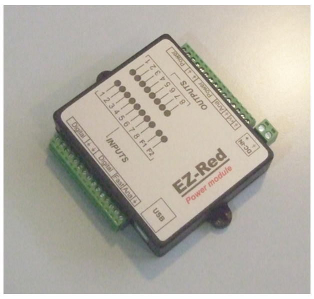
EZ-Red - I/O power module

"Console" mode with serial terminal emulation

Find other manuals at http://www.xonelectronics.it



Index

Introduction

The EZ-Red is an I/O module for PC which permits to operate electrical devices (5 to 30 volts) for building software-controlled automations with a personal computer. The device has 8 digital inputs, 8 digital outputs, 2 analog inputs (0-10 volts) and 2 analog outputs; moreover there are two fast and opto-coupled inputs which, if wanted, can connect to a quadrature encoder. The 8 digital outputs are "power" outputs that can drive inductive loads such electro valves and small CC motors.

This module can be used in three different modes:

1. As a dumb slave of the computer: the PC analyses the inputs and sets the outputs: the EZ-Red passively performs what the PC tell it to do.
2. As a stand-alone device, previously programmed with a PLC cycle written by the user and compiled by the supplied compiler. In this mode, no computer is required after the programming step.
3. In a combination of the two methods above: the module executes its internal configurable cycle, and the PC interacts with the module. Typically, the EZ-Red performs time-critical tasks and the PC modifies parameters, reads data for storing, implements a graphic user interface, and so on.

In any case, the communication between PC and EZ-Red can be done in two ways. The first is to use a Windows DLL (dynamic link library) which presents a high-level set of functions, hiding the low-level details of communication. The second way is technically simpler and is based on simple communication like a serial terminal. In both direction data is exchanged using ASCII text carrying commands or replies. The application program must implement a normal two-ways communication through a (virtual) serial port.

The EZ-Red has a configurable watch-dog, to protect from computer failures; if enabled, and the communication times out (the time-out is programmable), the module sets the outputs to a "safe" pattern – configurable.

This manual has been updated on april 2018, adding commands for reading and writing the internal memories of the PLC cycle. See the chapter DT, R and FLASH registers).

These commands are available from EZ-Red firmware v1.3.

FTDI Driver installation

In order to communicate with EZ-Red module is required first to install the correct FTDI drivers (Usb drivers for serial port emulation), available from the producer site (Ftdi) or from the XON Electronics site in the EZ-Red page. As soon as the EZ-Red is connected to the PC the drivers, if not already installed, will be asked or by the operating system. Please see the User Manual for further details.

Device connection

Power on the EZ-Red and connect it to the PC. The FTDI driver creates two virtual serial ports called A and B. To communicate with the module the port A must be used: look the COM name/number associated with port A and use that port.

Communication details

Communication parameters are 38400 baud, 8 data bit, no parity, 1 stop bit and no flow control.

Every data packet from PC to EZ-Red is composed of a series of ASCII characters terminated by a CR (carriage return, char number 13 corresponding to the Enter key). Space (char number 32) and LF (line feed, char number 10) are allowed but discarded by EZ-Red; moreover, EZ-Red is case insensitive - capital letters and lower-case ones are regarded as equal.

The maximum packet length is 76 characters; if more characters are sent, until the next CR, are ignored.

All the packets from PC to EZ-Red start with either a "?" (question mark, make a read) or ">" (greater than, make a write). The composition of the read command is the following:

Read command example				
?	**X**	**1**	**<CR>**	**(<LF>)**
Command (print)	Command Argument (X1)		Terminator	LF discarded if sent

The argument of the command, **X1** in this example, is an identifier known by EZ-Red; see later the list of all the known identifiers. If the sent packet is well formed and the identifier is valid, EZ-Red replies with the requested value:

Reply to read command				
X	**1**	**=**	**0**	***<CR>***
Identifier (X1)		Syntactic element	Value	Terminator

After the "=" (equal sign) there are always 1, 3 or 5 digits - it depends on the identifier kind.

The composition of a write command (of an output, or an internal identifier) is the following:

Write command example						
>	**Y**	**1**	**=**	**1**	**<CR>**	**(<LF>)**
Command ("set")	Command Argument (Y1)		Syntactical element	Value	Terminator	LF discarded if sent

The number after the equal must be a positive integer without sign and it must belong to the acceptable interval of the indicated identifier: for a digital output like Y1, only 0 and 1 are valid; for YBYTE (a byte comprising all the digital outputs) values range from 0 to 255.

When EZ-Red receives the packet (after having received the last CR), it sends the reply:

Positive response to write command		
O	**K**	**<CR>**
Command accepted (no error)		Terminator

If the sent packet does not begin with "?" or ">", no reply is received. If it does contain other kind of errors, EZ-Red replies with the following packet:

Reply to a packet containing errors					
E	**r**	**r**	**o**	**r**	**<CR>**
Negative acknowledgement					Terminator

Events (replies not solicited)

It is possible to instruct the EZ-Red to transmit automatically on certain events; this lets the application avoid a continuous poll, waiting instead or doing other things and latency times are also reduced.

Possible events to wait for are a change in the digital inputs or the execution of the instruction LOG executed by the PLC cycle of the module (see the Programming manual). Input change event has the following format:

Digital input change example packet																	
!	**X**	**B**	**=**	**0**	**0**	**5**	**<SPC>**	**E**	**N**	**C**	**=**	**1**	**2**	**3**	**4**	**5**	**<CR>**
	Digital inputs byte							Encoder value									

Data format for an event generated by a LOG instruction has the following format:

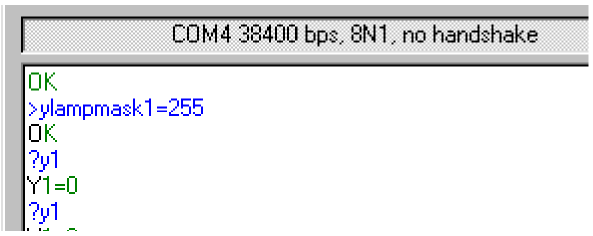
Cycle-generated event example packet																		
!	**L**	**O**	**G**	**=**	**0**	**0**	**5**	**<SPC>**	**D**	**A**	**T**	**=**	**3**	**0**	**0**	**4**	**5**	**<CR>**
	"Channel" of the LOG instruction								Value to communicate									

The LOG instruction reports the channel (first argument of the instruction) and a "DAT" (data) value - the second argument specified to the instruction. Examples are "LOG 5 ENCODERL", "LOG 8 WAITREMAIN1"; please refer to the programming manual.

Dialogue between PC and EZ-Red

After having sent a packet to the module, the PC must wait for a reply; characters sent from PC after the original packet but before a reply is received are discarded. EZ-Red replies fairly quickly, though (normally in 3/100 of a second) - it is hence recommended to wait a reply for a maximum of 50 milliseconds .

The normal dialogue is that the PC sends a request and EZ-Red replies. This flow can be polluted by the asynchronous events sent by EZ-Red: in particular, it can happen that EZ-Red starts the transmission of an event while it is already receiving a command from the PC. From the application point of view, the PC receives a reply which is incoherent with the command sent, and this is easily understandable because event packets always begin with a "!" (exclamation mark). The application must process the event (or discard it) and continue to wait for the intended reply.



COM4 38400 bps, 8N1, no handshake

```
OK
>ylampmask1=255
OK
?y1
Y1=0
?y1
```

Identifiers

All the operations involved in using the EZ-Red are implemented by reading or writing identifiers. Now a list of all the identifiers is presented, accompanied with explanations.

Reading inputs

The module has 8 digital inputs, 2 fast opto-coupled inputs (suitable also for an encoder) and 2 analog inputs. Some of the inputs also have hardware counter that can be read.

Digital inputs X1..X8, and XBYTE

They can be read one by one using the identifiers from X1 to X8, or all together using the identifier XBYTE which contains a number from 0 to 255 where every bit corresponds to a digital input. Digital input 1 is associated to the least significant bit, and so on up to the last, most significant bit which is associated to digital input 8.

By sending the command:

?X1***<CR>***

EZ-Red replies, for example:

X1=1***<CR>***

(X1 is 1: there is voltage on digital input 1); sending instead:

?XBYTE***<CR>***

EZ-Red could reply, for example:

XBYTE=192***<CR>***

(XBYTE=192 has set the bits of weight 128 e 64, corresponding to X7 e X8). The received number is always composed of three digits.

XCOUNT1 and XCOUNT2 (counters associated to X1 and X2)

The digital inputs X1 and X2 have two 16 bits hardware counters associated with them, which count the rising edges of the input signal. To read the counters, send:

"?XCOUNT1***<CR>***" (or "?XCOUNT2***<CR>***")

to receive in reply, for example:

XCOUNT1=01200***<CR>***

Analog inputs AIN1 and AIN2

The two analog inputs read a voltage from 0 to 10 volts, and yield a number from 0 to 255. A value of 255 means 10 volts are present in the input. To query the value of an analog input, send:

?AIN1***<CR>***

to obtain the reply, for example:

AIN1=160***<CR>***

The returned number is expressed in 256th of 10 volts; in order to know the real voltage the following formula can be applied: voltage = value*10 / 256; a value of 160 corresponds to 6.2 volts.

Fast inputs FX1 and FX2

These inputs are identical in concept to the normal ones and can be used in the same way. To read their status send the command "?FX1***<CR>***" or "?FX2***<CR>***": the reply, similar to that of X1..X8, contains 0 or 1.

These inputs are, at the hardware level, quite different from the normal ones – they are fast and opto-coupled, suitable for counting fast enough to connect a quadrature encoder; counters and encoder are explained below.

Counters FXCOUNT1 and FXCOUNT2

FX1 and FX2 have a hardware 32 bit internal counter which increments on every rising edge of the input signal. Using the terminal emulation mode (text commands) it is only possible to read the lower 16 bits by sending

"?FXCOUNT1***<CR>***" or "?FXCOUNT2***<CR>***"

ENCODER (connected to FX1 and FX2)

It is possible to connect a quadrature encoder (A and B channels) to the fast inputs FX1 and FX2; the EZ-Red decodes the A and B channel to obtain a positional value that can be read by sending:

?ENCODER***<CR>***

and the module replies, for example:

ENC=02490***<CR>***

The reply always contains a 5 digits number after the equal sign, which represents a 16 bit value even if the internal counter is 32 bits wide. A further “forward” step when the value is 65535 will reset it to 0; instead, a further “back” step when the value is 0 will being it to 65535.

It is possible to reset or assign an arbitrary value to this identifier (this operation is called “preset”), for example when the machinery reaches a particular switch. To do this, send the command:

>ENCODER=0***<CR>***

to reset the position. The module replies with:

OK***<CR>***

Digital power outputs Y1..Y8 (and YBYTE)

These identifiers are very similar to X1..X8 and XBYTE but they refer to the outputs, not the inputs. They can be read in the same way, for convenience, and also because the watch-dog can set them on its own.

Analog outputs AOUT1 and AOUT2

Again, these outputs can be read in addition to be written, for convenience and because the watch-dog can modify them if it fires.

Setting outputs

EZ-Red has 8 digital outputs and two analog outputs. The identifiers are Y1..Y8 and YBYTE for the digital ones; AOUT1 and AOUT2 for the analog ones. After any command that modifies these outputs, EZ-Red replies with "OK<CR>" if the command has been understood and accepted. For convenience, these identifiers can also be used in read commands. The power outputs Y1..Y8 also have an optional automatic blink pattern.

Outputs Y1..Y8 (power outputs), and YBYTE

To modify a single output, send the command:

>Y1=1***<CR>*** (turns on the output)

or:

>Y1=0***<CR>*** (turns off the output)

to modify a single output. As seen for the inputs X1..X8, for outputs too is available the identifier YBYTE that sets all the outputs in a single command:

>YBYTE=255***<CR>***

turns on all the power outputs.

Blinker YLAMPMASK1..YLAMPMASK8 (on Y1..Y8)

In order to simplify to make an output blink, every power output has a virtual blinker which can apply a pattern to every output, without the need for the PC to explicitly turn on and off the output repeatedly. To do this, a 16 bit pattern is employed, where every bit lasts 8/100 seconds; a bit set to 1 turns on the output, and a bit set to 0 turns the output off. These bits are examined, cyclically, from right (least significant) to left (most significant). For example, a pattern like "0000.0000.1111.1111" generates a slow and regular blinking with a duty cycle of 50%: the output stays on for 64/100 of a second, then off for 64/100, then the cycle restarts. The cited pattern is written "FF" in hexadecimal notation and 255 in decimal notation. A pattern of value "1" makes a short flash 8/100 of a second long every 128/100 of a second, while 0101.0101.0101.0101 makes a frantic blinking. To assign a blinking pattern to an output send a command like:

>YLAMPMASK1=255***<CR>***

The previous command make Y1 blink slowly. To turn off the blinking, send the pattern 0.

Se

If an output is assigned a blinking pattern different than zero, that output does not respond to the normal commands "Yn=x", not even if that is done by the PLC cycle. To make the output respond again, its blinking pattern must be reset to zero.

Analog outputs AOUT1 and AOUT2

Assign to these identifiers a number from 0 to 255 proportional to the output voltage desired, from 0 to 10 volts; the concept and translation formula is the same as what seen for AIN1 and AIN2. For example:

>AOUT1=128***<CR>***

will put 5 volts on analog output 1, because 128 is half of 256, so the voltage will be half of the maximum - 10 volts: the result is 5 volts.

.

More added functions

Introduction

What seen until now is enough to dialogate with EZ-Red to read inputs and set outputs. There are a few more features, accessible through identifiers too, which are explained below.

FLAGS (options and other internal signals)

FLAGS is an identifier of a byte made of 8 bits, and each of them indicates a particular state of the module; from a point of view, it is similar to the already seen XBYTE. Some of this bits (flags) can be manipulated in order to modify an option or to execute a function. Using console mode (text commands), these bits can only be managed in group with FLAGS: to know a single bit the whole group must be read, and a logic AND must be performed to mask out the unwanted bits. To modify a single flag, all the 8 must be read, the single bit(s) must be changed, and all the 8 bits have to be written back. To read the FLAGS byte send:

?FLAGS***<CR>***

EZ-Red reply is, for example:

FLAGS=082

To write the register send, for example:

>FLAGS=080***<CR>***

As usual, EZ-Red replies with "OK***<CR>***".

The following table explains the bits in FLAGS. The reported name is the one used in the PLC cycle (please refer to the Programming Manual):

N° of the bit	Name	Description
0 (*AND* with 1)	REPORTBACK	1 if the automatic report on inputs change is active; it can be written to turn on this function. Please refer below in this manual for more informations.
1 (*AND* with 2)	WDTFIRED	1 if the internal watch-dog is operating (fired). See the section Wach-dog in this document.
2 (*AND* with 4)	SENDTOPC	When written 1, an IOCHANGE transmission (report back event) is solicited. It returns to zero automatically. See the section Events.
3 (*AND* with 8)	CYCLERUN	Indicates if the PLC cycle is running (when 1) or not (when 0). It can be written in order to start/stop a PLC cycle (if programmed).
4 (*AND* with 16)	AUTOUPDATEXY	Shows what type of I/O updating is active for the PLC cycle. Refer to the Programming Manual.
5 (*AND* with 32)	DISABLEUSB	Turns off the communication; it has no effect on text commands mode.
6 (*AND* with 64)	WDTSTOPSCYCLE	Indicates whether the watch-dog, when fires, also stops the PLC cycle or not.
7 (*AND* with 128)	PWDPROTECT	Indicates whether the PLC program memory is password protected.

Events (automatic transmission)

EZ-Red can send data to the PC in automatic mode in addition to reply (or request) mode. This can happen in two situations: when the inputs change, if EZ-Red is instructed to do so; and when the PLC cycle executes the LOG instruction (see the Programming Manual).

IOCHANGE event (flag SENDTOPC)

The automatic transmission on inputs change is activated by setting to 1 the flag (bit) REPORTBACK (least significant bit of FLAGS, see previous section). When this mode is ON, a change in one or more inputs X1..X8 performs a transmission to the PC. It is also possible to programmatically start this kind of transmission by writing 1 to SENDTOPC bit of FLAGS. The transmission from EZ-Red to PC has the following format:

IOCHANGE transmission example																	
!	**X**	**B**	**=**	**0**	**0**	**5**	**<SPC>**	**E**	**N**	**C**	**=**	**1**	**2**	**3**	**4**	**5**	**<CR>**
	Inputs state							Value (position) of encoder									

Please note that this event carries the position of the encoder, but a change in position does NOT start an automatic transmission: encoders count too fast to transmit every single step.
Automatic transmission is limited/delayed by EZ-Red firmware to a maximum of 10 transmissions a second; without this limit communication congestions could arise. If a transmission gets delayed, the reported values are the very last values, not the one that were read before the delay. Giving this, it is possible to receive two consecutive packets stating that "XB=1", but this is correct and means that at least one bit in XBYTE has changed at least two times. This limitation also affects the explicit transmission request made by writing 1 to SENDTOPC.

LOG instruction (executed by the PLC program)

The PLC cycle can ask to send data to the PC programmatically, using the LOG instruction. Refer to the Programming manual for details. The PC receives a packet with the following format:

PLC cycle requested event format example																		
!	**L**	**O**	**G**	**=**	**0**	**0**	**5**	**<SPC>**	**D**	**A**	**T**	**=**	**3**	**0**	**0**	**4**	**5**	**<CR>**
	"Channel" of LOG instruction								Argument value									

DT registers, R registers and FLASH registers (from fw v1.3)

The PLC cycle of EZ-Red has a set of registers to work with: 64 virtual relays “R” (from R1 to R64), the 64 16-bit registers DT (from DT1 to DT64), and 32 bytes of non-volatile memory. The EZ-Red, from firmware v1.3, implements, in the simple text protocol, the possibility to read and write these registers too.

The virtual relays R1 .. R64 can be read and written using their normal identifier, like:

?R1<CR>	(read the content - 1 or 0 - of the register)
>R27=1<CR>	(write “1” to R27)

The R registers store bits: they only works with values 0 or 1.

The 64 16-bit registers can store values from 0 to 65535 (unsigned). They can be read and written using the same identifier in simple text protocol (form firmware version 1.3):

?DT1<CR>	(read the content of DT1)
>DT33=57<CR>	(write a value to the register DT33)

The 32 bytes of non-volatile memory are accessible in different formats (bit, byte, word) from the PLC cycle, but can only be read and written as bytes by the simple text protocol. The format is:

?FLASH1<CR>	(read the content of the first byte of non-volatile memory)
>FLASH28=255<CR>	(write a value to the flash, this FLASH28 corresponds to CONFIGCHR28)

--WARNING--: after a write operation to the flash memory, wait a few seconds (20 – 30) before powering off the EZ-Red. The EZ-Red waits a few seconds before actually updating the flash – this is done to improve the reliability of the memory.

FBACKS and FBMASK (power outputs feedbacks)

EZ-Red monitors the output lines and can detect an overload (which leads to high temperature) or an open circuit (symptom of a break in the cabling). When an Y output is off (no voltage), an open circuit is detected and the respective bit is set in FBACKS; when an Y output is on, an excessive current is detected and after some time the relative bit in FBACKS is set; otherwise, the bit is cleared.

The feedback circuitry reports this bits in couples: one bit (bit 0) for Y1 and Y2, one for Y3 and Y4, and so forth. If the FBACKS byte has the bit 0 high, is means that at least one of Y1 or Y2 has an error condition. If any of the first 4 bits of FBACKS is 1, then the red LED near the USB connector of EZ-Red is light.

The FBACKS byte can be read by sending:

?FBACKS***<CR>***

...or, available in firmware 1.3, there is also an alias for this:

?FBBYTE<CR>

An alarm condition detected by FBACKS can fire the watch-dog, if the corresponding bit in FBMASK is also set. FBMASK is all zero at start-up. FBMASK can be programmed by sending:

>FBMASK=xxx***<CR>***

where xxx is a number (from 0 to 15) containing the desired bit mask relative to bits in FBACKS (the firmware 1.3 has an alias for FBMASK named WDTFBBYTE).

The conformation of feedback circuitry suggests that, if an Y power output is not used (not connected), it must be held high to avoid detection of a broken circuit.

Watch-dog (integrity monitor)

A watch-dog is a supervisor mechanism that detects incongruities in some aspect of a system, in order to stop the system if it does not behave correctly, to avoid further damage; normally, the most important thing in that case is to set the outputs in a "safe" configuration.

EZ-Red monitors two aspects of the system: the feedbacks of the power outputs (see above FBACKS and FBMASK) and the reliability of the communication with the PC.

As seen before, if FBACKS sets a bit to 1 and the corresponding bit in FBMASK is also set, the watch-dog fires because an incongruent state is detected. Reliability of the communication is obtained by monitoring the time between command packets sent by the PC. If the time between two packets is higher than a programmed value, the communication is determined as failed and the watch-dog fires. The computer can stop to send commands because the communication breaks, or because the computer hangs. In any case, the computer is responsible for the correct execution of the system, so if the computer stops to send commands, the system can not continue to work.

WDTTIME (time-out of PC commands)

This identifier is a 16 bit integers expressing milliseconds, from 0 (disabled) to 65535 (more than 65 seconds). At start-up the value of WDTTIME is 0: there is no time-out between packets sent by the PC, so the watch-dog will not fire because of a communication failure. To set a different time-out send:

>WDTTIME=xxx***<CR>***

where xxx is a number from 0 to 65535 indicating the desired time-out in milliseconds (0=no time-out - watch-dog disabled).

When the watch-dog fires the following happens:

1. The WDTFIRED flag is set; it can be observed by reading FLAGS

2. If WDTSTOPSCYCLE (a bit in FLAGS) is set, the PLC cycle is stopped
3. YBYTE (the configuration of Y outputs) is set to to the value indicated by WDTOUTS
4. The analog outputs 1 and 2 (AOUT1 and AOUT2) are set to the value of WDTAOUT 1 and 2

WDTOUTS (output pattern for the watch-dog)

This byte contains the value to assign to YBYTE when the watch-dog fires. The default value is 192, which means that outputs Y1 to Y6 are off, and Y7 to Y8 are on. To set a different pattern send:

>WDTOUTS=xxx***<CR>***

where xxx is the desired pattern.

WDTAOUT1 and WDTAOUT2 (analog values for watch-dog)

These two bytes contain a value, from 0 to 255, to be assigned to AOUT1 and AOUT2 when the watch-dog fires. The default value, at start-up, is zero.

XON Electronics Srl
www.xonelectronics.it
info@xonelectronics.it

Internet product page is www.xonelectronics.eu/en/ez-red-power-io-module-for-pc

Please report any error or imprecision to web@xonelectronics.it