User's manual

Serial Communication Protocol

General information

This guide is designed to provide the most complete and exhaustive information on the serial communication protocol which has been implemented in the most advanced units of Lika Electronic's Posicontrol series. LECOM protocol (DIN ISO 1745) is a common standard for drive applications. It is intended for data exchange between one or more bus devices (Slave) and a host device (Master) in both RS-232 and RS- 485 serial communication interfaces. Information on setting the desired baud rate and data format as well as the pin-out of the data connector are clearly described in the specific "User's manual" of the unit you have to refer to.

Table of contents

1 – Unit address

This protocol supports unit addresses comprised between 11 and 99. They can be set either via keypad / DIL switch or, as an alternative, via serial setup using a personal computer / PLC. The unit addresses will be saved in the EEProm of the unit. The addresses must NOT contain any "0" because such numbers are reserved for collective addressing for several units. The general address "00" is intended for communication with all the units simultaneously linked to the network. Addresses such as "10" or "20" will communicate with all the units from 11 to 19 and from 21 to 29 respectively; and so on.
If the serial address of the unit should be unknown, you can run the SCAN function from the TOOLS menu of the operator software to find it out.
Ex factory, all the units are set to the default address "11".
Please note that any response delivered by a unit will be suppressed after access by the general address "00" or a collective address such as "20".

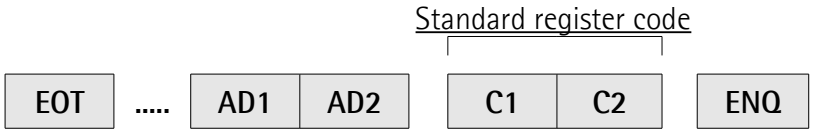
2 – Serial Access Codes

They are the Register Codes. To serially access the registers of a unit, the protocol uses either the **Standard Addressing** or the **Extended Addressing**, depending on the total number of registers to access to.
Please refer to the "User's manual" of the relevant unit to find out the code assignments and the mode of register addressing.
To clearly differentiate the serial access codes, the Extended Registers Codes are always preceded by an exclamation mark "!". Subcodes S1 and S2 must be always set to "0" unless other values are expressly stated in the "User's manual" of the unit.

3 – Reading from registers

When you need to read data from the operational registers (RAM), then you must send one of the following request strings (telegrams), depending on the addressing mode.

3.1 Reading standard register codes



EOT = Control character CTRL D (Hex 04)

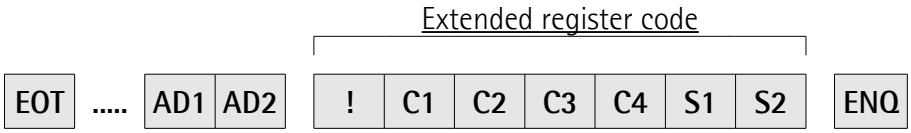
AD1 = Unit address, High byte
AD2 = Unit address, Low byte

C1 = Register code, High byte
C2 = Register code, Low byte

ENQ = Control character CTRL E (Hex 05)

For the complete list of valid register codes C1 and C2 please refer to the parameters list in the "User's manual" of the relevant unit.

3.2 Reading extended register codes



EOT = Control character CTRL D (Hex 04)

AD1 = Unit address, High byte
AD2 = Unit address, Low byte

! = Exclamation mark (Hex 21)
C1 = Register code, High byte
C4 = Register code, Low byte
S1 = Subcode, High byte
S2 = Subcode, Low byte

ENQ = Control character CTRL E (Hex 05)

For the complete list of valid register codes C1 and C2 please refer to the parameters list in the "User's manual" of the relevant unit.

3.3 Addressing examples

3.3.1 Example 1: standard addressing

You need to read the register having code "**03**" from the unit with device address "**31**".

			Unit address		Register code		
	EOT	...	3	1	0	3	ENQ
Hex:	04		33	31	30	33	05

3.3.2 Example 2: extended addressing

You need to read the register having code "**! 081A**" from the unit with device address "**11**".

			Unit address		Register code							
	EOT	...	1	1	!	0	8	1	A	0	0	ENQ
Hex:	04		31	31	21	30	38	31	41	30	30	05

Please note that figures 0-9 and characters A-F may be used for register addressing; A-F characters are expressed in ASCII hexadecimal codes (from 41 to 46).

3.4 Response to a valid request

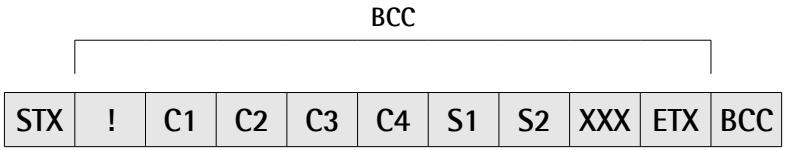
When a correct unit address and a valid register code are sent, the unit will respond issuing one of the following telegrams (depending on the addressing mode).

3.4.1 Response to a valid standard addressing

	BCC					
STX	C1	C2	XXX	ETX	BCC	

STX = Control character CTRL B (Hex 02)
XXX = Register code
ETX = Control character CTRL C (Hex 03)
BCC = Block check character

3.4.2 Response to a valid extended addressing



STX = Control character CTRL B (Hex 02)
XXX = Register code
ETX = Control character CTRL C (Hex 03)
BCC = Block check character

The total number of data characters marked with XXX in the tables above depends on the actual numeric value of the selected data register and may also be preceded by a minus sign ("-") in case of negative values. Preceding zeros are always deleted and does not appear in the telegram.
The BCC block check character is generated by an Exclusive-OR function over all characters between "C1" and "ETX" in the first "standard addressing" case, between "!" and "ETX" in the second "extended addressing" case ("C1", "!", "ETX" included). For further information refer to the section "6.1.1 Transmission of the data string" on page 26.

3.5 Response to an invalid request

When the request string contains an invalid or unknown register code (C1 – C4 or S1 – S2), then the response of unit will be as follows.

3.5.1 Response to an invalid standard addressing

STX	C1	C2	EOT

3.5.2 Response to an invalid extended addressing

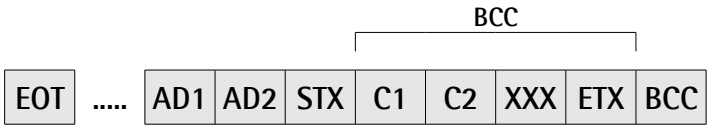
STX	!	C1	C2	C3	C4	S1	S2	EOT

In case of other different errors in the request telegram, the unit will just respond with "NAK" (Hex 15).

4 – Writing to registers

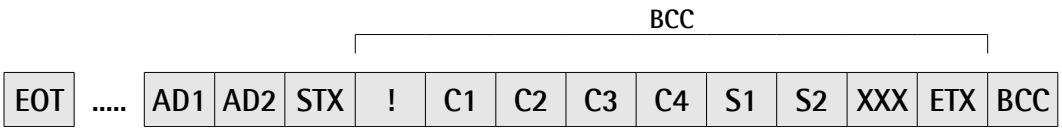
When you need to enter new values in the registers via personal computer, then you must send one of the following request telegrams, depending on the addressing mode.

4.1 Writing standard register codes



EOT = Control character CTRL D (Hex 04)
AD1 = Unit address, High byte
AD2 = Unit address, Low byte
STX = Control character CTRL B (Hex 02)
C1 = Register code, High byte
C2 = Register code, Low byte
XXX = New register data (ASCII code)
ETX = Control character CTRL C (Hex 03)
BCC = Block check character

4.2 Writing extended register codes



EOT = Control character CTRL D (Hex 04)
AD1 = Unit address, High byte
AD2 = Unit address, Low byte
! = Exclamation mark (Hex 21)
STX = Control character CTRL B (Hex 02)
C1 = Register code, High byte
C4 = Register code, Low byte
S1 = Subcode, High byte
S2 = Subcode, Low byte
XXX = New register data (ASCII code)
ETX = Control character CTRL C (Hex 03)
BCC = Block check character

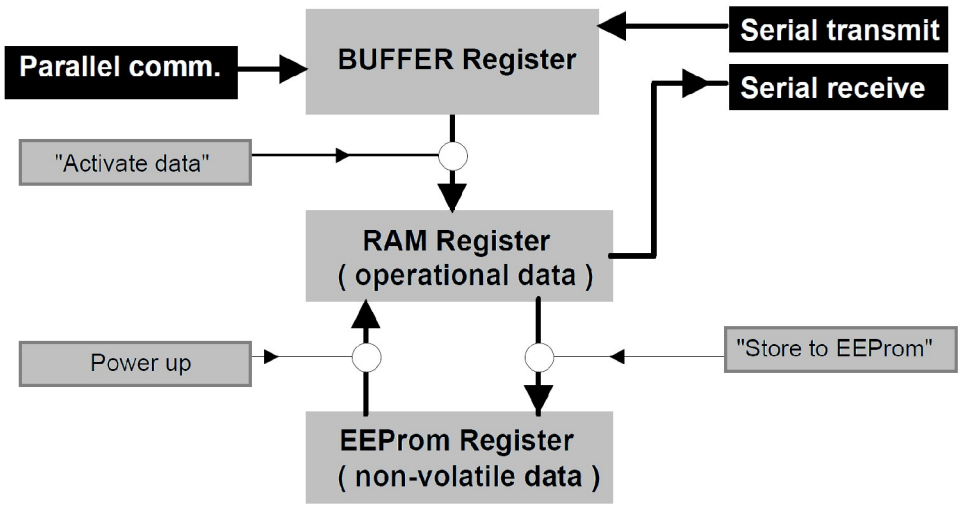
The data string marked with XXX in the tables above can have any number of characters and may also contain preceding zeros or a negative sign.
The BCC block check character is generated by an Exclusive-OR function over all characters between "C1" and "ETX" in the first "standard addressing" case, between "!" and "ETX" in the second "extended addressing" case ("C1", "!", "ETX" included).

In case of correct transmission of the above protocol, the unit will respond with "ACK" (Hex 06).
In case of any error, the unit will just respond with "NAK" (Hex 15).

NOTE
Data entered via serial transmission will be always stored in a temporary buffer register first and will not affect the current operation. To make them active and operational, then you must send the ACTIVATE DATA command.
This procedure is intended to allow the operator to enter a complete new set of parameters in the background of the unit without bearing on the production in process and to activate all the parameters at the same time by sending one single command.

4.3 Remarks about the registers organization

At any parameters read-out you will read the current operational data from the RAM. Parameters that have been stored in the buffer memory only and not yet activated cannot be read via serial request.
When the power is turned on, the unit automatically transfers the EEProm data to the RAM register. Serial register alteration carried out previously will be lost, unless it has been activated before and then stored to the EEProm.



5 – Sending the control commands

The same transmission protocol described in the previous sections is used also to send all control commands. However, the data string XXX uses one only character which is "**1**" when you need to switch the function ON and "**0**" when you need to switch it OFF again. Some of the serial commands will automatically reset to zero upon execution of the corresponding commands (such as ACTIVATE DATA or RESTORE TO EEPROM commands). Other commands (such as RESET, START/STOP, TRIM commands etc.) need to be set and even reset by using the same serial command.
For a complete list of the applicable command codes, please refer to the "User's manual" of the relevant unit.
All serial transmissions of a control command will get the same result as you get when you set the appropriate hardware input to "HIGH" logic level.

WARNING
There is a logical "OR" condition relating all hardware control inputs and their corresponding serial command flags. For this reason, it is compulsory to have both the hardware input and the serial command set to OFF at the same time when you need to switch the command OFF!
If, for instance, the RESET flag has been set to "1" via serial communication (RESET command is ON), the unit will be in its RESET state independently of the logical level of the hardware reset input. Thus the unit is in its normal operative state only when both the RESET serial flag is set to "0" and the RESET input has LOW logic level.
When the power is turned on, all serial command flags will be set to "0" automatically.

6 – Practical examples

6.1 Some practical examples of how the protocol works

The following example is intended to describe how to set a new value next to the Y register and send it to the Z unit.
Z unit uses a standard addressing for register access. Should you use a unit with extended addressing, the following example is still completely valid except the extended version of the address codes and the subcodes.
In the example the Z unit has serial **unit address "11"**, while Y register uses the serial **access code "00"**. You need to set the register to the **value "0.9873"**.

6.1.1 Transmission of the data string

First of all, you must transmit the following data string consisting of totally 13 ASCII characters.

No.	Expression	ASCII	Hex	Binary code Hi---------- ----------Lo	Comment
01	EOT	EOT	0 4	0 0 0 0 0 1 0 0	Control character initialization
02	AD1	1	3 1	0 0 1 1 0 0 0 1	Address, High byte
03	AD2	1	3 1	0 0 1 1 0 0 0 1	Address, Low byte
04	STX	STX	0 2	0 0 0 0 0 0 1 0	Control character
05	C1	0	3 0	0 0 1 1 0 0 0 0	Register code, High byte
06	C2	0	3 0	0 0 1 1 0 0 0 0	Register code, Low byte
07	X (data)	0	3 0	0 0 1 1 0 0 0 0	Factor, Highest digit
08	X (data)	9	3 9	0 0 1 1 1 0 0 1	
09	X (data)	8	3 8	0 0 1 1 1 0 0 0	
10	X (data)	7	3 7	0 0 1 1 0 1 1 1	
11	X (data)	3	3 3	0 0 1 1 0 0 1 1	Factor, Lowest digit
12	ETX	ETX	0 3	0 0 0 0 0 0 1 1	Control character
13	BCC	6	3 6	0 0 1 1 0 1 1 0	Block check character

Characters on a grey background are used to form the Block check character by means of an **Exclusive-OR function**. Now consider each of the 8 columns in the Binary code field.
In the HIGH BIT column (first column on the left in the Binary code field), you can find only zeros in all rows of the column, therefore the Exclusive-OR function value will result "0" and the High bit of the Block check character will be "0" in this column.
In the LOW BIT column (last column on the right in the Binary code field), you find the following sequence (from top -C1- to bottom -ETX-): 0 – 0 – 0 – 1 – 0 –

1 – 1 – 1. Also in this case the Exclusive-OR function value is "0", therefore the Low bit of the Block check character will be "0" in this column.
Thus we can state the following rule:

- when the number of "1" values in a column is **even**, the Block check bit must be "**0**" in the relevant column;
- otherwise, when the number of "1" values in a column is **odd**, the Block check bit must be "**1**" in the relevant column.

In the example above, the eight bits in the Block check character row are as follows: **0 – 0 – 1 – 1 – 0 – 1 – 1 – 0**. "0011 0110" binary value corresponds to "36" in hexadecimal notation and to "6" in ASCII character.

6.1.2 Waiting for acknowledgement

After correct transmission, the unit will acknowledge by responding "**ACK**" ("06" in hexadecimal notation, "0000 0110" in binary notation).
Should the unit respond sending "**NAK**" ("15" in hexadecimal notation), then this means that the transmission has been aborted because of an error such as a wrong BCC or an incorrect sequence of characters etc.
If the unit does not send back any response, this means that the transmission string is incomplete or the basic serial settings such as Baud rate or Data format are wrong.

6.1.3 Transmitting further parameters

Now we can transmit any further parameters as needed without affecting the machine processes.

6.1.4 Activating entered data

After all desired parameters have been sent successfully, we must activate the new settings to make them effective and operational. In the following example the Z unit has serial **unit address "11"**; we must write value "**1**" into the ACTIVATE DATA register having code "**67**" (C1 = 6; C2 = 7). Then we must send the following string:

No.	Expression	ASCII	Hex	Binary code Hi---------- ----------Lo	Comment
01	EOT	EOT	0 4	0 0 0 0 0 1 0 0	Control character initialization
02	AD1	1	3 1	0 0 1 1 0 0 0 1	Address, High byte
03	AD2	1	3 1	0 0 1 1 0 0 0 1	Address, Low byte
04	STX	STX	0 2	0 0 0 0 0 0 1 0	Control character
05	C1	6	3 6	0 0 1 1 0 1 1 0	Register code, High byte
06	C2	7	3 7	0 0 1 1 0 1 1 1	Register code, Low byte
07	X (data)	1	3 1	0 0 1 1 0 0 0 1	Activation command ON
08	ETX	ETX	0 3	0 0 0 0 0 0 1 1	Control character
09	BCC	3	3 3	0 0 1 1 0 0 1 1	Block check character

6.1.5 Saving data to EEProm

This operation is optional. If you do not send this command, the unit will use all data which has been transmitted and activated until it will be switched off. When you switch the unit on again, data will be uploaded from the EEProm.
The serial register code of the STORE command is "**68**" (C1 = 6; C2 = 8). Data value must be set to "**1**" to make the command operational (ON).

WARNING
The EEProm memory chip life time is limited to a total number of about 100,000 storage cycles. After this, saved data might be lost.

7 - ASCII code chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	TAB	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	space	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

The number of the row plus the number of the column are the hexadecimal code of each ASCII character.
For example: @ = row no. 4 + column no. 0; the hexadecimal code of character @ is 40.

This page intentionally left blank

This page intentionally left blank

Document release	Description
1.0	1st issue

Lika Electronic
Via S. Lorenzo, 25 - 36010 Carrè (VI) - Italy

Tel. +39 0445 806600
Fax +39 0445 806699

Italy: eMail info@lika.it - www.lika.it
World: eMail info@lika.biz - www.lika.biz