INDICE

3.0 - PROGRAMMAZIONE AVANZATA

4.0 - MODULI SPECIALI

5.0 – ESEMPI APPLICATIVI

1.0 - Generalità

Il sistema PIC*net* è un sistema per l'automazione degli impianti tecnologici con tecnologia "bus" completamente configurabile dall'utente. Mediante un opportuno programma è possibile definire le relazioni che legano le uscite del sistema con i suoi ingressi.
Tali relazioni risiedono in un programma utente che, una volta caricato nel modulo master del sistema, viene continuamente eseguito in tempo reale.

1.1 - Per cominciare

La programmazione avviene attraverso tre fasi (vedi anche Manuale di Installazione §3.0):

1 - Scrittura del programma utente.
La scrittura del programma può essere effettuata utilizzando un comune editor di testi in ambiente operativo DOS o preferibilmente l'editor integrato nel software PN LINK. I file generati dall'editor interno hanno l'estensione .pns.
I caratteri ammessi sono tutti quelli del set ASCII standard. Eventuali caratteri non ASCII presenti in una stringa verranno convertiti nel carattere '_'.

2 - Compilazione del programma utente.
Tale operazione si effettua lanciando il comando *compila* del sw PN LINK (da barra menù o mediante il pulsante di compilazione) in modo da generare il codice eseguibile che andrà scaricato sul modulo master.
In mancanza di errori il programma genererà un file eseguibile con estensione .pne.

Se il comando *compila* rileva errori di sintassi, invia un messaggio di allarme e segnala a video gli errori rilevati.

3 - Invio programma utente al modulo master.
Tale operazione si effettua lanciando il comando *invia*. Quando l'invio del programma è terminato sarà sufficiente metterlo in esecuzione premendo sul tasto di *avvio*. Da questo momento in poi il master continuerà l'esecuzione del programma fino a quando non verrà fermato da un comando di *stop*.

1.2 - Simbologia

Nel presente manuale e nei documenti collegati verranno utilizzati termini e simboli nel seguito riportati.

1.2.1 - Ingresso

Col termine ingresso si indica un generico elemento che fornisce al sistema PIC*net* una informazione di tipo logico (ON/OFF) o analogico relativa a quanto avviene sul campo (es. pulsante premuto, interruttore scattato, valore temperatura, ecc.).

Un generico ingresso digitale verrà sempre indicato con il simbolo Ix.y dove con x si intende il numero di modulo (indirizzo di sistema) e con y il numero dell'ingresso all'interno del modulo.
Il numero x può assumere tutti i valori tra 1 e 250, mentre y sarà compreso tra 1 e 2 per i moduli a 2 ingressi (PN2I2O/W/WP), tra 1 e 4 per i moduli dotati di 4 ingressi (PN4I4O) oppure tra 1 e 8 per i moduli ad 8 ingressi (PN8I8O, PN8I o PN8IW).

Un ingresso analogico verrà invece sempre indicato con il simbolo Ix dove con x si intende il numero del modulo (indirizzo di sistema).
Nel caso di moduli di ingresso a due canali ad 8 bit, l'indirizzo di sistema è unico per i due canali che vengono identificati con le lettere A e B.

Ad es. I2:A e I2:B indicano rispettivamente il primo ed il secondo canale del modulo di ingresso analogico n° 2.

1.2.2 - Uscita

Col termine uscita si indica un elemento generico che interviene sul campo su comando del sistema PIC*net* sulla base del valore attuale degli ingressi e delle relazioni di ingresso-uscita definite dall'utente nel suo programma.

Una generica uscita digitale verrà sempre indicata con il simbolo Ox.y dove con x si intende il numero di modulo (indirizzo di sistema) e con y il numero dell'uscita all'interno del modulo.
Il numero x può assumere tutti i valori tra 1 e 250, mentre y sarà compreso tra 1 e 2 per i moduli a 2 uscite (PN2I2O/W/WP), tra 1 e 4 per i moduli a 4 uscite (PN4I4O) oppure tra 1 e 8 per i moduli ad 8 uscite (PN8OC, PN8OT, PN8I8O o PN8OR).

Una uscita analogica verrà invece sempre indicata con il simbolo Ox dove con x si intende il numero del modulo (indirizzo di sistema).
Nel caso di moduli di uscita a due canali ad 8 bit, l'indirizzo di sistema è unico per i due canali che vengono identificati con le lettere A e B.

Ad es. O2:A e O2:B indicano rispettivamente il primo ed il secondo canale del modulo di uscita analogico n° 2.

Si noti che un'assegnazione di un'uscita analogica del tipo Om = In equivale all'insieme delle seguenti 8 assegnazioni:

Om.1 = In.1
Om.2 = In.2
Om.3 = In.3
.....
Om.7 = In.7
Om.8 = In.8

La stessa espressione può essere scritta indifferentemente in una delle seguenti forme tutte equivalenti:

```
Om = In;
Om = In:A;
Om:A = In;
Om:A = In:A;
```

Esiste anche la possibilità di accedere ai singoli bit che compongono il byte della grandezza convertita, ad es. per effettuare dei confronti di superamento di soglia.

Ad es. per verificare se un segnale è superiore al 50% del suo fondo scala basterà controllare che il bit più significativo sia diverso da zero.

Il valore attuale di una uscita può essere utilizzato all'interno di una espressione logica.
Sono ad esempio possibili espressioni del tipo:

```
O2.1 = T[ I2.1 | (I2.2 & !O2.1)];
```

in cui l'uscita O2.1 viene commutata da un passo passo comandato dall'ingresso I2.1 (sempre attivo) e dall'ingresso I2.2 (attivo solo se l'uscita è spenta).

Al momento del reset del sistema, tutte le uscite vengono automaticamente forzate al valore 0 o al valore che possedevano prima dello spegnimento del sistema a seconda dell'impostazione del flag di sistema RAMBACKUP (vedi par. 2.4.7).

1.2.3 – Operatori di confronto

In alcune funzioni di programmazione è richiesto l'utilizzo di operatori di confronto per decidere lo stato da assegnare ad un' uscita sulla base del valore di una determinata grandezza. Gli operatori di confronto utilizzabili sono i seguenti:

== : operatore 'uguale'
!= : operatore 'diverso
> : operatore 'maggiore'
< : operatore 'minore'
>= : operatore 'maggiore o uguale'
<= : operatore 'minore o uguale'

Ad es. nell'equazione O2.1 = I3 > 122, l'uscita n° 1 del modulo 2 verrà attivata quando l'ingresso analogico n° 3 supererà il valore di 122.

1.2.4 – Operatori matematici

E' possibile effettuare delle elaborazioni sui valori assunti da variabili interne o grandezze di ingresso analogico utilizzando i normali operatori matematici di somma, sottrazione, moltiplicazione e divisione.

Ad es. nell'equazione V2 = 15*I2 + 4*I3/3, la variabile V2 assume il valore risultante dall'espressione a secondo membro che dipende dai due ingressi analogici I2 e I3.

Non esiste alcun limite nell'uso degli operatori matematici con ingressi, variabili utente ed operatori di confronto che possono dunque essere liberamente utilizzati all'interno di una stessa espressione.
E' cioè possibile effettuare operazioni logiche su variabili intere (l'operazione viene eseguita su ogni singolo bit) o effettuare moltiplicazioni tra una variabile intera ed una variabile logica.

Ad es. posto che I2 e I3 siano moduli di ingresso (digitale o analogico) e che O8 sia un modulo di uscita (digitale o analogico), sarà possibile scrivere equazioni complesse del tipo:

```
O8 = 100*(I2 != 128) + 3*I2*I3.1;
```

1.2.5 – Display e messaggi

Col termine display si indica un elemento in grado di visualizzare dei messaggi del sistema.
Ogni modulo display ha la possibilità di memorizzare (in fase di inizializzazione) fino a 255 messaggi (1-255). La loro visualizzazione avviene su condizione secondo la sintassi descritta nel paragrafo 2.3.14.

Un generico display viene indicato con il simbolo Dy dove con y si intende il numero del modulo (indirizzo di sistema).
Il numero y può assumere tutti i valori compresi tra 1 e 250.
Il display del modulo master, se presente, è indirizzato con il valore 254.

Un generico messaggio viene indicato con il simbolo Mx dove con x si intende il numero del messaggio.
Il numero x può assumere tutti i valori tra 1 e 255.

1.2.6 - Allarmi

Col termine allarme si indica la presenza di una condizione di funzionamento anomala del sistema PIC*net* o dell'impianto in cui è inserito.
Mentre le condizioni di allarme del sistema PIC*net* sono già state predefinite e sono gestite in maniera automatica dal modulo master del sistema, gli allarmi di impianto sono liberamente definibili dall'utente.
Esiste cioè la possibilità di associare al verificarsi di una opportuna condizione logica, la segnalazione di un allarme.
Un generico allarme verrà indicato con il simbolo Ax dove con x si intende il numero di allarme, che può assumere tutti i valori tra 1 e 255.

Lo stato degli allarmi (si veda paragrafo 3.1-3.2) può essere utilizzato all'interno di espressioni logiche. Lo stato attivo di un allarme corrisponde al valore ON.

1.3 - Struttura di un programma e sintassi equazioni

Un programma utente è costituito da un insieme di equazioni che definiscono le relazioni richieste tra gli ingressi e le uscite del sistema.

Esiste inoltre la possibilità di definire alcuni elementi accessori del sistema (quali la data e ora, ecc.) direttamente all'interno di un programma mediante apposite variabili dette variabili di ambiente (vedi oltre par. 2.4).
La definizione delle variabili d'ambiente non è necessaria (in loro mancanza il sistema assume valori di default).

Ogni equazione è descritta da un'espressione del tipo:

Ox.y = f(Xn.m);

dove f(Xn.m) è una generica funzione che mediante operatori e funzioni (vedi oltre) definisce il legame tra gli ingressi del sistema, le uscite, gli allarmi e/o le variabili.

Nella scrittura delle equazioni si consideri che:

- **ogni equazione deve terminare con punto e virgola (;)**
- **un'equazione può essere scritta su più righe; in tal caso solo l'ultima riga del gruppo deve terminare con punto e virgola**
- **la presenza di spazi e tabulazioni all'interno di un'equazione non è significativa; non è tuttavia ammessa l'interruzione di un operatore o un simbolo, ad es.:**

O1.1=I2.2; **e** **O1.1 = I2.2;**

sono scritture entrambe valide, mentre:

O 1. 1=I 2 . 2;

è una scrittura errata.

- **eventuali commenti devono essere preceduti dai caratteri //.**
  Tutto ciò che segue il simbolo di commento "//" fino al termine della riga verrà ignorato dal compilatore.

Ciascun ingresso, variabile, allarme, ecc. può comparire più volte all'interno di un programma o di una stessa equazione, ma non è ammesso definire più volte la stessa uscita, la stessa variabile, lo stesso allarme, messaggio o display all'interno di un programma.

Non esiste una priorità di esecuzione tra i vari operatori logici; all'interno di una equazione composta da più operatori logici e funzioni, le singole istruzioni vengono eseguite da sinistra a destra.
Qualora si voglia alterare questa priorità di esecuzione occorre introdurre delle parentesi tonde opportune.

Ad es.:

```
O1.1 = I2.2 & I4.2 | I2.5;       e    O1.1 = (I2.2 & I4.2) | I2.5;
```

sono scritture equivalenti, mentre:

```
O1.1 = I2.2 & (I4.2 | I2.5);
```

è una scrittura differente perché prevede prima l'esecuzione dell'istruzione "or" (|) e successivamente l'esecuzione dell'istruzione "and" (&).

Per gli operatori matematici valgono invece le consuete regole nella priorità di esecuzione per cui gli operatori "*" (moltiplicazione) e "/" (divisione) hanno precedenza rispetto agli operatori "+" (addizione) e "-" (sottrazione).
Qualora in una stessa espressione vi siano più operatori matematici di pari priorità, questi vengono eseguiti da sinistra a destra.
Anche in questo caso, qualora si voglia alterare questa priorità di esecuzione occorre introdurre delle parentesi tonde opportune.

Ad es.:

```
O1 = I2 * 10 / 3 + 200;  e       O1 = ((I2 * 10) /3 ) + 200;
```

sono scritture equivalenti, mentre:

```
O1 = I2 * 100 / (3 + 200);
```

è una scrittura differente perché prevede prima l'esecuzione dell'istruzione "I2*100" e della somma (3+200) e successivamente l'esecuzione dell'istruzione "/" .

Poiché è possibile l'utilizzo di variabili dotate di segno (vedi oltre par. 3.6), è possibile anche utilizzare il segno "-" per indicare l'inversione di segno di un'espressione.

Ad esempio, posto che Var sia una variabile intera si potranno scrivere espressioni del tipo:

```
Var = - ( I2 * 3 ) - 100;
```

Esistono, predefinite, le seguenti costanti logiche: ON e OFF. Esse possono essere utilizzate in qualunque espressione logica al posto del valore di una grandezza. Non ha invece significato utilizzarle in espressioni sensibili ai fronti dei segnali.

Non esiste un limite fisso al numero di equazioni all'interno di un programma né al numero di funzioni o di ingressi che compaiono all'interno di una singola equazione.
Tale limite dipende dalla quantità di memoria utente a disposizione nel modulo master.

2.0 - Programmazione base

2.1 - Programmazione: operatori base

2.1.1 - Operatore "not"

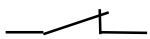
L'operatore "not" permette di ottenere il negato di una espressione. Nel sistema PIC*net* l'operatore "not" è rappresentato dal simbolo "!".

Es. : `O2.1 = !I3.2;`

L'uscita n° 1 del modulo di uscita 2 sarà 1 solo se l'ingresso n° 2 del modulo 3 è uguale a 0 e viceversa.

Il funzionamento dell'operatore "not" è rappresentato dalla seguente tabella:

Tabella logica (o della verità) operatore "not"



Input	*Output*
0	1
1	0

2.1.2 - Operatore "or"

L'operatore "or" lega un'uscita a due ingressi. L'uscita vale 1 solo se almeno uno dei due ingressi è uguale a 1.
L'operatore "or" permette di ottenere un funzionamento del tipo collegamento in parallelo.

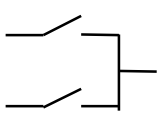
Nel sistema PIC*net* l'operatore "or" è rappresentato dal simbolo "|".

Es. : `O2.1 = I3.2 | I3.1;`

L'uscita n° 1 del modulo di uscita 2 sarà 1 se l'ingresso n° 2 o l'ingresso n° 1 del modulo 3 hanno valore 1.

Il funzionamento dell'operatore "or" è rappresentato dalla seguente tabella:

Tabella logica (o della verità) operatore "or"



I1	*I2*	*Output*
0	0	0
0	1	1
1	0	1
1	1	1

2.1.3 - Operatore "exor"

L'operatore "exor" lega un'uscita a due ingressi. L'uscita vale 1 solo se i due ingressi hanno valori differenti.
L'operatore "exor" permette di ottenere un funzionamento del tipo punto luce deviato.

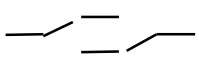
Nel sistema PIC*net* l'operatore "exor" è rappresentato dal simbolo "^".

Es. : `O2.1 = I3.2 ^ I3.1;`

L'uscita n° 1 del modulo di uscita 2 sarà 1 solo se l'ingresso n° 2 e l'ingresso n° 1 del modulo 3 hanno valori differenti.

Il funzionamento dell'operatore "exor" è rappresentato dalla seguente tabella:

Tabella logica (o della verità) operatore "exor"



I1	*I2*	*Output*
0	0	0
0	1	1
1	0	1
1	1	0

2.1.4 – Operatore "and"

L'operatore "and" lega un'uscita a due ingressi. L'uscita vale 1 solo se i due ingressi sono entrambi pari ad 1.
L'operatore "and" permette di ottenere un funzionamento del tipo collegamento in serie.

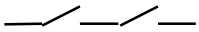
Nel sistema PIC*net* l'operatore "and" è rappresentato dal simbolo "&".

Es. : `O2.1 = I3.2 & I3.1;`

L'uscita n° 1 del modulo di uscita 2 sarà 1 solo se l'ingresso n° 2 e l'ingresso n° 1 del modulo 3 hanno entrambi valore 1.

Il funzionamento dell'operatore "and" è rappresentato dalla seguente tabella:

Tabella logica (o della verità) operatore "and"



I1	*I2*	*Output*
0	0	0
0	1	0
1	0	0
1	1	1

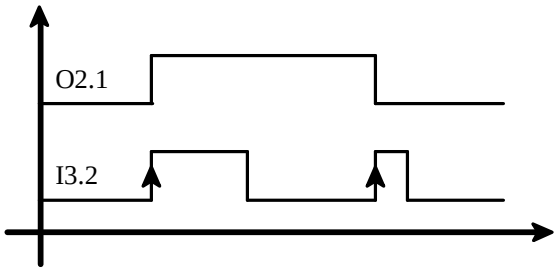
2.2 - Programmazione: operatori complessi

2.2.1 - Operatore "passo-passo" (TOGGLE)

L'operatore "passo-passo" permette di variare un'uscita in corrispondenza di fronti di salita di una espressione di ingresso.
Nel sistema PIC*net* l'operatore "passo-passo" è rappresentato dal simbolo "T[E]" dove E rappresenta l'espressione logica cui si applica (normalmente il valore di un ingresso).

Es. : `O2.1 = T[I3.2];`

L'uscita n° 1 del modulo di uscita 2 avrà il comportamento mostrato dal diagramma seguente dove si mostra l'andamento nel tempo dei segnali di ingresso e di uscita:

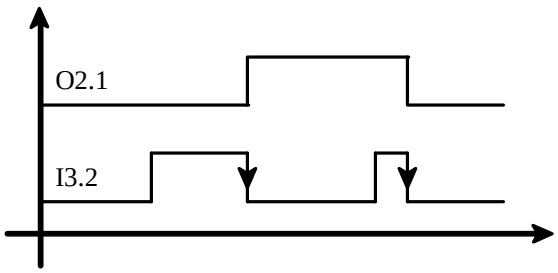


O2.1 = T[I3.2];

L'operatore "not" (simbolo "!") posto prima dell'espressione logica permette di modificare il funzionamento prima descritto utilizzando il fronte di discesa dei segnali anziché quello di salita per determinare la commutazione dell'uscita.

Es. : `O2.1 = T[!I3.2];`

L'uscita n° 1 del modulo di uscita 2 avrà il comportamento mostrato dal diagramma seguente dove si mostra l'andamento nel tempo dei segnali di ingresso e di uscita:



O2.1 = T[!I3.2];

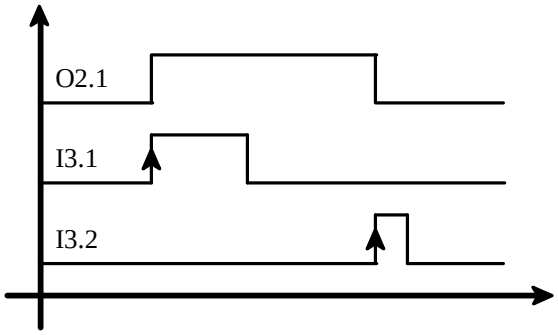
2.2.2 - Operatore "set-reset"

L'operatore "set-reset" permette di variare un'uscita in corrispondenza di variazioni di due espressioni di ingresso per ottenere un funzionamento del tipo "marcia-arresto".
Nel sistema PIC*net* l'operatore "set-reset" è rappresentato dal simbolo "SR[E1,E2]" dove E1 rappresenta l'espressione logica che attiva l'uscita e E2 l'espressione che la disattiva.

Es. : `O2.1 = SR[I3.1, I3.2];`

L'uscita n° 1 del modulo di uscita 2 assumerà il valore 1 sul fronte di salita dell'ingresso 1 del modulo 3 e si resetterà sul fronte di salita dell'ingresso 2.

Il comportamento sarà quello mostrato dal diagramma seguente dove si mostra l'andamento nel tempo dei segnali di ingresso e di uscita:



O2.1 = SR[I3.1, I3.2];

Essendo E1 ed E2 due generiche espressioni logiche, sarà possibile al loro interno una qualsiasi combinazione di operatori.
Ad esempio l'operatore "not" (simbolo "!") posto prima dell'espressione di "set" o di "reset" permette di modificare il funzionamento prima descritto utilizzando il fronte di discesa dei segnali anziché quello di salita per determinare la commutazione dell'ingresso.

Es. : `O2.1 = SR[!I3.1, I3.2];`

L'uscita n° 1 del modulo di uscita 2 assumerà il valore 1 sul fronte di discesa dell'ingresso 1 del modulo 3 e si resetterà sul fronte di salita dell'ingresso 2.

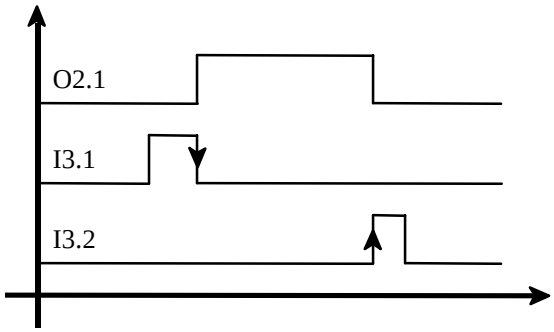
Es. : `O2.1 = SR[I3.1, !I3.2];`

L'uscita n° 1 del modulo di uscita 2 assumerà il valore 1 sul fronte di salita dell'ingresso 1 del modulo 3 e si resetterà sul fronte di discesa dell'ingresso 2.

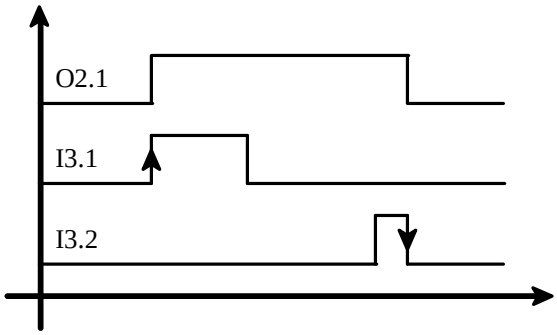
Es. : `O2.1 = SR[!I3.1, !I3.2];`

L'uscita n° 1 del modulo di uscita 2 assumerà il valore 1 sul fronte di discesa dell'ingresso 1 del modulo 3 e si resetterà sul fronte di discesa dell'ingresso 2.

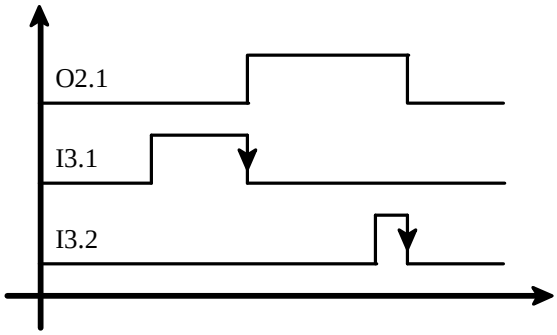
Il comportamento sarà quello mostrato dal diagramma seguente dove si mostra l'andamento nel tempo dei segnali di ingresso e di uscita:



O2.1 = SR[!I3.1, I3.2];



O2.1 = SR[I3.1, !I3.2];



O2.1 = SR[!I3.1, !I3.2];

2.3 - Programmazione: funzioni con parametri

2.3.1 - Funzione "timer"

La funzione "timer" permette di attivare un'uscita del sistema in funzione di una variazione di una espressione di ingresso con un ritardo programmabile. Si possono così realizzare i funzionamenti di timer ritardato all'eccitazione o alla diseccitazione.

Nel sistema PIC*net* la funzione "timer" è rappresentata dal simbolo "TMR" secondo la seguente sintassi:

TMR[t1, t2, E1, E2]

dove: t1 : tempo di ritardo all'eccitazione in decimi di secondo (max. 32767)
Il campo t1 può essere sia una costante che una variabile
t2 : tempo di ritardo alla diseccitazione in decimi di secondo (max. 32767)
Il campo t2 può essere sia una costante che una variabile
E1 : espressione logica di avvio temporizzazione (funziona sul fronte del segnale)
E2 : espressione logica di reset temporizzazione (funziona sul valore del segnale)

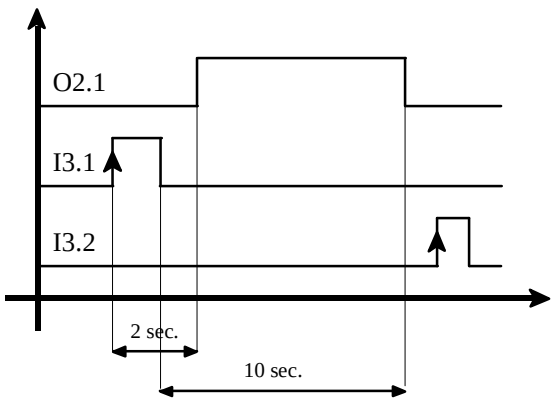
NOTA: se una temporizzazione risulta inferiore alla durata del ciclo di esecuzione del programma, non ne è garantito il corretto funzionamento.

Es. :

```
O2.1 = TMR[20, 100, I3.1, I3.2];
```

L'uscita n° 1 del modulo di uscita 2 si attiverà 2 secondi dopo il fronte di salita dell'ingresso 1 del modulo 3 e rimarrà attiva per 10 secondi dopo il fronte di discesa dello stesso ingresso. L'ingresso 2 del modulo 3 potrà resettare la temporizzazione in qualunque momento.

Il comportamento dell'uscita è mostrato dal diagramma seguente dove si mostra l'andamento nel tempo dei segnali di ingresso e di uscita:



L'operatore "not" (simbolo "!") posto prima di una espressione logica permette di modificare il funzionamento prima descritto utilizzando il fronte di discesa dei segnali anziché quello di salita per determinare la commutazione dell'uscita.

2.3.2 - Funzione "shot"

La funzione "shot" permette di attivare un'uscita del sistema in funzione di una variazione di una espressione di ingresso per un tempo programmabile. Si può così realizzare il funzionamento di un relè monostabile.

Nel sistema PIC*net* la funzione "shot" è rappresentata dal simbolo "SHOT" secondo la seguente sintassi:

SHOT[t1, E1]

dove: t1 : tempo di ritardo alla diseccitazione in decimi di secondo (max. 32767)
Il campo t1 può essere sia una costante che una variabile
E1 : espressione logica di avvio temporizzazione (funziona sul fronte del segnale)

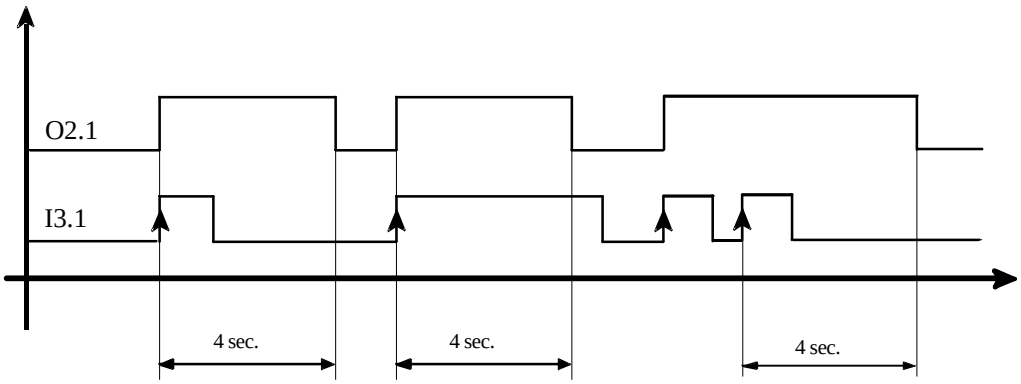
NOTA: se una temporizzazione risulta inferiore alla durata del ciclo di esecuzione del programma, non ne è garantito il corretto funzionamento.

Es. :

```
O2.1 = SHOT[40, I3.1];
```

L'uscita n° 1 del modulo di uscita 2 si attiverà sul fronte di salita dell'ingresso 1 del modulo 3 e rimarrà attiva per 4 secondi.

Il comportamento dell'uscita è mostrato dal diagramma seguente dove si mostra l'andamento nel tempo dei segnali di ingresso e di uscita:



L'operatore "not" (simbolo "!") posto prima di una espressione logica permette di modificare il funzionamento prima descritto utilizzando il fronte di discesa dei segnali anziché quello di salita per determinare la commutazione dell'uscita.

2.3.3 - Funzione "ritardo"

La funzione "ritardo" permette di attivare un'uscita del sistema con un ritardo prefissato rispetto ad un segnale in ingresso. Si possono così facilmente generare delle sequenze di eventi innescate da un'unica condizione di attivazione

Nel sistema PIC*net* la funzione "ritardo" è rappresentata dal simbolo "DELAY" secondo la seguente sintassi:

DELAY[t1, E1]

dove: t1 : tempo di ritardo alla attivazione in secondi (max. 32767)
Il campo t1 può essere sia una costante che una variabile
E1 : espressione logica di avvio sequenza (funziona sul fronte del segnale)

Es. : `O2.1 = DELAY[40, I3.1];`

L'uscita n° 1 del modulo di uscita 2 si attiverà 40 secondi dopo che si è verificato un fronte di salita dell'ingresso 1 del modulo 3 e si disattiverà 40 secondi dopo il fronte di discesa.

E' possibile interrompere una sequenza attivata aggiungendo un campo opzionale di reset.
In tal caso la sintassi completa è la seguente:

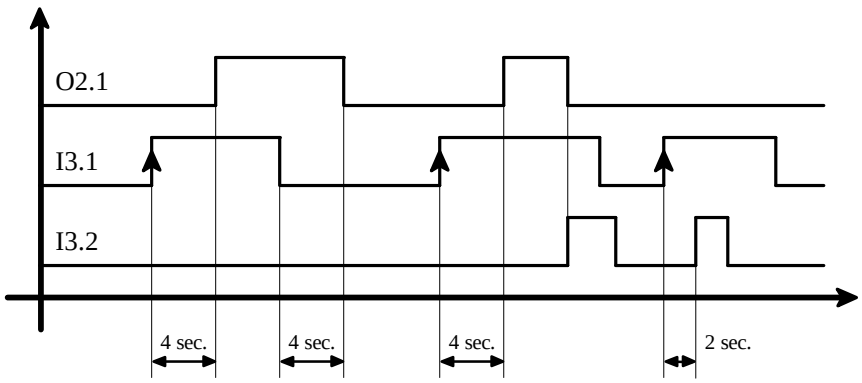
DELAY[t1, E1, E2]

dove: t1 : tempo di ritardo alla attivazione in secondi (max. 32767).
Il campo t1 può essere sia una costante che una variabile
E1 : espressione logica di avvio sequenza (funziona sul fronte del segnale)
E2 : espressione logica di reset sequenza (funziona sul valore del segnale)

Es. : `O2.1 = DELAY[4, I3.1, I3.2];`

L'uscita n° 1 del modulo di uscita 2 si attiverà 4 secondi dopo che si è verificato un fronte di salita dell'ingresso I3.1 e tornerà a zero 4 secondi dopo che si è verificato un fronte di discesa.
Se l'ingresso I3.2 viene attivato in qualunque momento la sequenza viene resettata e l'uscita torna a 0.

Il comportamento dell'uscita è mostrato dal diagramma seguente dove si mostra l'andamento nel tempo dei segnali di ingresso e di uscita:



2.3.4 - Funzione "PWM"

La funzione "pwm" permette di attivare un'uscita in modo intermittente in funzione di una variazione di una espressione logica.
Si può così realizzare il funzionamento di lampeggio controllato da un ingresso.

Nel sistema PIC*net* la funzione "pwm" è rappresentata dal simbolo "PWM" secondo la seguente sintassi:

PWM[t1, t2, E1]

dove: t1 : tempo di accensione in decimi di secondo (max. 32767)
Il campo t1 può essere sia una costante che una variabile
t2 : tempo di spegnimento in decimi di secondo (max. 32767)
Il campo t2 può essere sia una costante che una variabile
E1 : espressione logica di abilitazione dell'uscita (funziona sul valore del segnale)

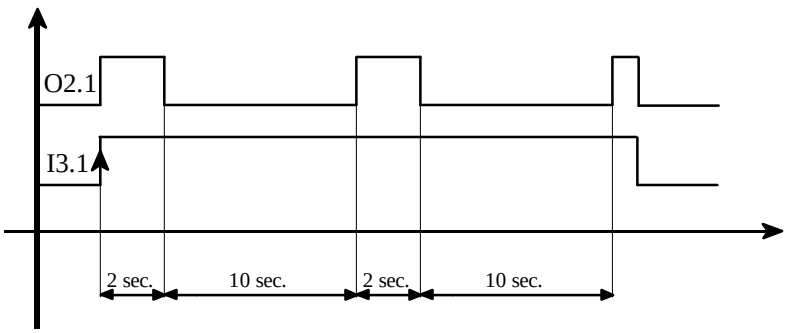
NOTA: se una temporizzazione risulta inferiore alla durata del ciclo di esecuzione del programma, non ne è garantito il corretto funzionamento.

Es. :

```
O2.1 = PWM[20, 100, I3.1];
```

L'uscita n° 1 del modulo di uscita 2 si attiverà ciclicamente per 2 secondi intervallati da 10 secondi di pausa dopo il fronte di salita dell'ingresso 1 del modulo 3 e continuerà tale funzionamento per un tempo indefinito (fino alla discesa dell'ingresso 1 del modulo 3).

Il comportamento dell'uscita è mostrato dal diagramma seguente dove si mostra l'andamento nel tempo dei segnali di ingresso e di uscita:



Oltre che per generare accensioni lampeggianti di uscite, la funzione PWM risulta molto utile all'interno di programmi per la generazione di segnali ad intervalli di tempo regolari, ad es. per visualizzare dei messaggi con una cadenza prefissata.

2.3.5 - Funzione "contatore singolo"

La funzione "contatore" permette di effettuare il conteggio di un segnale e di attivare un'uscita in funzione del valore assunto dal conteggio.
I valori estremi del conteggio sono 0 e 32767 e non possono essere superati.

Nel sistema PIC*net* la funzione "contatore singolo" è rappresentata dal simbolo "CNT1" secondo la seguente sintassi:

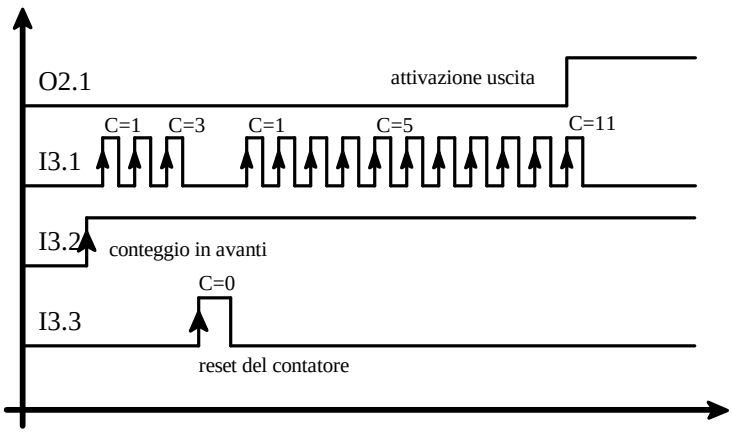
CNT1[E1, E2, E3]

dove: **E1 : espressione logica la cui transizione da 0 ad 1 modifica il conteggio**
E2 : espressione logica di direzione conteggio (funziona sul valore del segnale):
E2 = ON : conteggio in avanti (incremento)
E2 = OFF : conteggio all'indietro (decremento)
E3: espressione logica di reset del conteggio (funziona sul valore del segnale)

Es. : `O2.1 = CNT1[I3.1, I3.2, I3.3] >= 11;`

L'uscita n° 1 del modulo di uscita 2 si attiverà quando il contatore sarà maggiore o uguale al valore 11. Tale contatore verrà modificato dagli impulsi rilevati sull'ingresso I3.1, potrà essere resettato da un impulso sull'ingresso I3.3 e si incrementerà se I3.2 = ON, si decrementerà se I3.2 = OFF.

Il comportamento dell'uscita è mostrato dal diagramma seguente dove si mostra l'andamento nel tempo dei segnali di ingresso e di uscita:



O2.1 = CNT1[I3.1, I3.2, I3.3] >= 11;

Tutti i campi dell'espressione devono sempre essere presenti; qualora non sia richiesto il funzionamento di un parametro, questo dovrà essere sostituito da una costante logica (ON o OFF).

Es. :
```
O2.1 = CNT1[I3.1, ON, OFF] < 22;
```

L'uscita n° 1 del modulo di uscita 2 si attiverà quando il contatore sarà minore del valore 22. Tale contatore verrà sempre incrementato dagli impulsi rilevati sull'ingresso I3.1, e non sarà mai resettato.

Per soddisfare particolari applicazioni (ad es. per ottenere conteggi di valore superiore agli estremi di conteggio indicati) è possibile porre più contatori in cascata.

Es. :
```
V1.1 = CNT1[I3.1, ON, V1.1 | I2.3] ==3;
O2.4 = CNT1[V1.1, ON, I2.3] >=3;
```

Nell'esempio l'uscita n° 4 del modulo di uscita 2 si attiverà quando l'ingresso I3.1 avrà presentato 9 fronti di salita. Infatti il primo contatore conta fino a 3 e poi si autoresetta. Il secondo contatore sente i fronti di salita del primo (attraverso la variabile V1.1). L'ingresso I2.3 è in grado di resettare entrambi i contatori.

La funzione contatore restituisce in uscita un valore intero che può essere assegnato ad una variabile qualsiasi per successive elaborazioni.
Ad esempio le espressioni precedenti possono essere scritte anche nel seguente modo:

Es. :
```
V2 = CNT1[I3.1, ON, V1.1 | I2.3];
V1.1 = V2 ==3;
V3 = CNT1[V1.1, ON, I2.3];
O2.4 = V3 >=3;
```

2.3.6 - Funzione "contatore doppio"

La funzione "contatore doppio" permette di effettuare il conteggio su due segnali, di cui uno in incremento e l'altro in decremento, e di attivare un'uscita in funzione del valore assunto dal conteggio.
I valori estremi del conteggio sono 0 e 32767 e non possono essere superati.

Nel sistema PIC*net* la funzione "contatore doppio" è rappresentata dal simbolo "CNT2" secondo la seguente sintassi:

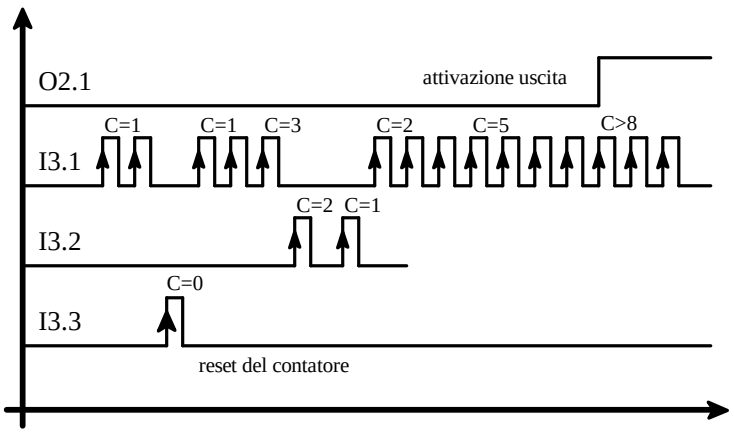
CNT2[E1, E2, E3]

dove: **E1 : espressione logica la cui transizione da 0 ad 1 incrementa il conteggio**
E2 : espressione logica la cui transizione da 0 ad 1 decrementa il conteggio
E3 : espressione logica di reset del conteggio (funziona sul valore del segnale)

Es. : `O2.1 = CNT2[I3.1, I3.2, I3.3] > 8;`

L'uscita n° 1 del modulo di uscita 2 si attiverà quando il contatore sarà maggiore al valore 8. Tale contatore verrà incrementato dagli impulsi rilevati sull'ingresso I3.1, decrementato dagli impulsi rilevati sull'ingresso I3.2 e potrà essere resettato da un impulso sull'ingresso I3.3.

Il comportamento dell'uscita è mostrato dal diagramma seguente dove si mostra l'andamento nel tempo dei segnali di ingresso e di uscita:



O2.1 = CNT2[I3.1, I3.2, I3.3] >8;

Tutti i campi dell'espressione devono sempre essere presenti; qualora non sia richiesto il funzionamento di un parametro, questo dovrà essere sostituito da una costante logica (ON o OFF).

Es. : `O2 = CNT2[I3.1, I3.2, OFF];`

Il modulo di uscita 2 configurerà le proprie uscite (se si tratta di un modulo di uscita digitale) o genererà un segnale analogico (se si tratta di un modulo di uscita analogico) in base al valore corrente del contatore. Tale contatore verrà modificato come nell'esempio precedente, ma non sarà mai resettato.

Anche in questo caso è possibile utilizzare più contatori in cascata.

2.3.7 - Funzione "programmatore settimanale"

In tutte le versioni del modulo Master esiste un orologio interno che permette l'uso della funzione "programmatore".
La funzione "programmatore" permette di attivare un'uscita in particolari momenti della giornata o della settimana.

Nel sistema PIC*net* la funzione "programmatore settimanale" è rappresentata dal simbolo "PRG1" secondo la seguente sintassi:

PRG1[Gx-Gy, hh:mm, hh:mm]

dove: Gx-Gy rappresenta l'intervallo di giorni in cui il comando è valido

I giorni della settimana sono rappresentati con le seguenti abbreviazioni:
LU = Lunedì
MA = Martedì
ME = Mercoledì
GI = Giovedì
VE = Venerdì
SA = Sabato
DO = Domenica

hh:mm rappresenta l'orario di attivazione e spegnimento in ore e minuti
Il valore deve essere compreso tra 00:00 e 23:59.

Tutti i campi dell'espressione devono sempre essere presenti.

Es. : `O2.1 = PRG1[LU-VE, 10:00, 11:00];`

L'uscita n° 1 del modulo di uscita 2 si attiverà tutti i giorni da Lunedì a Venerdì dalle 10 alle 11 di mattina.

Mediante un'opportuna impostazione dei campi è possibile attivare o disattivare il programmatore in modo permanente:

- **con orario inizio = orario fine, il programmatore è sempre abilitato (se entrambe i campi giorno sono diversi da zero)**
- **con almeno uno dei campi giorno a zero (gg inizio o gg fine) il programmatore è sempre disabiltato**

Es.:

```
Luce = PRG1[0-DO, 10:00, 10:00];       <- Sempre disabilitato
Luce = PRG1[0-DO, 10:00, 14:32];       <- Sempre disabilitato
Luce = PRG1[0-0,  10:00, 14:32];       <- Sempre disabilitato

Luce = PRG1[LU-DO, 10:00, 10:00];      <- Sempre abilitato
Luce = PRG1[LU-LU, 10:00, 10:00];      <- Sempre abilitato il Lunedì
```

2.3.8 - Funzione "programmatore utente"

In tutte le versioni del modulo Master dotate di display esiste la possibilità di definire fino a 128 programmazioni orarie utilizzando direttamente il display ed i tasti di configurazione del modulo stesso.

Tale possibilità è particolarmente utile in tutte le applicazioni in cui esiste la necessità di modificare frequentemente la programmazione di un evento (ad es. accensione delle luci notturne, comando impianto di irrigazione, ecc.) poiché tale modifica può essere effettuata direttamente dell'utente senza l'utilizzo di un personal computer e senza dover ricorrere ad una modifica del programma utente.

La funzione "programmatore utente" è funzionalmente identica alla funzione "programmatore settimanale" e permette di attivare un'uscita in particolari momenti della giornata o della settimana.

Nel sistema PIC*net* la funzione "programmatore utente" è rappresentata dal simbolo "UPRG" secondo la seguente sintassi:

UPRG[S1, Gx-Gy, hh:mm, hh:mm]

dove:

S1 : stringa (max 16 caratteri) che rappresenta il nome della programmazione utente; tale nome è quello che verrà mostrato sul display del modulo master nel menù di impostazione; i caratteri della stringa devono essere delimitati tra doppi apici.
Qualora il numero di caratteri sia inferiore a 16, la stringa verrà automaticamente centrata sul display.

Gx-Gy rappresenta l'intervallo di giorni in cui il comando è valido
I giorni della settimana sono rappresentati con le seguenti abbreviazioni:

LU	**=**	**Lunedì**
MA	**=**	**Martedì**
ME	**=**	**Mercoledì**
GI	**=**	**Giovedì**
VE	**=**	**Venerdì**
SA	**=**	**Sabato**
DO	**=**	**Domenica**
OFF	**=**	**Programmatore disabilitato (valore impostabile solo da interfaccia utente)**

hh:mm rappresenta l'orario di attivazione e spegnimento in ore e minuti
Il valore deve essere compreso tra 00:00 e 23:59.

Tutti i campi dell'espressione devono sempre essere presenti.

Es. : `O2.1 = UPRG[”Luce esterna”, VE-DO, 20:00, 23:00];`

L'uscita n° 1 del modulo di uscita 2 si attiverà tutti i giorni da Venerdì a Domenica dalle ore 20 alle ore 23.
Accedendo al menù di modifica delle programmazioni utente del modulo master, il programmatore sarà identificato dalla stringa specificata “Luce esterna”.

A differenza della funzione PRG1, i parametri indicati in fase di programmazione non sono fissi, ma possono essere modificati direttamente dall'utente utilizzando il tastierino ed il display del modulo master.

I valori indicati servono dunque solo per inizializzare il funzionamento del programmatore orario e vengono utilizzati dal programma solo se l'utente non ha effettuato modifiche di tali valori.
Qualora l'utente modifichi le impostazioni di un programmatore orario, tali modifiche vengono memorizzate dal master nella sua memoria interna e sono riutilizzate ogni volta che il programma utente viene eseguito.

Qualora il programma utente subisca delle modifiche e venga eseguito il comando di invio al master del nuovo programma, tutte le variazioni eventualmente fatte ai programmatori orari dall'utente vengono azzerate.

Nel menù di modifica delle programmazioni utente del modulo master, vengono visualizzati solo i programmatori orari effettivamente utilizzati tra quelli disponibili, ciascuno identificato con il proprio nome. Qualora nel programma utente non venga utilizzato alcun programmatore UPRG, il menù di modifica risulterà vuoto.
Tale menù, quando interrogato la prima volta, indicherà i valori di inizializzazione presenti nella definizione del programmatore UPRG, mentre le volte successive mostrerà i valori come modificati dall'utente.

Mediante un'opportuna impostazione dei campi è possibile attivare o disattivare il programmatore in modo permanente:

- **con orario inizio = orario fine, il programmatore è sempre abilitato (se entrambe i campi giorno sono diversi da zero)**
- **con almeno uno dei campi giorno impostato a OFF (gg inizio o gg fine) il programmatore è sempre disabiltato. La condizione OFF nel campo Gx o Gy (giorno Inizio/Fine) non puo essere specificata da programma ma solo da interfaccia utente nel menu "Programmatori Orari"**

Es.:

```
Luce_ext = UPRG["Luce esterna", VE-DO, 20:00, 23:00];  <- Funz. normale
Luce_ext = UPRG["Luce esterna", VE-VE, 20:00, 20:00];  <- Sempre abilitato il Venerdi
Luce_ext = UPRG["Luce esterna", LU-DO, 20:00, 20:00];  <- Sempre abilitato
```

2.3.9 - Funzione "programmatore intervallo"

In tutte le versioni del modulo Master esiste un orologio interno che permette l'uso della funzione "programmatore intervallo".
La funzione "programmatore intervallo" permette di attivare un'uscita in un particolare intervallo della giornata o della settimana.

Nel sistema PIC*net* la funzione "programmatore intervallo" è rappresentata dal simbolo "PRG2" secondo la seguente sintassi:

PRG2[Gx, hh:mm, Gy , hh:mm]

dove: Gx e Gy rappresentano il giorno di inizio e di fine dell'intervallo di attivazione dell'uscita

I giorni della settimana sono rappresentati con le seguenti abbreviazioni:

- **LU = Lunedì**
- **MA = Martedì**
- **ME = Mercoledì**
- **GI = Giovedì**
- **VE = Venerdì**
- **SA = Sabato**
- **DO = Domenica**

hh:mm rappresenta l'orario di attivazione e spegnimento in ore e minuti
Il valore deve essere compreso tra 00:00 e 23:59.

Tutti i campi dell'espressione devono sempre essere presenti.

Es. :
```
O2.1 = PRG2[MA, 10:00, VE, 11:00];
```

L'uscita n° 1 del modulo di uscita 2 si attiverà da Martedì alle 10:00 fino a Venerdì alle 11:00.

Mediante un'opportuna impostazione dei campi è possibile attivare o disattivare il programmatore in modo permanente:

- **con orario inizio = orario fine, il programmatore è sempre abilitato (se entrambe i campi giorno sono diversi da zero)**
- **con almeno uno dei campi giorno a zero (gg inizio o gg fine) il programmatore è sempre disabiltato**

Es.:
```
Pompa = PRG2[0, 10:00, MA, 12:00];     <- Sempre disabilitato
Pompa = PRG2[LU, 10:00, 0, 12:00];     <- Sempre disabilitato

Pompa = PRG2[LU, 10:00, MA, 12:00];    <- Funz. Normale
Pompa = PRG2[LU, 10:00, ME, 10:00];    <- Sempre abilitato
```

2.3.10 - Funzione "lettura orologio "

Nelle versioni del modulo Master dotate di orologio interno è possibile leggerne il valore in modo da effettuare controlli legati a opportune date, orari, ecc.

Nel sistema PIC*net* la funzione "lettura orologio" è rappresentata dal simbolo "RTC" secondo la seguente sintassi:

RTC(*campo*)

Dove *campo* indica l'informazione oraria che si vuole ottenere secondo le seguenti abbreviazioni:

HH	**=**	**ore (0-23)**
MM	**=**	**minuti (0-59)**
DD	**=**	**giorno della settimana (1-7)**
DM	**=**	**giorno del mese (1-31)**
MO	**=**	**mese dell'anno (1-12)**
YE	**=**	**anno (0-99)**

Per il giorno della settimana si assume che il Lunedì sia il primo giorno e la Domenica il settimo.
Tra parentesi è indicato il range di valori ammesso per ogni campo.

Es. :
```
O2.1 = RTC(DD) != 2 & RTC(DD) != 5 & RTC(HH) == 10;
```

L'uscita n° 1 del modulo di uscita 2 si attiverà tutti i giorni escluso il Martedì e il Venerdì dalle 10:00 alle 10:59.

2.3.11 - Funzione "isteresi"

La funzione "isteresi" permette di attivare un'uscita digitale in funzione del valore assunto da una variabile analogica. Tale funzione restituisce un valore ON se il valore della grandezza analogica risulta maggiore o uguale alla soglia superiore, OFF se minore o uguale alla soglia inferiore indicata.

Tale funzione risulta particolarmente utile in tutte le applicazioni di controllo temperatura o controllo di processo.

Nel sistema PIC*net* la funzione "isteresi" è rappresentata dal simbolo "IST" secondo la seguente sintassi:

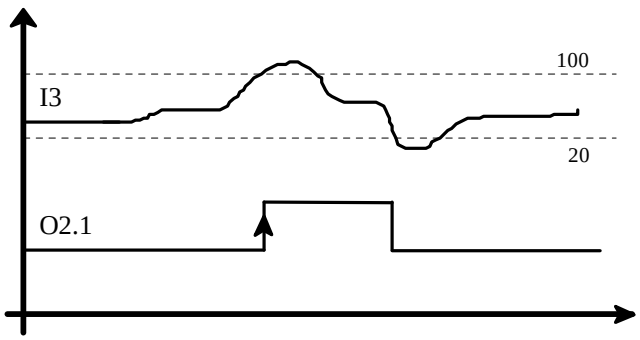
IST[E1, min, max]

dove: **E1 :** **grandezza di confronto**
min : **valore della soglia inferiore; il campo min può essere sia una costante che una variabile**
max : **valore della soglia superiore; il campo max può essere sia una costante che una variabile**

Es. : `O2.1 = IST[I3, 20, 100];`

L'uscita n° 1 del modulo di uscita 2 si attiverà quando il valore assunto dall'ingresso analogico n° 3 sarà uguale o maggiore a 100 e si resetterà quando il valore dell'ingresso sarà minore o pari al valore di 20.

Il comportamento dell'uscita è mostrato dal diagramma seguente dove si mostra l'andamento nel tempo dei segnali di ingresso e di uscita:



Come sempre l'operatore "not" (simbolo "!") posto prima della funzione IST permette di modificarne il funzionamento prima descritto utilizzando il fronte di discesa dei segnali anziché quello di salita per determinare la commutazione dell'uscita.

Tutti i campi dell'espressione devono sempre essere presenti; non è possibile, e non ha ovviamente alcun significato, indicare per la soglia inferiore un valore più grande della soglia superiore.

2.3.12 - Funzione "rampa"

La funzione "rampa" permette di generare dei valori crescenti o decrescenti di una grandezza per pilotare delle uscite analogiche o per effettuare dei controlli a soglia. Tale funzione restituisce un valore compreso tra i due estremi definiti.

La funzione rampa risulta particolarmente utile in tutte le applicazioni di dissolvenza luci o di controllo di processo.

Nel sistema PIC*net* la funzione "rampa" è rappresentata dal simbolo "RAMPA" secondo la seguente sintassi:

RAMPA[init, end, time, up, down]

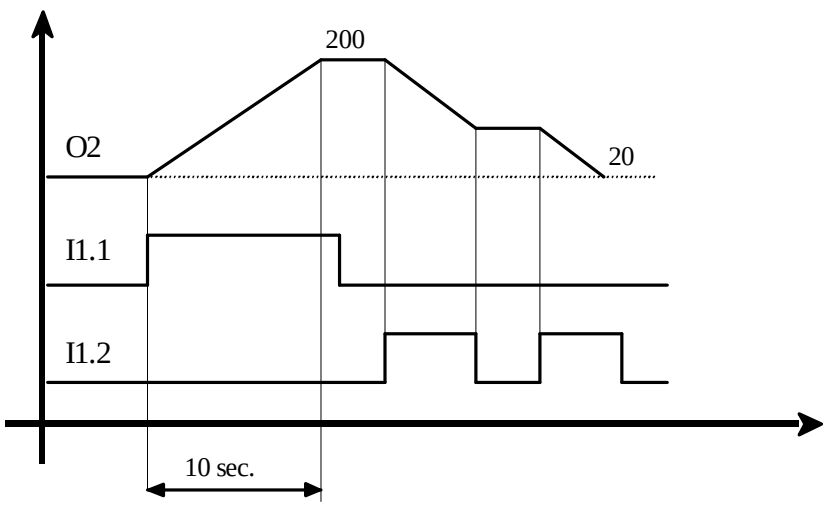
dove:
- **init :** valore di partenza della rampa
- **end :** valore finale della rampa
- **time :** tempo di escursione completa della rampa in decimi di secondo (max. 32767)
- **up :** espressione logica di abilitazione salita rampa (funziona sul valore del segnale)
- **down :** espressione logica di abilitazione discesa rampa (funziona sul valore del segnale)

Tutti i campi dell'espressione devono sempre essere presenti.

Es. : `O2 = RAMPA[20, 200, 100, I1.1, I1.2];`

L'uscita n° 2 assumerà un valore variabile da 20 a 200. Tale variazione verrà pilotata dagli ingressi I1.1 e I1.2 e l'intera escursione dell'uscita avverrà in 10 secondi.

Il comportamento dell'uscita è mostrato dal diagramma seguente dove si mostra l'andamento nel tempo dei segnali di ingresso e di uscita:



Come sempre l'operatore "not" (simbolo "!") posto prima delle espressioni up e down permette di modificarne il funzionamento prima descritto per determinare la variazione della rampa.
Qualora sia il comando *up* che il comando *down* siano attivi, la rampa non subisce modifiche.

E' possibile indicare per il campo "init" un valore maggiore di quello del campo "end". In tal caso i due campi *up* e *down* si scambiano di ruolo.

Esiste la possibilità di forzare il valore della rampa al valore iniziale o finale mediante due ulteriori condizioni logiche. In tal caso la sintassi completa è la seguente:

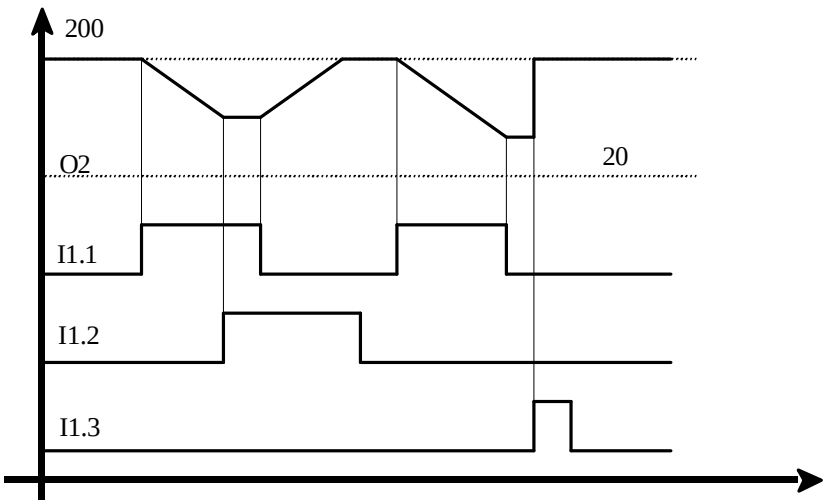
RAMPA[init, end, time, up, down, res1, res2]

dove:
- **init :** **valore di partenza della rampa**
- **end :** **valore finale della rampa**
- **time :** **tempo di escursione completa della rampa in decimi di secondo (max. 32767)**
- **up :** **espressione logica di abilitazione salita rampa (funziona sul valore del segnale)**
- **down :** **espressione logica di abilitazione discesa rampa (funziona sul valore del segnale)**
- **res1 :** **condizione logica di reset rampa al valore init (funziona sul valore del segnale)**
- **res2 :** **condizione logica di reset rampa al valore end (funziona sul valore del segnale) (se questo campo viene omesso si considera uguale a OFF)**

Es. :
```
O2 = RAMPA[200, 20, 100, I1.1, I1.2, I1.3, OFF];
```

L'uscita n° 2 assumerà un valore variabile da 20 a 200. Tale variazione verrà pilotata dagli ingressi I1.1 e I1.2 e l'intera escursione dell'uscita avverrà in 10 secondi. Un impulso sull'ingresso I1.3 provoca il reset della rampa al valore 200. Non è prevista la possibilità di provocare un reset della rampa a 20.

Il comportamento dell'uscita è mostrato dal diagramma seguente dove si mostra l'andamento nel tempo dei segnali di ingresso e di uscita:



2.3.13 - Funzione "dimmer"

Nei sistemi PIC*net* in cui sono installati dei moduli PNDIM o moduli di uscita analogica PNDA, esiste la possibilità di implementare le funzione tipiche di un dimmer.

Nel sistema PIC*net* la funzione "dimmer" è rappresentata dal simbolo "DIMMER" secondo la seguente sintassi:

Ox = DIMMER[up, down]

dove: Ox è l'indirizzo del generico modulo di uscita
up : condizione logica di aumento del valore dell'uscita
down : condizione logica di diminuzione del valore dell'uscita

Es. : O2 = **DIMMER[**I1.1, I1.2**]**;

Come in un normale dimmer l'uscita O2 viene alternativamente accesa o spenta da successive pressioni dei pulsanti connessi agli ingressi I1.1 e I1.2.

In caso di spegnimento, all'uscita viene assegnato il valore 0, mentre in caso di accensione l'uscita assume l'ultimo valore impostato.
La prolungata pressione di uno dei due pulsanti produce invece l'aumento (valore max. 255) o la diminuzione (valore min. 0) del valore dell'uscita O2.
Il tempo di attesa prima dell'inizio della dimmerazione è di defualt impostato a 1s. E' possibile modificare questo tempo mediante l'apposito registro del modulo master (vedi § 4.1).

L'intera escursione della regolazione dal valore minimo al valore massimo viene eseguita in circa 5 secondi (valore di default). E' possibile intervenire sulla velocità di escursione modificando l'apposito registro del modulo master (vedi § 4.1).

Nel caso di utilizzo di un modulo di uscita PNDIM, gli ingressi digitali presenti nel modulo stesso hanno le funzioni di "up" e "down" dei due canali dimmer solo se il modulo si trova in modalità di funzionamento locale (condizione di fabbrica o in caso di attivazione di un SETSLAVE per avaria bus).
Quando invece il modulo si trova connesso al bus PIC*net*, tali ingressi hanno il significato di generici ingressi digitali e non devono necessariamente essere associati ai campi "up" e "down" della funzione dimmer con cui si controlla il modulo PNDIM, il quale può dunque essere controllato da qualsiasi ingresso del sistema.

Come per qualsiasi altra funzione, i campi "up" e "down" possono essere rappresentati da un ingresso digitale (come nell'esempio) o da una qualsiasi condizione logica (combinazione di ingressi, altre funzioni, ecc.).

Qualora non si desideri regolare l'uscita da 0 al 100%, esiste la possibilità di indicare il valore minimo e massimo raggiungibili mediante due ulteriori campi della funzione DIMMER.
In tal caso la sintassi completa è la seguente:

Ox = DIMMER[lim_min, up, down]

oppure:

Ox = DIMMER[lim_min, lim_max, up, down]

dove: lim_min : valore minimo raggiungibile dalla regolazione di intensità (min. 0)
lim_max : valore massimo raggiungibile dalla regolazione di intensità (max. 255)

In alcune applicazioni (scenari luminosi, dissolvenze, ecc.) può essere utile poter forzare lo spegnimento o l'accensione al massimo valore mediante segnali di controllo diversi da quelli normalmente utilizzati per la regolazione luminosa.

Per risolvere il problema è possibile utilizzare la seguente sintassi estesa:

Ox = DIMMER[min, max, up, down, go_on, go_off]

dove: go_on : condizione logica che forza il valore max in uscita
go_off : condizione logica che forza lo spegnimento dell'uscita

Es. : O2 = DIMMER[10, 200, I1.1, I1.2, I2.1, I2.2];

Come in un normale dimmer l'uscita O2 viene alternativamente accesa o spenta da successive pressioni dei pulsanti connessi agli ingressi I1.1 e I1.2.
In caso di spegnimento, all'uscita viene assegnato il valore 0, mentre in caso di accensione l'uscita assume l'ultimo valore impostato.
Il valore dell'uscita può essere regolato tra un minimo di 10 ed un massimo di 200.

Attivando l'ingresso I2.1 è possibile in qualunque momento assegnare il valore 255 all'uscita O2.
Se l'ingresso I2.1 viene rimosso, l'uscita ritorna ad assumere il valore di regolazione precedente.

Attivando l'ingresso I2.2 è possibile in qualunque momento assegnare il valore 0 all'uscita O2.
Se l'ingresso I2.2 viene rimosso, l'uscita rimane spenta e solo l'attivazione di uno degli ingressi I1.1 o I1.2 fa assumere all'uscita il valore di regolazione precedente.

2.3.14 - Funzione "messaggio"

Nei sistemi PIC*net* in cui sono installati dei moduli di visualizzazione PNDIS (o con modulo master dotato di display), esiste la possibilità di definire dei messaggi da visualizzare al verificarsi di opportune condizioni.

Nel sistema PIC*net* la funzione "messaggio " è rappresentata dal simbolo "MESS" secondo la seguente sintassi:

Mx = MESS[S1, S2, V1, V2, C1]

dove: Mx è il numero del messaggio. Il numero x può assumere tutti i valori tra 1 e 255.
- **S1 : stringa (max 16 caratteri) che rappresenta la prima riga del messaggio su display; i caratteri della stringa devono essere delimitati tra doppi apici**
- **S2 : stringa (max 16 caratteri) che rappresenta la seconda riga del messaggio su display; i caratteri della stringa devono essere delimitati tra doppi apici**
- **V1 : flag che definisce il comportamento del relè presente sul modulo display; può assumere solo i valori ON o OFF**
- **V2 : flag che definisce il comportamento del cicalino presente sul modulo display; può assumere solo i valori ON o OFF**
- **C1 : espressione logica di consenso alla visualizzazione del messaggio**

Es. :
```
M2 = MESS["E' avvenuto", "un guasto", ON, OFF, I1.2];
```

Il messaggio 2 è del tipo (su due righe) "E' avvenuto un guasto" se I1.2 diviene attivo; in concomitanza viene attivato il relè mentre il cicalino non viene attivato.

Tutti i campi dell'espressione devono sempre essere presenti.

La funzione "messaggio" può essere utilizzata anche per la visualizzazione sul display del modulo master o dei moduli PNDIS del valore di variabili interne del programma (ingressi, uscite, variabili virtuali e variabili utente) semplicemente inserendo il nome della variabile da visualizzare all'interno della stringa S1 o S2. Il nome della variabile deve essere preceduto e seguito dal simbolo %.
Per ogni stringa del messaggio è possibile inserire un massimo di due variabili da visualizzare.

Ad esempio per visualizzare il numero di auto presenti in un posteggio, posto che I3.1 e I3.2 siano gli ingressi collegati ai sensori di ingresso ed uscita del posteggio (fotocellula o altro) si potrà scrivere:

```
V1 = CNT2[I3.1, I3.2, OFF];
M4 = MESS["Nel posteggio", "ci sono %V1% auto", OFF, OFF, PWM[10,40,ON]];
```

La variabile utente V1 contiene il valore delle auto presenti e non viene mai resettata. Il messaggio M4 mostra il valore di V1 sul display. Tale messaggio è continuamente rinfrescato ogni 5 secondi (attivato per 1 secondo con 4 secondi di pausa) mentre sia il cicalino che il relè di allarme sono disabilitati.

Un messaggio viene visualizzato quando si verifica la condizione logica di attivazione.
Una volta visualizzato, il messaggio permane sul display e verrà cancellato solo dall'attivazione di un nuovo messaggio o allarme o dalla pressione dei pulsanti posti sul frontale del modulo per la consultazione del menù utente.

2.3.15 - Funzione "variabili utente"

In tutte la versioni del modulo Master dotate di display esiste la possibilità di impostare il valore di grandezze utilizzabili all'interno del programma utilizzando direttamente il display ed i tasti di configurazione del modulo master.

Tale possibilità è particolarmente utile in tutte le applicazioni in cui esiste la necessità di modificare frequentemente il valore di un parametro (ad es. il livello di un'uscita analogica, la temporizzazione di un'accensione, ecc.) poiché tale modifica può essere effettuata direttamente dell'utente senza l'utilizzo di un personal computer e senza dover ricorrere ad una modifica del programma utente.

Nel sistema PIC*net* la funzione "variabili utente" è rappresentata dal simbolo "UVAR" secondo la seguente sintassi:

UVAR[S1, lim_min, lim_max, step]

dove:

S1 : stringa (max 16 caratteri) che rappresenta il nome della variabile utente; tale nome è quello che verrà mostrato sul display del modulo master nel menù di impostazione; i caratteri della stringa devono essere delimitati tra doppi apici.
Qualora il numero di caratteri sia inferiore a 16, la stringa verrà automaticamente centrata sul display.

lim_min : valore minimo raggiungibile dalla variabile (min. -32767)
lim_max : valore massimo raggiungibile dalla variabile (max. 32767)
step : valore con cui la variabile verrà incrementata o decrementata

Tutti i campi dell'espressione devono sempre essere presenti.

Es. : `O2 = UVAR[”Livello ballast”, 0, 200, 10];`

L'uscita del modulo di uscita analogica n. 2 assumerà il valore impostato dall'utente.
Tale valore potrà essere variato tra il valore minimo di 0 ed il valore massimo di 200 con gradini di 10 agendo sui pulsanti UP e DOWN presenti sull'interfaccia del modulo master.
Accedendo al menù di modifica delle variabili utente del modulo master, la variabile sarà identificata dalla stringa specificata “Livello ballast”.

All'avviamento del programma tutte le variabili così definite assumono il valore lim_min (valore di default).
E' possibile definire un diverso valore iniziale della variabile aggiungendo un campo opzionale secondo la seguente sintassi completa:

UVAR[S1, lim_min, lim_max, step, init]

dove:

init : valore iniziale della variabile (compreso tra lim_min e lim_max)

L'impostazione del valore iniziale viene utilizzata fino a quando l'utente non modifica il valore della variabile. Da quel momento ad ogni accensione o ripartenza del programma, la variabile assumerà l'ultimo valore impostato.

2.4 - Programmazione: variabili d'ambiente

Esiste la possibilità di definire all'interno del programma utente alcune variabili ("variabili d'ambiente") per impostare il funzionamento complessivo del sistema.
Le definizioni possono essere introdotte in ordine qualsiasi. Qualora non sia stata prevista la definizione di una variabile d'ambiente, il sistema attribuirà ad essa un valore predefinito ("valore di default").

2.4.1 - Impostazione Orologio

Esiste la possibilità (normalmente utilizzata in fase di debug del programma) di definire data ed ora da memorizzare nell'orologio interno (se presente).

La sintassi utilizzata è la seguente:

SETCLOCK = gg/mm/aa-hh:mm:ss;

con la quale si definiscono giorno, mese, anno, ore minuti e secondi per l'orologio del sistema.
Tale definizione verrà eseguita ogni volta che il programma utente viene fatto ripartire.

In mancanza di questa definizione, l'orologio mantiene la data e l'ora correnti.

2.4.2 - Impostazione Lingua

Esiste la possibilità di definire la lingua utilizzata per le comunicazioni del display del master e dei moduli display (se presenti). Tale definizione sostituisce quella impostata mediante il tastierino di configurazione presente sui moduli PNMAS e PNDIS.

SETCOUNTRY = code;

dove code = Codice paese (ANSI standard):
039 = Italia
001 = U.S.A.
033 = Francia
.....

In mancanza di questa definizione, viene mantenuto il valore corrente. Il valore di default è 039.

2.4.3 - Impostazione Velocità di Comunicazione con PC

Esiste la possibilità di definire la velocità di trasmissione della linea seriale RS232 di servizio del modulo master.

La sintassi utilizzata è la seguente:

SETBAUDPC = baudrate;

dove baudrate rappresenta una velocità di trasmissione accettabile (2.400, 4.800, 9.600, 19.200, 38.400).

In mancanza di questa definizione, la velocità di trasmissione è impostata a 38.400.

2.4.4 - Impostazione Velocità di Comunicazione Bus

Esiste la possibilità di definire la velocità di trasmissione sul bus tra il modulo master ed i moduli slave.

La sintassi utilizzata è la seguente:

SETBAUDBUS = baudrate;

dove baudrate rappresenta una velocità di trasmissione accettabile (5.000, 9.600, 18.000, 25.000, 41.000, 62.500).

In mancanza di questa definizione, la velocità di trasmissione è impostata a 9.600.

2.4.5 - Impostazione segnalazione allarmi

Esiste la possibilità di inibire il funzionamento del relè di scambio e dell'avvisatore acustico presenti sui moduli master PNMAS e di visualizzazione PNDIS al verificarsi di un allarme utente (vedi oltre par. 3.1). Tale impostazione prevale su quella indicata nella singola definizione dell'allarme.

La sintassi utilizzata è la seguente:

SETRELE = OFF;
per forzare la disattivazione del relè di allarme

SETBUZ = OFF;
per forzare lo spegnimento dell'avvisatore acustico.

In mancanza di queste definizioni, le definizioni assunte per default sono:

SETRELE = ON;
SETBUZ = ON;

che abilitano il funzionamento dei due dispositivi.

2.4.6 - Inizializzazione moduli

In alcuni moduli slave esiste la possibilità di configurare via software alcuni parametri di funzionamento (ad es. il valore delle uscite dei moduli PN8O e PN8OR). Tale impostazione avviene in fase di inizializzazione del sistema sulla base di una serie di definizioni utente.

La sintassi utilizzata è la seguente:

SETSLAVE(*addr*) = *const*;

dove *addr* è l'indirizzo del modulo e *const* è la costante di inizializzazione da inviare.
La costante specificata viene inviata dal master al modulo slave interessato all'accensione del sistema e viene da questo registrata nella sua memoria non volatile interna.

Inizializzazione Uscite

Nel caso di moduli slave di uscita (PN8O, PN8OR) o di moduli misti di ingresso-uscita (PN4I4O, PN2I2O, PN4I4OW, PN2I2OW), questa inizializzazione serve per indicare al modulo la configurazione delle uscite da assumere in caso di avaria di bus.

Infatti all'accensione del sistema il master provvede ad inizializzare le uscite a zero o ad un valore definito dall'utente con il comando INIT (vedi oltre par. 3.7) o al valore che possedevano prima dello spegnimento del sistema a seconda dell'impostazione del flag di sistema RAMBACKUP (vedi oltre par. 2.4.7).
Successivamente le uscite assumono la configurazione richiesta dal programma in esecuzione.
Se però un modulo non rileva più alcuna attività di bus (bus interrotto o avaria master) per un certo periodo, commuta le sue uscite al valore impostato con il comando SETSLAVE.

In tal modo è possibile gestire lo stato di particolari utenze in condizione di avaria del sistema.
Ad es.

```
SETSLAVE(02) = 10011111b;
```

indica che il modulo di uscita n°2 in caso di avaria bus dovrà commutare le sue uscite secondo il pattern indicato. Il bit più significativo (quello più a sinistra) si riferisce all'uscita n°8, il meno significativo (quello più a destra) alla n°1.

Occorre sempre indicare tutti gli 8 bit di configurazione anche nel caso di moduli con un numero inferiore di uscite (ad es. PN2I2O o PN4I4O). In tal caso i bit non utilizzati dovranno essere posti a 0.

Questa scrittura è equivalente alla seguente espressione (essendo 159 il numero decimale equivalente al numero binario 10011111b):

```
SETSLAVE(02) = 159;
```

Le uscite per le quali non viene specificata una configurazione, in caso di avaria bus mantengono l'ultimo valore assunto.

Nei moduli misti dotati sia di ingressi che di uscite a relè (PN2I2O – PN2I2OW – PN2I2OWP – PN4I4O – PN4I4OW – PN8I8O) esiste la possibilità di attivare una modalità locale in caso di avaria bus.
In tale modalità di funzionamento locale, ciascuna uscita dipende dal corrispondente ingresso con comando passo-passo o con comando mantenuto. In tal modo risulta sempre garantito il comando locale ad es. di un punto luce o la movimentazione di una tapparella anche in caso di avaria del sistema.

La modalità di funzionamento locale può essere attivata mediante la direttiva SETPAR (vedi § 3.14) secondo le seguenti corrispondenze:

- **indirizzo attivazione: 25**
- **costante disattivazione modalità locale: 255**
- **costante attivazione modalità comando mantenuto: 0**
- **costante attivazione modalità passo-passo: 1**

Ad es. per attivare il comando locale con comando mantenuto del modulo con indirizzo 100 si scriverà:

```
//-------------------------------------------------------------
//      Programma esempio di gestione
//      modulo PN2I2O con funzionamento in locale
//-------------------------------------------------------------
DEFINE     SLocMode   25       // indirizzo modalità locale
DEFINE     SLM_dis    255      // valore per disabilitazione
DEFINE     SLM_direct 0        // valore per abilitazione comando mantenuto
DEFINE     SLM_toggle 1        // valore per abilitazione comando passo-passo

SETPAR[100, SLocMode, SLM_direct];     // Imposto SLM = DIRETTO
```

In questo esempio, in caso di avaria del bus le uscite non subiranno nessuna modifica, ma verrà attivata la possibilità da parte dell'utente di attivare le uscite del modulo in modalità comando mantenuto agendo sul corrispondente ingresso (l'ingresso 1 comanderà l'uscita 1, l'ingresso 2 l'uscita 2, ecc.).

La modalità di funzionamento locale può convivere con il comando SETSLAVE essendo due funzionalità indipendenti che possono dunque essere anche attivate contemporaneamente.

Riprendendo l'esempio di prima:

```
//-------------------------------------------------------------
//      Programma esempio di gestione
//      modulo PN2I2O con funzionamento in locale
//-------------------------------------------------------------
DEFINE     SLocMode   25       // indirizzo modalità locale
DEFINE     SLM_dis    255      // valore per disabilitazione
DEFINE     SLM_direct 0        // valore per abilitazione comando mantenuto
DEFINE     SLM_toggle 1        // valore per abilitazione comando passo-passo

SETSLAVE(O100) = 00000011b;    // uscite tutte a ON (occorre indicare tutti i bit)
SETPAR[100, SLocMode, SLM_toggle];// Imposto SLM = TOGGLE
```

In questo esempio, in caso di avaria del bus le uscite commuteranno tutte sullo stato ON e verrà attivata la possibilità da parte dell'utente di attivare/disattivare le uscite del modulo in modalità passo-passo agendo sull' ingresso corrispondente.

E' possibile definire il tempo di inattività di bus (bus interrotto o avaria master) oltre il quale il modulo commuta le sue uscite al valore impostato con il comando SETSLAVE e/o attiva la funzionalità locale.

Tale definizione è effettuata con la seguente sintassi:

TIMESLAVE(*const*) ;

dove *const* è una costante (compresa tra 2 e 255) che indica il tempo di attesa in unità interne di programma.
In mancanza di tale indicazione il valore di default è 10 pari ad un'attesa di circa 20 secondi.

Ad ogni avviamento di programma il modulo master provvede ad inviare a tutti i moduli slave un comando di reset della funzione SETSLAVE che viene dunque disattivata, salvo poi ripristinarla se così previsto nel programma utente.
In tal modo è possibile disabilitare i SETSLAVE precedentemente memorizzati nei moduli slave semplicemente scaricando un programma che non preveda tale funzione.

Inizializzazione Ingressi

Nel caso di moduli slave di ingresso (PN8I, PN8IW) o di moduli misti di ingresso-uscita (PN4I4O, PN2I2O), questa inizializzazione serve per indicare al master la modalità di funzionamento da assumere in caso di guasto del modulo di ingresso coinvolto.

Infatti in caso di avaria di un modulo di ingresso, il master continua l'esecuzione del programma utente e l'aggiornamento delle uscite, ma non ha più informazioni valide sul valore degli ingressi collegati al modulo guasto.

A seconda del valore impostato con il comando SETSLAVE è possibile indicare comportamenti diversi:

a) SETSLAVE(*addr*) = *const*;

in questo caso si assume che in caso di avaria del modulo *addr* di ingresso, il valore letto sia *const*.

Ad es.

```
SETSLAVE(I2) = 10011111b;
```

indica che in caso di avaria del modulo di ingresso n°2, ai suoi ingressi viene assegnato il pattern indicato (il bit più significativo si riferisce all'uscita n°8, il meno significativo alla n°1).

Questa scrittura è equivalente alla seguente espressione:

```
SETSLAVE(I2) = 159;
```

Occorre sempre indicare tutti gli 8 bit di configurazione anche nel caso di moduli con un numero inferiore di ingressi (ad es. PN2I2O o PN4I4O). In tal caso i bit non utilizzati dovranno essere posti a 0.

b) SETSLAVE(*addr*) = LAST;

in caso di avaria del modulo *addr* di ingresso, il programma mantiene l'ultimo valore letto.

Ad es. :

SETSLAVE(I4) = LAST;

indica che in caso di avaria del modulo di ingresso n°4, il programma continua ad elaborare le equazioni utilizzando l'ultimo valore ricevuto per gli ingressi del modulo guasto.

c) SETSLAVE(*addr*) = STOP;

in caso di avaria del modulo *addr* di ingresso, il programma non esegue più le equazioni che contengono al loro interno ingressi del modulo guasto. Le uscite relative a tali equazioni non vengono dunque più modificate.

Ad es.

SETSLAVE(I4) = STOP;

indica che in caso di avaria del modulo di ingresso n°4, il programma non elabora più le equazioni che contengono gli ingressi I4.1, I4.2, I4.3, ecc.

In mancanza di queste definizioni, si assume per tutti i moduli di ingresso il valore STOP.

Nota: nella gestione di allarmi e messaggi, l'impostazione STOP per un modulo di ingresso viene gestita in modo identico all'impostazione LAST, per evitare la scomparsa di una segnalazione in caso di avaria.

2.4.7 - Reset moduli di uscita

Al momento del reset del sistema, tutte le uscite vengono automaticamente forzate al valore 0 (cui corrisponderà un'uscita chiusa o aperta a seconda del tipo di contatto NA o NC gestito).
In alcuni casi tale procedura può generare degli inconvenienti per l'utenza: se ad es. l'impianto è utilizzato per la gestione di un impianto di illuminazione e l'alimentazione del sistema bus viene momentaneamente a mancare per un black-out, al ritorno dell'alimentazione il sistema procederà al reset delle uscite provocando così lo spegnimento forzato di tutte le luci (e di altri apparecchi precedentemente accesi).
In tutte le versioni del modulo Master esiste la possibilità di evitare tali inconvenienti mediante la direttiva RAMBACKUP.

La sintassi utilizzata è la seguente:

RAMBACKUP;

Tale direttiva consente di inizializzare le uscite al valore che avevano prima dell'ultimo spegnimento del sistema e che viene memorizzato dal modulo master in una memoria dotata di batteria di back-up.
Il valore impostato al reset dalla direttiva RAMBACKUP ha una priorità inferiore all'eventuale valore di inizializzazione impostato con il comando INIT.

Qualora in presenza della direttiva RAMBACKUP si desideri resettare lo stato dell'impianto, nelle versioni di master superiori alla rev. 16.00 è possibile effettuare tale manovra accendendo l'unità centrale tenendo il pulsante ESC premuto.

3.0 - Programmazione avanzata

3.1 - Impostazione allarmi utente

Nel sistema PIC*net* esiste la possibilità di definire alcuni allarmi.
L'utente ha a disposizione 220 allarmi codificati tra 1 e 220. I codici tra 221 e 255 sono riservati per segnalazioni di allarme di sistema.

Nel sistema PIC*net* un allarme utente viene definito mediante il simbolo "ALM" secondo la seguente sintassi:

Ax = ALM[S1, S2, V1, V2, C1]

dove: **Ax è il numero dell'allarme. Il numero x può assumere tutti i valori tra 1 e 220.**
- **S1 : stringa (max 16 caratteri) che rappresenta la prima riga del messaggio associato all'allarme; i caratteri della stringa devono essere delimitati tra doppi apici**
- **S2 : stringa (max 16 caratteri) che rappresenta la seconda riga del messaggio associato all'allarme; i caratteri della stringa devono essere delimitati tra doppi apici**
- **V1 : flag che definisce il comportamento del relè presente sul modulo display; può assumere solo i valori ON o OFF**
- **V2 : flag che definisce il comportamento del cicalino presente sul modulo display; può assumere solo i valori ON o OFF**
- **C1 : espressione logica di visualizzazione dell'allarme**

Es. :
```
A80 = ALM["Guasto", "alla pompa", ON, OFF, I1.2];
```

L'allarme 80 verrà attivato se I1.2 diviene attivo; ad esso vengono associate le stringhe "Guasto" e "alla pompa". Al verificarsi della condizione di allarme verrà attivato il relè, ma non il cicalino.

Tutti i campi dell'espressione devono sempre essere presenti.

In mancanza di queste definizioni gli allarmi utente non esistono (situazione di default).
Gli allarmi utente, se presenti, vengono sempre memorizzati nello storico allarmi.
Il display del modulo master, se presente, visualizza sempre tutti gli allarmi attivi (utente e di sistema).
Un allarme rimane visualizzato solo se attivo ossia solo finché è verificata la condizione logica di attivazione.

Poiché un allarme può assumere solo i due valori logici ON e OFF, lo stato degli allarmi può essere utilizzato all'interno di espressioni logiche.

Es. :
```
O2.2 = A80 | I1.1;
```

L'uscita n° 2 del modulo di uscita 2 si attiverà al verificarsi dell'allarme utente n°80 o in corrispondenza di uno stato attivo sull'ingresso n°1 del modulo di ingresso 1.

3.2 - Allarmi di sistema

Esistono alcuni allarmi di sistema predefiniti ed associati ai codici compresi tra 221 e 255 (codici riservati).
Gli allarmi di sistema predefiniti equivalgono alle seguenti definizioni:

```
A221 = ALM["Conflitto bus", "", ON, ON, X];
A222 = ALM["Bad checksum", "", ON, ON, X];
A223 = ALM["Memory fail", "", ON, ON, X];
A224 = ALM["Modulo x non funzionante ", "", ON, ON, X];
A225 = ALM["Bus scollegato ", "", ON, ON, X];
```

dove con X viene indicata una condizione di sistema.
Gli allarmi di sistema vengono sempre memorizzati nello storico allarmi.

Anche gli allarmi di sistema, come gli allarmi utente, possono essere utilizzati all'interno di equazioni per attivare ad es. una segnalazione remota o per prendere determinate azioni al loro verificarsi.

3.3 – Impostazione "display"

Nei sistemi PIC*net* in cui sono installati dei moduli di visualizzazione PNDIS, esiste la possibilità di definire i messaggi e gli allarmi da visualizzare su ciascun modulo display presente.

Nel sistema PIC*net* il display viene programmato secondo la seguente sintassi:

Dx = *lista*

dove: **x rappresenta l'indirizzo del modulo display**
***lista* è l'elenco dei messaggi e degli allarmi visualizzabili separati da una virgola**

Il display del modulo master, se presente, è indirizzato con il valore 254.

Es. :
```
D2 = M1, A100, M22;
```

Sul modulo display n°2 verranno visualizzati solo i messaggi M1 e M22 o l'allarme A100 qualora vengano attivati.

Tutti i campi dell'espressione devono sempre essere presenti.
In mancanza di una definizione, un modulo display non visualizza nessun messaggio o allarme.
Inoltre non è possibile all'interno di un programma definire più volte uno stesso modulo display.

3.4 - Variabili virtuali

Esiste la possibilità di definire delle variabili utente da utilizzare all'interno delle equazioni. Tale utilizzo risulta comodo in tutti quei casi in cui un certo tipo di relazione tra gli ingressi ricorre frequentemente in più equazioni all'interno di uno stesso programma. In questo caso il risultato di tale relazione può essere depositato in una variabile di comodo detta appunto variabile virtuale.

La sintassi utilizzata per la definizione di una variabile virtuale è identica a quella delle normali equazioni dove al posto del simbolo di uscita si ponga quello di variabile.

Tutte le variabili virtuali sono variabili a 16 bit e possono essere utilizzate sia come variabili logiche che come variabili intere.
Nel primo caso (variabili logiche) sono indicate con il simbolo Vx.y dove con x si intende il numero della variabile e con y il bit all'interno della variabile stessa.
Il numero x può assumere tutti i valori tra 1 e 256 (2500 nelle versioni di master superiori alla rev. 16.00), mentre y sarà compreso tra 1 e 16.

Nel secondo caso (variabili intere) sono indicate con il simbolo Vx dove con x si intende il numero della variabile.
Il numero x può assumere tutti i valori tra 1 e 256 (2500 nelle versioni di master superiori alla rev. 16.00).

Ogni definizione di variabile virtuale è dunque equivalente all'assegnazione di una variabile di uscita (digitale o analogica) ed è descritta da un'espressione del tipo:

Vx.y = f(In.m);
o Vx = f(In.m);

essendo f(In.m) una generica funzione di ingressi del sistema.

Le variabili così definite possono essere liberamente utilizzate come termini generici all'interno delle equazioni semplificando la scrittura del programma e agevolando le sue modifiche successive.

Esempio 1. Si voglia comandare l'accensione di quattro lampade collegate alle uscite del modulo n°1 in funzione dello stato di un interruttore crepuscolare (connesso all'ingresso 1 del modulo 2) subordinato ad un consenso orario.
Il programma relativo sarà ad es.:

```
O1.1 = I2.1 & PRG1[Lu-Do, 18:00, 23:00];
O1.2 = I2.1 & PRG1[Lu-Do, 18:00, 23:00];
O1.3 = I2.1 & PRG1[Lu-Do, 18:00, 23:00];
O1.4 = I2.1 & PRG1[Lu-Do, 18:00, 23:00];
```

Utilizzando le variabili virtuali, lo stesso programma può essere così modificato:

```
V1.1 = I2.1 & PRG1[Lu-Do, 18:00, 23:00];
O1.1 = V1.1;
O1.2 = V1.1;
O1.3 = V1.1;
O1.4 = V1.1;
```

Oltre che per la maggior chiarezza e facilità di modifica, la seconda forma è certamente da preferire perché il programma relativo risulta eseguito dal modulo master in un tempo sensibilmente inferiore, riducendo i tempi di risposta del sistema.

Nel primo caso infatti l'unità di calcolo deve elaborare quattro programmazioni orarie e 4 operatori & per ogni ciclo di calcolo, mentre nel secondo esempio tanto la programmazione oraria quanto l'operatore & vengono eseguiti una sola volta ogni ciclo di calcolo con un notevole incremento delle prestazioni globali del sistema.

Esempio 2. Si voglia comandare l'accensione di due pompe connesse alle uscite del modulo n°210 al superamento di certi valori di temperatura rilevata da una termosonda collegata al modulo di ingresso analogico n°220.

Il programma relativo sarà ad es.:

```
O210.1 = (I220 / 4+20) >= 100;
O210.2 = (I220 / 4+20) >= 150;
```

Utilizzando le variabili virtuali, lo stesso programma può essere così modificato:

```
V11 =    (I220 / 4+20);
O210.1 = V11 >= 100;
O210.2 = V11 >= 150;
```

Anche in questo caso la seconda forma è da preferire perché il programma relativo risulta eseguito dal modulo master in un tempo inferiore, oltre che per la maggior chiarezza e facilità di modifica.

3.5 – Costanti utente

Per migliorare la lettura e la documentazione di un programma utente, è possibile utilizzare nelle equazioni delle stringhe di caratteri o costanti al posto delle espressioni richieste dalla sintassi del sistema.
A tal scopo occorre utilizzare la direttiva DEFINE secondo la seguente sintassi:

DEFINE *str1 str2*

dove: ***str1* è la stringa di comodo definita dall'utente (lunghezza minima 4 caratteri)**
***str2* è la stringa originaria da sostituire**

Es. :

```
//----------------------------
//  Comando luce esterna
//----------------------------
//-------------------------
//  Definizioni utente
//-------------------------
DEFINE START          16:30
DEFINE STOP           23:00
DEFINE LuceExt        O1.1
DEFINE Crepuscolare   I1.1
DEFINE Manuale        T[I1.2]
DEFINE Timer          TMR[0, 100, I1.3, OFF]
DEFINE Orologio       V1.1
//-------------------------
// Programma
//-------------------------
Orologio = PRG1[Lu-Ve, START, STOP];
LuceExt = Timer | Manuale | (Crepuscolare & Orologio);
```

In questo caso il sistema piloterà un'illuminazione esterna (collegata all'uscita O1.1) se riceverà un comando manuale passo-passo da un pulsante (connesso all'ingresso I1.2) o se verrà attivata una temporizzazione di 10 secondi mediante un pulsante (collegato all'ingresso I1.3) o se riceverà un consenso da un interruttore crepuscolare (collegato all'ingresso I1.1) subordinato ad una programmazione oraria dalle ore 16.30 alle 23 di tutti i giorni dal Lunedì al Venerdì.

Lo stesso programma può essere scritto utilizzando la sintassi tradizionale nel seguente modo:

```
V1.1 = PRG1[Lu-Ve, 16:30, 23:00];
O1.1 = TMR[0, 100, I1.3, OFF]  | T[I1.2] | (I1.1 & V1.1);
```

***NOTA*: La direttiva DEFINE non può essere utilizzata per sostituire delle stringhe all'interno di scritte definite dall'utente per la visualizzazione di allarmi o messaggi.** Ad es.:

```
DEFINE       Conta     V1
Conta = CNT2[I3.1, I3.2, OFF];
M4 = MESS["Nel posteggio", "ci sono %Conta% auto", OFF, OFF, PWM[50,50,ON]];  // errato
M5 = MESS["Nel posteggio", "ci sono %V1% auto", OFF, OFF, PWM[50,50,ON]];     // corretto
```

3.6 - Variabili utente

In aggiunta alle variabili virtuali già definite al par. 3.4, esiste la possibilità di definire delle variabili utente generiche da utilizzare all'interno delle equazioni. Questa definizione di variabili è di natura più generale in quanto tali variabili possono essere di vario tipo e non esistono vincoli sul nome associato alla variabile.

Le variabili utente sono identificate da un nome e devono essere dichiarate prima dell'uso secondo una delle seguenti sintassi:

INTEGER : *nome*	**(-32767 ÷ +32768)**
FLOAT : *nome*	**($-3*10^{38}$ ÷ $+3*10^{38}$)**
BOOL : *nome*	**(ON, OFF)**

dove: ***nome* è una stringa che definisce il nome della variabile utente.**
Sono nomi validi quelli composti esclusivamente da lettere e numeri.

Tra parentesi è indicato il range di valori ammesso per ogni tipo di variabile

Es.:

```
INTEGER :  persone
FLOAT :    temperatura
BOOL :     LuceEsterna
```

Le variabili di tipo BOOL rappresentano un'estensione delle variabili virtuali. Infatti la definizione di una variabile BOOL equivale a quella di una variabile virtuale logica (ad es. V4.2).
La definizione di una variabile INTEGER differisce invece da quella di una variabile virtuale analogica (ad es. V2) in quanto dotata di segno (può assumere sia valori positivi che negativi).

Le variabili così definite possono essere liberamente utilizzate come termini generici all'interno delle equazioni semplificando la scrittura del programma e agevolando le sue modifiche successive.
In ogni programma utente è possibile definire fino ad un massimo di 128 variabili per ogni tipo (512 nelle versioni di master superiori alla rev. 16.00).

***NOTA*: Le variabili utente così definite non possono avere nomi che compaiono per intero o in parte in una dichiarazione DEFINE.**

Ad es. non è possibile una scrittura del tipo:

```
DEFINE   Luce    V1.1
BOOL :   LuceEsterna
```

mentre la seguente scrittura è corretta:

```
DEFINE   Luce2   V1.1
BOOL :   LuceEsterna
```

3.7 – Inizializzazione variabili

Esiste la possibilità di impostare il valore che una variabile deve assumere alla partenza del programma. Tale impostazione avviene in fase di inizializzazione del sistema sulla base di una serie di definizioni utente.

La sintassi utilizzata è la seguente:

INIT *var* = *const*;

dove *var* è il nome della variabile interessata e *const* è la costante di inizializzazione da assegnare.

Anche se la direttiva INIT viene normalmente utilizzata con le variabili utente, tutte le variabili possono essere inizializzate (uscite, variabili utente, variabili virtuali) ad eccezione degli ingressi.

Ad es. sono inizializzazioni valide le seguenti:

```
FLOAT : pippo

INIT O2 = 255;
INIT V1.4 = ON;
INIT pippo = 12.5;
```

La costante specificata viene assegnata alla variabile all'accensione del sistema; successivamente la variabile assumerà il valore assegnato dal programma in esecuzione.
Nel caso di una variabile di uscita, l'inizializzazione con il comando INIT ha la priorità rispetto al valore assegnato mediante la direttiva RAMBACKUP (valore che la variabile possedeva prima dello spegnimento del sistema).
In ogni programma utente è possibile definire fino ad un massimo di 256 inizializzazioni, indipendentemente dal tipo della variabile interessata.

3.8 – Conversioni di tipo

In generale non è possibile eseguire operazioni matematiche tra variabili di tipo diverso.
Qualora in una stessa espressione vengano utilizzate insieme variabili di tipo INTEGER e FLOAT, occorre effettuare una conversione di tipo in modo da rendere le variabili interessate dall'operazione omogenee tra loro.

La sintassi utilizzata per trasformare in FLOAT una grandezza INTEGER è la seguente:

ITOF[*espr*]

dove *espr* è una grandezza o variabile di tipo intero.

La sintassi utilizzata per trasformare in INTEGER una grandezza FLOAT è la seguente:

FTOI[*espr*]

dove *espr* è una grandezza o variabile di tipo float. Durante la conversione la grandezza float viene arrotondata all'intero più vicino.

3.9 – Controllo del programma

Utilizzando la normale sintassi delle funzioni di programmazione non è possibile alterare il flusso del programma in esecuzione, dunque tutte le espressioni del programma sono sempre eseguite ad ogni iterazione.
Questo costringe a rendere complessa la scrittura delle espressioni in tutte le situazioni in cui è richiesto che una parte dell'impianto funzioni solo in una particolare condizione.

Ad es. si desideri comandare l'accensione di una luce (collegata all'uscita O1.1) in manuale mediante un pulsante passo-passo (connesso all'ingresso I100.2) oppure in automatico su consenso di un interruttore crepuscolare (collegato all'ingresso I2.2).
La commutazione dal funzionamento manuale a quello automatico sarà effettuata mediante un interruttore collegato all'ingresso I10.1.

Il programma può essere scritto utilizzando la sintassi tradizionale nel seguente modo:

```
O1.1 = (T[I100.2] & I10.1) | (I2.2 & !I10.1);
```

Esiste la possibilità di rendere il programma più efficiente utilizzando il costrutto di controllo secondo la seguente sintassi:

```
IF [condizione]
        .........
        equazione
        equazione
        .........
ENDIF
```

oppure

```
IF [condizione]

        .........
        equazione
        equazione
        .........
ELSE

        .........
        equazione
        equazione
        .........
ENDIF
```

dove: ***condizione*** **= espressione logica di abilitazione del primo set di istruzioni**

Utilizzando tale costrutto, il programma di prima può essere riscritto nel seguente modo:

```
IF [I10.1]
      O1.1 = T[I100.2];
ELSE
      O1.1 =I2.2;
ENDIF
```

3.10 – Macro funzioni

Nella scrittura di un programma complesso sovente capita che alcuni gruppi di istruzioni ricorrano più volte in maniera identica o con minime modifiche tra di loro.

Si pensi ad es. ad un programma per la gestione di un albergo. In tale applicazione le istruzioni relative alla gestione della camera (riporto allarmi, controllo accensioni, controllo temperatura, ecc.) si ripetono identicamente nel programma tante volte quanto sono le camere.
A ben considerare, tale gruppo di istruzioni realizza nel suo insieme una funzione complessa (nell'esempio la gestione di una camera).
Risulta dunque opportuno riunire tali gruppi di istruzioni all'interno di una nuova funzione utente (macro funzione) che verrà richiamata dal programma principale.

A tal scopo occorre utilizzare la direttiva MACRO secondo la seguente sintassi:

```
MACRO nome_macro (par1, par2, ..)
{
      .........
      equazione
      equazione
      .........
}
```

dove: ***nome_macro*** **è la stringa che definisce il nome della macro funzione definita dall'utente**
par1, par2, .. **sono i parametri che vengono ricevuti dalla macro funzione**

All'interno del programma la macro funzione viene utilizzata nel seguente modo:

```
      .........
      equazione
      nome_macro (var1,var2, ..);
      equazione
      .........
```

dove: ***nome_macro*** **è il nome della macro funzione già definita**
var1, var2, .. **sono i parametri che vengono passati alla macro funzione.**
Si noti che i parametri che vengono passati alla macro funzione verranno associati alle grandezze *par1, par2, ..* in base all'ordine con cui compaiono in parentesi.

Una macro funzione può ritornare un valore utilizzabile nel programma principale.
In tal caso la sintassi è la seguente:

```
MACRO nome_macro (par1, par2, ..)
{
        .........
        equazione
        equazione
        .........
        valore;
}
```

dove: ***valore*** **è la variabile il cui valore viene ritornato dalla macro funzione**

All'interno del programma la macro funzione viene utilizzata nel seguente modo:

```
        .........
        equazione
        var3 = nome_macro (var1,var2, ..);
        equazione
        .........
```

dove: ***var3*** **è il nome della variabile in cui viene depositato il valore restituito dalla macro**

Qualora in uno stesso programma vengano utilizzate macro funzioni e definizioni di costanti e variabili utente, l'ordine con cui il compilatore effettua la sostituzione è il seguente:

1 - sostituzione DEFINE
2 - sostituzione MACRO
3 - definizione variabili

Occorre dunque tener conto di tale ordine per evitare che ad es. in una definizione di MACRO compaiano variabili non ancora inizializzate dal compilatore.

Es. di programma con passaggio di parametri:

```
//-----------------------------
//  Comando luce esterna
//-----------------------------

//-------------------------
//  Definizioni utente
//-------------------------
DEFINE START 16:30
DEFINE STOP 0:00
DEFINE Giardino O1
DEFINE Scala O2
DEFINE Garage O3
DEFINE Crepuscolare I1.1
DEFINE Orologio V1.1

MACRO Notturna  (Out)
{
    Out.1 = Crepuscolare & Orologio;
    Out.2 = Crepuscolare;
}

//-------------------------
// Programma
//-------------------------
Orologio = PRG1[Lu-Ve, START, STOP];

Notturna  (Giardino);
Notturna  (Scala);
Notturna  (Garage);
```

Es. di programma senza passaggio di parametri:

```
//----------------------------------
//  Calcolo giorno dell'anno
//----------------------------------

//-------------------------
//  Definizioni utente
//-------------------------
DEFINE Capodanno 365
DEFINE Auguri V1.2
INTEGER : Oggi
INTEGER : Attesa
INTEGER : AttesaOld

MACRO GiornoAnno()
{
    31*(RTC(MO)==2) + 59*(RTC(MO)==3) + 90*(RTC(MO)==4) + 120*(RTC(MO)==5) +
    151*(RTC(MO)==6) + 181*(RTC(MO)==7) + 212*(RTC(MO)==8) + 243*(RTC(MO)==9) +
    272*(RTC(MO)==10) + 303*(RTC(MO)==11) + 334*(RTC(MO)==12) + RTC(DM);
}

//-------------------------
// Programma
//-------------------------

Oggi = GiornoAnno();
AttesaOld = Attesa;
Attesa = Capodanno – Oggi;
IF [ Oggi == Capodanno ]
     Auguri = ON;
ELSE
     Auguri = OFF;
ENDIF

V1.1 = !Auguri & (Attesa != AttesaOld);
M1 = MESS[“Buon Anno", "dal PICnet", ON, OFF, Auguri];
M2 = MESS[“Ancora %Attesa% giorni”, ”a Capodanno”, OFF, OFF,V1.1];
D254 = M1 , M2;
```

3.11 – Dichiarazione moduli

In alcune applicazioni di raccolta dati vengono installati dei moduli di ingresso o di uscita senza che il loro riferimento compaia nelle equazioni del programma utente. In questo caso la presenza in rete del modulo verrebbe ignorata dal sistema, che provvede ad interrogare periodicamente i soli moduli cui si fa esplicito riferimento nel programma utente.
In questi casi è possibile forzare l'interrogazione dei moduli non utilizzati dichiarandone la presenza con la funzione di dichiarazione secondo la seguente sintassi:

DECL(*lista*);

dove: ***lista*** **è l'elenco dei moduli da interrogare separati dalla virgola.**

Es. :

```
O2.1 = I1.4;
DECL(I3, I4, I10);
```

In questo caso il sistema interrogherà periodicamente i moduli n° 1 e 2 perché presenti in una equazione ed i moduli n° 3, 4 e 10 perché dichiarati.

3.12 – Aggiornamento ingressi

Nel normale funzionamento il modulo master provvede ciclicamente ad effettuare un'interrogazione di tutti i moduli di ingresso. Il tempo di rinfresco di tali moduli è variabile poiché dipende dal numero di moduli collegati in rete e dalla velocità di trasmissione bus, ed è in genere un tempo molto breve (tipicamente dell'ordine di alcuni decimi di secondo).
In impianti estesi con velocità di trasmissione bus non elevata, tuttavia, il rinfresco continuo di tutti gli ingressi può portare ad un certo rallentamento del sistema.
In tal caso è possibile evitare tale aggiornamento continuo e forzare manualmente in tempi prefissati il rinfresco di alcuni moduli di ingresso non prioritari o i cui ingressi sono lentamente variabili (ad es. moduli di ingresso analogico, moduli termostato, ecc).

Tale lettura forzata viene effettuata secondo la seguente sintassi:

READINPUT(*addr*);

dove: ***addr* è l'indirizzo del modulo di ingresso da interrogare.**

I moduli di ingresso il cui indirizzo compare in una istruzione READINPUT non sono più letti dal meccanismo di rinfresco automatico del programma.

Nel caso di moduli di ingresso a due canali, valgono le solite convenzioni sintattiche:

Es. :

```
READINPUT(I2:A);   //   legge il primo canale del modulo I2
READINPUT(I2:B);   //   legge il secondo canale del modulo I2
READINPUT(I2);     //   legge l'intero modulo I2 (se il modulo è a due canali li legge
                   //   entrambi)
```

3.13 – Impostazione e lettura parametri

Alcuni moduli hanno delle funzionalità autonome che dipendono dal valore di alcuni parametri impostati al momento dell'accensione del sistema e memorizzati all'interno del modulo stesso.
A titolo di esempio, sono parametri i valori di regolazione minima e massima della temperatura dei moduli PN*term*, alcune impostazione del modul PN MAS, ecc.

Per l'impostazione di tali parametri occorre utilizzare la direttiva SETPAR secondo la seguente sintassi:

SETPAR[*addr, par, value*];

dove: ***addr :*** **indirizzo del modulo**
par: **indirizzo del parametro da impostare**
value : **valore del parametro da impostare**

Il comando SETPAR viene eseguito all'accensione del sistema e viene ripetuto durante il funzionamento del programma ogni qualvolta il valore della variabile "*value*" venga modificato.

Es. :

```
SETPAR[10, 8, 22];
```

In questo caso il parametro n° 8 del modulo n° 10 viene impostato con il valore di 22.
Tale impostazione viene effettuata una sola volta all'accensione del sistema.

Es. :

```
SETPAR[10, 8, V1];
```

In questo caso il parametro n° 8 del modulo n° 10 viene impostato con il valore della variabile V1. Tale impostazione viene effettuata all'accensione del sistema e ripetuta ad ogni variazione della variabile V1.

Esiste la possibilità di condizionare l'aggiornamento di un parametro al verificarsi di una condizione logica. In tal caso la sintassi completa del comando è la seguente:

SETPAR[*addr, par, value, cond*];

dove: ***addr :*** **indirizzo del modulo**
par: **indirizzo del parametro da impostare**
value : **valore del parametro da impostare**
cond **: condizione logica di consenso all'impostazione (funziona sul fronte del segnale)**

Es. :

```
SETPAR[10, 8, V1, Inverno];
```

In questo caso il parametro n° 8 del modulo n° 10 viene impostato con il valore della variabile V1 solo quando la variabile utente "Inverno" diventa vera.

NOTA: non è possibile utilizzare il comando SETPAR all'interno di un costrutto IF ... ENDIF.
Per condizionare l'esecuzione di un comando SETPAR in base ad una certa condizione logica occorre specificare tale condizione nell'apposito campo del comando stesso.

Esiste ovviamente la possibilità di leggere il valore di un parametro come quello di altre variabili interne ai moduli slave, mediante il comando READ secondo la seguente sintassi:

READ[*addr, par*];

dove: ***addr :*** **indirizzo del modulo**
par: **indirizzo del parametro o della variabile da leggere**

Es. :

```
V1 = READ[10, 8];
```

In questo caso alla variabile V1 viene assegnato il valore del parametro n° 8 del modulo n° 10.

3.14 – Supervisione dell'impianto

Quando il sistema è connesso ad un personal computer è possibile mediante il software PN LINK effettuare il controllo dello stato degli ingressi e delle uscite dell'impianto.
Se previsto in fase di programmazione, è possibile anche interagire con l'impianto modificando lo stato delle sue uscite semplicemente cliccando con il mouse sull'uscita che si intende modificare.

Affinché la commutazione richiesta sull'uscita abbia effetto occorre abilitare tale funzionalità a livello di programma utente.
Tale abilitazione si ottiene facendo precedere l'assegnazione dell'uscita che si vuole modificare dal simbolo #.
Il software PN LINK permette di agire su ciascuna uscita attraverso una variabile di appoggio che viene richiamata nel programma utente con la sigla PN. Tale variabile può essere liberamente utilizzata all'interno dell'equazione relativa all'uscita interessata secondo una logica stabilita dall'utente.

Es. :

```
O1.1 =   T[I2.2];
#O1.2 =  I2.1 | I2.4 | PN;
```

In questo caso l'uscita O1.1 non può essere modificata da PN LINK, mentre l'uscita O1.2 verrà assegnata sulla base della sua equazione e di quanto imposto dal programma PN LINK.

In pratica la variabile PN è una variabile virtuale associata ad ogni uscita modificabile dalla supervisione che viene comandata dal software PN LINK secondo due differenti modalità: modalità *comando diretto* e modalità *passo-passo*.
Tali modalità sono selezionabili dall'apposita voce del menù di configurazione.

In modalità *comando diretto*, cliccando sull'uscita interessata si provoca l'attivazione della variabile PN associata per tutto il tempo della pressione del pulsante. La variabile PN ritorna a zero al momento del rilascio (funzionamento da pulsante).

In modalità *passo-passo*, cliccando su un'uscita si provoca la commutazione della variabile PN associata (funzionamento da interruttore).

Es. :

```
O1.1 =   T[I2.2];
#O1.2 =  T[I2.1 | PN];
#O4.1 =  T[I2.3] ^ PN;
```

In questo caso:

- **l'uscita O1.1 non può essere modificata da PN LINK,**
- **l'uscita O1.2 verrà commutata da un pulsante collegato all'ingresso I2.1 e dalla variabile PN (che dovrà dunque essere configurata in modalità comando diretto).**
- **l'uscita O4.1 verrà comandata in deviata da un passo-passo comandato da un pulsante di ingresso e dalla variabile PN gestita dal programma PN LINK (che in tal caso deve essere impostata sulla modalità comando passo-passo).**

4.0 - Moduli speciali

Alcuni moduli del sistema PIC*net* sono delle unità funzionali autonome in grado di svolgere funzioni complesse anche in assenza della connessione bus, ma che, in presenza del bus PIC*net*, permettono prestazioni non ottenibili in impianti con apparecchiature tradizionali.
Nei paragrafi che seguono vengono dunque prese in esame tali apparecchiature per metterne in evidenza le particolarità di programmazione.

4.1 – Modulo PN MAS

I moduli PN MAS (PN MAS-1RD, PN MAS-2RD, PN MAS-3RD) sono moduli master del sistema PICnet previsti per installazione da guida DIN.
Il modulo non dispone di ingressi e uscite a bordo, ma presenta una serie di funzionalità che possono essere gestite con i consueti comandi di lettura e scrittura all'indirizzo 254.

Oltre al display che, se presente, è indirizzato con il valore 254 (vedi § 3.3) le altre funzionalità possono essere gestite mediante la direttiva SETPAR (per la scrittura) o READ (per la lettura) (vedi § 3.14) ai seguenti indirizzi:

Grandezza	*Accesso*	*Indirizzo*	*Range*	*Default*
Abilitazione cicalino (equivale all'opzione nel Menu)	Scrittura	1	1: abilitazione 0: disabilitazione	x
Abilitazione relè (equivale all'opzione nel Menu)	Scrittura	2	1: abilitazione 0: disabilitazione	x
Retroilluminazione display (equivale all'opzione nel Menu)	Scrittura	6	1: abilitazione 0: disabilitazione	x
Set baud rete bus (equivale all'opzione nel Menu)	Scrittura	9	0 -> 5.000 1 -> 9.600 2 -> 18.000 3 -> 25.000 4 -> 41.600 5 -> 62.500	x
Intervallo interrogazione messaggi SMS in sec.	Scrittura	15	5-255	15
Attivazione cicalino	Scrittura	101	1: ON 0: OFF	- -
Attivazione relè	Scrittura	102	1: ON 0: OFF	- -
Set ora RTC interno (solo rev. 1.21.02)	Scrittura	100	0-23	- -
Set velocità rampa funzione DIMMER	Scrittura	104	1 - 64	5
Ritardo partenza rampa DIMMER	Scrittura	105	4 – 256 (decimi di secondo)	10
N. moduli utilizzati nel programma utente	Lettura	101	0 - 250	- -
N. moduli guasti	Lettura	102	0 - 250	- -

4.2 – Modulo PN MAS LITE

I moduli PN MAS LITE sono moduli master del sistema PICnet previsti per installazione da incasso in scatola 503.
Il modulo ha le stesse funzionalità delle versioni da guida DIN e può gestire fino a 16 moduli slave, ma non dispone delle seguenti risorse:

- **relè di scambio di allarme**
- **display di bordo e relativa interfaccia utente**
- **plug per connessione RS232**
- **cicalino di allarme.**

Il modulo dispone invece di un pannello frontale dotato di 8 pulsanti ed 8 led di segnalazione che vengono gestiti come normali ingressi ed uscite del sistema con indirizzo 254.
Come per il modulo PN PAN per i led di segnalazione è possibile scegliere tra un funzionamento ad accensione fissa o lampeggiante.
L'accensione fissa si ottiene attivando il corrispondente bit di uscita (bit 1-8), mentre per l'accensione lampeggiante occorre settare anche il corrispondente bit della parte alta (bit 9-16).

Data la mancanza del display e del cicalino di bordo, l'eventuale condizione di allarme del sistema viene evidenziata con il lampeggio a rotazione di tutti gli 8 led del pannello frontale.

Tale funzionalità può essere gestita mediante la direttiva SETPAR (vedi § 3.14) al seguente indirizzo:

- **abilitazione rotazione led: indirizzo 106 [default = 0]**
  (abilitazione = 0; disabilitazione = 1)

Esempio di definizione risorse:

```
//----------------------------
//      PN MAS LITE
//----------------------------
DEFINE  Puls_1     I254.1        // definizione pulsante 1
DEFINE  Puls_2     I254.2        // definizione pulsante 2
DEFINE  Puls_3     I254.3        // definizione pulsante 3
DEFINE  Puls_4     I254.4        // definizione pulsante 4
DEFINE  Puls_5     I254.5        // definizione pulsante 5
DEFINE  Puls_6     I254.6        // definizione pulsante 6
DEFINE  Puls_7     I254.7        // definizione pulsante 7
DEFINE  Puls_8     I254.8        // definizione pulsante 8

DEFINE  Spia_1     O254.1        // definizione led fisso 1
DEFINE  Spia_2     O254.2        // definizione led fisso 2
DEFINE  Spia_3     O254.3        // definizione led fisso 3
DEFINE  Spia_4     O254.4        // definizione led fisso 4
DEFINE  Spia_5     O254.5        // definizione led fisso 5
DEFINE  Spia_6     O254.6        // definizione led fisso 6
DEFINE  Spia_7     O254.7        // definizione led fisso 7
DEFINE  Spia_8     O254.8        // definizione led fisso 8

DEFINE  Lamp_1     O254.9        // definizione led lampeggiante 1
DEFINE  Lamp_2     O254.10       // definizione led lampeggiante 2
DEFINE  Lamp_3     O254.11       // definizione led lampeggiante 3
DEFINE  Lamp_4     O254.12       // definizione led lampeggiante 4
DEFINE  Lamp_5     O254.13       // definizione led lampeggiante 5
DEFINE  Lamp_6     O254.14       // definizione led lampeggiante 6
DEFINE  Lamp_7     O254.15       // definizione led lampeggiante 7
DEFINE  Lamp_8     O254.16       // definizione led lampeggiante 8
```

4.3 – Modulo PN DIM

I moduli PN DIM sono in grado di gestire 2 uscite con dimmer per la regolazione di carichi resistivi.
Le uscite sono tutte riferite ad una tensione di alimentazione esterna comune indipendente dalla tensione di alimentazione del sistema.
I moduli dispongono inoltre di 4 ingressi digitali non optoisolati normalmente aperti.
I contatti di ingresso vengono alimentati da una tensione derivata direttamente dall'alimentazione del bus.

I moduli PN DIM sono dunque moduli che riuniscono in un unico indirizzo:
- **due uscite analogiche ad 8 bit: livello canale 1 e canale 2**
- **quattro ingressi digitali: ingressi per incremento e decremento canale 1 e 2**

Per il modulo dimmer sono possibili due distinte modalità operative.

Funzionamento locale
In tale modalità il modulo PN DIM si comporta come un normale dimmer a due canali indipendenti.
I quattro ingressi disponibili svolgono le funzioni rispettivamente di ingresso "up" e "down" dei due canali di uscita. Come in un normale dimmer ciascuna uscita viene alternativamente accesa o spenta da successive pressioni dei pulsanti connessi agli ingressi del modulo.

In caso di spegnimento, l'uscita viene disalimentata, mentre in caso di accensione l'uscita assume l'ultimo valore impostato.
In maniera indipendente per ogni canale la prolungata pressione di uno dei due pulsanti produce invece l'aumento (valore max 100%) o la diminuzione (valore min. pari a circa il 10%) del valore dell'uscita relativa.

E' la modalità di funzionamento più comune e più semplice perché non comporta per il master nessuna elaborazione.
Tale modalità operativa è quella che il modulo assume alla prima accensione o in caso di avaria del bus PIC*net*.

Funzionamento remoto
Qualora connesso ad un bus PIC*net*, il modulo PN DIM abbandona la modalità di controllo locale e imposta le sue uscite sulla base delle direttive ricevute dal modulo master del sistema.
In tale modalità operativa i quattro ingressi presenti sul modulo assumono il significato di generici ingressi digitali del sistema e come tali possono essere utilizzati dal programma utente.

In tale modalità di funzionamento è dunque possibile in maniera centralizzata effettuare le seguenti operazioni:
- **accensione/spegnimento indipendente di ciascun canale**
- **regolazione indipendente di ciascun canale**

Il modulo PNDIM viene gestito come un normale modulo dotato di due uscite analogiche (vedi § 1.2.1) ed ingressi digitali secondo la seguente corrispondenza:

- uscita 1:	**uscita analogica primo canale**	**Ox:A**
- uscita 2:	**uscita analogica secondo canale**	**Ox:B**
- ingresso up canale 1:	**ingresso n° 1**	**Ix.1**
- ingresso down canale 1:	**ingresso n° 2**	**Ix.2**
- ingresso up canale 2:	**ingresso n° 3**	**Ix.3**
- ingresso down canale 2:	**ingresso n° 4**	**Ix.4**

essendo x l'indirizzo del modulo.

```
//-------------------------------------------------------------
//      Programma esempio di gestione
//      modulo PN DIM
//-------------------------------------------------------------
DEFINE    Out1         O4:a      // uscita dimmer canale 1
DEFINE    Out2         O4:b      // uscita dimmer canale 2
DEFINE    Up_ch1       I4.1      // ingresso up canale 1
DEFINE    Dw_ch1       I4.2      // ingresso down canale 1
DEFINE    Up_ch2       I4.3      // ingresso up canale 2
DEFINE    Dw_ch2       I4.4      // ingresso down canale 2

DEFINE    Spegni       I2.1      // ingresso pulsante spegnimento generale
DEFINE    Allarme      I2.2      // ingresso contatto allarme scattato generale
DEFINE    Scala        O5.4      // luce scala
DEFINE    Pulsante     I5.2      // pulsante accensione luce scala
DEFINE    Accendi      V1.1      // accensione temporizzata a seguito di un allarme


Accendi = SHOT[300, Allarme];     // in caso di allarme accendo le luci per 30 secondi

//-------------------------------------------------------------------------------------------
// Nel normale funzionamento la luce scala si accende e spegne comandata in passo-passo
// da un pulsante.
// La pressione del pulsante di spegnimento generale attiva il passo-passo solo se la luce è accesa
// (e quindi la spegne).
// Il comando di accensione allarme generale comanda direttamente l'uscita lasciando inalterato
// lo stato del passo-passo.
//-------------------------------------------------------------------------------------------
Scala = T[Pulsante  | (Scala & Spegni)] | Accendi;

//-------------------------------------------------------------------------------------------
//  Nel normale funzionamento il dimmer è comandato dai pulsanti previsti.
//  La pressione momentanea del pulsante di spegnimento generale comanda il canale solo
//  se la luce è accesa (e quindi la spegne istantaneamente).
//  La pressione prolungata del pulsante di spegnimento generale comanda la riduzione del canale
//  solo  se la luce è accesa (e quindi la spegne lentamente).
//  Il comando di accensione allarme generale comanda direttamente l'uscita lasciando inalterato
//  lo stato del dimmer.
//-------------------------------------------------------------------------------------------
Out1 = DIMMER[ Up_ch1, Dw_ch1 | ((Out1 != 0) & Spegni)] | (255*Accendi);
Out2 = DIMMER[ Up_ch2, Dw_ch2 | ((Out2 != 0) & Spegni)] | (255*Accendi);
```

4.4 – Modulo PN TERM

I moduli PN TERM sono moduli autonomi che svolgono tutte le funzioni di un normale termostato digitale. All'interno del modulo sono presenti il sensore di temperatura, il potenziometro per l'impostazione della temperatura desiderata (set-point) ed un relè di appoggio per il comando di apparecchi di condizionamento.

I moduli PN TERM sono dunque moduli che riuniscono in un unico indirizzo:

- **due ingressi analogici ad 8 bit: temperatura ambiente e set-point**
- **tre uscite digitali: commutazione estate/inverno, commutazione locale/remoto, comando relè**
- **tre ingressi digitali: ingresso ausiliario 1, ingresso ausiliario 2 e stato relè**

Per il modulo termostato sono possibili due distinte modalità operative.

Funzionamento locale

In tale modalità il modulo PN TERM si comporta come un normale termostato ambiente, effettuando la regolazione della temperatura (impostata mediante il potenziometro sul frontale) ed il comando del relè.
E' la modalità di funzionamento più comune e più semplice perché non comporta per il master nessuna elaborazione.

In tale modalità di funzionamento è comunque possibile in maniera centralizzata effettuare le seguenti operazioni:

- **commutazione estate/inverno**
- **lettura temperatura ambiente**
- **lettura set-point potenziometro**
- **lettura dello stato del relè (in modo da poter riportare in remoto il comando pompe o altro)**
- **impostazione del valore minimo del set-point (per impedire che l'utente chieda troppo freddo)**
- **impostazione del valore massimo del set-point (per impedire che l'utente chieda troppo caldo)**
- **compensazione offset di lettura**

In questa modalità di funzionamento, impostando allo stesso valore il minimo ed il massimo del set-point, si può imporre il riferimento di temperatura sostituendosi così al set-point dell'utente.
Questa possibilità risulta particolarmente utile per l'attivazione di scenari diversi, ad es. per forzare temperature di mantenimento o antigelo in assenza di persone nei locali.

Tale modalità operativa è quella che il modulo assume all'accensione o in caso di avaria del bus PIC*net*.

Funzionamento remoto

Forzando il funzionamento in remoto del modulo, il master si sostituisce al termostato nel pilotaggio del relè che può dunque essere attivato come una qualsiasi uscita digitale per permettere la realizzazione di funzioni più sofisticate (ad es. spegnimento del fan-coil di camera durante determinate fasce orarie o in caso di apertura della finestra, inseguimento di profili di temperatura complessi, ecc.).

Sul modulo è comunque disponibile un ingresso digitale non optoisolato che dispone di due distinte modalità di utilizzo. Nella prima modalità si tratta di un generico ingresso digitale a disposizione del sistema, mentre nella seconda modalità viene utilizzato per il forzamento del funzionamento in locale. In tal modo l'utente è sempre in grado di ripristinare la funzionalità locale del termostato by-passando il forzamento da remoto effettuato dal modulo master.

Il modulo PN TERM viene gestito come un normale modulo dotato di due ingressi analogici (vedi § 1.2.1) ed uscite ed ingressi digitali secondo la seguente corrispondenza:

- **temperatura ambiente: ingresso analogico primo canale Ix:A**
- **set-point utente: ingresso analogico secondo canale Ix:B**
- **comando relè di appoggio: uscita n° 1 (1=ON) Ox.1**
- **commutazione estate/inverno: uscita n° 2 (1=estate) Ox.2**
- **commutazione locale/remoto: uscita n° 3 (1=remoto) Ox.3**
- **modalità ingresso aux: uscita n° 5 (1=input generico) Ox.5**

essendo x l'indirizzo del modulo.

Le temperature vengono lette e scritte come variabili intere ad 8 bit con la seguente corrispondenza:

0.1953125 = 1 °C

Le soglie inferiore e superiore di regolazione della temperatura sono impostabile mediante la direttiva SETPAR (vedi § 3.14) ai seguenti indirizzi:

- **soglia massima: indirizzo 6**
- **soglia minima: indirizzo 7**
- **abilitazione soglie: indirizzo 8 (abilitazione = 0; disabilitazione = 255)**

E' inoltre possibile correggere eventuali offset di temperatura dovuti ad errate installazioni sempre mediante la direttiva SETPAR al seguente indirizzo:

- **offset di lettura: indirizzo 14**

Ad es. per forzare i limiti del modulo n° 10 si scriverà:

```
DEFINE      TempMin     102             // valore corrispondente a 20 °C
DEFINE      TempMax     108             // valore corrispondente a 21 °C
DEFINE      Abilita     0               // costante abilitazione soglie
DEFINE      Disabilita  255             // costante disabilitazione soglie
SETPAR[10, 7, TempMin];                 // set soglia minima
SETPAR[10, 6, TempMax];                 // set soglia massima
SETPAR[10, 8, Abilita];                 // abilita nuove soglie
```

Lo stato del relè e delle altre condizioni di funzionamento possono essere letti mediante la direttiva READ (vedi § 3.14) all'indirizzo 77 con la seguente corrispondenza:

- **estate/inverno: bit 2 (0 = Inverno, 1 = Estate)**
- **locale/remoto: bit 3 (0 = Locale, 1 = Remoto)**
- **ingresso aux 1 bit 4 (0 = OFF, 1 = ON)**
- **modalità ingresso aux bit 5 (0 = Forzamento, 1 = Input generico)**
- **stato relè: bit 6 (0 = OFF, 1 = ON)**
- **ingresso aux2 bit 7 (0 = OFF, 1 = ON)**

Ad es. per riportare a distanza sull'uscita O2.6 lo stato del relè del modulo n° 10 si scriverà:

```
V1 = READ[10, 77];      // leggo la parola di stato
O2.6 = V1.6;            // comando l'uscita secondo lo stato del relè
```

```
//-------------------------------------------------------------
//      Programma esempio di gestione
//      modulo PN TERM con funzionamento in locale
//-------------------------------------------------------------
DEFINE    KC            0.1953125   // costante di conversione temperatura
DEFINE    PNtemp        I1:a        // canale di lettura temperatura
DEFINE    PNsetp        I1:b        // canale di lettura set-point
DEFINE    PNrele        O1.1        // address di attivazione relè
DEFINE    PNEstInv      O1.2        // address di commutazione est/inv
DEFINE    PNRemLoc      O1.3        // address di commutazione locale/remoto
DEFINE    ReleOut       T[I4.1]     // ingresso pulsante per commutazione rele
DEFINE    EstateInverno T[I4.2]     // ingresso pulsante per commutazione est/inv
DEFINE    RemotoLocale  T[I4.3]     // ingresso pulsante per commutazione locale/remoto

FLOAT : Temperatura
FLOAT : Setpoint

Temperatura = KC * ITOF[PNtemp];     // converto il dato ricevuto in gradi °C
Setpoint = KC * ITOF[PNsetp];        // converto il dato ricevuto in gradi °C

PNrele = ReleOut;
PNEstInv = EstateInverno;
PNRemLoc = RemotoLocale;

//-------------------------------------------------------------
//       visualizzo la temperatura ed il set-point
//-------------------------------------------------------------
V1.1 = PWM[10,10,ON];                // sequencer messaggi
M1 = MESS["  T amb = %Temperatura%","  T imp = %Setpoint%",OFF,OFF,V1.1];
D254 = M1;
```

4.5 – Modulo PN LUX

**I moduli PN LUX sono moduli luxmetro utilizzati per il rilevamento della luminosità ambientale.
Oltre al sensore di luce sul modulo sono presenti due ingressi ausiliari ed un relè di appoggio.**

I moduli PN LUX sono dunque moduli che riuniscono in un unico indirizzo:

- **un ingresso analogico ad 8 bit**
- **un uscita digitale: comando relè**
- **tre ingressi digitali: ingresso ausiliario 1, ingresso ausiliario 2 e stato relè**

Il modulo PN LUX viene dunque gestito come un normale modulo dotato di un ingresso analogico (vedi § 1.2.1) ed uscite ed ingressi digitali secondo la seguente corrispondenza:

- luce ambiente:	**ingresso analogico**	**Ix:A**
- comando relè ausiliario:	**uscita n° 1 (1=ON)**	**Ox.1**

essendo x l'indirizzo del modulo.

La luminosità viene letta come variabile intera ad 8 bit con la seguente corrispondenza:

1 = 3.92 [lux]

Lo stato del relè e delle altre condizioni di funzionamento possono essere letti mediante la direttiva READ (vedi § 3.14) all'indirizzo 77 con la seguente corrispondenza:

- ingresso aux	**bit 4**	**(0 = OFF, 1 = ON)**
- stato relè:	**bit 6**	**(0 = OFF, 1 = ON)**
- ingresso aux 2	**bit 7**	**(0 = OFF, 1 = ON)**

Ad es. per regolare la luce artificiale mediante un modulo PN DA che controlla un reattore dimmerabile in modo da mantenere l'illuminamento complessivo ad un valore prefissato, si scriverà:

```
//-----------------------------------------------
//  Programma demo gestione luxmetro
//-----------------------------------------------

//----------------------------
//      n. 2 PN DA
//----------------------------
DEFINE  Out1_1          O2:A     //  uscita 0-10 primo canale
DEFINE  Out2_1          O2:B     //  uscita 0-10 secondo canale

//----------------------------
//      n. 3 PN 2I2O
//----------------------------
DEFINE  Rele1_1         O3.1     //  uscita relè primo canale
DEFINE  Rele2_1         O3.2     //  uscita relè secondo canale
DEFINE  In1_1           O3.1     //  ingresso aux primo canale
DEFINE  In2_1           O3.2     //  ingresso aux secondo canale

//----------------------------
//      n. 4 PN LUX
//----------------------------
DEFINE pnlux            I4:A
```

```
//-------------------------------------------------------------
//          Definizione variabili
//-------------------------------------------------------------
//-------------------------
//  Definizioni variabili
//-------------------------
INTEGER : errore
INTEGER : setpoint
INTEGER : lux

//-------------------------
//  Definizioni costanti
//-------------------------
DEFINE   Delta    1             // variazione minima uscita
DEFINE   KC       4             // approssimo la costante di lettura luxmetro 1=3.921569
DEFINE   Seq      V1

//-------------------------------------------------------------------
//
//                      Inizio Programma
//
//-------------------------------------------------------------------

lux = KC * ITOF[pnlux];     //  lettura lux
setpoint = UVAR["Lux richiesti", 100, 800, 5];
errore = setpoint - (pnlux * KC);
Seq = CNT1[PWM[5,5,ON],ON,Seq>=10];

//-----------------------------------------------
//  aggiorno la regolazione solo ogni 10 secondi
//-----------------------------------------------
IF [Seq==0]
    IF [errore > 0]
        IF [errore > 10]
            Out1_1 = Out1_1+Delta;  // per accelerare la regolazione aumentare il delta
            IF [Out1_1 >= 254] Out1_1=254; ENDIF
        ENDIF
    ELSE
        errore = (pnlux * KC) - setpoint;
        IF [errore > 10]
            Out1_1 = Out1_1-Delta;
            IF [Out1_1 <= 0] Out1_1=0; ENDIF
        ENDIF
    ENDIF
ENDIF

Rele1_1 = (Out1_1 != 0);


M1 = MESS[" Lux amb = %lux%"," Lux imp = %setpoint%",OFF,OFF,PWM[2,2,ON]];
D254 = M1;
```

4.6 – Modulo PN METEO

I moduli PN METEO sono moduli che racchiudono in un unico dispositivo una completa stazione meteorologica per la rilevazione di luce, vento e pioggia.

I moduli PN METEO sono dunque moduli che riuniscono in un unico indirizzo:

- **un ingresso analogico ad 8 bit: velocità vento**
- **un ingresso analogico ad 8 bit: luminosità ambiente**
- **un ingresso digitale: presenza pioggia**

Il modulo PN METEO viene gestito come un normale modulo dotato di due ingressi analogici (vedi § 1.2.1) secondo la seguente corrispondenza:

- **luce ambiente: ingresso analogico primo canale Ix:A**
- **velocità vento: ingresso analogico secondo canale Ix:B**

essendo x l'indirizzo del modulo.

La velocità del vento viene letta come variabile intera ad 8 bit con la seguente corrispondenza:
1 = 0.3 [km/h];

La luminosità viene letta come variabile intera ad 8 bit secondo due differenti scale di lettura settabili da software con la seguente corrispondenza:

alta risoluzione: 1 = 10 [lux];
bassa risoluzione: 1 = 1000 [lux];

La risoluzione di lettura può essere commutata anche dinamicamente scrivendo su un' uscita digitale secondo la seguente convenzione:

Ox.1 = ON = decine di lux OFF = migliaia di lux

In tal modo è possibile ad es. durante il giorno impostare la sensibilità più bassa per riconoscere la presenza di nuvole o sole e poi nel pomeriggio commutare sulla sensibilità più alta per utilizzare il modulo come sensore crepuscolare.

La presenza pioggia può essere letta mediante la direttiva READ (vedi § 3.14) all'indirizzo 0 con la seguente corrispondenza:

- **presenza pioggia bit 1 (0 = SECCO, 1 = PIOGGIA)**

Ad es. per gestire l'apertura/chiusura di lucernari o tende mediante un modulo PN Meteo, si potrà scrivere:

```
//-----------------------------------------------
//  Programma demo gestione PN Meteo
//-----------------------------------------------

//----------------------------
//      n. 1 PN2I2O
//----------------------------
DEFINE          Salita          O1.1          //  salita tenda
DEFINE          Discesa         O1.2          //  discesa tenda
DEFINE          Puls_up         I1.1          //  pulsante salita
DEFINE          Puls_dw         I1.2          //  pulsante discesa
```

```
//------------------------------
//  Definizioni variabili
//------------------------------
DEFINE          Meteo             100
DEFINE          Luce              V100
DEFINE          Vento             V101
DEFINE          Meteo_st          V102          // lettura stato meteo
DEFINE          Pioggia           V102.1        // lettura stato pioggia
DEFINE          SogliaLux         V103          // soglia intervento crepuscolare
DEFINE          SogliaVento       V104          // soglia vento
DEFINE          Notte             V1.1          // consenso crepuscolare
DEFINE          GoUp              V1.2          // moto tenda
DEFINE          GoDw              V1.3          // moto tenda
DEFINE          Apri              V1.4          // comando apertura generale
DEFINE          Chiudi            V1.5          // comando chiusura generale

//--------------------------------------
//  Definizioni costanti e macro
//--------------------------------------
DEFINE          Tmove             250           // durata max apertura/chiusura tenda
DEFINE          DLY               5             // attesa tra apertura e chiusura tenda

MACRO Tenda (Salita, Discesa, GoUp, GoDw, MotUp, MotDw, Time, AllUp, AllDw)
{
    GoUp = SR[Salita, GoUp];         // comando apertura
    GoDw = SR[Discesa, GoDw];        // comando chiusura
    MotUp = TMR[DLY, Time, GoUp | AllUp, (GoUp & MotUp) | GoDw | AllDw];
    MotDw = TMR[DLY, Time, GoDw | AllDw, (GoDw & MotDw) | GoUp | AllUp];
}

//--------------------------------------------------------------------
//                      Inizio Programma
//--------------------------------------------------------------------
SogliaLux = UVAR["Soglia crepusc.", 1,250,1,5];
SogliaVento = UVAR["Soglia vento", 1,250,1,5];

Luce = IMeteo:A;                 // Leggo luce
Vento = IMeteo:B;                // Leggo Vento
Meteo_st = READ[Meteo,0];        // Flag pioggia

Notte = (Luce <= SogliaLux);     // intervento crepuscolare
Chiudi = (Vento > SogliaVento) | Notte | Pioggia;
Apri = !Notte & !Chiudi;

Tenda (Puls_up, Puls_dw, GoUp, GoDw, Salita, Discesa, Tmove, Apri, Chiudi);
```

4.7 – Modulo PN GSM

I moduli PN GSM sono dei modem telefonici basati su tecnologia GSM dual band che, collegati alla seriale di servizio di un modulo master, permettono di svolgere tutta una serie di funzioni di telecontrollo e teleassistanza di un impianto senza la necessità di impegnare una linea telefonica spesso non disponibile. Come con un normale modem telefonico, con i moduli PN GSM è possibile effettuare connessioni remote con un impianto mediante il software PN LINK o PN LINK PRO ed effettuare quindi da postazione remota le stesse operazioni che si potrebbero svolgere connessi fisicamente sull'impianto:

- **supervisione del funzionamento dell'impianto**
- **comando remoto di utenze**
- **riporto allarmi**
- **modifiche del funzionamento dell'impianto mediante aggiornamento del programma utente**
- **aggiornamento data e ora del master**
- **rinumerazione indirizzo dei moduli.**

Trattandosi di apparecchi con tecnologia GSM, è però possibile, in aggiunta alle funzioni precedenti, la realizzazione di funzioni non disponibili con modem telefonici tradizionali:

- **invio di messaggi SMS a numeri telefonici predefiniti al verificarsi di una qualsiasi condizione logica di attivazione (ad es. un allarme)**
- **ricezione di messaggi SMS da un numero telefonico abilitato per l'attivazione di eventi (ad es. l'accensione o lo spegnimento di un utenza).**

La gestione dei moduli PN GSM avviene mediante tre semplici comandi per l'impostazione del modem, per l'invio e per la ricezione dei messaggi SMS secondo la seguente sintassi:

Inizializzazione modem

L'inizializzazione di un modem connesso alla seriale del master viene effettuata con la seguente sintassi:

INITMODEM(S1);

dove S1 è la stringa di comandi AT richiesta per inizializzare il modem utilizzato. Tale inizializzazione viene effettuata una sola volta all'avvio del programma.

Nel caso del modulo PN GSM, tale inizializzazione non è richiesta.

Invio di messaggi SMS

L'invio di messaggi SMS a telefoni cellulari predefiniti al verificarsi di una determinata condizione di attivazione è possibile secondo la seguente sintassi:

SEND[SMS,1, *ntel*, *stringa*, *cond*];

dove:

***ntel* : stringa contenente il n. telefonico del cellulare cui inviare il messaggio**
***stringa* : stringa contenente messaggio da inviare (max 160 caratteri)**
***cond* : condizione logica di attivazione del messaggio (funziona sul fronte del segnale)**

I primi due campi sono stati previsti per future applicazioni e non possono essere modificati.

Es.

```
SEND[SMS,1,"+393487055483","Prova PICnet", I1.2];
```

Il messaggio "Prova PICnet" viene inviato al n° di cellulare indicato quando l'ingresso I1.2 viene attivato.

Per la stringa contenente il messaggio valgono le stesse possibilità di formattazione già viste a proposito dei messaggi e allarmi utente compresa la possibilità di inserire campi variabili.

Es.

```
SEND[SMS,1,"+393487055483","Attenzione temperatura acqua = %V1% °C ", (V1> 50)];
```

In questo caso quando la variabile analogica V1 supera il valore prefissato, viene inviato un messaggio contenente l'effettivo valore della variabile stessa.

Per ogni messaggio è possibile inserire un massimo di due campi variabili.

Ricezione di messaggi SMS

La ricezione di comandi mediante messaggi SMS trasmessi da telefoni cellulari abilitati è possibile secondo la seguente sintassi:

SMS[*stringa*, *ntel*];

dove:

***stringa* : stringa contenente messaggio da riconoscere (max 160 caratteri)**
***ntel* : stringa contenente il n. telefonico del cellulare di provenienza abilitato**

Es. :

```
O2.1 = SMS[“Accendi Caldaia”, "+393487055483"];
```

In questo caso se dal cellulare indicato viene inviato il messaggio specificato nella stringa ("Accendi caldaia") l'uscita O2.1 viene attivata.

Per comodità di utilizzo è indifferente l'uso di caratteri maiuscoli o minuscoli all'interno della stringa.
Alla ricezione di un messaggio SMS valido, la funzione SMS restituisce un valore vero per 3 cicli di programma.

Per evitare che la presenza di caratteri spuri nel messaggio possa impedire il riconoscimento della stringa di comando, è possibile inserire il carattere asterisco al termine della stringa.
In tal caso qualunque messaggio ricevuto che contenga al suo interno la stringa specificata verrà riconosciuto come un messaggio valido.

Es. :

```
Caldaia_On = SMS[“Accendi Caldaia*”, "+393487055483"];
```

In questo caso l’uscita O2.1 viene attivata se dal cellulare indicato viene inviato un qualunque messaggio contenente la stringa “Accendi caldaia”.

Qualora non si voglia limitare l’invio dei comandi ad un solo cellulare, è possibile inserire il carattere asterisco nel campo *ntel* per abilitare qualunque cellulare all’invio di comandi.

Es. :

```
Caldaia_On = SMS[“Accendi Caldaia*”, "*"];
```

In questo caso l’uscita O2.1 viene attivata se viene ricevuto un qualunque messaggio contenente la stringa “Accendi caldaia”.

Questo permette ad es. di inviare comandi SMS all’impianto mediante provider internet che offrono tale servizio.

4.8 – Modulo PN IR

I moduli PN IR sono moduli di ingresso con comando remoto ad infrarossi. Si tratta in sostanza di un modulo di ingresso in cui la configurazione degli ingressi dipende dal comando inviato al modulo stesso da un telecomando a raggi infrarossi (Pn*remote*).
Il modulo trova applicazione prevalentemente in impianti civili e del terziario (alberghi, sale conferenze, uffici, ambienti per disabili, ecc.) e si compone di una scheda da incasso in scatola 503 ed un frutto dove è alloggiato il ricevitore ad infrarossi.
Il telecomando PN*remote* è dotato di 6 pulsanti di comando ed ogni modulo PN IR può riconoscere fino a 4 distinti telecomandi.

Per ciascun telecomando è possibile scegliere 3 distinte modalità di funzionamento agendo su opportuni ponticelli presenti sulla scheda:

1. **(modalità di default): ad ogni pulsante corrisponde un differente ingresso (pulsanti da 1 a 6);**
2. **il pulsante n. 6 ha la funzione di shift permettendo così di selezionare fino a 10 differenti ingressi (pulsanti da 1 a 5 e pulsanti da 1 a 5 preceduti dallo shift);**
3. **in questa modalità simile a quella di un telecomando TV, ogni pulsante può essere premuto singolarmente (primi 6 canali) o seguito dalla pressione di un secondo pulsante ottenendo così un totale di 42 ingressi differenti. Ad es. premendo prima il pulsante n. 2 e successivamente il pulsante n. 1 viene selezionato il canale 21.**

Il modulo PN IR è visto dal modulo master come un generico modulo di ingresso analogico che restituisce un codice differente a seconda del pulsante premuto, della modalità di funzionamento selezionata e del telecomando utilizzato.

Nel seguito è riportata la tabella di codifica dei pulsanti del telecomando.

//	Configurazione dei dip switch del telecomando		
//	OFF-OFF	ON-OFF	ON-ON
//	Codice di ritorno		
Modalità 1			
Canale 1	9	17	25
Canale 2	10	18	26
Canale 3	11	19	27
Canale 4	12	20	28
Canale 5	13	21	29
Canale 6	14	22	30
Modalità 2			
Canale 1	9	17	25
Canale 2	10	18	26
Canale 3	11	19	27
Canale 4	12	20	28
Canale 5	13	21	29
Canale 6	113	177	241
Canale 7	114	178	242
Canale 8	115	179	243
Canale 9	116	180	244
Canale 10	117	181	245

Modalità 3			
Canale 1	**9**	**17**	**25**
Canale 2	**10**	**18**	**26**
Canale 3	**11**	**19**	**27**
Canale 4	**12**	**20**	**28**
Canale 5	**13**	**21**	**29**
Canale 6	**14**	**22**	**30**
Canale 11	**73**	**137**	**201**
Canale 12	**74**	**138**	**202**
Canale 13	**75**	**139**	**203**
Canale 14	**76**	**140**	**204**
Canale 15	**77**	**141**	**205**
Canale 16	**78**	**142**	**206**
Canale 21	**81**	**145**	**209**
Canale 22	**82**	**146**	**210**
Canale 23	**83**	**147**	**211**
Canale 24	**84**	**148**	**212**
Canale 25	**85**	**149**	**213**
Canale 26	**86**	**150**	**214**
Canale 31	**89**	**153**	**217**
Canale 32	**90**	**154**	**218**
Canale 33	**91**	**155**	**219**
Canale 34	**92**	**156**	**220**
Canale 35	**93**	**157**	**221**
Canale 36	**94**	**158**	**222**
Canale 41	**97**	**161**	**225**
Canale 42	**98**	**162**	**226**
Canale 43	**99**	**163**	**227**
Canale 44	**100**	**164**	**228**
Canale 45	**101**	**165**	**229**
Canale 46	**102**	**166**	**230**
Canale 51	**105**	**169**	**233**
Canale 52	**106**	**170**	**234**
Canale 53	**107**	**171**	**235**
Canale 54	**108**	**172**	**236**
Canale 55	**109**	**173**	**237**
Canale 56	**110**	**174**	**238**
Canale 61	**113**	**177**	**241**
Canale 62	**114**	**178**	**242**
Canale 63	**115**	**179**	**243**
Canale 64	**116**	**180**	**244**
Canale 65	**117**	**181**	**245**
Canale 66	**118**	**182**	**246**

I codici di ritorno del telecomando PN*remote* sono dati ad 8 bit che possono essere letti con la consueta sintassi di un modulo analogico Ix:A.
Nel caso di utilizzo del telecomando "Pronto" Philips la codifica utilizzata è la codifica standard Sony a 12 bit ed il codice di ritorno deve essere ottenuto con una lettura a 16bit Ix:C.

Sul modulo sono inoltre disponibili un ingresso digitale ed una uscita generica con relè utilizzabili secondo la seguente corrispondenza:

- **ingresso ausiliario: direttiva READ indirizzo 0 READ[x,0]**
- **comando relè di appoggio: uscita n° 1 Ox.1**

Ad es. per riconoscere il comando inviato dal telecomando (con i dip switch codificati come OFF-OFF) al modulo n° 10 ed utilizzarlo per accendere e regolare delle luci si scriverà:

```
//-------------------------------------------------------------
//      Programma esempio di gestione
//      modulo PN IR con telecomando PNremote
//      in modalità 1 (6 canali differenti)
//          Moduli presenti
//          PNIR    indirizzo n. 10
//          PN8OR   indirizzo n. 1
//          PNDA    indirizzo n. 4
//-------------------------------------------------------------
DEFINE    P1          9         // codice pulsante 1
DEFINE    P2          10        // codice pulsante 2
DEFINE    P3          11        // codice pulsante 3
DEFINE    P4          12        // codice pulsante 4
DEFINE    P5          13        // codice pulsante 5
DEFINE    P6          14        // codice pulsante 6
DEFINE    Status      0         // codice lettura stato PN IR

DEFINE    Accendi     V1.1
DEFINE    Spegni      V1.2
DEFINE    Up          V1.3
DEFINE    Down        V1.4
DEFINE    Code        V2        // codice di ritorno
DEFINE    Stato       V3
DEFINE    Aux_in      V3.1

DEFINE    PNIR        10        // indirizzo modulo PNIR
DEFINE    Out1        O10.1     // uscita relè su modulo PNIR
DEFINE    Out2        O1.1      // uscita luce 1
DEFINE    Out3        O4        // uscita dimmer

Code = IPNIR;                   // leggo il codice

Accendi = (Code == P1);         // il pulsante 1 viene usato per accendere
Spegni = (Code == P2);          // il pulsante 2 viene usato per spegnere
Up = (Code == P4);              // il pulsante 4 viene usato per aumentare l'intensità
Down = (Code == P3);            // il pulsante 3 viene usato per diminuire l'intensità

Stato = READ[PNIR, Status];     // leggo lo stato dell'ingresso aux

Out1 = T[Aux_in];               // comando luce 1
Out2 = SR[Accendi, Spegni];     // comando luce 2
Out3 = DIMMER[Up, Down];        // comando luce 3
```

4.9 – Modulo PN BADGE

I moduli PN BADGE sono moduli previsti per il controllo accessi mediante tessere di riconoscimento a tecnologia transponder.
La tessera impiegata funziona mediante un campo elettromagnetico che la pone in comunicazione con il modulo PN BADGE senza bisogno di contatti fino ad una distanza di circa 4-6cm.
Il codice contenuto nella tessera viene ricevuto dal modulo PN BADGE che lo confronta con i codici abilitati precedentemente memorizzati e, in caso di esito positivo, attiva l'uscita a relè presente a bordo.
Ciascun modulo può memorizzare fino a 1024 diversi codici tessera che possono essere liberamente associati ad 8 diversi gruppi di abilitazione.

I moduli PN BADGE sono adatti per applicazioni nel settore civile e terziario e sono studiati per essere collocati all'interno di scatole da incasso tipo 503.

Ciascun modulo PN BADGE dispone a bordo delle seguenti risorse:

- **ingresso ausiliario in morsettiera**
- **relè di scambio per attivazione serratura**
- **relè di scambio multifunzione (uscita generica, luce di cortesia, abilitazione utenze camera)**

L'antenna frontale è disponibile in due diverse realizzazioni per applicazioni generiche (tipo G) e personalizzabile (tipo H).
In entrambe le realizzazioni sul frontale è presente un led giallo di presenza tensione/decodifica tessera ed un led verde a disposizione utente.
Nella versione tipo H sul frontale è disponibile un' ulteriore uscita led rossa a disposizione dell'utente ed un pulsante di ingresso generico (campanello ingresso o altro).

Per il modulo di controllo accessi sono possibili due distinte modalità operative.

Funzionamento locale
In tale modalità il modulo PN BADGE si comporta come una comune unità di controllo accessi indipendente.
E' la modalità di funzionamento più comune e più semplice perché non comporta per il master nessuna elaborazione.

In tale modalità di funzionamento è comunque possibile in maniera centralizzata dall'unità Master effettuare le seguenti operazioni:

- **cancellazione di tutti i codici tessera memorizzati su tutti i moduli PN BADGE presenti in rete**
- **cancellazione di un particolare codice tessera memorizzato su tutti i moduli presenti in rete**
- **cancellazione di tutti i codici tessera memorizzati su un particolare modulo PN BADGE**
- **cancellazione di un particolare codice tessera memorizzato su un particolare modulo PN BADGE**
- **memorizzazione di un nuovo codice tessera**
- **lettura codici tessera memorizzati**
- **lettura codice ultima tessera transitata**
- **abilitazione/disabilitazione di gruppo**

Tale modalità operativa è quella che il modulo assume alla prima accensione o in caso di avaria del bus PIC*net*.

In questa modalità se al modulo viene presentata una tessera il cui codice risulta memorizzato nel modulo stesso e che appartiene ad un gruppo abilitato, il modulo provvede ad eccitare il relè serratura. Tale attivazione può essere di tipo bistabile (alla prima lettura tessera si attiva il relè, alla successiva si disattiva, ecc.) o monostabile con un tempo di rilascio programmabile.

Il secondo relè dispone di quattro distinte modalità di funzionamento:

a) in questa modalità il relè è una generica uscita digitale comandabile dal master

b) in questa modalità il relè è previsto per la gestione di una luce di cortesia: al passaggio di una tessera valida, il relè viene eccitato per un tempo programmabile (indipendente dal tempo di eccitazione del relè serratura) in modo da permettere l'accensione di una luce temporizzata per agevolare l'accesso al locale

c) in questa modalità il relè è previsto per la gestione della forza motrice di camera: inserendo la tessera in una tasca porta-badge (il cui contatto di uscita andrà collegato all'ingresso libero in morsettiera), il relè viene eccitato e rimarrà in questo stato fintanto che l'ingresso in morsettiera risulta chiuso. All'estrazione della tessera (e quindi all'apertura dell'ingresso) il relè rimarrà eccitato per il tempo programmato in modo da permettere all'ospite l'uscita dalla camera e poi si disattiverà in modo da lasciare la camera non occupata priva di tensione.

d) in questa modalità il relè è previsto per la gestione della forza motrice di camera: al passaggio di una tessera valida, il relè viene eccitato per un tempo programmabile (indipendente dal tempo di eccitazione del relè serratura) in modo da attivare le funzioni di camera e permettere all'ospite l'accesso alla camera e l'inserimento della tessera in una tasca porta-badge il cui contatto di uscita andrà collegato all'ingresso libero in morsettiera.
Il relè rimarrà dunque eccitato fintanto che l'ingresso in morsettiera risulta chiuso. All'estrazione della tessera (e quindi all'apertura dell'ingresso) il relè rimarrà eccitato per il tempo programmato in modo da permettere all'ospite l'uscita dalla camera e poi si disattiverà in modo da lasciare la camera non occupata priva di tensione.

Funzionamento remoto

Forzando il funzionamento in remoto del modulo, il master si sostituisce al modulo di controllo nel pilotaggio del relè di serratura che può dunque essere attivato come una qualsiasi uscita digitale per permettere la realizzazione di funzioni più sofisticate (ad es. per consentire l'accesso a certe tessere in determinate fasce orarie, disabilitare un codice tessera allo scadere di un abbonamento, funzione anti pass-back, ecc.).

Le risorse del modulo PN BADGE vengono gestite in parte mediante comandi dedicati ed in parte come un normale modulo dotato di uscite digitali secondo la seguente corrispondenza:

- relè REL2 multifunzione:	**uscita n° 1 (1=ON)**	**Ox.1**
- led verde:	**uscita n° 2 (1=ON)**	**Ox.2**
- led rosso:	**uscita n° 3 (1=ON)**	**Ox.3**
- relè REL1 (serratura):	**uscita n° 4 (1=ON)**	**Ox.4**
- modalità locale/remoto REL1:	**uscita n° 5 (locale = 0; remoto = 1)**	**Ox.5**

essendo x l'indirizzo del modulo.

Le diverse impostazioni di funzionamento sono impostabile mediante la direttiva SETPAR (vedi § 3.14) ai seguenti indirizzi:

- modalità REL1:	**indirizzo 13** **monostabile = 0; bistabile = 1**	**[default = 0]**
- tempo attivazione REL1:	**indirizzo 12** **valore compreso tra 0 e 255 espresso in decimi di secondi**	**[default = 5]**
- modalità REL2:	**indirizzo 11** **relè generico = 0;** **luce cortesia = 1;** **FM camera = 2;** **luce cortesia e FM camera = 3;**	**[default = 0]**
- tempo attivazione REL2:	**indirizzo 10** **valore compreso tra 0 e 255 espresso in secondi**	**[default = 30]**

Lo stato dei relè e delle altre condizioni di funzionamento possono essere letti mediante la direttiva READ (vedi § 3.14) con la seguente corrispondenza:

- **codice ultima tessera letta: indirizzo 2**
- **stato risorse: indirizzo 0**
  - **ingresso ausiliario: bit 1 (0 = OFF, 1 = ON)**
  - **pulsante frontale: bit 2 (0 = OFF, 1 = ON)**
  - **stato relè REL1: bit 3 (0 = OFF, 1 = ON)**
  - **stato relè REL2: bit 4 (0 = OFF, 1 = ON)**

I diversi codici tessera memorizzati nei moduli PN BADGE possono essere suddivisi in distinti gruppi di abilitazione in modo da permettere facilmente all'unità master l'attivazione o disattivazione di tutte le tessere appartenenti ad un particolare gruppo in funzione ad es. di un consenso orario.

Tale gestione viene effettuata mediante l'istruzione "CARDGROUP" secondo la seguente sintassi:

CARDGROUP [*addr, group, flag, cond*];

dove: *addr* : indirizzo del modulo
**
group : numero del gruppo di tessere interessato compreso tra 1 e 8 (in rappresentazione binaria: un bit per ogni gruppo)**
**
flag : flag per l'abilitazione/disabilitazione del gruppo; può assumere solo i valori ON o OFF**
**
cond : condizione logica di consenso all'impostazione (funziona sul fronte del segnale)**

Le tessere possono essere suddivise in 8 diversi gruppi di appartenenza e con uno stesso comando CARDGROUP possono essere attivati/disattivati più gruppi contemporaneamente specificando il bit corrispondente nel campo *group*.

Es. :

```
CARDGROUP [10, 00000101b, ON, V1.1];
```

Equivalente a:

```
CARDGROUP [10, 5, ON, V1.1];
```

Le tessere appartenenti ai gruppi 1 e 4 memorizzate nel modulo Pnbadge n. 10 vengono abilitate da un fronte sulla variabile V1.1.

Es. :

```
CARDGROUP [10, 11111111b, ON, V1.1];
```

Equivalente a:

```
CARDGROUP [10, 255, OFF, V1.1];
```

Le tessere appartenenti a tutti i gruppi memorizzate nel modulo Pnbadge n. 10 vengono disabilitate da un fronte sulla variabile V1.1.

Ad es. si voglia segnalare a distanza l'ingresso di una persona in un locale ed abilitare un gruppo di tessere in funziona di una programmazione oraria.

```
//-------------------------------------------------------------
//      Programma esempio di gestione
//      modulo PN BADGE
//        Moduli presenti
//        PN BADGE               indirizzo n. 11
//        PN 8OR                 indirizzo n. 1
//-------------------------------------------------------------
DEFINE    Ufficio     11              // indirizzo del PN BADGE ufficio
DEFINE    Impiegati   2               // codice del gruppo tessere impiagati
DEFINE    Pulizie     4               // codice del gruppo tessere addetti pulizia
DEFINE    Status      0               // codice lettura stato PN BADGE

DEFINE    Lampeggio   O1.1

DEFINE    Clock_Imp   V1.1
DEFINE    Clock_Pul   V1.2
DEFINE    Start_Imp   V1.3
DEFINE    Stop_Imp    V1.4
DEFINE    Start_Pul   V1.5
DEFINE    Stop_Pul    V1.6

DEFINE    Stato       V2
DEFINE    Aux_in      V2.1
DEFINE    Campanello  V2.2
DEFINE    Serratura   V2.3
DEFINE    Rel_aux     V2.4

Clock_Imp = UPRG[“Orario impiegati”, Lu-Ve, 8:30, 18:30];
Start_Imp = SHOT[1, Clock_Imp];        // genero un impulso all’inizio del periodo
Stop_Imp = SHOT[1, !Clock_Imp];        // genero un impulso alla fine del periodo

Clock_Pul = UPRG[“Orario pulizie”, Lu-Ve, 18:30, 20:30];
Start_Pul = SHOT[1, Clock_Pul];        // genero un impulso all’inizio del periodo
Stop_Pul = SHOT[1, !Clock_ Pul];       // genero un impulso alla fine del periodo

CARDGROUP [Ufficio, Impiegati, ON, Start_Imp];
CARDGROUP [Ufficio, Impiegati, OFF, Stop_Imp];
CARDGROUP [Ufficio, Pulizie, ON, Start_Pul];
CARDGROUP [Ufficio, Pulizie, OFF, Stop_ Pul];

Stato = READ[Ufficio, Status];                      // leggo lo stato del lettore

Lampeggio = PWM[5, 5, SHOT[100, Serratura]];        // segnalo ingresso per 10 secondi
```

4.10 – Modulo PN PAN

I moduli PN PAN sono moduli di ingresso a pulsante con segnalazione luminosa.
Il modulo trova applicazione prevalentemente in impianti civili e del terziario (abitazioni, uffici, alberghi, ecc.) e si compone di una scheda da incasso in scatola 503 ed un pannello frontale dove sono alloggiati 8 pulsanti a membrana ed i corrispondenti led di segnalazione.
Il modulo viene gestito come un normale modulo dotato di 8 ingressi digitali (i pulsanti) ed 8 uscite digitali (i led di segnalazione) tra loro indipendenti e che vengono gestiti con i consueti comandi di lettura e scrittura.
Unica particolarità consiste nel funzionamento dei led di segnalazione per i quali è possibile scegliere tra un funzionamento ad accensione fissa o lampeggiante.
L'accensione fissa si ottiene attivando il corrispondente bit di uscita (bit 1-8), mentre per l'accensione lampeggiante occorre settare anche il corrispondente bit della parte alta (bit 9-16).

L'ulteriore ingresso ausiliario presente in morsettiera corrisponde al bit 9 (bit 1 della parte alta).

Ad es. si voglia comandare l'apertura di un cancello segnalando il suo stato secondo la seguente convenzione:

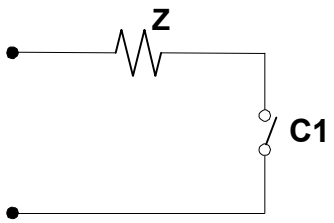
led spento :	**cancello chiuso**
led acceso :	**cancello aperto**
led lampeggiante :	**cancello in movimento**

```
//-------------------------------------------------------------
//      Programma esempio di gestione
//      cancello con modulo PN PAN
//         Moduli presenti
//         PN PAN                  indirizzo n. 1
//         PN4I4O                  indirizzo n. 2
//-------------------------------------------------------------
DEFINE     Apri        I1.1      // pulsante 1 pannello
DEFINE     Ferma       I1.2      // pulsante 2 pannello
DEFINE     Aperto      O1.1      // led 1 pannello accensione fissa
DEFINE     Fermato     O1.2      // led 2 pannello accensione fissa
DEFINE     Aprendo     O1.9      // led 1 pannello accensione lampeggiante

DEFINE     FC_chiuso   I2.1      // fine corsa chiusura
DEFINE     FC_aperto   I2.2      // fine corsa apertura
DEFINE     Apertura    O2.1      // comando relè apertura
DEFINE     Stop        O2.2      // comando relè stop

Apertura = Apri;
Stop = Ferma;

Aperto = !FC_chiuso;
Aprendo = Aperto & !FC_aperto;
Fermato = T[Ferma];
```

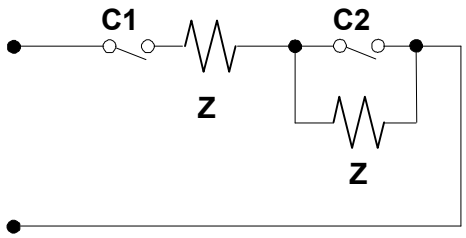
4.11 - Modulo di ingresso a controllo di impedenza PN8IBIL

I moduli PN 8IBIL sono moduli da guida DIN previsti per il controllo di impedenza di 8 ingressi analogici. Ogni ingresso può essere gestito in modalità BIL (ad impedenza bilanciata), oppure 2 BIL (ad impedenza con doppio bilanciamento) a seconda della programmazione effettuata (per maggiori dettagli sul funzionamento vedi Manuale di Installazione).

Circuito 1



Circuito 2



In condizione di riposo (contatti C1 e C2 chiusi e circuito integro) il led relativo all'ingresso risulta spento ed il corrispondente bit a 0.

Diversamente si possono avere i seguenti due casi:

Circuito 1

- **Se l'impedenza Z viene cortocircuitata, il corrispondente led risulta acceso ed il bit relativo (bit 1-8) risulta settato.**
- **Se il contatto C1 viene aperto, il corrispondente led risulta lampeggiante e risultano settati sia il bit relativo (bit 1-8) che il corrispondente bit della parte alta (bit 9-16).**

Circuito 2

- **Se l'impedenza Z viene cortocircuitata o viene aperto il contatto C2, il corrispondente led risulta acceso ed il bit relativo (bit 1-8) risulta settato.**
- **Se il contatto C1 viene aperto, il corrispondente led risulta lampeggiante e risultano settati sia il bit relativo (bit 1-8) che il corrispondente bit della parte alta (bit 9-16).**

4.12 – Modulo PN DIGITERM

Il modulo PN DIGITERM è particolarmente indicato per applicazioni in ambito alberghiero e consente la termoregolazione e la completa gestione dell'automazione camera (fatta eccezione per il controllo accessi) e dunque permette di affrontare impianti alberghieri anche complessi con consistenti economie.

Il modulo dispone nel complesso delle seguenti risorse:

INGRESSI
n. 7 ingressi optoisolati riferiti alla tensione di bus:

- **Tirante bagno**
- **Contatto porta**
- **Contatto finestra**
- **Tasca porta badge**
- **N. 3 ingressi programmabili**

n. 2 ingressi analogici per sonda NTC per rilievo temperatura ambiente camera e bagno

USCITE
n. 7 uscite relè:

- **Velocità fan coil minima**
- **Velocità fan coil media**
- **Velocità fan coil massima**
- **Luce di cortesia**
- **Abilitazione FM camera**
- **Elettrovalvola fan coil camera**
- **Elettrovalvola bagno**

n. 2 uscite analogiche per elettrovalvola proporzionale 0-10V camera e bagno

Le risorse del modulo PN DIGITERM vengono gestite in parte mediante comandi dedicati ed in parte come un normale modulo dotato di ingessi ed uscite digitali secondo la seguente corrispondenza:

Uscite fisiche

- **velocità 1 fan-coil:**	**uscita n° 1 (1=ON)**	**non accessibile**
- **velocità 2 fan-coil:**	**uscita n° 2 (1=ON)**	**non accessibile**
- **velocità 3 fan-coil:**	**uscita n° 3 (1=ON)**	**non accessibile**
- **luce di cortesia:**	**uscita n° 4 (1=ON)**	**Ox.4 (se abilitata)**
- **abilitazione Forza Motrice:**	**uscita n° 5 (1=ON)**	**Ox.5 (se abilitata)**
- **valvola camera:**	**uscita n° 6 (1=ON)**	**Ox.6 (se abilitata)**
- **valvola bagno:**	**uscita n° 7 (1=ON)**	**Ox.7 (se abilitata)**

Uscite logiche

- **ingresso ospite:**	**uscita n° 9 (1 = ospite entrato)**	**Ox.9**
- **estate/inverno:**	**uscita n° 10 (0 = Estate, 1 = Inverno)**	**Ox.10**
- **camera sfitta:**	**uscita n° 14 (1 = camera sfitta)**	**Ox.14**
- **ospite presente:**	**uscita n° 15 (1 = ospite presente)**	**Ox.15**

Ingressi fisici

- **allarme bagno:**	**ingresso n° 1**	**Ix.1**
- **contatto finestra:**	**ingresso n° 2**	**Ix.2**
- **tasca porta badge:**	**ingresso n° 3**	**Ix.3**
- **contatto porta:**	**ingresso n° 4**	**Ix.4**
- **ingresso libero 1:**	**ingresso n° 5**	**Ix.5**
- **ingresso libero 2:**	**ingresso n° 6**	**Ix.6**
- **ingresso libero 3:**	**ingresso n° 7**	**Ix.7**

Ingressi logici

- **lettura stato non disturbare: ingresso n° 9 Ix.9**
- **lettura stato camera riassettata: ingresso n° 10 Ix.10**
- **lettura stato allarme bagno: ingresso n° 11 Ix.11**
- **lettura stato camera sfitta/prenotata: ingresso n° 12 Ix.12**
- **lettura stato apertura prolungata porta e/o finestra: ingresso n° 13 Ix.13**
- **lettura stato giorno/notte: ingresso n° 16 Ix.16**

essendo x l'indirizzo del modulo.

Le diverse impostazioni di funzionamento sono invece gestite mediante la direttiva SETPAR (per la scrittura) o READ (per la lettura) (vedi § 3.14) ai seguenti indirizzi:

Grandezza	*Accesso*	*Indirizzo*	*Range*	*Default*
Temperatura camera [0.1 °C]	Lettura	1	0-500	- -
Setpoint temperatura camera [0.1 °C]	Lettura/scrittura	2	50-350	- -
Temperatura bagno [0.1 °C]	Lettura	3	0-500	- -
Uscita analogica canale A	Lettura/scrittura	4	0-255	0
Uscita analogica canale B	Lettura/scrittura	5	0-255	0
Limite massimo setpoint camera [0.1 °C]	Lettura/scrittura	6	50-350	350
Limite minimo setpoint camera [0.1 °C]	Lettura/scrittura	7	50-350	50
Setpoint camera con ospite assente [0.1 °C]	Lettura/scrittura	8	0-500	170
Setpoint camera con camera sfitta [0.1 °C]	Lettura/scrittura	9	0-500	150
Tempo accensione luce di cortesia [sec]	Lettura/scrittura	12	0-255	30
Tempo permanenza FM camera dopo uscita ospite [sec]	Lettura/scrittura	13	0-255	60
Offset negativo temperatura camera [0.1 °C]	Lettura/scrittura	14	0-100	0
Offset negativo temperatura bagno [0.1 °C]	Lettura/scrittura	15	0-100	0
Attesa da apertura finestra per spegnim. fan-coil [sec]	Lettura/scrittura	16	0-255	60
Attesa da apertura porta per spegnim. fan-coil [sec]	Lettura/scrittura	17	0-255	120
Gestione ingresso contatto finestra	Lettura/scrittura	18	0: considera 1: ignora	0
Gestione ingresso contatto porta	Lettura/scrittura	19	0: considera 1: ignora	1
Selezione ingressi attivi alti o bassi	Lettura/scrittura	20	0-255 1: attivo alto	245=ingressi 2 e 4 attivi bassi

Grandezza	*Accesso*	*Indirizzo*	*Range*	*Default*
Attivazione forzatura velocità fan-coil	Lettura/scrittura	22	0: disattivata 1: attivata	0
Velocità forzatura fan-coil	Lettura/scrittura	23	0-3	0
Attiva set-point bagno separato dal set-point camera	Lettura/scrittura	25	0: set-point diverso 1: set-point comune	1
Set-point bagno [0.1 °C]	Lettura/scrittura	26	0-500	18
Gestione contatti porta e finestra per la termoregolazione bagno	Lettura/scrittura	27	0: gestisci i sensori 1: ignora i sensori	0
Offset aumento temperatura per la visualizzazione display camera [0.1 °C]	Lettura/scrittura	28	0-100	0
Modalità visualizzazione display di camera	Lettura/scrittura	29	0: mostra l'orario 1: mostra temperatura in °C 3: mostra temperatura in °F	0
Velocità fan-coil selezionata sul pannello camera	Lettura/scrittura	30	0-3	1
Presenza pannello diplay di camera	Lettura/scrittura	31	1: pannello assente 0: pannello presente	0
Attivazione set-point temperatura extra	Lettura/scrittura	33	0: ignora 1: attiva	0
Set-point temperatura extra [0.1 °C]	Lettura/scrittura	34	0-500	- -
Velocità attuale fan-coil	Lettura	37	0-3	- -
Impostazione n. massimo velocità fan-coil	Lettura/scrittura	38	0-3	3
Lettura stato uscite relè 1-7	Lettura	39	0: OFF - 1:ON	- -
Allarme temperatura inferiore a 5° o superiore a 40°	Lettura	40	0: OK 1: fault camera 2 : fault bagno 3 : fault entrambe	- -
Impostazione luce di cortesia	Lettura/scrittura	41	0 : solo cortesia 1 : anche luce camera	0
Funzionamento ingresso n. 5 con registro n.41 a 1	Lettura/scrittura	42	1: passo-passo 0: comando mantenuto	1
Presenza ospite da remoto	Lettura/scrittura	47	0: disabilita 1: abilita	0
Impostazione presenza ospite (con registro n.47 a 1)	Lettura/scrittura	48	0: ospite assente 1: ospite presente	- -
Isteresi inferiore regolazione temperatura [0.1 °C]	Lettura/scrittura	56	0-50	3
Isteresi superiore regolazione temperatura [0.1 °C]	Lettura/scrittura	57	0-50	3
Illuminazione display pannello camera	Lettura/scrittura	58	0: sempre acceso 1: temporizzato	0
Ritardo spegnimento display dopo pressione pulsanti [ms]	Lettura/scrittura	59	0-300.000	5.000
Intensità diurna display	Lettura/scrittura	60	20(min)-1023(max)	1.023
Intensità notturna display	Lettura/scrittura	61	20(min)-1023(max)	50

Impostazione range impostazione temperatura	**Lettura/scrittura**	**62**	**0: non attivo 1: utilizza valori registri 6-7**	**0**
Tempo filtraggio ingressi digitali [ms]	**Lettura/scrittura**	**63**	**0-65.000**	**40**
Abilitazione comando manuale uscite Ox.4-Ox.7	**Lettura/scrittura**	**77**	**0: disabilitato 1: abilitato**	**01111000b**

Registri per la gestione lettore PN BADGE D.

Abilitazione PN BADGE D	**Lettura/scrittura**	**49**	**0: disabilita 1: abilita**	**0**
Tempo chiusura relé serratura in decimi di secondo	**Lettura/scrittura**	**50**	**0-100**	**15**
Abilitazione gruppi tessere	**Lettura/scrittura**	**51**	**0: disabilita gruppo 1: abilita gruppo Bit 1-8: gruppo 1-8**	**255**
Ultimo codice tessera letto	**Lettura**	**52**	**0-65534**	**- -**
Validità ultima tessera letta	**Lettura**	**53**	**0: non valida 1: valida**	**- -**
Cancellazione codici tessera	**Scrittura**	**55**	**1: cancella codici**	**- -**
Gruppo ultima tessera letta	**Lettura**	**74**	**1-8**	**- -**

Registri per la gestione pannello fuori porta PN DSP.

Attivazione pannello fuori camera	**Lettura/scrittura**	**45**	**0: disabilitato 1: abilita in autom. 2: abilita in manuale**	**0**
Stato pannello fuori camera: bit 1 = led verde bit 2 = led rosso bit 3 = led giallo bit 4 = pulsante campanello bit 5 = lampeggio led verde bit 6 = lampeggio led rosso bit 7 = lampeggio led giallo bit 8 = pulsazione led giallo	**Lettura/scrittura**	**46**	**0: non attivo 1: attivato**	**1000 0000b**

Nel seguito si riportano alcune particolarità della logica di funzionamento del sistema.

GESTIONE INGRESSI

Verifica entrata ospite in camera

L'entrata ospite in camera viene individuata dall'attivazione dell'uscita virtuale Ox.9 o tramite passaggio badge se si utilizza un PN BADGE D. L'attivazione dell'uscita Ox.9 dovrà essere prevista da programma utente in quei casi in cui non è installato il lettore di badge PN BADGE D, ma è presente o un altro modello di lettore di badge o un contatto sulla porta, che individua se è avvenuto.

Se si verifica un ingresso il sistema effettua le seguenti operazioni:

- **inizializzazione del timer che accende la luce di cortesia con un tempo impostabile tramite registro #12**
- **inizializzazione del timer che mantiene accesa la forza motrice della camera con un tempo impostabile tramite registro #13 in modo tale che se l'ospite non resta in camera, la forza motrice si disattiva dopo questo periodo**
- **verifica presenza ospite in camera: se nessun ospite è ancora presente in camera avviene l'impostazione della velocità del fan coil al minimo e viene attivato il flag di presenza in camera.**

Verifica dell'ingresso tasca porta badge

- **se l'ingresso si attiva, quindi il badge è inserito, avviene l'impostazione della velocità del fan coil al minimo e viene attivato il flag di presenza in camera (per ovviare ad una eventuale mancata segnalazione di ingresso dalla verifica entrata ospite in camera)**
- **finchè l'ingresso permane attivo (badge inserito) il timer forza motrice viene mantenuto resettato in modo tale da mantenere attiva la forza motrice per tutta la permanenza in camera**
- **se l'ingresso si disattiva (badge rimosso) viene impostata la velocità del fan coil al minimo e resettati i flag di presenza ospite in camera e quello di non disturbare eventualmente attivo.**

Nota: l'ingresso tasca porta badge può essere simulato dall' uscita virtuale Ox.15 previa abilitazione mediante il registro #47.

Verifica dell'ingresso di comando illuminazione camera (ingresso 5)

Se la modalità che prevede che la luce di cortesia sia anche la luce camera è abilitata tramite il registro #41, viene verificata la modalità di funzionamento di questo ingresso (passo-passo o comando mantenuto), impostata tramite il registro #42, e viene comandata di conseguenza l'uscita luce di cortesia.
Se tale modalità non è abilitata l'uscita viene mantenuta attiva fino a quando il timer luce di cortesia è diverso da zero a meno che non sia impostato il funzionamento in manuale tramite registro #77.

Verifica dell'ingresso contatto finestra

Se la gestione di questo ingresso è abilitata tramite il registro #18, viene conteggiata la durata dello stato attivo e se questa supera il tempo impostato tramite il registro #16, allora la termoregolazione della camera viene interrotta e viene attivato un flag leggibile tramite l'ingresso virtuale Ix.13. Questo flag può essere utilizzato per la visualizzazione della condizione di finestra sia sul sw di supervisione che sulll'eventuale pannello o lettore di badge fuori porta.

Verifica dell'ingresso contatto porta

Se la gestione di questo ingresso è abilitata tramite il registro #19, ogni volta che l'ingresso assume uno stato attivo viene accesa la luce di cortesia (se abilitata) e mantenuta accesa dopo la chiusura per un tempo impostabile tramite il registro #12.
Se inoltre l'ingresso si mantiene attivo per un tempo superiore alla soglia impostata tramite il registro #17, allora la termoregolazione della camera viene interrotta e viene attivato un flag leggibile tramite l'ingresso virtuale Ix.13.

GESTIONE CLIMATIZZAZIONE

Gestione set-point camera e bagno

- **Se l'ospite è presente in camera allora il set-point di camera è quello impostato tramite il pannello utente salvo limitazioni di minimo e massimo definite tramite i registri #6 e #7 rispettivamente. Quando l'ospite entra per la prima volta nella camera il valore del set-point sarà quello corrispondente al valore di temperatura "Camera occupata" nel sw di supervisione.**

- **Se l'ospite non è presente, ma la camera è prenotata, il set-point di camera è quello impostato tramite il registro #8 corrispondente al valore di temperatura "Camera libera" nel sw di supervisione.**

- **Se è attiva la forzatura del set-point da remoto tramite il registro #33, allora il set-point di camera diventa quello impostato tramite il registro #34 corrispondente al valore di "Temperatura forzata" nel sw di supervisione.**
  Nota: questa impostazione prevale sulle precedenti.

- **Se la camera è impostata come sfitta tramite l'uscita virtuale Ox.14 allora il set-point di camera diviene quello impostato tramite il registro #9 corrispondente al valore di temperatura "Camera sfitta" nel sw di supervisione.**
  Nota: questa impostazione prevale sulle precedenti.

- **Se tramite il registro #25 non è attiva l'impostazione che prevede che il bagno utilizzi lo stesso set-point della camera, allora il set-point del bagno sarò quello impostato tramite il registro #26, altrimenti verrà seguito quello della camera.**

Logica di gestione della climatizzazione

Il parametri che governo la logica di gestione sono:
- **la stagione impostata tramite l'uscita virtuale Ox.10**
- **le componenti inferiore e superiore dell'isteresi impostate tramite i registri #56 e #57**

Modalità inverno
Se la temperatura rilevata in camera è maggiore del set-point sommato alla componente superiore dell' isteresi allora il sistema setta il flag temperatura raggiunta, disattiva il condizionamento e imposta la velocità del fan coil al minimo (relè spenti e velocità minima predisposta sul display); se la temperatura misurata è minore del set-point con sottratta la componente inferiore dell'isteresi allora resetta il flag di temperatura raggiunta e attiva il condizionamento.

Modalità estate
Se la temperatura rilevata in nella camera è maggiore del set-point sommato alla componente superiore dell'isteresi viene attivato il condizionamento; se la temperatura misurata è minore del set-point con sottratta la componente inferiore dell'isteresi allora viene disattivato il fan-coil e la sua velocità viene impostata al minimo sul pannello di comando.

Le uscite adibite al controllo della valvola camera e bagno vengono comandate rispettivamente in funzione della velocità del fan coil (per la camera) e dello stato di temperatura raggiunta (per il bagno), salvo impostazione manuale tramite registro #77.

Se l'ospite non è presente in camera la velocità fan-coil viene selezionata in automatico in base alla differenza di temperatura ∆T tra valore misurato e valore impostato:

- **∆T < 2° à velocità minima**
- **2° < ∆T <4° à velocità media**
- **∆T > 4° à velocità massima**

Al rientro dell'ospite la velocità viene riportata al minimo e il set-point all'ultimo valore impostato dall'utente (utente uscito e rientrato) o al default di camera occupata (camera sfitta e poi prenotata).

Esempio gestione alberghiera con modulo PN DIGITERM e PN BADGE H:

```
//--------------------------------------
//    Gestione camere albergo
//--------------------------------------

//--------------------------------------
// DEFINIZIONI costanti
//--------------------------------------
DEFINE  ROOM_TEMP               1         // Temperatura camera
DEFINE  DPL_THRESHOLD           2         // Imposta threshold su display
DEFINE  MAX_THRESHOLD           6         // Massimo threshold accettato
DEFINE  MIN_THRESHOLD           7         // Minimo threshold acettato
DEFINE  ECONOMY_THRESHOLD       8         // Temperatura ospite assente
DEFINE  BATH_ALARM              21        // Allarme bagno
DEFINE  DPL_SPEED               30        // leggi la vel del fan coil 0..3

//-------------------------
//    Camera 101
//-------------------------
DEFINE  Digi_101          1
DEFINE  Bdg_101           2
DEFINE  Temp_101          V1
DEFINE  Set_101           V2
DEFINE  Virt_101          V3
DEFINE  Non_dist_101      V4
DEFINE  Vel_Fan_101       V5
DEFINE  All_WC_101        V6
DEFINE  Pulita_101        V7
DEFINE  Bdg_stato_101     V8
DEFINE  Prenota_101       V9

//-------------------------
//    Camera 102
//-------------------------
DEFINE  Digi_102          3
DEFINE  Bdg_102           4
DEFINE  Temp_102          V11
DEFINE  Set_102           V12
DEFINE  Virt_102          V13
DEFINE  Non_dist_102      V14
DEFINE  Vel_Fan_102       V15
DEFINE  All_WC_102        V16
DEFINE  Pulita_102        V17
DEFINE  Bdg_stato_102     V18
DEFINE  Prenota_102       V19

//-------------------------------
//      Definizione variabili
//-------------------------------
DEFINE  Estate            V1000.1
DEFINE  VCambioT          V1000.2
DEFINE  CambioT           V1000.3
DEFINE  Start             V1000.4
DEFINE  Cumulat_allarmi   V1000.5
DEFINE  Min_set           V1001
DEFINE  Max_set           V1002
DEFINE  Eco_set           V1003
DEFINE  Min_set_old       V1004
DEFINE  Max_set_old       V1005
DEFINE  Eco_set_old       V1006
DEFINE  Seq               V1007
DEFINE  Real_Set          V1008
DEFINE  Sfitta_set        V1009
DEFINE  Sfitta_set_old    V1010
```

```
DEFINE  Min_set_I         V1011
DEFINE  Max_set_I         V1012
DEFINE  Eco_set_I         V1013
DEFINE  Sfitta_set_I      V1014
DEFINE  Min_set_E         V1015
DEFINE  Max_set_E         V1016
DEFINE  Eco_set_E         V1017
DEFINE  Sfitta_set_E      V1018
DEFINE  PC_Min_set_I      V1019
DEFINE  PC_Max_set_I      V1020
DEFINE  PC_Eco_set_I      V1021
DEFINE  PC_Sfitta_set_I   V1022
DEFINE  PC_Min_set_E      V1023
DEFINE  PC_Max_set_E      V1024
DEFINE  PC_Eco_set_E      V1025
DEFINE  PC_Sfitta_set_E   V1026

//--------------------------------
//      Definizione macro camera
//--------------------------------
MACRO DigiCamera (N_Term,N_Bdg,S_Term,S_Badge,M_Temp,M_Set,M_fan,M_WC,M_clean)
{
     S_Badge=READ[N_Bdg,0];                 // stato del badge
     ON_Bdg.2=IN_Term.11;                   // rosso
     ON_Bdg.3=(IN_Term.2&!IN_Term.9)|PWM[10,10,IN_Term.9];     // giallo
     ON_Term.9=SHOT[10,S_Badge.3];          //entro in camera se relé serratura chiude
     ON_Term.10=Estate;                     //est/inv term
     SETPAR[N_Term, BATH_ALARM,0,M_WC.3];   // reset allarme bagno
     SETPAR[N_Term, MIN_THRESHOLD,Min_set,CambioT];           // soglia cliente presente
     SETPAR[N_Term, MAX_THRESHOLD,Max_set,CambioT];           // soglia cliente presente
     SETPAR[N_Term, ECONOMY_THRESHOLD,Eco_set,CambioT];       // soglia cliente fuori

     M_Temp=READ[N_Term, ROOM_TEMP];        // leggo temp
     M_Set=READ[N_Term, DPL_THRESHOLD];     // leggo impostata
     M_fan=READ[N_Term, DPL_SPEED];         // leggo la vel del fan impostata
     S_Term.1=IN_Term.9;      // copio il non dist in una virt per il pc
     IF[M_WC.3]
          ON_Term.13=OFF;     // reset allarme bagno
          M_WC.3=OFF;
     ENDIF
     M_WC.1=IN_Term.11;       // copio lo stato dell'all wc in una virt per il pc
     M_clean.1=IN_Term.10;    // copio lo stato della camera pulita in una virt per il pc
     IF[M_clean.3]
          ON_Term.12=OFF;     // reset camera pulita
          M_clean.3=OFF;
     ENDIF
}
```

```
//-------------------------------------
//     programma utente
//-------------------------------------
INIT Start=OFF;

// Verifico se sono state cambiate delle temp per rimandarle ai term
VCambioT=(Min_set_old!=Min_set)|(Max_set_old!=Max_set)|(Eco_set_old!=Eco_set)|(Sfitta_set
_old!=Sfitta_set);
CambioT=SHOT[10,VCambioT];

IF[CambioT]
     Min_set_old=Min_set;
     Max_set_old=Max_set;
     Eco_set_old=Eco_set;
     Sfitta_set_old=Sfitta_set;
ENDIF

IF[Estate]
     Sfitta_set=Sfitta_set_E;
     Eco_set=Eco_set_E;
     Max_set=Max_set_E;
     Min_set=Min_set_E;
ELSE
     Sfitta_set=Sfitta_set_I;
     Eco_set=Eco_set_I;
     Max_set=Max_set_I;
     Min_set=Min_set_I;
ENDIF

//-------------------------------------------------------------------------------------
//  converto i valori in gradi messi da PC in valori per il digiterm
//-------------------------------------------------------------------------------------
Min_set_I = PC_Min_set_I*10;
Max_set_I = PC_Max_set_I*10;
Eco_set_I = PC_Eco_set_I*10;
Sfitta_set_I = PC_Sfitta_set_I*10;
Min_set_E = PC_Min_set_E*10;
Max_set_E = PC_Max_set_E*10;
Eco_set_E = PC_Eco_set_E*10;
Sfitta_set_E = PC_Sfitta_set_E*10;

DigiCamera (Digi_101, Bdg_101, Non_dist_101, Bdg_stato_101, Temp_101, Set_101,
Vel_Fan_101, All_WC_101, Pulita_101);
DigiCamera (Digi_102, Bdg_102, Non_dist_102, Bdg_stato_102, Temp_102, Set_102,
Vel_Fan_102, All_WC_102, Pulita_102);

Start=ON;
```

4.13 – Modulo PN VIS

Il modulo PN VIS consente la visualizzazione alfanumerica su un display a tre cifre a led rossi ad alta intensità.
E' indicato particolarmente in impianti di chiamata ospedaliera o in case di riposo per la segnalazione degli allarmi in alternativa al modulo PN DIS che, date le sue dimensioni e forme, difficilmente si presta ad un'installazione da tavolo o a parete e non garantisce una sufficiente leggibilità a distanza.

Il modulo dispone a bordo di un uscita per un eventuale cicalino di allarme e di un ingresso per la sua tacitazione.

Le risorse del modulo PN VIS vengono gestite in parte mediante comandi dedicati ed in parte come un normale modulo dotato di ingessi ed uscite digitali secondo la seguente corrispondenza:

- **cifra numerica visualizzata: uscita a 16 bit Ox:C**
- **ingresso tacitazione: ingresso n° 1 Ix.1**

essendo x l'indirizzo del modulo.

L'uscita di servizio può essere comandata utilizzando la seguente sintassi:

WRITE[x, 0, Val];

essendo x l'indirizzo del modulo e Val il valore digitale da attribuire all'uscita.

Le diverse impostazioni di funzionamento sono impostabile mediante la direttiva SETPAR (vedi § 3.14) ai seguenti indirizzi:

- **posizione punto decimale: indirizzo 26 [default = 0]**
  0 = nessuna virgola
  1 = virgola cifra a destra (es. 344.)
  2 = virgola cifra centrale (es. 23.4)
  4 = virgola cifra a sinistra (es. 1.22)

La prima cifra a sinistra può essere utilizzata anche per rappresentare un carattere alfabetico.
Il carattere deve essere inviato mediante la direttiva SETPAR (vedi § 3.14) al seguente indirizzo:

- **prima cifra alfabetica: indirizzo 27 [default = 63]**

con la seguente corrispondenza:

"A" = 119	**"b" = 124**	**"C" = 57**	**"c" = 88**
"d" = 94	**"E" = 121**	**"F" = 113**	**"g" = 111**
"H" = 118	**"h" = 116**	**"i" = 16**	**"L" = 56**
"l" = 48	**"n" = 84**	**"O" = 63**	**"o" = 92**
"P" = 115	**"q" = 103**	**"r" = 80**	**"S" = 109**
"t" = 120	**"U" = 62**	**"u" = 28**	**"y" = 110**
"J" = 30	**"j" = 14**	**"-" = 64**	**"_" = 8**
"Space" = 0	**"dot" = 128**	**"equal" = 72**	**"zero" = 63**

Ad es. per visualizzare l'allarme proveniente da una serie di celle frigorifere, si scriverà:

```
//-------------------------------------------------------------
//      Gestione allarmi celle frigorifere
//-------------------------------------------------------------

//------------------------------
//          n. 01
//  PN 2I2O  cella 1
//------------------------------
DEFINE          Allarme_1         I1.1        // pulsante allarme cella 1
DEFINE          Reset_1           I1.2        // reset allarme cella 1
DEFINE          Buzzer_1          O1.1        // buzzer allarme cella 1
DEFINE          Luce_1            O1.2        // lampeggiante allarme cella 1

//------------------------------
//          n. 02
//  PN 2I2O  cella 2
//------------------------------
DEFINE          Allarme_2         I2.1        // pulsante allarme cella 2
DEFINE          Reset_2           I2.2        // reset allarme cella 2
DEFINE          Buzzer_2          O2.1        // buzzer allarme cella 2
DEFINE          Luce_2            O2.2        // lampeggiante allarme cella 2

//------------------------------
//          n. 03
//  PN 2I2O  cella 3
//------------------------------
DEFINE          Allarme_3         I3.1        // pulsante allarme cella 3
DEFINE          Reset_3           I3.2        // reset allarme cella 3
DEFINE          Buzzer_3          O3.1        // buzzer allarme cella 3
DEFINE          Luce_3            O3.2        // lampeggiante allarme cella 3

//----------------------------------
//          n. 05
//  PN VIS  punto di presidio
//----------------------------------
DEFINE          Cifra             O5:C        // display
DEFINE          Tacita            I5.1        // pulsante tacitazione allarmi
DEFINE          PNVIS             5           // indirizzo display
DEFINE          _C_               57          // codice lettera C

//-------------------------------------------------------------
//
//          Definizione variabili e costanti utente
//
//-------------------------------------------------------------
DEFINE          Allarmi           V1.1        // allarmi celle
DEFINE          Al_Cella_1        V1.2
DEFINE          Al_Cella_2        V1.3
DEFINE          Al_Cella_3        V1.4
DEFINE          Suoneria          V10         // stato buzzer esterno
DEFINE          Cont              V100



MACRO SetDisp (ch3, Val, DP, Addr, Num, Ingr)
{
     IF [Ingr&(Cont==Num)] OAddr:C = Val; ENDIF   // Assegno cifra
     SETPAR[Addr,26,DP,Ingr&(Cont==Num)];         // Locazione 26 => Decimal Point
     SETPAR[Addr,27,ch3,Ingr&(Cont==Num)];        // Locazione 27 => Digit 3'
     IF [!Ingr] Cont=Cont+1; ENDIF                // Incremento contatore se non allarmi
}
```

```
MACRO ClrDisp (Addr, Cond)
{
     IF [Cond] OAddr:C = 0; ENDIF       // Assegno cifra
     SETPAR[Addr,26,0,Cond];            // 0 = nessuna virgola
     SETPAR[Addr,27,63,Cond];           // Locazione 27 => Digit 3'
}

//---------------------------------------------------------------------
//
//                      Inizio Programma
//
//---------------------------------------------------------------------

Allarmi = Al_Cella_1 | Al_Cella_2 | Al_Cella_3;

Suoneria.1 = SR[Allarmi, Tacita];                  // suoneria allarme
WRITE[PNVIS, 0, Suoneria];

Al_Cella_1  = SR[Allarme_1 , Reset_1];
Al_Cella_2  = SR[Allarme_2 , Reset_2];
Al_Cella_3  = SR[Allarme_3 , Reset_3];

Buzzer_1 = Al_Cella_1;                          // buzzer allarme cella
Luce_1   = PWM[5,5,Al_Cella_1];                 // lampeggiante allarme cella
Buzzer_2 = Al_Cella_2;                          // buzzer allarme cella
Luce_2   = PWM[5,5,Al_Cella_2];                 // lampeggiante allarme cella
Buzzer_3 = Al_Cella_3;                          // buzzer allarme cella
Luce_3   = PWM[5,5,Al_Cella_3];                 // lampeggiante allarme cella


//-------------------------------------------------
//          Segnalazione luminosa
//-------------------------------------------------
Cont=CNT1[PWM[2,2,ON],ON,(Cont>=3)];              //Contatore per alternanza allarmi

SetDisp (_C_, 01, 4, PNVIS, 0, Al_Cella_1);
SetDisp (_C_, 02, 4, PNVIS, 1, Al_Cella_2);
SetDisp (_C_, 03, 4, PNVIS, 2, Al_Cella_3);

ClrDisp (PNVIS, !Allarmi);
```

4.14 – Modulo PN 4I2OA

Il modulo PN 4I2OA è un modulo misto ingresso/uscita che dispone di 4 ingressi digitali optoisolati normalmente aperti, 2 uscite a relè e 2 uscite analogiche 0-10V per il collegamento diretto di plafoniere con ballast dimmerabili.
Mediante le due uscite analogiche è possibile effettuare la regolazione luminosa indipendente di due gruppi di reattori, mentre le due uscite a relè sono utilizzate per il comando ON/OFF della plafoniera.
I 4 ingressi digitali presenti possono essere utilizzati per la regolazione luminosa oppure come ingressi generici del sistema.
Nel primo caso ad ogni canale corrisponde una coppia di ingressi per la regolazione up/down della luminosità: la pressione prolungata del pulsante collegato all'ingresso provoca la variazione luminosa, mentre la pressione breve provoca l'apertura o chiusura del relè.

Per il modulo PN 4I2OA sono possibili due distinte modalità operative.

Funzionamento locale

In tale modalità il modulo PN 4I2OA si comporta come un normale dimmer a due canali indipendenti.
I quattro ingressi disponibili svolgono le funzioni rispettivamente di ingresso "up" e "down" dei due canali di uscita. Come in un normale dimmer ciascuna uscita viene alternativamente accesa o spenta da successive pressioni dei pulsanti connessi agli ingressi del modulo.

In caso di spegnimento, l'uscita viene disalimentata, mentre in caso di accensione l'uscita assume l'ultimo valore impostato.
In maniera indipendente per ogni canale la prolungata pressione di uno dei due pulsanti produce invece l'aumento (valore max 100%) o la diminuzione (valore min. 0%) del valore dell'uscita relativa.

E' la modalità di funzionamento più comune e più semplice perché non comporta per il master nessuna elaborazione.
Tale modalità operativa è quella che il modulo assume alla prima accensione o in caso di avaria del bus PIC*net*.

Funzionamento remoto

Qualora connesso ad un bus PIC*net*, il modulo PN 4I2OA abbandona la modalità di controllo locale e imposta le sue uscite sulla base delle direttive ricevute dal modulo master del sistema.
In tale modalità operativa i quattro ingressi presenti sul modulo assumono il significato di generici ingressi digitali del sistema e come tali possono essere utilizzati dal programma utente.

In tale modalità di funzionamento è dunque possibile in maniera centralizzata effettuare le seguenti operazioni:

- **accensione/spegnimento indipendente di ciascun canale**
- **regolazione indipendente di ciascun canale**

Le risorse del modulo PN 4I2OA vengono gestite in parte mediante comandi dedicati ed in parte come un normale modulo dotato di ingessi ed uscite digitali secondo la seguente corrispondenza:

- uscita 1:	**uscita analogica primo canale**	**Ox:A**
- uscita 2:	**uscita analogica secondo canale**	**Ox:B**
- ingresso up canale 1:	**ingresso n° 1**	**Ix.1**
- ingresso down canale 1:	**ingresso n° 2**	**Ix.2**
- ingresso up canale 2:	**ingresso n° 3**	**Ix.3**
- ingresso down canale 2:	**ingresso n° 4**	**Ix.4**

essendo x l'indirizzo del modulo.

Le diverse impostazioni di funzionamento sono impostabile mediante la direttiva SETPAR (per la scrittura) o READ (per la lettura) (vedi § 3.14) ai seguenti indirizzi:

Grandezza	*Accesso*	*Indirizzo*	*Range*	*Default*
Range uscite analogiche	Lettura/scrittura	01	0: range 0-5V 1: range 0-10V	1
Attivazione relè uscita 1	Lettura/scrittura	02	0: disattivata 1: attivata	0
Attivazione relè uscita 2	Lettura/scrittura	03	0: disattivata 1: attivata	0
Modalità comando relè	Lettura/scrittura	04	0: comando utente 1: comando automatico	1
Modalità funzione ingressi	Lettura/scrittura	05	0: comandi separati 1: tasto unico	0
Forzamento locale	Lettura/scrittura	08	0: comando remoto 1: comando locale	0
Intervallo di tempo dissolvenza [ms]	Lettura/scrittura	09	1-255	20

Con il comando automatico dei relè (registro n. 04) il modulo provvede a diseccitare i relè quando la corrispondente uscita analogica assume il valore zero.
Questa impostazione (condizione di default) è particolarmente comoda nel funzionamento in remoto dal momento che ponendo a zero l'uscita di regolazione si diseccita il corrispondente relè senza la necessità di inviare uno specifico comando.

Impostando invece il comando utente dei relè occorre agire sui registri 02 e 03 per comandarne l'accensione e lo spegnimento.

Il registro 05 permette infine di scegliere la modalità di regolazione luminosa nel funzionamento in locale.
Con comandi separati (condizione di default) tutti i quattro ingressi vengono utilizzati per la regolazione secondo la corrispondenza prima indicata. Con comando a tasto unico invece i due ingressi n. 2 e n. 4 non sono utilizzati mentre gli ingressi n. 1 e 3 regolano rispettivamente i due canali.
L'attivazione breve dell'ingresso provoca l' ON/OFF dell'uscita. La pressione prolungata alternativamente ne regola l'intensità in aumento o in diminuzione.
Continuando a tenere attivo l'ingresso dopo che l'uscita corrispondente ha raggiunto il valore estremo (massimo o minimo) dopo una pausa l'uscita inizia la regolazione in senso opposto.

E' possibile introdurre un raccordo tra valori diversi di ogni uscita in modo da realizzare effetti di dissolvenza. La funzione di raccordo automatico introduce una rampa per passare dal valore iniziale dell'uscita al valore finale richiesto.
Il valore di default della rampa di raccordo è tale da introdure una variazione di un'unità ogni 20ms.
Questa impostazione può essere modificata scrivendo nel registro 09 l'intervallo di tempo di attesa per la variazione di un'unità. Impostando il registro a 1 si disattiva la funzionalità.

```
//-------------------------------------------------------------
//      Programma esempio di gestione
//      modulo PN 4I2OA
//      I pulsanti 1 e 2 dimmerano i 2 canali di uscita
//

//-------------------------------------------------------------
//
//          Definizione ingressi/uscite
//
//-------------------------------------------------------------

//-----------------------------------
//      Master PN MAS LITE
//-----------------------------------
DEFINE  Puls1   I254.1    // definizione pulsanti
DEFINE  Puls2   I254.2    // definizione pulsanti
DEFINE  Puls3   I254.3    // definizione pulsanti
DEFINE  Puls4   I254.4    // definizione pulsanti
DEFINE  Puls5   I254.5    // definizione pulsanti
DEFINE  Puls6   I254.6    // definizione pulsanti
DEFINE  Puls7   I254.7    // definizione pulsanti
DEFINE  Puls8   I254.8    // definizione pulsanti

DEFINE  Spia1   O254.1    // definizione led fissi
DEFINE  Spia2   O254.2    // definizione led fissi
DEFINE  Spia3   O254.3    // definizione led fissi
DEFINE  Spia4   O254.4    // definizione led fissi
DEFINE  Spia5   O254.5    // definizione led fissi
DEFINE  Spia6   O254.6    // definizione led fissi
DEFINE  Spia7   O254.7    // definizione led fissi
DEFINE  Spia8   O254.8    // definizione led fissi

DEFINE  Lamp1   O254.9    // definizione led lampeggianti
DEFINE  Lamp2   O254.10   // definizione led lampeggianti
DEFINE  Lamp3   O254.11   // definizione led lampeggianti
DEFINE  Lamp4   O254.12   // definizione led lampeggianti
DEFINE  Lamp5   O254.13   // definizione led lampeggianti
DEFINE  Lamp6   O254.14   // definizione led lampeggianti
DEFINE  Lamp7   O254.15   // definizione led lampeggianti
DEFINE  Lamp8   O254.16   // definizione led lampeggianti

//----------------------------
//      n. 2 PN4I2OA
//----------------------------
DEFINE  Out1_1  O2:A      //  uscita 0-10 primo canale
DEFINE  Out2_1  O2:B      //  uscita 0-10 secondo canale
DEFINE  In1_1   I2.1      //  ingresso up primo canale
DEFINE  In2_1   I2.2      //  ingresso down primo canale
DEFINE  In3_1   I2.3      //  ingresso up secondo canale
DEFINE  In4_1   I2.4      //  ingresso down secondo canale
```

```
//-------------------------------------------------------------
//          Definizione variabili
//-------------------------------------------------------------

//-------------------------
//  Definizioni virtuali
//-------------------------
DEFINE  Commut1      V1.1
DEFINE  Commut2      V1.2
DEFINE  Dimm1        V1.5
DEFINE  Dimm2        V1.6

DEFINE  Acceso1      V2.1
DEFINE  Acceso2      V2.2
DEFINE  Pulsanti     V2.5
DEFINE  Start        V2.6           // flag prima accensione

DEFINE  OldPuls1_1   V3.1           // memorie pulsanti
DEFINE  OldPuls2_1   V3.2

DEFINE  UpDw1        V5.1
DEFINE  UpDw2        V5.2
DEFINE  P_ch1        V5.5
DEFINE  P_ch2        V5.6

DEFINE  Conta        V10            // contatore attesa
DEFINE  Step         V11            // variazione in regolazione
DEFINE  Delta        V12            // variazione in dissolvenza

DEFINE  Canale1      V13            // valore attuale uscita 1
DEFINE  Canale2      V14            // valore attuale uscita 2


//-------------------------
//  Definizioni costanti
//-------------------------
DEFINE  WAIT_R       3              // attesa [0.5s] per regolare canale
DEFINE  MAX          255            // valore massimo uscita DA
DEFINE  MIN          30             // valore minimo uscita DA


//-------------------------------------------------------------------
//
//                     Inizio Programma
//
//-------------------------------------------------------------------
RAMBACKUP;

//----------------------------------
//  Impostazioni iniziali
//----------------------------------
SETSLAVE(Out1_1)=125;
SETSLAVE(Out2_1)=125;
TIMESLAVE(10);
SETPAR[2,4,1];                      // Pn4i2oa rele in automatico (default)
SETPAR[2,1,1];                      // Pn4i2oa tensione max 10V (default)
SETSLAVE(I1)= LAST;
```

```
//-----------------------------------------------
//  Inizializzazioni prima accensione
//-----------------------------------------------
IF [!Start]
     Canale1 = MAX;        // valore iniziale uscita 1
     Canale2 = MAX;        // valore iniziale uscita 2
     Step = 2;
     Delta = 2;
ENDIF

//---------------------------------
//  Regolazioni luminose canali
//---------------------------------
P_ch1 = Puls1 |  In1_1;
P_ch2 = Puls2 |  In2_1;

//-----------------------------------------------------
//          Gestione regolazione manuale
//-----------------------------------------------------
IF [OldPuls1_1 & (!P_ch1) & (Conta < WAIT_R)]
     Commut1 = ON;
     IF [Canale1 < MIN] Canale1 = MIN; ENDIF
ENDIF
IF [P_ch1 & (Conta >= WAIT_R)]
     Dimm1 = ON;
     IF [!Acceso1]   Commut1 = ON; ENDIF
ENDIF
OldPuls1_1 = P_ch1;

IF [OldPuls2_1 & (!P_ch2) & (Conta < WAIT_R)]
     Commut2 = ON;
     IF [Canale2 < MIN] Canale2 = MIN; ENDIF
ENDIF
IF [P_ch2 & (Conta >= WAIT_R)]
     Dimm2 = ON;
     IF [!Acceso2]   Commut2 = ON; ENDIF
ENDIF
OldPuls2_1 = P_ch2;

//---------------------------------
//  Regolazione canale 1
//---------------------------------
UpDw1 = T[Dimm1];
IF [UpDw1 & Dimm1]
     Canale1 = Canale1 + Step;
     IF [Canale1 > MAX] Canale1 = MAX; ENDIF
ENDIF

IF [!UpDw1 & Dimm1]
     Canale1 = Canale1 - Step;
     IF [Canale1 < MIN] Canale1 = MIN; ENDIF
ENDIF
```

```
//---------------------------------
//  Regolazione canale 2
//---------------------------------
UpDw2 = T[Dimm2];
IF [UpDw2 & Dimm2]
     Canale2 = Canale2 + Step;
     IF [Canale2 > MAX] Canale2 = MAX; ENDIF
ENDIF

IF [!UpDw2 & Dimm2]
     Canale2 = Canale2 - Step;
     IF [Canale2 < MIN] Canale2 = MIN; ENDIF
ENDIF

//-----------------------------------------
//     Gestione attivazione canali
//-----------------------------------------
Acceso1 = T[Commut1];
Acceso2 = T[Commut2];

Pulsanti = Puls1 | Puls2;
Conta = CNT1[PWM[2,3,ON], ON, !Pulsanti];

//--------------------------------
//  Gestione accensione canali
//--------------------------------
IF[Acceso1]
     Out1_1 = Canale1;
ELSE
     Out1_1 = 0;
ENDIF

IF[Acceso2]
     Out2_1 = Canale2;
ELSE
     Out2_1 = 0;
ENDIF

//----------------------
//  Settaggi finali
//----------------------
Spia1 = (Out1_1 != 0);
Spia2 = (Out2_1 != 0);

Commut1 = OFF;
Commut2 = OFF;
Dimm1 = OFF;
Dimm2 = OFF;
Start = ON;
```

4.15 – Modulo PN 2I2OWP

Il modulo PN 2I2OWP è un modulo misto ingresso/uscita che dispone di 6 ingressi digitali (di cui 4 a pulsante sul frontale) e di 2 uscite digitali a relè con fusibile di protezione integrato in modo da consentire il collegamento diretto di utenze senza necessità di ulteriori cablaggi esterni. L'alimentazione dei circuiti di potenza risulta protetta dalle sovracorrenti dal fusibile di bordo utilizzando fusibili rapidi con If=4A max.

Le risorse del modulo PN 2I2OWP vengono gestite come in un normale modulo dotato di ingressi e uscite digitali secondo la seguente corrispondenza:

- **pulsante 1: ingresso n° 1 (1=ON) Ix.1**
- **pulsante 2: ingresso n° 2 (1=ON) Ix.2**
- **pulsante 3: ingresso n° 3 (1=ON) Ix.3**
- **pulsante 4: ingresso n° 4 (1=ON) Ix.4**
- **ingresso aux 1: ingresso n° 5 (1=ON) Ix.5**
- **ingresso aux 2: ingresso n° 6 (1=ON) Ix.6**
- **relè REL1: uscita n° 1 (1=ON) Ox.1**
- **relè REL2: uscita n° 2 (1=ON) Ox.2**

essendo x l'indirizzo del modulo.

Lo stato del fusibile viene visualizzato sul pannello frontale con il led FUSE (led acceso=fusibile interrotto) e può essere rilevato dal modulo master.

- **stato fusibile: ingresso n° 7 (1=fusibile interrotto) Ix.7**

L'accensione del led avviene solo con almeno un'uscita attivata e carico collegato.

Ad es. si voglia comandare l'accensione di due luci dai primi due pulsanti o da due pulsanti collegati agli ingressi ausiliari. I pulsanti 3 e 4 svolgono la funzione rispettivamente di accensione e spegnimento totale.

```
//-------------------------------------------------------------
//      Programma esempio di gestione
//      modulo PN 2I2OWP
//

//-------------------------------------------------------------
//
//          Definizione ingressi/uscite
//
//-------------------------------------------------------------

DEFINE  Puls_1       I1.1      // definizione pulsanti
DEFINE  Puls_2       I1.2      // definizione pulsanti
DEFINE  Puls_3       I1.3      // definizione pulsanti
DEFINE  Puls_4       I1.4      // definizione pulsanti
DEFINE  In_Aux1      I1.5      // definizione ingressi
DEFINE  In_Aux2      I1.6      // definizione ingressi
DEFINE  Fuse         I1.7      // stato fusibile
DEFINE  Luce_1       O1.1      // definizione uscite
DEFINE  Luce_2       O1.2      // definizione uscite
DEFINE  Led_Fuse     O1.7      // led stato fusibile

Luce_1 = T[Puls_1 | In_Aux1 | (Puls_3 & !Luce_1) | (Puls_4 & Luce_1)];
Luce_2 = T[Puls_2 | In_Aux2 | (Puls_3 & !Luce_2) | (Puls_4 & Luce_2)];

A1=ALM["Allarme mancanza"," fusibile 1",OFF,ON, Fuse];
```

4.16 – Modulo PN 4I4OW

Il modulo PN 4I4OW è un modulo misto ingresso/uscita che dispone di 4 ingressi digitali optoisolati e di 4 uscite digitali a relè in modo da consentire il collegamento diretto di utenze senza necessità di ulteriori cablaggi esterni.

Le risorse del modulo PN 4I4OW vengono gestite in parte mediante comandi dedicati ed in parte come un normale modulo dotato di ingessi ed uscite digitali secondo la seguente corrispondenza:

- **ingresso 1:**	**ingresso n° 1 (1=ON)**	**Ix.1**
- **ingresso 2:**	**ingresso n° 2 (1=ON)**	**Ix.2**
- **ingresso 3:**	**ingresso n° 3 (1=ON)**	**Ix.3**
- **ingresso 4:**	**ingresso n° 4 (1=ON)**	**Ix.4**
- **relè REL1:**	**uscita n° 1 (1=ON)**	**Ox.1**
- **relè REL2:**	**uscita n° 2 (1=ON)**	**Ox.2**
- **relè REL3:**	**uscita n° 3 (1=ON)**	**Ox.3**
- **relè REL4:**	**uscita n° 4 (1=ON)**	**Ox.4**

essendo x l'indirizzo del modulo.

Su tutti gli ingressi è possibile inserire un filtro software per eliminare falsi contatti in caso di segnali molto disturbati. E' inoltre possibile selezionare da software lo stato di riposo dell'ingresso (NO o NC) senza dover coinvolgere il programma utente del modulo master.
Oltre che come generico modulo di ingresso e uscita, il modulo PN 4I4OW può essere utilizzato per svolgere particolari funzioni selezionabili in fase di inizializzazione del programma.

Funzione contatore impulsi

E' una funzionalità presente in automatico su tutti gli ingressi nel normale funzionamento.
Tale funzionalità permette il conteggio indipendente delle commutazioni di ciascuno dei quattro ingressi con una frequenza massima di 400Hz per ogni canale.
Il valore del conteggio viene letto con la direttiva READ e viene azzerato dopo ogni lettura, dunque il valore letto è sempre il numero di impulsi rilevati tra due letture successive.

Modalità tapparella

Attivando questa modalità gli ingressi e le uscite del modulo non vengono più gestiti dal modulo master, ma garantiscono il funzionamento indipendente di due automazioni (tapparelle o tende) secondo le seguente corrispondenza:

ingressi 1-2:	**pulsanti salita-discesa motore 1**
uscite 1-2:	**comandi salita-discesa motore 1**
ingressi 3-4:	**pulsanti salita-discesa motore 2**
uscite 3-4:	**comandi salita-discesa motore 2**

Lo stato degli ingressi e delle uscite è tuttavia accessibile da master per la realizzazione di funzioni complesse come l'apertura o chiusura centralizzata o in sequenza delle automazioni.
Per sua natura la modalità tapparella può essere impostata per coppie di ingressi/uscite e quindi solo i due bit meno significativi del registro sono operativi (bit 1 per la coppia 1-2; bit 2 per la coppia 3-4).

Modalità locale

Come per tutti i moduli misti, ossia dotati sia di ingressi che di uscite a relè, esiste la possibilità di attivare una modalità locale in caso di avaria bus.
In tale modalità di funzionamento, ciascuna uscita dipende dal corrispondente ingresso con comando passo-passo o con comando mantenuto. In tal modo risulta sempre garantito il comando locale ad es. di un punto luce anche in caso di avaria del sistema.

Tale modalità è automaticamente disabilitata su una coppia di ingressi/uscite se su quella coppia è attivata la modalità tapparella; è tuttavia possibile ad es. impostare contemporaneamente la modalità tapparella sulla coppia di ingressi/uscite 3-4, la modalità locale passo-passo sull'uscita 1 e quella con comando mantenuto sull'uscita 2.

Le diverse impostazioni di funzionamento sono gestite mediante la direttiva SETPAR (per la scrittura) o READ (per la lettura) (vedi § 3.14) ai seguenti indirizzi:

Grandezza	*Accesso*	*Indirizzo*	*Range*	*Default*
Tempo di filtraggio ingressi in ms	Lettura/scrittura	03	0-255	50
Modalità tapparella	Lettura/scrittura	05	0: disattivata 1: attivata	00b
Durata corsa tapparella in multipli di 0.5s	Scrittura	06	0-255	32
Conteggio impulsi ingresso 1	Lettura	07	0-255	0
Conteggio impulsi ingresso 2	Lettura	08	0-255	0
Conteggio impulsi ingresso 3	Lettura	09	0-255	0
Conteggio impulsi ingresso 4	Lettura	10	0-255	0
Stato uscite	Lettura	11	- -	0000b
Abilitazione modalità locale	Lettura/scrittura	25	0: disattivata 1: attivata	1
Selezione ingressi attivi alti (NO) o bassi (NC)	Lettura/scrittura	34	0: attivo basso 1: attivo alto	1111b
Modalità funzionamento locale	Lettura/scrittura	35	0: comando diretto 1: passo-passo	1111b

```
//-----------------------------------------------
//      Programma esempio di gestione
//      modulo PN 4I4OW
//-----------------------------------------------

//-----------------------------------------------
//          Definizione ingressi/uscite
//-----------------------------------------------
DEFINE  Puls_1       I1.1      // definizione pulsanti
DEFINE  Puls_2       I1.2      // definizione pulsanti
DEFINE  Puls_3       I1.3      // definizione pulsanti
DEFINE  Puls_4       I1.4      // definizione pulsanti
DEFINE  Luce_1       O1.1      // definizione uscite
DEFINE  Luce_2       O1.2      // definizione uscite
DEFINE  Salita       O1.3      // definizione uscite
DEFINE  Discesa      O1.4      // definizione uscite

DEFINE  Puls_Up      I10.1     // pulsante apertura centralizzata
DEFINE  Puls_Dw      I10.2     // pulsante chiusura centralizzata


//-----------------------------------------------
//          Definizione registri
//-----------------------------------------------
DEFINE  LOCAL_MODE      35     // Sceglie la modalità di lavoro di ogni singolo relè
DEFINE  USE_LOCAL       25     // Abilita modalità locale per intervento di SETSLAVE
DEFINE  INPUT_DEBOUNCE  3      // Tempo filtraggio ingressi espresso in millisecondi
DEFINE  SHUTTER_SEL     5      // Abilita modalità tapparella
DEFINE  SHUTTER_TIME    6      // Tempo escursione tapparella espresso in 0.5 secondi
DEFINE  COUNTER_I1      7      // Conteggio impulsi ingressi 1
DEFINE  COUNTER_I2      8      // Conteggio impulsi ingressi 2
DEFINE  COUNTER_I3      9      // Conteggio impulsi ingressi 3
DEFINE  COUNTER_I4      10     // Conteggio impulsi ingressi 4
DEFINE  INPUT_LEVEL     34     // Sceglie la polarità ingressi in modalità locale
DEFINE  RESET           170    // Resetta a valori di fabbrica


//--------------------------
// Definizioni di costanti
//--------------------------
DEFINE  MODULE          1      // Indirizzo modulo

//--------------------------
// Definizioni di variabili
//--------------------------
DEFINE  RelativeCnt     V1
DEFINE  TotalCnt        V2


//--------------------------
// Impostazioni iniziali
//--------------------------
SETSLAVE(OMODULE) = 00000001b;  // In caso di avaria il rele 1 chiude
TIMESLAVE(2);                   // Imposta il tempo di intervento del SETSLAVE

//--------------------------------------
// Imposto modalità locale uscite 1 e 2
// Rele 1: modalità diretta
// Rele 2: modalità passo passo
//--------------------------------------
SETPAR[MODULE, LOCAL_MODE, 00000010b];
SETPAR[MODULE, USE_LOCAL, 1];  // Abilito modalità locale dopo SETSLAVE
```

```
//---------------------------------------
// Imposto modalità tapparella per
// uscite 3 e 4 - escursione in 20 sec
//---------------------------------------
SETPAR[MODULE, SHUTTER_SEL, 00000010b];
SETPAR[MODULE, SHUTTER_TIME, 40];


//---------------------------------------
//      Equazioni impianto
//---------------------------------------

Luce_1 = T[Puls_1];
Luce_2 = T[Puls_2];

Salita = SHOT[10, Puls_Up];     // apertura centralizzata
Discesa = SHOT[10, Puls_Dw];    // chiusura centralizzata

// Leggo il contaimpulsi dell'ingresso 1
RelativeCnt = READ[MODULE, COUNTER_I1];

// Se il conteggio relativo non è nullo, ossia ci sono stati impulsi dall'ultima lettura,
// incremento il conteggio totale
IF[RelativeCnt != 0]
        TotalCnt = TotalCnt + RelativeCnt;
ENDIF;
```

4.17– Modulo PN 8I8O

Il modulo PN 8I8O è un modulo misto ingresso/uscita che dispone di 8 ingressi digitali optoisolati e di 8 uscite digitali a relè in modo da consentire il collegamento diretto di utenze senza necessità di ulteriori cablaggi esterni.

Le risorse del modulo PN 8I8O vengono gestite in parte mediante comandi dedicati ed in parte come un normale modulo dotato di ingessi ed uscite digitali secondo la seguente corrispondenza:

-	**ingresso 1:**	**ingresso n° 1 (1=ON)**	**Ix.1**
-	**ingresso 2:**	**ingresso n° 2 (1=ON)**	**Ix.2**
-	**ingresso 3:**	**ingresso n° 3 (1=ON)**	**Ix.3**
-	**ingresso 4:**	**ingresso n° 4 (1=ON)**	**Ix.4**
-	**ingresso 5:**	**ingresso n° 5 (1=ON)**	**Ix.5**
-	**ingresso 6:**	**ingresso n° 6 (1=ON)**	**Ix.6**
-	**ingresso 7:**	**ingresso n° 7 (1=ON)**	**Ix.7**
-	**ingresso 8:**	**ingresso n° 8 (1=ON)**	**Ix.8**
-	**relè REL1:**	**uscita n° 1 (1=ON)**	**Ox.1**
-	**relè REL2:**	**uscita n° 2 (1=ON)**	**Ox.2**
-	**relè REL3:**	**uscita n° 3 (1=ON)**	**Ox.3**
-	**relè REL4:**	**uscita n° 4 (1=ON)**	**Ox.4**
-	**relè REL5:**	**uscita n° 5 (1=ON)**	**Ox.5**
-	**relè REL6:**	**uscita n° 6 (1=ON)**	**Ox.6**
-	**relè REL7:**	**uscita n° 7 (1=ON)**	**Ox.7**
-	**relè REL8:**	**uscita n° 8 (1=ON)**	**Ox.8**

essendo x l'indirizzo del modulo.

Su tutti gli ingressi è possibile inserire un filtro software per eliminare falsi contatti in caso di segnali molto disturbati. E' inoltre possibile selezionare da software lo stato di riposo dell'ingresso (NO o NC) senza dover coinvolgere il programma utente del modulo master.
Oltre che come generico modulo di ingresso e uscita, il modulo PN 8I8O può essere utilizzato per svolgere particolari funzioni selezionabili in fase di inizializzazione del programma.

Modalità tapparella

Attivando questa modalità gli ingressi e le uscite del modulo non vengono più gestiti dal modulo master, ma garantiscono il funzionamento indipendente di quattro automazioni (tapparelle o tende) secondo le seguente corrispondenza:

ingressi 1-2:	**pulsanti salita-discesa motore 1**
uscite 1-2:	**comandi salita-discesa motore 1**
ingressi 3-4:	**pulsanti salita-discesa motore 2**
uscite 3-4:	**comandi salita-discesa motore 2**
ingressi 5-6:	**pulsanti salita-discesa motore 3**
uscite 5-6:	**comandi salita-discesa motore 3**
ingressi 7-8:	**pulsanti salita-discesa motore 4**
uscite 7-8:	**comandi salita-discesa motore 4**

Lo stato degli ingressi e delle uscite è tuttavia accessibile da master per la realizzazione di funzioni complesse come l'apertura o chiusura centralizzata o in sequenza delle automazioni.
Per sua natura la modalità tapparella può essere impostata per coppie di ingressi/uscite e quindi solo i quattro bit meno significativi del registro sono operativi (bit 1 per la coppia 1-2; bit 2 per la coppia 3-4; bit 3 per la coppia 5-6; bit 4 per la coppia 7-8).

Modalità locale

Come per tutti i moduli misti, ossia dotati sia di ingressi che di uscite a relè, esiste la possibilità di attivare una modalità locale in caso di avaria bus.
In tale modalità di funzionamento, ciascuna uscita dipende dal corrispondente ingresso con comando passo-passo o con comando mantenuto. In tal modo risulta sempre garantito il comando locale ad es. di un punto luce anche in caso di avaria del sistema.

Tale modalità è automaticamente disabilitata su una coppia di ingressi/uscite se su quella coppia è attivata la modalità tapparella; è tuttavia possibile ad es. impostare contemporaneamente la modalità tapparella sulle coppie di ingressi/uscite 3-4 e 5-6, la modalità locale passo-passo sull'uscita 1, 7 e 8 e quella con comando mantenuto sull'uscita 2.

Le diverse impostazioni di funzionamento sono gestite mediante la direttiva SETPAR (per la scrittura) o READ (per la lettura) (vedi § 3.14) ai seguenti indirizzi:

Grandezza	*Accesso*	*Indirizzo*	*Range*	*Default*
Tempo di filtraggio ingressi in ms	Lettura/scrittura	03	0-255	50
Modalità tapparella	Lettura/scrittura	05	0: disattivata 1: attivata	0000b
Durata corsa tapparella in multipli di 0.5s	Scrittura	06	0-255	32
Stato uscite	Lettura	11	- -	0000 0000b
Abilitazione modalità locale	Lettura/scrittura	25	0: disattivata 1: attivata	1
Selezione ingressi attivi alti (NO) o bassi (NC)	Lettura/scrittura	34	0: attivo basso 1: attivo alto	1111 1111b
Modalità funzionamento locale	Lettura/scrittura	35	0: comando diretto 1: passo-passo	1111 1111b

4.18 – Modulo PN HUMI

I moduli PN HUMI sono moduli con la funzionalità di umidostato e termostato ambiente.
Sul modulo sono inoltre presenti un ingresso ausiliario ed un relè di appoggio.

I moduli PN HUMI sono dunque moduli che riuniscono in un unico indirizzo:

- **due ingressi analogici ad 8 bit**
- **un uscita digitale: comando relè**
- **un ingresso digitale: ingresso ausiliario**

Il modulo PN HUMI viene dunque gestito come un normale modulo dotato di ingressi analogici ed uscite ed ingressi digitali secondo la seguente corrispondenza:

- **ingresso ausiliario: ingresso n° 1 (1=ON) Ix:1**
- **comando relè ausiliario: uscita n° 1 (1=ON) Ox.1**

essendo x l'indirizzo del modulo.

I valori di umidità e temperatura devono essere letti mediante la direttiva READ (vedi § 3.14) con la seguente corrispondenza:

GRANDEZZA	Indirizzo	Range
Umidità relativa (in decimi di punto percentuale)	**1**	**0-1.000 (0-100%)**
Temperatura (in decimi di grado)	**2**	**0-1.200 (0-120°C)**

NOTA: si raccomanda di interrogare il modulo a intervalli regolari (pausa minima 30s tra letture successive) perché l'interrogazione continua del sensore produce un riscaldamento dello stesso che si traduce in un errore di lettura.

E' tuttavia possibile correggere eventuali offset di temperatura dovuti ad errate installazioni mediante la direttiva SETPAR al seguente indirizzo:

- **offset di lettura temperatura: indirizzo 14**

Esempio di utilizzo del modulo:

```
//-----------------------------------------------
//  Programma demo gestione PN HUMI
//-----------------------------------------------

DEFINE  REG_HUMIDITY     1        //  Umidità relativa (x10)
DEFINE  REG_TEMPERATURE  2        //  Temperatura (x10)
DEFINE  MOD_ADDRESS      1        //  Indirizzo modulo PN HUMI

FLOAT : Humi                      //  Variabile valore convertito
FLOAT : Temp                      //  Variabile valore convertito

V1 = READ[MOD_ADDRESS, REG_HUMIDITY];       // Lettura registro umidità relativa
V2 = READ[MOD_ADDRESS, REG_TEMPERATURE];    // Lettura registro temperatura

Humi = ITOF[V1] / 10.0;           // Conversione del valore letto in %
Temp = ITOF[V2] / 10.0;           // Conversione del valore letto in °C

//-----------------------------------
// Visualizzazione dei dati letti
//-----------------------------------
M1 = MESS["RH: %Humi% %%    ","T:  %Temp% C   ", OFF, OFF, PWM[2, 2, ON]];
D254=M1;
```

4.19 – Modulo PN 4I2O2PT

Il modulo misto ingresso/uscita PN 4I2O2PT dispone di 4 ingressi digitali optoisolati, di 2 uscite digitali a relè e di 2 ingressi analogici per sonda PT100 o PT1000.

I moduli PN 4I2O2PT sono dunque moduli che riuniscono in un unico indirizzo:

- **due ingressi analogici ad 8 bit**
- **due uscite digitali: comando relè 1 e 2**
- **quattro ingressi digitali: ingresso ausiliari 1-4**

Il modulo PN 4I2O2PT viene dunque gestito come un normale modulo dotato di ingressi analogici (vedi §1.2.1) ed uscite ed ingressi digitali secondo la seguente corrispondenza:

- temperatura canale 1:	**ingresso analogico 1**	**Ix:A**
- temperatura canale 2:	**ingresso analogico 2**	**Ix:B**
- comando relè 1:	**uscita n° 1 (1=ON)**	**Ox.1**
- comando relè 2:	**uscita n° 2 (1=ON)**	**Ox.2**

essendo x l'indirizzo del modulo.

La temperatura viene letta come variabile intera ad 8 bit con la seguente corrispondenza:

1 = 0.196 [°C] scala 0 ÷ 50°C
1 = 0.392 [°C] scala 0 ÷ 100°C

Lo stato degli ingressi può essere letto mediante la direttiva READ (vedi § 3.14) all'indirizzo 0 con la seguente corrispondenza:

- ingresso 1	**bit 1**	**(0 = OFF, 1 = ON)**
- ingresso 2	**bit 2**	**(0 = OFF, 1 = ON)**
- ingresso 3	**bit 3**	**(0 = OFF, 1 = ON)**
- ingresso 4	**bit 4**	**(0 = OFF, 1 = ON)**

Esempio di utilizzo del modulo:

```
//-----------------------------------------------
//  Programma demo gestione PN 4I2O2PT
//-----------------------------------------------

DEFINE  CH_A            I1:A     //  ingresso primo canale
DEFINE  CH_B            I1:B     //  ingresso secondo canale
DEFINE  Aerotermo_1     O1.1     //  consenso aerotermo 1
DEFINE  Aerotermo_2     O1.2     //  consenso aerotermo 2

DEFINE  KC              0.196    //  fattore di conversione
DEFINE  KI              3        //  valore isteresi

DEFINE  Ingressi        V1
DEFINE  In_1            V1.1
DEFINE  In_2            V1.2
DEFINE  In_3            V1.3
DEFINE  In_4            V1.4
DEFINE  Fascia_1        V100.1
DEFINE  Fascia_2        V100.2
DEFINE  Fascia_On       V100.3
DEFINE  SetTerm         V101
DEFINE  SetTermNotte    V102
DEFINE  Temp_rif        V103

FLOAT : Temp1                    //  Variabile valore convertito canale 1
FLOAT : Temp2                    //  Variabile valore convertito canale 2
```

```
//-----------------------------------------------
//               Programma utente
//-----------------------------------------------

Ingressi = READ[1,0];             // Lettura registro stato ingressi

SetTerm = UVAR["Temp desiderata",5,35,1,18];          // set point giorno
SetTermNotte = UVAR["Temp notturna",5,15,1,6];        // set point notte

//-----------------------------------------------------------------
// il riscaldamento viene acceso nelle fasce orarie previste
// oppure per 30 minuti alla pressione del pulsante di zona
//-----------------------------------------------------------------
Fascia_1 = UPRG["Fascia Termo1", Lu-Sa, 6:30, 11:30];
Fascia_2 = UPRG["Fascia Termo2", Lu-Ve, 13:00, 18:30];
Fascia_On = Fascia_1 | Fascia_2 | SHOT[18000, In_2 | In_4];

Temp1 = KC*ITOF[CH_A];            // Conversione del valore letto in °C
Temp2 = KC*ITOF[CH_B];            // Conversione del valore letto in °C

IF [Fascia_On]
   Temp_rif = FTOI[ITOF[SetTerm] / KC];
   Aerotermo_1 = !IST[CH_A, Temp_rif - KI, Temp_rif + KI];  // isteresi zona 1
   Aerotermo_2 = !IST[CH_B, Temp_rif - KI, Temp_rif + KI];  // isteresi zona 2
ELSE
   Temp_rif = FTOI[ITOF[SetTermNotte] / KC];
   Aerotermo_1 = !IST[CH_A, Temp_rif - KI, Temp_rif + KI];  // isteresi zona 1
   Aerotermo_2 = !IST[CH_B, Temp_rif - KI, Temp_rif + KI];  // isteresi zona 2
ENDIF

//-------------------------------
//      Definizione allarmi
//-------------------------------
A1 = ALM["Scatto termica","Aerotermo 1",ON,ON,In_1];
A2 = ALM["Scatto termica","Aerotermo 2",ON,ON,In_3];

//-----------------------------------
// Visualizzazione dei dati letti
//-----------------------------------
M1 = MESS["T Cha: %Temp1%  ","T Chb: %Temp2%  ",OFF,OFF,PWM[1,1,ON]];
D254=M1;
```

4.20 – Modulo PN CVM

Il prodotto PN CVM è costituito da un modulo da guida DIN che permette l'interfacciamento al bus PICnet dello strumento di misura multifunzione per grandezze elettriche prodotto da ASITA mod. CVM-MINI CVM/M/RS4C2.

Gli analizzatori della gamma CVM-MINI sono strumenti programmabili per la misura, il calcolo e la visualizzazione di tutti i principali parametri delle linee elettriche trifase e trifase con neutro sia equilibrate che squilibrate.

Attraverso il modulo PN CVM è possibile riportare sul bus PICnet la maggior parte delle grandezze misurate dallo strumento.
Per ottimizzare il flusso di dati le grandezze sono state divise in due gruppi che vengono aggiornati con differente frequenza. Le grandezze ad alta priorità hanno un tempo di rinfresco di circa 10 secondi, mentre quelle a bassa priorità vengono aggiornate con frequenza di circa un minuto.
Nel seguito l'elenco delle grandezze disponibili ed il corrispondente indirizzo di lettura.

Tutte le grandezze sono accessibili mediante la direttiva READ (vedi § 3.14) con la seguente corrispondenza:

ALTA PRIORITA'	
Parametro	**Addr.**
Tensione istantanea – V1	0
Corrente istantanea – A1	1
Tensione istantanea – V2	2
Corrente istantanea – A2	3
Tensione istantanea – V3	4
Corrente istantanea – A3	5
Potenza Attiva trifase istantanea	6
Pot. Induttiva trifase istantanea	7
Pot. Capacitiva trifase istantanea	8
Fattore di potenza istantaneo	9
Frequenza istantanea	10
Tensione L1-L2 istantanea	11
Tensione L2-L3 istantanea	12
Tensione L3-L1 istantanea	13
Distorsione armonica istant. V1	14
Distorsione armonica istant. V2	15
Distorsione armonica istant. V3	16
Distorsione armonica istant. A1	17
Distorsione armonica istant. A2	18
Distorsione armonica istant. A3	19
Energia Attiva istantanea	20
Energia Induttiva istantanea	21
Energia Capacitiva istantanea	22
Potenza Apparente istantanea	23
Massima Domanda istantanea	24
Massima Domanda massima	25

BASSA PRIORITA'	
Parametro	**Addr.**
Tensione massima – V1	26
Corrente massima – A1	27
Tensione massima – V2	28
Corrente massima – A2	29
Tensione massima – V3	30
Corrente massima – A3	31
Potenza Attiva trifase massima	32
Pot. Induttiva trifase massima	33
Pot. Capacitiva trifase massima	34
Fattore di potenza massimo	35
Frequenza massima	36
Tensione L1-L2 massima	37
Tensione L2-L3 massima	38
Tensione L3-L1 massima	39
Distorsione armonica max. V1	40
Distorsione armonica max. V2	41
Distorsione armonica max. V3	42
Distorsione armonica max. A1	43
Distorsione armonica max. A2	44
Distorsione armonica max. A3	45
Potenza Apparente massima	46
Tensione minima – V1	47
Corrente minima – A1	48
Tensione minima – V2	49
Corrente minima – A2	50
Tensione minima – V3	51
Corrente minima – A3	52

BASSA PRIORITA' (continua)	
Parametro	**Addr.**
Potenza Attiva trifase minima	53
Pot. Induttiva trifase minima	54
Pot. Capacitiva trifase minima	55
Fattore di potenza minimo	56
Frequenza minima	57
Tensione L1-L2 minima	58
Tensione L2-L3 minima	59
Tensione L3-L1 minima	60
Distorsione armonica min. V1	61
Distorsione armonica min. V2	62
Distorsione armonica min. V3	63
Distorsione armonica min. A1	64
Distorsione armonica min. A2	65
Distorsione armonica min. A3	66
Potenza Apparente minima	67
Potenza attiva istantanea fase 1	68
Potenza reattiva istantanea fase 1	69
Fattore di potenza istant. fase 1	70

BASSA PRIORITA'	
Parametro	**Addr.**
Potenza attiva istantanea fase 2	71
Potenza reattiva istantanea fase 2	72
Fattore di potenza istant. fse 2	73
Potenza attiva istantanea fase 3	74
Potenza reattiva istantanea fase 3	75
Fattore di potenza istant. fse 3	76
Corrente di neutro istantanea	77
Potenza attiva massima fase 1	78
Potenza reattiva massima fase 1	79
Fattore di potenza max. fase 1	80
Potenza attiva massima fase 2	81
Potenza reattiva massima fase 2	82
Fattore di potenza max. fase 2	83
Potenza attiva massima fase 3	84
Potenza reattiva massima fase 3	85
Fattore di potenza max. fase 3	86
Corrente di neutro massima	87

Esempio di utilizzo del modulo:

```
//-----------------------------------------------
//  Programma demo gestione PN CVM
//-----------------------------------------------

DEFINE  CVM     22            // indirizzo del modulo

DECL(ICVM);                   // Dichiarazione fittizia per mantenere la comunicazione
                              // continua e per fare rilevare il modulo dal master

SETPAR[CVM, 130, 1];          // Imposta baudrate di comunicazione con CVM a 19200

V1 = READ[CVM, 0];            // Legge tensione L1
V2 = READ[CVM, 10];           // Legge frequenza
V3 = READ[CVM, 14];           // Legge THD% di V1

M1 = MESS["VL1 %V1% F %V2%   ", "THD %%V1 %V3%", OFF, OFF, PWM[2, 2, ON]];

D254 = M1;
```

4.21 – Modulo PN TAI

Il prodotto PN TAI è una scheda di interfaccia per centrali antintrusione mod. TECNOALARM TP16/256.

Mediante questa interfaccia l'intera centrale viene vista come un singolo modulo della rete PICnet, consentendo una completa integrazione dell'automazione impianti con la sicurezza.
Ciascuna zona o settore può essere oggetto di supervisione e controllo e può essere utilizzata per la realizzazione di scenari o sequenze di eventi.

Attraverso il modulo PN TAI è possibile riportare sul bus PICnet tutte le informazioni relative al funzionamento della centrale ed effettuare le principali impostazioni di funzionamento ed in particolare:

- **attivare / disattivare programmi**
- **inserire / escludere zone**
- **leggere lo stato di tamper di una zona**
- **leggere lo stato di intrusione di una zona**
- **leggere lo stato di inserimento / esclusione di una zona**
- **leggere lo stato dei programmi**

Le diverse impostazioni di funzionamento sono gestite mediante la direttiva SETPAR (per la scrittura) o READ (per la lettura) (vedi § 3.14) ai seguenti indirizzi:

REGISTRI di CENTRALE	
Parametro	**Addr.**
Primo numero password centrale	1
Secondo numero password centrale	**2**
Terzo numero password centrale	**3**
Quarto numero password centrale	4
Quinto numero password centrale	**5**
Sesto numero password centrale	**6**
Stato di esclusione delle zone. Ogni parola riporta lo stato di 16 zone bit per bit	**10-25**
Stato di allarme delle zone. Ogni parola riporta lo stato di 16 zone bit per bit	**26-41**
Stato di allarme tamper delle zone. Ogni parola riporta lo stato di 16 zone bit per bit	**42-57**
Stato di attività delle zone. Ogni parola riporta lo stato di 16 zone bit per bit	**58-73**
Stato di attività delle zone. Ogni parola riporta lo stato di 16 zone bit per bit	**74-89**
Attivazione zone	**90**
Disattivazione zone	**91**
Attivazione programmi	**106**
Disattivazione programmi	**107**

Esempio di utilizzo del modulo:

```
//-----------------------------------------------
//  Programma demo gestione PN TAI
//-----------------------------------------------

// Registri password di accesso alla centrale

DEFINE REG_CODE_1           1
DEFINE REG_CODE_2           2
DEFINE REG_CODE_3           3
DEFINE REG_CODE_4           4
DEFINE REG_CODE_5           5
DEFINE REG_CODE_6           6

// Registri stato esclusione delle zone 1->256
// Ogni registro è composto da 16bit, ogni bit rappresenta una zona in ordine crescente.
// Esempio: il bit meno significativo del REG_ZONE_EXCL_0 rappresenta la zona 1 mentre il
// più significativo la zona 16 e così via.

DEFINE REG_ZONE_EXCL_0      10
DEFINE REG_ZONE_EXCL_1      11
DEFINE REG_ZONE_EXCL_2      12
DEFINE REG_ZONE_EXCL_3      13
DEFINE REG_ZONE_EXCL_4      14
DEFINE REG_ZONE_EXCL_5      15
DEFINE REG_ZONE_EXCL_6      16
DEFINE REG_ZONE_EXCL_7      17
DEFINE REG_ZONE_EXCL_8      18
DEFINE REG_ZONE_EXCL_9      19
DEFINE REG_ZONE_EXCL_10     20
DEFINE REG_ZONE_EXCL_11     21
DEFINE REG_ZONE_EXCL_12     22
DEFINE REG_ZONE_EXCL_13     23
DEFINE REG_ZONE_EXCL_14     24
DEFINE REG_ZONE_EXCL_15     25

// Registri stato allarme intrusione delle zone 1->256
// Ogni registro è composto da 16bit, ogni bit rappresenta lo stato dell'allarme
// intrusione di una zona.

DEFINE REG_ZONE_ALM_IN_0    26
DEFINE REG_ZONE_ALM_IN_1    27
DEFINE REG_ZONE_ALM_IN_2    28
DEFINE REG_ZONE_ALM_IN_3    29
DEFINE REG_ZONE_ALM_IN_4    30
DEFINE REG_ZONE_ALM_IN_5    31
DEFINE REG_ZONE_ALM_IN_6    32
DEFINE REG_ZONE_ALM_IN_7    33
DEFINE REG_ZONE_ALM_IN_8    34
DEFINE REG_ZONE_ALM_IN_9    35
DEFINE REG_ZONE_ALM_IN_10   36
DEFINE REG_ZONE_ALM_IN_11   37
DEFINE REG_ZONE_ALM_IN_12   38
DEFINE REG_ZONE_ALM_IN_13   39
DEFINE REG_ZONE_ALM_IN_14   40
DEFINE REG_ZONE_ALM_IN_15   41

// Registri stato allarme tamper delle zone 1->256
// Ogni registro è composto da 16bit, ogni bit rappresenta lo stato dell'allarme tamper
// di una zona.

DEFINE REG_ZONE_TAMPER_0    42
DEFINE REG_ZONE_TAMPER_1    43
DEFINE REG_ZONE_TAMPER_2    44
DEFINE REG_ZONE_TAMPER_3    45
```

```
DEFINE REG_ZONE_TAMPER_4     46
DEFINE REG_ZONE_TAMPER_5     47
DEFINE REG_ZONE_TAMPER_6     48
DEFINE REG_ZONE_TAMPER_7     49
DEFINE REG_ZONE_TAMPER_8     50
DEFINE REG_ZONE_TAMPER_9     51
DEFINE REG_ZONE_TAMPER_10    52
DEFINE REG_ZONE_TAMPER_11    53
DEFINE REG_ZONE_TAMPER_12    54
DEFINE REG_ZONE_TAMPER_13    55
DEFINE REG_ZONE_TAMPER_14    56
DEFINE REG_ZONE_TAMPER_15    57

// Registri stato zona attiva delle zone 1->256
// Ogni registro è composto da 16bit, ogni bit rappresenta lo stato attiva / non attiva
// di una zona.

DEFINE REG_ZONE_ACTIVE_0     58
DEFINE REG_ZONE_ACTIVE_1     59
DEFINE REG_ZONE_ACTIVE_2     60
DEFINE REG_ZONE_ACTIVE_3     61
DEFINE REG_ZONE_ACTIVE_4     62
DEFINE REG_ZONE_ACTIVE_5     63
DEFINE REG_ZONE_ACTIVE_6     64
DEFINE REG_ZONE_ACTIVE_7     65
DEFINE REG_ZONE_ACTIVE_8     66
DEFINE REG_ZONE_ACTIVE_9     67
DEFINE REG_ZONE_ACTIVE_10    68
DEFINE REG_ZONE_ACTIVE_11    69
DEFINE REG_ZONE_ACTIVE_12    70
DEFINE REG_ZONE_ACTIVE_13    71
DEFINE REG_ZONE_ACTIVE_14    72
DEFINE REG_ZONE_ACTIVE_15    73

// Registri stato dei programmi 1->32
// Ogni registro è composto da 16bit, che a loro volta compongono due parole da 8bit
// che rappresentano lo stato di due programmi per descrizione bit vedere doc.

DEFINE REG_PROGRAM_STATE_0   74
DEFINE REG_PROGRAM_STATE_1   75
DEFINE REG_PROGRAM_STATE_2   76
DEFINE REG_PROGRAM_STATE_3   77
DEFINE REG_PROGRAM_STATE_4   78
DEFINE REG_PROGRAM_STATE_5   79
DEFINE REG_PROGRAM_STATE_6   80
DEFINE REG_PROGRAM_STATE_7   81
DEFINE REG_PROGRAM_STATE_8   82
DEFINE REG_PROGRAM_STATE_9   83
DEFINE REG_PROGRAM_STATE_10  84
DEFINE REG_PROGRAM_STATE_11  85
DEFINE REG_PROGRAM_STATE_12  86
DEFINE REG_PROGRAM_STATE_13  87
DEFINE REG_PROGRAM_STATE_14  88
DEFINE REG_PROGRAM_STATE_15  89

// Registri per l'attivaziopne / disattivazione delle zone 1->256
// Ogni registro è composto da 16bit, ogni bit rappresenta il trigger di attivazione/
// disattivazione di una zona.

DEFINE REG_ACTIVATE_ZONE_0   90
DEFINE REG_ACTIVATE_ZONE_1   91
DEFINE REG_ACTIVATE_ZONE_2   92
DEFINE REG_ACTIVATE_ZONE_3   93
DEFINE REG_ACTIVATE_ZONE_4   94
DEFINE REG_ACTIVATE_ZONE_5   95
DEFINE REG_ACTIVATE_ZONE_6   96
DEFINE REG_ACTIVATE_ZONE_7   97
DEFINE REG_ACTIVATE_ZONE_8   98
```

```
DEFINE REG_ACTIVATE_ZONE_9   99
DEFINE REG_ACTIVATE_ZONE_10  100
DEFINE REG_ACTIVATE_ZONE_11  101
DEFINE REG_ACTIVATE_ZONE_12  102
DEFINE REG_ACTIVATE_ZONE_13  103
DEFINE REG_ACTIVATE_ZONE_14  104
DEFINE REG_ACTIVATE_ZONE_15  105

// Registri per l'attivazione / disattivazione dei programmi definiti 1->32
// Ogni registro è composto da 16bit, ogni bit rappresenta il trigger di attivazione /
// disattivazione di una programma.

DEFINE REG_ACTIVATE_PRG_0    106
DEFINE REG_ACTIVATE_PRG_1    107

DEFINE MOD_ADDRESS           1        // Definizione indirizzo modulo

//---------------------------------
// Impostazione password 12345
//---------------------------------

SETPAR[MOD_ADDRESS, REG_CODE_1, 1];
SETPAR[MOD_ADDRESS, REG_CODE_2, 2];
SETPAR[MOD_ADDRESS, REG_CODE_3, 3];
SETPAR[MOD_ADDRESS, REG_CODE_4, 4];
SETPAR[MOD_ADDRESS, REG_CODE_5, 5];
SETPAR[MOD_ADDRESS, REG_CODE_6, 0];

//-------------------------------------------
// Lettura stato allarme intrusione zone 1->15
//-------------------------------------------

V1 = READ[MOD_ADDRESS, REG_ZONE_ALM_IN_0];

M1 = MESS["** INTRUSIONE **", "    ZONA 1    ", OFF, OFF, V1.1];
M2 = MESS["** INTRUSIONE **", "    ZONA 2    ", OFF, OFF, V1.2];
M3 = MESS["** INTRUSIONE **", "    ZONA 10   ", OFF, OFF, V1.10];

//           1     1        0
//           6543210987654321
//   515 = 0b0000001000000011 zone 1->16
//

SETPAR[MOD_ADDRESS, REG_ACTIVATE_ZONE_0, 515];

D254 = M1, M2, M3;
```

5.0 - Esempi applicativi

Gli esempi applicativi del sistema PIC*net* che seguono, sono forniti per dare all'utente alcune indicazioni sulle modalità con cui si possono risolvere alcuni problemi ricorrenti nella pratica impiantistica.
Tali esempi devono intendersi puramente indicativi in quanto non si propongono di esaurire tutte le problematiche che si riscontrano nelle reali applicazioni. Sinthesi srl non è imputabile a nessun titolo per eventuali danni a persone o cose derivanti dall'uso degli esempi contenuti in questa pubblicazione.

5.1 – Luci corridoio 1

Si voglia controllare l'accensione di un punto luce di un corridoio collegato all'uscita O1.1 da più pulsanti collegati a diversi ingressi del sistema.
Il problema si risolve semplicemente collegando tutti i pulsanti in parallelo mediante l'operatore OR per comandare un unico operatore passo-passo che comanda direttamente l'uscita.

Una possibile soluzione del problema è dunque la seguente:

```
DEFINE Pulsante1          I2.1
DEFINE Pulsante2          I2.2
DEFINE Pulsante3          I2.3
DEFINE Pulsante4          I2.4
DEFINE Luce1              O1.1
DEFINE Luce2              O1.2

Luce1 = T[Pulsante1 | Pulsante2 | Pulsante3 | Pulsante4];
Luce2 = Luce1;
```

5.2 – Luci corridoio 2

Con riferimento all'esempio precedente, si voglia comandare l'accensione di due distinti corridoi da due distinti gruppi di pulsanti. Si vuole inoltre disporre anche di un pulsante generale con il quale comandare l'accensione generale dei due corridoi.
Il problema si risolve semplicemente collegando in parallelo ai pulsanti mediante l'operatore OR anche il pulsante generale. Tale pulsante generale dovrà però essere attivo su un'uscita solo se è spenta in modo da accendere solo le luci spente e non spegnere quelle già accese.
Basterà dunque mettere tale pulsante generale in serie (mediante l'operatore AND) con l'uscita da controllare per ottenere il risultato voluto.

Una possibile soluzione del problema è dunque la seguente:

```
DEFINE Puls1              I2.1
DEFINE Puls2              I2.2
DEFINE Puls3              I2.3
DEFINE Puls4              I3.1
DEFINE Puls5              I3.2
DEFINE Puls6              I3.3
DEFINE PulsGenerale       I4.1
DEFINE Luce1              O1.1
DEFINE Luce2              O1.2

Luce1 = T[Puls1 | Puls2 | Puls3 | (PulsGenerale & !Luce1)];
Luce2 = T[Puls4 | Puls5 | Puls6 | (PulsGenerale & !Luce2)];
```

5.3 – Spegnimento generale luci

Con riferimento agli esempi precedenti, si voglia ora comandare l'accensione di due distinti punti luce da due distinti gruppi di pulsanti. Si vuole inoltre che la pressione prolungata (per 4 sec) di uno qualsiasi dei pulsanti di accensione provochi lo spegnimento di tutte le luci.
Il problema dello spegnimento centralizzato si risolve facendo in modo che la pressione di uno qualunque dei pulsanti di accensione faccia partire un contatore incrementato ogni secondo.
Se il contatore raggiunge il valore di 4 comanda lo spegnimento delle luci. Il contatore viene incrementato dall'uscita di un generatore di lampeggio PWM sempre attivo con tempi di accensione e spegnimento di 5 decimi di secondo (un lampeggio al secondo). Il contatore viene resettato se non ci sono pulsanti premuti.

Una possibile soluzione del problema è dunque la seguente:

```
DEFINE Puls1              I2.1
DEFINE Puls2              I2.2
DEFINE Puls3              I2.3
DEFINE Puls4              I3.1
DEFINE Pulsanti           V1.1
DEFINE Generale           V1.2
DEFINE Conta              V20
DEFINE Luce1              O1.1
DEFINE Luce2              O1.2

Pulsanti = Puls1 | Puls2 | Puls3 | Puls4;
Conta = CNT1[ PWM[5, 5, ON], ON, !Pulsanti ];
Generale = (Conta >= 4);

Luce1 = T[Puls1 | Puls2 | (Generale & Luce1)];
Luce2 = T[Puls3 | Puls4 | (Generale & Luce2)];
```

5.4 – Comando generale luci

Con riferimento agli esempi precedenti, si voglia ora comandare l'accensione di due distinti punti luce da due distinti pulsanti. Si vuole inoltre disporre di un terzo pulsante la cui pressione rapida provoca lo spegnimento generale delle luci; una pressione prolungata dello stesso pulsante invece provoca l'accensione generale di tutte le uscite.
Il problema si risolve per analogia all'esempio precedente controllando il raggiungimento di una opportuna soglia di un contatore incrementato ogni secondo dalla pressione del pulsante generale.

Una possibile soluzione del problema è dunque la seguente:

```
//------------------------------
//  Definizioni costanti
//------------------------------
DEFINE Stopcont            2
DEFINE Generale            I1.1         // pulsante di controllo luci generale
DEFINE Puls1               I2.1         // pulsanti di comando luce locali
DEFINE Puls2               I2.2
DEFINE Luce1               O1.1
DEFINE Luce2               O1.2

//------------------------------
//  Definizioni variabili
//------------------------------
INTEGER    : conta                      // durata della pressione del pulsante generale
BOOL       : old                        // stato precedente del pulsante generale
BOOL       : accendi
BOOL       : spegni

//------------------------------
//  Definizioni macro
//------------------------------
MACRO ControllaUscita (output, input)
{
   output = T[input | (!output & accendi) | ( output & spegni)];
}

//------------------------------
//  Programma principale
//------------------------------
if [old & (!Generale) & (conta < Stopcont)]   // se ho rilasciato il pulsante generale
     spegni = ON;
endif
if [Generale & (conta >= Stopcont)]           // se il pulsante generale è ancora premuto
     accendi = ON;
endif

ControllaUscita (Luce1, Puls1);
ControllaUscita (Luce2, Puls2);
conta = CNT1[ PWM[5, 5, Generale], ON, !Generale];

old = Generale;
accendi = OFF;
spegni = OFF;
```

5.5 – Comando luci notturne

Si voglia controllare l'accensione di una luce notturna collegata all'uscita O1.1 mediante un orologio programmatore. Un pulsante deve permettere l'accensione e lo spegnimento anticipato del punto luce, ma allo scadere del periodo di accensione oraria, la luce deve sempre spegnersi.
Il problema si risolve semplicemente utilizzando l'operatore di marcia-arresto.

Una prima possibile soluzione del problema è la seguente:

```
//-------------------------
//  Definizioni utente
//-------------------------
DEFINE START1              16:30
DEFINE STOP1               16:31
DEFINE START2              23:30
DEFINE STOP2               23:31
DEFINE Pulsante            I2.1
DEFINE LuceExt             O1.1
DEFINE Orologio1           V1.1
DEFINE Orologio2           V1.2
DEFINE Puls                V1.3

Puls = SR[Pulsante, Puls];      // trasformo la pressione pulsante in un impulso unitario
Orologio1 = PRG1[Lu-Ve, START1, STOP1];
Orologio2 = PRG1[Lu-Ve, START2, STOP2];
LuceExt = SR[ (Puls & !LuceExt) | Orologio1, (Puls & LuceExt) | Orologio2];
```

Una soluzione più elegante può essere ottenuta con un'unica programmazione oraria nel seguente modo:

```
//-------------------------
//  Definizioni utente
//-------------------------
DEFINE START               16:30
DEFINE STOP                23:30
DEFINE Pulsante            I2.1
DEFINE LuceExt             O1.1
DEFINE Orologio            V1.1
DEFINE Halt                V1.2
DEFINE Go                  V1.3
DEFINE Puls                V1.4

Puls = SR[Pulsante, Puls];      // trasformo la pressione pulsante in un impulso unitario
Orologio = PRG1[Lu-Ve, START, STOP];
Go = SHOT[ 1, Clock];           // genero un impulso all' accensione del clock
Halt = SHOT[ 1, !Clock];        // genero un impulso allo spegnimento del clock

LuceExt = SR[ (Puls & !LuceExt) | Go, (Puls & LuceExt) | Halt];
```

5.6 – Dimmer ad un solo pulsante

La funzione dimmer (vedi §2.3.13) prevede la presenza di due campi rispettivamente per l'aumento e la diminuzione dell'intensità dell'uscita oltre che per le normali operazioni di accensione e spegnimento.
Tale soluzione risulta sicuramente più comoda ed intuitiva per l'utente, ma in molti dimmer del commercio è presente un solo pulsante per lo svolgimento di tutte le funzioni prima elencate.
In tal caso pressioni successive dello stesso pulsante producono alternativamente l'aumento o la diminuzione dell'intensità dell'uscita.
Per riprodurre tale modalità di funzionamento utilizzando un solo pulsante per ciascun canale di un modulo PNDIM si può procedere come segue:

```
DEFINE Pulsante1           I2.1
DEFINE Pulsante2           I2.2
DEFINE Luce1               O1:A
DEFINE Luce2               O1:B
DEFINE UpDown1             V1.1
DEFINE UpDown2             V1.2

UpDown1 = T[Pulsante1];
UpDown2 = T[Pulsante2];
Luce1 = DIMMER[UpDown1 & Pulsante1, !UpDown1 & Pulsante1];
Luce2 = DIMMER[UpDown2 & Pulsante2, !UpDown2 & Pulsante2];
```

La funzione DIMMER può essere utilizzata anche nel controllo di apparecchi a lampade fluorescenti con ballast dimmerabili utilizzando per questo scopo un modulo di uscita analogica PNDA.
In questo caso è opportuno che l'alimentazione all'apparecchio venga fornita da un modulo di uscita a relè (in modo da poter gestire le diverse accensioni richieste in un ambiente), mentre il modulo di uscita analogica gestirà la regolazione luminosa (in genere unica per lo stesso ambiente).
Si avranno dunque vari pulsanti per le diverse accensioni, ed un pulsante per regolare la luminosità: la pressione prolungata di questo pulsante alternativamente aumenta e diminuisce la luminosità, mentre la pressione rapida attiva o disattiva il ballast.
L'accensione e lo spegnimento del ballast dovranno anche provocare rispettivamente l'alimentazione e la disalimentazione delle plafoniere che in precedenza erano state accese con il rispettivo pulsante di zona.
In questo caso nella funzione dimmer è opportuno impedire che la parzializzazione della luce arrivi a zero per evitare che l'utente non riesca ad attivare le accensioni usando i pulsanti di zona.

Un esempio applicativo può essere il seguente:

```
DEFINE Dimm                I2.1         // pulsante di regolazione luminosità
DEFINE Puls1               I2.2         // pulsante accensione zona 1
DEFINE Puls2               I2.3         // pulsante accensione zona 2
DEFINE Luce1               O1.1         // uscita accensione zona 1
DEFINE Luce2               O1.2         // uscita accensione zona 2
DEFINE Ballast             O2           // uscita analogica 0-10V controllo ballast
DEFINE UpDown              V1.1
DEFINE Minimo              20           // valore minimo della parzializzazione

UpDown = T[Dimm];
Ballast = DIMMER[Minimo, UpDown & Dimm, !UpDown & Dimm];
Luce1 = T[Puls1] & (Ballast !=0);
Luce2 = T[Puls2] & (Ballast !=0);
```

5.7 – Verifica ore di funzionamento centrale termica

Nell'esempio che segue viene gestito il funzionamento di una centrale termica di cui si vuole conoscere il tempo di funzionamento totale.
Gli orari di funzionamento della caldaia sono impostati mediante programmatori orari modificabili dall'utente.
La pompa di mandata viene fatta funzionare da un temporizzatore ritardato allo spegnimento e comandato dall'accensione del bruciatore, in modo che le pompe funzionino durante il tempo di accensione del bruciatore e si spengano con un certo ritardo per sfruttare il calore accumulato nell'acqua in circolo.
Il calcolo delle ore di funzionamento viene effettuato mediante una MACRO che conteggia ore e minuti di funzionamento del bruciatore. La durata di funzionamento, così come i possibili allarmi di funzionamento, viene visualizzata sul display del modulo master.

Una possibile soluzione del problema è dunque la seguente:

```
//------------------------------
//    Programma caldaia
//------------------------------
//
//           Modulo 4I4O
//   I1 = Termostato ambiente
//   I2 = Consenso impianto
//   I3 = Blocco bruciatore
//   I4 = Blocco pompa
//   O1 = comando bruciatore
//   O2 = comando pompe 1/2
//   O3 = scorta
//   O4 = comando pompa ricircolo

//------------------------------
//  Definizioni costanti
//------------------------------
DEFINE Caldaia              O1.1
DEFINE Pompe                O1.2
DEFINE Ricircolo            O1.4
DEFINE Termost              I1.1        // termostato ambiente abitazione
DEFINE Consenso             I1.2        // selettore di consenso accensione impianto
DEFINE Ritardo              6000        // ritardo allo spegnimento della pompa rispetto
                                        // alla caldaia (10 min)

//------------------------------
//  Definizioni variabili
//------------------------------
INTEGER       : NumOre
INTEGER       : NumMinuti
INTEGER       : Minuti
INTEGER       : MinutiOld
BOOL          : Clock1
BOOL          : Clock2
BOOL          : Clock3
```

```
//-----------------------------
//  Definizioni macro
//-----------------------------
MACRO ContaOre(N_Ore, N_Minuti, Consenso, Reset)
{
     MinutiOld = Minuti;
     Minuti = RTC(MM);

     IF [Consenso]
          IF [Minuti != MinutiOld]
               N_Minuti = N_Minuti+1;
          ENDIF
          IF [N_Minuti == 60]
               N_Ore = N_Ore+1;
               N_Minuti = 0;
          ENDIF
     ENDIF

     IF [Reset]
          N_Ore = 0;
          N_Minuti = 0;
     ENDIF
}

//-----------------------------
//  Programma principale
//-----------------------------

Clock1 = UPRG["Caldaia mattino", Lu-Do, 7:00, 7:30];
Clock2 = UPRG["Caldaia pomerig.", Lu-Do, 14:00, 15:00];
Clock3 = UPRG["Caldaia sera", Lu-Do, 18:00, 18:30];

ContaOre(NumOre, NumMinuti, Caldaia, !Consenso);

V1.4 = PWM [40,40,ON];
M1 = MESS["  Casa Rossi  ", " sistema attivo ", OFF, OFF, V1.4 & Consenso];
M2 = MESS["  Casa Rossi  ", "sistema inattivo", OFF, OFF, V1.4 & !Consenso];
M3 = MESS[" Impianto attivo", "da %NumOre%h e %NumMinuti%min", OFF, OFF, !V1.4 &
          Consenso];

A1 = ALM[" Allarme bruciatore ", " centrale termica  ", ON, OFF, I1.3];
A2= ALM[" Allarme pompa ", " centrale termica  ", ON, OFF, I1.4];

D254 = M1, M2, M3;

IF [Consenso]
     Caldaia = Clock1 | Clock2 | Clock3;
     Pompe = TMR[0, Ritardo, Caldaia, OFF] & Termost;
     Ricircolo = UPRG["Pompa ricircolo", Lu-Do, 7:00, 8:00];
ELSE
     Caldaia = 0;
     Pompe = 0;
     Ricircolo = 0;
ENDIF
```

5.8 – Gestione di un cancello a scorrimento

Nell'esempio che segue viene gestito il funzionamento di un cancello carraio del tipo a scorrimento.
Si suppone che il cancello possa essere comandato in apertura mediante radiocomando e in apertura e chiusura mediante un selettore a chiave. Un ulteriore pulsante permette di bloccare l'avanzamento del cancello.
Vengono inoltre gestiti gli ingressi relativi ai segnali provenienti dalle fotocellule e dalle coste mobili di sicurezza.

Una possibile soluzione del problema è la seguente:

```
//---------------------------------------
// Programma cancello elettrico
//---------------------------------------
//
//           Modulo 4I4O
//   I1 = contatto nc fine corsa apertura cancello
//   I2 = contatto nc fine corsa chiusura cancello
//   I3 = ingresso selettore apertura cancello
//   I4 = ingresso selettore chiusura cancello
//   O1 = uscita lampeggiante
//   O2 = uscita contattore apertura cancello
//   O3 = uscita contattore chiusura cancello
//   O4 = scorta
//
//           Modulo 8I
//   I1 = ingresso comando radio
//   I2 = ingresso fotocellule
//   I3 = ingresso costola mobile montante fisso
//   I4 = ingresso costola mobile montante mobile
//   I5 = ingresso stop cancello
//   I6/8 = scorta


//-------------------------
//  Definizioni I/O
//-------------------------
DEFINE FC_aperto          I1.1    // contatto nc fine corsa apertura cancello
DEFINE FC_chiuso          I1.2    // contatto nc fine corsa chiusura cancello
DEFINE SelettoreApri      I1.3    // ingresso selettore apertura cancello
DEFINE SelettoreChiudi    I1.4    // ingresso selettore chiusura cancello
DEFINE Radiocomando       I2.1    // ingresso comando radio
DEFINE Fotocellula        I2.2    // ingresso fotocellule
DEFINE CostolaFissa       I2.3    // ingresso costola montante fisso
DEFINE CostolaMobile      I2.4    // ingresso costola montante mobile
DEFINE Sto                I2.5    // ingresso stop cancello
DEFINE Luce               O1.1    // uscita lampeggiante
DEFINE Apertura           O1.2    // uscita contattore apertura cancello
DEFINE Chiusura           O1.3    // uscita contattore chiusura cancello


//----------------------------
//  Definizioni costanti
//----------------------------
DEFINE Chiuso             0
DEFINE Aprendo            1
DEFINE Aperto             2
DEFINE Chiudendo          3
DEFINE Fermo              4
DEFINE Tmove              300     // durata max apertura/chiusura
DEFINE Topen              200     // durata attesa cancello aperto
```

```
//-----------------------------
//  Definizioni variabili
//-----------------------------
DEFINE  StatoCancello     V2
DEFINE  Consenso          V3.1
DEFINE  Attesa            V3.2
DEFINE  Apriti            V3.3
DEFINE  Chiuditi          V3.4

INIT StatoCancello = Chiuso


//------------------------------
//  Programma principale
//------------------------------

Consenso = UPRG["Cons. cancello", Lu-Do, 6:30, 23:59];
IF [Consenso]
     M1 = MESS["  Cancello ", " in apertura  ", OFF, OFF, (StatoCancello==Aprendo)];
     M2 = MESS["  Cancello ", " aperto  ", OFF, OFF, (StatoCancello==Aperto)];
     M3 = MESS["  Cancello ", " in chiusura ", OFF, OFF, (StatoCancello== Chiudendo)];
     M4 = MESS["  Cancello ", " bloccato  ", OFF, OFF, (StatoCancello==Fermo)];

     D254 = M1, M2, M3, M4;

     Attesa = SHOT[Topen, (StatoCancello==Aperto)];      // attesa con cancello aperto
     Apriti = SHOT[Tmove, (StatoCancello==Aprendo)];     // attesa apertura cancello
     Chiuditi = SHOT[Tmove, (StatoCancello==Chiudendo)]; // attesa chiusura cancello

     IF [StatoCancello==Chiuso]
          IF [Radiocomando | SelettoreApri]
               StatoCancello = Aprendo;
          ENDIF
          Luce = OFF;
          Apertura = OFF;
          Chiusura = OFF;
     ELSE
          IF [StatoCancello==Aprendo]
               IF [SelettoreChiudi | CostolaFissa]
                    StatoCancello = Chiudendo;
               ENDIF

               IF [FC_aperto]
                    StatoCancello = Aperto;
               ENDIF

               IF [Stop]
                    StatoCancello = Fermo;
               ENDIF
               Luce = PWM[10,10,Apriti];
               Apertura = Apriti;
               Chiusura = OFF;
          ELSE
             IF [StatoCancello==Aperto]
                  IF [SelettoreChiudi | !Attesa]
                             StatoCancello = Chiudendo;
                  ENDIF
                  IF [Stop]
                       StatoCancello = Fermo;
                  ENDIF
                  Luce = OFF;
                  Apertura = OFF;
                  Chiusura = OFF;
             ELSE
```

```
                IF [StatoCancello==Chiudendo]
                     IF [Radiocomando | SelettoreApri | Fotocellula | CostolaMobile]
                          StatoCancello = Aprendo;
                     ENDIF
                     IF [FC_chiuso]
                          StatoCancello = Chiuso;
                     ENDIF
                     IF [Stop]
                          StatoCancello = Fermo;
                     ENDIF
                     Luce = PWM[10,10,Chiuditi];
                     Apertura = OFF;
                     Chiusura = Chiuditi;
                ELSE
                     IF [StatoCancello==Fermo]
                          IF [Radiocomando | SelettoreApri]
                             StatoCancello = Aprendo;
                          ENDIF
                          IF [SelettoreChiudi]
                             StatoCancello = Chiudendo;
                          ENDIF
                          Luce = OFF;
                          Apertura = OFF;
                          Chiusura = OFF;
                     ENDIF// IF [StatoCancello==Fermo]
                ENDIF// IF [StatoCancello==Chiudendo]
           ENDIF// IF [StatoCancello==Aperto]
        ENDIF// IF [StatoCancello==Aprendo]
     ENDIF// IF [StatoCancello==Chiuso]
ENDIF// IF [Consenso]
```

5.9 – Gestione irrigazione giardino 1

Nell'esempio che segue viene gestito il funzionamento di un impianto di irrigazione di un giardino suddiviso in quattro zone.
Il sistema provvederà ad attivare la pompa di irrigazione ed in successione le quattro elettrovalvole di zona. Per ogni zona deve essere possibile la scelta del tempo di irrigazione e deve inoltre essere possibile l'esclusione di ogni singola zona.
La situazione di funzionamento delle zone viene mostrata utilizzando il display del master e sfruttando la possibilità di visualizzare campi variabili all'interno delle scritte.

Una possibile soluzione del problema è la seguente:

```
//-----------------------------------------------------
//  Programma gestione irrigazione a zone
//-----------------------------------------------------

//-------------------------
//  Definizioni I/O
//-------------------------
DEFINE  TermPompaIrr   I3.1   // contatto na scatto termico pompa
DEFINE  AutoIrrig      I3.2   // ingresso selettore comando automatico/manuale irrigazione
DEFINE  ManuIrrig      I3.3   // ingresso selettore comando automatico/manuale irrigazione
DEFINE  ChangeZone     I3.4   // ingresso pulsante selezione zona
DEFINE  AbilZone       I3.5   // ingresso pulsante abilitazione/esclusione zona
DEFINE  IncrTime       I3.6   // ingresso pulsante incremento tempo irrigazione zona
DEFINE  DecrTime       I3.7   // ingresso pulsante decremento tempo irrigazione zona

DEFINE  IrrigZona1     O5.1   // uscita valvola zona 1
DEFINE  IrrigZona2     O5.2   // uscita valvola zona 2
DEFINE  IrrigZona3     O5.3   // uscita valvola zona 3
DEFINE  IrrigZona4     O5.4   // uscita valvola zona 4
DEFINE  PompaIrrig     O5.5   // uscita pompa irrigazione

//----------------------------
//  Definizioni costanti
//----------------------------
DEFINE  MinTime        5      // durata minima irrigazione di una zona in minuti
DEFINE  MaxTime        60     // durata massima irrigazione di una zona in minuti

//----------------------------
//  Definizioni variabili
//----------------------------
DEFINE  ClockIrriga     V1.1
DEFINE  Irriga          V1.2
DEFINE  ChangeOld       V1.3
DEFINE  AbilOld         V1.4
DEFINE  IncrOld         V1.5
DEFINE  DecrOld         V1.6
DEFINE  Reset           V1.7
DEFINE  Attiva          V1.8
DEFINE  Disattiva       V1.9

INTEGER:   Time1        // durata effettiva irrigazione zona 1
INTEGER:   Time2        // durata effettiva irrigazione zona 2
INTEGER:   Time3        // durata effettiva irrigazione zona 3
INTEGER:   Time4        // durata effettiva irrigazione zona 4
```

```
INTEGER:   Timer1       // durata irrigazione zona 1
INTEGER:   Timer2       // durata irrigazione zona 2
INTEGER:   Timer3       // durata irrigazione zona 3
INTEGER:   Timer4       // durata irrigazione zona 4
INTEGER:   Time         // tempo di passaggio zona
INTEGER:   Minuti       // conteggio minuti trascorsi da inizio ciclo
INTEGER:   ShowTime     // tempo di irrigazione della zona selezionata
INTEGER:   Zona         // zona da irrigare
INTEGER:   SelZone      // zona selezionata
INIT Zona = 1;
INIT SelZone = 0;
INIT Timer1 = 4;
INIT Timer2 = 4;
INIT Timer3 = 4;
INIT Timer4 = 4;


//------------------------------
//  Programma principale
//------------------------------

A1 = ALM["Blocco pompa", "  irrigazione", ON,ON,TermPompaIrr];

ClockIrriga  = UPRG["Irrigazione",LU-DO, 21:00, 22:00];
Irriga = (ClockIrriga & AutoIrrig) | ManuIrrig;
PompaIrrig = Irriga & !A1;
Minuti = CNT1[PWM[300,300,Irriga], ON, Reset];

//--------------------------------
//  Di default spengo tutto
//--------------------------------
IrrigZona1 = OFF;
IrrigZona2 = OFF;
IrrigZona3 = OFF;
IrrigZona4 = OFF;

//-----------------------------------------------
//  Gestione sequenza accensione zone
//-----------------------------------------------
IF [Irriga]
    Reset = OFF;
    IF [Zona == 1]
        IF [Minuti <= Time1]
            IrrigZona1 = ON;
        ELSE
            Zona = 2;
            Time = Time1+Time2;
        ENDIF
    ELSE
        IF [Zona == 2]
            IF [Minuti <= Time]
                IrrigZona2 = ON;
            ELSE
                Zona = 3;
                Time = Time+Time3;
            ENDIF
```

```
        ELSE
            IF [Zona == 3]
                IF [Minuti <= Time]
                    IrrigZona3 = ON;
                ELSE
                    Zona = 4;
                    Time = Time+Time4;
                ENDIF
            ELSE
                IF [Zona == 4]
                    IF [Minuti <= Time]
                        IrrigZona4 = ON;
                    ELSE
                        Zona = 1;
                        Reset = ON;
                    ENDIF
                ENDIF
            ENDIF
        ENDIF
    ENDIF
ENDIF


//-------------------------------------------------------------------------------------------
//                Gestione pulsante selezione zona
// Ad ogni pressione del pulsante si incrementa la zona selezionata
//-------------------------------------------------------------------------------------------
IF [ChangeZone & !ChangeOld]
    SelZone = SelZone + 1;
    IF [SelZone == 5]
        SelZone = 1;
    ENDIF
ENDIF
ChangeOld = ChangeZone;

//-------------------------------------------------------------------------------------------
//                Gestione pulsante esclusione zona
// Ad ogni pressione alternativamente si abilita o disabilita la zona selezionata
//-------------------------------------------------------------------------------------------
Disattiva = OFF;
Attiva = OFF;
IF [AbilZone & ! AbilOld]
    IF [SelZone == 1]
        IF [Time1 == 0]
            Time1 = Timer1;
            Attiva = ON;
        ELSE
            Time1 = 0;
            Disattiva = ON;
        ENDIF
    ENDIF
    IF [SelZone == 2]
        IF [Time2 == 0]
            Time2 = Timer2;
            Attiva = ON;
        ELSE
            Time2 = 0;
            Disattiva = ON;
        ENDIF
    ENDIF
```

```
    IF [SelZone == 3]
        IF [Time3 == 0]
            Time3 = Timer3;
            Attiva = ON;
        ELSE
            Time3 = 0;
            Disattiva = ON;
        ENDIF
    ENDIF
    IF [SelZone == 4]
        IF [Time4 == 0]
            Time4 = Timer4;
            Attiva = ON;
        ELSE
            Time4 = 0;
            Disattiva = ON;
        ENDIF
    ENDIF
ENDIF
AbilOld = AbilZone;

//-------------------------------------------------------------------------------------
//                  Gestione pulsante incremento tempi irrigazione
// Ad ogni pressione si incrementa il tempo di irrigazione della zona selezionata
//-------------------------------------------------------------------------------------
IF [IncrTime & !IncrOld]
    IF [SelZone == 1]
        Timer1 = Timer1 + 1;
        IF [Timer1 > MaxTime] Timer1 = MaxTime; ENDIF
        ShowTime = Timer1;
    ELSE
        IF [SelZone == 2]
            Timer2 = Timer2 + 1;
            IF [Timer2 > MaxTime] Timer2 = MaxTime; ENDIF
            ShowTime = Timer2;
        ELSE
            IF [SelZone == 3]
                Timer3 = Timer3 + 1;
                IF [Timer3 > MaxTime] Timer3 = MaxTime; ENDIF
                ShowTime = Timer3;
            ELSE
                Timer4 = Timer4 + 1;
                IF [Timer4 > MaxTime] Timer4 = MaxTime; ENDIF
                ShowTime = Timer4;
          ENDIF
        ENDIF
    ENDIF
ENDIF
IncrOld = IncrTime;



//-------------------------------------------------------------------------------------
//                  Gestione pulsante decremento tempi irrigazione
// Ad ogni pressione si diminuisce il tempo di irrigazione della zona selezionata
//-------------------------------------------------------------------------------------
IF [DecrTime & !DecrOld]
    IF [SelZone == 1]
        Timer1 = Timer1 - 1;
        IF [Timer1 < MinTime] Timer1 = MinTime; ENDIF
        ShowTime = Timer1;
    ELSE
        IF [SelZone == 2]
            Timer2 = Timer2 - 1;
            IF [Timer2 < MinTime] Timer2 = MinTime; ENDIF
            ShowTime = Timer2;
        ELSE
            IF [SelZone == 3]
```

```
                Timer3 = Timer3 - 1;
                IF [Timer3 < MinTime] Timer3 = MinTime; ENDIF
                ShowTime = Timer3;
            ELSE
                Timer4 = Timer4 - 1;
                IF [Timer4 < MinTime] Timer4 = MinTime; ENDIF
                ShowTime = Timer4;
          ENDIF
        ENDIF
    ENDIF
ENDIF
DecrOld = DecrTime;

M1 = MESS["Tempo irrigaz.", "zona %SelZone%: %ShowTime% min.", OFF,OFF,
IncrTime | DecrTime];
M2 = MESS["Attivazione", "zona %SelZone% irrigaz.", OFF,OFF, Attiva];
M3 = MESS["Disattivazione", "zona %SelZone% irrigaz.", OFF,OFF, Disattiva];
M4 = MESS["Controllo", "zona %SelZone% irrigaz.", OFF,OFF, ChangeZone];

D254 = M1, M2, M3, M4;
```

5.10 – Gestione irrigazione giardino 2

Nell'esempio che segue viene risolto lo stesso problema del punto precedente utilizzando la funzione variabili utente.
Il sistema provvederà ad attivare la pompa di irrigazione ed in successione le quattro elettrovalvole di zona. Per ogni zona deve essere possibile la scelta del tempo di irrigazione e deve inoltre essere possibile l'esclusione di ogni singola zona.
La situazione di funzionamento delle zone viene mostrata utilizzando il display del master e sfruttando la possibilità di visualizzare campi variabili all'interno delle scritte.

Una possibile soluzione del problema è la seguente:

```
//-------------------------------------------------
//      Gestione irrigazione giardino
//-------------------------------------------------

//-----------------------------------------------------
//  Programma gestione irrigazione a zone
//-----------------------------------------------------

//-------------------------
//  Definizioni I/O
//-------------------------
DEFINE  TermPompaIrr  I1.1  // contatto na scatto termico pompa
DEFINE  AutoIrrig     I1.2  // ingresso selettore comando automatico/manuale irrigazione
DEFINE  ManuIrrig     I1.3  // ingresso selettore comando automatico/manuale irrigazione

DEFINE  IrrigZona1    O2.1  // uscita valvola zona 1
DEFINE  IrrigZona2    O2.2  // uscita valvola zona 2
DEFINE  IrrigZona3    O2.3  // uscita valvola zona 3
DEFINE  IrrigZona4    O2.4  // uscita valvola zona 4
DEFINE  PompaIrrig    O2.5  // uscita pompa irrigazione

//----------------------------
//  Definizioni costanti
//----------------------------
DEFINE  MinTime      5    // durata minima irrigazione di una zona in minuti
DEFINE  MaxTime      60   // durata massima irrigazione di una zona in minuti

//----------------------------
//  Definizioni variabili
//----------------------------
DEFINE  ClockIrriga  V1.1
DEFINE  Irriga       V1.2
DEFINE  Reset        V1.3
DEFINE  Zona1        V2
DEFINE  Zona2        V3
DEFINE  Zona3        V4
DEFINE  Zona4        V5

DEFINE  Seq          V6

INTEGER:   Time1     // durata effettiva irrigazione zona 1
INTEGER:   Time2     // durata effettiva irrigazione zona 2
INTEGER:   Time3     // durata effettiva irrigazione zona 3
INTEGER:   Time4     // durata effettiva irrigazione zona 4
INTEGER:   Timer1    // durata irrigazione zona 1
```

```
INTEGER:   Timer2     // durata irrigazione zona 2
INTEGER:   Timer3     // durata irrigazione zona 3
INTEGER:   Timer4     // durata irrigazione zona 4
INTEGER:   Time       // tempo di passaggio zona
INTEGER:   Minuti     // conteggio minuti trascorsi da inizio ciclo
INTEGER:   ShowTime   // tempo di irrigazione della zona selezionata
INTEGER:   Zona       // zona da irrigare
INIT    Zona = 1;

Timer1 = UVAR["Temporiz. zona 1", MinTime, MaxTime, 1];
Timer2 = UVAR["Temporiz. zona 2", MinTime, MaxTime, 1];
Timer3 = UVAR["Temporiz. zona 3", MinTime, MaxTime, 1];
Timer4 = UVAR["Temporiz. zona 4", MinTime, MaxTime, 1];

Zona1 = UVAR["Attiva zona 1", 0, 1, 1];
Zona2 = UVAR["Attiva zona 2", 0, 1, 1];
Zona3 = UVAR["Attiva zona 3", 0, 1, 1];
Zona4 = UVAR["Attiva zona 4", 0, 1, 1];

//------------------------------
//  Programma principale
//------------------------------
A1 = ALM["Blocco pompa", "  irrigazione", ON,ON,TermPompaIrr];

ClockIrriga  = UPRG["Irrigazione",LU-DO, 21:00, 22:00];
Irriga = (ClockIrriga & AutoIrrig) | ManuIrrig;
PompaIrrig = Irriga & !A1;
Minuti = CNT1[PWM[300,300,Irriga], ON, Reset];

//--------------------------------
//  Di default spengo tutto
//--------------------------------
IrrigZona1 = OFF;
IrrigZona2 = OFF;
IrrigZona3 = OFF;
IrrigZona4 = OFF;

Time1 = Timer1 * Zona1;
Time2 = Timer2 * Zona2;
Time3 = Timer3 * Zona3;
Time4 = Timer4 * Zona4;

//-----------------------------------------------
//  Gestione sequenza accensione zone
//-----------------------------------------------
IF [Irriga]
   Reset = OFF;
   IF [Zona == 1]
      IF [Minuti < Time1]
         IrrigZona1 = ON;
      ELSE
         Zona = 2;
         Time = Time1+Time2;
      ENDIF
   ELSE
```

```
     IF [Zona == 2]
        IF [Minuti < Time]
           IrrigZona2 = ON;
        ELSE
           Zona = 3;
           Time = Time+Time3;
        ENDIF
     ELSE
        IF [Zona == 3]
           IF [Minuti < Time]
              IrrigZona3 = ON;
           ELSE
              Zona = 4;
              Time = Time+Time4;
           ENDIF
        ELSE
           IF [Zona == 4]
              IF [Minuti < Time]
                 IrrigZona4 = ON;
              ELSE
                 Time = 0;
                 Zona = 1;
                 Reset = ON;
              ENDIF
           ENDIF
        ENDIF
     ENDIF
   ENDIF
ENDIF


//-------------------------------
//      Definizione messaggi
//-------------------------------
Seq = CNT1[ PWM[20,20,ON], ON, Seq>4];       // sequencer messaggi

M1 = MESS["Zona 1 attiva", "Tempo =  %Timer1% min.", OFF,OFF, (Zona1 > 0) & (Seq==1)];
M2 = MESS["Zona 1 disattiva", "Tempo =  %Timer1% min.", OFF,OFF, (Zona1==0) & (Seq==1)];
M3 = MESS["Zona 2 attiva", "Tempo =  %Timer2% min.", OFF,OFF, (Zona2 > 0) & (Seq==2)];
M4 = MESS["Zona 2 disattiva", "Tempo =  %Timer2% min.", OFF,OFF, (Zona2==0) & (Seq==2)];
M5 = MESS["Zona 3 attiva", "Tempo =  %Timer3% min.", OFF,OFF, (Zona3 > 0) & (Seq==3)];
M6 = MESS["Zona 3 disattiva", "Tempo =  %Timer3% min.", OFF,OFF, (Zona3==0) & (Seq==3)];
M7 = MESS["Zona 4 attiva", "Tempo =  %Timer4% min.", OFF,OFF, (Zona4 > 0) & (Seq==4)];
M8 = MESS["Zona 4 disattiva", "Tempo =  %Timer4% min.", OFF,OFF, (Zona4==0) & (Seq==4)];

D254 = M1, M2, M3, M4, M5, M6, M7, M8;
```

5.11 – Regolazione temperatura 1

Nell'esempio che segue viene gestito il funzionamento di un impianto di riscaldamento mediante un modulo PN TERM gestito in remoto.
Il sistema provvede ad attivare la pompa di mandata per regolare la temperatura impostata dall'utente sul termostato durante l'orario lavorativo, mentre durante la notte viene forzata una temperatura di mantenimento più bassa.

Una possibile soluzione del problema è la seguente:

```
//-----------------------------------------------------
//  Programma gestione temperatura
//-----------------------------------------------------

//-------------------------
//      Definizioni I/O
//-------------------------
//---------------------------------
//          n. 180
//        modulo PNterm  uffici
//---------------------------------
DEFINE    PNtemp       I180:a    // canale di lettura temperatura
DEFINE    PNsetp       I180:b    // canale di lettura set-point
DEFINE    PNrele       O180.1    // address di attivazione rele'
DEFINE    PNEstInv     O180.2    // address di commutazione est/inv
DEFINE    PNRemLoc     O180.3    // address di commutazione locale/remoto
DEFINE    Estate       I200.1    // commutatore estate/inverno
DEFINE    Inverno      !I200.1

//-------------------------------
//     Definizione variabili
//-------------------------------
DEFINE    ClockUffici  V1.1      // programmatore orario uffici
INTEGER :  DeltaTemp

//-------------------------------
//      Definizione costanti
//-------------------------------
DEFINE    KC           0.1953125  // costante di conversione temperatura
DEFINE    TempMin      76         // valore corrispondente a 15°C
DEFINE    TempMax      82         // valore corrispondente a 16°C
DEFINE    DeltaMin     0          // valore corrispondente a 0°C
DEFINE    Offset       5          // valore corrispondente a 1°C
DEFINE    DeltaMax     10         // valore corrispondente a 2°C
DEFINE    Acceso       ON         // costante relè acceso
DEFINE     SetEst      ON         // costante funzionamento estivo
DEFINE     Remoto      ON         // costante funzionamento in remoto
DEFINE     Spento      OFF        // costante relè spento
DEFINE     SetInv      OFF        // costante funzionamento invernale
DEFINE     Locale      OFF        // costante funzionamento in locale
```

```
//---------------------------------------------------------------------
//                      Inizio Programma
//---------------------------------------------------------------------
RAMBACKUP;

//---------------------------------------------------------------
//  forzo funzionamento in remoto del termostato
//---------------------------------------------------------------
INIT    PNRemLoc = Remoto;

ClockUffici = UPRG["Orario uffici",Lu-Ve,8:00,18:30];

//-------------------------------------------
//  di notte mantengo la temperatura
//  ad un valore di default
//-------------------------------------------
IF [!ClockUffici]
    IF[Inverno]
        Pompa = !IST[PNtemp, TempMin, TempMax];        // comando pompa mandata
    ELSE
        Pompa = OFF;
    ENDIF
//-------------------------------------------
//  di giorno si segue il set-point
//  dell'utente
//-------------------------------------------
ELSE
    //--------------------------------------
    //  calcolo errore di temperatura
    //--------------------------------------
    DeltaTemp = PNtemp - PNsetp + Offset;

    IF[Inverno]
        Pompa = !IST[DeltaTemp, DeltaMin, DeltaMax];   // comando pompa mandata
    ELSE
        Pompa = IST[DeltaTemp, DeltaMin, DeltaMax];
    ENDIF
ENDIF
```

5.12 – Regolazione temperatura 2

Nell'esempio che segue viene gestito il funzionamento di un impianto di riscaldamento mediante un modulo PN TERM gestito in locale.
Il sistema provvede ad attivare la pompa di mandata per regolare la temperatura impostata dall'utente sul termostato durante l'orario lavorativo, mentre durante la notte viene forzata una temperatura di mantenimento più bassa.
Rispetto al caso precedente, la regolazione di temperatura viene effettuata localmente dal termostato, mentre il comando del relé viene riportato in remoto attraverso il bus.
L'impostazione della temperatura notturna avviene con la forzatura dei limiti minimo e massimo di regolazione.

Una possibile soluzione del problema è la seguente:

```
//-----------------------------------------------------
//  Programma gestione temperatura
//-----------------------------------------------------

//-------------------------
//  Definizioni I/O
//-------------------------
//---------------------------------
//      n. 180
//  modulo PNterm  uffici
//---------------------------------
DEFINE     PNtemp      I180:a      // canale di lettura temperatura
DEFINE     PNsetp      I180:b      // canale di lettura set-point
DEFINE     PNrele      O180.1      // address di attivazione rele'
DEFINE     PNEstInv    O180.2      // address di commutazione est/inv
DEFINE     PNRemLoc    O180.3      // address di commutazione locale/remoto

DEFINE     Estate      I200.1      // commutatore estate/inverno
DEFINE     Inverno     !I200.1

//-------------------------------
//     Definizione variabili
//-------------------------------
DEFINE     ClockUffici V1.1        // programmatore orario uffici

//-------------------------------
//      Definizione costanti
//-------------------------------
DEFINE     KC          0.1953125  // costante di conversione temperatura
DEFINE     MinSet      7          // indirizzo soglia minima
DEFINE     MaxSet      6          // indirizzo soglia massima
DEFINE     Soglie      8          // indirizzo abilitazione soglie
DEFINE     TempNott    76         // valore corrispondente a 15°C
DEFINE     Acceso      ON         // costante relè acceso
DEFINE      SetEst     ON         // costante funzionamento estivo
DEFINE      Remoto     ON         // costante funzionamento in remoto
DEFINE      Spento     OFF        // costante relè spento
DEFINE      SetInv     OFF        // costante funzionamento invernale
DEFINE      Locale     OFF        // costante funzionamento in locale
DEFINE      Abilita    0          // costante abilitazione soglie
DEFINE      Disabilita 255        // costante disabilitazione soglie
```

```
//---------------------------------------------------------------------
//                      Inizio Programma
//---------------------------------------------------------------------
RAMBACKUP;

//---------------------------------------------------------------------
//  forzo funzionamento in locale del termostato
//---------------------------------------------------------------------
INIT    PNRemLoc = Locale;

ClockUffici = UPRG["Orario uffici",Lu-Ve,8:00,18:30];

IF[Inverno]
    PNEstInv = SetInv;            // forzo funzionamento invernale
ELSE
    PNEstInv = SetEst;            // forzo funzionamento estivo
ENDIF

SETPAR[180, MinSet, TempNott];
SETPAR[180, MaxSet, TempNott];

IF [ClockUffici]
    //-------------------------------------------------------
    //  di giorno si segue il set-point dell'utente
    //-------------------------------------------------------
    SETPAR[180, Soglie, Disabilita];              // disabilita nuove soglie
ELSE
    //-------------------------------------------------------
    //  di notte si forza il set-point di temperatura
    //-------------------------------------------------------
    SETPAR[180, Soglie, Abilita];                 // abilita nuove soglie
ENDIF

V2 = READ[180, 77];               // leggo lo stato del relè termostato
Pompa = V2.6;                     // comando pompa mandata
```

5.13 –Gestione cambio ora solare/legale

Nelle versioni di modulo master successive alla rev. 1.21.02 è possibile modificare l'ora di sistema da programma utente (vedi § 4.1). In tal modo si può programmare l'aggiornamento automatico dell'ora al passaggio ora legale/solare.
Tale possibilità è superata nel caso delle revisioni successive alla rev. 1.21.05 in cui è implementato il cambio automatico dell'ora legale.

Una possibile soluzione del problema è la seguente:

```
//-----------------------------------------------------------------------------------------
//   Anno          Inizio              Fine
//-----------------------------------------------------------------------------------------
// 2006            26 Marzo 02:00      29 Ottobre 03:00
// 2007            25 Marzo 02:00      28 Ottobre 03:00
// 2008            30 Marzo 02:00      26 Ottobre 03:00
// 2009            29 Marzo 02:00      25 Ottobre 03:00
// 2010            28 Marzo 02:00      31 Ottobre 03:00
//
// Le date di cambio ora legale/solare sopra riportate (2006 a parte) sono presunte.
// Infatti entro il 2007 la Commissione europea dovrà presentare una relazione per
// rilevare vantaggi e svantaggi dell'ora legale e stabilire se conviene mantenerla e/o
// modificarla.
//
// Per aggiornamenti sulle date di cambio ora legale consultare la Gazzetta ufficiale
// delle Comunità europee DIRETTIVA 2000/84/CE DEL PARLAMENTO EUROPEO E DEL CONSIGLIO
// del 19 gennaio 2001 concernente le disposizioni relative all'ora legale.
//
//
// N.B.
// Siccome questa macro utilizza un flag x ricordarsi se e' stato
// effettuato il passaggio da ora solare a ora legale
// e' consigliabile prevedere nel programma la direttiva RAMBACKUP.
// Infatti il controllo per il passaggio di ora viene fatto il giorno xx del
// mese yy alle ore 2 o 3 di notte, quindi se il Master fosse
// acceso/spento durante questa coincidenza oraria e lo stato del flag non
// e' memorizzato (RAMBACKUP) l'ora viene ogni volta decrementata.
//
//-----------------------------------------------------------------------------------------

DEFINE    Sol_Leg      V100.1

MACRO ORALEGALE(Flags)
{
     // 02:00 26 Marzo 2006
     IF[(RTC(YE)==6) & (RTC(MO)==3) & (RTC(DM)==26) & (RTC(HH)==2)]
          SETPAR[254, 100, RTC(HH)+1];
          Flags = OFF;
     ENDIF

     // 03:00 29 Ottobre 2006
     IF[(RTC(YE)==6) & (RTC(MO)==10) & (RTC(DM)==29) & (RTC(HH)==3) & !Flags]
          SETPAR[254, 100, RTC(HH)-1];
          Flags = ON;
     ENDIF

     // 02:00 25 Marzo 2007
     IF[(RTC(YE)==7) & (RTC(MO)==3) & (RTC(DM)==25) & (RTC(HH)==2)]
          SETPAR[254, 100, RTC(HH)+1];
          Flags = OFF;
     ENDIF
```

```
     // 03:00 28 Ottobre 2007
     IF[(RTC(YE)==7) & (RTC(MO)==10) & (RTC(DM)==28) & (RTC(HH)==3) & !Flags]
          SETPAR[254, 100, RTC(HH)-1];
          Flags = ON;
     ENDIF

     // 02:00 30 Marzo 2008
     IF[(RTC(YE)==8) & (RTC(MO)==3) & (RTC(DM)==30) & (RTC(HH)==2)]
          SETPAR[254, 100, RTC(HH)+1];
          Flags = OFF;
     ENDIF

     // 03:00 26 Ottobre 2008
     IF[(RTC(YE)==8) & (RTC(MO)==10) & (RTC(DM)==26) & (RTC(HH)==3) & !Flags]
          SETPAR[254, 100, RTC(HH)-1];
          Flags = ON;
     ENDIF

     // 02:00 29 Marzo 2009
     IF[(RTC(YE)==9) & (RTC(MO)==3) & (RTC(DM)==29) & (RTC(HH)==2)]
          SETPAR[254, 100, RTC(HH)+1];
          Flags = OFF;
     ENDIF

     // 03:00 25 Ottobre 2009
     IF[(RTC(YE)==9) & (RTC(MO)==10) & (RTC(DM)==25) & (RTC(HH)==3) & !Flags]
          SETPAR[254, 100, RTC(HH)-1];
          Flags = ON;
     ENDIF

     // 02:00 28 Marzo 2010
     IF[(RTC(YE)==10) & (RTC(MO)==3) & (RTC(DM)==28) & (RTC(HH)==2)]
          SETPAR[254, 100, RTC(HH)+1];
          Flags = OFF;
     ENDIF

     // 03:00 31 Ottobre 2010
     IF[(RTC(YE)==10) & (RTC(MO)==10) & (RTC(DM)==31) & (RTC(HH)==3) & !Flags]
          SETPAR[254, 100, RTC(HH)-1];
          Flags = ON;
     ENDIF
}

//---------------------------------------------------------------------
//                      Inizio Programma
//---------------------------------------------------------------------
RAMBACKUP;


ORALEGALE(Sol_Leg);
```