



# Ergotech

## Green Drive

### High Torque Plastic Strain Wave Gear

**ErgotechGreenDrive** is a strain wave powered gearbox:

- Fully integrated with frameless motor and control system
- High torque density, brushless PMSM
- High-resolution positioning sensors
- RS485 or one-wire driver board
- Low weight due to technopolymers
- Post-industrial materials

## Summary

High Torque Plastic Strain Wave Gear .....	1
Communication between motor and PC via USB .....	3
Applications for communicating with the gearmotor .....	3
Program in Automation using artificial intelligence.....	3
The TCOM Module .....	4
Connectors for TCOM board .....	4
Communication between motor and PC with commercial boards.....	5
Direct communication between motor and PLC via RS485 .....	6
Communication protocol.....	7
Communication parameters.....	8
MOT1 - Communication parameters (read) .....	8
TMOT1 - Communication parameters (write) .....	9
MEANINGS.....	10
NOTES.....	10
TSIO communication module .....	11
Gearmotor and TSIO board registers .....	12
Wiring diagram when using only a PLC .....	13
Wiring diagram when using a PC .....	14
Connection interface.....	15

## Communication between motor and PC via USB

This is the simplest communication method that allows you to use all open-source applications developed to facilitate the use of the gearmotor and perform tests on the system where the gearmotor is integrated (see applications section).

In the EGD1 motors with the actual firmware (V2.3), the communication is fixed at 1 Mega Baud.

## Applications for communicating with the gearmotor

If you have chosen to communicate with the gearmotor via PC, you can use a series of open-source and "portable" applications (that do not require installation) designed to interact with the gearmotor and monitor its operation. Once you download the .zip file of the applications, extract it into any folder and launch "Automation.exe", which in turn will start all other applications.

You can download the full .zip file from this link: [www.ergotech.it](http://www.ergotech.it)

You can also download (eventually updated versions) individually from [www.theremino.com](http://www.theremino.com) through these links:

**Automation**: to program the entire system that includes multiple gearmotors and sensors, communicate with other applications, and interact with the user, including through sound and voice messages.

**Motors**: it manages the communication protocol with the gearmotor and any other modules, such as the TSIO, which handles I/O as an alternative to the HAL.

**HAL**: it handles input/output (ADC, digital I/O, PWM, counters, sensors, etc.).

**SlotViewer**: it displays the Slots that contain numerical values and text strings used to communicate between the system's applications.

**SignalScope**: it displays signals (motor positions, temperatures, speeds, etc.)

**Logger**: it updates a log file in real time with the desired signals (temperatures, speeds, etc.).

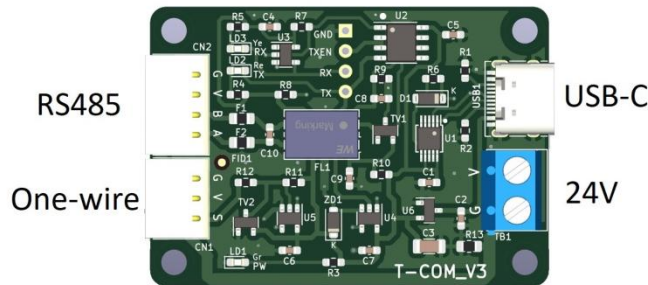
**Graphs**: it displays log files in real time

## Program in Automation using artificial intelligence.

If you design programs and systems with our applications, read this page!  
The LM notebook will search for what you need among hundreds of pages of documentation, and you can even have entire programs written for you, just ask!

## The TCOM Module

To allow communication between gearmotor and PC, you can use the TCOM module.



Top right: USB-C connector for PC connection.

Below: 24V power connector.

Top left: RS485 serial connector (communication protocol used in all Ergotech Green Drive 1 models). Bottom left: One-wire serial connector (for some models).

Advantages of TCOM:

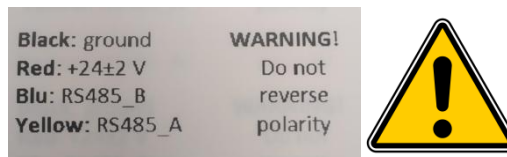
- Communicates in RS485 and One-wire.
- Provides connectors for both signals and power.

### Connectors for TCOM board

To connect the Ergotech Green Drive 1 to the PC via the TCOM board with RS485 communication protocol, you need to acquire terminals and a connector, available for purchase on all major e-commerce sites:

- [Molex 0008701039](#) terminals
- [Molex 50375043](#) 4-wire connector

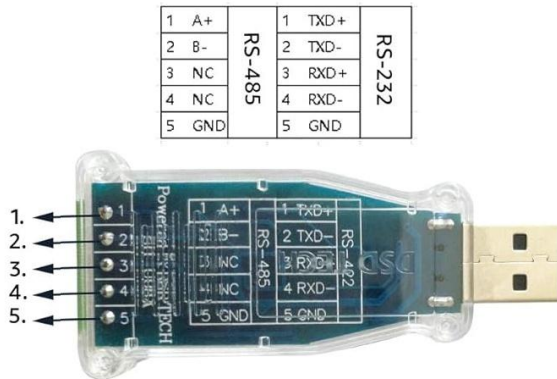
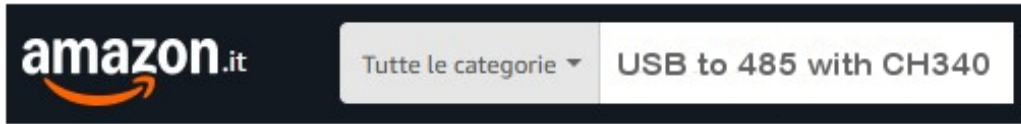
Wire order must match the TCOM board labels: Ground (G), Power (V), RS485\_B (B), and RS485\_A (A).



*EGD1 Label*

## Communication between motor and PC with commercial boards

Communication with the gearmotor via USB can also be done using commercial USB-to-RS485 boards (search "usb to 485 with CH340").



In this image, you can see a commercial board that we have tested and that works well. This particular model contains the CH340E chip, which communicates faster than others.

It is recommended to ensure the board uses the CH340 chip and not FTDI (which works but with lower performance and harder configuration). These boards only allow RS485 communication (not One-wire).

With these boards, you can only communicate via RS485 and not via one-wire as you could using the TCOM module and special gearmotors that support one-wire communication. Moreover, since these boards do not provide the TCOM module's power terminal, you must connect the 24 V directly between GND and +V on the 4-wire cable that goes to the motor chain.

## Direct communication between motor and PLC via RS485

To enable communication between the gearmotor and a PLC, you need to use an RS485 adapter module for the PLC— for example, the Siemens xxx module. To communicate directly with the motor, you must write a program in the PLC that follows the protocol and communication parameters described on the following pages.

You can design programs to control the motors in various ways, and many parameters aren't used to actually drive the motor. For example, you might read the temperature, set acceleration and speed limits, configure PID parameters, or manage the motor's identifier and password.

One important operation your PLC program should support is assigning each motor's ID even when they're all connected on the same serial line. Competing products force you to disconnect each motor individually and hook it up to a programmer, which can be very inconvenient.



To avoid that, we designed our motors so you can set their IDs manually by bringing a magnet close to them. When you start the identification routine, the program configures each motor to send a response when its sensor detects the magnet and then identifies them one by one.

Programming a PLC for such complex tasks can be quite challenging, so a good approach is to first use our PC applications and familiarize yourself with the core motor-control operations.

That way, you can monitor the communication line, identify and test the motors, and only afterward disconnect the PC and rely solely on the PLC.

Later on, you can also review the procedures used in the code of our "Motors" application—and copy them into your PLC without issue, since they're all open-source and free of copyright restrictions.

## Communication protocol

If you want to communicate directly with the gearmotor from the PLC, you must follow the protocol described below.

Communication occurs serially (8N1 – 8 bits, no parity, 1 stop bit) with packets structured according to the following schema.

Transmission packet (Instruction packet):

Header1	Header2	Packet ID	Length	Instruction	Param 1	...	Param N	Checksum
0xFF	0xFF	Packet ID	Length	Instruction	Param 1	...	Param N	CHKSUM

Reception packet (Return packet):

Header1	Header2	Packet ID	Length	Error	Param 1	...	Param N	Checksum
0xFF	0xFF	ID	Length	Error	Param 1	...	Param N	CHKSUM

For all protocol details (such as the composition of the checksum and the list of error bits), refer to the 'Communication Protocols' section of the following document:

[https://www.theremino.com/wp-content/uploads/files/Theremino\\_Motors\\_Help\\_ENG.pdf](https://www.theremino.com/wp-content/uploads/files/Theremino_Motors_Help_ENG.pdf)

## Communication parameters

Communication parameters for reading and writing to gearmotor registers are specified below.

### MOT1 - Communication parameters (read)

Adr.	Memory	Bytes	Default	Min	Max	Unit	Prot.	R/W
000	Serial Number	3		0	2 <sup>24</sup>			RO
003	Device Version	1		0	250			R
004	Device Sub-Version	1		0	250			R
005	Firmware Version	1		0	250			R
006	Firmware Sub-version	1		0	250			R
007	Boot loader version	1		0	250			R
010	Actual Position (Note 1)	4		-2 <sup>31</sup>	+2 <sup>31</sup>	Step		R/W
014	Actual Torque (Note 2)	2		0	32k	milli-nm		R
016	Actual Acceleration (Note 3)	2		0	65000	Rpm/S		R
018	Actual Velocity (Note 4)	2		-32k	+32k	Rpm		R
020	Actual Current	2		0		1 mA		R
022	Actual Voltage	2		0		1 V		R
024	Actual Temperature	1		0	250	1 °C		R
025	Moving Status (Note 5)	1		0	250			R
026	Hardware Error Status	1		0	250			R
027	ID Setting HallSensor value	2	2000	0	4095			R
030	MagSens1 A	2		0	5760			R
032	MagSens1 X	2		0	65535			R
034	MagSens1 Y	2		0	65535			R
036	MagSens2 A	2		0	5760			R
038	MagSens2 X	2		0	65535			R
040	MagSens2 Y	2		0	65535			R

## TMOT1 - Communication parameters (write)

Adr.	Memory	Bytes	Default	Min	Max	Unit	Prot.	R/W
050	Write protection (Note 6)	1	1	0	1			W
051	Working Mode (Note 7)	1	1	0	250		PRT	W
052	Reply Mode (Note 8)	1	0	0	1			W
053	Broadcast Code	1	254	200	250			W
054	Baud Rate Index	1	4	0	7			W
055	ID (device identifier) (Note 9)	1	1	0	199		PRT	W
056	ID Setting Tolerance	1	10	1	250			W
058	Goal Position (Note 1)	4	0	-2 <sup>31</sup>	+2 <sup>31</sup>	Step		W
062	Goal Torque (Note 2)	2	1000	0	32k	milli-nm		W
064	Goal Acceleration (Note 3)	2	10000	0	65000	Rpm/S		W
066	Goal Velocity (Note 4)	2	0	-32k	32k	Rpm		W
068	Position P Gain	1	20	0	250			W
069	Position I Gain	1	20	0	250			W
070	Position D Gain	1	20	0	250			W
071	Dead Band	1	0	0	250	Step		W
072	Zero Speed Current	1	0	0	255			W
080	MagSens1 GainAxis	1	0					W
081	MagSens1 Gain	1	0	0	255			W
082	MagSens1 OffsetX	1	0	-127	+127			W
083	MagSens1 OffsetY	1	0	-127	+127			W
084	MagSens1 GainAxis	1	0					W
085	MagSens1 Gain	1	0	0	255			W
086	MagSens1 OffsetX	1	0	-127	+127			W
087	MagSens1 OffsetY	1	0	-127	+127	Step		W
088	MagSens_WorkMode	1	0	0	255		PRT	W
100	Serial And Password (Note 10)	16					PRT	WO

## MEANINGS

PRT = Write protected

R = Read

W = Write

R/W = Read and Write

RO = Read Only

WO = Write Only

## NOTES

**(1) POSITION** - The unit of measurement is "step". Before the harmonic drive, there are  $384 * 60$  (23040) steps per turn. After the harmonic drive, there are  $384 * 60 * 20$  (460800) steps per turn. Writing zero to the Present Position will set the current motor position as the zero point. However, you can also assign a non-zero value, which will become the actual motor position.

**(2) TORQUE** - The unit of measurement is milli-newton on the output shaft (EGD1 max = 10 nm)

**(3) ACCELERATION** - The unit of measurement is RPM per second.

**(4) VELOCITY** - The unit of measurement is RPM.

**(5) MOVING STATUS** - 0 = Not moving.

**(6) PROTECTION** - Some parameters may also be written to flash memory to retain their values between power cycles. The <Write protection> must be turned off (set to 0) before modifying a value to write it to flash. See the "Prot." column.

**(7) WORKING MODE** - 0 = No operation / 1 = Position mode / 2 = Constant speed mode / 200 = StartSettingID / 201 = StopSettingID / 253 = RESET MOTOR. To modify this value, the WriteProtect must be unlocked by setting it to zero.

**(8) REPLY MODE** - 0 = Reply only for read instructions / 1 = Reply for all instructions.

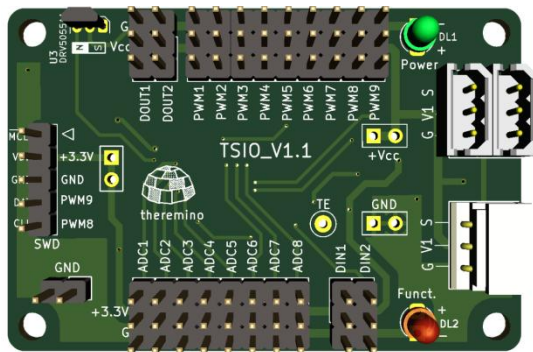
**(9) DEVICE ID** - This value is preserved across power cycles. To modify this value, the WriteProtect must be unlocked by setting it to zero. The maximum value is 199 because the broadcast codes range from 200 to 254.

**(10) SERIAL NUMBER** (3 bytes) + **PASSWORD** (12 bytes). After bootloader programming, the serial number is 0 and the password is not set (all bytes are 0). If you write a serial number without setting the password (all bytes are 0), the serial number will be programmed; this write operation can be repeated with different serial numbers until the password is set. If you write a serial number with the password set (not all bytes are 0), both the serial number and the associated password will be programmed. Subsequent serial number write operations will only succeed if associated with the same password. If the password is incorrect, the write operations will fail. To reset the password and serial number, the bootloader must be reprogrammed.

## TSIO communication module

This module can read and write analog and digital values, with the advantage that it is on the same communication line as the gearmotors, eliminating the need for additional connections from the PC or PLC

Currently, version 1.1 of the TSIO module can only communicate via One-wire, so it can only be used with the TCOM board and gearmotors in the One-wire version. There is also a version 1.2, which can be used on systems that communicate via RS485, but it will only be produced upon request.



All the details for communicating with the TSIO are summarized at this link:

[https://www.theremino.com/wp-content/uploads/files/Theremino\\_Motors\\_Help\\_ENG.pdf](https://www.theremino.com/wp-content/uploads/files/Theremino_Motors_Help_ENG.pdf)

## Gearmotor and TSIO board registers

Tables show the parameters used for TSIO I/O module communication, compatible with gearmotor communication lines.

TSIO v.xx		EGD1 v.xx	
SerialNumber,	000-003	SerialNumber,	000-003
DeviceVer,	003-001	DeviceVer,	003-001
DeviceSubVer,	004-001	DeviceSubVer,	004-001
FirmwareVer,	005-001	FirmwareVer,	005-001
FirmwareSubver,	006-001	FirmwareSubver,	006-001
BootLoaderVer,	007-001	BootLoaderVer,	007-001
ADC1,	010-002	ActualPosition,	010-004
ADC2,	012-002		
ADC3,	014-002	ActualTorque,	014-002
ADC4,	016-002	ActualAccel,	016-002
ADC5,	018-002	ActualVelocity,	018-002
ADC6,	020-002	ActualCurrent,	020-002
ADC7,	022-002	ActualVoltage,	022-002
ADC8,	024-002	ActualTemp,	024-001
		MovingStatus,	025-001
DIN1,	026-002	HwErrorStatus,	026-001
		IdSetHallSensor,	027-002
DIN2,	028-002	HallPhase (1..6),	029-001
ActualTemp,	030-002	MagSens1_A,	030-002
IdSetHallSensor,	032-002	MagSens1_X,	032-002
		MagSens1_Y,	034-002
		MagSens2_A,	036-002
		MagSens2_X,	038-002
		MagSens2_Y,	040-002
.....			
WriteProtect,	050-001	WriteProtect,	050-001
WorkingMode,	051-001	WorkingMode,	051-001
ReplayMode,	052-001	ReplayMode,	052-001
BaudRateIndex,	054-001	BaudRateIndex,	054-001
DeviceId,	055-001	DeviceId,	055-001
IdTolerance,	056-001	IdTolerance,	056-001
PWM1,	058-002	GoalPosition,	058-004
PWM2,	060-002		
PWM3,	062-002	GoalTorque,	062-002
PWM4,	064-002	GoalAccel,	064-002
PWM5,	066-002	GoalVelocity,	066-002
PWM6,	068-002	PositionPGain,	068-001
		PositionIGain,	069-001
PWM7,	070-002	PositionDGain,	070-001
		DeadBand,	071-001
PWM8,	072-002	ZeroSpeedCurrent,	072-001
PWM9,	074-002		
DOUT1,	076-002		
DOUT2,	078-002		
		MagSens1_GainAxis,	080-001
		MagSens1_Gain,	081-001
		MagSens1_OffsetX,	082-001
		MagSens1_OffsetY,	083-001
		MagSens2_GainAxis,	084-001
		MagSens2_Gain,	085-001
		MagSens2_OffsetX,	086-001
		MagSens2_OffsetY,	087-001
		MagSens_WorkMode,	088-001



## Wiring diagram when using only a PLC

This diagram is the simplest electrically, but it will require a lot of programming work on the PLC.

The PLC program will need to:

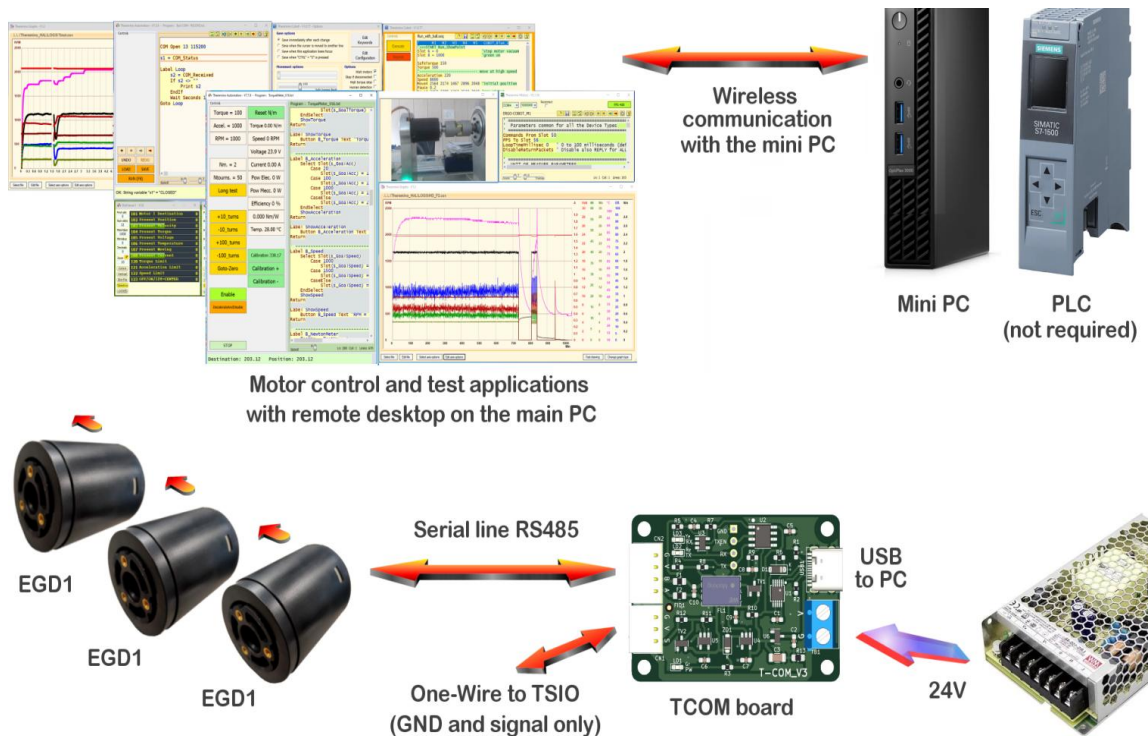
- Implement the communication protocol.
- Implement commands to read and write motor registers.
- Implement control functions, such as assigning IDs to motors.
- Manage serial communication with the motors.
- Send the destination to the motors, converting it from degrees to motor units.
- Read the achieved position and convert it to degrees.
- Read the temperature and other parameters of each motor and take them into account.
- Handle errors, perform necessary actions, and possibly alert the operators.
- Keep a log of the operations performed.

The information to write the program is all available in this document, and if necessary, you can also use parts of the open-source code we've written for the PC applications.

For example, instead of rewriting the function that creates the CRC, you could simply copy it and adapt it to the language used in the PLC. The same can be done with all other functions.

In any case, writing and testing this program could take an experienced programmer at least several months, and this time could be extended even more for complex projects.

## Wiring diagram when using a PC



We recommend using a PC in the control room and connecting it wirelessly to the Mini PC, which will not need a keyboard or screen.

The PLC shown in the image is not strictly required, but it could be useful for those accustomed to using them to control input/output signals (microswitches, relays, etc.). The PLC can then communicate with the PC in various ways, such as through text strings transmitted via serial and managed by our Automation application, or with other methods depending on the type of PLC used, such as APIs designed for controlling it.

If you are using the TSIO V1.1 module, be careful not to send 24 volts to it. Only connect the GND and signal to the TCOM board. The TSIO should be powered separately with a 5 to 12 V power supply, or with a StepDown converter that reduces the 24 volts to 5 volts.

In all cases, even with TSIO versions that support 24V, using a 5V StepDown would provide more current (up to 1 or 2 amps) to the controlled outputs, while using the TSIO's internal regulator limits the current to around 100 mA total.

## Connection interface

