EZ-Red

Modulo I/O di potenza per PC

Altri manuali sono disponibili sul sito http://www.xonelectronics.it

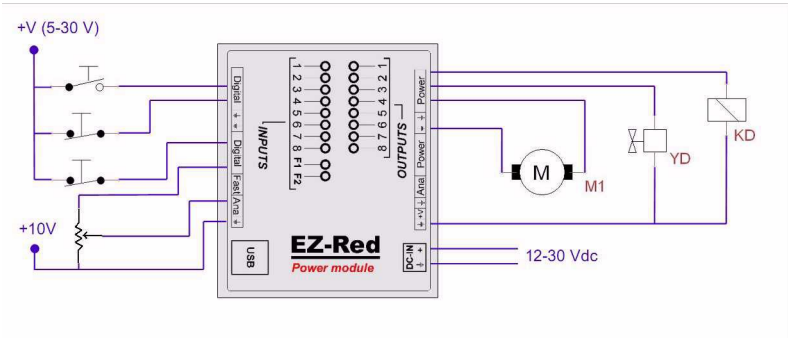
Indice

Introduzione....2
Alimentazione....2
Ingressi digitali a 24 volt....3
Ingressi veloci optoisolati....3
Ingressi analogici....3
Uscite di potenza a 24 volt....4
Uscite analogiche 0-10V....4
Uscita di alimentazione a 12 volt....4
Controllo di processo e acquisizione dati con il PC....5
Interfacciamento con DLL o con comandi di testo semplice (console)....5
PLC interno....5
Watch-dog interno (controllo di congruenza)....5
Variabili (word) di configurazione....5
Interfaccia USB....6
Installazione dei driver....6
Uso del libreria "ezreddll.dll"....7
C#....7
Delphi (pascal)....7
C++....7
Visual basic....7
Elenco delle funzioni....8
Dettagli delle funzioni....9
ezrd_open(comport: integer)....9
ezrd_close....9
ezrd_model: PChar (o char* in "C")....9
ezrd_int_ios(var ings, outs, da1, da2, an1, an2: byte)....10
ezrd_set_allouts(parm: cardinal)....11
ezrd_set_io(which1_8: cardinal; onoff: cardinal)....11
ezrd_up1....11
ezrd_dn1....11
ezrd_set_ana1(parm: cardinal)....11
ezrd_read_ain1(var parm: cardinal)....11
ezrd_read_ios(var onoff: cardinal)....11
ezrd_isup_io(which1_8: cardinal; var onoff: cardinal)....11
ezrd_isup1(var onoff: cardinal)....12
ezrd_readencoder(var flags, encoder: cardinal)....12
ezrd_SetWatchdog(tmout, ioconf, aout1, aout2: cardinal)....12

Introduzione

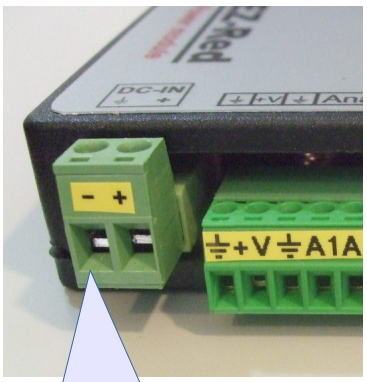
EZ-Red è un modulo di interfaccia di potenza a 24 volt, che si collega con l'interfaccia USB a un computer per controllare un processo esterno. Il modulo fornisce:

- 8 ingressi digitali a 24 volt (da 5 a 30 volt)
- 8 uscite di potenza a 24 volt, corrente 2A, anche per carichi induttivi
  - la tensione d'uscita è quella di alimentazione: da 15 a 30 volt
  - la corrente massima totale, sommando tutte le uscite, è di 4A
- 2 ingressi veloci (10 Khz), optoisolati (interfaccia encoder)
- 2 ingressi analogici 0-10V
- 2 uscite analogiche 0-10V
- 1 uscita ausiliaria di alimentazione a 12 volt, 200 mA
- Controllo, sulle uscite, di sovraccarico e carico interrotto
- Controllo sulle tensioni di alimentazione
- Controllo del processo in modo continuo (on-line), tramite chiamate "DLL"
- Watch-dog interno, con disposizione uscite configurabile
- PLC interno per memorizzare ed eseguire cicli in modo autonomo
- Memoria flash (non volatile) per memorizzare il ciclo del PLC



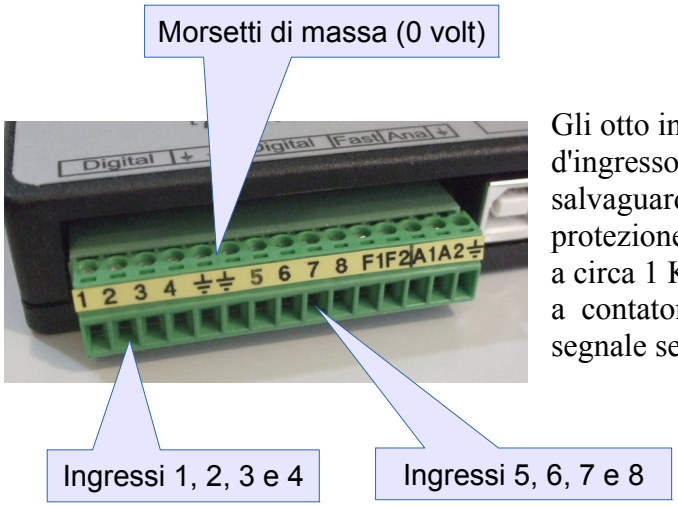
Lo schema mostra un esempio di collegamento: interruttori, potenziometri, motorini, relè ed elettrovalvole.

Alimentazione



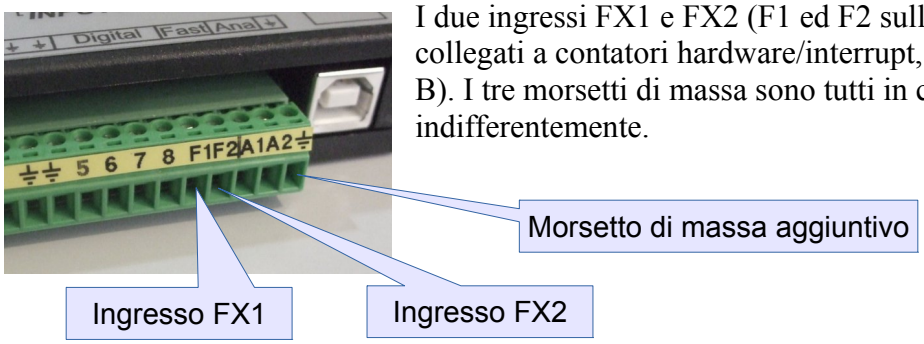
Il dispositivo deve essere alimentato con una tensione continua tra 15 e 30 volt. La massima corrente assorbita dal dispositivo è di 200 mA, a cui occorre sommare l'assorbimento dei dispositivi collegati alle uscite a 24 volt. Internamente viene generata una tensione ausiliaria +V di 12 volt; le due tensioni (alimentazione e +V) sono monitorate e possono essere lette dal computer o da ciclo. In caso di anomalia su queste tensioni, viene attivato il modo Emergenza. EZ-Red può essere alimentato anche con tensione inferiore a 15 volt (ma comunque superiore a 10 volt): in questo caso le uscite analogiche non raggiungono 10 volt e perdono linearità; anche l'uscita di alimentazione ausiliaria +V a 12 volt rimane stabilizzata con tensione di alimentazione non inferiore a 15 volt.

Ingressi digitali a 24 volt



Gli otto ingressi digitali X1..X8 hanno un'impedenza d'ingresso di 10 Kohm e un circuito di protezione a salvaguardia dei componenti interni. La circuiteria di protezione limita la frequenza massima dei segnali d'ingresso a circa 1 Khz. Due di questi ingressi, X1 e X2, sono associati a contatori hardware che permettono di contare i fronti del segnale senza dispendio di CPU.

Ingressi veloci optoisolati

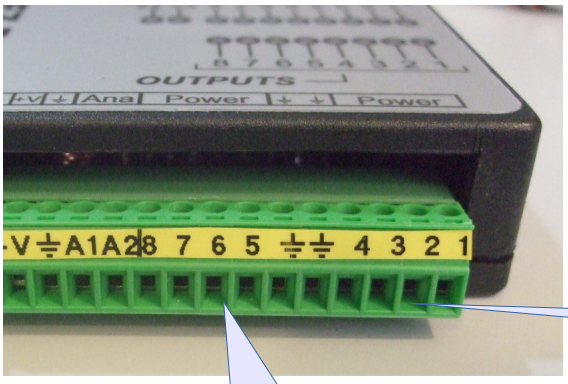


I due ingressi FX1 e FX2 (F1 ed F2 sulla morsettiera) sono optoisolati e collegati a contatori hardware/interrupt, con opzione encoder (canali A e B). I tre morsetti di massa sono tutti in comune e si possono utilizzare indifferentemente.

Ingressi analogici

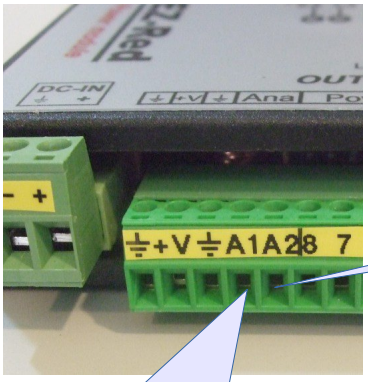
I due ingressi analogici AIN1 e AIN2 (A1 e A2 sulla morsettiera) accettano tensioni analogiche da 0-10 volt; hanno comunque un circuito di protezione che limita a 10 volt la tensione massima in ingresso.

Uscite di potenza a 24 volt



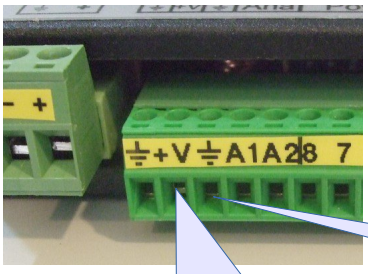
Le otto uscite di potenza Y1..Y8 possono fornire, ognuna, 2 ampere di corrente. Sono protette da sovraccarico e sovra-temperatura; è possibile inoltre rilevare l'assenza del carico. Lo stato di errore può essere monitorato con il computer o con il ciclo interno del PLC.

Uscite analogiche 0-10V



Le uscite analogiche AOUT1 e AOUT2 (A1 e A2 sulla morsettiera delle uscite) possono fornire circa 20 mA massimi, e dispongono di un circuito di protezione. La tensione analogica va da 0 a 10 volt.

Uscita di alimentazione a 12 volt

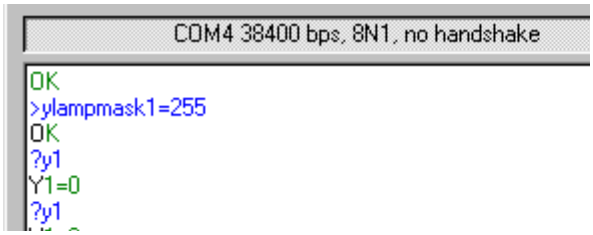


Sulla morsettiera delle uscite è disponibile una alimentazione ausiliaria di 12 volt, stabilizzata, con corrente massima di 300 mA, protetta da un fusibile auto ripristinante.

Controllo di processo e acquisizione dati con il PC

Attraverso l'interfaccia USB è possibile interagire con EZ-Red, usando chiamate dirette fornite dalla libreria DLL "ezreddll.dll", oppure attraverso l'invio di comandi di testo semplice. Con questi due metodi è possibile leggere lo stato di tutti gli ingressi e le variabili interne, impostare le uscite e gli stati interni del modulo, e quindi eseguire un controllo di processo e/o acquisizione dati.

Interfacciamento con DLL o con comandi di testo semplice (console)



Questo manuale descrive la libreria DLL; per l'utilizzo dei comandi di testo riferirsi al documento a parte intitolato ***EZ-Red - Modo terminale seriale***.

Anche se la libreria DLL è più adatta ai linguaggi compilati e LabView®, il modo *console* è completo ed efficace ed è utilizzabile anche con altri sistemi operativi, per esempio Linux.

PLC interno

Il modulo EZ-Red contiene un processore PLC interno, che gli permette di cooperare con il computer (il ciclo PLC ha tempi di risposta rapidi e precisi), oppure può far funzionare il modulo senza alcun computer collegato, come un normale PLC. Per utilizzare il PLC interno occorre scrivere un programma (riferirsi al ***Manuale di programmazione***), compilarlo, e memorizzarlo nella memoria non volatile di EZ-Red (riferirsi al ***Manuale di TSmon***).

Il programma PLC può essere protetto con una password e, inoltre, da ciclo è possibile attivare un bit di inibizione dell'interfaccia USB. Se da ciclo si attiva questo bit, il modulo diventa completamente dedicato alla gestione del ciclo e non è più possibile utilizzarlo in modo diverso (ma si può sbloccare fornendo la password corretta).

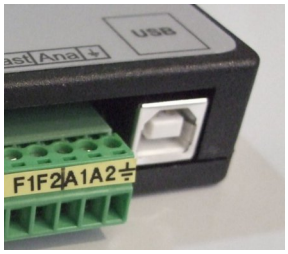
Watch-dog interno (controllo di congruenza)

Quando EZ-Red viene impiegato in un controllo di processo in combinazione con un PC, può essere desiderabile che, in caso di mal funzionamento del computer, il processo venga portato in uno stato sicuro, cioè con una disposizione delle uscite certa e non pericolosa. Il watch-dog, se abilitato, controlla che la comunicazione con il computer sia continua (entro un tempo configurabile di timeout) e, nel caso che la comunicazione s'interrompa, pone sulle uscite la configurazione (configurabile) di "emergenza". Opzionalmente, il watch-dog può arrestare il ciclo PLC. Oltre che per la mancanza di comunicazione con il PC, è possibile impostare l'intervento del watch-dog anche per il mal funzionamento di una uscita di potenza, o d'una anomalia della tensione di alimentazione. Nelle condizioni iniziali il watch-dog è disabilitato; può essere attivato da ciclo PLC o con l'apposita chiamata DLL.

Variabili (word) di configurazione

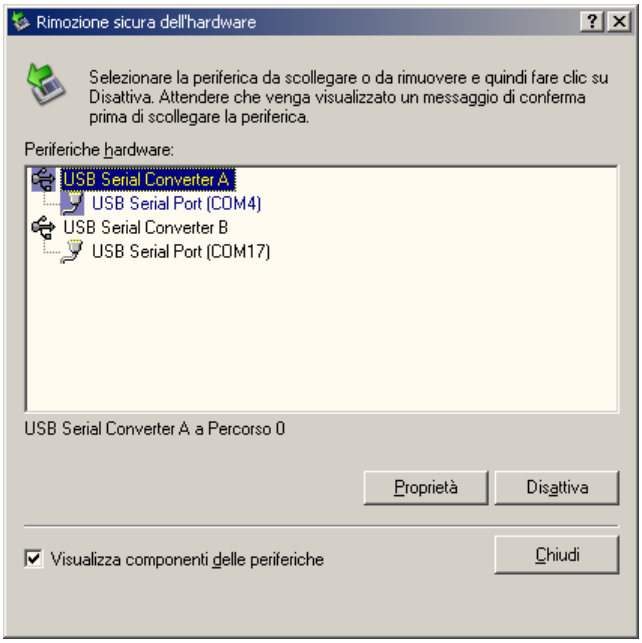
Nella memoria flash del dispositivo è possibile memorizzare, oltre a un ciclo PLC e l'eventuale password di protezione, anche una serie di word di configurazione. Questo meccanismo permette di rendere il ciclo PLC configurabile (con parametri), ed eventualmente fornire all'utilizzatore finale un programma per PC apposito per variare questi parametri.

Interfaccia USB



Il modulo EZ-Red si collega al computer tramite l'interfaccia USB, con un connettore standard “Type B” mostrato qui a fianco.

Con modulo alimentato, cavo USB inserito e driver di sistema operativo installati, vengono create due interfacce seriali virtuali; la prima di queste è il canale di comunicazione con EZ-Red (*USB Serial converter A*); la seconda (*USB Serial converter B*) non è usata.



L'immagine mostra che al primo canale, “A”, è associata la porta COM4. A questa porta occorre fare riferimento per comunicare con EZ-Red.

Il nome della porta può essere differente: è assegnato dal sistema operativo. Attraverso il pannello di gestione periferiche è possibile cambiare, entro certi limiti, il numero di porta assegnato. Riferirsi alle istruzioni specifiche del sistema operativo in uso.

Installazione dei driver

La comunicazione USB richiede l'installazione dei driver FTDI, forniti insieme a EZ-Red, scaricabili dal sito XON Electronics o dal direttamente dal sito di FTDI (www.ftdichip.com). Se non già installati in precedenza, essi vengono richiesti direttamente dal sistema operativo appena la periferica EZ-Red viene rilevata. In entrambi i casi occorre prima scaricare un file contenente tali driver, di tipo VCP (Virtual COM Port), e scompattare l'archivio in una cartella di propria scelta. Fatto questo, si può:

- Installare i driver prima di effettuare il collegamento hardware. Cliccare con il tasto destro sul file FTDIPORT.INF, e scegliere “Installa” dal menù contestuale, -oppure-
- Collegare e alimentare il modulo EZ-Red, e attendere che il sistema operativo faccia la richiesta per i driver; indicare di cercare i file necessari nella directory dove era stato scompattato il file archivio. E' possibile che il sistema operativo offra vie alternative per la ricerca e l'installazione del software necessario; se ciò accade, scegliere il modo preferito.

Uso del libreria "ezreddll.dll"

Copiare la libreria in una posizione raggiungibile dall'applicativo, e dichiarare le funzioni esportate dalla stessa (tutti i nomi sono completamente in minuscolo; in questo manuale si possono trovare scritte in maiuscolo per chiarezza). Solo le funzioni che si intende usare devono essere dichiarate, quelle non utilizzate possono essere tralasciate. E' possibile utilizzare pienamente il modulo usando due sole funzioni.

Quasi tutti i risultati delle funzioni sono interi con segno a 32 bit; EZRD_CLOSE non ritorna risultati, mentre EZRD_MODEL ritorna una stringa (stringa "C", terminata da un carattere NUL, come è normale nel sistema operativo Windows®). I parametri delle varie funzioni variano secondo la funzione usata. Tutte le funzioni usano il modello *stdcall*, che è lo standard (anche se non è l'unico modello disponibile) in ambiente Windows®. Il tipo di dato Cardinal indica un intero senza segno di 32 bit.

Prima di utilizzare qualsiasi altra funzione è necessario invocare EZRD_OPEN specificando il numero di porta COM da utilizzare (simulata da USB, per la porta "A"). EZRD_OPEN ritorna 0 se nessun errore è stato rilevato, oppure un numero diverso da 0 se si è verificato un errore (per esempio una porta non esistente o già impegnata). Poi, la funzione EZRD_INT_IOS permette di gestire tutti gli ingressi e le uscite. Per funzioni più avanzate, consultare l'elenco completo più avanti in questo manuale. Alcuni esempi di dichiarazione sono i seguenti.

C#

```
public class ExternalEZRD {
    [DllImport("ezreddll.dll")]
    public static extern int ezrd_open(int comport);
    [DllImport("ezreddll.dll")]
    public static extern void ezrd_close();
    [DllImport("ezreddll.dll")]
    public static extern int ezrd_int_ios(ref byte ings, ref byte outs, ref byte da1,
      ref byte da2, ref byte an1, ref byte an2);
}
```

Delphi (pascal)

```
mydll    = 'ezreddll.dll';
function  ezrd_open(comport: integer): integer; stdcall; external mydll;
procedure ezrd_close; stdcall; external mydll;
function  ezrd_int_ios(var ings, outs, da1, da2, an1, an2: byte): integer; stdcall;
  external mydll;
```

C++

```
extern "C" __declspec(dllimport) __stdcall {
  int ezrd_open(int comport);
  void ezrd_close();
  int ezrd_int_ios(unsigned char* ings, outs, da1, da2, an1, an2);
  }
```

Visual basic

```
Declare Function ezrd_open Lib "ezreddll.dll" (comport as integer) as integer
Declare Sub ezrd_close Lib "ezreddll.dll"
Declare Sub ezrd_int_ios Lib "ezreddll.dll" (ByRef ings as byte, ByRef outs as Byte, _
  ByRef da1 as Byte, ByRef da2 as Byte, ByRef an1 as Byte, ByRef an2 as Byte) as
integer
```

Elenco delle funzioni

Nome funzione	Parametri d'ingresso	Risultato	Descrizione
EZRD_OPEN	Numero di porta da utilizzare, es. "3" per "COM3"	Intero. Se 0, OK. Se diverso da 0, errore	Da richiamare prima di utilizzare qualsiasi altra funzione.
EZRD_CLOSE	Nessuno	Nessuno	Opzionale, serve per terminare il dialogo con la scheda driver.
EZRD_MODEL	Nessuno	Stringa NUL-terminata (stringa "C")	Opzionale, serve per conoscere il modello preciso del modulo EZ-Red
EZRD_INT_IOS	Vedere dettaglio	0=OK, oppure errore	Chiamata generale, multifunzione, che legge tutti gli ingressi e imposta tutte le uscite.
EZRD_SET_ALLOUTS	Maschera delle uscite	0=OK, oppure errore	Imposta contemporaneamente le 8 uscite di potenza.
EZRD_SET_IO	Uscita 1..8 e stato 1 o 0	0=OK, oppure errore	Imposta una singola uscita di potenza.
EZRD_UP1 ...fino a... EZRD_UP8	Nessuno	0=OK, oppure errore	Attiva (UP) l'uscita specificata.
EZRD_DN1 ...fino a... EZRD_DN8	Nessuno	0=OK, oppure errore	Spegne (DN) l'uscita specificata.
EZRD_SET_ANA1 ...e... EZRD_SET_ANA2	Valore da 0..255	0=OK, oppure errore	Imposta l'uscita analogica specificata in modo proporzionale: 0=0 volt, 255=10 volt.
EZRD_READ_AIN1 ...e... EZRD_READ_AIN2	Puntatore (indirizzo) della variabile da impostare	0=OK, oppure errore	Legge l'ingresso analogico 1 o 2, scrivendo nella variabile un valore da 0 a 255 proporzionale alla tensione d'ingresso.
EZRD_READ_IOS	Puntatore (indirizzo) della variabile da impostare	0=OK, oppure errore	Legge contemporaneamente tutti gli ingressi digitali, e ne pone la maschera nella variabile specificata.
EZRD_ISUP_IO	Numero d'ingresso da 1..8 e puntatore alla variabile da impostare	0=OK, oppure errore	Legge se l'ingresso specificato è a 1 o a 0.
EZRD_ISUP1 ...fino a... EZRD_ISUP8	Puntatore (indirizzo) della variabile da impostare	0=OK, oppure errore	Legge un ingresso e ritorna 0 o 1 nella variabile specificata.
EZRD_READENCODER	Puntatore (indirizzo) per i flag. Puntatore (indirizzo) per lettura	0=OK, oppure errore	Legge la posizione dell'encoder.
EZRD_PRESETENCODER	Nuovo valore int (long) da assegnare all'encoder	0=OK, oppure errore	Esegue il cosiddetto "preset" di un asse.
EZRD_SETWATCHDOG	Timeout (ms), configurazione uscite, valori per uscite analogiche 1 e 2.	0=OK, oppure errore	Configura o disattiva il watchdog di EZ-Red.

Dettagli delle funzioni

I paragrafi successivi spiegano in maggiore dettaglio il comportamento delle singole funzioni. La sintassi mostrata nel titolo è in Pascal (es. Delphi).

Tutte le funzioni che ritornano un risultato (quando contengono un segno di parentesi chiusa seguito dal due punti), ritornano il numero 0 per segnalare che il comando ha avuto successo, e un numero diverso da 0 per segnalare un codice di errore.

Sovente i parametri da passare alle funzioni sono interi a 32 bit senza segno. Quando i parametri devono essere passati per riferimento e non per valore, è perché essi vengono modificati dalla funzione in modo da raccogliere informazioni sullo stato. Il passaggio per riferimento si esegue con la parola chiave VAR in pascal, con la notazione "&" in C e con la parola BYREF in basic. La sintassi del C, a volte usata anche in altri ambienti di programmazione, è denotata talvolta come "passaggio di puntatore".

Una invocazione d'esempio in Pascal:

```
if EZRD_READ_IOS(ingressi)<>0 then writeln('Errore di comunicazione.');
```

mentre in linguaggio C diventa

```
if ( ezrd_read_ios(&ingressi) ) printf("Errore di comunicazione.");
```

In basic/VB:

```
If ezrd_read_ios(ingressi)<>0 Then print "Errore di comunicazione."
```

ezrd_open(comport: integer)

Serve per instaurare la comunicazione con il dispositivo EZ-Red. L'interfaccia USB crea due porte COM virtuali (riferirsi al numero della porta "A"); la DLL utilizza il numero di porta indicato per inviare e ricevere dati. Questa funzione va invocata una volta sola prima di poter accedere a qualsiasi altra. La libreria apre la porta indicata e interroga il dispositivo. Se tutte le fasi di aggancio vanno a buon fine, il risultato della funzione è zero. Se risulta impossibile aprire la porta (perché non esiste, è impegnata, o a causa di altri errori), il risultato è non-zero e riporta un codice di errore di Windows®. Se apertura della porta e comunicazione hanno successo, ma il dispositivo EZ-Red non risulta collegato, il codice di ritorno è 8.

ezrd_close

Serve per chiudere la comunicazione. Nessun risultato viene ritornato, e non è un errore chiudere la comunicazione se essa non era stata aperta. La porta COM in uso viene liberata. Qualsiasi ulteriore comando diverso da ezrd_open() provocherà un errore, dato che la comunicazione non è più disponibile.

ezrd_model: PChar (o char in "C")*

Questa è l'unica funzione che ritorna un risultato diverso da integer (intero a 32 bit). Serve per conoscere, in formato testuale, modello e versione del dispositivo EZ-Red. Il risultato fornito dalla funzione è un puntatore a una stringa *NUL-terminata* nel formato usato dal linguaggio C, comune in ambiente Windows®. Considerare questo puntatore di sola lettura; esso può anche essere NULL o puntare a un byte NUL: in entrambi i casi, significa che si è verificato un errore.

ezrd_int_ios(var ings, outs, da1, da2, an1, an2: byte)

Attraverso questa funzione è possibile interrogare tutti gli ingressi del modulo, e impostare tutte le uscite, con una chiamata unica. La funzione è abbastanza complicata: nei paragrafi successivi verranno illustrate funzioni più semplici.

Tutti i parametri sono interi di 8 bit senza segno (byte), e devono essere passati PER RIFERIMENTO (non per valore); il vantaggio di EZRD_INT_IOS è che con una sola chiamata è possibile eseguire qualsiasi operazione, o anche più operazioni contemporaneamente.

Prima di eseguire la chiamata occorre impostare le variabili nel seguente modo:

ings	maschera di bit per indicare quali uscite di potenza impostare (bit0=uscita 1 ... bit7=uscita8) I bit a 1 permettono di cambiare lo stato dell'uscita relativa, mentre se un bit è a zero, l'uscita relativa non viene modificata.
outs	maschera di bit per indicare lo stato voluto dell'uscita I bit considerati sono quelli corrispondenti nel parametro “ings”. Un bit di “outs” è ignorato se il bit corrispondente di “ings” è a 0.

Per esempio, per modificare solo l'uscita 1, occorre porre ings=1, e outs=1 (per accenderla) o outs=0 (per spegnerla). L'unico bit a 1 nella maschera INGS specifica che l'unica uscita che si vuole impostare è la 1; gli altri bit nella maschera OUTS verranno ignorati, e le relative uscite manterranno lo stato precedente. Per spegnere tutte le uscite usare INGS=255 e OUTS=0; per accenderle tutte usare INGS=255 e OUTS=255, e via dicendo. Similmente, le uscite analogiche DA1 e DA2 vengono abilitate dalla maschera AN2, in cui s'imposta il bit0 per DA1 e il bit1 per DA2. I valori da impostare a DA1 e DA2, se si desidera modificare le uscite analogiche, vanno da 0 a 255. Se non si vuole modificare alcuna uscita, ma solo leggere lo stato degli ingressi, basta porre INGS e AN2 a zero:

```
ings=0;  an2=0;     // nessuna uscita viene modificata
res=ezrd_int_ios(&ings, &outs, &da1, &da2, &an1, &an2);
```

Il seguente esempio modifica solo l'uscita analogica 1:

```
da1=128;            // circa 5 volt
ings=0;  an2=1;     // solo il bit0 di an2 a 1, per impostare l'uscita analogica 1
res=ezrd_int_ios(&ings, &outs, &da1, &da2, &an1, &an2);
```

Le righe seguenti spengono tutte le uscite digitali e portano a zero volt tutte le analogiche:

```
ings=255;           // tutti i bit da b0 a b7 messi a 1: “tutte le uscite”
outs=0;             // tutti i bit a 0: spegnere tutte le uscite digitali
an2=3;              // b0 e b1 alti; entrambe le uscite analogiche vengono scritte
da1=0;  da2=0;      // uscite analogiche a zero
res=ezrd_int_ios(&ings, &outs, &da1, &da2, &an1, &an2);
```

In qualunque caso, dopo l'esecuzione della chiamata, il valore delle variabili viene modificato per riflettere il nuovo stato di ingressi e uscite:

ings	riporta lo stato degli ingressi, un bit ciascuno: b0=ingresso 1; b7=ingresso 8
outs	riporta lo stato delle uscite digitali (un bit per ogni uscita)
da1	contiene, da 0 a 255, il valore del DAC (uscita analogica) 1
da2	contiene il valore del DAC per l'uscita analogica 2
an1	contiene il valore di tensione dell'ingresso analogico 1, da 0 a 255
an2	contiene il valore di tensione dell'ingresso analogico 2

Le funzioni descritte di seguito sono più semplici, però impegnano maggiormente la comunicazione tra computer e modulo EZ-Red. La velocità con cui il computer può inviare comandi al modulo è di circa 50 comando al secondo. Quindi, usare due comandi separati per impostare le uscite e leggere gli ingressi dimezza tale velocità, ma consente al software di controllo di essere più semplice.

ezrd_set_allouts(parm: cardinal)

Questa funzione imposta tutte le uscite digitali, con un bit per ogni uscita: il bit 0 corrisponde all'uscita 1, il bit 1 alla 2, e così via fino al bit 7 che corrisponde all'uscita 8. Si noti che tutte le uscite vengono comunque reimpostate - non c'è modo, usando questa funzione, di lasciare un'uscita nello stato che aveva in precedenza (a meno che lo stato non sia noto, e venga ribadito all'atto della chiamata).

ezrd_set_io(which1_8: cardinal; onoff: cardinal)

Questa funzione permette d'impostare una singola uscita digitale, lasciando invariate tutte le altre. Specificare, con un numero da 1 a 8 l'uscita da impostare, e con un valore 1 o 0 lo stato che l'uscita deve assumere. Il parametro ONOFF non è una maschera di bit - assume sempre e solo i valori 0 o 1.

ezrd_up1

Questa funzione imposta l'uscita digitale 1 al valore 1 (24 volt). Le altre funzioni simili: ezrd_up2, ezrd_up3 e così via fino a ezrd_up8, eseguono la stessa operazione sulle altre uscite digitali.

ezrd_dn1

Questa funzione spegne l'uscita digitale 1. Analogamente a quanto visto nel paragrafo precedente, esistono altre 7 funzioni dal nome simile, come ezrd_dn2, che agiscono sulle altre sette uscite.

ezrd_set_ana1(parm: cardinal)

Imposta, con un valore da 0 a 255, la tensione d'uscita del convertitore 1 (uscita analogica 1). La funzione ezrd_set_ana2 esegue la stessa operazione per l'uscita analogica 2.

ezrd_read_ain1(var parm: cardinal)

Legge la tensione presente sull'ingresso analogico 1, e la restituisce con un numero da 0 a 255. La funzione ezrd_read_ain2 serve per l'ingresso analogico 2.

ezrd_read_ios(var onoff: cardinal)

Questa funzione permette di leggere, in una sola chiamata, tutti gli ingressi digitali. Restituisce una maschera di bit con un 1 per ogni ingresso a 1 (+24 volt).

ezrd_isup_io(which1_8: cardinal; var onoff: cardinal)

Questa funzione permette di leggere lo stato di un singolo ingresso digitale; a differenza di quella descritta precedentemente, consente di evitare di eseguire calcoli su maschere di bit. Occorre passare (per valore) un numero da 1 a 8 per indicare l'ingresso da leggere, e un riferimento (byref o puntatore) a una variabile integer senza segno (32 bit) che conterrà, al ritorno della funzione, il valore 1 o 0 secondo

lo stato dell'ingresso specificato.

ezrd_isup1(var onoff: cardinal)

Questa funzione legge lo stato dell'ingresso 1, in modo analogo a quanto visto per la funzione precedente. Sono anche presenti le funzioni ezrd_isup2, ezrd_isup3..., per leggere gli altri 7 ingressi.

ezrd_readencoder(var flags, encoder: cardinal)

Permette di leggere la posizione dell'encoder, se questo è collegato agli ingressi veloci FX1 ed FX2. Insieme alla posizione dell'encoder viene fornito un insieme di flag (bit di stato):

Numero e nome	Maschera	Descrizione
1 WDTFIRED	and 2	Se ON, indica che il watchdog interno è intervenuto.
3 CYCLERUN	and 4	Indica che il PLC interno sta eseguendo un ciclo valido.
8 FX1	and 256	Stato ON/OFF dell'ingresso veloce FX1
9 FX2	and 512	Stato ON/OFF dell'ingresso veloce FX2
13 FLASHFAIL	and 8192	Indica che la memoria flash interna ha avuto un errore

ezrd_SetWatchdog(tmout, ioconf, aout1, aout2: cardinal)

Configura o disabilita il watch-dog di EZ-Red. TMOUT indica, in millisecondi (0..65000), il valore di timeout del watchdog; se impostato a zero, il watch-dog viene disabilitato. Diversamente, esso è abilitato: se EZ-Red non riceve almeno una comunicazione USB ogni TMOUT millisecondi, entra in condizione di allarme e pone le uscite secondo la configurazione impostata con questa funzione. IOCONF deve contenere la configurazione di bit delle otto uscite di potenza: nel bit 0 l'uscita 1, nel bit 1 l'uscita 2, e così via. Se IOCONF=0, in caso di watch-dog tutte le uscite di potenza vengono spente. Se IOCONF è uguale a 192 (bit 6 e 7 alti, impostazione di fabbrica), in caso di watch-dog le prime sei uscite diventano basse, e le ultime due alte. I valori di AOUT1 e AOUT2 (0..255) sono quelli da assegnare alle due uscite analogiche (0=0 volt, 255=10 volt).

XON ELECTRONICS SRL
WWW.XONELECTRONICS.IT
INFO@XONELECTRONICS.IT

Pagina internet del prodotto: http://www.xonelectronics.it/prodotti/industriali/EZ-Red

Si prega di segnalare errori o imprecisioni a web@xonelectronics.it