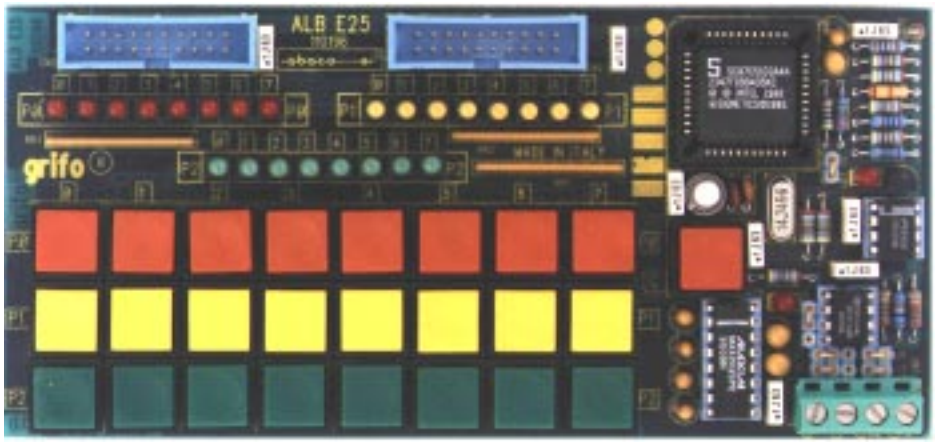
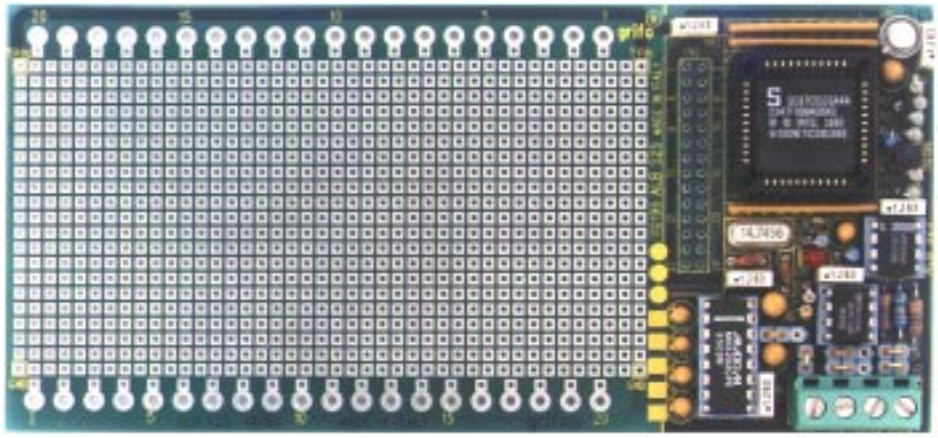
µP25

ALB E25 - ALB S25

microProcessor 25 I/O lines

TECHNICAL MANUAL





grifo®
ITALIAN TECHNOLOGY

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

µP25, ALB E25, ALB S25 Edition 5.10 *Rel. 21 June 2001*

abaco®, **GPC®**, **grifo®**, *are trademarks of* **grifo®**

µP25

ALB E25 - ALB S25

microProcessor 25 I/O lines

TECHNICAL MANUAL

µP25

Application based on microcontroller 87c51 matched with a specific management firmware developed for it. Code sequences sent through the serial line allow to: drive three 8 bit ports for I/O operations, eventually generating timed interrupts; acquire the 16 bit counter; read or write EEPROM messages; manage an A/D converter, a D/A converter and a LED display driver through high level commands. Two communication protocols can be selected: a point-to-point protocol or a Master-Slave network protocol.
Local Setup mode to configure completely the several sections.

ALB E25

Evaluation board based on **µP25**, provided with a set of LEDs, keys and connectors, that allow the user to control all the several resources **µP25** is provided with. Serial line configurable as RS 232 or RS 485.

ALB S25

Experimental board based on **µP25**, provided with a prototipal area where the user can develop any specific hardware to interface and the specialization of **µP25** I/O lines.
Serial line configurable as RS 232 or RS 485.

grifo®
ITALIAN TECHNOLOGY

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

abaco®, **GPC®**, **grifo®**, *are trademarks of* **grifo®**

DOCUMENTATION COPYRIGHT BY grifo® , ALL RIGHTS RESERVED

No part of this document may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language, in any form or by any means, either electronic, mechanical, magnetic, optical, chemical, manual, or otherwise, without the prior written consent of **grifo**®.

IMPORTANT

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

SYMBOLS DESCRIPTION

In the manual could appear the following symbols:

Attention: Generic danger

Attention: High voltage

Trade Marks

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

GENERAL INDEX

FIGURES INDEX

INTRODUCTION

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

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 rispect the actual safety normative. The protection of this container is not threshold to the only atmospheric agents, but specially to mechanic, electric, magnetic, etc. ones.

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

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

CARD VERSION

The present handbook is reported to the **ALB E25** and **ALB S25** cards whose release is:

ALB E25 -> 110196
ALB S25 -> 110196

and later. The validity of the bring informations is subordinate to the number of the card release. The user must always verify the correct correspondence among the two denotations. On the card the release number is present in more points both board printed diagram (serigraph) and printed circuit (for example between connectors CN2 and CN3 on **ALB E25** and between CN2 and the prototipal area on **ALB S25** both on the component side and on the solder side).

µP25 GENERAL INFORMATION

µP25 (microProcessor 25 I/O lines) is a powerful system designed specifically for the applications that require Input or Output lines management.
This system is based on the very famous microcontrollore **87c51**, with a management firmware developed for it; through comfortable code sequences sent through the serial line it allows tomanage several sections; In detail: **three 8 bit ports** are available for I/O operations like output lines control, eventually generating timed interrupts (square wave, setting or resetting the pin for a determined time, etc.), input lines acquisition eventually with debouncing, etc.
There is one more line that can be configured to work ad I/O line or as a trigger singal for the **16 bit counter** available to the user.
To work correctly **µP25** must be connected to a serial **EEPROM** that can be up to **2KBytes**; it contains the system configurations and is used to store up to **203 strings** of 10 characters each, available to the user.
µP25 is also provided with a set of high level commands to manage an **A/D** converter, a **D/A** converter and a **LED display driver**, opportunely interfaced through some I/O lines.
Two communication protocols are available with a baud rate up to 38.4 KBaud; the first is used to build a simple point-to-point communication buffered in **RS 232**, **RS 422** or **Current Loop**; the second is essential to develop a Master-Slave structure buffered in RS 485 or Current Loop. In this case, connecting several **µP25** devices to the network, it is possible to manage a very high number of I/O signals also located at remarkable distances through an unique Master board and an unique serial line.
For example it is possible to make automatic doors controls for buildings, signals repeaters, etc.

- Firmware implemented on microcontroller **87c51**.
- Local SETUP mode for a complete system configuration.
- Communication protocol for point-to-point or **Master-Slave** systems.
- Up to **128 µP25** devices can be connected in Master-Slave mode.
- Baud Rate selectable from **1200 Baud** up to **38.4 KBaud**.
- Up to **25 I/O lines** manageable for I/O operations.
- Possibility to generate signals like square wave, timed setting or resetting of a pin, etc. with high level commands.
- Supports several kinds of serial **EEPROMs** up to **2KBytes**, to store Setup and messagges.
- Up to **203 strings** each 10 characters long can be stored in EEPROM.
- **16 bit Counter** user available managed through high level commands.
- **12 bit A/D converter TLC 2543** interfaceable and manageable with high level commands.
- **16 bit D/A converter AD 420** interfaceable and manageable with high level commands.
- **LED display driver M5480** interfaceable and manageable with high level commands.

µP25 HARDWARE DESCRIPTION

Here follow an hardware description of the several section that make a minimal **µP25** system (please refer to the electric diagram in figure 1), with indication of the function performed by each section.

PORT 0 INPUT/OUTPUT LINES

It is made of the 8 lines called **P0.?** (pin 36÷43 of **µP25**), each of which can be configured as input or output. Commands available for this section allow both byte-level operations (read or write the whole port) and bit-level operations (read or write to a single line).
In addition, a set of commands not available for **ports 1 and 2** allow to perform **timed status changes** on the lines configured as output; in fact it is possible to generate a square wave, to activate a certain output for a given time, to generate a specified number of impulses, etc.

PORT 1 INPUT/OUTPUT LINES

It is made of the 8 lines called **P1.?** (pin 2÷9 of **µP25**), each of which can be configured as input or output. Commands available for this section allow both byte-level operations (read or write the whole port) and bit-level operations (read or write to a single line).

PORT 2 INPUT/OUTPUT LINES

It is made of the 8 lines called **P2.?** (pin 24÷31 of **µP25**), each of which can be configured as input, output or as management signals for **A/D converter**, **D/A converter** or **LED display driver** (as described further in this manual).
In case of configuration as I/O, the commands available for this section allow both byte-level operations (read or write the whole port) and bit-level operations (read or write to a single line).

INT0 LINE

It is connected to the signal **/INT0** (pin 14 of **µP25**), which can be configured as input, output or as trigger for the **16 bit counter** available to the user.
In this particualr case the counter will be increased by one each time a **falling edge** is detected on the above mentioned pin.
If the **INT0** line is configured as I/O, the instructions that allow to perform timed status changes descrived in one of the above paragraph will be available, in addition to the commands to set and acquire the pin status.

RESET SIGNAL

It is connected to **RST** signal (pin 10 of **µP25**), which must be connected to a specific circuitery that charges to generate a **reset** signal during the system power-on. Such signal, during this phase, must reach logic level **1**, keep the status for at least **5 µsec**, the return to logic level **0**.

RUN/SETUP AND RESISTOR NETWORKS SUPPLY

This circuitery drives the **/SETUP** signal (pin 17 of **µP25**) which has a double function: allows to select the working modality (**RUN** or **SETUP**) through the jumper, and allows to manage the resistor networks supply (signal SIP) of port **0**, **1** and **2** of **µP25** through the **PNP** transistor.
In detail:

JUMPER CONNECTED (SETUP MODE)
In this modality **µP25** sets all the ports as inputs and supplies the resistor networks as the transistor is led to carry the current by the jumper itself.

JUMPER NOT CONNECTED (RUN MODE)
In this modality **µP25** as first reads the setting data contained in the EEPROM, if these are valid configures the ports as inputs or outputs and only when comes to this point supplies the resistor networks (LED on). By acting this way, during the initialization phase immediatly after the power on it is warranted the logic status **0** to the pins configured as outputs, avoiding the possibility of unwanted activations of **NPN** devices connected to them.

EEPROM

It is the circuitery made of the serial **EEPROM** and its specific interfacing components. This section is driven through **PWM**, **SCL** and **SDA** signals (pin **15**, **18** and **19**).
In detail:

PWR	**(pin 15)**	->	EEPROM power supply circuit management signal
SCL	**(pin 18)**	->	Serial line **Clock** signal
SDA	**(pin 19)**	->	Serial line Data signal

The EEPROM power supply management circuitery warrants a better retention of the data stored in EEPROM, because this latter is supplied only during the information read and write phases. Such circuitery anyway is optional and can be omitted by leaving pin **PWR** of **µP25** disconnected.

POWER SUPPLY OF µP25

µP25 needs an unique tension for supply:

+5 Vdc: Supplies the CPU core and peripherals; must be in the range +5 Vdc ± 5% and must be provided through the specific pins.

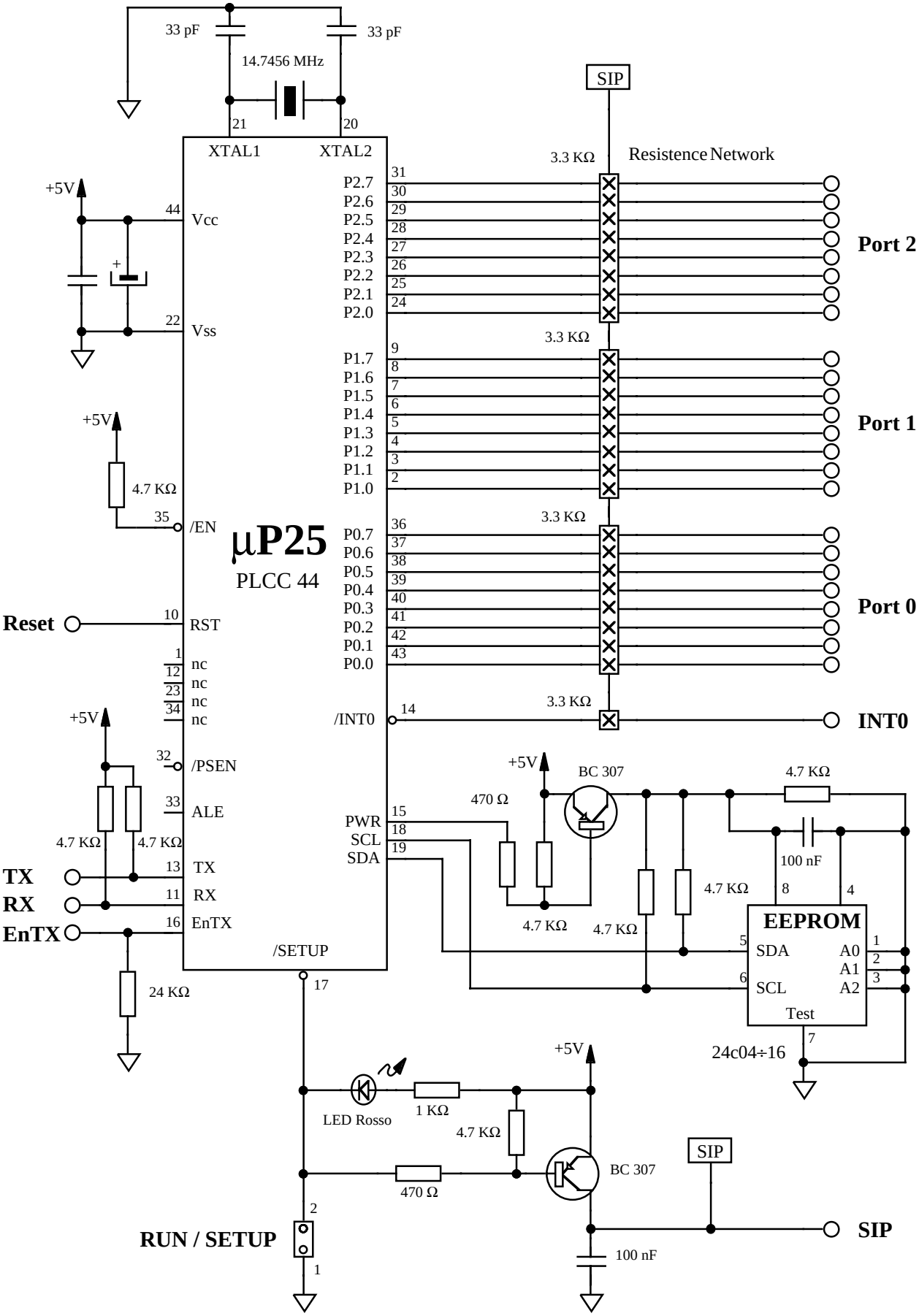


FIGURE 1: ELECTRIC DIAGRAM OF MINIMUM µP25-BASED SYSTEM

INTERFACEMENT OF A/D, D/A AND LED DISPLAY DRIVER

The lines of **µP25** port **2** allow to connect an **A/D converter**, a **D/A converter** and a **LED display driver**. These devices are managed through a set of high level commands and allow to make the application based on µP25 even more powerful and versatile, enlarging the range of projects and fields where it can be used.
Connection modalities of such peripheral devices to port **2** are shown in figure 2.

INTERFACEMENT OF A/D CONVERTER

The **Texas Instruments TLC 2543** A/D converter can be connected to **µP25**; this device has the following features:

- Conversion resolution **12** bit.
- **11** analog input channels.
- Conversion time **10 µsec**.
- **E**nd **O**f **C**onversion output pin.

To interface **TLC 2543** to **µP25** the user should perform the following connections:

Port 2.0 (pin **24** of **µP25**) -> **Clk** (pin **18** of **TLC 2543**) : Serial Clock signal
Port 2.1 (pin **25** of **µP25**) -> **Din** (pin **17** of **TLC 2543**) : Serial Data In signal
Port 2.2 (pin **26** of **µP25**) <- **Dout** (pin **16** of **TLC 2543**) : Serial Data Out signal
Port 2.3 (pin **27** of **µP25**) -> **CS** (pin **15** of **TLC 2543**) : Serial Chip Select signal

INTERFACEMENT OF D/A CONVERTER

The **Analog Devices AD 420** A/D converter can be connected to **µP25**; this device has the following features:

- Conversion resolution **16** bit.
- Output signal can be Tension or Current in the ranges **0÷10 Vdc**, **4÷20 mA**, **0÷20 mA**, and **0÷24 mA**.

To interface **AD 420** to **µP25** the user should perform the following connections:

Port 2.1 (pin **25** of **µP25**) -> **Din** (pin **9** of **AD 420**) : Serial Data In signal
Port 2.6 (pin **30** of **µP25**) <- **Clk** (pin **8** of **AD 420**) : Serial Clock signal
Port 2.7 (pin **31** of **µP25**) -> **Latch** (pin **7** of **AD 420**) : Output setting signal

INTERFACEMENT TO LED DISPLAY DRIVER

The **STM M5480** or the **National Semiconductor MM5480** LED display driver can be connected to **µP25**; these devices have the following features:

- **23** outputs with current generator drivers.
- Possibility to drive directly a **3 figures and half** LED display.
- Possibility to connect LEDs directly to the outputs, without resistors.

To interface **M5480** to **µP25** the user should perform the following connections:

Port 2.4 (pin **28** of **µP25**) -> **Din** (pin **16** of **M5480**) : Serial Data In signal
Port 2.5 (pin **29** of **µP25**) -> **Clk** (pin **18** of **M5480**) : Serial Clock signal

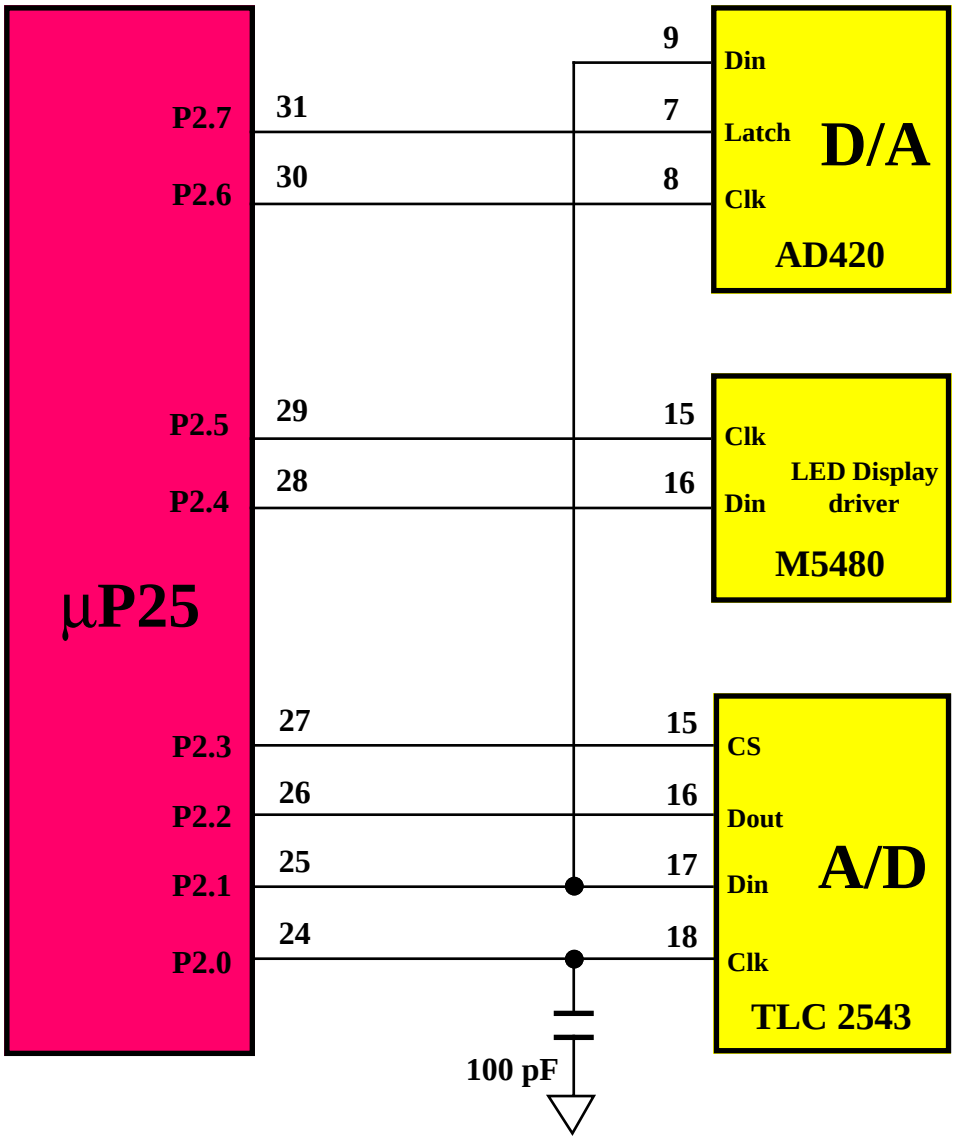


FIGURE 2: INTERFACEMENT OF A/D, D/A AND LED DISPLAY DRIVER TO µP25

SERIAL COMMUNICATION

It is driven thorugh **TX**, **RX** and **En TX** signals (pin **13**, **11** and **16** of **µP25**), which, if opportunely interfaced, allow to perform the serial connection between the master unit and the system based on **µP25**. The protocol can be **RS 232**, **RS 422** or **Current Loop** for the point-to-point communication (master unit connected to only one **µP25**), or **RS 485** or **Current Loop** for the Master-Slave communication.
µP25 pins of this section have the following meaning:

TX (pin **13** of **µP25**) -> Serial transmission signal
RX (pin **11** of **µP25**) -> Serial reception signal
En TX (pin **16** of **µP25**) <- Transmitter abilitation signal:
Logic level 0: Transmitter disabled
Logic level 1: Transmitter enabled

This latter signal is required only if the user wants to create a Master-Slave network base on **RS 485** serial protocol. In fact using this protocol the transmitting device must be actived only when **µP25** must send information to the Master unit.
In this case the interfacement circuitery based on **Texas Instruments SN75176** reported in the following figure should be realized.

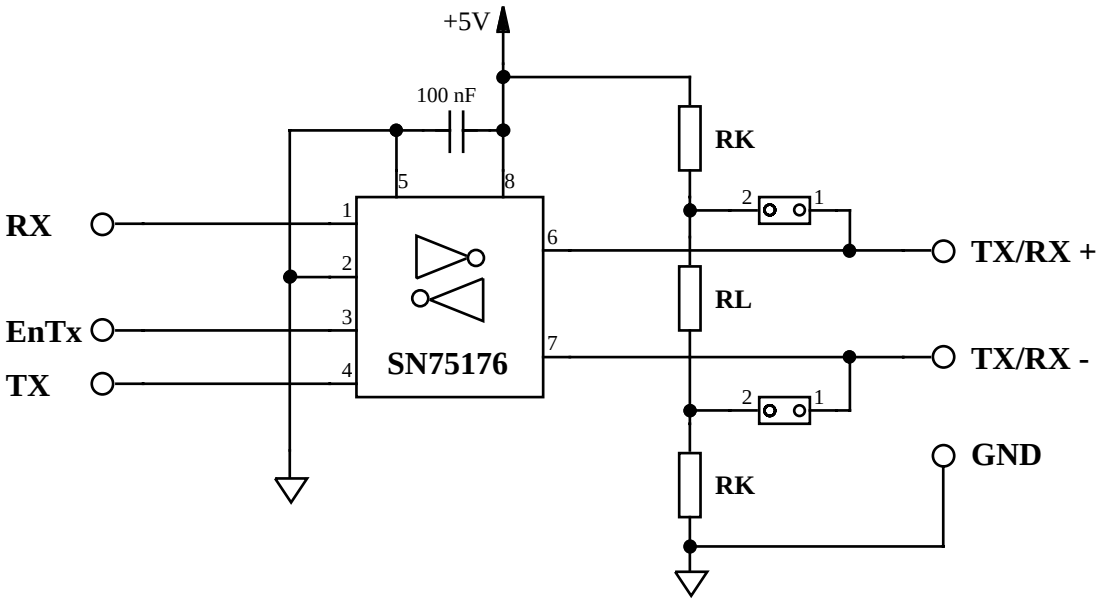


FIGURE 3: RS 485 INTERFACE CIRCUITERY BASED ON SN75176

The resistors **RL** and **RK** are the line **termination** circuitery; such circuitery can be enabled by connecting the two specific jumpers and must be enabled only in the units located at the beginning (**master** unit) and at the end of the network (**n-th slave** unit in a n units **daisy chain** network).
To determine the values of such components the user shold consider that the communication line must have a resistance of **60 Ω** and that the forcing resistors to Vcc and GND must have a value included in the range **5.6 kΩ** and **6.8 kΩ**.

So the values of **RL** and **RK** are:

Resistor RL: This component must be installed both on master unit and on n-th slave unit (the devices put at the extemities of the line), so: **RL = 120 Ω**

Resistor RK: These components may not be installed on master unit or on n-th slave unit, in this case: **RK = 2.7 kΩ÷3.3 kΩ.**
If these components are installed both on master unit and on n-th slave unit, the value must be: **RK = 5.6 kΩ÷6.8 kΩ.**

The following figure shows an example of daisy chain connection with one master unit and n slave units.

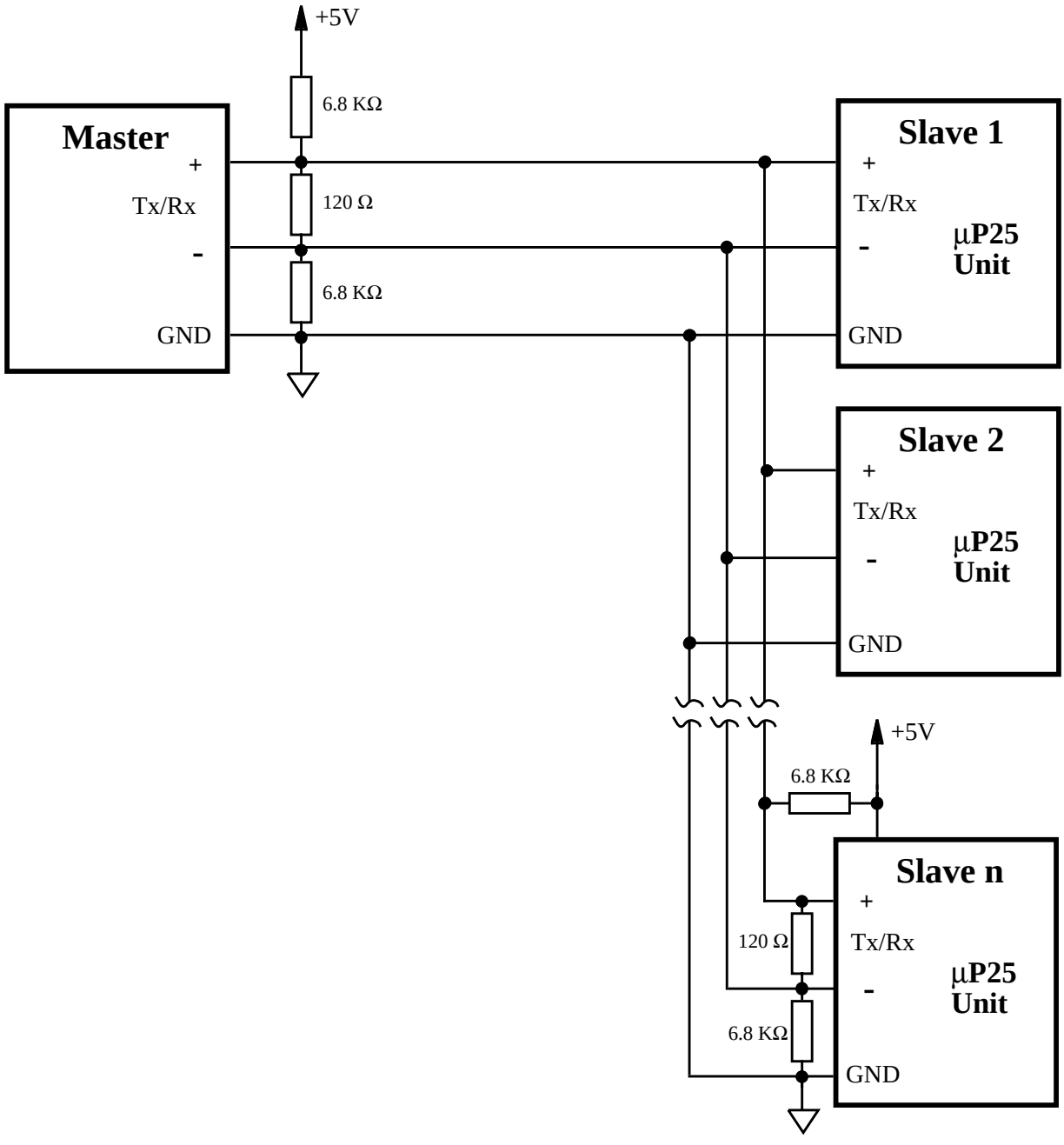


FIGURE 4: MASTER-SLAVE NETWORK COMMUNICATION EXAMPLE

TECHNICAL FEATURES OF µP25

GENERAL FEATURES OF µP25

On board resources:
- Local SETUP to completely configure the system.
- Up to 25 lines manageable for I/O operations, with possibility to generate square wave signals, timed setting or resetting of pins, etc. managed through high level commands.
- Supports several serial EEPROM sizes up to to 2 KBytes to store setup information and messages.
- Up to 203 strings of 10 characters each storable in EEPROM.
- 16 bit counter user available through high level commands.
- 12 bit A/D converter TLC 2543 through high level commands.
- 16 bit D/A converter AD 420 through high level commands.
- LED display driver M5480 through high level commands.

CPU core: 87c51

Communication protocols: Point-to-point or Master-Slave

Baud rate:	1200, 2400, 4800, 9600, 19200, or 38400
Stop bit:	1
Parity:	None
Data bits:	8 (point-to-point)
	9 (Master-Slave)

Up to 128 µP25 can be connected in Master-Slave

PHYSICAL FEATURES OF µP25

Package: PLCC 44 pins

Temperature range: from 0 to 70° C

Relative humidity: 20% up to 90% (without condensing)

ELECTRIC FEATURES OF µP25

Power supply: +5 Vdc ± 5%

Quartz frequence: 14.7456 MHz

Reset minimum duration: 5 µsec

µP25 SOFTWARE DESCRIPTION

µP25, as already said before, is an I/O device manageable through a set of commands sent across the serial line; this means that anything received is interpretated and executed and that eventual response codes are sent back across the same serial line.
µP25 firmware has a special working mode called "SETUP mode" that allows to configure all the several sections of the device.
Here follows a description of **µP25** functionalities, plus the complete set of command sequences and the combinations recgnized that must be used to take advantage of the device's main features. For each code or code sequence there is a doble description: mnemonic, in ASCII characters and numeric, in decimal and hexadecimal.

WORKING MODALITY SELECTION

µP25 has two different working modalities: "**SETUP mode**" and "**RUN mode**". The selection of the working mode to assume happens during the power on phase, testing the logic status of the **/SETUP** line (pin 15) of **µP25**; in detail:

/SETUP (pin 15)	=	Logic level **0**	->	*SETUP mode*
/SETUP (pin 15)	=	Logic level **1**	->	*RUN mode*

Here follows a complete description of all the commands available both for SETUP and for RUN mode.

SETUP MODE

In SETUP mode it is possible to set all the initialization parameters, like ports setting, kind of communication, baud rate, device name, EEPROM size, etc.
These parameters will ve stored in EEPROM and will define how the **µP25** is going to work in RUN mode.
In this working modality all ports are set as inputs and the communication protocol is fixed to the following values:

9600 Baud, 8 Bit, 1 Stop, NO Parity, Point-to-point communication

Master-Slave communication is not available in this phase.

SETUP MODE COMMANDS

Here follows a description of all the commands that **µP25** recognizes in SETUP mode.

EEPROM TYPE SETTING

Input sequence:

Dec Code:	***69***	***<EEPROM code>***
Hex Code:	***45***	***<EEPROM code>***
Mnemonic:	***E***	***ASCII(EEPROM code)***

The EEPROM type indicated by EEPROM code is installed.
Possible values for EEPROM code are:

1	*EEPROM type 24c02*
2	*EEPROM type 24c04*
4	*EEPROM type 24c16*

Example:
To select a serial EEPROM type 24c04, send the following sequence: **69 2**.

Response codes:
After sending the command to **µP25**, it returns a byte indicating whether the EEPROM code is correct or not. Such byte can be:

6	*(ACK)*	*Valid EEPROM code*
21 - 15 Hex	*(NACK)*	*EEPROM code not valid: command ignored*

If EEPROM code is valid it will be stored in **µP25** SRAM and will be saved into EEPROM only if a **Save Settings (S)** command is executed, as explained further.

BAUDE RATE SETTING

Input sequence:

Dec Code:	***66***	***<BAUD code>***
Hex Code:	***42***	***<BAUD code>***
Mnemonic:	***B***	***ASCII(BAUD code)***

The baud rate indicated by BAUD code will be used during RUN mode.
Possible values for BAUD code are:

0	*1200 Baud*
1	*2400 Baud*
2	*4800 Baud*
3	*9600 Baud*
4	*19200 Baud*
5	*38400 Baud*

Example:
To select a baud rate of 19200 Baud, send the following sequence: **66 4**.

Response codes:
After sending the command to **µP25**, it returns a byte indicating whether the BAUD code is correct or not. Such byte can be:

6	*(ACK)*	*Valid BAUD code*
21 - 15 Hex	*(NACK)*	*BAUD code not valid: command ignored*

If BAUD code is valid it will be stored in **µP25** SRAM and will be saved into EEPROM only if a **Save Settings (S)** command is executed, as explained further.

COMMUNICATION TYPE SETTING

Input sequence:

Dec Code:	***67***	***<Communication type code>***
Hex Code:	***43***	***<Communication type code>***
Mnemonic:	***C***	***ASCII(Communication type code)***

The communication type indicated by Communication type code is will be used during RUN mode. Possible values for Communication type code are:

0	*Point-to-point communication*
1	*9 bit Master-Slave communication*

Example:
To select 9 bit Master-Slave communication, send the following sequence: **67 1**.

Response codes:
After sending the command to **µP25**, it returns a byte indicating whether the Communication type code is correct or not. Such byte can be:

6	*(ACK)*	*Valid Communication type code*
21 - 15 Hex	*(NACK)*	*Communication type code not valid: command ignored*

If Communication type code is valid it will be stored in **µP25** SRAM and will be saved into EEPROM only if a **Save Settings (S)** command is executed, as explained further.

MASTER-SLAVE UNIT NAME SETTING

Input sequence:

Dec Code:	***78***	***<Name>***
Hex Code:	***4E***	***<Name>***
Mnemonic:	***N***	***ASCII(Name)***

This command sets the name that will be used during RUN mode in Master-Slave communication.
Possible values for the names are: ***128..255 (80..FF Hex)***.

Example:
To set the name 170, send the following sequence: **78 170**.

Response codes:
After sending the command to **µP25**, it returns a byte indicating whether the Name is correct or not. Such byte can be:

6	*(ACK)*	*Valid Communication type code*
21 - 15 Hex	*(NACK)*	*Name not valid: command ignored*

If Name is valid it will be stored in **µP25** SRAM and will be saved into EEPROM only if a **Save Settings (S)** command is executed, as explained further.

PORT SETTING

Input sequence:

Dec Code:	***80***	***<Port>***	***<Setting Byte>***
Hex Code:	***50***	***<Port>***	***<Setting Byte>***
Mnemonic:	***P***	***ASCII(Port)***	***ASCII(Setting Byte)***

This command stores the ports setting that will be used to initialize the port indicated by Port when starting to work in RUN mode.
Possible values for Port are: ***0, 1, 2***.
The meaning of Setting Byte is:

(bit 7) Px.7 Px.6 Px.5 Px.4 Px.3 Px.2 Px.1 Px.0 (bit 0)

Each bit of this byte indicates the setting of the corresponding line in the port indicated by Port:

Px.?	*0*	*This bit in the port indicated by Port is set as INPUT*
	1	*This bit in the port indicated by Port is set as OUTPUT*

If has been actived the management of A/D, D/A or M5480, the corresponding bits of port 2 can be set indifferently as inputs or outputs.

Example:
To set bits 0÷3 of port 1 as inputs and bits 4÷7 of the same port as outputs, send the following sequence: **80 1 240**.

Response codes:
After sending the command to **µP25**, it returns a byte indicating whether the command is correct or not. Such byte can be:

6	*(ACK)*	*Valid*
21 - 15 Hex	*(NACK)*	*Not valid: command ignored*

If the command is valid, the setting for the indicated port will be stored in **µP25** SRAM and will be saved into EEPROM only if a **Save Settings (S)** command is executed, as explained further.

PIN INT0 SETTING

Input sequence:

Dec Code:	***73***	***<INT0 code>***
Hex Code:	***49***	***<INT0 code>***
Mnemonic:	***I***	***ASCII(INT0 code)***

This command sets the way **INT0** (pin 14 of **µP25**) will be used during RUN mode.
Possible values are:

0	*Pin INT0 set as INPUT*
1	*Pin INT0 set as OUTPUT*
2	*Pin INT0 set as trigger for 16 bit counter*

Example:
To set pin INT0 as trigger for 16 bit counter, send the following sequence: **73 2**.

Response codes:
After sending the command to **µP25**, it returns a byte indicating whether the INT0 code is correct or not. Such byte can be:

6	*(ACK)*	*Valid INT0 code*
21 - 15 Hex	*(NACK)*	*INT0 code not valid: command ignored*

If INT0 code is valid it will be stored in **µP25** SRAM and will be saved into EEPROM only if a **Save Settings (S)** command is executed, as explained further.

A/D, D/A AND M5480 MANAGEMENT ACTIVATION

Input sequence:

Dec Code:	***70***	***<A.D.M. code>***
Hex Code:	***46***	***<A.D.M. code>***
Mnemonic:	***F***	***ASCII(A.D.M. code)***

This command indicates to **µP25** whether to activated or not during RUN mode the high level management of A/D converter **TLC 2543**, D/A converter **AD 420** or LED display driver **M 5480**. Possible values for A.D.M. code are:

Bit 0	*A/D converter **TLC2543** management **On/Off***
Bit 1	*D/A converter **AD 420** management **On/Off***
Bit 2	*LED display driver **M5480** management **On/Off***
Bit 3÷7	*Not used: **Must be set to 0***

Bits 0, 1 and 2 of A.D.M. code indicete whether to activate or not the management of the corresponding device.
In detail:

Bit x	*0*	*Device management ENABLED*
	1	*Device management DISABLED*

Example:
To manage A/D converter and M5480, send the following sequence: **70 5**.

Response codes:
After sending the command to **µP25**, it returns a byte indicating whether the A.D.M. code is correct or not. Such byte can be:

6	*(ACK)*	*Valid A.D.M. code*
21 - 15 Hex	*(NACK)*	*A.D.M. code not valid: command ignored*

If A.D.M. code is valid it will be stored in **µP25** SRAM and will be saved into EEPROM only if a **Save Settings (S)** command is executed, as explained further.

READ CURRENT CONFIGURATION

Input sequence:

Dec Code: ***76***
Hex Code: ***4C***
Mnemonic: ***L***

This command allows to read the current **µP25** configuration.

Response codes:
The **µP25** sends 9 bytes containing the following codes:

Byte 1	EEPROM code
Byte 2	BAUD code
Byte 3	Communication type code
Byte 4	Name of this unit in Master-Slave
Byte 5	PORT 0 setting
Byte 6	PORT 1 setting
Byte 7	PORT 2 setting
Byte 8	INT0 code
Byte 9	A.D.M. code

The meaning of these 9 bytes is the same reported in the previous paragraphs.

SAVE SETTINGS

Input sequence:

Dec Code: ***83***
Hex Code: ***53***
Mnemonic: ***S***

µP25 stores into serial EEPROM its current configuration, currently stored into its SRAM, then returns a code containing the result of the operation and enters in a infinite loop; at this point it is possible only to turn off the device.
If a parameter has not been changed, EEPROM will keep its previous value.

Response codes:
After saving its parameters, **µP25** returns a byte indicating whether the operation has been succesful or not. Such byte can be:

6	*(ACK)*	*Parameters stored in EEPROM without errors*
21 - 15 Hex	*(NACK)*	*Error while storing parameters in EEPROM*

SETUP MODE COMMANDS SUMMARIZING TABLE

Here follows the setup mode commands summarizing table.

COMMAND	CODE	HEX CODE	MNEMONIC
EEPROM Setting	69 EE code	45 EE code	E ASCII(EE code)
BAUD RATE Setting	66 Baud code	42 Baud code	B ASCII(Baud code)
Communication Type Setting	67 Comm code	43 Comm code	C ASCII(Comm code)
Master-Slave Unit Name setting	78 Name	4E Name	N ASCII(Name)
Port setting	80 Port Byte	50 Port Byte	P ASCII(Port) ASCII(Byte)
Pin INT0 Setting	73 Int0 code	49 Int0 code	I ASCII(Int0 code)
A/D, D/A And M5480 Management Activation	70 code	4D code	M ASCII(code)
Read Current Configuration	76	4C	L
Save Settings	83	53	S

FIGURE 5: SETUP MODE COMMANDS SUMMARIZING TABLE

RUN MODE

When entering in **RUN** mode the baud rate, communication protocol and other parameters stored in EEPROM are verified.
If they are valid, **µP25** sets the **/SETUP** signal (pin 15) to logic level **0**, performs the initialization and begins to stand by commands.
If they are not valid (e.g. EEPROM not initialized) the board sets the **/SETUP** signal (pin 15) to logic level **1**, then **µP25** enters in an infinite loop; at this point the user may only turn off the device.

EEPROM is NOT initialized by default, the the user must initialize it (SETUP mode) before attempting to use the board.

Baud rate and communication protocol for the **RUN** mode are settable (in **SETUP** mode), while data format is function of the selected communication mde, as follows:

Point-to-Point Communication:	**8 bit, 1 Stop, NO Parity**
Master-Slave:	**9 bit, 1 Stop, NO Parity**

In the following paragraphs the commands recognized in **RUN** mode are described.

GENERAL COMMANDS

MASTER RESET

Input Sequence:

Dec Code:	**65**	**97**
Hex Code:	**41**	**61**
Mnemonic:	**A**	**a**

Upon the reception of this command the firmware restores the initial condition that happens to be at the Power-ON; in detail:

Ports 0, 1 and 2:	They are reset and put into initial condition, so all the outputs are set to logic state 0. Eventual timings on port 0 are interrupted.
Pin INT0:	If configured as output, it is set to logic state 0.
16 bits Counter:	It is initialized to 0.
D/A converter:	If its management is enabled, it is initialized with combination 0.
M5480:	If its management is enabled, all its outputs are reset.

DIGITAL I/O PORT MANAGEMENT COMMANDS

Here follow the commands for managing, reading from and writing port 0, 1 and 2 of **µP25**.

OUTPUT PORT SET

Input Sequence:

Dec Code:	**87 <Port> <Nibble L Data> <Nibble H Data>**
Hex Code:	**57 <Port> <Nibble L Data> <Nibble H Data>**
Mnemonic:	**W ASCII(<Port>) ASCII(<Nibble L Data>) ASCII(<Nibble H Data>)**

The byte, sent as low nibble and high nibble, is written to the port indicated by <Port>.
<Port> can assume the values: **0**, **1** or **2**.
The <Data> byte must be sent in nibbles according to the following format:

Nibble L:	(MSB)	0	0	0	0	**Px.3**	**Px.2**	**Px.1**	**Px.0**	(LSB)
Nibble H:		0	0	0	0	**Px.7**	**Px.6**	**Px.5**	**Px.4**	

Where **Px.?** stands for the logic status, 0 or 1, that the corresponding bit of the specidied port must get.
Only the bits configured as outputs will be changed to the status indicated, if the whole port is set as input the command will be ignored, also if the sequence contains invalid data the command is ignored.
If the management of A/D, D/A or M5480 is enabled and the command specifes port 2, only the bits not used to managed the mentioned devices will be changed.

Example:
To write the byte 90 (5A Hex) to port 0, send the following sequence:
87 0 10 5.

Note:
If in the indicated port some bit are configured as inputs, the stauts change requested for the bits configured as output will not happen simultaneously but sequentially. The delay between two consecutive bit status changes will be about **50 µsec**.

INPUT PORT ACQUISITION

Input Sequence:

Dec Code:	**82**	**<Port>**
Hex Code:	**52**	**<Port>**
Mnemonic:	**R**	**ASCII(Port)**

The value read from the port indicated by <Port> is sent to the serial line. <Port> can be **0**, **1** or **2**.
Only the status of the bits configured as inputs will be acquired, while the **Px.?** positions corresponding to bits configures as outputs will be set to 0. So if the whole port is configured as output the returned value will be 0.

Response codes:
The data acquired by the port is returned as nibbles in the following format:

Nibble L:	(MSB)	0	0	0	0	**Px.3**	**Px.2**	**Px.1**	**Px.0**	(LSB)
Nibble H:		0	0	0	0	**Px.7**	**Px.6**	**Px.5**	**Px.4**	

Where **Px.?** stands for the logic status, 0 or 1, that the corresponding bit of the specidied port has.

Only the bits configured as inputs will be acquired, while the **Px.?** corresponding to bits confugured as outputs or used to manage A/D, D/A or M5480 will return **0**. If the whole port is set as input the command will be ignored, also if the sequence contains invalid data the command is ignored.

Example:
To read port 1, where the **90** (5A Hex) data is present, send the following sequence: **82 1.**
Then you will get as answer the following bytes: **10 5.**

READ PORT CONFIGURATION

Input Sequence:

Dec Code:	**70**
Hex Code:	**46**
Mnemonic:	**F**

The configuration bytes of ports 0, 1 and 2 of **µP25**, previously set in SETUP mode, are returned.

Response codes:
The three configuration bytes of ports 0, 1 and 2 are sent to the serial line as nibbles, in detail the following codes are sent:

Port0 Nibble L:	(MSB)	0	0	0	0	**P0.3**	**P0.2**	**P0.1**	**P0.0**	(LSB)
Nibble H:		0	0	0	0	**P0.7**	**P0.6**	**P0.5**	**P0.4**	
Port1 Nibble L:		0	0	0	0	**P1.3**	**P1.2**	**P1.1**	**P1.0**	
Nibble H:		0	0	0	0	**P1.7**	**P1.6**	**P1.5**	**P1.4**	
Port2 Nibble L:		0	0	0	0	**P2.3**	**P2.2**	**P2.1**	**P2.0**	
Nibble H:		0	0	0	0	**P2.7**	**P2.6**	**P2.5**	**P2.4**	

Each bit indicated with a pin name gives the configuration of that pin. In detail:

Px.?	*0*	*The indicated bit is configured as INPUT*
	1	*The indicated bit is configured as OUTPUT*

DIGITAL I/O BIT MANAGEMENT COMMANDS

Here follow the commands for managing, reading from and writing to single bits of port 0, 1 and 2 of µ**P25**.

OUTPUT BIT SET

Input Sequence:

Dec Code:	**83**	**<Port>**	**<Bit>**
Hex Code:	**53**	**<Port>**	**<Bit>**
Mnemonic:	**S**	**ASCII(Port)**	**ASCII(Bit)**

The output indicated by **<Bit>** in the port indicated by **<Port>** gets the logic status 1; **<Bit>** can range from **0** to **7**, **<Port>** can range from **0** to **3**.

If pin **INT0** is configured as output, to set it bit 0 of port 3 will have to be indicated, all other bits of port 3 will be considered wrong codes.

Note:
Eventual timings occourring on the output line are interrupted.
Only the bits configured as outputs will change their stauts, while indicating bits confugured as inputs or used to manage A/D, D/A or M5480 the command will be ignored, also if the sequence contains invalid data the command is ignored.

Example:
To set bit 4 of port 0, send the following sequence: **83** **0** **4.**

OUTPUT BIT CLEAR

Input Sequence:

Dec Code:	**67**	**<Port>**	**<Bit>**
Hex Code:	**43**	**<Port>**	**<Bit>**
Mnemonic:	**S**	**ASCII(Port)**	**ASCII(Bit)**

The output indicated by **<Bit>** in the port indicated by **<Port>** gets the logic status 0; **<Bit>** can range from **0** to **7**, **<Port>** can range from **0** to **3**.

If pin **INT0** is configured as output, to reset it bit 0 of port 3 will have to be indicated, all other bits of port 3 will be considered wrong codes.

Note:
Eventual timings occourring on the output line are interrupted.
Indicating bits confugured as inputs or used to manage A/D, D/A or M5480 the command will be ignored, also if the sequence contains invalid data the command is ignored.

Example:
To reset bit 7 of port 2, send the following sequence: **67** **2** **7.**

TIMED OUTPUT BIT SET

Input Sequence:

Dec Code:	**115**	**<Port>**	**<Bit>**	**<NibL>**	**<NibH>**
Hex Code:	**73**	**<Port>**	**<Bit>**	**<NibL>**	**<NibH>**
Mnemonic:	**s**	**ASCII(Port)**	**ASCII(Bit)**	**ASCII(NibL)**	**ASCII(NibH)**

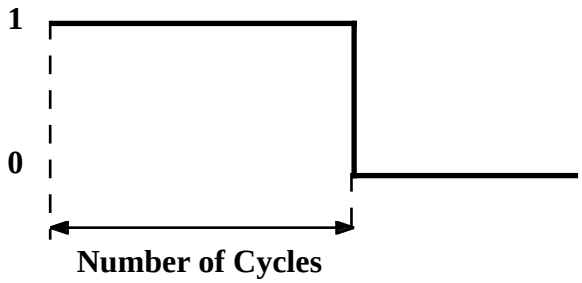


FIGURE 6: TIMED SET COMMAND

The output indicated by **<Bit>** in the port indicated by **<Port>** gets the logic status 1; **<Bit>** can range from **0** to **7**, **<Port>** can be selected between **0** and **3**, no timed output is possible on the remaining ports.

If pin **INT0** is configured as output, to set it in timed mode bit 0 of port 3 will have to be indicated, all other bits of port 3 will be considered wrong codes.

The selected output holds the logic status 1 for a time determined by the **<Nib>** byte, then it returns to logic status 0.
The byte must be sent to the serial line as nibbles in the following format:

Nibble L:	(MSB)	0	0	0	0	**Bit3**	**Bit2**	**Bit1**	**Bit0**	(LSB)
Nibble H:		0	0	0	0	**Bit7**	**Bit6**	**Bit5**	**Bit4**	

The timing value must range **1÷255** where one unit corresponds to **10 msec**.

Note:
Indicating bits confugured as inputs the command will be ignored, also if the sequence contains invalid data the command is ignored.

Example:
To set the pin of **INT0** for 500 msec, send the following sequence:
115 3 0 2 3.

TIMED OUTPUT BIT CLEAR

Input Sequence:

Dec Code:	**99**	**<Port>**	**<Bit>**	**<NibL>**	**<NibH>**
Hex Code:	**63**	**<Port>**	**<Bit>**	**<NibL>**	**<NibH>**
Mnemonic:	**c**	**ASCII(Port)**	**ASCII(Bit)**	**ASCII(NibL)**	**ASCII(NibH)**

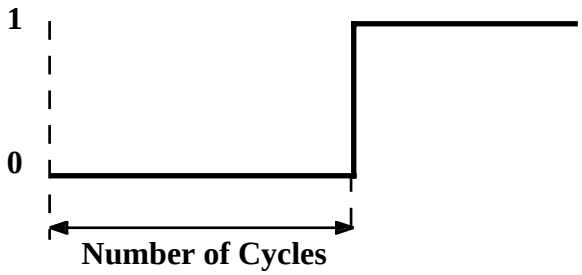


FIGURE 7: TIMED CLEAR COMMAND

The output indicated by **<Bit>** in the port indicated by **<Port>** gets the logic status 0; **<Bit>** can range from **0** to **7**, **<Port>** can be selected between **0** and **3**, no timed output is possible on the remaining ports.

If pin **INT0** is configured as output, to reset it in timed mode bit 0 of port 3 will have to be indicated, all other bits of port 3 will be considered wrong codes.

The selected output holds the logic status 0 for a time determined by the **<Nib>** byte, then it returns to logic status 1.
The byte must be sent to the serial line as nibbles in the following format:

Nibble L:	(MSB)	0	0	0	0	**Bit3**	**Bit2**	**Bit1**	**Bit0**	(LSB)
Nibble H:		0	0	0	0	**Bit7**	**Bit6**	**Bit5**	**Bit4**	

The timing value must range **1÷255** where one unit corresponds to **10 msec**.

Note:
Indicating bits confugured as inputs the command will be ignored, also if the sequence contains invalid data the command is ignored.

Example:
To reset bit 7 of port 0 for 1 sec, send the following sequence:
99 0 7 4 6.

SQUARE WAVE OUTPUT BIT

Input Sequence:

Dec Code:	**80**	**<Port>**	**<Bit>**	**<NibL>**	**<NibH>**
Hex Code:	**50**	**<Port>**	**<Bit>**	**<NibL>**	**<NibH>**
Mnemonic:	**P**	**ASCII(Port)**	**ASCII(Bit)**	**ASCII(NibL)**	**ASCII(NibH)**

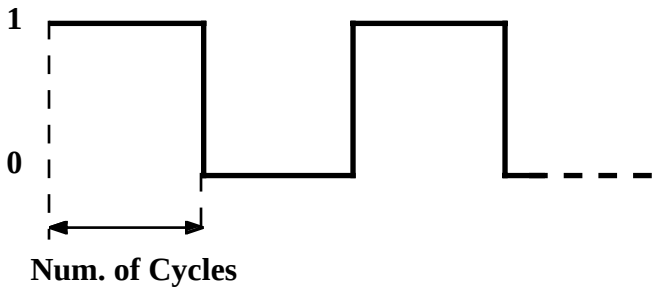


FIGURE 8: SQARE WAVE COMMAND

A sqare wave featuring 50% of duty cycle is generated to the output indicated by **<Bit>** in the port indicated by **<Port>**; **<Bit>** can range from **0** to **7**, **<Port>** can be selected between **0** and **3**, no timed output is possible on the remaining ports.

If pin **INT0** is configured as output, to apply to it this command bit 0 of port 3 will have to be indicated, all other bits of port 3 will be considered wrong codes.

The duration of the sqare wave is determined by the **<Nib>** byte, which indicates an half period. The byte must be sent to the serial line as nibbles in the following format:

Nibble L:	(MSB)	0	0	0	0	**Bit3**	**Bit2**	**Bit1**	**Bit0**	(LSB)
Nibble H:		0	0	0	0	**Bit7**	**Bit6**	**Bit5**	**Bit4**	

The timing value must range **1÷255** where one unit corresponds to **10 msec**.

Note:
Indicating bits confugured as inputs the command will be ignored, also if the sequence contains invalid data the command is ignored.

Example:
To reset bit 2 of port 0 for 200 msec, send the following sequence:
80 0 2 10 0.

SQUARE WAVE STARTING WITH "1" OUTPUT BIT

Input Sequence:

Dec Code:	**112**	**<Port>**	**<Bit>**	
	<NibL>	**<NibH>**	**<StaL>**	**<StaL>**
Hex Code:	**70**	**<Port>**	**<Bit>**	
	<NibL>	**<NibH>**	**<StaH>**	**<StaH>**
Mnemonic:	**p**	**ASCII(Port)**	**ASCII(Bit)**	
	ASCII(NibL)	**ASCII(NibH)**	**ASCII(StaL)**	**ASCII(StaH)**

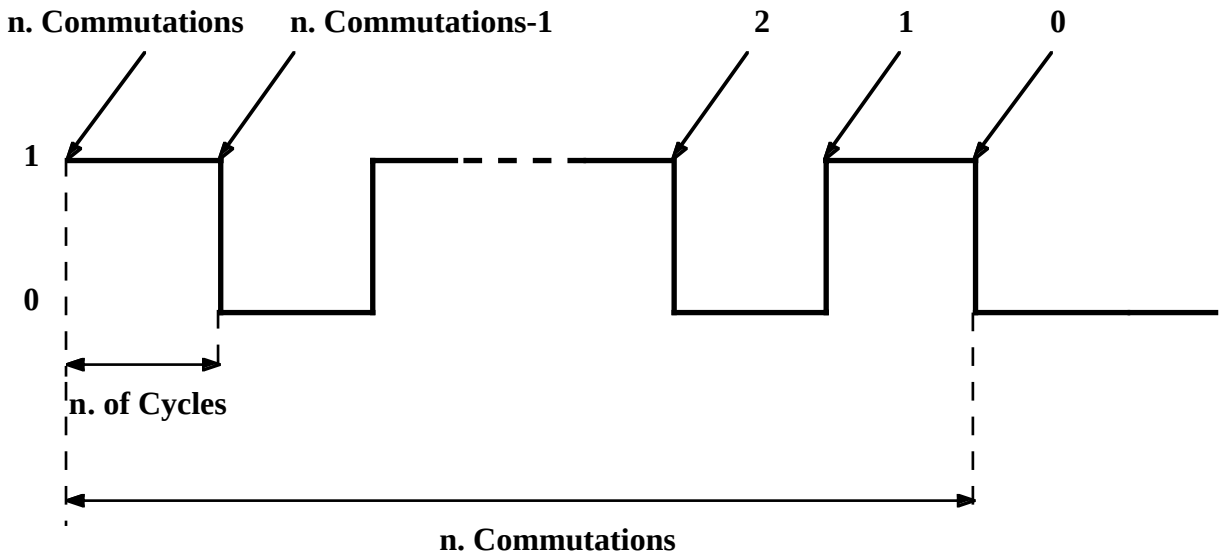


FIGURE 9: TIMED SQUARE WAVE COMMAND

A square wave featuring 50% of duty cycle and lasting for a specified time is generated to the output indicated by **<Bit>** in the port indicated by **<Port>**; **<Bit>** can range from **0** to **7**, **<Port>** can be selected between **0** and **3**, no timed output is possible on the remaining ports.

If pin **INT0** is configured as output, to apply to it this command bit 0 of port 3 will have to be indicated, all other bits of port 3 will be considered wrong codes.

The period of the sqare wave is determined by the **<Nib>** byte, which indicates an half period. The byte must be sent to the serial line as nibbles in the following format:

Nibble L:	(MSB)	0	0	0	0	**Bit3**	**Bit2**	**Bit1**	**Bit0**	(LSB)
Nibble H:		0	0	0	0	**Bit7**	**Bit6**	**Bit5**	**Bit4**	

The timing value must range **1÷255** where one unit corresponds to **10 msec**.
The total duration of the sqare wave is determined by the **<Sta>** byte, . The byte must be sent to the serial line as nibbles in the previous format.
It indicated the number of status toggles that the pin must perform before it return definitely to the logic status 0, as it can be seen in the figure, the number of status changes is **<Sta>**+1.

Note:
Indicating bits confugured as inputs the command will be ignored, also if the sequence contains invalid data the command is ignored.

Example:
To generate a square wave on pin INT0, with 200 msec of period, corresponding to 10 cycles, and 10 status toggles, send the following sequence:
112 3 0 10 0 9 0.

INPUT BIT

Input Sequence:

Dec Code:	**114**	**<Port>**	**<Bit>**
Hex Code:	**72**	**<Port>**	**<Bit>**
Mnemonic:	**r**	**ASCII(Port)**	**ASCII(Bit)**

The logic status of the bit indicated by **<Bit>** in the port indicated by **<Port>** is returned; **<Bit>** can range from **0** to **7**, **<Port>** can range from **0** to **3**.

If pin **INT0** is configured as input, to get its status bit 0 of port 3 will have to be indicated, all other bits of port 3 will be considered wrong codes.

Response codes:
The status of the indicated pin is returned, in detail:

0 *means that the pin has logic status 0*
1 *means that the pin has logic status 1*

Note:
Only the bits configured as inputs will be acquired, while bits confugured as outputs or used to manage A/D, D/A or M5480 will return **0**.
If the sequence contains invalid data the command is ignored.

Example:
To get the staus of pin 7 on port 2, which has logical status 1, send the following sequence:
114 2 7; the value returned will be **1.**

INPUT WITH DEBOUNCING

Input Sequence:

Dec Code:	**68**	**<Port>**	**<Bit>**	**<NibL>**	**<NibH>**
Hex Code:	**44**	**<Port>**	**<Bit>**	**<NibL>**	**<NibH>**
Mnemonic:	**D**	**ASCII(Port)**	**ASCII(Bit)**	**ASCII(NibL)**	**ASCII(NibH)**

The logic status of the bit indicated by **<Bit>** in the port indicated by **<Port>**, filtered by a debouncing procedure whose duration can be specified, is returned; **<Bit>** can range from **0** to **7**, **<Port>** can range from **0** to **3**.

If pin **INT0** is configured as input, to get its status bit 0 of port 3 will have to be indicated, all other bits of port 3 will be considered wrong codes.

The debouncing time is indicated by the value of **<Nib>** byte.
The timing value must range **1÷255** where one unit corresponds to **10 msec**.

Response codes:
The status of the indicated pin is returned, in detail:

0	*means that the pin has always kept the logic status 0 during the debouncing*
1	*means that the pin has always kept the logic status 1 during the debouncing*
7	*means that the pin haschanged the logic status during the debouncing*

Note:
Only the bits configured as inputs will be acquired, while bits confugured as outputs or used to manage A/D, D/A or M5480 will return **0**.
If the sequence contains invalid data the command is ignored.

Example:
To get the staus of pin 3 on port 1, which has logical status 0, with 50 msec of debouncing time, corresponding to 5 units, send the following sequence: **68 1 3 5 0.**

READ PIN INT0 CONFIGURATION

Input Sequence:

Dec Code:	**102**
Hex Code:	**6**
Mnemonic:	**f**

The configuration code of pin INT0, previously set in SETUP mode, is returned.

Response codes:
The values of pin INT0 configuration byte can be:

0	*pin INT0 configured as INPUT line*
1	*pin INT0 configured as OUTPUT line*
2	*pin INT0 configured as trigger for 16 bits counter*

MESSAGES MANAGEMENT COMMANDS

Here follow the commands for managing the messages in EEPROM.

LAST MEMORIZABLE MESSAGE ACQUISITION

Input Sequence:

Dec Code: **77**
Hex Code: **4D**
Mnemonic: **M**

This command allows the user to know the maximum number of messages that the EEPROM of **µP25** can store. This number depends on the memory device installed according to the following table:

EEPROM	N.MAX
24c02 (256 Bytes)	23
24c04 (512 Bytes)	48
24c16 (2048 Bytes)	202

FIGURE 10: MAXIMUM NUMBER OF MESSAGES STORABLE IN EEPROM

Response codes:
The number is returned in two nibbles according to the format:

Nibble L:	(MSB)	0	0	0	0	**Bit3**	**Bit2**	**Bit1**	**Bit0**	(LSB)
Nibble H:		0	0	0	0	**Bit7**	**Bit6**	**Bit5**	**Bit4**	

READY EEPROM REQUEST

Input Sequence:

Dec Code: **66**
Hex Code: **42**
Mnemonic: **B**

By this command the user may ask the firmaware if it is ready to manage a new EEPROM message, this command must be sent whenever the user needs to send one of the following messages management commands.

Response codes:
The firware returns the following codes:

0	EEPROM not ready to manage a new message
1	EEPROM ready to manage a new message

STORING A MESSAGE

Input Sequence:

Dec Code:	**69**	**<NibL Mess> <NibH Mess>**
		<NibL Chr.0> <NibH Chr.0>.....<NibL Chr.9> <NibH Chr.9>
Hex Code:	**45**	**<NibL Mess> <NibH Mess>**
		<NibL Chr.0> <NibH Chr.0>.....<NibL Chr.9> <NibH Chr.9>
Mnemonic:	**E**	**ASCII(NibL Mess) ASCII(NibH Mess)**
		ASCII(NibL Chr.0) ASCII(NibH Chr.0).....
		ASCII(NibL Chr.9) ASCII(NibH Chr.9)

The ten characters long message, whose code is indicated by **Mess**, is stored in EEPROM. The message number must range **0÷N.MAX** (see Figure 10 for the value of N.MAX) and must be sent in two nibbles as above indicated. The value of N.MAX may also be acquired by the specific command. The characters must be sent in two nibbles according to the following format:

Car. x	**Nibble L:**	(MSB)	0	0	0	0	**Bit3**	**Bit2**	**Bit1**	**Bit0** (LSB)
	Nibble H:		0	0	0	0	**Bit7**	**Bit6**	**Bit5**	**Bit4**

These bytes must range **0÷255** (**0÷FF** Hex).

Note:
If the sequence contains invalid data the command is ignored.

Example:
If you want to store as the message nubmer 16 the string "ABCDEFGHIJ" (corresponding to the codes: 65, 66, 67, 68, 69, 70, 71, 72, 73, 74), send the following sequence:
69 0 1 1 4 2 4 3 4 4 4 5 4 6 4 7 4 8 4 9 4 10 4.

READING A MESSAGE

Input Sequence:

Dec Code:	**76**	**<NibL Mess>**	**<NibH Mess>**
Hex Code:	**4C**	**<NibL Mess>**	**<NibH Mess>**
Mnemonic:	**L**	**ASCII(NibL Mess)**	**ASCII(NibH Mess)**

The ten characters long message is read from the EEPROM and sent on the serial connection. The message number must range **0÷N.MAX** (see Figure 10 for the value of N.MAX) and must be sent in two nibbles as above indicated. The value of N.MAX may also be acquired by the apposite command.

Note:
If the sequence contains invalid data the command is ignored.

Response codes:
The ten characters are returned in nibbles according to the format seen for the previous command.

Example:
If you want to read the message with number 16 stored in the previous example, you need to send the following sequence: **76 0 1.** The answer will be the sequence:
1 4 2 4 3 4 4 4 5 4 6 4 7 4 8 4 9 4 10 4.

16 BIT COUNTER MANAGEMENT COMMANDS

Here follow the commands to manage the 16 bit counter. Its value is incremented by the commutations of the pin INT0 when it is configured to trigger the counter.

16 BIT COUNTER READ

Input Sequence:

Dec Code:	**73**
Hex Code:	**49**
Mnemonic:	**I**

This command allows to acquire the current value of the 16 bit counter.

Response codes:
The sequence returned by the command is made of four bytes showing the 16 bit value currently stored in the counter register; this is sent in nibbles according to the following format:

Counter	(bit 0÷3)	: (MSB)	0	0	0	0	**Bit3**	**Bit2**	**Bit1**	**Bit0** (LSB)
	(bit 4÷7)	:	0	0	0	0	**Bit7**	**Bit6**	**Bit5**	**Bit4**
	(bit 8÷11)	:	0	0	0	0	**Bit11**	**Bit10**	**Bit9**	**Bit8**
	(bit 12÷15)	:	0	0	0	0	**Bit15**	**Bit14**	**Bit13**	**Bit12**

When the counter reaches the maximum value, which equals to **65535** (**FFFF Hex**), the next trigger impulse will set the counter to 0. If th epin INT0 is configurated as INPUT this command will always return **0**.

Example:
If you the counter register contains the value 23055 (5A0F Hex), sending the command **73** will return the following values: **15 0 10 5**.

16 BIT COUNTER RESET

Input Sequence:

Dec Code:	**88**	**120**
Hex Code:	**58**	**78**
Mnemonic:	**X**	**x**

Upon the reception of this command the firmware resets the 16 bit counterwith the new value **0**.

A/D, D/A AND M5480 MANAGEMENT COMMANDS

Here follow the high level commands available to manage A/D converter **TLC 2543**, D/A converter **AD 420** and LED display driver **M5480**.
These devices must be interfaced to **µP25** through the lines of **port 2** as described in the chapter dedicated to the hardware connections.
To take advantage of these commands it is essential to enable the management of the related peripheral through the specific commands of SETUP mode.

A/D CONVERSION ON ONE CHANNEL

Input Sequence:

Dec Code:	**81**	**<Channel>**
Hex Code:	**51**	**<Channel>**
Mnemonic:	**Q**	**ASCII(Channel)**

This command performs a conversion on one of the **TLC 2543** A/D converter channels, interfaced to **µP25** through bits **0÷3** of **port 2**. The number of the channel to convert is indicated in the parameter **<Channel>**, so it must be included in the range 0÷10.
The combination acquired is a 12 bit value, so it is included in the range 0÷4095 (0÷3FF Hex), in detail:

Combination = 0	->	$Vin = V_{ref-}$
Combination = 2048 (200 Hex)	->	$Vin = (V_{ref+} - V_{ref-}) / 2$
Combination = 4095 (3FF Hex)	->	$Vin = V_{ref+}$

Where Vin is the tension applied to the A/D input to be converted, while V_{ref+} and V_{ref-} mean, respectively, the positive and the negative A/D reference values.

Response codes:
The 12 bit combination is sent as nibbles, in 3 bytes, according to the following format:

Combination	(bit 0÷3)	: (MSB)	0	0	0	0	**Bit3**	**Bit2**	**Bit1**	**Bit0**	(LSB)
	(bit 4÷7)	:	0	0	0	0	**Bit7**	**Bit6**	**Bit5**	**Bit4**	
	(bit 8÷11)	:	0	0	0	0	**Bit11**	**Bit10**	**Bit9**	**Bit8**	

Note:
If the high level management of A/D converter is disabled, the request of a conversion will always return **0**.
If the sequence contains invalid data the command is ignored.

Example:
To execute a conversion on channel 5 of A/D converter, where a tension of value V_{ref+} is present, sending the following sequence: **81 5**.
µP25 will return the following values: **15 15 3**.

D/A OUTPUT MANAGEMENT

Input Sequence:

Dec Code:	***113***	***<Bit 0÷3>***	***<Bit 4÷7>***	***<Bit 8÷11>***	***<Bit 12÷15>***
Hex Code:	**71**	***<Bit 0÷3>***	***<Bit 4÷7>***	***<Bit 8÷11>***	***<Bit 12÷15>***
Mnemonic:	**q**	**ASCII(*Bit 0÷3*)**		**ASCII(*Bit 4÷7*)**	
		ASCII(*Bit 8÷11*)		**ASCII(*Bit 12÷15*)**	

Manages the analog output, setting it with the desired 16 bit combination, of A/D converter **AD 420** which is interfaced to **µP25** through bit **1**, **6** and **7** of **port 2**.
The 16 bit combination is sent as nibbles to the serial connection according to the following format:

Combination	(bit 0÷3)	:	(MSB)	0	0	0	0	**Bit3**	**Bit2**	**Bit1**	**Bit0** (LSB)
	(bit 4÷7)	:		0	0	0	0	**Bit7**	**Bit6**	**Bit5**	**Bit4**
	(bit 8÷11)	:		0	0	0	0	**Bit11**	**Bit10**	**Bit9**	**Bit8**
	(bit 12÷15)	:		0	0	0	0	**Bit15**	**Bit14**	**Bit13**	**Bit12**

The D/A converter analog output will assume a value, in V or mA, proportional to the value of the 16 bit combination, in detail:

Combination = 0	->	*Out = Smallest possible output*
Combination = 65535 (FFFF Hex)	->	*Out = Full range*

Smallest possible output and full range values depend on the range configured on the D/A converter through the specific jumpers, they can be **0÷10 V, 4÷20 mA, 0÷20 mA** or **0÷24 mA**.

Note:
If the high level management of D/A converter is disabled, the command will be ignored.
Also, if the sequence contains invalid data the command is ignored.

Example:
To set the combination value 23055 (5A0F Hex) on the D/A converter output, send the following sequence: **113 15 0 10 5**.

DISPLAY DRIVER MANAGEMENT

Input Sequence:

Dec Code:	***109***	***<Out 1÷4>***	***<Out 5÷8>***	***<Out 9÷12>***
		<Out 13÷16>	***<Out 17÷20>***	***<Out 21÷23>***
Hex Code:	**6D**	***<Out 1÷4>***	***<Out 5÷8>***	***<Out 9÷12>***
		<Out 13÷16>	***<Out 17÷20>***	***<Out 21÷23>***
Mnemonic:	**m**	**ASCII(*Out 1÷4*)**	**ASCII(*Out 5÷8*)**	**ASCII(*Out 9÷12*)**
		ASCII(*Out 13÷16*)	**ASCII(*Out 17÷20*)**	**ASCII(*Out 21÷23*)**

Sets or resets the 23 outputs of **M5480** LED display driver, interfaced to **µP25** through bit **4** and **5** of port **2**.
The six parameters allow to decide which lines must be set to logic status 1 and which must be reset to logic status 0 according to the followin correspondance:

Manages the analog output, setting it with the desired 16 bit combination, of A/D converter **AD 420** which is interfaced to **µP25** through bit **1**, **6** and **7** of **port 2**.
The 16 bit combination is sent as nibbles to the serial connection according to the following format:

M5480 Output	(Out 1÷4)	: (MSB)	0	0	0	0	**Out 4**	**Out 3**	**Out 2**	**Out 1**	(LSB)
	(Out 5÷8)	:	0	0	0	0	**Out 8**	**Out 7**	**Out 6**	**Out 5**	
	(Out 9÷12)	:	0	0	0	0	**Out 12**	**Out 11**	**Out 10**	**Out 9**	
	(Out 10÷16)	:	0	0	0	0	**Out 16**	**Out 15**	**Out 14**	**Out 13**	
	(Out 17÷20)	:	0	0	0	0	**Out 20**	**Out 19**	**Out 18**	**Out 17**	
	(Out 21÷23)	:	0	0	0	0	0	**Out 23**	**Out 22**	**Out 21**	

Each of the 23 outputs of **M5480** will be set or reset according to the status of the corresponding bit in the above described mask, in detail

Out x	*0*	*Output x of M5480 DISABLED*
	1	*Output x of M5480 ENABLED*

Note:
If the high level management of **M5480** is disabled, the command will be ignored.
Also, if the sequence contains invalid data the command is ignored.

Example:
To activate the outputs 1, 6, 8, 15, 22 and 23 of **M5480**, send the following sequence:
109 1 10 0 4 0 6.

READ CONFIGURATION BYTE OF A/D, D/A AND M5480

Input Sequence:

Dec Code: ***105***
Hex Code: **69**
Mnemonic: **i**

Allows to acquire the byte whose bits indcate whether are enabled or not the high level managements of A/D converter **TLC 2543**, D/A converter **AD 420** and LED display driver **M5480**.

Response code:
The byte returned through the serial line by **µP25** has the following meaning:

Bit 0	*Management of A/D converter TLC2543* ***On/Off***
Bit 1	*Management of A/D converter AD420* ***On/Off***
Bit 2	*Management of LED display driver M5480* ***On/Off***
Bit 3÷7	*Not used:* ***always return 0***

Bits 0, 1 and 2 indicate whether the high level management of a device is enabled or not, in detail:

Out x	*0*	*Output x of M5480 DISABLED*
	1	*Output x of M5480 ENABLED*

Note:
If the high level management of **M5480** is disabled, the command will be ignored.
Also, if the sequence contains invalid data the command is ignored.

Example:
If the high level management of A/D converter and M5480 are enabled, sending the command **105** will generate as anwer the byte **5**.

RUN MODE COMMANDS SUMMARIZING TABLE

Here follow the run mode commands summarizing tables.

COMMAND	CODE	HEX CODE	MNEMONIC
Output Port Set	87 port nib.L nib.H	57 port nib.L nib.H	W ASCII(port) ASCII(nib.L) ASCII(nib.H)
Input Port Acquisition	82 port	52 port	R ASCII(port)
Output Bit Set	83 port bit	53 port bit	S ASCII(port) ASCII(bit)
Output Bit Clear	67 port bit	43 port bit	C ASCII(port) ASCII(bit)
Timed Output Bit Set	115 port bit nib.L nib.H	73 port bit nib.L nib.H	s ASCII(port) ASCII(bit) ASCII(nib.L) ASCII(nib.H)
Timed Output Bit Clear	99 port bit nib.L nib.H	63 port bit nib.L nib.H	s ASCII(port) ASCII(bit) ASCII(nib.L) ASCII(nib.H)
Square Wave Output Bit	80 port bit nib.L nib.H	50 port bit nib.L nib.H	P ASCII(port) ASCII(bit) ASCII(nib.L) ASCII(nib.H)
Square Wave Starting With "1" Output Bit	112 port bit nib.L nib.H nib.L nib.H	70 port bit nib.L nib.H nib.L nib.H	p ASCII(port) ASCII(bit) ASCII(nib.L) ASCII(nib.H) ASCII(nib.L) ASCII(nib.H)
Input Bit	114 port bit	72 port bit	r ASCII(port) ASCII(bit)
Input Bit Debouncing	68 port bit nib.L nib.H	44 port bit nib.L nib.H	D ASCII(port) ASCII(bit) ASCII(nib.L) ASCII(nib.H)
16 Bit Counter Reset	88 120	58 78	X x
16 Bit Counter Read	73	49	I
Master Reset	65 97	41 61	A a

FIGURE 11: RUN MODE COMMANDS SUMMARIZING TABLE 1

COMMAND	CODE	HEX CODE	MNEMONIC
Ready EEPROM Request	66	42	B
Last Memorizable Message Acquisition	77	4D	M
Storing A Message	69 nib.L nib.H nib.L0 nib.H0 ..... nib.L9 nib.H9	45 nib.L nib.H nib.L0 nib.H0 ..... nib.L9 nib.H9	E ASCII(nib.L) ASCII(nib.H) ASCII(nib.L0) ASCII(nib.H0) ..... ASCII(nib.L9) ASCII(nib.H9)
Reading A Message	76 nib.L nib.H	4C nib.L nib.H	L ASCII(nib.L) ASCII(nib.H)
A/D Conversion On One Channel	81 Channel	51 Channel	Q ASCII(Channel)
D/A Output Management	113 nib.L0 nib.H0 nib.L1 nib.H1	71 nib.L0 nib.H0 nib.L1 nib.H1	q ASCII(nib.L0) ASCII(nib.H0) ASCII(nib.L1) ASCII(nib.H1)
Display Driver Management	109 nib.L0 nib.H0 nib.L1 nib.H1 nib.L2 nib.H2	6D nib.L0 nib.H0 nib.L1 nib.H1 nib.L2 nib.H2	m ASCII(nib.L0) ASCII(nib.H0) ASCII(nib.L1) ASCII(nib.H1) ASCII(nib.L2) ASCII(nib.H2)
Read Port Configuration	70	46	F
Read Pin INT0 Configuration	102	66	f
Read Configuration Byte Of A/D, D/A And M5480	105	69	i

FIGURE 12: RUN MODE COMMANDS SUMMARIZING TABLE 2

9 BITS MASTER-SLAVE COMMUNICATION MODE

The Master-Slave communication takes advantage of the 9 bits mode, this means that a nineth bit is used to distinguish between a call from a "**Master**" device to one of the "**Slave**" structures and a mere communication of datas between the Master and the selected Slave.
When the nineth bit is set to 1, the data byte must contain the name, or ID code, of the new target device, while if the nineth bit is set to 0 it is possible to send or receive informations from the selected target device.

If the communication is running under **µP25** protocol the ID code must be the byte set in **SETUP mode (NAME).** When this byte is received by a device (**with the nineth bit set to 1**) it recognizes itself and starts to wait for a string containing dats or commands (**with the nineth bit set to 0**); the string must be maximun **24 bytes** long.
It can contain only one command which requires to return an answer code through the serial line, more commands of this kind will be ignored.
The delay between two consecutive characters must be lower than **Time-Out,** because otherwise the string is considered terminated and the answering phase starts.
Here follows the list of Time-Outs related to the Baud Rate:

Baud Rate	***Time-Out***
38400 Baud	*550 µsec*
19200 Baud	*990 µsec*
9600 Baud	*1.54 msec*
4800 Baud	*3.08 msec*
2400 Baud	*6.105 msec*
1200 Baud	*12.1 msec*

When the Time-Out happens, the answer sequence begins; this is made of a byte containing the presence code **13** (**0D** Hex), or a data sequence requested by a read command sent during the previous call.

Example:
If a string containing the Port read command is sent, the answer to that call will be the presence code, while the answer to the next call will be the data acquired by the Port sent in the previous request.

After having sent the last character of the string the user will have to wait for a time:

"One char transmission time" + Time-Out

before receiving the first char of the answer sequence.

Example:
At 38.4 KBaud, After having sent the last character of the string the user will have to wait for about 810 µsec before to receive the first char of the answer sequence.

Note:
Between two successive calls, it is essential to wait for a time that depends on the number of commands sent and the kind of operations these commands require.
Each string of commands or data transmitted by the “Master” unit must always contain complete command sequences. If one of these sequences is incomplete it will be ignored, and so the successive sequences, even if complete.

EVALUATION BOARD ALB E25

ALB E25 (**ABACO**® Link Bus Evaluable 25 I/O lines) is an evaluation board based on **µP25** where it is interfaced to a set of keys, LEDs and connectors that allow the user to control all the resources this device is provided with.
The keys allow to test the several input modalities of the 25 input lines on µP25, while the LEDs allow to test the several output modalities. All these signals are also available on two comfortable 20 pins low profile connectors feauring standard **ABACO**® I/O pin out, that allow the user to interface **µP25** lines to an eventual user application.
This connection may be made easier using **BLOCK** modules of **FBC** serie, that allow to untangle the several signals from flat cable to more comfortable quick release screw terminal connectors.
ALB E25 board is also provided with a double serial interface, **RS 232** and **RS 485**, that allows to test the **point-to-point** and **Master-Slave** communication modalities of **µP25**.
A wide range of demo programs and examples of employ allow to use immediatly the board; such programs are available both for PC and for all the several **GPC**® cards of **grifo**® listing.
Main features of **ALB E25** are:

- Evaluation board based on **µP25**.
- **25** keys to test **µP25** input signals modalities.
- **25** LEDs of different colours to test **µP25** output signals modalities.
- 2 low profile 20 pins connectors feauring standard **ABACO**® I/O pin out to interface the I/O signals
- **Jumper** to select RUN/SETUP modes of **µP25**.
- **LED** to visualize the status of /SETUP signal of **µP25**.
- Serial EEPROM sized **512 byte** (24c04) to store messages and configuration of **µP25**; possibility to install EEPROM up to **2K Byte**.
- Serial line interface in **RS 232** and **RS 485**, selectable through jumper.
- Unique power supply **+5 Vdc**.

TECHNICAL FEATURES OF ALB E25

GENERAL FEATURES OF ALB E25

CPU:	87c51 with firmware **µP25**
Number of I/O Lines:	24 in ports called 0, 1 and 2, interfaced to LEDs, keys and connectors
Counter Lines:	pin INT0, connected to LED, key and connector, can be used as 25th I/O signal
EEPROM:	512 bytes (24c04) expandible up to 2K bytes
Oscillator:	14.7456 MHz quartz
Serial Interfaces:	RS 232 or RS 485 selectable through jumper

PHYSICAL FEATURES OF ALB E25

Size:	155x72 mm
Weight:	92 g
Connectors:	CN1: 4 pins screw terminal vertical connector CN2: 20 pins low profile vertical male connector CN3: 20 pins low profile vertical male connector
Temperature Range:	From 0 to 70 centigrad degreeses
Relative Humidity:	20% up to 90% (without condensing)

ELECTRIC FEATURES OF ALB E25

Power Supply:	+5 Vdc
Current Consumption:	**30 mA** when all LEDs are OFF **60 mA** when all LEDs are ON

INSTALLATION

In this chapter there are the information for a right installation and correct use of **ALB E25** card. The user can find the location and functions of each connectors, LEDs, keys, and some explanatory diagrams.

CONNECTIONS

The board has three connectors that can be linkeded to other devices or directly to the field, according to system requirements. In this paragraph there are connectors pin outs, a short signals description (including the signals direction) and connectors location, plus some figures that describe how the interface signals are connected on the card. To easily locate the connectors please refer to figure 19.

CN1 - CONNECTOR FOR POWER SUPPLY AND SERIAL LINE

CN1 is a 4 pins screw terminal connector that allows to supply **ALB E25** and to perform the serial connection. The screw terminals allow to crimp in extreme safety all the wires with diameter lower than 3 mm and to connect comfortably to the external world.

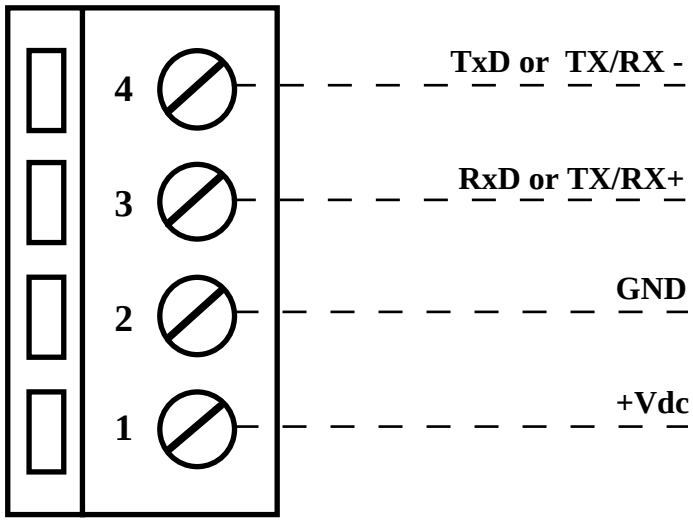


FIGURE 13: POWER SUPPLY AND SERIAL LINE CONNECTOR

Signals description:

GND	=		- Ground of power supply and serial line.
+Vdc	=	I	- Power supply +5 Vdc.
TxD	=	O	- RS 232 Transmit Data.
RxD	=	I	- RS 232 Receive Data.
TX/RX-	=	I/O	- RS 485 Negative Transmit/Receive differential line
TX/RX+	=	I/O	- RS 485 Positive Transmit/Receive differential line

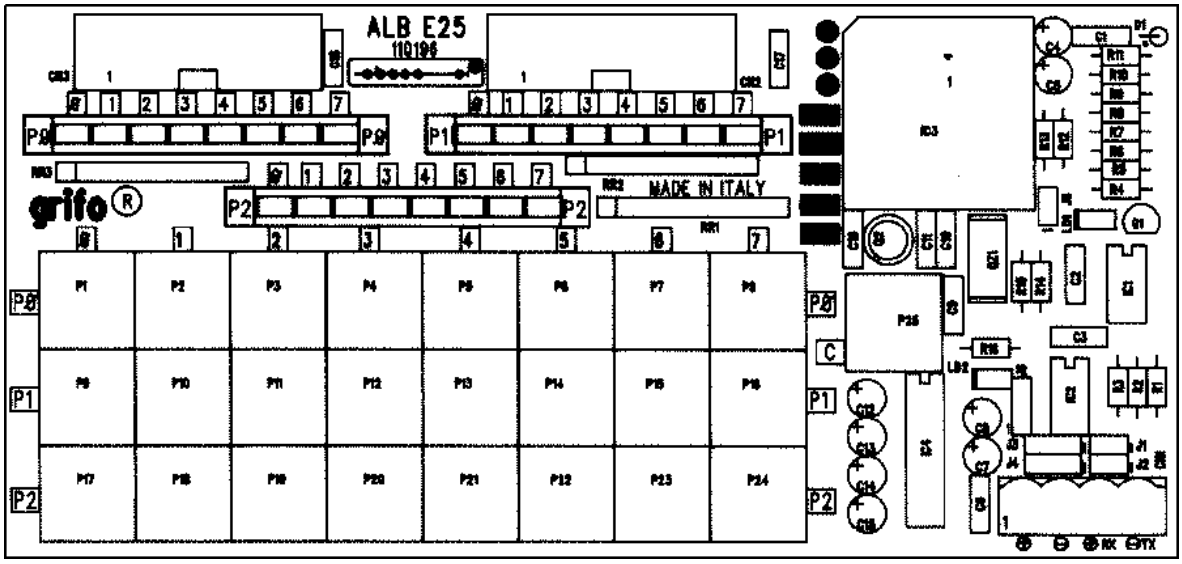


FIGURE 14: ALB E25 COMPONENTS MAP

FIGURE 15: ALB E25 CARD PHOTO

CN2 - CONNECTOR FOR I/O PORT 2 AND INT0

CN2 is a 20 pins low profile male verticalconnector with 2.54 mm pitch that allows to interface the I/O port 2 signals and the pin INT0 of **µP25** to the external world. It features standard pin out **ABACO®** I/O, the digital signals on this connector are TTL.

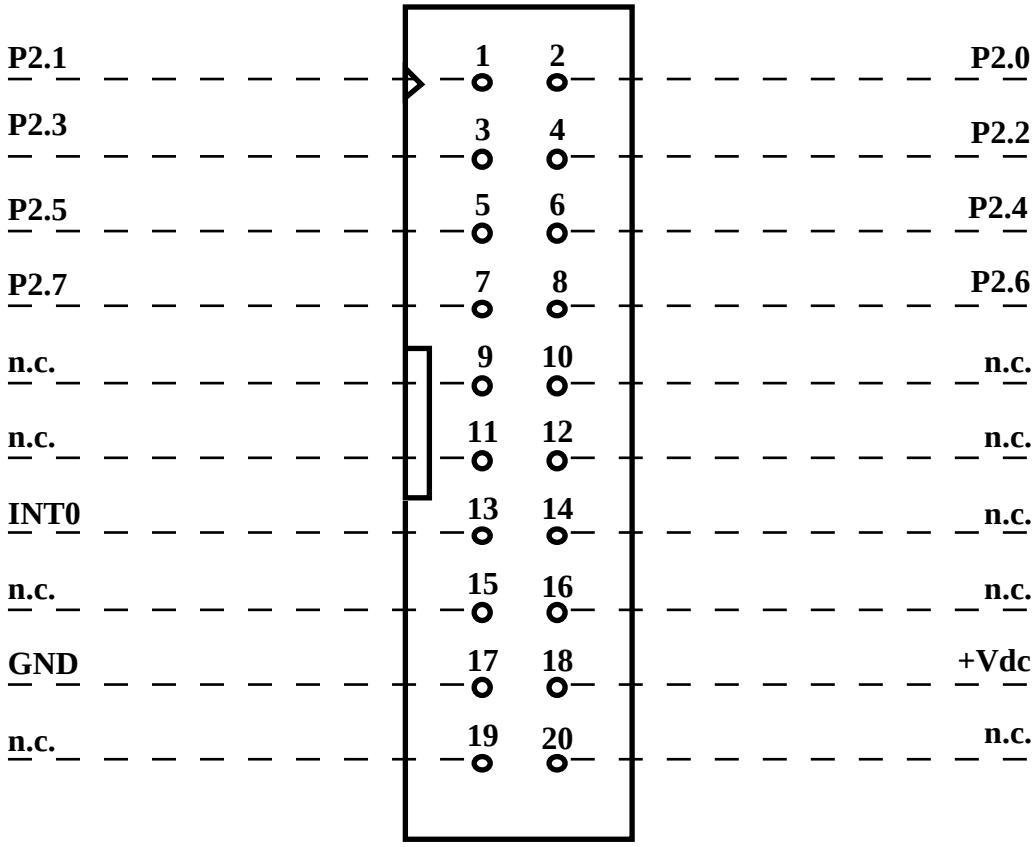


FIGURE 16: I/O PORT 2 AND INT0 CONNECTOR

Signals description:

GND	=		- Ground of power supply and serial line.
+Vdc	=	O	- Power supply +5 Vdc.
P2.n	=	I/O	- n-th digital signal of **µP25** port 2.
INT0	=	I/O	- Digital signal of **µP25** pin INT0.
n. c.	=	I	- No connection.

CN3 - CONNECTOR FOR I/O PORT 0 AND 1

CN3 is a 20 pins low profile male verticalconnector with 2.54 mm pitch that allows to interface the I/O ports 0 and 1 of **µP25** signals to the external world. It features standard pin out **ABACO®** I/O, the digital signals on this connector are TTL.

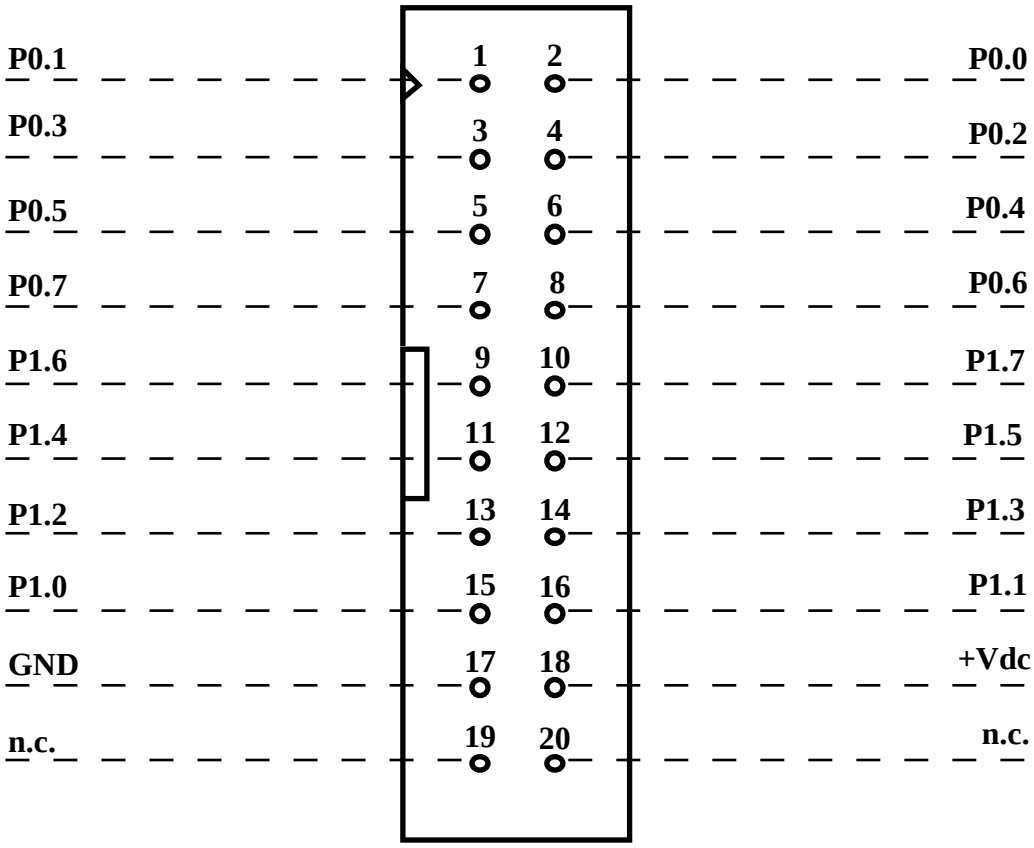


FIGURE 17: I/O PORT 0 AND 1 CONNECTOR

Signals description:

GND	=		- Ground of power supply and serial line.
+Vdc	=	O	- Power supply +5 Vdc.
P0.n	=	I/O	- n-th digital signal of **µP25** port 0.
P1.n	=	I/O	- n-th digital signal of **µP25** port 1.
n. c.	=	I	- No connection.

VISUAL SIGNALATIONS OF ALB E25

ALB E25 card is provided with 26 signalation LEDs to show several status informations, as described in the following table:

LEDs	COLOUR	PURPOSE
LD1	Red	Visualizes the logic status of signal /SETUP. When the LED is ON the signal is at logic status 0.
LD2	Red	Visualizes the logic status of signal INT0. When the LED is ON the signal is at logic status 0.
P0.0÷P0.7	Red	Visualize, respectively, the status of the 8 port 0 digital signals. When the LED is ON the signal is at logic status 0.
P1.0÷P1.7	Yellow	Visualize, respectively, the status of the 8 port 1 digital signals. When the LED is ON the signal is at logic status 0.
P2.0÷P2.7	Green	Visualize, respectively, the status of the 8 port 2 digital signals. When the LED is ON the signal is at logic status 0.

FIGURE 18: VISUAL SIGNALATIONS TABLE

The main purpose of LEDs is to show a visual indication about the card's status, making so easier debug and verify operations. All the LEDs are in the front of the board, near connector CN1. To easily locate these visual signalations please refer to figure 19.

KEYS OF ALB E25

ALB E25 board is provided with 25 keys by which it is possible to simulate the status changes of the **µP25** signals configured as inputs. Each key corresponds to a normally open contact that connects to the ground (logical level **0**) the line to which is connected when it is closed.
In detail:

Key Pressed	->	0V	=	Logic level **0**
Key Not Pressed	->	+5 Vdc	=	Logic level **1**

Here follows a description of the several keys; to easily locate them please refer to figure 20.

P1÷P8	->	They are connected, respectively, to signals P0.0÷P0.7 of **µP25**.
P9÷P16	->	They are connected, respectively, to signals P1.0÷P1.7 of **µP25**.
P17÷P24	->	They are connected, respectively, to signals P2.0÷P2.7 of **µP25**.
P25	->	IT is connected to signal INT0 of **µP25**.

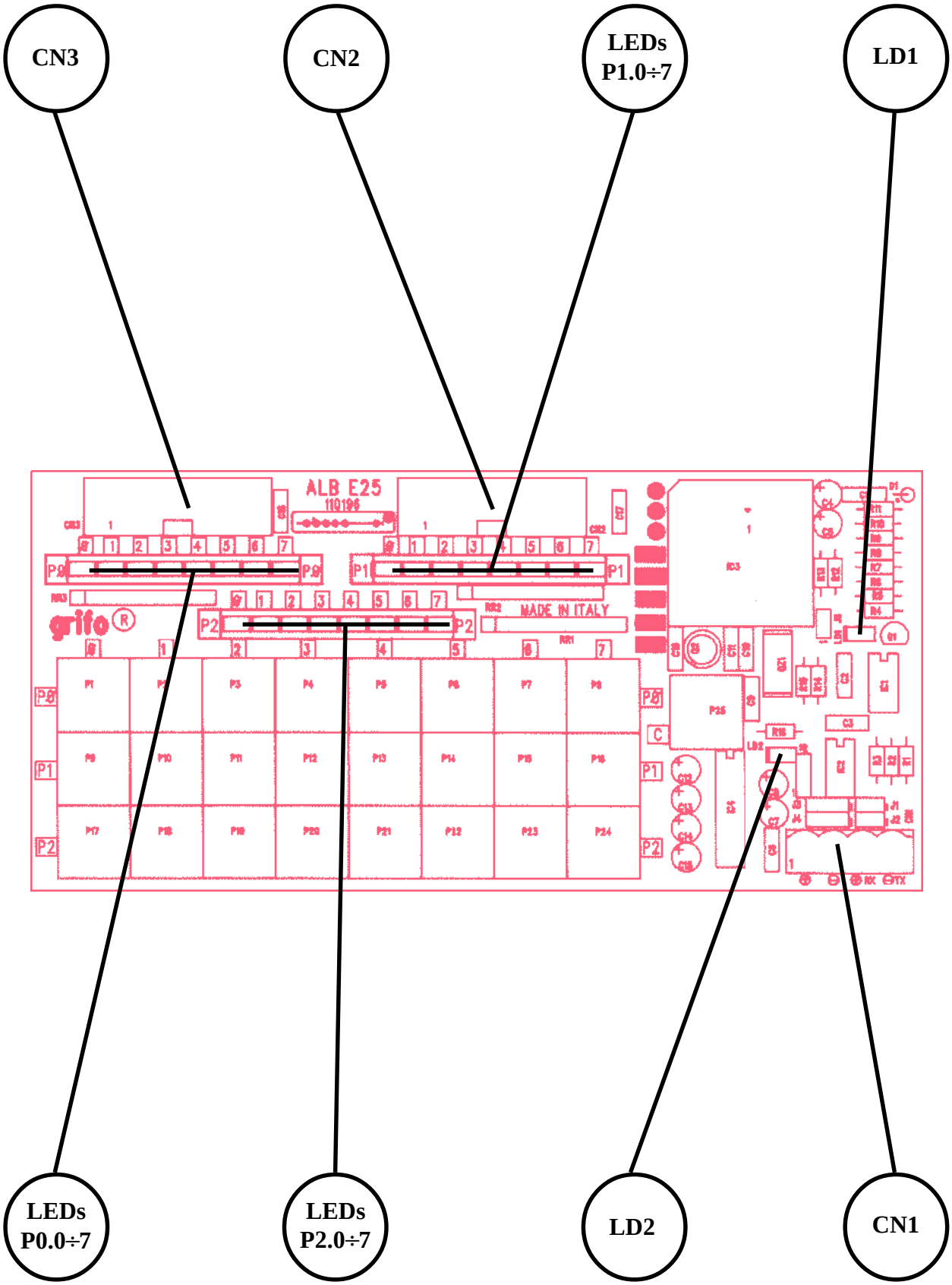


FIGURE 19: CONNECTORS AND LEDS LOCATION

JUMPERS

On **ALB E25** board there are 6 jumpers for card configuration. Below there is the jumpers list, location and function.

JUMPERS	N. PINS	PURPOSE
J1	2	Matched with jumper J2, it connects the termination circuitery for RS 485 serial line.
J2	2	Matched with jumper J1, it connects the termination circuitery for RS 485 serial line.
J3	3	Matched with jumpers J4 and J6, it allows to selects between RS 232 and RS 485 serial communication protocol.
J4	3	Matched with jumpers J3 and J6, it allows to selects between RS 232 and RS 485 serial communication protocol.
J5	2	Selects between RUN or SETUP working modality.
J6	3	Matched with jumpers J3 and J4, it allows to selects between RS 232 and RS 485 serial communication protocol.

FIGURE 20: JUMPERS SUMMARIZING TABLE

The following tables describe all the right connections of **ALB E25** jumpers with their relative functions. To recognize these valid connections, please refer to the board printed diagram (serigraph) or to figure 14 of this manual, where the pins numeration is listed; for recognizing jumpers location, please refer to figure 22.

2 PINS JUMPERS

JUMPER	CONNECTION	PURPOSE	DEF.
J1	Not connected	Matched with jumper J2, it does not connect the termination circuitery for RS 485.	
	Connected	Matched with jumper J2, it connects the termination circuitery for RS 485.	*
J2	Not connected	Matched with jumper J1, it does not connect the termination circuitery for RS 485.	
	Connected	Matched with jumper J1, it connects the termination circuitery for RS 485.	*
J5	Not connected	Selects the µ**P25** RUN working modality.	*
	Connected	SSelects the µ**P25** SETUP working modality.	

FIGURE 21: 2 PINS JUMPERS TABLE

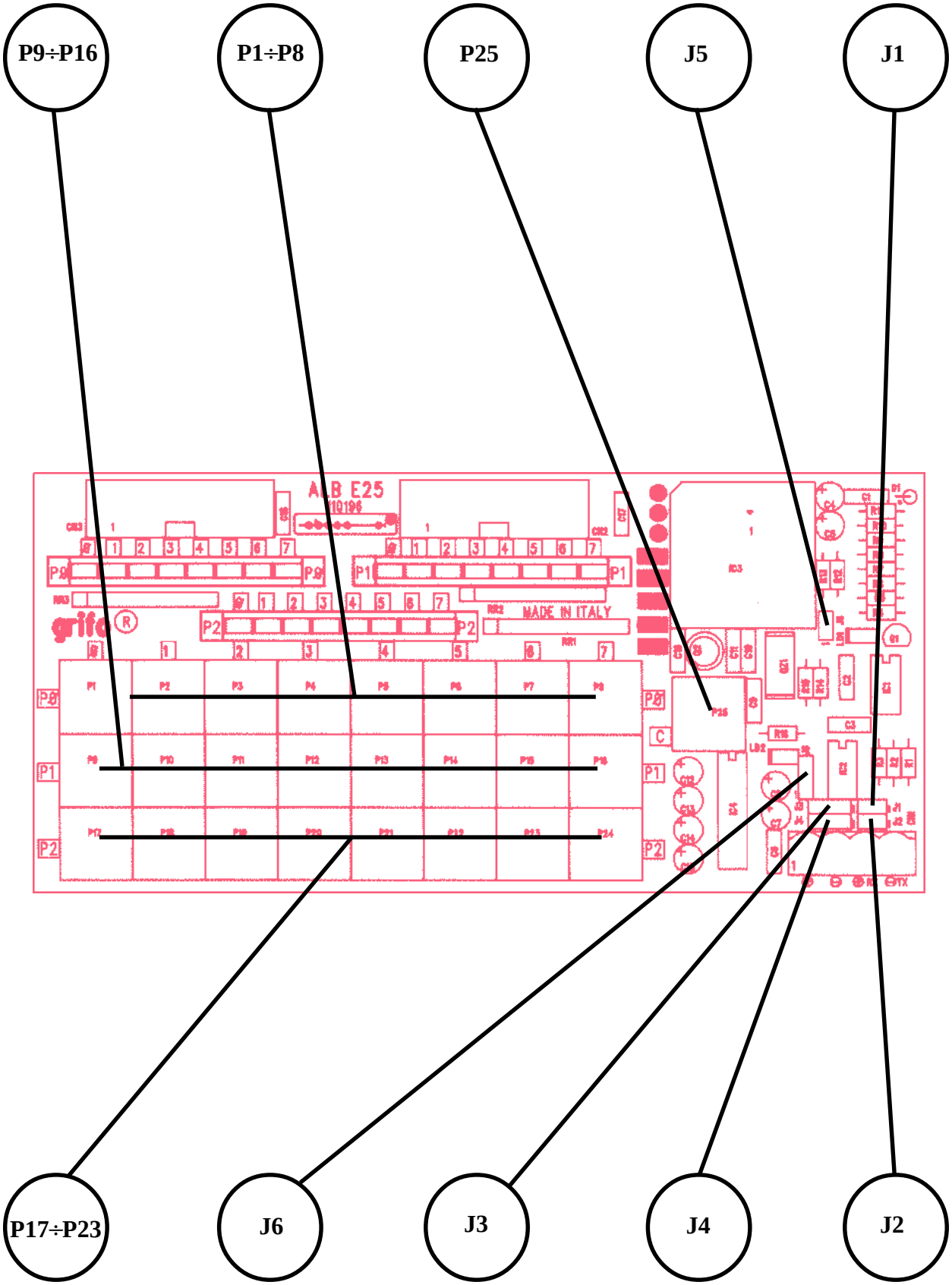


FIGURE 22: KEYS AND JUMPERS LOCATION

3 PINS JUMPERS

JUMPER	CONNECTION	PURPOSE	DEF.
J3	position 1-2	Matching with jumpers J4 and J6, it selects the RS 485 serial communication.	
	position 2-3	Matching with jumpers J4 and J6, it selects the RS 232 serial communication.	*
J4	position 1-2	Matching with jumpers J3 and J6, it selects the RS 485 serial communication.	
	position 2-3	Matching with jumpers J3 and J6, it selects the RS 232 serial communication.	*
J6	position 1-2	Matching with jumpers J3 and J4, it selects the RS 485 serial communication.	
	position 2-3	Matching with jumpers J3 and J4, it selects the RS 232 serial communication.	*

FIGURE 23: 3 PINS JUMPERS TABLE

The "*" used in the following tables, 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.

BOARD CONNNECTIONS

To prevent possible connecting problems between **ALB E25** board and the external systems, the user has to read carefully the information of the previous paragraphs and must follow these instructions:

- The TTL signals can be connected directly only to a device featuring the same type of interface. About the correspondance between logic signals and TTL output status, remember that a logic **0** generates a TTL 0 Vdc, while a logic **1** generates a TTL +5 Vdc.

POWER SUPPLY OF ALB E25

ALB E25 needs an unique tension for supply:

+5 Vdc: Supplies the on board logic; must be in the range +5 Vdc ± 5% and must be provided through the specific pins of connector CN1.

SERIAL COMMUNICATION SELECTION FOR ALB E25

ALB E25 board is provided with a serial communication line that can be buffered as **RS 232** or as **RS 485**. The selection between the two interfacing types is performed by hardware through an opportune jumpers configuration, as can be seen in the previous tables.
In detail:

J3, J4 and J6 in position 1-2	->	*Serial line configured in RS 485*
J3, J4 and J6 in position 2-3	->	*Serial line configured in RS 232*

NOTE
All the other positions of the jumpers are not valid and cause electric conflicts that may damage the board itself or damage the drivers of other devices connected to the board.

In case the RS 485 serial line is used, jumpers J1 and J2 allow to connect the termination circuitery of the differential transmission/reception line. Such circuitery is made with the forcing resistors (**R1=R3=3.3 KΩ**) and the line load (**R2=120 Ω**). In detail:

J1 and J2 connected	->	*Termination circuitery connected*
J1 and J2 not connected	->	*Termination circuitery not connected*

EXPERIMENTAL BOARD ALB S25

ALB S25 (**ABACO®** Link Bus Sperimentale 25 I/O lines) is an experimental board based on **µP25** provided with a wide experimental area where the user can develop an own hardware that can be interfaced to **µP25** I/O signals, to specialize the board according to the user application specifics. The experimental area is also provided with some features that easy the prototyping (power supply in several points, larger holers, etc.)
Another important feature of **ALB S25** is the possibility to cut the experimental area obtaining a very small module that can be installed as a *stack-through* module on a back plane developed by the user containing all the hardware for field interfacing simply using two pass-through strips.
Also, ALB S25 board is vided with a double serial interface, **RS 232** and **RS 485**, that allows to test the **point-to-point** and **Master-Slave** communication modalities of **µP25**.
A wide range of demo programs and examples of employ allow to use immediatly the board; such programs are available both for PC and for all the several **GPC®** cards of **grifo®** listing.
Main features of **ALB S25** are:

- Evaluation board based on **µP25**.
- **Prototyping area** where the user can develop an own hardware.
- Possibility to remove the prototyping area and to install the module obtained as stack-through on an user-developed back plane.
- **Jumper** to select RUN/SETUP modes of **µP25**.
- **LED** to visualize the status of /SETUP signal of **µP25**.
- Serial EEPROM sized **512 byte** (24c04) to store messages and configuration of **µP25**; possibility to install EEPROM up to **2K Byte**.
- Serial line interface in **RS 232** and **RS 485**, selectable through jumper.
- Unique power supply **+5 Vdc**.

TECHNICAL FEATURES OF ALB S25

GENERAL FEATURES OF ALB S25

CPU:	87c51 with firmware **µP25**
Number of I/O Lines:	24 in ports called 0, 1 and 2
Counter Lines:	pin INT0, connected to LED, key and connector, can be used as 25th I/O signal
EEPROM:	512 bytes (24c04) expandible up to 2K bytes
Oscillator:	14.7456 MHz quartz
Serial Interfaces:	RS 232 or RS 485 selectable through jumper

PHYSICAL FEATURES OF ALB S25

Size:	155x72 mm with experimental area 45x72 mm without experimental area
Weight:	55 g with experimental area
Connectors:	CN1: 4 pins screw terminal vertical connector CN2: 28 pins designed to install pass-through strips
Temperature Range:	From 0 to 70 centigrad degreeses
Relative Humidity:	20% up to 90% (without condensing)

ELECTRIC FEATURES OF ALB S25

Power Supply:	+5 Vdc
Current Consumption:	**35 mA**

INSTALLATION

In this chapter there are the information for a right installation and correct use of **ALB S25** card. The user can find the location and functions of each connectors, LED and some explanatory diagrams.

CONNECTIONS

The board has three connectors that can be linkeded to other devices or directly to the field, according to system requirements. In this paragraph there are connectors pin outs, a short signals description (including the signals direction) and connectors location, plus some figures that describe how the interface signals are connected on the card. To easily locate the connectors please refer to figure 28.

CN1 - CONNECTOR FOR POWER SUPPLY AND SERIAL LINE

CN1 is a 4 pins screw terminal connector that allows to supply **ALB S25** and to perform the serial connection. The screw terminals allow to crimp in extreme safety all the wires with diameter lower than 3 mm and to connect comfortably to the external world.

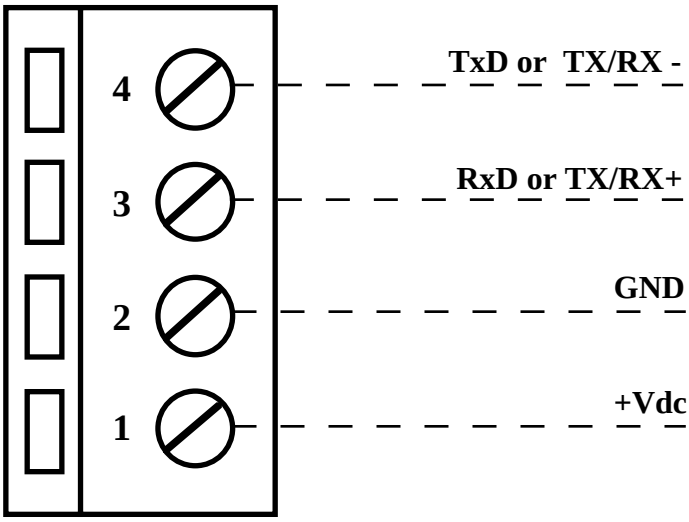


FIGURE 24: POWER SUPPLY AND SERIAL LINE CONNECTOR

Signals description:

GND	=		- Ground of power supply and serial line.
+Vdc	=	I	- Power supply +5 Vdc.
TxD	=	O	- RS 232 Transmit Data.
RxD	=	I	- RS 232 Receive Data.
TX/RX-	=	I/O	- RS 485 Negative Transmit/Receive differential line
TX/RX+	=	I/O	- RS 485 Positive Transmit/Receive differential line

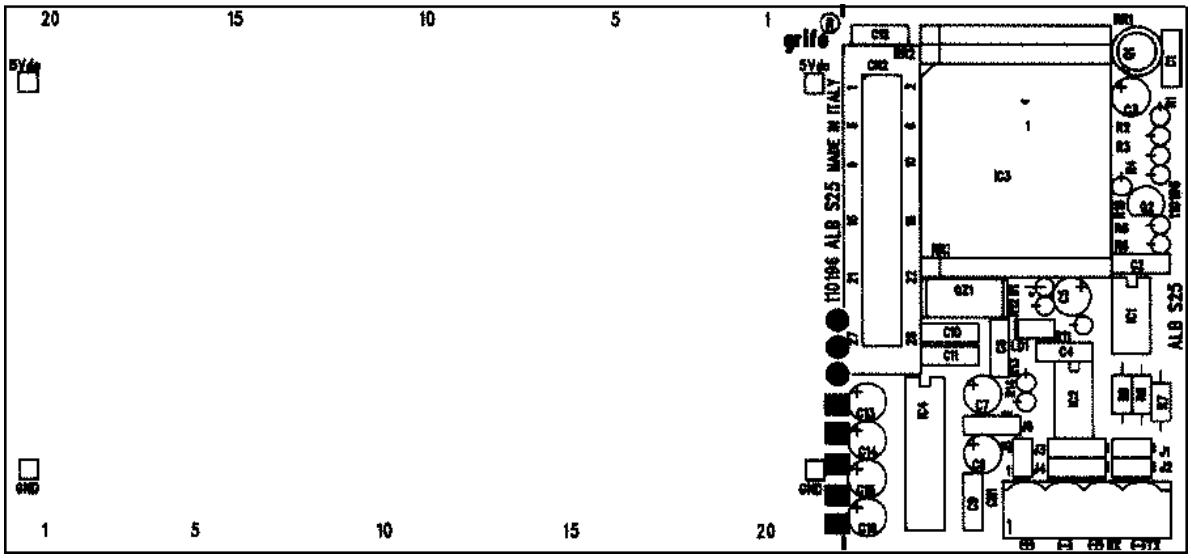


FIGURE 25: ALB S25 COMPONENTS MAP

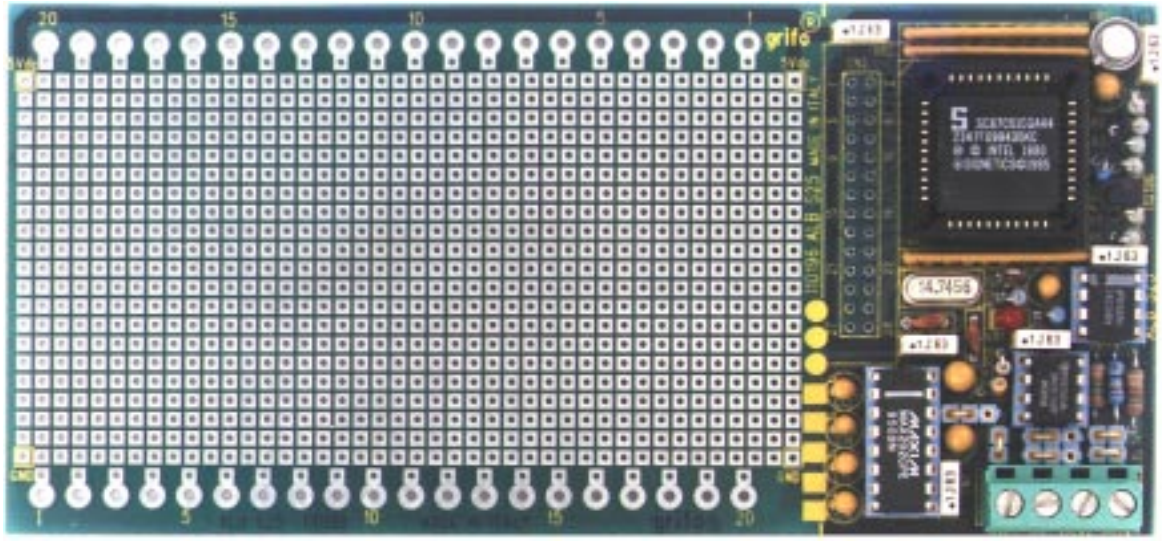


FIGURE 26: ALB S25 CARD PHOTO

CN2 - CONNECTOR FOR I/O PORT AND INT0

CN2 is a 28 pins connector with 2.54 mm pitch that allows to interface the I/O ports and INT0 signals to external hardware. The digital signals available on this connector are TTL; it is possible to install on CN2 two pass-through strips each with 14 pins for the *stack-through* installation of **ALB S25** without the experimental area.

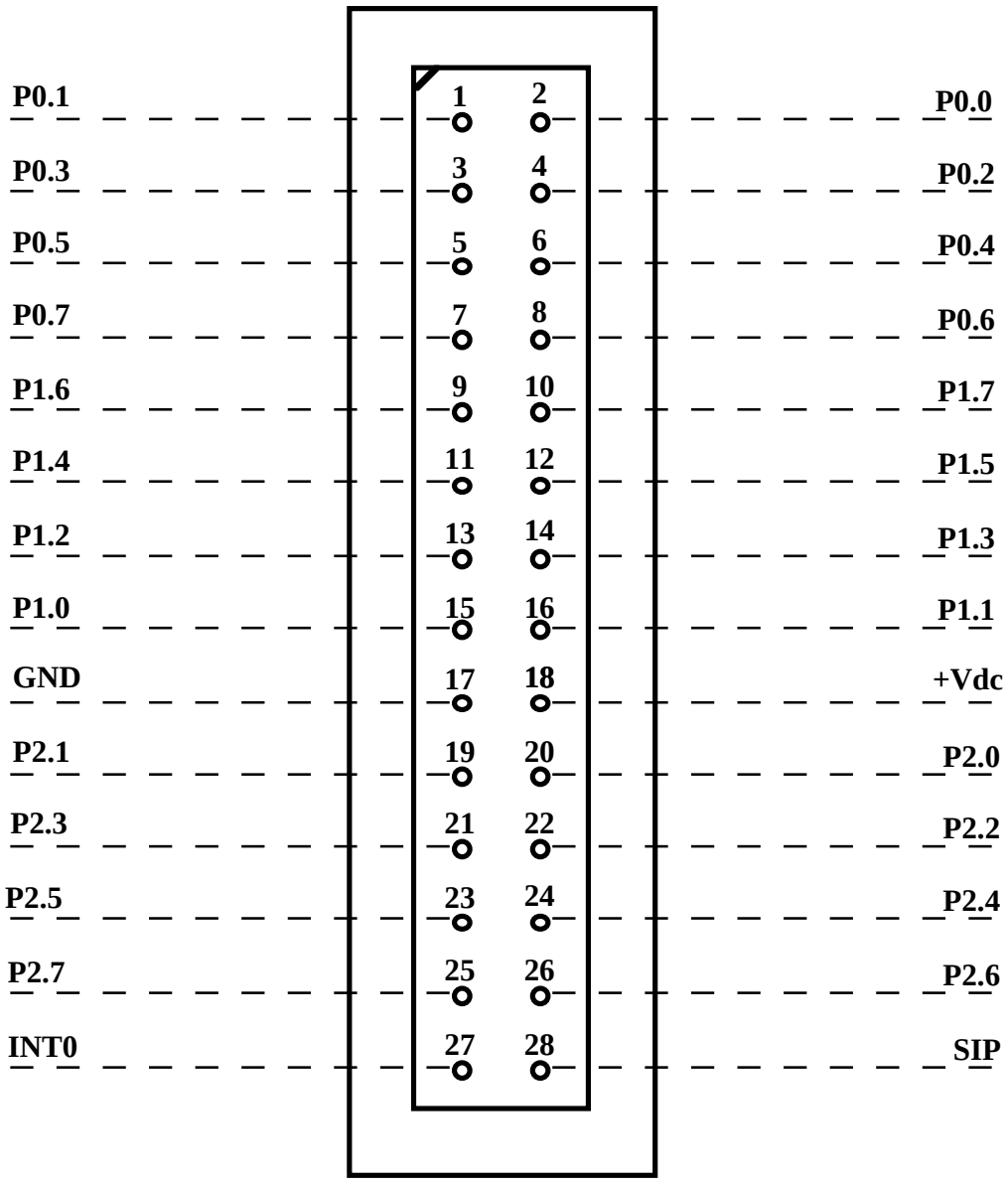


FIGURE 27: I/O PORT AND PIN INT0 CONNECTOR

Signals description:

GND	=		- Ground of power supply and serial line.
+Vdc	=	I/O	- Power supply +5 Vdc.
P0.n	=	I/O	- N-th digital line of µ**P25** port 0.
P1.n	=	I/O	- N-th digital line of µ**P25** port 1.
P2.n	=	I/O	- N-th digital line of µ**P25** port 2.
INT0	=	I/O	- Digital line of µ**P25** pin INT0.
SIP	=	O	- External hardware resistor network power supply.

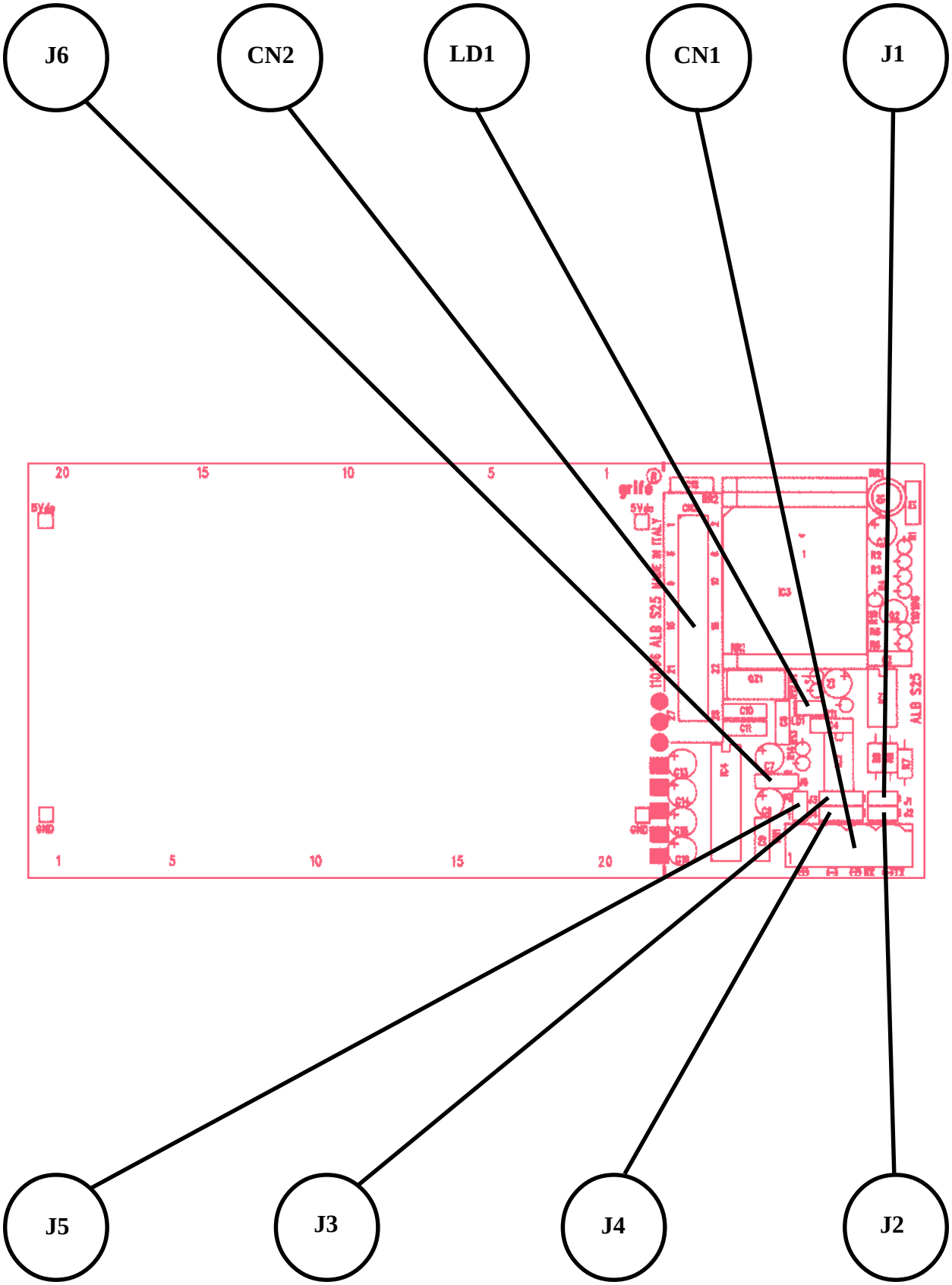


FIGURE 28: JUMPERS, LED AND CONNECTORS LOCATION

JUMPERS

On **ALB S25** board there are 6 jumpers for card configuration. Below there is the jumpers list, location and function.

JUMPERS	N. PINS	PURPOSE
J1	2	Matched with jumper J2, it connects the termination circuitery for RS 485 serial line.
J2	2	Matched with jumper J1, it connects the termination circuitery for RS 485 serial line.
J3	3	Matched with jumpers J4 and J6, it allows to selects between RS 232 and RS 485 serial communication protocol.
J4	3	Matched with jumpers J3 and J6, it allows to selects between RS 232 and RS 485 serial communication protocol.
J5	2	Selects between RUN or SETUP working modality.
J6	3	Matched with jumpers J3 and J4, it allows to selects between RS 232 and RS 485 serial communication protocol.

FIGURE 29: JUMPERS SUMMARIZING TABLE

The following tables describe all the right connections of **ALB S25** jumpers with their relative functions. To recognize these valid connections, please refer to the board printed diagram (serigraph) or to figure 25 of this manual, where the pins numeration is listed; for recognizing jumpers location, please refer to figure 28.

2 PINS JUMPERS

JUMPER	CONNECTION	PURPOSE	DEF.
J1	Not connected	Matched with jumper J2, it does not connect the termination circuitery for RS 485.	
	Connected	Matched with jumper J2, it connects the termination circuitery for RS 485.	*
J2	Not connected	Matched with jumper J1, it does not connect the termination circuitery for RS 485.	
	Connected	Matched with jumper J1, it connects the termination circuitery for RS 485.	*
J5	Not connected	Selects the µ**P25** RUN working modality.	*
	Connected	SSelects the µ**P25** SETUP working modality.	

FIGURE 30: 2 PINS JUMPERS TABLE

3 PINS JUMPERS

JUMPER	CONNECTION	PURPOSE	DEF.
J3	position 1-2	Matching with jumpers J4 and J6, it selects the RS 485 serial communication.	
	position 2-3	Matching with jumpers J4 and J6, it selects the RS 232 serial communication.	*
J4	position 1-2	Matching with jumpers J3 and J6, it selects the RS 485 serial communication.	
	position 2-3	Matching with jumpers J3 and J6, it selects the RS 232 serial communication.	*
J6	position 1-2	Matching with jumpers J3 and J4, it selects the RS 485 serial communication.	
	position 2-3	Matching with jumpers J3 and J4, it selects the RS 232 serial communication.	*

FIGURE 31: 3 PINS JUMPERS TABLE

The "*" used in the following tables, 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.

BOARD CONNNECTIONS

To prevent possible connecting problems between **ALB S25** board and the external systems, the user has to read carefully the information of the previous paragraphs and he must follow these instructions:

- The TTL signals can be connected directly only to a device featuring the same type of interface. About the correspondance between logic signals and TTL output status, remember that a logic **0** generates a TTL 0 Vdc, while a logic **1** generates a TTL +5 Vdc.

VISUAL SIGNALATIONS OF ALB S25

ALB S25 card is provided with one signalation LED, called LD1 and coloured in red, to show the logical status of /SETUP signal. In detail the /SETUP signal has logic level 0 whenever LD1 is ON. To easily locate the LED please refer to figure 28.

POWER SUPPLY OF ALB S25

ALB S25 needs an unique tension for supply:

+5 Vdc: Supplies the on board logic; must be in the range +5 Vdc ± 5% and must be provided through the specific pins of connector CN1.

SERIAL COMMUNICATION SELECTION FOR ALB S25

ALB S25 board is provided with a serial communication line that can be buffered as **RS 232** or as **RS 485**. The selection between the two interfacing types is performed by hardware through an opportune jumpers configuration, as can be seen in the previous tables.
In detail:

J3, J4 and J6 in position 1-2	->	*Serial line configured in RS 485*
J3, J4 and J6 in position 2-3	->	*Serial line configured in RS 232*

NOTE
All the other positions of the jumpers are not valid and cause electric conflicts that may damage the board itself or damage the drivers of other devices connected to the board.

In case the RS 485 serial line is used, jumpers J1 and J2 allow to connect the termination circuitery of the differential transmission/reception line. Such circuitery is made with the forcing resistors (**R7=R9=3.3 KΩ**) and the line load (**R8=120 Ω**). In detail:

J1 and J2 connected	->	*Termination circuitery connected*
J1 and J2 not connected	->	*Termination circuitery not connected*

EXTERNAL CARDS

ALB E25 and S25 can be connected to a wide range of **grifo®** boards,both intelligent and not, thanks to their standard serial line and through their **ABACO®** I/O compatible pin out..
Hereunder some of these cards are briefly described; ask the detailed information directly to **grifo®**, if required.

OBI 01 - OBI 02
Opto BLOCK Input NPN-PNP
Interface between 16 NPN, PNP optocoupled and displayed input lines, with screw terminal and **ABACO®** standard I/O 20 pins connector; power supply section; connection for DIN Ω rails.

OBI N8 - OBI P8
Opto BLOCK Input NPN-PNP
Interface between 8 NPN, PNP optocoupled and displayed input lines, with screw terminal and **ABACO®** standard I/O 20 pins connector; power supply section; connection for DIN Ω rails.

TBO 01 - TBO 08
Transistor BLOCK Output
Interface for **ABACO®** standard I/O 20 pins connector; 16 or 8 transistor output lines 45 Vdc 3 A open collector; screw terminal; optocoupled and displayed lines; connection for DIN 247277-1 and 3 rails.

RBO 01
Relé BLOCK Output
Interface for **ABACO®** standard I/O 20 pins connector; 8 displayed 5A or 10A relays; screw terminal; connection for DIN Ω rails.

RBO 08 - RBO 16
Relé BLOCK Output
Interface for **ABACO®** standard I/O 20 pins connector; 8 or 16 displayed Relays 3A with MOV; screw terminal; connection for DIN Ctype and Ω rails.

XBI 01
miXed BLOCK Input Output
Interface for **ABACO®** standard I/O 20 pins connector; 8 transistor output lines 45 Vdc 3A; 8 input lines; screw terminal; optocoupled and displayed I/O lines; connection for DIN 247277-1 and 3 rails.

XBI R4 - XBI T4
miXed BLOCK Input-Output
Interface for **ABACO®** standard I/O 20 pins connector; 4 Relays 3A with MOV or 4 optocoupled Transistors 3A open collectors; 4 input lines optocoupled; screw terminal; connection for DIN Ctype and Ω rails.

SBP 02
Switching Block Power 2.5 A
BLOCK format, **120x40x60 mm**,with plastic container for omega rails DIN 247277-1 e 247277-3; two comfortable screw terminal, quick release, 2 pins connectors; power LED; **w**ide **heat sinks**; protection on over temperature, over output voltage and short circuit; medium performance of **90%**; several models with input voltages from 10 to 46 Vdc and output voltages from 5 to 24 Vdc, 2.5 A.

SBP 05

Switching Block Power 5 A

BLOCK format, **120x110x100 mm**,with plastic container for omega rails DIN 247277-1 e 247277-3; two comfortable screw terminal, quick release, 4 and 5 pins connectors; power LED; Alarm LED; **w**ide **heat sinks**; protection on over temperature, over output voltage and short circuit; medium performance of **80%**; auxiliary input for **UPS** functions; several models with input voltages from 10 to 46 Vdc and output voltages from 5 to 24 Vdc, 5 A.

GPC® 51

General Purpose Controller fam. 51

Microprocessor family 51 INTEL including the masked BASIC chip; the board features: 16 I/O TTL lines; dip switch; 3 timer/counter; RS 232; 4 A/D converter signals resolution 11 bit; buzzer; on board EPROM programmer; RTC and 32K SRAM with Lithium battery back up; controlloer for display and keyboard.

GPC® 188F

General Purpose Controller 80C188

80C188 µP 20MHz; 1 RS 232 line; 1 RS 232, RS 422-485 or Current Loop line; 24 TTL I/O lines; 1M EPROM or 512K FLASH; 1M RAM Lithium battery backed; 8K serial EEPROM; RTC; Watch Dog; 8 Dip switch; 3 Timer Counter; 8 13 bit A/D lines; Power failure; activity LEDs; single power supply +5Vdc.

GPC® 15A

General Purpose Controller 84C15

Full CMOS card, 10÷20 MHz 84C15 CPU; 512K EPROM or FLASH; 128K RAM; 8K RAM and RTC backed; 8K serial EEPROM; 1 RS 232 line; 1 RS 232 line or RS 422-485 or Current Loop line; 32 or 40 TTL I/O lines; CTC; Watch dog; 2 Dip switches; Buzzer.

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 serail FLASH ; 1 serial line RS 232 plus 1 RS 232 or RS 422-485 or current loop; 40 I/O TTL; 2 timer/counter; 2 watch dog; dip switch; EEPROM; A/D converter with resolution 12 bit; activity LED.

GPC® 15R

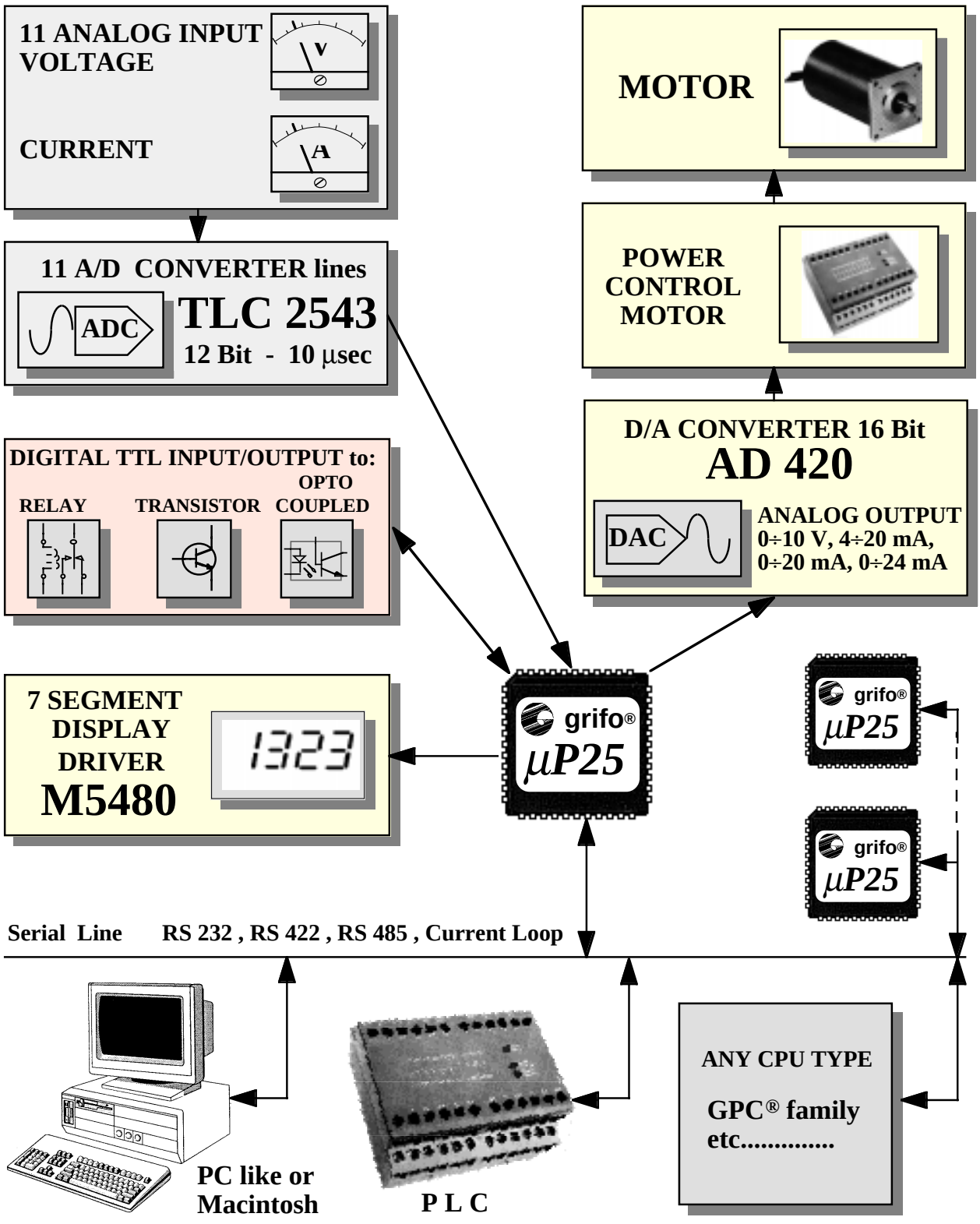
General Purpose Controller 84C15

84C15 µP, 10÷16 MHz; 1 RS 232 line; 1 RS 232 or RS 422-485 or C. L. line; 16÷24 TTL I/O lines; 16 Opto-in; 8 Relays; 4 Opto Coupled Timers Counters; 512K EPROM or FLASH; 512K RAM and RTC backed; 8K serial EEPROM; 8K Backed RAM modul; Buzzer; 1 Activity LED; Watch dog; 4÷12 readable DIPs; LCD Interface.

GPC® 323

General Purpose Controller 51 family

80C32 µP, 14 MHz; Full CMOS; 1 RS 232 line (software); 1 RS 232 or RS 422-485 or Current Loop line; 24 TTL I/O lines; 11 A/D 12 bits lines; 3 Timers Counters; 64K EPROM; 64K RAM; 32K RAM and RTC backed; 32K DIL EEPROM; 8K serial EEPROM; Buzzer; 2 Activity LED; Watch dog; 5 readable DIPs; LCD Interface.



µP25 Interconnections Blocks Diagram

FIGURE 32: POSSIBLE CONNECTIONS DIAGRAM

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; 64K EPROM; 64K RAM; 32K RAM and RTC backed; 32K DIL EEPROM; 8K serial EEPROM; 2 PWM lines; 1 Activity LED; Watch dog; 5 readable DIPs; LCD Interface.

GPC® 153

General Purpose Controller Z80

84C15 µP, 10÷16 MHz; Full CMOS; 1 RS 232 line; 1 RS 232 or RS 422-485 or Current Loop line; 16 TTL I/O lines; 8 A/D 12 bits lines; 2÷4 Timers Counters; 512K EPROM or FLASH; 512K RAM and RTC backed; 8K serial EEPROM; Buzzer; 1 Activity LED; Watch dog; 8 readable DIPs; LCD Interface.

GPC® 183

General Purpose Controller Z180

Z180 µP, 10÷16 MHz; Full CMOS; 1 RS 232 line; 1 RS 232 or RS 422-485 or Current Loop line; 24 TTL I/O lines; 11 A/D 12 bits lines; 2 Timers Counters; 512K EPROM or FLASH; 512K RAM and RTC backed; 8K serial EEPROM; Buzzer; 2 Activity LED; Watch dog; 4 readable DIPs; LCD Interface.

GPC® 324/D

"4" Type General Purpose Controller 80C32/320

80C32 or 80C320 µP, 14÷22 MHz; Full CMOS; 1 RS 232 line; 1 RS 232 or RS 422-485 or Current Loop line; 4÷16 TTL I/O lines; 3 Timers Counters; 64K EPROM; 64K RAM; 32K RAM backed; 32K DIL E2; 8K serial EEPROM; Watch dog; 1 readable DIP; LCD Interface; Abaco® I/O BUS; 5Vdc Power supply; Size: 100x50 mm.

GPC® 554

General Purpose Controller 80C552

Microprocessor 80C552 at 22 MHz; implementation completely CMOS; 32K EPROM; 32 K SRAM; 32 K EEPROM or SRAM; EEPROM; 2 RS 232 serial lines; 16 I/O TTL; 2 PWM lines; 16 bits Timer/Counter; Watch Dog; 6 signals A/D converter with resolution 10 bit; interface for **ABACO®** I/O BUS.

GPC® 154

"4" Type General Purpose Controller Z80

84C15 µP, 10÷16 MHz; Full CMOS; 1 RS 232 line; 1 RS 232 or RS 422-485 line; 16 TTL I/O lines; 2÷4 Timers Counters; 512K EPROM or FLASH; 512K RAM and RTC backed; 8K serial EEPROM; Watch dog; 2 readable DIPs; LCD Interface; Abaco® I/O BUS; 5Vdc Power supply; Size: 100x50 mm.

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 signals A/D converter with 12 bit resolution; interface for **ABACO®** I/O BUS.

GPC® 114

General Purpose Controller 68HC11

Microprocessor 68HC11A1 at 8 MHz; implementation completely CMOS; serie 4 format; 32K EPROM; 32K SRAM backed with Lithium battery; 32K EPROM, SRAM, EEPROM; RTC; 1 serial line RS 232 or RS 422-485; 10 I/O TTL; 3 timer/counter; watch dog; 8 signals A/D converter with resolution 8 bit; 1 asunchronous serial line; extremly low power consumption; interface for **ABACO®** I/O BUS.

GPC® 184

General Purpose Controller Z80195

Microprocessor Z80195 at 22 MHz; implementation completely CMOS; 512K EPROM or FLASH; 512K RAM; Back-Up with Lithium battery internal or external; 1 serial line RS 232 + 1 RS 232 or RS 422-485 or current loop + 1 TTL; 18 I/O TTL; 4 timer/counter 8 bits; 2 timer 16 bits; Watch Dog; Real Time Clock; activity LED; EEPROM; interface for **ABACO®** I/O BUS.

GPC® AM4

General Purpose Controller ATmega103

Microprocessor ATmega103 at 5.5 MHz; implementation completely CMOS; 128K internal FLASH; 32K SRAM; Back-Up with Lithium battery internal or external; 1 serial line RS 232 or RS 422-485 or current loop; 16 I/O TTL; 8 linee A/D resolution 10 bits; 2 timer/counter; Watch Dog; Real Time Clock; 4K internal EEPROM; interface for ISP programming; interface forr **ABACO®** I/O BUS.

BIBLIOGRAPHY

Here follows a list of manuls thst can be a source of further informations about the devices installed on **ALB E25** and **ALB S25**.

Manual PHILIPS: *8051-based 8-bit Microcontrollers*

Manual MAXIM: *Data Book - Volume III*

Manual TEXAS INSTRUMENTS: *RS-422 and RS-485 Interface Circuits*

Manual HEWLETT-PACKARD: *Optoelectronics Designer's Catalog*

Manual XICOR: *Data Book*

Manual SGS-THOMSON: *Industrial and Computer Peripheral ICs*

Manual NATIONAL SEMICONDUCTOR: *Interface Data Book*

Manual ANALOG DEVICES: *Design-In Reference Manual - G1944-200-8/94*

Tecnical note TEXAS INSTRUMENTS: *TLC2543 12-bit Analog-to Digital Converter*

Application Report TEXAS INSTRUMENTS: *Microcontroller based Data Acquisition using the TLC2543 12-bit serial-out ADC*

Please connect to the manifactures Web sites to get the latest version of all manuals and data sheets.

APPENDIX A: ALPHABETICAL INDEX

SYMBOLS

A

B

C

D

E

F

G

I

J

K

L

M

N

O

P

Q

R

S

T

V

W