OPERATOR PANEL
TOUCH SCREEN

TD320

Manuale
User manual

TABLE OF CONTENTS

Introduction

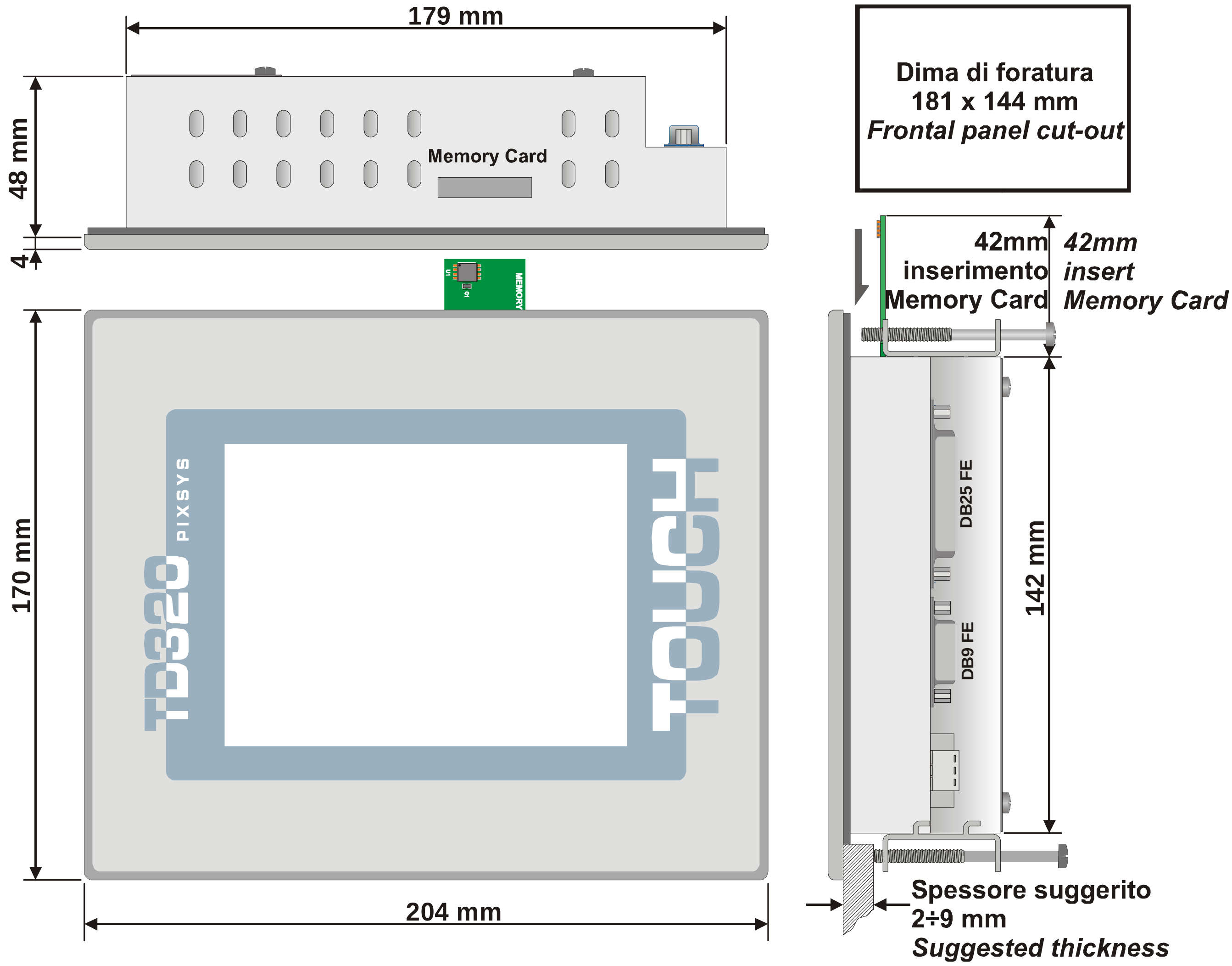
Thank you for having chosen a instrument. Model TD320 is a graphical touch screen terminal with an integrated PLC adapted for the supervision and control of systems where the participation of an operator (HMI) is necessary. The graphical resources are easily manageable from TdDesigner, a simple and versatile development environment, while the logic relative to the PLC is managed from the PLProg development environment, which is common to other devices (PL250, TCT500, etc).
The waterproof protection of the facade is IP54 and IP30 for the container.

Model identification

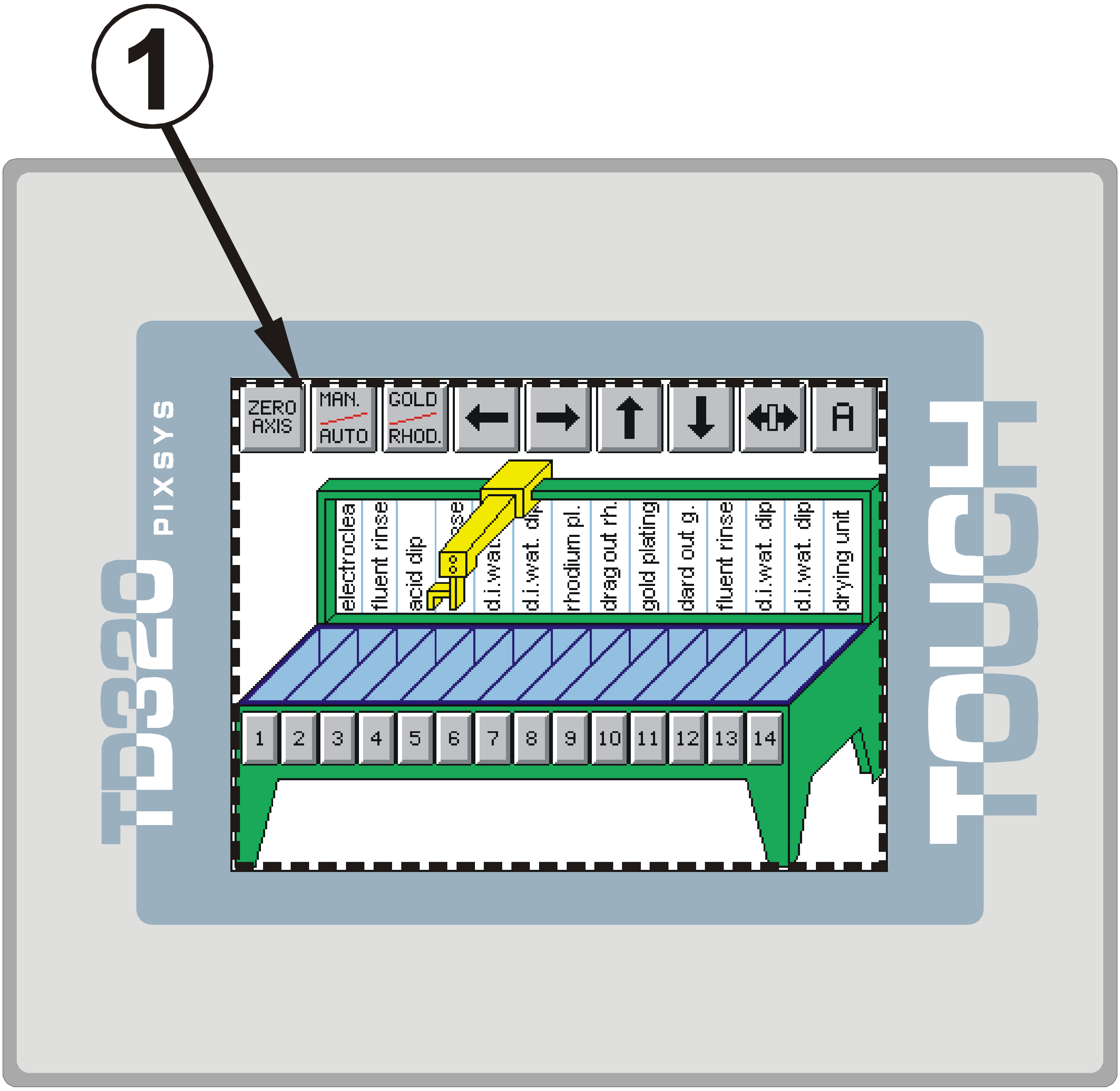
Only one version of the terminal TD320 is available, in low voltage, AC or DC. This is indicated in the model identification.

Ordering code		
TD320-	☐	
Power	**AD**	12…24V **AC/DC** ±15% 50/60Hz

1 Mechanical dimensions and installation



2 Display characteristics



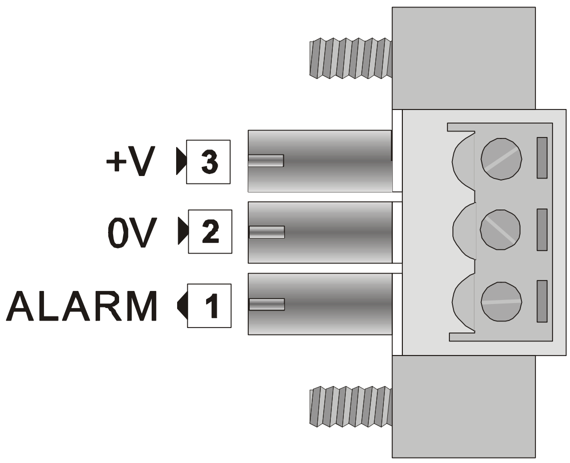
1	**DISPLAY**	**Type**: Back-lit LCD, resistive touch screen STN **Dimensions**: Active Area 5.7” 115.18(W)mm x 86.38(H)mm **Resolution**: 320x240 pixels **Colors**: 256 (8bit) **Importable Images**: bitmap of 256 colors (.bmp)

3 Electrical connections

Although this instrument is designed to resist the most difficult conditions present in industrial environments, it is good practice to observe the following precautions:

- Distinguish the line bringing live current from those of voltage
- Avoid the vicinity of groups of telecommand circuit breakers, electromagnetic contacts, and high-power motors.
- In particular, avoid the vicinity of power installations used to control phase.

3.1 Terminal clip connections



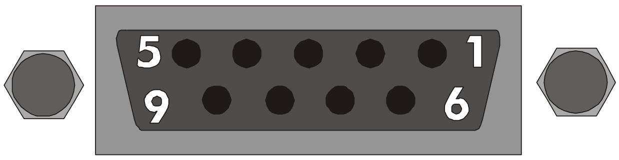
Power supply	
SUPPLY 12 to 24V ac/dc + 3 – 2	• 12…24V AC/DC ±15% 50/60Hz

Alarm output	
	With active contact (contact capacity **3A/250V~** resistive load) the voltage **+V** (power) is available between clips 1(+) and 2(-).

3.2 Serial ports of communication

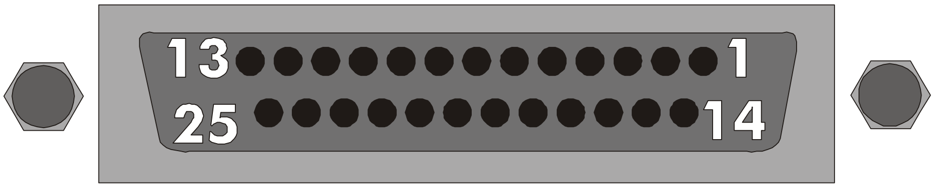
TD320 terminal communication with other devices is possible through serial connection with **RS485**, **RS232** and **RS422**.
The electrical signals are available in two connectors present at the back of the terminal: **post DB9** and **post DB25.**

DB9 FE



CONNECTOR	PIN NUMBER	SIGNAL	PORT
DB9 PINS	1	Not used	-
	2	RX – RS232 Program	COM2
	3	TX – RS232 Program	COM2
	4	RS485 -	EXP1
	5	GND RS485 / RS232	EXP1 / COM2
	6	TX – RS232	EXP1
	7	RX – RS232	EXP1
	8	Not used	-
	9	RS485 +	EXP1

DB25FE

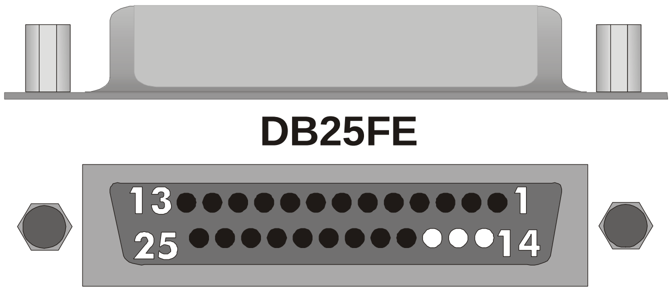


CONNECTOR	PIN N.	SIGNAL	PORT
DB25 PINS	1	Not used	-
	2	Not used	-
	3	Not used	-
	4	Not used	-
	5	GND (Common digital input)	-
	6	DI1 (digital input NPN)	-
	7	GND – RS232 Program	COM2 / EXP1
	8	DI2 (digital input NPN)	-
	9	DI3 (digital input NPN)	-
	10	DI4 (digital input NPN)	-
	11	RX – RS232 Program	COM2
	12	TX – RS232 Program	COM2
	13	DI5 (digital input NPN)	-
	14	GND isolated RS485 / RS422	COM1
	15	RS485+ / TX+ RS422	COM1
	16	RS485- / TX- RS422	COM1
	17	RX+ RS422	COM1
	18	RX- RS422	COM1
	19	DI6 (digital input NPN)	-
	20	DI7 (digital input NPN)	-
	21	RX – RS232	EXP1
	22	TX – RS232	EXP1
	23	DI8 (digital input NPN)	-
	24	Not used	-
	25	Not used	-

3.2.1 COM1 on DB25 pins

The communication port **COM1** is available in connection to 25 pins in interface RS485 or also RS422 (protocol, baud rate and format are settable).

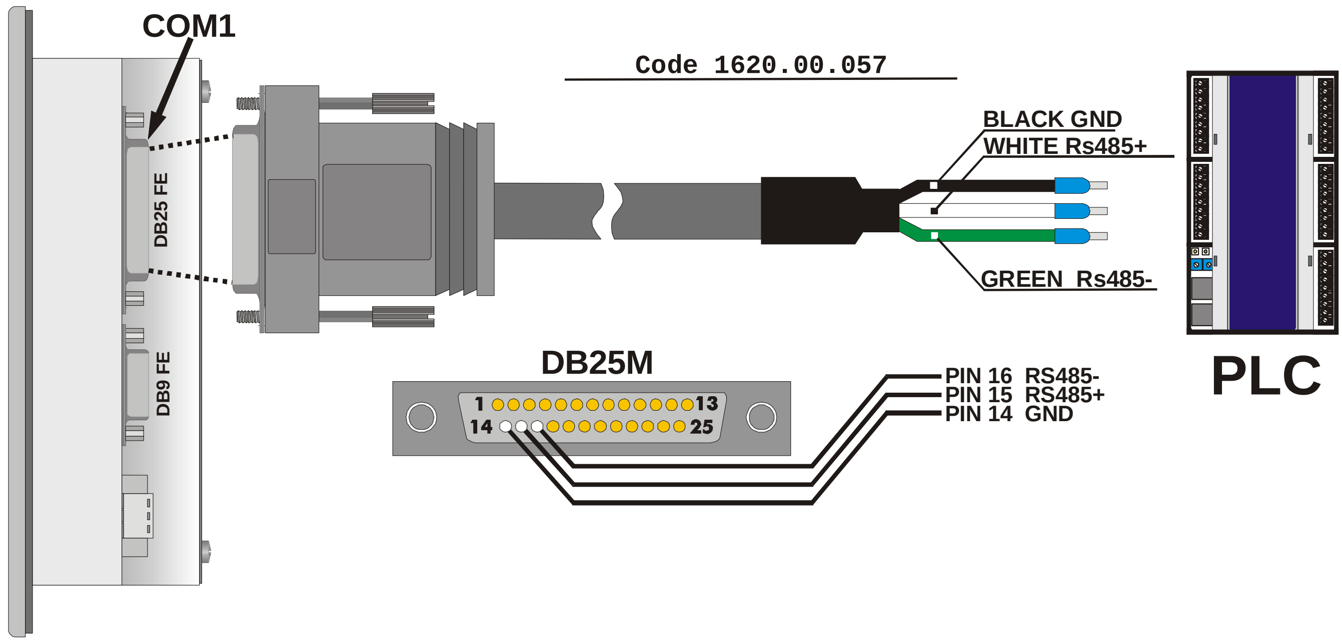
3.2.1.1 Interface RS485



Interface RS485 in DB25 (COM1)		
○	PIN 14	GND
○	PIN 15	RS485+
○	PIN 16	RS485-

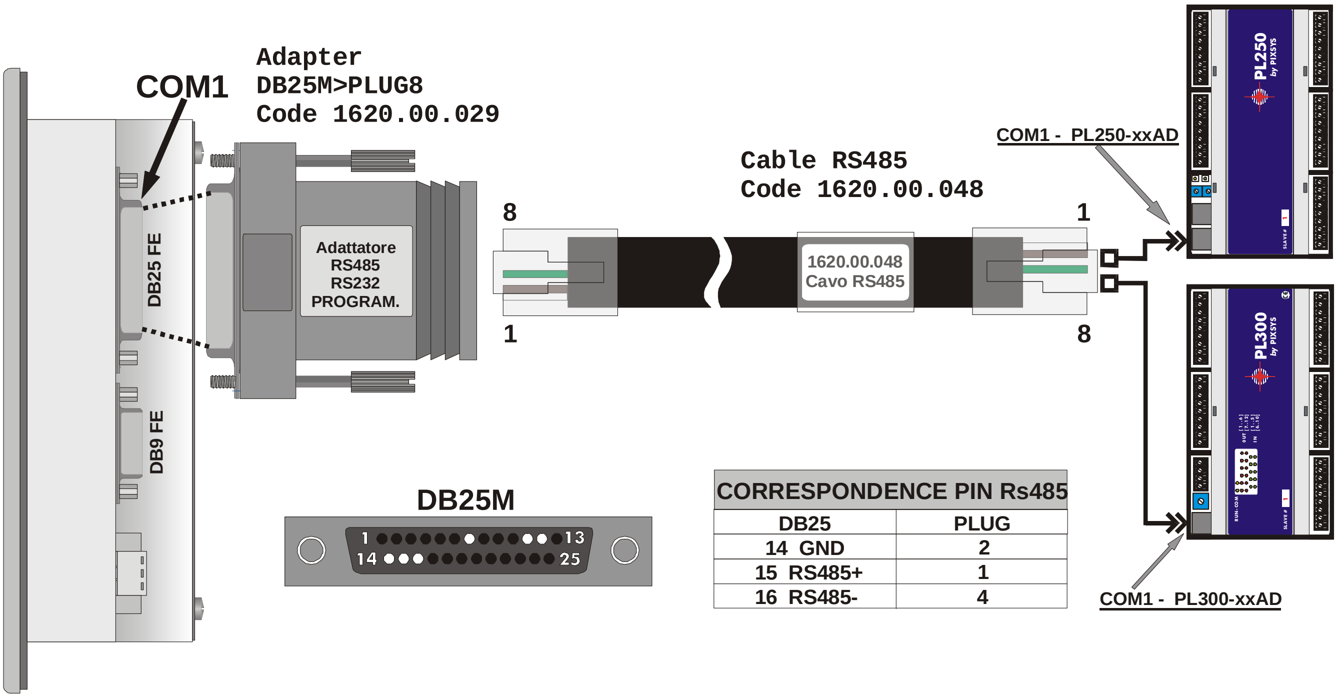
3.2.1.1.1 Cable for COM1 communications in generic RS485

A cable is available (code art. 1620.00.057, optional), that from connector DB25 provides COM1 in RS485 for a generic connection with other devices (for details regarding the communication protocols, consult other documentation).



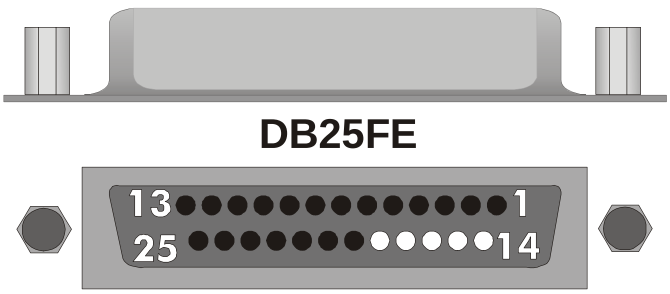
3.2.1.1.2 Cable for COM1 comm. in RS485 for PL250 / PL300

For communication with other devices (**PL250-XXAD** and **PL300-XXAD**) an (optional) cable is available that connects the port **COM1** in RS485 from connector DB25 of the terminal to the COM1 port on PLUG of the PLC.



CORRESPONDENCE PIN Rs485	
DB25	PLUG
14 GND	2
15 RS485+	1
16 RS485-	4

3.2.1.2 Interface RS422

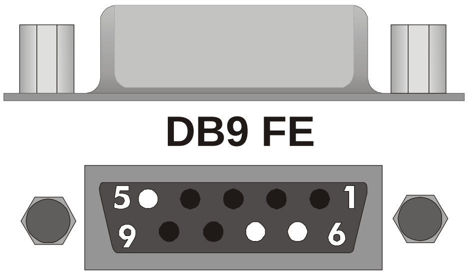


Interface RS422 in DB25 (COM1)		
○	PIN 14	GND
○	PIN 15	TX+
○	PIN 16	TX-
○	PIN 17	RX+
○	PIN 18	RX-

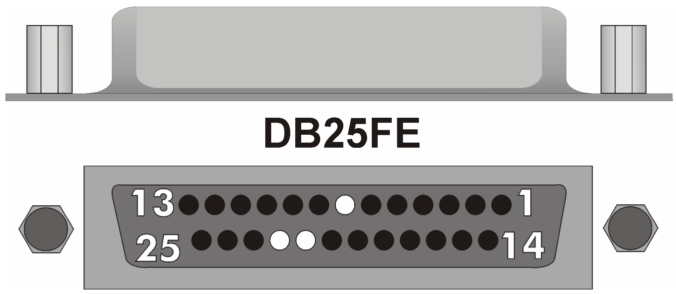
3.2.2 EXP1 on DB9 and DB25 connector pins

The communication port EXP1 is available in the DB9 connector pins in RS232 or RS485 interface and in the DB25 connector pins in RS232 interface (protocol, baud rate, and format are settable).

3.2.2.1 Interface RS232

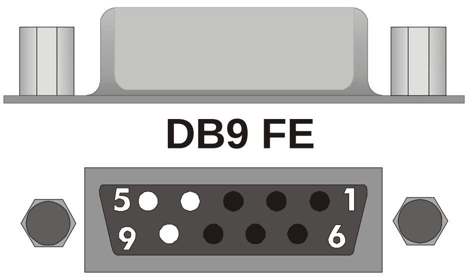


Interface RS232 in DB09 (EXP1)		
○	PIN 5	GND
○	PIN 6	TX - RS232
○	PIN 7	RX - RS232



Interface RS232 on DB25 (EXP1)		
○	PIN 7	GND
○	PIN 22	TX - RS232
○	PIN 21	RX - RS232

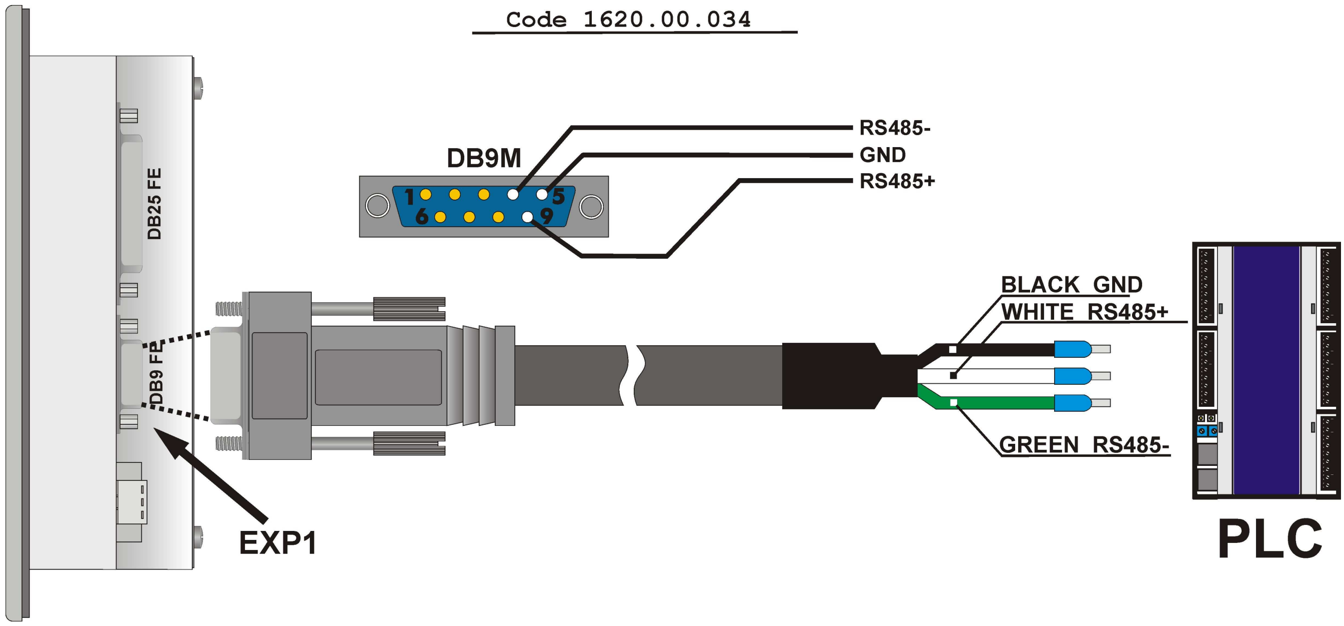
3.2.2.2 Interface RS485



Interface RS485 in DB09 (EXP1)		
○	PIN 5	GND
○	PIN 9	RS485+
○	PIN 4	RS485-

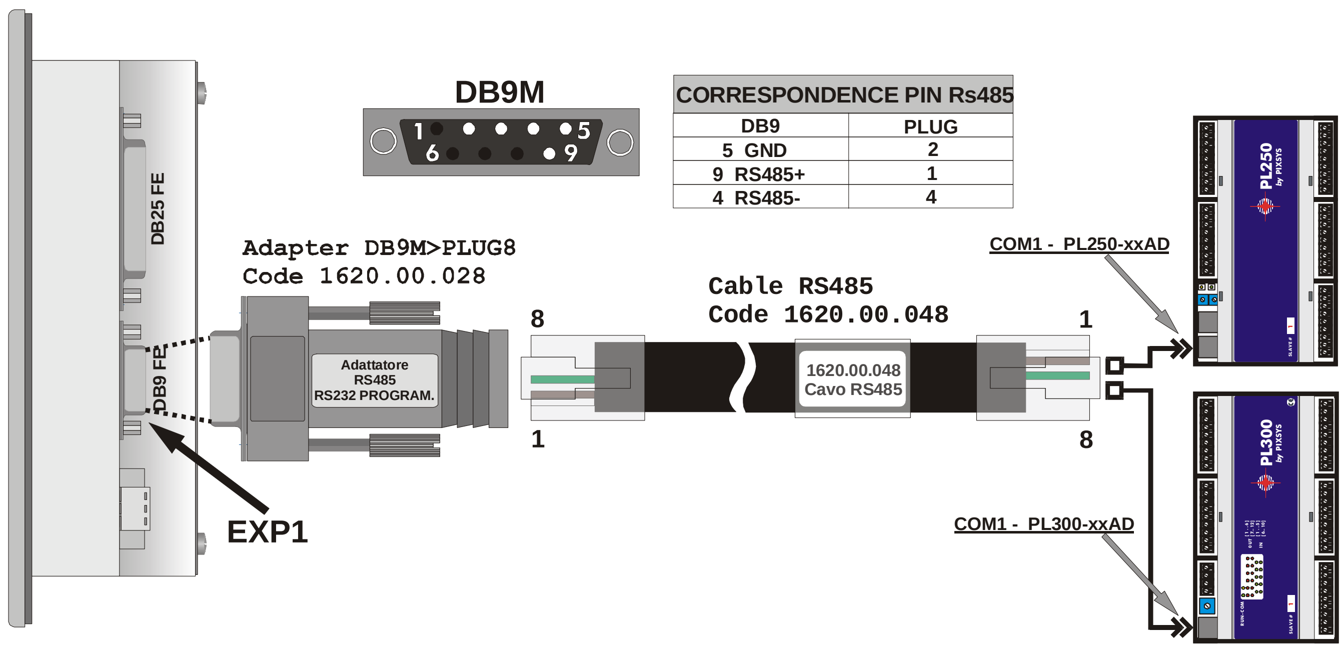
3.2.2.2.1 Cable of EXP1 communications in generic RS485

A cable is available (**code art. 1620.00.034**, optional), which provides EXP1 port from connector DB9 in RS485 for a generic connection with other devices (for details regarding the communication protocols, consult other documentation).



3.2.2.2.2 Cable of EXP1 comm. in RS485 for PL250 / PL300

For communication with other devices (**PL250-XXAD** and **PL300-XXAD**) an (optional) cable is available that connects port **EXP1** in RS485 from connector DB9 of the terminal to port COM1 on PLUG of the PLC..



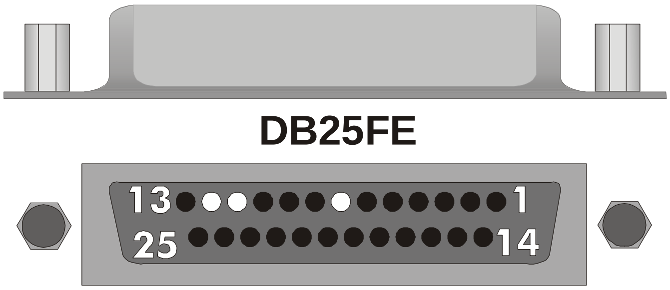
CORRESPONDENCE PIN Rs485	
DB9	PLUG
5 GND	2
9 RS485+	1
4 RS485-	4

3.2.3 COM2 Interface RS232

The communication port **COM2** is available either in the 25-pin connector or in the 9-pin connector, but only in interface RS232 (protocol **MODBUS SLAVE**, format **8,N,1**, baud rate settable).

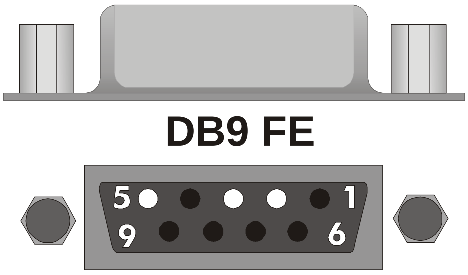
Usually this is the communications port used for programming the terminal through a PC (see Chapter 4).

3.2.3.1 COM2 on DB25 connector pins



Interface RS232 in DB25 (COM2)		
○	PIN 7	GND
○	PIN 12	TX - RS232
○	PIN 11	RX - RS232

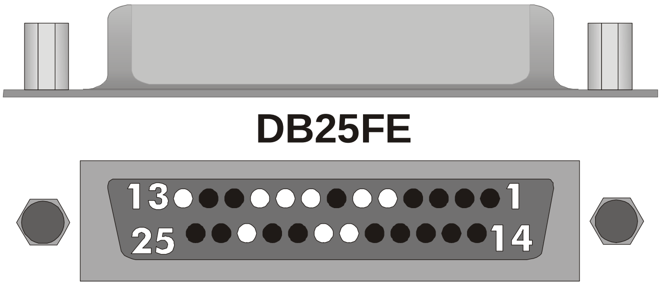
3.2.3.2 COM2 on DB9 connector pins



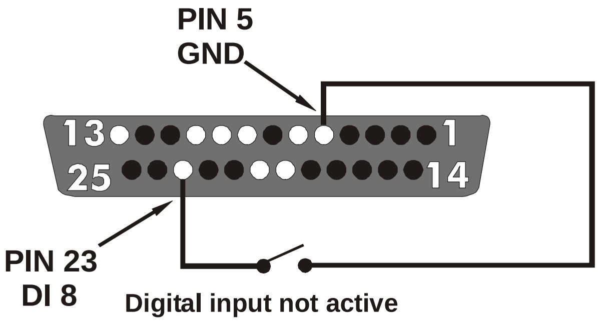
Interface RS232 in DB09 (COM2)		
○	PIN 5	GND
○	PIN 3	TX - RS232
○	PIN 2	RX - RS232

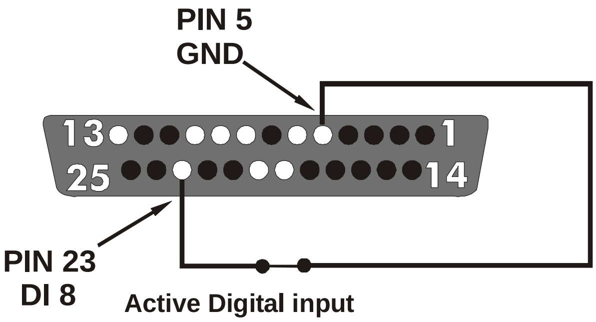
3.3 NPN digital inputs

In DB25 connector pins, 8 NPN digital inputs are present. The digital input is active if the respective PIN is short-circuited with GND (PIN 5, reference).



Digital inputs NPN		
○	PIN 5	Reference
○	PIN 6	DI 1
○	PIN 8	DI 2
○	PIN 9	DI 3
○	PIN 10	DI 4
○	PIN 13	DI 5
○	PIN 19	DI 6
○	PIN 20	DI 7
○	PIN 23	DI 8



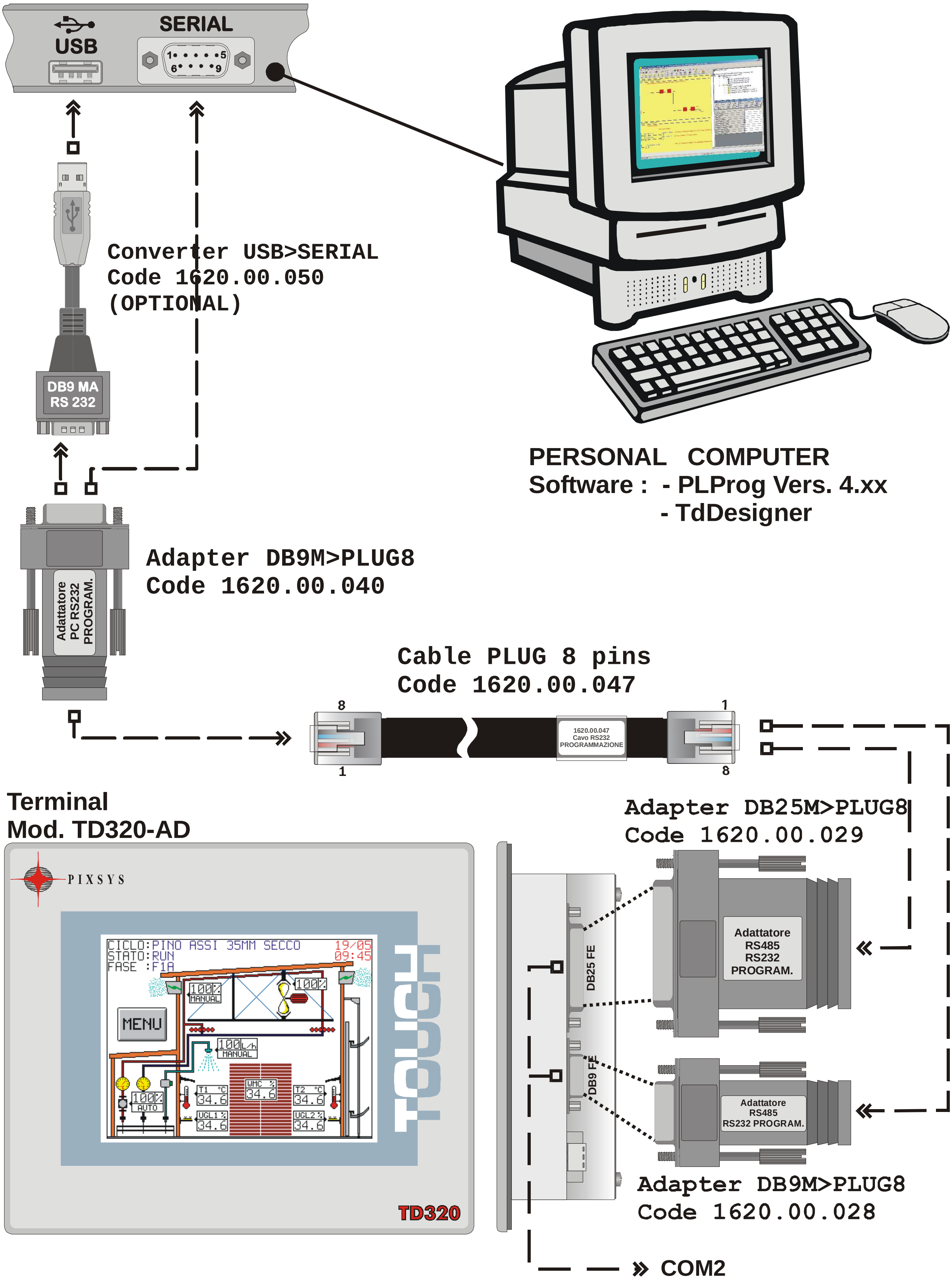


4 Programming the terminal

In order to program the terminal it is necessary to connect it to a PC. The development kit (optional, **code art. 2100.10.008**) provides the cable necessary for the connection and the development environment to create applications.

Programming involves the communication port **COM2**, present in both connectors, but only in interface RS232. There are 2 adapters on the side of the terminal that allow a user to program the terminal through connector DB25 or through DB9.

4.1 Starter Kit – Connection of the terminal to the PC



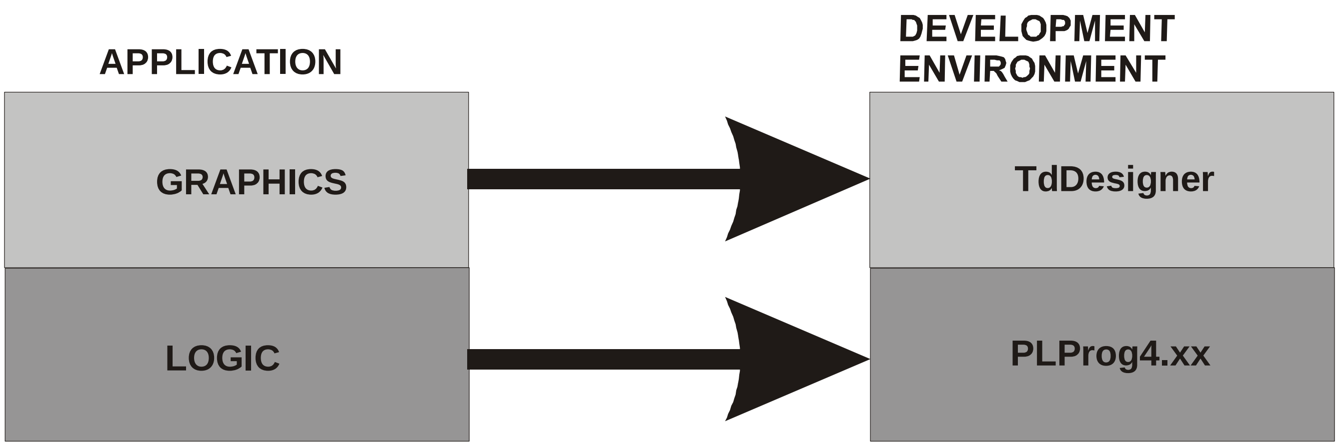
4.2 The development environment

The TD320 is a HMI graphical terminal with an integrated PLC. It allows a centralization of all the operational logic of the system that must be supervised and controlled.
The graphical part of the development environment must manage the visible pages and their fundamental items (e.g. synthesis, push-buttons, numerical and text edit boxes, images) and the interaction between various objects and the memory areas (the memory areas which they must reference for push-buttons, indicators and images).
The logic of the operation of the system, i.e. the way in which the memory areas must interact among each other, is instead managed by the PLC.

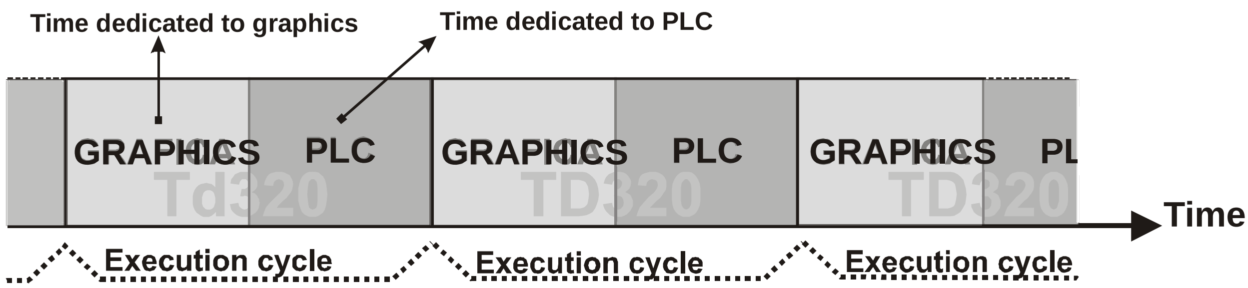
The TD320 terminal is also a PLC, therefore it manages graphics on one hand, and logic on the other, leaving other connected PLCs the sole task of "detecting the information" (e.g. digital and analog inputs, encoders etc.) and "to control the actuators" (e.g. digital and analog outputs etc.).

The development environment has two sub-environments:

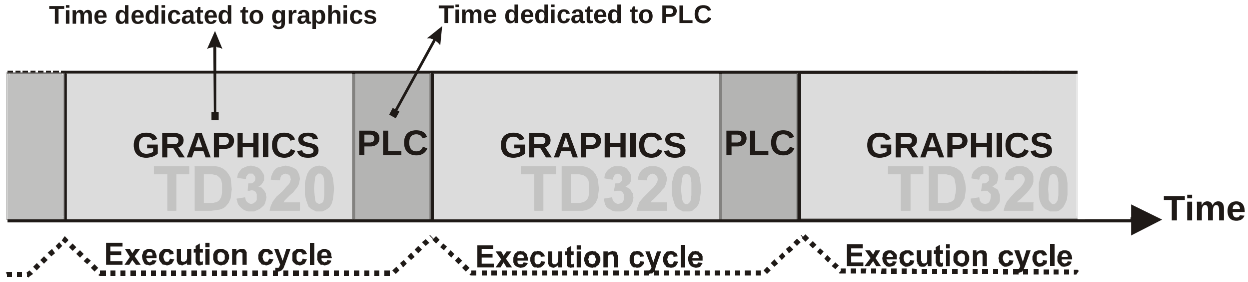
- **TdDesigner:** manages all resources that are strictly related to the graphics.
- **PLProg:** manages the interactions between the memory areas of the terminal (Ladder code, common to other PLCs, essentially the PL250 and TCT500).



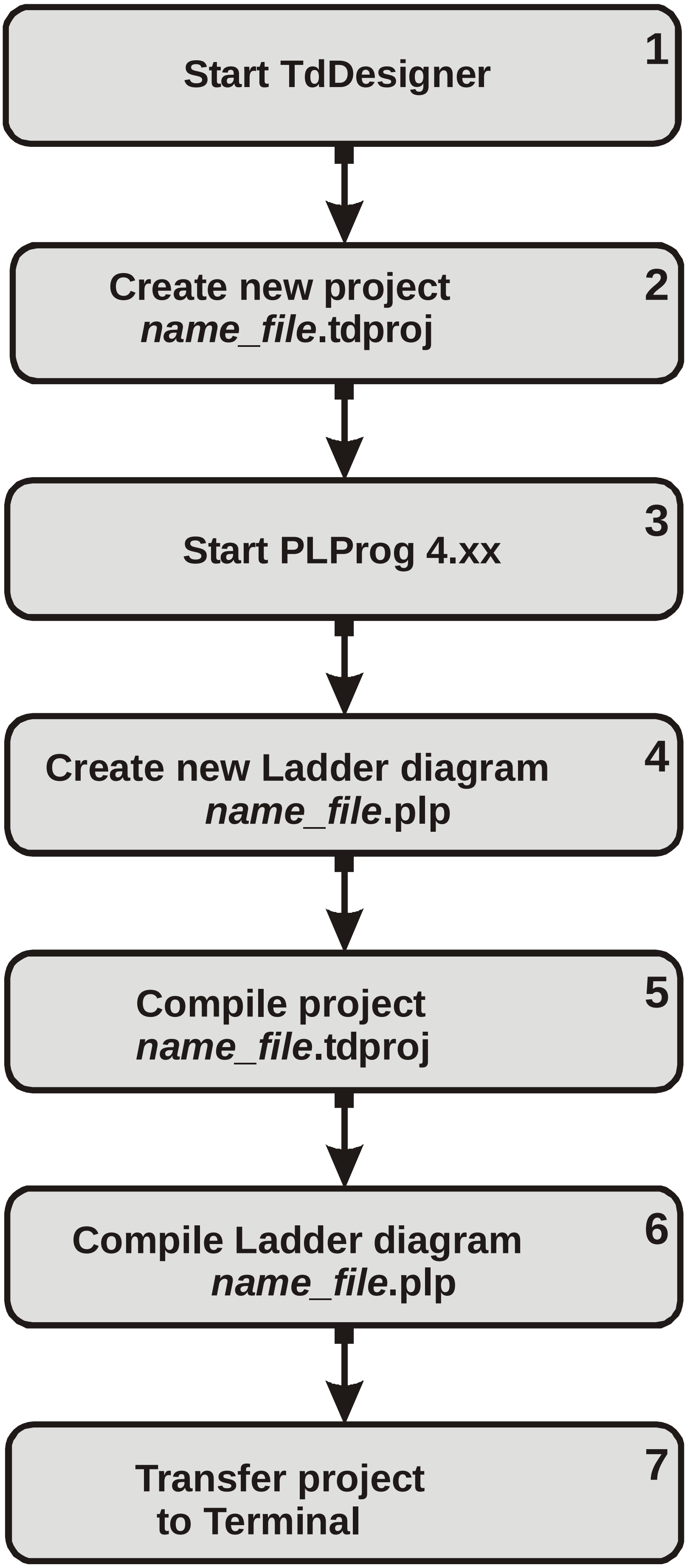
Any application managed by the TD320 terminal should therefore be realized using both the development environments, **implementing therefore two different files** strictly connected between them.
The operation of the terminal anticipates a division of the time dedicated to graphics management (implemented with TdDesigner) and of the time dedicated to the management of the PLC (implemented with PLProg 4.xx).
The default setup foresees an equal division of the execution cycle: the terminal will execute the instructions inherent for graphics for 50% of the time, and the Ladder instructions of the PLC for the other 50% (cyclically).



The time division is settable by the user (see chapter 5). An example is shown below in which 80% of the time is dedicated to the graphics and 20% to the PLC.



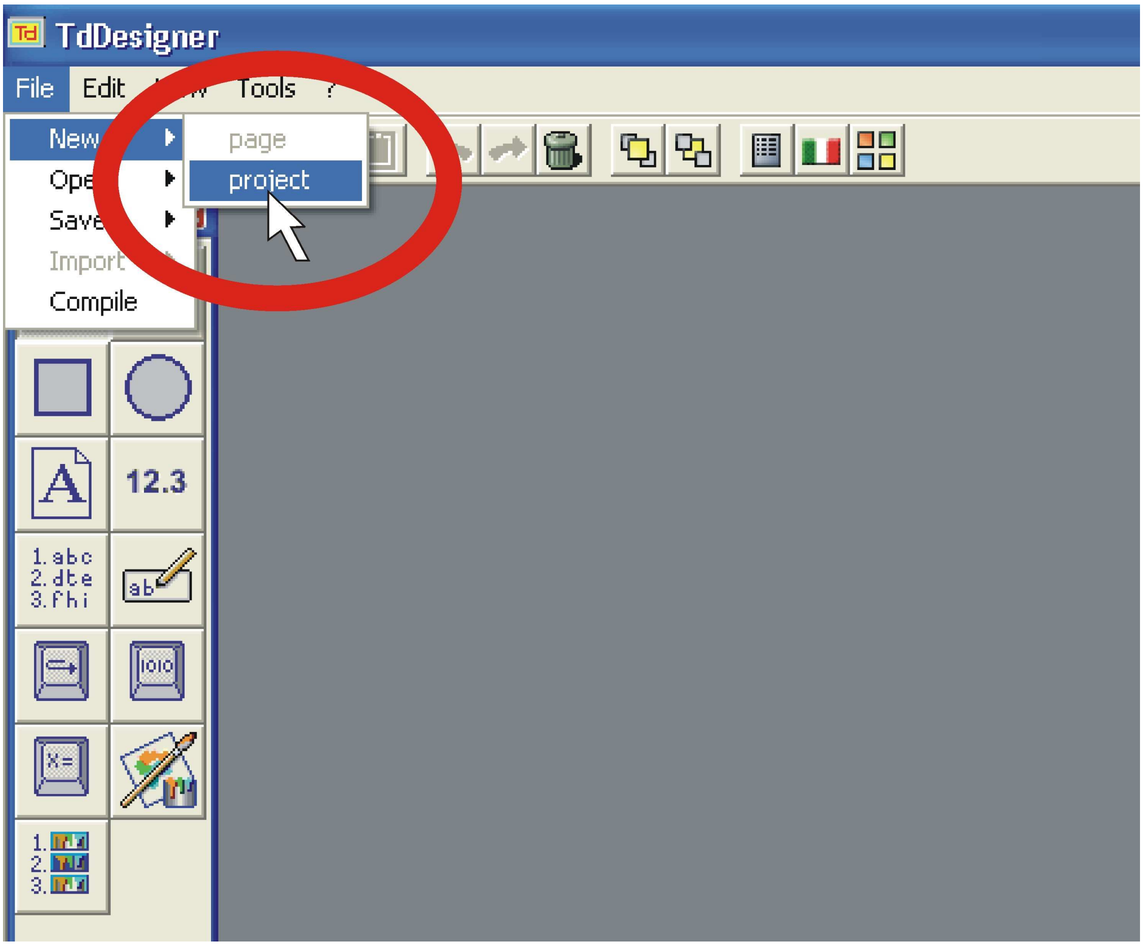
4.2.1 Creation of a new project



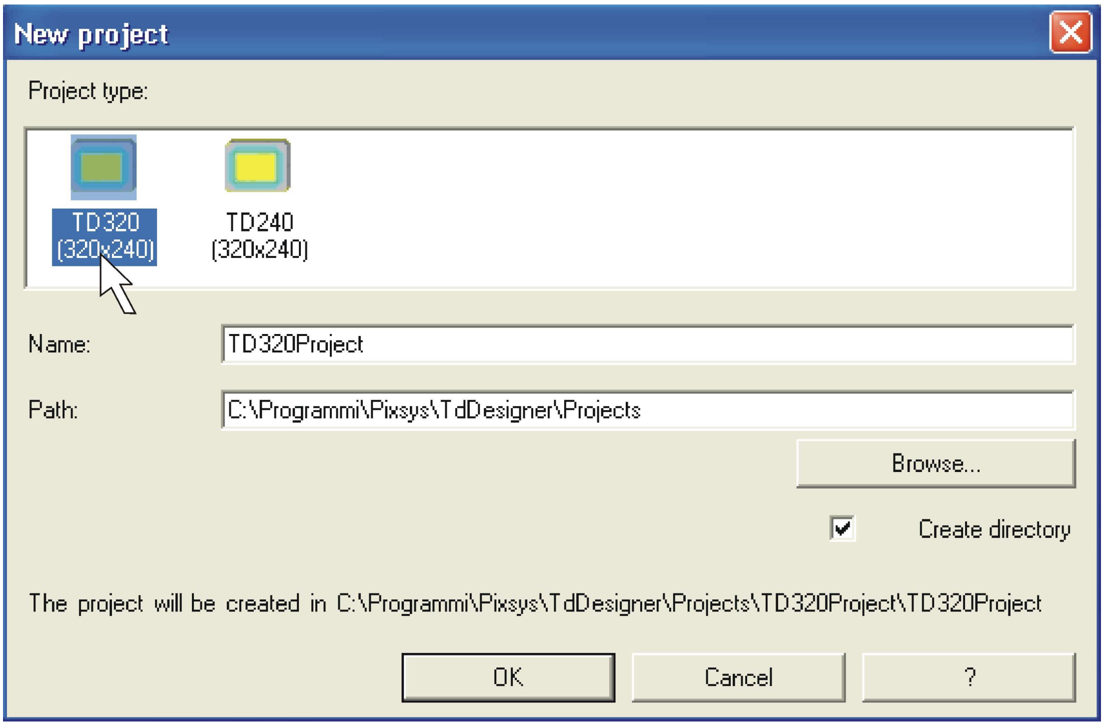
To create a new project and transfer it to the terminal, follow the procedure and described below:

1. **<u>Start TdDesigner</u>:** Start the **TdDesigner** software from the Start\Program menu or from the Desktop icon (automatically created at installation).

2. **<u>Create new project</u> *name_file*.tdproj:** Once the development environment is opened, create a new project as shown in the figure below:



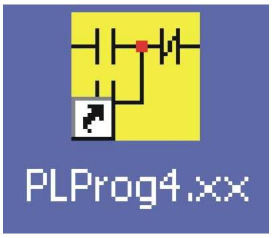
Select terminal TD320 (320x240 pixel display 5,7”).



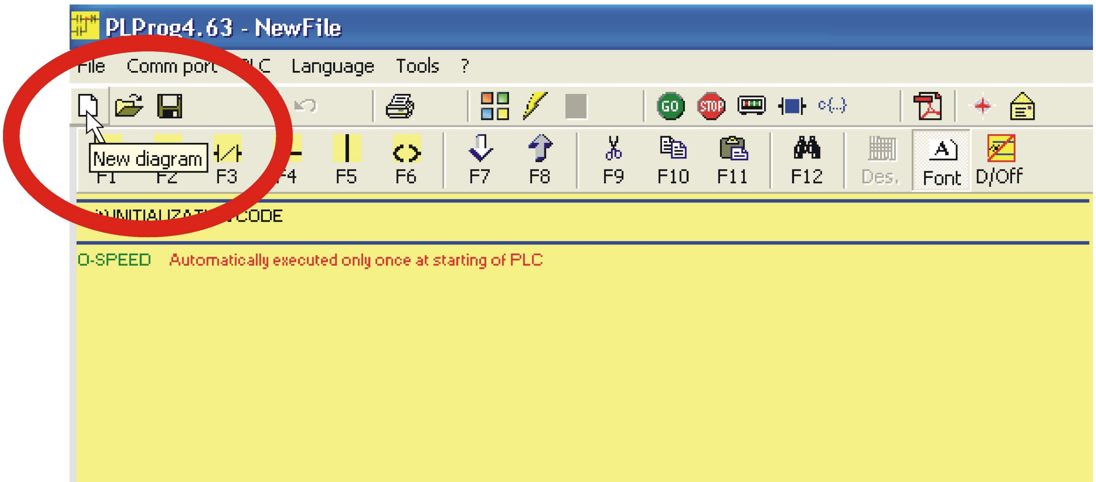
The development environment can put new project in a directory automatically created or in a folder chosen by user.

Graphics management is handled in other documentation, available with the development kit (**code art. 2100.10.008**) and assumed here as known by the user.

3. **Start PLProg 4.xx**: Start the **PLProg 4.xx** software from the Start\Program menu or the Desktop icon (automatically created at installation).



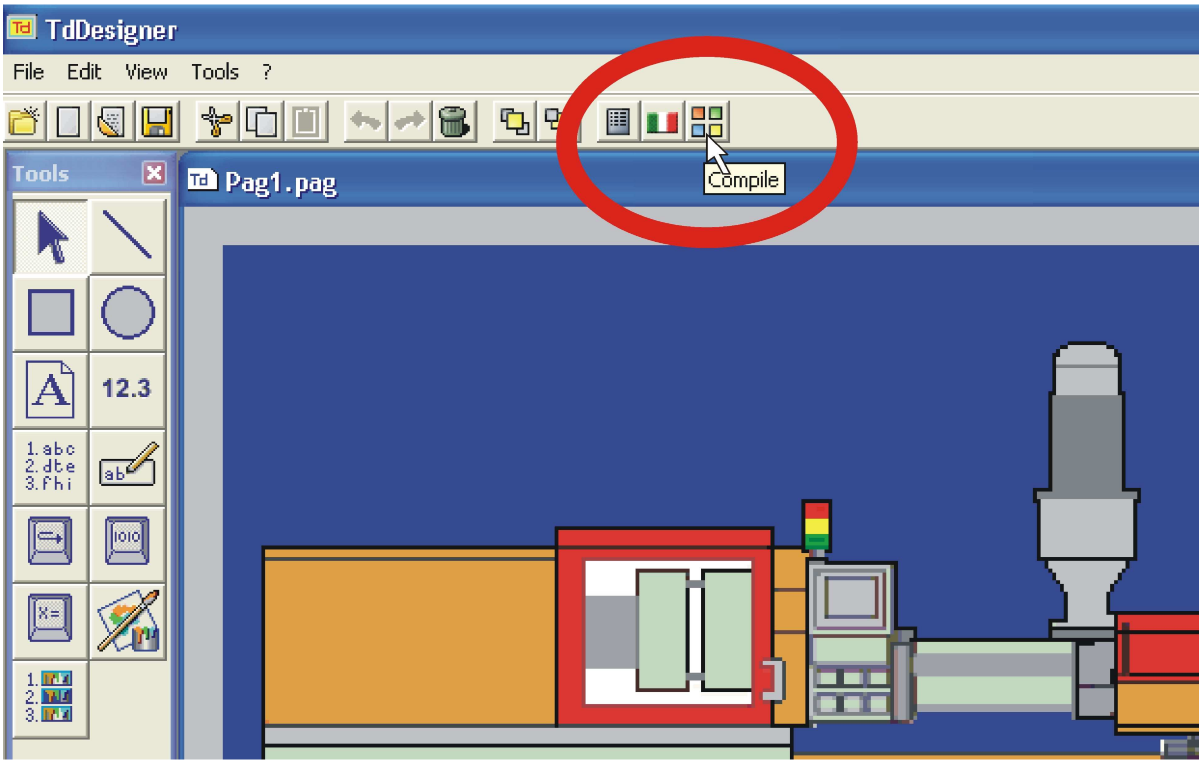
4. **Create new Ladder diagram file_name.plp**: Once the development environment is opened, create a new diagram as shown in the figure below:



A window will now open in the center of the screen: select the terminal TD320 in the item list <u>*Select CPU*</u>.

The guide to the software and the implementation of the Ladder code is available with the development kit (**code art. 2100.10.008**) and assumed here as known by the user.

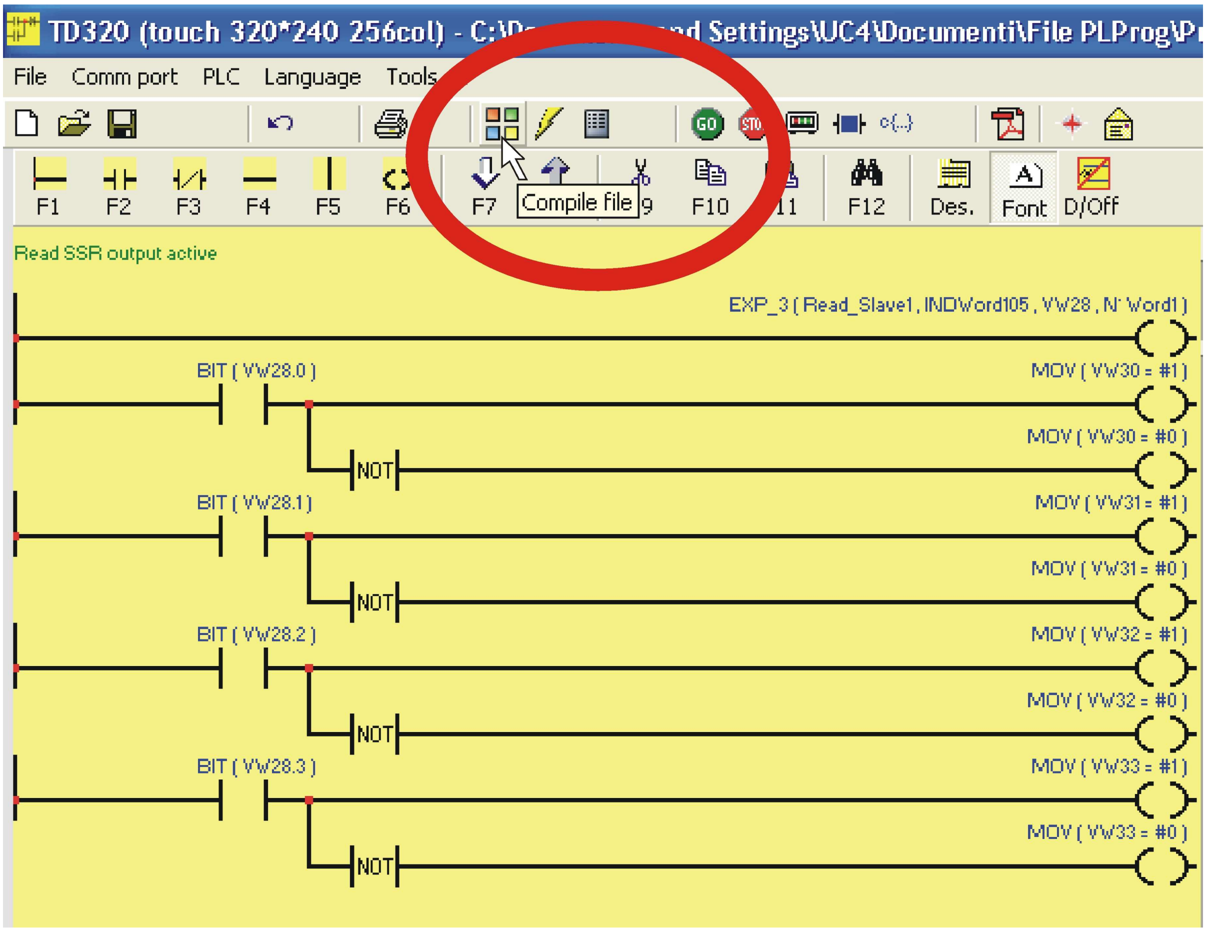
5. <u>**Compile project**</u> ***file_name*.tdproj:** Once the implementation of the graphics is finished, it is necessary to **compile** the project, as shown in the figure below.



This operation is necessary to make the project available as soon as implemented to the development environment **PLProg 4.xx**.
<u>The compilation has effect only if PLProg is open and the terminal TD320 has been selected as CPU</u>.

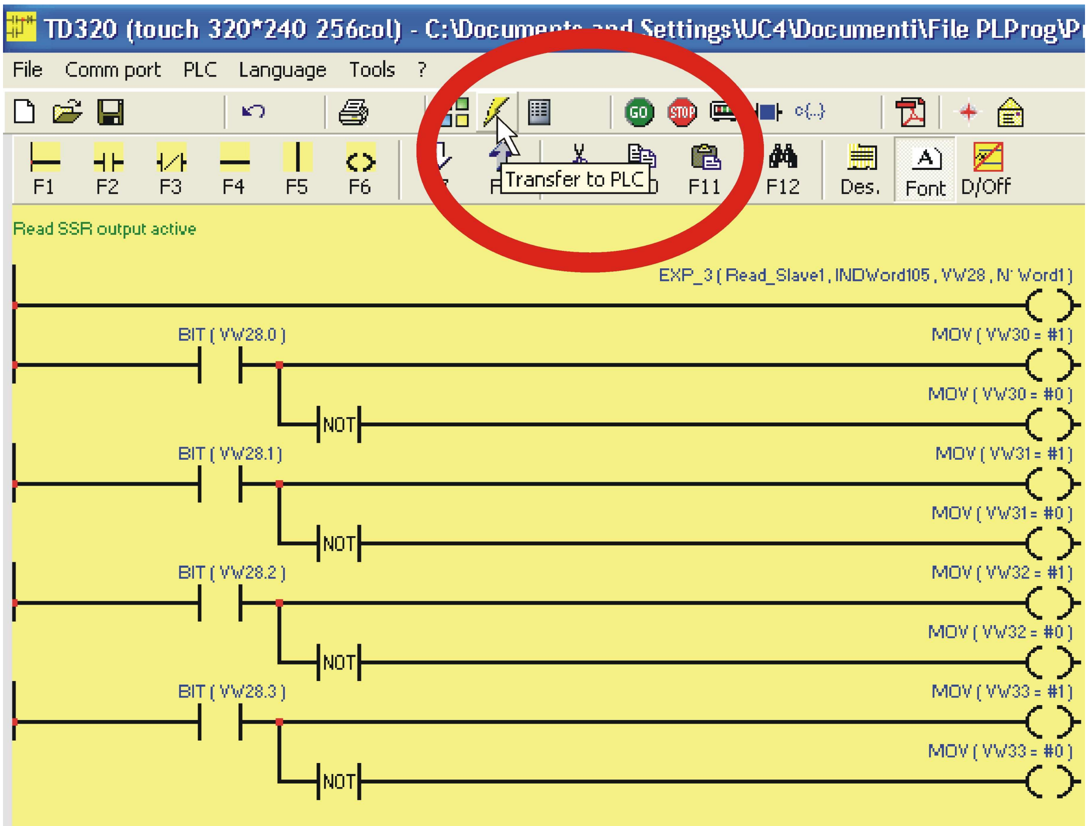
6. **<u>Compile Ladder diagram</u> *file_name*.plp:** Once the Ladder diagram sketch is finished, it is necessary to **compile** it, as shown in the figure below.
   It is this fundamental passage that creates the link between the Ladder file just compiled in the development environment of **PLProg4.xx** with the file previously compiled in the development environment of **TdDesigner.**

   Only with this operation will it be in fact possible to communicate to the terminal also the instructions inherent to the graphics of the created project.

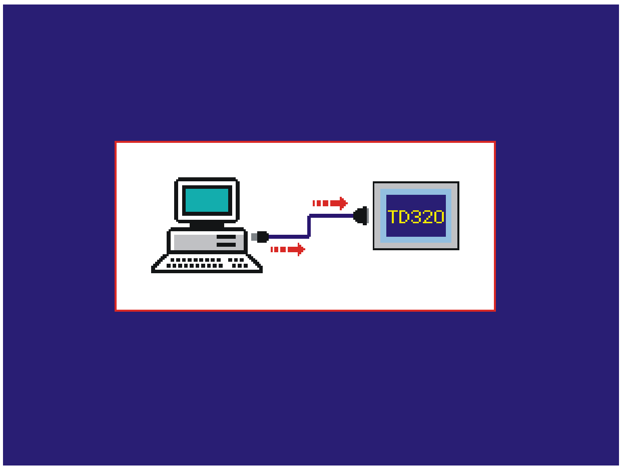


At this point, if saved by PLProg, file file_name.plp will contain both the PLC part and the graphical part (is not necessary that the file .tdproj has the same name of the file .plp).

7. **<u>Transfer the project to the terminal</u>:** If the compilation was successful, now one can carry out the download of the project, as shown in the figure below. The procedure transfers both the graphical part and the PLC part to the terminal



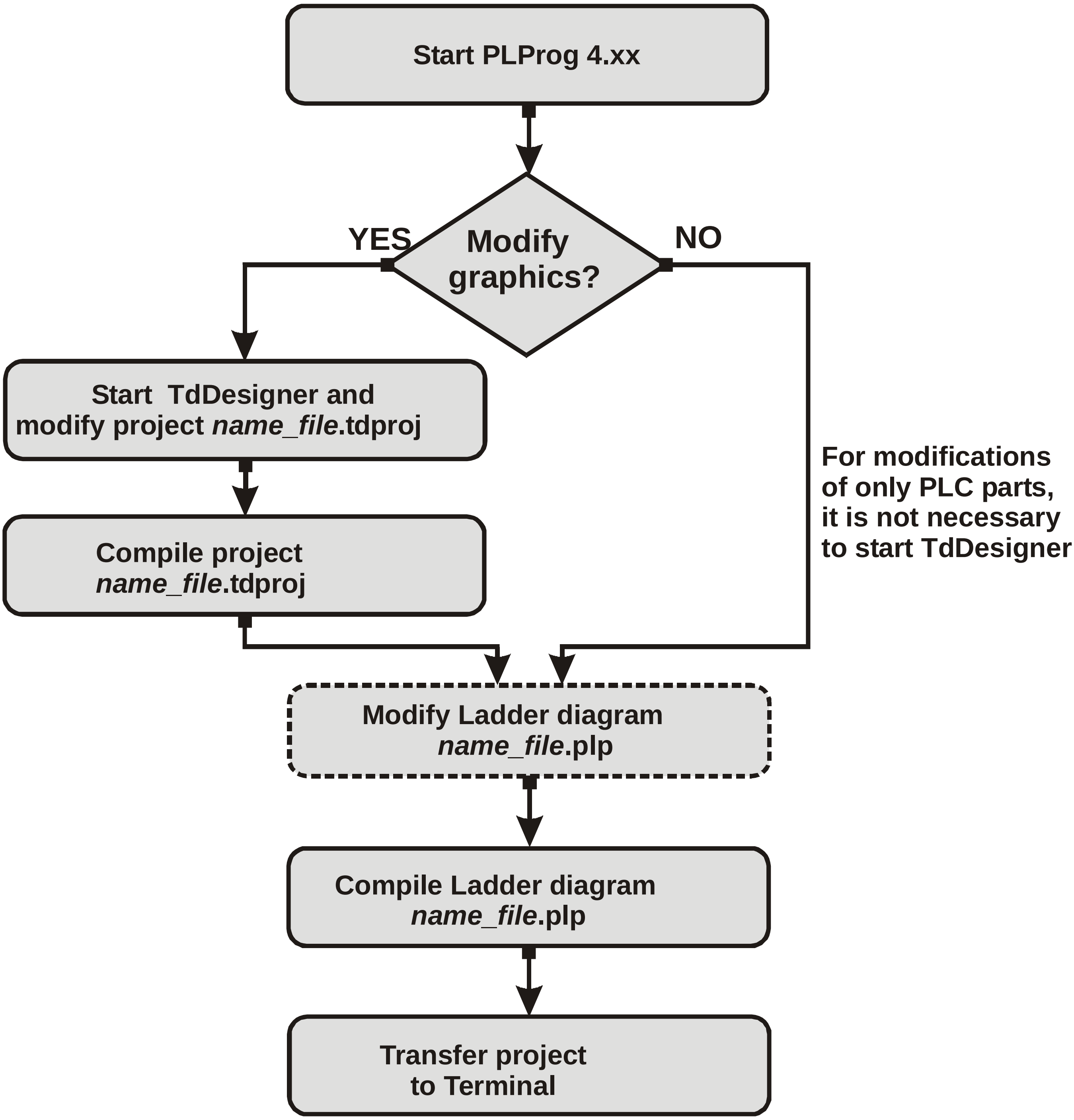
If the TD320 is connected correctly to the PC (see diagram of **section 4.1**), during the transfer the terminal will show this figure on the display:



At the end of the download, the terminal will execute the instructions of the entire application.

4.2.2 Modification of an already existing project

In the case in which an already existing project must be modified, follow the procedure below:



For eventual modifications of only the PLC part (as outlined) it is not necessary to start TdDesigner. The compilation of the project file_name.plp will maintain the graphics unchanged and will activate the modifications of the Ladder diagram.

5 Memory areas of the TD320

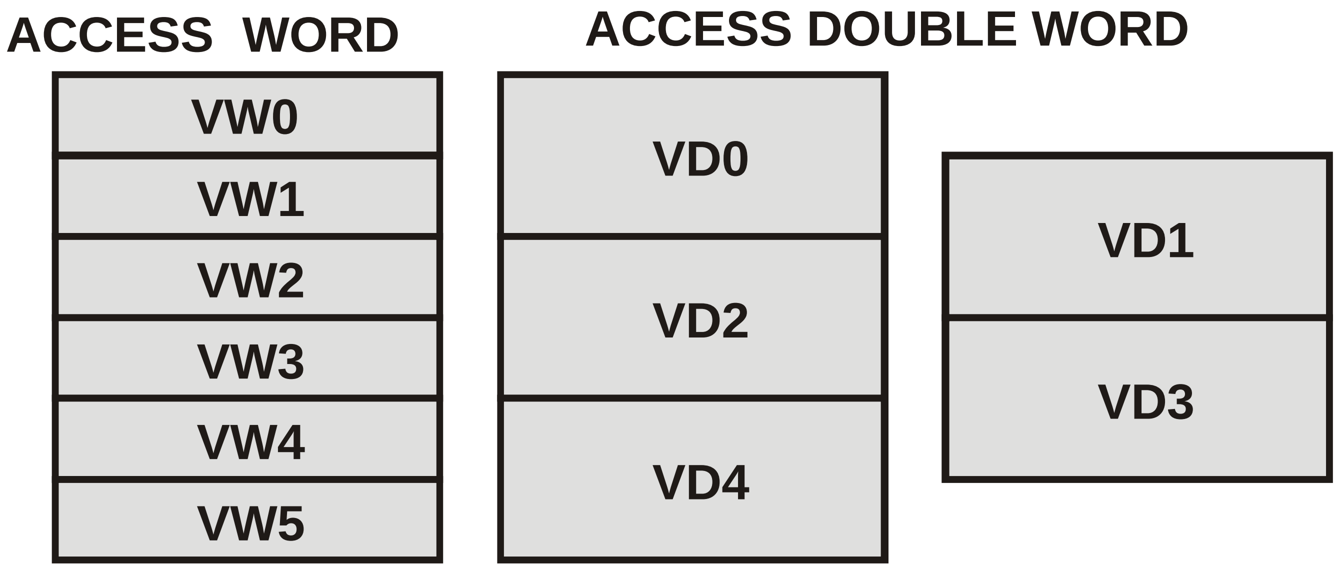
The TD320 makes memory areas available where it is possible to read or to write program data. Access to the various areas can occur through instructions that access a single bit (b), a byte (B), a word (W) or a double word (D).

SIGN	AREA	ACCESS
V	**Area of Variable V**	b, W, D
SM	Area of Special Marker	b, W, D
I	Area of Digital Inputs	b, W
AI	Area of Analog Inputs	b, W
Q	Area of Digital Outputs	b, W
M	Area of Marker	b, W
B	Area of Bistable	B
AQ	Area of Analog Outputs	b, W
T	Area of Timer	b, W
PT	Area of Preset Timer	b, W
C	Area of Counters	b, W
PV	Area of Preset Counters	b, W
EEP	Area of EEPROM	W
MMC	Area of EEPROM data	W
COM1	Area of buffer TX/RX port COM1	B
EXP	Area of buffer TX/RX port EXP1	B

5.1 Area of Variable V

Area variable V is a memory area used by the program to retain the data of the operations. It consists of 10000 locations of type word (5000 double word). Access can occur through operations on bits, words or double words. In the last case, the number of double words always makes reference to the organization by words, therefore in order to access consecutive double word variables it is necessary to increment by 2.

The memorized values are maintained even in the absence of power thanks to the rechargeable battery pad. Once charged, the battery maintains memorized data for approximately 6 months.



5.2 Area of Special Marker SM

Area special marker SM is the memory area used to retain all the data necessary for the Ladder program to interact with the TD320 hardware.
Some data are initialized at the start with default values indicated in the table below. In this area are the storage words that manage the events relative to the graphics, the PLC control bits and the setup for the serial ports of communication.
The table below describes the content of each single location of the special marker area, indicating the address for access through the ModBus protocol and the operation allowed at this location (R = read, W = write, R/W = read/write). The bits and words that do not appear in the tables are not used.

SM N°	ModBus Address	Description / Signification		
SM0	1000	**Bit state**		
		Bit 0	Bit RUN/STOP (1 = RUN). At startup this bit is always forced ON (PLC in RUN). In STOP the output relays of the PLC are disabled.	**R/W**
		Bit 1	Bit always ON for the first scan cycle of the main program. It becomes used, for example, to execute a subprogram of initialization.	**R**
		Bit 2	Bit that allows use of a 60-second clock impulse (ON for 30 seconds, OFF for 30 seconds).	**R**
		Bit 3	Bit that allows use of a 1-second clock impulse (ON for 0.5 seconds, OFF for 0.5 seconds).	**R**
		Bit 4	Bit clock of scan cycles that is active (ON) for a cycle and deactivated (OFF) for the successive cycle. It can be used as an input for counting scan cycles.	**R**
		Bit 6	Bit ON during the transmission phase of data on serial port COM1. It is automatically switched OFF at the end of the transmission.	**R**
		Bit 7	Bit ON during the transmission phase of data on serial port EXP1. It is automatically switched OFF at the end of the transmission.	**R**
		Bit 8	Bit ON during the transmission phase of data on serial port COM2. It is automatically switched OFF at the end of the transmission.	**R**

		Bit 9	This bit, if set ON, enables the serial port COM1 in "modem" mode. That means that the timeout between one character and another in reception is automatically fixed to 40 mS.	**R/W**
		Bit 10	This bit, if set ON, enables the serial port EXP1 in "modem" mode. That means that the timeout between one character and another in reception is automatically fixed to 40 mS.	**R/W**
		Bit 11	This bit, if set ON, enables the serial port COM2 in "modem" mode. That means that the timeout between one character and another in reception is automatically fixed to 40 mS.	**R/W**
SM1	1001	**Diagnostic bit anomaly / malfunction**		
		Bit 0	Bit ON in case of loss of data kept in the area "special marker" SM.	**R/W**
		Bit 1	Bit ON in case of loss of data kept in the area "variable V".	**R/W**
		Bit 2	Bit ON in case of loss of data kept in the area "EEProm".	**R/W**
		Bit 4	Bit ON in case of reset of the CPU or intervention of the watch-dog.	**R/W**
		Bit 5	Bit ON in case of stack overflow in the area reserved for RAM.	**R/W**
		Bit 7	Bit ON in case of anomaly / malfunction in the EEProm.	**R/W**
		Bit 8	Bit ON in case of anomaly / malfunction in the clock.	**R/W**
		Bit 10	Bit ON in case of stack overflow of the timer interrupt.	**R/W**
SM2	1002	**Device address**		
		Address (word) of ModBus protocol of the device. At startup, if ***SM1.0 = 1***, the value becomes initialized to "1", otherwise the previously saved data is maintained.		**R/W**

SM3	1003	**Cycle time**	
		Time of the last scan cycle of the program (resolution 100µS).	**R**
SM4	1004	**Minimal cycle time**	
		The minimal time found of the program scan cycle (resolution 100µS).	**R**
SM5	1005	**Maximum cycle time**	
		The maximum time found of the program scan cycle (resolution 100µS).	**R**
SM6	1006	**Interval of timer interrupt n°1**	
SM7	1007	**Interval of timer interrupt n°2**	
		Word that defines the interval of the timer interrupt. The value can be set between 1 and 100 ms (example:: SM6=1 → 1 ms, SM6=100 → 100 ms). For values of SM6 and SM7 not between 1 and 100, the correspondent interrupt is fixed to a default to 100 ms. At startup they are both fixed to a default of 100 → 100 ms. **In the Ladder code of the two interrupts, it is not allowed to use functions that access the areas of EEPROM and MMC**.	**R/W**
SM8	1008	**LCD contrast**	
		LCD display contrast 0…100 → 0…100%. At startup, if ***SM1.0 = 1***, the value is initialized to 50 → 50%, otherwise the previously saved data is maintained.	**R/W**
SM9	1009	**Minimal lamp time**	
		LCD back-lighting display 1…1000 → 1…1000 minutes, 0 → always lit. At startup, if ***SM1.0 = 1***, the value is initialized to 0 → always lit, otherwise the previously saved data is maintained.	**R/W**
SM10	1010	**Touch screen X**	
SM11	1011	**Touch screen Y**	
		Coordinates of the point of contact on the	**R**

		LCD display (X = 0…319, Y = 0…239) X=0, Y=0 → upper left corner When the display is not being touched, X = 500, Y = 500		
SM12	1012	**Touch screen FLAGS**		
		Bit 0	Bit ON in case of event: up, down or auto-repeat.	**R**
		Bit 1	Bit ON in case of down touch (pressure on the display).	**R**
		Bit 2	Bit ON in case of up touch (release of pressure on the display).	**R**
		Bit 3	Bit ON in case of touch pressure (continuous pressure on the display).	**R**
SM13	1013	**Language**		
		The number of languages for the text messages in the graphics is set from *TdDesigner* . This word defines the language for the currently visualized text messages (if ***n*** is the number of languages set by TdDesigner, SM13 can vary from ***0*** to ***n-1***). At startup, if ***SM1.0 = 1***, the value is initialized to 0 → first language, otherwise the selected language is maintained.		**R/W**
SM14	1014	**Number of visualized page**		
		Word that indicates the number of the visualized page (default 1, at startup the first page is always visualized).		**R**
SM15	1015	**Number of page to visualize**		
		Word that specifies the page number to visualize. Writing the number corresponding to a page physically created from the TdDesigner in this word will cause an immediate jump to that page; otherwise the visualized page will remain as it was before. After the page change, the word is set back to 0 automatically.		**R/W**

		At startup, if ***SM1.0 = 1***, the value is initialized to 0 → no change of page, otherwise the page previously chosen is maintained.	
SM16	1016	**Area of last variable modified**	
		Word that indicates (for a single scan cycle) the index corresponding to the last area of memory saved from the graphics. In detail, indices correspond to these areas:	**R**
		Area word V → 1 **Area word SM** → 2 **Area word AI** → 3 **Area word TR** → 4 **Area word AQ** → 5 **Area word I** → 6 **Area word Q** → 7 **Area word T** → 8 **Area word PT** → 9 **Area word C** → 10 **Area word PV** → 11 **Area double V** → 12 **Area double SM** → 13 **Area word M** → 14 **Area word EEPROM** → 15 **Area word MMC** → 16 **Area byte TX COM1** → 17 **Area byte RX COM1** → 18 **Area byte TX EXP1** → 19 **Area byte RX EXP1** → 20 **Area byte TX COM2** → 21 **Area byte RX COM2** → 22	
SM17	1017	**Memory area number of last variable modified**	
		Word that indicates (for a single scan cycle) the number of the last area of memory saved from the graphics. As an example, if the graphics modifies the variable **VW30**, , there will be, for the scan cycle following the modification, **SM16 = 1** and **SM17 = 30**. In	**R**

		the successive cycle the two areas will be automatically reset to 0.	
SM18	1018	**Time of buzzer activation (x10ms)**	
		Time buzzer is active in multiples of 10ms. The default value is 0xFFFF = 65536 = buzzer extinguished, which is set also at the end of the activation. If SM18 = 0, the buzzer will extinguish only by touch of the display.	**R/W**
SM19	1019	**TD320 digital outputs**	
		Bit 0 | Bit ON → Output relay active (between clips **1** and **2** of voltage **+V**)	**R/W**
SM20	1020	**CPU percentage for graphics**	
		Percentage of time used to execute instructions relative to the graphics. Possible values 10... 90 → 10…90%, default 50 → 50% (half time to graphics and half to PLC).	**R/W**
SM21	1021	**CPU percentage for graphics of page change**	
		Percentage of time used only to execute instructions relative to the change of a page. Once executed, the effective management of the time is decided by SM20. Possible values: 10…90 → 10…90%, default 50 → 50%	**R/W**
SM30	1030	**Seconds**	
		Internal clock seconds (0…59)	**R/W**
SM31	1031	**Minutes**	
		Internal clock minutes (0…59)	**R/W**
SM32	1032	**Hours**	
		Internal clock hours (0…23)	**R/W**
SM33	1033	**Days**	
		Internal clock days (1…31)	**R/W**
SM34	1034	**Months**	
		Internal clock months (1…12)	**R/W**
SM35	1035	**Year**	
		Internal clock year (0…99)	**R/W**
SM36	1036	**Day of the week**	
		Internal clock day of the week (0 → Sunday,	**R/W**

		6 → Saturday)		
SM37	1037	**Digital input filter**		
		It is possible to filter digital input signals by imposing a time of delay. If the state of the input changes, the new state will be accepted only if the input is maintained for the imposed time. The data will become accepted after the filter has eliminated disturbances and fixed the lines of the inputs on stable values. The TD320 supports filters with times of delay between 0 and 50ms (default 10ms).		**R/W**
SM38	1038	**Digital inputs**		
		Bit 0	Bit ON → Digital input DI1 active	**R**
		Bit 1	Bit ON → Digital input DI2 active	**R**
		Bit 2	Bit ON → Digital input DI3 active	**R**
		Bit 3	Bit ON → Digital input DI4 active	**R**
		Bit 4	Bit ON → Digital input DI5 active	**R**
		Bit 5	Bit ON → Digital input DI6 active	**R**
		Bit 6	Bit ON → Digital input DI7 active	**R**
		Bit 7	Bit ON → Digital input DI8 active	**R**
SM39	1039	**Conf. COM1 in mode Free-port**		
SM40	1040	**Conf. EXP1 in mode Free-port**		
SM41	1041	**Conf. COM2 in mode Free-port**		
		Word that enables the serial port to function in free-port mode and to set its parameters. Enabling this mode, the communications protocol using the serial port will be disabled, allowing direct access to the functions of transmission and reception of the data on the port. These parameters are initialized at startup to 0 (free-port mode disabled).		**R/W**
		Bit 0÷3	These bits set the communication velocity of the serial port in the free-port mode according to the following values (baud):	**R/W**

			0 → 110 6 → 4800 **1 → 150 7 → 9600** **2 → 300 8 → 19200** **3 → 600 9 → 28800** **4 → 1200 10 → 38400** **5 → 2400 11 → 57600**	
		Bit 4÷7	These bits set the format of serial port communication data in the free-port mode: 7-8 = number of bits of data, N = No parity control, O = Odd parity, E = Even parity, 1,2 = number of stop bits. **0 → 8, N, 1 6 → 8, N, 2** **1 → 8, O, 1 7 → 8, O, 2** **2 → 8, E, 1 8 → 8, E, 2** **3 → 7, N, 1 9 → 7, N, 2** **4 → 7, O, 1 10 → 7, O, 2** **5 → 7, E, 1 11 → 7, E, 2**	**R/W**
		Bit 8	Bit set to "1" enables the free-port mode. "0" returns the serial line control to the protocol selected during the programming phase.	**R/W**
SM42	1042	**Num. Byte in COM1 reception buffer**		
SM43	1043	**Num. Byte in EXP1 reception buffer**		
SM44	1044	**Num. Byte in COM2 reception buffer**		
		For each serial line, this word contains the number of valid characters present in the reception buffer. It is used in the free-port mode to control the number of characters received. Anything written to this word will set the value to zero, thus emptying the reception buffer.		**R/W**
SM45	1045	**COM1 serial baud rate (default 9600 baud)**		
SM49	1049	**EXP1 serial baud rate (default 9600 baud)**		
SM53	1053	**COM2 serial baud rate (default 57600 baud)**		

		The value that is set defines the communication velocity of the serial line for the ModBus protocol, if enabled (baud): Note: Because the modifications are active, it is necessary to set this word in the initialization code. In case no modification is made or if modifications are made in other parts of the program, the baud rate will remain at the default rate set at startup	**R/W**
		0 → 110 6 → 4800 **1 → 150 7 → 9600** **2 → 300 8 → 19200** **3 → 600 9 → 28800** **4 → 1200 10 → 38400** **5 → 2400 11 → 57600**	
SM46	1046	**COM1 serial format (default 8, N, 1)**	
SM50	1050	**EXP1 serial format (default 8, N, 1)**	
SM54	1054	**COM2 serial format (8,N,1 non modifiable)**	
		The value that is set defines the communications data format of the serial line for the ModBus protocol, if enabled. Note: Because the modifications are active, it is necessary to set this word in the initialization code. In case no modification is made or if modifications are made in other parts of the program, the baud rate will remain at the default rate set at startup.	**R/W**
		0 → 8, N, 1 6 → 8, N, 2 **1 → 8, O, 1 7 → 8, O, 2** **2 → 8, E, 1 8 → 8, E, 2** **3 → 7, N, 1 9 → 7, N, 2** **4 → 7, O, 1 10 → 7, O, 2** **5 → 7, E, 1 11 → 7, E, 2**	
SM47	1047	**COM1 RX/TX delay (default 20 mS)**	
SM51	1051	**EXP1 RX/TX delay (default 100 mS)**	
SM55	1055	**COM2 RX/TX delay (default 1 mS)**	

		The value set in mS defines: • **Protocol slave:** The minimum delay between the end of the serial reception of data coming from the master device to the start of transmission of the data of the reply from the TD320 (max 100 mS). • **Protocol master:** The maximum waiting period between the start of the transmission of interrogation data by the TD320, to the completed reception of the reply data from a slave device.	**R/W**
SM48	1048	**Num. Errors for signaling COM1**	
SM52	1052	**Num. Errors for signaling EXP1**	
SM56	1056	**Num. Errors for signaling COM2**	
		The value set in this word defines the number of consecutive communication errors after which an anomaly will be signaled in the respective bit of the word “Serial state”. The default value for all of the ports is 10.	**R/W**
SM57	1057	**Serial state COM1 1-16**	
SM58	1058	**Serial state COM1 17-32**	
SM59	1059	**Serial state COM1 33-48**	
SM60	1060	**Serial state COM1 49-64**	
SM61	1061	**Serial state COM1 65-80**	
SM62	1062	**Serial state COM1 81-96**	
SM63	1063	**Serial state COM1 97-112**	
SM64	1064	**Serial state COM1 113-128**	
SM65	1065	**Serial state COM1 129-144**	
SM66	1066	**Serial state COM1 145-160**	
SM67	1067	**Serial state COM1 161-176**	
SM68	1068	**Serial state COM1 177-192**	
SM69	1069	**Serial state COM1 193-208**	
SM70	1070	**Serial state COM1 209-224**	
SM71	1071	**Serial state COM1 225-240**	
SM72	1072	**Serial state COM1 241-256**	

		These words contain the state of COM1 serial communication. Each bit of each word signals a condition of missing communication (off line) or error for each of the data transmitted or received using the instructions COM_1-256 (for example, **SM66.9=1** indicates an error in the instruction number **COM-154(…)**). In the case of a serial line set to slave protocol, the error condition is signalled by putting a "1" in all of the bits of the word SM57. At startup, all of the words are initialized to 0.	**R**
SM73	1073	**Serial state EXP1 1-16**	
SM74	1074	**Serial state EXP1 17-32**	
SM75	1075	**Serial state EXP1 33-48**	
SM76	1076	**Serial state EXP1 49-64**	
SM77	1077	**Serial state EXP1 65-80**	
SM78	1078	**Serial state EXP1 81-96**	
SM79	1079	**Serial state EXP1 97-112**	
SM80	1080	**Serial state EXP1 113-128**	
SM81	1081	**Serial state EXP1 129-144**	
SM82	1082	**Serial state EXP1 145-160**	
SM83	1083	**Serial state EXP1 161-176**	
SM84	1084	**Serial state EXP1 177-192**	
SM85	1085	**Serial state EXP1 193-208**	
SM86	1086	**Serial state EXP1 209-224**	
SM87	1087	**Serial state EXP1 225-240**	
SM88	1088	**Serial state EXP1 241-256**	

		These words contain the state of EXP1 serial communication. Each bit of each word signals a condition of missing communication (off line) or error for each of the data transmitted or received using the instructions EXP_1-256 (for example, **SM80.4=1** indicates an error in the instruction number **EXP-117(…)**). In the case of a serial line set to slave protocol, the error condition is signalled by putting a "1" in all of the bits of the word SM73. At startup, all of the words are initialized to 0.	**R**
SM89	1089	**Serial state COM2 1-16**	
SM90	1090	**Serial state COM2 17-32**	
SM91	1091	**Serial state COM2 33-48**	
SM92	1092	**Serial state COM2 49-64**	
SM93	1093	**Serial state COM2 65-80**	
SM94	1094	**Serial state COM2 81-96**	
SM95	1095	**Serial state COM2 97-112**	
SM96	1096	**Serial state COM2 113-128**	
SM97	1097	**Serial state COM2 129-144**	
SM98	1098	**Serial state COM2 145-160**	
SM99	1099	**Serial state COM2 161-176**	
SM100	1100	**Serial state COM2 177-192**	
SM101	1101	**Serial state COM2 193-208**	
SM102	1102	**Serial state COM2 209-224**	
SM103	1103	**Serial state COM2 225-240**	
SM104	1104	**Serial state COM2 241-256**	
SM105	1105	**COM1 time-out number**	
SM107	1107	**EXP1 time-out number**	
SM109	1109	**COM2 time-out number**	
		If the corresponding port Is set to a Master protocol, this indicates the number of non-received packets of information during the communication. At startup, all counts are initialized to 0.	**R**
SM106	1106	**COM1 number of errors**	

SM108	1108	**EXP1 number of errors**	
SM110	1110	**COM2 number of errors**	
		If the corresponding port Is set to a Master protocol, this indicates the number of packets of information with errors during the communication. At startup, all counts are initialized to 0.	**R**
SM111	1111	**COM1 minimum delay for new transmission**#	
SM112	1112	**EXP1 minimum delay for new transmission**#	
SM113	1113	**COM2 minimum delay for new transmission**#	
		If the corresponding port is set to a Master protocol, this sets the minimum delay for a new transmission after the reply of a slave device. Possible values 0…100 → 0…100ms, default 5 → 5ms	**R/W**

5.3 Area of Digital Input I

Memory area I is composed of **32 words** and can be used to contain the state of the digital inputs read through the serial lines of other devices.
It is organized in words: each of the 16 bits of a word can represent the state of an input. It is accessible also in bits, in order to allow the control of each single input.

5.4 Area of Digital Output Q

Memory area Q is composed of **32 words** and can be used to contain the state of the digital outputs to then write them on serial lines of other devices.

Available from firmware version 1.12

It is organized in words: each of the 16 bits of a word can represent the state of an output. It is accessible also in bits, in order to allow the control of each single output.

5.5 Area of Marker M

Memory area M is comprised of **50 words** and contains the state of all the markers (contact bits) used in the program.
It is organized in words: each of the 16 bits of a word represents the state of a marker. For example, the state of the marker M5 is memorized in the bit 4 of word 1 in memory area M. The marker M5 is thus accessible as M1.4 (contact bit of the word), but also as single bit M5 (contact or electrical relay coil).

5.6 Area of Analog Inputs AI

Memory area AI is composed of **32 words** and can be used to contain the state of the analog inputs read from the serial lines of other devices.
It is organized in words: each can represent the state of an analog input

5.7 Area of Analog Outputs AQ

Memory area AI is composed of **32 words** and can be used to contain the state of the analog outputs read from the serial line of other devices
It is organized in words: each can represent the state of an analog output.

5.8 Areas of Timer T and Preset Timer PT

The area of memory for timer T is composed of **128 words**. If the timer is enabled, the variation of the contents of the area of memory is regulated by the type of timer, which is set at the moment of activation.

The area of memory for preset timer PT is composed of **128 words** and contains the values of activation of the contacts (preset) of the respective timers.
The areas are organized in signed words, thus the resolution of the timer and the preset timer is 16 bits (+32767).

5.9 Area of Counters C and Preset Counters PV

The memory area for counters C is composed of **64 words**. If a counter is enabled, the variation of the contents of the memory area is regulated by the type of counter.
The memory area for preset counters PV is composed of **64 words** and contains the values of activation of the (preset) contacts of the respective counters.
The areas are organized in words, thus the resolution of the counters and preset counters is 16 bits (from -32768 to +32767).

5.10 Area of Bistable Relay B

The area of memory for bistable relay B is composed of **128 bits**. It is organized by bits, thus each bistable relay is individualized by a single bit.

5.11 Area of EEProm

The area of memory EEProm is composed of **1000 words**. This memory is storage for data that must be maintained even if the TD320 remains off for very long periods (over 6 months). The data saved in this area are in fact tested at startup to verify their integrity, and any anomalies are signalled by activating the bit **SM1.2**, causing the initialization of the entire area to 0.
Access and writing to this area require a time significantly longer than any other (order of 30/40 mS), thus it is advisable not to use it for continual access (there is also a limit to the number of times that an EEProm cell can be written to, of an order of 1,000,000 times), but only to copy at startup the data stored here, for

example to memory area V, and then use area V for an access that is more rapid (order of 5/10µs).

5.12 Area of MMC

The memory area MMC is composed of **3000 words**. This is the memory storage where it is possible to save large quantities of data and maintain it even in the absence of power.

The memory is of type EEProm. The resulting access is thus slower than area V and SM and the TD320 executes no control of the integrity of the data stored in this area.

5.13 Area of TX/RX COM1

The memory area TX/RX COM1 is composed of **200 bytes**. This area is used to manage the data in transit on serial port COM1.
The first 100 bytes (TX-0…TX-99) are used to load the data to transmit; the last 100 bytes (RX-0…RX-99) are used to save the data received by the serial port COM1.
These bytes are useful only in the free-port mode, while in normal mode they are managed directly by the protocol selected in the programming phase.

5.14 Area of TX/RX EXP1

The memory area TX/RX EXP1 is composed of **200 bytes**. This area is used to manage the data in transit on the serial port EXP1.
The first 100 bytes (TX-0…TX-99) are used to load the data to transmit, the last 100 bytes (RX-0…RX-99) are used to save the data received by the serial port EXP1.

These bytes are useful only in the free-port mode, while in normal mode they are managed directly by the protocol selected in the programming phase.

6 Communication protocols

The TD329 can communicate with all devices that support the following serial protocols:

- ModBus RTU
- Nais Matsushita master

The terminal has 3 serial ports of communication (COM1, EXP, COM2), analyzed from the electrical point of view in chapter 3. Nonetheless, the ports are each managed in a different manner and will be analyzed separately.

6.1 Managing the communication port

The communication between the TD320 and other devices is managed by the PLC part of the terminal, thus the configuration of the port and the instructions must be implemented in the development environment PLProg 4.xx.
Generally the coils of the Ladder diagram are executed following the sequential order written in the diagram itself. The instruction related to the coil at line “n+1” is not executed until the full completion of the instruction related to the coil at line “n” (for coils positioned in the same column).

The control of transmission and reception of data is instead **asynchronous** with respect to the cycle of execution of the Ladder code.
When an instruction of read/write of a device must be executed (line “n”), control passes immediately to the next instruction (line “n+1”), without waiting for the data to be effectively read/written.

The effective transfer of the data in the serial line is executed in a manner that is independent to the normal scan of Ladder code, in different times according to the port that is used.

6.1.1 Ports COM1 and EXP1

The ports COM1 and EXP1 can be configured with protocol ModBus (master or slave), or Nais Matsushita master, Control Technique. These are the ports typically used for communication with other devices (PLC, etc.).
The control of the communication is carried out every 1 mS. This means that the corresponding flow of serial data will be controlled 1000 times a second.

6.1.2 Port COM2

The port COM2 can be configured only by using protocol ModBus slave. This port is used for programming the terminal by PC.
The control of the communication is carried out every scan cycle of the Ladder code.
This means that the flow of the data in the serial port COM2 will be controlled one time at the end of each scan cycle.

6.2 Protocol ModBus RTU

The ModBus on the serial line is a Master-Slave protocol. In a network with this type, there is a single node (the Master) that interrogates and commands the Slaves and processes the results. The Slave nodes typically do not transmit data unless specifically requested by the Master and do not communicate directly between each other.
A device in the serial line (a network node) is uniquely determined by an identification number (ID, variable from 1 to 255), called the ModBus Slave address: two devices cannot have the same address.

The addressees of a request (one or more Slave nodes) are selected by the Master by their ID, thus the data that transits on the line has a precise destination.
The Master controls the line: it doesn't have a specific ID address and can read or write data in words or bits with one or more Slave devices, specifying the destination ID.
Data read or written is saved in the destination device in registers identified by a specific ModBus address (variable from 1 to 65535). Each ModBus address can correspond to a register (word area of memory) or a single bit of a register (particular bit of an area of memory).

Refer to the following figure for the list of possible operations in a ModBus communication: reading and writing of a word or bit, single or multiple.

Main features of protocol Modbus RTU	
Baud-rate	Programmable
Format	8,N,1 (8 bit, no parity, 1 stop) (default)
Supported function	BITS READING (0x01, 0x02) WORDS READING (max 30 word) (0x03, 0x04) SINGLE BIT WRITING (0x05) SINGLE WORD WRITING (0x06) MULTIPLE BITS WRITING (0x0F) MULTIPLE WORDS WRITING (max 30 word) (0x10)
Error codes	ILLEGAL FUNCTION CODE (0x01) ILLEGAL DATA ADDRESS (0x02) ILLEGAL DATA VALUE (0x04)
Broadcast	Simultaneus writing to all connected slaves using address 0x00 and no answer from slaves.
Polling with unknown slave address	Polling using address 0xFF, any connected slave can answer

6.2.1 ModBus RTU Master

The protocol ModBus Master can be configured only for the ports COM1 and EXP1.

With this configuration the TD320 will have control of the transit of the data of the corresponding port. For each of the two ports, there can be active up to 256 frames (active packets) at the same time. Each frame corresponds to an instruction of direct communication:

- **Reading from a Slave:** Reading from the slave at the ModBus address corresponding to the data of interest is memorized in the registers of the Master. Each instruction can read up to 16 consecutive words.
- **Writing on a Slave:** Data of interest by the Master is written in the slave at the ModBus address corresponding to the data to overwrite. Each instruction can write up to 16 consecutive words.
- **Read/write on a Slave:** Normally data read from the slave is saved in the Master. When the data internal to the TD320 varies by effect of the program, it is useful to write the modified data into the Slave. Each instruction of read/write can operate only on 1 word.

6.2.2 ModBus RTU Slave

The protocol ModBus Slave can be configured for all three of the ports COM1, EXP1, and COM2.
With this configuration all of the resources of the terminal are available to the Master device that is eventually connected.

The following table indicates all of the data (word and bit) accessible by use of the ModBus protocol. Each area of memory corresponds to a distinct ModBus address (for the access of a word or a bit), variable from 0 to 65536.

The read/write access and the value given at startup of the TD320 are shown for each. Depending upon the initialization values, the following cases occur:

1. “**ROM**” fixed values defined by the program.

2. “**EEP**”, value stored in EEProm memory, maintained for at least 10 years even in the absence of power..

3. “**BUFF**”, value stored in RAM with the battery buffer. Also this data is maintained in the absence of power, but for a limited time (around 4 to 6 months).

4. “**DEFINED VALUE**” the value given to the data at startup corresponds to the value indicated in the table.

ACCESS TO WORD			
MODBUS ADDRESS	DESCRIPTION	READ/ WRITE	RESET VALUE
0	**Type of device**	R	ROM
1	Version of Firmware	R	ROM
2	Protocol activated on COM1	R	ROM
3	Protocol activated on EXP1	R	ROM
4	Protocol activated on COM2	R	ROM
5	**Address of protocol**	R	BUFF
6	**Version of BOOT**	R	ROM
10	**Clock seconds TD320**	R/W	BUFF
1000 ÷ 1199	Word area special marker SM	R/W	BUFF
2000 ÷ 2999	Word area variable V	R/W	BUFF
12000 ÷ 12127	Word area timer T	R/W	0
13000 ÷ 13127	Word area preset timer PT	R/W	0
14000 ÷ 14063	Word area counter C	R/W	0
15000 ÷ 15063	Word area preset counters PV	R/W	0
16000 ÷ 16099	**Word area buffer TX COM1**	R	0
16500 ÷ 16599	**Word area buffer RX COM1**	R	0
17000 ÷ 17099	**Word area buffer TX EXP1**	R	0
17500 ÷ 17599	**Word area buffer RX EXP1**	R	0
18000 ÷ 18099	**Word area buffer TX COM2**	R	0
18500 ÷ 18599	**Word area buffer RX COM2**	R	0

19000 ÷ 19031	**Word area analog input AI**	R	0
19200 ÷ 19215	**Word area trimmer TR**	R	0
19400 ÷ 19431	**Word area analog output AQ**	R	0
19800 ÷ 19927	**Word percentage proportional / integral / derived / output PID**		
19800	% Action proportional PID1	R	0
19801	% Action integral PID1	R	BUFF
19802	% Action derived PID1	R	BUFF
19803	% Output PID1	R	BUFF
19804	% Action proportional PID2	R	0
……	……		
19927	% Output PID128	R	BUFF
20000 ÷ 20999	**Word area EEProm**	R/W	EEP
30000 ÷ 59999	**Word area MMC**	R/W	EEP

ACCESS TO WORD			
MODBUS ADDRESS	DESCRIPTION (n.o. = "normally open")	READ/ WRITE	RESET VALUE
90	**Contacts n.o. positioners POS1÷POS16**	R	0
95	**Contacts n.o. tuning positioners POS1÷POS16**	R	0
100	Contacts n.o. digital inputs I1 ÷ I16	R	0
101	Contacts n.o. digital inputs I17 ÷ I32	R	0
……	……		
131	Contacts n.o. digital inputs I497 ÷ I512	R	0
150	Contacts n.o. digital outputs Q1 ÷ Q16	R	0
151	Contacts n.o. digital outputs Q17 ÷ Q32	R	0
……	……		
181	Contacts n.o. digital outputs Q497 ÷ Q512	R	0
200	Contacts n.o. bistable relays B1 ÷ B16	R/W	0
201	Contacts n.o. bistable relays B17 ÷ B32	R/W	0
……	……		

207	Contacts n.o. bistable relays B113 ÷ B128	R/W	0
250 251 …… 299	Contacts n.o. marker M1 ÷ M16 Contacts n.o. marker M17 ÷ M32 …… Contacts n.o. marker M785 ÷ M800	R R R	0 0 0
300 301 …… 307	Contacts n.o. timer T1 ÷ T16 Contacts n.o. timer T17 ÷ T32 …… Contacts n.o. timer T113 ÷ T128	R R R	0 0 0
350 351 352 353	Contacts n.o. counters C1 ÷ C16 Contacts n.o. counters C17 ÷ C32 Contacts n.o. counters C33 ÷ C48 Contacts n.o. counters C49 ÷ C64	R R R R	0 0 0 0

ACCESS TO BIT			
MODBUS ADDRESS	DESCRIPTION (n.o. = “normally open”)	READ/ WRITE	RESET VALUE
1440 1441 …… 1455	**Contact n.o. positioner POS1** Contact n.o. positioner POS2 **……** **Contact n.o. positioner POS15**	R R R	0 0 0
1520 1521 …… 1535	**Contact n.o. tuning position POS1** Contact n.o. tuning position POS2 **……** **Contact n.o. tuning position POS15**	R R R	0 0 0
1600 1601 …… 2111	**Contact n.o. digital input I1** Contact n.o. digital input I2 **……** **Contact n.o. digital input I512**	R/W R/W R/W	0 0 0
2400 2401 …… 2911	**Contact n.o. digital output Q1** Contact n.o. digital output Q2 **……** **Contact n.o. digital output Q512**	R/W R/W R/W	0 0 0
3200	**Contact n.o. bistable relay B1**	R/W	0

3201 …… 3327	Contact n.o. bistable relay B2 **……** **Contact n.o. bistable relay B128**	R/W R/W	0 0
4000 4001 …… 4799	**Contact n.o. marker M1** Contact n.o. marker M2 **……** **Contact n.o. marker M800**	R/W R/W R/W	0 0 0
4800 4801 …… 4927	**Contact n.o. timer T1** Contact n.o. timer T2 **……** **Contact n.o. timer T128**	R R R	0 0 0
5600 5601 …… 5663	**Contact n.o. counter C1** Contact n.o. counter C2 **……** **Contact n.o. counter C64**	R R R	0 0 0
16000 16001 …… 19199	**Bit 0 area special marker SM0** Bit 1 area special marker SM0 **……** **Bit 15 area special marker SM199**	R/W R/W R/W	BUFF BUFF BUFF
32000 32001 …… 63999	**Bit 0 area variables V0** Bit 1 area variables V0 **……** **Bit 15 area variables V2000**	R/W R/W R/W	BUFF BUFF BUFF

6.3 Protocol NAIS Matsushita Master

This is the protocol that permits the reading and writing of data (bit of word) of the PLC Nais Matsushita.
Generally, the communications interface is RS232, the velocity is 9600 baud (bits/sec), the format of communications 8,O,1 (8 bits of data, odd parity, 1 stop bit).
The following table indicates all of the elements that can be read/written from the PLC. The address of the bit o of the word to read or write is obtained by adding the real address of the bit/word (between Min and Max) to the value indicated in the column Offset. Each instruction “COM” or “EXP” can read or write to several consecutive data locations, the maximum number for each

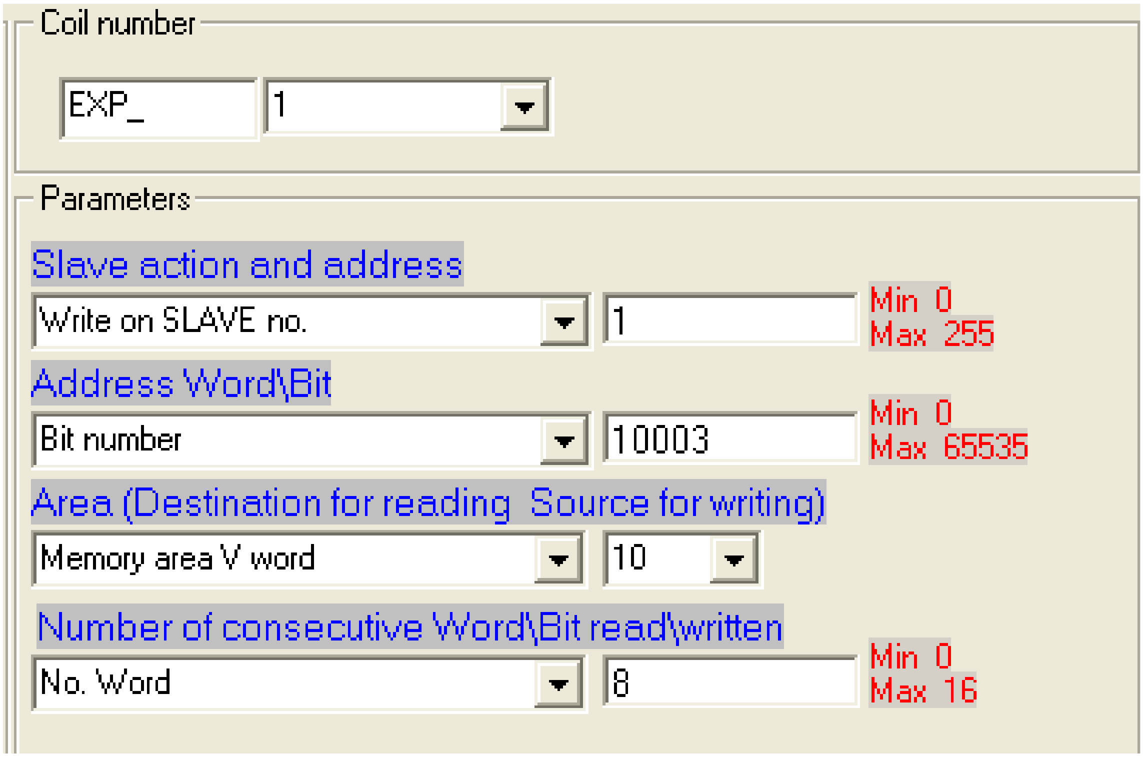
type of data is indicated in the column “Max number bit/word read/written consecutively”.

ACCESS TO BIT						
CONTACT	SYM	MIN	MAX	OFFSET	READ/ WRITE	MAX NUMBER OF BITS READ / WRITTEN CONSECUTIVELY
EXTERNAL INPUT	X	0	9999	0	R	8
EXTERNAL OUTPUT	Y	0	9999	10000	R/W	8
INTERNAL RELAY	R	0	9999	20000	R/W	8
LINK RELAY	L	0	9999	30000	R/W	8
TIMER	T	0	9999	40000	R	8
COUNTER	C	0	9999	50000	R	8

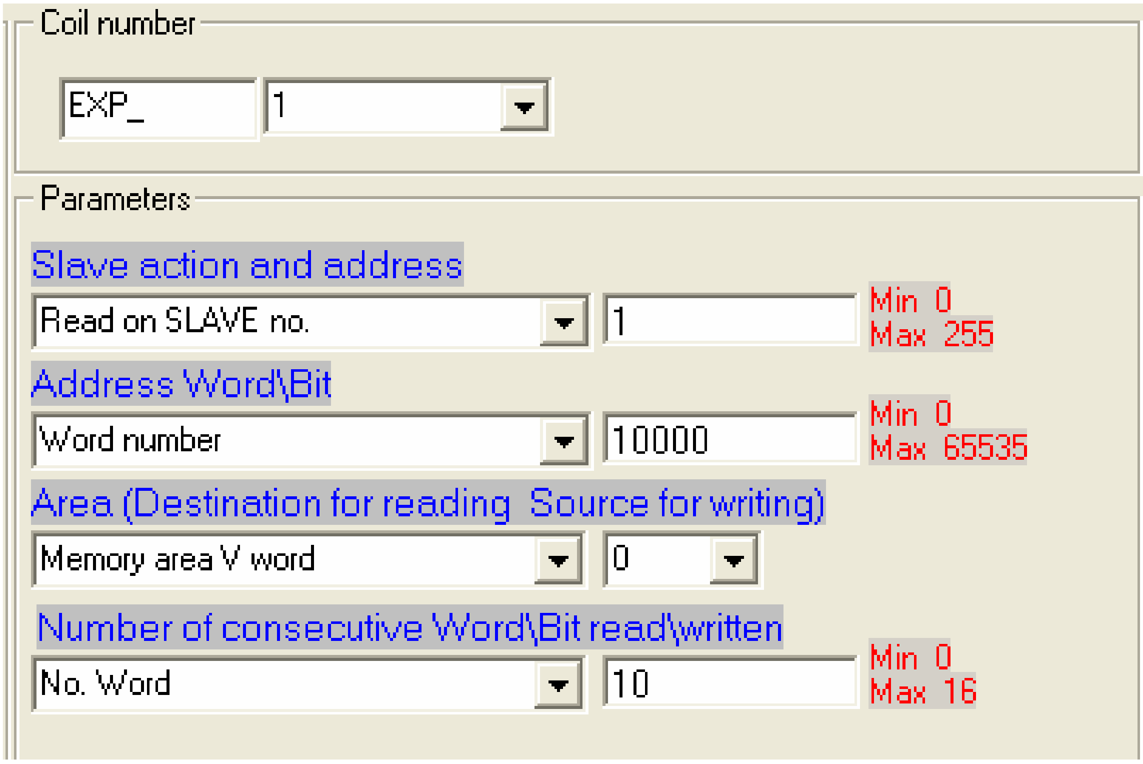
ACCESS TO WORD						
WORD NAME	SYM	MIN	MAX	OFFSET	READ/ WRITE	MAX NUMBER OF WORDS READ / WRITTEN CONSECUTIVELY
EXTERNAL INPUT	X	0	999	0	R	10
EXTERNAL OUTPUT	Y	0	999	1000	R/W	10(R) / 7 (W)
INTERNAL RELAY	R	0	999	2000	R/W	10(R) / 7 (W)
LINK RELAY	L	0	999	3000	R/W	10(R) / 7 (W)
TIMER	T	0	999	4000	R	10
COUNTER	C	0	999	5000	R	10
INDEX REG. X		0	0	6000	R/W	1
INDEX REG. Y		0	0	6001	R/W	1

INDEX REG. D		0	0	6002	R/W	1
DATA REGISTER	DT	0	9999	10000	R/W	10(R) / 7 (W)
LINK DATA REGISTER	LD	0	9999	20000	R/W	10(R) / 7 (W)
FILE REGISTER	FL	0	9999	30000	R/W	10(R) / 7 (W)
SET VALUE AREA		0	9999	40000	R/W	10(R) / 7 (W)
ELAPSED VALUE AREA		0	9999	50000	R/W	10(R) / 7 (W)

For the two examples shown below, the protocol NAIS Matsushita is selected for the port EXP1.
The illustrated instructions that follow write the contents of the 8 words from V10 to V17 of the TD320 in the register EXTERNAL OUTPUT of the PLC NAIS from Y3 to YA (Y10).



The following illustration, however, reads the register DATA REGISTER of the PLC NAIS, the 10 words from DT0 to DT9, and copies them in the area of V0 to V9 of the TD320.



7 Ladder programming of the TD320

Programming the PLC part of the TD320 is accomplished with the development environment **PLProg 4.xx**, which provides the user with all the resources necessary for creation of the Ladder diagram.
The compilation and download procedure discussed in chapter 4 allows the TD320 terminal to achieve the desired functionality.
The following describes all available elements (contacts and coils) and the relative characteristics for the creation of the diagram.

7.1 Digital input contacts I

Digital input contacts I can contain the state of the inputs read via serial lines of other devices, up to a maximum of 512.
A contact normally open (N.O.) is closed (ON) when the bit value is “1” (input active). A contact normally closed (N.C.) is opened (ON) when the bit value is “0” (input non-active).

7.2 Digital output contacts Q

The TD320 has 512 type “Q” outputs. These can be used to contain the state of eventual outputs of other devices, communicated by serial lines.
Each output has a coil and a related logical contact N.O. (normally open) or N.C. (normally closed). At activation of the coil “Q”, the related logical contact will close (if normally open) or will open (if normally closed).

7.3 Bistable relay B

There are 128 bistable relays available in the TD320. Each has a coil and related logical contact normally open or closed (N.O/N.C).
At activation of coil “B”, the related logical contact will change state, if it was closed it will open, if it was open it will close. A contact N.O. is closed (ON) when the bit value is “1”. A contact

N.C. is opened (ON) when the bit value is “0”. At the startup of the terminal, a contact N.O. will be open.

7.4 Timer T

The TD320 has 128 timers of 16 bits. Each is available in three modes of functioning:

- **TON “on-delay” of activation:** time begins counting when the coil is activated (ON). The timer bit (contact T) will be activated when the current timer value (word T) becomes greater than or equal to the pre-established time (preset, word PT). When the coil is deactivated (OFF), the current value of the timer is reset (zeroed). The timer stops in any case when it reaches the maximum value in signed 16-bits (+32767).
- **TOFF “off-delay” of deactivation**: allows delaying the deactivation of an output for a given period of time after the input has been deactivated. When the coil is activated (ON), the time bit (contact T) is immediately activated and the current value of the timer (word T) will be set to “0”. At the deactivation of the coil, the timer will count until the elapsed time becomes greater than or equal to the pre-established time (preset, word PT). Once reached, the timer bit deactivates and the current value stops advancing. If the input remains inactive for a time that is less than the pre-established time, the timer bit remains active. To start the count, the TOFF operation should sense a transition from state active to non-active (ON → OFF)
- **TONR with memory:** time begins counting when the coil is active (ON). The timer bit (contact T) is active when the current timer value (word T) becomes greater than or equal to the pre-established time (preset, word PT). When the coil is deactivated (OFF), the current value of the timer is maintained. Thus it is possible to accumulate time for more periods of activation of the coil. The current value of the timer can be reset with the operation MOV(Tx = #0). The timer stops in any case when it reaches the maximum value in signed 16-bits (+32767).

The time base can be selected between 10 mS, 100 mS, and 1S for each mode of functioning.
The current value of the timer is a multiple of the selected time base. For example, a current value of 50 in a timer with a base time of 10 mS corresponds to 500 mS, and with a base time of 1 S corresponds to 50 S.
The preset timer (PT) value can be a constant, or the contents of an area VW, SMW, AI, or TR.

7.5 Counters C

The TD320 has 64 counters of 16 bits. These are available in two modes of functioning:

- **MUP forward counter**: the counter bit (contact C) is activated when the current value (word C) is greater than or equal to the pre-established value (PV). The counts increments each time the input of the up-count Cx(UP) is active and decrements each time the input of the down-count Cx(DOWN) is active. The counter will be set to zero upon activation of the reset input Cx(RESET) or when the operation MOV(Cx=#0) is executed. Upon reaching the maximum value (32767), the rise of the next up-count will leave the current value unchanged. Similarly, upon reaching the minimum value (-32768) the rise of the next down-count will leave the current value unchanged. For the forward counters, the pre-established value (PV) is compared with the current value at the end of each cycle of the program. If the value is greater than or equal to the preset value, the counter bit activates (counter C), otherwise it is deactivated.
- **MDOWN backward counter**: the counter bit (contact C) is activated when the current value (word C) becomes equal to zero. The counter decrements from a pre-established value (PV) on the rise of the input of down-count Cx(DOWN) and increments on the rise of the input of up-count Cx(UP). Upon reaching the maximum value (32767), the rise of the next up-count will leave the current value unchanged. The counter resets the count bit (contact C) and loads the preset value

(PV) when the input Cx(RESET) becomes active. The counter in backward mode will stop counting when it reaches zero.

The preset value (PV) can be a constant, or the contents of an area VW, SMW, AI, or TR.

7.6 Mathematic formulas (FM)

The functions of math formulas FM execute mathematical operations (+, -, *, /, | [OR: logical inclusive or], & [logical AND], ^ [XOR: logical exclusive or], << [ROL: ROtate shift Left], >> [ROR: ROtate shift Right) between two operators and saves the result in another memory location. The operators can be numeric (constants) or refer to the available areas of memory (variables).

7.7 MOV assignments

The function MOV (move) assigns a numeric value (constant) or the contents of another location (source area) to a specified location in memory (destination area).
An instruction such as MOV(A=B) copies the contents of the memory location B to the memory location A.

7.8 BLKMOV multiple assignments

The function BLKMOV (block move) assigns a numerical value or the value from another (source) block of memory to a destination block of memory.
An instruction such as BLKMOV(A_i=B_i, num. data 8) copies the contents of memory B_i into the location of memory A_i, the contents of location B_{i+1} into the location A_{i+1}, ……and the contents of B_{i+7}into the location A_{i+7}.

7.9 MOVIND indexed assignments

MOVIND (move with index offset) assigns a numerical value (constant) or the value from another location of memory (variable source) to the specified location of memory (destination) as offset by an index for the source and/or destination.
This type of assignment permits various memory areas to be used as vectors of N locations each, where the value taken from another location is used as an “index”. It is possible to access the values n=0, n=1, …, n=N-1 of the area.
An instruction such as MOVIND(A[B]=C[D]) copies the contents of the memory location C[D] into the location A[B]. The index of area C is specified by D, which can be another memory location, and similarly B is the index of area A.

7.10 MOVTXT assignments

MOVTXT saves string characters passed as a function parameter to a specified location in memory. This function permits the following types of characters of the string in the memory area:

- ONE_CHARACTER_PER_WORD: in this format, each word of the destination area will contain a single character of the source string.
- TWO_CHARACTERS_PER_WORD in this format, each word in the destination area will contain two characters of the source string, starting with the high part. If string = “Example” then V[0] = Ex, V[1] = am, V[2] = pl. V[3] = e.

7.11 Digital input immediate contacts II

The digital input contacts II allow the immediate reading of the digital input state. The contact normally open is closed (ON) when the bit value is “1” (input active). The contact normally closed is open (ON) when the bit value is “0” (input non-active).

7.12 Contacts IF

The operations of conditional IF compare the values of two variables of any area of memory. It is possible to carry out the following types of comparison: = (equal), >= (greater than or equal), <= (less than or equal), > (more than), < (less than), <> (not equal). The contact is active when the comparison is true.

7.13 Functions SBIT and RBIT

The function SBIT (set bit) puts a “1” in a bit of a memory area when the coil of the function is at the active state.
The function RBIT (reset bit) puts a “0” in a bit of a memory area when the coil of the function is at the active state.
The index of the bit varies from 0 to 15 (the destination area is always a word), where bit 0 is the least-significant bit (LSB).

7.14 BIT contacts

This operation extracts the value of a bit of an area of memory.
A contact normally open is closed (ON) when the bit value is “1”.
A contact normally closed is open (ON) when the bit value is “0”.
The index of the bit varies from 0 to 15 (the destination area is always a word), where bit 0 is the least-significant bit (LSB).

7.15 RANGE functions

The function RANGE defines the value of the minimum and maxim limits for the analog inputs AI, for the trimmer TR, for the analog outputs AQ, and for the outputs of the PID.

RANGE(AI1, Min 10, Max 200)

The function imposes a minimum limit of 10 and maximum limit of 200 for the analog input AI1. If the analog input AI1 corresponds to a potentiometer (from a PLC via a serial communication), is used to establish the preset (PT) of a timer of base time 100 mS, this

provides a variable time from 1.0 to 20.0 seconds, according to the value of the potentiometer.
If input values exceed the limits set in the RANGE function, the output will be blocked to the minimum or maximum allowed value.
As for the output PID, the minimum and maximum values serve to calculate the value of the output generated by the algorithm of regulation. Let us consider the following example:

RANGE(PID1, Min 100, Max 500)

The function imposes the minimum limit of 100 and the maximum limit of 500 for the PID1 output. This means that an output of 0% corresponds to the minimum value imposed (100) and 100% will correspond to an output equal to the maximum value (500).

7.16 NOT contacts

The contact NOT modifies the state of the flow of current. The flow of current stops if it reaches a NOT contact and supplies energy if it doesn’t reach it.
The operation NOT inverts the logical value (0 → 1 and 1 → 0).

7.17 P and N contacts

The transition positive P contact activates the flow of current for one scan cycle of each transition from OFF to ON. The transition negative N contact activates the flow of current for one scan cycle of each transition from ON to OFF.
The instructions that follow in the diagram are thus executed only once (per scan cycle) for each transition that activates the contact.

7.18 SEND functions

The function SEND transmits the data through the serial line in free-port mode.
In this mode, enabled by the special markers SM39, SM40, and SM41, the protocol that normally manages the serial port is disabled and the Ladder program takes control of the port and of the transmission and reception buffers.

After having loaded the buffer with the data to transmit, activating the SEND function, which has parameters for the serial port and the number of characters to transmit, will cause the data to be sent on the serial line.
During the transmission phase, the bits SM0.6, SM0.7 or SM0.8 relative to the transmission port are set to “1”, while at the end of the transmission they will be set to “0”. It is possible to control an eventual reply of a connected device through the control of SM42, SM43, and SM44, which contain the number of characters received and saved in the reception buffer of each serial port. Any writing on any of these special markers causes the emptying of the buffer data in reception of the corresponding port.
Calls to the SEND function before the end of the preceding transmission or with free-port mode disabled are ignored by the program.

7.19 TunePOS and POS functions

The function “TunePOS” executes an auto-tuning procedure, indispensable for extracting the data of reaction time and axis inertia for which a positioning procedure is requested.
The function “POS” executes the positioning ON/OFF of the axis.
The functions operate on the variable area VD (double word), the address of the beginning of the area is requested as a parameter of the functions “TunePOS” and “POS”. The following table indicates how the data are organized in the area of the two functions from the address of the specified location:

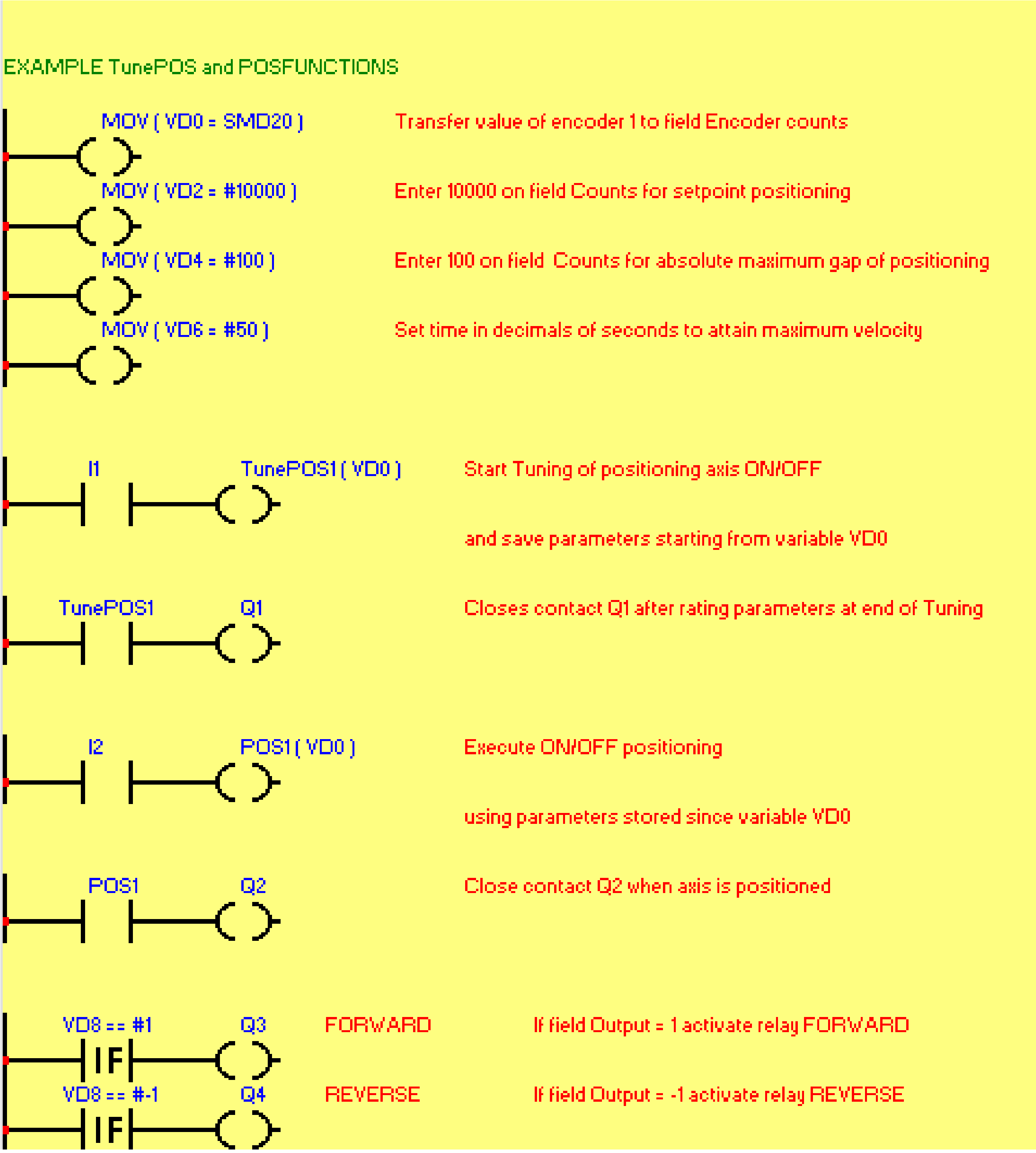
Address area VD	Contents
+0	Count for encoder
+2	Counts for setpoint positioning
+4	Counts for absolute maximum gap of positioning
+6	Time needed to attain max velocity (in decimals of seconds)
+8	State of the positioning output (0=stationary, 1=forward, 2=backward)

+10	Counts for forward inertia
+12	Counts for backward inertia
+14	Duration of minimum impulse (resolution 0.2 mS)
+16	Counts of movement after impulse of 100 mS
+18	Counts of movement after impulse of 500 mS
+20	Counts of movement after impulse of 1000 mS

For correct functioning, it is necessary to proceed as follows:

- Transfer the count of the encoder connected to a remote device (read via a serial line) in the field “Counts for encoder” (beginning area of memory).
- Set the count values to the desired position of the axis in the field “Counts for setpoint positioning”.
- Set the count values for the maximum gap of positioning in the field “Counts for absolute maximum gap of positioning”.
- Set the time, in decimals of seconds”, needed for the axis to attain maximum velocity.
- Activate the function “TunePOS” and wait that the contact TunePOS (normally open) closes to indicate the end of the procedure of auto-tuning the axis. At this point, the inertia data and the reaction time of the axis are automatically memorized in the indicated area of memory, remaining available for the function “POS”.
- Deactivate the function “TunePOS”.
- Activate the function “POS”. When the axis is positioned to the setting imposed (within the pre-established gap), the contact POS (normally open) will close, to indicate the end of positioning.
- Activate the outputs FORWARD and BACKWARD, reading the value of the field “Output” (VD+8). If the value of “Output” is “1”, it is necessary to activate the output FORWARD, if it is “-1”, it is necessary to activate the output BACKWARD, if it is “0” it is not necessary to activate any output.
- Set the value of the field “Output” to zero when the consent of the function “TunePOS” or “POS” is removed, to avoid that the output remains forced to forward or backward.

The following example shows the segment of Ladder code that implements the axis positions as explained in the procedure:



7.20 COM and EXP functions

The communication functions COM and EXP allow programming of the two serial ports (COM1 and EXP1) for the reading/writing of the data of the connected Slave devices, using the Master protocol selected in the project.
Such functions are active only when a protocol of communication of type Master is selected for the serial port within the project, that is, a protocol that allows the TD320 to take control of the line governing the flow of data with the slave devices.
The two functions are analogous, the only change is the serial port that is referenced. Taking into consideration that an Interface RS485 allows the connection of several devices to the same line, while the Interface RS232 allows connection of a single device to the TD320.
The instructions are active until the corresponding coil is activated, but keep in mind that, according to the protocol of communication, the time of updating the data can vary significantly and that at the moment of activation of the coil, the data read are not available instantly, but only after a certain time due to the delay of communication.
The instructions COM and EXP use the following parameters:

- Index (it is possible to set a maximum of 256 different serial interrogations on each port).
- Type of operation performed:
    - Reading: the TD320 continuously reads the data of the Slave device(s) and memorizes them in an area of internal memory.
    - Writing: the TD320 continuously writes the data to an area of internal memory in the Slave device(s).
    - Reading/Writing: the TD320 normally reads data of the Slave device and memorizes them in an area of internal memory; at the moment in which such internal data to the TD320 is modified by the program, the variations will be passed automatically to the Slave device through a write instruction (one datum at a time).

- Number of the Slave (address of communication of the Slave device).
- The type of data (word or bit).
- The address ModBus relative to the datum (or data) to transfer.
- The area of internal memory of the PL250A for reading or writing the data.
- The number of words (the instructions of reading and writing can transfer up to consecutive 16 bits/words).

7.21 StartPID, PID and SetOutPID functions

The functions StartPID, PID and SetOutPID allow the regulation of a size through an algorithm of action that is proportional, integral, and derived.
The function **StartPID** activates the regulation. The function can be activated a single time at startup or repeated at a later moment permitting the modification "on the fly" of the parameters of regulation. The integral action of the PID is zeroed only by calling the functions and fixing the integral time to "0". Otherwise, even in case of shutdown, the system will initialize the regulation maintaining as point of departure the same percentage of integral action, thus limiting the time of transition.
Parameters of the function **StartPID**:

- Proportional band
- Integral time
- Derived time
- Dead band

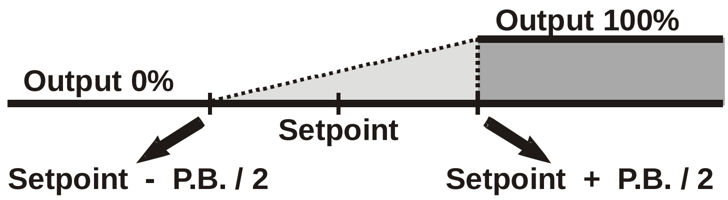
The parameters can be inserted in numerical format, or can refer to areas of memory. The integral time is expressed in the units of time in which the function PID is called (for instance, function PID called every 1 second, integral time expressed in seconds). The derived time, however, is expressed with an additional decimal digit with respect to the integral time. The proportional band and the dead band are instead expressed in numeric values equal to the setpoint and the process to regulate.

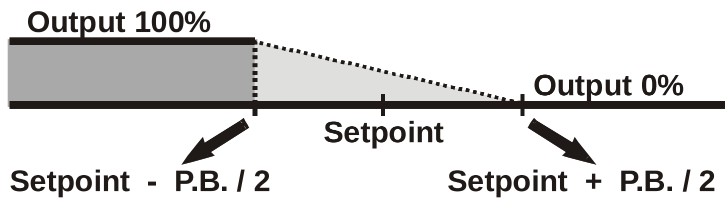
The parameters of the function **PID**:

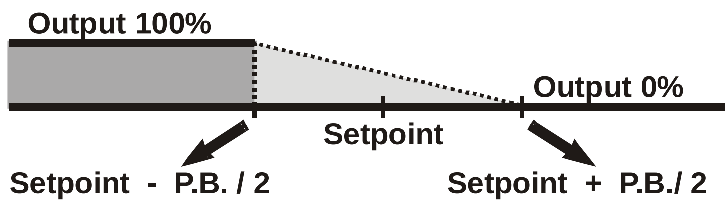
- Setpoint
- Process
- Output value
- Type of regulation action

The PID function, after acquiring setpoint, process, type of action and type of output, will set in the variable “Ouput value” the value obtained in the algorithm of regulation. Such a value will be obtained rescaling the percentage of the value between 0 and 10000 (0.00% ÷ 100.00%) between the minimum and maximum value of the PID output set by the RANGE function.

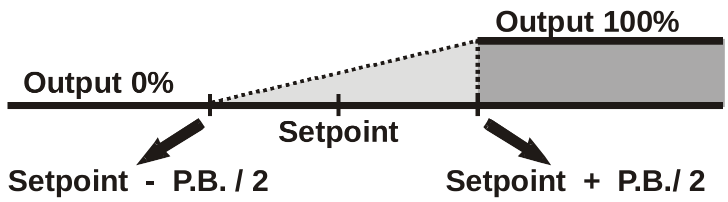
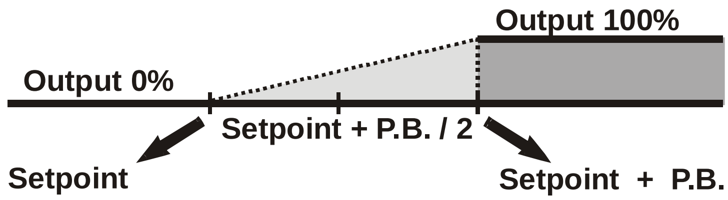
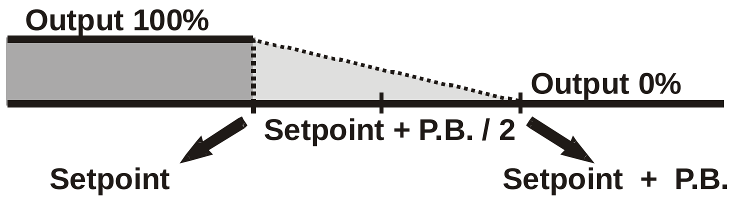
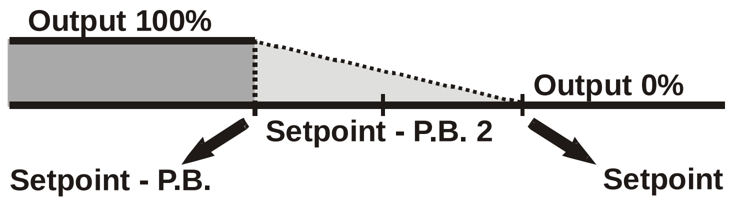
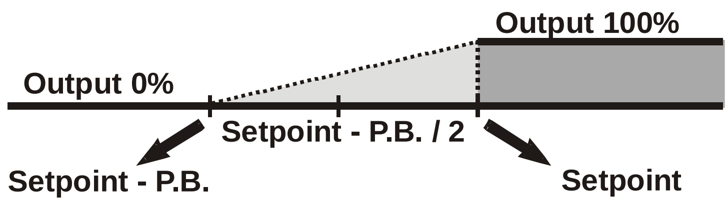
The following table indicates 8 types of regulation and the modulation intervals (the effective value between the interval is determined also by the actions integral and derivative, the table shows only the proportional components):

<table>
<tr><th>Type of regulation action</th><th>Intervals of modulation</th></tr>
<tr><td>Single direct action, 0</td><td>


</td></tr>
<tr><td>Single direct action, 1</td><td>


</td></tr>
<tr><td>Single inverse action, 0</td><td>


</td></tr>
</table>

Single inverse action, 1	
Double direct action, 0	
Double direct action, 1	
Double inverse action, 0	
Double inverse action, 1	

The PID function, for correct operation, must be called at the most regular intervals possible, thus by timer, or for more brief and precise times, by an internal interrupt.

The function **SetOutPID** is used for the regulation anticipated by the double function automatic/manual. It serves to avoid oscillation of size control in switching from manual mode to automatic by the PID algorithm.

The function uses the following parameters:

- Output value

The Output value is set by the PID automatically calculating the single percentages of the proportional and integral actions. In this mode, at the switching of manual function to automatic, the output value of the PID will take on the value set by manual and will initiate the regulation.
The function thus should be called only during the manual regulation phase, in order to maintain alignment of the output of the PID with that of manual. The function will automatically zero the derived action. The use of this function with the process outside of the proportional band sets the integral action to zero.

7.22 GENSET functions

The function GENSET automatically generates a setpoint variable rising or falling, with the possibility to set a ramp of acceleration or deceleration. The function GENSET operates on a series of variables in contiguous double words, starting from the location indicated as a parameter to the function.
The following table indicates how the data are organized in the memory area used by the function starting from the address of the specified location:

Address area VD	Contents
+0	State of the GENSET function 0 → Stop or end of movement 1 → Initialization function 2 → Ramp of acceleration 3 → Movement at constant velocity 4 → Ramp of deceleration
+2	Initial setpoint / setpoint calculated by the function GENSET (counts)
+4	Final setpoint (counts)
+6	Velocity of movement (counts*1000 / time unit)

+8	Duration of acceleration ramp (time unit)
+10	Duration of deceleration ramp (time unit)
+12	Instantaneous velocity of setpoint (counts*1000 / time unit)

For correct functioning, it is necessary to proceed as follows:

- Set the starting setpoint in location VD+2.
- Set the final setpoint in location VD+4.
- Set the maximum velocity of movement in location VD+6 in counts*1000 / time unit (so as to have 3 decimal digits. For example, setting 12345 corresponds to a velocity of 12.345 counts / time unit).
- Set the duration of the acceleration ramp in location VD+8 (expressed in time units, if the duration of the phase of acceleration should be 1 second, and the GENSET function is called by an interrupt of 1 mS, set 1000).
- Set the duration of the ramp of deceleration in location VD+10.
- Write “1” in the location VD (the location indicated as parameter of the function). This gives the “start” to the function that will automatically begin to write the generated setpoint in the location VD+2. The location VD will be also updated with the actual state, while the instantaneous velocity of the setpoint expressed with three decimal digits will be written in the location VD+12.

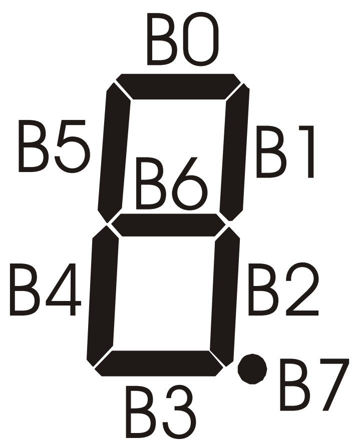
At the end of movement, when the location VD+2 attains the value of the final setpoint, the functional will automatically enter into a standby phase, indicated by the value “0” in the location VD. In this mode, the GENSET function can remain always enabled, even when movement is not necessary.

7.23 CONV functions

The function CONV converts the source data into one of the available formats:

- **TO_7SEG_SIGNED**: Converts the input data (a word with sign –32768..+32767) into a number specified in digits already transformed in code for 7-segment display. The function will take as parameters the number of digits to convert, starting from the least significant digit. The coded data will be saved (one digit per word) starting from the destination word and then in the successive words according to the number of digits requested.

- **TO_7SEG_UNSIGNED:** This is analogous to the above description with the difference that the data of origin is interpreted as a word without sign (0..65535). The code is comprised of a bit set to “1” if a segment should be lit, and “0” if the segment should remain dark. The association between the bits and the segments of the display is the following:



- **TO_ASCII_SIGNED:** Convert the input data (a word with sign –32768..+32767) into ASCII-coded digits. The function will take as parameters the number of digits to save. The coded data will be saved (one digit per word) starting from the destination word and then in the successive words according to the number of digits requested.

- **TO_ASCII_UNSIGNED:** This is analogous to the above description with the difference that the data of origin is interpreted as a word without sign (0..65535).

8 Notes / Updates

Notes / Updates

Notes / Updates