22 to register 129 of all devices.

Slot 22 32 To ModBus 0 40129 Slots 22 to 32 to registers 129 to 139

Examples of transfers from a device to a Slot

ModBus 2 40258 To Slot 22 Register 258 of device 2 towards Slot 22

ModBus 3 40259 To Slot 22 Register 259 of device 3 towards Slot 22

ModBus 1 40258 To Slot 100 107 Seven registers towards Slots 100 to 107

"Broadcast" programs

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 "0" is used instead of the device identifier.

Examples of broadcast transmissions

```
0 To ModBus 0 40013   'Immediate value to Return Delay
0 To ModBus 0 40014   'Immediate value to MinPos Limit
0 To ModBus 0 40015   'Immediate value to MaxPos Limit
32 To ModBus 0 40018 'Immediate value to PID-Proportional
32 To ModBus 0 40019 'Immediate value to PID-Derivative
 32 To ModBus 0 40020        'Immediate value to PID-Integral
```

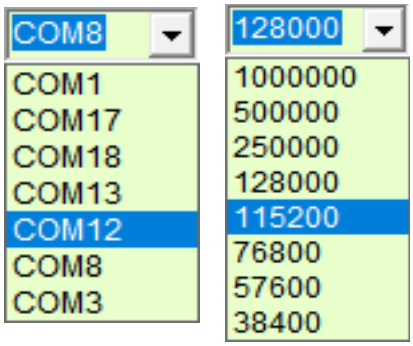
With this series of instructions, which is executed only once at start-up, all connected devices are initialized in the same way.

Example of a broadcast transmission that moves all the servos

```
Slot 100 To ModBus 0 40129
```

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

COM port settings



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

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.

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

To make the servo and application understand each other it is essential that the following two setting lines are at the beginning of the program and that they are correct.

Settings for FeeTech servos:

```
BaudRates 1M 500K 250K 128K 115200 76800 57600 38400
BaudRateRegister 6
```

The line `BaudRates` must contain **exactly** all the speeds that the servos accept and the speeds must be in the right order, otherwise the servos will be programmed with the wrong speed and will not work.

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.

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.

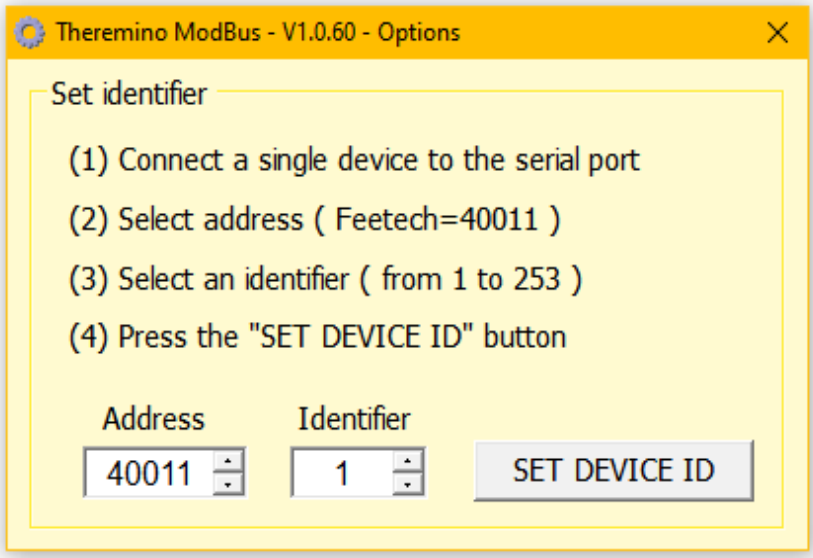
Note that at least one read command must be used, otherwise the text "Inactive" will appear instead of the FPS

Identifier setting

If two or more devices are connected on the same line and with the same ID, communication errors occur. In these cases the communication speed indicator may indicate very low FPS and cause the word "Disconnected" to blink.

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

- First of all make sure you can write to the EEPROM, as explained on the next page.
- Open the options panel using the gear tool located at the top right.



- Set the address (Address) of the identifier. Normally in FeeTech servos it is 40011.
- Connect a single device e make sure it is communicating. Possibly open and close the serial port box and also try to move it using a Broadcast command, such as this:**Slot 1 To ModBus 0 40129**
- Choose an identifier from 1 to 253 to assign to the device, making sure to use a different number for each of them.
- Press the SET DEVICE ID button.
- Repeat for all devices.

Enable writing of the EEPROM

FeeTech 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 the device identifier is changed it only acts in the RAM memory and this setting is then lost as soon as the supply voltage is turned off.

If only one servo was used then it would be enough to write a line of setting the identifier as the first line of the program and then set it at each power up. But if you are using multiple servos then each servo needs to remember its identifier so make sure to unlock the EEPROM before setting the identifier.

Example of command that enables EEPROM

```
0 To ModBus 0 40134   'Immediate value "0" to enable EEPROM writing
```

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 EEPROM lock register in position 40134.

The Sections and the Sections Slot

Using the sections may be necessary when the connected motors are numerous, say more than four, or 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. These commands are sent only if the value changes (check on the previous pages which commands to use to get the maximum speed).

On the other hand, the reading commands are very expensive in terms of time. While using the best commands, each engine 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.

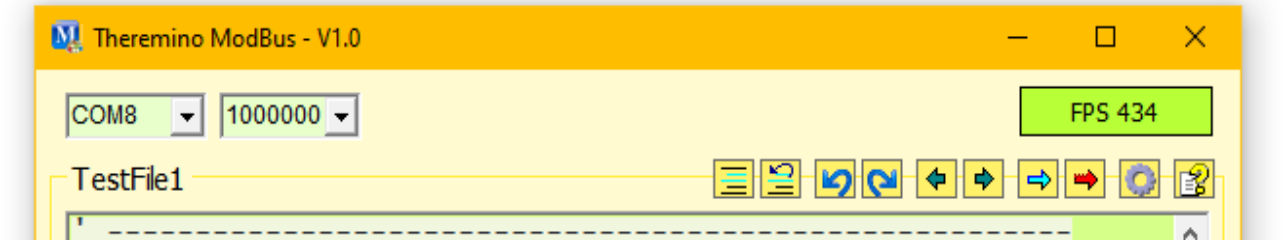
Example of using sections

```
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.

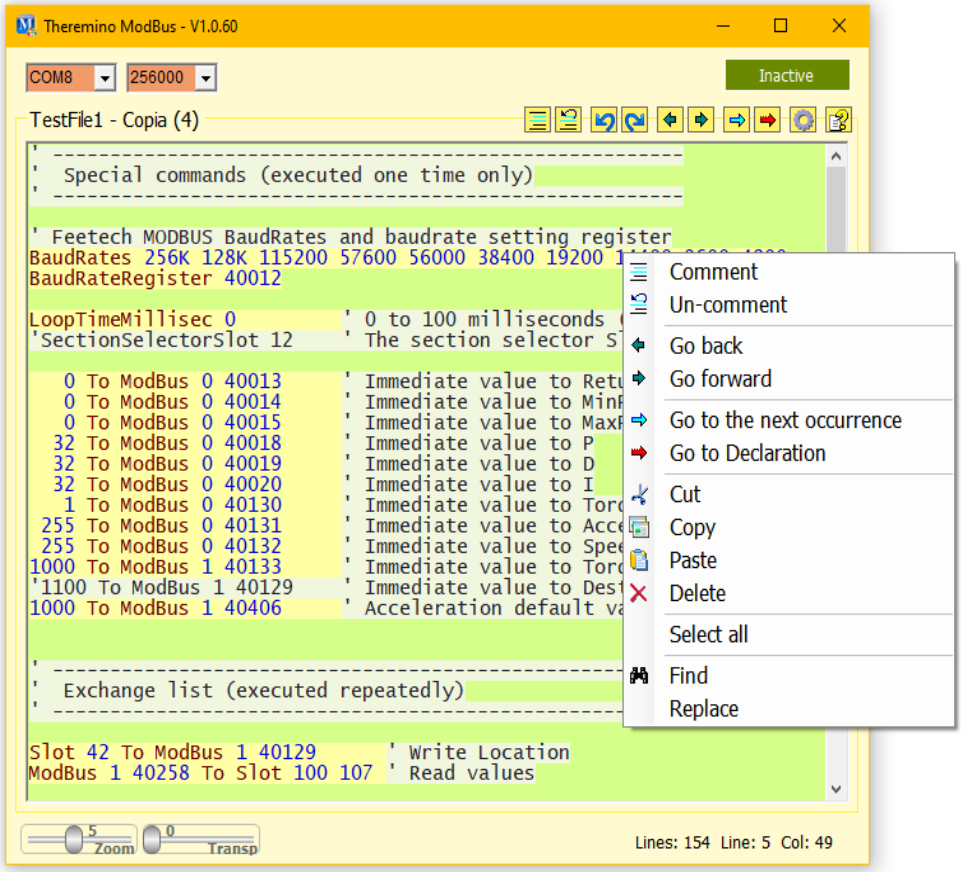
All statements preceding the command **Section 1**, are always performed. If you want you can write a**Section 0** at the beginning but it is not necessary.

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.

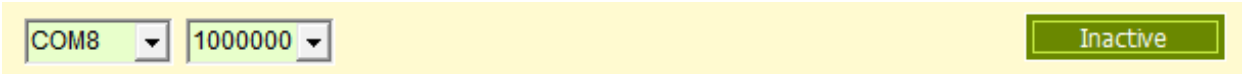
When using multiple motors the speed necessarily drops, but if you use the proper instructions you can check a dozen devices before dropping below 50 FPS and starting to have problems.

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.

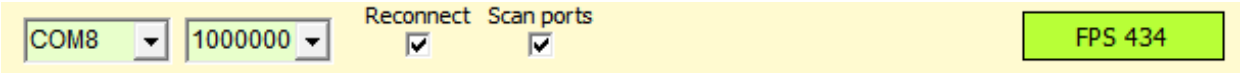


In this case the COM8 port is connected but the word "Inactive" indicates that no return packets are being received.

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 or Sync write 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.

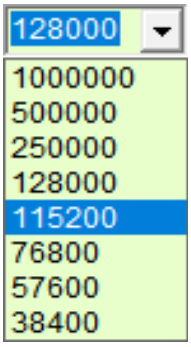
Increase the communication speed

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.

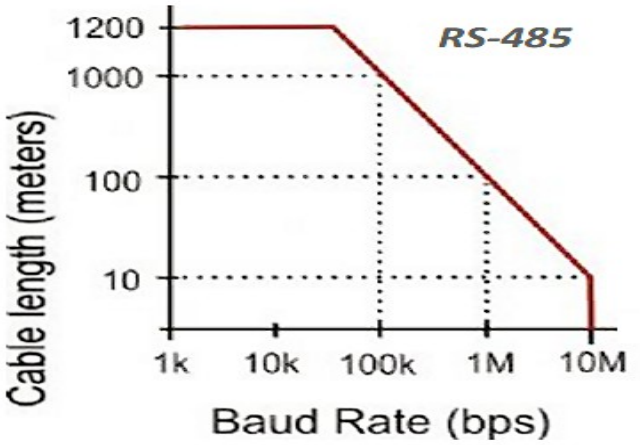
There are no problems when controlling one device, but achieving these speeds with three or more connected devices requires careful programming.

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



FeeTech servos work up to 1 mega bit and Dynamixel up to 4 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 ModBus 0 40013** 'Immediate value to Return Delay

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

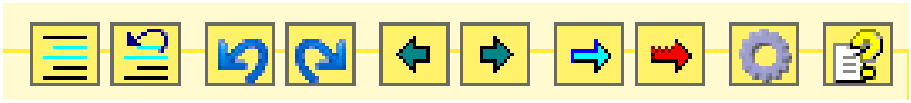
To write the registers, if possible, it is good to use multiple writes, as in this example which reads three Slots (42, 43, 44) and writes them in three registers starting from 40129.

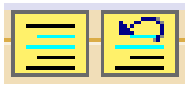
Example: **Slot 42 44 To ModBus 1 40129**

Only multiple read instructions will be used to read, as in this example which reads eight registers starting at 40258 and writes them to registers 100 to 107.

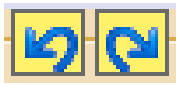
Example: **ModBus 1 40258 To Slot 100 107**

The toolbar

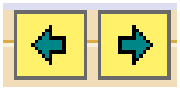




The first two buttons comment and de-comment the selected text.



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

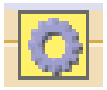


The two dark ARROWS move the cursor, and also the visible page, to the previously visited program sections.

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 the devices.

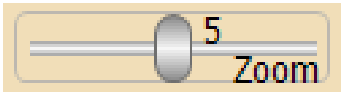
The question mark opens the instruction file (Help) in the chosen language. For this command to work you must copy the Help file of your preferred language into the "Docs" folder.
The latest Help files are downloaded from ***This Page***

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*.

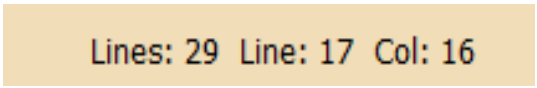
The controls of the lower bar

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.



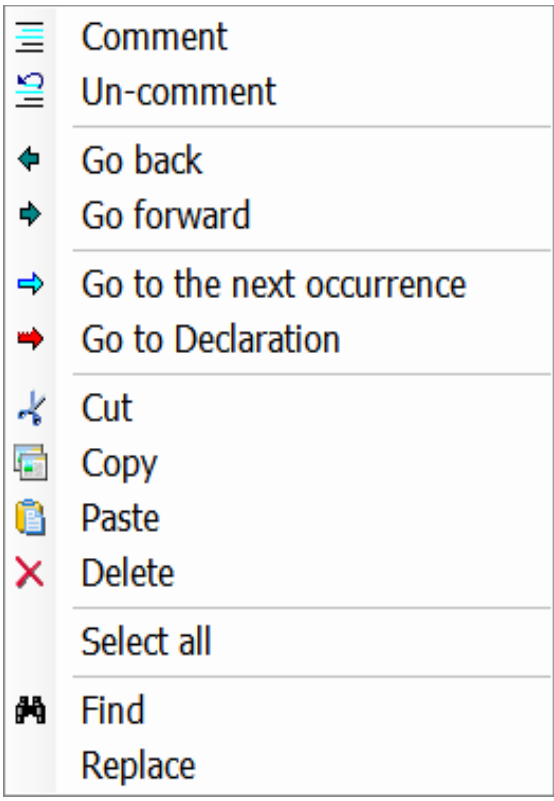
The right part of the bottom bar shows information about the program:

- 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.

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