DUEMMEGI

SISTEMI DI AUTOMAZIONE INDUSTRIALE

CONTATTO

MCP - Modulo di Controllo Programmabile

Manuale d'Uso per MCP ver. 4.x e MCP Plus

DUEMMEGI s.r.l. - Via Longhena, 4 - 20139 MILANO
Tel. 02/57300377 - FAX 02/55213686

Rel.: 4.3
Aprile 2001

I1- Variazioni riportate in questo manuale (rel.4.3) rispetto alla precedente versione (rel.4.2)

- Aggiunti maggiori dettagli su MAPPA RAM di MCP
- Aggiunti maggiori dettagli ed esempi su protocollo MODBUS
- Inserita Appendice C con tabelle per localizzare i punti virtuali in RAM

Variazioni del manuale rel.4.2 rispetto alla rel.4.1

- Si fa riferimento anche a MCP Plus
- l'indirizzo di MCP è un numero tra 1 e 255; il numero massimo di MCP collegabili sulla stessa linea RS485 è 31
- aggiunta direttiva PROTOCOL
- aggiunti schemi e descrizione MCP Plus
- aggiunto paragrafo su porte seriali di MCP Plus
- aggiornate caratteristiche tecniche
- aggiunte dimensioni di ingombro MCP Plus
- il protocollo "standard" (o proprietario) di MCP è stato denominato FXP
- aggiunti maggiori dettagli sull'implementazione del driver
- aggiunta appendice B su protocollo MODBUS di MCP Plus

I2- Cosa cambia tra MCP standard (versione 4.x) e MCP Plus

MCP Plus è il nuovo modulo di controllo programmabile della linea Contatto DUEMMEGI. Il contenitore è modulare di **dimensioni 9M** (come la versione standard). Le differenze rispetto alla versione standard sono le seguenti:

- l'uscita per il display seriale è stata eliminata
- è stata aggiunta una **porta RS485** per il collegamento di più MCP Plus in multi-drop
- sia la porta RS232 che quella RS485 sono **galvanicamente isolate** dal resto dei circuiti (non sono però isolate tra di loro) senza richiedere alcuna alimentazione supplementare esterna
- oltre al protocollo MCP standard (FXP), sono stati integrati i **protocolli MODBUS e JOHNSON CONTROL**, eliminando così la necessità di convertitori di protocolli esterni

Nota: le porte RS232 e RS485 sono mutuamente esclusive: la commutazione da una porta all'altra avviene tramite il segnale di ingresso DSR sul connettore RS232.

INDICE

1- INTRODUZIONE

MCP è un modulo di controllo per il sistema "bus" Contatto. MCP è acronimo di **M**odulo di **C**ontrollo **P**rogrammabile, in quanto la gestione del sistema può essere stabilita dall'utente secondo regole, dette equazioni, che legano le uscite agli ingressi.

Per utilizzare al meglio MCP è necessario utilizzare un software di supporto allo sviluppo dei programmi ed alla messa in servizio; questo programma lavora su Personal Computer in ambiente WINDOWS® ed è denominato MCPTOOLS. Per l'uso di questo programma si rimanda alla documentazione relativa.

Essenzialmente il software di supporto MCPTOOLS comprende:

- un editor di testi per la scrittura del programma
- un compilatore che consente di tradurre un file ASCII contenente le equazioni di funzionamento in un file binario adatto ad essere trasferito nella memoria non volatile (di tipo FLASH) del modulo MCP
- un simulatore per la verifica del programma, o di parte di esso, prima che questo venga trasferito nella memoria di MCP
- una sezione che gestisce il trasferimento del programma a MCP (o viceversa)
- un visualizzatore di mappa, ossia la rappresentazione grafica dello stato dei moduli in campo (ingressi e uscite), stato dei contatori, mappa dei punti virtuali ecc.

Nel seguito di questo manuale sono descritti gli operatori disponibili per le equazioni e la relativa sintassi; viene invece sottintesa la conoscenza, da parte del lettore, del sistema Contatto. ***In questo manuale ci si riferirà a MCP versione 4.x e a MCP Plus semplicemente con la dicitura MCP*** (salvo dove non diversamente specificato)

ATTENZIONE: ***questo manuale si riferisce a MCP versione 4.x e a MCP Plus. Per verificare la versione del modulo MCP in Vostro possesso, controllare l'etichetta gialla posta sul retro del contenitore: i primi tre caratteri del codice stampigliato indicano la versione; per esempio F445100N indica la versione 4.1. Per un corretto uso di questa versione tenere presente quanto segue:***

- ***un programma scritto per versioni precedenti di MCP è generalmente compatibile con la versione 4.x***
- ***un programma scritto per la versione 4.x potrebbe non essere compatibile con versioni precedenti***

IMPORTANTE: ***i programmi scritti per MCP versione 4.x o per MCP Plus DEVONO essere compilati con la release 4.x del compilatore integrato nel software di supporto MCPTOOLS.***

PRUDENZA: ***MCP contiene una batteria ricaricabile NiCd o NiMH: siate sicuri di rimuovere la batteria prima di gettare il modulo. La batteria va eliminata in modo sicuro, secondo le leggi vigenti nel rispetto dell'ambiente.***

1.1- Considerazioni generali

Il modulo CONTATTO MCP può processare lo stato e le variazioni degli ingressi per comandare le uscite o per eseguire funzioni particolari, quali, per esempio, il conteggio di impulsi oppure il controllo di un dispositivo seriale per la visualizzazione di messaggi.

Sono a disposizione inoltre **1000 punti virtuali**, che si differenziano dai punti reali (chiamati d'ora in poi semplicemente ingressi o uscite), in quanto non sono associati a nessun punto reale in campo, ma sono solamente il risultato di combinazioni di ingressi (reali e/o virtuali); tale risultato viene memorizzato nella memoria RAM del modulo MCP.

I punti virtuali consentono quindi di creare variabili d'appoggio per particolari funzioni ed equazioni. I punti virtuali possono essere considerati e gestiti come ulteriori ingressi (virtuali) ed essere utilizzati per controllare uscite reali o ulteriori punti virtuali; per queste ragioni, nel seguito di questo manuale, i termini **"punto virtuale"**, **"ingresso virtuale"** e **"uscita virtuale"** saranno considerati equivalenti.

I punti virtuali sono inoltre obbligatori per alcune funzioni specifiche come verrà descritto nei relativi paragrafi che seguono.

MCP versione 4.x **permette di definire punti virtuali in funzione di altri punti virtuali** senza limiti di concatenamento.

In realtà i ***punti virtuali a disposizione sono 992***, in quanto i seguenti sono riservati come segue:

- **V1000**: diventa attivo quando MCP rileva uno o più moduli non funzionanti (corrisponde allo stato del led MOD.F su MCP)
- **V999**: diventa attivo quando MCP rileva una condizione di bus guasto (corrisponde allo stato del led BUS.F su MCP)
- **V998**: diventa attivo quando MCP termina la fase di inizializzazione che fa seguito ad una riprogrammazione o quando si accende l'alimentazione
- **V997**: è un punto virtuale che cambia stato ogni 0.5 secondi; questo punto può quindi essere utilizzato come base dei tempi per implementare ad esempio funzioni di lampeggio, per fare aumentare o diminuire il contenuto di un contatore a intervalli regolari di 1 secondo, ecc.
- **V996**: riservata
- **V995**: riservata
- **V994**: quando è attiva MCP considera, per il calcolo delle fasce orarie, l'orario corrente letto dall'orologio interno + 1 ora; in questo modo è possibile commutare tra ora solare (V994=0) e ora legale (V994=1). Anche quando viene richiesta l'ora dalla porta seriale (mediante MCPTOOLS o altro programma), la risposta di MCP sarà in accordo con lo stato della V994
- **V993**: riservata

Questi punti virtuali riservati, con l'eccezione di V994, possono essere utilizzati come punti di "sola lettura", vale a dire possono essere impiegati solo come ingressi di equazioni, ad esempio quando si voglia segnalare, attraverso una uscita reale di un modulo oppure via modem, la presenza di uno o più moduli guasti.

Nota: *se si utilizza il punto virtuale V994 per gestire ora legale/solare, è obbligatorio che l'orologio interno di MCP venga sempre impostato sull'ora solare; infatti la V994 non cambia le impostazioni dell'orologio interno, ma semplicemente forza il calcolo di un'ora in più quando vengono valutate le equazioni contenenti fasce orarie oppure quando viene richiesta l'ora attraverso la linea seriale RS232.*

2- EQUAZIONI: TIPOLOGIE E SINTASSI

Le equazioni possono essere suddivise in:

1. Equazioni logiche
   - L'uscita dipende da un ingresso o dalla combinazione di più ingressi, reali e/o virtuali, in serie o in parallelo, eventualmente complementati
2. Equazioni con operatori semplici
   - Gli operatori base sono **RESET** e **SET**, utilizzati per il controllo di pulsanti e la simulazione di relè MARCIA/ARRESTO
3. Equazioni con operatori speciali.
   - Gli operatori speciali consentono l'implementazione di funzioni più complesse quali il conteggio di impulsi, la gestione di visualizzatori di testi, timer, ecc.

2.1- Equazioni logiche

Le equazioni logiche sono le più semplici e consentono la combinazione di ingressi per controllare una particolare uscita (reale e/o virtuale).

Non vi sono limiti, se non nella capacità di memoria, al numero di ingressi che possono essere combinati tra loro. Il sistema CONTATTO consente un massimo di 1016 ingressi e **tutti possono essere utilizzati per il controllo di un'uscita**. Inoltre **lo stesso ingresso può controllare più uscite**.

Le equazioni nella loro sintassi generale sono di due tipi:

```
Ox.y = f ( Ij.k, Vn )

Vn = f( Ij.k, Vn )
```

dove `Ox.y` e `Vn` indicano l'uscita da controllare e `f( Ij.k, Vn )` è la combinazione degli ingressi reali e/o virtuali che controllano l'uscita.

Le notazioni **`x.y`** e **`j.k`** (o altre simili utilizzate nel seguito) stanno ad indicare l'indirizzo del modulo (**`x`** e **`j`**) ed il relativo punto (**`y`** e **`k`**). Per esempio, **`Ox.y`** indica l'uscita **`y`** (compresa tra 1 e 8) del modulo avente indirizzo **`x`** (compreso tra 1 e 127). Allo stesso modo **`Ij.k`** indica l'ingresso **`k`** del modulo di indirizzo **`j`**.

La funzione che definisce una uscita può comprendere uno o più ingressi reali e/o virtuali legati tra loro da operatori AND (simbolo **`&`**) e da operatori OR (simbolo |); è inoltre possibile ribaltare la logica di un ingresso facendolo precedere dal simbolo **`!`** (operatore NOT o di negazione).

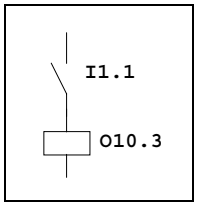
L'operatore **`&`** equivale, nella notazione elettromeccanica, alla serie di contatti, mentre l'operatore | equivale al parallelo. L'operatore di negazione corrisponde invece a sostituire un contatto normalmente aperto con un contatto normalmente chiuso.

I seguenti esempi mostrano l'equivalenza tra una combinazione logica di MCP e lo schema elettromeccanico.

Esempi:

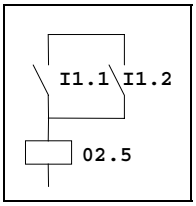
`O10.3 = I1.1`

Questa è l'equazione più semplice, che lega un ingresso (1 del modulo 1) ad un'uscita (3 del modulo 10).



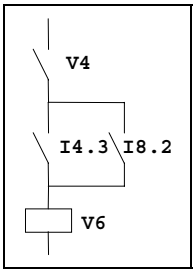
`O2.5 = (I1.1 | I1.2)`

L'equazione rappresenta il parallelo di due ingressi. Le parentesi in questo caso sono facoltative; esse possono essere usate per una maggiore leggibilità.



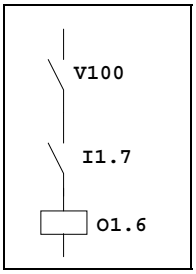
`V6 = (I4.3 | I8.2) & V4`

L'equazione rappresenta il parallelo di due ingressi in serie ad un ulteriore ingresso; l'uscita è, ad esempio, virtuale. In questo caso le parentesi hanno un significato ben preciso e, come descritto più avanti, sono necessarie.



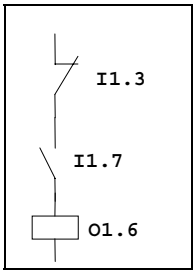
`O1.6 = V100 & I1.7`

L'equazione rappresenta la serie di due ingressi, di cui uno virtuale ed uno reale.



`O1.6 = !I1.3 & I1.7`

L'equazione rappresenta la serie di due ingressi di cui uno negato; ciò significa che il contatto normalmente aperto collegato al punto `I1.3`, viene interpretato come normalmente chiuso.

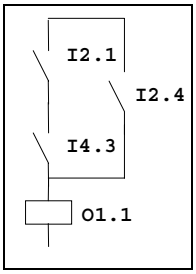


La precedenza degli operatori `&` e `|`, nel caso siano presenti entrambi, è la seguente:

1. Operatore `&`
2. Operatore `|`

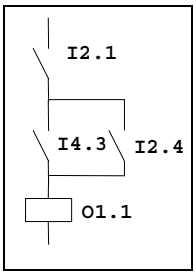
Tale precedenza può essere modificata utilizzando le parentesi tonde `(` e `)`.

Esempi:



`O1.1 = I2.1 & I4.3 | I2.4`

L'equazione è il parallelo di due termini: `I2.1 & I4.3` e `I2.4.`
Notare la differenza con l'esempio che segue.



`O1.1 = I2.1 & (I4.3 | I2.4)`

L'operatore | viene valutato prima dell'`&`. L'equazione è equivalente a:
`O1.1 = I2.1 & I4.3 | I2.1 & I2.4`
Il termine serie è cioè comune ad entrambi i termini parallelo. Notare la differenza con l'esempio precedente.

In fase di valutazione dell'equazione viene considerato lo stato dell'ingresso corrente. Come già ricordato, è possibile invertire la logica di un ingresso, sia esso reale o virtuale, utilizzando l'operatore NOT (`!`, punto esclamativo). In tal caso viene considerato il complemento del rispettivo ingresso o termine.

Esempi:

`O1.1 = !I2.4` L'uscita viene disattivata quando `I2.4` viene attivato.

`O1.1 = !(I2.4 & I4.5)` L'uscita viene disattivata quando entrambi gli ingressi sono attivi e attivata quando almeno uno di essi è non attivo. L'equazione è equivalente a: `O1.1 = !I2.4 | !I4.5`.

Nota: come si vede dall'ultimo esempio mostrato, la negazione di una intera equazione composta da una combinazione di ingressi, equivale a negare ogni singolo ingresso e a scambiare gli operatori `&` con | e viceversa. Tale regola è propria dell'algebra booleana ed è sempre applicabile in equazioni di questo tipo.

2.2- Equazioni con operatori semplici

Gli operatori semplici sono due:

1. **RESET**
2. **SET**

Tali operatori sono applicabili ad ingressi (reali o virtuali) e permettono di implementare funzioni specifiche.

L'operatore **SET**, indicato dal simbolo **S**, e l'operatore **RESET**, indicato dal simbolo **R,** trovano largo impiego in tutti quei casi ove si voglia implementare in modo semplice una sequenza di **MARCIA-ARRESTO**.
Esso opera sull'uscita attivandola quando l'ingresso di **SET** diventa attivo e mantenendola attiva anche dopo che lo stesso ingresso è tornato a riposo (autoritenuta dell'uscita). All'attivazione dell'ingresso **RESET**, l'uscita viene disattivata.

L'equazione generale che definisce una sequenza **SET-RESET** è:

Ox,y = SIj.k & RIn.m

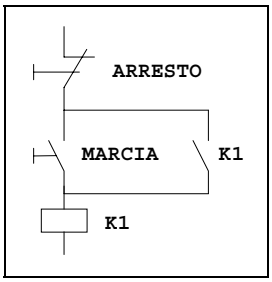
dove **Ij.k** è l'ingresso che attiva l'uscita **Ox.y** e **In.m** è l'ingresso che la spegne. Sia l'ingresso che l'uscita nella precedente equazione possono essere anche punti virtuali.

Nota: benchè nello schema elettromeccanico il pulsante di arresto (**RESET**) sia rappresentato come normalmente chiuso, il contatto reale da collegare all'ingresso del modulo deve essere normalmente aperto; per maggiori dettagli, vedere gli esempi che seguono.

Gli operatori **R** e **S** devono essere scritti come prefisso al relativo ingresso e possono essere utilizzati anche in combinazione con l'operatore di negazione (**!**). In tal caso gli operatori **R** e **S** devono precedere il simbolo di negazione.

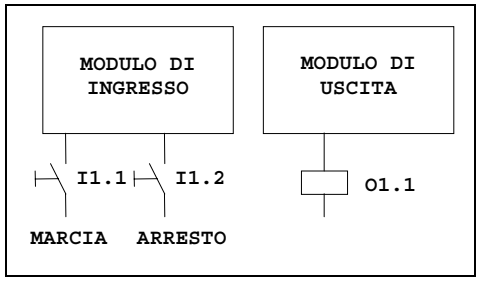
Esempi:

La figura a lato rappresenta l' equivalente elettromeccanico di una cella MARCIA-ARRESTO, ottenuta utilizzando un relè, un pulsante normalmente aperto ed un pulsante normalmente chiuso. Il contatto K1 in parallelo al pulsante di marcia implementa l'autoritenuta del relè.



L'equazione equivalente di una cella MARCIA-ARRESTO con MCP diventa:

O1.1 = SI1.1 & RI1.2



Notare, come già segnalato in precedenza, che entrambi i pulsanti collegati al modulo di ingresso sono normalmente aperti; in questo modo l'attivazione dell'uscita avviene alla chiusura del pulsante di marcia e la disattivazione alla chiusura del pulsante di arresto. Nel caso in cui si voglia ribaltare la logica dei pulsanti, si deve utilizzare l'operatore NOT come nel seguente esempio:

O1.1 = SI1.1 & R!I1.2

In questo caso l'attivazione dell'uscita avviene alla chiusura del pulsante di marcia e la disattivazione alla apertura del pulsante di arresto.

Nota: **il simbolo & tra operatori SET-RESET è obbligatorio.**

Gli operatori **R** e **S** possono anche essere combinati con ingressi privi di operatore che fungono in tal modo da consensi:

O1.5 = I2.3 & RI2.1 & SI4.6 L'ingresso **I2.3** è un consenso, poichè quando non è attivo forza lo spegnimento dell'uscita; si fa notare che se l'uscita si disattiva a causa della apertura di **I2.3**, l'uscita rimane disattivata sino a che non si richiude **I4.6** insieme a **I2.3** esattamente come avviene nel caso elettromeccanico.

2.3- Equazioni con operatori speciali

Gli operatori speciali consentono alcune funzioni particolari, sia per il controllo delle uscite che per la gestione di messaggi ed allarmi su visualizzatori di testi della linea **DUEMMEGI DISP**. Gli operatori speciali sono:

1. **GENERATORE DI CODICE BINARIO**
2. **CONTATORE**
3. **CONTROLLO PERIFERICA SERIALE (non disponibile per MCP Plus)**
4. **SOGLIA ANALOGICA**
5. **TIMER**
6. **OROLOGIO PROGRAMMATORE**
7. **CONTROLLO MODULI ANALOGICI**
8. **TOGGLE**
9. **GESTIONE MODULO CONTATORE A 16 BIT**
10. **GESTIONE MODULO DI INGRESSO ANALOGICO A 16 BIT**
11. **CONFIGURAZIONE**
12. **INDIRIZZO**
13. **CODICE DI IDENTIFICAZIONE**
14. **EVENTI**
15. **MODEM**
16. **PROTOCOLLO (solo per MCP Plus)**

2.3.1- Generatore di codice binario

Questo operatore consente di **associare un codice binario di 8 bit ai punti virtuali** del sistema e di inviarlo ad uno o più moduli di uscita se il relativo punto virtuale risulta attivo. Nel caso più uscite fossero attive contemporaneamente, MCP invia ciclicamente i rispettivi codici binari con un periodo di circa 2 secondi (funzione di rinfresco o ciclatura). Se tutti i punti virtuali che controllano un blocco binario tornano allo stato di riposo, viene inviato il codice 0 (zero).

L'operatore per la generazione di codice binario può quindi essere utilizzato per controllare un display ad ingresso binario, collegato ad un modulo di uscita (MODPNP) oppure per controllare direttamente un visualizzatore **DUEMMEGI DISP-BUS**. Utilizzando le uscite virtuali è possibile avere messaggi relativi ad un ingresso o combinazioni di più ingressi (reali e/o virtuali).
Potendo inoltre specificare per ogni modulo di uscita quali sono i codici da inviare e quali sono i punti virtuali che generano tali codici, è possibile gestire più di un display, ognuno con i propri messaggi ed indipendente dagli altri.

L'equazione del blocco **BINARY**, nella sua forma generale, è la seguente:

```
BINARY x (  \
      Bn=Vi \
      Bm=Vj \
      ...   \
      )
```

dove **Vi** deve essere specificato utilizzando le equazioni opportune. Il simbolo "\" di andata a capo (vedi capitolo riguardante la scrittura delle equazioni) è obbligatorio. La parola chiave **BINARY** identifica l'inizio del blocco ed è seguita dall'indirizzo **x** del modulo di uscita sul quale trasferire il codice binario. Seguono poi, racchiusi tra le parentesi **(** e **)**, i termini **Bn=Vi**, che definiscono il codice binario **n** da inviare quando la corrispondente uscita virtuale **i**-esima diventa attiva. Il codice binario può assumere i valori tra 1 e 255.

Nota: i termini `Vi` a destra del segno di uguale **non possono essere combinati con altri termini nè possono essere preceduti dall'operatore di negazione (`!`).** Ogni punto virtuale può essere presente solo una volta nello stesso blocco `BINARY` (significa cioè che un ingresso non può generare più di un codice binario sulla stessa uscita), mentre un codice binario può essere ripetuto più volte (più uscite virtuali possono generare lo stesso codice sulla stessa uscita); vedere l'esempio che segue per maggiori dettagli.

Esempio:

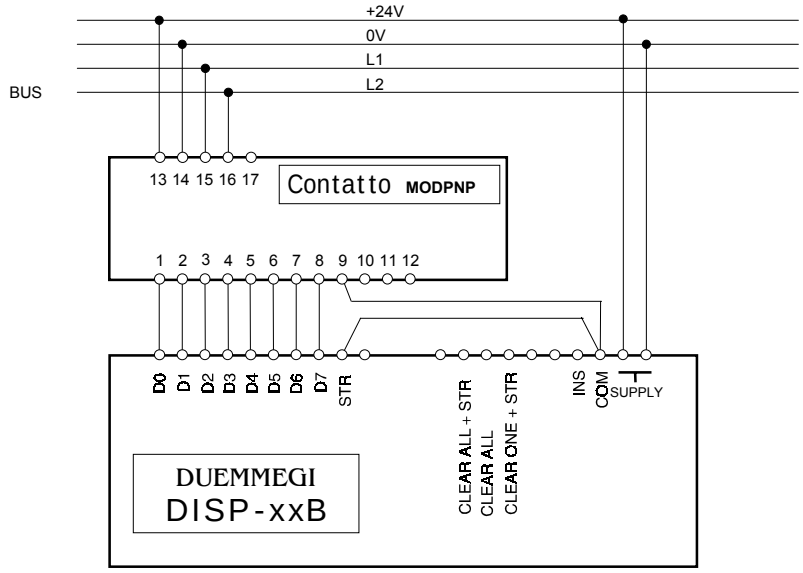
```
V1 = I1.1
V3 = I1.1 & I1.3
V5 = (I1.5 & I2.3) | I2.1
V8 = RI6.2 & SI4.6
BINARY 4 ( B1=V1 B2=V3 B3=V5 B26=V8 )
```

4 uscite virtuali generano 4 codici diversi. Notare che il simbolo di andata a capo (\) non è presente in quanto l'equazione è scritta su una sola riga.

```
BINARY 12 ( B1=V1 B2=V3 B1=V5 B26=V8 )
```

Sono presenti due uscite virtuali che generano lo stesso codice binario (`V1` e `V5`).

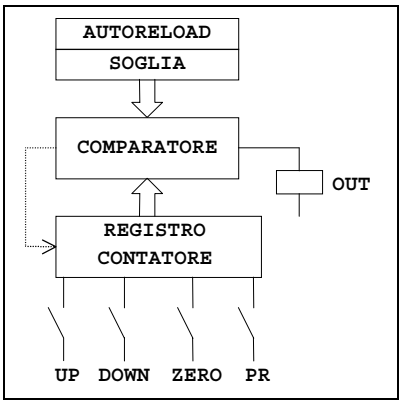
Il display binario **DUEMMEGI** DISP-XXB (XX dipende dal modello, ossia da quanti messaggi si vogliono visualizzare) deve essere collegato al modulo di uscita MODPNP come illustrato nello schema seguente.



Per i collegamenti del display su bus **DUEMMEGI** DISP-BUS si rimanda al manuale dello stesso.

2.3.2- Contatore

E' possibile definire alcune locazioni di memoria di MCP come contatori ed utilizzarle per il conteggio di impulsi provenienti da uno o più ingressi del sistema.
In funzione quindi del contenuto del contatore e di un valore prestabilito, detto soglia, si può controllare un'uscita come schematizzato dalla figura seguente.



Ogni contatore può essere controllato da almeno quattro ingressi, ognuno dei quali con una specifica funzione:

1. Ingresso per il conteggio in avanti (**U**, up)
2. Ingresso per il conteggio all'indietro (**D**, down)
3. Ingresso per l'azzeramento (**Z**, zero)
4. Ingresso per il caricamento del contatore con un valore definito (**P**, preset); gli ingressi di preset possono essere più di uno, con valori di caricamento diversi

Il contatore, in funzione delle variazione dei suoi ingressi, viene aggiornato e quindi confrontato con il valore della soglia, per il controllo dell'uscita.

L'uscita e gli ingressi di un contatore possono essere sia reali che virtuali. Il comparatore che confronta il valore si soglia con il contenuto del registro permette tre tipi di relazioni:

1. Uscita OFF se contatore < soglia e uscita ON se contatore >= soglia.
2. Uscita ON se contatore <= soglia e uscita OFF se contatore > soglia.
3. Uscita ON se contatore = soglia e uscita OFF se contatore diverso dalla soglia.

Il contatore può essere definito a 8 o 16 bit, consentendo un conteggio fra 0 e 255 nel primo caso e fra 0 e 65535 nel secondo. E' possibile utilizzare contemporaneamente sia contatori a 8 bit che contatori a 16 bit a patto che il loro numero rispetti la seguente regola:

(numero contatori a 8 bit) + 2 x (numero contatori a 16 bit) <= 512

La sintassi da utilizzare per la scrittura della relativa equazione è:

```
Oa.b = Cn {op} T,R U[s1]Ic.d & D[s2]Ie.f & ZIg.h & P[p1]Ii.m & Oq
```

dove:

Oa.b è l'uscita controllata dalla funzione contatore

Cn {op} T è la regola di confronto tra il contenuto del contatore n-esimo e un valore di soglia **T**; **n** può assumere valori tra 0 e 511 (vedere nel seguito il significato di questo parametro).
{op} può essere uno dei seguenti simboli:

- **=** (uguale)
- **>** (maggiore o uguale)
- **<** (minore o uguale)

T è un valore numerico compreso tra 0 e 255 per i contatori a 8 bit e tra 0 e 65535 per quelli a 16 bit; **per dichiarare che un contatore è a 16 bit, il valore T deve essere preceduto dal simbolo #**. Il valore **T** può essere anche un puntatore ad un altro contatore, vale a dire che si può definire, come valore di soglia, **il contenuto del registro di un altro contatore** (vedere nel seguito per maggiori dettagli)

`R` è il valore (facoltativo) di autoreset o autoreload per rendere il contatore autoazzerabile o autocaricabile: quando il conteggio in avanti raggiunge il valore `R`, il contatore viene automaticamente azzerato, mentre quando il conteggio all'indietro scende sotto il valore 0, il contatore viene ricaricato con il valore `R-1`. Il valore di `R` può essere un puntatore ad un altro registro contatore, vale a dire che si può definire, come valore di autoreset, **il contenuto del registro di un altro contatore** (vedere nel seguito per maggiori dettagli)
Nota: se non viene specificato il valore per l' autoazzeramento/autoreload, il conteggio viene arrestato a 0 (nel caso di conteggio all'indietro) e al valore massimo consentito (nel caso di conteggio in avanti), evitando l'underflow o l'overflow del contatore.

`U[s1]Ic.d` è la definizione dell'ingresso che fa incrementare il contatore (**U**P) di una quantità `[s1]`; in altre parole `s1` è un numero che rappresenta **lo step dell'incremento e può assumere valori tra 1 e 32** compresi. **Lo step deve essere racchiuso tra parentesi quadre e non è obbligatorio** (quindi se non è specificato viene assunto uno step pari a 1).

`D[s2]Ie.f` è la definizione dell'ingresso che fa decrementare il contatore (**D**OWN) di una quantità `[s2]`; in altre parole `s2` è un numero che rappresenta **lo step del decremento e può assumere valori tra 1 e 32** compresi. **Lo step deve essere racchiuso tra parentesi quadre e non è obbligatorio** (quindi se non è specificato viene assunto uno step pari a 1).

`ZIg.h` è la definizione dell'ingresso che fa azzerare il contatore (**Z**ERO)

`P[p1]Ii.m` è la definizione dell'ingresso che provoca il caricamento del contatore con il valore `p1` (**P**RESET); `p1` è un numero compreso tra:

1 e 255	per i contatori a 8 bit
1 e 65535	per i contatori a 16 bit

Il valore di preset deve essere racchiuso tra parentesi quadre e non può essere pari a zero. Si possono inoltre definire più ingressi di preset (vedere esempi che seguono)

`Oq` è un modulo di uscita **reale** di indirizzo `q`; il contenuto del contatore verrà trasferito su questo modulo di uscita. Se il contatore è a 16 bit, verrà trasferita sul modulo di uscita solo la parte alta, ovvero il byte più significativo (si ricorda che i moduli di uscita sono a 8 bit). **È possibile specificare più di una uscita** sulla quale trasferire il contenuto del registro contatore.

Il simbolo & che lega i vari termini è obbligatorio ed è l'unico operatore che può essere utilizzato nell'equazione che definisce un contatore.

Le definizioni di **U**P, **D**OWN, **Z**ERO e **P**RESET possono essere tutte oppure solo in parte presenti. Inoltre, come già accennato sopra, **il PRESET può essere presente più volte nella stessa equazione** con ingressi diversi, in modo da permettere il precaricamento di vari valori.
Il conteggio, avanti o indietro, l'azzeramento e i preset sono normalmente effettuati sulla transizione OFF→ON dell'ingresso. Volendo utilizzare come evento la transizione ON→OFF, si può utilizzare l'operatore di negazione NOT (`!`) davanti al relativo ingresso.

Tutti gli ingressi e tutte le uscite presenti in una funzione contatore possono essere **sia reali che virtuali, ad eccezione per i termini** `Oq` che possono essere solo reali.

I contatori della versione 4.x (o superiori) di MCP, a differenza delle precedenti, devono essere numerati (es. `C1, C2, C3`***, ecc.). In questo modo risulta più semplice l'individuazione dell'indirizzo del registro, all'interno della memoria RAM di MCP, del contatore sul quale si vuole eventualmente agire (leggere o scrivere) da un sistema di supervisione. Quando si assegna un numero ad un contatore si deve tenere presente che un contatore a 16 bit occupa 2 celle di memoria, per cui esso occuperà sia il numero di registro che gli è stato esplicitamente assegnato sia il successivo (che quindi non potrà essere usato nella definizione di un altro contatore).*** L'esempio che segue illustra quanto appena detto:

```
V1 = C0 > 10 UI1.1 & DI1.2          // Contatore C0 a 8 bit (1 registro,
                                    // chiamato C0 come esplicitamente
                                    // dichiarato)
V2 = C1 > #300 U I1.3 & D I1.4      // Contatore C1 a 16 bit (2 registri,
                                    // chiamati C1, come esplicitamente
                                    // dichiarato, e C2 sottinteso)
V3 = C3 > 128 UI1.5 & DI1.6         // Contatore C3 a 8 bit (1 registro,
                                    // chiamato C3 come esplicitamente
                                    // dichiarato; non poteva essere chiamato C2
                                    // in quanto già utilizzato nella precedente
                                    // equazione)
```

Non è comunque necessario che i contatori vengano numerati in ordine crescente, e non è vietato lasciare dei contatori inutilizzati tra uno e l'altro.

Come sopra ricordato, la versione 4.x di MCP (o superiori) **consente di definire, come valore di soglia `T` e/o come valore di autoreset/autoreload `R`, il contenuto di un altro contatore**; in questo modo è possibile avere **contatori a soglia variabile** e quindi impostabile da supervisore (il quale andrà a scrivere il valore voluto nel relativo registro), oppure impostabile mediante UP/DOWN del contatore che contiene il valore di soglia.
Quando un contatore è definito a 16 bit (2 byte), il numero di identificazione più basso contiene la parte (byte) più significativa, mentre l'identificativo successivo contiene la parte (byte) meno significativa. Ad esempio, se `C4` è stato definito come contatore a 16 bit, il registro `C4` contiene il byte più significativo, mentre `C5` quello meno significativo e quindi il supervisore dovrà scrivere entrambi i registri.

Gli esempi che seguono chiariscono meglio quanto appena illustrato.

Esempio 1:

```
O1.1 = C1 > C2,C3 UI1.1 & DI1.2
```

il contenuto del contatore `C1` viene confrontato con un valore di soglia che è il contenuto del registro contatore `C2`. Il contatore `C1`, inoltre, viene incrementato (a passi di una unità) ad ogni commutazione OFF→ON dell'ingresso `I1.1` e viene decrementato (a passi di una unità) ad ogni commutazione OFF→ON di `I1.2`. L'uscita `O1.1` sarà attivata o disattivata in base al risultato del confronto. Inoltre, quando il conteggio `C1` raggiunge il valore contenuto nel registro `C3`, il contatore `C1` viene azzerato; se invece il conteggio `C1` scende sotto il valore 0, allora il contatore `C1` viene ricaricato con il contenuto del registro `C3` diminuito di una unità.
Il supervisore potrà, attraverso la linea seriale RS232 di MCP, scrivere all'indirizzo RAM del contatore `C2` e/o `C3` in modo da impostare la soglia e il valore di autoreset/autoreload al valore voluto. **Notare che se i due parametri sono impostati solo da supervisore, non è necessario definire `C2` e `C3` nel programma di MCP**.

Esempio 2:

```
O1.1 = C1 > C2 UI1.1 & DI1.2
V1 = C2 > 1 U[10]I1.3 & D[10]I1.4
```

il contenuto del contatore `C1` viene confrontato con un valore di soglia che è il contenuto del registro `C2` il quale, a sua volta, viene incrementato, a passi di 10, da `I1.3` e decrementato, a passi di 10, da `I1.4`.
`C2` potrebbe comunque essere controllato anche dal supervisore.
`V1` e il valore `1` della soglia di `C2` servono, in questo esempio, solo a permettere la definizione del contatore `C2`; in altre parole, `V1` potrebbe non essere utilizzato come ingresso per altre equazioni dello stesso programma, ed il valore di soglia potrebbe essere un valore "fittizio" nel senso che potrebbe essere uguale a qualsiasi valore tra 0 e 255 senza per questo alterare il funzionamento del programma (lo stesso dicasi per l'operatore di confronto).
Seguono altri esempi a carattere generale che illustrano l'uso dei contatori.

Esempi:

```
O1.1 = C1 < 10 UI1.1 & DI1.2          // O1.1 verrà accesa quando il contenuto del
                                      // contatore è minore o uguale a 10. Lo step
                                      // di incremento e decremento è pari a 1

V34 = C2 > 125 U[25]I1.1 & DI1.2      // V34 verrà accesa quando il contenuto del
                                      // contatore è maggiore o uguale a 125. Lo
                                      // step di incremento è pari a 25 mentre
                                      // quello di decremento è pari a 1

O78.5 = C8 > #312 UV1 & DV2 & P[300]V998 // il contatore (a 16 bit) viene
                                         // precaricato con il valore 300
                                         // quando la V998 diventa attiva, vale
                                         // a dire all'accensione di MCP
                                         // (vedere il significato della V998 a
                                         // pag.4 del manuale)

O1.4 = C3 > 10,15 UI1.1 & DI1.2       // il contatore attiva l'uscita quando il
                                      // conteggio è maggiore o uguale a 10 e si
                                      // autoazzera quando supera 14. In caso di
                                      // conteggio all' indietro, il contatore
                                      // viene ricaricato con il valore 14 quando
                                      // il conteggio scende sotto zero.
```

L'esempio che segue illustra come si potrebbe realizzare **l'impianto semaforico di un parcheggio**, dove il numero massimo di posti auto può essere variato dal supervisore scrivendo nei registri C2-C3 (16 bit).
ATTENZIONE: ricordare sempre che un contatore a 16 bit occupa 2 posizioni, per cui il supervisore, per cambiare il valore di soglia, dovrà scrivere in entrambi i registri C2 e C3.

```
V1 = C2 > #1 P[312]V998               // precarica C2 (16 bit) con il valore
                                      // 312 (solo al reset di MCP)
V2 = C0 > #C2 UI1.1 & DI1.2 & ZI1.3   // confronta C0 (16 bit) con il
                                      // contenuto di C2
O1.1 = V2                             // accendi O1.1 se C0>= C2 (semaforo
                                      // rosso)
O1.2 = !V2                            // accendi O1.2 se C0 < C2 (semaforo
                                      // verde)
```

L'esempio che segue mostra come sia possibile **controllare due dimmer per regolare l'intensità luminosa** di alcune lampade. Mediante 2 pulsanti (UP e DOWN) si incrementa o decrementa il valore del contatore a passi di 15 (step) e il risultato viene inviato ai moduli di uscita analogici `O1` e `O2` a loro volta collegati ad altrettanti dimmer elettronici. Mediante 2 pulsanti connessi a `I1.4` e `I1.5` è inoltre possibile selezionare la luminosità ad altrettanti valori predefiniti (es. un terzo e due terzi del massimo). Attivando un pulsante connesso a `I1.6`, infine, è possibile azzerare il contatore e quindi spegnere completamente le luci. Notare che `V1` e il valore 255 sono "fittizi" e servono unicamente, in questo caso, a definire il contatore. ***ATTENZIONE:*** il valore di preset deve essere diverso da 0 e **il valore di step deve essere compreso tra 1 e 32**. Notare anche come sia possibile definire più preset nella stessa equazione.

```
V1 = C1 > 255 \
     U[15]I1.1 & D[15]I1.2 & \   // up step 15 con I1.1 e down step 15 con I1.2
     P[85]I1.4 &  \              // preset 85 quando si attiva I1.4
     P[170]I1.5 & \              // preset 170 quando si attiva I1.5
     ZI1.6 & \                   // azzera il contatore quando si attiva I1.6
     O1 & O2                     // trasferisce il contenuto del contatore su O1
                                 // e O2
```

2.3.3- Controllo periferica seriale

Questo operatore è dedicato al controllo di un display seriale (**DUEMMEGI DISP-S**) connesso all'apposita presa seriale RS232 di MCP ed eventualmente di una stampante collegata al display. In relazione alle variazioni degli ingressi definiti è possibile inviare particolari comandi al display per la visualizzazione di messaggi di allarme e/o di stato. E' inoltre possibile la gestione congiunta di una uscita (normalmente per comandare una sirena in caso di allarme) e di due ingressi (pulsanti di tacitazione e reset).

ATTENZIONE: questa funzione non è disponibile per MCP Plus.

L'equazione che definisce la lista dei messaggi con i relativi ingressi, i pulsanti di ACK e RESET e l´uscita per la sirena è la seguente:

```
SERIAL ( \
      ACK = Vn \
      RST = Vm \
      SIREN = Vp \
      DEFAULT = PRN MEM SIREN \
      Z001 = Vx1 opzioni \
      Z002 = Vx2 opzioni \
      ... \
      )
```

Il simbolo "\" di andata a capo (vedi capitolo riguardante la scrittura delle equazioni) è obbligatorio.
La parola chiave **SERIAL** definisce l'inizio del blocco di informazioni che devono essere racchiuse fra le parentesi tonde **(** e **).** Vi sono poi tre parametri necessari per la gestione degli allarmi: **ACK**, **RST** e **SIREN** che definiscono rispettivamente gli ingressi ai quali sono collegati i pulsanti di tacitazione della sirena, di reset degli allarmi e l'uscita virtuale per il controllo della sirena. Tali parametri sono facoltativi e possono quindi essere omessi.

La linea dei **DEFAULT** indica le tre opzioni che possono essere **PRN** o **NOPRN**, **MEM** o **NOMEM** e **SIREN** o **NOSIREN** (è indifferente utilizzare caratteri minuscoli o maiuscoli). Tali default indicano le opzioni da assumere per le successive linee **dove non sia diversamente specificato**. Se la linea **DEFAULT** non viene specificata i default assunti sono **MEM, PRN e SIREN.**
Segue poi la lista dei codici da inviare al display (numero del messaggio) con il corrispondente ingresso virtuale (con eventuali opzioni diverse da quelle di default).
E' tassativo che tutti i punti definiti all'interno del blocco SERIAL siano virtuali; questi punti virtuali vanno quindi definiti a priori legandoli a punti o a combinazioni di punti reali come descritto in precedenza (a meno che non siano V999 e/o V1000, vedi paragrafo 1.1); inoltre i termini **Vi** a destra del segno di uguale non possono essere combinati con altri termini ma possono essere preceduti dall'operatore di negazione (**!**) come mostrato nell'esempio alla fine del presente paragrafo.

Quando un punto virtuale tra quelli specificati diventa attivo, il corrispondente codice numerico viene inviato al display per richiamarne il relativo messaggio ed inoltre vengono eseguite le azioni corrispondenti alle tre opzioni definite per quell'uscita:

1. L'uscita viene inserita nella coda di refresh e memorizzata
2. Viene inviato il comando di stampa (se l'opzione **PRN** è abilitata). Nessun comando con l'opzione **NOPRN**
3. Viene attivata la sirena se abilitata (**SIREN**). L'opzione **NOSIREN** non attiva la sirena

L'opzione **MEM** (o **NOMEM**) definisce il comportamento della sequenza quando il punto virtuale torna disattivo. Nel caso **NOMEM** esso viene cancellato dalla coda di rinfresco ciclico, mentre nel caso **MEM** rimane memorizzato e solo un comando di reset può rimuoverlo (a patto, ovviamente, che lo stesso punto virtuale sia tornato a riposo).

MCP continua ad inviare in modo ciclico i codici corrispondenti alle uscite virtuali attive e/o memorizzate fino a che la coda non viene svuotata (in modo automatico se **NOMEM** o manuale se **MEM**). In tal caso il codice 000 viene inviato al display. La procedura di reset è condizionata alla tacitazione della sirena, e quindi all'acquisizione dell'allarme da parte dell'operatore. In altre parole il reset non ha alcun effetto se prima non è stata eseguita la tacitazione della sirena (**ACK**).

Riassumendo e facendo riferimento alla forma sintattica generale sopra riportata, i significati dei simboli sono i seguenti:

Vn	punto virtuale per la tacitazione
Vm	punto virtuale per il reset
Vp	punto virtuale per la sirena
Vx1	punto virtuale associato al messagio relativo al codice ESC Z 001 (richiamo messaggio 1)
Vx2	punto virtuale associato al messagio relativo al codice ESC Z 002 (richiamo messaggio 2)
opzioni	La lista delle opzioni per il corrispondente messaggio (**MEM** o **NOMEM**, **PRN** o **NOPRN**, **SIREN** o **NOSIREN**)**:** **MEM**: il messaggio viene memorizzato fino al reset **NOMEM**: il messaggio viene rimosso non appena l´ingresso relativo torna allo stato di riposo **PRN**: il messaggio viene stampato **NOPRN**: il messaggio non viene stampato **SIREN**: l'ingresso attiva la sirena **NOSIREN**: l'ingresso non attiva la sirena
DEFAULT	Le opzioni di default da considerare dove non diversamente specificato

Note:

1. Le linee relative alla definizione dell'ingresso di ACK, di RST, dell'uscita per la sirena e la linea dei default non sono obbligatorie.
2. In mancanza della linea dei default vengono assunte le seguenti opzioni: **MEM PRN SIREN.**
3. I punti virtuali relativi all'ACK e al RST, se richiesti, devono essere specificati da apposite equazioni.
4. Il punto virtuale relativo alla sirena dovrà comandare poi un'uscita reale (anche se non espressamente obbligatorio).
5. I punti virtuali relativi ai messaggi devono essere specificati da apposite equazioni.
6. La tacitazione ed il reset degli allarmi vengono eseguiti conformemente alla sequenza ISA-M, per cui se l'allarme non viene prima riconosciuto, non può essere cancellato.
7. Se al momento del reset sono ancora presenti allarmi, essi verranno visualizzati ma la sirena non viene comandata sino a che non interviene un nuovo allarme (vale a dire un allarme non ancora presente nella coda).
8. I punti virtuali possono essere preceduti dall'operatore NOT (!). In tal caso lo stato di tali ingressi viene complementato.
9. È possibile generare la chiamata di un messaggio anche nel caso in cui si verifica una condizione di modulo guasto o di bus guasto, utilizzando i punti virtuali V999 e V1000 (vedi paragrafo 1.1)

Esempio:

```
V100 = I1.1                    Ingresso per la tacitazione
V200 = I1.2                    Ingresso per il reset
O1.1 = V300                    Uscita per la prima sirena (ad esempio locale)
O2.1 = V300                    Uscita per la seconda sirena (ad esempio remota)

V1 = I2.1                      Definizione allarme per messaggio 1
V2 = I2.2 | I2.3 | I2.4        Definizione allarme per messaggio 2
V3 = I2.5 & I2.6               Definizione allarme per messaggio 3
V4 = CLOCK( 12:00, 13:00)      Definizione stato per messaggio 4 (il significato di CLOCK verrà
                               descritto in un paragrafo successivo)

SERIAL ( \                     Inizio blocco SERIAL
  ACK = V100 \                 Comando di tacitazione
  RST = V200 \                 Comando di reset
  SIREN = V300 \               Controllo della sirena
  DEFAULT = MEM NOPRN SIREN \  Opzioni di default: memorizzazione allarmi, nessuna stampa e
                               attivazione della sirena
  Z001 = V1 \                  Allarme 1
  Z002 = !V2 \                 Allarme 2 (logica complementata)
  Z003 = V3 PRN \              Allarme 3 con stampa
  Z004 = V4 NOMEM NOSIREN \    Visualizzazione di stato (in quanto non memorizzato e senza sirena)
)                              Fine blocco SERIAL
```

2.3.4- Soglia

L'operatore **SOGLIA** consente di associare un punto di uscita digitale ad un modulo di ingresso analogico e controllare tale uscita in funzione del valore assunto dall'ingresso e di una soglia specificata. La versione 4.x di MCP (o superiori) permette di definire **valori di soglia e di isteresi non solo fissi ma anche variabili**, secondo la seguente sintassi:

```
Oa.b = Aj {op} T,H
```

dove:

Oa.b è l'uscita controllata dalla funzione soglia (può essere anche virtuale)

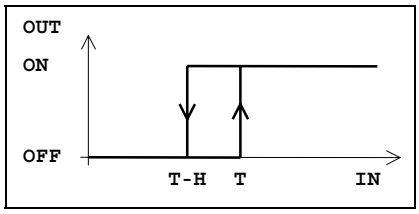
Aj identifica il modulo di ingresso analogico di indirizzo **j**

{op} è l'operatore di confronto che può essere:
- `>` (maggiore o uguale)
- `<` (minore o uguale)

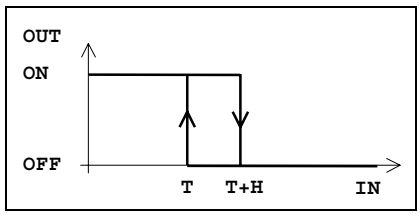
T è la soglia, ovvero un valore numerico compreso tra 0 e 255 oppure l'identificativo di un registro contatore che contiene il valore di soglia

H è il valore di isteresi, ovvero un valore numerico compreso tra 0 e 255 oppure l'identificativo di un registro contatore che contiene il valore di isteresi

Il significato dell'isteresi è diverso a seconda dal segno del confronto (< o >) come qui di seguito descritto:



> l'uscita viene accesa quando l'ingresso è maggiore o uguale alla soglia (**T**) e viene spenta quando l'ingresso diventa minore o uguale alla soglia diminuita dell'isteresi (**T - H**)



< l'uscita viene accesa quando l'ingresso è minore o uguale alla soglia (**T**) e viene spenta quando l'ingresso diventa maggiore o uguale alla soglia aumentata dell'isteresi (**T + H**)

L'isteresi risulta molto utile, se non indispensabile, in tutti quei processi in cui è necessario evitare continue commutazioni dell'uscita quando il valore di ingresso è nell'intorno della soglia (es. termoregolazione mediante termostato, controllo di livello, ecc.). **Se nell'equazione si omette l'isteresi, essa viene assunta uguale a zero**.

È possibile legare tra loro più termini mediante operatori AND (**&**) e OR (|).

I valori di **T** ed **H** devono essere compresi fra 0 e 255 (valore di fondo scala dei moduli di ingresso analogici della linea Contatto).

Esempi:

`O1.1 = A1 > 240,2`	Segnalazione valore sopra soglia con isteresi pari a 2: l'uscita viene accesa quando il valore di ingresso è maggiore o uguale a 240 e viene spenta quando il valore di ingresso è minore o uguale a 238.
`V2 = A1 < 40,2 | A2 <30,4`	Segnalazione valore sotto soglia di due ingressi: l'uscita (ad esempio virtuale) viene accesa quando il valore del modulo 1 scende sotto 40 oppure quando il valore del modulo 2 scende sotto 30 (per le isteresi vale quanto detto sopra).
`O1.4 = A1 < 128 & A1 > 30`	L'uscita viene accesa quando il valore di ingresso è maggiore di 30 ma contemporaneamente minore di 128 (in altre parole quando il valore di ingresso è compreso tra 30 e 128).

Come sopra ricordato, la versione 4.x di MCP (o superiori) **consente di definire, come valore di soglia T e/o come valore di isteresi H, il contenuto di un registro contatore**; in questo modo è possibile avere **funzioni a soglia e isteresi variabili** e quindi impostabili da supervisore (il quale andrà a scrivere il valore voluto nel relativo registro), oppure impostabile mediante UP/DOWN del contatore che contiene il valore di soglia o di isteresi. **I registri contatori che vengono usati nelle funzioni soglia possono essere solo a 8 bit**. Gli esempi che seguono chiariscono quanto appena illustrato.

Esempi:

Nell'esempio che segue, l'uscita `O1.1` viene attivata quando il valore letto dal modulo di ingresso analogico di indirizzo 1 è maggiore del valore contenuto nel registro contatore `C1` ma minore del valore contenuto nel registro `C2`. Le isteresi sono, in questo esempio, fisse e pari a 2.

Al reset di MCP (**V998** attiva), il contatore **C1** viene precaricato con il valore 64 e il **C2** con 128; a questo punto un eventuale sistema di supervisione può, attraverso la porta seriale di MCP, cambiare i valori di soglia semplicemente scrivendo nei relativi registri.
V1, **V2** e il valore 1 nelle due equazioni dei contatori sono puramente "fittizi" e servono solo a definire i contatori da "precaricare" con i valori indicati.
Si ricorda che i registri contatori in una funzione soglia possono essere solo a 8 bit.

```
V1 = C1 > 1 P[64]V998
V2 = C2 > 1 P[128]V998
O1.1 = A1 > C1,2 & A1 < C2,2        // O1.1 accesa se A1 compreso tra i valori
                                    // contenuti in C1 e C2
```

Nel seguente esempio, sia il valore di soglia che quello di isteresi sono prelevati da registri contatori (rispettivamente **C4** e **C8**); poichè in questo esempio non si richiede di precaricare i due contatori con valori definiti e i due registri sono controllati esclusivamente dal supervisore, allora non è necessario includere nel programma due equazioni che definiscano esplicitamente **C4** e **C8**.

```
V100 = A22 < C4,C8
```

Nell'esempio che segue l'uscita **O2.3** viene attivata quando **A10** è maggiore o uguale al valore contenuto nel registro **C20**; quest'ultimo, a sua volta, viene incrementato da **I1.1** e decrementato da **I1.2**.

```
V34 = C20 < 1 UI1.1 & DI1.2
O2.3 = A10 > C20,5
```

2.3.5- Timer

I timer consentono di comandare un'uscita (reale o virtuale) con un ritardo programmabile rispetto al segnale di comando; si possono definire **sino ad un massimo di 256 timer**.
Il ritardo può essere all'eccitazione, alla diseccitazione od ad entrambe e può variare da 0 a 6553,5 secondi (poco più di 1 ora e 49 minuti) con risoluzione di 0.1 secondi.
La versione 4.x di MCP (o superiori) permette di definire **ritardi di eccitazione e diseccitazione non solo fissi ma anche variabili**, secondo la seguente sintassi:

```
Oa.b = TIMERn (Ij.k,e,d)
```

oppure:

```
Oa.b = TIMERn (TIj.k,0,d)
```

oppure ancora:

```
Oa.b = TIMERn R(TIj.k,0,d)
```

dove:

Oa.b è l'uscita controllata dalla funzione timer (reale o virtuale)
n è l'identificativo, **facoltativo**, del timer (numero compreso tra 0 e 255); per maggiori dettagli su questo parametro, vedere nel seguito
Ij.k è l'ingresso che comanda la funzione timer (reale o virtuale)
e è il ritardo all'eccitazione (in decimi di secondo) dell'uscita che segue ad una variazione dell'ingresso, ovvero un valore numerico compreso tra 0 e 65535 oppure l'identificativo di un registro contatore **a 16 bit** che contiene il valore del ritardo all'eccitazione
d è il ritardo alla diseccitazione (in decimi di secondo) dell'uscita che segue ad una variazione dell'ingresso, ovvero un valore numerico compreso tra 0 e 65535 oppure l'identificativo di un registro contatore **a 16 bit** che contiene il valore del ritardo all'eccitazione
T è un simbolo che, quando posto davanti all'ingresso, fa funzionare il timer come monostabile, vale a dire che sull'uscita viene generato un impulso di durata pari ad un tempo **d** e "triggerato" dalla variazione OFF→ON dell'ingresso specificato (oppure dalla variazione ON→OFF se l'ingresso è

preceduto dal simbolo di negazione **!**); il tempo **e**, nel caso del monostabile, non ha significato e deve essere pari a zero

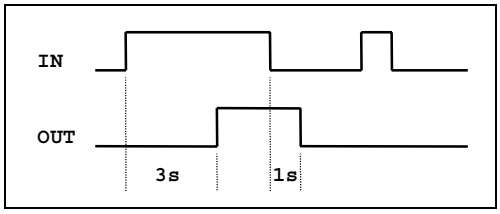
R è un simbolo che, quando posta appena prima della apertura di parentesi di una funzione timer **monostabile**, lo rende "retriggerabile", vale a dire che l'impulso sull'uscita ha la durata specificata a partire dall'ultima variazione OFF→ON dell'ingresso specificato (oppure OFF→ON se l'ingresso è preceduto dal simbolo di negazione **!**); si noti che il simbolo R ha senso solo per i monostabili

L'ingresso reale **Ij.k** (o virtuale) può essere in tutti i casi preceduto dal simbolo di negazione **!** (NOT). In tal caso la logica dell'ingresso viene complementata.
Se il parametro di ingresso (Ij.k) è preceduto dalla lettera T il timer funziona da monostabile: genera cioè un impulso sulla corrispondente uscita di durata d (in questo caso il tempo e non viene considerato, vale a dire che viene assunto uguale a zero).

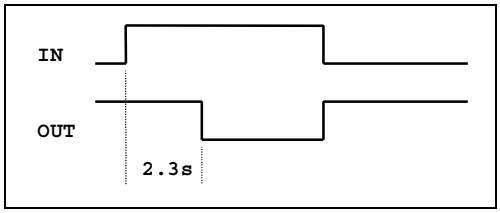
Il parametro n non è obbligatorio; esso serve solo per identificare in modo semplice il timer quando è richiesta la lettura e/o la scrittura dei relativi registri da PC o da supervisore.

Esempi:

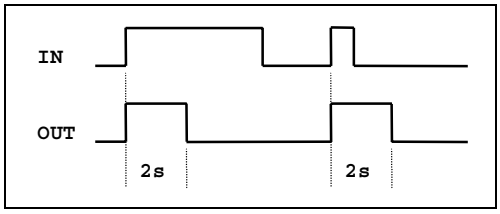
O1.1 = TIMER8(I2.5, 30, 10) Ritardo all'eccitazione di 3 secondi e alla diseccitazione di 1 secondo; notare che se l'ingresso diventa attivo per un tempo minore del ritardo di eccitazione, l'uscita non commuta. È stato utilizzato il timer numero 8.



V23 = TIMER(!I1.1, 0, 23) Uscita complementare rispetto all'ingresso, nessun ritardo all'eccitazione, 2.3 secondi di ritardo alla diseccitazione. Questa equazione non specifica quale timer è stato utilizzato

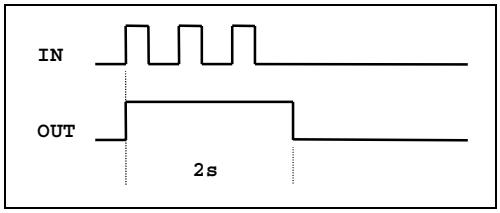


O1.1 = TIMER (TI1.1, 0, 20) Impulso di due secondi quando **I1.1** si chiude (***monostabile***); all'apertura dell'ingresso non succede nulla. Notare che l'impulso sull'uscita viene generato anche se l'ingresso rimane attivo per un tempo inferiore alla durata dell'impulso stesso.

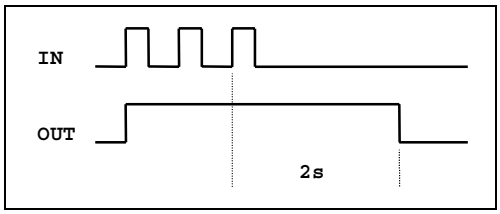


Nota: nel caso del **monostabile non retriggerabile**, l'ingresso che fa partire l'impulso viene ignorato per tutta la durata dell'impulso stesso, come qui di fianco illustrato (in altre parole il monostabile **non** è retriggerabile). L'equazione è ancora:

O1.1 = TIMER (TI1.1, 0, 20)



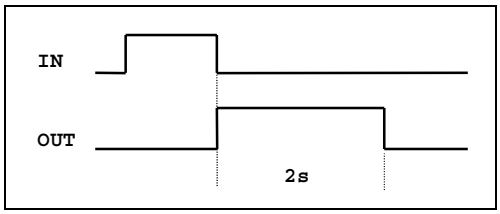
La figura a fianco mostra invece come si comporta un **monostabile retriggerabile**; l'equazione, in questo caso è:




```
O1.1 = TIMER R ( TI1.1, 0, 20)
```

La durata dell'impulso sarà ancora pari a 2 secondi, ma a partire dall'ultima variazione OFF→ON dell'ingresso.

La negazione dell'ingresso di un monostabile, sia esso retriggerabile o meno, dà luogo ad un impulso positivo sull'uscita ma "triggerato" sulla variazione ON→OFF come indicato dalla figura a lato. L'equazione è:




```
O1.1 = TIMER ( T!I1.1, 0, 20)
```

Come sopra ricordato, la versione 4.x di MCP (o superiori) **consente di definire, come valore del ritardo di eccitazione e/o come valore del ritardo di diseccitazione, il contenuto di un contatore**; in questo modo è possibile avere **funzioni timer a ritardi variabili** e quindi impostabili da supervisore (il quale andrà a scrivere il valore voluto nel relativo registro), oppure impostabile mediante UP/DOWN del contatore che contiene il valori dei ritardi. **I registri contatori che vengono usati nelle funzioni timer devono essere *esclusivamente* a 16 bit**.
Gli esempi che seguono chiariscono quanto appena illustrato.

Esempi:

Nell'esempio che segue, l'uscita `O1.1` viene attivata con un ritardo pari a al valore contenuto nel registro contatore `C0` e viene disattivata con un ritardo pari al valore contenuto nel registro `C2`. Al reset di MCP (V998 attiva), il contatore `C0` viene precaricato con il valore 20 (pari a un ritardo di 2 secondi) e il `C2` con 40 (pari a un ritardo di 4 secondi); a questo punto un eventuale sistema di supervisione può, attraverso la porta seriale di MCP, cambiare i valori dei ritardi semplicemente scrivendo nei relativi registri (`C0-C1` e `C2-C3`). `V1`, `V2` e il valore `#1` nelle due equazioni dei contatori sono puramente "fittizi" e servono solo a definire i contatori da "precaricare" con i valori indicati. Si ricorda che i registri contatori in una funzione timer possono essere solo a 16 bit, il che significa mettere `#` davanti a soglia nella definizione del contatore. ATTENZIONE: un contatore a 16 bit occupa 2 posizioni.

```
V1 = C0 > #1 P[20]V998
V2 = C2 > #1 P[40]V998
O1.1 = TIMER(I1.1, C0, C2)
```

Nell'esempio che segue, i due ritardi sono prelevati da registri contatori (rispettivamente `C4` e `C6`); poichè in questo esempio non si richiede di precaricare i due contatori con valori definiti e i due registri (o meglio quattro: `C4-C5` e `C6-C7`) sono controllati esclusivamente dal supervisore, allora non è necessario includere nel programma due equazioni che definiscano esplicitamente `C4` e `C6`.

```
V100 = TIMER(V8, C4, C6)
```

Nell'esempio che segue l'uscita `O2.3` viene attivata con un ritardo pari al valore contenuto nel registro `C20`; quest'ultimo, a sua volta, viene incrementato da `I1.1` e decrementato da `I1.2`. Il ritardo alla diseccitazione è fisso e pari a 5 secondi. Notare che `C20` è stato definito a 16 bit, in quanto deve controllare una funzione `TIMER`.

```
V34 = C20 < #1 UI1.1 & DI1.2
O2.3 = TIMER(I2.1, C20, 50)
```

2.3.6- Orologio programmatore

La funzione orologio consente la programmazione delle uscite in funzione di fasce orarie.
La programmazione può essere **giornaliera o settimanale**, nel qual caso deve essere specificato, oltre l'orario, anche il giorno della settimana.
Sono possibili le seguenti sintassi:

```
Ox.y = CLOCK(ON,OFF | ...)
```

L'uscita controllata, **Ox.y**, può essere sia reale che virtuale.
ON e **OFF** indicano rispettivamente gli orari di accensione e di spegnimento nel seguente formato:

```
GG:HH:MM
```

dove:

GG indica il giorno della settimana e può assumere i valori da 1 a 7: **1=LUN, 2=MAR, 3= MER, 4=GIO, 5=VEN, 6=SAB, 7=DOM**. **Se omesso la programmazione viene assunta giornaliera**.
E' possibile (e raccomandabile), in luogo del numero, utilizzare le prime tre lettere del giorno (**LUN, MAR, MER, GIO, VEN, SAB, DOM**).

HH ore (00 ÷ 23). Questo parametro deve essere indicato nella notazione 24 ore: 7 significa quindi le 7 di mattina, 19 significa le 7 di sera.

MM minuti (0 ÷ 59).

| simbolo dell'operatore OR per combinare più fasce orarie.

Esempi:

`O1.1 = CLOCK(8:15, 17:30)`	Accensione dell'uscita **O1.1** tutti i giorni alle 8:15 e spegnimento alle 17:30.
`O1.3 = CLOCK(1:8:00, 5:20:00)`	Accensione dell'uscita **O1.3** il Lunedì alle 8:00 e spegnimento alle 20:00 del Venerdì.
`O1.3 = CLOCK(LUN:8:00, VEN:20:00)`	Equazione esattamente identica alla precedente.

Nota: l'ora **24:00** non è consentita; utilizzare la notazione **00:00**. Fare attenzione però che questa notazione è riferita alla mattina del giorno specificato; ad esempio:

```
O1.1 = CLOCK(LUN:8:00,MER:0:00)
```

significa che l'uscita viene sarà accesa tutti i lunedi mattina alle 8 e sarà spenta tutti i mercoledi mattina alle 0, che equivale alla mezzanotte di ogni martedi.

Con la versione 4.x di MCP (o superiori), **è possibile avere fasce orarie variabili** (es. impostabili da supervisore) appoggiandosi a registri contatori. Sono possibili due casi:

1. giorno della settimana (GG) fisso e orario (HH:MM) variabile; la sintassi in questo caso è:

```
Ox.y = CLOCK( Cv, Cw )
```

oppure:

```
Ox.y = CLOCK( GGa:Cv, GGb:Cw )
```

In questo caso i contatori **Cv** e **Cw** contengono un numero compreso tra 0 e 1439, corrispondente al **numero di minuti a partire dalle ore** **0:00** (1439 = **23:59**); nella prima notazione la fascia oraria è giornaliera, mentre nella seconda la fascia oraria è settimanale, ma con i giorni della settimana fissi (**GGa** e **GGb**).

2. giorno della settimana e orario (GG:HH:MM) variabili; la sintassi in questo caso è:

```
Ox.y = CLOCK( xxx:Cv, xxx:Cw )
```

oppure:

```
Ox.y = CLOCK( 8:Cv, 8:Cw )
```

In questo caso i contatori **Cv** e **Cw** contengono un numero compreso tra 0 e 10079, corrispondente al **numero di minuti a partire dalle ore** **0:00** del Lunedi (10079 = **23:59** della Domenica); **xxx** nella prima notazione o **8** nella seconda sono "simboli" fissi e servono ad indicare che il contenuto del registro contatore che segue contiene il numero di minuti settimanali.

ATTENZIONE: i registri contatori che vengono usati nelle funzioni CLOCK possono essere solo a 16 bit.

Gli esempi che seguono chiariscono quanto appena illustrato.

Esempi:

`O1.1 = CLOCK(C0, C2)`	Accensione giornaliera all'orario specificato dal contenuto del registro C0 e spegnimento all'orario specificato dal registro C2. Se C0 contiene il valore 480 e C2 contiene il valore 1020, allora l'uscita sarà accesa dalle 8:00 alle 17:00
`O1.1 = CLOCK(MAR:C0, GIO:C2)`	Accensione settimanale dal Martedi all'orario specificato dal contenuto del registro C0 e spegnimento il Giovedi all'orario specificato dal registro C2. Se C0 contiene il valore 675 e C2 contiene il valore 1422, allora l'uscita sarà accesa dal Martedi alle 11:15 al Giovedi alle 23:42.
`O1.1 = CLOCK(xxx:C0, xxx:C2)`	Accensione dal giorno ed orario specificato dal contenuto del registro C0 al giorno ed orario specificato dal registro C2. Se C0 contiene il valore 675 e C2 contiene il valore 6780, allora l'uscita sarà accesa dal Lunedi alle 11:15 al Venerdi alle 17:00.
`O1.1 = CLOCK(8:C0, 8:C2)`	Questa equazione è esattamente identica alla precedente.

Come già accennato è possibile programmare più fasce orarie combinando tra loro diverse coppie **ON,OFF** mediante l'operatore | (OR) come indicato negli esempi che seguono. **L'operatore OR è l'unico ammesso nella funzione** **CLOCK**.

Esempi:

`V4 = CLOCK(MAR:8:00, MAR:12:00 | GIO:14:30, SAB:00:00)`	Attivazione di **V4** dalle 8:00 alle 12:00 del martedì e dalle 14:30 del Giovedì alla mezzanotte del Venerdì (equivalente alle 0:00 del Sabato).
`V2 = CLOCK(LUN:C4, MAR:C6 | VEN:14:30, SAB:00:00)`	Attivazione di **V2** il Lunedi dall'orario specificato in C4 al Martedi all'orario specificato in C6 e dalle 14:30 del Venerdi alle 0:00 del Sabato (equivalente alla mezzanotte del Venerdi). In questo esempio i giorni di accensione e spegnimento sono fissi.
`V1 = CLOCK(xxx:C0, xxx:C2 | xxx:C4, xxx:C6)`	Attivazione di **V1** dall'orario specificato in C0 all'orario specificato in C2 e dall'orario specificato in C4 all'orario specificato in C6. I registri contatori, in questo caso contengono anche l'informazione del giorno della settimana.

2.3.7- Analogico

Benchè l'equazione per il riporto di un segnale da un modulo di ingresso analogico ad un modulo di uscita analogico può essere sostituita da 8 relazioni punto-punto fra ingresso e uscita (specificando che ogni bit dell'uscita sia uguale al corrispondente bit dell'ingresso), l'uso dell'operatore analogico consente una maggiore velocità di elaborazione e la possibilità di eseguire semplici operazioni aritmetiche sui valori letti dai moduli. L'equazione nel formato più semplice è la seguente:

Ox = Coeff Ai

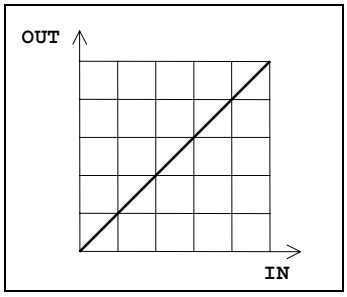
dove **Ox** indica il modulo di uscita analogico con indirizzo **x**, **Ai** è l' ingresso analogico di indirizzo **i**, e **Coeff** è il fattore di scala che può essere compreso tra 1 e 256.
Per l'operatore analogico vale quanto segue:

1. Il termine base **Coeff Ai** può mancare del parametro **Coeff** (in tal caso è assunto uguale ad 1) o dell'ingresso **Ai** (in tal caso viene considerato un termine costante pari a **Coeff**).
2. Più termini base **Coeff Ai** possono essere sommati, sottratti, moltiplicati e divisi tra loro. Le operazioni algebriche indicate vengono eseguite nell'ordine scritto, da sinistra a destra. Gli operatori per le operazioni algebriche sono **+** (somma), **-** (sottrazione), ***** (moltiplicazione) e **/** (divisione).
3. Non sono ammesse parentesi.

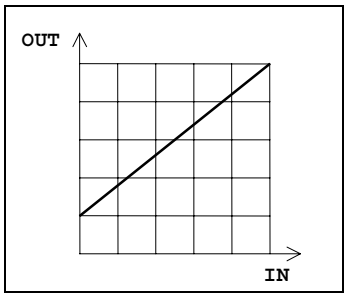
4. Ogni termine è il prodotto di due numeri di 8 bit (**Coeff** moltiplicato il valore del segnale di ingresso) . Il risultato è un termine di 16 bit. Le operazioni vengono effettuate considerando numeri a 16 bit ma vengono comunque limitate a valori rappresentabili con 16 bit, per cui se si eccede il valore 65535 (massimo numero ottenibile con 16 bit) il risultato viene posto uguale a 65535. Se in seguito alla sottrazione si ottiene un numero negativo, il risultato viene posto uguale a 0. Le limitazioni del risultato valgono sia per il risultato finale che per quelli parziali.
5. Il valore finale viene troncato a un numero di 8 bit (max 255) per poter essere assegnato all'uscita; in altre parole, se il valore finale eccede 255 allora viene posto uguale a 255.

Esempi:

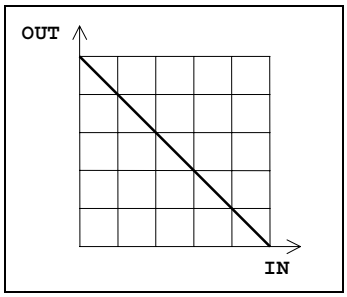
O1 = A2 Semplice riporto di un segnale dall'ingresso analogico 2 all'uscita analogica 1.



O1 = 4A1 / 5 + 51 L'equazione scala di quattro quinti il valore di ingresso e aggiunge un offset (51).



O1 = 255 - A1 Uscita complementare rispetto all'ingresso.



O1 = A1 + 3A2 Somma di A1 e 3 volte A2. Se A1 = 10 e A2 = 5 allora l'uscita risulta uguale a 25

O1 = A1 + 3 * A2 Prodotto di A2 per la somma di A1 e 3: si noti la differenza con l'esempio precedente. In questo caso viene eseguita la somma A1 + 3 e il risultato viene poi moltiplicato per A2. Se A1 = 10 e A2 = 5 allora l'uscita è pari a 65. In questo caso l'equazione è costituita da tre termini anzichè due come nell'esempio precedente.

2.3.8- Toggle

L'operatore toggle, lettera **T**, inverte lo stato dell'uscita ad ogni variazione OFF→ON dello stato dell'ingresso. Lo stato dell'ingresso viene quindi ignorato fino a successive sue variazioni. L'utilizzo più comune di questo operatore è la simulazione di relè **PASSO-PASSO.** L'operatore TOGGLE permette inoltre la definizione, facoltativa, di un ingresso che forza l'accensione dell'uscita e di un ingresso che ne forza lo spegnimento, indipendentemente dallo stato dell'uscita stessa (SET e RESET asincrono). L'equazione nel formato generale è la seguente:

```
Ox.y = TIj.k | RIh.l | SIm.n
```

dove **Ij.k** è l'ingresso che genera la commutazione dell'uscita, **Ih.l** è l'ingresso che ne forza lo spegnimento e **Im.n** è l'ingresso che ne forza l'accensione.
L'uscita e gli ingressi che rientrano nell'equazione dell'operatore TOGGLE **possono essere anche virtuali**. E' inoltre possibile provocare la commutazione, il set ed il reset dell'uscita **da più ingressi utilizzando il simbolo | (OR)** come illustrato negli esempi che seguono. **Non è invece ammesso il simbolo &**.
Se si vuole invertire la logica dell'ingresso, sia esso un ingresso di toggle, di set o di reset, è sufficiente anteporre al relativo ingresso il simbolo !.

Esempi:

`O1.1 = TI6.1` Simulazione relè **PASSO-PASSO** con variazione attiva OFF→ON dell'ingresso.

`O1.1 = T!I6.1` Simulazione relè **PASSO-PASSO** con variazione attiva ON→OFF dell'ingresso. Il simbolo **!** modifica infatti la variazione dell'ingresso che fa commutare l'uscita.

`V100 = TI1.1 | TI1.2` In questo caso l'uscita comandata è virtuale.

`O1.1 = TI1.1 | TI1.2 | SI1.3 | SI1.4 | RI1.5 | RI1.6` Sono presenti due pulsanti per il comando di commutazione della uscita, due comandi di set e due comandi di reset.

2.3.9- Gestione modulo contatore a 16 bit

Il modulo contatore a 16 bit (codice MODCNT) **è un modulo esterno** che conta gli impulsi presenti ai suoi ingressi e memorizza il totale del conteggio nella propria memoria non volatile. Il trattamento di questo modulo da parte di MCP richiede una gestione particolare, specificata da una equazione non molto diversa da quella che gestisce i contatori interni. Ogni modulo contatore può avere più di un ingresso di conteggio (4 nel caso del modulo MODCNT), per cui **è necessario indicare, nell'equazione, il numero del canale sul quale si vuole operare**. La sintassi generale è la seguente:

```
Ox.y = CIj.k {op} T Zip.q
```

dove:

`Ox.y` rappresenta l'uscita da controllare (può anche essere un punto virtuale)
`CIj.k` indica il modulo contatore avente indirizzo **j** (compreso tra 1 e 127) ed il numero del canale **k** (compreso tra 1 e 4)

`ZIp.q` è l'ingresso (reale o virtuale) per l'azzeramento del contatore; questo ingresso è facoltativo

`{op}` è la regola di confronto tra contenuto del contatore e soglia `T`.
`{op}` può essere uno dei seguenti simboli:
- `=` (uguale)
- `>` (maggiore o uguale)
- `<` (minore o uguale)

`T` è la soglia ed è un valore numerico compreso tra 0 e 65535. Il valore `T` può essere anche un puntatore ad un altro contatore, vale a dire che si può definire, come valore di soglia, **il contenuto del registro di un contatore**

Come appena enunciato, **è possibile definire, come valore di soglia `T`, il contenuto di un registro contatore**; in questo modo è possibile avere **contatori a soglia variabile** e quindi impostabile da supervisore (il quale andrà a scrivere il valore voluto nel relativo registro), oppure impostabile mediante UP/DOWN del contatore che contiene il valore di soglia.
I registri contatori usati come soglia di questa funzione devono essere *esclusivamente* a 16 bit.

Esempi:

`O1.1 = CI10.1 > 100 ZI1.1` L'uscita è controllata dal contatore 1 del modulo di indirizzo 10; l'uscita è attiva se il conteggio è maggiore o uguale a 100. L'ingresso `I1.1`, quando attivato, azzera il contenuto del contatore.

`V10 = CI1.4 > C2` La virtuale `V10` è attivata quando il conteggio del canale 4 del modulo contatore 1 è uguale o maggiore del contenuto del registro contatore `C2` (che deve essere a 16 bit).

2.3.10- Gestione modulo analogico di ingresso a 16 bit

La gestione del modulo di ingresso analogico a 16 bit da parte di MCP richiede una gestione particolare, specificata da una equazione non molto diversa da quella dell'operatore soglia già esposta per i moduli a 8 bit. Ogni modulo analogico a 16 bit può avere più di un ingresso, per cui **è necessario indicare, nell'equazione, il numero del canale sul quale si vuole operare**. La sintassi generale è la seguente:

```
Ox.y = Ai.j {op} T,H
```

dove:

`Ox.y` è l'uscita controllata dalla funzione soglia (può essere anche virtuale)

`Ai.j` identifica il modulo di ingresso analogico a 16 bit di indirizzo `i` ed il canale `j`

`{op}` è l'operatore di confronto che può essere:
- `>` (maggiore o uguale)
- `<` (minore o uguale)

`T` è la soglia, ovvero un valore numerico compreso tra 0 e 65535 oppure l'identificativo di un registro contatore che contiene il valore di soglia

`H` è il valore di isteresi, ovvero un valore numerico compreso tra 0 e 65535 oppure l'identificativo di un registro contatore che contiene il valore di isteresi

Come per l'operatore soglia standard, è possibile legare tra loro più termini mediante operatori AND (`&`) e OR (|).

Come sopra ricordato, la versione 4.x di MCP (o superiori) **consente di definire, come valore di soglia `T` e/o come valore di isteresi `H`, il contenuto di un registro contatore**; in questo modo è possibile avere **funzioni a soglia e isteresi variabili** e quindi impostabili da supervisore (il quale andrà a scrivere il valore voluto nel relativo registro), oppure impostabile mediante UP/DOWN del contatore che contiene il valore di soglia o di isteresi. **I registri contatori che vengono usati nelle funzioni soglia con moduli di ingresso analogici del tipo qui descritto possono essere esclusivamente 16 bit**. Gli esempi che seguono chiariscono quanto appena illustrato.

Esempi:

`O1.1 = A1.1 > 2400,2`	L'uscita è attiva quando l'equivalente binario del segnale analogico applicato al canale 1 del modulo 1 è maggiore o uguale a 2400; l'isteresi è pari a 2 step.
`V2 = A1.4 < 400,2 | A2.3 < 300,4`	La virtuale `V2` è attivata quando il canale 4 del modulo 1 è minore o uguale a 400, oppure quando il canale 3 del modulo 2 è minore o uguale a 300; le isteresi sono rispettivamente 2 step e 4 step.
`V8 = A1.3 > C1,20`	La virtuale `V8` è attivata quando il canale 3 del modulo 1 è maggiore o uguale al contenuto del registro contatore `C1` (a 16 bit); l'isteresi è fissa e pari a 20.
`V8 = A1.3 > C1,C4`	La virtuale `V8` è attivata quando il canale 3 del modulo 1 è maggiore o uguale al contenuto del registro contatore `C1` (a 16 bit); l'isteresi è pari al contenuto del registro contatore `C4`.

2.3.11- Configurazione

La configurazione del sistema, vale a dire quanti e quali moduli sono collegati al bus del modulo MCP, viene dedotta automaticamente dalle equazioni scritte. In base agli ingressi e alle uscite viene costruita la mappa dei moduli presenti che verranno inseriti nel ciclo di polling di MCP.
E´ possibile che, per particolari esigenze, il sistema abbia **alcuni moduli che non siano necessariamente legati ad alcuna equazione**, come ad esempio moduli di ingresso per la rilevazione di segnali tramite un supervisore, moduli di uscita comandati esclusivamente da computer, ecc. **In tal caso è comunque necessario segnalarne la presenza al MCP**, in modo tale da poter essere interrogati e rilevarne sia lo stato che un eventuale malfunzionamento.
L'equazione che descrive la configurazione dei moduli aggiuntivi (che non rientrano nelle equazioni e quindi non automaticamente inseriti nella mappa di configurazione) è la seguente:

```
CONF ( Ii & Ij & … & Ox & Oy & … & Ck & Cu & … & Av & Aw & … )
```

dove `i, j, x, y,` ecc. sono gli indirizzi dei moduli (1÷127) e le lettere `I` e `O` indicano se l'indirizzo che segue è relativo ad un modulo di ingresso o di uscita; i termini `C` identificano i moduli contatori multicanale MODCNT di indirizzo `K`, `u`, ecc. (`k`, `u`, ecc. compresi tra 1 e 127); i termini `A` identificano i moduli analogici di ingresso a 16 bit di indirizzo `v`, `w`, ecc. (`v`, `w`, ecc. compresi tra 1 e 127).
Il simbolo `&` è obbligatorio e l'equazione può essere spezzata su più righe utilizzando il solito simbolo `\` di andata a capo (vedere capitolo riguardante la scrittura delle equazioni).
L'operatore `CONF` non può essere applicato ai punti virtuali.

Esempi:

`CONF ( I1 & I12 & O4 & O65 )` Aggiunge alla mappa del sistema i moduli di ingresso con indirizzo 1 e 12 ed i moduli di uscita 4 e 65.

`CONF ( I1 & I2 & O7 & O22 & C50 & C51 & A80 )` Aggiunge alla mappa del sistema i moduli di ingresso con indirizzo 1 e 2, i moduli di uscita 7 e 22, i moduli contatori con indirizzo 50 e 51 ed il modulo analogico con indirizzo 80.

2.3.12- Indirizzo

È possibile collegare più moduli MCP su una rete RS485 (con opportuni convertitori di segnale per MCP standard o direttamente con MCP Plus). In questo caso ogni MCP deve essere univocamente individuabile nella rete mediante un numero detto indirizzo. L'indirizzo viene assegnato mediante l'equazione:

`ADDRESS = n`

dove `n` è un numero fra 1 e 255 che identifica il modulo MCP. **L´ indirizzo 0 è riconosciuto da ogni MCP**, indipendentemente dal proprio indirizzo. Può essere pertanto utilizzato per comunicare con qualsiasi MCP assicurandosi che un solo modulo alla volta sia collegato, per evitare conflitti nella comunicazione.

ATTENZIONE: nonostante l'indirizzo possa assumere valori da 1 a 255, il numero "fisico" di MCP collegati sulla stessa linea RS485 deve essere minore o uguale a 31.

Esempio:

`ADDRESS = 4` Assegnazione dell'indirizzo 4. Il modulo MCP così programmato risponderà alle richieste che contengono nel campo indirizzo i valori 4 e 0 (quest'ultimo è detto anche indirizzo jolly).

2.3.13- Codice di identificazione

Ogni MCP può essere identificato con un nome. Il codice di identificazione è utile per il riconoscimento dell'unità in reti con più MCP (utilizzando la linea RS485) oppure per la gestione tramite modem.
Il codice di identificazione è una stringa ***alfanumerica*** composta al massimo di **63 caratteri** e viene assegnato con la seguente equazione:

`ID = (Codice di Identificazione)`

dove tra parentesi è indicato il codice desiderato.
Non è possibile utilizzare spazi e parentesi tonde all'interno della stringa di identificazione. Per una maggiore leggibilità si consiglia di usare, al posto degli spazi, il carattere ´_´ (sottolineatura o underscore).

Esempio:

`ID = (Impianto_cabina_23_MILANO)` Assegnazione del codice di identificazione. Gli spazi sono sostituiti dal carattere sottolineatura (underscore).

2.3.14- Memorizzazione eventi

La versione 4.x di MCP consente di memorizzare, in ordine cronologico insieme a data e ora, **sino a 1023** eventi corrispondenti alla variazione di stato di uno o più punti che possono essere **solo virtuali**. L'equazione che definisce gli ingressi virtuali che, se attivi, generano la memorizzazione dell'evento è:

```
EVENT ( \
      Vn {ON} {OFF} \
      ... \
      )
```

dove si è indicato con:

Vn Il punto virtuale la cui variazione di stato deve essere memorizzata

ON indica che la memorizzazione viene effettuata sulla transizione 0⟶1 del punto virtuale

OFF indica che la memorizzazione viene effettuata sulla transizione 1⟶0 del punto virtuale

Quando viene definita la funzione **EVENT**, MCP riserva automaticamente una porzione di RAM per memorizzare in ordine cronologico gli eventi specificati nel blocco (1023 max). **Ognuno di questi eventi viene memorizzato con il numero del punto virtuale cui si riferisce, stato del punto, data e ora dell'occorrenza dell'evento**.

Questa lista di eventi può essere letta e azzerata da PC o supervisore collegato alla porta seriale di MCP, agendo sulla memoria RAM di MCP come qui descritto:

- gli eventi sono memorizzati a partire dall'indirizzo 2000h (esadecimale)
- i due byte 2000h e 2001h contengono il numero di eventi memorizzati (2000h è il byte più significativo)
- gli indirizzi 2002h e 2003h non sono utilizzati
- le informazioni riguardanti ciascun evento sono memorizzate a partire dall'indirizzo 2004h in blocchi da 4 byte per evento, ognuno dei quali contiene:

<table>
<tr><td>-</td><td colspan="3">Indirizzo n punto virtuale (7 bit)</td></tr>
<tr><td colspan="2">Mese (4 bit)</td><td>Stato</td><td>Punto p (3 bit)</td></tr>
<tr><td colspan="3">Giorno (5 bit)</td><td>Ora MSB (3 bit)</td></tr>
<tr><td>Ora LSB (2)</td><td colspan="3">Minuti (6 bit)</td></tr>
</table>

Il numero del punto virtuale associato all'evento è dato da `(n-1)*8 + (p+1)`; *Stato* indica lo stato del punto virtuale, *Mese*, *Giorno*, *Ora* e *Minuti* indicano il momento in cui l'evento si è verificato.

La coda è cronologica e può contenere fino a 1023 blocchi. **Ulteriori eventi vengono persi**. Il supervisore, per azzerare la lista degli eventi, deve semplicemente resettare il numero di eventi presenti, il che significa che deve azzerare i due byte agli indirizzi 2000h e 2001h.

Note:

1. I punti virtuali devono essere specificati da apposite equazioni contenenti combinazioni di ingressi reali e/o virtuali. **Se i parametri ON e OFF sono presenti entrambi**, la memorizzazione avviene sia sulla variazione 1⟶0 che su quella 0⟶1
2. E' possibile memorizzare il verificarsi di una condizione di modulo guasto o di bus guasto utilizzando i punti virtuali V999 e V1000 (vedi paragrafo relativo sul manuale di MCP)
3. Il simbolo \ di andata a capo (vedi capitolo riguardante la scrittura delle equazioni) è obbligatorio

Esempio:

```
V1 = I2.1                       Definizione evento 1
V2 = I2.2 | I2.3 | I2.4         Definizione evento 2
V3 = I2.5 & I2.6                Definizione evento 3

EVENT (             \           Inizio blocco
       V1 ON        \           Evento 1, alla transizione 0→1 di V1
       V2 OFF       \           Evento 2, alla transizione 1→0 di V2
       V3 ON OFF    \           Evento 3, ad entrambe le transizioni 0→1 e 1→0 di V3
)
```

2.3.15- Modem

Il modulo MCP consente di essere programmato in modo da generare una chiamata via modem al verificarsi di transizioni su uno o più punti che possono essere **solo virtuali**. L'equazione che definisce gli ingressi virtuali che, se attivi, provocano una chiamata via modem è la seguente:

```
MODEM ( \
      TEL1 = x \
      TEL2 = y \
      TEL3 = w \
      TEL4 = z \
      Vn {ON} {OFF} \
      ... \
      )
```

dove si è indicato con:

`Vn`	il punto virtuale la cui transizione deve generare la chiamata via modem
`ON`	specifica che la chiamata deve avvenire alla transizione 0→1 del punto virtuale
`OFF`	specifica che la chiamata deve avvenire alla transizione 1→0 del punto virtuale
`x,y,w,z`	numeri telefonici da chiamare (completi di prefisso)

Il simbolo \ di andata a capo (vedi capitolo riguardante la scrittura delle equazioni) è obbligatorio.
Nel blocco MODEM è **possibile includere sino a 3 stringhe di comandi AT** per configurare automaticamente il modem alla accensione del modulo MCP; la lunghezza massima di ognuna di queste stringhe è di 17 caratteri. Le stringhe di configurazione devono essere posizionate, all'interno del blocco MODEM, prima dei 4 numeri di telefono; l'esempio riportato alla fine di questo paragrafo mostra l'utilizzo delle stringhe di configurazione.

Note:

1. I punti virtuali devono essere specificati da apposite equazioni contenenti combinazioni di ingressi reali e/o virtuali. Se i parametri ON e OFF sono presenti entrambi, la chiamata viene effettuata sia sulla variazione 1→0 che su quella 0→1
2. È possibile generare la chiamata anche nel caso in cui si verifica una condizione di modulo guasto o di bus guasto, utilizzando i punti virtuali V999 e V1000 (vedi relativo paragrafo)
3. È possibile inserire sino a quattro numeri telefonici TEL1, TEL2, TEL3 e TEL4. L'ordine di chiamata è da TEL1 a TEL4. **E´ obbligatorio definire tutti i quattro numeri** (sono ammesse anche ripetizioni dello stesso numero)
4. I numeri telefonici possono essere di **15 cifre max**
5. Non è obbligatorio, anche se raccomandabile, definire le stringhe di configurazione MODEM

Al verificarsi di uno degli eventi, MCP chiama, attraverso il modem, il primo numero specificato; se questo è occupato o non risponde entro un tempo limite di 90 secondi, la chiamata viene ripetuta per altre due volte, poi MCP passa al numero successivo. Se dopo tutti i tentativi effettuati MCP non ha ricevuto risposta, allora la sequenza si ferma; se invece almeno uno dei numeri chiamati era occupato, allora la sequenza riparte dal primo numero.
La postazione che risponde alla chiamata **deve inviare a MCP una richiesta del codice di identificazione per confermare l'avvenuta acquisizione degli allarmi; in caso contrario MCP procede nella sequenza**.

Quando viene definita la funzione modem, MCP riserva automaticamente una porzione di RAM per memorizzare in ordine cronologico **sino a 127** degli eventi specificati nel blocco. **Ognuno di questi eventi viene memorizzato con il numero del punto virtuale cui si riferisce, stato del punto, data e ora dell'occorrenza dell'evento.**
ATTENZIONE: la mappatura degli eventi all'interno della memoria RAM di MCP versione 4.x è variata rispetto alle precedenti come qui di seguito riportato:

- gli eventi sono memorizzati a partire dall'indirizzo 300h (esadecimale)
- il byte 300h contiene il numero di allarmi memorizzati
- gli indirizzi 301h, 302h e 303h non sono utilizzati
- le informazioni riguardanti ciascun evento sono memorizzate a partire dall'indirizzo 304h in blocchi da 4 byte per allarme, ognuno dei quali contiene:

<table>
<tr><td>-</td><td colspan="3">Indirizzo n punto virtuale (7 bit)</td></tr>
<tr><td colspan="2">Mese (4 bit)</td><td>Stato</td><td>Punto p (3 bit)</td></tr>
<tr><td colspan="3">Giorno (5 bit)</td><td>Ora MSB (3 bit)</td></tr>
<tr><td>Ora LSB (2)</td><td colspan="3">Minuti (6 bit)</td></tr>
</table>

Il numero del punto virtuale associato all'allarme è dato da `(n-1)*8 + (p+1)`; *Stato* indica lo stato del punto virtuale, *Mese*, *Giorno*, *Ora* e *Minuti* indicano il momento in cui l'evento si è verificato.
La coda è cronologica e può contenere **fino a 127 blocchi. Ulteriori allarmi vengono persi**. Il supervisore, per azzerare la lista degli allarmi modem, deve semplicemente resettare il numero di eventi presenti, il che significa che deve azzerare il byte all'indirizzo 300h.

La lista degli eventi relativi alla funzione **`MODEM`** può essere letta e azzerata da PC o supervisore collegato alla porta seriale di MCP.

Esempi:

```
V1 = I2.1                       Definizione allarme 1
V2 = I2.2 | I2.3 | I2.4         Definizione allarme 2
V3 = I2.5 & I2.6                Definizione allarme 3

MODEM ( \                       Inizio blocco
   AT&F0E0          \           Stringa 1 di configurazione MODEM
   AT-K0X3&K0&D0    \           Stringa 2 di configurazione MODEM
   ATT&WZ           \           Stringa 3 di configurazione MODEM
   TEL1 = 02112233 \            Numero telefonico 1
   TEL2 = 12345678 \            Numero telefonico 2
   TEL3 = 112       \           Numero telefonico 3
   TEL4 = 113       \           Numero telefonico 4
```

```
   V1 ON          \      Allarme 1, alla transizione 0→1 di V1
   V2 OFF         \      Allarme 2, alla transizione 1→0 di V2
   V3 ON OFF      \      Allarme 3, ad entrambe le transizioni 0→1 e 1→0 di V3
)                        Fine blocco
```

Note: le stringhe di configurazione nell'esempio riportato si riferiscono ad un MODEM DIGICOM. Applicando l'alimentazione al modulo MCP, a modem acceso e collegato via RS232, le stringhe di configurazione vengono trasferite; è raccomandabile includere sempre queste stringhe nel blocco MODEM.
I numeri di telefono possono contenere anche caratteri speciali, come la virgola per immettere qualche secondo di pausa prima di proseguire nella composizione del numero (ad esempio quando la chiamata viene effettuata da centralino che richiede lo 0 per accedere alla linea esterna).

Il collegamento seriale RS232 tra modulo MCP e modem è a 3 fili (TX, RX e comune).

Per maggiori informazioni riguardante la configurazione del modem per un corretto interfacciamento con MCP, è disponibile una nota applicativa sull'argomento.

2.3.16- Protocollo

Il modulo **MCP Plus** è in grado di scambiare informazioni con il mondo esterno attraverso il protocollo MODBUS (RTU Protocol) oppure con il protocollo JOHNSON CONTROL VND. Questi protocolli sono direttamente integrati in MCP Plus e convivono con il protocollo proprietario FXP (vedi nel seguito).
I protocolli MODBUS e JOHNSON sono normalmente disabilitati; per attivare il protocollo MODBUS è necessario includere nel programma di MCP Plus la seguente direttiva:

```
PROTOCOL = MODBUS
```

Allo stesso modo per attivare il protocollo JOHNSON è necessario includere nel programma di MCP Plus la seguente direttiva:

```
PROTOCOL = JC
```

Il protocollo proprietario FXP (vale a dire il normale protocollo di comunicazione di MCP) è sempre attivo, nel senso che ad una richiesta secondo il protocollo proprietario, MCP Plus risponde secondo lo stesso protocollo proprietario.

Note:
- I protocolli MODBUS e JOHNSON **non** possono essere contemporaneamente abilitati.
- Nel caso fosse abilitato il protocollo JOHNSON, MCP Plus risponderà solo a richieste con protocollo FXP di indirizzo zero (indirizzo jolly); la risposta di MCP Plus conterrà sempre l'indirizzo definito mediante la direttiva ADDRESS.
- Nel caso fosse abilitato il protocollo MODBUS, MCP Plus risponderà solo a richieste con protocollo FXP di indirizzo zero (indirizzo jolly) e di indirizzo definito mediante la direttiva ADDRESS; la risposta di MCP Plus, in questo caso, conterrà sempre l'indirizzo definito mediante la direttiva ADDRESS.

3- SCRITTURA DELLE EQUAZIONI

La scrittura delle equazioni è la prima fase della programmazione del modulo MCP. Le equazioni devono essere scritte secondo le sintassi riportate nei precedenti paragrafi.

Per scrivere le equazioni **si deve utilizzare il software di supporto MCPTOOLS** fornito da **DUEMMEGI** insieme al modulo MCP; questo programma lavora su Personal Computer in ambiente WINDOWS® e, oltre alle operazioni di scrittura delle equazioni, consente la una facile messa in servizio. ***Per i dettagli sull'utilizzo di questo programma si rimanda alla relativa documentazione o all'help in linea dello stesso.***

Essenzialmente il software di supporto MCPTOOLS comprende:

- un editor di testi per la scrittura del programma
- un compilatore che consente di tradurre il file ASCII contenente le equazioni di funzionamento in un file binario adatto ad essere trasferito nella memoria non volatile (di tipo FLASH) del modulo MCP
- un simulatore per la verifica del programma, o di parte di esso, prima che questo venga trasferito nella memoria di MCP
- una sezione che gestisce il trasferimento del programma a MCP (o viceversa)
- un visualizzatore di mappa, ossia la rappresentazione grafica dello stato dei moduli in campo (ingressi e uscite), stato dei contatori, mappa dei punti virtuali ecc.

Il file che contiene le equazioni è in formato ASCII e deve avere estensione **.EQU**; ad esempio:

***nomefile*.EQU**

dove *nomefile* è il nome del file contenente le equazioni e può essere qualsiasi (nei limiti della sintassi consentita da WINDOWS®).
L'estensione .EQU è **obbligatoria** in quanto le fasi successive (compilazione e programmazione) richiedono che il file di partenza abbia tale estensione.

La programmazione del modulo MCP avviene in 3 fasi successive, tutte supportate dal software MCPTOOLS:

1. creazione (o editazione) del file *nomefile.EQU* contenente le equazioni in formato leggibile (ASCII)
2. compilazione di *nomefile.EQU*, vale a dire conversione del file ASCII in un file *nomefile*.BIN scritto in un formato adatto ad essere trasferito nella memoria del modulo MCP
3. trasferimento di *nomefile*.BIN nella memoria di MCP

Se durante la fase 2 vengono rilevati errori sintattici, questi vengono segnalati dal compilatore di MCPTOOLS insieme ad informazioni circa il tipo di errore e il numero di riga ove si è verificato.

3.1- Regole per la scrittura delle equazioni

Le equazioni devono essere scritte rispettando la sintassi descritta nei relativi paragrafi (logica, contatore, timer, ecc ...).

Valgono inoltre le seguenti regole:

- Gli spazi e il carattere di tabulazione non sono significativi. Essi vengono ignorati in fase di compilazione. **Si raccomanda vivamente, in ogni caso, di dividere i vari termini ed i vari operatori di una equazione con almeno uno spazio, in modo da rendere più facilmente leggibile il programma**
- Un'equazione può essere divisa su più righe utilizzando il carattere di "andata a capo" \ (barra rovesciata) alla fine della riga per indicare il suo proseguimento sulla riga successiva
- L'equazione termina alla fine della riga (se non è presente il carattere \ )
- I caratteri `//` (due barre diritte consecutive) indicano un commento: tutto ciò che segue tali caratteri (essi inclusi) fino alla fine della riga è considerato commento e ignorato in fase di compilazione. **I commenti risultano molto utili per una maggiore chiarezza e documentazione del programma,** soprattutto in caso di modifche effettuate in un secondo tempo. Si consiglia quindi di usarli sempre per descrivere ogni equazione
- Si possono utilizzare indifferentemente caratteri maiuscoli o minuscoli

In alternativa agli identificatori degli ingressi e delle uscite **`Ij.k, Ox.y, Vn, Aj`** è possibile utilizzare variabili definite dall´utente tramite la direttiva **`#define`** come di seguito illustrato:

```
#define     %Pompa1%    O1.1  // Definizione uscita
#define     %Comando%   I1.1  // Definizione ingresso

%Pompa1% = %Comando%          // Equazione
```

L'equazione precedente è identica a:

```
O1.1 = I1.1
```

ma chiaramente è più facilmente leggibile ed interpretabile. Le variabili definite con la direttiva **`#define`** **devono essere racchiuse tra due caratteri `%` e non possono contenere spazi o altri caratteri `%`**. Inoltre non vi è distinzione fra maiuscolo e minuscolo.

Esempio di file di equazioni:

```
//////////////////////
// Definizioni /////////
//////////////////////
#define     %LuceScale%       O1.1
#define     %ComPiano1%       I1.1
#define     %ComPiano2%       I1.2
#define     %ComPiano3%       I1.3

// Punti virtuali
V1 = %ComPiano1% | %ComPiano2% | %ComPiano3%  // OR dei pulsanti di comando

// Comando Uscite
%LuceScale% = TIMER ( V1, 0, 100)             // Accensione luce scale
```

L'esempio illustrato comanda la luce scale in seguito alla pressione di uno dei tre pulsanti di comando posti sui tre piani. La luce rimane accesa per 10 secondi, a partire dal rilascio del pulsante, dopo di che si spegne automaticamente. Lo stesso file può essere scritto, senza utilizzare le definizioni, nel modo seguente:

```
// Comandi da pulsanti
V1 = I1.1 | I1.2 | I1.3         // OR dei pulsanti di comando
// Comando Uscite
O1.1 = TIMER ( V1, 0, 100)      // Accensione luce scale
```

Come si può notare, usando la notazione con i **`#define`**, la leggibilità delle equazioni risulta migliore in quanto più mnemonica.
Per la scrittura del programma non è necessario che il modulo MCP sia collegato al PC.

3.2- Compilazione delle equazioni

La compilazione è la seconda fase del processo di programmazione di MCP. **Il file contenente le equazioni scritte (estensione .EQU) deve essere compilato mediante l'apposita voce di menu di MCPTOOLS**.
Il compilatore processa le equazioni scritte, ne verifica la sintassi, la congruenza, controlla eventuali errori e compatta le informazioni in un file binario avente lo stesso nome del file .EQU di partenza ma con estensione .BIN. Il formato binario non è un formato stampabile ma risulta compatto ed adatto alle capacità di memoria di MCP.

Per la compilazione del programma non è necessario che il modulo MCP sia collegato al PC.

Se durante la compilazione vengono rilevati uno o più errori nelle equazioni, essi vengono visualizzati sullo schermo e la compilazione continua per analizzare le equazioni successive, ma alla fine non viene creato alcun file binario.

È inoltre possibile che il compilatore segnali alcune avvertenze (WARNING): queste stanno ad indicare che non sono stati rilevati errori tale da impedire la creazione del file .BIN ma che comunque ci sono alcune incongruenze che vanno verificate prima di trasferire il programma a MCP.

Per la lista del significato dei messaggi di errore generati dal compilatore, consultare l'Appendice.

3.3- Programmazione della memoria di MCP

La programmazione del modulo MCP consiste nel **trasferimento nella sua memoria FLASH del file binario** con la configurazione del sistema e la codifica delle equazioni. È questa la terza ed ultima fase dopo la scrittura e la compilazione delle equazioni.

Il trasferimento avviene, attraverso la porta seriale RS232 del PC collegata alla porta seriale di MCP, **mediante l'apposita voce di menu di MCPTOOLS**.

Il trasferimento del programma da PC a MCP richiede che il modulo MCP sia alimentato e collegato al PC mediante l'apposito cavo RS232 fornito in dotazione.

*Nota: **MCP viene fornito con la velocità di comunicazione configurata a 19200 baud**; nel caso sia necessario utilizzare una velocità più bassa, è necessario impostare la velocità voluta spostando i jumper o i microswitch interni di MCP come descritto nel prossimo capitolo.*

4- MESSA IN SERVIZIO

4.1- Connessioni

Il modulo **MCP standard** è disponibile in due versioni:

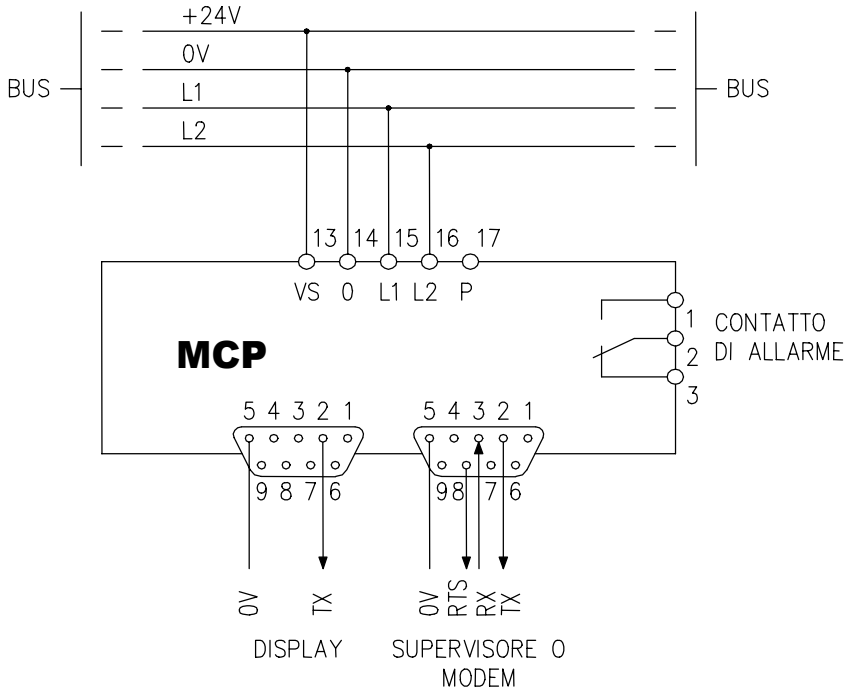
- in contenitore da tavolo denominato **MCP**
- in contenitore modulare DIN (dimensione 9 moduli) denominato **MCP/MOD**

Il modulo **MCP Plus** è disponibile solo in contenitore modulare DIN (dimensione 9 moduli)

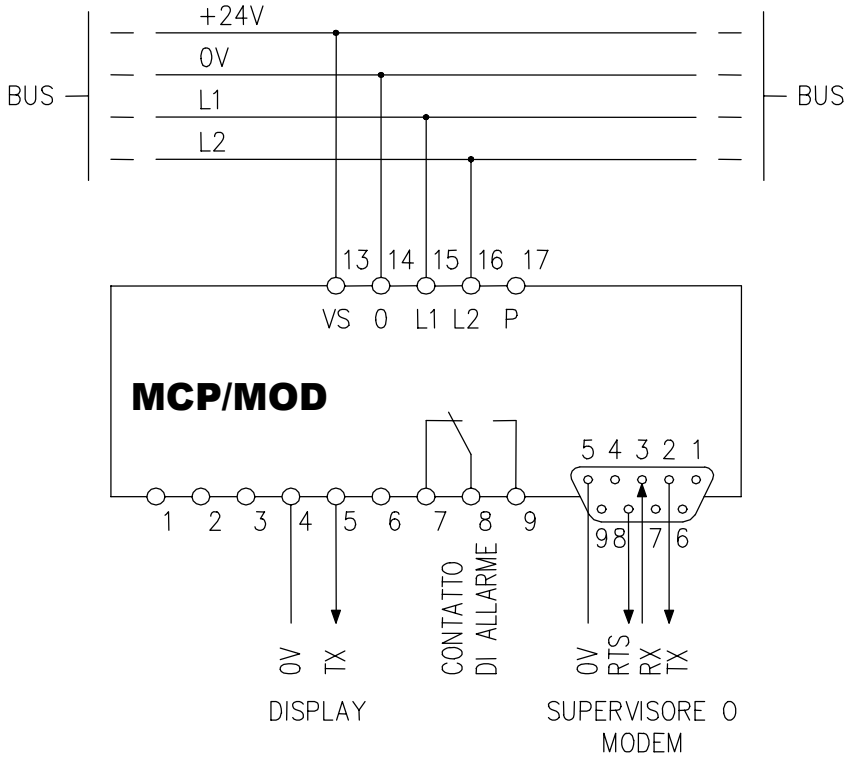
Tutte le versioni sono provviste di una morsettiera estraibile a 5 poli per il collegamento al bus di sistema; un relè interno, riportato a morsettiera, consente di segnalare eventuali malfunzionamenti di sistema (modulo guasto, bus difettoso, ecc.) mediante dispositivi esterni quali sirena, lampeggiante, ecc. . Questo relè è **eccitato in condizioni normali** e si diseccita in caso di anomalia in modo che avvenga una segnalazione di guasto anche in caso di interruzione della linea di alimentazione. Il ripristino del relè è automatico, nel senso che, quando l'anomalia scompare, esso ritorna allo stato "normale" (cioè si rieccita). Dato il funzionamento appena descritto, risulta chiaro che l'eventuale dispositivo di segnalazione di guasto **va collegato al contatto normalmente chiuso** del relè; la portata massima di questo contatto è di 5A @ 250Vca.
Una porta seriale RS232 consente il collegamento tra MCP e Personal Computer (o modem); per la versione MCP standard è inoltre disponibile una ulteriore uscita seriale RS232 per il collegamento di un eventuale visualizzatore di messaggi, mentre per la versione MCP Plus, al posto dell'uscita per il visualizzatore, è presente una porta RS485 per il collegamento in multi-drop.

Le seguenti figure mostrano i collegamenti da effettuare, per le varie versioni, verso la linea bus e la descrizione degli altri morsetti e connettori; notare che il morsetto 17 va lasciato libero.

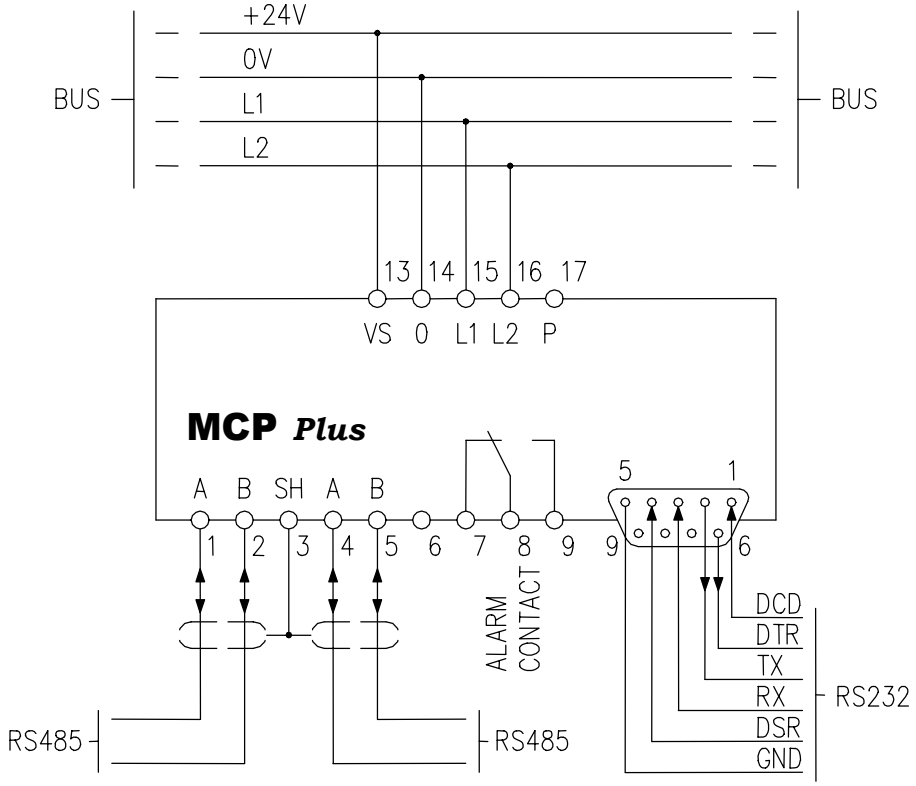
Connessioni modulo MCP



Connessioni modulo MCP/MOD



Connessioni modulo MCP Plus



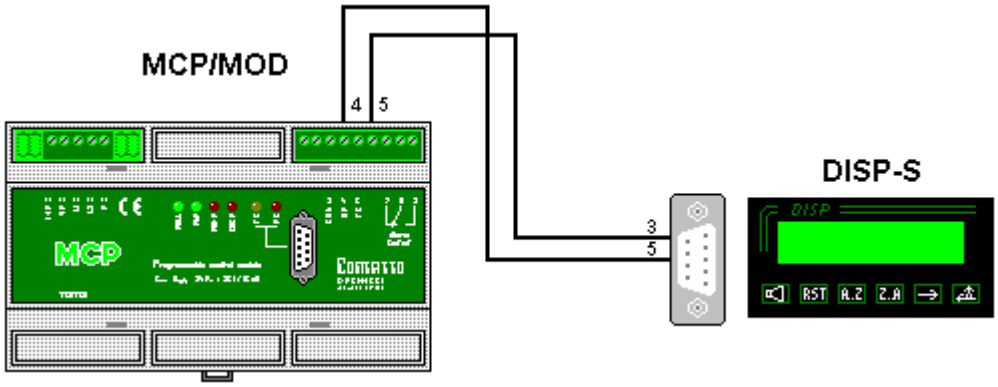
Per tutte le versioni, l'ulteriore morsettiera a 5 poli identificata dalla dicitura "TESTER" serve per il collegamento del tester **Contatto FXPRO** come descritto nel prossimo capitolo.

4.2- Collegamento del display seriale DISP-S

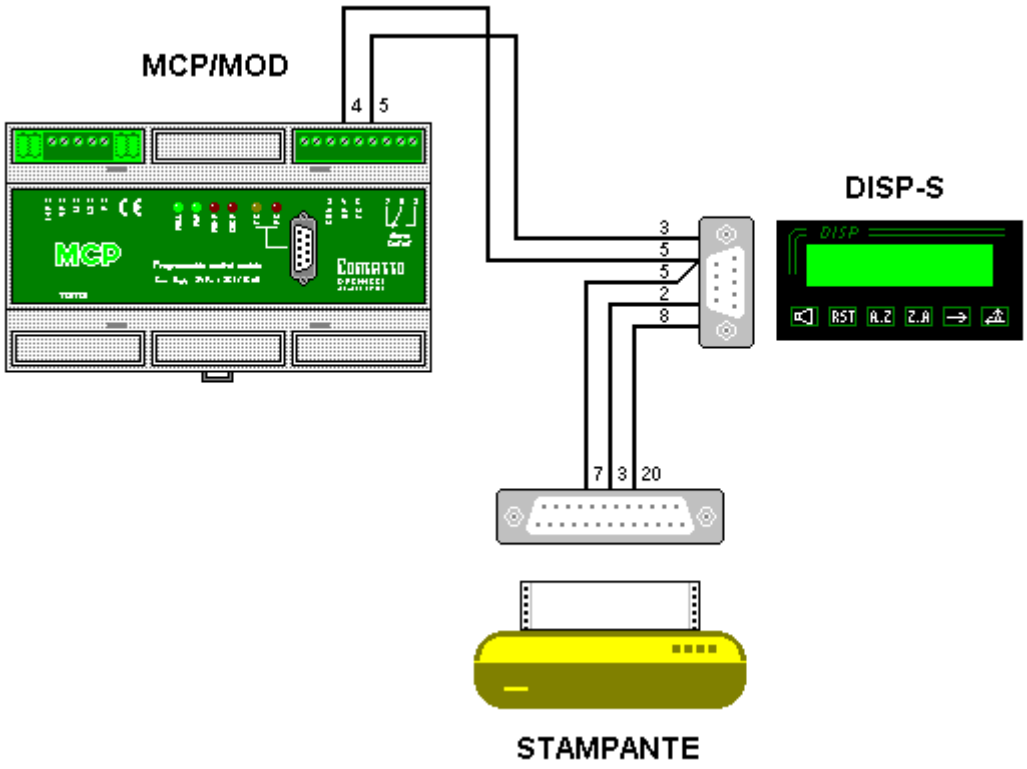
ATTENZIONE: questo paragrafo si applica solo a MCP standard.

Le figure seguenti mostrano come collegare un visualizzatore seriale **DUEMMEGI DISP-S** ed eventualmente una stampante seriale al modulo MCP standard nelle due versioni esistenti. **Tutti i connettori a vaschetta indicati sono maschi.**

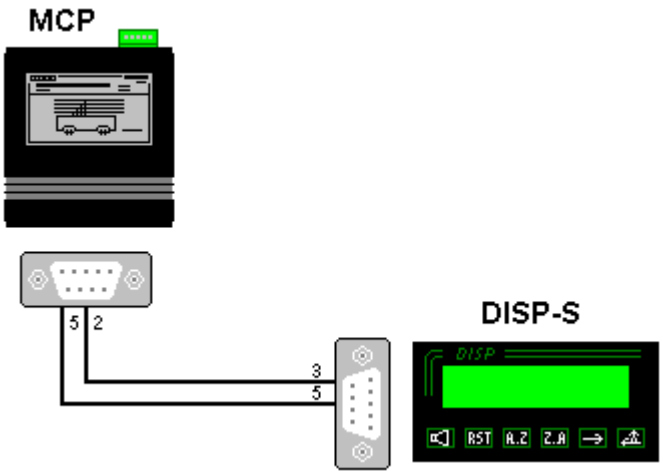
Da MCP/MOD a DISP-S



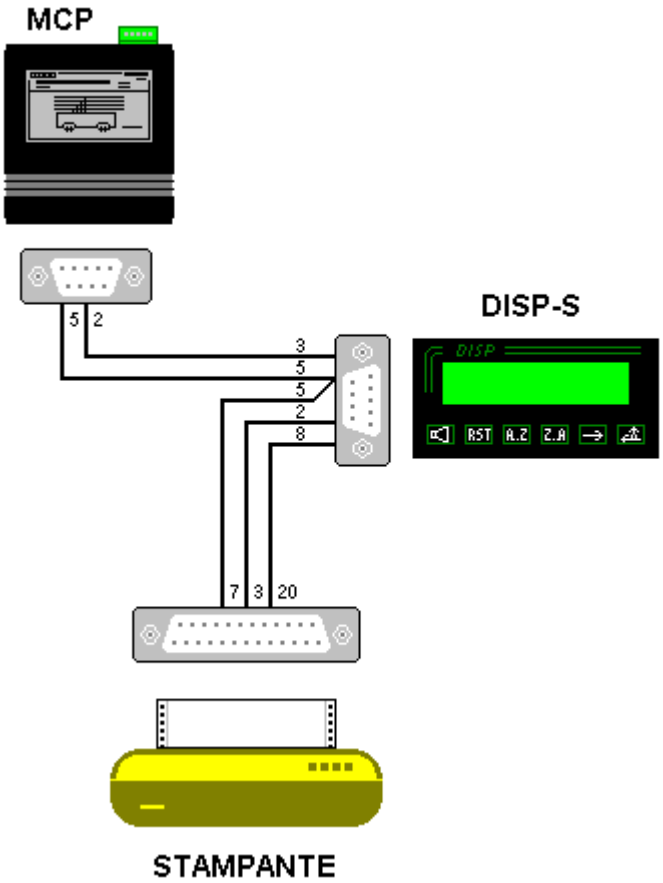
Da MCP/MOD a DISP-S e stampante seriale



Da MCP a DISP-S



Da MCP a DISP-S e stampante seriale



4.3- Selezione della velocità di comunicazione seriale

La velocità di comunicazione seriale tra MCP (sia standard che Plus) è fissata in fabbrica a 19200 Baud; se per qualsiasi motivo si volesse cambiare tale velocità, procedere come qui di seguito descritto. Le possibili velocità sono:

- 19200 Baud
- 9600 Baud
- 4800 Baud
- 2400 Baud

Per la versione da tavolo (MCP), aprire il modulo togliendo le due viti poste sul fondo del contenitore e spostare i due ponticelli (jumper) posti sul lato componenti della scheda a circuito stampato attenendosi alla tabella serigrafata sulla scheda stessa.

Per la versione modulare (MCP/MOD e MCP Plus), rimuovere il coperchietto posto tra la morsettiera bus e la morsettiera del contatto di allarme; agire quindi sul dip-switch posto sotto il coperchietto attenendosi alla tabella serigrafata sulla scheda.

Per rendere operativa la selezione fatta è necessario spegnere e poi riaccendere il modulo.

ATTENZIONE: La velocità di comunicazione sull'uscita display di MCP e MCP/MOD è sempre la stessa selezionata per la porta seriale principale (Supervisor/Modem).

4.4- Porte seriali RS232 e RS485 di MCP Plus

MCP Plus dispone sia di porta RS232 (su pannello frontale) che di porta RS485 (morsetti da 1 a 5). Queste due porte sono **galvanicamente isolate dal resto dei circuiti** mediante alcuni fotoaccoppiatori e un convertitore cc/cc interni (non è quindi richiesta alcuna alimentazione supplementare esterna).
Le due porte RS232 e RS485 **non sono però isolate tra loro**.

Le due porte sono **"mutuamente esclusive"**, in quanto non sono indipendenti tra loro e quindi non possono essere usate contemporaneamente; la commutazione da una porta all'altra avviene mediante il segnale di ingresso DSR sul connettore RS232 secondo la seguente logica:

- DSR attivo (+10V): porta RS232 abilitata e RS485 disabilitata (sia in ricezione che in trasmissione)
- DSR non attivo (-10V o non collegato): porta RS485 abilitata e RS232 disabilitata

In pratica, avendo un MCP Plus collegato in una rete RS485, è sufficiente inserire il connettore RS232 in arrivo da un PC (o altro) per disconnettere il dispositivo dalla rete e connetterlo al PC stesso (a patto ovviamente che il segnale DSR venga gestito come descritto prima); ciò facilita eventuali riprogrammazioni o verifiche di MCP Plus in quanto tali operazioni si possono eseguire senza disconnetterlo "fisicamente" dalla rete.

La porta RS485 di MCP Plus è sdoppiata su 4 morsetti (più uno per lo schermo) in modo da facilitare i collegamenti multidrop: in pratica i morsetti 1 e 4 (segnale "A") sono internamente collegati tra di loro, così come i morsetti 2 e 5 (segnale "B").

ATTENZIONE: come per tutte le reti RS485, ***non sono ammesse derivazioni di tipo radiale***; inoltre la linea ***deve essere terminata sia all'inizio che alla fine con una resistenza da 120 Ohm 1/2W*** tra i morsetti A e B. Il numero massimo di dispositivi sulla rete RS485 deve inoltre essere limitato a 32.

5- DIAGNOSTICA

5.1- Diagnosi del sistema CONTATTO *con MCP*

Il modulo MCP prevede la segnalazione dei guasti di sistema tramite due LED di colore rosso e fornisce su apposita morsettiera un contatto di scambio come descritto nel precedente paragrafo.

I due led rossi forniscono le segnalazioni di:

- **MOD.F** (modulo guasto)
- **BUS.F** (bus non funzionante)

mentre il contatto del relè interno commuta in seguito al verificarsi di uno dei guasti citati oppure per mancanza di alimentazione (sicurezza intrinseca).

La ricerca dei moduli guasti, può essere effettuata in due modi:

1- Utilizzando il tester/programmatore **FXPRO**.
E' il metodo più rapido per ricercare i moduli guasti. Il modulo MCP è provvisto di un connettore a 5 poli per il collegamento del tester. Con il tasto di verifica sul tester è possibile risalire all'indirizzo del modulo che non risponde a MCP.

N.B.: **Durante la diagnosi le uscite vengono spente. Esse tornano al loro stato corretto al termine della diagnosi.**

2- Mediante **Personal Computer**.
È il metodo più potente e versatile. La diagnosi è possibile collegando tramite cavo RS232 il modulo MCP al Personal Computer sul quale è installato il programma di supporto MCPTOOLS fornito in dotazione e già menzionato nei capitoli precedenti. Attraverso la voce di menu Supervisor Show Maps è possibile visualizzare i moduli che rientrano nel programma caricato in MCP, permettendo la ricerca di eventuali moduli guasti (in questo caso la colorazione a video dei moduli guasti è rossa). Per maggiori dettagli, si rimanda alle istruzioni del programma MCPTOOLS.

Nel caso di segnalazione di BUS NON FUNZIONANTE è necessaria la verifica dei collegamenti del bus. Questo guasto si verifica quando il modulo MCP non riesce a trasmettere il suo segnale sul bus (L1 e L2).
Date le molteplici cause di questo tipo di anomalia, si consiglia di contattare **DUEMMEGI**, non prima comunque di aver effettuato un accurato controllo delle connessioni. **Sono disponibili, su richiesta, schede tecniche che guidano l'operatore nella ricerca del guasto.**

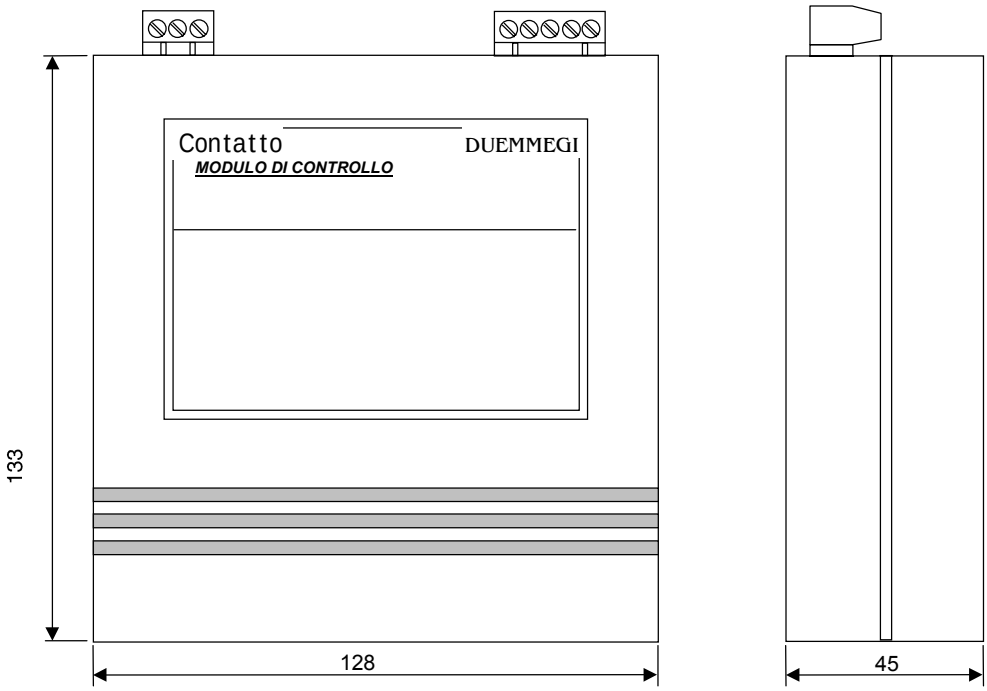
I due led verdi presenti sul modulo MCP segnalano l'attività sul bus. In particolare, il led **POLL** segnala l'inizio del ciclo di polling e risulta essere lampeggiante con frequenza inversamente proporzionale al numero di moduli presenti. Se la memoria di MCP non è programmata correttamente, questo led risulta acceso in modo continuo.
Il led **VAR** segnala invece il verificarsi di una variazione di stato su uno o più moduli di ingresso. Se il led VAR rimane acceso in modo fisso, oppure è frequentemente lampeggiante insieme al led MOD.F nonostante non siano presenti variazioni sui moduli di ingresso, allora significa che esistono due o più moduli dello stesso tipo (IN o OUT) aventi lo stesso indirizzo; in questo caso utilizzare la funzione Supervisor Show Maps del programma **MCPTOOLS**, già descritto nei paragrafi precedenti, per localizzare i moduli doppi (in questo caso la colorazione a video dei moduli doppi è gialla).

6- CARATTERISTICHE TECNICHE

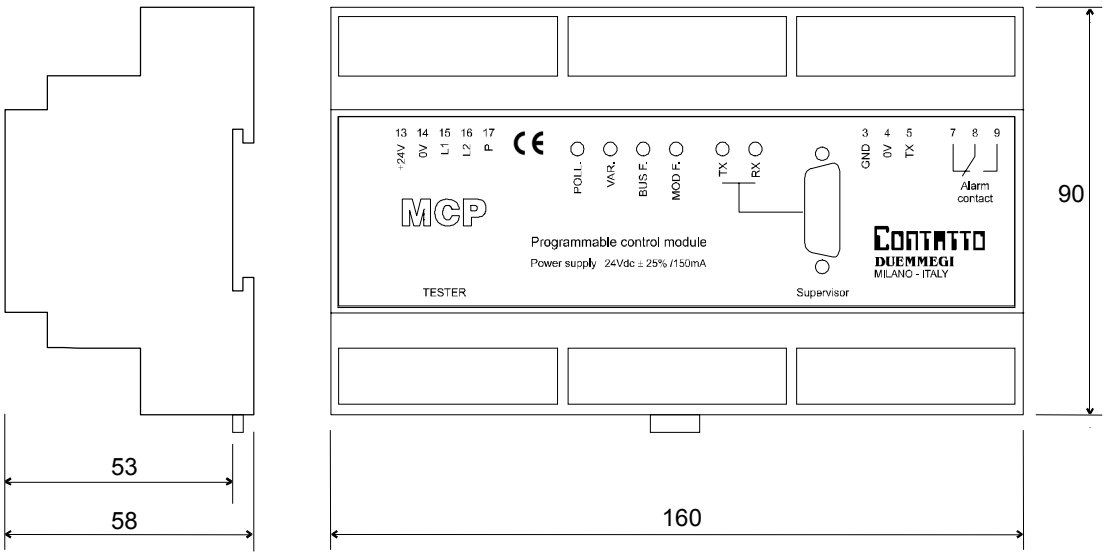
Tensione di alimentazione	24Vcc ± 25%
Assorbimento massimo	150mA
Portata contatto di allarme	5A@ 250Vca AC1
Numero di processori interni	1
Tempo di reazione medio ingresso → uscita	25msec
Memoria di programma utente	Tipo FLASH 128 Kbytes
Memoria RAM	32 Kbytes
Numero di punti virtuali disponibili	1000
Numero di timer	256 con tempi da 0 a 6553 secondi, risoluzione 0.1 sec.
Numero di contatori	512 a 8 bit (con possibilità di configurazione a 16 bit)
Orologio programmatore	Settimanale (sino a 1016 punti indipendenti)
Punti di ingresso digitali	1016
Punti di uscita digitali	1016
Punti di ingresso analogici	Sino a 127 (ogni analogico occupa 8 punti digitali)
Punti di uscita analogici	Sino a 127 (ogni analogico occupa 8 punti digitali)
Porte seriali disponibili	MCP e MCP/MOD: RS232 non optoisolate MCP Plus: RS232 e RS485 optoisolate
Periferiche collegabili	- Modem - Videoterminali touch screen - Display seriale con gestione allarmi (no MCP Plus) - Display binario con gestione allarmi - Sistemi di supervisione su PC
Interfacciabilità verso altri sistemi	MODBUS: - MCP Plus: protocollo MODBUS integrato - MCP e MCP/MOD: con convertitore di protocollo esterno codice **DUEMMEGI CDP** JOHNSON CONTROL: - MCP Plus: protocollo JOHNSON integrato

7- DIMENSIONI DI INGOMBRO

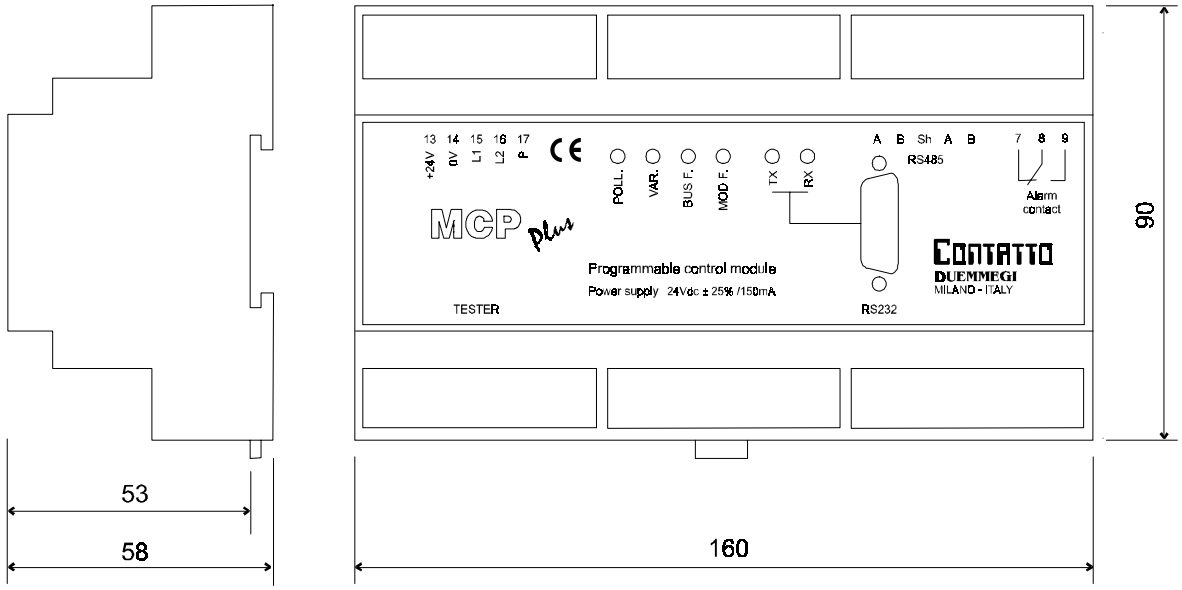
7.1- Dimensioni MCP



7.2- Dimensioni MCP/MOD



7.3- Dimensioni MCP Plus



8- APPENDICE A: PROTOCOLLO DI COMUNICAZIONE FXP

8.1- Formato e definizione dei messaggi del protocollo FXP

Il protocollo proprietario utilizzato da MCP è denominato **FXP**; questo protocollo, appositamente sviluppato per interfacciare MCP con il mondo esterno (PC, PLC, ecc.) è di tipo **NRZ con 1 bit di start, 8 bit dati, nessuna parità, 1 bit di stop**. Il baud rate è configurabile come descritto al paragrafo 4.3 tramite i ponticelli all'interno di MCP alle seguenti velocità: 2400, 4800, 9600, 19200 baud. **MCP si comporta da slave**, per cui risponde alle interrogazioni di un HOST Computer.

Nel seguito i dati numerici rappresentati con la notazione **0x** si intendono in formato esadecimale.
I messaggi tra MCP e HOST hanno il seguente formato:

Indirizzo	Codice	# Byte	Dato 1	.....	Dato N	ChkSum H	ChkSum L

Dove:

- **Indirizzo**: 1 byte, indirizzo nodo MCP, L'indirizzo 00H è valido per ogni nodo
- **Codice**: 1 byte, identificatore del messaggio
- **# Byte**: 1 byte, numero di byte dati che seguono
- **Dato 1 ÷ N** N byte di dati
- **ChkSum**: 2 byte (high,low) di checksum, pari alla somma complementata dei byte del messaggio, inclusi il codice ed il numero di byte.

I messaggi disponibili sono:

Richieste da HOST a MCP

Codice	# Byte	Byte dati	Funzione
0x13	Da 0x03 a 0xFF	I primi tre byte indicano l'Indirizzo di partenza; i rimanenti (#Byte - 3) sono i dati da scrivere in RAM. L'indirizzo è: 0x00 0xXX 0xXX: RAM esterna 0x01 0xXX 0xXX: RAM interna (registri)	Scrittura memoria RAM a partire dall'indirizzo specificato dai primi tre byte.
0x14	0x04	Indirizzo di partenza (3 byte), numero di byte da leggere (1 byte) (0x00 = 256 byte).. 0x00 0xXX 0xXX: RAM esterna 0x01 0xXX 0xXX: RAM interna (registri)	Lettura memoria RAM a partire dall'indirizzo specificato dai primi tre byte. Il numero di byte è specificato dal quarto byte.
0x15	0x01	Indirizzo del modulo MSB=0 ⇒ modulo di ingresso MSB=1 ⇒ modulo di uscita	Richiesta di stato. Il comando è utile in fase di diagnostica.
0x16	0x03	Indirizzo, Nuovo stato, Maschera	Comando delle uscite reali . La maschera indica quali uscite modificare.
0x17	0x02	Indirizzo, Maschera	Uscite reali in manuale. La maschera (bit a 1) indica quali uscite porre in modo manuale (non vengono valutate le corrispondenti equazioni)
0x18	0x02	Indirizzo, Maschera	Uscite reali in automatico. La maschera (bit a 1) indica quali uscite porre in modo automatico (vengono valutate le corrispondenti equazioni). Al momento del comando viene forzato il ricalcolo dell'equazione.

0x19	0x02	Bit 15: Stato Bit 9÷0: Numero uscita virtuale (0-1000)	Comando uscite virtuali. Il comando forza il calcolo delle equazioni relative alle uscite reali dipendenti dall'uscita virtuale.
0x1A	0x02	'I', 'D'	Richiesta di codice di identificazione.
0x1B	Da 0x00 a 0xFF	Caratteri ASCII	Copia su linea seriale DISPLAY/PRINTER.
0x1C	0x06	Ore, Minuti, Giorno della settimana, Data, Mese, Anno (BCD)	Programmazione orologio.
0x1D	0x00		Lettura orologio.

MCP è in grado di riconoscere inoltre alcuni messaggi tipici dei **MODEM**; questi messaggi sono i seguenti:

- *OK,*
- *CONNECT XXXX*
- *BUSY*
- *RING*
- *NO CARRIER, NO DIALTONE, NO ANSWER*

preceduti e seguiti da 0x0D-0x0A (CR LF)

Risposte da MCP a HOST

Codice	# Byte	Byte dati	Funzione
0x13	0x01	0x00: scrittuta valida 0xFF: errore di scrittura.	Risposta al comando di scrittura memoria RAM.
0x14	0xXX	XX byte letti dalla RAM a partire dall'indirizzo specificato nella richiesta. Il numero XX è anch'esso specificato nella richiesta. (XX = 0x00 = 256 byte)	Risposta al comando di lettura memoria RAM.
0x15	0x03	Flag, Indirizzo del modulo, Stato. Flag. 0 = 1: modulo non previsto Flag.1 = 1: modulo non risponde	Risposta alla richiesta di stato.
0x16	0x00		Risposta al comando delle uscite reali
0x17	0x00		Risposta al comando uscite reali in manuale.
0x18	0x00		Risposta al comando uscite reali in automatico.
0x19	0x00		Risposta al comando delle uscite virtuali.
0x1A	0x40	64 caratteri di identificazione	Risposta alla richiesta di codice di identificazione.
0x1B	0x00		Risposta al comando di copia su seriale.
0x1C	0x00		Risposta al comando di programmazione dell'orologio.
0x1D	0x06	Ore, Minuti, Giorno della settimana, Data, Mese, Anno (BCD)	Risposta alla richiesta di lettura orologio.

8.2- Mappatura RAM

La seguente tabella riporta la mappatura nella RAM di MCP dei parametri di uso più comune.

8.2.1- Mappa memoria RAM esterna

Indirizzo (Hex)	Contenuto	Commenti
0001-007F	Mappa degli stati dei moduli di ingresso	127 moduli di ingresso
0081-00FF	Mappa degli stati dei moduli di uscita	127 moduli di uscita
0101-017D	Mappa degli stati correnti dei punti virtuali	1000 punti virtuali (solamente digitali)
0181-01FF	Modalità controllo uscite reali	127 moduli di uscita se bit = 0 (automatico)→ l'uscita viene controllata in funzione dell'equazione se bit = 1 (manuale)→ l'uscita viene controllata da RS232
0201-027D	Mappa dei nuovi stati dei punti virtuali	1000 punti virtuali (solamente digitali)
0289	Anno in formato BCD	Letto dal chip orologio interno. **Nota 4**
028A	Giorno della settimana BCD	Letto dal chip orologio interno; 0=Domenica….6=Sabato. **Nota 4**
028B	Mese in formato BCD	Letto dal chip orologio interno. **Nota 4**
028C	Giorno del mese in formato BCD	Letto dal chip orologio interno. **Nota 4**
028D	Ore in formato BCD	Letto dal chip orologio interno. **Nota 4**
028E	Minuti in formato BCD	Letto dal chip orologio interno. **Nota 4**
028F	Secondi in formato BCD	Letto dal chip orologio interno. **Nota 4**
0300-04FF	Allarmi per modem	512 byte con gli allarmi che hanno causato la chiamata via modem. **Nota 3**
0500-06FF	Registri contatori	512 contatori a 8 bit / 256 contatori a 16 bit. **Nota 5**
0700-077F	Coda di refresh per device seriale	Se bit = 1 il messaggio corrispondente all'uscita virtuale deve essere ciclato. **Nota 1**
0780-07BF	Contatori variati	32 byte = 512 bit. Se bit = 1 il contenuto del contatore è variato
1000-13FF	Timer	1024 byte per 256 timer. **Nota 2**
1408-17FF	Mappa moduli contatori 16 bit	1016 byte per 127 moduli a 4 canali a 16 bit. **Nota 5**
1808-1BFF	Mappa moduli analogici di ingresso 16 bit	1016 byte per 127 moduli a 4 canali a 16 bit. **Nota 5**
2000-2FFF	Lista eventi in ordine cronologico	4096 byte per memorizzare 1023 eventi con data e ora. **Nota 3**
3000-3080	Azzeramento moduli contatori ext.	

Nota 1: Quando un allarme, che richiede un messaggio al display seriale, diventa attivo, viene inserito nella coda di refresh (bit corrispondente a 1). Esso viene rimosso o quando diventa disattivo, se programmato come NOMEM, oppure quando viene eseguito il RESET dall'operatore, se programmato come MEM. Ad intervalli regolari di 2 secondi, MCP invia al display seriale il messaggio da visualizzare in accordo con i dati nella coda. Il refresh non viene eseguito in ordine cronologico.
Nota 2: Ogni timer richiede 4 byte in RAM: 2 byte per il timer, 11 bit per l'uscita da controllare, 5 flag.
Nota 3: All'occorrenza di un allarme programmato per generare la chiamata via modem o per la memorizzazione dell'evento, si inseriscono nella lista 4 byte indicanti il punto virtuale che ha causato l'evento, con data e ora. In tal modo è possibile avere l'ordine cronologico di 127 allarmi modem o 1023 eventi. La lista deve essere azzerata dal supervisore.
Nota 4: Valido solo per MCP Plus. Le celle da 0289 a 028F contengono l'immagine dello stato corrente del chip orologio interno di MCP; queste celle possono anche essere scritte, nel qual caso l'orologio interno viene aggiornato con i nuovi parametri. Questa possibilità permette la lettura e la scrittura dell'orologio utilizzando le funzioni di lettura e scrittura della memoria RAM invece che i comandi specifici; in tal modo risulta possibile leggere ed impostare l'orologio anche mediante protocollo MODBUS. Tutte le volte che si cambia uno di questi parametri, i secondi vengono comunque posti uguali a zero.
Nota 5: Nel caso di registro 16 bit, la cella di indirizzo più basso contiene il byte più significativo e la cella di indirizzo più alto contiene il byte meno significativo del registro stesso.

8.2.2- Mappa memoria RAM interna

Indirizzo (Hex)	Contenuto	Commenti
013F	Indirizzo più alto	Indirizzo più alto tra i moduli che rientrano nel programma caricato in MCP (per FXPRO)
0140-014F	Configurazione moduli di ingresso	128 flag indicanti i moduli di ingresso previsti nel programma caricato
0150-015F	Configurazione moduli di uscita	128 flag indicanti i moduli di uscita previsti nel programma caricato
0160-016F	Moduli di ingresso guasti	128 flag indicanti i moduli di ingresso guasti
0170-017F	Moduli di uscita guasti	128 flag indicanti i moduli di uscita guasti
0180-018F	Moduli di ingresso doppi	128 flag indicanti i moduli di ingresso doppi (con stesso indirizzo)
0190-019F	Moduli di uscita doppi	128 flag indicanti i moduli di uscita doppi (con stesso indirizzo)
01C0-01CF	Configurazione moduli contatori 16 bit	128 flag indicanti i moduli contatori 16 bit previsti nel programma caricato
01D0-01DF	Configurazione moduli analogici 16 bit	128 flag indicanti i moduli analogici 16 bit di ingresso presenti

8.3- Guida per l'implementazione di un driver di comunicazione con MCP mediante protocollo FXP

Nota preliminare: in questo paragrafo la notazione x%8 indica x modulo 8 ed equivale al resto della divisione di x per 8; la notazione INT(x) indica invece la parte intera di x.

Per calcolare x modulo 8 procedere come segue:

1. dividere x per 8
2. sottrarre al risultato del punto 1 la parte intera dello stesso
3. moltiplicare per 8 il risultato del punto 2: il valore risultante è il modulo 8 del numero di partenza; questo risultato è sempre un numero intero compreso tra 0 e 7.

Esempio di calcolo di 43%8:

1. 43 : 8 = 5.375
2. 5.375 – 5 = 0.375
3. 0.375 x 8 = 3

Richiesta stati ingressi-uscite

La richiesta dello stato degli ingressi e delle uscite può essere fatto in due modi:

1. Utilizzando il messaggio con codice 0x015. In questo modo, oltre allo stato MCP risponde con un byte di flag con le informazioni circa la configurazione e la funzionalità del modulo.
2. Utilizzando il messaggio di lettura RAM 0x14: MCP risponde con il solo stato del modulo. Questo comando può essere utilizzato per scaricare in un blocco lo stato di tutti i moduli. La mappa degli stati dei moduli di ingresso parte dall'indirizzo 1 fino a 127 (0x01 ÷ 0x7F), quella delle uscite da 129 a 255 (0x81 ÷ 0xFF). All'indirizzo RAM **x** è memorizzato lo stato del modulo di ingresso avente indirizzo **x**, all'indirizzo RAM **x+128** è memorizzato lo stato del modulo di uscita **x**.

Richiesta stato punti virtuali

La richiesta dello stato delle uscite (punti) virtuali può essere fatto utilizzando il messaggio di lettura RAM 0x14. La mappa degli stati dei punti virtuali parte dall'indirizzo 257 fino a 381 (0x101 ÷ 0x17D) **compresi**. Il punto virtuale **x** è il bit **(x - 1)%8** della locazione RAM:

$$257 + INT\left(\frac{x-1}{8}\right)$$

Notare che gli indirizzi RAM per la lettura dei punti virtuali sono diversi da quelli per la scrittura.
L'appendice C riporta una tabella per localizzare facilmente indirizzo RAM e bit relativi ad un dato punto virtuale.

Comando uscite reali

Le uscite reali possono essere comandate in due modi:

1. Utilizzando l'apposito comando con codice 0x16: in tal caso MCP invia su bus istantaneamente il nuovo stato dell'uscita
2. Utilizzando il comando di scrittura in RAM all'indirizzo relativo allo stato del modulo di uscita (da 129 a 255, vale a dire 0x81 ÷ 0xFF). MCP invierà il comando all'uscita con il normale ciclo di polling (il ritardo massimo dell'aggiornamento dell'uscita è pari alla durata del ciclo di polling). Il modulo di uscita **x** si trova all'indirizzo RAM **x+128**.

Controllo uscite reali

Le uscite reali possono assumere lo stato derivante dal calcolo dell'equazione relativa o meno. Ciò può essere controllato con i comandi 0x17 e 0x18 oppure scrivendo direttamente in RAM i bit di controllo per ciascuna uscita. Tali bit si trovano a partire dall'indirizzo 385 fino all'indirizzo 511 (0x181 ÷ 0x1FF). Se il bit è 0 l'uscita è il risultato del calcolo dell'equazione, se il bit è 1 il calcolo dell'equazione è sospeso per quell'uscita. Il modulo di uscita **x** è controllato dalla cella RAM che si trova all'indirizzo **x+384**.

Comando uscite virtuali

Le uscite virtuali possono essere comandate in due modi:

1. Utilizzando l'apposito comando con codice 0x19
2. Utilizzando il comando di scrittura in RAM all'indirizzo relativo al punto virtuale desiderato (da 513 a 637 **compresi**, vale a dire 0x201 ÷ 0x27D). Il punto virtuale **x** è il bit **(x - 1)%8** della locazione RAM:

$$513 + INT\left(\frac{x-1}{8}\right)$$

Notare che gli indirizzi RAM per la lettura dei punti virtuali sono diversi da quelli per la scrittura.
L'appendice C riporta una tabella per localizzare facilmente indirizzo RAM e bit relativi ad un dato punto virtuale.

Soft-reset di MCP

Scrivendo il valore 0 in RAM agli indirizzi 640-647 (0x280 ÷ 0x287), si provoca l'inizializzazione del modulo MCP (reset).

Registri Contatori Interni

I contatori sono mappati in RAM a partire dall'indirizzo 1280 (0x500) con offset ricavabile dal numero del contatore considerato (vedere Capitolo 2). La lettura e la scrittura dei contatori deve essere fatta con i comandi di lettura e scrittura della RAM agli indirizzi specificati. Il contatore Cn si trova alla locazione RAM:
(n + 1280).
Nel caso di registro contatore a 16 bit, la cella indirizzata contiene il byte più significativo mentre la cella successiva contiene il byte meno significativo del contatore stesso.

Moduli Contatori Esterni 4 canali a 16 bit

I moduli contatori esterni 4 canali a 16 bit sono mappati in RAM a partire dall'indirizzo 5128 (0x1408). La lettura e la scrittura deve essere fatta con i comandi di lettura e scrittura della RAM agli indirizzi specificati. Il canale Ch (compreso tra 1 e 4) del contatore esterno con indirizzo Addr si trova alla locazione RAM:
(Addr - 1) x 8 + 2 x (Ch - 1) + 5128.
La cella indirizzata contiene il byte più significativo mentre la cella successiva contiene il byte meno significativo del contatore stesso.

Moduli Analogici di ingresso 4 canali a 16 bit o 12 bit

I moduli analogici di ingresso 4 canali a 16 o 12 bit sono mappati in RAM a partire dall'indirizzo 6152 (0x1808). La lettura deve essere fatta con il comando di lettura della RAM agli indirizzi specificati. Il canale Ch (compreso tra 1 e 4) del modulo con indirizzo Addr si trova alla locazione RAM:
(Addr - 1) x 8 + 2 x (Ch - 1) + 6152.
La cella indirizzata contiene il byte più significativo mentre la cella successiva contiene il byte meno significativo del registro stesso.

Eventi

Gli eventi sono memorizzati in RAM a partire dall'indirizzo 8192 (0x2000). Il numero di eventi memorizzati si trova in due byte agli indirizzi 8192 (più significativo) e 8193 (meno significativo) (0x2000 e 0x2001). Le informazioni sono divise in un massimo di 1023 blocchi di quattro byte ciascuno dei quali contiene:

-	Indirizzo n punto virtuale (7 bit)		
Mese (4 bit)		Stato	Punto p (3 bit)
Giorno (5 bit)			Ora MSB (3 bit)
Ora LSB (2)	Minuti (6 bit)		

Il numero del punto virtuale associato all'evento è dato da **(n-1)*8 + (p+1).** *Stato* indica lo stato del punto virtuale, *Mese, giorno ora* e *minuti* indicano il momento dell'evento.

La coda è cronologica e può contenere fino a 1023 blocchi. Ulteriori eventi vengono persi. Il supervisore deve azzerare il numero di eventi nella coda che si trova, come detto sopra, agli indirizzi 8192 e 8193 (0x2000 ÷ 0x2001) per poter riabilitare MCP alla registrazione di ulteriori eventi.

Allarmi MODEM

Gli allarmi pendenti che hanno causato la chiamata via modem si trovano in RAM a partire dall'indirizzo 768 (0x300). All'indirizzo 768 (0x300) si trova un byte con il numero di eventi memorizzati. Le informazioni sono divise in un massimo di 127 blocchi di quattro byte ciascuno dei quali contiene:

-	Indirizzo n punto virtuale (7 bit)		
Mese (4 bit)		Stato	Punto p (3 bit)
Giorno (5 bit)			Ora MSB (3 bit)
Ora LSB (2)	Minuti (6 bit)		

Il numero del punto virtuale associato all'allarme è dato da **(n-1)*8 + (p+1).** *Stato* indica lo stato del punto virtuale, *Mese, giorno ora* e *minuti* indicano il momento dell'evento.

La coda è cronologica e può contenere fino a 127 blocchi. Ulteriori allarmi vengono persi. Il supervisore deve azzerare il numero degli allarmi nella coda che si trova all'indirizzo 768 (0x300) per poter riabilitare MCP alla registrazione di ulteriori eventi.
La conferma dell'acquisizione della chiamata deve essere fatta con la richiesta del codice di identificazione. In caso contrario MCP continuerebbe il processo di chiamata con il numero successivo.

Configurazione del sistema

La configurazione dei moduli di ingresso è memorizzata nella RAM ***interna*** del microcontrollore all'indirizzo 320 (0x140) in 16 byte consecutivi. La configurazione del modulo **x** sarà il bit **x%8** della locazione:

$$320 + \mathrm{INT}\left(\frac{x}{8}\right)$$

Se il bit è 1 allora il modulo rientra nel programma di polling, altrimenti no. In modo analogo la configurazione dei moduli di uscita si trova nella RAM interna a partire dall'indirizzo 336 (0x150).

Moduli guasti

I moduli guasti sono memorizzati nella RAM ***interna*** del microcontrollore in modo analogo a quello descritto per la configurazione del sistema, a partire dall'indirizzo 352 (0x160) per i moduli di ingresso e 368 (0x170) per i moduli di uscita.

Moduli doppi

I moduli doppi sono memorizzati nella RAM ***interna*** del microcontrollore in modo analogo a quello descritto per la configurazione del sistema, a partire dall'indirizzo 384 (0x0180) per i moduli di ingresso e 400 (0x190) per i moduli di uscita.

9- APPENDICE B: PROTOCOLLO DI COMUNICAZIONE MODBUS

9.1- Caratteristiche generali

MCP Plus è in grado di scambiare informazioni con il mondo esterno attraverso i protocolli MODBUS (RTU Protocol) e JOHNSON CONTROL (VND). Questi protocolli sono direttamente integrati in MCP Plus e convivono, se abilitati mediante la direttiva PROTOCOL (vedi relativo paragrafo), con il protocollo proprietario descritto in precedenza; ciò significa che:

- ad una richiesta in protocollo MODBUS, se abilitato, MCP Plus risponde secondo il protocollo MODBUS
- ad una richiesta in protocollo JOHNSON, se abilitato, MCP Plus risponde secondo il protocollo JOHNSON
- ad una richiesta in protocollo proprietario, MCP Plus risponde secondo il protocollo proprietario

In questa appendice verrà riportata una traccia per l'utilizzo del protocollo MODBUS; per quanto riguarda il protocollo JOHNSON, fare riferimento alla apposita documentazione.
I parametri di comunicazione MODBUS utilizzati da MCP Plus sono i seguenti:

- **1 bit di start**
- **8 bit dati**
- **nessuna parità**
- **1 o 2 bit di stop (a rilevazione automatica)**

Il baud rate è configurabile come descritto al paragrafo 4.3 tramite i ponticelli all'interno di MCP Plus alle seguenti velocità: 2400, 4800, 9600, 19200 baud. **MCP Plus si comporta da slave (periferica MODBUS)**, per cui risponde alle interrogazioni di un MASTER MODBUS DEVICE.
In una rete MODBUS ogni periferica deve avere un proprio indirizzo (station address); l'indirizzo di MCP Plus si imposta includendo nel programma la funzione ADDRESS descritta al paragrafo 2.3.12.
Per localizzare i punti di ingresso e di uscita (reali e/o virtuali), i registri ecc., fare riferimento alla mappa di memoria RAM esterna riportata al paragrafo 8.2.1.

9.2- Funzioni MODBUS supportate

MCP Plus supporta le seguenti funzioni MODBUS:

Codice funzione	Descrizione
1	Lettura stato uscite
2	Lettura stato ingressi
3	Lettura registri (memoria)
4	Lettura ingressi analogici
5	Comando di un singolo punto di uscita digitale
6	Scrittura di un singolo registro
15	Comando di uscite multiple
16	Scrittura di registri multipli
17	Richiesta tipo dispositivo

9.3- Esempi di funzioni MODBUS

Il presente paragrafo illustra alcuni esempi di funzioni MODBUS (richiesta e risposta) tra le più utilizzate; si ricorda che MCP Plus, in un sistema MODBUS, è una periferica SLAVE, vale a dire che risponde alle interrogazioni di un dispositivo MASTER.

Gli esempi che seguono servono per individuare le funzioni MODBUS da utilizzare per il colloquio con MCP Plus; gli attuali driver MODBUS implementati nei dispositivi più comuni (PLC, software di supervisione per PC, videoterminali, ecc.), mettono a disposizione una piattaforma di sviluppo e una interfaccia utente che ne semplifica notevolmente la messa a punto rispetto a quanto descritto nei prossimi paragrafi. In pratica la messa a punto della comunicazione tra sistema MASTER e MCP Plus si riduce alla configurazione del driver di comunicazione messo a disposizione dal produttore del sistema MASTER, per cui occorre fare riferimento anche al manuale utente dello stesso.
Le notazioni che seguono, se non diversamente specificato, si intendono in formato decimale.

9.3.1- Funzione 1: Lettura dello stato delle uscite

La funzione MODBUS 1 permette di leggere lo stato delle uscite; è necessario specificare:

- **un punto di partenza (Start)**; questo valore **deve essere multiplo di 8**. Detto **i** l'indirizzo del modulo reale del sistema Contatto a partire dal quale si vuole leggere lo stato delle uscite, il valore di Start sarà pari a: **(i - 1) x 8**. Valori consentiti: **da 0 a 1008**; notare che questo numero non è l'indirizzo fisico del modulo di uscita.
- **quanti punti di uscita si vogliono leggere (Number)**; in pratica quanti moduli di indirizzo consecutivo si vogliono leggere. Per evitare confusioni, è bene che questo valore sia **multiplo di 8** e pari al numero di moduli che si vogliono leggere moltiplicato 8. Valori consentiti: da **8 a 1016**.

MCP Plus risponderà con un numero di byte pari al Number diviso 8.

Esempio:
Si vuole leggere lo stato delle uscite del modulo 25, ad esempio un MOD8R che, come noto, possiede 8 punti di uscita. I parametri da passare al driver MODBUS sono:

Start: 192
Number: 8

MCP Plus risponderà con 1 byte contenente lo stato dei punti di uscita del modulo 25, codificato secondo il codice binario (1=uscita accesa, 0=uscita spenta). Il bit meno significativo corrisponde al punto di uscita 1, quello più significativo al punto di uscita 8.

9.3.2- Funzione 2: Lettura dello stato degli ingressi

La funzione MODBUS 2 permette di leggere lo stato degli ingressi; è necessario specificare:

- **un punto di partenza (Start)**; questo valore **deve essere multiplo di 8**. Detto **i** l'indirizzo del modulo reale del sistema Contatto a partire dal quale si vuole leggere lo stato degli ingressi, il valore di Start sarà pari a: **(i - 1) x 8**. Valori consentiti: **da 0 a 1008**; notare che questo numero non è l'indirizzo fisico del modulo di ingresso.
- **quanti punti di ingresso si vogliono leggere (Number)**; in pratica quanti moduli di indirizzo consecutivo si vogliono leggere. Per evitare confusioni, è bene che questo valore sia **multiplo di 8** e pari al numero di moduli che si vogliono leggere moltiplicato 8. Valori consentiti: da **8 a 1016**.

MCP Plus risponderà con un numero di byte pari al Number diviso 8.

Esempio 1:
Si vuole leggere lo stato degli ingressi del modulo 43, ad esempio un MOD8I/A che, come noto, possiede 8 punti di ingresso. I parametri da passare al driver MODBUS sono:

Start: **336**
Number: **8**

MCP Plus risponderà con 1 byte contenente lo stato dei punti di ingresso del modulo 43, codificato secondo il codice binario (1=ingresso attivo, 0=ingresso non attivo). Il bit meno significativo corrisponde al punto di ingresso 1, quello più significativo al punto di ingresso 8.

Esempio 2:
Si vuole leggere lo stato degli ingressi dei moduli 57, 58, 59 e 60, ad esempio tutti MOD8I/A che, come noto, possiedono 8 punti di ingresso ciascuno. I parametri da passare al driver MODBUS sono:

Start: **448**
Number: **32**

MCP Plus risponderà con 4 byte contenenti ciascuno lo stato dei punti di ingresso dei moduli dal 57 (primo byte) al 60 (ultimo byte) compresi.

9.3.3- Funzione 3: Lettura dei registri (memoria)

La funzione MODBUS 3 **è la più utilizzata**, in quanto di uso generale, e permette di leggere il contenuto della memoria RAM di MCP Plus che contiene praticamente tutte le informazioni sullo stato del sistema.
È necessario specificare:

- **un punto di partenza (Start)**; questo valore **è l'indirizzo della cella RAM** a partire dalla quale si vuole leggere. Valori consentiti: **da 0 a 12416** (in esadecimale da 0x0000 a 0x3080) per la **memoria esterna** e **da 33087 a 33247** (in esadecimale da 0x813F a 0x81DF) per la **memoria interna** del microcontrollore (vedi paragrafo 8.2)
- **quanti registri si vogliono leggere (Number)**; in pratica quante celle RAM consecutive si vogliono leggere. Valori consentiti: da **1 a 125**.

MCP Plus risponde con un numero di WORD pari al Number specificato (vale a dire **un numero di byte pari al doppio del Number specificato**).

Ogni WORD nella risposta di MCP sarà formata da:

- **il byte più significativo pari a zero e quello meno significativo contenente il dato della cella indirizzata per le locazioni RAM da 0x0000 a 0x04FF comprese e per le locazioni della RAM interna del microcontrollore**
- **il byte più significativo contenente il dato della cella indirizzata e quello meno significativo contenente il dato della cella successiva per le locazioni RAM da 0x0500 a 0x307F comprese**

La funzione MODBUS 3 può essere utilizzata per leggere lo stato di ingressi e uscite reali, lo stato dei punti virtuali, il contenuto dei contatori, ecc.; in pratica si può richiedere qualsiasi informazione presente nella mappa RAM (vedi paragrafo 8.2), compresa ora e data corrente del chip orologio interno di MCP.

Esempio 1:
Si vuole leggere lo **stato delle uscite** del modulo 25, ad esempio un MOD8R; in alternativa alla funzione 1, è possibile utilizzare la funzione 3. La posizione della cella RAM contenente lo stato del modulo di uscita **i** è pari a **i+128**, per cui, per quanto riguarda il modulo 25, sarà necessario passare al driver MODBUS i seguenti parametri:

Start: **153**
Number: **1**

MCP Plus risponderà con una WORD il cui byte più significativo è uguale a zero e quello meno significativo contiene lo stato dei punti di uscita del modulo 25, codificato secondo il codice binario (1=uscita accesa, 0=uscita spenta). Il bit meno significativo corrisponde al punto di uscita 1, quello più significativo al punto di uscita 8.

Esempio 2:
Si vuole leggere lo **stato degli ingressi** del modulo 43, ad esempio un MOD8I/A; in alternativa alla funzione 2, è possibile utilizzare la funzione 3. La posizione della cella RAM contenente lo stato del modulo di ingresso **i** è pari a **i**, per cui, per quanto riguarda il modulo 43, sarà necessario passare al driver MODBUS i seguenti parametri:

Start: **43**
Number: **1**

MCP Plus risponderà con una WORD il cui byte più significativo è uguale a zero e quello meno significativo contiene lo stato dei punti di ingresso del modulo 43, codificato secondo il codice binario (1=ingresso attivo, 0=ingresso non attivo). Il bit meno significativo corrisponde al punto di ingresso 1, quello più significativo al punto di ingresso 8.

Esempio 3:
Si vuole leggere lo **stato degli ingressi** dei moduli 57, 58, 59 e 60, ad esempio tutti MOD8I/A, mediante la funzione 3. I parametri da passare al driver MODBUS sono:

Start: **57**
Number: **4**

MCP Plus risponderà con 4 WORD (8 byte), ognuna della quali ha il byte più significativo uguale a zero, mentre quello meno significativo contiene lo stato dei punti di ingresso dei moduli 57, 58, 59 e 60 codificato secondo il codice binario (1=ingresso attivo, 0=ingresso non attivo). Il bit meno significativo corrisponde al punto di ingresso 1, quello più significativo al punto di ingresso 8.

Esempio 4:
Si vuole leggere lo **stato del punto virtuale** V328 mediante la funzione 3. Il punto virtuale **Vx** si trova, all'interno della mappa RAM, alla locazione:

$$257 + \mathrm{INT}\left(\frac{x-1}{8}\right)$$

Poiché un punto virtuale occupa un solo bit, si dovrà specificare quale è quello ad esso associato; questo è dato da:

$$(x-1)\%8$$

(per il significato dei simboli utilizzati in queste formule, fare riferimento al paragrafo 8.3).
In alternativa, l'appendice C riporta una tabella per localizzare facilmente indirizzo RAM e bit relativi ad un dato punto virtuale.

Notare che gli indirizzi RAM per la lettura dei punti virtuali sono diversi da quelli per la scrittura.

I parametri da passare al driver MODBUS, per quanto riguarda il punto virtuale V328, sono quindi:

Start: **297**
Number: **1**
Bit: **7**

Nota: il numero del bit è sempre un numero compreso tra 0 e 7.

MCP Plus risponderà con 1 WORD (2 byte), il cui byte più significativo è uguale a zero, mentre quello meno significativo contiene lo stato dei punti virtuali da V321 (bit meno significativo) a V328 (bit più significativo). I punti virtuali sono codificati in binario (1=punto attivo, 0=punto non attivo).

Esempio 5:
Si vuole leggere l'**indirizzo di eventuali moduli di ingresso guasti** mediante la funzione 3. L'elenco dei moduli di ingresso guasti si trova nella memoria interna del microcontrollore come riportato nella tabella del paragrafo 8.2.2; per poter accedere alla memoria interna, all'indirizzo riportato nella mappa si deve sommare 32768 (in esadecimale 0x8000). Per leggere l'intera mappa dei moduli di ingresso guasti, bisogna passare al driver MODBUS i seguenti parametri:

Start: **33120**
Number: **16**

MCP Plus risponderà con 16 WORD (32 byte), ognuna delle quali ha il byte più significativo uguale a zero, mentre quello meno significativo contiene la condizione dei moduli di ingresso in accordo alla seguenti regole:

- il bit meno significativo della prima WORD (bit 0) è la condizione del modulo di ingresso di indirizzo 0 (che non esiste e che quindi va scartato)
- il bit 1 della prima WORD è la condizione del modulo di ingresso di indirizzo 1
- .
- .
- il bit 7 della prima WORD è la condizione del modulo di ingresso di indirizzo 7
- il bit 0 della seconda WORD è la condizione del modulo di ingresso di indirizzo 8
- .
- .
- il bit 7 della sedicesima WORD è la condizione del modulo di ingresso di indirizzo 127

Quando un bit vale "1", significa che il modulo ad esso associato è guasto (oppure non presente).

Questo esempio si può applicare, semplicemente variando l'indirizzo in base alla mappa riportata al paragrafo 8.2.2, a tutte le altre informazioni relative alla condizione o alla configurazione dei moduli (sia di ingresso che di uscita).

Esempio 6:
Si vuole leggere il **valore analogico del canale 1** del modulo analogico a 12 bit di indirizzo 8, ad esempio un MOD4AM12; La posizione delle celle RAM contenenti il valore del canale **Ch** del modulo di ingresso analogico di indirizzo **Addr** è pari a **(Addr - 1) x 8 + 2 x (Ch - 1) + 6152** per cui, per quanto riguarda il canale 1 del modulo 8, sarà necessario passare al driver MODBUS i seguenti parametri:

Start: **6208**
Number: **1**

MCP Plus risponderà con una WORD (16 bit) contenente il valore richiesto.

9.3.4- Funzione 5: Comando di un singolo punto di uscita digitale

La funzione 5 consente di forzare lo stato di un singolo punto di uscita reale; è necessario specificare:

- **il numero del punto di uscita reale da forzare (Number)**; detto **i** l'indirizzo del modulo reale del sistema Contatto di cui si vuole variare un punto di uscita e detto **p** il punto di uscita che si vuole variare, allora Number dovrà essere impostato a **[(i –1) x 8 + p – 1]**. I valori ammessi per **i** sono da 1 a 127, mentre per **p** sono da 1 a 8 .
- **nuovo stato del punto di uscita** (1=acceso, 0=spento).

Esempio:
Si vuole accendere il punto 3 del modulo di uscita di indirizzo 29. I parametri da passare al driver MODBUS sono:

Number: **226**
Status: **1**

9.3.5- Funzione 16: Scrittura di registri multipli (memoria)

La funzione 16 consente di scrivere nella memoria RAM esterna di MCP Plus che contiene praticamente tutte le informazioni sullo stato del sistema. Questa funzione, assieme alla 3, **è la più utilizzata**.
È necessario specificare:

- **un punto di partenza (Start)**; questo valore **è l'indirizzo della cella RAM** a partire dalla quale si vuole scrivere il nuovo valore. Valori consentiti per Number: **da 0 a 12416** (in esadecimale da 0x0000 a 0x3080). La **memoria interna** del microcontrollore NON DEVE ESSERE MODIFICATA
- **quanti registri si vogliono scrivere (Number)**; in pratica quante celle RAM consecutive si vogliono scrivere. Valori consentiti: da **1 a 125**.
- **i valori che si vogliono scrivere (Data)** nelle celle specificate; ogni dato (tanti quanti specificati da Number) deve essere formato da due byte (una WORD), dove il byte più significativo è pari a zero e quello meno significativo contiene il valore voluto

I driver MODBUS mettono normalmente a disposizione la possibilità di scrivere una o più WORD per intero (utile nel caso si voglia cambiare ad esempio il contenuto di un contatore o variare una uscita analogica), oppure di variare un singolo bit (ad esempio per comandare una singola uscita reale o per variare lo stato di un punto virtuale).

La funzione MODBUS 16 può quindi essere utilizzata per variare lo stato di un intero modulo di uscita (sia digitale che analogico), lo stato di un solo punto di uscita di un modulo, lo stato dei punti virtuali, il contenuto dei contatori, ecc.

Nel caso si voglia variare un solo bit di un registro mediante la funzione 16, bisogna tenere conto dello stato degli altri bit dello stesso registro in quanto la scrittura avviene sull'intera WORD; in pratica i driver MODBUS ne tengono automaticamente conto, in quanto, quando la scrittura deve essere a livello di bit, eseguono in successione le seguenti due funzioni:

1. lettura, mediante la funzione 3, della WORD che comprende il bit che si vuole variare
2. scrittura, mediante la funzione 16, della WORD letta ma con il bit interessato variato

MCP tratterà automaticamente la WORD ricevuta dal MASTER MODBUS:

- **forzando il byte più significativo a zero per le locazioni RAM da 0x0000 a 0x04FF comprese e per le locazioni della RAM interna del microcontrollore**
- **lasciando la WORD inalterata (vale a dire come spedita dal MASTER MODBUS) per le locazioni RAM da 0x0500 a 0x307F comprese; in questo caso la parte più significativa della WORD verrà**

scritta alla locazione indirizzata mentre la parte meno significativa verrà scritta alla locazione successiva

La funzione MODBUS 16 può essere utilizzata anche per impostare data e ora del chip orologio interno di MCP Plus come illustrato in uno dei prossimi esempi.

Esempio 1:
Si vuole accendere il punto 3 del modulo di uscita di indirizzo 29. In alternativa alla funzione 5, è possibile utilizzare la funzione 16. La posizione della cella RAM contenente lo stato del modulo di uscita **i** è pari a **i+128**, per cui, per quanto riguarda il modulo 29, sarà necessario passare al driver MODBUS i seguenti parametri:

Start: **157**
Number: **1** (normalmente, in questo caso, questo parametro non viene richiesto dal driver)
Bit: **2**
Valore: **1** (oppure ON, dipende dal driver utilizzato)

Nota: il punto 3 di un modulo di uscita corrisponde al bit 2 della WORD, in quanto i punti di uscita reali del sistema Contatto sono numerati da 1 a 8, mentre il driver MODBUS "ragiona" su bit da 0 a 7.

L'esecuzione di questa funzione prevede, come descritto prima, che il driver MODBUS legga la cella di memoria 157 mediante la funzione 3, cambi il bit 2 al valore letto, e poi trasferisca il nuovo valore nella cella 157 mediante la funzione 16. Questa procedura, normalmente, viene eseguita automaticamente dal driver MODBUS del sistema MASTER (PLC, software di supervisione per PC, videoterminale, ecc.).

Esempio 2:
Si vogliono accendere tutte le uscite del modulo di indirizzo 29. Si utilizza la funzione 16. La posizione della cella RAM contenente lo stato del modulo di uscita **i** è pari a **i+128**, per cui, per quanto riguarda il modulo 29, sarà necessario passare al driver MODBUS i seguenti parametri:

Start: **157**
Number: **1**
Valore: **255**

In questo caso viene scritto direttamente il valore 255 nella cella di memoria 157. I driver MODBUS consentono inoltre di eseguire operazione sia matematiche che logiche tra il valore corrente della cella ed un valore prefissato (ad esempio un EXOR tra il valore attuale di un modulo di uscita ed il valore 255 per invertire lo stato di ogni uscita del modulo stesso) e poi di scrivere il risultato nella stessa cella.

Esempio 3:
Si vuole attivare il punto virtuale V751 mediante la funzione 16. Il punto virtuale **Vx** si trova, all'interno della mappa RAM, alla locazione:

$$513 + INT\left(\frac{x-1}{8}\right)$$

Poiché un punto virtuale occupa un solo bit, si dovrà specificare quale è quello ad esso associato; questo è dato da:

$$(x - 1)\%8$$

(per il significato dei simboli utilizzati in queste formule, fare riferimento al paragrafo 8.3). In alternativa, l'appendice C riporta una tabella per localizzare facilmente indirizzo RAM e bit relativi ad un dato punto virtuale.
Notare che gli indirizzi RAM per la lettura dei punti virtuali sono diversi da quelli per la scrittura.

Il punto virtuale V751 si è quindi il bit 6 della cella RAM 606; sarà necessario passare al driver MODBUS i seguenti parametri:

Start: **606**
Number: **1** (normalmente, in questo caso, questo parametro non viene richiesto dal driver)
Bit: **6**
Valore: **1** (oppure ON, dipende dal driver utilizzato)

L'esecuzione di questa funzione prevede, come descritto prima, che il driver MODBUS legga la cella di memoria 606 mediante la funzione 3, cambi il bit 6 al valore letto, e poi trasferisca il nuovo valore nella cella 606 mediante la funzione 16. Questa procedura è necessaria in quanto la cella 606 contiene gli stati dei punti virtuali da V745 a V752; poiché si vogliono mantenere invariati gli stati degli altri punti virtuali, risulta evidente come sia indispensabile la lettura preliminare della cella. Questa sequenza, comunque, viene eseguita automaticamente dal driver MODBUS del sistema MASTER.

Esempio 4:
Si vuole scrivere il valore 157 nel registro contatore C22 (si ricorda che nel sistema Contatto i contatori sono numerati da 0 a 512). Si utilizza la funzione 16. L'indirizzo della cella RAM contenente il valore del contatore **Cn** è dato da **1280+n**, per cui, per quanto riguarda il contatore C22, sarà necessario passare al driver MODBUS i seguenti parametri:

Start: **1302**
Number: **1**
Valore: **157**

In questo caso viene scritto direttamente il valore 157 nella cella di memoria 1302.

Esempio 5:
Si vuole impostare i minuti dell'orologio interno di MCP a 36; dalla mappa RAM di MCP risulta che le informazioni relative a data e ora sono:

0x0289	Anno in formato BCD
0x028A	Giorno della settimana BCD
0x028B	Mese in formato BCD
0x028C	Giorno del mese in formato BCD
0x028D	Ore in formato BCD
0x028E	Minuti in formato BCD
0x028F	Secondi in formato BCD

I parametri da passare al driver MODBUS saranno dunque:

Start: **654**
Number: **1**
Valore: **54**

In questo caso viene scritto direttamente il valore 36 nella cella di memoria 654 (0x028E). L'orologio interno verrà di conseguenza aggiornato con il nuovo valore dei minuti (si ricorda che i secondi verranno automaticamente azzerati).

Notare che il valore passato è 54 (decimale), in quanto il registro dei minuti, così come per tutti i registri relativi ai parametri dell'orologio, richiede il formato BCD; infatti, 36 in formato BCD corrisponde a 54 decimale.

10- APPENDICE C: TABELLE PUNTI VIRTUALI

10.1- Lettura punti virtuali

	0100	0101	0102	0103	0104	0105	0106	0107	0108	0109	010A	010B	010C	010D	010E	010F
Bit 0	-	V1	V9	V17	V25	V33	V41	V49	V57	V65	V73	V81	V89	V97	V105	V113
Bit 1	-	V2	V10	V18	V26	V34	V42	V50	V58	V66	V74	V82	V90	V98	V106	V114
Bit 2	-	V3	V11	V19	V27	V35	V43	V51	V59	V67	V75	V83	V91	V99	V107	V115
Bit 3	-	V4	V12	V20	V28	V36	V44	V52	V60	V68	V76	V84	V92	V100	V108	V116
Bit 4	-	V5	V13	V21	V29	V37	V45	V53	V61	V69	V77	V85	V93	V101	V109	V117
Bit 5	-	V6	V14	V22	V30	V38	V46	V54	V62	V70	V78	V86	V94	V102	V110	V118
Bit 6	-	V7	V15	V23	V31	V39	V47	V55	V63	V71	V79	V87	V95	V103	V111	V119
Bit 7	-	V8	V16	V24	V32	V40	V48	V56	V64	V72	V80	V88	V96	V104	V112	V120

	0110	0111	0112	0113	0114	0115	0116	0117	0118	0119	011A	011B	011C	011D	011E	011F
Bit 0	V121	V129	V137	V145	V153	V161	V169	V177	V185	V193	V201	V209	V217	V225	V233	V241
Bit 1	V122	V130	V138	V146	V154	V162	V170	V178	V186	V194	V202	V210	V218	V226	V234	V242
Bit 2	V123	V131	V139	V147	V155	V163	V171	V179	V187	V195	V203	V211	V219	V227	V235	V243
Bit 3	V124	V132	V140	V148	V156	V164	V172	V180	V188	V196	V204	V212	V220	V228	V236	V244
Bit 4	V125	V133	V141	V149	V157	V165	V173	V181	V189	V197	V205	V213	V221	V229	V237	V245
Bit 5	V126	V134	V142	V150	V158	V166	V174	V182	V190	V198	V206	V214	V222	V230	V238	V246
Bit 6	V127	V135	V143	V151	V159	V167	V175	V183	V191	V199	V202	V215	V223	V231	V239	V247
Bit 7	V128	V136	V144	V152	V160	V168	V176	V184	V192	V200	V208	V216	V224	V232	V240	V248

	0120	0121	0122	0123	0124	0125	0126	0127	0128	0129	012A	012B	012C	012D	012E	012F
Bit 0	V249	V257	V265	V273	V281	V289	V297	V305	V313	V321	V329	V337	V345	V353	V361	V369
Bit 1	V250	V258	V266	V274	V282	V290	V298	V306	V314	V322	V330	V338	V346	V354	V362	V370
Bit 2	V251	V259	V267	V275	V283	V291	V299	V307	V315	V323	V331	V339	V347	V355	V363	V371
Bit 3	V252	V260	V268	V276	V284	V292	V300	V308	V316	V324	V332	V340	V348	V356	V364	V372
Bit 4	V253	V261	V269	V277	V285	V293	V301	V309	V317	V325	V333	V341	V349	V357	V365	V373
Bit 5	V254	V262	V270	V278	V286	V294	V302	V310	V318	V326	V334	V342	V350	V358	V366	V374
Bit 6	V255	V263	V271	V279	V287	V295	V303	V311	V319	V327	V335	V343	V351	V359	V367	V375
Bit 7	V256	V264	V272	V280	V288	V296	V304	V312	V320	V328	V336	V344	V352	V360	V368	V376

	0130	0131	0132	0133	0134	0135	0136	0137	0138	0139	013A	013B	013C	013D	013E	013F
Bit 0	V377	V385	V393	V401	V409	V417	V425	V433	V441	V449	V457	V465	V473	V481	V489	V497
Bit 1	V378	V386	V394	V402	V410	V418	V426	V434	V442	V450	V458	V466	V474	V482	V490	V498
Bit 2	V379	V387	V395	V403	V411	V419	V427	V435	V443	V451	V459	V467	V475	V483	V491	V499
Bit 3	V380	V388	V396	V404	V412	V420	V428	V436	V444	V452	V460	V468	V476	V484	V492	V500
Bit 4	V381	V389	V397	V405	V413	V421	V429	V437	V445	V453	V461	V469	V477	V485	V493	V501
Bit 5	V382	V390	V398	V406	V414	V422	V430	V438	V446	V454	V462	V470	V478	V486	V494	V502
Bit 6	V383	V391	V399	V402	V415	V423	V431	V439	V447	V455	V463	V471	V479	V487	V495	V503
Bit 7	V384	V392	V400	V408	V416	V424	V432	V440	V448	V456	V464	V472	V480	V488	V496	V504

	0140	0141	0142	0143	0144	0145	0146	0147	0148	0149	014A	014B	014C	014D	014E	014F
Bit 0	V505	V513	V521	V529	V537	V545	V553	V561	V569	V577	V585	V593	V601	V609	V617	V625
Bit 1	V506	V514	V522	V530	V538	V546	V554	V562	V570	V578	V586	V594	V602	V610	V618	V626
Bit 2	V507	V515	V523	V531	V539	V547	V555	V563	V571	V579	V587	V595	V603	V611	V619	V627
Bit 3	V508	V516	V524	V532	V540	V548	V556	V564	V572	V580	V588	V596	V604	V612	V620	V628
Bit 4	V509	V517	V525	V533	V541	V549	V557	V565	V573	V581	V589	V597	V605	V613	V621	V629
Bit 5	V510	V518	V526	V534	V542	V550	V558	V566	V574	V582	V590	V598	V606	V614	V622	V630
Bit 6	V511	V519	V527	V535	V543	V551	V559	V567	V575	V583	V591	V599	V602	V615	V623	V631
Bit 7	V512	V520	V528	V536	V544	V552	V560	V568	V576	V584	V592	V600	V608	V616	V624	V632

	0150	0151	0152	0153	0154	0155	0156	0157	0158	0159	015A	015B	015C	015D	015E	015F
Bit 0	V633	V641	V649	V657	V665	V673	V681	V689	V697	V705	V713	V721	V729	V737	V745	V753
Bit 1	V634	V642	V650	V658	V666	V674	V682	V690	V698	V706	V714	V722	V730	V738	V746	V754
Bit 2	V635	V643	V651	V659	V667	V675	V683	V691	V699	V707	V715	V723	V731	V739	V747	V755
Bit 3	V636	V644	V652	V660	V668	V676	V684	V692	V700	V708	V716	V724	V732	V740	V748	V756
Bit 4	V637	V645	V653	V661	V669	V677	V685	V693	V701	V709	V717	V725	V733	V741	V749	V757
Bit 5	V638	V646	V654	V662	V670	V678	V686	V694	V702	V710	V718	V726	V734	V742	V750	V758
Bit 6	V639	V647	V655	V663	V671	V679	V687	V695	V703	V711	V719	V727	V735	V743	V751	V759
Bit 7	V640	V648	V656	V664	V672	V680	V688	V696	V704	V712	V720	V728	V736	V744	V752	V760

	0160	0161	0162	0163	0164	0165	0166	0167	0168	0169	016A	016B	016C	016D	016E	016F
Bit 0	V761	V769	V777	V785	V793	V801	V809	V817	V825	V833	V841	V849	V857	V865	V873	V881
Bit 1	V762	V770	V778	V786	V794	V802	V810	V818	V826	V834	V842	V850	V858	V866	V874	V882
Bit 2	V763	V771	V779	V787	V795	V803	V811	V819	V827	V835	V843	V851	V859	V867	V875	V883
Bit 3	V764	V772	V780	V788	V796	V804	V812	V820	V828	V836	V844	V852	V860	V868	V876	V884
Bit 4	V765	V773	V781	V789	V797	V805	V813	V821	V829	V837	V845	V853	V861	V869	V877	V885
Bit 5	V766	V774	V782	V790	V798	V806	V814	V822	V830	V838	V846	V854	V862	V870	V878	V886
Bit 6	V767	V775	V783	V791	V799	V802	V815	V823	V831	V839	V847	V855	V863	V871	V879	V887
Bit 7	V768	V776	V784	V792	V800	V808	V816	V824	V832	V840	V848	V856	V864	V872	V880	V888

	0170	0171	0172	0173	0174	0175	0176	0177	0178	0179	017A	017B	017C	017D	017E	017F
Bit 0	V889	V897	V905	V913	V921	V929	V937	V945	V953	V961	V969	V977	V985	V993	-	-
Bit 1	V890	V898	V906	V914	V922	V930	V938	V946	V954	V962	V970	V978	V986	V994	-	-
Bit 2	V891	V899	V907	V915	V923	V931	V939	V947	V955	V963	V971	V979	V987	V995	-	-
Bit 3	V892	V900	V908	V916	V924	V932	V940	V948	V956	V964	V972	V980	V988	V996	-	-
Bit 4	V893	V901	V909	V917	V925	V933	V941	V949	V957	V965	V973	V981	V989	V997	-	-
Bit 5	V894	V902	V910	V918	V926	V934	V942	V950	V958	V966	V974	V982	V990	V998	-	-
Bit 6	V895	V903	V911	V919	V927	V935	V943	V951	V959	V967	V975	V983	V991	V999	-	-
Bit 7	V896	V904	V912	V920	V928	V936	V944	V952	V960	V968	V976	V984	V992	**V1000**	-	-

10.2- Scrittura punti virtuali

	0200	0201	0202	0203	0204	0205	0206	0207	0208	0209	020A	020B	020C	020D	020E	020F
Bit 0	-	V1	V9	V17	V25	V33	V41	V49	V57	V65	V73	V81	V89	V97	V105	V113
Bit 1	-	V2	V10	V18	V26	V34	V42	V50	V58	V66	V74	V82	V90	V98	V106	V114
Bit 2	-	V3	V11	V19	V27	V35	V43	V51	V59	V67	V75	V83	V91	V99	V107	V115
Bit 3	-	V4	V12	V20	V28	V36	V44	V52	V60	V68	V76	V84	V92	V100	V108	V116
Bit 4	-	V5	V13	V21	V29	V37	V45	V53	V61	V69	V77	V85	V93	V101	V109	V117
Bit 5	-	V6	V14	V22	V30	V38	V46	V54	V62	V70	V78	V86	V94	V102	V110	V118
Bit 6	-	V7	V15	V23	V31	V39	V47	V55	V63	V71	V79	V87	V95	V103	V111	V119
Bit 7	-	V8	V16	V24	V32	V40	V48	V56	V64	V72	V80	V88	V96	V104	V112	V120

	0210	0211	0212	0213	0214	0215	0216	0217	0218	0219	021A	021B	021C	021D	021E	021F
Bit 0	V121	V129	V137	V145	V153	V161	V169	V177	V185	V193	V201	V209	V217	V225	V233	V241
Bit 1	V122	V130	V138	V146	V154	V162	V170	V178	V186	V194	V202	V210	V218	V226	V234	V242
Bit 2	V123	V131	V139	V147	V155	V163	V171	V179	V187	V195	V203	V211	V219	V227	V235	V243
Bit 3	V124	V132	V140	V148	V156	V164	V172	V180	V188	V196	V204	V212	V220	V228	V236	V244
Bit 4	V125	V133	V141	V149	V157	V165	V173	V181	V189	V197	V205	V213	V221	V229	V237	V245
Bit 5	V126	V134	V142	V150	V158	V166	V174	V182	V190	V198	V206	V214	V222	V230	V238	V246
Bit 6	V127	V135	V143	V151	V159	V167	V175	V183	V191	V199	V202	V215	V223	V231	V239	V247
Bit 7	V128	V136	V144	V152	V160	V168	V176	V184	V192	V200	V208	V216	V224	V232	V240	V248

	0220	0221	0222	0223	0224	0225	0226	0227	0228	0229	022A	022B	022C	022D	022E	022F
Bit 0	V249	V257	V265	V273	V281	V289	V297	V305	V313	V321	V329	V337	V345	V353	V361	V369
Bit 1	V250	V258	V266	V274	V282	V290	V298	V306	V314	V322	V330	V338	V346	V354	V362	V370
Bit 2	V251	V259	V267	V275	V283	V291	V299	V307	V315	V323	V331	V339	V347	V355	V363	V371
Bit 3	V252	V260	V268	V276	V284	V292	V300	V308	V316	V324	V332	V340	V348	V356	V364	V372
Bit 4	V253	V261	V269	V277	V285	V293	V301	V309	V317	V325	V333	V341	V349	V357	V365	V373
Bit 5	V254	V262	V270	V278	V286	V294	V302	V310	V318	V326	V334	V342	V350	V358	V366	V374
Bit 6	V255	V263	V271	V279	V287	V295	V303	V311	V319	V327	V335	V343	V351	V359	V367	V375
Bit 7	V256	V264	V272	V280	V288	V296	V304	V312	V320	V328	V336	V344	V352	V360	V368	V376

	0230	0231	0232	0233	0234	0235	0236	0237	0238	0239	023A	023B	023C	023D	023E	023F
Bit 0	V377	V385	V393	V401	V409	V417	V425	V433	V441	V449	V457	V465	V473	V481	V489	V497
Bit 1	V378	V386	V394	V402	V410	V418	V426	V434	V442	V450	V458	V466	V474	V482	V490	V498
Bit 2	V379	V387	V395	V403	V411	V419	V427	V435	V443	V451	V459	V467	V475	V483	V491	V499
Bit 3	V380	V388	V396	V404	V412	V420	V428	V436	V444	V452	V460	V468	V476	V484	V492	V500
Bit 4	V381	V389	V397	V405	V413	V421	V429	V437	V445	V453	V461	V469	V477	V485	V493	V501
Bit 5	V382	V390	V398	V406	V414	V422	V430	V438	V446	V454	V462	V470	V478	V486	V494	V502
Bit 6	V383	V391	V399	V402	V415	V423	V431	V439	V447	V455	V463	V471	V479	V487	V495	V503
Bit 7	V384	V392	V400	V408	V416	V424	V432	V440	V448	V456	V464	V472	V480	V488	V496	V504

	0240	0241	0242	0243	0244	0245	0246	0247	0248	0249	024A	024B	024C	024D	024E	024F
Bit 0	V505	V513	V521	V529	V537	V545	V553	V561	V569	V577	V585	V593	V601	V609	V617	V625
Bit 1	V506	V514	V522	V530	V538	V546	V554	V562	V570	V578	V586	V594	V602	V610	V618	V626
Bit 2	V507	V515	V523	V531	V539	V547	V555	V563	V571	V579	V587	V595	V603	V611	V619	V627
Bit 3	V508	V516	V524	V532	V540	V548	V556	V564	V572	V580	V588	V596	V604	V612	V620	V628
Bit 4	V509	V517	V525	V533	V541	V549	V557	V565	V573	V581	V589	V597	V605	V613	V621	V629
Bit 5	V510	V518	V526	V534	V542	V550	V558	V566	V574	V582	V590	V598	V606	V614	V622	V630
Bit 6	V511	V519	V527	V535	V543	V551	V559	V567	V575	V583	V591	V599	V602	V615	V623	V631
Bit 7	V512	V520	V528	V536	V544	V552	V560	V568	V576	V584	V592	V600	V608	V616	V624	V632

	0250	0251	0252	0253	0254	0255	0256	0257	0258	0259	025A	025B	025C	025D	025E	025F
Bit 0	V633	V641	V649	V657	V665	V673	V681	V689	V697	V705	V713	V721	V729	V737	V745	V753
Bit 1	V634	V642	V650	V658	V666	V674	V682	V690	V698	V706	V714	V722	V730	V738	V746	V754
Bit 2	V635	V643	V651	V659	V667	V675	V683	V691	V699	V707	V715	V723	V731	V739	V747	V755
Bit 3	V636	V644	V652	V660	V668	V676	V684	V692	V700	V708	V716	V724	V732	V740	V748	V756
Bit 4	V637	V645	V653	V661	V669	V677	V685	V693	V701	V709	V717	V725	V733	V741	V749	V757
Bit 5	V638	V646	V654	V662	V670	V678	V686	V694	V702	V710	V718	V726	V734	V742	V750	V758
Bit 6	V639	V647	V655	V663	V671	V679	V687	V695	V703	V711	V719	V727	V735	V743	V751	V759
Bit 7	V640	V648	V656	V664	V672	V680	V688	V696	V704	V712	V720	V728	V736	V744	V752	V760

	0260	0261	0262	0263	0264	0265	0266	0267	0268	0269	026A	026B	026C	026D	026E	026F
Bit 0	V761	V769	V777	V785	V793	V801	V809	V817	V825	V833	V841	V849	V857	V865	V873	V881
Bit 1	V762	V770	V778	V786	V794	V802	V810	V818	V826	V834	V842	V850	V858	V866	V874	V882
Bit 2	V763	V771	V779	V787	V795	V803	V811	V819	V827	V835	V843	V851	V859	V867	V875	V883
Bit 3	V764	V772	V780	V788	V796	V804	V812	V820	V828	V836	V844	V852	V860	V868	V876	V884
Bit 4	V765	V773	V781	V789	V797	V805	V813	V821	V829	V837	V845	V853	V861	V869	V877	V885
Bit 5	V766	V774	V782	V790	V798	V806	V814	V822	V830	V838	V846	V854	V862	V870	V878	V886
Bit 6	V767	V775	V783	V791	V799	V802	V815	V823	V831	V839	V847	V855	V863	V871	V879	V887
Bit 7	V768	V776	V784	V792	V800	V808	V816	V824	V832	V840	V848	V856	V864	V872	V880	V888

	0270	0271	0272	0273	0274	0275	0276	0277	0278	0279	027A	027B	027C	027D	027E	027F
Bit 0	V889	V897	V905	V913	V921	V929	V937	V945	V953	V961	V969	V977	V985	V993	-	-
Bit 1	V890	V898	V906	V914	V922	V930	V938	V946	V954	V962	V970	V978	V986	V994	-	-
Bit 2	V891	V899	V907	V915	V923	V931	V939	V947	V955	V963	V971	V979	V987	V995	-	-
Bit 3	V892	V900	V908	V916	V924	V932	V940	V948	V956	V964	V972	V980	V988	V996	-	-
Bit 4	V893	V901	V909	V917	V925	V933	V941	V949	V957	V965	V973	V981	V989	V997	-	-
Bit 5	V894	V902	V910	V918	V926	V934	V942	V950	V958	V966	V974	V982	V990	V998	-	-
Bit 6	V895	V903	V911	V919	V927	V935	V943	V951	V959	V967	V975	V983	V991	V999	-	-
Bit 7	V896	V904	V912	V920	V928	V936	V944	V952	V960	V968	V976	V984	V992	**V1000**	-	-