Technical Support
Bulletin No. 18 – ModBus Tips

Sommario

Chiarimenti

Di seguito apporteremo alcuni esempi di glossario al fine di poter dare a tutti la possibilità di comprendere e seguire il presente bollettino:

Numero **Binario**: il numero binario è una quantità espressa in base 2, ovvero si utilizzano solamente 2 caratteri (0 , 1). Questo sistema numerico è utilizzato dai calcolatori elettronici i quali interpretano il numero utilizzando un segnale logico (on , off) [Fig1]

Numero **decimale**: il sistema decimale è il sistema metrico che giornalmente si utilizza. Esso esprime una quantità utilizzando 10 caratteri, ovvero base 10 (0,1,2….8,9). Permette di fare calcoli algebrici più agevolmente

Numero **esadecimale**: il numero esadecimale esprime una quantità utilizzando 16 caratteri, ovvero base sedici, permette di lavorare più agevolmente con numeri grandi. Il sistema è preferito nell'ambito della programmazione. Si compone di (0,1,2,….,9,A,B,C,D,E,F). Per convenzione un numero espresso in esadecimale viene preceduto da H (es. HFF) oppure 0x (es. OxFF)

Bit: è l'abbreviazione di Binary Digit e definisce la quantità di informazione. Un bit è la minima quantità riconosciuta da calcolatore e rappresenta uno stato logico (Vero\falso; On\Off, etc) [Fig1]

Bit più \ meno significativo: a volte si parla di bit più significativo o MSB, il quale è il primo sulla sinistra del numero indicato. Esso è quello che ha più peso in termini di valore assoluto. Il bit meno significativo è il bit primo a destra ed è quello che a meno peso in termini di valore assoluto.

Byte: è la quantità necessaria di bit per definire un carattere alfanumerico. Ogni carattere è costituito da una sequenza di 8 bits. [Fig1]

Nibble: Questa unità di misura viene utilizzata per esprimere un insieme di 4 bits, ovvero la metà di un byte. Solitamente è utilizzato per descrivere ogni cifra esadecimale. Dal valore 0000 al valore 1111 è quindi rappresentabile da H0 ad HF [Fig1]

Word: Unità di misura che fissa la lunghezza di informazione a 16bits, oppure 2 bytes. Nei nostri controllori che utilizzano il protocollo ModBus-RTU per convenzione ogni informazione richiesta o spedita ha lunghezza di word o suoi multipli. [Fig1]

Server: Il dispositivo server è colui che in grado di richiedere informazioni o inviare comandi ai dispositivi che ad esso sono connessi attraverso un bus fisico come può essere un network EIA\TIA-485-A (comunemente detta RS485). Sono utilizzati anche sinonimi come Master o parent.

Client: Dispositivo connesso al bus dati e che risponde a richieste specifiche inviate dal server. Sono utilizzati anche sinonimi come Slave o sibling

Frame: il frame è il messaggio completo che viene scambiato fra il server ed il client

Broadcast: un messaggio trasmesso in broadcasting indica che non è diretto a un destinatario specifico ma destinato a chiunque sia in ascolto. Ad un messaggio broadcast non viene ritornata risposta.

Controllo di parità: Per prevenire errori di trasmissione dovuti a disturbi transitori viene utilizzato il controllo di parità. Viene perciò definito il parametro di parità sia nel server che nel client. Questo metodo non fa altro che aggiungere ad ogni byte spedito (8 bits) un ulteriore bit di parità. Se viene definita una parità ODD (dispari) ogni qualvolta si spedisce un byte viene aggiunto un bit a 1 o 0 in modo che il numero di valori logici alti sia dispari.
Es. Il byte che vogliamo spedire è 11011011 il quale contiene un numero pari (6 bits) di valori logici alti (1). Se definiamo che il controllo di parità è di tipo ODD (dispari) andremo ad aggiungere un bit a valore logico alto in modo che il totale passi da 6 a 7 bits a valore alto.

Bit di Start \ Stop: Solitamente viene richiesto di configurare almeno il numero di bits di stop in quanto ogni Server già definisce 1bit di start. Questi due bits servono per definire l'inizio e la fine del dato in una comunicazione seriale asincrona. Tale tipo di comunicazione invia i bits in sequenza uno dietro l'altro, per cui chi riceve deve saper dove inizia e finisce il pacchetto dati. Spedire un bit significa portare la linea seriale a valore logico alto (1) o tenerla a valore logico basso (0) per un tempo definito. Questo tempo definisce la velocità di trasmissione, o baudrate, della seriale.
Ad esempio in una seriale RS485 ove sia ha un Baudrate di 9600bit\sec (caratteristico nei nostri controlli) si ha che in un secondo vengono trasmessi 9600 bit. Il tutto implica che un bit ha una durata caratteristica di 0,1ms circa (1 sec \ 9600 bits)
Il meccanismo di trasmissione si traduce in questo. Il server porta a valore logico alto la seriale per il tempo di un bit (bit di start), successivamente invia gli 8 bits che formano il byte dato, e segue poi il bit di parità. In infine la linea seriale rimane ancora alta per il tempo di 1\2 bits (1 o 2 bit di stop)

Time-out: Un altro parametro da definire nel Server è il time-out ovvero il tempo d'attesa di una risposta da parte del client. Prima di intraprendere un'altra azione. Il time-out è funzione della velocità di trasmissione e del tempo utile impiegato dal client ad elaborare la richiesta e formulare la risposta.
Il tempo di invio del pacchetto dati su di una seriale a 9600bit\sec è:

$$(0,1 \text{ X } 11) \text{ X } 20 = 20\text{mS}$$

Dove 0,1 è il tempo necessario per definire un singolo bit, 11 è il numero caratteristico di bits da spedire per ogni singolo byte (1bit start + 8 bit dato + 1bit parità +1 bit stop) , mentre 20 è il numero di byte di un messaggio generico (potrebbe essere superiore in accordo con la quantità di dai richiesti\spediti)
Segue quindi che per inviare il messaggio il server impiega 20ms, altrettanti se non superiori servono per l'invio della risposta da parte del server.
Il time out quindi viene impostato considerando

il tempo di andata richiesta + tempo elaborazione + tempo ritorno risposta + tempo di silenzio

Per i nostri controlli solitamente un time-out tipico potrebbe essere di 200mS.

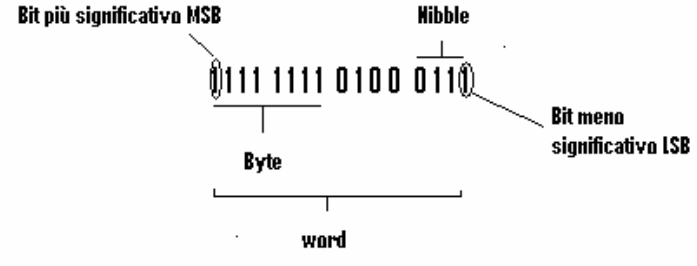
ModBus-RTU: Quando viene utilizzato il metodo di trasmissione RTU, i dati vengono scambiati in formato binario dove ogni byte di informazione è codificato in un byte di comunicazione. L'inizio dei messaggi viene preceduto da un silenzio di 3,5 bits. Se il client identifica un silenzio di 1,5 bits presume che un nuovo messaggio stia per arrivare e cancella il buffer di ricezione. Questo implica che il messaggio deve essere trasmesso in maniera continuativa.

ModBus-ASCII:, Quando viene utilizzato la comunicazione di tipo ASCII, tutti i messaggi sono codificati in valori esadecimali i quali rappresentano un carattere leggibile ASCII. I caratteri utilizzati sono da 0 a 9 e da A a F. Per ogni byte di informazione 2 sono i caratteri necessari da trasferire in quanto ogni carattere definisce solo 4 bits di comunicazione nel sistema esadecimale. Utilizzando questo tipo di comunicazione vengono inviati dei caratteri per stabilire l'inizio e la fine del frame il carattere ascii " **:** " viene utilizzato come inizio ed ogni messaggio viene terminato con il carattere "Cr|LF" Il vantaggio del metodo Ascii è che tollera un attesa fra bytes fino ad un secondo

Area Fisica: La memoria Ram è suddivisa in diversi settori chiamati aree che hanno un proprio indice reale. Non è un dato necessario all'utente che necessiti di accedere ad una risorsa.

Area logica: Il firmware gestisce la ri-allocazione dei settori fisici della RAM. Li ri-organizza in modo da ottimizzare il processo di calcolo. All'utente sono necessari gli indirizzi logici al fine di accedere alla risorsa di interess

(fig1) Esempio di numero binario:



Comandi implementati.
Nei controllori Eliwell attualmente sono implementati solo alcuni comandi previsti dal protocollo ModBus ed in particolare sono il comando 3 e 16 intensi in decimale oppure 0x03 e 0x10 se intesi in esadecimale.
I comandi citati 3 e 16 sono implementati in tutti i controlli Eliwell che integrano il ModBus-RTU indistintamente dal modello. In solo alcuni controllori sono disponibili altri comandi come il 43 ed il 4, ovvero 0x2B ed 0x04

Comando 0x03: Il comando definisce la lettura di registri multipli consecutivi (holding registers). Il blocco definito dal protocollo come Holding registers, sono dei dati che possono essere variati dal programma applicativo ed hanno dimensione di 16 bits ovvero 1 word.

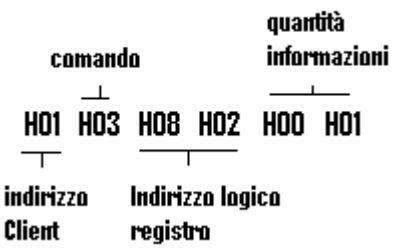
Comando 0x04: Il comando definisce la lettura multipla di input forniti da un sistema di inputs\outputs. In quei sistemi dove la memoria è organizzata in blocchi, con diversi comandi si accede direttamente alla risorse necessarie. Tutti i controlli Eliwell che si pongono come client rispetto alla rete, non integrano questo comando in quanto tutte le risorse vengono allocate in un unico blocco in memoria e gestite come holding registers. L'unico controllore che integra questo comando è l'XT Pro in quanto esso può porsi come Client rispetto alla rete oppure essere lui stesso il master che richiede informazioni ed invia comandi.

Comando 0x10: Il comando definisce la scrittura di registri multipli e contigui (holding registers) Come prima la dimensione dei dati e di 16 bits ovvero 1 word.

Comando 0x2B/0x0E:Il comando definisce la sola lettura della descrizione o identificativo del dispositivo Client. Esso permette di acquisire dal dispositivo informazioni come identificativo del costruttore, codice prodotto. I comando si compone di sotto comandi dei quali alcuni sono obbligatori per i costruttori che implementino il comodano 43 (0x2B), altri invece sono facoltativi.
I sottocomandi obbligatori sono 0x00, 0x01,0x02. Nell'ordine essi restituiscono l'identificativo del produttore, identificativo prodotto, versione. Nel caso dei nostri controlli ove questo comando sia implementato restituiscono INVENSYS come ID produttore, Maschera firmware come Id prodotto, e release come versione.

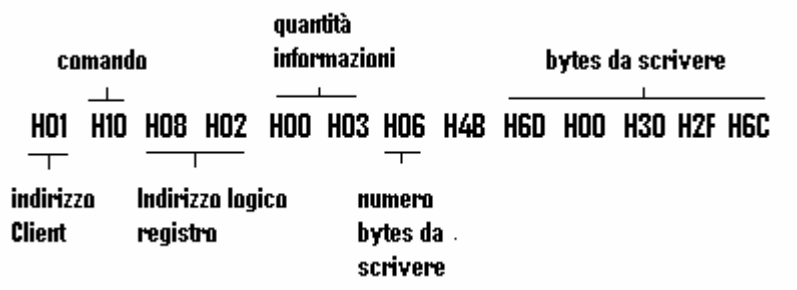
Esempio di comandi o frames
Prendiamo ad esempio il messaggio di lettura inviato dal server verso il client con indirizzo.1
Si fa presente che l'indirizzo 0 nel protocollo ModBus è riservato per messaggio di broadcasting

Esempio di lettura registro (0x03):



Nel dettaglio il primo byte definisce l'indirizzo del dispositivo (H01). Il secondo byte definisce il comando di lettura (H03)
Il terzo ed il quarto byte definiscono nell'ordine la parte alta e bassa della word che indirizzano il registro da leggere.
Il quinto ed il sesto byte definiscono rispettivamente la parte alta e bassa della word che definisce la quantità di informazioni. L'esempio ci dice che partendo dall'indirizzo logico 0x08 0x02 andrò a leggere un solo registro.
Se il sesto byte fosse ad esempio 0x03 anziché 0x01, significa che partendo dall'indirizzo 0x08 0x02 andrò a leggere 3 registri contigui.

Esempio di scrittura registro (0x10)



Come prima il primo byte indica l'indirizzo del client a cui il messaggio è diretto, mentre il secondo byte indica il comando. Vediamo quindi che entrambi hanno dimensione di un byte.
Il terzo e quarto byte (H08 H02) indicano la parte alta e bassa della word del registro di partenza su cui scrivere. Si vede che un registro ha dimensione di 2 bytes (word).
Il quinto ed il sesto byte indica il numero di registri consecutivi da scrivere. In questo caso sono 3 i registri da scrivere (H00 H03).
Il settimo byte (H06) indica il numero di bytes che seguono e che verranno scritti. Questo numero risulta essere sempre il doppio del numero che lo precede nel frame essendo ogni registro di dimensione pari a 2 bytes,

Calcolo dell'area logica
In alcuni manuali non vengono elencati gli indirizzi logici dei registri, ma per ogni area viene indicato il suo offset e l'indirizzo fisico della risorsa.
Ad esempio nella serie Ech200B, gli ingressi digitali si trovano in area RAM che ha indice 8, la quale ha un offset fisico di 0x4F. Per gli ingressi digitali, i quali vengono accumulati in una unica word, l'indirizzo fisico dichiarato è 0x63
Per ottenere l'indirizzo logico da leggere la procedura da fare è la seguente.

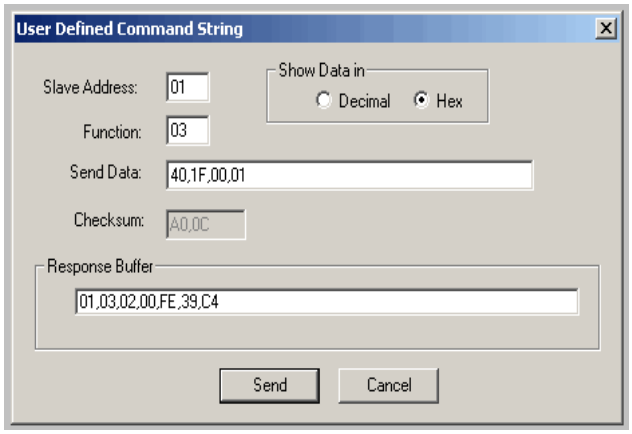
Dato che:
1. l'indirizzo del registro viene espresso da una word, ovvero 16bits
2. L'indice dell'area viene espressa nella word utilizzando i 5 bits più significativi
3. l'indirizzo logico del registro viene espresso utilizzando i restanti 11bits

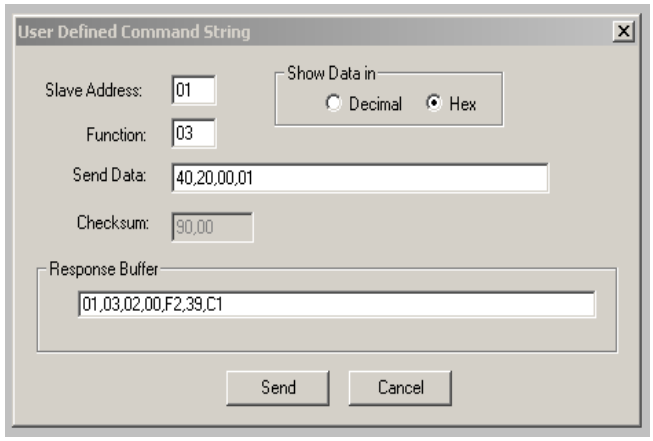
L'indice dell'area nel nostro caso è 8, ovvero in binario 01000 (5 bits)
L'indice logico è h63 – h4F, ovvero h14. L'indirizzo logico ottenuto in binario equivale a 00000010100

Messo tutta assieme 01000 00000010100, il che diventa in esadecimale 0x 4014
Il nostro registro ha indirizzo logico h4014

Lettura di variabile con segno

Per rappresentare numeri interi negativi si usa il cosiddetto complemento a 2.
Portiamo un esempio e supponiamo di voler rappresentare usando n bits un numero senza segno. Sappiamo che con n bits possiamo rappresentare un numero decimale contenuto nell'insieme da 0 a 2^n -1.
Supponendo che abbiamo a disposizione 3 bits, ne consegue che posso rappresentare un numero decimale da 0 a 7. Volendo fare il passo successivo e rappresentare un numero negativo dobbiamo utilizzare il bit più significativo per indicare il segno. Ne consegue che i numeri positivi possono essere rappresentati nell'intervallo da 0 a [2^{n-1}], mentre il valori negativi possono essere rappresentati nell'intervallo da [-2^{n-1}] a [-1]. l'intervallo totale è quindi definito all'interno di [2^{n-1}] e [2^{n-1} -1].
Consideriamo ancora di avere 3 bits, se il più significativo è destinato al segno ne conseguee che posso rappresentare valori decimali contenuti fra -4 e 3.
Nella rappresentazione in complemento a 2 il bit più significativo è definito anche bit di segno
Analizziamo ora la lettura di un valore con segno come la sonda St1. Sappiamo che il suo valore è definito all'interno di una word, ovvero 16 bits. Da questo ricaviamo che i valori interi rappresentabili sono -32768 e 32767.





Nell'esempio in figura sopra vediamo che la sonda St1 ha valore 0x FEF2, ovvero 1111111011110010.
Vediamo che il bit piu significativo si trova a valore alto (1) per cui sappiamo che il numero è negativo.
A questo punto complementiamo ad 1 il numero binario (funzione logica NOT) ed al risultato aggiungiamo1

Complemento ad 1 : 0000000100001101
+ 1

0000000100001110

Il valore assoluto ottenuto ed espresso in decimale è 270. Sappiamo che il valore va inteso negativo e ne consegue che la sonda St1 vale -27,0 °C

Lo stesso risultato si ottiene se prendo il valore FFFF, ovvero 1111111111111111, e gli sottraggo il valore FEF2 (vedi sopra) e sommo 1.

```
1111111111111111
1111111011110010
               +     1
 -------------------------
0000000100001110
```

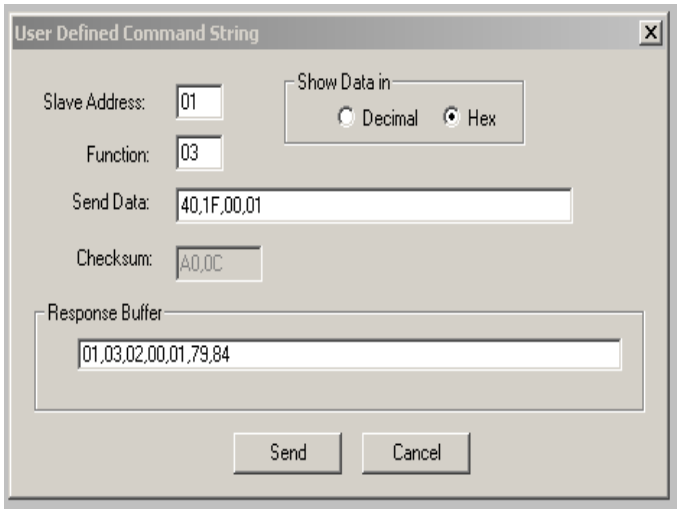
Risulta essere quindi più semplice ricavarsi la formula

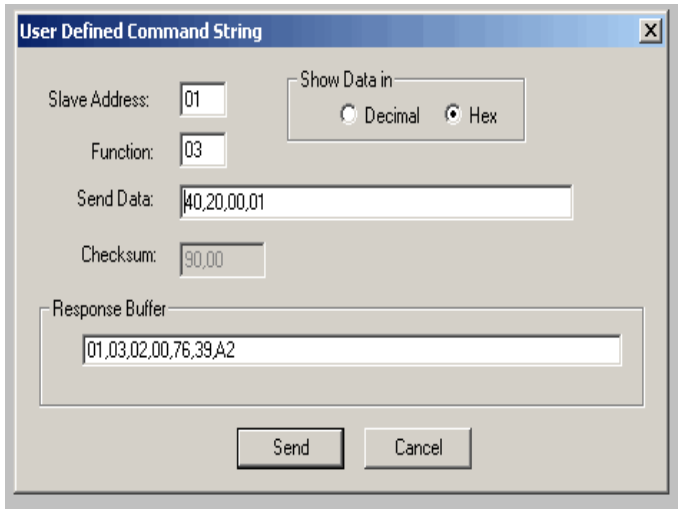
(FFFF- Value) +1

Esempi di comando con Ech200B

1. **Lettura sonda**

Il primo esempio riporta la lettura della sonda di regolazione St1. In questo controllore il valore delle sonde è diviso su 2 words. È stato preso volutamente l'Ech200B perché è l'unico prodotto Eliwell ad avere questa variante per la lettura sonda suddivisa su 2 words. Ad esempio la sonda St1 si trova agli indirizzi fisici 6E ed 6F, dove nel 6E trovo il byte piu significativo e 6F il byte meno significativo.

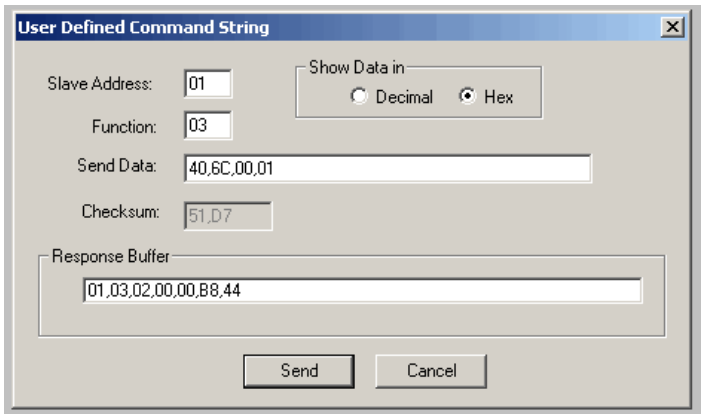


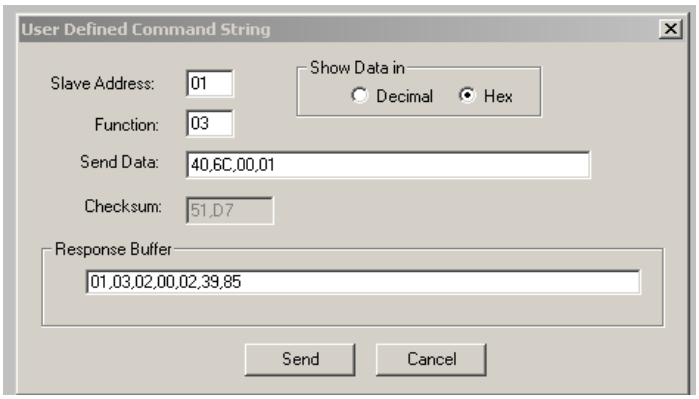


All'indirizzo logico 0x4020 il controllore risponde con una word. Prendo la parte bassa è la utilizzo come byte meno significativo per la sonda., mentre all'indirizzo 0x401F prendo la parte bassa e la utilizzo come byte più significativo. Mi trovo quindi ad avere una word composta da 0x01 e 0x76 cioè la mia sonda vale in esadecimale 0176, ed in decimale 374. ne consegue che il valore letto in °C è di 37,4 °C..

2. **Lettura allarme**

Andiamo ora a vedere la lettura di un allarme come l'alta pressione digitale, ovvero intervento pressostato.
Gli allarmi come le sonde di trovano in RAM quindi area 8 .
Essendo gli allarmi attivi o non, essi sono raggruppati in words e descritti dallo stato di un bit alla specifica posizione indicata dal manuale. Ad esempio l'allarme di alta pressione si trova all'indirizzo fisico 0xBB ed è descritto dallo stato del bit di indice 1 (bit_1)
Come prima dobbiamo ricavare l'indirizzo logico. In questo caso l'indirizzo è 0x406C.

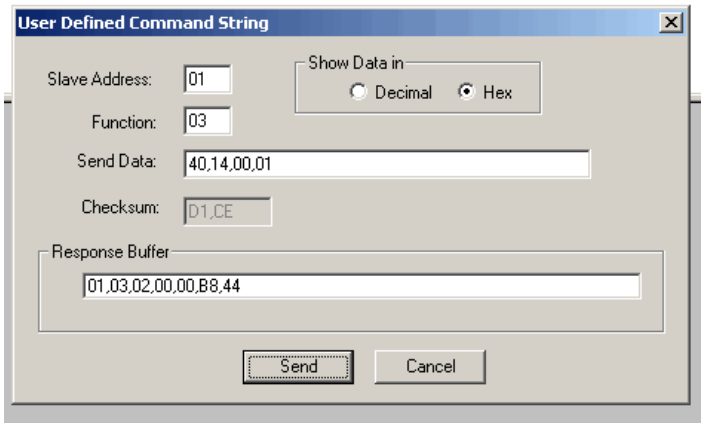


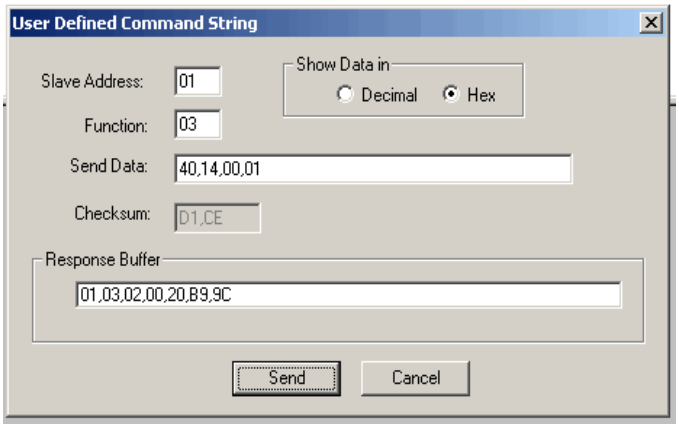


Nell'immagine di sinistra vediamo che non ci sono allarmi presenti all'indirizzo, perché il controllo ci risponde con una word di valore 0x 0000. Nell'immagine di destra invece il regolatore ci risponde con una word di valore 0x 0002.
Tradotto in valore binario la word vale 0000 0000 0000 0010.
Da questo vediamo che il bit con indice 1 (bit_1) si trova a valore logico alto e ne consegue che è presente l'allarme di alta pressione digitale.

3. Lettura stato ingresso digitale

Lo stato degli ingressi digitali è riportato in area RAM (area 8) . Vi sono riportati gli statti dei 5 ingressi previsti ed inoltre anche lo stato delle sonde se configurate come digitali.
Andiamo a vedere ad esempio l'ingresso digitale 5, il quale si trova all'indirizzo logico 0x4014.



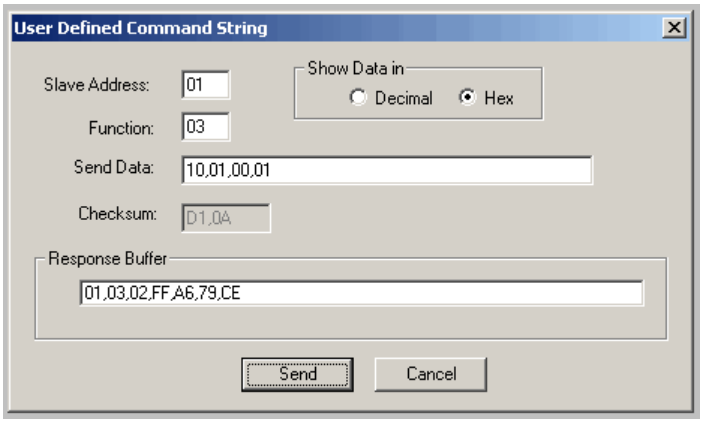


Nell'immagine di sinistra vediamo che il controllore risponde al comando con una word che ha valore 0x 0000, da cui ricaviamo il valore relativo valore binario 0000 0000 0000 0000.
Lo stato del digitale 5 si trova, come possiamo ricavare dal manuale , all'indice 5 (bit_5) da cui ricaviamo lo stato del digitale. In questo caso il valore logico del bit 5 è basso il che significa che l'ingresso è aperto.
Nell'esempio di sinistra il dispositivo risponde con una word di valore 0x 0020, che riportato in binario vale 0000 0000 0010 0000.

Esempi di comando con Ert400B

1. **Lettura sonda**

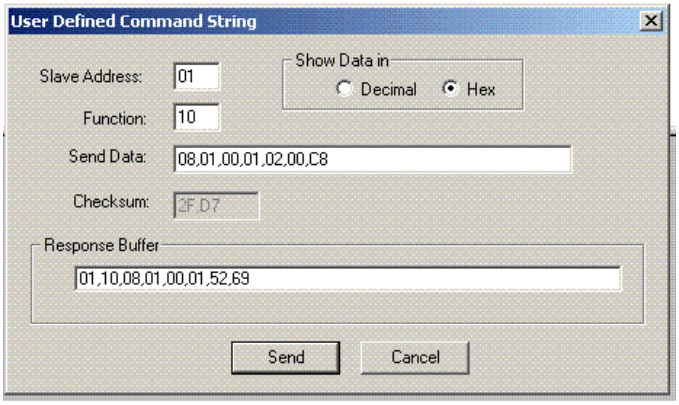
Prendiamo ad esempio ora la lettura della sonda di regolazione, la quale ha valore negativo per rendere più esaustiva la prova.
Come vediamo dal manuale dell'ERT 400 ci vengono forniti gia direttamente gli indirizzi logici delle risorse e non si rende necessario tenere in considerazione l'offset dell'area.
Nel esempio considerato la sonda St1 di regolazione si trova all'indirizzo logico 4097, ovvero 0x10 01. l'intero valore sonda è, in questo controllore, ha lunghezza di una word ma i byte sono contigui.
Non si rende perciò necessario leggere due words e mettere assieme i due bytes meno significativi.



Il controllo ci risponde con un valore 0xFF A6, che in decimale è uguale 65446.
Come abbiamo visto precedentemente per ottenere il modulo del valore sonda dobbiamo complementare a due, ovvero (FF FF – FF A6) +1 = 5A
Il valore assoluto della nostra sonda è di 90 °C\10. La temperatura di regolazione è quindi -9,0°C

2. **Scrittura variabile**

Vediamo ora un esempio di scrittura variabile utilizzando il setpoint. Supponiamo di voler cambiare il valore del set cooling e di volerlo portare a 20.0°C.



La variabile Set cooling si trova all'indirizzo logico 2049, ovvero in 0x 08 01.
Andremo ad'utilizzare il comando di scrittura 0x10.
Il frame si compone come:

01: indirizzo di rete del client
10: comando di scrittura
0801: area logica ove è locato il setpoint cooling
0001: numero di variabili da scrivere
02: numeri di bytes da scrivere
00C8: valore in esadecimale da scrivere, ovvero 200 (20,0°C)

DECLINAZIONE DI RESPONSABILITA’

La presente pubblicazione é di esclusiva proprietà della Eliwell la quale pone il divieto assoluto di riproduzione e divulgazione se non espressamente autorizzata dalla Eliwell stessa.

Ogni cura é stata posta nella realizzazione di questo documento; tuttavia la Eliwell non può assumersi alcuna responsabilità derivante dall’utilizzo della stessa.

Lo stesso dicasi per ogni persona o società coinvolta nella creazione e stesura di questo documento.

La Eliwell si riserva il diritto di apportare qualsiasi modifica, estetica o funzionale, senza preavviso alcuno ed in qualsiasi momento.

Eliwell Controls s.r.l.
Via dell’Industria, 15 • Zona Industriale Paludi • 32010 Pieve d’Alpago (BL) ITALY
Telephone +39 0437 986 111 • Facsimile +39 0437 989 066
Sales +39 0437 986 100 (Italy) • +39 0437 986 200 (other countries)
Email: saleseliwell@invensyscontrols.com
Technical helpline +39 0437 986 300 • E-mail techsuppeliwell@invensyscontrols.com
www.eliwell.it