QTP 12

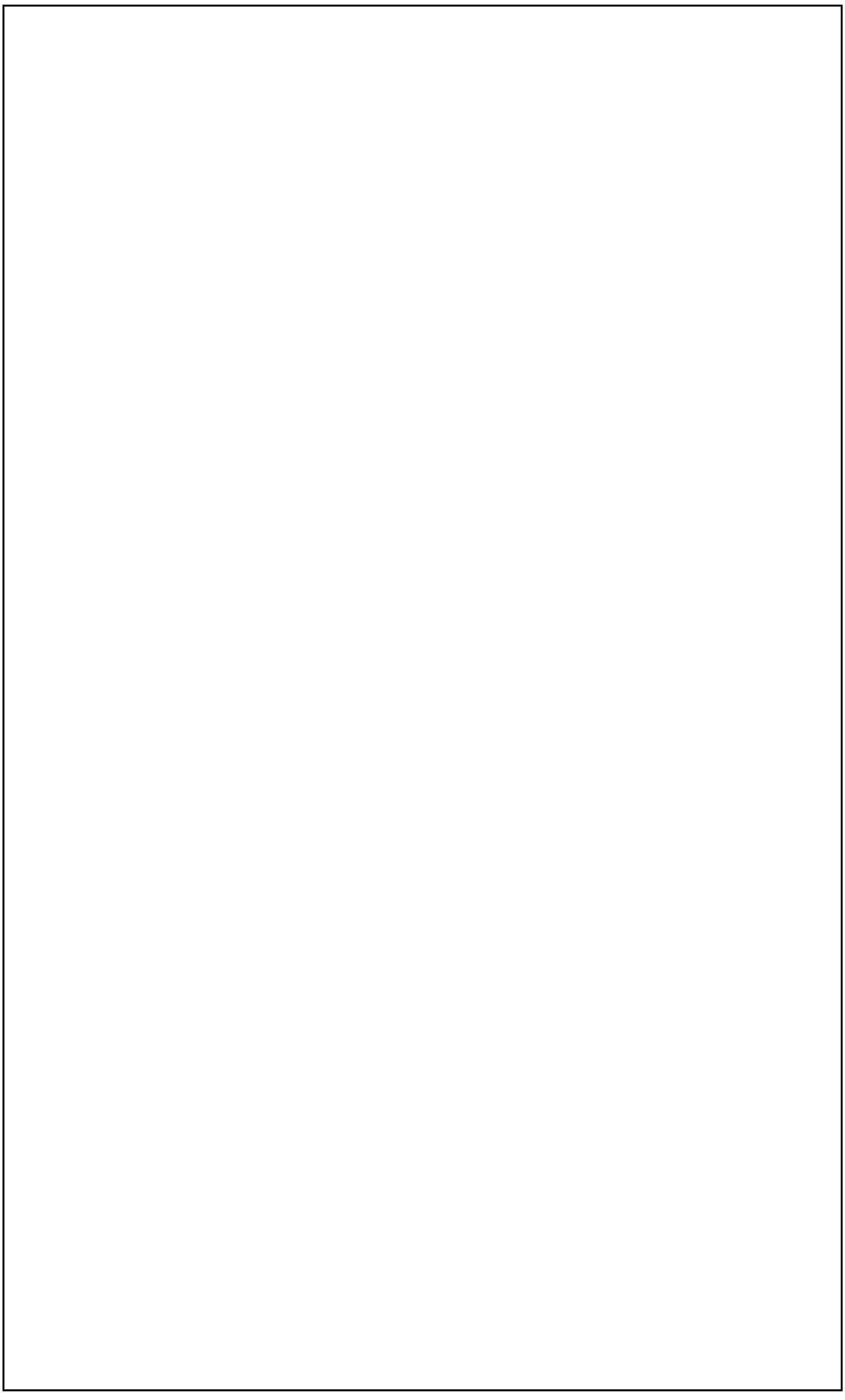
Quick Terminal Panel 12 keys

USER MANUAL

grifo®
ITALIAN TECHNOLOGY

Via dell' Artigiano, 8/6
40016 San Giorgio di Piano
(Bologna) ITALY
E-mail: grifo@grifo.it
http://www.grifo.it http://www.grifo.com
Tel. +39 051 892.052 (a.r.) FAX: +39 051 893.661

QTP 12 Rel. 5.50 *Edition 04 January 2007*



QTP 12

Quick Terminal Panel 12 keys

USER MANUAL

Operator interface provided with a very low price and interesting visualization capabilities. Dimension: **78x144 mm**; **9 mm** frontal; **28 mm** rear. Aluminium **Container** shape with front plastics frame, provided of mounting **Brackets**. Front panel with keyboard, protected by anti scratch polyester cover, with **IP 54** protection. Front panel mounting. Equipped with **3** displays models: **Alphanumeric LCD** or **Fluorescent** by **20x2** characters and **Graphic** Fluorescent by **140x16 pixels**; adjustable **Brightness** of VFD display and **Contrast** of LCD display. Membrane keypad with **12** keys provided of double serigraphy: numeric and function. Features of debounce, autorepeat, keycklick and disable for the keys pressed.
1 LED and **Buzzer** managed by software in different modes. Panel name, and/or **LED** indicator, **Personalization Label** slot. **1** digital outputs,completely driven by software. **EEPROMs** for setup, messages, user characters, keys codes, etc. Up to **3371** different **messages** can be saved and displayed, even with auto scrolling mode. **Real Time Clock** (RTC) backed by dedicated **Lithium** battery; complete **Alarm Clock** capable to drive the digital output.
CAN communication line provided of proper line driver. Asynchronous serial line configurable in **RS 232**, **RS 422**, **RS 485** or **Current Loop**. Synchronous line in **I2C BUS**. Network connection through proper protocols. Local **setup** for operating modalities selection. Up to **256 Different Characters** visible; **8** user characters provided of selectable **Pattern**.
Wide range **DC** or **AC** power supply from **5 Vdc** to **24 Vac**; total power consumption change according with used configuration, from **1.6** to **2.5 W**. On board electronic protection aganist voltage peaks, by **TransZorb**. Comfortable connectors for a fast cablation, with standard pin outs.
Possibility to require customized keyboard and program packages.

grifo®
ITALIAN TECHNOLOGY
Via dell' Artigiano, 8/6
40016 San Giorgio di Piano
(Bologna) ITALY
E-mail: grifo@grifo.it
http://www.grifo.it http://www.grifo.com
Tel. +39 051 892.052 (a.r.) FAX: +39 051 893.661

QTP 12 Rel. 5.50 *Edition 04 January 2007*

DOCUMENTATION COPYRIGHT BY grifo®, ALL RIGHTS RESERVED

IMPORTANT

Although all the information contained herein have been carefully verified, grifo® assumes no responsability for errors that might appear in this document, or for damage to things or persons resulting from technical errors, omission and improper use of this manual and of the related software and hardware.
grifo® reserves the right to change the contents and form of this document, as well as the features and specification of its products at any time, without prior notice, to obtain always the best product.
For specific informations on the components mounted on the card, please refer to the Data Book of the builder or second sources.

SYMBOLS DESCRIPTION

In the manual could appear the following symbols:


Attention: Generic danger


Attention: High voltage


Attention: ESD sensitive device

Trade Marks

abaco, **GPC®**, grifo® : are trade marks of grifo®.
Other Product and Company names listed, are trade marks of their respective companies.

GENERAL INDEX

FIGURES INDEX

INTRODUCTION

The use of these devices has turned - **IN EXCLUSIVE WAY** - to specialized personnel.

This device is not a **safe component** as defined in directive **98-37/CE**.

Pins of module are not provided with any kind of ESD protection. Many pins of the card are directly connected to their respective pins of on board's components and these last are sensitive to electrostatic noises. So personnel who handles the product/s is invited to take all necessary precautions that avoid possible damages caused by electrostatic discharges.

The purpose of this handbook is to give the necessary information to the cognizant and sure use of the products. They are the result of a continual and systematic elaboration of data and technical tests saved and validated from the manufacturer, related to the inside modes of certainty and quality of the information.

The reported data are destined- IN EXCLUSIVE WAY- to specialized users, that can interact with the devices in safety conditions for the persons, for the machine and for the enviroment, impersonating an elementary diagnostic of breakdowns and of malfunction conditions by performing simple functional verify operations , in the height respect of the actual safety and health norms.

The informations for the installation, the assemblage, the dismantlement, the handling, the adjustment, the reparation and the contingent accessories, devices, installation, etc. are destined - and then executable - always and in exclusive way from specialized warned and educated personnel, or directly from the AUTHORIZED TECHNICAL ASSISTANCE, in the height respect of the manufacturer recommendations and the actual safety and health norms.

The devices can't be used outside a box. The user must always insert the cards in a container that rispect the actual safety normative. The protection of this container is not threshold to the only atmospheric agents, but specially to mechanic, electric, magnetic, etc. ones.

To be on good terms with the products, is necessary guarantee legibility and conservation of the manual, also for future references. In case of deterioration or more easily for technical updates, consult the AUTHORIZED TECHNICAL ASSISTANCE directly.

To prevent problems during card utilization, it is a good practice to read carefully all the informations of this manual. After this reading, the user can use the general index and the alphabetical index, respectly at the begining and at the end of the manual, to find information in a faster and more easy way.

grifo® provid this documentation "as is" without warranty of any kind. In no event shall **grifo®** be liable for indirect, special, incidental or consequential damages of any kind arising from any error in this documentation, including any loss or interruption of business, profits, use , or data. Moreover is not guaranteed the updating of the product for new computers or new operating systems, that will become available in the future.

VERSION

This handbook make reference to printed circuit version **160805** and to firmware version **2.1** and following ones. The validity of the information contained in this manual is subordinated to the version numbers on the used panel, and so the user must always verify the correct correspondence beetween the notations. The version numbers are reported in several places on the electronic part of the product, and following figure shows the most accessible ones. Obviously if the version must be checked, then it must be extracted from the metallic container: a simple pressure on **QTP 12** connectors, or on the printed circuit reachable from rear container window, is sufficient. When on the front panel there are two black screws, they must be previously unscrewed (for details see APPENDIX C).

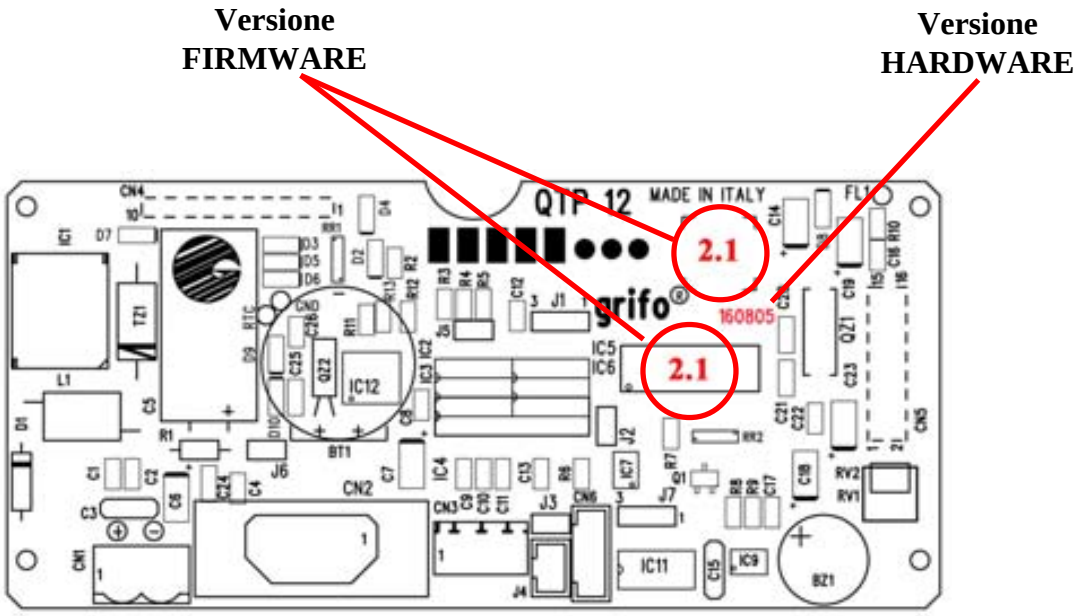


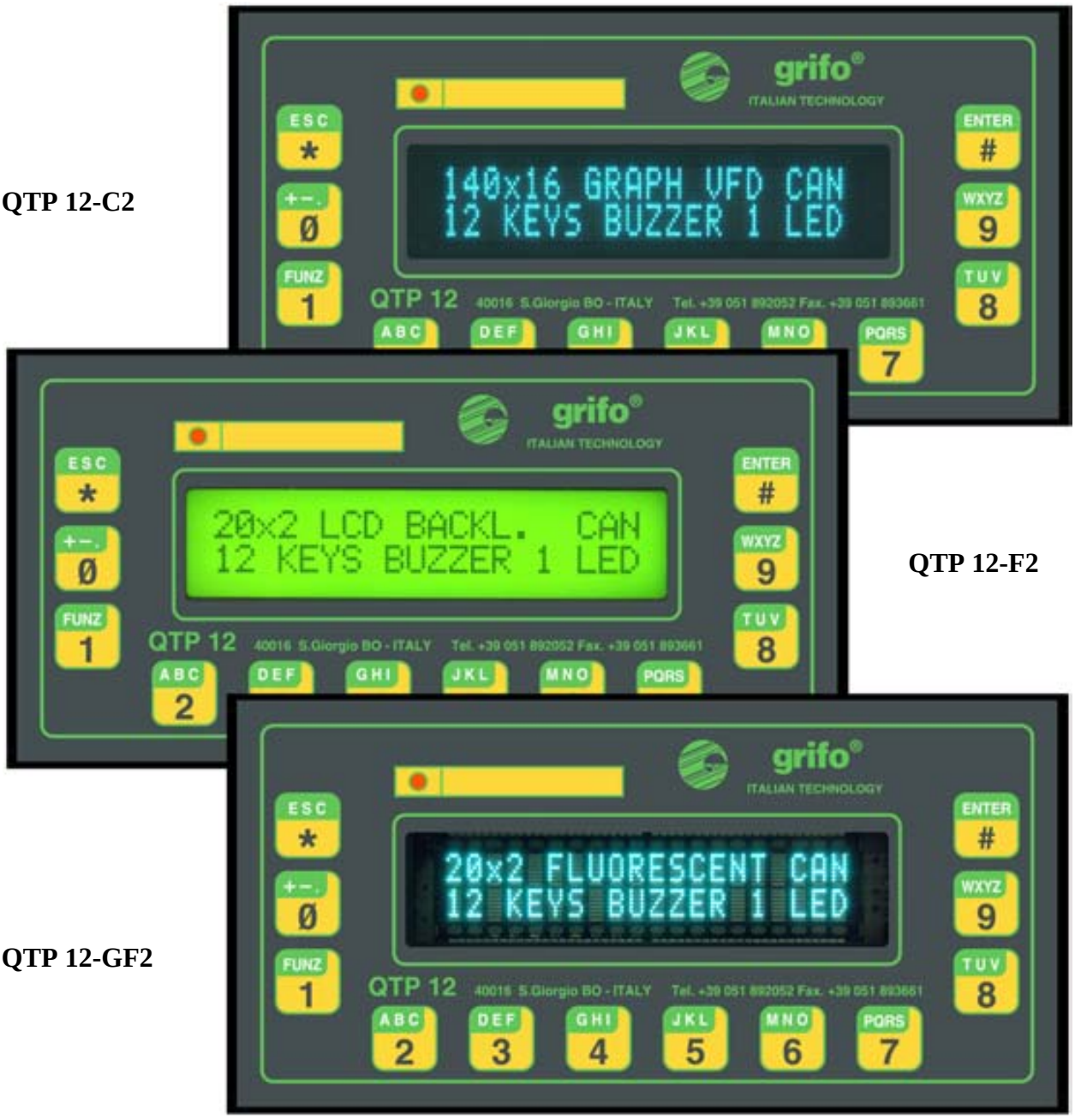
FIGURE 1: LOCATION OF HARDWARE AND FIRMWARE VERSION

The firmware version number can be also directly required to the terminal by using a dedicated command.
Normally the **QTP 12** is always supplied with the latest firmware version that is available but, for specific requirements, the user can receive also a different version; he must carefully specify this particular condition in the order.

GENERAL INFORMATION

QTP 12 basically is an operator interface provided of IP 54 protection on front side, specifically designed for a direct use on automatic machines. Among the most important aims of **QTP 12** we can remind the representation of information and the simplified input of user selections; moreover the availability of interesting additional features make it the right component to solve many applications in the civil, domestic and/or industrial fields, always by keeping an optimum price/performance ratio.
QTP 12 is available with alphanumeric displays by 20 characters for 2 lines in two different types LCD with LED backlight or Fluorescent and with graohic display 140x16 pixels. In addition on the front panel there are: a 12 keys membrane keyboard, a personalization label slot (used to carry a name or the user's own logo), and one indicator LED.
A pratical and robust metallic container, in aluminium shape with the standard DIN 72x144 size allows a direct mounting in front panel modality. A back side openings allows to reach the connectors that can be used for the required wirings. The enclosed brackets supplied with **QTP 12** let the user mount and/or unmount the terminal by performing a simple rectangle digging up, on the support panel, that normally is the front side of the electric box.
QTP 12 is the best choice whenever the user needs to show information, messages, status, etc. and the 12 keys are sufficient for user interaction; in fact **QTP 12** gives the possibility to store in the on board serial **EEPROM** up to 3371 messages. These messages can be shown on the display, also in sliding mode, by simply sending a proper command sequence, through communication line. With this feature the master program space and its execution time are optimized or even erased, in fact messages must not be sent to the panel every time, they are already stored inside EEPROM of the **QTP 12**. Furthermore it is possible to get messages back through the communication line and read them again. So **QTP 12** can be used as little mass memory where the user can save and read set-up informations, passwords, identification codes, etc. The horizontal scrolling attribute for the saved messages, let the user displays more information on less space: on the first row of the display up to 200 characters can be shown in a self managed sliding modality.
The module's asynchronous serial line can be buffered with the most frequently used electric protocols and thanks to this feature the **QTP 12** can be connected to each systems available on the market. Furthermore low cost networks of **QTP**s can be realized where many different operator panels can be contemporaneously managed. Alternatively the terminal can be driven through a synchronous I2C BUS line that allows the connection on local networks. The interconnection with other devices is ensured also by the optional CAN line, that increase the possible applications fields and improves overall network performances.
CPU section features 16K FLASH with ISP interface for comfortable programming through the asynchronous serial line. This allows an easy development environment for the user application that doesn't require any additional system, with a considerable cost reductions. The user can write the application program with 8051 compatible code, by using one of the numerous comfortable development tools.
The **QTP 12** is able to execute an entire range of display commands including: clear the entire screen or part of it, cursor position and movement, buzzer activation, characters definition, messages management, etc., with command codes compatibile to **ADDS Viewpoint** standard. Many other commands allow the use of the other resources of the operator panel, at high level; in other words the user doesn't have to directly interact on the hardware sections but he must simply use the provided commands.
Main features of **QTP 12**, including the available otions, are as follows:

QTP 12-C2



QTP 12-F2

QTP 12-GF2

FIGURE 2: AVAILABLE MODELS

- Dimension: front size **72x144 mm**; fore depth of **9 mm**; back depth of **28 mm**.
- Remarkably **low price**.
- **Alluminium container** with frontal frame in plastic, provided with **mounting clamps**.
- **Front panel** with keyboard and display window protected by anti scratch **polyester cover**.
- **Membrane** keypad with **12 keys** provided of double serigraphy: numeric and function.
- **Debouncing**, **autorepeat** and **keycklick** functions for the keys pressed.
- The **code** of the pressed key can be **changed** and moreover the not used keys can be even **disabled**.
- **Surface** or flush panel **mounting**.
- **IP 54** standard **protection** on front side.
- **3** different **models**, with different displays types:
  - **QTP 12-C2**: alphanumeric **LCD** backlight, **2** lines of **20** characters
  - **QTP 12-F2**: alphanumeric **Fluorescent**, **2** lines of **20** characters
  - **QTP 12-GF2**: graphic **Fluorescent**, **140** by **16** pixels
- Characters dimensions:
  - **QTP 12-C2**: **3.2 x 4.9** mm
  - **QTP 12-F2**: **2.4 x 4.7** mm
  - **QTP 12-GF2**: **1.5 x 3.6** mm up to **5.0 x 10.2** mm
- **LED backlite** of LCD display.
- Comfortable regulation of LCD display contrast and VFD display brightness in order to obtain always the best **visibility** in any environmental conditions.
- **Buzzer** for BELL, keypressed and acoustic signals, all driven by software.
- **1** indicator **LED** managed by software in different modes.
- **Personalization label slot** for panel name, and/or name of LED.
- **I51** family **microprocessor**, with software selectable clock.
- Different memory types: **16K FLASH** EPROM; **2K FLASH** EPROM for Boot Loader; **0,5K RAM**; up to **64K+2K EEPROM**; **240** bytes backed **SRAM**.
- **EEPROMs** used for permanent storage of setup parameters, messages, user characters, keys codes, etc.
- Memorization on **EEPROMs** and visualization on display, of maximum **3371** different **messages**, even with auto **scrolling** mode.
- The text **messages** managed by firmware **reduce** the user program and thus the communicated data.
- Possibility to save and load data from the on board **non volatile memories** (backed SRAM and EEPROM).
- Asynchronous serial line with **RS 232** or **RS 422**, **RS 485**, passive **Current Loop** electric protocols.
- Synchronous **I2C BUS** communication line.
- **CAN** communication line provided of proper line driver.
- **Network** connection through the available serial line and proper **logic protocols**, up to **256** different **units**.
- Several **physic protocols** selectable for all the serial communication lines.
- Functionality as **serial <-> I2C BUS converter**, capable to manage each peripheral devices with this interface (temperature sensors, A/D and D/A converters, etc.).
- **Local setup** for required operating modalities.
- 8 user characters provided of **selectable patterns**.
- Up to **256 different characters** defined on display and thus visible.
- Real Time Clock (**RTC**) backed by proper **Lithium** battery.
- **1 digital output**, 0/5V open collector type, connected to pods of internal printed circuit

board and driven by different commands.
- Possibility to **set** and **acquire** the Real Time Clock and its seven temporal parameter (hours, minutes, seconds, day, month, year and week day). The current **date** and **time** can be autonomously **visualized** on display, with attributes and positions defined by user.
- Management of a complete **alarm clock** that can be set on hour, minutes, seconds, day and month. When the alarm time is reached the **digital output** can be **enabled**, with a selectable activation **endurance** time.
- **Transparent** functional modality: the data received by user program, if they are not commands, are directly visualized on display while the keys pressed and possible responses of the commands are returned to the same program. This modality is normally defined **dumb terminal**.
- Tens of commands dedicated to visualization and other operations, compatible with **ADDS Wiew-Point** standard.
- Possibility to enable an **autonomous visualization** with different attributes that is automatically shown at **power on**.
- **5** comfortable and standard **connectors** for a fast cablation.
- Wide range **DC** or **AC** power supply from **5 Vdc** to **24 Vac**.
- Total **power consumption** change according with used configuration, from **1.6** to **2.5 W**.
- On board electronic protection aganist voltage peaks, by **TransZorb™**.
- For specific requirements about front panel, consumption, functionality and price, please contact directly **grifo®**.

Here follows a description of the board's functional blocks, with an indication of the operations performed by each one.

BUZZER

QTP 12 has a circuitry that generates a steady sound, based on a capacitive buzzer. By software, through some specific comands, this circuitry can be enabled, disabled or intermittent, it can generate a simple beep, it can signal a key pressed and it can signalize possible malfunctions.
When, after a power on, the card generates a fixed or intermittent sound and it doesn't work correctly, there is a wrong condition that must be resolved: please contact **grifo®** technicians.

ASYNCHRONOUS SERIAL LINE

The most diffused communication with the master unit is performed through an asynchronous serial line, that in default configuration, is electrically configured in **RS 232** but using a proper indication in the order, it can be configured in:

RS 422	->	**.RS422** option
RS 485	->	**.RS485** option
Current Loop	->	**.CLOOP** option

The physical protocol of the serial line is completely configurable through a dedicated setup modality that let the user select the values listed in TECHNICAL FEATURES chapter, by the simple use of four keys. Finally the logic protocol can be point to point or master slave type, using the nineth bit technique; this latter, when used in conjunction with one of the options above described, allows the connection of many **QTPs** on a network and to communicate with terminals of the same or different type, easily and efficiently.

KEYBOARD

QTP 12 has a membrane keyboard with **12 keys** located around the display that offer a cheap solution for user data input even when the data are heterogeneous and complex. All the keys are metallic dome type so they provide a tactile sensation of the key pressed and they withstand the knocks and bumps of industrial life. All the keys have a standard label (see figure ??) that satisfy the normal man-machine interface requirements. Remarkable is the presence of numeric digits, the whole alphabet and some functions that allows to input any kind of data and to execute any kind of command.
Moreover the keys are equipped with autorepeat and they are totally software reconfigurable or on the other hand the code returned when a key is pressed can be changed or disabled. It is also possible to switch on/off the **keyclick** function, i.e the buzzer short activation each time a key is pressed.
Please remark that the four central keys may assume a variable functionality, in case of graphic display installed. This allows to draw on screen their function names and change them according to needs of the application (e. g. START, STOP, INS, DEL, ALRM, INFO, etc.).
Four keys are used to define some of the functional parameters, as described in proper paragraph LOCAL SETUP.
In addition, a personalization label can be added on the frontal of the keyboard in order to customize and/or identificate the terminal, as described in APPENDIX C.

EEPROM

QTP 12 has a base EEPROM (2 KBytes) for storing setup, communication protocol, identification name, keys codes, user characters patterns, messagges, and so on. Many of the stored data have vital importance so a serial EEPROM has been choosen to obtain the best warranties on validity and maintenance of the saved information, even when power supply is not available.
It is really interesting the feature of 20 characters messages that can be first saved and then read or shown on the display at any moments, just giving a proper command to the terminal, with the right message identification number or numbers. **QTP 12** also manages the visualization of these messages in scrolling mode: on an single line it shows more text than it could be visible in normal condition.
The number of managed messages can be increased by ordering the **QTP 12** with one of the optional and additional EEPROM:

.EE128 (16 Kbytes)
.EE256 (32 Kbytes)
.EE512 (64 Kbytes)

For detailed information about messages please read COMMANDS FOR MESSAGES MANAGEMENT paragraph.

ON BOARD POWER SUPPLY

One of the most important peculiarity of **QTP 12** is its own switching power supply that requires an input voltage variable from **8÷24 Vac** or **10÷38 Vdc**; this section generates all the voltages used by the module.
As alternative, **QTP 12** without power supply can be ordered (by using the codes **.5Vdc** or **.ALIM**): in this case a +5 Vdc stabilized power supply must be provided by an external source.
For detailed information about power supply section, please refer to ELECTRIC FEATURES and POWER SUPPLY paragraphs.

DISPLAY

QTP 12 is available with **Fluorescent** or backlit **LCD** alphanumeric displays **20x2** characters or **graphic fluoresecnt 140x16** pixels.
LEDs backlight of LCD models ensures a good visibility even when the environmental lighting changes and if it necessary the user can modify the contrast regulation by acting on a specific trimmer; viceversa on VFD display the brightness can be regulated by software. Another important features of **QTP 12** displays is their wide viewing angle that allows a good visibility from each frontal position. Further information on each display are reported in TECHICAL FEATURES chapter and APPENDIX B and D.
As described in the chapter dedicated to commands, **QTP 12** with graphic display (**QTP 12-GF2**) can execute all kinds of commands (graphic and alphanumeric), while **QTP 12** with alphanumeric display (**QTP 12-C2** and **QTP 12-F2**), of course, cannot execute graphic commands.
The user must choose the right display (so the right **QTP 12** model) that is sufficient for the information to visualize and for his visibility requirements. For specific requirements on current consumption, visibility and price, the card can be provided even with LCD display not backlighted: for detailed information about these options and their availability, please contact directly **grifo®** offices.

CAN INTERFACE

QTP 12 can have, as option, a complete CAN interface that supports the **BasicCAN** and **PeliCAN 2.0B** standards protocols. With this feature the user can afford and solve many problems as: high speed data trasfer, long distance communication, autonomous errors management, multimaster and multislave networks support, etc.
The code used to order this option is: **.CAN**

FIGURE 3: QTP 12 COMPLETE VIEW

I2C BUS LINE

Through the synchronous serial interface in I2C BUS the **QTP 12** can perform two different communications:

- slave mode = the command unit operates as a master and communicate to **QTP** either the commands and the responses through the I2C BUS line; it is supported the communication in short local networks, with other units of the same and/or different type.
- master mode = the **QTP** communicates with peripheral devices in I2C BUS (sensors, A/D, D/A, etc.) and it acts as a converter; naturally the operations to perform on the line are decided by the command unit that communicate with **QTP**, through the asynchronous serial line.

The physic protocol of the described modes is partially configured through the proper program of local setup, that allows to select the values described in TECHNICAL FEATURES chapter, by the simple use of only four keys. Further information about the communication between **QTP 12** and other units are reported in previous paragraphs.

REAL TIME CLOCK

QTP 12 can have, as option, a Real Time Clock backed by on board Lithium battery, that manages hours, minutes, seconds, day, month, year and week day. This device is manageable by the user with appropriate software commands which allow to set time and date, to read these data or to display them on display with a given position plus format and to manage a clock alarm.This option adds a complete time information, autonomously managed by **QTP 12**, and it makes available a serial real time clock to the external command unit. This unit should control the elapsed time, activate procedures on time based events, calculate production values in a time period, start or stop processes at fixed time of a day, etc.

The code used to order this option is: **.RTC**

QTP 12 LIBRARY

For the **QTP 12** it is available a library that allow the user to decide the complete functionality of the operator panel. In this condition the **QTP** firmware is not the one described in this manual but it is developed by the user through comfortable high level programming languages. Anyway the language take advantage from the numerous functions described in the COMMANDS chapter, in fact they are grouped in the library that must be simply linked with the user firmware. This choice really simplify the **QTP** management and reduces the developing time. For detailed information about this possibility please refer to proper manual.

TECHNICAL FEATURES

GENERAL FEATURES

Resources:	IP 54 frontal Metallic container complete of mounting clamps 1 indicator LED driven by software Membrane keyboard with 12 keys, software reconfigurable Slot pocket for personalization label Buzzer for beep, keyclick or acoustic feedback Full duplex, asynchronous serial line, buffered in RS 232 or RS 422, RS 485, Current Loop (options) Synchronous I2C BUS serial line in master and/or slave modality CAN interface (option) Alphanumeric or graphic display in three different models Circuitry that regulates LCD display contrast Real Time Clock backed by lithium battery (option) 1 digital outputs in open collector (option) Switching power supply
Displays:	alphanumeric LCD 20x2 chars, LED backligt alphanumeric Fluorescent 20x2 chars graphic Fluorescent 140x16 pixels
CPU:	89C5115 or 89C51CC02 with 14.7456 MHz crystal *Default: 89C5115*
Memories:	16K FLASH EPROM 2K FLASH EPROM for Boot Loader 0,5K RAM 2K EEPROM up to 64K EEPROM (option) 240 byte backed SRAM (option)
Power on time:	100 msec
Timing resolution:	2,5 msec
Base EEPROM write time:	8 msec
Optional EEPROM write time:	5 msec
Keys autorepeat time:	After 500 msec and then every 100 msec
Buzzer intermittent time:	500 msec
LED intermittent time:	500 msec
Messages shift time:	500 msec
RTC visualization time:	500 msec

User EEPROM bytes: 40

User backed SRAM bytes: 224

Messages number: 95, 914, 1733, 3371
Default: *95*

Max units on network: 256 with asynchronous line and Master-Slave 9 bits
128 with synchronous I2C BUS line

Communication: Selectable between Normal, Master-Slave 9 bits, I2C BUS
Default: *Normal*

Communication physic protocol in Normal, Master-Slave 9 bits:

Baud rate:	1200, 2400, 4800, 9600, 19200, 38400
Stop bit:	1 or 2
Parity:	none
Bits x chr:	8, 9
Slave Address:	from 00H to FFH at step of 1
Default:	*19200 Baud, 1 Stop, No parity, 8 Bits, Slave Address = 80H*

Communication physic protocol in I2C BUS:

Bit rate:	from 500 to15000 bits/second
Modality:	Slave
Slave Address:	from 00H to FEH at step of 2
Default:	*Slave Address = 80H*

Receive buffer size: 40 characters

Transmit buffer size: 20 characters

PHYSICAL FEATURES

Size: DIN 72x144: 144 x 72 x 37 mm (W x H x D)
156 x 72 x 80 mm (W x H x D) with clamps
See outline dimension in APPENDIX C

Size of breaking for mount: 138 (min) x 66 (min) x 10 (max) mm (W x H x D)
See outline dimension in APPENDIX C

Pixels size:

LCD 20x2:	0.5 x 0.6 mm (W x H)
VFD 20x2:	0.4 x 0.5 mm (W x H)
VFD 140x16:	0.4 x 0.6 mm (W x H)

See dimension in APPENDIX D

Characters size:

LCD 20x2:	5 x 7 dots = 3.2 x 4.9 mm (W x H)
VFD 20x2:	5 x 7 dots = 2.4 x 4.7 mm (W x H)
VFD 140x16:	from 3 x 5 dots = 1.5 x 3.6 mm (W x H)
	to 10 x 14 dots = 5.0 x 10.2 mm (W x H)

See dimension in APPENDIX D

Viewing area size:	LCD 20x2: 73.5 x 11.5 mm (W x H) VFD 20x2: 70.8 x 11.5 mm (W x H) VFD 140x16: 69.9 x 11.5 mm (W x H) See dimension in APPENDIX D
Weight:	320 g max.
Mounting:	Surface or front panel mounting, through provided clamps At sight on a bearing surface
Temperature range:	From 0 to 50 °C
Relative humidity:	20% up to 90% (without condense)
Connectors:	CN1: quick release screw terminal, 2 pins, male, pitch 5 CN2: D type connector, vertical, 9 pins, female CN3: quick release screw terminal, 3 pins, male, pitch 3.5 CN6: AMP MODU II, vertical, 4 pins, male, pitch 2.54 J4: AMP MODU II, vertical, 2 pins, male, pitch 2.54

FIGURE 4: REAR VIEW

ELECTRIC FEATURES

Power voltage: +10÷38 Vdc , 8÷24 Vac (*)
or +5 Vdc ± 5% (option)

Power consumption: see next table (*)

DISPLAY model	*Consumption max. +5 Vdc*	*Consumption max. 10÷40 Vdc 8÷24 Vac*
LCD alphanumeric 20x2 backlight: QTP 12-C2	250 mA	1,7 W
Fluorescent alphanumeric 20x2: QTP 12-F2	230 mA	1,6 W
Fluorescent graphic 140x16: QTP 12-GF2	360 mA	2,5 W

FIGURE 5: CONSUMPTIONS TABLE

Output power supply voltage: +5.0 Vdc
Current available on +5 Vdc output: 400 mA - consumption max. +5 Vdc
RS 232 extravoltage protection: ±15 KV
RS 422-485 line impedance: 60 Ω
RS 422-485 termination: line termination resistor: 120 Ω
pull-up resistor on positive: 3.3 KΩ
pull-down resistor on negative: 3.3 KΩ
CAN line impedance: 60 Ω
CAN termination circuit: 120 Ω ressitor, disconnectable
Pull up resistors on I2C BUS: 10 KΩ
Back up battery: 3 V Lithium; 180 mAh; CR2032 model
Back up current: 3.5 µA
Max current on relays: 5 A
Max voltage on relays: 30 Vdc

(*) The data are referenced to 20 C° environmental work temperature (for further information please refer to chapter POWER SUPPLY).
The table on figure 5 lists the **QTP 12** power consumption referred to the different display types that can be ordered; for the wide range power supply are described the required power, in place of the current, already corrected with efficiency factor of the on board power supply section.
To reduce consumptions of **QTP 12** with LCD display it is possible to order particular models without backlight: for further information and availability, please contact directly **grifo®**.

INSTALLATION

In this chapter there are the information for a right installation and correct use of the terminal **QTP 12**. In detail there are the locations and functions of each connector, of the user settable jumpers, of the battery and any other information concerning hardware configuration.

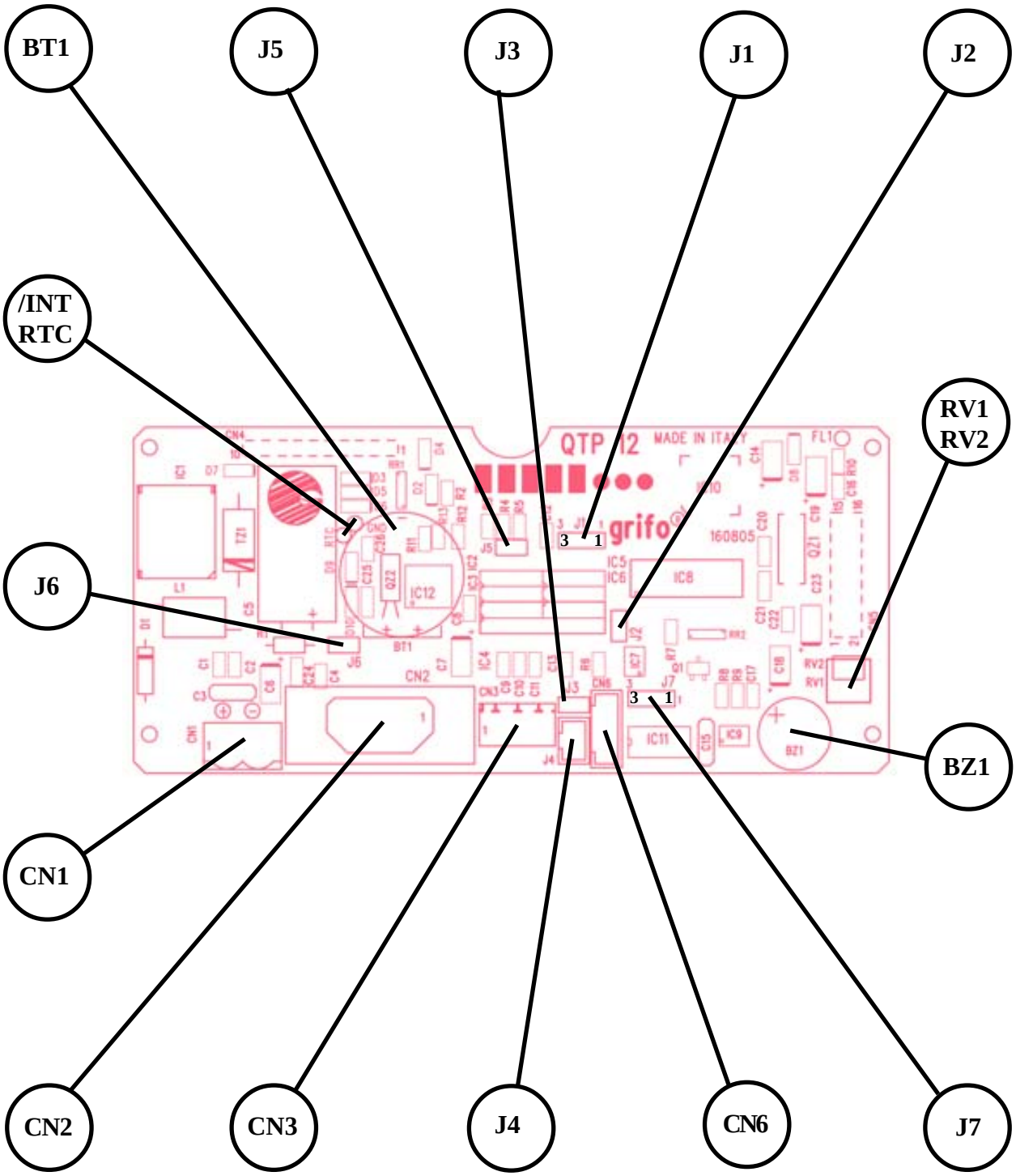


FIGURE 6: LOCATION OF JUMPERS, CONNECTORS, BUZZER, BATTERY, ETC.

CONNECTIONS

QTP 12 terminal has 5 connectors that can be linkeded to other devices or directly to the field, according to system requirements. Below are reported the pin outs, the meaning of the connected signals (including their directions) and some connection examples, that simplify and speed the installation phase. In addition the figures 4 and 6 show the connectors position on the board, to simplify their recognitions.
All the connectors are accessible from the back of the alluminum container, through a proper breaking in the rear side that allows comfortable insertion and deinsertion.

CN1 - POWER SUPPLY CONNECTOR

CN1 is a vertical, 2 pins, male, quick release screw terminal connector, with 5 mm pitch.
On CN1 must be connected the single power supply voltage for the terminal that can be one out of three different types, as described by following figures

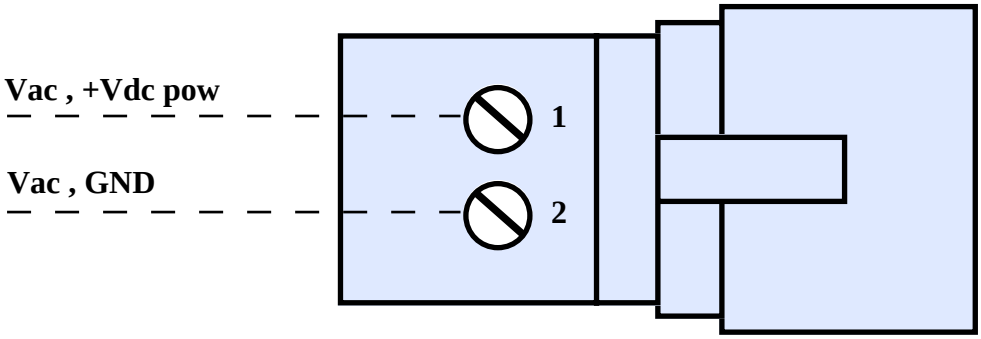


FIGURE 7: CN1 - POWER SUPPLY CONNECTOR

Signals description:

Vac = I - AC power supply lines connected to on board switching section; these signals must be in the range **8÷24 Vac**.

+Vdc pow = I - DC power supply lines connected to on board switching section (**+10÷+38 Vdc**) or stabilized (**+5 Vdc**) voltage connected to on board logic, according to ordered configuration.

GND = - Ground signal for DC power supply.

NOTE For further information about power supply configurations, please refer to paragraph POWER SUPPLY.

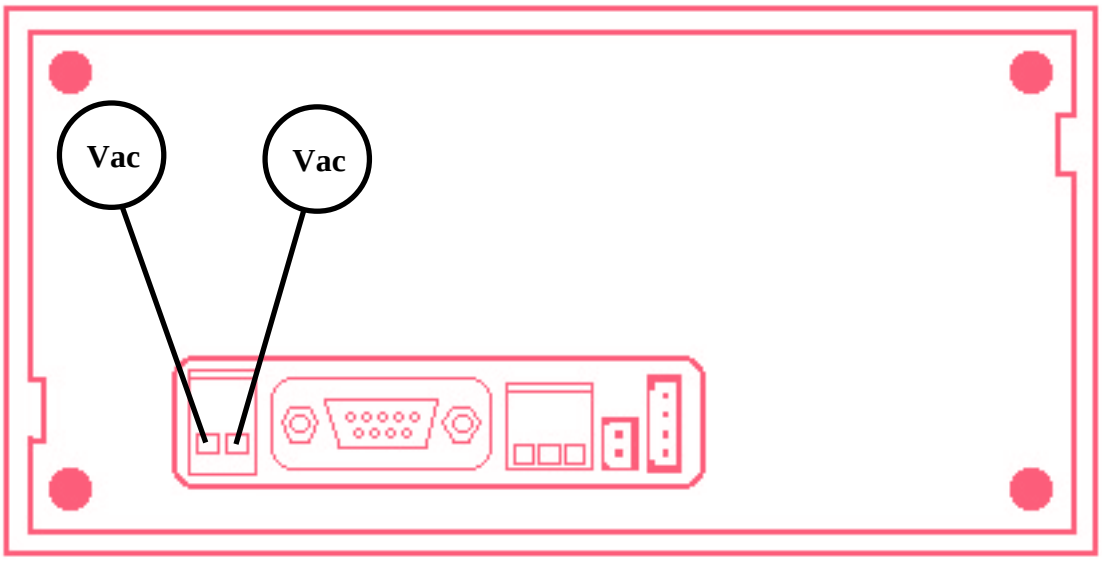


FIGURE 8: AC POWER SUPPLY 8÷24 VAC

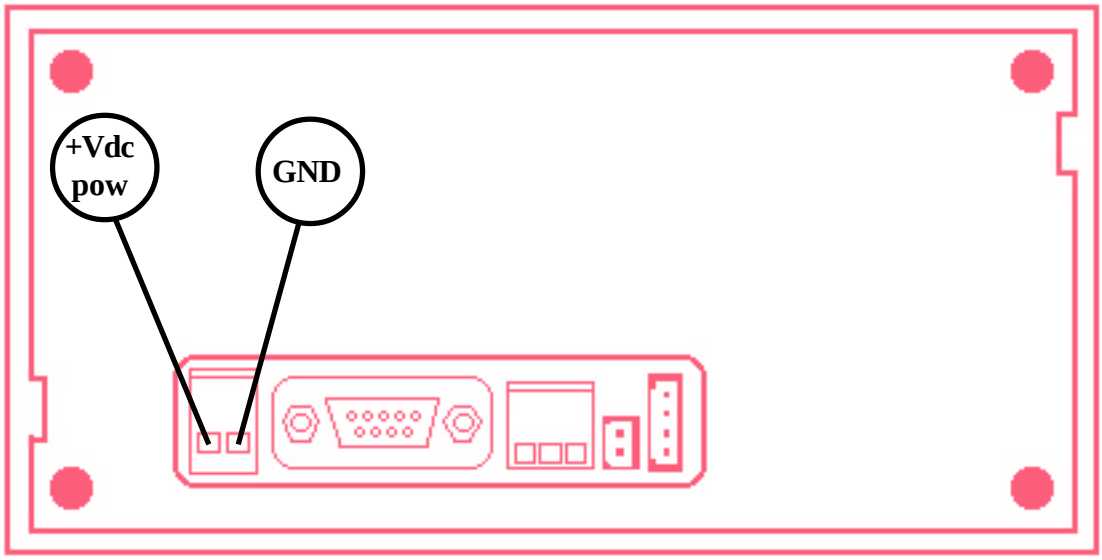


FIGURE 9: DC POWER SUPPLY +10÷38 VDC

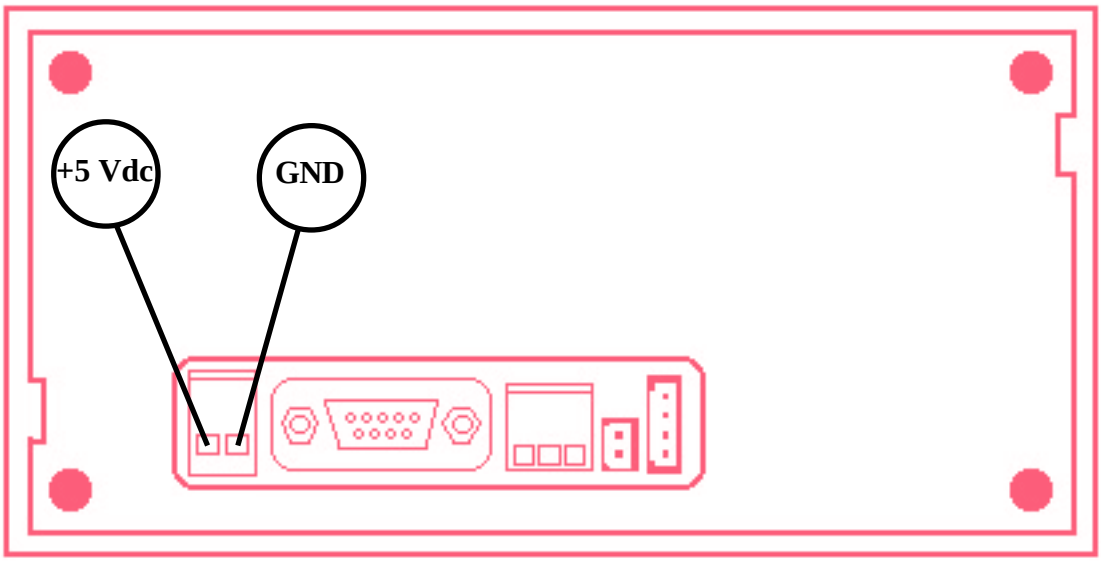


FIGURE 10: STABILIZED POWER SUPPLY +5 VDC (OPTION)

CN3 - CAN INTERFACE CONNECTOR

CN3 is a vertical, 3 pins, male, quick release screw terminal connector, with 3.5 mm pitch. Through CN3 must be connected the CAN serial communication line by following the standard rules defined by the same protocol. Signals placement has been designed to reduce interferences and to obtain a fast and comfortable node connection on the field CAN bus. The connector is available only when the .CAN option has been ordered.

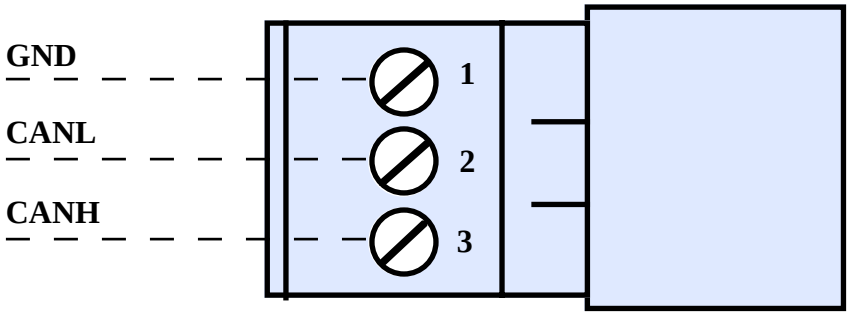


FIGURE 11: CN3 - CAN INTERFACE CONNECTOR

Signals description:

CANH = I/O - Differential line high for CAN interface.
CANL = I/O - Differential line low for CAN interface.
GND = - Ground signal.

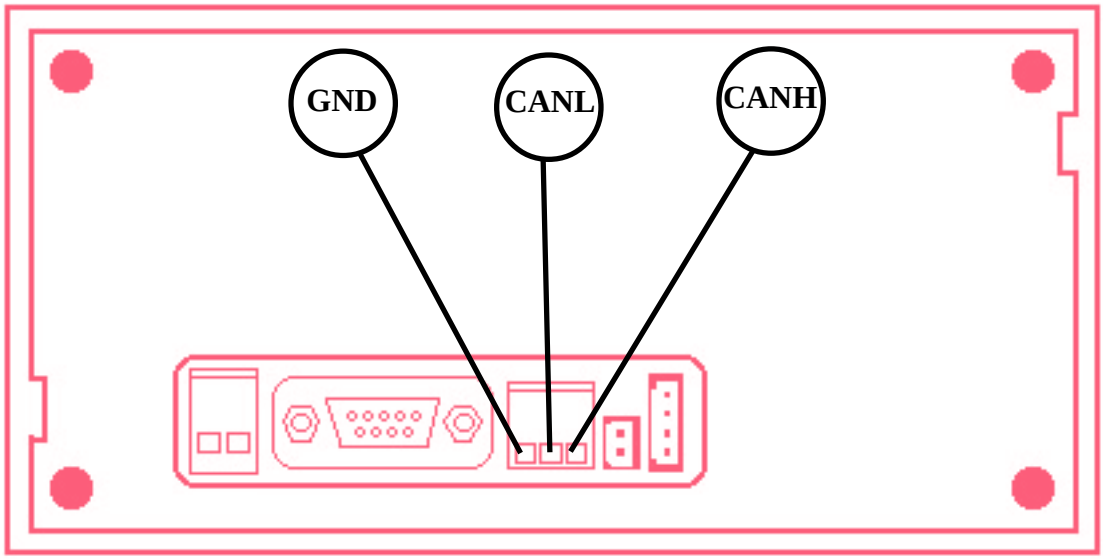


FIGURE 12: CAN LINE CONNECTION

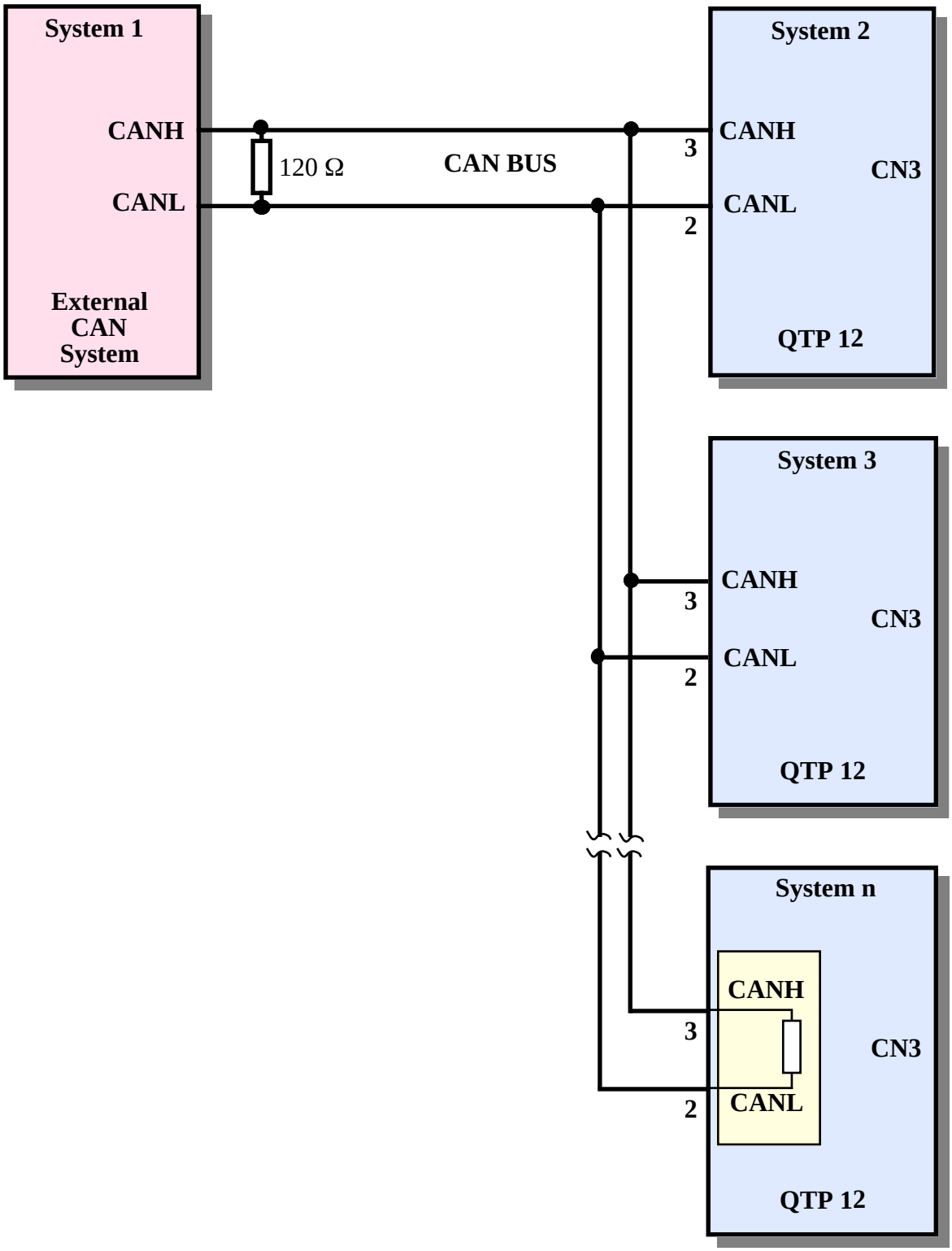


FIGURE 13: CAN NETWORK CONNECTION EXAMPLE

Please remind that a CAN network must have a line impedance of 60 Ω and for this reason two termination resistors (120 Ω) must be placed at its extrems, respectevely near the units that are at the greatest distance. On **QTP 12** the terminating circuitry is already installed: it can be connected or not through specific jumper, as explained later.
When the system to link on the CAN line have very different potentials, it is possible to connect also the grounds of the same systems, that is pin 1 of CN3. In this way any possible problems of communication and/or incorrect working, are solved.

CN6 - I2C BUS LINE CONNECTOR

CN6 is a vertical, 4 pins, male, AMP MODU II connector, with 2.54 mm pitch.
Through CN6 can be connected the synchronous communication line in I2C BUS. The signals connected respect the international normatives defined by this standard of communication and include also the power supply voltage generated on board, that can be used to supply power at external devices and/or systems. On the other hand the signals placement has been designed to reduce interferences and it is the same one available on great part of grifo® cards, to speed up the connection of different units.
The female connector for CN6 is directly available between grifo® accessories, and it can be ordered by using the codes **CKS.AMP4** or **AMP4.Cable**, as described in APPENDIX E of the manual.

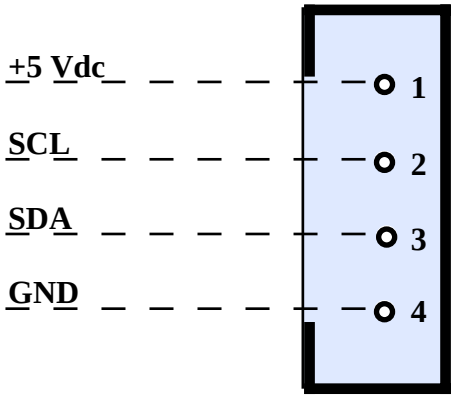


FIGURE 14: CN6 - I2C BUS LINE CONNECTOR

Signals description:

SDA = I/O - Data signal for I2C BUS communication.
SCL = I/O - Clock signal for I2C BUS communication.
+5 Vdc = O - +5 Vdc power supply signal.
GND = - Ground signal.

A complete description of I2C BUS communication is reported in next paragraphs I2C BUS COMMUNICATION and COMMAND FOR I2C BUS COMMUNICATION AS MASTER The following figures show a connection example diagram with a generic I2C BUS master unit, both in point to point and network mode:

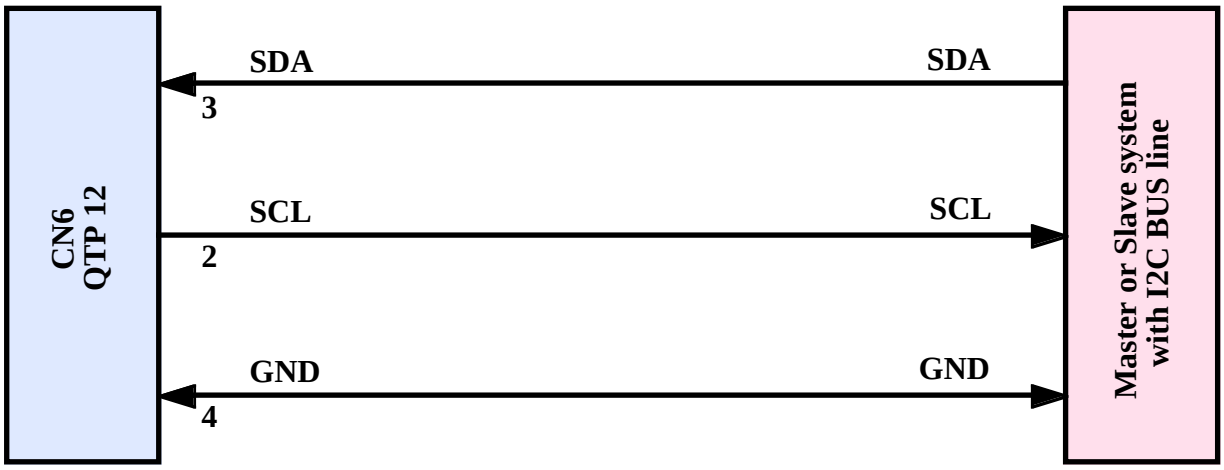


FIGURE 15: CONNECTION EXAMPLE FOR I2C BUS POINT TO POINT COMMUNICATION

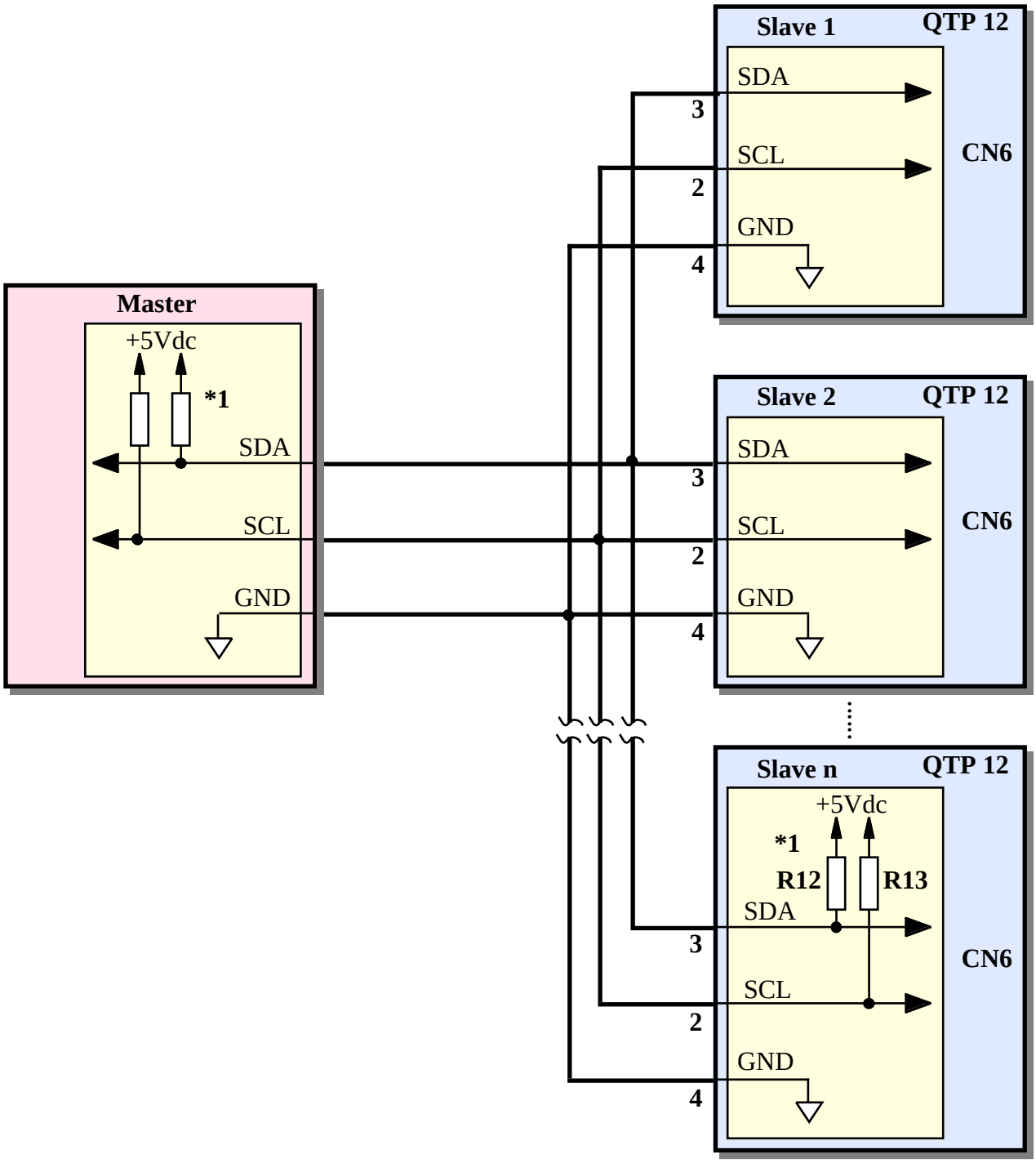


FIGURE 16: CONNECTION EXAMPLE FOR I2C BUS NETWORK COMMUNICATION

Please remind that in a I2C BUS network must be connected two pull up resistors at the net extrems, respectevely near the master unit and the slave unit at the greatest distance from the master.
On **QTP 12** these resistors are always present in default configuration and they have the value described in ELECTRIC FEATURES paragraph. The user must select or configure the I2C BUS devices to connect, by taking care of this feature. In detail on **QTP 12** the described resistors (***1**) must be removed on the units that are not at the line extremities, as shown in previous figure, on slaves 1 and 2.
For further information please refer to document "*THE I2C-BUS SPECIFICATIONS*", from PHILIPS semiconductors.

J4- BOOT LOADER ACTIVATION CONNECTOR

J4 is a vertical, 2 pins, male, AMP MODU II connector, with 2.54 mm pitch.
This connector enables the DEBUG modality of **QTP 12** that allows the user to reprogram the internal FLASH EPROM. Normally, this operation is necessary only when the user must develop its own management program, for example in conjunction with the library firmware **.LIB**.

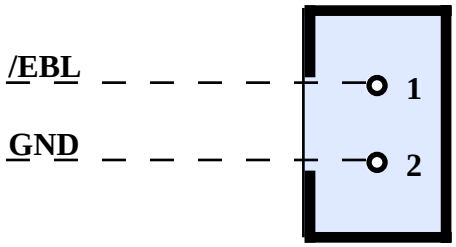


FIGURE 17: J4 - BOOT LOADER ACTIVATION CONNECTOR

Signals description:

/EBL = I/O - Boot Loader enable signal.
GND = - Ground signal.

The J4 connector can be used also as a simple 2 pins jumper (as described on figure ??) infact to start Boot Loader the /EBL signal must be enabled, that means connect it to ground. In this specific condition the Boot Loader is enabled by simply inserting a jumper on the connector. Viceversa, when the activation must be remoted, the female connector for J4 must be used: it is directly available between grifo® accessories, and it can be ordered by using the codes **CKS.AMP2** or **AMP2.Cable**, as described in APPENDIX E of the manual.

CN2 - SERIAL LINE CONNECTOR

CN2 is a D type, 9 pins, female, vertical connector.
On CN2 are available all the signals of the asyncronous serial line, buffered with one of the diffused electric standards RS 232, RS 422, RS 485 or Current Loop, that allows the complete management of the panel. Placing of the signals has been designed to reduce interferences and electrical noises and to simplify connections with other systems, while the electric protocols follow the CCITT directives of the used standard.

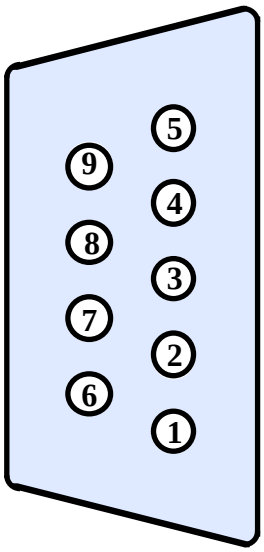


FIGURE 18: CN2 - SERIAL LINE CONNECTOR

Pin	*Signal*	*Direction*	*Description*
RS 232 serial line:			
2	**RX RS232**	= I	- Receive data for RS 232.
3	**TX RS232**	= O	- Transmit data for RS 232.
5	**GND**	=	- Ground signal.
RS 422 serial line:			
1	**RX- RS422**	= I	- Negative receive data for RS 422.
2	**RX+ RS422**	= I	- Positive receive data for RS 422.
3	**TX- RS422**	= O	- Negative transmit data for RS 422.
4	**TX+ RS422**	= O	- Positive transmit data for RS 422.
5	**GND**	=	- Ground signal.
RS 485 serial line:			
1	**RXTX- RS485**	= I/O	- Negative receive and transmit data for RS 485.
2	**RXTX+ RS485**	= I/O	- Positive receive and transmit data for RS 485.
5	**GND**	=	- Ground signal.
Current Loop serial line:			
9	**RX- C.L.**	= I	- Negative receive data for Current Loop.
8	**RX+ C.L.**	= I	- Positive receive data for Current Loop.
7	**TX- C.L.**	= O	- Negative transmit data for Current Loop.
6	**TX+ C.L.**	= O	- Positive transmit data for Current Loop.

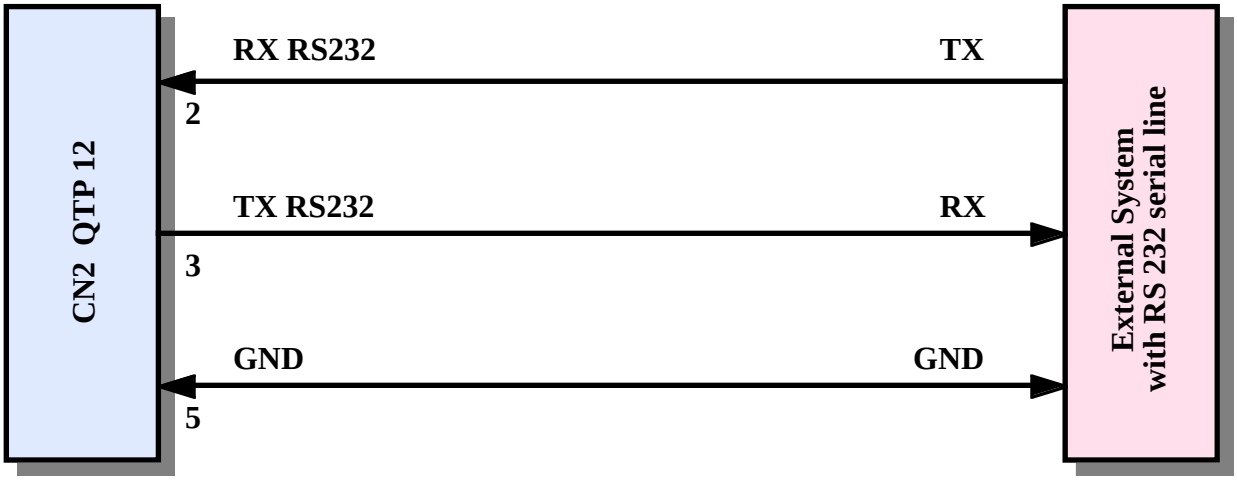


FIGURE 19: RS 232 POINT TO POINT CONNECTION EXAMPLE

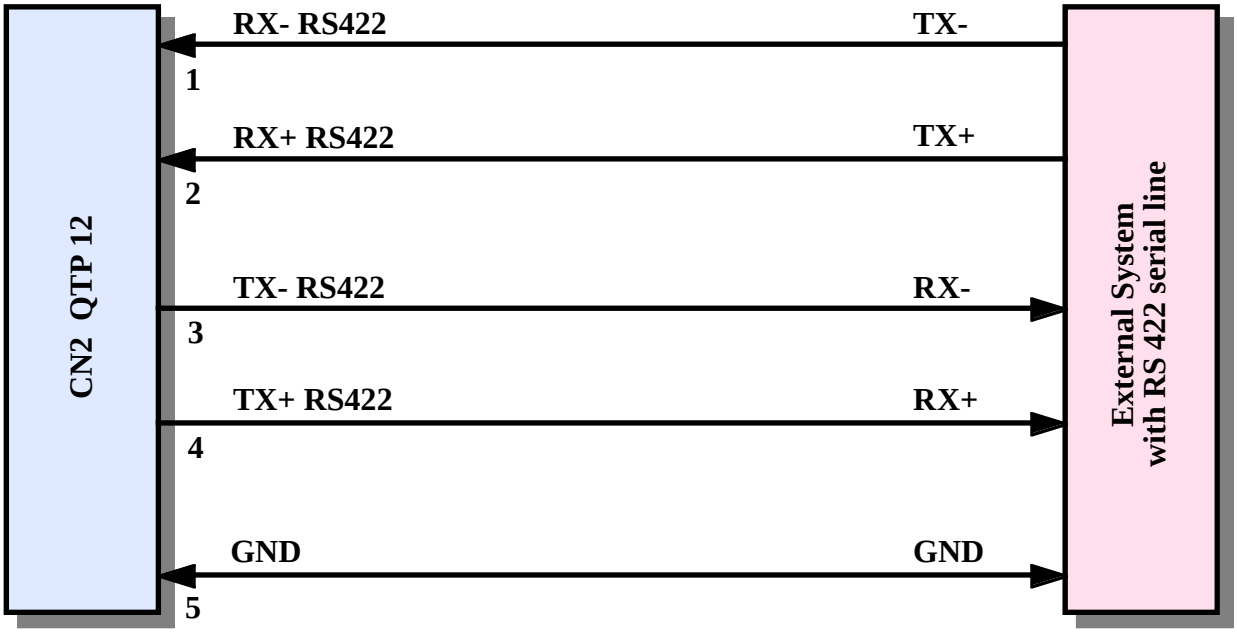


FIGURE 20: RS 422 POINT TO POINT CONNECTION EXAMPLE

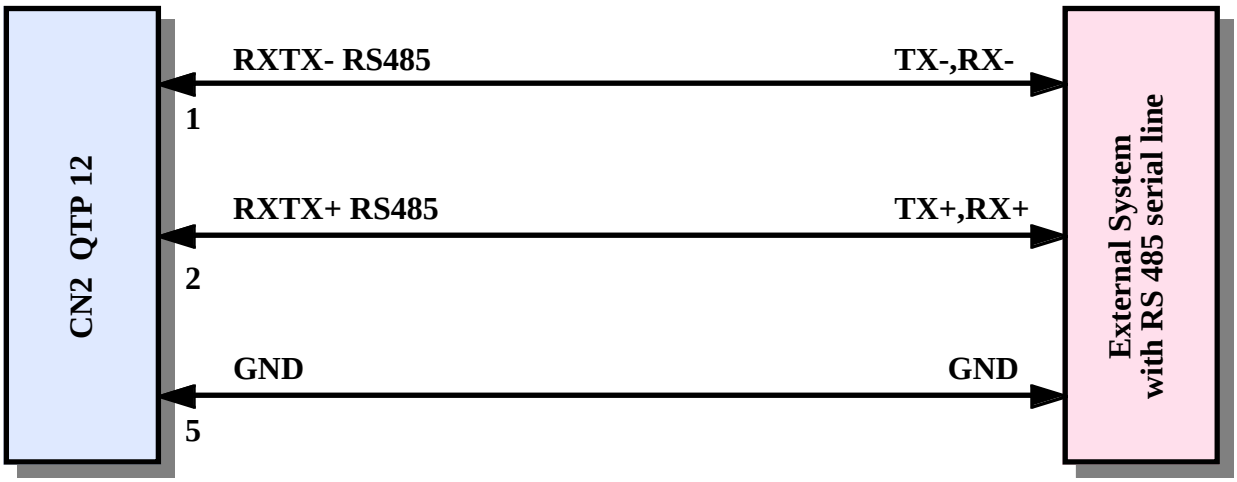


FIGURE 21: RS 485 POINT TO POINT CONNECTION EXAMPLE

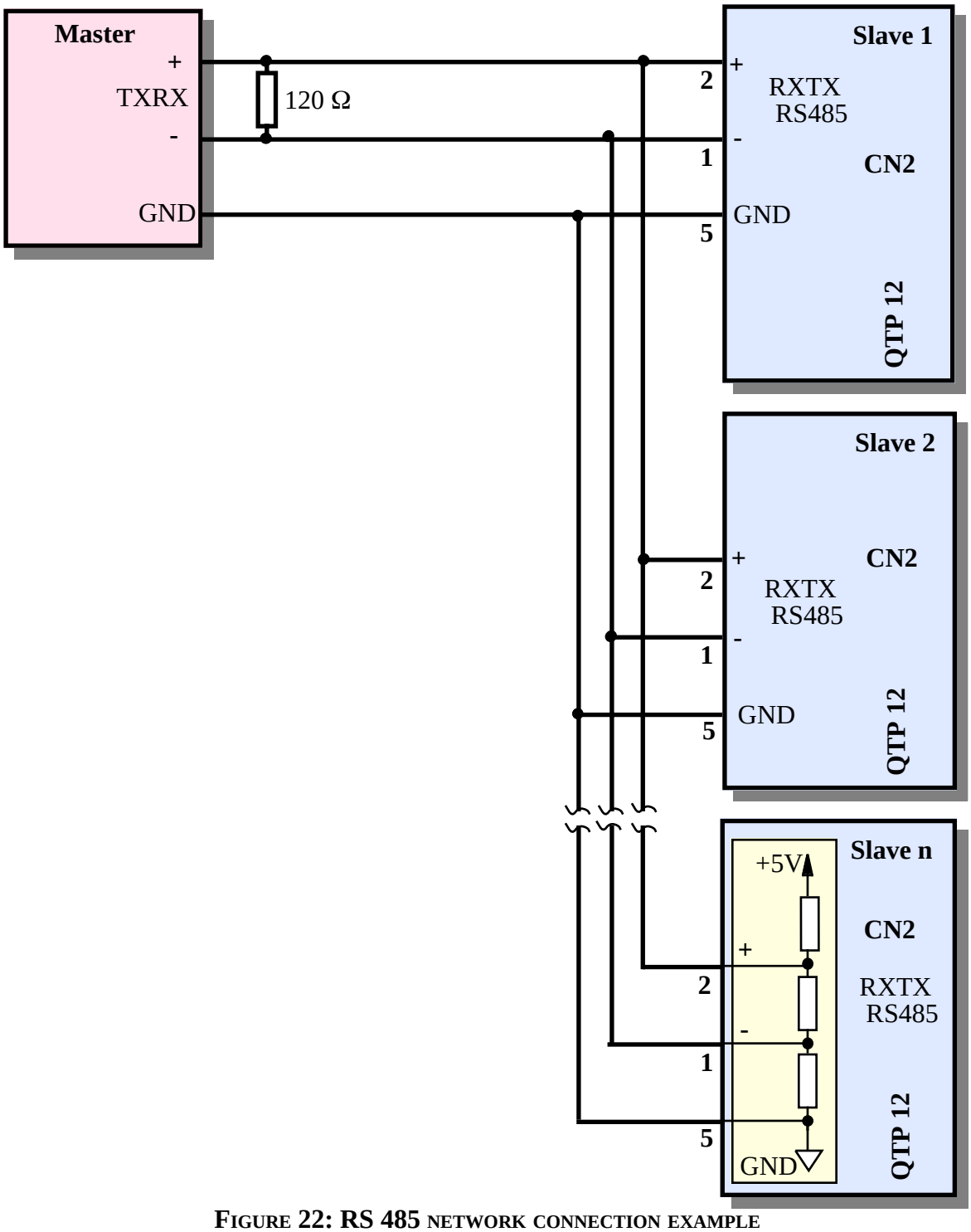


FIGURE 22: RS 485 NETWORK CONNECTION EXAMPLE

Please remark that in a RS 485 network two forcing resistors must be connected across the net and two termination resistors (120 Ω)must be placed at its extrems, respectevely near the Master unit and the Slave unit at the greatest distance from the Master.
Forcing and terminating circuitry is installed on **QTP 12** board and it can be enabled or disabled through specific jumers, as explained later.
About termination resistor of Master unit,connect it only if not already present (for example many RS 232-RS 485 converters already have it inside).
For further information please refr to TEXAS INSTRUMENTS Data-Book , "*RS 422 and RS 485 Interface Cicuits*", the introduction about RS 422-485.

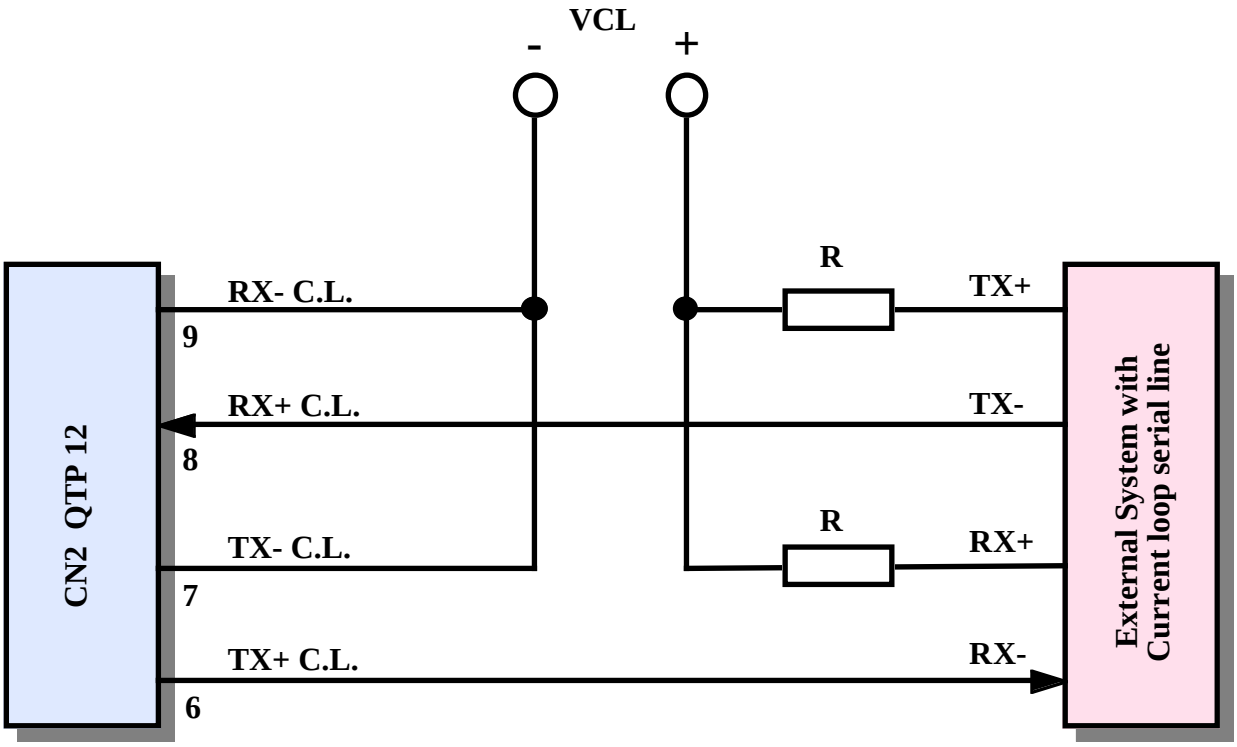


FIGURE 23: CURRENT LOOP 4 WIRES POINT TO POINT CONNECTION EXAMPLE

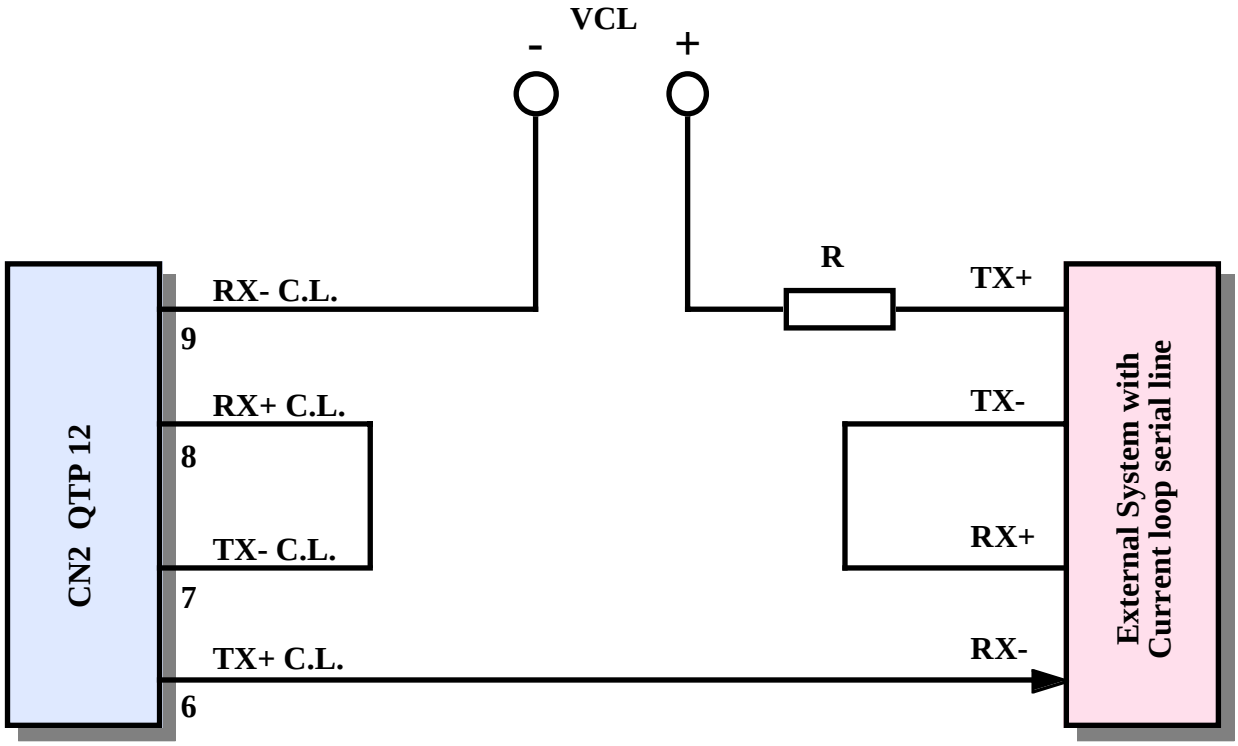


FIGURE 24: CURRENT LOOP 2 WIRES POINT TO POINT CONNECTION EXAMPLE

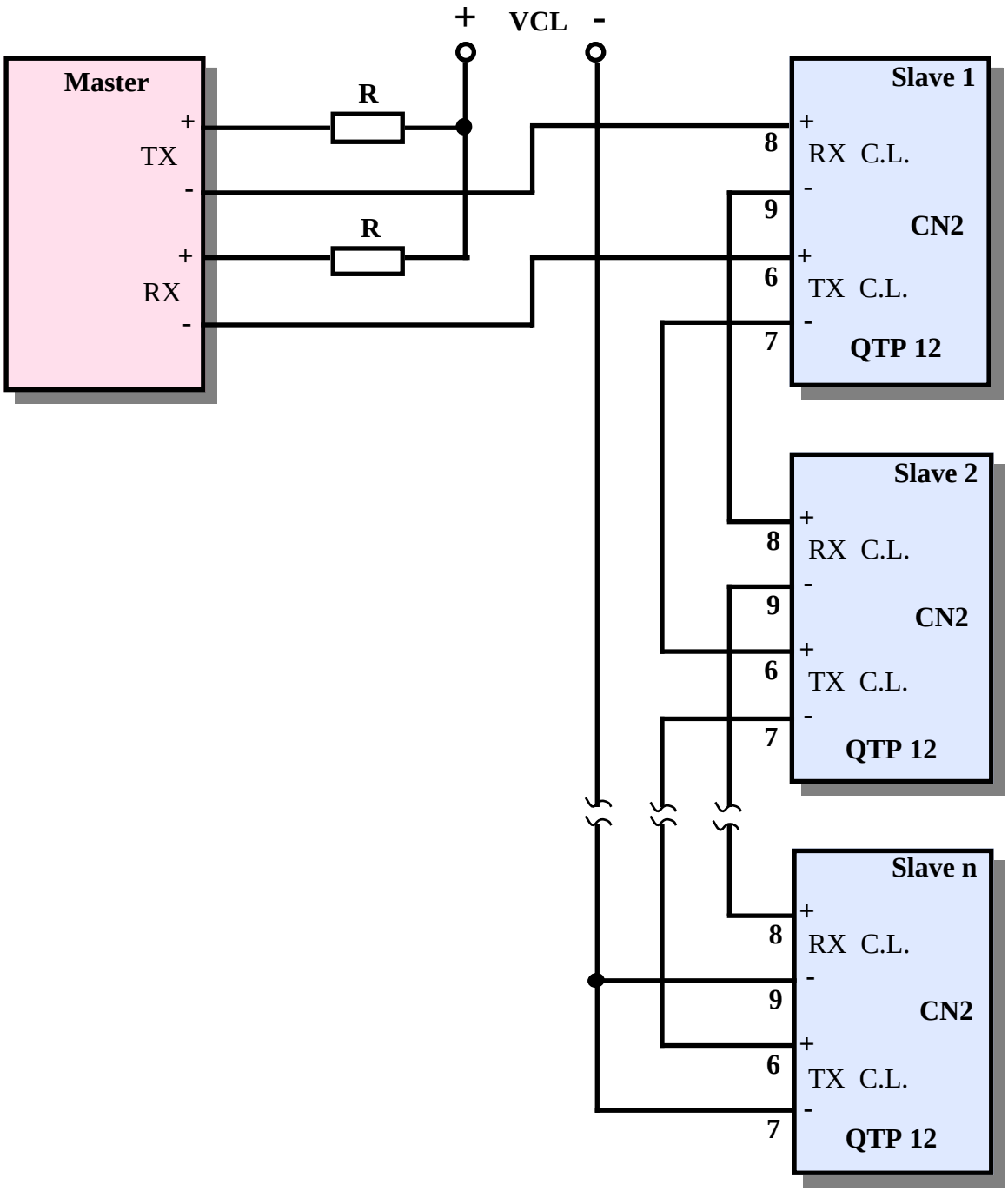


FIGURE 25: CURRENT LOOP NETWORK CONNECTION EXAMPLE

Possible Current Loop connections are two: 2 wires and 4 wires. These connections are shown in figures 23÷25 where it is possible to see the voltage for **VCL** and the resistances for current limitation (**R**). The supply voltage varies in compliance with the number of connected devices and voltage drop on the connection cable.
The choice of the values for these components must be done cosidering that:
- circulation of a **20 mA** current must be guaranteed;
- potential drop on each transmitter is about **2.35 V** with a 20 mA current;
- potential drop on each receiver is about **2.52 V** cwith a 20 mA current;
- in case of shortciruit each transmitter must dissipate at most **125 mW**;
- in case of shortciruit each receiver must dissipate at most **90 mW**.

For further info please refer to HEWLETT-PACKARD Data Book, (**HCPL 4100** and **4200** devices).

JUMPERS

On **QTP 12** there are seven jumpers for card configuration and by connecting them, the user can perform some selections that regards the working conditions of the card. Here below there is the jumpers list and relative functions in the possible connection modalities:

JUMPER	CONNECTION	PURPOSE	DEF.
J1	position 1-2	Configures serial line for RS 485 standard elettric protocol (2 wires half duplex).	
	position 2-3	Configures serial line for RS 422 standard elettric protocol (4 wires half duplex or full duplex).	*
J2 , J5	not connected	Do not connect termination and forcing circuitery to RS 422, RS 485 serial line.	*
	connected	Connect termination and forcing circuitery to RS 422, RS 485 serial line.	
J3	not connected	Does not connect 120 Ω termination resistor to CAN line.	*
	connected	Connects 120 Ω line termination resistor to CAN line.	
J4	not connected	It selects the RUN modality at power on by executing the program saved on FLASH (used only for .LIB version).	*
	connected	It selects the DEBUG modality at power on by executing the Boot Loader (sed only for .LIB version).	
J6	not connected	On board battery BT1 not connected to back up circuitry.	
	connected	On board battery BT1 onnected to back up circuitry.	*
J7	position 1-2	Write protection of optional EEPROM not enabled.	*
	position 2-3	Write protection of optional EEPROM enabled.	

FIGURE 26: JUMPERS TABLE

To recognize the valid connections and locations of these jumpers, please refer to the board printed diagram (serigraph) or to figure 6 of this manual, where the pins numeration is listed.
In previous table the "*****" denotes the default connection, or on the other hand the connection setup at the end of testing phase, that is the configuration the user receives. The user can check the default configuration af all the modificable features, also in the APPENDIX E at the end of the manual.
Further information about purpose of the **QTP 12** jumpers are reported in the following paragraphs, that describe the sections where the same jumpers are used.

FIGURE 27: COMPONENTS MAP SOLDER SIDE

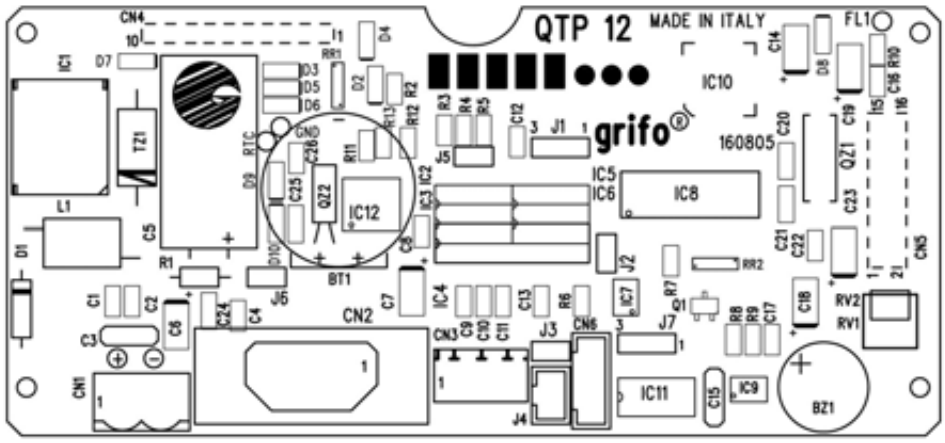


FIGURE 28: COMPONENTS MAP COMPONENTS SIDE

SERIAL LINE CONFIGURATION

Serial line of **QTP 12** can be buffered in RS 232, RS 422, RS 485 or Current Loop. By software the serial line can be programmed to operate with all the standard physical protocols, in fact the bits per character, parity, stop bits and baud rates can be decided by an opportune local setup procedure (see homonymous paragraph). Through the local setup can be selected also the logic protocol of communication, among the available modalities.
By hardware can be selected which one of the electric standards is used, through jumpers connection (as described in the previous table) and drivers installation. Some devices needed for RS 422, RS 485 and Current Loop configurations are not mounted on the board in standard configuration; this is why each fist non-standard (non RS 232) serial configuration must be always performed by **grifo®** technicians. At this point the user can change autonomously the configuration following the below information:

- SERIAL LINE IN RS 232 (default configuration)

			IC4	= driver MAX 202
			IC2	= no device
J1	=	indifferent	IC5	= no device
J2 , J5	=	not connected	IC3	= no device
			IC6	= no device

- SERIAL LINE IN CURRENT LOOP (option **.CLOOP**)

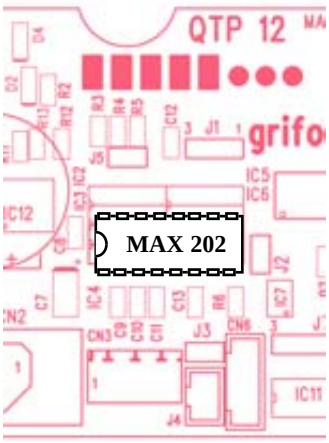
			IC4	= no device
			IC2	= no device
J1	=	indifferent	IC5	= no device
J2 , J5	=	not connected	IC3	= driver HP 4200
			IC6	= driver HP 4100

Please remark that Current Loop serial interface is passive, so it must be connected an active current loop serial line, that is a line provided with its own power supply, like described in figures 23÷25. Current Loop interface can be employed to make both point to point and multi points connections through a 2 wires or a 4 wires connection.

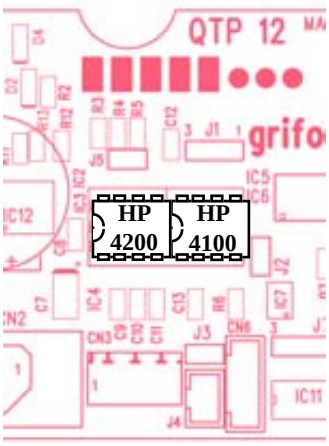
- SERIAL LINE IN RS 422 (option **.RS422**)

			IC4	= no device
			IC2	= driver SN 75176 or MAX 483
J1	=	position 2-3	IC5	= driver SN 75176 or MAX 483
J2 , J5	=	(*)	IC3	= no device
			IC6	= no device

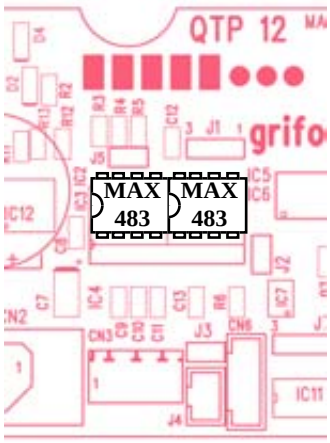
RS 422 electric protocol can be used to make 4 wires, full duplex, connections either in multi points or point to point systems.
Transmitter abilitation, essential in networks connections, is managed directly by **QTP 12** firmware by selecting the Master-Slave 9 bits logic protocol.



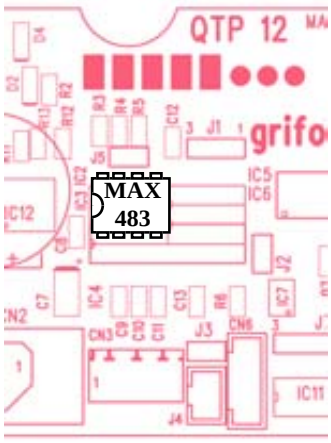
Serial line in RS 232



Serial line in Current Loop



Serial line in RS 422



Serial line in RS 485

FIGURE 29: LOCATIONS OF DRIVERS FOR SERIAL COMMUNICATION

- SERIAL LINE IN RS 485 (option **.RS485**)

			IC4	= no device
			IC2	= driver SN 75176 or MAX 483
J1	=	position 1-2	IC5	= no device
J2 , J5	=	(*)	IC3	= no device
			IC6	= no device

In this modality the signals to use are pins 1 and 2 of connector CN2, that become transmission or reception lines according to the status defined by firmware; the last must be configured with logic protocol Master-Slave 9 bits. The RS 485 electric protocol can be used to make 2 wires half duplex connections both in multi points networks and point to point connection.

(*) When the RS 422 or RS 485 serial line are used, it is possible to connect the termination and forcing circuit on the line, by using J2 and J5 jumpers. This circuit must be always connected in case of point to point connections, while in case of multi points connections it must be connected olny in the farest boards, that is on the edges of the commmunication line.
During a power on, the RS 485 driver is in reception and RS 422 transmission driver is disabled, to avoid confilcts on the communication line.

For further information about serial communication please refer to the connection examples of figures 19÷25.

OPTIONAL EEPROM PROTECTION

With jumper J7 the user can protect the optional EEPROM of **QTP 12** towards the write operations. In detail with jumper in position 1-2 the EEPROM is not protected and it can be written through proper commands, viceversa in position 2-3 the device can only be read.
Please remind that the first 95 messages and all the other data saved in base EEPROM (see paragraph DATA STORED ON EEPROM), are not interested by jumper J7 configuration.
The most important purpose of this jumper is to avoid unwanted writing and/or modifications of the numerous messages saved inside optional EEPROM, especially when they are stable. Normally the user must perform the following operations, during the installation phase:

- disable the protection by connecting J7 in position 1-2 (default configuration);

- save all the invariable messages, by taking advantage of specific commands for messages provided of number greather than 95, through a dedicated program (i.e. QTP EDIT), or a proper modality of management program;

- enable the protection by connecting J7 in position 2-3;

- at this point the management program can use the saved and protected messages through the read commands, only.

FIGURE 30: QTP 12-C2

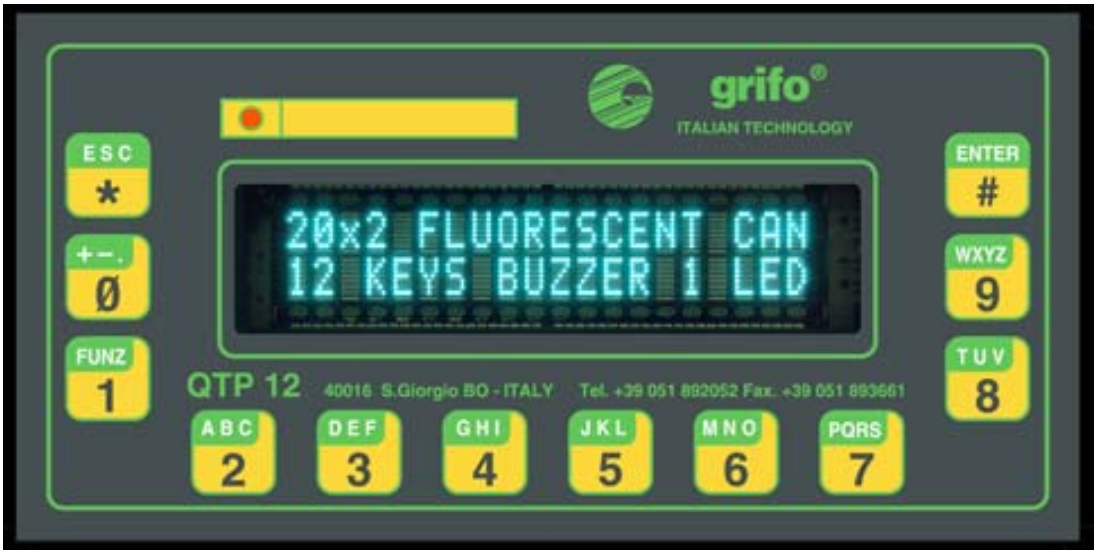


FIGURE 31: QTP 12-F2

FIGURE 32: QTP 12-GF2

POWER SUPPLY

QTP 12 terminal is provided with a power supply section that solves in a efficent and comfortable way the problem to supply the board, in any situation. It generates energy for all sections of the board: control logic, display, backlight, keyboard, LED, serial interfaces, CAN interface, I2C BUS line, real time clock, and buzzer.
Here follow the voltages required from **QTP** according to card configuration together with the relative right connection:

Default version: This configuration includes a switching power supply that requires 10÷38 Vdc or 8÷24 Vac provided through CN1 (polarity must be respected in case of DC supply). This allows to supply the terminal using standard industrial and commercial power sources like transformers, batteries, solar cells, etc. A comfortable and inexpensive solution for default version power supply can be the **EXPS-1** product that can be directly connected to the terminal starting from mains.
Please remind that on board switching section is provided with single diode rectifier, so in case of DC supply, all ground signals of the terminal (GND) are at the same potential. When a single AC source is used to supply different units (both some **QTP 12** or other cards provided of supply section with single diode rectifier), please ensure that the two phases of AC voltage must be connected at the same input pins of power supply connector. Whenever this rule is not satisfied dangerous malfunctions or damages can rise up on all the connected devices. For example, if we call Phase1 and Phase2 the two signals of the AC voltage, then Phase1 must be always connected to positive inputs (Vac, +Vdc pow) and Phase2 must be connected to negative input (Vac, GND). Complete information and details can be found on paragraph CN1 - POWER SUPPLY CONNECTOR.
This is the default version, normally delivered without further specifications, in the order.

.5Vdc or **.ALIM** version: This configuration is not provided of any power supply section, so a +5 Vdc ± 5% stabilized supply voltage must be provided by an external source, through CN1 connector (polarity must be respected also in this case). This allows to provide energy to the terminal through laboratory power supply, other cards, etc.
This version is a particular OEM configuration only, to directly agree upon **grifo®**.

Selection of power supply section must be performed during the order phase, in fact it involves a different hardware configuration that must be made by **grifo®** technicians.
The **QTP 12** is always provided with a **TransZorb™** protection circuit to avoid damages from incorrect voltages and/or break down of power supply section. It is also provided with a distribuited filtering circuitry that saves the terminal from disturbs or noise from the field, improving the overall system performances.
For further information please refer to paragraph ELECTRIC FEATURES.

FIGURE 33: POWER SUPPLY EXPS-1

BACK UP

When **QTP 12** is ordered with the optional real time clock (**.RTC**), it is provided of a lithium battery that keeps the time and the content of SRAM even when power supply is off. The user can cannect or not this battery to back up circuitry, by acting on dedicated jumper J6, as described in figure 26. The card is supplied with the jumper connected to preserve the clock counting and the SRAM content in each operating condition.
Whenever the **QTP 12** is not used for a long time, or the application doesn't need the back up circuit, it is suggested to prevent the battery discharge by removing the jumper J6.
Obviously if the J6 connection must be changed, then the card must be extracted from the metallic container: a simple pressure on **QTP 12** connectors, or on the printed circuit reachable from rear container window, is sufficient. When on the front panel there are two black screws, they must be previously unscrewed (for details see APPENDIX C). When printed circuit is extracted from container, the J6 location can be easily found by using figure 6.

CONTRAST REGULATION TRIMMER

On **QTP 12** board there is a trimmer that defines the contrast on LCD displays. This trimmer, named RV1 or RV2 is set by **grifo®** to obtain the best display visibility in each working conditions and normally the user must not change its position. In case of specific requirements, as external light very low or very high, it can be changed by little rotation in both directions until the visibility is improved. For recognizing the location of contrast regulation trimmer, please refer to figure 6.

NOTE The regulation is possible only with LCD display, installed on **QTP 12-C2**. When **QTP 12-F2** or **QTP 12-GF2**, with fluorescent display, is used the trimmer for contrast regulation has no effects on visibility. The user can change the VFD display visibility only by using the command FLUORESCENT DISPLAY BRIGHTNESS SETTING.

CAN INTERFACE CONNECTION

Jumper J3 connects or does not connect termination resistor of CAN line, as described on figure 26. The CAN BUS must be a differential line with 60 Ω of impedance so termination reistors must be connected to obtain this impedance value. In detail, this connection must be always made in case of point to point communications, while in multi points communications it must be connected only in the cards at the greatest distance, that is at the ends of the CAN line (please see example of figure 13).
The right CAN termination contributes considerably to obtain a correct communication; in fact the **QTP 12** on board interface can suppres transients and avoids radio frequence and electromagnetic noises, only when connection to the filed is correctly made.
CAN line is not galvanically isolated (as descrived in previous paragraph POWER SUPPLY) from on board generated supply voltage. Ground of CAN line is connected to on board GND and it is available on a pin of CN3 connector. This latter can be used to equilibrate difference of potentials amongst several CAN systems, but also to shield physical connection, when shielded cable is used for CAN line, to obtain the greatest protection against external noise.

SOFTWARE DESCRIPTION

As already stated **QTP 12** is a complete video terminal. It shows on the display any characters received from communication line, except the commands that are recognized and executed, and it transmits back, on the same line, the possible results of the executed commands and the codes of keys pressed. In other words it acts a slave dumb terminal controlled by an external command unit that can be placed even at a long distance. These operations are automatically performed by the on board firmware that is programmed and executed from **QTP 12** CPU.
The on board firmware manages also a local setup which allows the user to define some working conditions by using the display and the keyboard of **QTP 12**.
This chapter describes the main features of **QTP 12** software functionalities, while the following one reports a detailed description of the recognized comand sequences, that can be used to benefit of all the potentialities of the terminal.
In corrispondence of the first order, on the received grifo® CD, are supplied many complete and useful demo programs either in executable and source format; these can be used as received with no modifications, for a first test of the product and then changed, or partially used, to develop the user application program.

LOCAL SETUP

Thanks to a proper local setup mode the user can select some parameters of communication protocol, define some working conditions and restore the base EEPROM content. This mode can be easily and intuitively used, thanks to the on board display and four keys of **QTP 12**.
In detail the user must:

a) Press the keys **ESC** ***** and **+-. 0**, simultaneously power on the **QTP 12** and then mantain the keys pressed for at least half of a second.

b) At this point setup mode is entered, on the display appears the "**Local Setup V.x.y**" string and with keys **WXYZ 9** and **ENTER #** the current configuration parameters, and its current values, shall be changed as below described:

c) Press the key **ENTER #** to change current menu, recognized by the following messages:

"**COMMUNIC.**"	to change the communication type eual to logic protocol
"**BAUD RATE**"	to change the communication baud rate
"**STOP BIT**"	to change the stop bit number
"**KEY-CLICK**"	to change the keyclick mode
"**SLAVE ADD.**"	first digit of identification name in hexadecimal
"**SLAVE ADD.**"	second digit of identification name in hexadecimal
"**EE DATA**"	initializes data in EEPROM
"**INTRTC FN.**"	to change function of the digital output managed by RTC
"**SAVE and EXIT**"	to exit from setup mode

d) Press the key **WXYZ 9** to change current value of displayed menu:

COMMUNIC.:	**Norm.**, **I2C**, **M.S.9** that are the 3 communication mode	(*def.=Norm.*)
BAUD RATE:	**38400**, **19200**, **9600**, **4800**, **2400** or **1200** baud	(*def.=19200*)
STOP BIT:	**1** or **2** with Normal protocol	(*def.=1*)
	1 with Master-Slave 9 bits protocol	

KEY-CLICK:	**ON** or **OFF**	*(def.=ON)*
SLAVE ADD.:	Changes the digit enclosed in "><" from **0** to **F**	(*def.=80H*)
EE DATA:	**NOINI** or **INIT**	(*def.=NOINI*)
INTRTC FN.:	**USER** or **ALARM**	(*def.=USER*)
SAVE and EXIT:	exits setup and configures **QTP 12** with selected parameters	

e) Once the necessary values have been set by using the modalities described in points **c** and **d**, select the **SAVE and EXIT** menu and press the **WXYZ 9** key to confirm.

Once exited from setup mode, the selected parameters are saved on EEPROM and they are mantained until another local setup is executed; immediately after the terminal starts its normal functionality. The *default* values (before reported between round brackets) are those set at the end of testing phase, that is the configuration the user receives.
Available options for menus BAUD RATE and STOP BIT define the physical communication protocol that has other two parameters unchangeable and set to no parity and 8 bits per character or 9 bits when Master-Slave 9 bits is selected. Options of remaining menus are described in the following paragraphs.

NOTE: Please remind that local setup mode can be entered only during power on, when previously described conditions are recognized in fact if described keys are pressed at the same time during normal operation then setup mode will not start.
The local setup is normally executed only one time after the first installation, from the customer or installer, that configures the **QTP 12** according with requirements of the developed application. So it regards expert staff but not the final user that handle it as a simple, ready operator interface unit.

KEYBOARD ACQUISITION

When **QTP 12** recognizes a key pressure, it transmits the relative code on communication line. This happens immediately when Normal communcation is selected, while in case of Master-Slave 9 bits or I2C BUS communications, the code is saved in the transmission buffer and then it is sent only upon reception of specific request from command unit, by using the rules described in the following paragraphs.
Moreover an **auto repeat** function of the stroked key is implemented so when **QTP 12** recognizes the pressure on a key, for a time greater than **0.5 sec**. it will start the transmission of its code about each **0.1 sec**. and it lasts until that key is released.
If the **keyclick** function is enabled when the code of the pressed key is transmitted, the on board buzzer also generates a loud beep that sonorously signalize the event to the user. Whenever the buzzer is already enabled or intermittent, then the keyclick disables it for a little time period, to ensure the acoustic event recognition in any circumstance.
When two or more keys are contemporaneously pressed it is transmitted only the code of the key with higher number, or on the other end, the key with number 11 (ENTER #) has the highest priority while the key number 0 (ESC *) has the lower priority.
Another feature provided by **QTP 12** is the complete reconfiguration of the key codes performed by user application program; in other words it is possible to change the code returned when a key is pressed and even disable the key. This features really simplifies the development of management software, in fact the command unit can dinamically change the keyboard functionality.

KEYS CODES

Below there is a figure that shows the keys numbers and locations and another figure with the default codes, that **QTP 12** terminal returns on communication line, when a key is pressed. As for the command sequences the code are shown in deciml, hexadecimal and ASCII mnemonic format, through the standard ASCII symbol table:

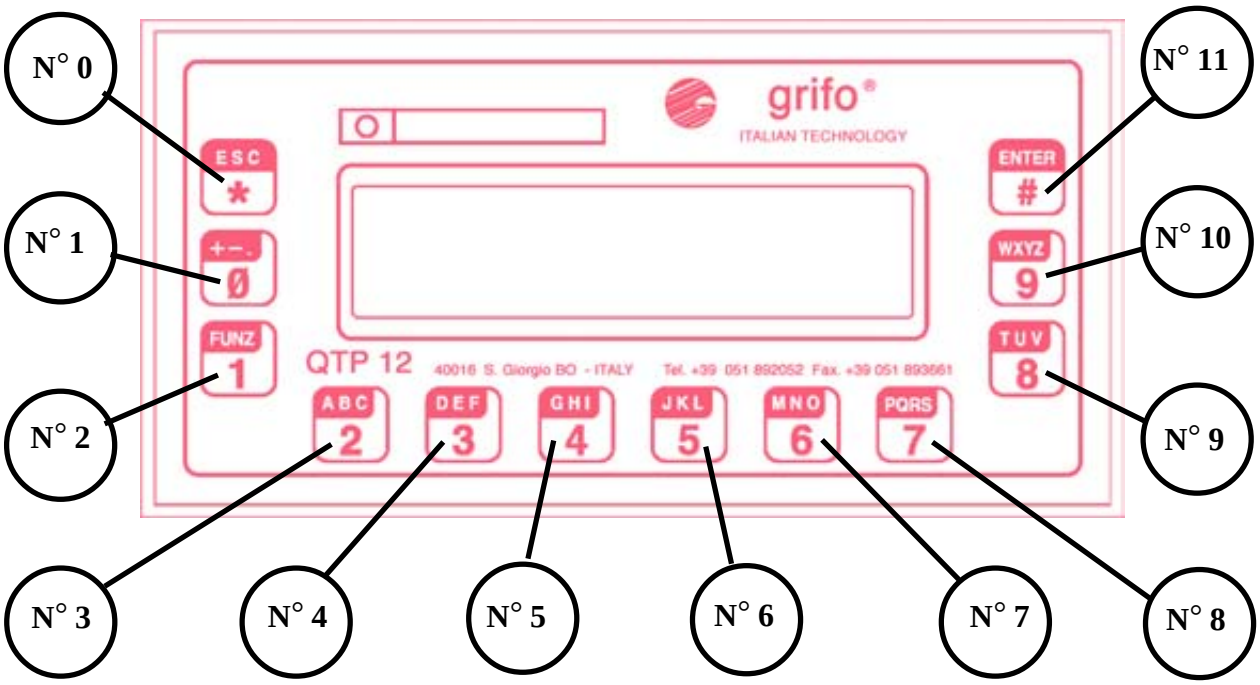


FIGURA 34: KEYS NUBERS AND LOCATION

The keys numbers on figure 34 are unchangeable and they are necessary for the command KEY CODE RECONFIGURATION, described in following chapter, to univocally identify all the keys. On the other end, the keys codes are programmable and they can be used by command unit to recognizes the keys pressed. The table of figure 35 lists the codes in default configuration, that is the one the user receive after an order and reparation, or after an EEPROM inizialization.

KEY NUMBER	*SERIGRAPHY*	*CODE*	*HEX CODE*	*MNEMONIC*
0	ESC *	42	2A	*
1	+-. 0	48	30	0
2	FUNZ 1	49	31	1
3	ABC 2	50	32	2
4	DEF 3	51	33	3
5	GHI 4	52	34	4
6	JKL 5	53	35	5
7	MNO 6	54	36	6
8	PQRS 7	55	37	7
9	TUV 8	56	38	8
10	WXYZ 9	57	39	9
11	ENTER #	35	23	#

FIGURE 35: DEFAULT KEYS CODES

COMMUNICATION BUFFERS

QTP 12 is provided of two communication buffers that simplify the management and increase its flexibility, in fact they reduce the waiting time of the connected command unit.
The first is a receive buffer: it is **40 bytes** long, it memorizes each character received from command unit and then it is examined at the end of the currently executed operation. Naturally when commands that requires a long execution time (delete commands, EEPROMs management commands, messages shift, etc.) are continuosly received, the buffer can become full and it will overflow. When overflow occours the first locations of the buffer are overwritten by each next received characters, and they are defenitely lost. The command unit must stop the transmission until the **QTP 12** has emptied the receive buffer and it is still ready to receive other data. In practice the user must insert suitable delays in communication, experimentally calibrated, to avoid overflow of the buffer.
The second is a transmit buffer: it is **20 bytes** long, it memorizes each character that must be sent to command unit and it is filled with the keys pressed codes and the executed commands response. When Normal communication is selected the transmit buffer is not used infact data are immediately transmitted, viceversa when I2C BUS or Master-Slave 9 bits communications are used the data remain in transmit buffer until the command unit requires them. If the command unit doesn't receive data from **QTP 12**, this buffer become full. When this filling occours all following data are no more saved in the transmit buffer, and these are defenitely lost. So the command unit must manage data reception from **QTP 12** at least in two situations: before to send commands provided of responses (to empty the buffer for the same response) and periodically (to get the possible keys pressed).

DATA STORED ON EEPROM

The base EEPROM of **QTP 12** stores a set of data used and/or changed through the specific commands and local setup. The choice of EEPROM memory type has been performed to obtain the best warranties on data validity and endurance, naturally even when power supply is not available. The detailed description on each one of the data saved on EEPROM is reported in the following chapter, in the paragraphs relative to commands that directly use them.
With menu EE DATA of local setup the user can select to leave unchanged these data (NOINI option) or to set them at their default values (INIT option) that is the configuration received after an order or a reparation. In details, by selecting the INIT option, the base EEPROM data will have the following values:

presence byte	->	255 (FFH)
keys codes	->	those described in table of figure ??
power on visualization	->	none
patterns of user defineable characters	->	255 (FFH)
messages	->	255 (FFH)
user EEPROM bytes	->	255 (FFH)

Once exit from local setup a string is shown on the display together with a scrolling bar of ***** (asterisk) that inform about the status progress of the operation. The displayed ***** are 10 and the execution time of the described initialization phase is 20 seconds approximately.
Please remind that the INIT option will initialize only the base EEPROM, while the optional EEPROM mantain its original contents. With this option of all the available messages, only the first 95 are deleted.
The user must be very careful with EEPROM initialization, in fact all data previously saved are defenitely lost.

CHARACTERS VISUALIZATION ON DISPLAY

QTP 12 shows on its display all the received characters having a code included in the range **0÷255** (**00÷FF Hex**) including the one that identifies a command sequence (27 = 1BH), as described later. The character is visualized on the current cursor position and this latter will go to the next position; if it is placed on the last character of the display (right down corner), it will be placed on Home position (left up corner).
The corrispondence between codes and displayed characters is defined by the following rules:

Codes	Characters
0 ÷ 15 (00÷0F Hex)	User defineable
16 ÷ 31 (10÷1F Hex)	Special and different, according with installed display
32 ÷ 127 (20÷7F Hex)	Standard ASCII
128 ÷ 255 (80÷FF Hex)	Special and different, according with installed display

To allow visualization of special characters, that have same codes of the one character commands, a specific command has been provided that selects the operating mode of **QTP 12** among the two available:

command the characters are not displayed and the relative commands are executed;
representation the characters are always displayed.

After a power on it is automatically selected the command mode to make immediately utilizable every functionalities. The commands composed by a sequence of two or more characters, that always start with ESC = 27 = 1BH, are anyhow interpreted and executed independently from the selected operating mode.
Every models of **QTP 12** has 8 user characters that can be defined and/or stored according to application requirements, and then shown on the display, as explained in the further paragraph COMMANDS FOR USER CHARACTERS.
About special characters please refer to APPENDIX B and remind that it is possible to get also different display models, provided of different special characters, but everything must be directly prearranged with **grifo®**.

FIGURE 36: CHARACTERS AVAILABLE ON QTP 12-GF2

COMMUNICATION MODALITIES

QTP 12 supports three different serial communication modalities:

Norm. Normal communication uses the asynchronous serial line on CN2 and it supports 8 bits per character, no parity plus stop bit and baud rate selected by user, through local setup. This communication mode is suitable for point to point connections in RS 232, RS 422 and Current Loop.
For detailed information about this modality please read proper paragraph NORMAL COMMUNICATION.

I2C I2C BUS communication uses the synchronous serial line on CN6 and it supports a bit rate from 500 to 15000 bits per second, as slave (either receive or transmit), with a 7 bits Slave Address selected by user, through local setup. This communication mode is suitable for point to point or network connections.
For detailed information about this modality please read proper paragraph I2C BUS COMMUNICATION.

M.S.9 Master-Slave 9 bits communication uses the asynchronous serial line on CN2 and it supports 9 bits per character, no parity, one stop bit plus baud rate selected by user, through local setup. This communication mode is suitable for point to point connections (with all electric protocols) or network (with RS 485, RS 422 and Current Loop electric protocols).
For detailed information about this modality please read proper paragraph MASTER-SLAVE 9 BITS COMMUNICATION.

Local setup allows to select communication modality, as described in the specific paragraph, while electric protocol must be defined when the terminal is ordered or changed as described in SERIAL LINE CONFIGURATION paragraph.

MASTER-SLAVE 9 BITS COMMUNICATION

The Master-Slave 9 bits mode uses a particular communication technique; in addition to the 8 data bits also a nineth bit is managed and it discriminates between a call coming from the "**master**" device to any of the "**slave**" units, and a normal transmission between master and the currently selected slave. When 9th bit is placed at 1, the 8 data bits of the same character has to contain the identification address, of the device required for communication, while by placing this bit at 0, it is possible to take out or supply info at the selected device.
When **QTP 12** is used, the identification address must be that one selected through the local setup program, on the "SLAVE ADD." menus.
When this byte is sent (with 9th bit set to 1) the **QTP 12** recognizes itself and it waits the string containing chars, data or commands. In this string there could be only a command that involves the return of a response, to send via serial line from **QTP** part; if there is more than one command with response, the results of the remaining ones are ignored.
Between the transmission of a character and the next one there must be a time interval shorter than **Time Out**, in fact when this time period is elapsed , the **QTP 12** will consider the command sequence terminated and it will begin the answering phase. The Time Out values for each baud rates are below described:

Baud Rate	Time Out	Character transmission time
38400 Baud	550 µsec	287 µsec
19200 Baud	990 µsec	573 µsec
9600 Baud	1540 µsec	1146 µsec
4800 Baud	3080 µsec	2292 µsec
2400 Baud	6105 µsec	4584 µsec
1200 Baud	12100 µsec	9167 µsec

Master unit, once completed the transmission of the last character of the command sequence, must wait for a time equal to:

Character transission time + Time Out

before to receive the first character of the response string, transmitted by the **QTP 12**. The answer consists in a character containing the possible code of key pressed (**255** = **FF Hex** means no keys pressed), or a characters sequence that coincide with the response of the command sent in the previous interrogation. Please remind that response is provided also when master unit transmit a command sequence with only the identification name: this simplifies the check for available keys pressed or invalid command.

Several demo programs, coded in different programming languages, are provided with **QTP 12**. They implement Master-Slave 9 bits communication and can be used directly by the user or modified according to the specific needs.
When the master unit is a PC, the user can also take advantage of comfortable **DLL** libraries that allow to manage Master-Slave 9 bits communication at high level, this means without having to worry about nineth bit, timings, possible electric protocol converters, etc. Also these libraries are provided with the first purchase, complete of user documentation, on a floppy disk or a CD rom.

NOTES:

1) To ensure right commands execution, between a call and the next one it is necessary to wait for a time that is proportional to the number of commands sent, and type of operations they involve.

2) If the master unit doesn't support 9 bits communication, it is possible to simulate this bit by using the parity bit and programming its value properly, before any characters tranmission, according to this scheme:

If the character to transmit has even number of "1" bits

*If 9th bit must be **1*** -> *Program parity to ODD*

*If 9th bit must be **0*** -> *Program parity to EVEN*

If the character to transmit has odd number of "1" bits

*If 9th bit must be **1*** -> *Program parity to EVEN*

*If 9th bit must be **0*** -> *Program parity to ODD*

3) When automatic visualizations on display are enabled (scrolling messages, date and time visualization, etc.) the time between two calls, in addition to the time indicated at point **1**, must be about **12000 µsec**.

4) In a single communication between master unit and **QTP 12** can be transmitted many characters to show and some commands to execute, taking care to doesn't fill the receive buffer, as described in COMMUNICATION BUFFER paragraph.

The following flow chart shows all the described features:

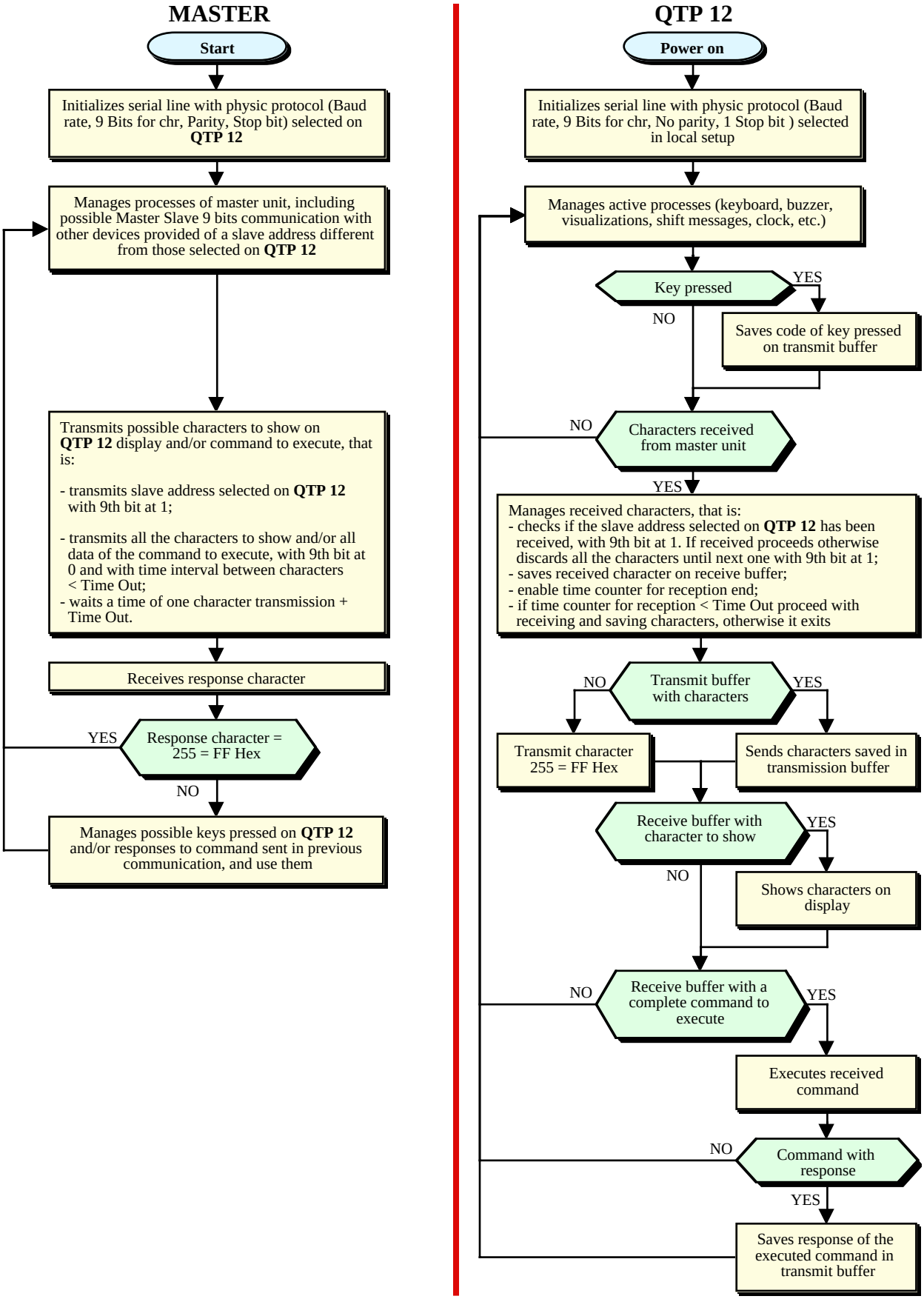


FIGURE 37: FLOW CHART FOR MASTER-SALVE 9 BITS COMMUNICATION

To explain better the Master-Slave 9 bits protocol, here follows an example where master unit sends three commands to **QTP 12** (reading of version number, a string to show on display and a check for possible keys pressed) with a 38.4K baud rate and identification address (SLAVE ADD.) set to 80H value:

Master	*QTP 12*
Sends "Reading of version number" command, that is the characters sequence: 80H with nineth bit set to 1 1BH with nineth bit set to 0 56H with nineth bit set to 0 with a delay between the characters lower than 550 µsec	Receives characters of the command and verifies the end with a 550 µsec Time out
Waits for 837 µsec	Recognizes command sequence, executes the command and stores response for next interrogation
Receives one character response	Sends the response, which is the code 255=data not available, with nineth bit set to 0
Sends a string to show on the display, that is the characters sequence: 80H with nineth bit set to 1 1st character of string with nineth bit set to 0 2nd character of string with nineth bit set to 0 : : : : : : : : : with a delay between the characters lower than 550 µsec	Receives characters of the command and verifies the end with a 550 µsec Time Out
Waits for 837 µsec	Recognizes command sequence and shows on the display the characters of the string
Receives three response characters with the version number previously requested	Transmits saved response which is the version number required by previous command, with nineth bit set 0
Sends check command for answer data and/or keys pressed, that is the characters sequence: 80H with nineth bit set to 1	Receives characters of the command and verifies the end with a 550 µsec Time Out
Waits for 837 µsec	Recognizes sequence without commands so performs no operation
Receives one or more characters corresponding to codes of possible keys pressed	Sends the response, which is the code 255 or possible key pressed code, with nineth bit set to 0

FIGURE 38: EXAMPLE OF MASTER-SLAVE 9 BITS COMMUNICATION

I2C BUS COMMUNICATION

The system that communicates with **QTP 12** in this modality must operate as master, either in transmit and receive mode, following the rules of I2C BUS standard protocol detailed descripted in the document "*THE I2C-BUS SPECIFICATIONS*", from PHILIPS semiconductors.
This modality requires a syncronization between the systems in communication, as illustrated in the following flow charts:

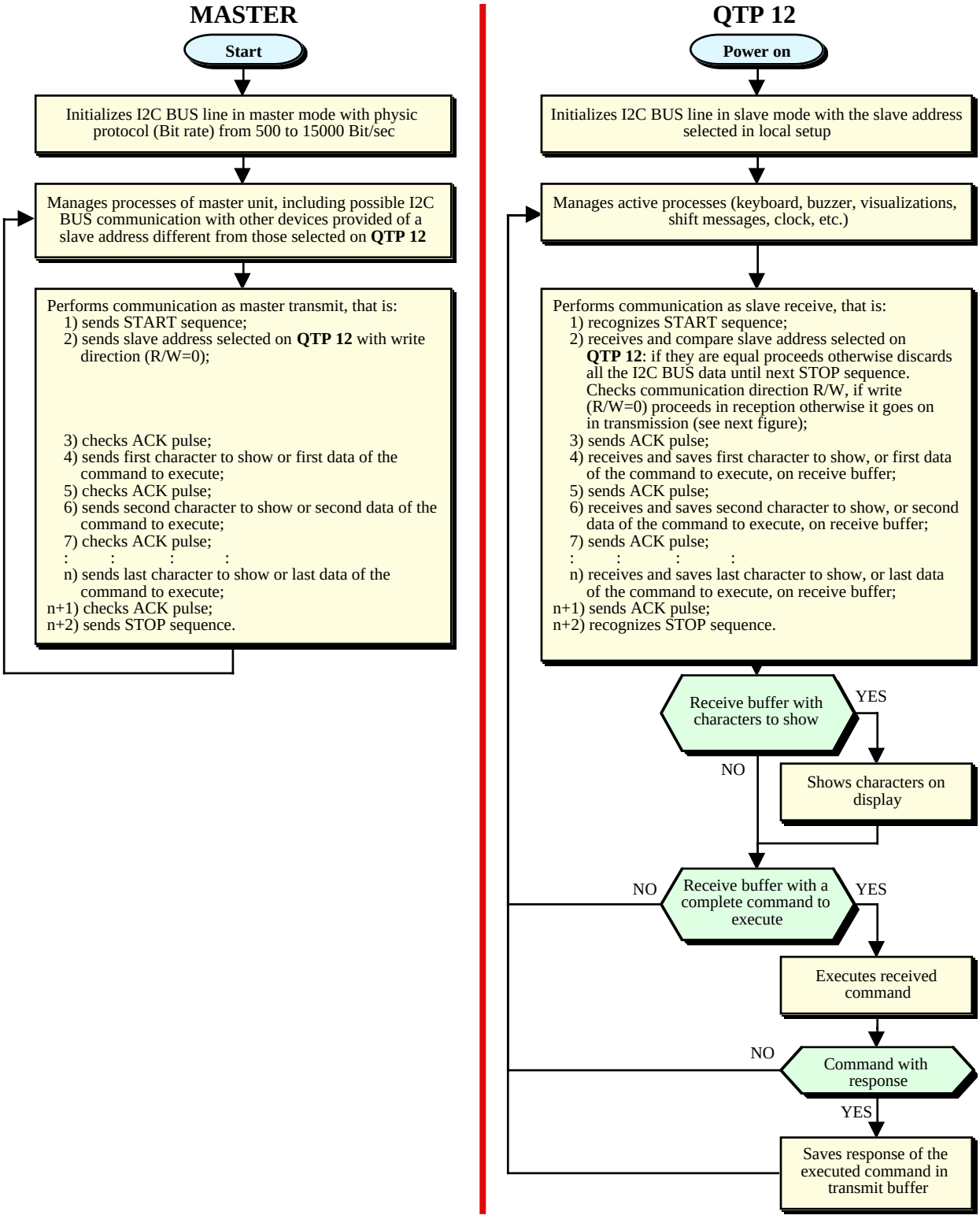


FIGURE 39: FLOW CHART FOR MASTER -> QTP 12 COMMUNICATION IN I2C BUS

The master must perform a communication with write data direction to supply the characters to visualize and/or the command sequences to execute, and perform a communication with read data direction to get the possible codes of keys pressed and/or the possible answers to the supplied commands.
Each communication involves only the **QTP 12** with the slave address equal to those defined in local setup of the terminal, inside "SLAVE ADD." menus. When an I2C BUS network is used, each **QTP 12** must be set with a different slave address, and different from the slave addresses of the other possible I2C BUS devices connected to same network.
In order to simplify the complete management, the first data returned by **QTP 12** after a read communication, always coincides with the number of characters available in the transmission buffer, that is the number of data the master must receive. Thus the muster unit could terminate the communication with proper STOP sequence, only when it has received all these data.

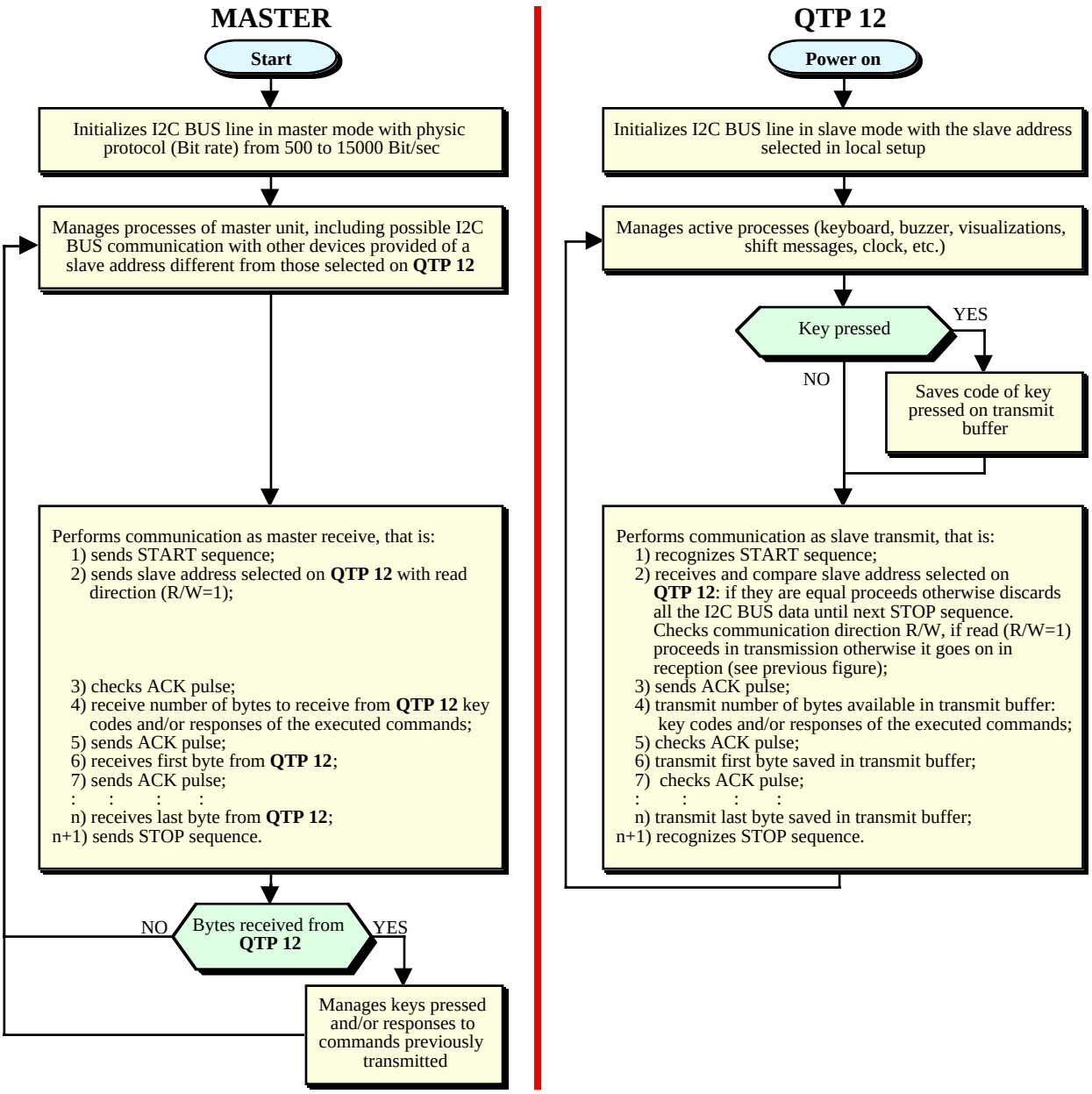


FIGURE 40: FLOW CHART FOR QTP 12 -> MASTER COMMUNICATION IN I2C BUS

NOTES:

1) To ensure right commands execution, between a communication and the next one it is necessary to wait for a time that is proportional to the number of commands sent and type of operations they involve.

2) When automatic visualizations on display are enabled (scrolling messages, date and time visualization, etc.) the time between two calls, in addition to the time indicated at point **1**, must be about **12000 μsec**.

3) During a communication from master unit to **QTP 12** it can be transferred many characters to visualize and command to execute, taking care to doesn't overflow the receive buffer, as described in paragraph COMMUNICATION BUFFERS.

4) The communications from **QTP 12** to master unit must be planed to doesn't overflow the transmit buffer, as described in paragraph COMMUNICATION BUFFERS.

5) The slave address defined in local setup is 7 bits wide but it is managed as 8 bits value, with the least significant bit (R/W) fixed to 0; so 128 different even values can be selected, in the range 00÷FEH. Moreover when the options .EExxx and .RTC are installed the addresses **160** (**A0 Hex**) and **162** (**A2 Hex**) can't be used.

6) When an I2C BUS network connection is used, performs all the configurations described in CN6 - I2C BUS LINE CONNECTOR paragraph, in order to ensure that the line is correctly terminated, from the electric point of view (see figure 16).

7) The **QTP 12** doesn't support the enhancements of I2C BUS protocol (as 10 bits addressing, fast mode, high speed mode, etc.) and the reserved slave addresses: these features can't be used by master unit.

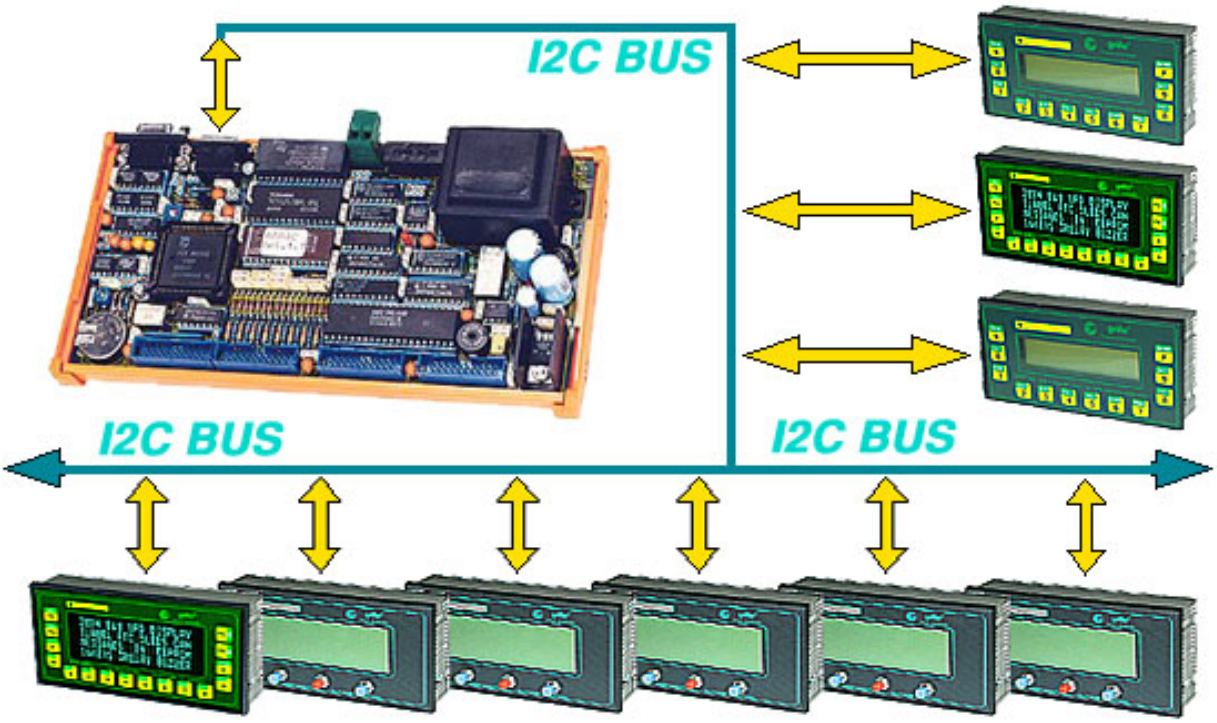


FIGURE 41: I2C BUS NETWORK CONNECTION

NORMAL COMMUNICATION

The system that communicates with **QTP 12** (defined master) in this mode must only transmit the characters to visualize and/or the command sequences to execute, and manage the reception of characters that are the codes of the possible key pressed and/or the possible responses to the supplied commands. This mode doesn't require any syncronization between the two systems in communication and each events is immediately processed from **QTP 12**, as illustrated in the following flow chart:

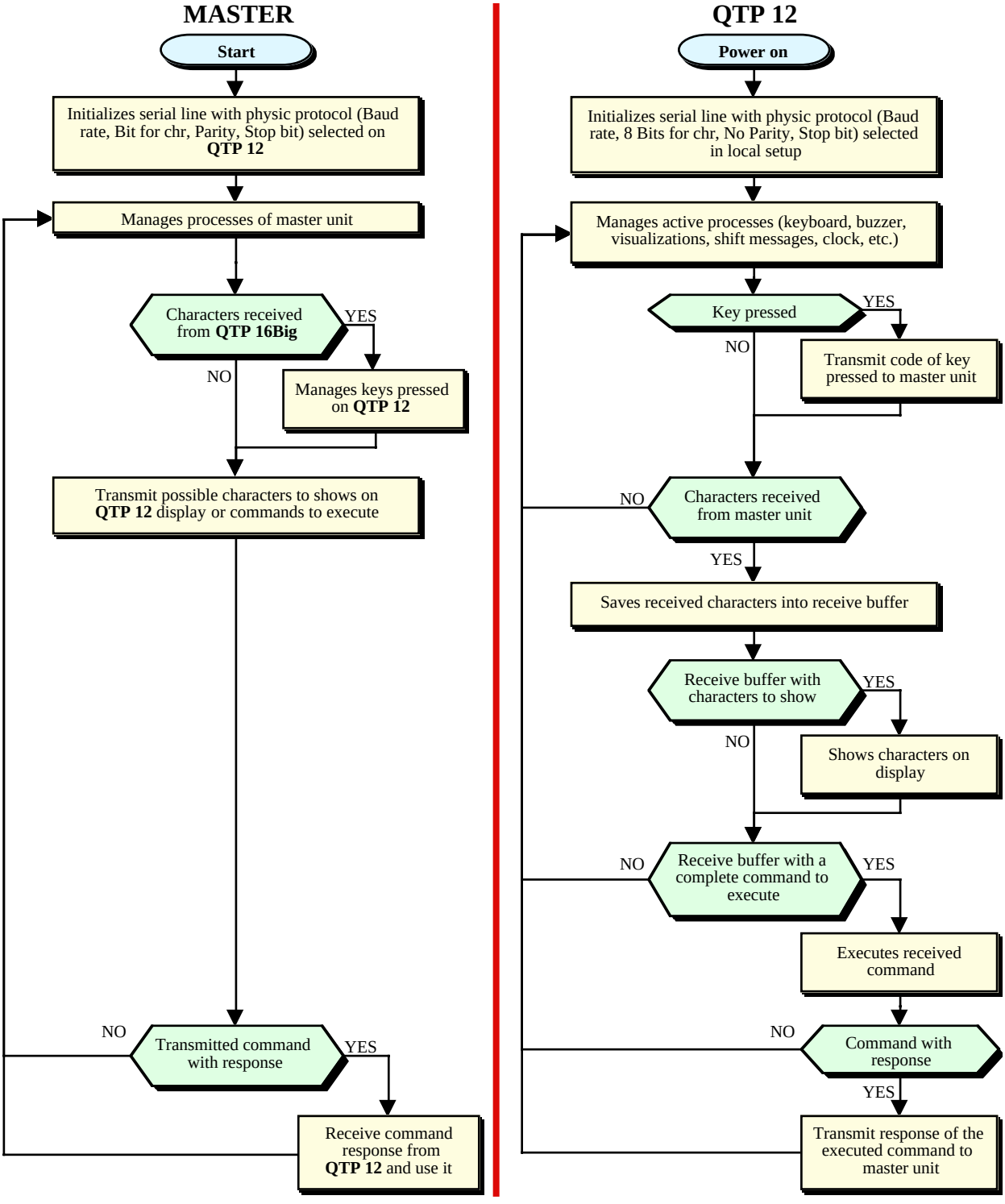


FIGURE 42: FLOW CHART FOR NORMAL COMNICATION

HOW TO START

In this paragraph are listed the operations that must be performed to start using the **QTP 12** in a pratical and fast way, solving the typical beginners problems. The paragraph contains interesting information even for the users that already know the product and its operating modes, in fact there is the description of a fast functional test. The following steps assume that the command unit is a Personal Computer (provided of one free RS 232 serial line and a generic operating system, up to Windows 98), to allow any user to execute them correctly.

A) *Establish connections:*

A1) Perform the serial connection described in figure 43 or on the other hand connect the two communication signals (TX RS232, RX RS232) and the reference ground signal (GND), to free COMx serial port of the PC. It can be easily discovered that this connection cable is reversed and it can be conveniently orderd to **grifo®**, with the code **CCR 9+9R**.

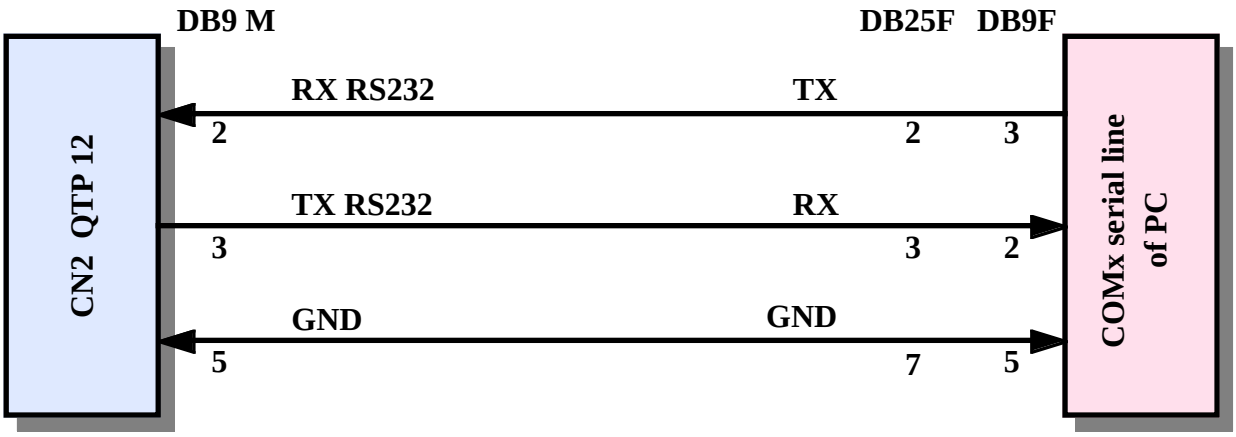


FIGURE 43: RS 232 CONNECTION WITH PC

A2) Supply power voltage on CN1 and check that buzzer is immediately disabled and a blinking block cursor is displayed in the left up corner of the display.

B) *Use of demo program:*

B1) On the floppy disks or **grifo®** CD rom received with the first purchase, it is available the file PRQTP12.EXE, that is a demo program for PC that comunicate through RS 232 line with **QTP**. This file once found, must be copied in a comfortable folder on the hard disk of the used PC.

B2) Execute the program copied at point B1 and compile its start questions, by selecting the mounted display type and the possible mounted options. At this point press a key on PC to continue without execute the local setup, in fact the shown configuration coincides with the default one already set on the received **QTP 12**.

B3) Carry on demo program execution and check that the operations described on PC monitor are correctly executed on **QTP**; when required interact with the same program in order to test all the available commands, until the end of demo program is reached.

C) *Use of terminal emulation:*

C1) Found the HYPERTERMINAL communication program on the PC, that normally is located on Windows menu: "Start | Program | Accessories | Communication", and execute it.

C2) Through the HYPERTERMINAL properties windows, define the communication parameters to:

Connect	directly to COM x (those used at point A1)
Bit rate	19200
Data Bits	8
Parity	No
Stop Bit	1
Flow control	None

and wait the presentation of communication window.

C3) At this point type something on PC keyboard and check that pressed keys are shown on **QTP 12** display. For compltеteness it can be tested also the effects of some commands by typing their codes sequences always on PC keyboard (this operation is simplified by contemporaneous pression of ALT key and of digits of the decimal code, on the numeric pad: for example to transmit the clear page command with decimal code 12, you can press contemporaneously the ALT key and first the keys 1 and then 2).

C4) Press the keys of **QTP 12** and check that they are correctly shown on PC monitor, with the codes corrispondence reported on figure 35.

When during execution of the steps above described a problem or a malfunction is found, we suggest to read and repeat again all the steps carefully and then ensure that default configuration values are saved, through local setup. If malfunction persists please contact directly **grifo®** technician.

DEMO PROGRAMS

In correspondence of the first purchase together with **QTP 12** it is supplied a floppy disk or a **grifo®** CD where are saved numerous demo programs that allow to test and estimate immediately the received product. These programs are provided both in executable and source format and they are coded with many high level programming languages (C, PASCAL, BASIC, etc.) either for PC platforms or **grifo®** microprocessor cards (as **GPC®**, Mini Modules, etc).
As described in HOW TO START paragraph the programs named PRQTP12.* use all the commands of **QTP** through a simple iteraction with the user; but many other demos are supplied capable, for example, to: drive **QTP** connected to a serial network, manage messages, use the Master-Slave 9 bits protocol with DLL libraries, perform scenographic presentation, communicate with I2C BUS line, etc. The user can examine the remarks of these demos and decide himself if they are interesting.
All the demo programs can be used directly or modified or partially used, according to applications requirements, without any autorizathion, license or additional cost. Furthermore in case of unusual requirements or combinations, specific new demo programs can be obtained, after proper agreenment with **grifo®**.

COMMANDS

This chapter describes all the commands available in **QTP16Big** firmwares and each relative input and output parameters. The commands are divided in subgroups according with their functions and for each code, or codes sequence, there is a double description: the mnemonic one through the ASCII characters and the numeric one under decimal and hexadecimal format.
The commands respect the **ADDS Viewpoint** standard so all the sequences begin with **ESC** character corresponding to the **27** decimal code (**1B Hex**).
A rich list of demo programs (supplied in source and executable format) shows the pratical use modalities of commands: we suggest to add these demo programs, received during first purchase on CD or floppy disk, to this chapter documentation.
A summarized descriptions of all the available commands, their parameters and possible responses, are reported in the tables of APPENDIX A.

COMMANDS FOR CURSOR POSITION

Here follows the list of the commands that acts on cursor position.

CURSOR LEFT

Code: ***21***
Hex code: ***15***
Mnemonic: ***NACK***

The cursor is shifted of one position to the left without modifying the display contents. If the cursor is in Home position, it will be placed in the last position of the last row of the display (down, right corner).

CURSOR RIGHT

Code: ***6***
Hex code: ***6***
Mnemonic: ***ACK***

The cursor is shifted of one position to the right. If the cursor is placed in the last position of the last row, il will be moved to the Home position that is the first position in the first row (up, left corner).

CURSOR DOWN

Code: ***10***
Hex code: ***A***
Mnemonic: ***LF***

The cursor will be moved to the line below but it will remain in the same column. If the cursor is in the last display line, it will be moved to the first display line.

CURSOR UP

Code: ***26***
Hex code: ***1A***
Mnemonic: ***SUB***

The cursor will be moved to the line above but it will remain in the same column. If the cursor is in the first display line, it will be moved to the last display line.

HOME

Code: ***1***
Hex code: ***1***
Mnemonic: ***SOH***

The cursor is moved to Home position that is the first line, first column of the display, or on the other hand the up, left corner.

CARRIAGE RETURN

Code: ***13***
Hex code: ***D***
Mnemonic: ***CR***

The cursor is moved to the beginning of the line where it was located.

CARRIAGE RETURN+LINE FEED

Code: ***29***
Hex code: ***1D***
Mnemonic: ***GS***

The cursor is moved to the beginning of line below the one where it was located. If the cursor is at the last display line, it will be moved to the beginning of the first line, i.e Home position.

ABSOLUTE PLACEMENT OF ALPHANUMERIC CURSOR

Code: ***27 89 r c***
Hex code: ***1B 59 r c***
Mnemonic: ***ESC Y ASCII(r) ASCII(c)***

The cursor is moved to the absolute position indicated by **r** and **c** parameters.
These characters are the row and column values of the new desidered position referred to coordinate 0, 0 of the Home position, plus a constant offset of **32** (**20 Hex**). The position is expressed in alphanumeric mode so their valid values ranges respectively are 32÷33 and 32÷51. When row and/or column values are not compatible with the specified ranges, the command is ignored.
If, for example, the user wants to place the cursor on the second line, third column (row 1, column 2), then the following sequence must be sent:

27 89 33 34 or **1B 59 21 22 Hex** or **ESC Y ! "**

COMMANDS FOR CHARACTERS ERASURE

Below are described all the commands that deletes one or more characters from the display.

BACKSPACE

Code: ***8***
Hex code: ***8***
Mnemonic: ***BS***

This command moves the cursor one character to the left and it erase the contents of the reached position.
If the cursor is in Home position, it will be erased the last character of the last row of the display.

CLEAR PAGE

Code: ***12***
Hex code: ***C***
Mnemonic: ***FF***

This command clears all data on the display and it moves the cursor to Home position.

CLEAR LINE

Code: ***25***
Hex code: ***19***
Mnemonic: ***EM***

This command erases all characters displayed on the current line and it moves the cursor to the first column of the same line.

CLEAR END OF LINE

Code: ***27 75***
Hex code: ***1B 4B***
Mnemonic: ***ESC K***

This command erases all characters displayed from the current cursor position to the end of line inclusive. The cursor mantains the previous position.
If, for example, the cursor is at the beginning of a display line, the complete line will be erased.

CLEAR END OF PAGE

Code: ***27 107***
Hex code: ***1B 6B***
Mnemonic: ***ESC k***

This command erases all characters displayed from the current cursor position to the end of display inclusive. The cursor mantains the previous position.
If, for example, the cursor is at Home position, the complete display will be erased.

COMMANDS FOR CURSOR ATTRIBUTES MANAGEMENT

Below are listed the commands that define the possible cursor attributes and styles.

CURSOR OFF

Code: ***27 80***
Hex code: ***1B 50***
Mnemonic: ***ESC P***

The cursor is disabled and it is not more visible.

STEADY STATIC CURSOR ON

Code: ***27 79***
Hex code: ***1B 4F***
Mnemonic: ***ESC O***

The cursor is enabled and so it is visible as a not blinking line placed under the character displayed on the current cursor position.

NOTE: This command can't be used on **QTP 12-GF2** model provided of graphic display: in this condition the command has no effect.

BLINKING BLOCK CURSOR ON

Code: ***27 81***
Hex code: ***1B 51***
Mnemonic: ***ESC Q***

The cursor is enabled and it is visible as a blinking rectangular block that is alternatively visualized with the character displayed on the current cursor position.

COMMANDS FOR GENERAL FUNCTIONS

In the following paragraphs are described all the general purpose commands that manage some features of **QTP 12** firmwares. These commands do not come into the other subgroups and for this reason they are described in this specific paragraph.

READ FIRMWARE VERSION

Code: ***27 86***
Hex code: ***1B 56***
Mnemonic: ***ESC V***

The command returns a string of 3 characters containing the management firmware version that is resident and executed by **QTP 12**.
For example with firmware version 2.1 the following characters will be returned:

50 46 49 or **32 2E 31 Hex** or **2.1**

READ CARD CODE

Code: ***27 160***
Hex code: ***1B A0***
Mnemonic: ***ESC ASCII(160)***

The firmware returns the card code that in case of **QTP 12** coincides with value **9** (**09 Hex**). This command is useful especially when on the same serial network there are many different devices and the command unit must recognize them. Naturally the card code identifies the product in a univocal manner.

FLUORESCENT DISPLAY BRIGHTNESS SETTING

Code: ***27 108 lum***
Hex code: ***1B 6C lum***
Mnemonic: ***ESC l ASCII(lum)***

Sets fluorescent display brightness to one of the four possible values, passed in **lum** parameter, with the following corrispondence:

0	*(00 Hex)*	->	*Brightness at 100%*
1	*(01 Hex)*	->	*Brightness at 75%*
2	*(02 Hex)*	->	*Brightness at 50%*
3	*(03 Hex)*	->	*Brightness at 25%*

If brightness parameter is not valid, command is ignored.

NOTE This command is available only with models **QTP 12-F2** and **QTP 12-GF2**, provided of fluorescent display. In case of **QTP 12-C2** with LCD display, command must not be sent because it produces the visualization of an undesired character and a shift in all the next received data (alternatively see the CONTRAST REGULATION TRIMMER paragraph).

OPERATING MODE SELECTION

Code: ***27 65 mode***
Hex code: ***1B 41 mode***
Mnemonic: ***ESC A ASCII(mode)***

It defines the operating mode for the first special characters (those provided of code less than 32 = 20H) and the single character commands. The selected modality is defined by value of **mode** parameter, with the following correspondence:

0 (00 Hex) -> Command mode
255 (FF Hex) -> Representation mode

If **mode** value is not one of the above described, the command is ignored. Further information about operating mode are reported in CHARACTERS VISUALIZATION ON DISPLAY paragraph.

COMMUNICATION RESET

Code: ***27 163***
Hex code: ***1B A3***
Mnemonic: ***ESC ASCII(163)***

Tnis command reinitializes communication, with no modifications on the other process in execution. Naturally communication is referenced to data exchange between **QTP 12** and command unit, so it is indipendent from used veichle. In detail the command performs the following operations:

- clears receive buffer;
- eliminates possible characters of response still to return, from transmit buffer;
- eliminates possible pressed keys still to return;
- terminates the managements of all the commands under execution that redirect the supplied characters (message storage, I2C BUS communication as master, etc.).

GENERAL RESET

Code: ***27 162***
Hex code: ***1B A2***
Mnemonic: ***ESC ASCII(162)***

Tnis command performs a general reset of **QTP 12** and it set again an initial condition similar to those available after a power on. In detail the command performs the following operations:

- resets communication as described in previous command;
- clears display and stops possible scrolling messages;
- disables indicator LED, buzzer and possible intermittent attributes;
- disables and deactivates the optional clock alarm;
- disables the digital output;
- reloads the setting saved on EEPROM that are keyclick mode, key codes, user characters patterns, identification slave address, communication protocol, etc.

NOTE: This command uses the on board EEPROM, so before executing it is better to check the EEPROM ready status through the proper command. The execution time of this command is some tens of msec.: when after this command many other commands must be sent, it is preferible insert a suitable delay that avoid receive buffer overflow.

BEEP

Code: ***7***
Hex code: ***7***
Menomonic: ***BEL***

The buzzer is enabled for a time of 0.1 second. If buzzer was already enable then it is disabled for the same time period, so the audible effect of this command is always recognizable.

BUZZER, LED ACTIVATION

Code: ***27 50 dev attr***
Hex code: ***1B 32 dev attr***
Mnemonic: ***ESC 2 ASCII(dev) ASCII(attr)***

The on board device selected by **dev** parameter is driven using attribute specified in **attr** parameter. In details the devices have the following correspondence:

0 (00 Hex) -> Indicator LED
255 (FF Hex) -> Buzzer

while the attribute can assume the following values:

0 (00 Hex) -> device OFF
255 (FF Hex) -> device ON
85 (55 Hex) -> device intermittent

If parameters are not valid, command is ignored.
The intermittent function is completely autonomous and it doesn't requires any intervent from user side.
The buzzer and the indicator LED are always disabled after power on but they can be changed by command unit to recall operator attention, to signalize an allarm, to show a predefined status, etc.
For example, to activate the LED with intermittent attribute, the following sequence must be sent:

27 50 0 85 or **1B 32 00 55 Hex** or **ESC 2 NUL U**

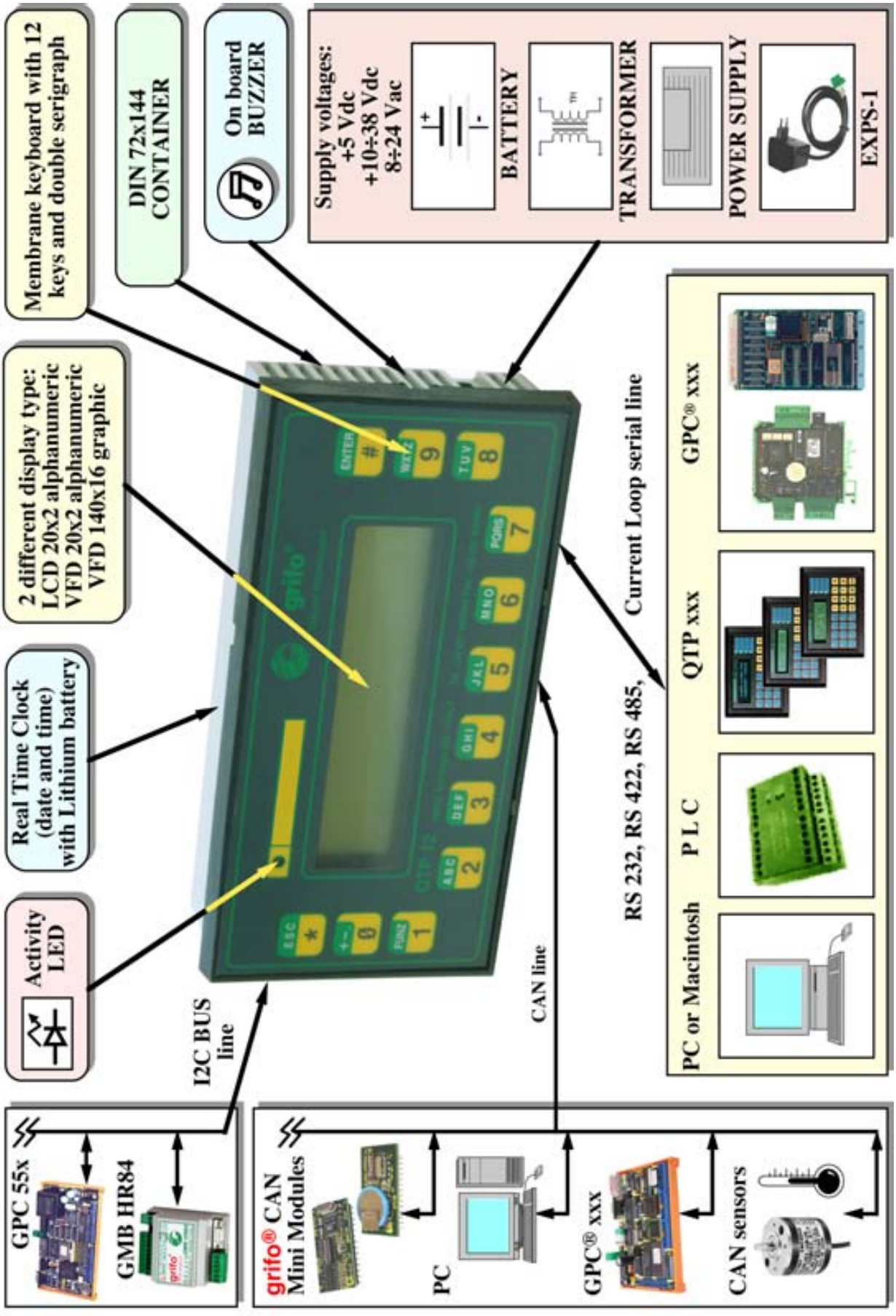


FIGURE 44: AVAILABLE CONNECTIONS DIAGRAM

COMMANDS FOR EEPROM

In this paragraph are described some of the commands that directly manages the data saved on EEPROM/s of **QTP 12**; there are other commands that indirectly use this memory devices but they are described in next paragraphs.

REQUEST FOR EEPROM AVAILABILITY

Code: ***27 51***
Hex code: ***1B 33***
Mnemonic: ***ESC 3***

This command checks if the **QTP 12** is ready for management of its on board EEPROM/s. This command must be executed any time there are data to be read or write on this type of memories.
When **QTP 12** receives this command, it returns one of the following codes:

6	*(06 Hex)*	*(ACK)*	*->*	***QTP 12** ready*
21	*(15 Hex)*	*(NACK)*	*->*	***QTP 12** not ready*

If firmware sends back the NACK code, it is not yet possible to memorize a new data on EEPROM or get an already saved one.

WRITE OF PRESENCE BYTE

Code: ***27 33 78 byte***
Hex code: ***1B 21 4E byte***
Mnemonic: ***ESC ! N ASCII(byte)***

This command sets the card presence byte with the value indicated in the **byte** parameter that must be included in **0÷255** (**0÷FF Hex**) range.
This byte has a reserved allocation on the on board base EEPROM that, once it is set with the desidered value, it allows for example, to verify that **QTP 12** runs correctly, or if there are some communication problems on the serial line.

NOTE: This command uses the on board EEPROM, so before executing it is better to check the EEPROM availability through the proper command; in fact if it is not ready the command is ignored.

READ PRESENCE BYTE

Code: ***27 33 110***
Hex code: ***1B 21 6E***
Mnemonic: ***ESC ! n***

The command sends back the value of its presence byte.
For example, it can be useful to verify the presence, or the correct running, of the card and its firmware.

NOTE: This command uses the on board EEPROM, so before executing it is better to check the EEPROM availability through the proper command; in fact if it is not ready the command is not executed and the code **21** (**15 Hex**) = **NAK** is returned.

WRITE BYTE ON EEPROM

Code: ***27 164 addl addh byte***
Hex code: ***1B A4 addl addh byte***
Mnemonic: ***ESC ASCII(164) ASCII(addl) ASCII(addh) ASCII(byte)***

The value passed in **byte** parameter, included in range **0÷255** (**0÷FF Hex**), is write in the user EEPROM location for general use, identified by **addh addl** address.
The user EEPROM is a reserved area in the base EEPROM for general purpose, directly managed at byte level with no use of the other commands for messages, presence bytes, etc. The typical uses of this area are the memorization of configurations, setups, identifications, etc. that must be mantained also when power supply is absent. The address that identifies the used location is 16 bits wide and **addh**, **addl** are respectly the most and less significant part. The user EEPROM with the **QTP 12** firmwares has a size of 40 bytes, so the **addl** parameter must be included in range **0÷39** (**0÷27H**) while **addh** must always be 0. This choice has been made for compatibility with future expansions and other terminals.
When the command sequence contains not valid data, the command is ignored.
If, for example, the user wants to write the value 100 at address 35 of user EEPROM, then the following sequence must be sent:

27 164 35 0 100 or **1B A4 23 00 64 Hex** or **ESC ASCII(164) # NUL d**

NOTE: This command uses the on board EEPROM, so before executing it is better to check the EEPROM availability through the proper command; in fact if it is not ready the command is ignored.

READ BYTE FROM EEPROM

Code: ***27 165 addl addh***
Hex code: ***1B A5 addl addh***
Mnemonic: ***ESC ASCII(165) ASCII(addl) ASCII(addh)***

The value saved in user EEPROM location identified by **addh addl** address is read and returned. As described in the previous command the value of first parameter must be included in range **0÷39** (**0÷27H**) while the value of second parameter must always be 0. The returned data is a single character that is included in **0÷255** (**0÷FF Hex**) range.
When the command sequence contains not valid data, the command is ignored.

NOTE: This command uses the on board EEPROM, so before executing it is better to check the EEPROM availability through the proper command; in fact if it is not ready the command is not executed and the code **21** (**15 Hex**) = **NAK** is returned.

COMMANDS FOR KEYBOARD MANAGEMENT

Below are described the commands that can be used to manage the keys available on **QTP 12**. Detailed information about keys management and codes transmitted by the terminal, are available in KEYBOARD ACQUISITION paragraph.

KEY RECONFIGURATION

Code:	***27 55 key n. code***
Hex code:	***1B 37 key n. code***
Mnemonic:	***ESC 7 ASCII(key n.) ASCII(code)***

When the selected **key n.** is reconfigured, each time it is pressed, the card will send the new specified **code** on communication line.
The value of **key n.** to be reconfigured must be included in the range **0÷11** (**00÷0B Hex**) otherwise the command is ignored, and it will substitute the key described in figure 34.
The **code** value can vary in the range **0÷254** (**00÷FE Hex**) to obtain the same code when key is pressed, but value **255** (**FF Hex**) indicates that the key is disabled and when it will be pressed the **QTP** will not send any code.
Figure 35 reports the default key codes and the paragraph DATA STORED ON EEPROM indicates how to restore these codes in case of unwanted changes.

NOTE: This command writes data on the on board EEPROM, so before executing it is better to check the EEPROM availability through the proper command; in fact if it is not ready the command is ignored.

KEYCLICK ON WITHOUT MEMORIZATION

Code:	***27 53***
Hex code:	***1B 35***
Mnemonic:	***ESC 5***

The **keyclick** function is switched on so there is a timed sound feedback when a key is pressed (whenever the buzzer is already enabled, then it is disabled for the same time, in order to always ensure the key pressed recognition). This setting is not saved inside the on board EEPROM so if the terminal is powered off and on it goes back to the previous condition, defined and saved in local setup mode.

KEYCLICK OFF WITHOUT MEMORIZATION

Code:	***27 54***
Hex code:	***1B 36***
Mnemonic:	***ESC 6***

The **keyclick** function is disabled so there is not sound feedback when a key is pressed.
This setting is not saved inside the on board EEPROM so if the terminal is powered off and on it goes back to the previous condition, defined and saved in local setup mode.

KEYCLICK ON WITH MEMORIZATION

Code:	***27 33 53***
Hex code:	***1B 21 35***
Mnemonic:	***ESC ! 5***

This command enables **keyclick** function, so there is an audible feedback when a key is pressed (whenever the buzzer is already enabled, then it is disabled for the same time, in order to always ensure the key pressed recognition). This setting is stored on the on board EEPROM so if the card is turned off and on, it keeps the current condition.

NOTE: This command writes data on the on board EEPROM, so before executing it is better to check the EEPROM availability through the proper command; in fact if it is not ready the command is ignored.

KEYCLICK OFF WITH MEMORIZATION

Code:	***27 33 54***
Hex code:	***1B 21 36***
Mnemonic:	***ESC ! 6***

This command disables **keyclick** function, so there is not audible feedback when a key is pressed. This setting is stored on the on board EEPROM so if the card is turned off and on, it keeps the current condition.

NOTE: This command writes data on the on board EEPROM, so before executing it is better to check the EEPROM availability through the proper command; in fact if it is not ready the command is ignored.

FIGURE 45: FRONT PANEL WITH KEYBOARD

COMMANDS FOR USER CHARACTERS

QTP 12 lets the user define and show up to 8 user characters; those characters can be used to represent on display special characters, pseudo graphic characters, special symbols, etc. that are not still available in the same display (please refer to tables in appendix B).
The user characters can be defined and saved with a pattern equal to a 5 x 8 pixels matrix, so organized:

	Columns of pixels				
Rows of pixels	Pat 0.4	Pat 0.3	Pat 0.2	Pat 0.1	Pat 0.0
	Pat 1.4	Pat 1.3	Pat 1.2	Pat 1.1	Pat 1.0
	Pat 2.4	Pat 2.3	Pat 2.2	Pat 2.1	Pat 2.0
	Pat 3.4	Pat 3.3	Pat 3.2	Pat 3.1	Pat 3.0
	Pat 4.4	Pat 4.3	Pat 4.2	Pat 4.1	Pat 4.0
	Pat 5.4	Pat 5.3	Pat 5.2	Pat 5.1	Pat 5.0
	Pat 6.4	Pat 6.3	Pat 6.2	Pat 6.1	Pat 6.0
	Pat 7.4	Pat 7.3	Pat 7.2	Pat 7.1	Pat 7.0

FIGURE 46: USER CHARACTERS PATTERN

The user characters representation is really simple in fact it is sufficient to send the proper code (0 to 7 equal to 8 to 15) with a previous setting of representation mode, through OPERATING MODE SELECTION command.
When the user characters are saved, their patterns are written on EEPROM and then they are reloaded on display any time the terminal is powered on or when GENERAL RESET command is executed.

NOTE: Please remind that on models with fluorescent displays the character has a 5 x 7 pixels matrix (Pat 0÷Pat 6) and the last row of the pattern is not displayed. Furthermore on **QTP 12-F2** the value of Pat 7.4 pixel defines the status of all the five pixels Pat 7.4÷Pat 7.0, or in other words it defines the status of underline attribute of the defined character.

DEFINITION OF USER CHARACTER

Code: ***27 66 nchar Pat0 ÷ Pat7***
Hex code: ***1B 42 nchar Pat0 ÷ Pat7***
Mnemonic: ***ESC B ASCII(nchar) ASCII(Pat0) ÷ ASCII(Pat7)***

After the two command identification codes, other 9 bytes must be sent with the following meaning:

nchar	*(0÷7)*	*(00÷7 Hex)*	->	*Number of user character to define*
Pat0	*(0÷31)*	*(00÷1F Hex)*	->	*First byte of pattern equal to first high row of character.*
:	*:*		->	*:* *:* *:* *:*
Pat7	*(0÷31)*	*(00÷1F Hex)*	->	*Seventh byte of pattern equal to last low row of character.*

This command loads on the display the pattern of the user character **nchar** with the value placed in the eight bytes **Pat0 ÷ Pat7**, as described in figure 46; the pattern is only defined but not saved, so when **QTP 12** is turned off and on, the user character **nchar** doesn't mantain the supplied pattern. For example to define the user character 5 as an empty rectangle with maximum size, the following sequence has to be sent:

27 66 5 31 17 17 17 17 17 17 31 or
1B 42 05 1F 11 11 11 11 11 11 1F Hex

DEFINITION AND MEMORIZATION OF USER CHARACTER

Code: ***27 33 66 nchar Pat0 ÷ Pat7***
Hex code: ***1B 21 42 nchar Pat 0 ÷ Pat7***
Mnemonic: ***ESC ! B ASCII(nchar) ASCII(Pat0) ÷ ASCII(Pat7)***

After the three command identification codes, other 9 bytes must be sent with the following meaning:

nchar	*(0÷7)*	*(00÷7 Hex)*	->	*Number of user character to define and save*
Pat0	*(0÷31)*	*(00÷1F Hex)*	->	*First byte of pattern equal to first high row of character.*
:	*:*		->	*:* *:* *:* *:*
Pat7	*(0÷31)*	*(00÷1F Hex)*	->	*Seventh byte of pattern equal to last low row of character.*

This command loads on the display the pattern of the user character **nchar** with the value placed in the eight bytes **Pat0 ÷ Pat7**, as described in figure 46; moreover the pattern is also saved on EEPROM, so if **QTP 12** is turned off and on, the user character **nchar** mantain the supplied pattern.

NOTE: This command uses the on board EEPROM, so before executing it is better to check the EEPROM availability through the proper command; in fact if it is not ready the command is ignored.
Execution time of the command is about 80 msec. When the command has been transmitted and several commands must follow, it is better to insert a delay to avoid receive buffer overflow.

COMMANDS FOR MESSAGE MANAGEMENT

In the following paragraphs are described all the commands that manage messages, available in **QTP 12** terminal. The messages are 20 characters sequence that can be saved on board EEPROM/s and then reloaded or represented on display, simply by suppling the same message identification number. The most important function of messages is the possibility to show constant information on the display (i.e. alarms, equipment status, user instructions, etc.) without the transmission of the numerous characters of these information but only the few characters of the command. The **QTP 12** firmware manages the scrolling messages visualization, too; with this feature on a single line of display can be shown more text that continuosly shift from right to left.
Additionally the messages coincide with the entity used by power on visualization command, described in a following paragraph.
Please remind that a comfortable program for PC, named **QTP EDIT**, allows any user to edit the messages, save and load them on PC disks and transmit/receive them directly to/from **QTP** serially connected to same PC.
In the default configuration the **QTP 12** install a base EEPROM with a size of 2048 bytes, that can store up to 95 messages; through an optional EEPROM, that must be specified in the order phase, the maximum number of messages can be increased up to 3371, as described in next table. When the user has special requirements about EEPROM sizes, other different dimensions can be obtained, but they must be previously agreed upon grifo®.

READING OF MAX MESSAGE NUMBER

Code: ***27 110***
Hex code: ***1B 6E***
Mnemonic: ***ESC n***

This comand returns the number of the last messages that can be saved. It varys in compliace with the size of the EEPROM/s installed on the card, as reported in the below table:

Version (option)	*EEPROM size*	*Messages n°*	*Last message n°*	*Last group n° (max.grp) and last messago n° of group (max.msg)*
-	2K Bytes	95 (005FH)	94 (5EH)	0 (00H) 94 (5EH)
.EE128	2+16K Bytes	914 (0392H)	255 (FFH)	3 (03H) 145 (91H)
.EE256	2+32K Bytes	1733 (06C5H)	255 (FFH)	6 (06H) 196 (C4H)
.EE512	2+64K Bytes	3371 (0D2BH)	255 (FFH)	13 (0DH) 42 (2AH)

FIGURE 47: NUMBER OF MESSAGES ON EEPROM

This command has been implemented for compatibility and interchangeability with other **QTP** operator panels and it returns a valid response only when optional EEPROM is not available. To obtain always the right messages number it is suggested the following command.

READING OF LAST GROUP AND MESSAGE MANAGED

Code:	***27 33 109***
Hex code:	***1B 21 6D***
Mnemonic:	***ESC ! m***

This comand returns the number of the last group of messages that can be saved and the number of the last message inside this group. Both these values change according to EEPROM/s sizes installed on the board, as described in the last column of previous table. As convention we define **max.grp** and **max.msg** the two numbers returned as response by this command, and they will be used in all the following descriptions.
All the numerous messages of **QTP 12** have been divided in groups of 256 messages, to allow their identification. With this tecnique it is really easy and fast to convert the two returned values in the number of the last message = **max.grp** * 256 + **max.msg**.

SELECT CURRENT MESSAGE GROUP

Code:	***27 33 77 grp***
Hex code:	***1B 21 4D grp***
Mnemonic:	***ESC ! M ASCII(grp)***

It selects the message group **grp** that must be used with the following commands for messages management. The message group identifies a set of 256 messages, as described in previous command, that has been adopted to easily address all the messages with a byte codification of the message number. The valid values for message group are those included in range **0÷max.grp** (where **max.grp** is the last group reported on figure 47), otherwise the command is not executed.
After a power on or a GENERAL RESET command, it is always selected the the first message group, that is those with number **0** (**00 Hex**).
If, for example, the user wants to select the group for message number 300, then the following sequence must be sent:

27 33 77 1 or **1B 21 4D 01 Hex** or **ESC ! M SOH**

MESSAGE STORAGE

Code:	**27 33 67 mess.n. chr.0 ÷ chr.19**
Hex code:	**1B 21 43 mess.n. chr.0 ÷ chr.13 Hex**
Mnemonic:	**ESC ! C ASCII(mess.n.) ASCII(chr.0) ÷ ASCII(chr.19)**

This command stores the 20 characters message, identified by **mess.n.** parameter, on the on board EEPROM/s, in the currently selected message group. The 20 chars which form the message must be visualizable on the display so they must be included in the range **0÷255** (**00÷FF Hex**). The message number must be included in the range **0÷max.msg** when the last message group is selected, otherwise in the range **0÷255** (**00÷FF Hex**), or in other words identify one of the available messages. If this number is out of range, the command is ignored.

NOTE: This command uses the on board EEPROM, so before executing it is better to check the EEPROM availability through the proper command; in fact if it is not ready the command is ignored.
The execution of this command depends on jumper J7 configuration, as described in paragraph OPTIONAL EEPROM PROTECTION.

MESSAGE READING

Code:	***27 33 69 mess.n.***
Hex code:	***1B 21 45 mess.n.***
Mnemonic:	***ESC ! E ASCII(mess.n.)***

This command reads from EEPROM/s the 20 characters message identified by **mess.n.** parameter, in the currently selected message group, and it returns this message, beginning from the first char of the string.
The message number must be included in the range **0÷max.msg** when the last message group is selected, otherwise in the range **0÷255** (**00÷FF Hex**), or in other words identify one of the available messages. If this number is out of range, the command is ignored.

NOTE: This command uses the on board EEPROM, so before executing it is better to check the EEPROM availability through the proper command; in fact if it is not ready the command is not executed and 20 characters with the code **21** (**15 Hex**) = **NAK** are returned.

VISUALIZATION OF MESSAGES

Code:	***27 33 68 mess.n. n***
Hex code:	***1B 21 44 mess.n. n***
Mnemonic :	***ESC ! D ASCII(mess.n.) ASCII(n)***

This command visualizes **n** 20 characters messages on the display, beginning from current cursor position.
The first of the **n** messages is that one having the number corresponding to **mess.n.**, in the currently selected message group, while the remaining messages are those immediately subsequents in EEPROM/s.
The **mess.n.** value and the number of the following messages defined by **n**, must be included in the range **0÷max.msg** when the last message group is selected, otherwise in the range **0÷255** (**00÷FF Hex**), or in other words identify one of the available messages. If these numbers are out of range, the command is ignored..
The **n** quantity of messages to be visualized depends on the maximum number of characters of the installed display; on **QTP 12** this number is 40 and so the maximum number of messages is 2. In other words the **n** parameter can be set with a value in the range **1÷2** and if it is out of this range, the command is ignored.
Once the command is executed the cursor is placed in the next position of the last character visualized; if the last character of the said message occupies the last position of the display, the cursor will be placed in Home position.
For example, to visualize the messages number 10 and 11, first of all it will be necessary to send the command that select message group 0, and then the following sequence:

27 33 68 10 2 or **1B 21 44 0A 02 Hex** or **ESC ! D LF STX**

NOTE: This command uses the on board EEPROM, so before executing it is better to check the EEPROM availability through the proper command; in fact if it is not ready the command is delayed until the operation under execution is completed.

SCROLLING MESSAGES VISUALIZATION

Code:	**27 33 83 mess.n. n.chr**
Hex code:	**1B 21 53 mess.n. n.chr**
Mnemonic :	**ESC ! S ASCII(mess.n.) ASCII(n.chr)**

This command visualizes a string, long **n.chr** characters, on the first line of display in sliding mode. The string is shifted from right to left and so the user can visualize on a single line (the first) many information, more than the normal 20 characters.
The string of **n.chr** characters, begins with the first character of the **mess.n.** message already stored in EEPROM/s, in the currently selected message group, and continues with next characters (always saved in following EEPROM/s messages).
The **mess.n.** value must be included in the range **0÷max.msg** when the last message group is selected, otherwise in the range **0÷255** (**00÷FF Hex**), or in other words identify one of the available messages. If this number is out of range, the command is ignored.
Instead the **n.chr** parameter must range in the following values:

0	Stops the scrolling messages visualization (the **mess.n.** value doesn't care).
20÷200	Enables sliding visualization of the specified characters.

If **n.chr.** value is out of the specified ranges or it points after the last character of the last message stored in EEPROM/s, the command will be ignored.
The scrolling messages visualization is always performed on the first display line and the cursor position and attributes are mantained.
For example, if you wish to visualize a 35 characters string in sliding mode, composed by message 10 (20 characters) and by the first 15 characters of message 11, first of all it will be necessary to send the command that select message group 0, and then the following sequence:

27 33 83 10 35 or **1B 21 53 0A 23 Hex** or **ESC ! S LF #**

NOTE: This command uses the on board EEPROM, so before executing it is better to check the EEPROM availability through the proper command; in fact if it is not ready the command is delayed until the operation under execution is completed.
The message visualization in sliding mode is managed in background and so there is an increased firmware execution time, that causes a subsequent slowing down on interpretation of the data coming from command unit. This is the reason why it is necessary to wait for few **msec** between the transmission of 20÷30 bytes data blocks when many information and/or commands are sent to **QTP 12**. In this way misunderstanding and interpreting problems of the received data, caused by receive buffer overflow, are completely avoid.

SET AUTOMATIC VISUALIZATION

Code: ***27 150 255 mess.n len shift r c***
Hex code: ***1B 96 FF mess.n len shift r c***
Mnemonic: ***ESC ASCII(150) ASCII(255) ASCII(mess.n) ASCII(len) ASCII(shift) ASCII(r) ASCII(c)***

This command set the power on visualization of **QTP 12** that is a possible representation automatically displayed, immediately after a power on phase, and that stay on display until the first data is received from command unit.
In this paragraph the term visualization and representation always refer to the visual results on the display and it can be selected among many possibilities, defined by the proper parameters ot the command. In detail it is possible to display: a **single message** in any position, a **static messages sequence** (screen) in any position and an **auto scrolling messages** sequence only on the first row. For this purpose the parameters have the following meaning:

mess.n *it coincides with the number of the first message to show, of the group 0, and the possible others, are those immediately successive on EEPROM. The* ***mess.n*** *value, in order to be valid, must respect the conditions below described:*

- ***0≤mess.n≤msggrp0*** *-> to enable visualization or, in other words, it must select an available message (where maxgrp0 is the number of the last message in group 0, described in the fourth column of figure 47)*
- ***mess.n=255 (FFH)*** *-> to disable visualization*

len *it coincides with the length of visualization and it is expressed in a different way according with visualization attribute:*

- static visualization (shift=0): it corresponds to messages number. The ***len*** *value, in order to be valid, must respect the following conditions:*

- ***mess.n+len<=msggrp0*** *-> it must select available messages in group 0 (where maxgrp0 is the number of the last message in group 0, described in the fourth column of figure 47)*
- ***1 ≤ len ≤ 2*** *-> it must not exceed the display dimensions equal to 2 messages of 20 characters max.*

- scrolling visualization (shift=255): it corresponds to characters number. The ***len*** *value, in order to be valid, must respect the following conditions:*

- ***len=0*** *-> stops the scrolling in execution*
- ***20≤length≤200*** *-> the range of shiftable characters*

shift *it coincides with the scrolling visualization attribute and it can assume two possible values:*

0	*00H*	*NUL*	*->*	*static visualization*
255	*FFH*	*ASCII(255)*	*->*	*scrolling visualization*

r *when the representation is static (shift=0) it coincides with the row where the visualization starts and its valid values range is* ***0÷1****. When the visualization has the scroll attribute (shift=255) the representation occurs always on the first line of display and the parameter value doesn't care.*

c *when the representation is static (shift=0) it coincides with the column where the visualization starts and it valid values range is* ***0÷19****. When the visualization has the scroll attribute (shift=255) the representation occurs always on the first line of display and the parameter value doesn't care.*

Whenever in the received sequence there are not valid data, the command is ignored viceversa the automatic visualization is immediately saved on EEPROM in order to mantain it when power off and on occours. In fact this command arranges the **QTP 12** as a system, that show the visualization before than any communications occour with the command unit, and for this reason the visualization is saved, recalled and managed by the single terminal.
The power on visualization uses only the messages that belong to first group 0 and the message group currently selected when the command is received, is completely indifferent.
Among the typical use of this command there are: the visualization of the general information (name, address, telephone, etc.) of the company that developed the application, the timed visualization of the firmware and/or software version in execution, the immediate visualization of instructions for the operator that has just powered on the machine. Moreover the command adds another application field for the **QTP 12** as a stand alone visualization system, that can even work without any external command unit.
For example, if you wish to enable a power on view, with static visualization of messages 10 and 11 from the first row of display, it will be necessary to send the following sequence:

27 150 255 10 2 0 0 0 or **1B 96 FF 0A 02 00 00 00 Hex** or
ESC ASCII(150) ASCII(255) LF STX NUL NUL NUL

NOTE: This command uses the on board EEPROM, so before executing it is better to check the EEPROM availability through the proper command; in fact if it is not ready the command is ignored.
When a command unit is connected to **QTP 12** ensures that it doesn't transmit any characters (even spurious ones) during power on phase, or the power on messages will be immediately removed from display.

COMMANDS FOR I2C BUS COMMUNICATION AS MASTER

On **QTP 12** are available a group of commands let the command unit communicate in master mode to all the I2C BUS peripherals that operates as slaves. This commands coincide with the essential elements that once properly combined allows to communicate with any device provided of this standard (temperature sensors, A/D and D/A converters, etc.).
Naturally these commands are superfluos when communication with **QTP 12** is already performed on I2C BUS line infact in this case the command unit can directly exchange data with the I2C BUS peripherals, as it already does with **QTP**. Viceversa the commands become really useful when the command unit communicate through the asynchronous serial line and in this condition **QTP 12** acts as a **serial <-> I2C BUS converter**.
The figure 48 shows a possible connection diagram for some I2C BUS peripherals that can be managed by these commnads; certainly the operator panel resources can be expanded with a reduced cost and a short development time of management software.
About physical protocol of I2C BUS line managed by these commands, the following features are used:

- Bit rate: 50000 bits per second
- Mode: Master (transmit and receive)
- Slave Address: all the even addresses in range **0÷254** (**00÷FE Hex**), except the values **160** (**A0 Hex**) , **162** (**A2 Hex**) already used on board and the one defined in local setup of **QTP**.

About electric connection, please remind that the I2C BUS line in Master mode coincides with the one used for Slave mode. The connection is always performed through CN6, by following the indications already reported in paragraph CN6 - I2C BUS LINE CONNECTOR.

START I2C BUS

Code: ***27 250***
Hex code: ***1B FA***
Mnemonic: ***ESC ASCII(250)***

This command generates the start sequence on the I2C BUS line.

STOP I2C BUS

Code: ***27 251***
Hex code: ***1B FB***
Mnemonic: ***ESC ASCII(251)***

This command generates the stop sequence on the I2C BUS line.

TRANSMIT BYTE ON I2C BUS

Code: *27 252 byte*
Hex code: *1B FC byte*
Mnemonic: *ESC ASCII(252) ASCII(byte)*

This command transmits the **byte** passed as parameter on the I2C BUS line and it reads the acknowmedge bit from peripheral. At the end of operation it always returns a character with the read bit status that can assume the possible values **0** (**00 Hex**) or **1** (**01 Hex**).
The command can be used to perform many operations required by I2C BUS communication, infact all the data exchanged with this standard are organized in bytes (Slave Address, commands, addresses, etc.) that must be sent to peripheral devices.

RECEIVE BYTE FROM I2C BUS

Code: *27 253 ack*
Hex code: *1B FD ack*
Mnemonic: *ESC ASCII(253) ASCII(ack)*

This command receive a byte from the I2C BUS line and returns it. Then it performs the following acknowledge operations, according with **ack** parameter value:

0 (00 Hex) -> *sends bit at 0 (**ACK**)*
1 (01 Hex) -> *sends bit at 1 (**NAK**)*

The command can be used to perform many operations required by I2C BUS communication, infact all the data exchanged with this standard are organized in bytes (data, status, addresses, etc.) that must be received from peripheral devices.

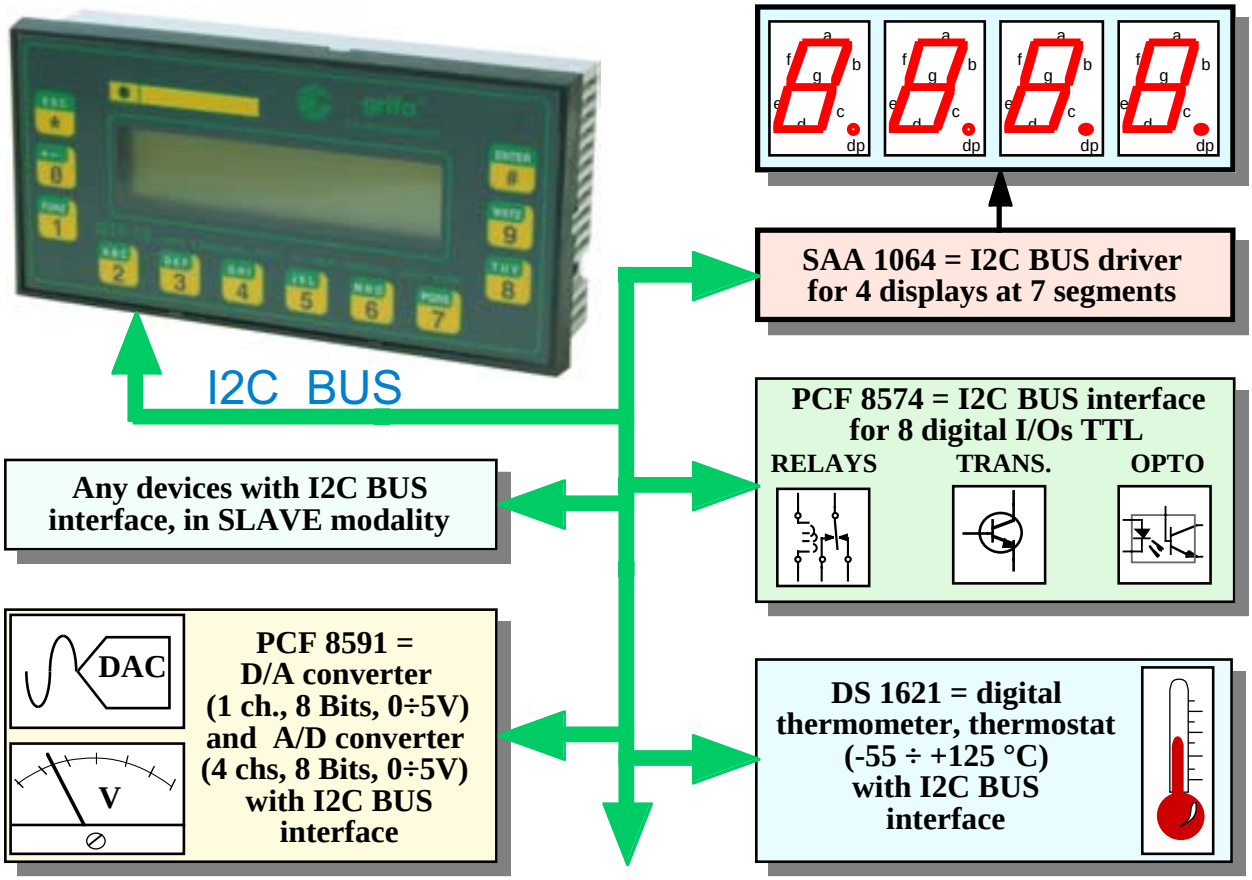


FIGURE 48: CONNECTION OF I2C BUS LINE AS MASTER

COMMANDS FOR SRAM AND CLOCK

Here follow commands that manage the backed Real Time Clock and SRAM available on **QTP 12.RTC**. Whenever the .RTC option is not ordered all the commands described in this paragraph are ignored because the necessary hardware components are not installed.
The correct updating of the clock and the SRAM data preservation is ensured also when power supply is not available, as described in BACK UP paragraph.
Please note that 16 bytes of the 240 provided by backed SRAM are reserved for internal functionalities and consequently only 224 bytes are available for users.
Among the most important functions of the SRAM+RTC we remind the following examples: the automatic visualization on display of current time and date; the use of a complete clock through one of the communication lines; the management of a clock alarm that autonomously check a predefined time; the activation of one of the relay outputs when the clock alarm time is reached; the memorization and the acquisition of data that change values continuosly; etc.

WRITE BYTE ON BACKED SRAM

Code: *27 33 71 addr byte*
Hex code: *1B 21 47 addr byte*
Mnemonic: *ESC ! G ASCII(addre) ASCII(byte)*

It writes the value passed in **byte** parameter, variable in range **0÷255 (00÷FF Hex**), on the backed SRAM. The address where memorize this byte is passed in **addr** parameter and it must be included in the range **32÷255** (**20÷FF Hex**) otherwise the command is ignored.

READ BYTE FROM BACKED SRAM

Code: *27 33 103 addr*
Hex code: *1B 21 67 addr*
Mnemonic: *ESC ! g ASCII(addr)*

It returns a value that is the byte stored on backed SRAM, at the address specified by the **addr** parameter. The address must be included in the range **32÷255** (**20÷FF Hex**) otherwise the command is ignored.

SET CLOCK

Code: *27 33 70 hou min sec day mon yea wee*
Hex Code: *1B 21 46 hou min sec day mon yea wee*
Mnemonic: *ESC ! F ASCII(hou) ASCII(min) ASCII(sec) ASCII(day) ASCII(mon) ASCII(yea) ASCII(wee)*

The on board Real Time Clock is set with the data contained in the passed parameters; if one of these ones has a value not included in the allowed range, the command is ignored and none of the RTC fields are changed.
Here under it is listed the detailed meaning of the 7 parameters above described, and their validity ranges.

PARAMETER	*RANGE*	*MEANING*
hou	0÷23 (00H÷17H)	Hours
min	0÷59 (00H÷3BH)	Minutes
sec	0÷59 (00H÷3BH)	Seconds
day	0÷31 (00H÷1FH)	Day of month
mon	1÷12 (00H÷0CH)	Month
yea	0÷99 (00H÷63H)	Year
wee	0÷6 (00H÷06H)	Day of week: 0 -> Sunday : : : 6 -> Saturday

FIGURE 49: REAL TIME CLOCK PARAMETERS

NOTE: To ensure the validity of the two digits year, managed by clock of **QTP 12.RTC**, it is absolutely necessary to send the SET CLOCK command at least one time each 4 years. This is a quite normal condition in fact tipically the clock time is adjusted one or two times for any year.

ACQUIRE CLOCK

Code: *27 33 102*
Hex Code: *1B 21 66*
Mnemonic: *ESC ! f*

The command returns 7 values, named hou, min, sec, day, mon, yea, wee, that corresponds to the current time and date parameters, acquired from the on board Real Time Clock.
The meaning of these bytes is the one explained in previous table.

SHOW TIME ON DISPLAY

Code: ***27 33 116 r c frm***
Hex Code: ***1B 21 74 r c frm***
Mnemonic: ***ESC ! t ASCII(r) ASCII(c) ASCII(frm)***

The time acquired from the on board Real Time Clock is displayed starting from the display position passed in **r** and **c** parameters. These express the row and column values of display referred to Home position with coordinate 0, 0, plus a constant offset of **32** (**20 Hex**). The position is expressed in alphanumeric mode so their valid values ranges respectively are 32÷33 and 32÷51. When row and/or column values are not compatible with the specified ranges, the command is ignored.
The **frm** parameter is used to specify the visualization format, with the following meaning:

Bit 0	->	1	Enables the automatic visualization of time in the display position defined by **r** and **c**.
		0	Disables the automatic visualization of time. The values of **r** and **c** are not used.
Bit 1	->	1	The time is visualized in AM/PM format: **HH:MM:SSm** where **HH** are the hours, **MM** the minutes, **SS** the seconds and **m** is the AM (**a**) or PM (**p**) indication.
		0	The time is visualized in 24H format: **HH:MM:SS** where **HH** are the hours, **MM** the minutes and **SS** the seconds.
Bit 2	->	1	Enables the alternate visualization of time and date on the same position of display.
		0	Disables the alternate visualization of time and date.
Bit 3÷7	->	0	Reserved for future expansions. They must be set to **0** value.

For example, if you wish to visualize the time in 24 H format, starting from the 10th character of the 2nd row (row 1, column 9), it will be necessary to send the following sequence:
27 33 116 33 41 1 or **1B 21 74 21 29 01 Hex** or **ESC ! t !) SOH**

NOTE: The time visualization is managed in background and so there is a slowing down on interpretation of data coming from command unit. This is the reason why it is necessary to wait for few **msec** between the transmission of 20÷30 bytes data blocks when many information and/or commands are sent to **QTP 12**. In this way misunderstanding and interpreting problems of the received data, caused by receive buffer overflow, are completely avoid.

SHOW DATE ON DISPLAY

Code: ***27 33 100 r c frm***
Hex code: ***1B 21 64 r c frm***
Mnemonic: ***ESC ! d ASCII(r) ASCII(c) ASCII(frm)***

The date acquired from the on board Real Time Clock is displayed starting from the display position passed in **r** and **c** parameters. These express the row and column values of display referred to Home position with coordinate 0, 0, plus a constant offset of **32** (**20 Hex**). The position is expressed in alphanumeric mode so their valid values ranges respectively are 32÷33 and 32÷51. When row and/or column values are not compatible with the specified ranges, the command is ignored.
The **frm** parameter is used to specify the visualization format, with the following meaning:

Bit 0 ->	1	Enables the automatic visualization of date in the display position defined by **r** and **c**.
	0	Disables the automatic visualization of date. The values of **r** and **c** are not used.
Bit 1 ->	1	The date is visualized in English format: **MM-DD-YY**, where **MM** is the month, **DD** the day and **YY** the year.
	0	The date is visualized in Italian format: **DD-MM-YY**, where **DD** is the day, **MM** the month and **YY** the year.
Bit 2 ->	1	Enables the alternate visualization of date and time on the same position of display.
	0	Disables the alternate visualization of date and time.
Bit 3÷7 ->	0	Reserved for future expansions. They must be set to "0" value.

Please note that the week day is not displayed.
The alternate visualization of date and time, on the same display position, is obtained by sending both the commands SHOW TIME and SHOW DATA with the bit **frm.2**=1 and with the same coordinate **r, c** values.
For example, if you wish to visualize the date in English format MM-DD-YY, starting from the 10th character of the 2nd row (row 1, column 9), it will be necessary to send the following sequence:

27 33 100 33 41 3 or **1B 21 64 21 29 03 Hex** or **ESC ! d !) ETX**

NOTE: The datee visualization is managed in background and so there is a slowing down on interpretation of data coming from command unit. This is the reason why it is necessary to wait for few **msec** between the transmission of 20÷30 bytes data blocks when many information and/or commands are sent to **QTP 12**. In this way misunderstanding and interpreting problems of the received data, caused by receive buffer overflow, are completely avoid.

SET CLOCK ALARM

Code:	***27 33 70 hou min sec day mon dton***
Hex Code:	***1B 21 46 hou min sec day mon dton***
Mnemonic:	***ESC ! F ASCII(hou) ASCII(min) ASCII(sec) ASCII(day) ASCII(mon) ASCII(dton)***

The command sets and enables the on board clock alarm with the data contained in the passed parameters; if one of these ones has a value not included in the allowed range (see figure 49), the command is ignored.
Once received the command, the **QTP 12** clock alarm is set and it is also enabled and deactivated. The word enabled means that the current time and date are compared with those of the clock alarm and, when all the parameters match, the clock alarm is activated. At this point the clock remains active for the time period passed in **dton** parameter and then it is deactivated and disabled. The **dton** alarm activation time is expressed in tens of seconds, with the following corrispondence:

dton =	0	->	alarm remains active 1 second
	1	->	alarm remains active 10 seconds
	2	->	alarm remains active 20 seconds
	:	:	: : : : :
	255	->	alarm remains active 2550 seconds

When the menu **INTRTC FN.** of local setup is set to **ALARM** option, during the clock alarm activation time, it is enabled also the /INTRTC digital output, connected to homonimous pods, in a completely automatic mode. In this condition the user can act even on an external actuator with no requirements of additional operations (read paragraph COMMANDS FOR DIGITAL OUTPUT MANAGEMENT for additional information on the use of this output).
The user must remind the following notes that concern the clock alarm:

NOTE:
- After a power on the alarm clock is always disabled and deactivated; so a possible enable, performed before the power off, is lost. Viceversa the defined alarm parameters are mantained and they can be acquired with the proper ACQUIRE CLOCK ALARM command.
- The command SET CLOCK doesn't affect the clock alarm setting and functionality.
- The described SET CLOCK ALARM command always enables and deactivates the clock alarm, indipendently from the previous status. A possible alarm activation time under execution it is interrupted and the /INTRTC output is not active.
- At the end of activation time the alarm clock is disabled and deactivated; in other words it can be re-enabled only with a new SET CLOCK ALARM command.
- The current status of the clock alarm, included enable and active conditions, can be comfortably acquired through the following command.

ACQUIRE CLOCK ALARM

Code: ***27 33 97***
Hex Code: ***1B 21 61***
Mnemonic: ***ESC ! a***

The command returns the 5 clock alarm parameters (hou, min, sec, day, mon), defined with the last command SET CLOCK ALARM, and a sixth value with the current clock alarm status, with the following information:

Bit 0 -> clock alarm enabled status
- 1 Clock alarm enabled and waiting for the defined time
- 0 Clock alarm disabled (no controls are performed)

Bit 1 -> clock alarm active status
- 1 Clock alarm activated, that means predefined time reached and it waits the elapsing of the alarm activation time **dton**. In this condition the /INTRTC output is active.
- 0 Clock alarm deactivated, that means predefined time is not still reached or it has already been reached and the alarm activation time **dton** is finished. In this condition the /INTRTC output is not active.

Bit 2÷7 -> 0 Not used.

Through this command the user can easily check if the current time and date have reached and matched the time already set with the previous command. So he must not provide complicated time controls, that involves numerous crossed checks, in his application software.

COMMANDS FOR DIGITAL OUTPUT MANAGEMENT

Below are listed the commands that manage the optional digital output, available in the **QTP 12.RTC**. Please remind that this outputs is available on two specific pods on printed circuit board, as described in figure 6 and that the same output can't drive directly the field signals; in other words, in order to drive every loads, it must be properly buffered.
Whenever the .RTC option is not ordered all the commands described in this paragraph are ignored because the necessary hardware components are not installed and the /INTRTC output is not available. Moreover the commands related to digital output are executed only if the menu **INTRTC FN.**. of the local setup is set to **USER** option.
Among the most important features of the digital outputs it can be listed, for example, the management of power relays, lamps, electric valves, motors, heaters, etc. or any other actuators that assume only the two state of active/deactive (ON/OFF).

WRITE ALL DIGITAL OUTPUTS

Code: ***27 166 out***
Hex code: ***1B A6 out***
Mnemonic: ***ESC ASCII(166) ASCII(out)***

All the digital relays outputs are set with **out** value, according to following correspondence:

(MSB) 0 0 0 0 0 **NO OUT 3** **NO OUT2** **NO OUT1** (LSB)

Where **NO OUT n** stands for the logic state, **0** (output deactive=relay contact open) or **1** (output active=relay contact closed), that the respective relay outputs, on CN7, must assume.
When the received sequence contains invalid data the command is ignored.
If, for example, only the NO OUT3 and NO OUT1 outputs must be enabled, then the following sequence must be sent:

27 166 5 or **1B A6 05 Hex** or **ESC ASCII(166) ENQ**

ENABLE SINGLE DIGITAL OUTPUT

Code: ***27 168 bit***
Hex code: ***1B A8 bit***
Mnemonic: ***ESC ASCII(168) ASCII(bit)***

This command sets to logic state **1** (output active=relay contact closed) the relay digital output identified by **bit** parameter, that has the following correspondence with CN7 signals:

1 -> NO OUT1 **2** -> NO OUT2 **3** -> NO OUT3

When the received sequence contains invalid data, the command is ignored.
If, for example, the output NO OUT2 must be enabled with no modifications on the remaining outputs, then the following sequence must be sent:

27 168 2 or **1B A8 02 Hex** or **ESC ASCII(168) STX**

DISABLE SINGLE DIGITAL OUTPUT

Code: ***27 170 bit***
Hex code: ***1B AA bit***
Mnemonic: ***ESC ASCII(170) ASCII(bit)***

This command sets to logic state **0** (output deactive=relay contact opened) the relay digital output identified by **bit** parameter, that has the following correspondence with CN7 signals:

1 -> NO OUT1 **2** -> NO OUT2 **3** -> NO OUT3

When the received sequence contains invalid data, the command is ignored.

If, for example, the output NO OUT1 must be disabled with no modifications on the remaining outputs, then the following sequence must be sent:

27 170 1 or **1B AA 01 Hex** or **ESC ASCII(170) SOH**

GRAPHIC COMMANDS

QTP 12-GF2, featuring a 140 x 16 pixels graphic display, allows the possibility to show graphic images, histograms, characters with different font and size, diagrams, etc.
A set of graphic commands is availabe; such commands are based on pixels (smallest visible entity) orgranized in the following coordinates system:

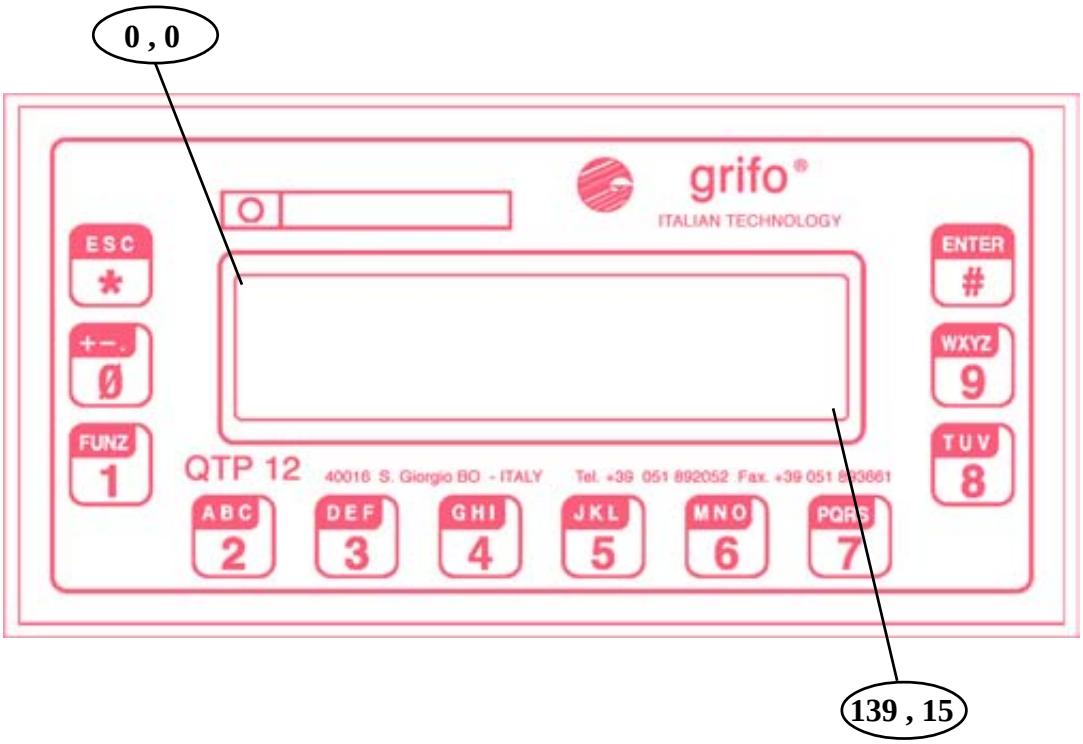


FIGURE 50: COORDINATES OF PIXELS ON GRAPHIC DISPLAY

NOTE Please remark that following graphic commands can be used only on **QTP 12-GF2** model while the remaining models do not recognize them as commands; consequently they show the command codes sequence on display. In addition, **QTP 12-GF2** recognizes both graphic and alphanumeric commands.

ALPHANUMERIC MODE SETTING

Code: ***27 208***
Hex Code: ***1B D0***
Mnemonic: ***ESC ASCII(208)***

This command sets alphanumeric representation mode, which allows to use all alphanumeric commands.
After command execution, the cursor (if enabled) is shown in the last position decided by previous commands.
After power on, alphanumeric mode is immediatly set.

FIGURE 51: FIRST GRAPHIC EXAMPLE

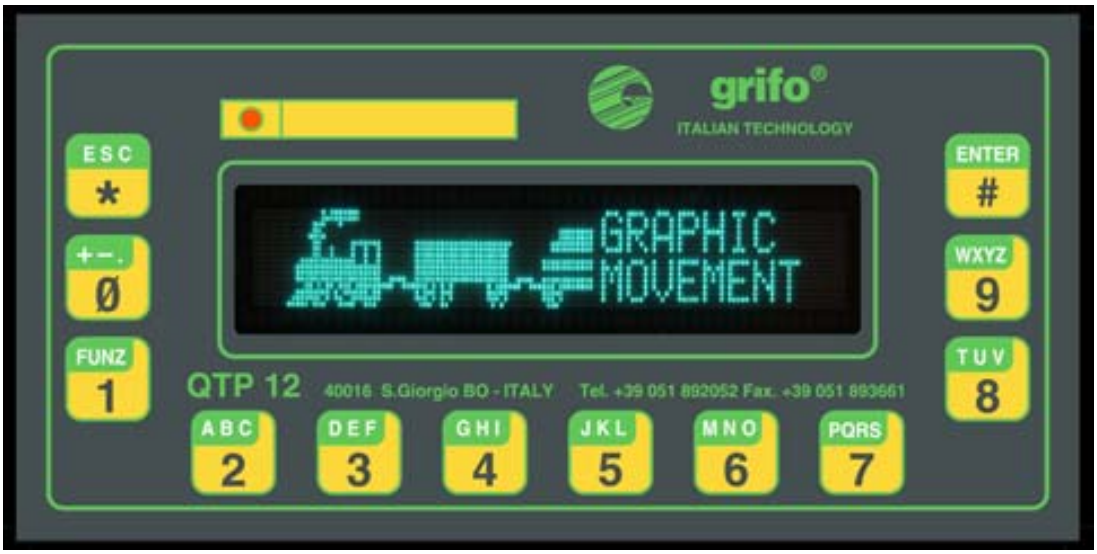


FIGURE 52: SECOND GRAPHIC EXAMPLE

GRAPHIC MODE SETTING

Code: ***27 209***
Hex Code: ***1B D1***
Mnemonic: ***ESC ASCII(209)***

This command sets graphic mode, it enables the interpretation of data as graphic and not as commands.
After execution of this command there is no effect on display, but characters received are not interpreted as commands any more; they go directly to the display.
One-character commands have no effect until alphanumeric mode is restored.
On other hand, commands starting with ESC = 27 = 1BH are alwats enalbed, despite the mode selected.
After power on, alphanumeric mode is set by default.

GRAPHIC CURSOR ABSOLUTE POSITION

Code: ***27 206 y x 0***
Hex Code: ***1B CE y x 00***
Mnemonic: ***ESC ASCII(206) ASCII(y) ASCII(x) NUL***

Moves the cursor to the point of coordinates **x** and **y**; the position indicated by these two numers is absolute, so it is not affected by all other parameters and is beyond the range of normal alphanumeric positioning.
Characters received after this command are displayed from indicated point, and are drawn to the right and to the top.
Values of coordinates **y** and **x** must be in the range **0÷15** and **0÷139**, that is included in size of display used.

NOTE: Code **0** (**NUL**) is present for compatibility with future versions of firmware: it must be always transmitted anyway.

GRAPHIC AREA SETTING

Code: ***27 241 x1 y1 x2 y2 cmd***
Hex Code: ***1B F1 x1 y1 x2 y2 cmd***
Mnemonic: ***ESC ASCII(241) ASCII(x1) ASCII(y1) ASCII(x2) ASCII(y2) ASCII(cmd)***

Sets graphic work area and the action to make on it.
Top left corner of area is set with coordinates **x1** and **y** , bottom right corner is **x2** and **y2**.
Values of **y1**, **y2** and **x1**, **x2** must be in the ranges **0÷15** and **0÷139**, that is included in size of display used.
Byte **cmd** selcts the action to perform to graphic area also according to the next bytes received from serial line:

cmd = *67 (43 Hex) C -> Clears selected area.*

70 (46 Hex) F -> Fills selected area.

72 (48 Hex) H -> Draw the following horizontal graphic data with horizontal shift.

73 (49 Hex) I -> Inverts selected area.

79 (4F Hex) O -> Draws a frame around selected area.

86 (56 Hex) V -> Draw the following vertical graphic data with horizontal shift.

104 (68 Hex) h -> Draw the following horizontal graphic data with vertical shift.

111 (6F Hex) o -> Delete a frame around selected area.

118 (76 Hex) v -> Draw the following vertical graphic data with vertical shift.

About commands that draw data in graphc area (H,h,V,v), the next bytes sent to the terminal are used as graphic data that will define pixels activation of display. The correspondance between pixels and bits of these bytes is explained in following figures 53÷57 where all the four organization and shift modes are described. In addition please remind that logic status **1** of a bit correspond to **activation** of corresponding pixel and viceversa.
This command can stop its execution in two different conditions:

- when sufficient bytes have been received and the drawing of the selected graphic area is complete;
- when a different command is received; in this condition the command graph area setting is interrupted and on display it will be displayed only the pixels received up to that istance.

For example, to draw an arrow like the one in the following figure on the top left corner of display:

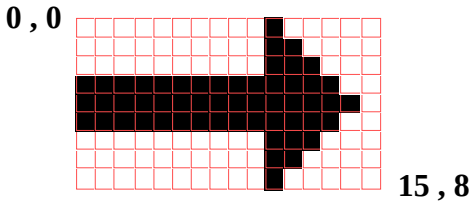


FIGURE 53: EXAMPLE OF GRAPHIC DRAWING

the command unit first send the sequence:

27 241 0 0 15 8 72 or **1B F1 00 00 0F 08 48 Hex**

then the sequence of graphic data:

0	**0**	**0**	**255**	**255**	**255**	**0**	**0**	**0**	**32**	**48**	**56**	**252**	**254**	**252**	**56**	**48**	**32**	or
00	**00**	**00**	**FF**	**FF**	**FF**	**00**	**00**	**00**	**20**	**30**	**38**	**FC**	**FE**	**FC**	**38**	**30**	**20**	**Hex**

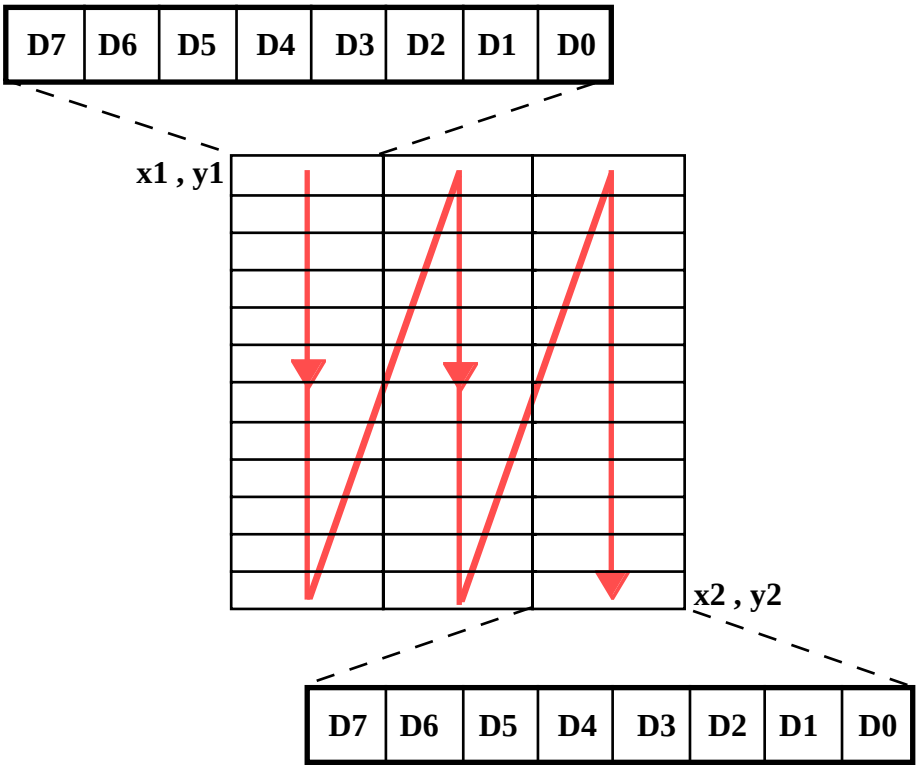


FIGURE 54: HORIZONTAL DATA AND HORIZONTAL SHIFT

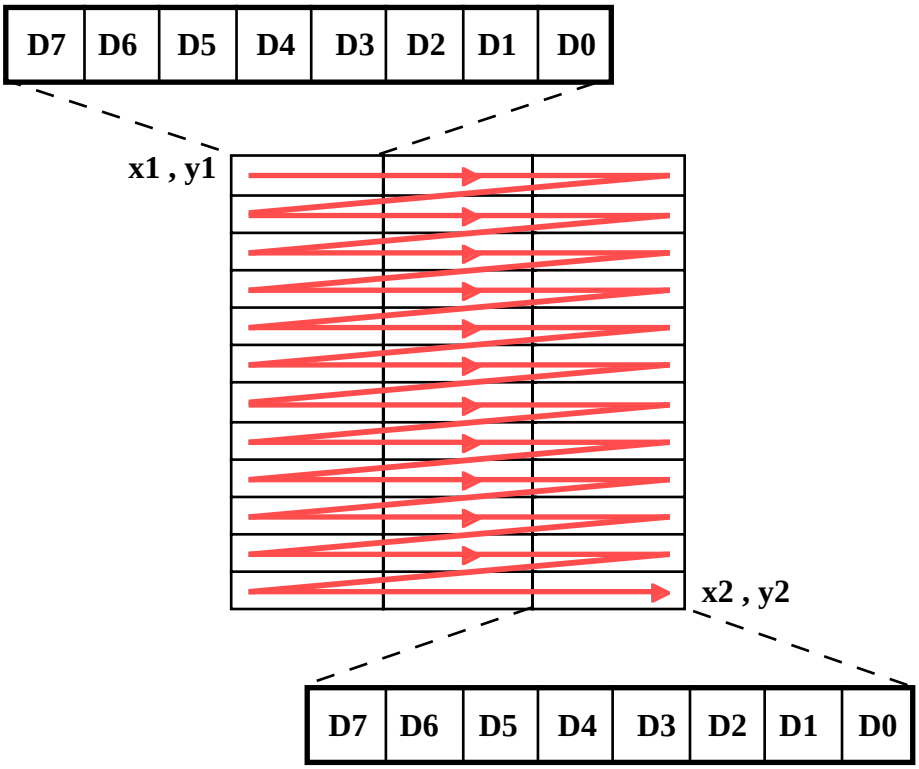


FIGURE 55: HORIZONTAL DATA AND VERTICAL SHIFT

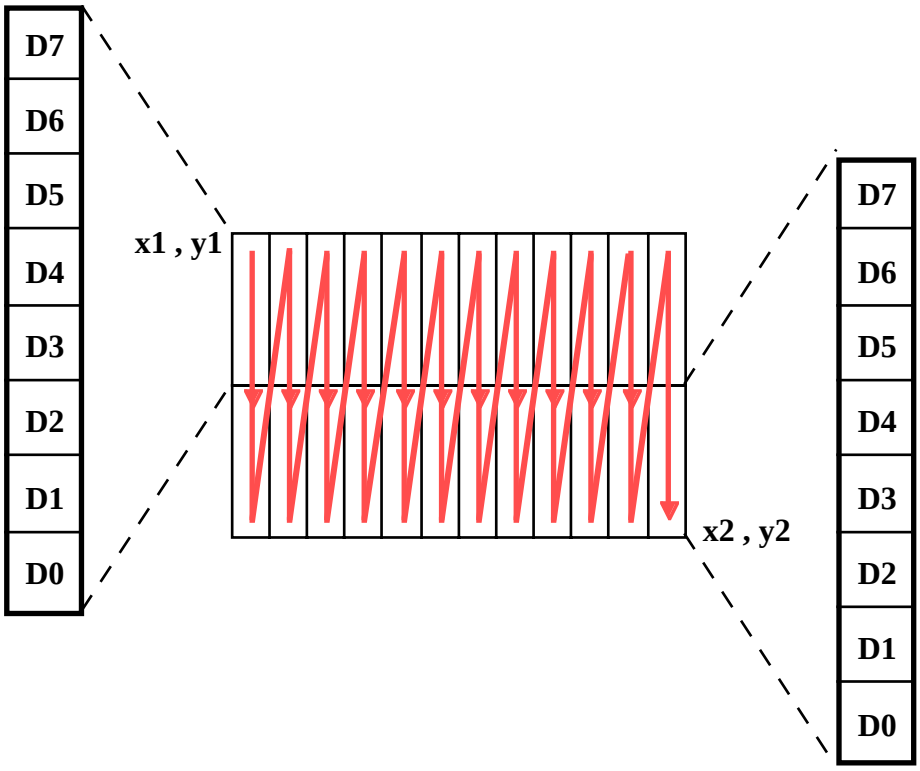


FIGURE 56: VERTICAL DATA AND HORIZONTAL SHIFT

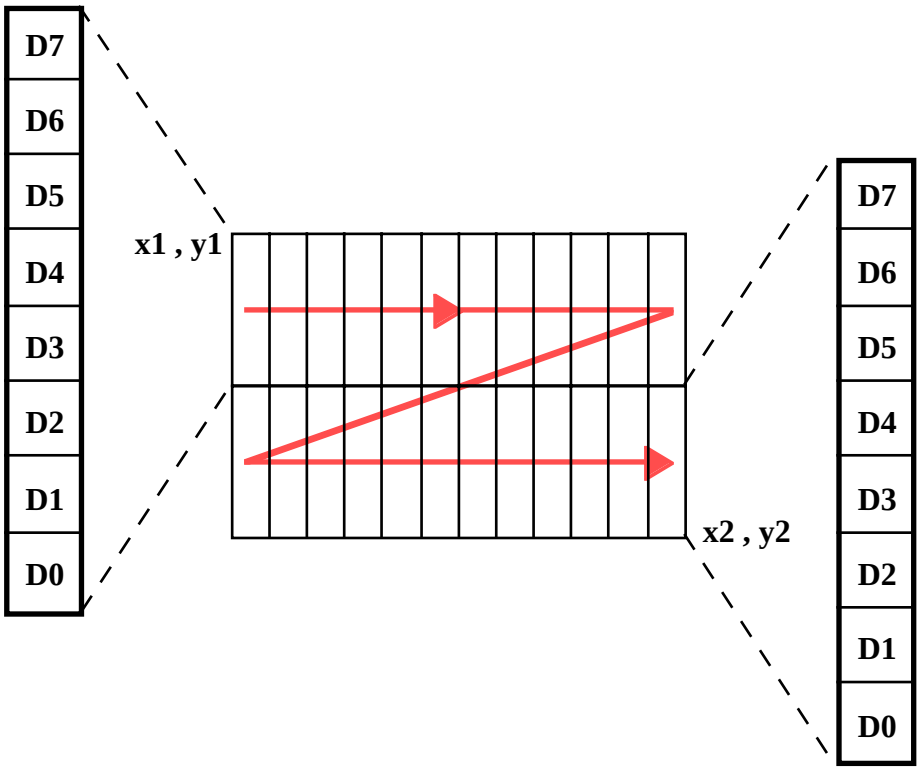


FIGURE 57: VERTICAL DATA AND VERTICAL SHIFT

GRAPHIC FONT SETTING

Code: ***27 242 font***
Hex Code: ***1B F2 font***
Mnemonic: ***ESC ASCII(242) ASCII(font)***

Selects the font for next alphanumeric characters visualization, in graphic mode.
When the graphic mode is selected and a graph area drawing command is not under exectution (as already stated in this conditions the received bytes are used ad graph data), the received byte are shown on display as characters, anyway. In this mode the characters font can be selected, and it is different from the one in pure alphanumeric mode.
The characters font for text displayed in graphic mode can be selected with parameter **font**, as below described

font	=	*65*	*(41 Hex)*	*A*	*-> Proportional spacing minifont 3x5÷5x5 pixels.*
		97	*(61 Hex)*	*a*	*-> Proportional spacing minifont 3x5÷5x5 pixels.*
		66	*(42 Hex)*	*B*	*-> Katakana font 5x7 pixels.*
		67	*(43 Hex)*	*C*	*-> Katakana font 10x14 pixels.*
		98	*(62 Hex)*	*b*	*-> Europe font 5x7 pixels.*
		99	*(63 Hex)*	*c*	*-> Europe font 10x14 pixels.*
		49	*(31 Hex)*	*1*	*-> 1 pixel line spacing.*
		50	*(32 Hex)*	*2*	*-> 2 pixels line spacing.*

The first six font selection commands are mutually exclusive, while the line spacing selections add their effect to the font selection commands. So, each of the five fonts can be set with line spacing of 1 or 2 pixels, obtaining 10 different fonts.
This selection is valid only in graphic mode, in alphanumeric mode only the classic font shown in figure B1, with line spacing 1 pixel, is used.
After a power on alphanumeric mode is enalbed by default, while for graphic mode the font Katakana 5x7, with a line spacing of 1 pixel, is automatically enabled.
For further information about available characters with described fonts please refer to appendix B, while figure 36 shows three of the ten avaialble fonts.

APPENDIX A: COMMANDS SUMMARY TABLES

The following tables list a summary of all the commands recognized by **QTP 12** firmware. Please remind that these commands are compatible with ADDS Viewpoint standard. As in all the other descriptions of the manual, the codes are reported in three formats: decimal, hexadecimal and mnemonic, while the last column reports the number of data returned by command as response.

COMMAND	CODE	HEX CODE	MNEMONIC	Ret.
Home	01	01	SOH	0
Cursor left	21	15	NACK	0
Cursor right	06	06	ACK	0
Cursor down	10	0A	LF	0
Cursor up	26	1A	SUB	0
Carriage return	13	0D	CR	0
Carraige return+line feed	29	1D	GS	0
Alphanumeric cursor absolute position	27 89 r c	1B 59 r c	ESC Y ASCII(r) ASCII(c)	0
Back space	08	08	BS	0
Clear page	12	0C	FF	0
Clear line	25	19	EM	0
Clear end of line	27 75	1B 4B	ESC K	0
Clear end of page	27 107	1B 6B	ESC k	0
Cursor off	27 80	1B 50	ESC P	0
Steady cursor on	27 79	1B 4F	ESC O	0
Blinkling block cursor on	27 81	1B 51	ESC Q	0
Reading of version number	27 86	1B 56	ESC V	3
Reading card code	27 160	1B A0	ESC ASCII(160)	1
Operating mode selection	27 65 mode	1B 41 mode	ESC A ASCII(mode)	0
General reset	27 162	1B A2	ESC ASCII(162)	0
Communication reset	27 163	1B A3	ESC ASCII(163)	0
Fluorescent display brightness setting	27 108 lum	1B 6C lum	ESC l ASCII(lum)	0
Beep	07	07	BEL	0
LEDs, Buzzer activation	27 50 dev attr	1B 32 dev attr	ESC 2 ASCII(dev) ASCII(attr)	0

FIGURE A1: COMMAND CODES SUMMARY TABLE (1 OF 4)

COMMAND	CODE	HEX CODE	MNEMONIC	Ret.
Request of EEPROM availability	27 51	1B 33	ESC 3	1
Writing presence byte	27 33 78 byte	1B 21 4E byte	ESC ! N ASCII(byte)	0
Reading presence byte	27 33 110	1B 21 6E	ESC ! n	1
Write byte on EEPROM	27 164 addl addh byte	1B A4 addl addh byte	ESC ASCII(164) ASCII(addl) ASCII(addh) ASCII(byte)	0
Read byte from EEPROM	27 165 addl addh	1B A5 addl addh	ESC ASCII(165) ASCII(addl) ASCII(addh)	1
Key code reconfiguration	27 55 key n. cod.	1B 37 key n. cod.	ESC 7 ASCII(key n.) ASCII(cod.)	0
Keyclick on without memorization	27 53	1B 35	ESC 5	0
Keyclick off without memorization	27 54	1B 36	ESC 6	0
Keyclick on with memorization	27 33 53	1B 21 35	ESC ! 5	0
Keyclick off with memorization	27 33 54	1B 21 36	ESC ! 6	0
Definition of user character	27 66 nchar Pat0÷Pat7	1B 42 nchar Pat0÷Pat7	ESC B ASCII(nchar) ASCII(Pat0)÷ASCII(Pat7)	0
Definition and memorization of user character	27 33 66 nchar Pat0÷Pat7	1B 21 42 nchar Pat0÷Pat7	ESC ! B ASCII(nchar) ASCII(Pat0)÷ASCII(Pat7)	0
Write digital output	27 166 out	1B A6 out	ESC ASCII(166) ASCII(out)	0
Enable single digital output	27 168 bit	1B A8 bit	ESC ASCII(168) ASCII(bit)	0
Disable single digital output	27 170 bit	1B AA bit	ESC ASCII(170) ASCII(bit)	0

FIGURE A2: COMMAND CODES SUMMARY TABLE (2 OF 4)

COMMAND	CODE	HEX CODE	MNEMONIC	Ret.
Reading of max message number	27 110	1B 6E	ESC n	1
Reading of last group and message managed	27 33 109	1B 21 6D	ESC ! m	2
Select current message group	27 33 77 grp	1B 21 4D grp	ESC ! M grp	0
Message storage	27 33 67 mess.n. chr.0÷chr.19	1B 21 43 mess.n. chr.0÷chr.13	ESC ! C ASCII(mess.n.) ASCII(chr.0)÷ASCII(chr19)	0
Message reading	27 33 69 mess.n.	1B 21 45 mess.n.	ESC ! E ASCII(mess.n.)	20
Visualization of n messagges	27 33 68 mess.n. n	1B 21 44 mess.n. n	ESC ! D ASCII(mess.n.) ASCII(n)	0
Scrolling messages visualization	27 33 83 mess.n. n.chr	1B 21 53 mess.n. n.chr	ESC ! S ASCII(mess.n.) ASCII(n.chr)	0
Set automatic visualization	27 150 255 mess.n. len shift r c	1B 96 FF mess.n. len shift r c	ESC ASCII(150) ASCII(255) ASCII(mess.n.) ASCII(len) ASCII(shift) ASCII(r) ASCII(c)	0
Start I2CBUS	27 250	1B FA	ESC ASCII(250)	0
Stop I2CBUS	27 251	1B FB	ESC ASCII(251)	0
Transmit byte on I2CBUS	27 252 byte	1B FC byte	ESC ASCII(252) ASCII(byte)	1
Receive byte from I2CBUS	27 253 ack	1B FD ack	ESC ASCII(253) ASCII(ack)	1

FIGURE A3: COMMAND CODES SUMMARY TABLE (3 OF 4)

COMMAND	CODE	HEX CODE	MNEMONIC	Ret.
Write byte on backed SRAM	27 33 71 addr byte	1B 21 47 addr byte	ESC ! G ASCII(addr) ASCII(byte)	0
Read byte from backed SRAM	27 33 103 addr	1B 21 67 addr	ESC ! g ASCII(addr)	1
Set clock	27 33 70 hou min sec day mon yea wee	1B 21 46 hou min sec day mon yea wee	ESC ! F ASCII(hou) ASCII(min) ASCII(sec) ASCII(day) ASCII(mon) ASCII(yea) ASCII(wee)	0
Acquire clock	27 33 102	1B 21 6	ESC ! f	7
Show time	27 33 116 r c frm	1B 21 74 r c frm	ESC ! t ASCII(r) ASCII(c) ASCII(frm)	0
Show data	27 33 100 r c frm	1B 21 64 r c frm	ESC ! d ASCII(r) ASCII(c) ASCII(frm)	0
Set clock alarm	27 33 65 hou min sec day mon dton	1B 21 41 hou min sec day mon dton	ESC ! A ASCII(hou) ASCII(min) ASCII(sec) ASCII(day) ASCII(mon) ASCII(dton)	0
Acquire clock alarm	27 33 97	1B 21 61	ESC ! a	6
Graphic cursor absolute position	27 206 y x 0	1B CE y x 0	ESC ASCII(206) ASCII(y) ASCII(x) NUL	0
Alphanumeric mode setting	27 208	1B D0	ESC ASCII(208)	0
Graphic mode setting	27 209	1B D1	ESC ASCII(209)	0
Graphic area setting	27 241 x1 y1 x2 y2 cmd	1B F1 x1 y1	ESC ASCII(241) ASCII(x1) ASCII(y1) ASCII(x2) ASCII(y2) ASCII(cmd)	0
Graphic font setting	27 242 font	1B F2 font	ESC ASCII(242) ASCII(font)	0

FIGURE A4: COMMAND CODES SUMMARY TABLE (4 OF 4)

APPENDIX B: DISPLAY CHARACTERS

The following tables show the characters sets displayed on **QTP 12** for all the possible received codes, according with ordered display, and so model, and according with functionality mode preselected through proper commands.
Even the not ASCII characters (or special characters) change when the display type changes; if the user requires a characters set different from those described in the following figures, he can directly contact **grifo®**.

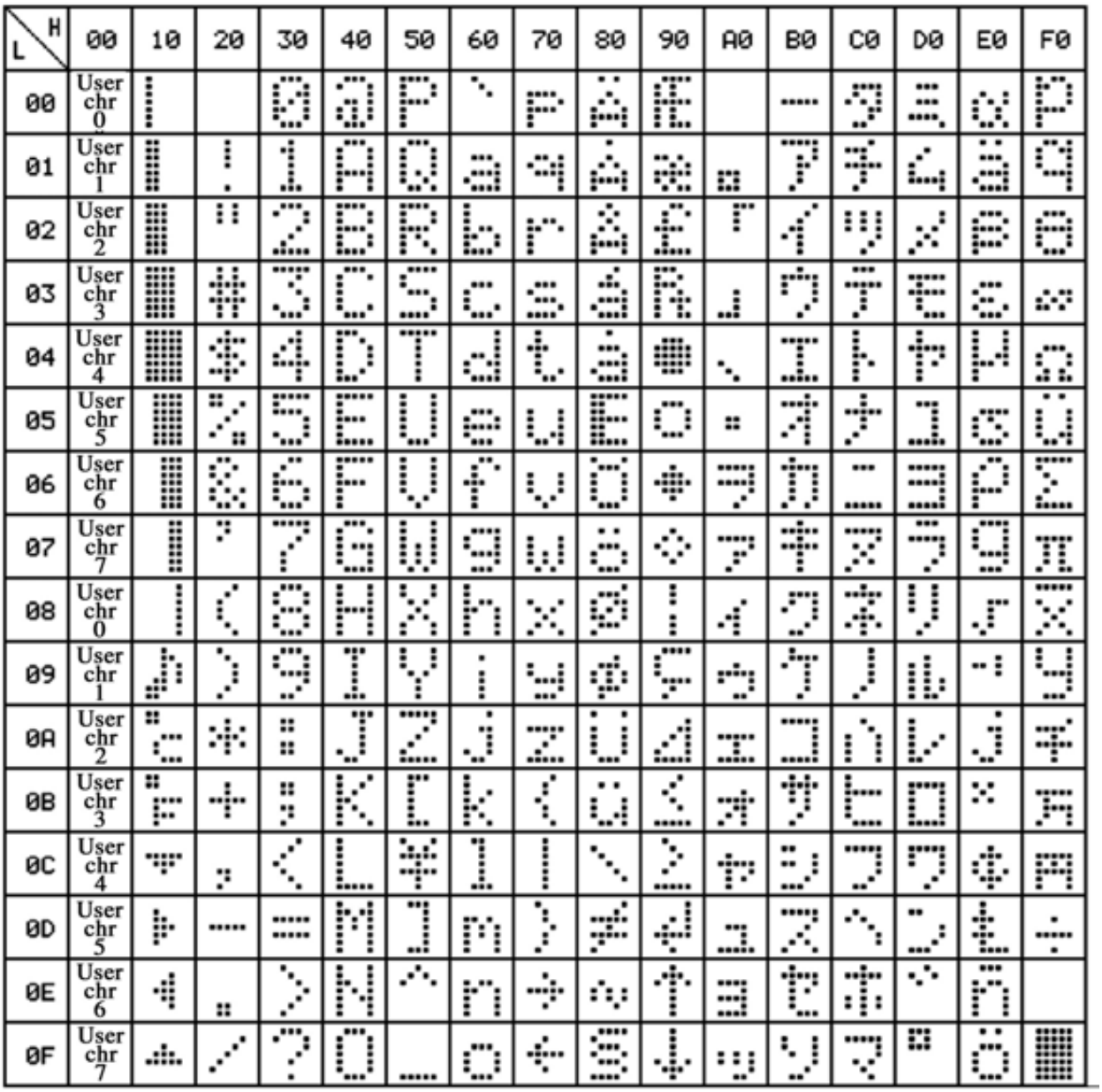


FIGURE B1: QTP 12-F2, GF2 IN ALPHANUMERIC MODE CHARACTERS TABLE

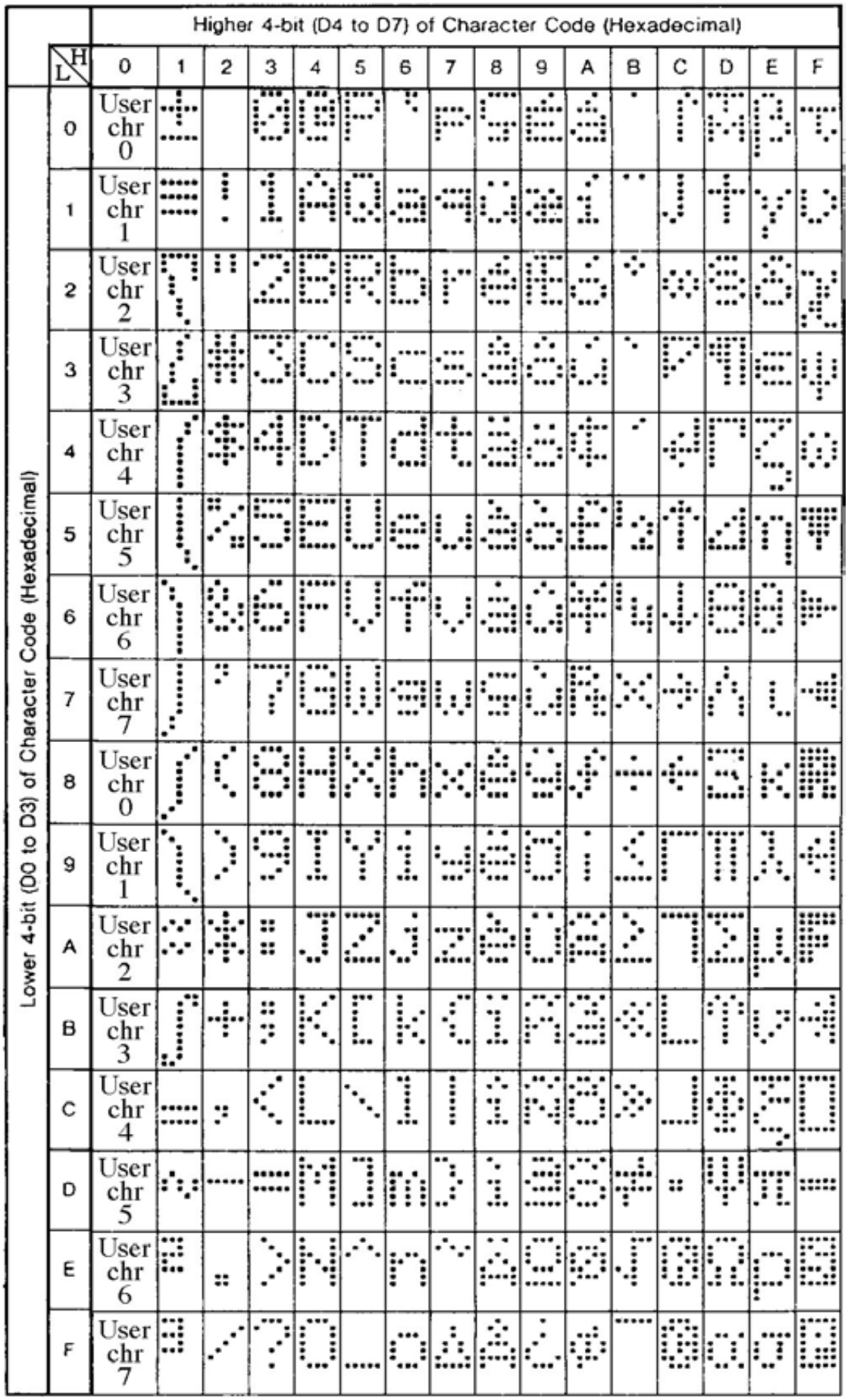


FIGURE B2: QTP 12-C2 CHARACTERS TABLE

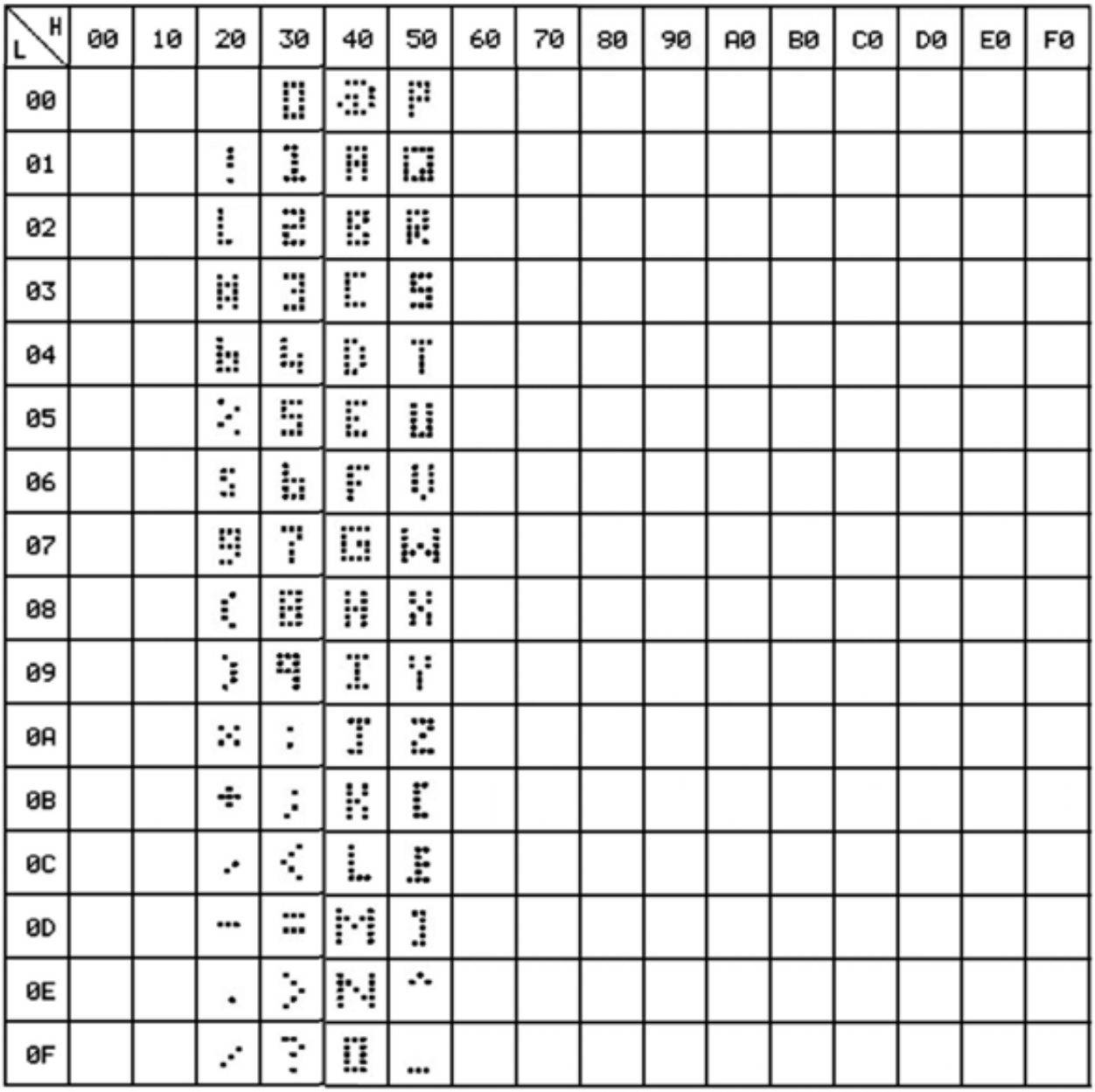


FIGURE B3: QTP 12-GF2 MINIFONT IN GRAPHIC MODE CHARACTERS TABLE

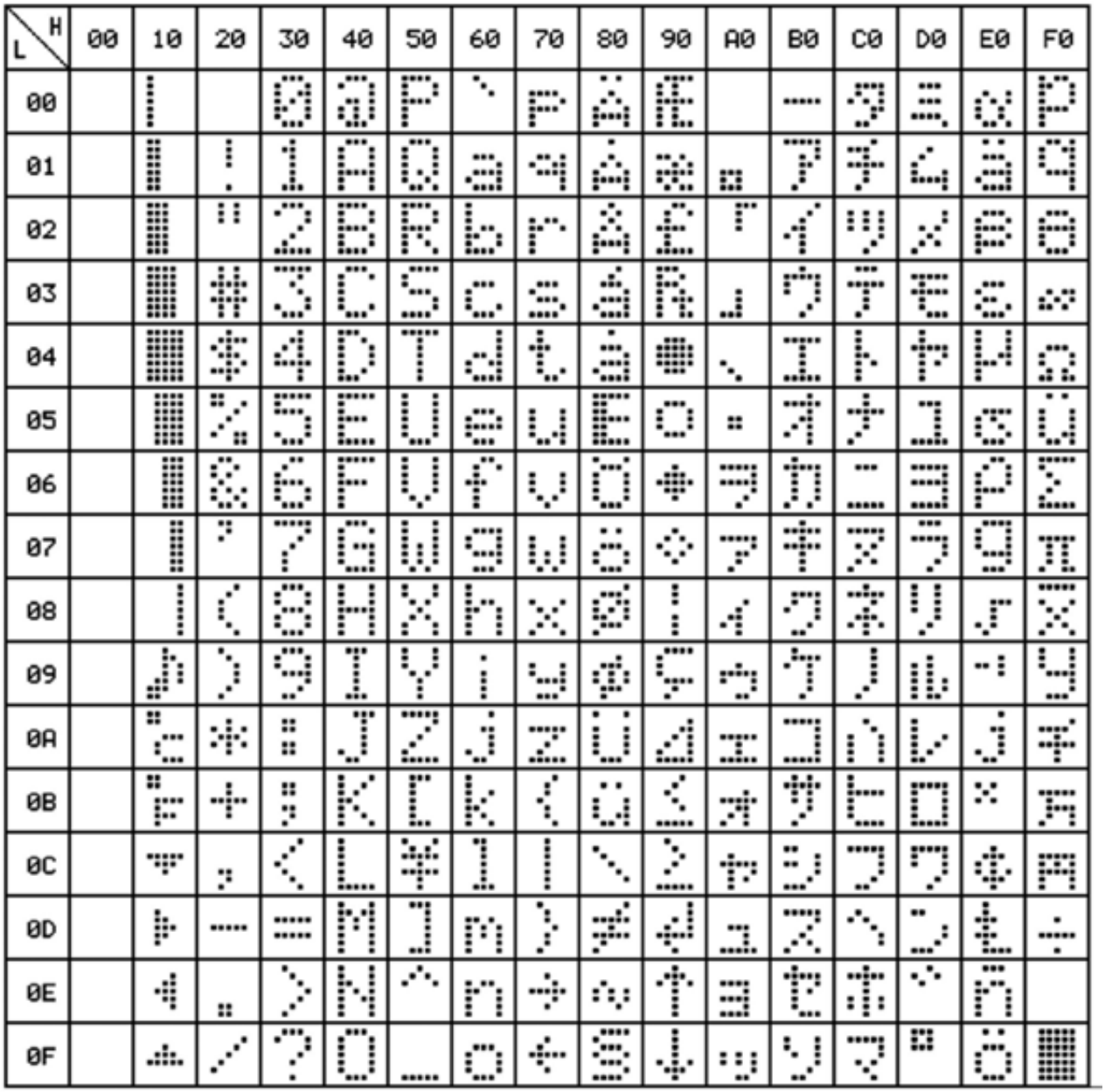


FIGURE B4: QTP 12-GF2 FONT KATAKANA IN GRAPHIC MODE CHARACTERS TABLE

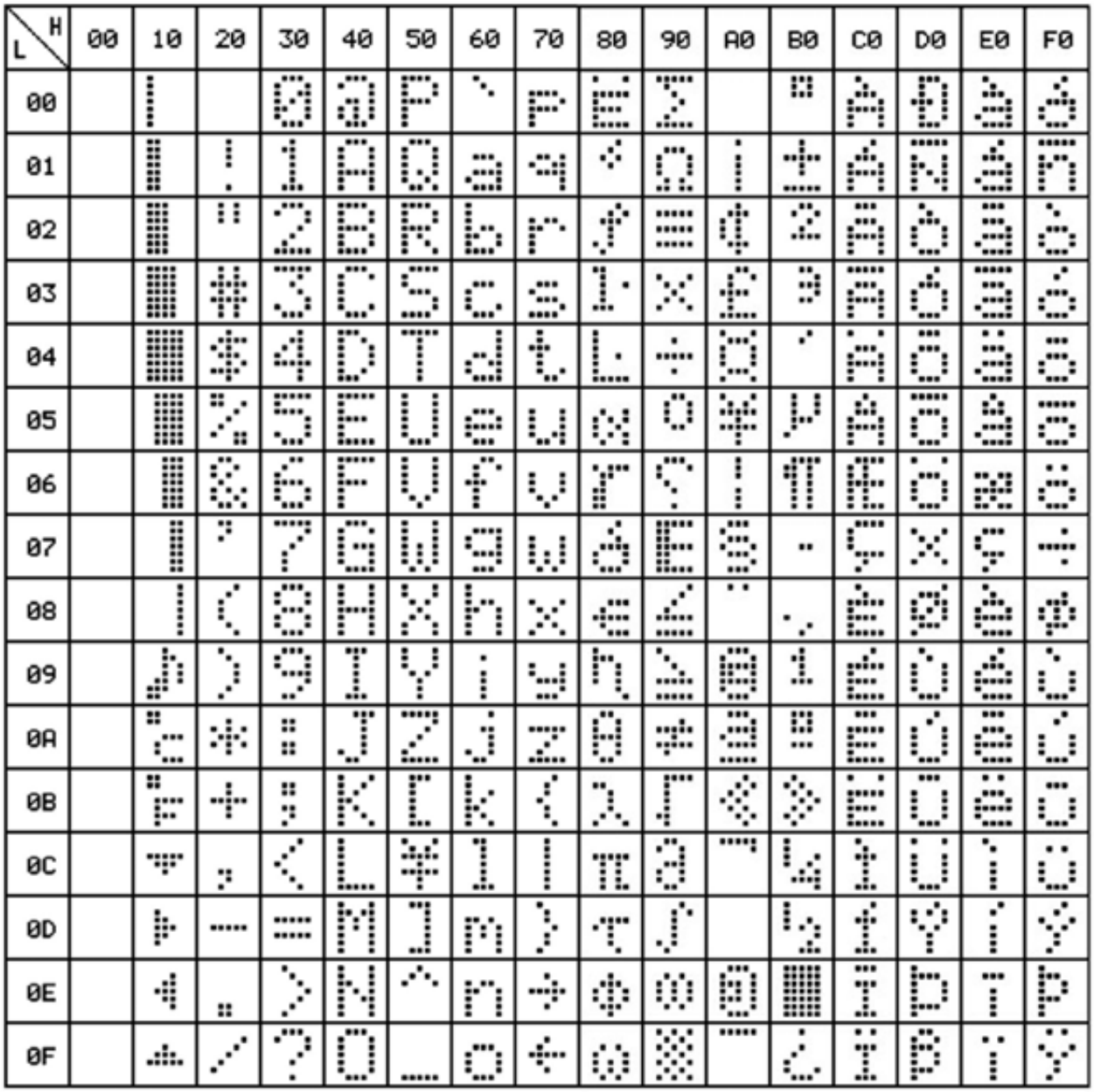


FIGURE B5: QTP 12-GF2 FONT EUROPEAN IN GRAPHIC MODE CHARACTERS TABLE

APPENDIX C: MOUNTING NOTES

QTP 12 is provided complete of metallic container and some additional parts that simplify the mounting. Inside this appendix there are all the information concerning this operations, together with the instructions to open the container and to personalize it.

TERMINAL DIMENSIONS

In the following figure there are dimensions of terminal **QTP 12** complete of external metallic container, attached frontal plastic frame and mounting clamps. Dimensions are in **mm** and the drawing is in scale.

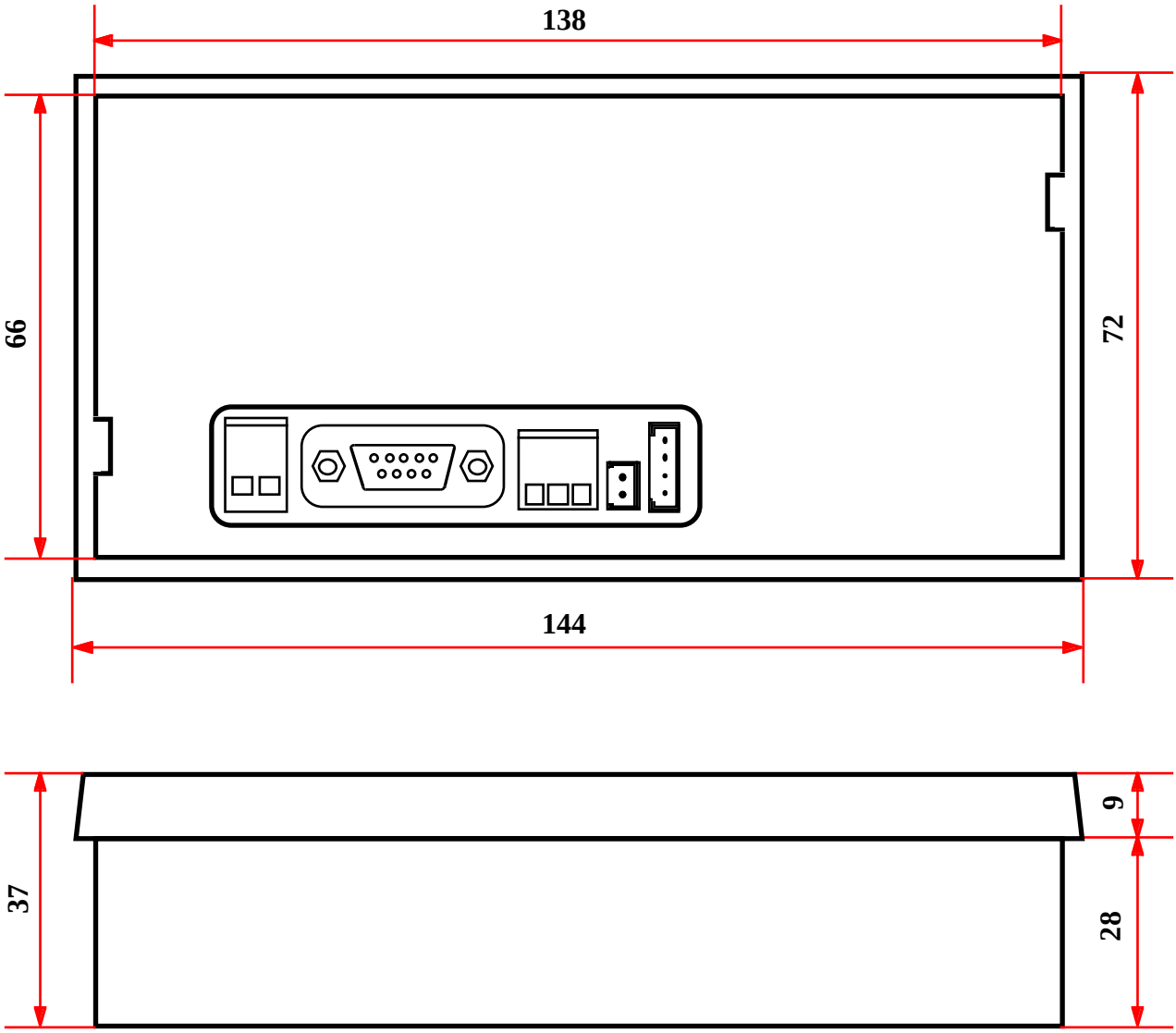


FIGURE C1: QTP 12 DIMENSIONS

The dimensions of previous figure refer to container only, but occupied area can be slightly greater by considering also mounting clamps and screws, described in following figures, up to a maximum of 156 x 72 x 80 mm (W x H x D).

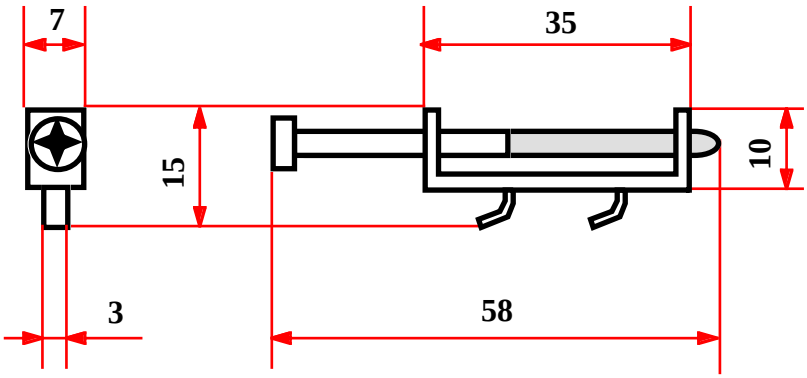


FIGURE C2: MOUNTING CLAMP DDIMENSIONS

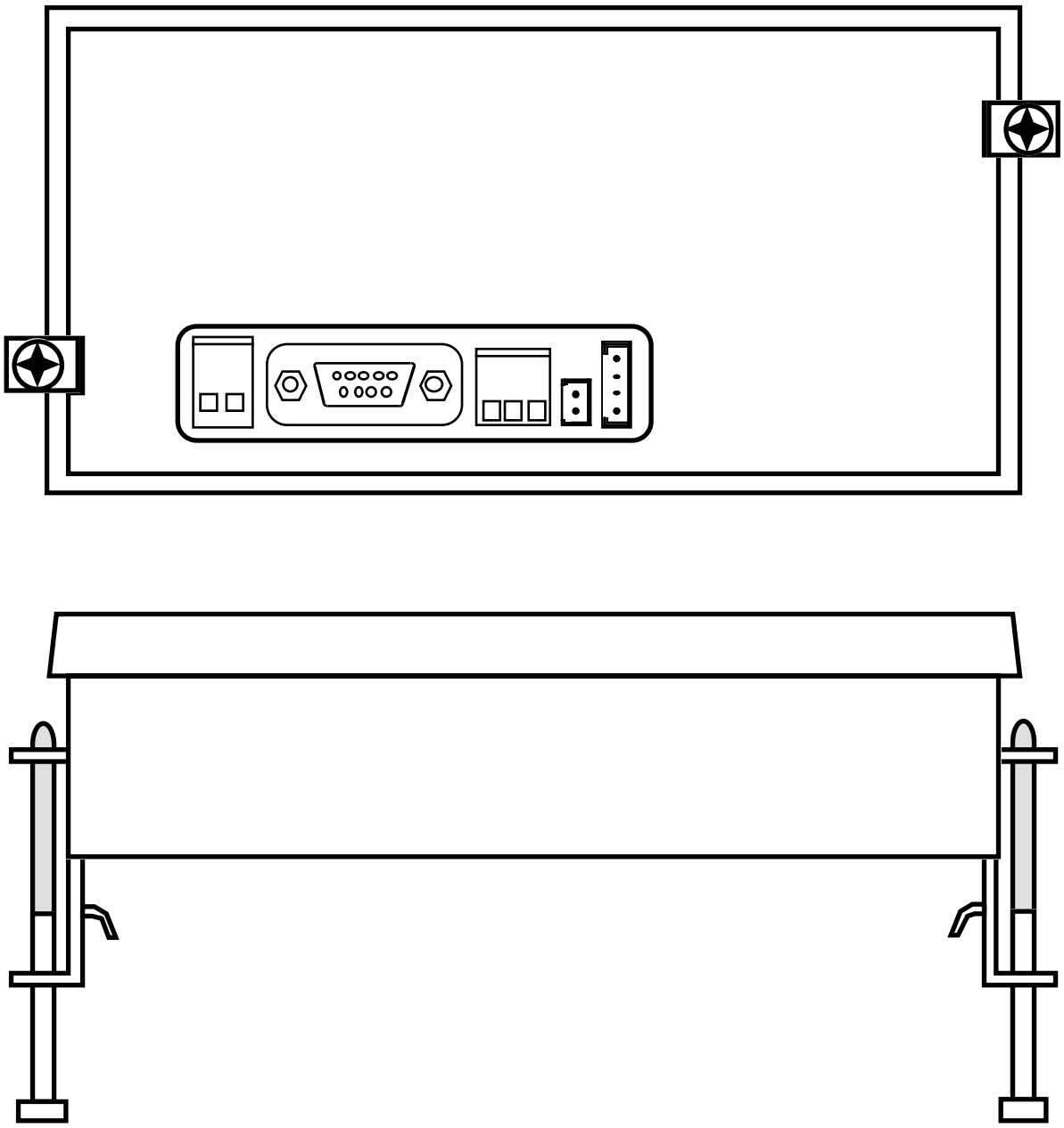


FIGURE C3: QTP 12 + MONTING CLAMP VIEW

FRONT PANEL MOUNTING

The provided mounting mode is the front panel one that is possible on any panel with 10 mm maximum thickness and fixing is done by two clamps provided with **QTP 12**.
Installation operations are exteremly easy and they are below summarized:

1) make a rectangular breaking on mounting panel like those in the following figure;

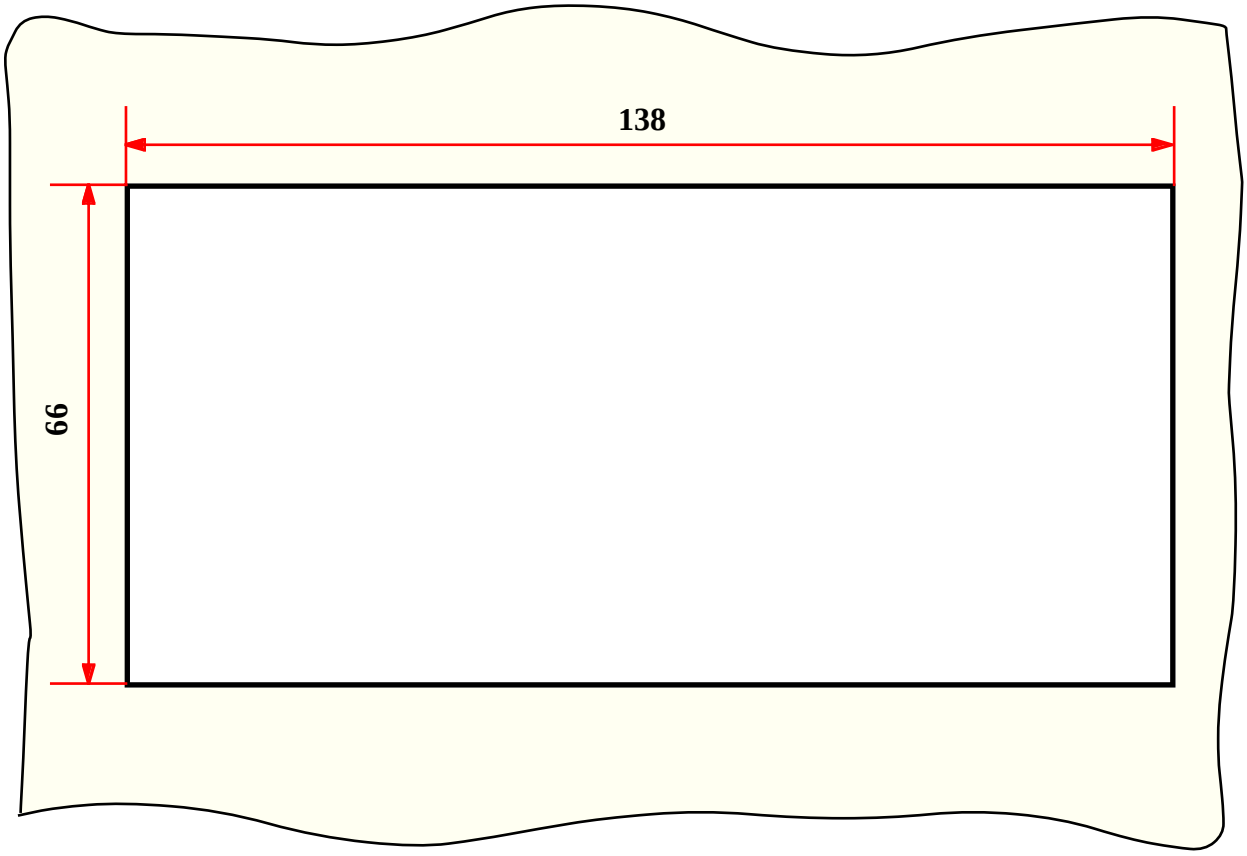


FIGURE C4: BREAKING FOR INSTALLATION

2) screw the two screws on the two **C** clamps, keeping the sharpened part close to the screw cut hole of clamp;

3) insert **QTP 12** in the breaking made at point 1;

4) dock the two clamps prepared at point 2 to the specific side breakings of **QTP 12** container, taking care that the first hook of the clamp, the one near screw-cut hole, enters correctly in the proper lateral buttonhole of the containter (figure C3 shows the result of these instructions);

5) screw the two clamps until the **QTP 12** container is firmly docked to mounting panel;

6) insert the connettors on the back side.

PERSONALIZATION LABEL INSERTION

Frontal of **QTP 12** is provided with a pocket where the user can insert a personalization label with his logo, an identification code, the terminal function, or anything else.
If the label is required please inser it before mounting **QTP**. Label must be thin but rather rigid, for example made of 160 g/m^2 paper or polyester or polycarbonate sheets. Here follow the suggested dimensions, in millimeters, of personalization label; please nothe that the white zone is the area contained in the transparent window, or in other words, the visible part:

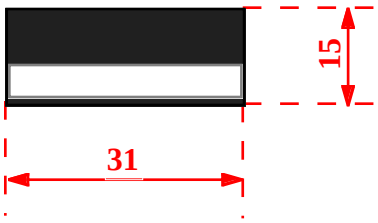


FIGURE C5: PERSONALIZATION LABEL DIMENSIONS

Here follow the operations required to insert personalization label inside the **QTP 12**:

1) Unscrew the two black screws on frontal panel (if present).

2) Separate the group metallic carter + plastic frame from the group front panel + printed circuit. A simple pressure on **QTP 12** connectors, or on the printed circuit always from the backside connectors window, it is sufficient.

3) Now the front panel is ready to insert the personalization label; this latter must be inserted from the bottom side, using the specific pocket located on the back of front panel, as shown in following figure. As described on figure C5, lenght of label must be greater than height of window to simplify the insertion and extraction.

4) Remount terminal **QTP 12**, following the previous steps in reversed order.

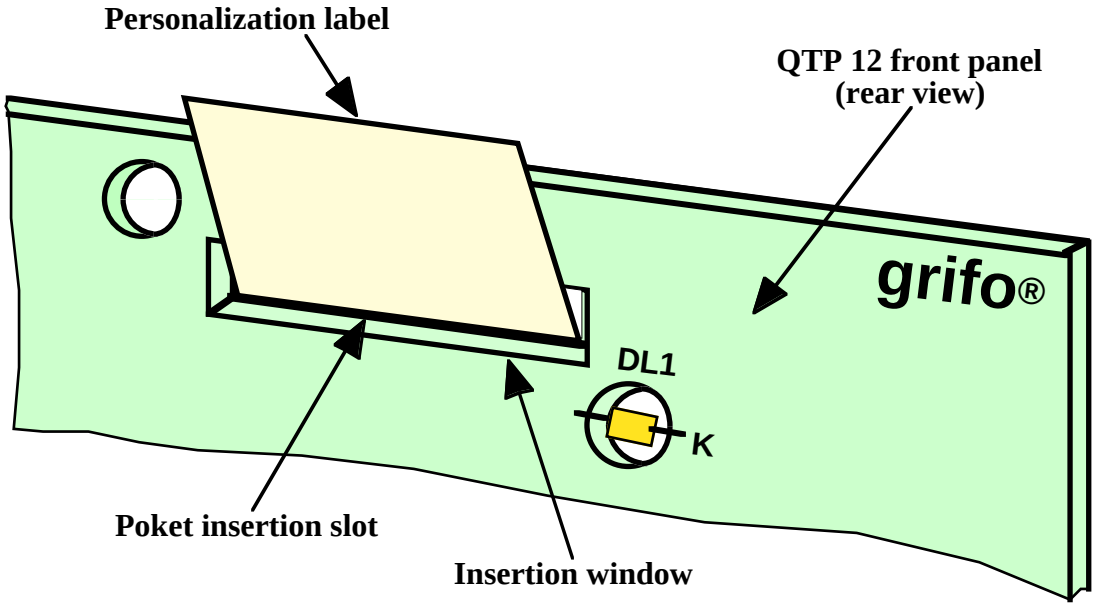


FIGURE C6: PERSONALIZATION LABEL INSERTION

FIXING FRONT PANEL TO CONTAINER

QTP 12 by default is provided with front panel (keyboard+printed circuit board) jointed in plastic frame of the back metallic container. Terminal anyway allows a better mechanical docking between this two groups, by using two specific screws; this avoid accidental separations of front panel.

Here follows the operations that must be performed, to ensure such docking:

1) Separate the group metallic carter + plastic frame from the group front panel + printed circuit. A simple pressure on backside **QTP 12** connectors, or on the printed circuit always from the backside connectors window, is normally sufficient.

2) Of the six screws, that dock the plastic frame to the back metallic container, unscrew the two central ones.

3) On front panel, in correspondence with these central screws, there are two holes provided with flare, visible only from the back side. It is sufficient to perforate the polyester layer that covers the frontal, to make these holes accessible.

4) Remount everything, using the same screws removed at point **2**, that will be screwed on the front panel with keyboard and not on the plastic frame any more.

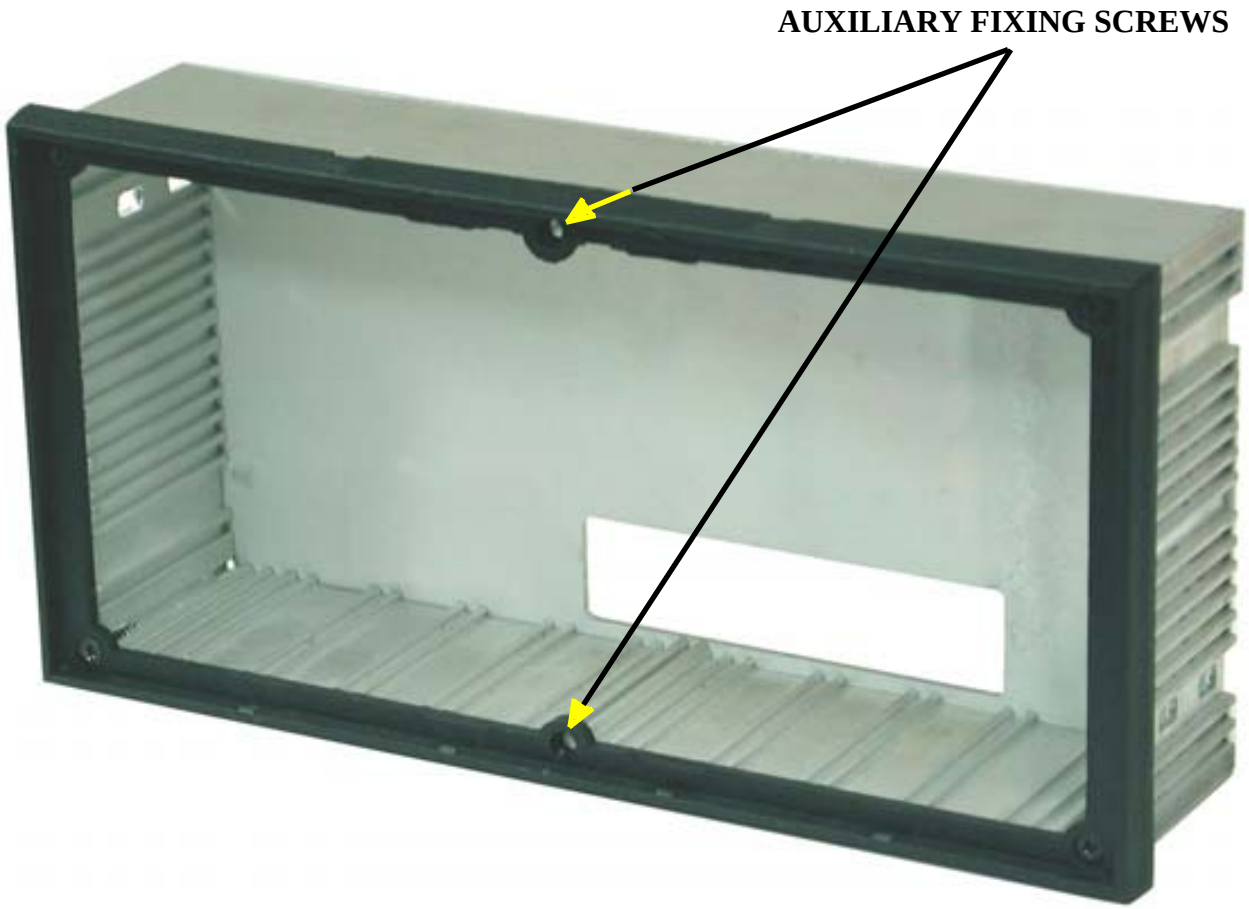


FIGURE C7: SCREWS FOR FRONT PANEL FIXING

APPENDIX D: VIEW AREA AND CHARACTERS DIMENSIONS

The following figures report the dimensions expressed in mm and in scale of the three display models used on **QTP 12**. Whenever a larger visible area and/or a greater number of characters are necessary, please remind that are available many other **QTP** models and/or display types; in these conditions it is suggested to contact directly **grifo®**.

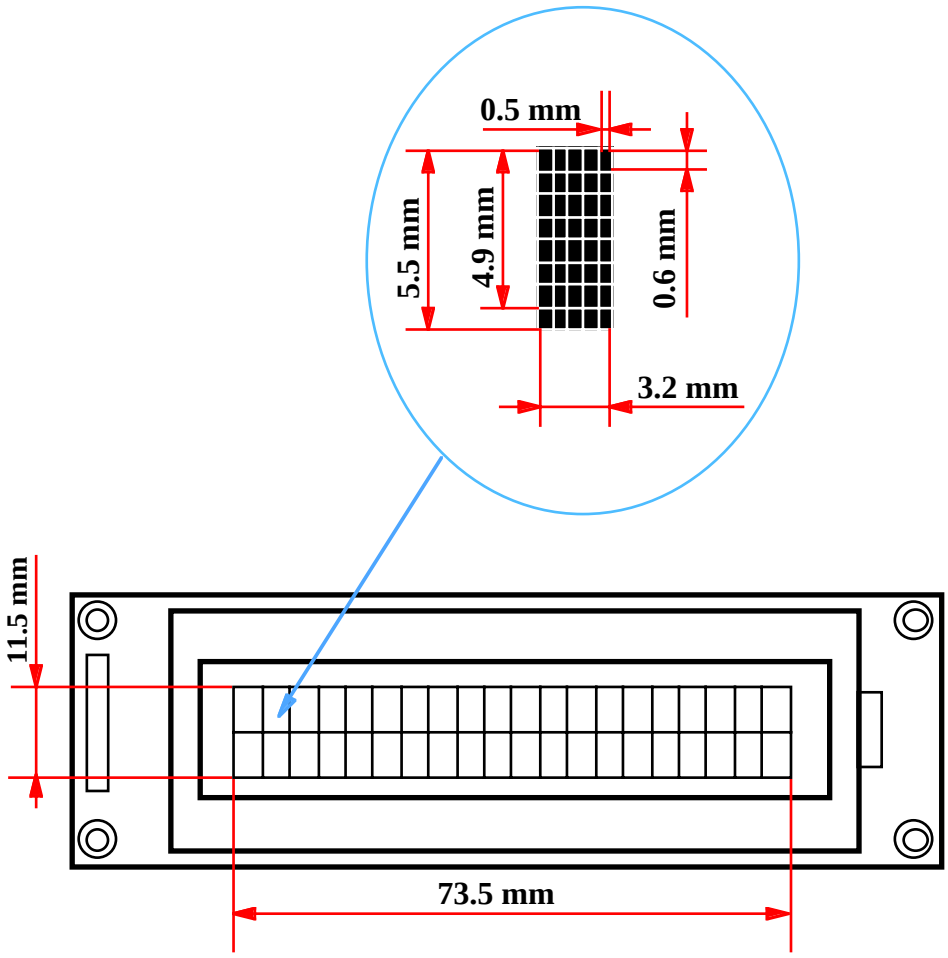


FIGURE D1: DISPLAY DIMENSIONS OF QTP 12-C2

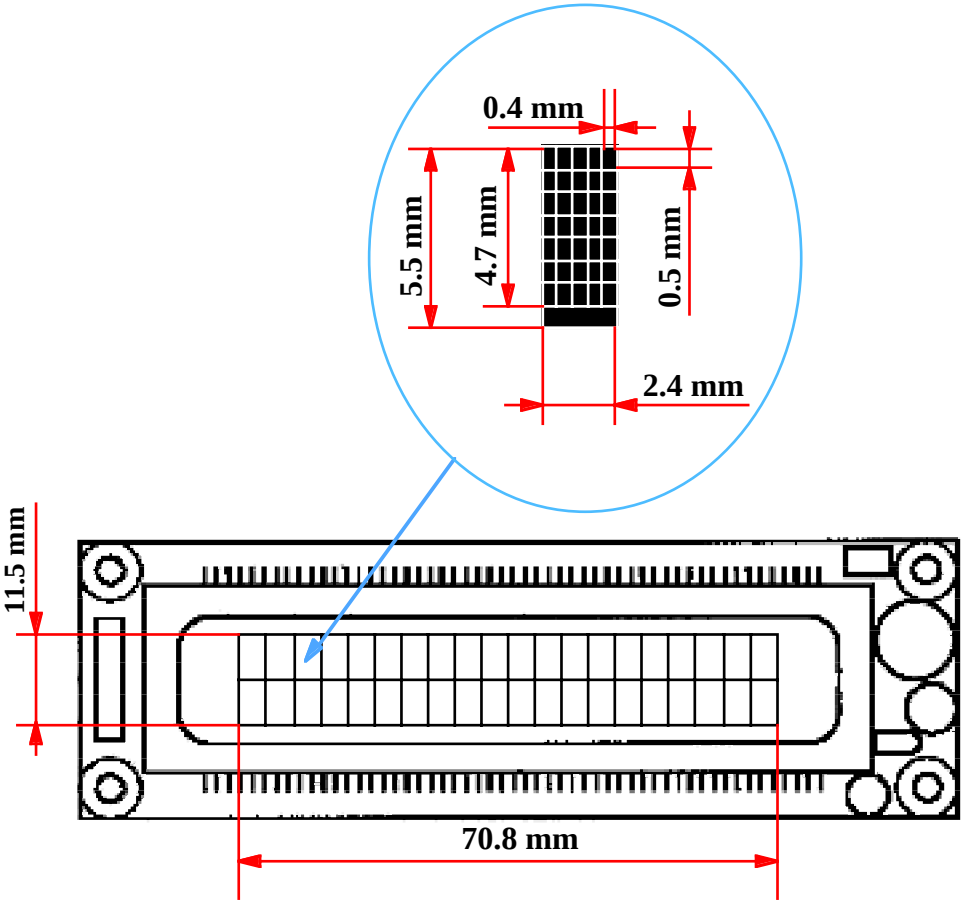


FIGURE D2: DISPLAY DIMENSIONS OF QTP 12-F2

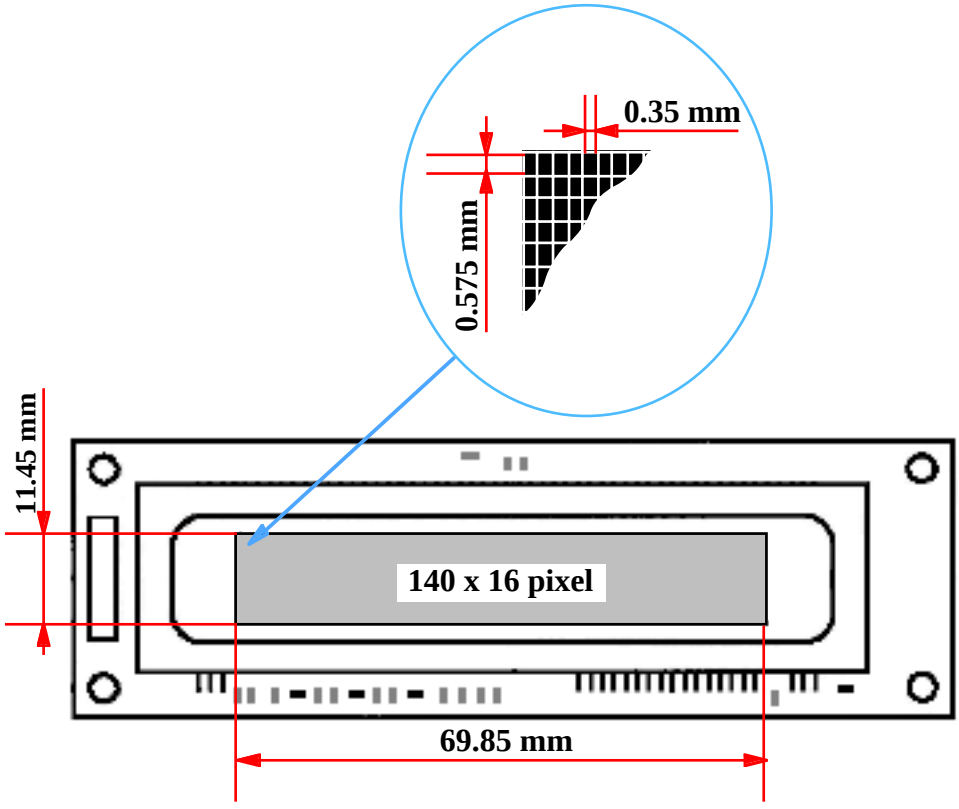


FIGURE D3: DISPLAY DIMENSIONS OF QTP 12-GF2

APPENDIX E: DEFAULT CONFIG., OPTIONS, ACCESSORIES

In corrispondence of the first purchase, or after a reparation, the **QTP 12** is supplied in its base configuration. The features of this configuration has been described many times in the manual (by using also the name default configuration) and in this appendix they are summarized, opportunely divided in the following tables.

PARAMETER	*DEFAULT SETTING*	*FUNCTION*
COMMUNIC.	Norm.	Serial communication on asynchronous serial line in normal mode
BAUD RATE	19200	Communication speeed on asynchronous serial line
BIT x CHR	8	Bits per character on asynchronous serial line
STOP BIT	1	Stop bits on asynchronous serial line
PARITY	None	Parity check on asynchronous serial line
KEY-CLICK	ON	Keyclick enabled when keys are pressed
SLAVE ADD.	80H	**QTP** address for network communication
EE DATA	INIT	Data of base EEPROM, initializzed
INTRTC FN.	USER	/INTRTC signal (connected to printed circuits pods) configured as user digital output

FIGURE E1: LOCAL SETUP DEFAULT CONFIGURATION

The values listed in previous table can be modified through the local setup modality, as described with details in the homonimous paragraph.

JUMPER	*DEFAULT CONNECTION*	*FUNCTION*
J1	position 2-3	Configures serial line for RS 422 standard elettric protocol
J2 , J5	not connected	Do not connect termination and forcing circuitery to RS 422, RS 485 serial line.
J3	not connected	Does not connect 120 Ω termination resistor to CAN line.
J4	not connected	Selects the RUN modality
J6	connected	On board battery BT1 onnected to back up circuitry.
J7	position 1-2	Write protection of optional EEPROM not enabled.

FIGURE E2: JUMPERS DEFAULT CONFIGURATION

Please remind that the jumpers default configuration proposed is the one relative to base version of terminal, that is without any options.

During the order phase the user can add to **QTP 12**, the following features:

OPTION	*DESCRIPTION*
.CAN	CAN communication line
.RS422	Asynchronous serial communication line in RS 422
.RS485	Asynchronous serial communication line in RS 485
.CLOOP	Asynchronous serial communication line in passive Current Loop
.EE128	Additional EEPROM with 16K Bytes size
.EE256	Additional EEPROM with 32K Bytes size
.EE512	Additional EEPROM with 64K Bytes size
.RTC	Section with Real Time Clock and SRAM backed by battery
.5Vdc or .ALIM	Power supply voltage at +5 Vdc

FIGURE E3: OPTIONS TABLE

All these options are described in the pragraphs of the manual that illustrate the functionalities and the use of the same additional features. It is suggested to use the final alphabetical index, placed in following APPENDIX F, to found these paragraphs in a short time.

In addition there are a list of accessories that simplify and speed up the use of the module. Among these ones we remind the following available products:

- **AMP2.Cable** complete connector with 2 coloured wires, 1 metre length;

FIGURE E4: AMP2.CABLE CONNECTION ACCESSORY

- **CKS.AMP2** kit composed by female AMP Mod II 2 pins, plus 2 contacts to crimp;

FIGURE E5: CKS.AMP2 CONNECTION ACCESSORY

These components can be acquired directly from AMP dealers by using P/N 280358 and P/N 182206-2.

- **AMP8.Cable** complete connector with 4 coloured wires, 1 metre length;

FIGURE E6: AMP4.CABLE CONNECTION ACCESSORY

- **CKS.AMP4** kit composed by female AMP Mod II 4 pins, plus 4 contacts to crimp;

FIGURE E7: CKS.AMP4 CONNECTION ACCESSORY

These components can be acquired directly from AMP dealers by using P/N 280359 and P/N 182206-2.

- **EXPS-1** power supply for direct connection to mains voltage at 230 Vac, 50 Hz, that generates an output voltage of 24 Vdc, 300 mA compatible for **QTP 12**. The photo of this accessories is already available in previous pages of manual, on figure 33.

APPENDIX F: ALPHABETICAL INDEX

D