Overall dimension: **72x144 mm** frontal frame size; **9 mm** frontal frame depth; **28 mm** rear metallic housing depth. Remarkably Low Price. Aluminium container with front plastic frame. Front panel with Keyboard and display window protected by anti scratch Polyester cover. Case with rear mounting brackets. Front panel mounting. **IP 54** standard protection on front side. Available with 3 different models of displays: alphanumeric **LCD** and **VFD** with **20x2** chrs. and graphic with **140x16** pixels. Viewing area size: **82x18 mm**. Membrane keypad with **12** keys provided of double serigraphy: numeric and function. Status **LED** indicator managed by software. Panel name, and/or **LED** indicator, personalization label slot. **Buzzer** for acoustic signal driven by software. **151 family microcontroller**, with **14.7** or **29.4 MHz** software selectable clock speed. 3 different memory types: **32K FLASH EPROM; 1.2K RAM; 2K EEPROM**. Serial line configurable in **RS 232, RS 422, RS 485** or **Current Loop.** **CAN** communication line provided of proper line driver. **8** optocoupled digital Inputs, both **NPN** or **PNP** type. Some inputs are connected to hardware counters and Interrupts. **4** relay digital outputs, up to **5 A**. Comfortable quick release, screw terminal connectors for a fast cabling. **2 TTL I/O** lines on proper connector, suitable for **I2C BUS, 1-Wire, SPI, etc.**, devices management. Wide range **DC or AC** power supply from **5 Vdc** to **24 Vac**. Total **power** consumption change according with used configuration, from **2.5 to 3.3 W**. On board protection aganist voltage peaks by **TransZorb™**. Wide range of developement software tools and programming languages, such as: **Assembler; FORTH; PASCAL Compilers (SYS51PW); C Compilers (HTC51, SYS51CW, DDS MICRO C 51, µC/51); Real Time Operating Systems (CMX RTX); Contacts Logic (Ladder WORK); BASIC compilers (BXC51, BASCOM 8051); terminal emulation (GET 51), etc.** Supplied with two different management Firmwares: Serial or Library. Customized keyboard and program packages.
IMPORTANT

Although all the information contained herein have been carefully verified, grifo® assumes no responsibility 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

INTRODUCTION ................................................................................................................... 1

HARDWARE AND FIRMWARE VERSION ........................................................................... 2

GENERAL INFORMATION .................................................................................................... 2

SERIAL LINE ................................................................................................................... 4

INDICATOR LED ............................................................................................................. 6

DISPLAY ......................................................................................................................... 6

KEYBOARD .................................................................................................................... 6

BUZZER ............................................................................................................................ 7

MEMORY DEVICES ........................................................................................................... 7

ON BOARD POWER SUPPLY ........................................................................................... 7

MICROCONTROLLER ....................................................................................................... 8

CAN INTERFACE ............................................................................................................... 8

TTL I/O LINES ................................................................................................................... 8

OPTOCOUPLED DIGITAL INPUTS .................................................................................... 8

RELAYS DIGITAL OUTPUTS ............................................................................................. 8

TECHNICAL FEATURES .................................................................................................... 10

GENERAL FEATURES ..................................................................................................... 10

PHYSICAL FEATURES ..................................................................................................... 11

ELECTRIC FEATURES ..................................................................................................... 12

INSTALLATION ................................................................................................................... 14

CONNECTIONS ................................................................................................................... 14

CN1 - POWER SUPPLY CONNECTOR ........................................................................... 14

CN2 - SERIAL LINE CONNECTOR ............................................................................... 16

CN3 - CAN INTERFACE CONNECTOR ........................................................................... 22

CN4 - RELAYS DIGITAL OUTPUTS CONNECTOR ....................................................... 24

CN5 - OPTOCOUPLED DIGITAL INPUTS CONNECTOR ........................................ 26

J6 - TTL I/O, 1-WIRE, I2C BUS CONNECTOR ............................................................. 28

CONTRAST REGULATION TRIMMER ............................................................................. 30

JUMPERS ................................................................................................................................ 30

SERIAL LINE CONFIGURATION ....................................................................................... 32

CAN INTERFACE CONNECTION ..................................................................................... 34

INTERRUPTS ................................................................................................................... 35

CONNECTOR SIGNALS INTERFACEMENT ...................................................................... 35

POWER SUPPLY .............................................................................................................. 36

SOFTWARE DESCRIPTION ............................................................................................... 38

DATA STORED ON EEPROM .......................................................................................... 38

CHARACTERS VISUALIZATION ON DISPLAY ................................................................... 39

RECEIVE BUFFER ............................................................................................................ 40

KEYBOARD ACQUISITION ............................................................................................... 40

KEYS CODES ................................................................................................................... 40
VISUALIZATION OF N MESSAGES ................................................................. 70
SCROLLING MESSAGES VISUALIZATION .................................................. 71
COMMANDS FOR KEYBOARD MANAGEMENT ........................................... 72
KEY CODE RECONFIGURATION ................................................................. 72
KEYCLICK ON WITHOUT MEMORIZATION ............................................... 72
KEYCLICK OFF WITHOUT MEMORIZATION ............................................. 72
KEYCLICK ON WITH MEMORIZATION ..................................................... 73
KEYCLICK OFF WITH MEMORIZATION .................................................... 73
COMMANDS FOR USER CHARACTERS ..................................................... 74
DEFINITION OF USER CHARACTER ....................................................... 75
DEFINITION AND MEMORIZATION OF USER CHARACTER .................... 75
COMMANDS FOR CURSOR ATTRIBUTES MANAGEMENT ....................... 76
CURSOR OFF ............................................................................................ 76
BLINKING BLOCK CURSOR ON ............................................................. 76
COMMANDS FOR BUFFERED DIGITAL I/OS MANAGEMENT .................... 77
WRITE ALL DIGITAL OUTPUTS ............................................................... 77
ACQUIRE ALL DIGITAL INPUTS ............................................................... 77
ENABLE SINGLE DIGITAL OUTPUT ......................................................... 78
TIMED ENABLE OF SINGLE DIGITAL OUTPUT ....................................... 78
DISABLE SINGLE DIGITAL OUTPUT ....................................................... 79
TIMED DISABLE OF SINGLE DIGITAL OUTPUT ....................................... 79
ACQUIRE SINGLE DIGITAL INPUT .......................................................... 80
ACQUIRE SINGLE DIGITAL INPUT WITH DEBOUNCING ....................... 80
TIMED SQUARE WAVE STARTING WITH "1" ON SINGLE DIGITAL OUTPUT ... 81
TIMED SQUARE WAVE STARTING WITH "0" ON SINGLE DIGITAL OUTPUT ... 82
SQUARE WAVE ON SINGLE DIGITAL OUTPUT ....................................... 83
COMMANDS FOR ALTERNATIVE FUNCTIONS OF DIGITAL I/OS ............... 84
SET MULTIFUNCTIONS PINS .................................................................. 84
READ MULTIFUNCTIONS PINS SETTING ................................................. 84
READ COUNTER ..................................................................................... 85
RESET COUNTER .................................................................................. 85
GRAPHIC COMMANDS ........................................................................... 86
ALPHANUMERIC MODE SETTING .......................................................... 86
GRAPHIC MODE SETTING .................................................................... 88
GRAPHIC CURSOR ABSOLUTE POSITION ........................................... 88
GRAPHIC AREA SETTING ..................................................................... 88
GRAPHIC FONT SETTING ..................................................................... 92
COMMAND FOR 1-WIRE COMMUNICATION ........................................... 93
RESET 1-WIRE LINE .............................................................................. 93
WRITE BIT ON 1-WIRE LINE ............................................................... 94
READ BIT FROM 1-WIRE LINE ............................................................ 94
WRITE N BYTES ON 1-WIRE LINE ...................................................... 95
READ N BYTES FROM 1-WIRE BUS .................................................... 95
"READ ROM" ON 1-WIRE LINE ............................................................ 96
"SKIP ROM" ON 1-WIRE LINE .............................................................. 96
"MATCH ROM" ON 1-WIRE LINE .......................................................... 97
"ALARM SERCH" ON 1-WIRE LINE ...................................................... 98
FIGURES INDEX

FIGURE 1: BLOCKS DIAGRAM ................................................................. 3
FIGURE 2: PHOTO OF AVAILABLE MODELS ........................................... 5
FIGURE 3: QTP 12/R84 COMPLETE PHOTO ........................................... 9
FIGURE 4: CONSUMPTIONS TABLE ........................................................ 12
FIGURE 5: QTP 12/R84-C2 PHOTO ....................................................... 13
FIGURE 6: QTP 12/R84-F2 PHOTO ....................................................... 13
FIGURE 7: QTP 12/R84-GF2 PHOTO .................................................... 13
FIGURE 8: CN1 - POWER SUPPLY CONNECTOR .................................... 14
FIGURE 9: AC POWER SUPPLY CONNECTION 8 + 24 Vac ±5% ............ 15
FIGURE 10: DC POWER SUPPLY CONNECTION +10 + 40 Vdc ±5% ...... 15
FIGURE 11: STABILIZED POWER SUPPLY CONNECTION +5 Vdc ±5% ... 15
FIGURE 12: CN2 - SERIAL LINE CONNECTOR ......................................... 16
FIGURE 13: SERIAL COMMUNICATION DIAGRAM ................................... 17
FIGURE 14: RS 232 POINT TO POINT CONNECTION EXAMPLE ............ 18
FIGURE 15: RS 422 POINT TO POINT CONNECTION EXAMPLE ............ 18
FIGURE 16: RS 485 POINT TO POINT CONNECTION EXAMPLE ............ 18
FIGURE 17: RS 485 NETWORK CONNECTION EXAMPLE .......................... 19
FIGURE 18: CURRENT LOOP 4 WIRES POINT TO POINT CONNECTION EXAMPLE ............................................. 20
FIGURE 19: CURRENT LOOP 2 WIRES POINT TO POINT CONNECTION EXAMPLE ............................................. 20
FIGURE 20: CURRENT LOOP NETWORK CONNECTION EXAMPLE ........... 21
FIGURE 21: CN3 - CAN INTERFACE CONNECTOR ..................................... 22
FIGURE 22: CAN LINE CONNECTION ................................................... 22
FIGURE 23: CAN NETWORK CONNECTION EXAMPLE ................................ 23
FIGURE 24: CN4 - RELAYS DIGITAL OUTPUTS CONNECTOR ..................... 24
FIGURE 25: RELAYS OUTPUTS CONNECTION .......................................... 24
FIGURE 26: RELAYS OUTPUTS BLOCK DIAGRAM .................................... 25
FIGURE 27: CN5 - OPTOCOUPLER DIGITAL INPUTS CONNECTOR ........... 26
FIGURE 28: OPTOCOUPLER INPUTS BLOCK DIAGRAM ............................ 27
FIGURE 29: OPTOCOUPLER INPUTS CONNECTION ................................... 27
FIGURE 30: TTL I/O, 1-WIRE, I2C BUS CONNECTOR ............................... 28
FIGURE 31: TTL I/O, 1-WIRE, I2C BUS BLOCK DIAGRAM ....................... 28
FIGURE 32: JUMPERS, CONNECTORS, TRIMMER, ETC LOCATION ............ 29
FIGURE 33: JUMPERS TABLE .............................................................. 30
FIGURE 34: COMPONENTS MAP SOLDER SIDE ....................................... 31
FIGURE 35: COMPONENTS MAP COMPONENTS SIDE ............................... 31
FIGURE 36: DRIVERS LOCATION FOR SERIAL COMMUNICATION ............ 33
FIGURE 37: POWER SUPPLY EXPS-2 PHOTO ....................................... 37
FIGURE 38: PHOTO OF CHARACTERS AVAILABLE ON QTP 12/R84-GF2 ... 39
FIGURE 39: KEYS NUMERATION AND LOCATION .................................... 41
FIGURE 40: DEFAULT KEYS CODES ....................................................... 41
FIGURE 41: MASTER SLAVE PROTOCOL COMMUNICATION EXAMPLE ...... 45
FIGURE 42: RS 232 CONNECTION WITH P.C. AND SERIAL FIRMWARE ....... 46
FIGURE 43: CODE AREA ORGANIZATION WITH LIBRARY FIRMWARE ........ 49
FIGURE 44: RAM USE WITH LIBRARY FIRMWARE .................................... 51
FIGURE 45: DEVELOPMENT MODE WITH LIBRARY FIRMWARE ............... 55
FIGURE 46: RS 232 CONNECTION WITH P.C. AND LIBRARY FIRMWARE ...... 56
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 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 etc. installation are destined - and then executable - always and in exclusive way from specialized warned and educated personnel, or directly from the TECHNICAL AUTHORIZED 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 respect 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.
HARDWARE AND FIRMWARE VERSION

This handbook make reference to printed circuit version 100403 and to firmware version 1.3 and following ones. The validity of the information contained in this manual is subordinated to the firmware release number, so the user must always verify the correct correspondence between the notations. The printed circuit version is reported in several places, for example over the serial connector CN2 on the component side. Inside the device, the firmware release number is written on the label stuck on the CPU or it can be obtained by a proper command.

GENERAL INFORMATION

QTP 12/R84 is a powerful module complete of inputs, outputs and operator interface. It contains a complete PLC controller capable to manage completely a specific application in a minimum size, specifically designed for industrial and/or domestic use. QTP 12/R84 is available with alphanumeric displays by 20 characters for 2 lines in two different types: LCD type with LED backligt or Fluorescent, or alternatively with graphic Fluorescent display with 140x16 pixels. In addition on the front panel there are: a 12 keys membrane keyboard, a label slot used to carry a name for the QTP 12/R84 or the user’s own logo, and a status LED. A practical and robust metallic container, in the standard DIN 72x144 size allows a direct mounting in front panel modality with a protected front side and back openings for connections. The enclosed brackets supplied with QTP 12/R84 let the user mount and/or unmount the terminal by performing a simple rectangle digging up on the support panel. Alternatively it can be used on a table with no elements addition. The module’s serial line can be buffered with the most frequently used electric protocols and thanks to this feature the QTP 12/R84 can be connected to each systems available on the market and moreover low cost networks can be realized where many different units can be contemporaneously managed. The interconnection with other devices is ensured also by the optional CAN line, that increase the possible applications fields and gives better performance of the complete network. CPU section features 32K FLASH with ISP interface for comfortable programming through serial line; this allows an easy development environment 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 high level development tools as the convenient BASIC compiler BASCOM 8051; the efficient C compiler µC/51 or the well known LadderWORK for the users that want contacts programming. Moreover are available different ready to use management firmwares for faster and easier development, that are briefly described below.

Hardware features of QTP 12/R84, including possible options, are as follows:

- Overall dimension: 72x144 mm frontal frame size; 9 mm frontal frame depth; 28 mm rear metallic housing depth.
- Remarkably low price.
- Aluminium container with front plastic frame.
- Front panel with Keyboard and display window protected by anti scratch Polyester cover.
- Case with rear mounting brackets.
- Front panel mounting.
- IP 54 standard protection on front side.
FIGURE 1: BLOCKS DIAGRAM
- 3 different models with different displays type:
  QTP 12/R84-C2: alphanumeric LCD display, LED backlight, with 2 lines for 20 chrs.
  QTP 12/R84-F2: alphanumeric Fluorescent display with 2 lines for 20 characters.
  QTP 12/R84-GF2: graphic Fluorescent display with 140 x 16 pixels.
- Viewing area size: 82x18 mm.
- Membrane keypad with 12 keys provided of double serigraphy: numeric and function.
- Status LED indicator managed by software.
- Panel name, and/or LED indicator, personalization label slot.
- Buzzer for acoustic signal driven by software.
- 16-bit family microcontroller, with 14.7 or 29.4 MHz software selectable clock speed.
- 3 different memory types: 32K FLASH EPROM; 1,2K RAM; 2K EEPROM.
- Serial line configurable in RS 232, RS 422, RS 485 or Current Loop.
- CAN communication line provided of proper line driver.
- 8 optocoupled digital inputs, both NPN or PNP type.
- Some inputs are connected to hardware counters and interrupts.
- 4 relay digital outputs, up to 5 A.
- Comfortable quick release, screw terminal connectors for a fast cablation.
- 2 TTL I/O lines on proper connector, suitable for I2C BUS, 1-Wire, SPI, etc., devices management.
- Wide range DC or AC power supply from 5 Vdc to 24 Vac.
- Total power consumption change according with used configuration, from 2.5 to 3.3 W.
- On board protection against voltage peaks by TransZorb™.
- Wide range of development software tools and programming languages, such as: assembler; FORTH; PASCAL compilers (SYS51PW); C compilers (HTC 51, SYS51CW, DDS MICRO C 51, µC/51); real time operating systems (CMX RTX); contacts logic (Ladder WORK); BASIC compilers (BXC51, BASCOM 8051); terminal emulation (GET 51), etc.
- Supplied with two different management firmwares: serial or library.
- Customized keyboard and program packages.
- For specific requirements about consumption and price, the LCD display can be not backlit (please contact grifo®).

Here follows a description of the board's functional blocks, with an indication of the operations performed by each one. The interconnections of these blocks and their general features are also described on figure 1.

SERIAL LINE

QTP 12/R84 has one serial communication line. By default condition it is configured as RS 232, but using a proper indication in the ordering phase, it can be configured in:

<table>
<thead>
<tr>
<th>Serial Line</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>RS 422</td>
<td>RS422 option</td>
</tr>
<tr>
<td>RS 485</td>
<td>RS485 option</td>
</tr>
<tr>
<td>Current loop</td>
<td>CLOOP option</td>
</tr>
</tbody>
</table>

Both the physical and logic communication protocol are completely configurable through the executed firmware, as described in proper paragraphs of SOFTWARE DESCRIPTION chapter. The numerous configurations of serial line offer a wide expansion and interconnection possibilities, including development of cheap networks with master slave communication.
Figure 2: Photo of available models
INDICATOR LED

**QTP 12/R84** has 1 indicator LED for different visual signals, placed near the personalization label. The main purpose of LED is to show a visual indication about the card's status, making so easier debug and verify operations even from a long distance from the panel, where the display is not readable. In this way its three different status (enabled, disabled or blinking) can show different operating condition as an alarm, an alert, a ready machine, etc. The described functionalities are totally autonomous and they are managed by comfortable command described in COMMANDS chapter.

DISPLAY

**QTP 12/R84** accepts three different display: **graphic fluorescent** with **140x16** pixels, **alphanumeric fluorescent** with **20x2** characters, **alphanumeric backlighted** LCD with **20x2** characters. LEDs backlighting of LCD model ensures a good visibility even when the environmental lighting changes and if it necessary the user can modify the contrast regulation by moving a specific trimmer. Another important feature of **QTP 12/R84** displays is their wide viewing angle that allows a good visibility from each frontal position. Further information on each display are reported in TECHNICAL FEATURES chapter.

As described in the chapter dedicated to commands, **QTP 12/R84** with graphic display (**QTP 12/R84-GF2**) can execute all kinds of commands (graphic and alphanumeric), while **QTP 12/R84** with alphanumeric display (**QTP 12/R84-C2** and **QTP 12/R84-F2**), of course, cannot execute graphic commands.

If the number of display characters is not sufficient please remind that the available firmwares, as described in COMMANDS chapter, manages also the scrolling messages: on a single line of display can be shown more text that continuously shift from right to left.

The user must choose the right display (so the right **QTP 12/R84** 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 with LCD display not backlit: for detailed information about these options and their availability, please contact directly grifo® offices.

KEYBOARD

**QTP 12/R84** 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 49) 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, when graphic display is installed. This allows to draw their function names in proximity of the physical keys and change them.
according to needs of the application (e.g. START, STOP, INS, DEL, ALRM, INFO, etc.). In addition, a personalization label can be added on the frontal of the keyboard to customize and/or identify the terminal, as described in APPENDIX C. Detailed information about keyboard managements performed by each firmwares are reported in proper chapters SOFTWARE DESCRIPTION and COMMANDS.

BUZZER

QTP 12/R84 has a circuitry that generates a steady sound, based on a capacitive buzzer. By software through specific comands, described in SOFTWARE DESCRIPTION and COMMANDS chapters, this circuitry can be enabled, disabled or intermittent, it can generate a simple beep, it can be linked to a key pressure, just to get the keyclick function, or it can signalize possible malfunctions. In latter case, 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.

MEMORY DEVICES

The QTP 12/R84 has three different memory types:
- 32K Bytes of FLASH EPROM for code,
- 256 Bytes of internal RAM for data and stack plus 1K Bytes of external ERAM for data,
- 2K Bytes of EEPROM for data,

that can be used from the user according with his requirements. The memory configuration available on the board is fixed and it can't be changed by the user; whenever it is not sufficient we suggest to contact directly grifo®. It is really interesting and useful the EEPROM that keeps data also when power supply is failed; in this way the card is always able to maintain parameters, system status and configuration, etc. in each working conditions.

The addressing of memory devices is automatically controlled by microcontroller that locates all the resource inside its addressing space obtaining a comfortable use with each software development tools, available for the QTP 12/R84. The developed firmwares use all the described memories for their normal activities and thus they allocates buffers, counters, status, setting, characters codes, messages, user characters, etc. Further information are reported in SOFTWARE DESCRIPTION chapter.

ON BOARD POWER SUPPLY

One of the most important peculiarity of QTP 12/R84 is its own switching power supply that requires an input voltage variable from 8÷24 Vac ±5% or 10÷38 Vdc ±5%. This section generates all the voltages used by the module. As alternative, QTP 12/R84 without power supply can be ordered (using the code .5Vdc or .ALIM) and in this case a +5 Vdc ±5% stabilized power supply must be provided by an external source. Detailed information on power supply section and other supply voltages, are reported in ELECTRIC FEATURES chapter and in POWER SUPPLY paragraph.
MICROCONTROLLER

The QTP 12/R84 can use the microcontroller T89C51CC01 or T89C51AC2 produced by ATMEL. These 8 bits processors are code compatible with the world wide used INTEL '51 family and they have an extended instruction set, fast execution time, easy use of all kind of memories, an efficient interrupts management and a rich list of integrated hardware peripheral devices. For further information on the listed microcontroller, please refer to specific documentation of the manufacturing company, or to APPENDIX D of this manual.

CAN INTERFACE

QTP 12/R84 can have, as option, of a complete CAN interface that supports the BasicCAN and PeliCAN 2.0B standards protocol. With this feature the user can afford and solve many problems as: high speed data transfer, long distance communication, autonomous errors management, multimaster and multislave networks support, etc. The code used to order this option is: .CAN

TTL I/O LINES

On one connector of QTP 12/R84 are available 2 digital I/O lines at TTL level. The functions of these lines is completely user defined and thanks to available management firmwares are supported also autonomous functions dedicated to SPI, 1-WIRE and I2C BUS standards. These interfaces allow connection of devices provided of the same communication standard, obtaining a local expansion of module ratings. Connector has been designed to allow mounting of the expansion both externally and internally of metallic container and to satisfy any need of the user. A wide set of software examples explains the management of most common peripheral expansions like A/D and D/A converters, display drivers, memories, temperature sensors, electronic keys, etc.

OPTOCOUPLED DIGITAL INPUTS

The card features 8 NPN and/or PNP inputs connected to a quick release screw terminal connector that are directly acquired by 8 I/O lines of microcontroller, through a galvanically isolated and buffered interface. These lines have been selected to take advantage of microcontroller internal peripherals, so the inputs can generate interrupts, be counted by hardware counters, etc. Optocoupled inputs are powered by a specific voltage called +Vopto that must be provided from an external source.

RELAYS DIGITAL OUTPUTS

The board is provided of 4 relays outputs 5 A, with normally open contacts. Each line is driven directly by an I/O line of microcontroller, buffered through a specific driver and connected to a comfortable quick release screw terminal connector to easy interface the field signals. Also in this case, microcontroller signals have been selected to take advantage of internal hardware peripherals.
FIGURE 3: QTP 12/R84 COMPLETE PHOTO
TECHNICAL FEATURES

GENERAL FEATURES

Resources:
- IP54 frontal
- Status LED software manageable
- Membrane keyboard with 12 metallic dome keys
- Buzzer for beep, feedback and keyclick driven by software
- RS 232 serial line, it can be buffered in RS 422, RS 485 or Current loop (option)
- CAN interface (option)
- Alphanumeric or graphic display in 3 different models
- Trimmer to set LCD display contrast
- 8 digital inputs optocoupled NPN and PNP type
- 4 digital outputs buffered with relays
- 2 TTL I/O digital lines
- 1 switching power supply

Displays:
- alphanumeric LCD 20x2 LED backligt
- alphanumeric VFD 20x2
- graphic VFD 140x16

Microcontroller:
- T89C51AC2 or T89C51CC01

Memory:
- 32K x 8 FLASH EPROM
- 256 x 8 internal RAM
- 1K x 8 external RAM
- 2K x 8 EEPROM

Clock frequency:
- 14.7456 MHz (duplicable by software)

Opto input max frequency:
- 13 KHz

Reset time after power on:
- 160 msec

EEPROM max write time:
- 8 msec

Physic comm. protocol:
- Baud rate: 1200, 2400, 4800, 9600, 19200, 38400
- Stop bit: 1 or 2
- Parity: none
- Bits x chr: 8, 9
- Default: 19200 Baud, 1 Stop, No parity, 8 Bits

Logic comm. protocol (#):
- Selectable between normal and master slave (Default: normal)

Receive buffer size (#):
- 30 characters

Transmission buffer size(#):
- 20 characters
Timing resolution (#): 10 msec
Timing precision (#): 2,5 msec
Keys autorepeat (#): After 500 ms and then every 100 ms
Messages number (#): 95
User EEPROM bytes (#): 40

(#{}) = Feature with serial or library firmware.

**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

Characters size: 
- LCD 20x2: 5 x 7 dots, 3.20 x 4.85 mm (W x H)
- Fluorescent 20x2: 5 x 7 dots, 2.40 x 4.70 mm (W x H)
- Fluorescent 140x16: from 5 x 7 dots, 1.50 x 3.62 mm (W x H) to 10 x 14 dots, 5.00 x 10.15 mm (W x H)

Weight: 330 g max.

Mounting: Surface or front panel mounting through proper 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 3,5
- CN2: D type, 9 pins, female
- CN3: quick release screw terminal, 3 pins, male, pitch 3.5
- CN4: quick release screw terminal, 6 pins, male, pitch 3.5
- CN5: quick release screw terminal, 9 pins, male, pitch 3.5
- J6: vertical strip, 4 pins, male, pitch 2.54
ELECTRIC FEATURES

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

Power consumption: See next table

Output power voltage: +5.0 Vdc

Available current on output power voltage: 400 mA - max consumption +5 Vdc (*)
1000 mA - max consumption +5 Vdc (only GF2)

Max current on relays: 5A (resistive load)

Max voltage on relays: 30 Vdc

Optocoupled input current: 2÷9 mA

Optocoupled inputs voltage: +V opto = 8÷30 Vdc (*)

RS 422-485 line impedance: 60 Ω

RS 422-485 termination circuit:
- pull-up resistor on positive: 3.3 KΩ
- pull-down resistor on negative: 3.3 KΩ
- line termination resistor: 120 Ω

CAN line impedance: 60 Ω

CAN termination circuit: 120 Ω resisor, disconnectable

(*) The data are referenced to 20 C° environmental work temperature (for further information please refer to chapter POWER SUPPLY).

Hereunder is listed the QTP 12/R84 power consumption referred to the different display models; for the wide range voltages, are reported the required powers already corrected with efficiency of on board power supply secion:

<table>
<thead>
<tr>
<th>DISPLAY model</th>
<th>Max consumption +5 Vdc ±5%</th>
<th>Max consumption 10÷38 Vdc ±5%</th>
<th>Max consumption 8÷24 Vac ±5%</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCD20x2 alphanumeric backlit: QTP 12/R84-C2</td>
<td>380 mA</td>
<td>2,7 W</td>
<td></td>
</tr>
<tr>
<td>VFD alphanumeric 20x2: QTP 12/R84-F2</td>
<td>360 mA</td>
<td>2,5 W</td>
<td></td>
</tr>
<tr>
<td>VFD graphic 140x16: QTP 12/R84-GF2</td>
<td>475 mA</td>
<td>3,3 W</td>
<td></td>
</tr>
</tbody>
</table>

FIGURE 4: CONSUMPTIONS TABLE

To reduce consumptions of QTP 12/R84 with LCD display it is possible to order particular models without backlighting: for further information please contact directly grifo®.
FIGURE 5: QTP 12/R84-C2 PHOTO

FIGURE 6: QTP 12/R84-F2 PHOTO

FIGURE 7: QTP 12/R84-GF2 PHOTO
INSTALLATION

In this chapter there are the information for a right installation and correct use of the terminal QTP 12/R84. In detail there are the locations and functions of each connector, of the user settable jumpers, of the trimmer and each other information about hardware configuration of the product.

CONNECTIONS

QTP 12/R84 module has 6 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 some figures show the connectors and signals position on the board to simplify their recognitions.

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 2 pins, vertical, quick release screw terminal connector with 3.5 mm pitch.

On CN1 must be connected the power supply voltage for the terminal that can be one out of three different types, as described in following figures:

![Figure 8: CN1 - Power Supply Connector](image)

Signals description:

<table>
<thead>
<tr>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vac</td>
<td>I - AC power supply lines connected to on board switching section; these signals accept an alternate 8 ± 24 Vac ±5% voltage.</td>
</tr>
<tr>
<td>+Vdc pow</td>
<td>I - DC power supply lines connected to on board switching section (+10 ± 38 Vdc ±5%) or stabilized (+5 Vdc ±5%) voltage connected to on board logic, according with ordered configuration.</td>
</tr>
<tr>
<td>GND</td>
<td>- DC power supply ground signal.</td>
</tr>
</tbody>
</table>

NOTE For further information about power supply configurations, please refer to paragraph POWER SUPPLY.
**Figure 9**: AC power supply connection 8 ± 24 Vac ±5%

**Figure 10**: DC power supply connection +10 ± 40 Vdc ±5%

**Figure 11**: Stabilized power supply connection +5 Vdc ±5%
CN2 - SERIAL LINE CONNECTOR

CN2 is a D type, 9 pins, female, vertical connector. On CN2 are available all the signals of the asynchronous serial line buffered with one of the electric standards RS 232, RS 422, RS 485 or Current loop. Placing of the signals has been designed to reduce interference and electrical noise and to simplify connections with other systems, while the electric protocols follow the CCITT directive.

**FIGURE 12: CN2 - SERIAL LINE CONNECTOR**

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>RX RS232</td>
<td>= I -</td>
<td>Receive data for RS 232.</td>
</tr>
<tr>
<td>3</td>
<td>TX RS232</td>
<td>= O -</td>
<td>Transmit data for RS 232.</td>
</tr>
<tr>
<td>5</td>
<td>GND</td>
<td>= -</td>
<td>Ground signal.</td>
</tr>
</tbody>
</table>

**RS 232 serial line:**

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RX- RS422</td>
<td>= I -</td>
<td>Negative receive data for RS 422.</td>
</tr>
<tr>
<td>2</td>
<td>RX+ RS422</td>
<td>= I -</td>
<td>Positive receive data for RS 422.</td>
</tr>
<tr>
<td>3</td>
<td>TX- RS422</td>
<td>= O -</td>
<td>Negative transmit data for RS 422.</td>
</tr>
<tr>
<td>4</td>
<td>TX+ RS422</td>
<td>= O -</td>
<td>Positive transmit data for RS 422.</td>
</tr>
<tr>
<td>5</td>
<td>GND</td>
<td>= -</td>
<td>Ground signal.</td>
</tr>
</tbody>
</table>

**RS 422 serial line:**

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RX- RS485</td>
<td>= I/O -</td>
<td>Negative receive and transmit data for RS 485.</td>
</tr>
<tr>
<td>2</td>
<td>RX+ RS485</td>
<td>= I/O -</td>
<td>Positive receive and transmit data for RS 485.</td>
</tr>
<tr>
<td>5</td>
<td>GND</td>
<td>= -</td>
<td>Ground signal.</td>
</tr>
</tbody>
</table>

**RS 485 serial line:**

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>RX- C.L.</td>
<td>= I -</td>
<td>Negative receive data for Current loop.</td>
</tr>
<tr>
<td>8</td>
<td>RX+ C.L.</td>
<td>= I -</td>
<td>Positive receive data for Current loop.</td>
</tr>
<tr>
<td>7</td>
<td>TX- C.L.</td>
<td>= O -</td>
<td>Negative transmit data for Current loop.</td>
</tr>
<tr>
<td>6</td>
<td>TX+ C.L.</td>
<td>= O -</td>
<td>Positive transmit data for Current loop.</td>
</tr>
</tbody>
</table>

**Current Loop serial line:**

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>RX- C.L.</td>
<td>= I -</td>
<td>Negative receive data for Current loop.</td>
</tr>
<tr>
<td>8</td>
<td>RX+ C.L.</td>
<td>= I -</td>
<td>Positive receive data for Current loop.</td>
</tr>
<tr>
<td>7</td>
<td>TX- C.L.</td>
<td>= O -</td>
<td>Negative transmit data for Current loop.</td>
</tr>
<tr>
<td>6</td>
<td>TX+ C.L.</td>
<td>= O -</td>
<td>Positive transmit data for Current loop.</td>
</tr>
</tbody>
</table>

**FIGURE 12: CN2 - SERIAL LINE CONNECTOR**
Figure 13: Serial communication diagram
**Figure 14: RS 232 Point to Point Connection Example**

- **CN2 QTP 12/R84**
  - 2 RX RS232
  - 3 TX RS232
  - 5 GND

- **External System**
  - TX
  - RX
  - GND

**Figure 15: RS 422 Point to Point Connection Example**

- **CN2 QTP 12/R84**
  - 1 RX+ RS422
  - 2 TX- RS422
  - 3 TX+ RS422
  - 4 RX+
  - 5 GND

- **External System**
  - TX-
  - RX-
  - RX+
  - TX+
  - GND

**Figure 16: RS 485 Point to Point Connection Example**

- **CN2 QTP 12/R84**
  - 1 RXTX- RS485
  - 2 RXTX+ RS485
  - 5 GND

- **External System**
  - TX-, RX-
  - TX+, RX+
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 extremis, respectively near the master unit and the slave unit at the greatest distance from the master.

Forcing and terminating circuitry is installed on QTP 12/R84 board. It can be enabled or disabled through specific jumpers, as explained later.

The master unit termination resistor must be added only if it is not already available inside (many RS 232-RS 485 converters already have it).

For further information please refer to TEXAS INSTRUMENTS Data-Book, "RS 422 and RS 485 Interface Circuits", the introduction about RS 422-485.
Figure 18: Current loop 4 wires point to point connection example

Figure 19: Current loop 2 wires point to point connection example
Possible Current loop connections are two: 2 wires and 4 wires. These connections are shown in figures 18+20 where it is possible to see the voltage that supply the loop (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 considering 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 with a 20 mA current;
- in case of shortcircuit each transmitter must dissipate at most 125 mW;
- in case of shortcircuit each receiver must dissipate at most 90 mW.

For further info please refer to HEWLETT PACKARD data book, in the section that describes the current loop opto couplers, for HCPL 4100 and 4200 devices.
CN3 - CAN INTERFACE CONNECTOR

CN3 is a 3 pins, vertical, quick release screw terminal connector with 3.5 mm pitch. Through CN3 can be connected the CAN serial communication line by following the standard rules defined by the same protocol. Signal placement has been designed to reduce interference and to obtain a fast and comfortable node connection on the field CAN bus.

**Figure 21: CN3 - CAN interface connector**

Signals description:

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

**Figure 22: CAN line connection**
Please remind that a CAN network must have two termination resistors (120 $\Omega$) placed at its extremities, respectively near the master unit and the slave unit at the greatest distance from the master.

On QTP 12/R84 the terminating circuitry is already installed: it can be connected or not through specific jumper, as explained later, in paragraph JUMPERS.

When the system to link on the CAN line have very different potentials, it is possible to connect also the grounds of the systems, that is pin 1 of CN3, to solve possible problems of communication and/or correct working.

**FIGURE 23: CAN NETWORK CONNECTION EXAMPLE**
CN4 - RELAYS DIGITAL OUTPUTS CONNECTOR

CN4 is a 6 pins, vertical, quick release screw terminal connector with 3.5 mm pitch. This connector allows to connect the normally open contacts and common signals of the 4 relays outputs available on QTP 12/R84. Please remind that maximum (resistive) load for each line is 5 A and maximum voltage is 30 Vdc. These lines are driven by microcontroller signals, opportunely buffered, that have been carefully selected to simplify software management (please refer to chapter COMMANDS).

Signals description:

- **OUT An** = O - Normally open contact for relay n of group A.
- **COMMON OUT A** = - Common contact for relays of group A.
- **OUT Bn** = O - Normally open contact for relay n of group B.
- **COMMON OUT B** = - Common contact for relays of group B.

**Figure 24: CN4 - Relays digital outputs connector**

**Figure 25: Relays outputs connection**
Relays are driven by 4 PNP transistors that are driven by as many I/O pins of microcontroller.

Previous figures 25 and 26 show respectly the external and the on board connection for the relays digital outputs. As describe in these figures, there are two groups of relays, named A and B, provided of their own common terminal. This allows connection of external loads even when they are supplied by two different sources, making the cabling of the whole system very easier.

**FIGURE 26: RELAYS OUTPUTS BLOCK DIAGRAM**
CN5 - OPTOCOUPLED DIGITAL INPUTS CONNECTOR

CN5 is a 9 pins, vertical, quick release screw terminal connector with 3.5 mm pitch. CN5 is used to connect the 8 optocoupled NPN or PNP input signals that QTP 12/R84 manages. In addition to digital inputs, on the connector there is also the common signal where the positive or negative terminal of galvanically isolated +Vopto voltage must be connected. The input lines are connected to some microcontroller I/O pins that have been carefully selected to take advantage of its internal peripherals, so they can generate interrupts, can be counted by hardware counters, etc., as described in COMMANDS chapter.

**FIGURE 27: CN5 - OPTOCOUPLED DIGITAL INPUTS CONNECTOR**

Signals description:

- **IN n** = I - Optocoupled input n, NPN or PNP type.
- **CNT n** = I - Optocoupled input for counter n, NPN or PNP type.
- **COMMON IN** = - Positive (NPN) or negative (PNP) terminal of external supply voltage +Vopto.

Input lines are optocoupled and provided with low-pass filter; this warrants a grade of protection for internal electronics against external noise. The use of a bidirectional circuit where current can circulate in both directions makes input lines suitable both for PNP and NPN drivers. Supply voltage for optocouplers (+Vopto) must be provided from external source always on connector CN5 and must be compliant to the specifications written in ELECTRIC FEATURES paragraph. The following figures 28 and 29 describe respectively the on board and the external connection for the optocoupled digital inputs.
**Figure 28: Optocoupled Inputs Block Diagram**

- **CPU**
- **Opto Coulplers**
- **Low Width, Frequency Filters**
- **CN5**
- **PIN 1-8**
- **COMMON IN**
- **PIN 9**

**Figure 29: Optocoupled Inputs Connection**

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>CN5</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN 1</td>
<td>IN 2</td>
<td>IN 3</td>
<td>IN 4</td>
<td>IN 5</td>
<td>IN 6</td>
<td>IN 7</td>
<td>IN 8</td>
<td>COMMON IN</td>
<td>+V opto</td>
</tr>
</tbody>
</table>

- **NPN type**
- **ISOLATED DC POWER SUPPLY**

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>CN5</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN 1</td>
<td>IN 2</td>
<td>IN 3</td>
<td>IN 4</td>
<td>IN 5</td>
<td>IN 6</td>
<td>IN 7</td>
<td>IN 8</td>
<td>COMMON IN</td>
<td>+V opto</td>
</tr>
</tbody>
</table>

- **PNP type**
- **ISOLATED DC POWER SUPPLY**

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>CN5</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN 1</td>
<td>IN 2</td>
<td>IN 3</td>
<td>IN 4</td>
<td>IN 5</td>
<td>IN 6</td>
<td>IN 7</td>
<td>IN 8</td>
<td>COMMON IN</td>
<td>GND opto</td>
</tr>
</tbody>
</table>
J6 - TTL I/O, 1-WIRE, I2C BUS CONNECTOR

J6 is a 4 pins, male, vertical, strip connector with 2.54 mm pitch. On J6 are available two digital I/Os and the signals with +5 Vdc power supply generated by on board switching regulator, that can be used to comfortably supply some external systems or devices. Thanks to these lines and a dedicated management firmware can be connected many peripheral devices that follows the I2C BUS, SPI, 1-WIRE, etc., standards. Signals are TTL compliant, according to each standards specifications, and their disposition has been designed to reduce interferences and to simplify the connection.

J6 is not accessible from breaking in the rear side of container, so for its connection first open the QTP12/R84 box and then insert the proper female connector and let the wires go through the same breaking; alternatively the added electronics can be mounted on a little printed circuit that is connected to J6 in piggy back mode, inside the aluminum container.

The female connector for J6 can be ordered to grifo® with the code CKS.AMP4 (kit composed by a female AMP Mod II 4 pins connector plus 4 contact to crimp) or AMP4 CABLE (ready to use connector with 4 wires long 1 metre).

Signals description:

- **P2.n** = I/O - Signal connected to Port 2.n of microcontroller.
- **1-WIRE n** = I/O - Signal for management of 1-WIRE line n.
- **SDA** = I/O - Data signal of I2C BUS software line.
- **SCL** = O - Clock signal of I2C BUS software line.
- **+5 Vdc** = O - Positive of +5 Vdc power supply.
- **GND** = - Ground signal.

Remind that pin 2 and 3 of J6 are electrically connected to a 10 KΩ pull up resistor, as described in the following figure:

![Figure 31: TTL I/O, 1-WIRE, I2C BUS Block Diagram](image)
FIGURE 32: JUMPERS, CONNECTORS, TRIMMER, ETC. LOCATION
CONTRAST REGULATION TRIMMER

On QTP 12/R84 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 32.

JUMPERS

On QTP 12/R84 there are five jumpers for card configuration and connecting them, the user can perform some selections that regards the working conditions of the card. Here below there is the jumpers list, location and function in the possible connection modalities:

<table>
<thead>
<tr>
<th>JUMPER</th>
<th>CONNECTION</th>
<th>FUNCTION</th>
<th>DEF.</th>
</tr>
</thead>
<tbody>
<tr>
<td>J1, J2</td>
<td>not connected</td>
<td>Termination and forcing circuit not connected to RS 422, RS 485 serial line.</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>connected</td>
<td>Termination and forcing circuit connected to RS 422, RS 485 serial line.</td>
<td></td>
</tr>
<tr>
<td>J3</td>
<td>position 1-2</td>
<td>Configures serial line for RS 485 standard electric protocol (2 wires half duplex).</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>position 2-3</td>
<td>Configures serial line for RS 422 standard electric protocol (4 wires full duplex or half duplex).</td>
<td></td>
</tr>
<tr>
<td>J4</td>
<td>not connected</td>
<td>120 Ω termination resistor not connected to CAN line.</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>connected</td>
<td>120 Ω termination resistor connected to CAN line.</td>
<td></td>
</tr>
<tr>
<td>P1</td>
<td>not connected</td>
<td>Selects RUN mode that executes application program saved on FLASH EPROM.</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>connected</td>
<td>Selects DEBUG mode that executes boot loader (see FLASH EPROM PROGRAMMING paragraph).</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 33: Jumpers table**

To recognize these valid connections, please refer to the board printed diagram (serigraph) or to figure 35 of this manual, where the pins numeration is listed; for recognizing jumpers location, please refer to figure 32 again.

The "*" denotes the default connection, or on the other hand the connection set up at the end of testing phase, that is the configuration the user receives.

Further information about purpose of the QTP 12/R84 jumpers are reported in the following paragraphs, that describe the section where the same jumpers are used.
Figure 34: Components map solder side

Figure 35: Components map components side
SERIAL LINE CONFIGURATION

Serial line of QTP 12/R84 can be buffered in RS 232, RS 422, RS 485 or Current loop. By software the serial line can be programmed to operate with some logical protocols and many standard physical protocols that can be decided through opportune firmware modalities.

By hardware can be selected which one of these electric standards is used, through jumpers connection (as described in the previous tables) and drivers installation. In the following paragraphs there are all the information on all serial communication configurations. 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 first not standard (not RS 232) serial configuration must be always ordered, and thus performed, by grifo® technicians. This far the user can change in autonomy the configuration following the informations below:

- SERIAL LINE IN RS 232 (default configuration)

  IC4 = driver MAX 202
  IC2 = no device
  J3 = indifferent
  J1, J2 = not connected
  IC5 = no device
  IC3 = no device
  IC6 = no device

- SERIAL LINE IN CURRENT LOOP (option .CLOOP)

  IC4 = no device
  IC2 = no device
  J3 = indifferent
  J1, J2 = not connected
  IC5 = no device
  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 18÷20. Current loop interface can be used to make both point to point and multi point or network connections through a 2 wires or a 4 wires connection.

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

  IC4 = no device
  IC2 = driver SN 75176 or MAX 483
  J3 = position 2-3
  IC5 = no device
  J1, J2 = (*)
  IC3 = no device
  IC6 = no device

RS 422 electric protocol can be used to make 4 wires full duplex connections both point to point and multi points network. RS 422 transmitter ability, essential in networks, is managed directly by QTP 12/R84 serial firmware (by selecting the master slave logic protocol) or by P2.5 signal of microcontroller, when library firmware is used, as follows:

  P2.5 = DIR = low level  = logic state 0  ->  transmitter enabled
  P2.5 = DIR = high level = logic state 1  ->  transmitter disabled

In point to point systems, signal P2.5 = DIR can be always kept low (transmitter always enabled), while in multi point networks transmitter must be enabled only when a transmission is performed.
RS 232 serial line

Current loop serial line

RS 422 serial line

RS 485 serial line

**Figure 36: Drivers location for serial communication**
- SERIAL LINE IN RS 485 (option .RS 485)

<table>
<thead>
<tr>
<th>J1</th>
<th>IC4</th>
<th>= no device</th>
</tr>
</thead>
<tbody>
<tr>
<td>J2</td>
<td>IC2</td>
<td>= driver SN 75176 or MAX 483</td>
</tr>
<tr>
<td></td>
<td>IC5</td>
<td>= no device</td>
</tr>
<tr>
<td></td>
<td>IC3</td>
<td>= no device</td>
</tr>
<tr>
<td></td>
<td>IC6</td>
<td>= no device</td>
</tr>
</tbody>
</table>

In this modality the signals to use are pins 1 and 2 of connector CN2, that become transmission or reception lines according to status defined by firmware. RS 485 electric protocol can be used to make 2 wires half duplex connections either in point to point or multi points networks. RS 485 line direction is managed directly by QTP 12/R84 serial firmware (by selecting the master slave logic protocol) or by P2.5 signal of microcontroller, when library firmware is used, as follows:

- P2.5 = DIR = low level = logic state 0 -> line in transmission
- P2.5 = DIR = high level = logic state 1 -> line in reception

With this kind of serial communication all the transmitted characters are also received and so the user is allowed to verify autonomously the succes of transmission. In fact, any conflict on the line can be recognized by testing the received character, after each transmission.

(*) 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 J1 and J2 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 only in the farthest boards, that is on the edges of the communication line.

During a reset or a power on, RS 485 driver is in reception and RS 422 transmission driver is disabled, to avoid conflicts on the line.

For further information about serial communication please refer to the connection examples of figures 14+20.

CAN INTERFACE CONNECTION

Jumper J4 connects or does not connect CAN termination resistor, as described in figure 33. The CAN BUS must be a differential line with 60 Ω impedance and the termination resistors 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 lines (please see example of figure 23). Correct CAN termination contributes remarkably to correct communication; in fact the QTP 12/R84 on board interface can suppress transients and avoids radio frequency and electromagnetic disturb, only if connection to the field is made correctly.

CAN line is not galvanically isolated (as described in paragraph POWER SUPPLY) from board 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, using shielded cable, to obtain the greatest protection against external noise.
INTERRUPTS

A remarkable feature of **QTP 12/R84** is the powerful interrupts management. Here follows a short description of which devices can generate interrupts and their modalities; for further information about interrupts management please refer to the microprocessor data sheet or APPENDIX D of this manual.

- Optocoupled input IN 5 on CN5 -> Generates an /INT0 = P3.2 of microcontroller.
- Optocoupled input IN 6 on CN5 -> Generates an /INT1 = P3.3 of microcontroller.
- Microcontroller peripherals -> Generate an internal interrupt. In detail the possible microcontroller interrupt sources are: Timer 0, Timer 1, Timer 2, PCA and its compare and capture modes, UART, CAN, etc.

An interrupt management section (ICU integrated in microcontroller), allows to enable, disable, mask and prioritize, so the application program has always the possibility to react promptly to every event, deciding also the priority of contemporary interrupts.

Finally some interrupt sources are used by the developed firmwares, as described in the SOFTWARE DESCRIPTION chapter.

CONNECTOR SIGNALS INTERFAECEMENT

To prevent possible connecting problems between **QTP 12/R84** and the external systems, the user has to read carefully the previous paragraph information (both text and figures) and he must follow these instructions:

- All TTL signals must follow the rules of this electric standard. The connected digital signals must be always referred to card ground (GND) and then the 0V level corresponds to logic state 0, while the 5V level corresponds to logic state 1.
- For optocoupled input signals, both the contact to acquire and external +Vopto must be connected in serie. In detail contacts (as switches, relays, proximities, etc.) must perform the following connection between signals on CN5:
  
<table>
<thead>
<tr>
<th>SIGNAL</th>
<th>NPN</th>
<th>PNP</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN x</td>
<td>GND opto</td>
<td>+V opto</td>
</tr>
<tr>
<td>COMMON IN</td>
<td>+Vopto</td>
<td>GND opto</td>
</tr>
</tbody>
</table>

To avoid problems with electric noise, it is suggestable to keep galvanically isolated +Vopto and board power supply, this means to keep separate GND and GND opto signals.

- Relays outputs must be connected directly to the load to drive (remote control switches, power relays, electric valves, etc.). Board contacts are normally open and can drive up to **5 A, 30 Vdc**. To drive load with different supply voltages, two common signals are available for two couples of relays.
- For RS 232, RS 422, RS 485, Current loop and CAN signals the user must follow the standard rules of each one of these protocols, defined by CCITT directive.
- Also I2C BUS, SPI and 1-WIRE signals are at TTL level, as defined by the same standards; for completeness it is remarked that in a network, with several devices and rather long, it is better to study the connection lay out. Do not forget that lines are already provided of 10 KΩ pull up resistor.
POWER SUPPLY

Terminal QTP 12/R84 is provided with a power supply section that solves in an efficient and comfortable way the problem to supply the boards in any situation. It generates energy for all sections of the board: control logic, display, backlighting, serial interfaces, CAN interface, LED, buzzer, keyboard, digital inputs, digital outputs, etc.

Here follow voltages required according to card configuration:

Default version: This configuration includes a switching power supply that requires 10+38 Vdc ±5% or 8+24 Vac ±5% that must be provided through CN1 (polarity must be respected in case of DC voltage). This allows to supply the terminal using standard power sources of industrial environments, like transformers, batteries, solar cells, etc. A comfortable and inexpensive solution for default version power supply can be the EXPS-1 and EXPS-2 products that can be connected directly to the terminal starting from mains.

Please remark that on board switching supply is provided with single diode rectifier, so in case of DC supply, all ground signals of the terminal (GND) are at the same potential.
This is the default version, normally delivered without further specifications in the order.

Version .5Vdc or .ALIM: 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 (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 option configuration only, to directly agree upon grifo®.

+V opto: Independently from the selected power supply configuration, QTP 12/R84 always requires a second supply voltage, named +V opto that supplies the optocouplers of digital input section; it must be included in 8+30 Vdc range and must be provided on connector CN5.

Selection of power supply section must be performed during the order phase, in fact this means a different hardware configuration that must be made by grifo® technicians.

Current available to supply external loads with on board generated +5 Vdc and GND signals, for example through J6 connector, must be less than:

- 400 mA - max consumption on +5 Vdc when QTP 12/R84-C2 or F2 is used
- 1000 mA - max consumption on +5 Vdc when QTP 12/R84-GF2 is used

for example, in case of QTP 12/R84-F2, it becomes:
- 400 mA - 360 mA = 40 mA

To warrant highest immunity against noise and so a correct working of the cards, it is essential that the two supply voltages are galvanically isolated. In order to obtain this EXPS-2 can be ordered: it has a double galvanic isolated transformer that accepts a mains input voltage.

QTP 12/R84 is always provided with a TransZorb™ based protection circuit to avoid damages from incorrect voltages and break-down of power supply section. It is also provided with a distributed 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 37: POWER SUPPLY EXPS-2 PHOTO
SOFTWARE DESCRIPTION

Two different firmwares have been developed in order to simplify and speed the QTP 12/R84 use; infact they allow an high level development of user applications without the typical problems caused by direct management of the hardware. The two firmwares are options that can be defined in the order, they have been named serial (.SER) and library (.LIB) and they have many common features but important differences in use modalities, as below described.

It is important to remind that if no options are specified in the order, then the default configuration with library firmware is supplied and thus only the serial firmware must be properly specified. Both firmwares are supplied together with complete and usefull demo programs either in executable or 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.

A detailed description of the numerous commands available in both firmwares, necessary to take advantages from all the QTP 12/R84 features, is reported in following chapter.

DATA STORED ON EEPROM

On board EEPROM the firmwares of QTP 12/R84 store data used and/or changed through the specific commands. 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 use them.

When the card is received from an order or a reparation, the EEPROM is supplied already set with its default values, that are:

- presence byte -> 255 (FFH)
- keys codes -> those reported in table of figure 40
- patterns of user defineable characters -> 255 (FFH)
- messages -> 255 (FFH)
- user byte for general use -> 255 (FFH)

Whenever the user desires to reset the default configuration on all data saved in EEPROM, the firmwares provide the modalities below described: user must be nery careful with EEPROM initialization, in fact all previously saved data are lost.

- serial firmware: the menu EEPROM DATA of local setup allows the user to decide whether to leave these data unchanged (NOINI option) or to set them to their default value (INIT option). By selecting the INIT option, once exit from local set up a string is shown on the display with a scrolling bar; * (asterisk) inform about the status progress of the operation. Initialization phase execution time is about 20 seconds while the number of printed * is 10.

- library firmware: a specific function that first initializes EEPROM and then the firmware, can be called by user. In other words the user must perform the call described in INTEGRATION AND USE OF LIBRARY FIRMWARE paragraph. Also with this firmware the initialization phase execution time is about 20 seconds.
CHARACTERS VISUALIZATION ON DISPLAY

The firmware of QTP 12/R84 show 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). 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 correspondence between codes and displayed characters is defined by the following rules:

<table>
<thead>
<tr>
<th>Codes</th>
<th>Characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 ÷ 15</td>
<td>User defined</td>
</tr>
<tr>
<td>16 ÷ 31</td>
<td>Special and different according with installed display</td>
</tr>
<tr>
<td>32 ÷ 127</td>
<td>Standard ASCII</td>
</tr>
<tr>
<td>128 ÷ 255</td>
<td>Special and different according with installed display</td>
</tr>
</tbody>
</table>

To allow representation of special characters, that have same codes of some one character commands, a specific command has been provided that selects the operating mode of QTP 12/R84 firmware, among the two available:

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

After a power on it is automatically selected the command mode to make immediately utilizable each functionality. 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.

Through the firmwares every model of QTP 12/R84 has 8 user characters that can be defined and/or stored and shown on the display, as further explained in the paragraph USER CHARACTERS COMMANDS.

About special characters please refer to APPENDIX B and remind that it is possible to get different display models, provided of different special characters, but everything must be directly arranged with grifo®.

![Figure 38: Photo of Characters available on QTP 12/R84-GF2](image-url)
RECEIVE BUFFER

QTP 12/R84 firmwares are provided with a reception buffer that simplify the management from user side, in fact it reduces the waiting time on the developed user application program. Each received character is immediately saved inside this buffer (30 bytes long) and processed at the end of the currently executed command.

- When serial firmware is used and commands that requires a long execution time (delete commands, EEPROM management commands, graphic commands, etc.) are continuously received, the buffer will become full and will overflow. When overflow occurs last location of the buffer is overwritten by each next received characters, and these are definitively lost. The master unit must stop the transmission until the QTP 12/R84 has emptied the receive buffer and it is still ready to receive other data. In practice the user must insert suitable delays between the commands transmission, to leave sufficient time to QTP for executing the required operations and to avoid the complete filling of reception buffer.

- When library firmware is used the buffer can't overflow and so the user application program doesn't need any delay.

KEYBOARD ACQUISITION

When QTP 12/R84 firmwares recognize a key pressure, they returns the specific code to user application program. Moreover an auto repeat function of the stroked key is implemented so when QTP 12/R84 firmware recognizes the pressure on a key for a time grater than 0.5 seconds it will return its code about each 0.1 seconds and it lasts until that key is released. If the keyclick function is enabled when the code of the pressed key is returned, the on board buzzer also generates a loud beep that sonorously signalize the event to the user. If buzzer was already enabled then it is disabled for a little time period, to ensure the acustic event recognition in any circumstance. Another features provided by firmwares 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.

- When serial firmware is used and normal communication is selected, the key code is immediately transmitted on serial line, viceversa if master slave communication is selected the code is sent only upon reception of specific request of master unit, as described in the following paragraphs. Moreover with this firmware four keys are used to set some function modalities, as described in proper LOCAL SET UP paragraph.

- When library firmware is used the key codes are temporarily saved in a specific transmission buffer and then returned to user application program when this requires them.

KEYS CODES

The following figures show the key numbers and position and a table with the default codes that QTP 12/R84 firmwares return when a key is pressed. As for the command sequences the codes are shown in decimal, hexadecimal and ASCII mnemonic format.
These codes are those returned under default condition, i.e. the configuration the user receives, but they can be comfortably reconfigured by using a specific command. This feature really simplifies the management software development in fact the master unit can change the codes according with his requirements and it can also disable the keys.
SERIAL FIRMWARE .SER

This firmware transforms QTP 12/R84 in a complete serial video terminal and for this reason any characters received from serial line, if it is not a command, is shown on the display and codes of any pressed keys, are serial transmitted to the master control unit. The user can't interfere on code executed by QTP, that is in fact used as a closed and finished product, but only on program executed by master control unit (P.C., PLC, CPU card, etc.) that communicates with QTP through serial line. on this line are naturally interchanged also the parameters and the results of the rich list of commands.

Moreover the serial firmware can use two different communication mode, equal to two logical protocols, named normal and master slave; the first is suitable for point to point connections while the second, once coupled with a proper electric protocol (RS 422, RS 485, Current loop), allows networks connections. In the latter condition more QTPs and other units of the same or different type, can be driven by a single master unit in a practical and comfortable way. As the serial firmware is completely based on serial communication, it manages also a local set up which allows to set the physic communication protocol by using the keys and the display of QTP 12/R84.SER.

In the following five paragraphs are described the most important features of the serial firmware.

LOCAL SET UP WITH SERIAL FIRMWARE

Thanks to a proper local set up mode, some parameters of communication protocol, the key click mode and EEPROM initialization can be set by the user with the simple use of 4 keys.

To enter set up mode the user must power on the QTP 12/R84 and simultaneously he must press the keys number 0 and number 1 (serigraph * and 0) for at least half of a second.

When the set up mode is entered, on the display appears the "** Local Setup **" string and with keys number 10 and number 11 (serigraph 9 and #) the configuration parameters shall be changed as below described:

Key N° 11 (#): Changes current menu, recognized by the following messages:

"COMMUNICATION" to change the communication type
"BAUD RATE" to change the communication baud rate
"STOP BIT" to change the stop bit number
"KEY-CLICK" to change the keyclick mode
"NAME (Hex)" first digits of hexadecimal identification name
"NAME (Hex)" second digits of hexadecimal identification name
"EEPROM DATA" initializes data on EEPROM
"SAVE and EXIT" to exit from set up mode

Key N° 10 (9): Changes current value of menu, with the following possibilities:

COMMUNICATION: Norm. or M.-S. for normal or master slave protocol (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 protocol
KEYCLICK: ON or OFF (def=ON)
NAME (Hex): Changes the digits enclosed in "<>" from 0 to F (def=80H)
EEPROM DATA: NOINI or INIT (def=NOINI)
SAVE and EXIT: exits set up and configures QTP 12/R84 with selected parameters
Once exited from set up mode, the selected parameters are saved on EEPROM and they are maintained until another local set up is executed and then terminal starts its normal functionality. The default values before reported 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 if master slave is selected).

The options of remaining menus are instead described in the previous and following parameters.

NOTE
Please remind that set up mode can be entered only during power up, when previously described condition is recognized; in fact if the keys 0 and 1 are pressed at the same time during normal operation of QTP 12/R84.SER, the set up mode will not start and the code of the pressed keys will be transmitted on the serial line.

The local set up is normally executed only one time after the first installation, so it regards expert staff and not the final user of QTP 12/R84.SER that use it as a simple operator interface unit.

COMMUNICATION MODALITIES WITH SERIAL FIRMWARE

QTP 12/R84.SER provides two different serial communication modalities on its asynchronous serial line:

Norm. Normal communication uses 8 bits per character, no parity plus stop bit and baud rate selected by the user through local setup. This communication mode is suitable for point to point connections in RS 232, RS 422 and Current loop.

M.-S. Master slave communication uses 9 bits per character, no parity, one stop bit plus baud rate selected by the 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 further information about master slave mode, please refer to next paragraph.

Local set up 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 COMMUNICATION MODE

The master slave mode uses the 9 bits communication technique. In addition to the 8 data bits also a ninth bit is managed and it recognizes between a call coming from the "master" device to any of the "slave" structures, and a normal info transmission between master and the currently selected device.

When 9th bit is placed at 1, the 8 data bits of the same character has to contain the identification name of the device required for communication, while by placing this particular bit at 0, it is possible to take out or supply info at the selected device.
When QTP 12/R84.SER is used, the identification name must be that one selected by the local set up program on the "NAME (Hex)" entry. When this byte is sent (with 9th bit set to 1) the QTP 12/R84.SER recognizes itself and it waits the string containing chars, data or commands. In this string there must only be a command that involves the return of an answer to send via serial line from QTP part; if there is more than one command with answers, the results of the remaining ones are ignored.

Between the transmission of a character and the next one there must be an time interval shorter than the **Time Out**, as elapsed this delay, the QTP 12/R84.SER will consider the data string terminated and it will begin the answering phase. The Time Out values for each baud rates is below described:

<table>
<thead>
<tr>
<th>Baud Rate</th>
<th>Time Out</th>
<th>Character transmission time</th>
</tr>
</thead>
<tbody>
<tr>
<td>38400 Baud</td>
<td>550 µsec</td>
<td>287 µsec</td>
</tr>
<tr>
<td>19200 Baud</td>
<td>990 µsec</td>
<td>573 µsec</td>
</tr>
<tr>
<td>9600 Baud</td>
<td>1540 µsec</td>
<td>1146 µsec</td>
</tr>
<tr>
<td>4800 Baud</td>
<td>3080 µsec</td>
<td>2292 µsec</td>
</tr>
<tr>
<td>2400 Baud</td>
<td>6105 µsec</td>
<td>4584 µsec</td>
</tr>
<tr>
<td>1200 Baud</td>
<td>12100 µsec</td>
<td>9167 µsec</td>
</tr>
</tbody>
</table>

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

```
“character transmission time”+“Time out”
```

before to receive the first character of the answering string, transmitted by the QTP 12/R84.SER. The answer consists in a byte containing the code of the pressed key (255 = FF Hex if no key is pressed) or a characters sequence that coincide with the answer of the command sent in the previous interrogation. Please remark that answer is provided also when master unit transmit a command sequence with only the identification name: this simplifies the check for keys pressed or invalid commands.

Several demo programs, coded in different programming languages, are provided with QTP 12/R84.SER. They implement master slave communication and can be used directly by the user or modified according to the specific needs. When the master unit is a P.C., the user can also take advantage of comfortable DLL libraries that allow to manage master slave communication at high level, this means without having to worry about management of nineth bit, timings, eventual 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 command 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 cannot communicate using 9 bits, it is possible to simulate this communication mode by using the parity bit and programming its value properly, before any characters transmission, according to this scheme:

   **If the character to transmit has EVEN number of "1" bits**

   - **If 9th bit must be 1** -> **Set parity to ODD**
   - **If 9th bit must be 0** -> **Set parity to EVEN**
If the character to transmit has ODD number of "1" bits
- If 9th bit must be 1 -> Set parity to EVEN
- If 9th bit must be 0 -> Set parity to ODD

3) If the scrolling messages mode is enabled, the time between two calls, in addition to the time indicated at point 1, must be about 12000 µsec.

To explain better the master slave protocol, here follows an example where master unit sends three commands to QTP 12/R84.SER (reading of version number, a string to show and a check for possible keys pressed) with baud rate 38.4 KBAud and identification name 80H:

<table>
<thead>
<tr>
<th>Master</th>
<th>QTP 12/R84.SER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sends “Reading of version number” command, that is the characters sequence: 80H with ninth bit set to 1 1BH with ninth bit set to 0 56H with ninth bit set to 0 with a delay between characters lower than 550 µsec</td>
<td>Receives character of the command and verifies the end with 550 µsec Time Out</td>
</tr>
<tr>
<td>Waits for 837 µsec</td>
<td>Recognizes command sequence, executes the command and stores answer for next command</td>
</tr>
<tr>
<td>Receives one character of answer</td>
<td>Sends the answer, which is the code of the possible key pressed with ninth bit set to 0</td>
</tr>
<tr>
<td>Sends a string to show on the display, that is the character sequence: 80H with ninth bit set to 1 1° character of string with ninth bit set to 0 2° character of string with ninth bit set to 0 : : : : : : with a delay between characters lower than 550 µsec</td>
<td>Receives character of the command and verifies the end with 550 µsec Time Out</td>
</tr>
<tr>
<td>Waits for 837 µsec</td>
<td>Recognizes command sequence and shows on the display the characters of the string</td>
</tr>
<tr>
<td>Receives three characters of answer with the version number previously requested</td>
<td>Transmits saved response which is the version number required in previous command, with ninth bits set 0</td>
</tr>
<tr>
<td>Sends key pressed check command, that is the character sequence: 80H with ninth bit set to 1</td>
<td>Receives character of the command and verifies the end with 550 µsec Time Out</td>
</tr>
<tr>
<td>Waits for 837 µsec</td>
<td>Recognizes sequence without commands so performs no operation</td>
</tr>
<tr>
<td>Receives one character of answer corresponding to code of eventual key pressed</td>
<td>Sends the answer, which is the code of the possible key pressed with ninth bit set to 0</td>
</tr>
</tbody>
</table>

**Figure 41: Master Slave Protocol Communication Example**
HOW TO START WITH SERIAL FIRMWARE

In this paragraph are listed the operations that must be performed to start using the **QTP 12/R84.SER** in a practical 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 are the serial connection example with a development P.C., the description of a fast functional test, etc. The following steps assume that a standard Personal Computer (provided of one free RS 232 serial line and a generic operating system, up to Windows 98) is available, to allow any user to execute them.

A) **Serial connection between QTP 12/R84.SER and P.C.:**
   A1) Perform the serial connection described in figure 42 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 P.C. It can be easily discovered that this connection cable is reversed and it can be conveniently ordered to grifo® with the code CCR 9+9R.

   ![Figure 42: RS 232 connection with P.C. and serial firmware](image)

   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.

   A3) Press some keys of QTP and check that the relative keyclick is generated by internal buzzer.

B) **Use of demo program:**
   B1) On the floppy disk or CD rom received with the first purchase, is available the file PRQTP12R.EXE, that contains the executable code for the serial firmware demo. This file once found, must be copied in a comfortable folder on the hard disk of the used P.C.

   B2) Execute the program copied at point B1, compile its start questions by selecting a normal communication and the mounted display type. At this point press a key on P.C. to continue without execute the local setup in fact the shown configuration coincides with the default one already set on the received **QTP 12/R84.SER**.

   B3) Carry on demo program execution and check that the operations described on P.C. monitor are correctly executed on QTP; when required interact with the same program in such a way as 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 P.C., that normally is located on Windows menu: "Start | Program | Accessories | Communication", and execute it.
C2) Through the HYPERTERMINAL properties windows, setup 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 P.C. keyboard and check that pressed keys are shown on QTP 12/R84.SER display and that pressing QTP keys, their codes appear on P.C. monitor. For completeness it can be tested also the effects of some commands by typing their codes sequences always on P.C. 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 the keys 1 and 2).

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 ensure that default configuration values are saved, through local setup. If malfunction persists please contact directly grifo® technician.

DEMO PROGRAMS FOR SERIAL FIRMWARE

In correspondence of the first purchase together with QTP 12/R84.SER it is supplied a floppy disk or a grifo® CD where are saved numerous demo programs that allow to test and weigh 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 P.C. platforms or grifo® microprocessor cards (as GPC®, Mini Module, etc).

As described in HOW TO START WITH SERIAL FIRMWARE paragraph the programs named PRQTP12R.* use all the commands of QTP with a simple interaction with the user; but many other demos are supplied capable, for example, to: drive QTP connected to a serial network, manage 1-WIRE sensors, manage messages, use the master slave protocol with DLL libraries, 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 authorizathion, license or additional cost. Furthermore in case of unusual requirements or combinations, specific new demo programs can be obtained, after proper agreement with grifo®.
With this firmware QTP 12/R84 operates as a powerful process controller, complete of operator interface, capable to work stand alone or combined with other systems. The user must interfere on code executed by QTP, in such a way as to develop an application program that satisfy his requirements. This object is comfortably reached thanks to the rich and complete list of commands that can be called directly with relative parameters and results. Such commands normally satisfy all the typical problems of industrial environments and they are minutely described in homonymous chapter. So QTP 12/R84.LIB is not a complete product ready to use in fact before it must be specialized by the user. The specialization can be performed with simple and low cost development tools, either at high or low level, and they make QTP a very flexible and versatile product. In fact the user application program that specializes it allows to solve any problems, even those with high complexity, and to realize different automations by using the same hardware. On the contrary of serial firmware the library one doesn't use the asynchronous serial line of QTP: so the user application program can communicate with other devices with no restriction both in point to point and network connection, with any logic protocol. Thanks to selected development tools the serial line can be used as a debug channel that let the user found out the errors and reduces the time necessary to obtain a finished system.

In the following six paragraphs are described the most important features of the library firmware.

INTEGRATION AND USE OF LIBRARY Firmware

The library firmware has been developed with the following aims:
- linkable with all the available programming languages;
- reduce the used hardware resources;
- maintain compatibility of use with firmwares of other QTPs;
- provide easy techniques for the commands calls and parameters exchange;
- cover the normal and diffused requests of industrial environments;
that have defined the integration and use modalities of the same firmware, inside the user application program.

It is important to remind that the user of library firmware must have a basic knowledge of the used microcontroller and of embedded software development in fact the following documentation doesn't supply these information but it uses them. If the user have not this know how he can found it in software development tool documentation and in microcontroller data sheets, reported in APPENDIX D of the manual.

In this paragraph are listed all the general information about integration and use of library firmware suitable for every user, with each development tools supplied by grifo® or other companies.

The integration and use of library firmware needs some hardware and software tools opportunely specified in following descriptions. Their complete documentation is provided inside the same tools and it is not duplicated in this manual. Among these tools the most important is a standard Personal Computer (up to now called development P.C.) provided of one free RS 232 serial line and a generic operating system, from Windows 95.

In conclusion the operations necessary to integrate and use the library firmware are:
a) Install the software development tool preselected to realize the application program, on the development P.C. Generally all software packages available for the mounted microprocessor (or in other words the numerous tools for the 51 family) can be used, as described in next DEVELOPMENT SOFTWARE FOR LIBRARY FIRMWARE paragraph.

b) Install the ISP (In System Programming) utility program on the development P.C., that is the FLIP program capable to communicate with microcontroller boot loader, through a serial line, and that allows to read, erase and program the FLASH EPROM memory. Further information can be found in next FLASH EPROM PROGRAMMING paragraph.

c) Prearrange the software development tool to ensure that the generated application program reserves the hardware resource used by library firmware. As illustrated in figures 43, 44 and in next RESOURCES USED BY LIBRARY FIRMWARE paragraph, the application program can't use the last area of FLASH, some internal RAM areas, one timer counter, the on board EEPROM, etc.

d) Physically the library firmware coincides with an executable code that must be saved at the end of the code area of microcontroller, as described on figure 43. This code is supplied in QTP12Rxx.HEX file that, thanks to its HEX format, could be directly used for FLASH burning. On this memory, in addition to described firmware, it must be saved also the executable code of user application program at the beginning of area code obtaining its immediate start after a power on or a reset.
The transit from application program to *library firmware* is performed through a proper entry points table, located at fixed addresses, that makes a "link bridge" between the two codes saved in the single code area.

The choice of allocation addresses of the three areas on FLASH has been carefully made to obtain the maximum free space for the user application program and to have fixed entry points that don't change also when the *library firmware* is updated and/or expanded.

With this choice the user can employ the new firmware version by simply reprogramming it on FLASH, with no intervents or modifications on his application program.

e) The START LIB.FW value is established by *library firmware*, so it depends from the firmware version; with the current version 1.3 it is fixed at 6F00H and anyhow it can be easily obtained by loading the QTP12Rxx.HEX file (where xx corresponds to version number) and checking its start address.

The user, after each generation of his application program, must verify that the END APPL. SW address is lower than START LIB.FW address or, in other words, that the two codes are not overlapped. This verification is easily performed in fact normally all the software development tools (assemblers, compilers, languages, etc.) inform about the dimension of the generated code and it will be sufficient to compare these information with the START LIB.FW value, before described.

f) Redirect the interrupts service routines for microcontroller /INT0, TIMER0, /INT1, TIMER1 hardware sources to the library firmware entry points illustrated in figure 43. The redirections must be performed following the rules of the development tool and normally they coincides with absolute jump instructions (i.e. LJMP 7FE5H) to entry address, placed inside the relative interrupt service routine. Please remind that of the 4 interrupts to redirect only those for TIMER 0 is always necessary while the other 3 must be redirected only when the relative commands are used, as described in next RESOURCES USED BY LIBRARY FIRMWARE paragraph.

g) The commands execution, the parameters and results exchange and the use of the *library firmware* are simplified by three procedures, with as many entry points, with the following features:

**CONSOLE STATUS:** returns the status of the presence of data that the firmware must send to application program; the data can be either a keypressed code or the answer of a command previously sent. The procedure has no input parameters and a single output parameter, saved in accumulator register, that coincides with the number of characters ready to be sent to application program. This number of characters has also a status function in fact if it is zeroed there are no data and viceversa.

**CONSOLE INPUT:** waits the availability of a data that firmware must send to application program and returns this data; also for this procedure the data can be either a keypressed code or the answer of a command previously sent. The procedure has no input parameters and a single output parameter, saved in accumulator register, that coincides with the described data.

**CONSOLE OUTPUT:** sends a data from application program to firmware; the data can be a character to show on display or a command to execute or its possible parameters. The procedure has a single input parameter, saved in accumulator register, that coincides with the described data and no output parameters.

In the application program the *library firmware* is used with the three described procedures; if console management procedures of the software development tools, that already have a compatible structure, are redirected on these procedures it is obtained a remarkable simplification and an incomparable use flexibility. This is the reason why the prefix CONSOLE has been used in the procedures names. So the high level instructions PRINT, PRINTF, KBHIT, SCANF, INPUT, etc. of a C or BASIC language development tools, automatically call the three entry procedures of the *library firmware* and all their possibilities can be used.
**Figure 44: RAM Use with Library Firmware**

- **FFH**: Internal Peripherals Registers Area
- **B6H**: Special Function Registers
- **80H**: User Application Program Area
- **55H**: Library Firmware Area
- **24H**: User Bit Area
- **20H**: Registers Banks R0÷R7
- **00H**: User Application Program Area
- **03FFH**: External RAM
- **0000H**: User Application Program Area
g) After a reset or a power on the user application program must prearrange the library firmware for next operations. These initializations are performed by two proper procedures, with as many entry points, with the following features:

**FW INITIALIZATION:** executes all the initialization operations like: setting of variables; clear buffers; disable buzzer, status LED, digital outputs, counters; setup and clear of display; setup of blinking cursor in Home position; load of user characters patterns; setup of saved keyclick mode; enable keyboard scanning; enable time based functions; etc. The procedure has no input nor output parameters.

**EEPROM INITIALIZATION:** initializes the EEPROM with default data described in DATA STORED ON EEPROM paragraph and then executes all the initialization operation listed for FW INITIALIZATION. The procedure has a single input parameter, saved in accumulator register, that coincides with the type of mounted display and no output parameters. The input parameter has the following meaning:

- 0  ->  Alphanumeric LCD 20x2 display  (QTP 12/R84-C2.LIB)
- 1  ->  Alphanumeric VFD 20x2 display  (QTP 12/R84-F2.LIB)
- 2  ->  Graphic VFD 140x16 display   (QTP 12/R84-GF2.LIB)

Remind that the execution time of this procedures is about 20 seconds.

Normally these procedures must be called only one time at the beginning of user application program, by using the rules of the development tool and normally they coincide with an absolute call instructions (i.e. LCALL 7FF4H) to entry address, preceded by a possible setting of the input parameter. The EEPROM INITIALIZATION procedure must be called only one time to maintain endurance of the EEPROM that it writes: its typical use are in correspondence of the first installation or when default settings must be restored after wrong and unwanted modifications.

h) Once developed the user application program that uses the library firmware with all the features described in the previous points, it must be saved on FLASH EPROM of QTP 12/R84.LIB together with the library firmware, as described in FLASH EPROM PROGRAMMING paragraph.

i) At this point the QTP 12/R84.LIB is complete and ready to be used and tested inside the real application system. The debug of the obtained application program can be done with the modalities of the used software development tool; the serial line (not used by library firmware) is an excellent candidate for this function.

RESOURCES USED BY LIBRARY FIRMWARE

The library firmware offers commands that allows to easily manage the numerous resources of the board as display, keyboard, buzzer, LED, digital outputs, digital inputs, EEPROM memory, 1-WIRE interface, etc. These commands need an additional list of QTP 12/R84 hardware resources that are briefly described in this chapter, together with the use limitation of the user application program:

**Code area in microcontroller FLASH EPROM:** coincides with an area at the end of FLASH EPROM, already described in points (d), (e) of previous paragraph. On this area is saved the code of library firmware and it must not be absolutely used from application program, to avoid malfunctions of the entire system. Some software development tools (as BASCOM 8051) can be configured to autonomously advice the user when the generated code reaches a prefixed limit (END APPL. SW >= START LIB. FW).

**Data area in microcontroller internal RAM:** coincides with two internal RAM areas (the first located in the direct access area and the second located in the indirect access area of
microcontroller) where are saved all the flags, variables, buffers of the *library firmware*. These areas are located at fixed addresses described on figure 44 and they must not be absolutely used from application program, to avoid malfunctions of the entire system.

The protection and reservation of these memories is obtained following the rules of the used software development tool and normally it is performed with compiler directives, setting of the possible project, variables declaration that are located but never used, setting of user start up code, etc. The software development tools normally shows windows that reports the program generation results that allow the user to easily check the safeguard of these areas and to discover possible malfunctions during test phase.

The choice of *library firmware* memory usage has been done carefully with the intention to leave unused a portion of any microcontroller memory type, that is: 4 bytes addressable at bit level equal to 32 user bits, 43 bytes with direct access, 48 bytes with indirect access and finally 1024 bytes of external RAM. Moreover the completeness of the commands offered from firmware drastically reduces the requirements of memory the user application program, up to few work variables and the stack.

**Stack area of microcontroller:** the *library firmware* have not its own stack and it uses those of application program. During configuration of the selected software development tool, the user must consider the stack size required from firmware, that in worst conditions can reach the 17 bytes.

**TIMER0 timer counter of microcontroller:** all time based process of *library firmware* are managed through a periodic interrupt generated by TIMER0 of microcontroller. The user application program can not use this resource and it must redirect the microcontroller standard interrupt vector (000BH) to specific entry point (7FE8H) of firmware. Once initialized the firmware services this interrupt each 2.5 msec and consequently slow down the execution of user application program. The performances reduction depends upon the used commands and upon process in execution: in some circumstances the delay can reach the tens of msec, as properly indicated in the commands descriptions.

**TIMER1 timer counter of microcontroller:** to manage the commands relative to counter 1 the *library firmware* uses the TIMER1 of microcontroller, set as hardware counter with interrupt reload. If the user application program use firmware counter 1 then it can not use this resource and it must redirect the microcontroller standard interrupt vector (001BH) to specific entry point (7FEE8H) of firmware.

**TIMER2 timer counter of microcontroller:** to manage the commands relative to counter 2 the *library firmware* uses the TIMER2 of microcontroller, set as hardware counter. If the user application program use firmware counter 2 then it can not use this resource.

**/INT0 interrupt signal of microcontroller:** to manage the commands relative to counter 3 the *library firmware* uses the /INT0 interrupt signal of microcontroller. If the user application program use firmware counter 3 then it can not use this resource and it must redirect the microcontroller standard interrupt vector (0003H) to specific entry point (7FE5H) of firmware.

**/INT1 interrupt signal of microcontroller:** to manage the commands relative to counter 4 the *library firmware* uses the /INT1 interrupt signal of microcontroller. If the user application program use firmware counter 4 then it can not use this resource and it must redirect the microcontroller standard interrupt vector (0013H) to specific entry point (7FEBH) of firmware.
FLASH EPROM PROGRAMMING

As described in previous paragraphs the library firmware must be saved on the microcontroller FLASH EPROM together with user application program, as illustrated in figure 43. The FLASH management is performed through an ISP technique (In System Programming) that reduces the cost and the time for development in fact it eliminates the use of external EPROMs, programmer, eraser, etc.

The ISP programmation requires only the standard development P.C. that executes a proper management program named FLIP (FLexible In system Programming), interacts with a boot loader available on microcontroller side and it is capable to read, erase, verify, program either the FLASH or the EEPROM memory. Everything happen through a simple serial connection between development P.C. and QTP 12/R84.LIB normally done with RS 232 serial line or with CAN line (for this last possibility contact directly grifo®).

Jumper P1 selects the QTP 12/R84 operating mode between the two available:

<table>
<thead>
<tr>
<th>P1</th>
<th>Operating mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not connected</td>
<td>RUN mode</td>
</tr>
<tr>
<td>Connected</td>
<td>DEBUG mode</td>
</tr>
</tbody>
</table>

In RUN mode after a power on the application program saved in FLASH is always executed, independently by external conditions, while in DEBUG mode the power on causes the execution of microcontroller boot loader and thus allows the ISP programmation For further information on ISP programmation please consult the specific technical documentation released by ATMEL, while in this paragraph we remind that:

- in the "Device | Select" window select the right microcontroller according to ordered QTP, or in detail the T89C51AC2 in case of QTP 12/R84 and T89C51CC01 in case of QTP 12/R84.CAN;
- in the "Settings | Communication | RS232" window select the maximum available baud rate (115200) and reduce it only if communication errors come out;
- in the "Buffer | Options" window set to NO the option "Reset Buffer Before Loading" and select the option "Whole buffer": this ensures a correct loading of both the files to program;
- with the FLIP load and then program always two .HEX file: those with the code of application program generated by software development tool and the library firmware saved on QTP12Rxx.HEX file (where xx correspond to version number).

The paragraph HOW TO START WITH LIBRARY FIRMWARE includes a complete example of FLASH EPROM programming, accompanied by photographs of the FLIP most important windows.

DEVELOPMENT SOFTWARE FOR LIBRARY FIRMWARE

A wide selection of software development tools can be used, allowing the best development of the user application program that have been described many times in the previous paragraphs. Generally all software packages available for the mounted microprocessor (or in other words the numerous tools for the 51 family) can be used, either at high and low level.

All the software development tools supplied by grifo® always include many example programs, libraries with console redirection, header files and accessories that integrate the library firmware and make it ready to use for each commands and section of the card.
Among these we remind:

**BASCOM 8051:** It is a low price cross compiler for BASIC source programs. It is a powerful software tool that includes editor, BASIC compiler and simulator in an easy to use integrated development environment for Windows. Many memory models, data types and direct use of hardware resources instructions, are available. Delivered on a CD for Windows with user manual.

**μC/51:** It is a comfortable, low cost, software package with a complete IDE that allows to use an editor, and ANSI C compiler, and assembler, a linker and a remote source level debugger user configurable. Sources of main libraries and of remote debugger are included, and so several utilities and demo programs. Delivered on a CD for Windows with user manual.

**LADDER WORK:** It is an easy to use system to generate automation application using the very famous and diffused contacts logic. It includes a graphic editor to place and connect hardware components of the card (like digital I/O, counters, A/D, etc.) like on an electric diagram and define their properties, an efficient compiler to create the executable code and an utility to download it to card memories. Integrated IDE makes comfortable use of all these tools. Delivered on a CD for Windows with user manual and hardware key.

---

**Library firmware commands:**

- Home
- Cursor right
- Absolute cursor position
- Clear page
- Clear line
- Set multifunction pin
- Read counter
- Square wave on dig. out
- Acquire dig. input
- Acquire dig. in with deb.
- Graphic cursor position
- Graphic area setting
- Graphic font setting

---

**User application program**

```c
void main(void)
{
    init_cpu();                                        // Initializes mounted CPU
    ini_qtp12r_fw();              // Initialize library fw with data saved on QTP
    CONS_QTP=1;                                                 // Console on QTP
    putc(0x0C);                                             // Command CLEAR PAGE
    printf("Product pieces=");                        // Shows windows on display
    cmd_qtp("\x1B\x01\x01");                // Command SET PIN 1 AS COUNTER 2
    for (;;)                                                // Endless loop start
        pos_cur_alf(0,15);                                 // Places cursor in 0,15
        printf(" %5d",word);                   // Associated to IN 1, and shows it
        if (strhlp[0]==1)                         // If IN8 enabled: general allarm
            pos_cur_alf(1,0);                                 // Places cursor in 1,0
            printf(" GENERAL ALLARM");                // Shows allarm
        // endif
    //endfor                                                    // Endless loop end
}
```

---

**Figure 45: Development mode with library firmware**
HOW TO START WITH LIBRARY FIRMWARE

In this paragraph are listed the operations that must be performed to start using the QTP 12/R84.LIB in a practical 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 are the serial connection example with a development P.C., the description of a fast functional test, etc. The following steps assume that a standard development Personal Computer (provided of one free RS 232 serial line and a generic operating system, like Windows 95 or greater) is available, to allow any user to execute them.

A) Serial connection between QTP 12/R84.LIB and development P.C.:  
A1) Perform the serial connection described in figure 42 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 P.C. It can be easily discovered that this connection cable is reversed and it can be conveniently ordered to grifo® with the code CCR 9+9R.

A2) Found the HYPERTERMINAL communication program on the development P.C., that normally is located on Windows menu: "Start | Program | Accessories | Communication", execute it and through the HYPERTERMINAL properties windows, setup 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.

A3) Set RUN mode, that is jumper P1 not connected.

A4) Supply power voltage on CN1 and check that buzzer is immediately disabled, on display is shown the message "QTP 12/R84 library fw demo program" with a blinking block cursor in the right down corner and that on development P.C. appears the demo program presentation message. Each QTP 12/R84.LIB, received for the first time, is delivered with its demo program and library firmware already saved in FLASH and arranged to allow automatic start at power on: if the demo program presentation screen doesn’t appear please check again the serial connection and the P1 right configurations.

A5) Follow the instructions of the demo and check execution of all the commands of library firmware: the user must interact with demo either through serial console on development P.C. and the resources of same QTP.
A6) When demo execution is terminated turn off QTP power supply.
A7) Exit from HYPERTERMINAL program on development P.C.

B) Reprogram of FLASH with demo program:
B1) On the floppy disk or grifo® CD rom received with the first purchase, are available the files PRFWQ12R.HEX and QTP12Rxx.HEX (where xx corresponds to version number), that contains respectively the executable code of the demo and the library firmware code, already saved on FLASH as described in point A4. This files once found, must be copied in a comfortable folder on the hard disk of the development P.C.
B2) Always on grifo® CD is also available the utility program FLIP: locate and install it in a comfortable folder of development P.C. hard disk. FLIP manages QTP 12/R84.LIB microcontroller memories programming through the simple serial connection made at point A1, as described in FLASH EPROM PROGRAMMING paragraph.
B3) Set DEBUG mode, that is P1 connected.
B4) Close each program executed on development P.C. that could use the serial line COMx of the development P.C., as the terminal emulator HYPERTERMINAL.
B5) Power on the card and check that buzzer is enabled and remains active during execution of following points B6÷B17.
B6) Run the ISP programming software FLIP (version >= 1.8.2), installed at point B2.
B7) Select the device to program by clicking the first button on the top left, picking the correct name in the device selection window and press OK. the selection of the device to program must be done according to ordered QTP, or in detail T89C51AC2 in case of QTP 12/R84 and T89C51CC01 in case of QTP 12/R84.CAN.

![Figure 47: FLIP settings window (1 of 4)](image)

B8) Select the communication mode for ISP programmation of QTP by clicking the second button on the top left, picking in sequence: RS 232, the serial port of development P.C. used in point A1, 115200 Baud and the then press Connect. At this point the FLIP start communication with microcontroller boot loader and fill in a list of data in its main window. If communication fails and after about 20 seconds the window "Timeout error" appear, try in sequence to: reduce communication baud rate to 19200; repeat points B3÷B8; check serial connection by repeating the points A1÷A5.
B9) Open the "Buffer | Options" window set to NO the option "Reset Buffer Before Loading" and select the option "Whole buffer": this ensures a correct loading of both the files to program;

![Figure 48: FLIP settings window (2 of 4)](image)

B10) Load the two files to write in FLASH described at point B1 (that is QTP12Rxx.HEX and PRFWQ12R.HEX) by executing two times the following operations: click the third button on top right and select the file using the dialog box.

B11) Select all the check boxes in the frame "Operations Flow" like in figure 50, to let FLIP execute the four operations: erase, blank check, program and verify.

B12) Make sure that main windows of FLIP looks like figure 50; in details for the boxes: "Size", "Hex file:“, "Signature Bytes" and "BSB / EB / SBV" the data must exactly match.

![Figure 49: FLIP settings window (3 of 4)](image)
B13) Press button "Run" in the main window to start the preselected ISP operations.

B14) Wait the end of ISP operation execution. The status bar on the bottom reports operation progress and text box in the bottom left reports operation status; the check boxes become red and then green when the respective operation is successfully completed. Thus wait for "Verify" check box to become green.

B15) Close FLIP.

B16) Start the terminal emulator configured like in point A2. The HYPERTERMINAL settings and execution can be also obtained by a simple double click on the icon of a specific configuration file (with extension .HT) that can be created directly by HYPERTERMINAL, with the save option of the "File" menu.

B17) Set RUN mode, that is P1 not connected.

B18) Power off and then on the card: the demo program, that use the library firmware just saved in FLASH, must start as described in point A4.

C) Creation of executable code of demo program:

C1) Install on the hard disk of the development P.C. the software environment selected to develop the application program. As described in the chapter DEVELOPMENT SOFTWARE FOR LIBRARY Firmware there are many different software tools that satisfy any customers requirements.

C2) On floppy disk or grifo® CD in addition to file with the executable code of the demo program, described at point B1, there are also the source file of the same. These have an extension that
identifies the used software development tools (for example PRFWQ12R.BAS for BASCOM 8051 or PRFWQ12R.C for μC/51) and they are properly organized inside demo programs tables available on CD, together with possible definition and header files (89C51CC01.DAT and FWQ12R.BAS for BASCOM 8051, CANARY.H and FWQ12R.H for μC/51, etc.). Once these files have been located they must be copied in a comfortable folder on the hard disk of development P.C.

C3) Compile the source file by using the selected software tools: the file PRFWQ12R.HEX must be obtained equal to those available on grifo® CD and already used at points B.

C4) Reperform the programmation of the obtained HEX file in the QTP FLASH, by executing again the points B3÷B18. About the FLIP settings, please remind that they could be inserted only the first time, then saved in a proper configuration file that once opened, reconfigure automatically all the FLIP options.

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 if malfunction persists please contact directly grifo® technicians.

Instead when execution of all the steps above described is right, the user has realized his first application program that coincides with demo of QTP 12/R84.LIB. At this point it is possible to modify the source of the demo/s program according to application requirements and test the obtained program with the steps above listed (from B3 to C3) in cyclic mode, until the developed application program is completely well running. When this focus is reached the development P.C. can be eliminated, by obtaining a self running product, as below described:

D) Final preparation of application:
D1) Set the RUN mode (jumper P1 not connected) and disconnect development P.C.

DEMO PROGRAMS FOR LIBRARY Firmware

In correspondence of the first purchase together with QTP 12/R84.LIB it is supplied a floppy disk or a grifo® CD where are saved numerous demo programs that allow to test and weigh immediately the received product. These programs are provided both in executable and source format and they are realized with all the software development tools suggested by grifo®, that are C compiler μC/51, BASIC compiler BASCOM 8051, contact logic LADDER WORK.

As described in HOW TO START WITH LIBRARY Firmware paragraph the programs named PRFWQ12R.* use all the commands of QTP with a simple interaction with the user; they manage the display in both alphanumeric and graphic modes, the keyboard, the EEPROM, the messages, the digital buffered I/Os, the hardware counters, some 1-WIRE sensors, 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 authorization, license or additional cost. Furthermore in case of unusual requirements or combinations, specific new demo programs can be obtained, after proper agreement with grifo®.
COMMANDS

This chapter describes all the commands available in QTP 12/R84 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 View Point 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 the chapter documentation.

COMMANDS FOR CURSOR POSITION

Here follows the list of the cursor positioning commands.

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.

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, it will be moved to the Home position that is the first position in the first row.

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 \textbf{32 (20 Hex)}. The position is expressed in alphanumeric mode so their valid values ranges respectively are 32÷33 and 32÷51.

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:

\textbf{27 89 33 34 or 1B 59 21 22 Hex or ESC Y ! "}

If row and/or column values are not compatible with the installed display, the command is ignored.
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 position to the left and it erase the contents of the reached cell.
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 EEPROM

In the following paragraphs are described the commands that directly manage the data saved on QTP 12/R84 on board EEPROM; there are other commands that indirectly use this memory device 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/R84 is ready for management of its on board EEPROM. This command must be executed any time there are data to be read or write on this type of memory.

When QTP 12/R84 firmware receives this command, it answers with the following codes:

<table>
<thead>
<tr>
<th>Code</th>
<th>(Hex)</th>
<th>(Mnemonic)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>06</td>
<td>ACK</td>
<td>QTP 12/R84 ready</td>
</tr>
<tr>
<td>21</td>
<td>15</td>
<td>NACK</td>
<td>QTP 12/R84 not ready</td>
</tr>
</tbody>
</table>

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 range.

This byte has a reserved allocation on the on board EEPROM that, once it is set with the desired value, it allows for example, to verify that QTP 12/R84 runs correctly, or if there are some communication problems, when serial firmware is used.

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 firmware sends back the value of its presence byte.

For example, this command 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 for general use is a reserved area in the on board EEPROM 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 maintained also when power supply is absent. The address that identifies the used location is 16 bits wide and addh, addl are respectively the most and less significant part. The user EEPROM with the QTP 12/R84 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 addl addh 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 GENERAL FUNCTIONS

In the following paragraphs are described all the general purpose commands that manage some features of QTP 12/R84 firmwares. These commands do not come into the other subgroups and for this reason they are described in a proper paragraph.

READ FIRMWARE VERSION

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

The firmware returns a string of 3 characters containing the management firmware version that is resident and executed by QTP 12/R84.
For example with firmware version 1.3 the following characters will be returned:

```
49 46 51 or 31 2E 33 Hex or 1.3
```

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/R84 coincides with value 2 (02 Hex). This command is useful especially when serial firmware is used, on the communication serial line there are many different devices and the master 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:

<table>
<thead>
<tr>
<th>lum parameter</th>
<th>Brightness</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (00 Hex)</td>
<td>Brightness at 100%</td>
</tr>
<tr>
<td>1 (01 Hex)</td>
<td>Brightness at 75%</td>
</tr>
<tr>
<td>2 (02 Hex)</td>
<td>Brightness at 50%</td>
</tr>
<tr>
<td>3 (03 Hex)</td>
<td>Brightness at 25%</td>
</tr>
</tbody>
</table>

If parameter is not valid, command is ignored.

NOTE This command is available only with models QTP 12/R84-F2 and QTP 12/R84-GF2. In case of QTP 12/R84-C2 with display LCD, command must not be sent because it produces the visualization of an undesired character and a shift in all the next received data.
OPERATING MODE SELECTION

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

It defines the operating mode for the 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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (00 Hex)</td>
<td>Command mode</td>
</tr>
<tr>
<td>255 (FF Hex)</td>
<td>Representation mode</td>
</tr>
</tbody>
</table>

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

COMMUNICATION RESET

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

This command reinitializes communication, with no modifications on the other process in execution. Naturally communication is referenced to data exchange between used firmware and user application program, so it is independent from used vehicle. In detail the command performs the following operations:
- clears reception buffer;
- eliminates possible characters of answer still to return;
- eliminates possible pressed keys still to return;
- terminates all the commands reception in execution and their character redirection (message storage, graphic mode, write bytes on 1-WIRE line, etc.).

GENERAL RESET

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

This command performs a general reset of the executed firmware and it resets up 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;
- disables all digital outputs and possible timed management;
- resets all counters;
- disables all alternative functions on each signal;
- clears display and stops possible scrolling messages;
- disable status LED and buzzer plus possible intermittent attribute;
- reloads the setting saved on EEPROM that are keyclick mode, key codes, user characters patterns, identification name, communication protocol, etc.

**NOTE:** This command uses the on board EEPROM, so before executing it is better to check the EEPROM availability through the proper command. The execution time of this command is some tens of msec.: when serial firmware .SER is used and after this command many other commands must be sent, it is preferable 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.

LED AND BUZZER ACTIVATION

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

Device indicated in `device` parameter is modified using attribute specified in `attr` parameter. In detail device are so identified:

- 0 (00 Hex) -> status LED
- 255 (FF Hex) -> buzzer

while available attributes are:

- 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.

For example, to activate the status LED with blinking attribute, the following sequence must be sent: 27 50 0 85 or 1B 32 00 55 Hex or ESC 2 NUL U
COMMANDS FOR MESSAGE MANAGEMENT

In the following paragraphs are described all the commands that manage messages, available in QTP 12/R84 firmwares. The messages are 20 characters sequence that can be saved on board EEPROM 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. allarms, equipment status, etc.) without the transmission of the numerous characters of this information but only the few characters of the command. The QTP 12/R84 firmwares manage the scrolling messages visualization, too; with this feature on a single line of display can be shown more text that continuously shift from right to left.

When serial firmware .SER is used, 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 PC.

QTP 12/R84 features one EEPROM with a size of 2048 bytes that can store up to 95 messages identified by a number from 0 to 94.

READING OF MAX MESSAGE NUMBER

<table>
<thead>
<tr>
<th>Code:</th>
<th>27 110</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hex code:</td>
<td>1B 6E</td>
</tr>
<tr>
<td>Mnemonic:</td>
<td>ESC n</td>
</tr>
</tbody>
</table>

This command returns the number of the last messages that can be saved on EEPROM. It is always 94 (5E Hex) as previously described. This command is important for other QTP models that has a variable messages number and it has been implemented on QTP 12/R84 for compatibility and interchangeability with all grifo® operator panels.

MESSAGE STORAGE

<table>
<thead>
<tr>
<th>Code:</th>
<th>27 33 67 mess.num. chr. 0 ÷ chr.19</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hex code:</td>
<td>1B 21 43 mess.num. chr. 0 ÷ chr.13 Hex</td>
</tr>
<tr>
<td>Mnemonic:</td>
<td>ESC ! C ASCII(mess.num. ) ASCII(chr.0) ÷ ASCII(chr.19)</td>
</tr>
</tbody>
</table>

This command stores the 20 characters message, identified by mess.num, parameter, on the on board EEPROM. The 20 chars which form the message must be visualizable on the display so they must be included in the range 0÷255 (0÷FF Hex). The message number must be included in the range of 0÷94 to select one of the available messages.

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.
MESSAGE READING

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

This command reads the 20 characters message identified by `mess.num.` parameter, from the EEPROM and it returns this message, beginning from the first char of the string. The message number must be included in the range of 0÷94 to select 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 the code 21 (15 Hex) = NAK is returned.

VISUALIZATION OF N MESSAGES

**Code:** 27 33 68 mess.num. n  
**Hex code:** 1B 21 44 mess.num. n  
**Mnemonic:** ESC ! D ASCII(mess.num.) 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.num.` while the remaining messages are those immediately subsequents in EEPROM. The `mess.num.` value and the number of the following messages defined by n, must be included in the range 0÷94, to select only the available messages. If these number are out of range the command is ignored.

The n quantity of messages to be visualized depends on the model of the installed display. For alphanumeric mode all displays can show at most 40 characthers so the maximum number of messages is 2. In other words the n parameter can ranges from 1 to 2 and if its value is out of this range, 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, it will be necessary to send 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.num. n.chr  
**Hex code:** 1B 21 53 mess.num. n.chr  
**Mnemonic:** ESC ! S ASCII(mess.num.) ASCII(n.chr)

This command visualizes a `n.chr` characters string on the display first line 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.num.` message already stored in EEPROM and continues with next characters always saved in following EEPROM messages.

The `mess.num.` value must be included in the range 0÷94, to select one of the available messages. If the value is out of range this command is ignored.

Instead the `n.chr` parameter must range in the following values:

- **0** Stops the scrolling messages visualization (the `mess.num.` 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, the command will be ignored.

The scrolling messages visualization is always performed on the first display line and the cursor position and attributes are maintained.

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, it will be necessary to send 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. This causes a subsequent slowing down of user application program execution when *library firmware *.LIB is used, or a subsequent slowing down of commands interpretation when *serial firmware *.SER is used. This is the reason why it is necessary to wait for few **msec** between the transmission of 20÷30 bytes data blocks when the last firmware is used. In this way misunderstanding and interpreting problems of the received data, caused by receive buffer overflow, are completely avoid.
COMMANDS FOR KEYBOARD MANAGEMENT

Below are described the commands that can be used to manage the QTP 12/R84 keyboard. Detailed information about keys management and codes returned by the firmwares are available in KEYBOARD ACQUISITION paragraph.

KEY CODE RECONFIGURATION

<table>
<thead>
<tr>
<th>Code:</th>
<th>27 55 key n. code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hex code:</td>
<td>1B 37 key n. code</td>
</tr>
<tr>
<td>Mnemonic:</td>
<td>ESC 7 ASCII(key n.) ASCII(code)</td>
</tr>
</tbody>
</table>

When the selected key n. is reconfigured, each time it is pressed, the firmware will return the new specified code on serial line.

The value of key n. to reconfigure is obtained by figure 39 and it must be included in the range 0÷11 (0÷B Hex) otherwise the command is ignored.

If the code value is included in range 0÷254 (00÷FE Hex) then the firmware return this code when the relative key is pressed; but if code parameter has 255 (FF Hex) value then the key is disabled and when it will be pressed nothing happen.

Figure 40 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 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.

KEYCLICK ON WITHOUT MEMORIZATION

<table>
<thead>
<tr>
<th>Code:</th>
<th>27 53</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hex code:</td>
<td>1B 35</td>
</tr>
<tr>
<td>Mnemonic:</td>
<td>ESC 5</td>
</tr>
</tbody>
</table>

The keyclick function is switched on so there is a sound feedback when a key is pressed: in detail the buzzer is enabled for a short time if it was disabled and viceversa it is disabled for a short time if it was enabled.

This setting is not saved inside the on board EEPROM so after a power or initialization it goes back to the previous condition, already saved on this memory.

KEYCLICK OFF WITHOUT MEMORIZATION

<table>
<thead>
<tr>
<th>Code:</th>
<th>27 54</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hex code:</td>
<td>1B 36</td>
</tr>
<tr>
<td>Mnemonic:</td>
<td>ESC 6</td>
</tr>
</tbody>
</table>

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 after a power or initialization it goes back to the previous condition, already saved on this memory.
KEYCLICK ON WITH MEMORIZATION

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

The **keyclick** function is switched on so there is a sound feedback when a key is pressed: in detail the buzzer is enabled for a short time if it was disabled and vice versa it is disabled for a short time if it was enabled.

The command store this setting on the on board EEPROM and so maintained also after a power or initialization.

**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.

KEYCLICK OFF WITH MEMORIZATION

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

The **keyclick** function is disabled so there is not sound feedback when a key is pressed.

The command store this setting on the on board EEPROM and so maintained also after a power or initialization.

**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.

---

![Figure 51: Photo of frontal with keyboard](image)
COMMANDS FOR USER CHARACTERS

QTP 12/R84 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 table in appendix B). The user characters can be defined and saved with a pattern equal to a 5 x 8 pixels matrix, so organized:

![Figure 52: User Characters Pattern](image)

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 character are saved their patterns are written on EEPROM and then they are reloaded on display any time the terminal is powered on or initialized.

**NOTE:** On QTP 12/R84-F2 the character has a 5 x 7 pixels matrix (Pat 0+Pat 6) and 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 Pat 0 + Pat 7
**Hex code:** 1B 42 nchar Pat 0 + Pat 7

**Mnemonic:** ESC B ASCII(nchar) ASCII(Pat 0) + ASCII(Pat 7)

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

- **nchar** (0÷7) (00÷7 Hex) -&gt; Number of user character to define
- **Pat 0** (0÷31) (00÷1F Hex) -&gt; First byte of pattern equal to first high row of character.
- **Pat 7** (0÷31) (00÷1F Hex) -&gt; 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 `Pat 0 ÷ Pat 7`, as described in figure 52; the pattern is only defined but not saved, so if QTP 12/R84 is turned off and on or initialized, the user character `nchar` doesn't maintain 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 31
1B 42 05 1F 11 11 11 11 1F Hex
```

DEFINITION AND MEMORIZATION OF USER CHARACTER

**Code:** 27 33 66 nchar Pat 0 + Pat 7
**Hex code:** 1B 21 42 nchar Pat 0 + Pat 7

**Mnemonic:** ESC ! B ASCII(nchar) ASCII(Pat 0) + ASCII(Pat 7)

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

- **nchar** (0÷7) (00÷7 Hex) -&gt; Number of user character to define and save
- **Pat 0** (0÷31) (00÷1F Hex) -&gt; First byte of pattern equal to first high row of character.
- **Pat 7** (0÷31) (00÷1F Hex) -&gt; 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 `Pat 0 ÷ Pat 7`, as described in figure 52; moreover the pattern is also saved on EEPROM, so if QTP 12/R84 is turned off and on or initialized, the user character `nchar` maintain 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 serial firmware .SER is used and several commands must follow, it is better to insert a delay to avoid receive buffer overflow.
COMMANDS FOR CURSOR ATTRIBUTES MANAGEMENT

Below are listed the commands that define the possible cursor attribute. Please remind that cursor can be visible only in alphanumerical mode, while in graphic mode it is managed but not shown. It is possible, anyway, to define position and style for cursor also in graphic mode by using alternatively graphic and alphanumerical commands.

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 current position character.

NOTE: This command can be used only on model QTP 12/R84-F2 and QTP 12/R84-C2 that has alphanumerical display. On QTP 12/R84-GF2, with graphic display, the command has no effects.

BLINKING BLOCK CURSOR ON

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

The cursor is enabled and so it is visible as a blinking rectangular block that is alternatively visualized with the character displayed on the current cursor position.
COMMANDS FOR BUFFERED DIGITAL I/Os MANAGEMENT

Below are listed the commands that manage the buffered digital I/O signals available in QTP 12/R84 firmwares. As described in all following paragraphs the digital I/Os are referenced by the same name used in connectors descriptions (see figures 24+29) and when required, a sequential numeration starting from 1.

WRITE ALL DIGITAL OUTPUTS

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

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

(MSB) 0 0 0 0 OUT B2 OUT B1 OUT A2 OUT A1 (LSB)

Where OUT Nn stands for the logic state, 0 (output disabled) or 1 (output enabled), that the respective relay outputs, on CN4, must assume.

Any possible timed commands in execution on all the output lines are interrupted.

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

If, for example, only the OUT A2 and OUT B2 outputs must be enabled, then the following sequence must be sent:

27 166 10 or 1B A6 0A Hex or ESC ASCII(166) LF

ACQUIRE ALL DIGITAL INPUTS

| Code:     | 27 167          |
| Hex code: | 1B A7           |
| Mnemonic: | ESC ASCII(167) |

The command first acquires and then returns the status of all optocoupled digital inputs IN0÷7 connected to CN5. The status is returned as a value with the following correspondence:

(MSB) IN 8 IN 7 IN 6 IN 5 IN 4 IN 3 IN 2 IN 1 (LSB)

Where the bits IN n, stand for the logic state 0 (ON = input closed) or 1 (OFF = input open), currently found on the relative input lines.

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

If, for example, on CN5 are closed only the IN 6 and IN5 inputs, then the following data is returned as answer of the command:

48 or 30 Hex or 0
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 enabled) the relay digital output identified by bit parameter, that has the following correspondence with CN4 signals:

1 -> OUT A1  2 -> OUT A2  3 -> OUT B1  4 -> OUT B2

Any possible timed command in execution on the specified output line is interrupted. When the command sequence contains invalid data, the command is ignored.

If, for example, the output OUT A2 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
```

TIMED ENABLE OF SINGLE DIGITAL OUTPUT

**Code:** 27 169 bit tmp
**Hex code:** 1B A9 bit tmp
**Mnemonic:** ESC ASCII(169) ASCII(bit) ASCII(tmp)

This command sets to logic state 1 (output enabled), for a specified time period, the relay digital output identified by bit parameter, that has the following correspondence with CN4 signals:

1 -> OUT A1  2 -> OUT A2  3 -> OUT B1  4 -> OUT B2

The selected output remains in this state for a period of time specified by tmp parameter and then returns to logic state 0 (output disabled); the time period is measured in tens of msec and it must be included in range 1÷255.

Any possible timed command in execution on the specified output line is interrupted. When the command sequence contains invalid data, the command is ignored.

If, for example, the output OUT A2 must be enabled for 500 msec, with no modifications on the remaining outputs, then the following sequence must be sent:

```
27 169 2 50  or  1B A9 02 32 Hex  or  ESC ASCII(169) STX 2
```
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 disabled) the relay digital output identified by bit parameter, that has the following correspondence with CN4 signals:

1 -> OUT A1  
2 -> OUT A2  
3 -> OUT B1  
4 -> OUT B2

Any possible timed command in execution on the specified output line is interrupted. When the command sequence contains invalid data, the command is ignored.

If, for example, the output OUT A1 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
```

TIMED DISABLE OF SINGLE DIGITAL OUTPUT

**Code:** 27 171 bit tmp
**Hex code:** 1B AB bit tmp
**Mnemonic:** ESC ASCII(171) ASCII(bit) ASCII(tmp)

This command sets to logic state 0 (output disabled), for a specified time period, the relay digital output identified by bit parameter, that has the following correspondence with CN4 signals:

1 -> OUT A1  
2 -> OUT A2  
3 -> OUT B1  
4 -> OUT B2

The selected output remains in this state for a period of time specified by tmp parameter and then returns to logic state 1 (output enabled); the time period is measured in tens of msec and it must be included in range 1÷255.

```
1 = Enable
0 = Disable
```

**FIGURE 54: TIMED DISABLE OF SINGLE DIGITAL OUTPUT, COMMAND**

Any possible timed command in execution on the specified output line is interrupted. When the command sequence contains invalid data, the command is ignored.

If, for example, the output OUT A1 must be disabled for 550 msec, with no modifications on the remaining outputs, then the following sequence must be sent:

```
27 171 1 55  
or 1B AB 01 37 Hex  
or ESC ASCII(171) SOH 7
```
ACQUIRE SINGLE DIGITAL INPUT

Code: 27 172 bit
Hex code: 1B AC bit
Mnemonic: ESC ASCII(172) ASCII(bit)

The command acquires and returns the status of buffered digital input specified by bit parameter, that has the following correspondence with CN5 signals:

1 -> IN 1  
2 -> IN 2  
7 -> IN 7  
8 -> IN 8

The returned value coincides with the logic status of the selected input; in detail the value is 0 for closed input or 1 for open input.

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

If, for example, the status of input IN 7 must be acquired, then the following sequence must be sent:

27 172 7 or 1B AC 07 Hex or ESC ASCII(172) BEL

ACQUIRE SINGLE DIGITAL INPUT WITH DEBOUNCING

Code: 27 173 bit tmp
Hex code: 1B AD bit tmp
Mnemonic: ESC ASCII(173) ASCII(bit) ASCII(tmp)

The command acquires and returns the debounced status of buffered digital input specified by bit parameter, that has the following correspondence with CN5 signals:

1 -> IN 1  
2 -> IN 2  
7 -> IN 7  
8 -> IN 8

Contrary to previous command the acquisition is performed with a debouncing process that belong a time period, measured in tens of msec, defined by tmp parameter included in range 1÷255.

The returned value can assume one of the following values:

0 -> Input closed during the whole debouncing time
1 -> Input opened during the whole debouncing time
7 -> Input changed during the debouncing time

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

If, for example, the status of input IN 7 must be acquired, with a 50 msec debouncing time, then the following sequence must be sent:

27 173 7 5 or 1B AD 07 05 Hex or ESC ASCII(173) BEL ENQ

NOTE: The execution time of this command is equal to the debouncing time passes in tmp parameter, so the answer is returned once this time is elapsed. When serial firmware .SER is used and several commands must follow, it is better to insert a delay to avoid receive buffer overflow.
TIMED SQUARE WAVE STARTING WITH "1" ON SINGLE DIGITAL OUTPUT

**Code:** 27 174 bit tmp per.n.
**Hex code:** 1B AE bit tmp per.n.
**Mnemonic:** ESC ASCII(174) ASCII(bit) ASCII(tmp) ASCII(per.n.)

The command generates a timed square wave, with starting logic state 1 and 50% duty cycle (output enabled and disabled alternatively for equal times), on the relay digital output identified by bit parameter, that has the following correspondence with CN4 signals:

1 -> OUT A1  
2 -> OUT A2  
3 -> OUT B1  
4 -> OUT B2

The duration of enable and disable time, or time period, is defined by tmp parameter, measured in tens of msec, that must be included in range 1÷255.

Instead the per.n. parameter specifies the number of periods, or status changes, required on the selected outputs; as described in the following figure the firmware generates per.n. periods equal to per.n.+1 commutations. Also the value of this data must be included in range 1÷255.

**Figure 55: Timed Square Wave Starting with 1, Command**

Any possible timed command in execution on the specified output line is interrupted. When the command sequence contains invalid data, the command is ignored.

If, for example, on the output OUT B2 must be generated a timed square wave of 10 periods, each one with 250 msec time period, with no modifications on the remaining outputs, then the following sequence must be sent:

27 174 4 25 10 or 1B AE 04 19 0A Hex or ESC ASCII(174) EOT EM LF

**NOTE:** As illustrated in figure 55, the final status of the output will depend on the performed periods number; in particular an even number of periods number (= odd number of commutations) will leave the output in status 1 (output enabled) and viceversa.
TIMED SQUARE WAVE STARTING WITH "0" ON SINGLE DIGITAL OUTPUT

**Code:**  27 175 bit tmp per.n.
**Hex code:**  1B AF bit tmp per.n.
**Mnemonic:**  ESC ASCII(175) ASCII(bit) ASCII(tmp) ASCII(per.n.)

The command generates a timed square wave, with starting logic state 0 and 50% duty cycle (output enabled and disabled alternatively for equal times), on the relay digital output identified by **bit** parameter, that has the following correspondence with CN4 signals:

1 -> OUT A1   2 -> OUT A2   3 -> OUT B1   4 -> OUT B2

The duration of enable and disable time, or time period, is defined by **tmp** parameter, measured in tens of msec, that must be included in range 1÷255.

Instead the **per.n.** parameter specifies the number of periods, or status changes, required on the selected outputs; as described in the following figure the firmware generates **per.n.** periods equal to **per.n.+1** commutations. Also the value of this data must be included in range 1÷255.

Any possible timed command in execution on the specified output line is interrupted.

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

If, for example, on the output OUT B2 must be generated a timed square wave of 40 periods, each one with 200 msec time period, with no modifications on the remaining outputs, then the following sequence must be sent:

`27 175 4 20 40` or `1B AE 04 14 28 Hex` or `ESC ASCII(175) EOT DC4`

**NOTE:** As illustrated in figure 56, the final status of the output will depend on the performed periods number; in particular an even number of periods number (= odd number of commutations) will leave the output in status **0** (output disabled) and viceversa.
SQARE WAVE ON SINGLE DIGITAL OUTPUT

*Code:* 27 176 bit tmp  
*Hex code:* 1B B0 bit tmp  
*Mnemonic:* ESC ASCII(176) ASCII(bit) ASCII(tmp)

The command generates a continuous square wave with 50% duty cycle (output enabled and disabled alternatively for equal times), on the relay digital output identified by the `bit` parameter, that has the following correspondence with CN4 signals:

1 -> OUT A1  
2 -> OUT A2  
3 -> OUT B1  
4 -> OUT B2

The duration of enable and disable time, or time period, is defined by the `tmp` parameter, measured in tens of msec, that must be included in range 1÷255.

![Figure 57: Square Wave, Command](image)

Any possible timed command in execution on the specified output line is interrupted. When the command sequence contains invalid data, the command is ignored.

If, for example, on the output OUT B1 must be generated a square wave with 500 msec time period (equal to a 1000 msec wave period and thus a 1 Hz frequency), with no modifications on the remaining outputs, then the following sequence must be sent:

27 176 3 50 or 1B B0 03 32 Hex or ESC ASCII(176) ETX 2

**NOTE:** The square wave generation started by this command can be stopped by any other command that performs operation on single digital output; the most appropriated are the ENABLE and DISABLE SINGLE DIGITAL OUTPUT that moreover allow to define a required status on the output.

Among the most frequently uses of this command, and the previous ones, we remind the time base generation required by external systems, the repeated and periodic automation commands, (i.e. the pieces folding), the status indicators command (semaphores, hooters, etc.), the programmed moving of pieces, etc.
COMMANDS FOR ALTERNATIVE FUNCTIONS OF DIGITAL I/OS

In addition to buffered digital I/Os management described in previous paragraphs, the QTP 12/R84 firmwares offer other functions, defined alternative functions, that always solve the typical problems of industrial automation. The digital I/O signals used by these commands are referenced with the same names used in connectors descriptions (see figures 24+29).

SET MULTIFUNCTIONS PINS

<table>
<thead>
<tr>
<th>Code:</th>
<th>27 179 bit cnf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hex code:</td>
<td>1B B3 bit cnf</td>
</tr>
<tr>
<td>Mnemonic:</td>
<td>ESC ASCII(179) ASCII(bit) ASCII(cnf)</td>
</tr>
</tbody>
</table>

Defines the configuration of buffered digital I/O signals identified by bit parameter, using the following correspondence with CN4 and CN5 signals:

<table>
<thead>
<tr>
<th>bit</th>
<th>cnf</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1+12</td>
<td>0</td>
<td>Normal input output line, without alternative functions</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>16 bits hw counter associated to Timer 2 of microcontroller</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>16 bits hw/fw counter associated to /INT0 signal of microcontroller</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>16 bits hw/fw counter associated to /INT1 signal of microcontroller</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>16 bits hw counter associated to Timer 1 of microcontroller</td>
</tr>
</tbody>
</table>

The pin configuration, passed in cnf parameter, defines its functionality and to make the command more flexible and expandible, four different configurations, numbered from 0 to 3, have been provided. In current firmwares version are available only few alternative functions on some pins, as illustrated in the following list:

Whenever the command sequence sent to firmware includes not valid data, the command is ignored. After a power on or initialization, all the alternative functions are disabled and all I/O signals are configured as simple digital inputs and outputs.

If, for example, the digital input IN 1 must be configured as 16 bits counter, then the following sequence must be sent:

```
27 179 1 1 or 1B B3 01 01 Hex or ESC ASCII(179) SOH SOH
```

READ MULTIFUNCTIONS PINS SETTING

<table>
<thead>
<tr>
<th>Code:</th>
<th>27 180 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hex code:</td>
<td>1B B4 bit</td>
</tr>
<tr>
<td>Mnemonic:</td>
<td>ESC ASCII(180) ASCII(bit)</td>
</tr>
</tbody>
</table>

Returns the configuration of buffered digital I/O signal identified by bit parameter, with the modalities described in previous paragraph.

If, for example, the configuration of digital input IN 6 must be acquired, then the following sequence must be sent:

```
27 180 6 or 1B B4 06 Hex or ESC ASCII(180) ACK
```

and the value 1 is returned if the input is configured as 16 bits counter, otherwise the value 0 if the input have no alternative functions.
READ COUNTER

**Code:** 27 177 cnt  
**Hex code:** 1B B1 cnt  
**Mnemonic:** ESC ASCII(177) ASCII(cnt)

Returns the current value of 16 bits counter, identified by \texttt{cnt} parameter with the following correspondence:

<table>
<thead>
<tr>
<th>cnt</th>
<th>signal</th>
<th>Counter</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IN 8</td>
<td>16 bits hw counter associated to Timer 1 of microcontroller</td>
</tr>
<tr>
<td>2</td>
<td>IN 1</td>
<td>16 bits hw counter associated to Timer 2 of microcontroller</td>
</tr>
<tr>
<td>3</td>
<td>IN 5</td>
<td>16 bits hw/fw counter associated to /INT0 signal of microcontroller</td>
</tr>
<tr>
<td>4</td>
<td>IN 6</td>
<td>16 bits hw/fw counter associated to /INT1 signal of microcontroller</td>
</tr>
</tbody>
</table>

When the command is executed two values are returned that forms the 16 bits combination currently saved inside the counter: the first one is the low byte (LSB) and the second is the high byte (MSB). Independently from the used counter the firmware recognizes as pulses to count the falling edges of the signal, equal to closing of the optocoupled input. When counter reaches its maximum value that is 65535 (FFFF Hex), in correspondence of next pulse on input signal the new value of combination will be 0.

If the pin used as input signal is not configured as counter (through proper SET MULTIFUNCTIONS PINS command) this command always returns a 0 combination.

Whenever the command sequence sent to firmware includes invalid data, the command is ignored.

After a power on or initialization, all the counters are reset to 0 value.

If, for example, the counter 2 associated to digital input IN 1 must be read, then the following sequence must be sent:

\[
27 \ 177 \ 2 \quad \text{or} \quad 1B \ B1 \ 02 \text{ Hex} \quad \text{or} \quad \text{ESC ASCII(177) STX}
\]

and assuming that the two values 100 and 15 (64 and 0F Hex) are returned, the counter currently contains the combination 3940 (0F64 Hex).

RESET COUNTER

**Code:** 27 178 cnt  
**Hex code:** 1B B2 cnt  
**Mnemonic:** ESC ASCII(178) ASCII(cnt)

The command resets the current 16 bits value of the counter identified by \texttt{cnt} parameter, with the correspondence reported in previous paragraph.

Among the most frequently uses of the commands relative to counters, we remind the product pieces counting, the position sensors acquisition (even bidirectional encoders), the people transition counting inside access control systems, etc.
COMMAN D S FOR G RAP HICS

QTP 12/R84-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., through a short group of simple graphic commands.

All graphic commands are based on the smallest visible entity of display, that are points or pixels that are organized in the coordinates system described in the following figure:

![Figure 58: Coordinates of graphic display pixels](image)

**NOTE:** Please remind that following graphic commands can be used only on model QTP 12/R84-GF2 while the remaining models do not recognize them as commands so they show the character of the command sequence on display.

In addition, QTP 12/R84-GF2 recognizes both graphic and alphanumeric commands already described in previous paragraphs.

**ALPHANUMERIC MODE SETTING**

<table>
<thead>
<tr>
<th>Code:</th>
<th>27 208</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hex code:</td>
<td>1B D0</td>
</tr>
<tr>
<td>Mnemonic:</td>
<td>ESC ASCII(208)</td>
</tr>
</tbody>
</table>

This command sets alphanumeric representation mode, which allows to use all alphanumeric commands. When command is executed, the cursor (if enabled) is shown in the last position decided by previous commands.

After power on or initialization it is immediately selected alphanumeric mode to make all functionalities available.
Figure 59: First graphic example

Figure 60: Second graphic example
**GRAPHIC MODE SETTING**

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

This command sets graphic mode, that enables the interpretation of caharacters sent to firmware as graphic data and not as commands. When this command is executed there is no effect on display, but characters received are no more checked for one characters commands and they go directly to graphic display. So one character commands are not executed until alphanumeric mode is restored. On the other hand, two or more characters commands, starting with ESC = 27 = 1BH, are always checked and executed, independently from selected mode.

After power on or initialization, alphanumeric mode is automatically selected by firmwares to allow the use of all commands.

**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 parameters is absolute, so it is not affected by all other settings and it is independent from normal alphanumeric cursor placement. Characters to show, received after this command are displayed from indicated point, and they 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 are the size of used display.

If, for example, the graphic cursor must be placed on pixel with coordinates (10, 100), then the following sequence must be sent:

27 206 100 10 0  or  1B CE 64 0A 00 Hex  or  ESC ASCII(206) d LF NUL

**NOTE:** Code 0 (NUL) described at the end of command sequence, is present for compatibility with future expansions and for compatibility with other terminals: it must be always transmitted anyway to ensure correct command execution.

**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)

Defines graphic work area and the action to make on it. The graphic area has a top left corner equal to pixel with coordinates x1, y1 and bottom right corner placed on coordinates x2, y2. The values of y1, y2 and x1, x2 must be respectively in the ranges 0+15 and 0+139, that are the size of used display.

Byte cmd selects the action to perform on the defined graphic area and thus the function of the next bytes the firmware receive, as described in the following list:
\[ \text{cmd} = 67 \text{ (43 Hex) C} \rightarrow \text{Clears selected area.} \]

\[ 70 \text{ (46 Hex) F} \rightarrow \text{Fills selected area.} \]

\[ 72 \text{ (48 Hex) H} \rightarrow \text{Draws the selected area with following horizontal graphic data, with horizontal shift.} \]

\[ 73 \text{ (49 Hex) I} \rightarrow \text{Inverts selected area.} \]

\[ 79 \text{ (4F Hex) O} \rightarrow \text{Draws a frame around selected area.} \]

\[ 86 \text{ (56 Hex) V} \rightarrow \text{Draws the selected area with following vertical graphic data, with horizontal shift.} \]

\[ 104 \text{ (68 Hex) h} \rightarrow \text{Draws the selected area with following horizontal graphic data, with vertical shift.} \]

\[ 111 \text{ (6F Hex) o} \rightarrow \text{Deletes a frame around selected area.} \]

\[ 118 \text{ (76 Hex) v} \rightarrow \text{Draws the selected area with following vertical graphic data, with vertical shift.} \]

About commands that draw data in graphic area (H,h,V,v) next bytes sent to the terminal are used as graphic data that decide pixels activation of display. The correspondance between display pixels and bits of these bytes is explained in figures 62-65 where all the four organization and shift modes are described.

Logic status 1 of a bit correspond to \textbf{activation} of corresponding pixel and viceversa logic status 0 of bit correspond to \textbf{deactivation} of pixel.

The draw selected area commands can be completed in two ways: by filling all the selected area or by interruption caused from another command; naturally this latter condition stops execution of the first command and so only pixels already represented at that moment will be visualized.

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

\[ \text{F I G U R E 61: E X A M P L E O F G R A P H I C D R A W I N G} \]

First send the command sequence:

\[ 27 241 0 0 15 8 72 \]  \quad \text{or}  \quad 1B F1 00 00 0F 08 48 \text{ Hex} \]

And then graphic data sequence:

\[ \begin{array}{cccccccccccc}
0 & 0 & 0 & 255 & 255 & 255 & 0 & 0 & 0 & 32 & 48 & 56 & 252 & 254 & 252 & 56 & 48 & 32 \quad \text{or} \\
00 & 00 & 00 & FF & FF & FF & 00 & 00 & 00 & 20 & 30 & 38 & FC & FE & FC & 38 & 30 & 20 \quad \text{Hex}
\end{array} \]
FIGURE 62: Horizontal graphic data and horizontal shift

FIGURE 63: Horizontal graphic data and vertical shift
**Figure 64:** Vertical graphic data and horizontal shift

**Figure 65:** Vertical graphic 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 used for next alphanumeric characters visualization, in graphic mode. If graphic mode is selected and a drawing area command is not under execution (as already stated this command uses received bytes as graphic data), then the received bytes are shown on display as characters, anyway. In this last condition a font can be selected that is different from the one used in alphanumeric mode.

The available graphic fonts can be selected with parameter **font**:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>65</td>
<td>A -&gt; Proportional spacing minifont, 3x5÷5x5 pixels.</td>
</tr>
<tr>
<td>97</td>
<td>a -&gt; Proportional spacing minifont, 3x5÷5x5 pixels.</td>
</tr>
<tr>
<td>66</td>
<td>B -&gt; Katakana font, 5x7 pixels.</td>
</tr>
<tr>
<td>67</td>
<td>C -&gt; Katakana font, 10x14 pixels.</td>
</tr>
<tr>
<td>98</td>
<td>b -&gt; European font, 5x7 pixels.</td>
</tr>
<tr>
<td>99</td>
<td>c -&gt; European font, 10x14 pixels.</td>
</tr>
<tr>
<td>49</td>
<td>1 -&gt; 1 pixel interspacing.</td>
</tr>
<tr>
<td>50</td>
<td>2 -&gt; 2 pixels interspacing.</td>
</tr>
</tbody>
</table>

The first six font selection commands are mutually exclusive, while the interspacing selection commands add their effect to the first ones. So, each of the five fonts can be set with interspacing of 1 or 2 pixels, obtaining 10 different fonts. The font selected is used only in graphic mode, while in alphanumeric mode only the classic font shown in figure B1 with 1 pixel interspacing, is used.

After a power on or initialization, alphanumeric mode is automatically enabled and the Katakana font, 5x7, 1 pixel interspacing is automatically selected for graphic mode.

For further information about available characters with described fonts, please refer to APPENDIX B, while figure 38 shows a photo with three of the ten available fonts.
COMMANDS FOR 1-WIRE COMMUNICATION

The TTL I/O lines available on QTP 12/R84 are used by firmwares to implement two communication BUS with 1-WIRE protocol. Through the commands explained in the following paragraphs it is possible to manage the several devices developed to work with this standard like temperature sensors, memories, Dallas iButton™, etc.

The TTL I/O signals available on connector J6 are used as follows:

- **Pin 2 J6** -> 1-WIRE line n. 1
- **Pin 3 J6** -> 1-WIRE line n. 2

As stated in the following paragraphs, the high level commands available do not support the presence of more than one device on each 1-WIRE line in fact, for example, there is not the command “Search ROM”, whose task is to search for ROM codes of the device connected to the line.

It is possible, anyway, to manage a network connection of many 1-WIRE units taking advantage of the low level commands (line reset, get and set a bit status, read and write bytes); in this case the implementation becomes more articulate and it requires also an heavy data exchange between the user application program and the QTP 12/R84 firmware.

So it is suggestable to connect the board to at most two 1-WIRE devices; the following figure reports a connection example with two temperature sensors, Dallas DS18s20.

![Connection Example with Two 1-WIRE Devices](image)

*Figure 66: Connection example with two 1-WIRE devices*

Anyhow any users is invited to get some general documentation about 1-WIRE protocol in fact this manual uses the typical names, data and commands of this standard but it doesn't provide specific information on their meanings.
RESET 1-WIRE LINE

Code: 27 181 wire
Hex code: 1B B5 wire
Mnemonic: ESC ASCII(181) ASCII(wire)

The commands generates a reset sequence on the 1-WIRE line indicated by wire parameter. The value of wire must be 1 or 2, otherwise the command is ignored. At the end of the reset sequence it is acquired and returned the presence impulse of a possible device connected on the line with the following correspondence:

\[\begin{align*}
0 & \quad \text{1-WIRE device present and ready to receive commands} \\
1 & \quad \text{1-WIRE device not present}
\end{align*}\]

If, for example, the reset sequence must be generated and presence impulse must be read on 1-WIRE line n. 1, then the following sequence must be sent:
27 181 1 or 1B B5 01 Hex or ESC ASCII(181) SOH

WRITE BIT ON 1-WIRE LINE

Code: 27 182 wire bit
Hex code: 1B B6 wire bit
Mnemonic: ESC ASCII(182) ASCII(wire) ASCII(bit)

The bit value passed as parameter is generated on the 1-WIRE line indicated by wire parameter. The valid values for the two parameters respectively are 0, 1 and 1, 2 otherwise the command is ignored.

If, for example, the bit 1 must be wrote on 1-WIRE line n. 2, then the following sequence must be sent:
27 182 2 1 or 1B B6 02 01 Hex or ESC ASCII(182) STX SOH

READ BIT FROM 1-WIRE LINE

Code: 27 183 wire
Hex code: 1B B7 wire
Mnemonic: ESC ASCII(183) ASCII(wire)

The command acquires a bit from the 1-WIRE line indicated by wire parameter. The valid value for wire are 1 or 2, otherwise the command is ignored.
Once the bit read operation is performed the command returns the logic status (0 or 1) of the bit acquired from the specified line.

If, for example, a bit must be read from 1-WIRE line n. 1, then the following sequence must be sent:
27 183 1 or 1B B7 01 Hex or ESC ASCII(183) SOH
WRITE N BYTES ON 1-WIRE LINE

Code:  27 184 wire n dat0 + datn  
Hex code:  1B B8 wire n dat0 + datn  
Mnemonic:  ESC ASCII(184) ASCII(wire) ASCII(n) ASCII(dat0) + ASCII(datn)

The number of bytes specified by parameter n are sent on the 1-WIRE line indicated by wire parameter. The value of this last parameter must be 1 or 2; the maximum number of bytes to send after command is 10, so the value of n parameter must be in range 1+10 (01+0A Hex). Finally the n bytes dat0 + datn can assume any value in range 0+255 (0+FF Hex).

Whenever the command sequence contains invalid data, the command is ignored.

If there are problem on 1-WIRE line or no device is connected, the command is anyhow executed obtaining no effects.

If, for example, the three bytes 23, 118 and 80 must be wrote on 1-WIRE line n. 2, then the following sequence must be sent:

27 184 2 3 23 118 80 or 1B B8 02 03 17 76 50 Hex or ESC ASCII(184) STX ETX ETB u P

READ N BYTES FROM 1-WIRE LINE

Code:  27 185 wire n  
Hex code:  1B B9 wire n  
Mnemonic:  ESC ASCII(185) ASCII(wire) ASCII(n)

The bytes specified by n parameter are read from the 1-WIRE line indicated by wire parameter. The value of this last parameter must be 1 or 2; the maximum number of bytes to read with command is 10, so the value of n parameter must be in range 1+10 (01+0A Hex).

Whenever the command sequence contains invalid data, the command is ignored.

Once the read operation is performed the command returns the n bytes, acquired from the specified line, with the same order of reading.

If there are problem on 1-WIRE line or no device is connected, the command is anyhow executed obtaining no effects and all the returned data assume the value 255 (FF Hex).

If, for example, four bytes must be read from 1-WIRE line n. 2, then the following sequence must be sent:

27 185 2 4 or 1B B9 02 04 Hex or ESC ASCII(185) STX EOT  
and assuming that the 1-WIRE device supplies the byte 23, 118, 80 and 13 during reading, the following data are returned:

23 118 80 13 or 17 76 50 0D Hex or ETB u P CR
"READ ROM" ON 1-WIRE LINE

**Code:** 27 186 wire  
**Hex code:** 1B BA wire  
**Mnemonic:** ESC ASCII(186) ASCII(wire)

This command performs the following operations on the 1-WIRE line specified by `wire` parameter:

a) The reset sequence is sent to test the presence of the device on the specified line and to prepare it to receive next command.

b) If the device is present, the "Read ROM" command (code 33 Hex of 1-WIRE protocol) is sent, and then the device ROM code is acquired.

The value of `wire` parameter must be 1 or 2, otherwise the command is ignored.

The response of this command is made of 8 bytes (`rom0÷rom7`) with the following meaning:

- **Device 1-WIRE present and command executed successfully:**  
  the 8 bytes of the answer are the device ROM code: `rom0` = Family code, `rom1÷rom6` = Serial number and `rom7` = CRC.

- **Device 1-WIRE not present and command not sent:**  
  the 8 bytes of the answer are all 0's.

If, for example, the ROM code of the device connected to 1-WIRE line n. 2 must be read, then the following sequence must be sent:

27 186 2  or  1B BA 02 Hex  or  ESC ASCII(186) STX

and assuming that the ROM code of 1-WIRE device is: Family code=16, Serial number=56, 198, 13, 0, 8, 0 and CRC=226, the following data are returned:

16 56 198 13 0 8 0 226  or  10 38 C6 0D 00 08 00 E2 Hex  or  DLE 8 ASCII(198) CR NUL BS NUL ASCII(226)

"SKIP ROM" ON 1-WIRE LINE

**Code:** 27 188 wire  
**Hex code:** 1B BC wire  
**Mnemonic:** ESC ASCII(188) ASCII(wire)

This command performs the following operations on the 1-WIRE line specified by `wire` parameter:

a) The reset sequence is sent to test the presence of the device on the specified line and to prepare it to receive next command.

b) If the device is present, the "Skip ROM" command (code CC Hex of 1-WIRE protocol) is sent.

The value of `wire` parameter must be 1 or 2, otherwise the command is ignored.

At the end of described operations an answer data is returned with next possible values:

0  ->  Device 1-WIRE present and command sent
1  ->  Device 1-WIRE not present and command not sent

If, for example, the "Skip ROM" command must be sent to device connected to 1-WIRE line n. 1, then the following sequence must be sent:

27 188 1  or  1B BC 01 Hex  or  ESC ASCII(188) SOH
"MATCH ROM" ON 1-WIRE LINE

| Code:   | 27 187 wire rom0 ∵ rom7 |
| Hex code: | 1B BB wire rom0 ∵ rom7 |
| Mnemonic: | ESC ASCII(187) ASCII(wire) ASCII(rom0) ∵ ASCII(rom7) |

This command performs the following operations on the 1-WIRE line specified by wire parameter:

a) The reset sequence is sent to test the presence of the device on the specified line and to prepare it to receive next command.

b) If the device is present, the "Match ROM" command (code 55 Hex of 1-WIRE protocol) is sent, followed by the passed ROM code: 
   - rom0 = Family code,
   - rom1 ∵ rom6 = Serial number,
   - rom7 = CRC.

The value of wire parameter must be 1 or 2, while the 8 bytes rom0 ∵ rom7 may vary in the range 0 ∵ 255 (00 ∵ FF Hex).

Whenever the command sequence contains invalid data, the command is ignored.
At the end of described operations an answer data is returned with next possible values:

| 0 | -> Device 1-WIRE present and command sent |
| 1 | -> Device 1-WIRE not present and command not sent |

If, for example, the "Match ROM" command must be sent to device connected to 1-WIRE line n. 1 to match the ROM code: Family code=16, Serial number=56, 198, 13, 0, 8, 0 and CRC=226, then the following sequence must be sent:

27 187 1 16 56 198 13 0 8 0 226 or 1B BB 01 10 38 C6 0D 00 08 00 E2 Hex
or ESC ASCII(187) SOH DLE 8 ASCII(198) CR NUL BS NUL ASCII(226)

**Figure 67: Functional diagram of 1-WIRE protocol**
"ALARM SERCH" ON 1-WIRE LINE

Code: 27 189 wire  
Hex code: 1B BD wire  
Mnemonic: ESC ASCII(189) ASCII(wire)

This command performs the following operations on the 1-WIRE line specified by wire parameter:

a) The reset sequence is sent to test the presence of the device on the specified line and to prepare it to receive next command.

b) If the device is present, the "Alarm search" command (code EC Hex of 1-WIRE protocol) is sent and then it is checked if the device has the alarm flag set or not.

The value of wire parameter must be 1 or 2, otherwise the command is ignored.

At the end of described operations an answer data is returned with next possible values:

0 -> Device 1-WIRE present with alarm flag not set  
1 -> Device 1-WIRE present with alarm flag set  
7 -> Device 1-WIRE not present and command not sent

If, for example, the "Alarm Search" command must be sent to device connected to 1-WIRE line n. 1, then the following sequence must be sent:

27 189 1 or 1B BD 01 Hex or ESC ASCII(189) SOH
EXTERNAL CARDS

The typical application of **QTP 12/R84** are those that require interaction between operator and controlled machine for data, status, measures, information exchanges, actuations. The card can be connected to a wide range of programmable devices provided of one serial communication line produced by grifo®, or to many system of other companies.

Hereunder some of these cards are briefly described; ask the detailed information directly to grifo®, or search it on grifo® CD or web site, if required.

**MSI 01**  
Multi Serial Interface 1 line
Interface card for TTL serial line that is buffered in RS 232, RS 422, RS 485, or Current loop line. The TTL line is on a mini screw connector and the buffered one is on standard 6 pins plug connector.

**IBC 01**  
Interface Block Communication 1 line
Conversion card for serial communication, 2 RS 232 lines; 1 RS 422 or RS 485 line; 1 optical fibre line; selectable DTE/DCE interface; status LEDs; quick connection for DIN 46277-1 and 3 rails; wide range power supply.

**GPC® 553**  
General Purpose Controller 80C552
80C552 µP, 22÷33 MHz; 1 RS 232 line (software); 1 RS 232 or RS 422-485 or Current loop line; 16 TTL I/O lines; 8 A/D 10 bits lines; 3 Timers Counters; 32K EPROM; 32K RAM; socket for 32K RAM,RTC backed, EEPROM, FLASH; 8K serial EEPROM; 2 PWM lines; 1 activity LED; Watch dog; 5 readable DIPs; LCD Interface; **ABACO®** I/O BUS expansion interface.Wide power supply.

**GPC® 884**  
General Purpose Controller Am188ES
Microprocessor AMD Am188ES up to 40 MHz16 bits; implementation completely CMOS; serie 4 format; 512K EPROM or FLASH; 512K SRAM backed with Lithium battery; RTC; 1 RS 232 serial line + 1 RS 232 or RS 422-485 or Current loop; 16 I/O TTL; 3 timer/counter; watch dog; EEPROM; 11 A/D converter signals with 12 bits resolution; interface for **ABACO®** I/O BUS.

**GPC® 150**  
General Purpose Controller 84C15
Microprocessor Z80 at 16 MHz; implementation completely CMOS; 512K EPROM or FLASH; 512K SRAM; RTC; Back up through external Lithium battery; 4M serial FLASH; 1 serial line RS 232 plus 1 RS 232 or RS 422-485 or Current loop; 40 I/O TTL; 2 timer/counters; 2 watch dogs; dip switch; serial EEPROM; A/D converter with resolution 12 bit; activity and status LEDs.

**GPC® R/T94**  
General Purpose Relays/Transistors 9 inputs 4 outputs
CMOS card, 14 MHz 89C4051 CPU; 4K FLASH; 128 byte RAM; 256 byte SRAM+RTC backed through battery; 1K serial EEPROM; 1 RS 232, RS 422, RS 485 or Current loop line; 9 optocoupled NPN inputs; 4 relays outputs (5 A) or transistor (4A 45 Vdc) optocoupled; I/O lines displayed by LEDs; 1 counters 16 bits; analog inputs with 11 bits precision. Fast release screw terminal connectors. +5 Vdc power supply or 8÷24 Vac wide range; plastic container for Ω DIN 46277-1 and 3 rails.

**QTP 12/R84 Rel. 5.00**
QTP G28
Quick Terminal Panel - LCD Graphic, 28 keys
Operator panel with LCD display 240x128 pixels, CFC backligt; Optocoupled RS 232 line and additional RS 232,422,485. Current loop line; CAN line controller and interface; EEPROM for set up; 256K EPROM or FLASH; 128K RAM; RTC and RAM lithium backed; possibility of renaming keys, LEDs and panel name; Buzzer, 28 keys and 16 LEDs with blinking attribute, manageable by software; built in power supply; reader of magnetic badge and relay option. Plastic and metallic container, IP65 on front side. High level firmware with capabilities of dump terminal provided of commands for primary graphic objects.

UCC A2
UART Comunication Card 2 lines
2 indipendent serial lines in RS 232, RS 422, RS 485 or Current loop. For each line: 3 character receive buffer; baud rate (from 50 to 115K baud), parity, stop bits and bits per character programmable by software; 2 handshake signals. Comunication managed by SCC 85C304 UART; configuration dip switch; status LEDs; interrupts generation; 8 bits BUS; normal addressing.

GMB HR84
grifo® Mini Block Housing, 8 opto inputs, 4 relays outputs
8 optocoupled inputs NPN or PNP visualized through LEDs; some inputs can be counter or interrupt source; 4 relay outputs up to 5 A visualized through 4 LEDs; some outputs can make PCA functions for automatic timed commands; Serial line in RS 232, RS 422, RS 485, Current Loop or TTL. Expansion connector for I2C BUS, possible CAN line and possible analog signal. Switching power supply; logic protection through TransZorb™; DC or AC power supply from 12 Vdc up to 24 Vac.

CAN GM1
CAN grifo® Mini Module T89C51CC01
28 pins CAN Mini Module based on CPU Atmel T89C51CC01 with 32K FLASH; 256 Bytes SRAM;1K ERAM; 2K FLASH for Bootloader; 2K EEPROM; 3 Timer Counter and 5 sections of high performance Timer Counter (PWM, watch dog, compare); RTC + 240 Bytes SRAM, backed by a Lithium battery; I2C BUS; 17 lines of I/O TTL; 8 A/D 10 bit; RS 232; CAN; 2 status LEDs; Dip configuration switch; etc.

SBP 02-xx
Switch BLOCK Power xx version
Low cost switching power supply able to generate voltage from +5 to +40 Vdc and current up to 2.5 A; Input from 12 to 24 Vac; screw terminal quick release connector. Container for DIN C type and Ω rails; very small size; different protection security.

EXPS-2
EXternal Power Supply 2 tensioni
Plug power supply 75x55x90 mm provided with plastic container. Input voltage: 230 Vac, 50 Hz. Output voltages: 24 Vdc, 200 mA and 18 Vac, 300 mA galvanically isolatated. Standard connection for mains power in input and quick release screw terminal connector for output voltages. LED to signal output voltage presence.
**FIGURE 68: AVAILABLE CONNECTIONS DIAGRAM**
BIBLIOGRAPHY

In this chapter there is a complete list of technical data books and sheets, where the user can find all the necessary documentations on the components mounted on QTP 12/R84 board.

ATMEL manual: 
AT89 series

HEWLETT PACKARD manual: 
Optoelectronics Designer’s Catalog

MAXIM manual: 
New Releases Data Book - Volume IV

NATIONAL data sheets: 
LM2825 - Simple Switcer

PHILIPS manual: 
Application notes and development tools for 80C51 microcontrollers

PHILIPS manual: 
80C51 - Based 8-Bit Microcontrollers

PHILIPS manual: 
FC-bus compatible ICs

S.E. data sheets: 
SI series - Switching power supply

SGS-THOMSON manual: 
Small signal transistor - Data Book

TAKAMISAWA manual: 
Relays index Book

TEXAS INSTRUMENTS manual: 
The TTL Data Book - SN54/74 Families

TEXAS INSTRUMENTS manual: 
RS-422 and RS-485 Interface Circuits

TOSHIBA manual: 
Photo couplers - Data Book

The described manuals can be requested directly to manufacturer or local dealers. Alternatively this information and/or upgrades can be found in specific internet web pages, of the listed companies.
APPENDIX A: COMMANDS SUMMARY TABLES

The following tables list a summary of all the commands recognized by QTP 12/R84 firmwares. 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.

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

**FIGURE A1: COMMAND CODES SUMMARY TABLE (1 OF 4)**
<table>
<thead>
<tr>
<th>Command</th>
<th>Code</th>
<th>HEX code</th>
<th>Mnemonic</th>
<th>Ret.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request of EEPROM availability</td>
<td>27 51</td>
<td>1B 33</td>
<td>ESC 3</td>
<td>1</td>
</tr>
<tr>
<td>Write presence byte</td>
<td>27 33 78 byte</td>
<td>1B 21 4E byte</td>
<td>ESC ! N ASCII(byte)</td>
<td>0</td>
</tr>
<tr>
<td>Read presence byte</td>
<td>27 33 110</td>
<td>1B 21 6E</td>
<td>ESC ! n</td>
<td>1</td>
</tr>
<tr>
<td>Write byte on EEPROM</td>
<td>27 164 addl addh byte</td>
<td>1B A4 addl addh byte</td>
<td>ESC ASCII(164) ASCII(addl) ASCII(addh) ASCII(byte)</td>
<td>0</td>
</tr>
<tr>
<td>Read byte from EEPROM</td>
<td>27 165 addl addh</td>
<td>1B A5 addl addh</td>
<td>ESC ASCII(165) ASCII(addl) ASCII(addh)</td>
<td>1</td>
</tr>
<tr>
<td>Key code reconfiguration</td>
<td>27 55</td>
<td>1B 37</td>
<td>ESC 7 ASCII(key n.) ASCII(cod.)</td>
<td>0</td>
</tr>
<tr>
<td>Keyclick on without memorization</td>
<td>27 53</td>
<td>1B 35</td>
<td>ESC 5</td>
<td>0</td>
</tr>
<tr>
<td>Keyclick off without memorization</td>
<td>27 54</td>
<td>1B 36</td>
<td>ESC 6</td>
<td>0</td>
</tr>
<tr>
<td>Keyclick on with memorization</td>
<td>27 33 53</td>
<td>1B 21 35</td>
<td>ESC ! 5</td>
<td>0</td>
</tr>
<tr>
<td>Keyclick off with memorization</td>
<td>27 33 54</td>
<td>1B 21 36</td>
<td>ESC ! 6</td>
<td>0</td>
</tr>
<tr>
<td>Definition of user character</td>
<td>27 66 nchar Pat0÷Pat7</td>
<td>1B 42 nchar Pat0÷Pat7</td>
<td>ESC B ASCII(nchar) ASCII(Pat0÷ASCII(Pat7))</td>
<td>0</td>
</tr>
<tr>
<td>Definition and memorization of user character</td>
<td>27 33 66 nchar Pat0÷Pat7</td>
<td>1B 21 42 nchar Pat0÷Pat7</td>
<td>ESC ! B ASCII(nchar) ASCII(Pat0÷ASCII(Pat7))</td>
<td>0</td>
</tr>
<tr>
<td>Reading of max message number</td>
<td>27 110</td>
<td>1B 6E</td>
<td>ESC n</td>
<td>1</td>
</tr>
<tr>
<td>Message storage</td>
<td>27 33 67 mess.n. chr.0÷chr.19</td>
<td>1B 21 43 mess.n. chr.0÷chr.13</td>
<td>ESC ! C ASCII(mess.n.) ASCII(chr.0÷ASCII(chr.19))</td>
<td>0</td>
</tr>
<tr>
<td>Message reading</td>
<td>27 33 69 mess.n.</td>
<td>1B 21 45 mess.n.</td>
<td>ESC ! E ASCII(mess.n.)</td>
<td>20</td>
</tr>
<tr>
<td>Visualization of n messages</td>
<td>27 33 68 mess.n. n</td>
<td>1B 21 44 mess.n. n</td>
<td>ESC ! D ASCII(mess.n.) ASCII(n)</td>
<td>0</td>
</tr>
<tr>
<td>Scrolling messages visualization</td>
<td>27 33 83 mess.n. n.chr</td>
<td>1B 21 53 mess.n. n.chr</td>
<td>ESC ! S ASCII(mess.n.) ASCII(n.chr)</td>
<td>0</td>
</tr>
</tbody>
</table>

**Figure A2: Command codes summary table (2 of 4)**
<table>
<thead>
<tr>
<th>Command</th>
<th>Code</th>
<th>HEX code</th>
<th>Mnemonic</th>
<th>Ret.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write all digital outputs</td>
<td>27 166</td>
<td>A6 out</td>
<td>ESC ASCII(166) ASCII(out)</td>
<td>0</td>
</tr>
<tr>
<td>Acquire all digital inputs</td>
<td>27 167</td>
<td>A7</td>
<td>ESC ASCII(167)</td>
<td>1</td>
</tr>
<tr>
<td>Enable single digital output</td>
<td>27 168</td>
<td>A8 bit</td>
<td>ESC ASCII(168) ASCII(bit)</td>
<td>0</td>
</tr>
<tr>
<td>Timed enable of single digital output</td>
<td>27 169</td>
<td>A9 bit</td>
<td>ASCII(bit) ASCII(tmp)</td>
<td>0</td>
</tr>
<tr>
<td>Disable single digital output</td>
<td>27 170</td>
<td>AA bit</td>
<td>ESC ASCII(170) ASCII(bit)</td>
<td>0</td>
</tr>
<tr>
<td>Timed disable of single digital output</td>
<td>27 171</td>
<td>AB bit</td>
<td>ESC ASCII(171) ASCII(bit)</td>
<td>0</td>
</tr>
<tr>
<td>Acquire single digital input</td>
<td>27 172</td>
<td>AC bit</td>
<td>ESC ASCII(172) ASCII(bit)</td>
<td>1</td>
</tr>
<tr>
<td>Acquire single digital input with debouncing</td>
<td>27 173</td>
<td>AD bit</td>
<td>ESC ASCII(173) ASCII(bit)</td>
<td>1</td>
</tr>
<tr>
<td>Timed square wave starting with &quot;1&quot; on single digital output</td>
<td>27 174</td>
<td>AE bit</td>
<td>ESC ASCII(174) ASCII(bit)</td>
<td>0</td>
</tr>
<tr>
<td>Timed square wave starting with &quot;0&quot; on single digital output</td>
<td>27 175</td>
<td>AF bit</td>
<td>ESC ASCII(175) ASCII(bit)</td>
<td>0</td>
</tr>
<tr>
<td>Square wave on single digital output</td>
<td>27 176</td>
<td>B0 bit</td>
<td>ESC ASCII(176) ASCII(bit)</td>
<td>0</td>
</tr>
<tr>
<td>Read counter</td>
<td>27 177</td>
<td>B1 cnt</td>
<td>ESC ASCII(177) ASCII(cnt)</td>
<td>2</td>
</tr>
<tr>
<td>Reset counter</td>
<td>27 178</td>
<td>B2 cnt</td>
<td>ESC ASCII(178) ASCII(cnt)</td>
<td>0</td>
</tr>
<tr>
<td>Set multifunction pins</td>
<td>27 179</td>
<td>B3 bit</td>
<td>ESC ASCII(179) ASCII(bit)</td>
<td>0</td>
</tr>
<tr>
<td>Read multifunction pins setting</td>
<td>27 180</td>
<td>B4 bit</td>
<td>ESC ASCII(180) ASCII(bit)</td>
<td>1</td>
</tr>
</tbody>
</table>

**Figure A3: Command codes summary table (3 of 4)**
<table>
<thead>
<tr>
<th>Command</th>
<th>Code</th>
<th>HEX code</th>
<th>Mnemonic</th>
<th>Ret.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset 1-WIRE line</td>
<td>27 181</td>
<td>1B B5 wire</td>
<td>ESC ASCII(181) ASCII(wire)</td>
<td>1</td>
</tr>
<tr>
<td>Write bit on 1-WIRE line</td>
<td>27 182</td>
<td>1B B6 wire bit</td>
<td>ESC ASCII(182) ASCII(wire) ASCII(bit)</td>
<td>0</td>
</tr>
<tr>
<td>Read bit from 1-WIRE line</td>
<td>27 183</td>
<td>1B B7 wire</td>
<td>ESC ASCII(183) ASCII(wire)</td>
<td>1</td>
</tr>
<tr>
<td>Write n bytes on 1-WIRE line</td>
<td>27 184</td>
<td>1B B8 wire n</td>
<td>ESC ASCII(184) ASCII(wire) ASCII(n) ASCII(dat0÷datn)</td>
<td>0</td>
</tr>
<tr>
<td>Read n bytes from 1-WIRE line</td>
<td>27 185</td>
<td>1B B9 wire n</td>
<td>ESC ASCII(185) ASCII(wire) ASCII(n)</td>
<td>n</td>
</tr>
<tr>
<td>&quot;Read ROM&quot; on 1-WIRE line</td>
<td>27 186</td>
<td>1B BA wire</td>
<td>ESC ASCII(186) ASCII(wire)</td>
<td>8</td>
</tr>
<tr>
<td>&quot;Match ROM&quot; on 1-WIRE line</td>
<td>27 187</td>
<td>1B BB wire rom0÷rom7</td>
<td>ESC ASCII(187) ASCII(wire) ASCII(rom0÷rom7)</td>
<td>1</td>
</tr>
<tr>
<td>&quot;Skip ROM&quot; on 1-WIRE line</td>
<td>27 188</td>
<td>1B BC wire</td>
<td>ESC ASCII(188) ASCII(wire)</td>
<td>1</td>
</tr>
<tr>
<td>&quot;Alarm search&quot; on 1-WIRE line</td>
<td>27 189</td>
<td>1B BD wire</td>
<td>ESC ASCII(189) ASCII(wire)</td>
<td>1</td>
</tr>
<tr>
<td>Graphic cursor absolute position</td>
<td>27 206</td>
<td>1B CE y x</td>
<td>ESC ASCII(206) ASCII(y) ASCII(x)</td>
<td>0</td>
</tr>
<tr>
<td>Alphanumeric mode setting</td>
<td>27 208</td>
<td>1B D0</td>
<td>ESC ASCII(208)</td>
<td>0</td>
</tr>
<tr>
<td>Graphic mode setting</td>
<td>27 209</td>
<td>1B D1</td>
<td>ESC ASCII(209)</td>
<td>0</td>
</tr>
<tr>
<td>Graphic area setting</td>
<td>27 241</td>
<td>1B F1 x1 y1 x2 y2 cmd</td>
<td>ESC ASCII(241) ASCII(x1) ASCII(y1) ASCII(x2) ASCII(y2) ASCII(cmd)</td>
<td>0</td>
</tr>
<tr>
<td>Graphic font setting</td>
<td>27 242</td>
<td>1B F2 font</td>
<td>ESC ASCII(242) ASCII(font)</td>
<td>0</td>
</tr>
</tbody>
</table>

**FIGURE A4: COMMAND CODES SUMMARY TABLE (4 OF 4)**
APPENDIX B: DISPLAY CHARACTERS

The following tables shows the characters sets displayed on QTP 12/R84 for all the possible received characters, according with ordered display and model and according with functionality mode preselected through proper commands.
Even the not ASCII characters (or special characters) change when the display type changes and if the user requires a character set different from those described in the following figures, he can directly contact grifo®.

![Character Table](image)

**Figure B1: QTP 12/R84-F2, GF2 Characters Table, in Alphanumeric Mode**
| Figure B2: QTP 12/R84-C2 Characters Table |
**Figure B3: QTP 12/R84-GF2 Minifont Characters Table, in Graphic Mode**
**Figure B4: QTP 12/R84-GF2 Katakana Font Characters Table, in Graphic Mode**
**Figure B5: QTP 12/R84-GF2 European Font Characters Table, in Graphic Mode**
APPENDIX C: MOUNTING NOTES

TERMINAL DIMENSIONS

Here are dimensions of terminal QTP 12/R84 external metallic container and frontal plastic frame. Dimensions are in mm and scale is 1:1.

The dimensions of previous figure refer to container only, but occupied area can be slightly greater by considering also mounting clamps and screws, up to a maximum of 156 x 72 x 80 mm (W x H x D).
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/R84. Installation operations are extremely easy and they are below summarized:

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

![Figure C2: Breaking for installation](image)

2) screw the two clamps, keeping the sharpened part close to the screw-cut edge of clamp;

3) insert QTP 12/R84 in the breaking made at point 1;

4) dock the two clamps prepared at point 2 to the specific side breakings of QTP 12/R84 container, taking care that the first hook of the clamp, the one near screw-cut edge, enters correctly in the proper lateral hole of the container;

5) screw the clamps until the QTP 12/R84 container is firmly docked to mounting panel;

6) connect the connectors.
PERSONALIZATION LABEL INSERTION

Frontal of QTP 12/R84 is provided with a pocket where the user can insert a personalization label with his logo, an identification code, purpose of status LED, or anything else. **If the label is required please insert it before mounting QTP.** Label must be thin but rather rigid, for example made of 160 g/m² paper or polyester or polycarbonate sheets. Here follow the suggested dimensions, in millimeters, of personalization label; please note that the white zone is the area contained in the transparent window, or in other words, the visible part:

![Dimensions of Personalization Label](image)

**FIGURE C3: DIMENSIONS OF PERSONALIZATION LABEL**

Here follow the operations required to insert personalization label inside the QTP 12/R84.

1) Unscrew the two black screws of frontal panel (if present).
2) Separate the group metallic carter + plastic frame from the group keyboard + printed circuit. A simple pressure on QTP 12/R84 connectors is sufficient.
3) Now keyboard is ready to insert the personalization label; this latter must be inserted from the top side, using the specific pocket located on the back of keyboard panel, as shown in following figure. As described on figure C3, length of label must be greater than height of window to simplify the insertion and extraction.
4) Remount terminal QTP 12/R84, following the previous steps in reversed order.

![Personalization Label Insertion](image)

**FIGURE C4: PERSONALIZATION LABEL INSERTION**
FIXING FRONTAL PANEL TO CONTAINER

QTP 12/R84 by default is provided with front panel (keyboard+printed circuit board) tongued in plastic frame+metallic carter, of back container. Terminal anyway allows a better mechanical docking between these two groups by using two specific screws, that 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 keyboard + printed circuit. A simple pressure on the backside connectors, or on the printed circuit always from the backside connectors window, is normally sufficient.

2) Unscrew the 2 central screws to separate the frame from the rear carter.

3) On the front panel, in correspondence to those 2 central screws, there are 2 countersink holes which are visible only in the rear view. To get these two holes accessible, the user needs to hole the polycarbonate panel covering.

4) Remount the panel by using the two screws of point "2". They will be screwed on the front panel instead of the frontal frame.
APPENDIX D: DATA SHEETS

grifo® provides a completely free technical documentation service to make available data sheets of on board components, through its web site. In this chapter the user found the complete and ready to use links and URLs to these information, together with the first pages of the same documents. To use our technical documentation service just connect to our site www.grifo.com and click its icon.

T89C51CC01 AND T89C51AC2
Link: Home | Technical documentation Service | ATMEL WµC | Data-Sheet T89C51CC01
URL: http://www.grifo.com/PRESS/DOK/Temic/AT89C51CC01.pdf

Features
- 80C51 Core Architecture
- 256 Bytes of On-chip RAM
- 1K Bytes of On-chip ERAM
- 32K Bytes of On-chip Flash Memory
  - Data Retention: 10 Years at 85°C
  - Read/Write Cycle: 10K
- 2K Bytes of On-chip Flash for Bootloader
- 2K Bytes of On-chip EEPROM
  - Read/Write Cycle: 100K
- 14-sources 4-level Interrupts
- Three 16-bit Timers/Counters
- Full Duplex UART Compatible 80C51
- Maximum Crystal Frequency 40 MHz
  - In X2 Mode, 20 MHz (CPU Core, 40 MHz)
- Five Ports: 32 + 2 Digital I/O Lines
- Five-channel 16-bit PCA with:
  - PWM (8-bit)
  - High-speed Output
  - Timer and Edge Capture
- Double Data Pointer
- 21-bit WatchDog Timer (7 Programmable Bits)
- A 10-bit Resolution Analog to Digital Converter (ADC) with 8 Multiplexed Inputs
- Full CAN Controller:
  - Fully Compliant with CAN Rev2.0A and 2.0B
  - Optimized Structure for Communication Management (Via SFR)
  - 15 Independent Message Objects:
    - Each Message Object Programmable on Transmission or Reception
    - Individual Tag and Mask Filters up to 29-bit Identifier/Channel
    - 8-byte Cyclic Data Register (FIFO)/Message Object
    - 16-bit Status and Control Register/Message Object
    - 16-bit Time-Stamping Register/Message Object
    - CAN Specific 2.0 Part A or 2.0 Part B Programmable for Each Message Object
    - Access to Message Object Control and Data Registers Via SFR
    - Programmable Reception Buffer Length Up To 15 Message Objects
    - Priority Management of Reception of Hits on Several Message Objects at the Same Time (Basic CAN Feature)
    - Priority Management for Transmission
    - Message Object Overrun Interrupt
  - Supports:
    - Time Triggered Communication
    - Autobaud and Listening Mode
    - Programmable Automatic Reply Mode
    - 1-Mbit’s Maximum Transfer Rate at 8 MHz\(^1\) Crystal Frequency in X2 Mode
    - Readable Error Counters
    - Programmable Link to On-chip Timer for Time Stamping and Network Synchronization
    - Independent Baud Rate Prescaler
    - Data, Remote, Error and Overload Frame Handling
- On-chip Emulation Logic (Enhanced Hook System)
- Power Saving Modes:
  - Idle Mode
  - Power-down Mode

---

1. At BRP = 1 sampling point will be fixed.
- Power Supply: 5V ± 10% (or 3V (1) ± 10%)
- Temperature Range: Industrial (-40° to +85°C)
- Packages: VQFP44, PLCC44, CA-BGA64

**Description**

The T89C51CC01 is the first member of the CANary™ family of 8-bit microcontrollers dedicated to CAN network applications.

In X2 mode a maximum external clock rate of 20 MHz reaches a 300 ns cycle time.

Besides the full CAN controller T89C51CC01 provides 32K Bytes of Flash memory including In-System-Programming (ISP), 2K Bytes Boot Flash Memory, 2K Bytes EEPROM and 1.2-Kbyte RAM.

Primary attention is paid to the reduction of the electro-magnetic emission of T89C51CC01.

**Block Diagram**

![Block Diagram of T89C51CC01](image)

Notes:
1. 8 analog Inputs/8 Digital I/O
2. 2-Bit I/O Port

1. Ask for availability

2. T89C51CC01
APPENDIX E: ALPHABETICAL INDEX

Simboli
+V opto  12
+V opto  26, 35
.LIB  38, 48
.SER  42
/INT0  35, 53
/INT1  35, 53
µC/51  55
1-WIRE  8, 28, 35, 93
51 family  8, 54
6 - TTL I/O, 1-WIRE, I2C BUS connector  28
9 bits  43
9th bit  44

A

ABSOLUTE PLACEMENT OF ALPHANUMERIC CURSOR, command  62
AC  14
Access control systems  85
Accessories  28, 46, 56
ACQUIRE ALL DIGITAL INPUTS, command  77
ACQUIRE SINGLE DIGITAL INPUT, command  80
ACQUIRE SINGLE DIGITAL INPUT WITH DEBOUNCING, command  80
ADDs View Point  61
"ALARM SERCH" ON 1-WIRE LINE, command  98
Allarm  6
Alphanumeric  6
Alphanumeric mode  86, B-1
ALPHANUMERIC MODE SETTING, command  86
Alternative functions  84
Application program  49, 54, 60
ASCII  39, B-1
Assistance  1
Autorepeat  6, 11, 40

B

Backligt  6
BACKSPACE, command  63
BASCOM 8051  55
BasicCAN  8
BEEP, command  68
Bibliography  102
Bit access RAM  51
Bits per character  43
BLINKING BLOCK CURSOR ON, command  76
Blocks diagram  3
Boot loader 54
Brightness 66
Buad rate 42
Buffer 10, 40
Buzzer 7, 68

C
CAN 8, 12, 22, 23, 34, 35
Card code 66
CARRIAGE RETURN, command 62
CARRIAGE RETURN+LINE FEED, command 62
CCITT 35
Characters 11, 39, B-1
Clamps C-1, C-2
CLEAR END OF LINE, command 63
CLEAR END OF PAGE, command 63
CLEAR LINE, command 63
CLEAR PAGE, command 63
Cleats area 89
Clock 10
CN1 - Power supply connector 14
CN2 - serial line connector 16
CN3 - CAN interface connector 22
CN4 - Relays digital outputs connector 24
CN5 - Optocoupled digital inputs connector 26
Code 7, 50
Code area 49
Column 62
Command mode 39, 67
Commands 39, 61, A-1
Commands for 1-WIRE communication 93
Commands for alternative functions of digital I/Os 84
Commands for buffered digital I/Os management 77
Commands for characters erasure 63
Commands for cursor attributes management 76
Commands for cursor position 61
Commands for EEPROM 64
Commands for general functions 66
Commands for graphics 86
Commands for keyboard management 72
Commands for message management 69
Commands for user characters 74
Common 24
Communication 67
block diagram 17
electric protocol 30, 32
line 34
logic protocol 10, 42
master slave  42, 43
normal  42, 43
physic 42, 43
COMmunication RESET, command  67
Compilers  55
Components map  31
Components side  31
COMrx  46, 56
Connections  14, 101
Connectors  11, 14, 29
   CN1  14
   CN2  16
   CN3  22
   CN4  24
   CN5  26
   J6  28
Console input  50
Console output  50
Console redirection  50
Console status  50
Consumption  36
Contacts  8, 24
Contacts logic  55
Container  1, 113
Contrast  30
Controller  48
Counter  53, 84, 85
Current  12
Current loop  4, 16, 20, 32, 35
Cursor  61, 76, 88
CURSOR DOWN, command  61
CURSOR LEFT, command  61
CURSOR OFF, command  76
CURSOR RIGHT, command  61
CURSOR UP, command  62

D
Data  7, 38
Data endurance  38
Data sheets  D-1
DC  14, 36
Debouncing  80
DEBUG mode  30, 54
Default configuration  30, 32, 36, 38, 41
DEFINITION AND MEMORIZATION OF USER CHARACTER, command  75
DEFINITION OF USER CHARACTER, command  75
Delay  71, 75
Delays 40
Demo programs 46, 47, 56, 60
Development 55
Development P.C. 56
Digital inputs 8, 26, 27, 35, 77, 80
Digital outputs 8, 24, 35, 77
Dimensions C-1
DIR 32, 34
Direct acces RAM 51
Directive 1, 16, 35
DISABLE SINGLE DIGITAL OUTPUT, command 79
Display 6, 10, 52, B-1
Distance 34
DLL libraries 44, 47
Draws area 89
Drivers 17, 33
DS18s20 93
Dump terminal 42
Duty cycle 81

E

EEPROM 7, 10, 11, 38, 42, 52, 64
EEPROM for general use 65
EEPROM initialization 52
Efficiency 12
ENABLE SINGLE DIGITAL OUTPUT, command 78
Encoders 85
Entry points 49
ESD 1
European font 92, 111
Executable code 59
Expansion 99
EXPS-1 36
EXPS-2 36, 100
External cards 99
Extra voltages 36

F

Features
electric 12
general 2, 10
physical 11
technical 10
Fills area 89
Filters 27, 36
Firmware 2, 11, 32, 34, 38, 66, A-1
Firmware initialization 42, 52
First purchase 47, 60
FLASH EPROM  7, 10, 49, 52, 54
FLASH programming  54, 57
FLIP  54, 57
FLUORESCENT DISPLAY BRIGHTNESS SETTING, command  66
Fonts  92, B-3
Frequency  10, 83
Front panel  C-2
Frontal  73
Frontal panel fixing  C-4

G
Galvanically isolated  8
GENERAL RESET, command  67
Graphic  6, 86
Graphic area  88
GRAPHIC AREA SETTING, command  88
GRAPHIC CURSOR ABSOLUTE POSITION, command  88
Graphic example  87
GRAPHIC FONT SETTING, command  92
Graphic mode  88, B-3, B-4, B-5
GRAPHIC MODE SETTING, command  88
Ground  34, 35
Group A  24
Group B  24

H
Hardware  2
HOME, command  62
Hooters  83
Horizontal data and horizontal shift  90
Horizontal data and vertical shift  90
How to start  46, 56
Humidity  11
HYPERTERMINAL  47, 56

I
I2C BUS  8, 28, 35
Identification name  42, 43
Impedance  12, 34
In System Programming  49, 54
Indirect access RAM  51
Industrial environments  48
Information  2
Initialization  67
Installation  14
Integration of library firmware  48
Interfacement  35
Interrupt 53
Interrupts 35, 50
Interspacing 92
Introduction 1
Inverts area 89
IP54 10
ISP 49, 54

J
Jumpers 29, 30, 54

K
Katakana font 92, B-4
KEY CODE RECONFIGURATION, command 72
Keyboard 6, 40, 72
Keyclick 6, 40, 42, 72
KEYCLICK OFF WITH MEMORIZATION, command 73
KEYCLICK OFF WITHOUT MEMORIZATION, command 72
KEYCLICK ON WITH MEMORIZATION, command 73
KEYCLICK ON WITHOUT MEMORIZATION, command 72
Keys 6, 40
Keys codes 38, 40, 44, 72
Keys location 41
Keys numeration 41

L
Label 6, C-3
LADDER WORK 55
LCD 6, 10
LED 6, 68
LED AND BUZZER ACTIVATION, command 68
Library firmware 38, 48
License 47, 60
Line direction 34
Loads 24, 36
Local setup 42
Location 29
Logic protocol 10, 43

M
Malfunction 7, 47, 60
Map 31
Master 43
Master slave 42, 43
Master slave example 45
Master unit 42
"MATCH ROM" ON 1-WIRE LINE, command 97
Membrane 6
Memory 7, 10, 49, 51, 52
MESSAGE READNG, command 70
MESSAGE STORAGE, command 69
Messages 69
Messages 11, 38
Microcontroller 8, 10, 35, 57, 85, 102, D-1
Mini font 92, B-3
Models 4, 6
Mounting 11, C-1, C-2
Mounting breaking C-2

N
Name 42
Network 19, 21, 23
Noisy 34, 36
Normally open 24
NPN 26, 35

O
Operating mode 39, 67
OPERATING MODE SELECTION, command 67
Operating system 46, 56
Options 4, 7, 8, 32, 34, 36
Optocoupled 8, 12, 26, 27, 35, 77
Overflow 40, 71, 75

P
P.C. connection 46, 56
Parity 43, 44
Patterns 38, 74
PeliCAN 8
Personalization C-3
Photo complete 9
Photo of available models 5
Photo of characters 39
Physic protocol 10
Pixels 86, 88, 89
PNP 26, 35
Pocket 115
Polarity 36
Power 12
Power consumption 12
Power on 7, 10, 34, 67, 86
Power supply 7, 12, 14, 28, 36
Presence byte 38, 64
PRFWQ12R.HEX  57
Priority  35
PRQTP12R.EXE  46

Q
QTP EDIT  69
QTP12Rxx.HEX  49, 57

R
RAM  7, 10, 51, 52
READ BIT FROM 1-WIRE LINE, command  94
READ BYTE FROM EEPROM, command  65
READ CARD CODE, command  66
READ COUNTER, command  85
READ Firmware VERSION, command  66
READ MULTIFUNCTIONS PINS SETTING, command  84
READ N BYTES FROM 1-WIRE LINE, command  95
READ PRESENCE BYTE, command  64
"READ ROM" ON 1-WIRE LINE, command  96
READING OF MAX MESSAGE NUMBER, command  69
Receive buffer  10, 40
Relays  8, 12, 24, 35, 77, 81
Remarks  47
Representation mode  39, 67
REQUEST FOR EEPROM AVAILABILITY, command  64
Requirements  48, 52, 93
Reset  10, 67, 94
RESET 1-WIRE LINE, command  94
RESET COUNTER, command  85
Resources  10, 48, 52
ROM code  96, 97
Row  62
RS 232  4, 16, 18, 32, 35, 46, 56
RS 422  4, 12, 16, 18, 32, 35
RS 485  4, 12, 16, 19, 34, 35
Rules  1
RUN mode  30, 54

S
Safety  1
Screws  C-4
Scrolling  71
SCROLLING MESSAGES VISUALIZATION, command  71
Semaphores  83
Serial firmware  38, 42
Serial line  4, 16, 32, 46, 56
SET MULTIFUNCTIONS PINS, command  84
ITALIAN TECHNOLOGY

Setup  42
Signals connections  35
Size  11, C-1
"SKIP ROM" ON 1-WIRE LINE, command  96
Slave  43
Software  38
Software development tools  48, 54
Solder side  31
Sound  7, 68
Special characters  39, B-1
Special function registers  51
SPI  8, 28, 35
Square wave  81, 82, 83
SQUARE WAVE ON DIGITAL OUTPUT, command  83
Stack  7, 53
STEADY STATIC CURSOR ON, command  76
Stop bit  42
Switching  7, 36

T
T89C51AC2  10, D-1
T89C51CC01  10, D-1
Temperature  11
Temperature sensors  93
Terminal emulation  46
Termination  12, 19, 23, 30, 34
Time out  44
TIMED DISABLE OF SINGLE DIGITAL OUTPUT, command  79
TIMED ENABLE OF SINGLE DIGITAL OUTPUT, command  78
TIMED SQUARE WAVE STARTING WITH "0" ON OUTPUT, comm  82
TIMED SQUARE WAVE STARTING WITH "1" ON OUTPUT, com  81
Timeout error  57
TIMER 0  50
TIMER0  53
TIMER1  53
TIMER2  53
Timing  10, 11, 40, 44, 78, 81
Transmission buffer  10
Transmission time  44
Transmitter  32
TransZorb™  36
Trimmer  29, 30
TTL  8, 28, 35
TTL I/O lines  8, 28, 93

U
Use of library firmware  48
User applications  38, 54
User bytes 38, 65
User characters 38, 39, 74, B-1

V

VCL 21
Vectors 53
Version 2, 66
Vertical data and horizontal shift 91
Vertical data and vertical shift 91
VFD 6, 10
Video terminal 42
Visibility 6, 30
Visual indication 6
VISUALIZATION OF N MESSAGES, command 70
Voltages 7, 12, 14, 36

W

Warranty 1
Weight 11
WRITE ALL DIGITAL OUTPUTS, command 77
WRITE BIT ON 1-WIRE LINE, command 94
WRITE BYTE ON EEPROM, command 65
WRITE N BYTES ON 1-WIRE LINE, command 95
WRITE OF PRESENCE BYTE, command 64