MANUALE D'USO
MICROPLC
MERLINO

CONTROLLORE A LOGICA PROGRAMMABILE

Manuale V.2.1
Marzo 99

C&P di Coppi Angelo.
ELETTRONICA E AUTOMAZIONE

C&P di Coppi Angelo
Elettronica e Automazione
53021 Abbadia San Salvatore
Siena
P.IVA 00846080521
Tel.0577 -777358

Le informazioni presenti in questo documento sono soggette a modifica a totale discrezione della ditta
C&P di Coppi Angelo

Questo manuale fa riferimento al MicroPLC Mod. MERLINO ed è esclusivo
per questo modello.

IMPORTANTE :

La ***C&P*** non risponde in alcun modo dei danni causati da un ***improprio*** uso sia dell' Hardware che del Software del MicroPLC MERLINO.

• INTRODUZIONE

Questo manuale fornisce tutte le istruzioni necessarie per l'installazione e la operatività del MicroPLC mod. **MERLINO.**

Contenuto della confezione:

1	***MICROPLC modello MERLINO***
1	***Floppy Disk 1,44 Mb contenente il Software***
1	***Manuale D'uso in formato elettronico pdf***

• PREFAZIONE

Il **MicroPLC MERLINO** è un controllore a logica programmabile atto a sostituire logiche elettromeccaniche complesse.Grazie alla implementazione di un linguaggio molto flessibile e potente ad un hardware molto ricco in termini sia di I/O che periferiche dedicate ha uno spettro di applicazioni molto vario anche in quelle applicazioni che si distinguono per complessità o velocità di risposta.
Con il software fornito ,la programmazione, per chi lavora in campo elettrotecnico , risulta intuitiva e getta un ponte tra schema a contatti in forma classica e programmazione avanzata.
Visto i numerosi componenti interni implementati e l'ottimo rapporto prezzo prestazioni il MicroPLC MERLINO risulta la scelta vincente in tutte quelle applicazioni dove la flessibilità di applicazione , tempi di intervento e scorte a magazzino sono parte prevalente del progetto.

• Caratteristiche tecniche:

Caratteristiche Generali

Tensione di alimentazione	20-28 VDC. (protezione contro l'inversione di polarità)
Corrente di alimentazione	350 ma MAX (protezione interna tramite fusibile ripristinabile)
Temperatura di lavoro	da 0 °C a 50°C
Temperatura di immagazzinamento	da –10°C a +60°C
Umidità relativa massima	da 30% all'80% senza condensa
Ritenzione dei dati	Per 5 anni in assenza di tensione tramite batteria al Litio interna a 25°C

Caratteristiche degli ingressi

Tensione di ingresso	da 12 a 28 VDC	
	tensione di ON	12VDC
	tensione di OFF	11VDC
Tipo di isolamento	Fotoaccoppiatore in corrente alternata	
Isolamento	500VAC per un minuto tra ingresso e massa	
Resistenza di ingresso	Circa 3KΩ	

Caratteristiche delle uscite

Tipo di uscita	Contatto a relè normalmente aperto
Massimo Carico	2 Ampere 250 VAC cosφ 1, 1Ampere 110 VDC per morsetti a vite 2Ampere 110VAC cosφ 1, 1Ampere 55 VDC per morsetti sfilabili.
Isolamento	500VAC per un minuto tra uscite e massa
Vita Meccanica	30000000 di cicli.
Vita Elettrica	>100000 cicli a 2Ampere 250 Vac

Caratteristiche degli ingressi analogici

Range	0-5 VDC
Impedenza ingresso	>=100khoms
Risoluzione	1024 punti
Sampling	10 hz
Protezione	Resistenza e diodi e VDR 5,5V 10A
Protezione EMI	Filtro a T su ogni I/O att.>40db a 10 Mhz

Caratteristiche delle uscite analogiche

Range	0-5 VDC
Risoluzione	255 punti
Sampling	Definibile dall'utente
Tipo di uscita	Emitter follower
Protezione EMI	Filtro a T su ogni I/O att.>40db a 10 Mhz

Caratteristiche dell'ingresso per encoder

Range	32bit con segno,relativo assoluot oppure 2 contatori separati
Frequenza Max impulsi	500 khz
Sampling	Definibile dall'utente da 25hz a 5Khz
Impdedenza ingresso	1Kohms livello TTL

Caratteristiche delle porte di comunicazione

Standard	RS-485 a 4 fili differenziale
Massima distanza	1Km su cavo twistato
Velocità di comuniucazione	da 600 baud a 115,2 Kbaud selezionabile via Software
Protezione ESD	5Kv –Human Body Model
Protezione EMI	Con Filtro a T su ogni I/O att.>40db a 10 Mhz

Caratteristiche Protezioni Funzionamento

Watch-Dog Software	Reset dopo 523 ms dall'ultimo refresh,ripartenza automatica
Watch-Dog Hardware	Intervento a 4,5Volt alimentazione µP,ripartenmza automatica

Caratteristiche generali del sistema

Software	Linguaggio Interpretato appositamente studiato per realizzzare istruzioni potenti e veloci. Gestione di 10 tipi di dati e 7 modi di indirizzamento per un totale di circa 600 istruzioni di base, tipologia del linguaggio comparabile con assembler ad alto livello per CPU a 32 bit.(es 68000 PDP11) Implementazione di 1000 registri di sistema e gestione di tutte le operazioni registro registro. (registri come memoria)Implementazione di processore logico per istruzioni ad 1 bit per istruzioni accumulatore registro.Tutti i registri possono essere letti come contatti e scritti come bobine per totale di 32000 rele.Gestione di registri specifici come ingressi uscite timer etc. Possibilita per l'utente di interagire direttamente con libreria di funzioni di sistema, registri della CPU fisica e 3 interrupt utente.Gestione completa del linguaggio da programma compilatore

• INSTALLAZIONE HARDWARE

Descrizione dei Connettori

Il **MIcroPLC MERLINO** Viene fornito in un contenitore modulare standard DI56001 con connettore a vite a 36 poli.(esiste anche versione con morsetti estraibili)

TABELLA CONNESSIONI**:**

N°	**DESCRIZIONE**	**N°**	**DESCRIZIONE**
1	COMUNE INGRESSI	***19***	TX-
2	INGRESSO 1	***20***	TX+
3	INGRESSO 2	***21***	RX-
4	INGRESSO 3	***22***	RX+
5	INGRESSO 4	***23***	INGRESSO ANALOGICO 0 (OUT AN. 0)
6	INGRESSO 5	***24***	INGRESSO ANALOGICO 1 (OUT AN. 1)
7	INGRESSO 6	***25***	INGRESSO ANALOGICO 2
8	INGRESSO 7	***26***	COMUNE INGRESSI ANALOGICI
9	INGRESSO 8	***27***	**ALIMENTAZIONE +VCC (16.. 28VDC)**
10		***28***	**ALIMENTAZIONE GND + EARTH**
11	USCITA 7 NA	***29***	USCITA 0 NA
12	USCITA 7 C	***30***	USCITA 0 C
13	USCITA 3 NA	***31***	USCITA 4 NA
14	USCITA 3 C	***32***	USCITA 4 C
15	USCITA 6 NA	***33***	USCITA 1 NA
16	USCITA 6 C	***34***	USCITA 1 C
17	USCITA 2 NA	***35***	USCITA 5 NA
18	USCITA 2 C	***36***	USCITA 5 C

Display LCD

Il pannello frontale e' rimovibile per permettere l'accesso agli 8 tasti di sistema.All'accensione della macchina dopo i test funzionali di sistema la macchina se ha un programma valido in memoria lo esegue altrimenti si posiziona nello stato di stop. In questa fase possono essere impostati vari parametri di sistema e sono accessibili tutte le 100 pagine definite nel sistema.

Istallazione

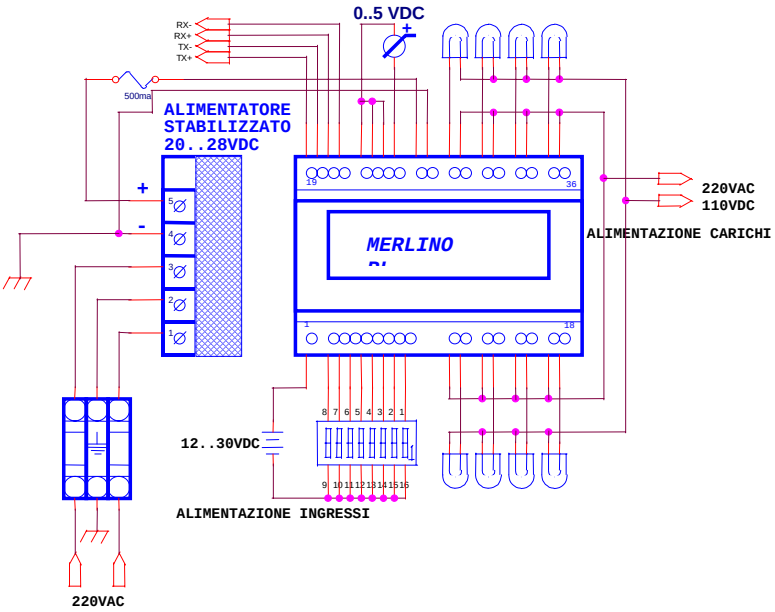
Connettere ai morsetti di alimentazione N° 26 e N°27 una tensione continua da **16VDC a 28 VDC MAX**
con un ripple inferiore a ±10% con una corrente di almeno 400 ma separata galvanicamente dal resto del circuito.Il polo di alimentazione Negativo **deve essere collegato a terra** per poter scaricare eventuali distrurbi intercettati dal filtro EMI di alimentazione interno in ingresso del PLC

ATTENZIONE!

Calcolare la massima corrente per ogni cavo e seguire le appropriate procedure di cablaggio.L'inosservanza di queste misure può causare gravi danni alle persone e alla macchina

Spegnere il MicroPLC prima di collegare i circuiti
Tuti i cavi di segnali a basso livello devono essere stesi separatamente dagli altri circuiti
I circuiti in AC devono essere separati dai circuiti in CC.
I circuiti non devono essere cablati vicino a dispositivi che possono essere una potenziale fonte di interferenze elettriche.Se si verificano gravi problemi di disturbo può darsi che sia necessario usare ulteriore filtraggio dell'alimentazione.
Etichettare sempre tutti i cavi da e per tutti i circuiti di ingresso uscita.

ESEMPIO INSTALLAZIONE



ATTENZIONE!

Tutti i circuiti di sicurezza del sistema non devono passare attraverso il MicroPLC. Tutti gli arresti di emergenza e/o i blocchi di sicurezza devono agire direttamente sui circuiti di alimentazione ausiliari. Possono entrare all' interno del plc solo per funzioni di monitoraggio, senza alcun effetto operativo.

Il MicroPLC appena ricevuta alimentazione esegue il reset della macchina e si predispone al funzionamento,Vengono eseguiti in successione test dei tasti,test della Ram,test della tensione di programmazione, controllo dello stato della memoria di programma e nel caso vi sia un programma valido nella memoria flash viene eseguito.Il MicroPLC può essere impostato in stop in qualsiasi momento premendo contemporamenamente i 4 tasti dul lato destro sotto il pannello frontale.In questo caso il PLC viene forzato in stop con velocita di comunicazione COM1 a 115Kbaud e numero PLC (della rete) a 1.

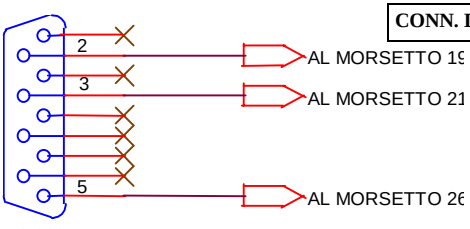
Connessioni verso Personal Computer o PLC master

Collegare i cavi di connessione seriale nel seguente modo e controllare la disposizione dei segnali sul vostro convertitore RS-232 >> RS-485

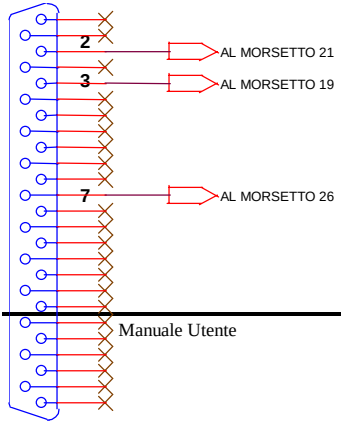
Morsetto	
21	RX+
20	RX-
19	TX+
18	TX-

Connessioni in standar RS-232

Il ***microPLC Merlino*** supporta lo standard RS-485 e non lo standard RS-232.Eseguendo il seguente collegamento si riesce generalmente ad eseguire un collegamento anche se per brevi distanze (2metri)

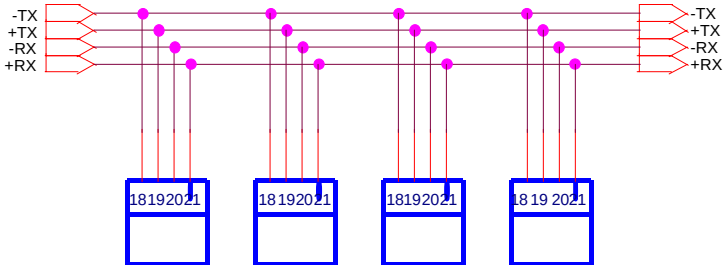


DB9	Morsetto
2	**19**
3	**21**
5	**26**



DB25	Morsetto
2	**21**
3	**19**
7	**26**

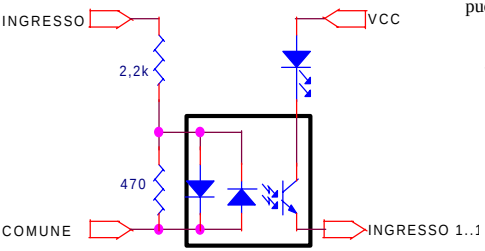
Se si effettua il collegamento di un MicroPLC in una rete di MicroPLC collegare tutti e 4 i fili in parallelo a tutti i dispositivi. Esempio di connessione in rete



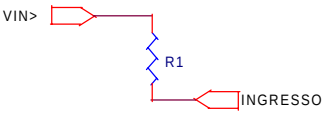
Connessioni degli ingressi digitali

Gli ingressi accettano una tensione da 12VDC a 28VDC , a comune si può mettere sia il polo positivo che il polo negativo.Un ingresso viene visto come chiuso(ON) quando gli viene applicata una tensione ai morsetti corrispondenti.

Il circuito di ingresso è il seguente



Per aumentare la tensione in ingresso si può inserire un resistenza esterna.



VALORI DI R1

48 VDC	9,1K-0,5WATT
110 VDC	22K-1WATT

Connessioni delle uscite digitali

Le uscite sono contatti normalmente aperti di relè fisici interni. La portata ammessa massima è di 2Ampere a 250 VAC cos φ 1 o 1 Ampere 110 VDC. (morsetti a vite)

All'interno del MicroPLC non è presente nessun circuito atto alla eliminazione dei disturbi generati nella chiusura del contatto .

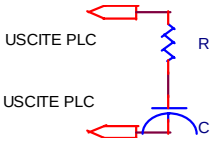
Questi disturbi possono essere causati da fenomeni transitori in presenza di carichi fortemente induttivi specialmente se alimentati in corrente continua. Questi disturbi, di persè non sono nocivi al MicroPLC , se correttamente alimentato tramite un alimentatore separato galvanicamente dal sistema, visto che all'interno della macchina sono presenti una serie di circuiti atti a eliminare questo inconveniente.

Ma poiché la natura e l' ampiezza di questi fenomeni non è prevedibile a priori, perché dipendente da un numero di fattori che va al di là della semplice tensione di alimentazione dei carichi (disposizione dei componenti, passaggio dei cavi,fenomeni indotti sia elettrici che meccanici) è sempre buona norma cercare di eliminare il più possibile la generazione degli stessi.

Esistono in commercio tutta una serie di accessori adatti a questo scopo :

Contattori a basso assorbimento appositamente studiati per l'uso con PLC
Zoccoli per relè con filtro incorporato
Cavi schermati etc.

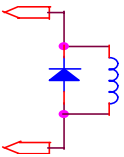
Generalmente per ovviare a questo fenomeno basta costruire un semplice circuito da porre in parallelo ai contatti di uscita o in parallelo al carico.
Il circuito più comunemente usato da porre in parallelo ai contatti è una resistenza in serie ad un condensatore:



VALORI RC COMUNEMENTE USATI

V	I	R	C
24	2A	100 ohm	0,01 mf 100v
48	2A	100 ohm	0,01 mf 250v
110	2A	100 ohm	0,47 mf 400 v
220	2A	200 ohm	0,22 mf 630v

Attenzione :
Il filtro sopradescritto quando usato in circuiti alimentati a corrente alternata in parallelo ai contatti N.A. lascia passare una certa corrente residua ,dovuta alla impedenza del filtro.Se si alimenta carichi a bassa corrente questa corrente può mantenere il carico in fase di rilascio, inoltre a contatto aperto il filtro lascia scorrere una debole corrente attraverso i suoi capi, per questo motivo conviene mettere il filtro (se necessario) sul carico.
Se i carichi sono induttivi a corrente continua è buona norma mettere in parallelo al carico un diodo polazizzato inversamente in modo da cortocircuitare la tensione inversa generata al rilascio.



DIODI COMUNEMENTE USATI

24 VDC	2A	IN4004
48 VDC	2A	IN4004
110 VDC	1A	IN4007
220 VDC	0,5 A	IN4007

Esempio:

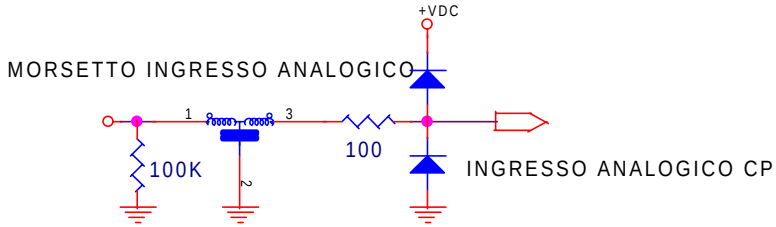
Nel circuito stampato della base del MicroPLC sono presenti due pad liberi per ogni contatto in modo da poter inserire un componente qualora si renda necessario (diodo ,VDR o Condensatore) all'interno del PLC

Connessione ingressi analogici
Il convertitore analogico digitale è integrato nel microcontrollore per cui bisogna porre particolare attenzione ai collegamenti con questo dispositivo ,infatti una eccessiva tensione di ingresso può distruggere sia il canale analogico che tutto il processore impedendo il funzionamento del sistema.
E' buona norma generalmente mettere degli isolatori di segnale agli ingressi analogici, con tali dispositivi non sono necessarie altri accorgimenti,visto che eventuali anomalie del segnale sono a carico di questi

dispositivi.Nel caso che si intenda collegare direttamente un segnale all'ingresso analogico bisogna assicurarsi che in alcun modo la tensione di ingresso possa avere valori superiori ai
6 volt

ATTENZIONE

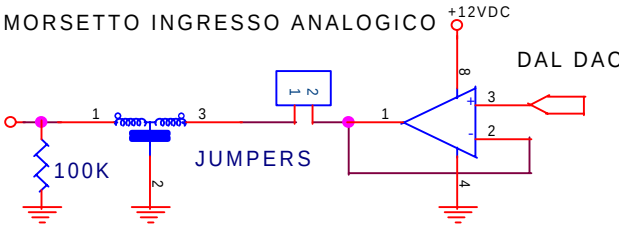
Gli ingressi analogici sono canali ad alta impedenza(>=100K) per cui è facile indurre tensioni indotte o Ricevere alti segnali di rumore specialmente per collegamenti a distanza.
Si consiglia in fase di cablaggio di separare i collegamenti che portano segnali analogici dal resto dei circuiti e eseguire un cablaggio a se stante mentre per segnali che arrivano dal campo misure con distanze apprezzabili ,se non è proprio possibile istallare un separatore galvanico di segnale,almeno procedere alla protezione del canale analogico tramite soppressori di segnale a 5VDC.
Segue schema di ingresso dei primi tre canali analogici:



Connessioni uscite analogiche.
Il ***microPLC Merlino*** mette a disposizione del programmatore anche due uscite analogiche a 8 bit.quando si imposta l' uscita analogica 0 (1) si perde il canale anlogico 6 (7). Nella versione startdard della macchina le uscite analogiche non sono fisicamente connesse ai morsetti
di uscita .Il segnale è comunque presente nel connettore di espansione.Allo scopo di agevolare applicazioni minime ove non necessita un numero elevato di morsetti e quindi di acquisto della espanzione hardware ,ad esempio se sia richiesto una sola uscita analogica e un solo ingresso è presente dentro al PLC un connettore che permette attraverso jumpers di connettere l'ingresso analogico 0 (1) all' uscita analogica(0).In questo modo si perde fino a 4 canali analogici ma non è necessaria nessuna aggiunta hardware.Il connettore in questione si trova ,guardando il PLC dall'alto in basso a destra nella scheda centrale sotto al connettore della tastiera .
pin out del connettore dip

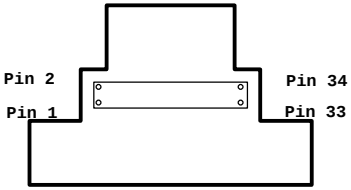
1	2	3	4	5
Riprogrammazione Sistema Operativo			Out.An. Zero Al morsettto 22	Out.An. Uno Al morsettto 23

Segue schema di uscita canali analogici:



Sul lato destro della macchina si trova un connettore a 34 poli per le espansioni.Le connessioni sono le seguenti:

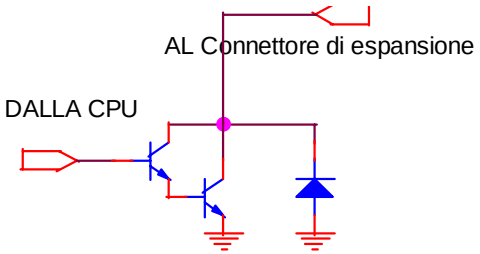
Pin	Qualifica
1	Ingresso analogico n°3
2	Ingresso analogico n°5
3	Ingresso analogico n°4
4	Ingresso analogico n°6(DAC 0)
5	Uscita a collettore aperto n°9
6	Ingresso analogico n°7 (DAC 1)
7	Uscita a collettore aperto n°10
8	Ingresso TTL n°16
9	Uscita a collettore aperto n°11
10	Ingresso TTL n°15
11	Uscita a collettore aperto n°12
12	Ingresso TTL n°14
13	Uscita a collettore aperto n°13
14	Ingresso TTL n°13
15	Uscita a collettore aperto n°14
16	Ingresso TTL n°12
17	Uscita a collettore aperto n°15
18	Ingresso TTL n°11
19	Uscita a collettore aperto n°16
20	Ingresso TTL n°10
21	Uscita a collettore aperto n°17
22	Ingresso TTL n°9



Pin	Qualifica
23	Uscita a collettore aperto n°18
24	Ingresso TTL n°17
25	Trasmissione COM0
26	Ingresso TTL n°18
27	Ricezione COM0
28	Ingresso Encoder TTL A
29	CTS Com 0
30	Ingresso Encoder TTL B
31	+5VDC 50 ma
32	+12VDC 150 ma
33	GND sistema
34	GND sistema

Uscite connettore di espansione

Le uscite del connettore di espansione sono abilitate e funzionano a collettore apeto.Massima tensione applicabile 30VDC con un carico per canale non superiore 50ma. Tipologia:

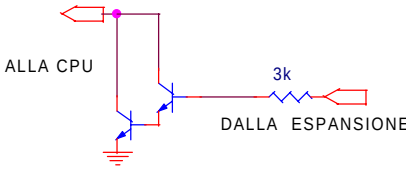


Ingressi connettore di espansione.

Gli ingressi del connettore di espansione sono a livello TTL e sono direttamente connessi alla CPU.Per questo ,nel caso sia necessario avere più ingressi a disposizione bisogna implementare un circuito a fotoaccoppiatore allo scopo di separare galvanicamente la macchina dal mondo esterno.

Questo tipo di ingressi se necessario possono anche essere riconfigurati come uscite ausiliarie.Allo scopo di facilitare tale operazione vedere la **sys** ExtOut nel capitolo "**Funzioni di sistema**".

Ingresso Encoder connettore espansione
L'ingresso per encoder è a livello TTL ed è connesso alla CPU tramite un buffer.La massima frequenza ammessa è 500 Khz.



Porta Com 0 connettore espansione
Questi pin sono connessi direttamente alla CPU e non possono essere usati se non attraverso un opportuno drivers che da livello TTL trasformi i segnali in standard RS-485 o RS-232

Ubicazione .
Generalmente l'ambiente di lavoro del MicroPLC sarà un quadro elettrico.
E' buona norma cercare una disposizione della macchina la più isolata possibile
dai seguenti fenomeni di interferenza:

Generatori di calore quali scaldiglie,trasformatori a pieno carico, etc.
Generatori di forti vibrazioni meccaniche quali contattori di potenza etc.
Generatori di armoniche quali azionamenti o avviatori trifasi di potenza.
Generatori di microonde quali circuiti risonanti R L C.
Ambienti a forte schock climatici.

- ***Sistema Operativo***

Il ***microPLC Merlino*** è un PLC basato su un microcontrollore a 16\32 bit dell'ultima generazione in tecnologia flash. Grazie a questa tecnologia si possono memorizzare direttamente i programmi all'interno del microcontrollore rendendo il sistema molto immune al rumore . Grazie agli accessi alla memoria a zero stati di attesa(16 Mhx, 125 ns istruzione , istruzione maxxima 3 cicli di clockj) il sistema risulta essere molto veloce. Inoltre questa tecnologia mette a disposizione degli utenti la possibilita' di aggiornamenti software a livello di sistema operativo potendo riprogrammare tutta la memoria del processore direttamente dall'utente via seriale .Questo salto di qualita della CPU non poteva essere pienamente recepito dall'utente finale con un linguaggio di programmazione derivato da macchine a 8 bit.Per questo e' stato indirizzato un particolare sforzo tecnico nello sviluppo di un linguaggio che possa offrire la piena potenzialita dell ' hardware a disposizione.
Nel sistema operativo sono stati implementati diversi processi che possone essere così suddivisi:

- Processo principale
- Comunicazione Seriale canale 0 e 1
- Timer di sistema
- Gestione Encoder
- Gestione Errori Hardware
- Gestione Interrupt Utente
- Programmazione Memoria Flash

Questi processi sono in parte gestiti tramite interrupt assicurando grazie alla velocità del microcontrollore una apprezzabile prestazione del sistema.
Il sistema operativo occupa circa 90Kbyte di memoria flash lasciando così a disposizione dell'utente 32 kb di flash per il codice di programma.Il linguaggio di programmazione implementato è un ***interprete di linguaggio assembler*** . Questa scelta è stata decisa per due motivi principali il primo è di non specializzare in modo eccessivo il sistema nel solo campo dei PLC con un linguaggio troppo orientato a questo mondo, mentre il secondo è quello di ottenere un linguaggio comunque potente. Inoltre tutte le istruzioni implementate sono state ottimizzate per una esecuzione alla massima velocità possibile,qtutte le istruzioni sono chiamate a funzioni ottimizzate e in genere non occupano più di 2 microsendi per l'esecuzione . Grazie ad un notevole numero di registri messi a disposizione del programmatore e alla totale ortogonalità delle istruzioni, il linguaggio risulta semplice e molto potente.Infine questo linguaggio lascia aperta la strada per una implementazione di compilatori ad alto livello , ad esempio ladder basic, pascal o C ,che usino questo linguaggio come risultato finale.

- **Linguaggio Assembler**

Il ***microPLC Merlino*** mette a disposizione per il programmatore 1000 registri generali a 32 bit come memorie di sistema,900 accessibili e 100 come stack di sistema, 7 modi di indirizzamento dati e 10 formati diversi .Tutti i registri possono essere manipolati nello stesso modo senza distinzione , ma i primi 300 sono utilizzati dal sistema operativo. Infatti in questi primi registri vengono memorizzati tutte le variabili necessarie al corretto funzionamento del sistema. Queste stesse variabili sono condivisibili direttamente con l'utente in modo trasparente. Ad esempio il registro numero 20 (R(20)) contiene lo stato delle uscite . Impostando questo registro vengono attivate automaticamente le corrispondenti uscite fisiche.
Ad ogni registro si può' accedere con diversi formati definiti dalla quantità in bit del registro che si vuole modificare.
La dimensione di default e' quella a 32 bit ,tale dimensione e' definita "**long**" e viene identificata con la estensione "**.l**". Ogni dato **long** e' costituito da due dati interi che sono definiti come "**low word**" per i 16 Bit meno significativi e "**high word**" per i 16 Bit piu' significativi. Questi dati sono rispettivamente indentificati dall'estensione "**.lw**" e "**.hw**". Il dato a 16 bit **low word** e' a sua volta costituito da due dati a 8 bit ciascuno ,gli 8 bit meno significativi sono definiti come "**low low byte**" e vengono individuati dalla estensione "**.llb**" , mentre gli 8 bit piu significativi sono definiti come "**low high byte**" e vengono individuati dalla estensione "**.lhb**". Anche il dato a 16 bit **high word** e' a sua volta costituito da due dati a 8 bit ciascuno ,gli 8 bit meno significativi sono definiti come "**high low byte**" e vengono individuati dalla estensione "**.hlb**" , mentre gli 8 bit piu significativi sono definiti come "**hgh high byte**" e vengono individuati dalla estensione "**.hhb**".
Riassumendo , possiamo schematizzare la seguente tabella :

31								**long**							**0**
31			**high word**				**16**	**15**			**low word**				**0**
31	**hhb**		**24**	**23**	**hlb**		**16**	**15**	**lhb**		**8**	**7**	**llb**		**0**

Grazie a questo tipo d'accesso ai dati si può decidere la quantità di bit che l'operazione che si vuole compiere può modificare.
Ad esempio ,per sottolineare la libertà che offre questo sistema di programmazione possiamo con un istruzione di rotazione ,ruotare appunto di una posizione solo un bit nel dato hlb (16..23) in una sola operazione (rotr.hlb r(n)), questa stessa operazione utilizzando anche assembler evoluti su processori reali a 32 bit richiede piu istruzioni .
Ad ogni registro si può accedere anche come singolo bit. questo formato non ha un'estensione ma è identificato da una serie d'istruzioni dedicate allo scopo. Mentre per le operazioni sui registri non e' necessario un registro con la funzione d'accumulatore per le operazioni a bit questo e indispensabile, per avere una corrispondenza univoca tra lo schema disegnato e il programma ottenuto. Nel sistema operativo la funzione d'accumulatore è assegnata al bit zero del registro numero 9. Ad esempio l'istruzione **ldb r(n).x** legge il bit alla posizione **x** (0..31) del registro **n** (0..900) lo stato del bit viene scritto sul **bit 0 di r(9)**.
Sono definiti anche i formati "**word"** e "**byte"** e vengono identificati rispettivamente dalla estensione "**.w**" e "**.b**" e hanno lo stesso significato di **low word** e **low low byte** servono solo in abbinamento ad istruzioni di collegamento con la CPU fisica del PLC e proprio per evidenziare questa diversità sono stati immessi nel sistema
Ogni istruzione ha diversi modi d'accesso ai dati, i piu importanti
Sono tre:

- **Per valore** **[#]**
- **Su memoria diretta** **[R]**
- **Indiretto su Memoria** **[@R]**

Inoltre poiché un'istruzione ha fino un massimo di tre dati si possono avere una o piu combinazioni dei modi descritti.
Ad esempio l'istruzione **mov.l #123456789, R(100)** imposta i 32 bit della memoria **R(100)** con il valore **123456789.**

Il PLC riconosce 80 istruzioni diverse che combinate per estensione e modo d'indirizzamento forniscono circa 600 diverse istruzioni.
L'area di programma di sistema va da 0 a 0x7fff ,questo segmento di memoria è mappato nel processore dall'indirizzo fisico 0x18000 a 0x1ffff. Questo serve al programmatore perché non avendo specializzato il linguaggio è valido qualsiasi indirizzo a 32 bit come sorgente ,quindi se ad esempio si definisce una stringa nel codice di programma viene preso dal linguaggio come indirizzo sorgente lo spiazzamento sul programma dell'indirizzo della stringa sommato a 0x18000. L'area di memoria ram del sistema è di 4Kbyte 1000 long .Quest'area è tutta dedicata ai registri di sistema.Gli ultimi 100 registri sono dedicati allo stack e sono più che sufficenti se non sono abilitati interrupt utente.Nel qual caso bisogna monitorare i registri dopo l'esecuzione di programma o comunque lascire circa 15 registri per ogni interrupt utente utilizzato .Gli accessi ai registri possono essere diretti, indiretti o attraverso istruzioni di I/O (WR oRD).Per specificare un registro in una istruzione che lo prevede bisogna solo passarlo come numero e non come indirizzo. Ovvero se un parametro di una qualsiasi istruzione prevede un registro è ha valore 345 significa che è riferito a R(345).Questa notazione vale anche per il modo indiretto ovvero se il registro r(n) contiene 100 (r(n)=100) un indirizzamento indiretto su r(n) (@r(n)) significa che interesserà il registro r(100) perché r(n) punta a r(100).Il funzionamento delle istruzioni di I/O è diverso perché queste istruzioni sono state implementate come collegamento alla CPU fisica del sistema e debbono fare riferimento alla mappatura hardware del sistema.La memoria RAM inizia a 0xfef10 sino a 0xfff10.Proprio per questo motivo se si passa come valore un registro, (#r(n)) il linguaggio assembler decodifica questa richiesta e ritorna l'indirizzo fisico del registro 0xfef10+n*4.
Mappa del sistema operativo:

Sistema operativo	
Flash Eprom	Address
Sistema Operativo	0x00000 0x17FFF
Area Programmi Utenti	0x18000

Sistema operativo	
RAM	Address
Registro 0	0xFEF10
Registro 1000	0xFFF10

Tramite le istruzioni di I/O si puo rimappare anche il funzionamento dei registri fisici della CPU.
Ad esempio per impostare un ingresso come uscita.Per avere una mappa completa della CPU fisica fare riferimento ai manuali tecnici e di programmazione della CPU utilizzata nel sistema ovvero H3048F16 Hitachi.

- **Sintassi**

Visto che il linguaggio non implica nessun registro specializzato per scopi particolari almeno dal punto di vista software bisogna analizzare alcune particolarità .
Buona parte delle istruzioni implicano un registro sorgente e un registro destinazione,l'operazione che si esegue ha come uscita il registro destinazione.Ad esempio l'istruzione :

ADD.L B, A

Equivale a

A = A+ B

Che espressa in altri linguaggi ha la notazione :

C PASCAL BASIC

A+=B A:=A+B A=A+B

Bisogna subito puntualizzare che l'ordine dei parametri è essenziale , infatti l'istruzione del tipo

ADD.L A, B

Equivale a

B = B+ A

Si può generare confusione nei confronti se non si analizza con precisione quale parametro è interessato dal test.Ad esempio:

GTR.L A, B, indirizzo

Esegue un salto ad **indirizzo** se è vero che

A > B

mentre

GTR.L B, A, indirizzo

È esattamente l'opposto e esegue un salto ad **indirizzo** se è vero che

B > A

Il linguaggio assembler in dotazione alla macchina (*.esm) ha poche direttive che sono così riassunte:

DB	Definisce un byte nell'area programma es: DB 10,0xfe,0b11110101 Le virgole indicano una successiva definizione
DW	Definisce un intero a 16 bit nell'area programma es: DW 10,0xfe45,0b11110101001111 Le virgole indicano una successiva definizione
DM	Definisce una stringa di caratteri DM "Hello World",10,13 Le virgole indicano o una successiva dichiarazione o la presenza di una definizione di byte
DL	Definisce un long che può essere un valore o un puntatore (a qualsiasi oggetto).Inoltre accetta anche la sintassi Che definisce un numero floating point.Offset tra le varie dichiarazioni è 4.Es DL 123456789,0x1fffffff,r(300),r(300)+2 DL 0f1234.2345 Le virgole indicano una successiva dichiarazione.
EQU	Definisce che una etichetta è un valore non un indirizzo.Es: ACCBIT: EQU 9 Associa ad ACCBIT il valore 9.Ad ogni successiva valutazione del campo ACCBIT viene ad esso sostituito il valore 9.
#include "def"	Prima di elaborare il file corrente ingloba il file <**def**>
Etichette	Ogni nome di lunghezza inferiore a 26 caratteri è una etichetta ovvero è un indirizzo che deve essere definito nello svolgimento del programma.La definizione si ottiene facendo seguire il nome dai due punti es: jbz salto

.
salto:
Il programma assimila tutti i nomi e le istruzioni a caratteri grandi per cui scrivere salto o SalTO è identico.Nella linea di codice che definisce una etichetta si può aggiungere solo un commento ,**impostare eventuali istruzioni genera un errore.**

Commenti Qualsiasi carattere preceduto da punto e virgola è un commento al programma.Il programma salta tutti i caratteri presenti fino a che non incontra una nuova linea.

Come tutti i PLC anche il ***microPLC merlino*** basa il suo funzionamento sulla scansione di un programma in un loop infinito.In tutti i microPLC della C&P il ciclo di scansione è definito da una istruzione di tipo START per inizio scansione e END per la fine.Nel programma possono trovarsi più istruzioni di questo tipo , nel qual caso si ha un loop dinamico in base al contesto del programma.
Bisogna bene assimilare quando si programma con questo tipo di logica i concetti di sincronismo tra il programma e gli eventi generati da altri processi onde prevenire errori nella programmazione.Alla esecuzione dell'istruzione start sono letti gli ingressi ,letti i contatti dei timer,degli orologi etc,per cui questi contatti avranno uno stato stabile per tutta una scansione di programma e non possono generare ambiguità.Se però ad esempio si legge lo stato di un contatto speciale a 1/10 sec più volte entro la scansione di un programma lo stato di questo contatto può nelle varie parti del programma essere diverso perché questi bit sono impostati direttamente dalla routine di interrupt del sistema.Per ovviare a questo inconveniente vi è un sistema semplicicssimo,appoggiare su un rele interno questo contatto e usare i contatti del relè al posto di quelli del contatto speciale.In questo modo la lettura in diversi punti del programma del solito contatto darà sempre un risultato unico.Questa tecnica è generale è deve essere utilizzata ogni qualvolta si usa contatti che possono essere variabili(eventi asincrooni) durante la scansione del programma.Ricordarsi però che un qualsiasi contatto asincrono deve essere sincronizzato solo se compare lo stesso più volte entro il loop di scansione programma.Segue tabella che indica quali sono i contatti sincroni e asincroni

Sincroni	**Asincroni**
Contatori	Contatti speciali
Timer	Tasti
Rele	Tutte le memorie ad interrupt
Uscite	
Ingressi	
Orologi	
Tasto con Fronte	

Nel prossimo capitolo sono elencate e spiegate tutte le istruzioni presenti nel sistema.

Complemento di un bit in modo assoluto **ABSCPLB**

Istruzione	Sorgente	Destinazione	Esempio
ABSCPLB		R(nnn).nbit	ABSCPLB R(430).04
ABSCPLB	R(nnn)	R(nnn)	ABSCPLB R(300),R(301)
ABSCPLB		@R(nnn).nbit	ABSCPLB @R(400).31
ABSCPLB	@R(nnn)	R(nnn)	ABSCPLB @R(300),R(301)

Descrizione:

Questa istruzione esegue il complemento del bit indicato da nbit (numero che va da 0 a 31) all'interno di un registro indicato con la sintassi R(nnn) (valore compreso nei range da 0 a 900)

ESEMPI:

	Bit da 31…..0 della locazione R(300)
ABSCPLB R(300). 0	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.0
Dopo l'esec. Della istruz.	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.1

	Bit da 31…..0 della locazione R(300)
ABSCPLB R(300),R(301)	1.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x
Contenuto del registro R(301)	31
	Bit da 31…..0 della locazione R(300)
Dopo l'esec. Della istruz.	0.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x

	Bit da 31…..0 della locazione R(450)
ABSCPLB @R(300).10	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.0.x.x.x.x.x.x.x.x.x.x.x
Contenuto del registro R(300)	450
	Bit da 31…..0 della locazione R(450)
Dopo l'esec. Della istruz.	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.1.x.x.x.x.x.x.x.x

ABSCPLB @R(300),R(301)	320
Contenuto del registro R(301)	31
	Bit da 31…..0 della locazione R(320)
Prima dell'esec. della istruz.	0.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x
Dopo l'esec. della istruz.	1.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x

Reset di un bit in modo assoluto **ABSRESB**

Istruzione	Sorgente	Destinazione	Esempio
ABSRESB		R(nnn).nbit	ABSRESB R(430).04
ABSRESB	R(nnn)	R(nnn)	ABSRESB R(300),R(301)
ABSRESB		@R(nnn).nbit	ABSRESB @R(400).31
ABSRESB	@R(nnn)	R(nnn)	ABSRESB @R(300),R(301)

Descrizione:

Questa istruzione forza allo stato 0 il bit indicato da nbit (numero che va da 0 a 31) all'interno di un registro indicato con la sintassi R(nnn) (valore compreso nei range da 0 a 900)

ESEMPI:

	Bit da 31…..0 della locazione R(300)
ABSRESB R(300). 0	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.1
Dopo l'esec. Della istruz.	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.0
	Bit da 31…..0 della locazione R(300)
ABSRESB R(300),R(301)	1.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x
Contenuto del registro R(301)	31
	Bit da 31…..0 della locazione R(300)
Dopo l'esec. Della istruz.	0.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x
	Bit da 31…..0 della locazione R(450)
ABSRESB @R(300).10	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.1.x.x.x.x.x.x.x.x.x
Contenuto del registro R(300)	450
	Bit da 31…..0 della locazione R(450)
Dopo l'esec. Della istruz.	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.0.x.x.x.x.x.x.x.x.x
ABSRESB @R(300),R(301)	320
Contenuto del registro R(301)	31
	Bit da 31…..0 della locazione R(320)
Prima dell'esec. della istruz.	1.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x
Dopo l'esec. della istruz.	0.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x

Set di un bit in modo assoluto **ABSSETB**

Istruzione	Sorgente	Destinazione	Esempio
ABSSETB		R(nnn).nbit	ABSSETB R(430).04
ABSSETB	R(nnn)	R(nnn)	ABSSETB R(300),R(301)
ABSSETB		@R(nnn).nbit	ABSSETB @R(400).31
ABSSETB	@R(nnn)	R(nnn)	ABSSETB @R(300),R(301)

Descrizione:

Questa istruzione forza allo stato 1 il bit indicato da nbit (numero che va da 0 a 31) all'interno di un registro indicato con la sintassi R(nnn) (valore compreso nei range da 0 a 900)

ESEMPI:

	Bit da 31…..0 della locazione R(300)
ABSSETB R(300). 0	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.0
Dopo l'esec. Della istruz.	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.1
	Bit da 31…..0 della locazione R(300)
ABSSETB R(300),R(301)	0.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x
Contenuto del registro R(301)	31
	Bit da 31…..0 della locazione R(300)
Dopo l'esec. Della istruz.	1.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x
	Bit da 31…..0 della locazione R(450)
ABSSETB @R(300).10	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.0.x.x.x.x.x.x.x.x.x.x.x
Contenuto del registro R(300)	450
	Bit da 31…..0 della locazione R(450)
Dopo l'esec. Della istruz.	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.1.x.x.x.x.x.x.x.x.x.x

ABSSETB @R(300),R(301)	320
Contenuto del registro R(301)	31
	Bit da 31…..0 della locazione R(320)
Prima dell'esec. della istruz.	0.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x
Dopo l'esec. della istruz.	1.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x

Somma aritmetica ADD

Istruzione	Sorgente	Destinazione	Esempio
ADD.*	#Valore	R(nnn)	ADD.L #1000,R(300)
ADD.*	R(nnn)	R(nnn)	ADD.L R(300),R(301)
ADD.*	R(nnn)	@R(nnn)	ADD.L R(300),@R(301)

* Formato Istruzione

HHB	HLB	LHB	LLB
HW		LW	
L			

Dimensione formato
Byte
Word
Long

Descrizione:

Questa istruzione esegue la somma tra l'operando sorgente del formato prestabilito (BYTE, WORD, LONG) ed il registro destinazione. Mette il risultato all'interno del registro destinazione specificato. Nell' operando sorgente possono essere specificati i seguenti formati :

Decimale	Esadecimale	Binario
#VALORE	# 0xVALORE	# 0bVALORE

R(nnn) = numero che indica un registro interno, ed è compreso tra 0 e 900.

ESEMPI:

	HHB	HLB	LHB	LLB	
Prima dell' l'esec. della istruz.	x	x	x	100	Contenuto R(300)
ADD.LLB #100, R(300)					
Dopo l'esecuzione della istruz.	x	x	x	200	Contenuto R(300)

	HW	LW	
Prima dell' l'esec. della istruz.	1000	X	Contenuto R(320)
ADD.HW #01200, R(320)			
Dopo l'esecuzione della istruz.	2200	X	Contenuto R(320)

	R(310)	R(320)
Prima dell' l'esec. della istruz.	1800	3200
ADD.L R(310), R(320)		
Dopo l'esecuzione della istruz.	1800	5000

	R(310)	R(320)
Prima dell' l'esec. della istruz.	135000	1000
Contenuto di R(300)	320	
ADD.L R(310), @R(300)		
Dopo l'esecuzione della istruz.	135000	136000

And logico **AND**

Istruzione	Sorgente	Destinazione	Esempio
AND.*	#Valore	R(nnn)	AND.LW #0FF00,R(300)
AND.*	R(nnn)	R(nnn)	AND.L R(300),R(301)
AND.*	R(nnn)	@R(nnn)	AND.L R(300),@R(301)

* Formato Istruzione

HHB	HLB	LHB	LLB
HW		LW	
L			

Dimensione formato
Byte
Word
Long

Descrizione:

Questa istruzione esegue l' AND logico tra l'operando sorgente del formato prestabilito (BYTE, WORD, LONG) ed il registro destinazione. Mette il risultato all'interno del registro destinazione specificato. Nell' operando sorgente possono essere specificati i seguenti formati :

Decimale	Esadecimale	Binario
#VALORE	# 0xVALORE	# 0bVALORE

R(nnn) = numero che indica un registro interno, ed è compreso tra 0 e 900.

ESEMPI:

HHB HLB LHB LLB

Prima dell' l'esec. della istruz.	x	x	x	0F	Contenuto R(300)
AND.LLB #0xF0, R(300)					
Dopo l'esecuzione della istruz.	x	x	x	00	Contenuto R(300)

HW LW

Prima dell' l'esec. della istruz.	FFFF	X	Contenuto R(320)
AND.HW #0xFFF0, R(320)			
Dopo l'esecuzione della istruz.	FFF0	X	Contenuto R(320)

	R(310)	R(320)
Prima dell' l'esec. della istruz.	0FF0FF	FFFFFF
AND.L R(310), R(320)		
Dopo l'esecuzione della istruz.	0FF0FF	0FF0FF

	R(310)	R(320)
Prima dell' l'esec. della istruz.	00000F	FFFFFF
Contenuto di R(300)	320	
AND.L R(310), @R(300)		
Dopo l'esecuzione della istruz.	00000F	00000F

And logico a bit **ANDB**

Istruzione	Sorgente	Destinazione	Esempio
ANDB		R(nnn).nbit	ANDB R(430).4
ANDB	R(nnn)	R(nnn)	ANDB R(300),R(301)
ANDB		@R(nnn).nbit	ANDB @R(400).31
ANDB	@R(nnn)	R(nnn)	ANDB (@R(300),R(301)

Descrizione:

Questa istruzione esegue l' AND del bit indicato da nbit (numero che va da 0 a 31) all'interno di un registro indicato con la sintassi R(nnn) (valore compreso nei range da 0 a 900) e l'accumulatore per istruzioni a BIT che è identificato come il BIT 0 del registro 9. Il risultato dell'operazione viene messo nell'accumulatore.

ESEMPI:

	Bit da 31…..0 della locazione R(300)
ANDB R(300). 0	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.0
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 1
Dopo l'esec. Della istruz.	ACCUMULATORE R(009).0 = 0

	Bit da 31…..0 della locazione R(300)
ANDB R(300),R(301)	1.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x
Contenuto del registro R(301)	31
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 1
Dopo l'esec. Della istruz.	ACCUMULATORE R(009).0 = 1

	Bit da 31…..0 della locazione R(450)
ANDB @R(300).10	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.0.x.x.x.x.x.x.x.x.x.x.x
Contenuto del registro R(300)	450
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 1
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 0

Contenuto del registro R(300)	320
Contenuto del registro R(301)	31
	Bit da 31…..0 della locazione R(320)
Prima dell'esec. della istruz.	1.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 1
ANDB @R(300),R(301)	
Dopo l'esec. della istruz.	ACCUMULATORE R(009).0 = 1

Nand logico a bit **ANDB !**

Istruzione	Sorgente	Destinazione	Esempio
ANDB !		R(nnn).nbit	ANDB !(R(430).4)
ANDB !	R(nnn)	R(nnn)	ANDB !(R(300),R(301))
ANDB !		@R(nnn).nbit	ANDB !(@R(400).31)
ANDB !	@R(nnn)	R(nnn)	ANDB !(@R(300),R(301))

Descrizione:

Questa istruzione esegue l' AND tra il bit NEGATO indicato da nbit (numero che va da 0 a 31) all'interno di un registro indicato con la sintassi R(nnn) (valore compreso nei range da 0 a 900) e l'accumulatore per istruzioni a BIT che è identificato come il BIT 0 del registro 9. Il risultato dell'operazione viene messo nell'accumulatore.

ESEMPI:

	Bit da 31…..0 della locazione R(300)
ANDB !(R(300). 0)	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.0
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 1
Dopo l'esec. Della istruz.	ACCUMULATORE R(009).0 = 1

	Bit da 31…..0 della locazione R(300)
ANDB !(R(300),R(301))	1.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x
Contenuto del registro R(301)	31
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 1
Dopo l'esec. Della istruz.	ACCUMULATORE R(009).0 = 0

	Bit da 31…..0 della locazione R(450)
ANDB !(@R(300).10)	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.0.x.x.x.x.x.x.x.x.x.x
Contenuto del registro R(300)	450
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 1
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 1

Contenuto del registro R(300)	320
Contenuto del registro R(301)	31
	Bit da 31…..0 della locazione R(320)
Prima dell'esec. della istruz.	1.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 1
ANDB !(@R(300),R(301))	
Dopo l'esec. della istruz.	ACCUMULATORE R(009).0 = 0

Complemento a di un bit **CPLB**

Istruzione	Sorgente	Destinazione	Esempio
CPLB		R(nnn).nbit	CPLB R(430).04
CPLB	R(nnn)	R(nnn)	CPLB R(300),R(301)
CPLB		@R(nnn).nbit	CPLB @R(400).31
CPLB	@R(nnn)	R(nnn)	CPLB @R(300),R(301)

Descrizione:

Questa istruzione esegue il complemento del bit indicato da nbit (numero che va da 0 a 31) all'interno di un registro indicato con la sintassi R(nnn) (valore compreso nei range da 0 a 900).
L'ISTRUZIONE VIENE ESEGUITA SOLO SE IL CONTENUTO DELL'ACCUMULATORE A BIT (R (009).0 E' ALLO STATO LOGICO 1.

ESEMPI:

In tutti gli esempi sottoriportati si è considerato l'accumulatore nella condizione logica 1, se fosse stato a 0 l'istruzione non sarebbe stata eseguita.

	Bit da 31…..0 della locazione R(300)
CPLB R(300). 0	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.0
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 1
	Bit da 31…..0 della locazione R(300)
Dopo l'esec. Della istruz.	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.1
	Bit da 31…..0 della locazione R(300)
CPLB R(300),R(301)	1.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x
Contenuto del registro R(301)	31
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 1
Dopo l'esec. Della istruz.	0.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x
	Bit da 31…..0 della locazione R(450)
CPLB @R(300).10	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.0.x.x.x.x.x.x.x.x.x.x.x.x
Contenuto del registro R(300)	450
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 1
BIT ACC R(009).0	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.1.x.x.x.x.x.x.x.x.x.x.x
Contenuto del registro R(300)	320
Contenuto del registro R(301)	31
	Bit da 31…..0 della locazione R(320)
Prima dell'esec. della istruz.	1.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 1
CPLB @R(300),R(301)	
Dopo l'esec. della istruz.	0.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x

Decrementa valore **DEC**

Istruzione	Sorgente	Destinazione	Esempio
DEC1.*		R(300)	DEC1.L R(300)
DEC1.*		@R(300)	DEC1.L @R(300)
DEC2.*		R(300)	DEC2.L R(300)
DEC2.*		@R(300)	DEC2.L @R(300)
DEC4.*		R(300)	DEC4.L R(300)
DEC4.*		@R(300)	DEC4.L @R(300)

* Formato Istruzione	Dimensione formato
L	Long

Descrizione:

Questa istruzione sottrae il valore (1, 2 o 4) alla variabile definita in destinazione. Il risultato della operazione viene nuovamente memorizzato nel registro destinazione.

R(nnn) = numero che indica un registro interno, ed è compreso tra 0 e 900.

ESEMPI:

R(310)

DEC1.L R(310)	1000
Dopo l'esecuzione della istruz.	999

	R(300)	R(320)
DEC1.L @R(300)	320	2500
Dopo l'esecuzione della istruz.	320	2499

R(310)

DEC2.L R(310)	1000
Dopo l'esecuzione della istruz.	998

	R(300)	R(320)
DEC2.L @R(300)	320	2500
Dopo l'esecuzione della istruz.	320	2498

R(310)

DEC4.L R(310)	1000
Dopo l'esecuzione della istruz.	996

	R(300)	R(320)
DEC4.L @R(300)	320	2500
Dopo l'esecuzione della istruz.	320	2496

Divisione aritmetica **DIV**

Istruzione	Sorgente	Destinazione	Esempio
DIV.*	#Valore	R(nnn)	DIV.LW #1000,R(300)
DIV.*	R(nnn)	R(nnn)	DIV.L R(300),R(301)
DIV.*	R(nnn)	@R(nnn)	DIV.HW R(300),@R(301)

* Formato Istruzione

		Dimensione formato
HW	LW	Word
L		Long

Descrizione:

Questa istruzione esegue l'operazione aritmetica di divisione tra l'operando destinazione ed il sorgente del formato prestabilito (WORD, LONG). Il risultato viene memorizzato nel registro destinazione specificato, mentre il resto della divisione viene memorizzato nel Registro R(0).
Nell' operando sorgente possono essere specificati i seguenti formati :

Decimale	Esadecimale	Binario
#VALORE	# 0xVALORE	# 0bVALORE

R(nnn) = numero che indica un registro interno, ed è compreso tra 0 e 900.

ESEMPI:

HW LW

Prima dell' l'esec. della istruz.	4800	x	Contenuto R(320)
DIV.HW #01200, R(320)			
Dopo l'esecuzione della istruz.	4	x	Contenuto R(320)
Registro R(0)	0	0	Nessun resto

	R(310)	R(320)	
Prima dell' l'esec. della istruz.	16	7800	
DIV.L R(310), R(320)			
Dopo l'esecuzione della istruz.	HW	LW	
Contenuto di R(320)	0	487	
Registro R(0)	0	5	Resto

	R(310)	R(320)	
Prima dell' l'esec. della istruz.	1000	13400	
Contenuto di R(300)	320		
DIV.L R(310), @R(300)			
Dopo l'esecuzione della istruz.	0	13	
Registro R(0)	0	4	Resto

Decrementa e salta se non è zero **DJNZ**

Istruzione	Sorgente	Destinazione	Esempio
DJNZ	R(nnn)	LABEL	DJNZ R(300), SALTASU
DJNZ	@R(nnn)	LABEL	DJNZ @R(300), SALTASU
DJNZ	R(nnn)	R(nnn)	DJNZ R(300), R(400)
DJNZ	@R(nnn)	R(nnn)	DJNZ @R(300), R(400)

LABEL = Stringa che definisce l'indirizzo di salto

Descrizione:

Questa istruzione decrementa il contenuto del registro sorgente, e setta il contatore di programma all'indirizzo specificato dalla LABEL indicata in destinazione se il contenuto del registro sorgente è diverso da zero.
Nel caso la condizione non venga soddisfatta il program counter passerà all'istruzione successiva.

R(nnn) = numero che indica un registro interno, ed è compreso tra 0 e 900.

ESEMPI:

```
MOV.L     #10,R(300)      ;assegna alla variabile R(300) il valore 10
SALTASU:
NOP
DJNZ   R(300), SALTASU  ;R(300) = R(300) -1 = 9 e salta all'indirizzo SALTASU
                         ; sino a quando R(300) ≠ 0
```

```
MOV.L  #320,R(300)       ;la variabile R(300)  conterrà 320 (sarà poi la locaz. 320)
MOV.L  #10,R(320)        ; la variabile R(320) conterrà il valore 10
CONT:
NOP
DJNZ     @R(300), CONT   ;decrementa il registro puntato da R(300) che in questo caso
              ; è 320 e salta all'indirizzo CONT  finchè R(320) ≠ 0
```

```
MOV.L  #10,R(300)        ;la variabile R(300)  conterrà 10
MOV.L  #CONT,R(320)      ; la variabile R(320) conterrà il valore corrispondente
NOP                       ; all'indirizzo CONT
CONT:
NOP
DJNZ     R(300), R(320)  ;decrementa il registro  R(300) che in questo caso è 10
                          ;e salta all'indirizzo contenuto nella locazione R(320) che  è
                         ;CONT  sino a quando R(300) ≠ 0
```

Decrementa e salta se è zero **DJZ**

Istruzione	Sorgente	Destinazione	Esempio
DJZ	R(nnn)	LABEL	DJZ R(300), SALTASU
DJZ	@R(nnn)	LABEL	DJZ @R(300), SALTASU
DJZ	R(nnn)	R(nnn)	DJZ R(300), R(400)
DJZ	@R(nnn)	R(nnn)	DJZ @R(300), R(400)

LABEL = Stringa che definisce l'indirizzo di salto

Descrizione:

Questa istruzione decrementa il contenuto del registro sorgente, e setta il contatore di programma all'indirizzo specificato dalla LABEL indicata in destinazione se il contenuto del registro sorgente è uguale a zero.
Nel caso la condizione non venga soddisfatta il program counter passerà all'istruzione successiva.

R(nnn) = numero che indica un registro interno, ed è compreso tra 0 e 900.

ESEMPI:

```
;istruzione  DJZ  decrementa e salta se zero
mov.l                   #350,r(900)             ;R(900) = 350
mov.l                   #899,r(898)             ;R(898) = 899
mov.l                   #50,@r(898)             ;R(899) = 50
mov.l                   #NOAZZRAM,r(897)   ;R(897) = indirizzo NOAZZRAM
AZZRAM:
mov.l                   #0x5555aaaa,@r(900)     ;scrive il valore esad. 5555AAAA nella
nop                                             ;locazione puntata da R(900) e quindi R(350)
dec1.l          r(900)                          ;R(900)= 350-1
djz                     @r(898),r(897)          ;@R(898)= @R(898) –1 e cioè R(899)=50-1
nop                                             ;e salta all'indirizzo NOAZZRAM se R(899)=0
jmp                     AZZRAM

NOAZZRAM:
mov.l                   #400,r(800)             ;R(800) = 400
mov.l                   #50,r(798)              ;R(798) = 50
mov.l                   # NOAZZE1,r(797)        ;R(797) = indirizzo NOAZZE1
AZZRAM1:
mov.l                   #0x4444bbbb,@r(800)     ;Il valore puntato da R(800) e cioè R(400)
nop                                             ;conterrà il numero esad. 4444BBBB
dec1.l                  r(800)                  ;R(800) = R(800)-1 e cioè 400 = 400-1
djz                     r(798),r(797)           ;decrementa R(798) e quindi 50 = 50-1
nop                                             ;e salta all'indir. Contenuto in R(797)
nop                                             ;se R(798) = 0
jmp                     AZZRAM1
```

Fine programma **END**

Istruzione	Sorgente	Destinazione	Esempio
END			END

Descrizione:

Quando il program counter incontra questa istruzione il PLC esegue una serie di operazioni :

- Viene copiata la RAM il cui contenuto è lo stato delle uscite, sulle uscite fisiche dell'Hardware del PLC. Quindi compilando un file assembler che non ha al termine l'istruzione END non si ha nessuna condizione di errore, ma le uscite dei relè e delle espansioni parallele non verranno rinfrescate perché non viene effettuata nessuna copia di trasferimento dalla RAM contenente l'ultima condizione richiesta e l'uscita fisica vera e propria.
- Viene aggiornato il BIT 24 del registro R(74) che viene settato solo per la prima scansione di un programma
- Viene aggiornato il BIT 11 del registro R(74) che viene invertito ogni scansione di programma.
- Imposta il contatore di programma all'indirizzo dove si trova l'istruzione START per reinizializzare la scansione di programma.
- Esegue una verifica sul contatore di stack e da un errore se questo non è azzerato.

Salta se uguale **EQL**

Istruzione	Sorgente	Destinazione	Ind. di salto	Esempio
EQL.*	#Valore	R(nnn)	Label	EQL.LW #07FF,R(300),LABEL
EQL.*	R(nnn)	R(nnn)	Label	EQL.L R(300),R(301),LABEL
EQL.*	#Valore	@R(nnn)	Label	EQL.HW 1500 ,R(300),LABEL
EQL.*	R(nnn)	@R(nnn)	Label	EQL.L R(300),@R(301)

* Formato Istruzione

				Dimensione formato
HHB	HLB	LHB	LLB	Byte
HW		LW		Word
L				Long

Non ammessa quando la sorgente = #valore

Descrizione:

Questa istruzione compara la sorgente con la destinazione e salta all'indirizzo specificato se la sorgente è uguale alla destinazione.
Nel caso la condizione non venga soddisfatta il program counter passerà all'istruzione successiva.

Decimale	Esadecimale	Binario
#VALORE	# 0xVALORE	# 0bVALORE

R(nnn) = numero che indica un registro interno, ed è compreso tra 0 e 900.

ESEMPI:

```
INIZIO:
EQL.LLB   #0x24,R(300),EQU_1  ;se LLB del registro R(300)= 24
NOP                           ;salta all'indirizzo EQU_1
JMP       INIZIO              ;altrimenti salta all'ind. INIZIO
EQU_1:
EQL.HW    R(300),R(301),SECOMP ;se HW.R(300)=HW.R(301) vai a SECOMP
JMP       EQU_1
SECOMP:
MOV.L   #302,R(900)
EQL.L   #0xffff,@R(900),TERC ;se 0xFFFF= R(302) vai a TERC
JMP       SECOMP
TERC:
EQL.LLB   r(300),@r(900),QUARC ;se R(300)= R(302) vai a QUARC
JMP       TERC
QUARC:
```

Estende un valore **EXT**

Istruzione	Sorgente	Destinazione	Esempio
EXT.*		R(nnn)	EXT.W R(300)
EXT.*		@R(nnn)	EXT.L @R(301)

* Formato Istruzione	Dimensione formato
W	Word
L	Long

Descrizione:

Questa istruzione estende il bit di segno (bit 7) nella istruzione di lunghezza WORD e (bit 15) in quella di lunghezza LONG ai bit piu significativi dopo il bit di segno, (bit da 8 a 15) per l'istruzione con estensione WORD e (bit da 16 a 32) per quella con estensione LONG.

R(nnn) = numero che indica un registro interno, ed è compreso tra 0 e 900.

ESEMPI:

	HHB	HLB	LHB	LLB	
Prima dell' l'esec. della istruz.	x	x	x	01010000	Contenuto R(320)
EXT.W R(320)					
Dopo l'esecuzione della istruz.	x		0000000001010000		Contenuto R(320)

	HW	R(310)	LW
Prima dell' l'esec. della istruz.	x		1011000011001100
EXT.L R(310)			
Dopo l'esecuzione della istruz.	HW		LW
Contenuto di R(320)	1111111111111111		1011000011001100

Contenuto di R(300)	320	
Prima dell' l'esec. della istruz.	x	0011000011001100
EXT.L @R(300)	HW R(320)	LW
Dopo l'esecuzione della istruz.	0000000000000000	0011000011001100

Salta se maggiore o uguale **GEQ**

Istruzione	Sorgente	Destinazione	Ind. di salto	Esempio
GEQ.*	#Valore	R(nnn)	Label	GEQ.LW #07FF,R(300),LABEL
GEQ.*	R(nnn)	R(nnn)	Label	GEQ.L R(300),R(301),LABEL
GEQ.*	#Valore	@R(nnn)	Label	GEQ.HW 1500 ,R(300),LABEL
GEQ.*	R(nnn)	@R(nnn)	Label	GEQ.L R(300),@R(301)

* Formato Istruzione Dimensione formato

HHB	HLB	LHB	LLB	Byte
HW		LW		Word
L				Long

Non ammessa quando la sorgente = #valore

Descrizione:

Questa istruzione compara la sorgente con la destinazione e salta all'indirizzo specificato se la sorgente è maggiore o uguale alla destinazione.
Nella comparazione si terrà conto del segno, per cui se il bit di segno è = 1 il valore è da considerare negativo e quindi minore di qualsiasi numero avente bit di segno = 0.
Nel caso la condizione non venga soddisfatta il program counter passerà all'istruzione successiva.

Decimale	Esadecimale	Binario
#VALORE	# 0xVALORE	# 0bVALORE

R(nnn) = numero che indica un registro interno, ed è compreso tra 0 e 900.

ESEMPI:

```
INIZIO:
GEQ.LLB   #0x24,R(300),EQU_1  ; 24 >= di LLB del registro R(300)
NOP                           ;salta all'indirizzo EQU_1
JMP       INIZIO              ;altrimenti salta all'ind. INIZIO
EQU_1:
GEQ.HW    R(300),R(301),SECOMP ;se HW.R(300)>=HW.R(301) vai a SECOMP
JMP       EQU_1                ;altrimenti salta all'ind. EQU_1
SECOMP:
MOV.L   #302,R(900)
GEQ.LW  #0xffff,@R(900),TERC ;se 0xFFFF >= R(302) vai a TERC
NOP             ;in questo caso tenere che 0xFFFF è un
NOP             ;numero negativo.
JMP       SECOMP
TERC:
GEQ.LLB   R(300),@R(900),QUARC ;se R(300)>= R(302) vai a QUARC
JMP       TERC
QUARC:
```

Salta se maggiore G**TR**

Istruzione	Sorgente	Destinazione	Ind. di salto	Esempio
GTR.*	#Valore	R(nnn)	Label	GTR.LW #07FF,R(300),LABEL
GTR.*	R(nnn)	R(nnn)	Label	GTR.L R(300),R(301),LABEL
GTR.*	#Valore	@R(nnn)	Label	GTR.HW 1500 ,R(300),LABEL
GTR.*	R(nnn)	@R(nnn)	Label	GTR.L R(300),@R(301)

* Formato Istruzione

				Dimensione formato
HHB	HLB	LHB	LLB	Byte
HW		LW		Word
L				Long

Non ammessa quando la sorgente = #valore

Descrizione:

Questa istruzione compara la sorgente con la destinazione e salta all'indirizzo specificato se la sorgente è maggiore della destinazione.
Nella comparazione si terrà conto del segno, per cui se il bit di segno è = 1 il valore è da considerare negativo e quindi minore di qualsiasi numero avente bit di segno = 0.
Nel caso la condizione non venga soddisfatta il program counter passerà all'istruzione successiva

Decimale	Esadecimale	Binario
#VALORE	# 0xVALORE	# 0bVALORE

R(nnn) = numero che indica un registro interno, ed è compreso tra 0 e 900.

ESEMPI:

```
INIZIO:
GTR.LLB   #0x24,R(300),EQU_1  ;se LLB del registro 24 > R(300)
NOP                           ;salta all'indirizzo EQU_1
JMP       INIZIO              ;altrimenti salta all'ind. INIZIO
EQU_1:
GTR.HW    R(300),R(301),SECOMP ;se HW.R(300)> HW.R(301) vai a SECOMP
JMP       EQU_1                ;altrimenti salta all'ind. EQU_1
SECOMP:
MOV.L   #302,R(900)
GTR.LW  #0xffff,@R(900),TERC ;se 0xFFFF > R(302) vai a TERC
NOP             ;in questo caso tenere che 0xFFFF è un
NOP             ;numero negativo.
JMP       SECOMP
TERC:
GTR.LLB   R(300),@R(900),QUARC ;se R(300)> R(302) vai a QUARC
JMP       TERC
QUARC:
```

Incrementa valore **INC**

Istruzione	Sorgente	Destinazione	Esempio
INC1.*		R(300)	INC1.L R(300)
INC1.*		@R(300)	INC1.L @R(300)
INC2.*		R(300)	INC2.L R(300)
INC2.*		@R(300)	INC2.L @R(300)
INC4.*		R(300)	INC4.L R(300)
INC4.*		@R(300)	INC4.L @R(300)

* Formato Istruzione	Dimensione formato
L	Long

Descrizione:

Questa istruzione somma il valore (1, 2 o 4) alla variabile definita in destinazione. Il nuovo valore viene nuovamente memorizzato nel registro destinazione.

R(nnn) = numero che indica un registro interno, ed è compreso tra 0 e 900.

ESEMPI:

	R(310)
INC1.L R(310)	1000
Dopo l'esecuzione della istruz.	1001

	R(300)	R(320)
INC1.L @R(300)	320	2500
Dopo l'esecuzione della istruz.	320	2501

	R(310)
INC2.L R(310)	1000
Dopo l'esecuzione della istruz.	1002

	R(300)	R(320)
INC2.L @R(300)	320	2500
Dopo l'esecuzione della istruz.	320	2502

	R(310)
INC4.L R(310)	1000
Dopo l'esecuzione della istruz.	1004

	R(300)	R(320)
INC4.L @R(300)	320	2500
Dopo l'esecuzione della istruz.	320	2504

Compara un bit e salta se non è zero **JBNZ**

Istruzione	Sorgente	Destinazione	Esempio
JBNZ		LABEL	JBNZ SALTASU
JBNZ		R(nnn)	JBNZ R(300)

LABEL = Stringa che definisce l'indirizzo di salto

Descrizione:

Questa istruzione salta all'indirizzo specificato da LABEL o dal registro r(nnn) se l'accumulatore a bit (bit 0 del registro R(009)) è diverso da 0. Nel caso la condizione non venga soddisfatta il contatore di programma passerà all'istruzione successiva.

R(nnn) = numero che indica un registro interno, ed è compreso tra 0 e 900.

ESEMPI:

```
ABSSETB  R(009).0                ; forzo a 1 l'accumulatore
NOP
JBNZ   ESCI                      ; salta all'indirizzo ESCI perché R(009) non è 0
          NOP
ESCI:
          MOV.L      #FUORI,R(300)      ;assegno ad R(300) l'indirizzo di salto "FUORI"
          ABSSETB R(009).0              ; forzo a 1 l'accumulatore
          JBNZ  R(300)                  ;salta all'indirizzo FUORI perché R(009) non è 0
          NOP
FUORI:
```

Compara un bit e salta se è zero **JBZ**

Istruzione	Sorgente	Destinazione	Esempio
JBZ		LABEL	JBZ SALTASU
JBZ		R(nnn)	JBZ R(300)

LABEL = Stringa che definisce l'indirizzo di salto

Descrizione:

Questa istruzione salta all'indirizzo specificato da LABEL o dal registro r(nnn) se l'accumulatore a bit (bit 0 del registro R(009)) è uguale 0. Nel caso la condizione non venga soddisfatta il contatore di programma passerà all'istruzione successiva.

R(nnn) = numero che indica un registro interno, ed è compreso tra 0 e 900.

ESEMPI:

```
ABSRESB  R(009).0                ; forzo a 0 l'accumulatore
NOP
JBZ   ESCI                       ; salta all'indirizzo ESCI perché R(009) è 0
          NOP
ESCI:
NOP
          MOV.L      #FUORI,R(300)        ;assegno ad R(300) l'indirizzo di salto "FUORI"
          ABSRESB R(009).0                ; forzo a 0 l'accumulatore
          JBZ  R(300)                     ;salta all'indirizzo FUORI perché R(009) è 0
          NOP
FUORI:
NOP
```

Salto incondizionato **JMP**

Istruzione	Sorgente	Destinazione	Esempio
JMP		LABEL	JMP LABEL
JMP		R(nnn)	JMP R(300)
JMP		@R(nnn)	JMP @R(400)

LABEL = Stringa che definisce l'indirizzo di salto

Descrizione:

Questa istruzione esegue un salto incondizionato all'indirizzo specificato da LABEL o dal contenuto di un registro.

R(nnn) = numero che indica un registro interno, ed è compreso tra 0 e 900.

ESEMPI:

```
QUI:
        NOP
        NOP
        JMP   QUI
        MOV.L    #OLTRE,R(419)
        NOP
        JMP   R(419)                ;Program counter salta all'indirizzo OLTRE
        NOP                         ;scritto all'interno di R(419)
        NOP
OLTRE:
        MOV.L  #ALT_IND,R(300)
        MOV.L  #300,R(310)
        JMP   @R(310)               ;P.C. salta all'indirizzo ALT_IND scritto
        NOP                         ;scritto all'interno di R(300) e puntato
        NOP                         ;dal registro R(310)
ALT_IND:
```

Scrivi un bit nell'accumulatore **LDB**

Istruzione	Sorgente	Destinazione	Esempio
LDB		R(nnn).nbit	LDB R(430).4
LDB	R(nnn)	R(nnn)	LDB R(300),R(301)
LDB		@R(nnn).nbit	LDB @R(400).31
LDB	@R(nnn)	R(nnn)	LDB @R(300),R(301)

Descrizione:

Questa istruzione carica l'accumulatore per istruzioni a BIT (identificato come il BIT 0 del registro 9) con il bit contenuto da nbit (numero che va da 0 a 31) di un registro indicato con la sintassi R(nnn) (valore compreso nei range da 0 a 900).

ESEMPI:

Bit da 31…..0 della locazione R(300)

LDB R(300). 0	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.0
Dopo l'esec. Della istruz.	ACCUMULATORE R(009).0 = 0

Bit da 31…..0 della locazione R(300)

LDB R(300),R(301)	1.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x
Contenuto del registro R(301)	31
Dopo l'esec. Della istruz.	ACCUMULATORE R(009).0 = 1

Bit da 31…..0 della locazione R(450)

LDB @R(300).10	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.0.x.x.x.x.x.x.x.x.x.x.x
Contenuto del registro R(300)	450
Dopo l'esec. Della istruz.	ACCUMULATORE R(009).0 = 0

Contenuto del registro R(300)	320
Contenuto del registro R(301)	31

Bit da 31…..0 della locazione R(320)

LDB @R(300),R(301)	1.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x
Dopo l'esec. della istruz.	ACCUMULATORE R(009).0 = 1

Scrivi un bit negato nell'accumulatore **LDB !**

Istruzione	Sorgente	Destinazione	Esempio
LDB!		R(nnn).nbit	LDB !(R(430).4)
LDB!	R(nnn)	R(nnn)	LDB !(R(300),R(301))
LDB!		@R(nnn).nbit	LDB !(@R(400).31)
LDB!	@R(nnn)	R(nnn)	LDB !(@R(300),R(301))

Descrizione:

Questa istruzione carica l'accumulatore per istruzioni a BIT (identificato come il BIT 0 del registro 9) con il complemento del bit contenuto da nbit (numero che va da 0 a 31) di un registro indicato con la sintassi R(nnn) (valore compreso nei range da 0 a 900).

ESEMPI:

Bit da 31…..0 della locazione R(300)

LDB ! (R(300). 0)	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.0
Dopo l'esec. Della istruz.	ACCUMULATORE R(009).0 = 1

Bit da 31…..0 della locazione R(300)

LDB ! (R(300),R(301))	1.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x
Contenuto del registro R(301)	31
Dopo l'esec. Della istruz.	ACCUMULATORE R(009).0 = 0

Bit da 31…..0 della locazione R(450)

LDB !(@R(300).10)	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.0.x.x.x.x.x.x.x.x.x.x
Contenuto del registro R(300)	450
Dopo l'esec. Della istruz.	ACCUMULATORE R(009).0 = 1

Contenuto del registro R(300)	320
Contenuto del registro R(301)	31

Bit da 31…..0 della locazione R(320)

LDB !(@R(300),R(301))	1.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x
Dopo l'esec. della istruz.	ACCUMULATORE R(009).0 = 0

Salta se è minore o uguale **LEQ**

Istruzione	Sorgente	Destinazione	Ind. di salto	Esempio
LEQ.*	#Valore	R(nnn)	Label	LEQ.LW #07FF,R(300),LABEL
LEQ.*	R(nnn)	R(nnn)	Label	LEQ.L R(300),R(301),LABEL
LEQ.*	#Valore	@R(nnn)	Label	LEQ.HW 1500 ,R(300),LABEL
LEQ.*	R(nnn)	@R(nnn)	Label	LEQ.L R(300),@R(301)

* Formato Istruzione Dimensione formato

HHB	HLB	LHB	LLB	Byte
HW		LW		Word
L				Long

Non ammessa quando la sorgente = #valore

Descrizione:

Questa istruzione compara la sorgente con la destinazione e salta all'indirizzo specificato se la sorgente è minore o uguale alla destinazione.
Nella comparazione si terrà conto del segno, per cui se il bit di segno è = 1 il valore è da considerare negativo e quindi minore di qualsiasi numero avente bit di segno = 0.
Nel caso la condizione non venga soddisfatta il program counter passerà all'istruzione successiva.

Decimale	Esadecimale	Binario
#VALORE	# 0xVALORE	# 0bVALORE

R(nnn) = numero che indica un registro interno, ed è compreso tra 0 e 900.

ESEMPI:

```
INIZIO:
LEQ.LLB   #0x24,R(300),EQU_1  ;24 <= di LLB del registro R(300)
NOP                           ;salta all'indirizzo EQU_1
JMP       INIZIO              ;altrimenti salta all'ind. INIZIO
EQU_1:
LEQ.HW    R(300),R(301),SECOMP ;se HW.R(300)<=HW.R(301) vai a SECOMP
JMP       EQU_1                ;altrimenti salta all'ind. EQU_1
SECOMP:
MOV.L  #302,R(900)
LEQ.LW  #0xffff,@R(900),TERC ;se 0xFFFF <= R(302) vai a TERC
NOP           ;in questo caso consid.che 0xFFFF è un
NOP           ;numero negativo.
JMP       SECOMP
TERC:
LEQ.LLB   R(300),@R(900),QUARC ;se R(300)<= R(302) vai a QUARC
JMP       TERC
QUARC:
```

Salta se è minore **LES**

Istruzione	Sorgente	Destinazione	Ind. di salto	Esempio
LES.*	#Valore	R(nnn)	Label	LES.LW #07FF,R(300),LABEL
LES.*	R(nnn)	R(nnn)	Label	LES.L R(300),R(301),LABEL
LES.*	#Valore	@R(nnn)	Label	LES.HW 1500 ,R(300),LABEL
LES.*	R(nnn)	@R(nnn)	Label	LES.L R(300),@R(301)

* Formato Istruzione Dimensione formato

HHB	HLB	LHB	LLB	Byte
HW		LW		Word
L				Long

Non ammessa quando la sorgente = #valore

Descrizione:

Questa istruzione compara la sorgente con la destinazione e salta all'indirizzo specificato se la sorgente è minore della destinazione.
Nella comparazione si terrà conto del segno, per cui se il bit di segno è = 1 il valore è da considerare negativo e quindi minore di qualsiasi numero avente bit di segno = 0.
Nel caso la condizione non venga soddisfatta il program counter passerà all'istruzione successiva

Decimale	Esadecimale	Binario
#VALORE	# 0xVALORE	# 0bVALORE

R(nnn) = numero che indica un registro interno, ed è compreso tra 0 e 900.

ESEMPI:

```
INIZIO:
LES.LLB    #0x24,R(300),EQU_1  ;24 < di LLB del registro R(300)
NOP                            ;salta all'indirizzo EQU_1
JMP        INIZIO              ;altrimenti salta all'ind. INIZIO
EQU_1:
LES.HW     R(300),R(301),SECOMP ;se HW.R(300)< HW.R(301) vai a SECOMP
JMP        EQU_1                ;altrimenti salta all'ind. EQU_1
SECOMP:
MOV.L   #302,R(900)
LES.LW  #0xffff,@R(900),TERC ;se 0xFFFF < R(302) vai a TERC
NOP             ;in questo caso tenere che 0xFFFF è un
NOP             ;numero negativo.
JMP        SECOMP
TERC:
LES.LLB    R(300),@R(900),QUARC ;se R(300)< R(302) vai a QUARC
JMP        TERC
QUARC:
```

Trasferisce valori tra registri **MOV**

Istruzione	Sorgente	Destinazione	Esempio
MOV.*	#Valore	R(nnn)	MOV.L #1000,R(300)
MOV.*	#Valore	@R(nnn)	MOV.L #1000,@R(300)
MOV.*	R(nnn)	R(nnn)	MOV.L R(300),R(301)
MOV.*	R(nnn)	@R(nnn)	MOV.L R(300),@R(301)
MOV.*	@R(nnn)	R(nnn)	MOV.L @R(300),R(301)

* Formato Istruzione

HHB	HLB	LHB	LLB
HW		LW	
L			

Dimensione formato

Byte
Word
Long

Descrizione:

Questa istruzione trasferisce l'operando sorgente del formato prestabilito (BYTE, WORD, LONG) all'interno del registro destinazione specificato.
Nell' operando sorgente possono essere specificati i seguenti formati :

Decimale	Esadecimale	Binario	Floating point
#VALORE	# 0xVALORE	# 0bVALORE	# 0f VALORE

R(nnn) = numero che indica un registro interno, ed è compreso tra 0 e 900.

ESEMPI:

MOV.LLB #100, R(300)
MOV.HLB #100, R(300)

HHB	HLB	LHB	LLB	
x	x	x	100	Contenuto R(300)
x	100	x	x	Contenuto R(300)

MOV.LW #0xFFFF, R(320)

HW	LW	
x	FFFF	Contenuto R(320)

MOV.HW #41500, @R(450)
Dopo l'esecuzione della istruz.

300		Contenuto R(450)
41500	x	Contenuto R(300)

MOV.HHB R(301),R(300)
Dopo l'esecuzione della istruz

R(301)				R(300)			
AA	55	FF	22	43	x	x	x
AA	55	FF	22	AA	x	x	x

MOV.L R(310),@R(320)
Dopo l'esecuzione della istruz.

R(310)	R(320)	R(315)
0x1200FFFF	315	0xFFFFFFFF
0x1200FFFF	315	0x1200FFFF

MOV.L @R(400),R(870)
Dopo l'esecuzione della istruz.

R(400)	R(401)	R(870)
401	0x22334455	0xFFFFFFFF
401	0x22334455	0x22334455

Operazione aritmetica di moltiplicazione **MULT**

Istruzione	Sorgente	Destinazione	Esempio
MULT.*	#Valore	R(nnn)	MULT.L #1000,R(300)
MULT.*	R(nnn)	R(nnn)	MULT.L R(300),R(301)
MULT.*	R(nnn)	@R(nnn)	MULT.L R(300),@R(301)

* Formato Istruzione

				Dimensione formato
HHB	HLB	LHB	LLB	Byte
HW		LW		Word
L				Long

Descrizione:

Questa istruzione esegue il prodotto tra l'operando sorgente del formato prestabilito (BYTE, WORD, LONG) ed il registro destinazione. Mette il risultato all'interno del registro destinazione specificato. Nell' operando sorgente possono essere specificati i seguenti formati :

Decimale	Esadecimale	Binario
#VALORE	# 0xVALORE	# 0bVALORE

R(nnn) = numero che indica un registro interno, ed è compreso tra 0 e 900.

ESEMPI:

Prima dell' l'esec. della istruz.
MULT.LLB #10, R(300)
Dopo l'esecuzione della istruz.

HHB	HLB	LHB	LLB	
x	x	x	10	Contenuto R(300)
x	x	x	100	Contenuto R(300)

Prima dell' l'esec. della istruz.
MULT.HW #2, R(320)
Dopo l'esecuzione della istruz.

HW	LW	
1000	x	Contenuto R(320)
2000	x	Contenuto R(320)

Prima dell' l'esec. della istruz.
MULT.L R(310), R(320)
Dopo l'esecuzione della istruz.

R(310)	R(320)
18000	32
18000	576000

Prima dell' l'esec. della istruz.
Contenuto di R(300)
MULT.L R(310), @R(300)
Dopo l'esecuzione della istruz.

R(310)	R(320)
135000	16
320	
135000	2160000

Complemento a due **NEG**

Istruzione	Sorgente	Destinazione	Esempio
NEG.*		R(nnn)	NEG.L R(300)
NEG.*		@R(nnn)	NEG.L @R(301)

* Formato Istruzione

HHB	HLB	LHB	LLB
HW		LW	
L			

Dimensione formato
Byte
Word
Long

Descrizione:

Questa istruzione esegue il complemento a due del registro destinazione (Rd = 0 - Rd). Memorizza il risultato all'interno del registro destinazione specificato.

R(nnn) = numero che indica un registro interno, ed è compreso tra 0 e 900.

ESEMPI:

	HHB	HLB	LHB	LLB	
Prima dell' l'esec. della istruz.	x	x	x	7Fh	Contenuto R(300)
NEG.LLB R(300)					
Dopo l'esecuzione della istruz.	x	x	x	81h	Contenuto R(300)

	HW	LW	
Prima dell' l'esec. della istruz.	3FFFh	x	Contenuto R(320)
NEG.HW R(320)			
Dopo l'esecuzione della istruz.	C001h	x	Contenuto R(320)

	R(310)	R(320)
Prima dell' l'esec. Della istruz.	320	80000h
NEG.L @ R(310)		
Dopo l'esecuzione della istruz.	320	FFF80000h

Salta se diverso **NEQ**

Istruzione	Sorgente	Destinazione	Ind. di salto	Esempio
NEQ.*	#Valore	R(nnn)	Label	NEQ.LW #07FF,R(300),LABEL
NEQ.*	R(nnn)	R(nnn)	Label	NEQ.L R(300),R(301),LABEL
NEQ.*	#Valore	@R(nnn)	Label	NEQ.HW 1500 ,R(300),LABEL
NEQ.*	R(nnn)	@R(nnn)	Label	NEQ.L R(300),@R(301)

* Formato Istruzione

HHB	HLB	LHB	LLB
HW		LW	
L			

Dimensione formato
Byte
Word
Long

Descrizione:

Questa istruzione compara la sorgente con la destinazione e salta all'indirizzo specificato se la sorgente è diversa dalla destinazione.
Nel caso la condizione non venga soddisfatta il program counter passerà all'istruzione successiva.

Decimale	Esadecimale	Binario
#VALORE	# 0xVALORE	# 0bVALORE

R(nnn) = numero che indica un registro interno, ed è compreso tra 0 e 900.

ESEMPI:

```
INIZIO:
NEQ.LLB   #0x24,R(300),EQU_1  ;se LLB del registro R(300) ≠ 24
NOP                           ;salta all'indirizzo EQU_1
JMP       INIZIO              ;altrimenti salta all'ind. INIZIO
EQU_1:
NEQ.HW    R(300),R(301),SECOMP ;se HW.R(300) ≠ HW.R(301) vai a SECOMP
JMP       EQU_1
SECOMP:
MOV.L   #302,R(900)
NEQ.LW  #0xffff,@R(900),TERC ;se 0xFFFF ≠ R(302) vai a TERC
JMP       SECOMP
TERC:
NEQ.LLB   r(300),@r(900),QUARC ;se R(300) ≠ R(302) vai a QUARC
JMP       TERC
QUARC:
```

Nessuna operazione **NOP**

Istruzione	Sorgente	Destinazione	Esempio
NOP			NOP

Descrizione:

Questa istruzione incrementa il program counter causando l'esecuzione della istruzione successiva. Lo stato di tutti i registri non cambia.

ESEMPIO:

```
MOV.L #10,R(800)    ; memorizza il valore 10 all'interno del registro R(800)
NOP                 ; incrementa solamente il program counter
NOP                 ; incrementa solamente il program counter
NOP                 ; incrementa solamente il program counter
MOV.L #20,R(801)    ; memorizza il valore 20 all'interno del registro R(801)
```

Complemento ad uno **NOT**

Istruzione	Sorgente	Destinazione	Esempio
NOT.*		R(nnn)	NOT.L R(300)
NOT.*		@R(nnn)	NOT.L @R(301)

* Formato Istruzione

HHB	HLB	LHB	LLB	Byte
HW		LW		Word
L				Long

Dimensione formato

Descrizione:

Questa istruzione esegue il complemento del registro destinazione (Rd = - Rd). Memorizza il risultato all'interno del registro destinazione specificato.

R(nnn) = numero che indica un registro interno, ed è compreso tra 0 e 900.

ESEMPI:

	HHB	HLB	LHB	LLB	
Prima dell' l'esec. della istruz.	x	x	x	00h	Contenuto R(300)
NOT.LLB R(300)					
Dopo l'esecuzione della istruz.	x	x	x	FFh	Contenuto R(300)

	HW	LW	
Prima dell' l'esec. Della istruz.	AA55h	x	Contenuto R(320)
NOT.HW R(320)			
Dopo l'esecuzione della istruz.	55AAh	x	Contenuto R(320)

	R(310)	R(320)
Prima dell' l'esec. Della istruz.	320	000FFFFFh
NOT.L @ R(310)		
Dopo l'esecuzione della istruz.	320	FFF00000h

NOT di un bit **NOTB**

Istruzione	Sorgente	Destinazione	Esempio
NOTB			NOTB

Descrizione:

Questa istruzione esegue il NOT del BIT 0 registro 009 (accumulatore logico a BIT)

ESEMPIO:

```
ABSRESB     ; memorizza il valore 0 all'interno del registro R(009).0
NOP         ; incrementa solamente il program counter
NOTB        ; nega il contenuto di R(009).0
NOP         ;dopo l'istruzione NOTB l'accumulatore conterra 1
NOP         ;invece di 0.
```

Or logico **OR**

Istruzione	Sorgente	Destinazione	Esempio
OR.*	#Valore	R(nnn)	OR.LW #0FF00,R(300)
OR.*	R(nnn)	R(nnn)	OR.L R(300),R(301)
OR.*	R(nnn)	@R(nnn)	OR.L R(300),@R(301)

* Formato Istruzione

HHB	HLB	LHB	LLB
HW		LW	
L			

Dimensione formato
Byte
Word
Long

Descrizione:

Questa istruzione esegue l' OR logico tra l'operando sorgente del formato prestabilito (BYTE, WORD, LONG) ed il registro destinazione. Mette il risultato all'interno del registro destinazione specificato. Nell' operando sorgente possono essere specificati i seguenti formati :

Decimale	Esadecimale	Binario
#VALORE	# 0xVALORE	# 0bVALORE

R(nnn) = numero che indica un registro interno, ed è compreso tra 0 e 900.

ESEMPI:

HHB HLB LHB LLB

Prima dell' l'esec. della istruz.	x	x	x	0F	Contenuto R(300)
OR.LLB #0xF0, R(300)					
Dopo l'esecuzione della istruz.	x	x	x	FF	Contenuto R(300)

HW LW

Prima dell' l'esec. della istruz.	0F0F	x	Contenuto R(320)
OR.HW #0xFFF0, R(320)			
Dopo l'esecuzione della istruz.	FFFF	x	Contenuto R(320)

R(310) R(320)

Prima dell' l'esec. della istruz.	0000FF	FF00FF
OR.L R(310), R(320)		
Dopo l'esecuzione della istruz.	0000FF	FF00FF

R(310) R(320)

Prima dell' l'esec. della istruz.	00000F	000000
Contenuto di R(300)	320	
OR.L R(310), @R(300)		
Dopo l'esecuzione della istruz.	00000F	00000F

Or logico di un bit **ORB**

Istruzione	Sorgente	Destinazione	Esempio
ORB		R(nnn).nbit	ORB R(430).4
ORB	R(nnn)	R(nnn)	ORB R(300),R(301)
ORB		@R(nnn).nbit	ORB @R(400).31
ORB	@R(nnn)	R(nnn)	ORB (@R(300),R(301)

Descrizione:

Questa istruzione esegue l' OR del bit indicato da nbit (numero che va da 0 a 31) all'interno di un registro indicato con la sintassi R(nnn) (valore compreso nei range da 0 a 900) e l'accumulatore per istruzioni a BIT che è identificato come il BIT 0 del registro 9. Il risultato dell'operazione viene memorizzato nell'accumulatore.

ESEMPI:

	Bit da 31…..0 della locazione R(300)
ORB R(300). 0	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.1
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 0
Dopo l'esec. Della istruz.	ACCUMULATORE R(009).0 = 1

	Bit da 31…..0 della locazione R(300)
ORB R(300),R(301)	0.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x
Contenuto del registro R(301)	31
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 1
Dopo l'esec. Della istruz.	ACCUMULATORE R(009).0 = 1

	Bit da 31…..0 della locazione R(450)
ORB @R(300).10	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.0.x.x.x.x.x.x.x.x.x
Contenuto del registro R(300)	450
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 1
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 0

Contenuto del registro R(300)	320
Contenuto del registro R(301)	31
	Bit da 31…..0 della locazione R(320)
Prima dell'esec. della istruz.	1.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 1
ORB @R(300),R(301)	
Dopo l'esec. della istruz.	ACCUMULATORE R(009).0 = 1

Nor logico di un bit **ORB !**

Istruzione	Sorgente	Destinazione	Esempio
ORB!		R(nnn).nbit	ORB ! (R(430).4)
ORB!	R(nnn)	R(nnn)	ORB !(R(300),R(301))
ORB!		@R(nnn).nbit	ORB ! (@R(400).31)
ORB!	@R(nnn)	R(nnn)	ORB !(@R(300),R(301))

Descrizione:

Questa istruzione esegue l' OR del bit NEGATO indicato da nbit (numero che va da 0 a 31) all'interno di un registro indicato con la sintassi R(nnn) (valore compreso nei range da 0 a 900) e l'accumulatore per istruzioni a BIT che è identificato come il BIT 0 del registro 9. Il risultato dell'operazione viene memorizzato nell'accumulatore.

ESEMPI:

	Bit da 31…..0 della locazione R(300)
ORB !(R(300). 0)	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.1
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 0
Dopo l'esec. Della istruz.	ACCUMULATORE R(009).0 = 0

	Bit da 31…..0 della locazione R(300)
ORB ! (R(300),R(301))	0.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x
Contenuto del registro R(301)	31
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 0
Dopo l'esec. Della istruz.	ACCUMULATORE R(009).0 = 1

	Bit da 31…..0 della locazione R(450)
ORB ! (@R(300).10)	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.0.x.x.x.x.x.x.x.x.x.x.x
Contenuto del registro R(300)	450
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 0
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 1

Contenuto del registro R(300)	320
Contenuto del registro R(301)	31
	Bit da 31…..0 della locazione R(320)
Prima dell'esec. della istruz.	1.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 0
ORB ! (@R(300),R(301))	
Dopo l'esec. della istruz.	ACCUMULATORE R(009).0 = 0

Or logico di un bit con il bit di stack **ORBPOP**

Istruzione	Sorgente	Destinazione	Esempio
ORBPOP			ORBPOP

Descrizione:

Questa istruzione esegue l' OR logico a bit tra l'accumulatore ed il contenuto del BIT localizzato all'interno dello stack ed inserito per mezzo dell'istruzione PUSHB.

ESEMPI:

ORBPOP	CONTENUTO BITSTACK = 1
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 0
Dopo l'esec. Della istruz.	ACCUMULATORE R(009).0 = 1

Preleva un valore dallo stack **POP**

Istruzione	Sorgente	Destinazione	Esempio
POP.*		R(nnn)	POP.LW R(300)
POP.*		@R(nnn)	POP.HW @R(301)

* Formato Istruzione

		Dimensione formato
HW	LW	Word
L		Long

Descrizione:

L'istruzione POP preleva dallo stack il primo valore (della lunghezza definita WORD o LONG) posizionato nella cima dello stack stesso dall'istruzione PUSH e lo memorizza sul registro specificato in destinazione.
Il contatore di STACK viene decrementato del numero di BYTE corrispondenti alla lunghezza del formato della istruzione POP:

- 2 BYTE per la lunghezza WORD
- 4 BYTE per la lunghezza LONG.

E' importante ricordare che il contatore di STACK *deve essere zero quando il program counter incontra l'istruzione* END *altrimenti verrà generato un errore di* STACK *e verrà settato un bit di sistema identificato dal* BIT 18 *del registro* R4.
Affinchè questo errore non si presenti è sufficiente far coincidere il numero e la lunghezza delle istruzioni PUSH *e* POP.

R(nnn) = numero che indica un registro interno, ed è compreso tra 0 e 900.

ESEMPI:

	HW	LW	
Prima dell' l'esec. della istruz.	4800	x	Contenuto R(320)
POP.LW R(320)			
Cont. WORD STACK	5200		
Dopo l'esecuzione della istruz.	4800	5200	Contenuto R(320)

	R(350)	R(320)
Prima dell' l'esec. della istruz.	320	7800
Cont. LONG STACK	137000	
POP.L @R(350)		
Dopo l'esecuzione della istruz.	HW	LW
Contenuto di R(320)	137000	

Preleva un bit dallo stack **POPB**

Istruzione	Sorgente	Destinazione	Esempio
POPB			POPB

Descrizione:

Questa istruzione estrae dalla cima dello STACK il contenuto a 32bit posizionato sulla cima dello stack stesso e lo memorizza nell'accumulatore a BIT R(009). Di tutti e 32 i bit si terrà conto solamente del primo, che è il meno significativo (BIT 0), ed il quale è l'unico ad essere utilizzato nelle istruzioni a BIT. Questa istruzione può essere considerata identica alla istruzione POP.L R(nnn) , con la sola differenza che la destinazione è fissa ed è accumulatore R(009).

E' importante ricordare che il contatore di STACK *deve essere zero quando il program counter incontra l'istruzione* END *altrimenti verrà generato un errore di* STACK *e verrà settato un bit di sistema identificato dal* BIT 18 *del registro* R4.
Affinchè questo errore non si presenti è sufficiente far coincidere il numero e la lunghezza delle istruzioni PUSH *e* POP

ESEMPI:

POPB	CONTENUTO BIT 0 STACK = 1
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 0
Dopo l'esec. Della istruz.	ACCUMULATORE R(009).0 = 1

Memorizza un valore nello stack **PUSH**

Istruzione	Sorgente	Destinazione	Esempio
PUSH.*	R(nnn)		PUSH.LW R(300)
PUSH.*	@R(nnn)		PUSH.HW @R(301)

* Formato Istruzione

HW	LW
L	

Dimensione formato
Word
Long

Descrizione:

L'istruzione PUSH memorizza nello stack il valore (della lunghezza definita WORD o LONG) prelevato dal registro sorgente specificato.
Il contatore di STACK viene incrementato del numero di BYTE corrispondenti alla lunghezza del formato della istruzione PUSH:

- 2 BYTE per la lunghezza WORD
- 4 BYTE per la lunghezza LONG.

E' importante ricordare che il contatore di STACK *deve essere zero quando il program counter incontra l'istruzione* END *altrimenti verrà generato un errore di* STACK *e verrà settato un bit di sistema identificato dal* BIT 18 *del registro* R4.
Affinchè questo errore non si presenti è sufficiente far coincidere il numero e la lunghezza delle istruzioni PUSH *e* POP.

R(nnn) = numero che indica un registro interno, ed è compreso tra 0 e 900.

ESEMPI:

	HW	LW	
Prima dell' l'esec. della istruz.	x	4800	Contenuto R(320)
PUSH.LW R(320)			
Cont. WORD STACK	xxxxxxxxxxxxxx		
Dopo l'esecuzione della istruz.	4800		Contenuto STACK

	R(350)	R(320)
Prima dell' l'esec. della istruz.	320	7800
Contenuto di LONG STACK	xxxxxxxxxxxxxxxxx	
PUSH.L @R(350)		
Dopo l'esecuzione della istruz.		
Contenuto dello STACK	7800	

Memorizza un bit nello stack **PUSHB**

Istruzione	Sorgente	Destinazione	Esempio
PUSHB			PUSHB

Descrizione:

Questa istruzione memorizza nella cima dello STACK il contenuto a 32bit dell'accumulatore a BIT R(009). Questa istruzione può essere considerata identica alla istruzione PUSH.L R(nnn) , con la sola differenza che la sorgente è fissa ed è accumulatore R(009).

E' importante ricordare che il contatore di STACK *deve essere zero quando il program counter incontra l'istruzione* END *altrimenti verrà generato un errore di* STACK *e verrà settato un bit di sistema identificato dal* BIT 18 *del registro* R4.
Affinchè questo errore non si presenti è sufficiente far coincidere il numero e la lunghezza delle istruzioni PUSH *e* POP.

ESEMPI:

PUSHB	ACCUMULATORE R(009).0 = 1
Dopo l'esec. Della istruz.	
Cima dello STACK	VALORE A 32 BIT DEL REGISTRO R(009)

Leggi da un indirizzo assoluto **RD**

Istruzione	Sorgente	Destinazione	Esempio
RD.*	@Valore	R(nnn)	RD.B @0xFEF10,R(300)
RD.*	@R(nnn)	R(nnn)	RD.L @R(400),R(310)

* Formato Istruzione	Dimensione formato
B	Byte
W	Word
L	Long

Descrizione:

Questa istruzione legge un BYTE, WORD o LONG da un indirizzo a 32bit e lo memorizza su un registro specificato in destinazione.
Con questo tipo di istruzione è possibile leggere anche tutti quei registri interni non disponibili con le normali istruzioni MOV. Questo permette di entrare in modo più approfondito nell' hardware del sistema riuscendo a leggere anche i registri interni del processore.

Decimale	Esadecimale	Binario
@VALORE	@ 0xVALORE	@0bVALORE

R(nnn) = numero che indica un registro interno, ed è compreso tra 0 e 900.

ESEMPI:

```
MOV.L  #00112233,R(300)   ;memorizzo il valore LUNGO specificato dentro R(300)
NOP                        ; il valore FF3C0 corrisponde all'indirizzo del registro R(300)
RD.B   @FF3C3,R(310)      ;leggo il BYTE memorizzato alla locazione FF3C3
NOP                        ; e lo memorizzo nel  LLB del registro R(310).
RD.W   @FF3C2,R(311)      ;leggo il WORD memorizzato alle locazioni FF3C2  ed FF3C3
NOP                        ; e lo memorizzo nel WORD LW del registro R(311)
RD.L   @FF3C0,R(312)      ;leggo tutto il registro R(300) e lo memorizzo nel registro R(312)

;IL PROGRAMMA CHE SEGUE E' IDENTICO AL PRECEDENTE SOLO CHE FUNZIONA
;MEMORIZZANDO GLI INDIRIZZI ALL'INTERNO DI REGISTRI ED UTILIZZANDOLI
;IN MODO INDIRETTO.

MOV.L  #00112233,R(300)   ;memorizzo il valore LUNGO specificato dentro R(300)
MOV.L  #FF3C3,R(301)
MOV.L  #FF3C2,R(302)
MOV.L  #FF3C0,R(303)

RD.B   @R(301),R(310)
RD.W   @R(302),R(311)
RD.L   @R(303),R(312)
```

Reset di un bit **RESB**

Istruzione	Sorgente	Destinazione	Esempio
RESB		R(nnn).nbit	RESB R(430).04
RESB	R(nnn)	R(nnn)	RESB R(300),R(301)
RESB		@R(nnn).nbit	RESB @R(400).31
RESB	@R(nnn)	R(nnn)	RESB @R(300),R(301)

Descrizione:

Questa istruzione forza allo stato logico 0 il bit indicato da nbit (numero che va da 0 a 31) all'interno di un registro indicato con la sintassi R(nnn) (valore compreso nei range da 0 a 900).
L'ISTRUZIONE VIENE ESEGUITA SOLO SE IL CONTENUTO DELL'ACCUMULATORE A BIT (R (009).0 E' ALLO STATO LOGICO 1.

ESEMPI:

In tutti gli esempi sottoriportati si è considerato l'accumulatore nella condizione logica 1, se fosse stato a 0 l'istruzione non sarebbe stata eseguita.

	Bit da 31…..0 della locazione R(300)
RESB R(300). 0	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.1
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 1
	Bit da 31…..0 della locazione R(300)
Dopo l'esec. Della istruz.	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.0
	Bit da 31…..0 della locazione R(300)
RESB R(300),R(301)	1.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x
Contenuto del registro R(301)	31
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 1
Dopo l'esec. Della istruz.	0.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x
	Bit da 31…..0 della locazione R(450)
RESB @R(300).10	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.1.x.x.x.x.x.x.x.x.x.x
Contenuto del registro R(300)	450
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 1
BIT ACC R(009).0	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.0.x.x.x.x.x.x.x.x.x
Contenuto del registro R(300)	320
Contenuto del registro R(301)	31
	Bit da 31…..0 della locazione R(320)
Prima dell'esec. della istruz.	1.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 1
RESB @R(300),R(301)	
Dopo l'esec. della istruz.	0.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x

Reset contatore **RESC**

Istruzione	Sorgente	Destinazione	Esempio
RESC	#VALORE		RESC #5
RESC	R(nnn)		RESC R(300)
RESC	@R(nnn)		RESC @R(400)

Descrizione:

Questa istruzione esegue il reset di un contatore identificato dalla sorgente #VALORE (numero decimale da 0 a 23). Dopo l'esecuzione della istruzione il set del contatore viene riportato :

- Al valore 0 se il contatore è impostato UP
- Al valore di preset se il contatore è impostato come DOWN.
- I contatti del contatore utilizzati lungo il programma vengono rispettivamente riportati nelle condizioni di inizio conteggio, aperti se erano NA o chiusi se erano NC.

L'ISTRUZIONE VIENE ESEGUITA SOLO SE IL CONTENUTO DELL'ACCUMULATORE A BIT (R (009).0 E' ALLO STATO LOGICO 1.

ESEMPI:

In tutti gli esempi sottoriportati si è considerato l'accumulatore nella condizione logica 1, se fosse stato a 0 l'istruzione non sarebbe stata eseguita.

```
NOP
RESC #5              ; azzera il contatore N° 5
NOP

MOV.L   #10,R(300)
RESC  R(300)         ; azzera il contatore N° 10 memorizzato come valore
NOP                  ; nel registro R(300)
NOP

MOV.L #23,R(310)
MOV.L #310,R(320)
RESC @R(320)         ; in questo caso viene azzerato il contatore N°23 memorizzato
NOP                  ; nel registro R(310), ma puntato dal registro R(320).
```

Ritorno da subroutine **RET**

Istruzione	Sorgente	Destinazione	Esempio
RET			RET

Descrizione:

Questa istruzione viene utilizzata al termine di una subroutine chiamata precedentemente con l'istruzione SUB.
Dopo l'istruzione viene prelevato dallo STACK il valore del contatore di programma dell'istruzione successiva alla chiamata SUB, e quindi eseguita l'istruzione successiva.

ESEMPIO:

```
NOP
NOP
SUB     AZZRAM          ;chiamata della subroutine AZZRAM
MOV.L   #425,R(400)

AZZRAM:
ADD R(500),R(300)
RET                     ; il controllo ritorna al programma principale
NOP                     ; e si esegue l'istruzione successiva alla SUB.
```

Ritorno da interrupt **RETI**

Istruzione	Sorgente	Destinazione	Esempio
RETI			RETI

Descrizione:

Questa istruzione viene utilizzata al termine di una routine di interrupt per passare il controllo nuovamente al programma principale.
L'evento di interrupt è asincrono rispetto al programma principale, per cui quando si verifica il contatore di programma della istruzione corrente viene memorizzato sullo STACK, per poi essere nuovamente riprelevato dopo l'istruzione RETI.
In questo modo si mantiene traccia della posizione del programma corrente al momento in cui si presenta la condizione di interrupt.
Per saperne di più sul funzionamento degli interrupt fare riferimento al capitolo funzioni di sistema.

ESEMPIO:

```
MOV.L  #mysvrtim,R(300)          ;memorizzo l'indirizzo della routine di interrupt
        MOV.L    #10000,R(301)
        SYS      #43,R(300)      ;chiamo la routine di interrupt
        .
        .
mysvrtim:                        ;questa è una routine di interrupt
        ABSCPLB  R(20).0
        RETI                     ; ritorna dalla routine di interrupt.
```

Rotazione a sinistra **ROTL**

Istruzione	Sorgente	Destinazione	Esempio
ROTL.*		R(nnn)	ROTL.LLB R(300)
ROTL.*		@R(nnn)	ROTL.LW @R(320)

* Formato Istruzione

HHB	HLB	LHB	LLB
HW		LW	
L			

Dimensione formato
Byte
Word
Long

Descrizione:
Questa istruzione esegue una rotazione da sinistra a destra di un bit nel registro selezionato.
Il bit più significativo viene ruotato sul bit meno significativo.
La rotazione avviene solo all'interno del formato prescelto come indicato negli esempi sottoriportati.

R(nnn) = numero che indica un registro interno, ed è compreso tra 0 e 900.

ESEMPI:

	HHB	HLB	LHB	LLB	
Prima dell' l'esec. della istruz.	x	x	x	80	Contenuto R(300)
ROTL.LLB R(300)					
Dopo l'esecuzione della istruz.	x	x	x	01	Contenuto R(300)

	HW	LW	
Prima dell' l'esec. della istruz.	00000001	x	Contenuto R(320)
ROTL.HW R(320)			
Dopo l'esecuzione della istruz.	00000010	x	Contenuto R(320)

	HW	LW	
Prima dell' l'esec. della istruz.	x	10000000	Contenuto R(310)
ROTL.LW R(310)			
Dopo l'esecuzione della istruz.	x	00000001	Contenuto R(310)

Prima dell' l'esec. della istruz.	1000100010001000	Contenuto R(320)
Contenuto di R(300)	320	Contenuto R(300)
ROTL.L @R(300)		
Dopo l'esecuzione della istruz.	0001000100010001	Contenuto R(310)

Rotazione a destra **ROTR**

Istruzione	Sorgente	Destinazione	Esempio
ROTR.*		R(nnn)	ROTR.LLB R(300)
ROTR.*		@R(nnn)	ROTR.LW @R(320)

* Formato Istruzione

				Dimensione formato
HHB	HLB	LHB	LLB	Byte
HW		LW		Word
L				Long

Descrizione:
Questa istruzione esegue una rotazione da destra a sinistra di un bit nel registro selezionato.
Il bit meno significativo viene ruotato sul bit più significativo.
La rotazione avviene solo all'interno del formato prescelto come indicato negli esempi sottoriportati.

R(nnn) = numero che indica un registro interno, ed è compreso tra 0 e 900.

ESEMPI:

	HHB	HLB	LHB	LLB	
Prima dell' l'esec. della istruz.	x	x	x	01	Contenuto R(300)
ROTR.LLB R(300)					
Dopo l'esecuzione della istruz.	x	x	x	80	Contenuto R(300)

	HW	LW	
Prima dell' l'esec. della istruz.	00000010	x	Contenuto R(320)
ROTR.HW R(320)			
Dopo l'esecuzione della istruz.	00000001	x	Contenuto R(320)

	HW	LW	
Prima dell' l'esec. della istruz.	x	00000001	Contenuto R(310)
ROTR.LW R(310)			
Dopo l'esecuzione della istruz.	x	10000000	Contenuto R(310)

Prima dell' l'esec. della istruz.	0001000100010001	Contenuto R(320)
Contenuto di R(300)	320	Contenuto R(300)
ROTR.L @R(300)		
Dopo l'esecuzione della istruz.	1000100010001000	Contenuto R(310)

Set fronte di salita **RSEDGE**

Istruzione	Sorgente	Destinazione	Esempio
RSEDGE		#VALORE	RSEDGE
RSEDGE		@R(nnn)	RSEDGE

Descrizione:

Questa istruzione è denominata " FRONTE DI SALITA " ed ha il compito di mantenere il proprio BIT allo stato logico 1 ogni volta che l'ACCUMULATORE (R009.0) è allo stato logico 1.
Questo Bit rimarrà ad 1 solo per una scansione di programma per poi ritornare subito a zero alla scansione di programma successiva e rimanervi per tutto il tempo che il bit dell'accumulatore è allo stato logico 1.
La condizione verrà resettata solo quando l'accumulatore ritorna allo stato logico 0.

R(nnn) = numero che indica un registro interno, ed è compreso tra 0 e 900.

#VALORE = numero decimale da 0 a 31.

ESEMPI:

```
ABSSETB   R(009).0
RSEGDE     #0                  ; Il FRONTE N°0 rimarrà ad 1 per una scansione di programma.
NOP

ABSSETB   R(009).0
MOV.L      #31,R(300)
RSEGDE     R(300)              ; Il FRONTE N°31 rimarrà ad 1 per una scansione di programma.
NOP
```

Setta un bit **SETB**

Istruzione	Sorgente	Destinazione	Esempio
SETB		R(nnn).nbit	SETB R(430).04
SETB	R(nnn)	R(nnn)	SETB R(300),R(301)
SETB		@R(nnn).nbit	SETB @R(400).31
SETB	@R(nnn)	R(nnn)	SETB @R(300),R(301)

Descrizione:

Questa istruzione forza allo stato logico 1 il bit indicato da nbit (numero che va da 0 a 31) all'interno di un registro indicato con la sintassi R(nnn) (valore compreso nei range da 0 a 900).
L'ISTRUZIONE VIENE ESEGUITA SOLO SE IL CONTENUTO DELL'ACCUMULATORE A BIT R (009).0 E' ALLO STATO LOGICO 1.

ESEMPI:

In tutti gli esempi sottoriportati si è considerato l'accumulatore nella condizione logica 1, se fosse stato a 0 l'istruzione non sarebbe stata eseguita.

	Bit da 31…..0 della locazione R(300)
SETB R(300). 0	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.0
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 1
	Bit da 31…..0 della locazione R(300)
Dopo l'esec. Della istruz.	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.1
	Bit da 31…..0 della locazione R(300)
SETB R(300),R(301)	0.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x
Contenuto del registro R(301)	31
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 1
Dopo l'esec. Della istruz.	1.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x
	Bit da 31…..0 della locazione R(450)
SETB @R(300).10	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.0.x.x.x.x.x.x.x.x.x.x.x.x.x
Contenuto del registro R(300)	450
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 1
BIT ACC R(009).0	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.1.x.x.x.x.x.x.x.x.x.x.x
Contenuto del registro R(300)	320
Contenuto del registro R(301)	31
	Bit da 31…..0 della locazione R(320)
Prima dell'esec. della istruz.	0.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 1
SETB @R(300),R(301)	
Dopo l'esec. della istruz.	1.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x

Settaggio parametri orologio **SETCLOCK**

SETCLOCK	PARAMETRI SET	MODO DI FUNZIONAMENTO
SETCLOCK	#VALORE	#VALORE
SETCLOCK	#VALORE	R(nnn)
SETCLOCK	#VALORE	@R(nnn)
SETCLOCK	R(nnn)	R(nnn)
SETCLOCK	R(nnn)	@R(nnn)

PARAMETRI SET #VALORE: indica il set dell'orologio, e dipende dal modo di funzionamento.

MODI FUNZIONAMENTO # VALORE di LLB:
1 START su HH-MM
2 START su HH-MM STOP su HH-MM
3 START su ME-GG-HH-MM
4 START su ME-GG STOP su ME-GG
5 START su GG giorno della settimana
6 START su GG giorno settimana stop GG settimana

#VALORE di LHB: va da 0 a 7 e definisce il numero di orologio selezionato.

Descrizione:

Questa istruzione inizializza uno degli 8 orologi interni al PLC. Ogni orologio viene identificato con 2 parametri che nell'ordine sono:
PARAMETRI SET : valore che indica i numeri di SET dell'orologio. Per inserire i valori è più comodo utilizzare la notazione esadecimale. I parametri saranno memorizzati in un registro a 32 bit del numero di orologio selezionato secondo la modalità indicata.

REGISTRO OROLOGIO CORRENTE

HHB	HLB	LHB	LLB	MODO DI FUNZIONAMENTO
00	00	HH	MM	START su ORA e MINUTI (HH-MM)
HHstop	MMstop	HHstart	MM start	START su HH-MM e STOP su HH-MM
GG	ME	HH	MM	START su ME-GG-HH-MM
Ggstop	MEstop	GGstart	MMstart	START su ME-GG STOP su ME-GG
00	00	GG	00	START su GG della settimana
00	00	GGstart	GGstop	START su GG settimana STOP su giorno sett

MODO FUNZIONAMENTO :

- 1 START su HH-MM. l'orologio selezionato chiuderà il proprio contatto quando l'ora ed i minuti selezionati corrisponderanno all'ora ed i minuti dell'orologio hardware interno. Questo contatto rimarrà chiuso per tutto il minuto corrente.
- 2 START su HH-MM STOP su HH-MM. L'orologio selezionato chiuderà il proprio contatto al raggiungimento dei valori prefissati per poi riaprirsi al raggiungimento di HH-MM STOP.
- 3 START su ME-GG-HH-MM. Il contatto dell'orologio si chiuderà al raggiungimento dei valori impostati per poi riaprirsi il minuto successivo.
- 4 START su ME-GG STOP su ME-GG. Il contatto dell'orologio si chiuderà al raggiungimento del Mese e Giorno prefissati su START e si riaprirà un minuto dopo aver raggiunto i parametri ME-GG STOP.

- 5 START su GG della settimana. L'orologio chiuderà il proprio contatto al raggiungimento del giorno settimanale prefissato, e lo riaprirà un minuto dopo il passaggio del giorno.
- 6 START su GG della settimana STOP su GG della settimana. L'orologio chiuderà il proprio contatto al raggiungimento del giorno settimanale prefissato, e lo riaprirà un minuto dopo il passaggio del giorno di STOP impostato.

ESEMPI:

```
SETCLOCK  #0x0A1E0A1E, #0x0002  ; i due valori vengono memorizzati nei due registri nel
NOP                             ; seguente modo.
;orologio N° 1 impostato nel modo 2 ORA START = 10,30  e ORA STOP = 10,33.
```

REGISTRO OROLOGIO CORRENTE (32 bit)				REGISTRO MODO DI FUNZION. (16 bit)	
HHB	HLB	LHB	LLB	LHB	LLB
0A=10	1E=30	0A=10	21=33	0	2
Ora Stop	Min stop	Ora Start	Min Start	Orologio 1	Modo 2

```
SETCLOCK  #0x00A1E, #0x0001  ; i due valori vengono memorizzati nei due registri nel
NOP                          ;nel seguente modo.
;Orologio N° 1 impostato nel modo 1 ORA START = 12,30.
```

REGISTRO OROLOGIO CORRENTE (32 bit)				REGISTRO MODO DI FUNZION. (16 bit)	
HHB	HLB	LHB	LLB	LHB	LLB
00	00	0C=12	1E=30	0	1
		Ora Start	Min Start	Orologio 1	Modo 1

```
;stesso esempio del precedente ma con i valori dei settaggi memorizzati in due registri

MOV.L   #0xA1E,R(300)
MOV.L   #1,R(301)
SETCLOCK  R(300),R(301)

;stesso esempio del precedente ma con il #VALORE di modo puntato dal registro R(310)

MOV.L   #0xA1E,R(300)
MOV.L   #1,R(301)
MOV.L   #301,R(310)
SETCLOCK  R(300),@R(310)
```

Settaggio parametri contatore **SETCOUNTER**

SETCOUNTER	MODO CONTEGGIO.	VALORE PRESET	NUMERO CONTAT.
SETCOUNTER	#VALORE	#VALORE	#VALORE
SETCOUNTER	#VALORE	R(nnn)	#VALORE
SETCOUNTER	#VALORE	@R(nnn)	#VALORE
SETCOUNTER	R(nnn)	R(nnn)	#VALORE
SETCOUNTER	R(nnn)	@R(nnn)	#VALORE

MODO CONTEGGIO	# VALORE: 1 UP 2 DOWN 3 AUOT-UP 4 AUTO-DOWN	R(nnn) =0-900	
VALORE PRESET	#VALORE = 0-32000	R(nnn) =0-900	@R(nnn)=0-900
NUMERO CONTATORE	#VALORE = 0-23		

Descrizione:

Questa istruzione inizializza uno dei 24 contatori interni al PLC. Ogni contatore viene identificato con 3 parametri che nell'ordine sono:

- NUMERO CONTATORE : valore compreso tra 0 e 23 (massimo numero contatori inserbile)
- VALORE DI PRESET : identifica il valore di fine o inizio conteggio (UP/DOWN)
- MODO CONTEGGIO : 1 = UP, 2 = DOWN, 3 = AUTO-UP, 4 = AUTO-DOWN.

ESEMPI:

```
SETCOUNTER      #1,32000,1    ;CONTATORE N°2 impostato in modo UP a 32000 impulsi

MOV.L           #2000,R(300)
SETCOUNTER      #2,R(300),0   ;CONTATORE N°1 impostato in modo DOWN a 2000 impulsi
NOP                           ; valore memorizzato nel registro R(300).

;programma identico al precedente ma con il registro R(300) puntato dal registro R(310)
MOV.L           #2000,R(300)
MOV.L           #300,R(310)
SETCOUNTER      #2,@R(310),0 ;CONTATORE N°1 impostato  DOWN a 2000 impulsi
NOP                           ; valore memorizzato nel registro R(300).

MOV.L           #1,R(400)
MOV.L           #3450,R(300)
SETCOUNTER      R(400),R(300),23 ;CONT. N°23 impostato a3450(R300) impulsi UP R(400)

programma identico al precedente ma con il registro R(300) puntato dal registro R(310)
MOV.L           #1,R(400)
MOV.L           #3450,R(300)
MOV.L           #300,R(310)
SETCOUNTER      R(400),@R(310),23 ;CONT. N° 23(R400) impostato a3450(R300) impulsi UP
```

Settaggio parametri DAC **SETDAC**

SETDAC	NUMERO DAC
SETDAC	#VALORE
SETDAC	R(nnn)

NUMERO DAC #VALORE = 0/1 R(nnn) = 0 -900

Descrizione:

Con questa istruzione viene abilitato uno dei 2 DAC (convertitore digitale analogico) ad 8 bit disponibili sul PLC. I DAC vengono inseriti a scapito di 2 dei 3 ADC (convertitore analogico digitale) a 10 bit disponibili a morsettiera. Per abilitare i DAC oltre che alla istruzione SETDAC è necessario cortocicuitare 2 ponticelli strip all'interno del PLC.

ESEMPI:

```
SETDAC    #1            ; DAC N° 2 abilitato.

MOV.L     #0,R(300)
SETDAC    R(300)        ;DAC N° 1 abilitato.
```

Settaggio parametri registro encoder **SETENCODER**

SETENCODER	MODO	VALORE DIVISORE
SETENCODER	#VALORE	#VALORE
SETENCODER	#VALORE	R(nnn)
SETENCODER	#VALORE	@R(nnn)

MODO :	# VALORE**:**	MODO CK
	0 ENCODER (VALORE ASSOLUTO)	CK INT.
	1 CONTATORI (VALORI ASSOLUTI)	CK INT.
	2 ENCODER CLOCK ESTERNO (VALORE ASSOLUTO)	CK EXT.
	3 CONTATORE 1 (VALORE ASSOLUTO)	CK EXT.
	4 CONTATORE 2 (VALORE ASSOLUTO)	CK EXT.
	256 ENCODER (VALORE RELATIVO)	CK INT.
	257 CONTATORI (VALORI REALTIVI)	CK INT.
	258 ENCODER CLOCK ESTERNO (VALORE RELATIVO)	CK EXT.
	259 CONTATORE 1 (VALORE RELATIVO)	CK EXT.
	260 CONTATORE 2 (VALORE RELATIVO)	CK EXT.
VAL.DIVISORE	#VALORE	
	225 - 65535 campionamento su CLOCK INTERNO	
	0 - 65535 campionamento su CLOCK ESTERNO	

Descrizione:

Questa istruzione inizializza il modo di funzionamento della lettura da encoder o da contatori esterni nelle seguenti modalità:

ENCODER ASSOLUTO MODO= 0 RELATIVO MODO = 256
Si imposta il modo di funzionamento su encoder assoluto a due fasi A B. Il conteggio è aggiornato tenendo conto del segno (senso di rotazione) degli impulsi di clock.
Il campo frequenza in questo caso imposta la frequenza alla quale il conteggio viene aggiornato.Se, ad esempio, impostiamo 5000hz, ogni 200microsecondi sarà aggiornata la posizione dell'encoder.
La memoria Encoder contiene la posizione assoluta mentre la memoria contatore 1 contiene il valore relativo all'ultimo rinfresco.
CONTATORI ASSOLUTO MODO= 1 RELATIVO MODO = 257
Si imposta il modo di funzionamento su due contatori diversi su canale A e canale B. Il conteggio è assoluto, su ogni clock del canale corrispondente, senza segno (sempre positivo).
Il campo frequenza in questo caso imposta la frequenza alla quale il conteggio viene aggiornato. Se ad esempio mettiamo 5000hz ogni 200micor secondi saranno aggiornate le posizioni dei contatori
ENCODER SU CLOCK AUTOMATICO ASSOLUTO MODO= 2 RELATIVO MODO = 258
Si imposta il modo di funzionamento su encoder assoluto a due fasi A B. Il conteggio e aggiornato tenendo conto del segno (senzo di rotazione) degli impulsi di clock
Il campo frequenza in questo caso imposta il divisore con cui vengono divisi gli impulsi di ingresso per generare l'interrupt di aggiornamento (modo raggiungimento posizione veloce)
CONTATORE SU CANALE 0 ASSOLUTO MODO= 3 RELATIVO MODO = 259
Si imposta il modo di funzionamento su due contatori diversi su canale A o canale B. Il conteggio è assoluto su ogni clock del canale corrispondente e senza segno (sempre positivo).
Il campo frequenza in questo caso imposta il divisore con cui vengono divisi gli impulsi di ingresso del canale A per generare l'interrupt di aggiornamento (misura di frequenza).

CONTATORE SU CANALE 1 ASSOLUTO MODO= 4 RELATIVO MODO = 260
Si imposta il modo di funzionamento su due contatori diversi su canale A o canale B. Il conteggio è assoluto su ogni clock del canale corrispondente e senza segno (sempre positivo).
Il campo frequenza in questo caso imposta il divisore con cui vengono divisi gli impulsi di ingresso del canale B per generare l'interrupt di aggiornamento (misura di frequenza).
Se una modalità indica una lettura ASSOLUTA sul registroa 32 bit dove viene memorizzato il valore dell' encoder R(275) troveremo il valore ASSOLUTO della misura dal primo campionamento effettuato, mentre nel registro successivo R(276) verrà memorizzato il valore RELATIVO rispetto al penultimo campionamento effettuato R(276) = V(t) – V(t-1) dove V(t) valore assoluto corrente e V(t-1) valore assoluto rilevato nel campionamento precedente.
Se , invece, la modalità indica una lettura RELATIVA sul registro R(275) viene memorizzato il valore RELATIVO rispetto al penultimo campionamento effettuato R(276) = V(t) – V(t-1) dove V(t) valore assoluto corrente e V(t-1) valore assoluto rilevato nel campionamento predcedente.

VALORI DIVISORE #VALORE :
Nel caso sia impostato un settaggio su CLOCK INTERNO imposta il divisore per ottenere la frequenza di refresh richiesta . In questo caso #VALORE sarà compreso tra 2950 e 65535; infatti dividendo la frequenza di clock del processore che è pari a 14,7456 MHZ per il valore minimo si ottiene una frequenza di campionamento pari a 5000Hz che è la massima frequenza di campionamento del valore letto da encoder. Utilizzando lo stesso procedimento per il massimo numero inseribile in #VALORE si otterrà la minima frequenza di campionamento pari a 225 Hz.

Nel caso che la impostazione sia su CLOCK ESTERNO il parametro #VALORE è un numero compreso tra 0 e 65535, in questo caso il numero inserito è un divisore della frequenza di clock dell'encoder o di un eventuale contatore esterno. Supponendo che la frequenza di clock sia 1Mhz se si dividesse per un #VALORE di 1000 verrà generato un interrupt lettutra encoder o contatore ogni 1000HZ.

ESEMPI:

```
SETENCODER  #0,#7373    ;imposta encoder su MODO 0 VALORE ASSOLUTO E CLOCK
NOP                     ;INTERNO CON FREQUENZA DI CAMPIONAMENTO
NOP                     ;PARI A 2000Hz, 0,5 mS.

SETENCODER #256,#14745 ;imposta encoder  su MODO 256 VALORE RELATIVO e CLOCK
NOP                     ;INTERNO con FREQUENZA di CAMPIONAMENTO PARI  a
NOP                     ; 1000HZ, 1mS.

;questa istruzione è uguale alla precedente perché ,il valore del divisore è stato impostato su R(300)
MOV.L  #14745,R(300)
SETENCODER #256,r(300)

;stessa istruzione ma R(300) è puntato da R(310) in modo indiretto.
MOV.L #14745,R(300)
MOV.L #300,R(310)
SETENCODER #256,@R(310)
```

Settaggio parametri registri di posizionamento **SETPOS**

SETPOS	VALORE a 32bit	N° POSIZIONATORE
SETPOS	#VALORE	#VALORE
SETPOS	#VALORE	R(nnn)
SETPOS	#VALORE	@R(nnn)
SETPOS	R(nnn)	R(nnn)
SETPOS	R(nnn)	@R(nnn)

Decimale	Esadecimale	Binario	
#VALORE	# 0xVALORE	# 0bVALORE	VALORE A 32 bit con segno

N° POSIZIONATORE va da 0-7
R(nnn) = numero che indica un registro interno, ed è compreso tra 0 e 900.

Descrizione:

Questa istruzione memorizza su uno degli 8 registri dedicati ai posizionatori da R(279) ad R(286) un valore con segno a 32bit. Quando il PLC legge uan posizione da ENCODER la confronta ad interrupt (in base al tempo campionato ed impostato con l'istruzione SETENCODER) con il valore e/o i valori memorizzati sui registri di posizionamento. Dal risulato del confronto vengono settati i bit interni del registro R(44):

- BIT da 0..7 per i rispettivi posizionatori da 0..7 nel caso che il valore ENCODER registro R(275) SIA >= del #VALORE memorizzato sul posizionatore interessato.
- BIT da 8..15 per i rispettivi posizionatori da 0..7 nel caso che il valore ENCODER registro R(275) SIA < del #VALORE memorizzato sul posizionatore interessato.

ESEMPI:

```
SETPOS     #1000000000, #0  ;memorizzo  nel  registro R(279)  il  valore decimale di 1.000.000.000
LDB          R(44).0          ;se il valore letto dall'encoder è > di R(279)  setto ad 1 il BIT 0 di R(44)
STB          R(46).0          ; e lo memorizzo sul rele interno 1.

MOV.L   #135000,R(300)      ;memorizzo il valore decimale 135.000 nel registro R(300)
SETPOS   R(300),#1          ;memorizzo il contenuto del registro R(300) sul registro del
NOP                         ; posizionatore N°2

;identica alla precedente. Qui il registro R(300)  è puntato in modo indiretto da R(310).
MOV.L   #300,R(310)
MOV.L   #135000,R(300)      ;memorizzo il valore decimale 135.000 nel registro R(300)
SETPOS  @R(310),#1          ;memorizzo il contenuto del registro R(300) sul registro del
NOP                         ; posizionatore N°2

;identica alla precedente. Qui il registro R(300)  è puntato in modo indiretto da R(310).
; ed il numero del posizionatore è contenuto nel registro R(302)
MOV.L   #300,R(310)
MOV.L   #1,R(302)
MOV.L   #135000,R(300)      ;memorizzo il valore decimale 135.000 nel registro R(300)
SETPOS  @R(310),R(302)      ;memorizzo il contenuto del registro R(300) sul registro del POS. N°1
```

Settaggio parametri timers **SETTIMER**

SETTIMER	MODO TIMER	VALORE PRESET	NUMERO TIMER
SETTIMER	#VALORE	#VALORE	#VALORE
SETTIMER	#VALORE	R(nnn)	#VALORE
SETTIMER	#VALORE	@R(nnn)	#VALORE
SETTIMER	R(nnn)	R(nnn)	#VALORE
SETTIMER	R(nnn)	@R(nnn)	#VALORE

MODO TIMER	# VALORE: **0** 1/10 DI SECONDO **1** SECONDI **2** MINUTI **3** ORE	R(nnn) =0-900	
VALORE PRESET	#VALORE = 0-32000	R(nnn) =0-900	@R(nnn)=0-900
NUMERO TIMER	#VALORE = 0-23		

Descrizione:

Questa istruzione inizializza uno dei 24 timer interni al PLC. Ogni timer viene identificato con 3 parametri che nell'ordinr sono:

- NUMERO TIMER : valore compreso tra 0 e 23 (massimo numero contatori inserbile)
- VALORE DI PRESET : identifica il valore di tempo impostato compreso tra 0 e 32000
- MODO CONTEGGIO : **1** = 1/10 SEC, **2** = SEC , **3** = MINUTI, **4** = ORE.

ESEMPI:

```
SETTIMER    #2,32000,0    ;TIMER N°1 impostato con base dei tempi a SEC  tempo 32000 .

MOV.L              #2000,R(300)
SETTIMER           #1,R(300),2   ;Timer N°3 con base dei tempi a 1/10 di Sec. Tempo 2000
NOP                              ; valore memorizzato nel registro R(300).

;programma identico al precedente ma con il registro R(300) puntato dal registro R(310)
MOV.L              #2000,R(300)
MOV.L              #300,R(310)
SETTIMER           #1,@R(310),2 ; Timer N°3 con base dei tempi a 1/10 di Sec. Tempo 2000
 NOP                             ; valore memorizzato nel registro R(300).

MOV.L              #3,R(400)
MOV.L              #3450,R(300)
SETTIMER           R(400),R(300),1 ;TIMER N° 2 con base dei tempi a MIN

programma identico al precedente ma con il registro R(300) puntato dal registro R(310)
MOV.L              #3,R(400)
MOV.L              #3450,R(300)
MOV.L              #300,R(310)
SETTIMER           #R(400),@R(310),1
```

Shift logico a sinistra **SHFL**

Istruzione	Sorgente	Destinazione	Esempio
SHFL.*		R(nnn)	SHFL.LLB R(300)
SHFL.*		@R(nnn)	SHFL.LW @R(320)

* Formato Istruzione

HHB	HLB	LHB	LLB
HW		LW	
L			

Dimensione formato
Byte
Word
Long

Descrizione:
Questa istruzione esegue una spostamento a sinistra di un bit nel registro selezionato.
Il bit più significativo viene perso e sul bit meno significativo entra un bit di valore 0.
Lo shift avviene solo all'interno del formato prescelto come indicato negli esempi sottoriportati.

R(nnn) = numero che indica un registro interno, ed è compreso tra 0 e 900.

ESEMPI:

	HHB	HLB	LHB	LLB	
Prima dell' l'esec. Della istruz.	x	x	x	80	Contenuto R(300)
SHFL.LLB R(300)					
Dopo l'esecuzione della istruz.	x	x	x	00	Contenuto R(300)

	HW	LW	
Prima dell' l'esec. Della istruz.	00000001	x	Contenuto R(320)
SHFL.HW R(320)			
Dopo l'esecuzione della istruz.	00000010	x	Contenuto R(320)

	HW	LW	
Prima dell' l'esec. Della istruz.	x	11000000	Contenuto R(310)
SHFL.LW R(310)			
Dopo l'esecuzione della istruz.	x	10000000	Contenuto R(310)

Prima dell' l'esec. Della istruz.	1000100010001000	Contenuto R(320)
Contenuto di R(300)	320	Contenuto R(300)
SHFL.L @R(300)		
Dopo l'esecuzione della istruz.	0001000100010000	Contenuto R(310)

Shift logico a destra **SHFR**

Istruzione	Sorgente	Destinazione	Esempio
SHFR.*		R(nnn)	SHFR.LLB R(300)
SHFR.*		@R(nnn)	SHFR.LW @R(320)

* Formato Istruzione

HHB	HLB	LHB	LLB
HW		LW	
L			

Dimensione formato
Byte
Word
Long

Descrizione:
Questa istruzione esegue una spostamento a destra di un bit nel registro selezionato.
Il bit meno significativo viene perso e sul bit più significativo entra un bit di valore 0.
Lo shift avviene solo all'interno del formato prescelto come indicato negli esempi sottoriportati.

R(nnn) = numero che indica un registro interno, ed è compreso tra 0 e 900.

ESEMPI:

	HHB	HLB	LHB	LLB	
Prima dell' l'esec. della istruz.	x	x	x	80	Contenuto R(300)
SHFR.LLB R(300)					
Dopo l'esecuzione della istruz.	x	x	x	40	Contenuto R(300)

	HW	LW	
Prima dell' l'esec. della istruz.	00000101	x	Contenuto R(320)
SHFR.HW R(320)			
Dopo l'esecuzione della istruz.	00000010	x	Contenuto R(320)

	HW	LW	
Prima dell' l'esec. della istruz.	x	11000000	Contenuto R(310)
SHFR.LW R(310)			
Dopo l'esecuzione della istruz.	x	01100000	Contenuto R(310)

Prima dell' l'esec. della istruz.	1000100010001000	Contenuto R(320)
Contenuto di R(300)	320	Contenuto R(300)
SHFR.L @R(300)		
Dopo l'esecuzione della istruz.	0100010001000100	Contenuto R(310)

Start scansione programma **START**

Istruzione	Sorgente	Destinazione	Esempio
START			START

Descrizione:

Quando il program counter incontra questa istruzione il PLC esegue una serie di operazioni :

- Vengono copiati gli ingressi hardware sulla la RAM apposita . Quindi compilando un file assembler che non ha l'istruzione START non si ha nessuna condizione di errore, ma gli ingressi non verranno rinfrescati perché non viene effettuata nessuna copia di trasferimento alla RAM contenente l'ultima condizione richiesta .
- Viene aggiornato il BIT 11 del registro R(74) che viene invertito ogni scansione di programma.
- Viene abilitata la condizione di interrupt se è chiamata dall'apposita funzione SYS.

Memorizza lo stato dell'accumulatore in un registro **STB**

Istruzione	Sorgente	Destinazione	Esempio
STB		R(nnn).nbit	STB R(430).4
STB	R(nnn)	R(nnn)	STB R(300),R(301)
STB		@R(nnn).nbit	STB @R(400).31
STB	@R(nnn)	R(nnn)	STB (@R(300),R(301)

Descrizione:

Questa istruzione carica il contenuto dell'accumulatore per istruzioni a BIT (identificato come il BIT 0 del registro 9) sul bit contenuto da **nbit** (numero che va da 0 a 31) di un registro indicato con la sintassi **R(nnn)** (valore compreso nei range da 0 a 900).

ESEMPI:

STB R(46). 0	ACCUMULATORE R(009).0 = 1
Prima dell'esec della istruz	R(46).0 = 0
Dopo l'esec della istruz.	R(46).0 = 1

STB R(46),R(301)	ACCUMULATORE R(009).0 = 1
Contenuto del registro R(301)	31
Prima dell'esec della istruz	R(46).31 = 0
Dopo l'esec. Della istruz.	R(46).31 = 1

STB @R(300).10	ACCUMULATORE R(009).0 = 0
Contenuto del registro R(300)	46
Prima dell'esec della istruz	R(46).10 = 1
Dopo l'esec. Della istruz.	R(46).10 = 0

Contenuto del registro R(300)	46
Contenuto del registro R(301)	31
STB @R(300),R(301)	ACCUMULATORE R(009).0 = 1
Prima dell'esec della istruz	R(46).31 = 0
Dopo l'esec. della istruz.	R(46).31 = 1

Memorizza lo stato dell'accumulatore su un contatore **STC**

Istruzione	Sorgente	Destinazione	Esempio
STC	#VALORE		STC #1
STC	R(nnn)		STC R(300)
STC	@R(nnn)		STC @R(420)

#VALORE = numero compreso tra 0 e 23
R(nnn) = valore compreso tra 0 e 900

Descrizione:

Questa istruzione carica il contenuto dell'accumulatore per istruzioni a BIT (identificato come il BIT 0 del registro 9) sul bit del registro di stato dei contatori (bits compresi tra 0 e 23) .

- Se l'accumulatore = 0 non si avrà nessuna operazione.
- Se l'accumularore = 1 il contatore interessato si incrementa di 1 se in modo UP o si decrementa di 1 se impostato in modo DOWN.

ESEMPI:

STC #0	ACCUMULATORE R(009).0 = 1
Prima dell'esec della istruz	Contatore N° 1 = xxxx
Dopo l'esec della istruz.	Contatore N°1= xxxx+1

STC R(301)	ACCUMULATORE R(009).0 = 1
Contenuto del registro R(301)	23
Prima dell'esec della istruz	Contatore N° 23 = xxxx
Dopo l'esec. Della istruz.	Contatore N° 23 = xxxx+1

STC @R(300)	ACCUMULATORE R(009).0 = 0
Contenuto del registro R(300)	1
Prima dell'esec della istruz	Contatore N° 2 = xxxx
Dopo l'esec. Della istruz.	Contatore N° 2 = xxxx

Memorizza lo stato dell'accumulatore su un orologio **STCK**

Istruzione	Sorgente	Destinazione	Esempio
STK	#VALORE		STK #1
STK	R(nnn)		STK R(312)
STK	@R(nnn)		STK @(423)

#VALORE = numero compreso tra 0 e 7
R(nnn) = valore compreso tra 0 e 900

Descrizione:

Questa istruzione carica il contenuto dell'accumulatore per istruzioni a BIT (identificato come il BIT 0 del registro 9) sul bit contenuto dal registro di stato degli orologi (bits che vanno da 0 a 7) .

- Se l'accumulatore = 0 non si avrà nessuna operazione.
- Se l'accumularore = 1 l'orologio impostato dalla istruzione SETCLOCK si avvierà con la modalità prevista.

ESEMPI:

STK #0	ACCUMULATORE R(009).0 = 1
Prima dell'esec della istruz	Orologio N° 1 disabilitato
Dopo l'esec della istruz.	Orologio N° 1 abilitato

STK R(301)	ACCUMULATORE R(009).0 = 1
Contenuto del registro R(301)	7
Prima dell'esec della istruz	Orologio N° 7 disabilitato STOP
Dopo l'esec. Della istruz.	Orologio N° 7 abilitato RUN

STK @R(300)	ACCUMULATORE R(009).0 = 0
Contenuto del registro R(300)	1
Prima dell'esec della istruz	Se Orologio N°2 abilitato RUN, altrimenti STOP
Dopo l'esec. Della istruz.	Orologio N° 2 STOP

Memorizza lo stato dell'accumulatore su un timer **STT**

Istruzione	Sorgente	Destinazione	Esempio
STT	#VALORE		STT #1
STT	R(nnn)		STT R(312)
STT	@R(nnn)		STT @(423)

#VALORE = numero compreso tra 0 e 23
R(nnn) = valore compreso tra 0 e 900

Descrizione:

Questa istruzione carica il contenuto dell'accumulatore per istruzioni a BIT (identificato come il BIT 0 del registro 9) sul bit contenuto dal registro di stato dei timer (bits che vanno da 0 a 23) .

- Se l'accumulatore = 0 non si avrà nessuna operazione.
- Se l'accumularore = 1 il TIMER impostato dalla istruzione SETTIMER si avvierà con la modalità prevista .

ESEMPI:

STT #0	ACCUMULATORE R(009).0 = 1
Prima dell'esec della istruz	Timer N° 1 disabilitato
Dopo l'esec della istruz.	Timer N° 1 abilitato

STT R(301)	ACCUMULATORE R(009).0 = 1
Contenuto del registro R(301)	7
Prima dell'esec della istruz	Timer N° 7 disabilitato STOP
Dopo l'esec. Della istruz.	Timer N° 7 abilitato RUN

STT @R(300)	ACCUMULATORE R(009).0 = 0
Contenuto del registro R(300)	1
Prima dell'esec della istruz	Se Timer N°2 abilitato RUN, altrimenti STOP
Dopo l'esec. Della istruz.	Timer N° 2 STOP

Chiamata subroutine **SUB**

Istruzione	Sorgente	Destinazione	Esempio
SUB		LABEL	SUB LABEL
SUB		R(nnn)	SUB R(300)
SUB		@R(nnn)	SUB @R(400)

LABEL = Stringa che definisce l'indirizzo di salto

Descrizione:

Questa istruzione esegue un salto ad un sottoprogramma memorizzato all'indirizzo specificato da LABEL o dal contenuto di un registro.
Il contatore di programma viene salvato sullo STACK per poi essere nuovamente prelevato dall'istruzione RET.
R(nnn) = numero che indica un registro interno, ed è compreso tra 0 e 900.

ESEMPI:

```
            NOP
            SUB   AZZRAM
            MOV.L #1567,R(300)
            END
AZZRAM:
            MOV.L  #10,R(300)
            MOV.L  #400,R(302)
AZZ1        :
MOV.L  #0,@R(302)
            INC1.L  R(302)
            DJNZ  R(300), AZZ1
            RET
;il programma sotto riportato assegna l'indirizzo della subroutine al registro R(303)
            MOV.L #AZZRAM,R(303)
            SUB R(300)
            .
            .
AZZRAM:
NOP
            NOP
            RET
;In questo programma viene assegnato l'indirizzo di salto al registro R(303) che però è puntato
;indirettamente dal registro R(304)
            MOV.L #AZZRAM,R(303)
            MOV.L #303,R(304)
            SUB @R(304)
            .
AZZRAM:
            NOP
            RET
```

Operazione aritmetica di sottrazione **SUBC**

Istruzione	Sorgente	Destinazione	Esempio
SUBC.*	#Valore	R(nnn)	SUBC.L #1000,R(300)
SUBC.*	R(nnn)	R(nnn)	SUBC.L R(300),R(301)
SUBC.*	R(nnn)	@R(nnn)	SUBC.L R(300),@R(301)

* Formato Istruzione

				Dimensione formato
HHB	HLB	LHB	LLB	Byte
HW		LW		Word
L				Long

Descrizione:

Questa istruzione esegue la sottrazione tra l'operando sorgente del formato prestabilito (BYTE, WORD, LONG) ed il registro destinazione. Mette il risultato all'interno del registro destinazione specificato. Nell' operando sorgente possono essere specificati i seguenti formati :

Decimale	Esadecimale	Binario
#VALORE	# 0xVALORE	# 0bVALORE

R(nnn) = numero che indica un registro interno, ed è compreso tra 0 e 900.

ESEMPI:

Prima dell' l'esec. della istruz.
SUBC.LLB #100, R(300)
Dopo l'esecuzione della istruz.

HHB	HLB	LHB	LLB	
x	x	x	200	Contenuto R(300)
x	x	x	100	Contenuto R(300)

Prima dell' l'esec. della istruz.
SUBC.HW #01200, R(320)
Dopo l'esecuzione della istruz.

HW	LW	
2200	x	Contenuto R(320)
1000	x	Contenuto R(320)

Prima dell' l'esec. della istruz.
SUBC.L R(310), R(320)
Dopo l'esecuzione della istruz.

R(310)	R(320)
1800	4000
1800	2200

Prima dell' l'esec. della istruz.
Contenuto di R(300)
SUBC.L R(310), @R(300)
Dopo l'esecuzione della istruz.

R(310)	R(320)
135000	235000
320	
135000	100000

Scambia i dati di un registro **SWAP**

Istruzione	Sorgente	Destinazione	Esempio
SWAP.*		R(nnn)	SWAP.LW ,R(300)
SWAP.*		@R(nnn)	SWAP.L @R(301)

* Formato Istruzione

		Dimensione formato
HW	LW	Word
L		Long

R(nnn) = numero che indica un registro interno, ed è compreso tra 0 e 900.

Descrizione:
Questa istruzione scambia i byte interni ad un registro secondo la dimensione formato prescelta.
Nel formato WORD vengono scambiati il BYTE più significativo con il meno significativo del WORD interessato (LW o HW).
Nel formato LONG vengono scambiati il WORD più significativo con quello meno significativo

ESEMPI:

	HHB	HLB	LHB	LLB	
Prima dell' l'esec. della istruz.	x	x	35	200	Contenuto R(300)
SWAP.LW R(300)					
Dopo l'esecuzione della istruz.	x	x	200	35	Contenuto R(300)

	HW	LW	
Prima dell' l'esec. della istruz.	2200	4300	Contenuto R(320)
SWAP.L R(320)			
Dopo l'esecuzione della istruz.	4300	2200	Contenuto R(320)

	HW	LW	
Prima dell' l'esec. della istruz.	1800	4000	R(300)
SWAP.L @R(400)	R(400)=300		
Dopo l'esecuzione della istruz.	4000	1800	R(300)

	HHB	HLB	LHB	LLB	
Prima dell' l'esec. della istruz.	50	32	x	x	R(320)
Contenuto di R(300)	320				
SWAP.HW @R(300)					
Dopo l'esecuzione della istruz.	32	50	x	x	R(320)

Chiamata a funzione di sistema **SYS**

Istruzione	N° FUNZIONE	REG. BASE	Esempio
SYS	#Valore	R(nnn)	SYS #1,R(300)
SYS	R(nnn)	R(nnn)	SYS R(301),R(300)
SYS	@R(nnn)	R(nnn)	SYS @R(302),R(300)

Descrizione:

Questa istruzione chiama una funzione di sistema già prememorizzata nel sistema operativo del PLC.
I parametri da passare alla SYS sono:

- NUMERO DI FUNZIONE che può essere inserito direttamente o memorizzato in un registro.
- REGISTRO DI BASE che è il primo registro dove la funzione memorizza i parametri con cui ritorna la funzione stessa . Ogni funzione può ritornare da 1 a 4 parametri. Considerando, ad esempio, una funzione che ritorna con 3 parametri, ed avendo selezionato il Registro R(400) come REGISTRO di BASE, i valori di ritorno della SYS verranno memorizzati rispettivamente nei registri R(400),R(401),R(402).

ESEMPI:

```
SYS #1,R(510)          ;chiama la funzione di sistema 1 e ritorna con i valori memorizzandoli a
NOP                    ; partire dal registro R(510).

;stesso programma del precedente ma il numero di SYS è memorizzato sul registro R(300)
MOV.L   #1,R(300)
SYS  R(300),R(510)

;come sopra ma il registro su cui è memorizzata la SYS è puntato indirettamente da R(320)
MOV.L   #1,R(300)
MOV.L   #300,R(320)
SYS  R(300),R(510)
```

Scrive i dati ad un indirizzo assoluto **WR**

Istruzione	Sorgente	Destinazione	Esempio
WR.*	R(nnn)	@Valore	WR.B R(300),@0xFEF10
WR.*	R(nnn)	@R(nnn)	WR.L R(400), @R(310)

* Formato Istruzione	Dimensione formato
B	Byte
W	Word
L	Long

Descrizione:

Questa istruzione legge un BYTE, WORD o LONG da un registro interno a 32bit e ne memorizza il contenuto ad un indirizzo diretto a 32 bit.
Con questo tipo di istruzione è possibile scrivere anche tutti quei registri interni non disponibili con le normali istruzioni MOV. Questo permette di entrare in modo più approfondito nell' hardware del sistema riuscendo a scrivere anche i registri interni del processore.

Decimale	Esadecimale	Binario
@VALORE	@ 0xVALORE	@0bVALORE

R(nnn) = numero che indica un registro interno, ed è compreso tra 0 e 900.

ESEMPI:

```
MOV.L  #0x00112233,R(301)   ;memorizzo il valore LUNGO specificato dentro R(301)
NOP                          ; il valore esad. 00112233
WR.B   R(310),@0xFF3C3      ;scrivo il BYTE memorizzato in R(310) alla locazione FF3C3
NOP                          ; LLB del registro R(300) indirizzo fisico = 0xFF3C3.
WR.W  R(311),@0xFF3C2       ;scrive il WORD memorizzato in R(311) alle locazioni FF3C2  ed FF3C3
NOP                          ;
WR.L    R(312), @0xFF3C0    ;scrivo tutto il registro R(312) nel registro R(300)

;IL PROGRAMMA CHE SEGUE E' IDENTICO AL PRECEDENTE SOLO CHE FUNZIONA
;MEMORIZZANDO GLI INDIRIZZI ALL'INTERNO DI REGISTRI ED UTILIZZANDOLI
;IN MODO INDIRETTO.

MOV.L  #00112233,R(301)   ;memorizzo il valore LUNGO specificato dentro R(301)
MOV.L  #FF3C3,R(301)
MOV.L  #FF3C2,R(302)
MOV.L  #FF3C0,R(303)

WR.B        R(310),@R(301)
WR.W        R(311),@R(302)
WR.L        R(312),@R(303)
```

Or esclusivo **XOR**

Istruzione	Sorgente	Destinazione	Esempio
XOR.*	#Valore	R(nnn)	XOR.LW #0FF00,R(300)
XOR.*	R(nnn)	R(nnn)	XOR.L R(300),R(301)
XOR.*	R(nnn)	@R(nnn)	XOR.L R(300),@R(301)

* Formato Istruzione

				Dimensione formato
HHB	HLB	LHB	LLB	Byte
HW		LW		Word
L				Long

Descrizione:

Questa istruzione esegue l' XOR logico (OR esclusivo) tra l'operando sorgente del formato prestabilito (BYTE, WORD, LONG) ed il registro destinazione. Mette il risultato all'interno del registro destinazione specificato. Nell' operando sorgente possono essere specificati i seguenti formati :

Decimale	Esadecimale	Binario
#VALORE	# 0xVALORE	# 0bVALORE

R(nnn) = numero che indica un registro interno, ed è compreso tra 0 e 900.

ESEMPI:

	HHB	HLB	LHB	LLB	
Prima dell' l'esec. della istruz.	x	x	x	0F	Contenuto R(300)
XOR.LLB #0xFF, R(300)					
Dopo l'esecuzione della istruz.	x	x	x	F0	Contenuto R(300)

	HW	LW	
Prima dell' l'esec. della istruz.	0F0F	x	Contenuto R(320)
XOR.HW #0xFFF0, R(320)			
Dopo l'esecuzione della istruz.	F0FF	x	Contenuto R(320)

	R(310)	R(320)
Prima dell' l'esec. della istruz.	0000FF	FF00FF
XOR.L R(310), R(320)		
Dopo l'esecuzione della istruz.	0000FF	FF0000

	R(310)	R(320)
Prima dell' l'esec. della istruz.	00000F	0000F0
Contenuto di R(300)	320	
XOR.L R(310), @R(300)		
Dopo l'esecuzione della istruz.	00000F	0000FF

Or esclusivo a bit **XORB**

Istruzione	Sorgente	Destinazione	Esempio
XORB		R(nnn).nbit	XORB R(430).4
XORB	R(nnn)	R(nnn)	XORB R(300),R(301)
XORB		@R(nnn).nbit	XORB @R(400).31
XORB	@R(nnn)	R(nnn)	XORB (@R(300),R(301)

Descrizione:

Questa istruzione esegue l' XOR (OR ESCLUSIVO) del bit indicato da **nbit** (numero che va da 0 a 31) all'interno di un registro indicato con la sintassi **R(nnn)** (valore compreso nei range da 0 a 900) e l'accumulatore per istruzioni a BIT che è identificato come il BIT 0 del registro 9. Il risultato dell'operazione viene memorizzato nell'accumulatore.

ESEMPI:

	Bit da 31…..0 della locazione R(300)
XORB R(300). 0	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.1
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 0
Dopo l'esec. Della istruz.	ACCUMULATORE R(009).0 = 1

	Bit da 31…..0 della locazione R(300)
XORB R(300),R(301)	0.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x
Contenuto del registro R(301)	31
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 0
Dopo l'esec. Della istruz.	ACCUMULATORE R(009).0 = 0

	Bit da 31…..0 della locazione R(450)
XORB @R(300).10	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.1.x.x.x.x.x.x.x.x.x.x
Contenuto del registro R(300)	450
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 1
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 0

Contenuto del registro R(300)	320
Contenuto del registro R(301)	31
	Bit da 31…..0 della locazione R(320)
Prima dell'esec. della istruz.	1.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 1
XORB @R(300),R(301)	
Dopo l'esec. della istruz.	ACCUMULATORE R(009).0 = 0

Or esclusivo a bit negato **XORB!**

Istruzione	Sorgente	Destinazione	Esempio
XORB!		R(nnn).nbit	XORB ! (R(430).4)
XORB!	R(nnn)	R(nnn)	XORB ! (R(300),R(301))
XORB!		@R(nnn).nbit	XORB ! (@R(400).31)
XORB!	@R(nnn)	R(nnn)	XORB ! (@R(300),R(301))

Descrizione:

Questa istruzione esegue l' XOR del bit NEGATO indicato da **nbit** (numero che va da 0 a 31) all'interno di un registro indicato con la sintassi **R(nnn)** (valore compreso nei range da 0 a 900) e l'accumulatore per istruzioni a BIT che è identificato come il BIT 0 del registro 9. Il risultato dell'operazione viene memorizzato nell'accumulatore.

ESEMPI:

	Bit da 31…..0 della locazione R(300)
XORB ! (R(300). 0)	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.1
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 0
Dopo l'esec. Della istruz.	ACCUMULATORE R(009).0 = 0

	Bit da 31…..0 della locazione R(300)
XORB !(R(300),R(301))	0.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x
Contenuto del registro R(301)	31
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 0
Dopo l'esec. Della istruz.	ACCUMULATORE R(009).0 = 1

	Bit da 31…..0 della locazione R(450)
XORB ! (@R(300).10)	x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.1.x.x.x.x.x.x.x.x.x.x
Contenuto del registro R(300)	450
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 1
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 1

Contenuto del registro R(300)	320
Contenuto del registro R(301)	31
	Bit da 31…..0 della locazione R(320)
Prima dell'esec. della istruz.	1.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x.x
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 0
XORB !(@R(300),R(301))	
Dopo l'esec. della istruz.	ACCUMULATORE R(009).0 = 0

Or esclusivo con il bit di stack **XORBPOP**

Istruzione	Sorgente	Destinazione	Esempio
XORBPOP			XORBPOP

Descrizione:

Questa istruzione esegue l' XOR logico a bit tra l'accumulatore ed il contenuto del BIT localizzato all'interno dell stack ed inserito per mezzo dell'istruzione PUSHB.

ESEMPI:

XORBPOP	CONTENUTO BITSTACK = 1
BIT ACC R(009).0	ACCUMULATORE R(009).0 = 0
Dopo l'esec. Della istruz.	ACCUMULATORE R(009).0 = 1

- ## *COMUNICAZIONE SERIALE*

La comunicazione seriale del ***MicroPLC Merlino*** è stata appositamente progettata in modo da realizzare sistemi distribuiti punto a punto con un'ampia varietà di comandi e semplice codifica decodifica dei messaggi scambiati. Ogni ***MicroPLC Merlino*** ha due porte di comunicazione seriale ***Com 0*** e ***Com 1.*** All'accensione della macchina queste porte sono impostate :

Com 0 ***57600*** baud ***8*** dati ***1*** Stop ***N***essuna parità
Com 1 ***115200*** baud ***8*** dati ***1*** Stop ***N***essuna parità

Ad ogni ***MicroPLC Merlino*** è assegnato un numero di scheda che va da 0 a 255 si può così gestire reti attraverso un computer di supervisione. La porta di comunicazione specifica per questo scopo è la ***Com 1*** .Inoltre ogni ***MicroPLC Merlino*** può funzionare da ***Master*** dalla porta di comunicazione ***Com 0*** e visto che sono stati implementati i protocolli di gestione sia delle espansioni ***EXP8IO*** che per supportare ***MicroPLC D1208*** ,oltre naturalmente ad altri ***MicroPLC Merlino***, si possono costruire reti piramidali accentrate, dall'alto verso il basso.La configurazione minima per una rete master slave è composta da 1 a 5 slave .In questa tipologia si può direttamente comandare altri 40 I/O senza nessuna aggiunta hardware oltre i 5 slave. Considerando inoltre la presenza della interfaccia di comunicazione RS-485 si può distribuire la rete in modalità a stella con un raggio di 1Km. La porta di comunicazione ***Com 1*** funziona solo da slave la porta ***Com 0*** può funzionare anche da master.

- ## *protocollo di comunicazione:*

Ogni ***MicroPLC Merlino*** può essere configurato con velocità seriale da 600 baud a 115 Kbaud 8 bit di dati , 1 di stop, nessuna parità. Il protocollo di comunicazione è derivato dal ***MicroPLC 1208*** e uso lo stesso modo di formattazione dei messaggi. Lo scambio di messaggi tra un computer di supervisione o tra ***MicroPLC Merlino*** master e slave è identico e si basa su scambio pacchetti di dati a lunghezza variabile che includono le seguenti informazioni:

carattere di **START** messaggio	255
Numero del ***MicroPLC Merlino*** interessato del messaggio.	da 0 a 255
Numero di caratteri complessivi di tutto il messaggio.	0…255
Comando ***MicroPLC***	
Eventuali dati in base al Comando che interessa il messaggio.	
Checksum messaggio,somma a 256 di tutti i byte di questo messaggio dal Carattre di START alla checksum esclusa.	

esempio:

START	NUM PLC	LEN MSG	COMANDO	MOD 256
0xff	0x01	x05	0x00	0x05
255	1	5	0	5

dove

0xff	Comando di start messaggio
0x01	riferito al *MicroPLC Merlino* numero 1
0x05	lunghezza messaggio 5 bytes
0x00	Comando di test presenza scheda (non prevede nessuno scanbio di dati)
0x05	somma di tutti i bytes precedenti.

Ogni volta che un ***MicroPLC Merlino*** riceve un messaggio indirizzato ad un'altro ***MicroPLC*** si disinteressa di questo e aspetta l'inizio di un nuovo messaggio. Quando riceve un messaggio indirizzato a se risponde in maniera adeguata al Comando ricevuto. Considerando che per la comunicazone il ***MicroPLC Merlino h***a molti comandi diversi che implicano un numero eventuale di dati variabile risulta difficile calcolare i tempi di comunicazione che possono intercorrere in una rete formata da diversi ***MicroPLC***. Facendo riferimento al suo predecessore bisogna però rilevare che la risposta avviene immediatamente dopo la ricezione dell'ultimo carattere ricevuto senza i 20ms che invece attendeva il ***MicroPLC D1208*** .Il ***MicroPLC Merlino*** può ricevere e trasmettere contemporaneamente da ambedue i canali di comunicazione. Per lo scambio di messaggi è stato assunto che ogni messaggio di risposta sia costituito con il Comando che è stato ricevuto nel messaggio di interrogazione sommato a 128.In questo modo è stato limitato a 127 il numero massimo di Comandi eventualmente disponibili per la comunicazione del ***MicroPLC*** ma si riesce a risalire dal messaggio di risposta al messaggio di interrogazione e questo è stato ritenuto importante soprattutto nello sviluppo di programmi di supervisione, che così possono implementare la comunicazione con buffer di comunicazione e eventuale decodifica in processi separati.In ogni risposta ad una qualsiasi interrogazione seriale il ***MicroPLC Merlino*** inserisce nel messaggio di risposta il carattere 0x00 prima del carattere di Start messaggio (0xff). Questo carattere non viene considerato nel campo che indica la lunghezza del messaggio.

- ***Comandi Comunicazione Seriale:***

Segue elenco dettagliato dei comandi ammessi per la comunicazione seriale del ***MicroPLC***.
Buona parte di questi comandi sono stati implementati per rispettare una piena compatibilità con il suo predecessore D1208, ma ad attenta lettura appare evidente che la maggior parte di questi si può rimpiazzare con i comandi di lettura scrittura della memoria che tra l'altro sono stati potenziati, quindi utile anche in previsione di compatibilità future l'uso massiccio di questi comandi a discapito di quelli dedicati.
Sintassi :
Allo scopo di non complicare ulteriormente la leggibilità dei messaggi di esempio si applicano le seguenti abbreviazioni:

START	Valore 255 (0xff) Comando di start messaggio
CHKSUM	CheckSum somma algebrica di tutti i byte componenti il messaggio ,modulo 256 esempio 255,01,05,12,CHKSUM=255+1->256-> allora 5+12=**17**
LEN	Numero dei byte componenti il messaggio
PLC	Numero PLC interessato dal messaggio 0..255

- **Comando TEST_SCHEDA valore 0x00**

Serve essenzialmente per vedere la presenza di una scheda nella rete dei ***MicroPLC***,
a questo Comando si ha in risposta **ACKW** (valore 0x80)
esempio :

Messaggio di interrogazione

START	**PLC**	**LEN**	**TEST_SCHEDA**	**CHK**

Messaggio di risposta

START	**PLC**	**LEN**	**TEST_SCHEDA+128**	**CHK**

- **Comando WRITE_INPUT valore 0x03**[1]

Serve per scrivere il solo stato degli ingressi del PLC
esempio :

Messaggio di interrogazione								
START	**PLC**	**LEN**	**WRITE_INPUT**	**hhb**	**hlb**	**lhb**	**llb**	**CHK**
Messaggio di risposta								
START	**PLC**	**LEN**	**WRITE_INPUT+128**					**CHK**

nel ***MicroPLC Merlino*** la lettura degli ingressi viene effettuata ogni passaggio per l'istruzione start se il PLC è in Run oppure di continuo se il PLC si trova nello stato di stop per cui quando si forza lo stato degli ingressi tramite comunicazione seriale questi vengono modificati solo per il tempo che rimane al prossimo aggiornamento automatico. Di solito questa azione basta a forzare lo stato degli ingressi quando il ***MicroPLC Merlino*** sta eseguendo un programma.

- **Comando WRITE_OUTPUT valore 0x04**

Quando un ***MicroPLC Merlino*** riceve questo Comando come risposta da un messaggio che contiene il solo Comando di risposta. Nel area dati del messaggio di interrogazione vi sono bytes a cui vengono impostate le uscite**.** Nel ***MicroPLC*** lo stato delle uscite viene controllato da un flag con il quale si possono abilitare o disabilitare , quindi quando si usa questo Comando bisogna accertarsi dello stato di questo flag. Inoltre se nel ***MicroPLC Merlino*** vi è un programma in **RUN** questi reimpostera autonomamente le uscite in base alla logica di programma. Se invece il ***MicroPLC Merlino*** si trova nelle condizioni di **STOP** ad una forzatura di stato seriale si imposta le uscite che rimarranno in questo stato fino alla prossima scrittura da seriale.

Messaggio di interrogazione								
START	**PLC**	**LEN**	**WRITE_OUTPUT**	**hhb**	**hlb**	**lhb**	**llb**	**CHK**
Messaggio di risposta								
START	**PLC**	**LEN**	**WRITE_OUTPUT+128**					**CHK**

- **Comando WR_O_RD_I valore 0x05**

Questo comando serve per la comunicazione master slave verso ***MicroPLC D1208 o EXP8IO*** .Per questo usa una forma del messaggio compatibile con questi apparecchi

Messaggio di interrogazione				Impostazione uscite				
START	**PLC**	**LEN**	**WR_O_RD_I**	**hhb**	**hlb**	**lhb**	**llb**	**CHK**
Messaggio di risposta				Stato degli ingressi				
START	**PLC**	**LEN**	**WR_O_RD_I +128**	**hhb**	**hlb**	**lhb**	**llb**	**CHK**

- **Comando WR_O_RD_O valore 0x06**

Questo comando imposta lo stato delle uscite e ritorna la lettura di queste.
E'stato inserito per il funzionamento del ***MicroPLC Merlino*** come scheda remota su computer. Con questo comando infatti si ha automaticamente verifica di quanto scritto.

Messaggio di interrogazione				Impostazione uscite				
START	**PLC**	**LEN**	**WR_O_RD_O**	**hhb**	**hlb**	**lhb**	**llb**	**CHK**
Messaggio di risposta				Stato delle uscite				
START	**PLC**	**LEN**	**WR_O_RD_O +128**	**hhb**	**hlb**	**lhb**	**llb**	**CHK**

[1] I comandi con valore 1,2 non sono stati implementati

- **Comando WR_O_RD_IO Valore 0x07**

Con questo Comando si ha la combinazione dei comandi **WRITE_OUTPUT ,LETTURA degli INGRESSI e LETTURA delle USCITE** Infatti il ***MicroPLC Merlino*** ,riceve nell'area dati del messaggio di interrogazione il bytes a cui impostare le uscite e risponde ponendo nell'area dati del messaggio di risposta lo stato degli ingressi e lo stato delle uscite .Questo Comando è stato inserito per il funzionamento del ***MicroPLC Merlino*** come scheda remota su computer. Con questo comando infatti si ha automaticamente verifica di quanto scritto e in più lo stato degli ingressi tutto in una sola comunicazione.

Messaggio di interrogazione				Impostazione uscite				
START	**PLC**	**LEN**	**WR_O_RD_IO**	**hhb**	**Hlb**	**lhb**	**llb**	**CHK**
Messaggio di risposta				Stato delle uscite				
START	**PLC**	**LEN**	**WR_O_RD_O +128**	**hhb**	**Hlb**	**lhb**	**llb**	
				Stato degli ingressi				
				hhb	**Hlb**	**lhb**	**llb**	**CHK**

- **Comando READ_ANI Valore 0x08**

Questo comando ritorna le memorie che contengono i valori analogici .

Messaggio di interrogazione								
START	**PLC**	**LEN**	**READ_ANI**					**CHK**
Messaggio di risposta				ANI0		ANI1		
START	**PLC**	**LEN**	**READ_ANI +128**	**hb**	**Lb**	**hb**	**lb**	
ANI2		ANI3		ANI4		ANI5		
Hb	**lb**	**hb**	**lb**	**hb**	**Lb**	**hb**	**lb**	
ANI6		ANI7		DAC0		DAC1		
Hb	**lb**	**hb**	**lb**	**hb**	**lb**	**hb**	**lb**	**CHK**

- **Comando READ_ENC valore 0x09**

Questo comando ritorna le memorie che riguardano l'encoder.

Messaggio di interrogazione								
START	**PLC**	**LEN**	**READ_ENC**					**CHK**
Messaggio di risposta				Stato Encoder				
START	**PLC**	**LEN**	**READ_ENC +128**	**hhb**	**hlb**	**lhb**	**llb**	
Stato Contatore 0				Stato Contatore 1				
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Impostazione rate				Posizione 0				
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Posizione 1				Posizione 2				
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Posizione 3				Posizione 4				
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Posizione 5				Posizione 6				
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Posizione 7								
hhb	**hlb**	**lhb**	**llb**					**CHK**

- **Comando RESET valore 0x0a**

Imposta lo Stop del PLC ,reimposta le variabili interne allo stato iniziale

Messaggio di interrogazione

START	**PLC**	**LEN**	**RESET**		**CHK**
Messaggio di risposta					
START	**PLC**	**LEN**	**RESET +128**		**CHK**

- **Comando CHECK_TYPE valore 0x0b**

interroga il ***MicroPLC*** per saper e di che tipo è l'area dati del messaggio di risposta si trovano in sequenza il tipo di scheda ,il numero degli ingressi e il numero delle uscite e dei canali analogici.

Messaggio di interrogazione								
START	**PLC**	**LEN**	**CHECK_TYPE**					**CHK**
Messaggio di risposta				Tipo	in	out	ani	
START	**PLC**	**LEN**	**CHECK_TYPE +128**	04	18	18	04	**CHK**

- **Comando READ_PLC valore 0x0c**

Legge lo stato interno dei parametri principali di tutto il ***MicroPLC***.

Messaggio di interrogazione								
START	**PLC**	**LEN**	**READ_PLC**					**CHK**
Messaggio di risposta				Stato Uscite				
START	**PLC**	**LEN**	**READ_PLC +128**	**hhb**	**hlb**	**lhb**	**llb**	
Stato Ingressi				Stato bobine contatori				
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Stato bobine timer				Stato bobine orologi				
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Stato fronti				Stato bobine relè blocco 1				
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Stato bobine relè blocco 2				Stato bobine relè blocco 3				
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Stato bobine relè blocco 4				Stato bobine posizionatori				
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Stato Contatti Speciali								
hhb	**hlb**	**lhb**	**llb**					**CHK**

- **Comando WRITE_OROL valore 0x0d**

Imposta lo stato delle bobine di uscita di un orologio

Messaggio di interrogazione				Dato ingresso				
START	**PLC**	**LEN**	**WRITE_OROL**	**hhb**	**hlb**	**lhb**	**llb**	**CHK**
Messaggio di risposta								
START	**PLC**	**LEN**	**WRITE_OROL +128**					**CHK**

- **Comando READ_SLAVE1 valore 0x0e**

Legge lo stato interno dei parametri principali per il ***MicroPLC***. Slave 1

Messaggio di interrogazione					
START	**PLC**	**LEN**	**READ_SLAVE1**		**CHK**
Messaggio di risposta				Stato Uscite	

START	**PLC**	**LEN**	**READ_ SLAVE1 +128**	**hhb**	**hlb**	**lhb**	**llb**	
Stato Ingressi				Stato bobine contatori				
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Stato bobine timer				Stato bobine rele uscita				
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Stato bobine rele ingresso				Memoria di configurazione				
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Mem. EXCHOS1				Mem. EXCHOS1+1				
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Mem. EXCHOS1+2				Mem. EXCHOS1+3				
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Mem. EXCHIS1				Mem. EXCHIS1+2				
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Mem. EXCHIS1+2				Mem. EXCHIS1+3				
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	**CHK**

- **Comando READ_SLAVE2 valore 0x0f**

Legge lo stato interno dei parametri principali per il ***MicroPLC***. Slave 2

Messaggio di interrogazione

START	**PLC**	**LEN**	**READ_SLAVE2**					**CHK**
Messaggio di risposta				Stato Uscite				
START	**PLC**	**LEN**	**READ_ SLAVE2 +128**	**hhb**	**hlb**	**lhb**	**llb**	
Stato Ingressi				Stato bobine contatori				
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Stato bobine timer				Stato bobine rele uscita				
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Stato bobine rele ingresso				Memoria di configurazione				
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Mem. EXCHOS2				Mem. EXCHOS2+1				
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Mem. EXCHOS2+2				Mem. EXCHOS2+3				
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Mem. EXCHIS2				Mem. EXCHIS2+2				
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Mem. EXCHIS2+2				Mem. EXCHIS2+3				
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	**CHK**

- **Comando SET_FLAG valore 0x10**

Imposta lo stato del flag delle uscite.

Messaggio di interrogazione				Dato impostazione	
START	**PLC**	**LEN**	**SET_FLAG**	**8 bit**	**CHK**
Messaggio di risposta				Dato impostato	
START	**PLC**	**LEN**	**SET_FLAG +128**	**8 bit**	**CHK**

- **Comando READ_SLAVE3 valore 0x11**

Legge lo stato interno dei parametri principali per il ***MicroPLC***. Slave 3

Messaggio di interrogazione

START	**PLC**	**LEN**	**READ_SLAVE3**					**CHK**
Messaggio di risposta				Stato Uscite				
START	**PLC**	**LEN**	**READ_ SLAVE3 +128**	**hhb**	**hlb**	**lhb**	**llb**	
Stato Ingressi				Stato bobine contatori				
hhb	**Hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Stato bobine timer				Stato bobine rele uscita				
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Stato bobine rele ingresso				Memoria di configurazione				
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Mem. EXCHOS3				Mem. EXCHOS3+1				
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Mem. EXCHOS3+2				Mem. EXCHOS3+3				
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Mem. EXCHIS3				Mem. EXCHIS3+2				
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Mem. EXCHIS3+2				Mem. EXCHIS3+3				
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	**CHK**

- **Comando READ_SLAVE4 valore 0x12**

Legge lo stato interno dei parametri principali per il ***MicroPLC***. Slave 4

Messaggio di interrogazione

START	**PLC**	**LEN**	**READ_SLAVE4**					**CHK**
Messaggio di risposta				Stato Uscite				
START	**PLC**	**LEN**	**READ_ SLAVE4 +128**	**hhb**	**hlb**	**lhb**	**llb**	
Stato Ingressi				Stato bobine contatori				
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Stato bobine timer				Stato bobine rele uscita				
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Stato bobine rele ingresso				Memoria di configurazione				
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Mem. EXCHOS4				Mem. EXCHOS4+1				
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Mem. EXCHOS4+2				Mem. EXCHOS4+3				
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Mem. EXCHIS4				Mem. EXCHIS4+2				
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Mem. EXCHIS4+2				Mem. EXCHIS4+3				
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	**CHK**

- **Comando WR_S_RD_S valore 0x13**

Questo comando è usato per l'interscambio dati tra ***Merlino e Merlino*** e scambia pacchetti di 16 caratteri. Funziona in due modi diversi se il PLC è in impostato in modo master oppure no. Un PLC che gestisce una rete di ***MicroPLC Merlino*** in modo master scrive in uscita il contenuto delle bobine di interscambio ***EXCHOSn*** e legge e copia il risultato nelle bobine di ingresso ***EXCHISn .***
Il ***MicroPLC Merlino*** che invece è slave quando riceve questo comando copia i dati passati nelle memorie ***EXCHIS1*** e risponde con il contenuto delle memorie ***EXCHOS1.***Supponendo presente una rete formata da un Master e 5 slave possiamo riassumere la seguente tabella:

PLC MASTER	
Offset	**Offset**
R(85)	**EXCHOS1**
R(86)	**EXCHO1S1**
R(87)	**EXCHO2S1**
R(88)	**EXCHO3S1**
R(105)	**EXCHIS1**
R(106)	**EXCHI1S1**
R(107)	**EXCHI2S1**
R(108)	**EXCHI3S1**

PLC SLAVE 1	
Offset	**Offset**
R(105)	**EXCHIS1**
R(106)	**EXCHI1S1**
R(107)	**EXCHI2S1**
R(108)	**EXCHI3S1**
R(85)	**EXCHOS1**
R(86)	**EXCHO1S1**
R(87)	**EXCHO2S1**
R(88)	**EXCHO3S1**

PLC MASTER	
Offset	**Offset**
R(89)	**EXCHOS2**
R(90)	**EXCHO1S2**
R(91)	**EXCHO2S2**
R(92)	**EXCHO3S2**
R(109)	**EXCHIS2**
R(110)	**EXCHI1S2**
R(111)	**EXCHI2S2**
R(112)	**EXCHI3S2**

PLC SLAVE 2	
Offset	**Offset**
R(105)	**EXCHIS1**
R(106)	**EXCHI1S1**
R(107)	**EXCHI2S1**
R(108)	**EXCHI3S1**
R(85)	**EXCHOS1**
R(86)	**EXCHO1S1**
R(87)	**EXCHO2S1**
R(88)	**EXCHO3S1**

PLC MASTER	
Offset	**Offset**
R(93)	**EXCHOS3**
R(94)	**EXCHO1S3**
R(95)	**EXCHO2S3**
R(96)	**EXCHO3S3**
R(113)	**EXCHIS3**
R(114)	**EXCHI1S3**
R(115)	**EXCHI2S3**
R(116)	**EXCHI3S3**

PLC SLAVE 3	
Offset	**Offset**
R(105)	**EXCHIS1**
R(106)	**EXCHI1S1**
R(107)	**EXCHI2S1**
R(108)	**EXCHI3S1**
R(85)	**EXCHOS1**
R(86)	**EXCHO1S1**
R(87)	**EXCHO2S1**
R(88)	**EXCHO3S1**

PLC MASTER	
Offset	**Offset**
R(97)	**EXCHOS2**
R(98)	**EXCHO1S2**
R(99)	**EXCHO2S2**
R(100)	**EXCHO3S2**
R(117)	**EXCHIS2**
R(118)	**EXCHI1S2**
R(119)	**EXCHI2S2**
R(120)	**EXCHI3S2**

PLC SLAVE 4	
Offset	**Offset**
R(105)	**EXCHIS1**
R(106)	**EXCHI1S1**
R(107)	**EXCHI2S1**
R(108)	**EXCHI3S1**
R(85)	**EXCHOS1**
R(86)	**EXCHO1S1**
R(87)	**EXCHO2S1**
R(88)	**EXCHO3S1**

PLC MASTER	
Offset	**Offset**
R(100)	**EXCHOS5**
R(101)	**EXCHO1S5**
R(102)	**EXCHO2S5**
R(103)	**EXCHO3S5**
R(121)	**EXCHIS2**
R(122)	**EXCHI1S2**
R(123)	**EXCHI2S2**
R(124)	**EXCHI3S2**

PLC SLAVE 5	
Offset	**Offset**
R(105)	**EXCHIS1**
R(106)	**EXCHI1S1**
R(107)	**EXCHI2S1**
R(108)	**EXCHI3S1**
R(85)	**EXCHOS1**
R(86)	**EXCHO1S1**
R(87)	**EXCHO2S1**
R(88)	**EXCHO3S1**

Messaggio di interrogazione						EXCHOSn		
START	**PLC**	**LEN**	**WR_S_RD_S**	**hhb**	**hlb**	**lhb**	**llb**	
	EXCHO1Sn				EXCHO2Sn			
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
	EXCHO3Sn							
hhb	**hlb**	**lhb**	**llb**					**CHK**
Messaggio di risposta					EXCHISn			
START	**PLC**	**LEN**	**WR_S_RD_S +128**	**hhb**	**hlb**	**lhb**	**llb**	
	EXCHI1Sn				EXCHI2Sn			
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
	EXCHI3Sn							
hhb	**hlb**	**lhb**	**llb**					**CHK**

- **Comando CHANGE_SPEED valore 0x14**

Cambia la velocità di comunicazione seriale. Il primo byte nell'area dati del messaggio di interrogazione contiene la nuova costante per la comunicazione seriale.
Le costanti ammesse sono le seguenti :

Definizione	Valore	
CBR_600	**1**	velocità 600 baud
CBR_1200	**2**	velocità 1200 baud
CBR_2400	**3**	velocità 2400 baud
CBR_4800	**4**	velocità 4800 baud
CBR_9600	**5**	velocità 9600 baud
CBR_19200	**6**	velocità 19200 baud
CBR_28800	**7**	velocità 28800 baud
CBR_57600	**8**	velocità 57600 baud
CBR_115000	**0**	velocità 115200 baud

I valori diversi dai precedenti vengono scartati dal ***MicroPLC Merlino*** e non cambia velocità seriale. Si ottiene comunque un messaggio di risposta valido nella vecchia velocità seriale.
Nota:
Quando il ***MicroPLC Merlino*** riceve questo messaggio valido cambia subito velocità di comunicazione seriale e risponde alla nuova velocità seriale. Questo impone di stare attenti nella programmazione di un eventuale computer supervisore poiché se per errori di comunizione seriale il comando non viene acquisito il PLC e non cambia velocità seriale mentre il computer esegue interogazioni alla nuova velocità ,bisogna impostare un timeout oltre il quale reimpostare la vecchia velocità di comunicazione.

Messaggio di interrogazione				Nuova impostazione	
START	**PLC**	**LEN**	**CHANGE_SPEED**	**Da 0 a 8**	**CHK**
Messaggio di risposta				Nuova impostazione	
START	**PLC**	**LEN**	**CHANGE_SPEED +128**	**Da 0 a 8**	**CHK**

- **Comando LOAD_PRO valore 0x15**

Caricare un programma sul ***MicroPLC Merlino*** può essere effettuato solo tramite via seriale usando il Comando LOAD_PRO .Con ogni messaggio che contiene questo Comando si riesce a programmare 64 bytes di programma nell'area programma della memoria flash del PLC.

Messaggio di interrogazione				Indirizzo memoria programma			
START	**PLC**	**LEN**	**LOAD_PRO**	**ADDR hb**		**ADDR lb**	
Char 0	**Char 1**	**Char 2**	**Char 3**	**Char 4**	**Char 5**	**Char 6**	**Char 7**
Char 8	**Char 9**	**Char 10**	**Char 11**	**Char 12**	**Char 13**	**Char 14**	**Char 15**

Char 16	**Char 17**	**Char 18**	**Char 19**	**Char 20**	**Char 21**	**Char 22**	**Char 23**
Char 24	**Char 25**	**Char 26**	**Char 27**	**Char 28**	**Char 29**	**Char 30**	**Char 31**
Char 32	**Char 33**	**Char 34**	**Char 35**	**Char 36**	**Char 37**	**Char 38**	**Char 39**
Char 40	**Char 41**	**Char 42**	**Char 43**	**Char 44**	**Char 45**	**Char 46**	**Char 47**
Char 48	**Char 49**	**Char 50**	**Char 51**	**Char 52**	**Char 53**	**Char 54**	**Char 55**
Char 56	**Char 57**	**Char 58**	**Char 59**	**Char 60**	**Char 61**	**Char 62**	**Char 63**
CHK							

Messaggio di risposta Address

START	**PLC**	**LEN**	**LOAD_PRO +128**	**ADDR hb**	**ADDR lb**	**CHK**

Questo messaggio viene monitorato solo se il PLC si trova nello stato STOP, altrimenti viene completamente ignorato. Appena il ***MicroPLC Merlino*** disabilita le uscite azzera gli errori del PLC controllo, se l'indirizzo è valido e se la memoria flash è vuota. Se l'indirizzo non è nel range giusto il messaggio viene ignorato mentre se la memoria flash è programmata imposta il bit corrispondente negli errori di sistema (**4**), quindi esegue la programmazione di questo segmento di programma .Sono ammesse 40 prove di programmazione per byte per cui nel display possono apparire messaggi che indicano che il byte non si è programmato , ma questi si riferiscono a un solo tentativo. Se veramente un byte non si è programmato viene impostato il bit corrispondente negli errori di sistema (**3**).Infine ritorna un messaggio di risposta classico con il Comando messaggio pari a LOAD_PRO+128 con un campo dati che rappresenta l'indirizzo in WORD del segmento programmato.

- **Comando SAVE_PRO valore 0x16**

Per riprendere un programma dal ***MicroPLC Merlino*** bisogna usare il Comando SAVE_PRO. In ogni messaggio di interrogazione che contiene questo Comando nell'area dati va anche impostato l'indirizzo nella memoria programma da dove prendere i dati. Quando il ***MicroPLC Merlino*** riceve questo messaggio controllo, se l'indirizzo è valido e quindi costruisce e spedisce il messaggio di risposta. Se non vi sono errori ritorna un messaggio con un'area dati che rappresenta 64 byte di programma del segmento richiesto. Appena ricevuto questo messaggio il PLC disabilita lo stato delle uscite imposta lo stato di STOP e azzera gli errori di sistema. Quindi costruisce il messaggio di risposta e lo invia.

Messaggio di interrogazione Indirizzo memoria programma

START	**PLC**	**LEN**	**SAVE_PRO**	**ADDR hb**	**ADDR lb**

Messaggio di risposta Address

START	**LEN**	**PLC**	**SAVE_PRO +128**		**ADDR hb**	**ADDR lb**	**CHK**
Char 0	**Char 1**	**Char 2**	**Char 3**	**Char 4**	**Char 5**	**Char 6**	**Char 7**
Char 8	**Char 9**	**Char 10**	**Char 11**	**Char 12**	**Char 13**	**Char 14**	**Char 15**
Char 16	**Char 17**	**Char 18**	**Char 19**	**Char 20**	**Char 21**	**Char 22**	**Char 23**
Char 24	**Char 25**	**Char 26**	**Char 27**	**Char 28**	**Char 29**	**Char 30**	**Char 31**
Char 32	**Char 33**	**Char 34**	**Char 35**	**Char 36**	**Char 37**	**Char 38**	**Char 39**
Char 40	**Char 41**	**Char 42**	**Char 43**	**Char 44**	**Char 45**	**Char 46**	**Char 47**
Char 48	**Char 49**	**Char 50**	**Char 51**	**Char 52**	**Char 53**	**Char 54**	**Char 55**
Char 56	**Char 57**	**Char 58**	**Char 59**	**Char 60**	**Char 61**	**Char 62**	**Char 63**
CHK							

- **Comando RUN valore 0x17**

Quando si esegue questo Comando il ***MicroPLC Merlino*** resetta tutte le bobine interne dei contatori ,i timer, lo stato del processore logico ed esegue il programma in memoria.
Se non vi è un programma valido in memoria o vi è un errore di sistema il ***MicroPLC Merlino*** passa autonomamente in stato di STOP ,setta il flag di errore , disabilita le uscite.

Messaggio di interrogazione

START	**PLC**	**LEN**	**RUN**	**CHK**

Messaggio di risposta DA SYSSTATOPLC R(5)=1

START	**PLC**	**LEN**	**RUN +128**	**hhb**	**hlb**	**lhb**	**llb**	**CHK**

- **Comando STOP valore 0x18**

Quando si esegue questo Comando il ***MicroPLC Merlino*** congela tutto lo stato interno e non esegue nessun passo di programma.

Messaggio di interrogazione

START	**PLC**	**LEN**	**STOP**					**CHK**
Messaggio di risposta				DA SYSSTATOPLC R(5)=0				
START	**PLC**	**LEN**	**STOP+128**	**hhb**	**hlb**	**lhb**	**llb**	**CHK**

- **Comando TRACE valore 0x19**

Impostata lo stato di debug remoto per il plc.
Nello stato di debug remoto il ***MicroPLC Merlino*** permette il debug passo passo del programma interno. Quando il ***MicroPLC Merlino*** riceve un Comando TRACE valido imposta lo stato del PLC in modalita debug remoto e imposta le memorie dedicate al debug a zero .

Messaggio di interrogazione

START	**PLC**	**LEN**	**TRACE**					**CHK**
Messaggio di risposta				DA SYSSTATOPLC R(5)=6				
START	**PLC**	**LEN**	**TRACE+128**	**hhb**	**hlb**	**lhb**	**llb**	**CHK**

- **Comando TRACE_RUN Valore 0x1a**

Permette il debug del programma in memoria del PLC. Ad ogni passo viene associato un passo di programma. Con questo Comando si riesce a fare una valida azione di debug dei programmi logici del ***MicroPLC***. Prima di usare questo Comando bisogna impostare la modalità debug remoto usando l'apposita istruzione TRACE.

Messaggio di interrogazione

START	**PLC**	**LEN**	**TRACE_RUN**					**CHK**
Messaggio di risposta				DA **SYSSTATOPLC** R(5)=6				
START	**PLC**	**LEN**	**TRACE_RUN+128**	**hhb**	**hlb**	**lhb**	**llb**	
Da **SYSACC** r(9)				Da **SYSPC** r(10)				
Hhb	**Hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Da **SYSCS** r(11)				Da **SYSDEBUG** r(14)				
Hhb	**Hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Da **SYSDEBUG1** r(15)				Da **SYSBREKP1** r(16)				
Hhb	**Hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Da **SYSBREKP2** r(17)				Da **SYSIND** r(18)				
Hhb	**Hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Da **SYSIND1** r(19)				Da **SYSERRPLC** r(4)				
Hhb	**Hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	**CHK**

Memorie	Contenuto
SYSSTATOPLC	Nella Sessione di debug questa memoria contiene 6
SYSACC	Stato dell'accumulatore all'uscita dell'istruzione
SYSPC	Stato del contatore di programma all'uscita dell'istruzione
SYSCS	Stato del contatore gli annidamento dello stack all'uscita dell'istruzione
SYSDEBUG	Hw parametro 0 che identifica l'istruzione lw parametro 1 che identifica la il parametro destinazione
SYSDEBUG1	Hw parametro 2 che identifica la il parametro sorgente lw eventuale parametro 3 se presente nella istruzione
SYSBREKP1	E' uguale al contenuto della memoria definita dal parametro sorgente
SYSBREKP2	E' uguale al contenuto della memoria definita dal parametro destinazione
SYSIND	E' uguale al contenuto della memoria indiretta puntata dal contenuto

	della memoria del parametro sorgente
SYSIND1	E' uguale al contenuto della memoria indiretta puntata dal contenuto della memoria del parametro destinazione
SYSERRPLC	Errori di sistema

- **Comando READ_SLAVE5 valore 0x1b**

Legge lo stato interno dei parametri principali per il ***MicroPLC***. Slave 5

Messaggio di interrogazione								
START	**PLC**	**LEN**	**READ_SLAVE5**					**CHK**
Messaggio di risposta				Stato Uscite				
START	**PLC**	**LEN**	**READ_ SLAVE5 +128**	**hhb**	**hlb**	**lhb**	**llb**	
Stato Ingressi				Stato bobine contatori				
Hhb	**Hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Stato bobine timer				Stato bobine rele uscita				
Hhb	**Hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Stato bobine rele ingresso				Memoria di configurazione				
Hhb	**Hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Mem. EXCHOS5				Mem. EXCHOS5+1				
Hhb	**Hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Mem. EXCHOS5+2				Mem. EXCHOS5+3				
Hhb	**Hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Mem. EXCHIS5				Mem. EXCHIS5+2				
Hhb	**Hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
Mem. EXCHIS5+2				Mem. EXCHIS5+3				
Hhb	**Hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	**CHK**

- **Comando READ_MEMORY1 valore 0x1c**

Legge il contenuto di un qualsiasi registro del PLC

Messaggio di interrogazione				Indirizzo registro da leggere		
START	**PLC**	**LEN**	**READ_MEMORY1**	**ADDR hb**	**ADDR lb**	**CHK**
Messaggio di risposta				Address		
START	**PLC**	**LEN**	**READ_MEMORY1+128**	**ADDR hb**	**ADDR lb**	
Contenuto della memoria						
Hhb		**hlb**	**lhb**	**llb**	**CHK**	

- **Comando WRITE_MEMORY1 valore 0x1d**

Scrive il contenuto di un qualsiasi registro del PLC

Messaggio di interrogazione				Indirizzo registro da leggere		
START	**PLC**	**LEN**	**WRITE_MEMORY1**	**ADDR hb**	**ADDR lb**	
Contenuto della memoria						
hhb		**hlb**	**lhb**	**llb**	**CHK**	
Messaggio di risposta				Address		
START	**PLC**	**LEN**	**WRITE_MEMORY1+128**	**ADDR hb**	**ADDR lb**	**CHK**

- **[2]Comando WR_S_RD_I valore 0x21**

Con questo Comando si imposta le bobine interne di ingresso dati e si legge lo stato degli ingressi del PLC

Messaggio di interrogazione					EXCHISn			
START	**PLC**	**LEN**	**WR_S_RD_I**	**hhb**	**hlb**	**lhb**	**llb**	
	EXCHI1Sn				EXCHI2Sn			
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
	EXCHI3Sn							
hhb	**hlb**	**lhb**	**llb**					**CHK**
Messaggio di risposta					Ingressi PLC			
START	**PLC**	**LEN**	**WR_S_RD_I +128**	**hhb**	**hlb**	**lhb**	**llb**	**CHK**

- **Comando WR_S_RD_T valore 0x22**

Con questo Comando si impostano le bobine interne di ingresso dati e si legge lo stato delle bobine dei timer del PLC

Messaggio di interrogazione					EXCHISn			
START	**PLC**	**LEN**	**WR_S_RD_T**	**hhb**	**hlb**	**lhb**	**llb**	
	EXCHI1Sn				EXCHI2Sn			
hhb	**hlb**	**lhb**	**Llb**	**hhb**	**hlb**	**lhb**	**llb**	
	EXCHI3Sn							
hhb	**hlb**	**lhb**	**Llb**					**CHK**
Messaggio di risposta					Bobine timer PLC			
START	**PLC**	**LEN**	**WR_S_RD_T+128**	**hhb**	**hlb**	**lhb**	**llb**	**CHK**

- **Comando WR_O_RD_T valore 0x23**

Con questo Comando si impostano le uscite e si legge lo stato delle bobine dei timer del PLC

Messaggio di interrogazione					Uscite del PLC			
START	**PLC**	**LEN**	**WR_O_RD_T**	**hhb**	**hlb**	**lhb**	**llb**	**CHK**
Messaggio di risposta					Bobine timer PLC			
START	**PLC**	**LEN**	**WR_O_RD_T+128**	**hhb**	**hlb**	**lhb**	**llb**	**CHK**

- **Comando WR_S_RD_C valore 0x24**

Con questo Comando si impostano le bobine interne di ingresso dati e si legge lo stato delle bobine dei contatori del PLC

Messaggio di interrogazione					EXCHISn			
START	**PLC**	**LEN**	**WR_S_RD_C**	**hhb**	**hlb**	**lhb**	**llb**	
	EXCHI1Sn				EXCHI2Sn			
hhb	**Hlb**	**lhb**	**Llb**	**hhb**	**hlb**	**lhb**	**llb**	
	EXCHI3Sn							
hhb	**Hlb**	**lhb**	**Llb**					**CHK**
Messaggio di risposta					Bobine Contatori PLC			
START	**PLC**	**LEN**	**WR_S_RD_C+128**	**hhb**	**hlb**	**lhb**	**llb**	**CHK**

[2] I comandi 0x1e,0x1f,0x20 non sono stati implementati

- **Comando WR_O_RD_C valore 0x25**

Con questo Comando si impostano le uscite del e si legge lo stato delle bobine dei contatori del PLC

Messaggio di interrogazione					Uscite del PLC			
START	**PLC**	**LEN**	**WR_O_RD_C**	**hhb**	**hlb**	**lhb**	**llb**	**CHK**
Messaggio di risposta					Bobine contatori PLC			
START	**PLC**	**LEN**	**WR_O_RD_C+128**	**hhb**	**hlb**	**lhb**	**llb**	**CHK**

- **[3]Comando WRITE_TIMER valore 0x29**

Con questo Comando si imposta lo stato nelle bobine dei timer.

Messaggio di interrogazione					Stato a cui impostare le bobine dei Timer			
START	**PLC**	**LEN**	**WRITE_TIMER**	**hhb**	**hlb**	**lhb**	**llb**	**CHK**
Messaggio di risposta								
START	**PLC**	**LEN**	**WRITE_TIMER +128**					**CHK**

- **Comando WRITE_COUNTER valore 0x2A**

Con questo Comando si imposta lo stato nelle bobine dei timer.

Messaggio di interrogazione					Stato a cui impostare le bobine dei Contatori			
START	**PLC**	**LEN**	**WRITE_COUNTER**	**hhb**	**hlb**	**lhb**	**llb**	**CHK**
Messaggio di risposta								
START	**PLC**	**LEN**	**WRITE_ COUNTER +128**					**CHK**

- **Comando WRITE_RELE valore 0x2b**

Con questo Comando si imposta lo stato nelle bobine dei relè interni.

Messaggio di interrogazione					Blocco 1 R(46)			
START	**LEN**	**PLC**	**LEN**	**hhb**	**hlb**	**lhb**	**llb**	
	Blocco 2 R(47)				Blocco 3 R(48)			
hhb	**Hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
	Blocco 4 R(49)				Blocco 5 R(50)			
hhb	**Hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
	Blocco 6 R(51)				Blocco 7 R(52)			
hhb	**Hlb**	**lhb**	**llb**	**hhb**	**hlb**	**lhb**	**llb**	
	Blocco 8 R(53)							
hhb	**Hlb**	**lhb**	**llb**					**CHK**
Messaggio di risposta								
START	**PLC**	**LEN**	**WRITE_RELE+128**					**CHK**

[3] I comandi 0x26,0x27,0x28 non sono stati implementati

- **Comando WRITE_TIME valore 0x2c**

Con questo Comando si imposta L'orologio di sistema.

Messaggio di interrogazione				ORA	MIN	SEC	GG	
START	**PLC**	**LEN**	**WRITE_TIME**	**Char**	**Char**	**Char**	**Char**	
MM	**AA**	**NumG**						
Char	**Char**	**Char**						**CHK**
Messaggio di risposta								
START	**PLC**	**LEN**	**WRITE_TIME+128**					**CHK**

- [4]**Comando WRITE_NUM_PLC valore 0x2e**

Con questo Comando si imposta il numero di PLC. Nell'area dati del messaggio di impostazione vi è un bytes che è il numero a cui viene impostato il PLC.

Messaggio di interrogazione				Nuovo Numero di PLC	
START	**PLC**	**LEN**	**WRITE_NUM_PLC**	**Char**	**CHK**
Messaggio di risposta					
START	**PLC**	**LEN**	**WRITE_NUM_PLC +128**		**CHK**

- [5]**Comando READ_ST_TIMER valore 0x31**

Con questo Comando si legge l'impostazione di un timer nel formato di comunicazione compatibile con il plc D1208.Nell'area dati del messaggio di interrogazione vi deve essere un bytes che contiene il numero di timer di cui si vuole leggere le impostazioni. Sono ammessi valori da 0 a 23.Nell'area dati del messaggio di risposta vi sono 7 bytes che indicano i seguenti campi:

Bytes	**Descrizione**
1	numero di timer a cui sono riferiti questi dati
2	high byte valore a cui è settato questo timer(set/256)
3	low byte resto del valore di set
4	high byte valore a cui è questo timer
5	low byte resto del valore a cui è questo timer
6	byte che indica la scala cui è settato: • 1 timer impostato a decimi di secondo • 2 timer impostato a secondi • 3 timer impostato a minuti • 4 timer impostato a ore qualsiasi altro valore indica una impostazione errata del timer
7	byte che indica se il timer è abilitato o disabilitato

Messaggio di interrogazione				Numero di timer	
START	**PLC**	**LEN**	**READ_ST_TIMER**	**Char**	**CHK**
Messaggio di risposta					
START	**PLC**	**LEN**	**READ_ST_TIMER +128**	**Char** **Char** **Char** **Char**	
Char	**Char**	**Char**			**CHK**

[4] Il comando 0x2d non è implementato
[5] Il comando 0x2f,0x30 non è implementato

- **Comando READ_ST_COUNTER valore 0x32**

Con questo Comando si legge l'impostazione di un contatore nel formato di comunicazione compatibile con il plc D1208.Nell'area dati del messaggio di interrogazione vi deve essere un bytes che contiene il numero di contatore di cui si vuole leggere le impostazioni. Sono ammessi valori da 0 a 23.Nell'area dati del messaggio di risposta vi sono 7 bytes che indicano i seguenti campi:

Bytes	Descrizione
1	numero di contatore a cui sono riferiti questi dati
2	high byte valore a cui è settato questo contatore(set/256)
3	low byte resto del valore di set
4	high byte valore a cui è questo contatore
5	low byte resto del valore a cui è questo contatore
6	byte che indica la scala cui è settato: • 1 Conta su • 2 Conta giu • 3 Auto su (come timer veloce) • 4 Auto giu(come timer veloce) qualsiasi altro valore indica una impostazione errata del contatore
7	byte che indica se il contatore è abilitatto o disabilitato

Messaggio di interrogazione Numero di Contatore

START	**PLC**	**LEN**	**READ_ST_COUNTER**	**Char**	**CHK**

Messaggio di risposta

START	**PLC**	**LEN**	**READ_ST_COUNTER +128**	**Char**	**Char**	**Char**	**Char**
Char	**Char**	**Char**					**CHK**

- **[6]Comando READ_ST_CLOCK valore 0x35**

Con questo Comando si legge l'impostazione di un orologio. Nell'area dati del messaggio di interrogazione vi deve essere un bytes che contiene il numero di orologio di cui si vuole leggere le impostazioni. Sono ammessi valori da 0 a 3.Nell'area dati del messaggio di risposta vi sono 7 bytes che indicano i seguenti campi:

Bytes	Descrizione
1	numero di orologio a cui sono riferiti questi dati
2	1° dato impostazione
3	1° dato impostazione
4	1° dato impostazione
5	1° dato impostazione
6	Byte che indica il modo a cui è impostato l'orologio **0x01** orologio solo start ora e minuti **0x02** orologio start stop ora minuti start ora minuti sto **0x03** orologio solo start ora minuti giorno mese **0x04** orologio start stop giorno mese start giorno mese stop **0x05** orologio solo start numero del giorno della settimana **0x06** orologio start stop giorno settimana start giorno della settimana stop qualsiasi altro valore indica una impostazione errata dell'orologio

[6] I Comandi 0x33,0x34 non sono implementati

7	Byte che indica se l'orologio è abilitatto o disabilitato

Messaggio di interrogazione Numero di Orologio

START	**PLC**	**LEN**	**READ_ST_CLOCK**	**Char**		**CHK**

Messaggio di risposta

START	**PLC**	**LEN**	**READ_ST_CLOCK +128**	**Char**	**Char**	**Char**	**Char**
Char	**Char**	**Char**					**CHK**

- **Comando WRITE_ST_TIMER valore 0x36**

Con questo Comando si imposta un timer con un messaggio di comunicazione compatibile con il plc D1208. Nell'area dati del messaggio di interrogazione vi devono essere 7 bytes per l'impostazione decodificati come nel messaggio di lettura del timer (vedi READ_ST_TIMER).

Messaggio di interrogazione Numero di timer

START	**PLC**	**LEN**	**WRITE_ST_TIMER**	**Char**	**Char**	**Char**	**Char**	
Char	**Char**	**Char**						**CHK**

Messaggio di risposta

START	**PLC**	**LEN**	**WRITE _ST_TIMER+128**	**CHK**

- **Comando WRITE_ST_COUNTER valore 0x37**

Con questo Comando si imposta un contatore con un messaggio di comunicazione compatibile con il plc D1208. Nell'area dati del messaggio di interrogazione vi devono essere 7 bytes per l'impostazione decodificati come nel messaggio di lettura del contatore (vedi READ_ST_COUNTER).

Messaggio di interrogazione Numero di timer

START	**PLC**	**LEN**	**WRITE_ST_COUNTER**	**Char**	**Char**	**Char**	**Char**
Char	**Char**	**Char**					**CHK**

Messaggio di risposta

START	**PLC**	**LEN**	**WRITE_ST_COUNTER+128**	**CHK**

- **Comando WRITE_ST_CLOCK valore 0x38**

Con questo comando si imposta un'orologio con un messaggio di comunicazione compatibile con il plc D1208. Nell'area dati del messaggio di interrogazione vi devono essere 7 bytes per l'impostazione decodificati come nel messaggio di lettura del orologio (vedi READ_ST_CLOCK).

Messaggio di interrogazione Numero di timer

START	**PLC**	**LEN**	**WRITE_ST_CLOCK**	**Char**	**Char**	**Char**	**Char**	
Char	**Char**	**Char**						**CHK**

Messaggio di risposta

START	**PLC**	**LEN**	**WRITE_ST_CLOCK+128**	**CHK**

- **Comando RET_RUN valore 0x39**

Reimposta lo stato run nel PLC.E' stata implementata solo per compatibilità con D1208.

Messaggio di interrogazione				
START	**PLC**	**LEN**	**RET_RUN**	**CHK**
Messaggio di risposta				
START	**PLC**	**LEN**	**RET_RUN+128**	**CHK**

- **Comando READ_MEMORY valore 0x3b**

Permette di leggere 16 registri di sistema contemporaneamente in tutto il range dei registri 0..900.
Nel messaggio di interrogazione nel campo dati bisogna specificare l'indice dei registri da leggere

Messaggio di interrogazione				ADDR		
START	**PLC**	**LEN**	**READ_MEMORY**	**High byte**	**Low byte**	**CHK**
Messaggio di risposta				ADDR		
START	**PLC**	**LEN**	**READ_MEMORY +128**	**High byte**	**Low byte**	
		R [ADDR]			R [ADDR+1]	
hhb	**Hlb**	**lhb**	**llb**	**hhb**	**hlb** **lhb** **llb**	
		R [ADDR+2]			R [ADDR+3]	
hhb	**Hlb**	**lhb**	**llb**	**hhb**	**hlb** **lhb** **llb**	
		R [ADDR+4]			R [ADDR+5]	
hhb	**Hlb**	**lhb**	**llb**	**hhb**	**hlb** **lhb** **llb**	
		R [ADDR+6]			R [ADDR+7]	
hhb	**Hlb**	**lhb**	**llb**	**hhb**	**hlb** **lhb** **llb**	
		R [ADDR+8]			R [ADDR+9]	
hhb	**Hlb**	**lhb**	**llb**	**hhb**	**hlb** **lhb** **llb**	
		R [ADDR+10]			R [ADDR+11]	
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb** **lhb** **llb**	
		R [ADDR+12]			R [ADDR+13]	
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb** **lhb** **llb**	
		R [ADDR+14]			R [ADDR+15]	
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb** **lhb** **llb**	**CHK**

- **Comando WRITE_MEMORY valore 0x3c**

Permette di scrivere 16 registri di sistema contemporaneamente in tutto il range dei registri 0..900.
Nel messaggio di interrogazione nel campo dati bisogna specificare l'indice dei registri da scrivere, oltre ovviamente ai dati da impostare

Messaggio di interrogazione				ADDR	
START	**PLC**	**LEN**	**WRITE_MEMORY**	**High byte**	**Low byte**
		R [ADDR]			R [ADDR+1]
Hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb** **lhb** **llb**
		R [ADDR+2]			R [ADDR+3]
Hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb** **lhb** **llb**
		R [ADDR+4]			R [ADDR+5]
Hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb** **lhb** **llb**
		R [ADDR+6]			R [ADDR+7]
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb** **lhb** **llb**
		R [ADDR+8]			R [ADDR+9]
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb** **lhb** **llb**
		R [ADDR+10]			R [ADDR+11]
hhb	**hlb**	**lhb**	**llb**	**hhb**	**hlb** **lhb** **llb**

	R [ADDR+12]					R [ADDR+13]		
hhb	**hlb**	**lhb**	**llb**		**hhb**	**hlb**	**lhb**	**llb**	
	R [ADDR+14]					R [ADDR+15]		
hhb	**hlb**	**lhb**	**llb**		**hhb**	**hlb**	**lhb**	**llb**	**CHK**
Messaggio di risposta						ADDR			
START	**PLC**	**LEN**	**WRITE_MEMORY+128**			**High byte**	**Low byte**		**CHK**

- **[7]Comando WRITE_ROW valore 0x43**

Permette di scrivere una riga sul display LCD.
Il messaggio nell'area dati deve essere composto dalla posizione x,y del cursore e da una stringa al massimo di 16 caratteri.

Messaggio di interrogazione

START	**PLC**	**LEN**	**WRITE_ROW**	**X**	**Y**	**Char**	**Char**	
Char	**Char**	**Char**	**0 (fine stringa)**					**CHK**

Messaggio di risposta

START	**PLC**	**LEN**	**WRITE_ROW+128**	**CHK**

- **Comando READ_ROW valore 0x44**

Permette di leggere una riga sul display LCD.

Messaggio di interrogazione						Numero di riga 0..1		
START	**PLC**	**LEN**	**READ_ROW**			**Char**		**CHK**
Messaggio di risposta						Numero di riga 0..1		
START	**PLC**	**LEN**	**READ_ROW+128**			**Char**		
Char	**Char**	**Char**	**Char**	**Char**	**Char**	**Char**	**Char**	
Char	**Char**	**Char**	**Char**	**Char**	**Char**	**Char**	**Char**	
Stringa di 16 caratteri con il messaggio della riga								**CHK**

- **Comando WRITE_LCD valore 0x45**

Scrive tutta la pagina corrente solo come messaggio .La variabile LASTPAGE viene impostata a NULL.
Nell'area dati devono essere presenti i 32 caratteri che compongono il messaggio.

Messaggio di interrogazione						32 bytes del messaggio		
START	**PLC**	**LEN**	**WRITE_LCD**					
Char	**Char**	**Char**	**Char**	**Char**	**Char**	**Char**	**Char**	
Char	**Char**	**Char**	**Char**	**Char**	**Char**	**Char**	**Char**	
Char	**Char**	**Char**	**Char**	**Char**	**Char**	**Char**	**Char**	
Char	**Char**	**Char**	**Char**	**Char**	**Char**	**Char**	**Char**	
Messaggio di risposta						Numero di riga 0..1		
START	**PLC**	**LEN**	**WRITE_LCD +128**					**CHK**

- **Comando READ_LCD valore 0x46**

Legge tutta la pagina corrente del display lcd Nell'area dati è presente un carattere che indica l'offset del della stringa

[7] I comandi 0x3d,0x3e,0x3f,0x40,0x41,0x42 non sono implementati

Messaggio di interrogazione					Posizione di start			
START	**PLC**	**LEN**	**READ_LCD**			**Char**		**CHK**
Messaggio di risposta								
START	**PLC**	**LEN**	**READ_LCD +128**					
Char	**Char**	**Char**	**Char**	**Char**	**Char**	**Char**	**Char**	
Char	**Char**	**Char**	**Char**	**Char**	**Char**	**Char**	**Char**	
Char	**Char**	**Char**	**Char**	**Char**	**Char**	**Char**	**Char**	
Char	**Char**	**Char**	**Char**	**Char**	**Char**	**Char**	**Char**	
		32 caratteri costituenti la pagina corrente						**CHK**

• Data Base

Il sistema operativo usa i primi 300 (0..299)registri .Questo spazio di memoria viene identificato come data base del sistema operativo.Segue la tabella descrittiva di ogni singolo registro.

[8]Riferimento	Addr	Offset	Aiuto
SYSADDRSTART	0	**R(0)**	• Valore di ritorno da funzione di sistema operativo • Indirizzo del primo byte programmato all'accensione • Indirizzo programmazione per flash eprom
SYSADDRSTOP	4	**R(1)**	Indirizzo stop programmazione flash eprom
SYSFLAG	8	**R(2)**	Parametri programmazione flash eprom
SYSNUMPLC	12	**R(3)**	• LLB numero del PLC • LHB numero di PLC slave per richiesta • HLB disponibile future espansioni • HHB disponibile future espansioni
SYSERRPLC	16	**R(4)**	ERRORI sul PLC disposizione a bit: • Bit 0 : PLC in modo programmazione • Bit 1 : Il byte è programmato • Bit 2 : Manca Tensione di programmazione • Bit 3 : Il byte non si è programmato • Bit 4 : Il byte non si è cancellato • Bit 16 : Errore istruzione sconosciuta • Bit 17 : Errore Funzione sconosciuta • Bit 18 : Errore di Stack • Bit 19 : Errore Matematica (divisione per zero) • Bit 20 : Errore indirizzo
SYSSTATOPLC	20	**R(5)**	STATO PLC : (**long**) • 0 : STOP • 1 : RUN • 2 : STEP • 3 : BREAK • 4 : ERASE • 5 : LOAD • 6 : DEBUG REMOTO

[8] Con Riferimento si intende la etichetta cosi come è definita nel file Merlino.def

SYSFLAGOUT	24	**R(**6**)**	(**long**)0 = uscite disabilitate 1 = Uscite abilitate
SYSVELCOM	28	**R(7)**	**hw**= Com 1 **lw** = Com 0 Velocità di comunicazione seriale
SYSCHIAVE	32	**R(8)**	(**long**) Chiave di accesso
SYSACC	36	**R(**9**)**	(**bit 0**) Accumulatore per operazioni a bit
SYSPC	40	**R(**10**)**	Program Counter su posizione scansione del programma (**PC**)
SYSCS	44	**R(**11**)**	Contatore degli annidamenti dello stack (**CS**)
SYSFUP	48	**R(**12**)**	Copia dello stato dei Rivelatori di fronte per codifica
SYSSTART	52	**R(**13**)**	Indirizzo di rientro su istruzione START
SYSDEBUG	56	**R(**14**)**	Parametro 0 e 1 per sessione debug
SYSDEBUG1	60	**R(**15**)**	Parametro 2 e 3 per sessione debug
SYSBREKP1	64	**R(**16**)**	Indirizzo breakpoint 0
SYSBREKP2	68	**R(**17**)**	Indirizzo breakpoint 1
SYSDISP0	72	**R(**18**)**	Usato dalla sessione debug
SYSDISP1	76	**R(**19**)**	Usato dalla sessione debug
OUT	80	**R(**20**)**	32 USCITE PLC interne Bit 0=Uscita 0 Bit 31 =Uscita 31 • Se si imposta in questa memoria un bit l'uscita corrispondente si attiva (OUT1..OUT32 ladder)
OUT1	84	**R(**21**)**	32 USCITE PLC SLAVE NUMERO 1 (OUT33..OUT64)
OUT2	88	**R(**22**)**	32 USCITE PLC SLAVE NUMERO 2 (OUT65..OUT96)
OUT3	92	**R(**23**)**	32 USCITE PLC SLAVE NUMERO 3 (OUT97..OUT128)
OUT4	96	**R(**24**)**	32 USCITE PLC SLAVE NUMERO 4 (OUT129.OUT160)
OUT5	100	**R(**25**)**	32 USCITE PLC SLAVE NUMERO 5 (OUT161..OUT192)
INP	104	**R(**26**)**	32 INGRESSI INTERNI Bit 0= Ingresso 0 Bit 31 =Uscita 31 • Ingressi 0..24 fanno riferimento ai primi 18 ingressi fisici quindi se si attiva un ingresso il bit corrispondente in questa memoria va allo stato logico 1 • Gli ultimi 8 bit rispecchiano lo stato logico degli ingressi analogici un valore inferiore a 2,5volt da stato logico 0 altrimenti se la tensione è superiore a 3 volt da stato logico 1 (IN1..IN32 ladder)
INP1	108	**R(**27**)**	32 INGRESSI PLC SLAVE NUMERO 1
INP2	112	**R(**28**)**	32 INGRESSI PLC SLAVE NUMERO 2
INP3	116	**R(**29**)**	32 INGRESSI PLC SLAVE NUMERO 3
INP4	120	**R(**30**)**	32 INGRESSI PLC SLAVE NUMERO 4
INP5	124	**R(**31**)**	32 INGRESSI PLC SLAVE NUMERO 5
COUNT	128	**R(**32 **)**	32 BOBINE CONTATORI INTERNI • Se un contatore viene impostato ad un valore al superamento dello stesso la bobina corrispondente è settata a 1
COUNT1	132	**R(**33 **)**	32 BOBINE CONTATORI PLC SLAVE NUMERO 1
COUNT2	136	**R(**34 **)**	32 BOBINE CONTATORI PLC SLAVE NUMERO 2
COUNT3	140	**R(**35 **)**	32 BOBINE CONTATORI PLC SLAVE NUMERO 3
COUNT4	144	**R(**36 **)**	32 BOBINE CONTATORI PLC SLAVE NUMERO 4
COUNT5	148	**R(**37 **)**	32 BOBINE CONTATORI PLC SLAVE NUMERO 5
TIMER	152	**R(**38**)**	32 BOBINE TIMER INTERNI • Se un timer viene impostato ad un valore al superamento dello stesso la bobina corrispondente è settata a 1
TIMER1	156	**R(**39**)**	32 BOBINE TIMER PLC SLAVE NUMERO 1
TIMER2	160	**R(**40 **)**	32 BOBINE TIMER PLC SLAVE NUMERO 2
TIMER3	164	**R(**41 **)**	32 BOBINE TIMER PLC SLAVE NUMERO 3
TIMER4	168	**R(**42 **)**	32 BOBINE TIMER PLC SLAVE NUMERO 4
TIMER5	172	**R(**43 **)**	32 BOBINE TIMER PLC SLAVE NUMERO 5
BENC	176	**R(**44 **)**	32 BOBINE POSIZIONE : • Sono abilitate solo le prime 16 bobine il cui stato dipende dalla posizione dell'encoder rispetto alla posizione

			assegnata da SETPOS .Le prime 8 bobine intervengono per ENCODER>=POS.Le seconde 8 bobine intervengono per ENCODER<POS.
OROL	180	**R(**45 **)**	32 BOBINE OROLOGI INTERNI • Sono abilitate solo i primi 8 orologi e quindi le prime 8 bobine.Quando un orologio è nel range indicato dalla impostazione la bobina corrispondente è settata
R1	184	**R(**46 **)**	32 BOBINE RELE 1 INTERNI (impostati a 0 all'accenzione)
R2	188	**R(**47 **)**	32 BOBINE RELE 2 INTERNI (impostati a 0 all'accenzione)
R3	192	**R(**48 **)**	32 BOBINE RELE 2 INTERNI (impostati a 0 all'accenzione)
R4	196	**R(**49 **)**	32 BOBINE RELE 3 INTERNI (impostati a 0 all'accenzione)
R5	200	**R(**50 **)**	32 BOBINE RELE 4 INTERNI (stato ritenuto all'accenzione)
R6	204	**R(**51 **)**	32 BOBINE RELE 5 INTERNI (stato ritenuto all'accenzione)
R7	208	**R(**52 **)**	32 BOBINE RELE 6 INTERNI (stato ritenuto all'accenzione)
R8	212	**R(**53 **)**	32 BOBINE RELE 7 INTERNI (stato ritenuto all'accenzione)
RSO1	216	**R(**54 **)**	32 (**llb** dato **a** D1208 o EXP8IO) dati a PLC SLAVE 1
RSO2	220	**R(**55 **)**	32 (**llb** dato **a** D1208 o EXP8IO) dati a PLC SLAVE 2
RSO3	224	**R(**56**)**	32 (**llb** dato **a** D1208 o EXP8IO) dati a PLC SLAVE 3
RSO4	228	**R(**57**)**	32 (**llb** dato **a** D1208 o EXP8IO) dati a PLC SLAVE 4
RSO5	232	**R(**58**)**	32 (**llb** dato **a** D1208 o EXP8IO) dati a PLC SLAVE 5
RSI1	236	**R(**59**)**	32 (**llb** dato **da** D1208 o EXP8IO) dati da PLC SLAVE 1
RSI2	240	**R(**60**)**	32 (**llb** dato **da** D1208 o EXP8IO) dati da PLC SLAVE 2
RSI3	244	**R(**61**)**	32 (**llb** dato **da** D1208 o EXP8IO) dati da PLC SLAVE 3
RSI4	248	**R(**62**)**	32 (**llb** dato **da** D1208 o EXP8IO) dati da PLC SLAVE 4
RSI5	252	**R(**63**)**	32 (**llb** dato **da** D1208 o EXP8IO) dati da PLC SLAVE 5
ANI	256	**R(**64**)**	**Lw=**Ingr.Analogico 0 **Hw=** Ingr.Analogico 0 ris.10bit
ANI1	260	**R(**65**)**	**Lw=**Ingr.Analogico 2 **Hw=** Ingr.Analogico 1 ris.10bit
ANI2	264	**R(**66**)**	**Lw=**Ingr.Analogico 4 **Hw=** Ingr.Analogico 3 ris.10bit
ANI3	268	**R(**67**)**	**Lw=**Ingr.Analogico 6 **Hw=** Ingr.Analogico 7 ris.10bit
DAC	272	**R(**68**)**	**Lw=**Out.Analogico 0 **Hw=** Out.Analogico 0 ris.8bit
TIM	276	**R(**69**)**	Temporizzazioni 50SIMI,10MI,2SEC,10SEC **llb….hhb**
TIM1	280	**R(**70**)**	Temporizzazioni 30SEC,2MIN,5MIN,10MIN **llb….hhb**
TIM2	284	**R(**71**)**	Orologio di sistema **llb..hhb** • LLB = Decimi di secondo • LHB = Secondi • HLB = Minuti • HHB = Ore
TIM3	288	**R(**72**)**	Orologio di sistema **llb..hhb** • LLB = Giorno • LHB = Mese • HLB = Anno • HHB = numero del giono (0=domenica 7=sabat0)
DELAY	292	**R(**73**)**	Var. per ritardi del sistema viene incrementata ogni 20mms
SI	296	**R(**74**)**	Contatti Speciali blocco 1: • Bit 0 Inverte lo stato ogni decimo di secondo • Bit 1 Inverte lo stato ogni mezzo secondo • Bit 2 Inverte lo stato ogni secondo • Bit 3 Inverte lo stato ogni 2 secondi • Bit 4 Inverte lo stato ogni 10 secondi • Bit 5 Inverte lo stato ogni 30 secondi • Bit 6 Inverte lo stato ogni minuto • Bit 7 Inverte lo stato ogni 2 minuti • Bit 8 Inverte lo stato ogni 5 minuti • Bit 9 Inverte lo stato ogni 30 minuti • Bit 10 Inverte lo stato ogni istruzione start

			• Bit 11 Inverte lo stato primo ciclo di programma • Bit 16 Inizio trasmissione Com 0 =1 fine Tx 0 • Bit 17 Inizio trasmissione Com 1 =1 fine Tx 0 • Bit 18 Inizio Ricezione Com 0 =1 fine Rx 0 • Bit 19 Inizio Ricezione Com 1 =1 fine Rx 0 • Bit 20 a 1 se ricevuto messaggio da EXP8IO slave • Bit 21 a 1 se ricevuto messaggio da D1208 slave • Bit 22 a 1 se ricevuto messaggio da Merlino slave • (bit 20..22 vanno resettati manualmente) • Bit 23 Nuovo messaggio a Slave qualsiasi tipo • Bit 24 Inverte lo stato ogni istruzione End • Bit 25 Fine Conversione Analogica ADC imp. 1 • (bit 25 va resettato manualmente) • Bit 31 Inverte lo stato ogni interrupt Encoder semforo di posizionamento
SI1	300	**R(**75**)**	Contatti Speciali blocco 2: • Bit 0 … Bit 23 bobina istantanea timer 0..23
SI2	304	**R(**76**)**	Contatti Speciali blocco 3: • Bit 0 … Bit 23 bobina istantanea su abilitazione contatore 0..23 (se contatore abilitato conteggio bobina a 1)
SI3	308	**R(**77**)**	Contatti Speciali blocco 4:disponibili future espansioni
KEY	312	**R(**78**)**	Tastiera si sistema: • LLB = valore diretto dalla porta ingresso tasti • LHB = usato come rilevatore di fronte tasto premuto • HLB = contatto tasto premuto con in serie rilevatore di fronte • HHB = contatto tasto premuto diretto • Bit 0 hhb tasto F3 (istantaneo su hlb) • Bit 1 hhb tasto F2 (istantaneo su hlb) • Bit 2 hhb tasto F1 (istantaneo su hlb) • Bit 3 hhb tasto F4 (istantaneo su hlb) • Bit 4 hhb tasto UP (istantaneo su hlb) • Bit 5 hhb tasto DW (istantaneo su hlb) • Bit 6 hhb tasto ESC (istantaneo su hlb) • Bit 7 hhb tasto ENTER (istantaneo su hlb)
VAL	316	**R(**79**)**	Copia dell'ultimo valore immesso da tastiera
CONF1	320	**R(**80**)**	CONFIGURAZIONE SLAVE 1: Sono ammessi i seguenti parametri: 1. Il modulo slave è una EXP8IO,sono scritte 8 uscite e letti 8 ingressi .Le uscite sono impostate dal byte LLB della memoria OUT1(2..5) gli ingressi sono copiati nel byte LLB INP1(2..5) 2. Il modulo slave è un PLC D1208 usato come EXP8IO,sono scritte 8 uscite e letti 8 ingressi .Le uscite sono impostate dal byte LLB della memoria OUT1(2..5) gli ingressi sono copiati nel byte LLB INP1(2..5) 3. Il modulo slave è un PLC D1208 sono scritte 8 uscite e ritorna lo Stato degli 8 timer interni il PLC .Le uscite sono impostate dal byte LLB della memoria OUT1(2..5) lo stato dei timer viene copiato nel byte LLB della memoria TIMER1(2…5) 4. Il modulo slave è un PLC D1208 sono scritte 8 uscite e ritorna lo Stato degli 8 contatori interni il PLC .Le uscite

			sono impostate dal byte LLB della memoria OUT1(2..5) lo stato dei contatori viene copiato nel byte LLB della memoria COUNT1(2…5) 5. Il modulo slave è un PLC D1208 sono scritti 8 rele interni e ritorna lo stato degli 8 ingressi il PLC .Lo stato dei rele da scrivere è impostato dal byte LLB della memoria RSO1(2..5) lo stato degli ingressi viene copiato nel byte LLB della memoria INPUT1(2…5) 6. Il modulo slave è un PLC D1208 sono scritti 8 rele interni e ritorna lo stato degli 8 timer del PLC .Lo stato dei rele da scrivere è impostato dal byte LLB della memoria RSO1(2..5) lo stato dei timer viene copiato nel byte LLB della memoria TIMER1(2…5) 7. Il modulo slave è un PLC D1208 sono scritti 8 rele interni e ritorna lo stato degli 8 Contatori del PLC .Lo stato dei rele da scrivere è impostato dal byte LLB della memoria RSO1(2..5) lo stato dei Contatori viene copiato nel byte LLB della memoria COUNT1(2…5) 8. Il modulo slave è un PLC D1208 sono scritti 8 rele interni e ritorna lo stato di 8 rele interni del PLC .Lo stato dei rele da scrivere è impostato dal byte LLB della memoria RSO1(2..5) lo stato dei rele letti viene copiato nel byte LLB della memoria RSI1(2…5) 9. Il modulo slave è un PLC MERLINO (modo intracom). Sono passati messaggi a pacchetti di 16 byte.I byte di invio a Merlino Slave iniziano alla memoria EXCHO1S1(2…5).I byte ricevuti da merlino Slave sono copiati a partire dalla memoria EXCHI1S1(2..5). E' lasciato al programmatore l'arbitraggio dei messaggi,suggerimento : il primo byte è un contatore che identifica il messaggio ,quando nei dati di ingresso ritorna lo stesso contatore si può cambiare messaggio
CONF2	324	**R(**81**)**	CONFIGURAZIONE SLAVE 2 (vedi **CONF1**)
CONF3	328	**R(**82**)**	CONFIGURAZIONE SLAVE 3 (vedi **CONF1**)
CONF4	332	**R(**83**)**	CONFIGURAZIONE SLAVE 4 (vedi **CONF1**)
CONF5	336	**R(**84**)**	CONFIGURAZIONE SLAVE 5 (vedi **CONF1**)
EXCHOS1	340	**R(**85**)**	INTRACOM A MERLINO SLAVE 1 Questi 16 byte sono
EXCHO1S1	344	**R(**86**)**	INTRACOM A MERLINO SLAVE 1 passati alla memoria
EXCHO2S1	348	**R(**87**)**	INTRACOM A MERLINO SLAVE 1 EXCHIS1 del
EXCHO3S1	352	**R(**88**)**	INTRACOM A MERLINO SLAVE 1 MERLINO slave 1
EXCHOS2	356	**R(**89**)**	INTRACOM A MERLINO SLAVE 2 Questi 16 byte sono
EXCHO1S2	360	**R(**90**)**	INTRACOM A MERLINO SLAVE 2 passati alla memoria
EXCHO2S2	364	**R(**91**)**	INTRACOM A MERLINO SLAVE 2 EXCHIS2 del
EXCHO3S2	368	**R(**92**)**	INTRACOM A MERLINO SLAVE 2 MERLINO slave 2
EXCHOS3	372	**R(**93**)**	INTRACOM A MERLINO SLAVE 3 Questi 16 byte sono
EXCHO1S3	376	**R(**94**)**	INTRACOM A MERLINO SLAVE 3 passati alla memoria
EXCHO2S3	380	**R(**95**)**	INTRACOM A MERLINO SLAVE 3 EXCHIS3 del
EXCHO3S3	384	**R(**96**)**	INTRACOM A MERLINO SLAVE 3 MERLINO slave 3
EXCHOS4	388	**R(**97**)**	INTRACOM A MERLINO SLAVE 4 Questi 16 byte sono
EXCHO1S4	392	**R(**98**)**	INTRACOM A MERLINO SLAVE 4 passati alla memoria
EXCHO2S4	396	**R(**99**)**	INTRACOM A MERLINO SLAVE 4 EXCHIS4 del
EXCHO3S4	400	**R(**100**)**	INTRACOM A MERLINO SLAVE 4 MERLINO slave 4
EXCHOS5	404	**R(**101**)**	INTRACOM A MERLINO SLAVE 5 Questi 16 byte sono
EXCHO1S5	408	**R(**102**)**	INTRACOM A MERLINO SLAVE 5 passati alla memoria
EXCHO2S5	412	**R(**103**)**	INTRACOM A MERLINO SLAVE 5 EXCHIS5 del
EXCHO3S5	416	**R(**104**)**	INTRACOM A MERLINO SLAVE 5 MERLINO slave 5

EXCHIS1	420	**R**(105)	INTRACOM DA MERLINO SLAVE 1 Questi 16 byte sono
EXCHI1S1	424	**R**(106)	INTRACOM DA MERLINO SLAVE 1 i dati della memoria
EXCHI2S1	428	**R**(107)	INTRACOM DA MERLINO SLAVE 1 EXCHOS1 del
EXCHI3S1	432	**R**(108)	INTRACOM DA MERLINO SLAVE 1 MERLINO slave 1
EXCHIS2	436	**R**(109)	INTRACOM DA MERLINO SLAVE 2 Questi 16 byte sono
EXCHI1S2	440	**R**(110)	INTRACOM DA MERLINO SLAVE 2 i dati della memoria
EXCHI2S2	444	**R**(111)	INTRACOM DA MERLINO SLAVE 2 EXCHOS2 del
EXCHI3S2	448	**R**(112)	INTRACOM DA MERLINO SLAVE 2 MERLINO slave 2
EXCHIS3	452	**R**(113)	INTRACOM DA MERLINO SLAVE 3 Questi 16 byte sono
EXCHI1S3	456	**R**(114)	INTRACOM DA MERLINO SLAVE 3 i dati della memoria
EXCHI2S3	460	**R**(115)	INTRACOM DA MERLINO SLAVE 3 EXCHOS3 del
EXCHI3S3	464	**R**(116)	INTRACOM DA MERLINO SLAVE 3 MERLINO slave 3
EXCHIS4	468	**R**(117)	INTRACOM DA MERLINO SLAVE 4 Questi 16 byte sono
EXCHI1S4	472	**R**(118)	INTRACOM DA MERLINO SLAVE 4 i dati della memoria
EXCHI2S4	476	**R**(119)	INTRACOM DA MERLINO SLAVE 4 EXCHOS4 del
EXCHI3S4	480	**R**(120)	INTRACOM DA MERLINO SLAVE 4 MERLINO slave 4
EXCHIS5	484	**R**(121)	INTRACOM DA MERLINO SLAVE 5 Questi 16 byte sono
EXCHI1S5	488	**R**(122)	INTRACOM DA MERLINO SLAVE 5 i dati della memoria
EXCHI2S5	492	**R**(123)	INTRACOM DA MERLINO SLAVE 5 EXCHOS5 del
EXCHI3S5	496	**R**(124)	INTRACOM DA MERLINO SLAVE 5 MERLINO slave 5
COM0OUT0	500	**R**(125)	Buffer di uscita seriale porta di comunicazione COM 0.
COM0OUT1	504	**R**(126)	72 byte in uso dell'interrupt di trasmissione seriale 0.
COM0OUT2	508	**R**(127)	In questo buffer vengono appoggiati tutti i messaggi in
COM0OUT3	512	**R**(128)	trasmissione mentre ad interrupt sono inviati
COM0OUT4	516	**R**(129)	
COM0OUT5	520	**R**(130)	
COM0OUT6	524	**R**(131)	
COM0OUT7	528	**R**(132)	
COM0OUT8	532	**R**(133)	
COM0OUT9	536	**R**(134)	
COM0OUT10	540	**R**(135)	
COM0OUT11	544	**R**(136)	
COM0OUT12	548	**R**(137)	
COM0OUT13	552	**R**(138)	
COM0OUT14	556	**R**(139)	
COM0OUT15	560	**R**(140)	
COM0OUT16	564	**R**(141)	
COM0OUT17	568	**R**(142)	
COM0INP0	572	**R**(143)	Buffer di ingresso seriale porta di comunicazione COM 0.
COM0INP1	576	**R**(144)	72 byte in uso dell'interrupt di ricezione seriale 0.
COM0INP2	580	**R**(145)	In questo buffer vengono appoggiati tutti i caratteri ricevuti nei
COM0INP3	584	**R**(146)	messaggi in ricezione prima di essere decodificati
COM0INP4	588	**R**(147)	
COM0INP5	592	**R**(148)	
COM0INP6	596	**R**(149)	
COM0INP7	600	**R**(150)	
COM0INP8	604	**R**(151)	
COM0INP9	608	**R**(152)	
COM0INP10	612	**R**(153)	
COM0INP11	616	**R**(154)	
COM0INP12	620	**R**(155)	
COM0INP13	624	**R**(156)	
COM0INP14	628	**R**(157)	
COM0INP15	632	**R**(158)	
COM0INP16	636	**R**(159)	

COM0INP17	640	**R**(160)	
COM1OUT0	644	**R**(161)	Buffer di uscita seriale porta di comunicazione COM 1.
COM1OUT1	648	**R**(162)	72 byte in uso dell'interrupt di trasmissione seriale 1.
COM1OUT2	652	**R**(163)	In questo buffer vengono appoggiati tutti i messaggi in
COM1OUT3	656	**R**(164)	trasmissione mentre ad interrupt sono inviati
COM1OUT4	660	**R**(165)	
COM1OUT5	664	**R**(166)	
COM1OUT6	668	**R**(167)	
COM1OUT7	672	**R**(168)	
COM1OUT8	676	**R**(169)	
COM1OUT9	680	**R**(170)	
COM1OUT10	684	**R**(171)	
COM1OUT11	688	**R**(172)	
COM1OUT12	692	**R**(173)	
COM1OUT13	696	**R**(174)	
COM1OUT14	700	**R**(175)	
COM1OUT15	704	**R**(176)	
COM1OUT16	708	**R**(177)	
COM1OUT17	712	**R**(178)	
COM1INP0	716	**R**(179)	Buffer di ingresso seriale porta di comunicazione COM 1.
COM1INP1	720	**R**(180)	72 byte in uso dell'interrupt di ricezione seriale 1.
COM1INP2	724	**R**(181)	In questo buffer vengono appoggiati tutti i caratteri ricevuti nei
COM1INP3	728	**R**(182)	messaggi in ricezione prima di essere decodificati
COM1INP4	732	**R**(183)	
COM1INP5	736	**R**(184)	
COM1INP6	740	**R**(185)	
COM1INP7	744	**R**(186)	
COM1INP8	748	**R**(187)	
COM1INP9	752	**R**(188)	
COM1INP10	756	**R**(189)	
COM1INP11	760	**R**(190)	
COM1INP12	764	**R**(191)	
COM1INP13	768	**R**(192)	
COM1INP14	772	**R**(193)	
COM1INP15	776	**R**(194)	
COM1INP16	780	**R**(195)	
COM1INP17	784	**R**(196)	
CLONE	788	**R**(197)	Buffer di 32 byte che contiene una copia dei messaggi del
CLONE1	792	**R**(198)	display lcd.Serve per poter monitorare le scritte sul display via
CLONE2	796	**R**(199)	seriale senza perdere tempo.
CLONE3	800	**R**(200)	
CLONE4	804	**R**(201)	
CLONE5	808	**R**(202)	
CLONE6	812	**R**(203)	
CLONE7	816	**R**(204)	
DISPLAY	820	**R**(205)	Variabili in uso alla visualizzazione del display: • LLB Posizione attuale di scrittura copia display • LHB contatore per visualizzazione pagina stato • HLB Poszione cursore X • HHB Posizione Cursore Y
LASTPAGE	824	**R**(206)	Ultima pagina visualizzata.Contiene il puntatore al descrittore di pagina.Viene usata dal molte **SY**S di visualizzazione
COMOUT0	828	**R**(207)	Variabili necessarie all'interrupt trasmissione seriale COM 0
COMOUT1	832	**R**(208)	Variabili necessarie all'interrupt trasmissione seriale COM 1
COMINP0	836	**R**(209)	Variabili necessarie all'interrupt ricezione seriale COM 0

COMINP1	840	**R(**210**)**	Variabili necessarie all'interrupt ricezione seriale COM 1 Se viene reimpostato l'interrupt seriale il byte LLB contiene il dato ricevuto
MODECOM	844	**R(**211**)**	Modo di trasmissione seriale : • LLB Com 0 se 0 modo binario se 1 modo ascii • LHB Com 1 se0 modo binario se 1 modo ascii
TIMEOUT	848	**R(**212**)**	Contatori time Out Com 0 sia TX (**lw**) che RX(**hw**)
TIMEOUT1	852	**R(**213**)**	Contatori time Out Com 1 sia TX (**lw**) che RX(**hw**)
FLAGTCO	856	**R(**214**)**	Bit presenza componenti nel programma: Istruzioni di impostazione come SetTimer,SetPos impostano la presenza di un componente di tipo appropriato nel programma questo per indicarne il successivo rinfresco,in modo da non attuare funzioni superflue. • Bit 0 presenza Timer a decimi secondo • Bit 1 presenza Timer a secondi • Bit 2 presenza Timer a minuti • Bit 3 presenza Timer a ore • Bit 4 presenza contatori come timer veloci • Bit 5 presenza controlli su orologi • Bit 6 presenza controlli su orologi • Bit 7 presenza interrupt rimappato su com 1 • Bit 8 presenza interrupt utente 1 • Bit 9 presenza interrupt utente 2
TIMIMP0	860	**R(**215**)**	Timer 0 **lw** = Set impostazione **lw** = valore attuale
TIMIMP1	864	**R(**216**)**	Timer 1 **lw** = Set impostazione **lw** = valore attuale
TIMIMP2	868	**R(**217**)**	Timer 2 **lw** = Set impostazione **lw** = valore attuale
TIMIMP3	872	**R(**218**)**	Timer 3 **lw** = Set impostazione **lw** = valore attuale
TIMIMP4	876	**R(**219**)**	Timer 4 **lw** = Set impostazione **lw** = valore attuale
TIMIMP5	880	**R(**220**)**	Timer 5 **lw** = Set impostazione **lw** = valore attuale
TIMIMP6	884	**R(**221**)**	Timer 6 **lw** = Set impostazione **lw** = valore attuale
TIMIMP7	888	**R(**222**)**	Timer 7 **lw** = Set impostazione **lw** = valore attuale
TIMIMP8	892	**R(**223**)**	Timer 8 **lw** = Set impostazione **lw** = valore attuale
TIMIMP9	896	**R(**224**)**	Timer 9 **lw** = Set impostazione **lw** = valore attuale
TIMIMP10	900	**R(**225**)**	Timer 10 **lw** = Set impostazione **lw** = valore attuale
TIMIMP11	904	**R(**226**)**	Timer 11 **lw** = Set impostazione **lw** = valore attuale
TIMIMP12	908	**R(**227**)**	Timer 12 **lw** = Set impostazione **lw** = valore attuale
TIMIMP13	912	**R(**228**)**	Timer 13 **lw** = Set impostazione **lw** = valore attuale
TIMIMP14	916	**R(**229**)**	Timer 14 **lw** = Set impostazione **lw** = valore attuale
TIMIMP15	920	**R(**230**)**	Timer 15 **lw** = Set impostazione **lw** = valore attuale
TIMIMP16	924	**R(**231**)**	Timer 16 **lw** = Set impostazione **lw** = valore attuale
TIMIMP17	928	**R(**232**)**	Timer 17 **lw** = Set impostazione **lw** = valore attuale
TIMIMP18	932	**R(**233**)**	Timer 18 **lw** = Set impostazione **lw** = valore attuale
TIMIMP19	936	**R(**234**)**	Timer 19 **lw** = Set impostazione **lw** = valore attuale
TIMIMP20	940	**R(**235**)**	Timer 20 **lw** = Set impostazione **lw** = valore attuale
TIMIMP21	944	**R(**236**)**	Timer 21 **lw** = Set impostazione **lw** = valore attuale
TIMIMP22	948	**R(**237**)**	Timer 22 **lw** = Set impostazione **lw** = valore attuale
TIMIMP23	952	**R(**238**)**	Timer 23 **lw** = Set impostazione **lw** = valore attuale
TIMMODE0	956	**R(**239**)**	**llb** modo timer 0……**hhb** modo timer 3
TIMMODE1	960	**R(**240**)**	**llb** modo timer 4……**hhb** modo timer 7
TIMMODE2	964	**R(**241**)**	**llb** modo timer 8……**hhb** modo timer 11
TIMMODE3	968	**R(**242**)**	**llb** modo timer 12…..**hhb** modo timer 15
TIMMODE4	972	**R(**243**)**	**llb** modo timer 16…..**hhb** modo timer 19
TIMMODE5	976	**R(**244**)**	**llb** modo timer 20…..**hhb** modo timer 23
CONIMP0	980	**R(**245**)**	Counter 0 **lw** = Set impostazione **lw** = valore attuale
CONIMP1	984	**R(**246**)**	Counter 1 **lw** = Set impostazione **lw** = valore attuale

CONIMP2	988	**R(247)**	Counter 2 **lw** = Set impostazione **lw** = valore attuale
CONIMP3	992	**R(248)**	Counter 3 **lw** = Set impostazione **lw** = valore attuale
CONIMP4	996	**R(249)**	Counter 4 **lw** = Set impostazione **lw** = valore attuale
CONIMP5	1000	**R(250)**	Counter 5 **lw** = Set impostazione **lw** = valore attuale
CONIMP6	1004	**R(251)**	Counter 6 **lw** = Set impostazione **lw** = valore attuale
CONIMP7	1008	**R(252)**	Counter 7 **lw** = Set impostazione **lw** = valore attuale
CONIMP8	1012	**R(253)**	Counter 8 **lw** = Set impostazione **lw** = valore attuale
CONIMP9	1016	**R(254)**	Counter 9 **lw** = Set impostazione **lw** = valore attuale
CONIMP10	1020	**R(255)**	Counter 10 **lw** = Set impostazione **lw** = valore attuale
CONIMP11	1024	**R(256)**	Counter 11 **lw** = Set impostazione **lw** = valore attuale
CONIMP12	1028	**R(257)**	Counter 12 **lw** = Set impostazione **lw** = valore attuale
CONIMP13	1032	**R(258)**	Counter 13 **lw** = Set impostazione **lw** = valore attuale
CONIMP14	1036	**R(259)**	Counter 14 **lw** = Set impostazione **lw** = valore attuale
CONIMP15	1040	**R(260)**	Counter 15 **lw** = Set impostazione **lw** = valore attuale
CONIMP16	1044	**R(261)**	Counter 16 **lw** = Set impostazione **lw** = valore attuale
CONIMP17	1048	**R(262)**	Counter 17 **lw** = Set impostazione **lw** = valore attuale
CONIMP18	1052	**R(263)**	Counter 18 **lw** = Set impostazione **lw** = valore attuale
CONIMP19	1056	**R(264)**	Counter 19 **lw** = Set impostazione **lw** = valore attuale
CONIMP20	1060	**R(265)**	Counter 20 **lw** = Set impostazione **lw** = valore attuale
CONIMP21	1064	**R(266)**	Counter 21 **lw** = Set impostazione **lw** = valore attuale
CONIMP22	1068	**R(267)**	Counter 22 **lw** = Set impostazione **lw** = valore attuale
CONIMP23	1072	**R(268)**	Counter 23 **lw** = Set impostazione **lw** = valore attuale
CONMODE0	1076	**R(269)**	**llb** modo Counter 0……..**hhb** modo Counter 3
CONMODE1	1080	**R(270)**	**llb** modo Counter 4……..**hhb** modo Counter 7
CONMODE2	1084	**R(271)**	**llb** modo Counter 8……..**hhb** modo Counter 11
CONMODE3	1088	**R(272)**	**llb** modo Counter 12……**hhb** modo Counter 15
CONMODE4	1092	**R(273)**	**llb** modo Counter 16……**hhb** modo Counter 19
CONMODE5	1096	**R(274)**	**llb** modo Counter 20……**hhb** modo Counter 23
ENC	1100	**R(275)**	Posizione dell Encoder
HCOUNT0	1104	**R(276)**	Posizione contatore hardware 0 o encoder relativo
HCOUNT1	1108	**R(277)**	Posizione contatore hardware 1
IMPRATE	1112	**R(278)**	RATE REFRESH ENCODER (**lw**) + modo (**hhb)**
ENCPOS	1116	**R(279)**	Posizione assoluta per set posizione 0
ENCPOS1	1120	**R(280)**	Posizione assoluta per set posizione 1
ENCPOS2	1124	**R(281)**	Posizione assoluta per set posizione 2
ENCPOS3	1128	**R(282)**	Posizione assoluta per set posizione 3
ENCPOS4	1132	**R(283)**	Posizione assoluta per set posizione 4
ENCPOS5	1136	**R(284)**	Posizione assoluta per set posizione 5
ENCPOS6	1140	**R(285)**	Posizione assoluta per set posizione 6
ENCPOS7	1144	**R(286)**	Posizione assoluta per set posizione 7
ORIMP0	1148	**R(287)**	OROLOGIO 0 IMPOSTAZIONI
ORIMP1	1152	**R(288)**	OROLOGIO 1 IMPOSTAZIONI
ORIMP2	1156	**R(289)**	OROLOGIO 2 IMPOSTAZIONI
ORIMP3	1160	**R(290)**	OROLOGIO 3 IMPOSTAZIONI
ORIMP4	1164	**R(291)**	OROLOGIO 4 IMPOSTAZIONI
ORIMP5	1168	**R(292)**	OROLOGIO 5 IMPOSTAZIONI
ORIMP6	1172	**R(293)**	OROLOGIO 6 IMPOSTAZIONI
ORIMP7	1176	**R(294)**	OROLOGIO 7 IMPOSTAZIONI
ORMODE	1180	**R(295)**	OROLOGIO IMPOSTAZIONE MODO 0..3
ORMODE1	1184	**R(296)**	OROLOGIO IMPOSTAZIONE MODO 4..7
SVRCOM1	1188	**R(297)**	Interrupt utente su seriale COM 1
SVRTIM0	1192	**R(298)**	Interrupt utente 0
SVRTIM1	1196	**R(299)**	Interrupt utente 1

I registri da R(300) a r(900) sono a disposizione del programmatore .Tutti i registri sono tamponati contro la mancanza di tensione e i dati vengono conservati.

- **Reti Master – Slave**

Il ***microPLC Merlino*** permette la gestione di reti Master-Slave via seriale.
Nel sistema operativo sono stati implementati tutti i messaggi necessari alla gestione di tutte le periferiche della serie ***microPLC*** ovvero :

Exp 8IO
D1208
MERLINO

Grazie alla presenza a bordo del PLC di due porte di comunicazione seriali e alla gestione di diversi tipi di apparecchiature periferiche si possono avere reti master slave più complesse rispetto a quelle che si potevano implementare con il Il ***microPLC D1208.***Rispetto a questa apparente complessità va subito puntualizzato che proprio grazie alla presenza di due porte di comunicazione seriali con il ***microPLC Merlino*** si può monitorare una rete Master-Slave comunque complessa attraverso la porta COM 1 e un computer supervisore.
Sempre grazie alle due porte di comunicazione seriale si può impostare un ***microPLC Merlino*** come master dal lato COM 0 e come slave dal lato COM 1.In questo modo si possono implementare reti particolarmente complesse .
Per eseguire correttamente una connessione Master – Slave bisogna seguire una determinata procedura sia hardware che software.

- **Collegamenti tra Master e Slave**

Dal punto di vista hardware una connessione tra master e slave viene effettuata collegando fisicamente le macchine in modo che :

1	Tutti gli apparati slave risultino in parallelo
2	I segnali di trasmissione di un master (TX+,TX-) siano connessi ai segnali di ricezione (RX+,RX-) di tutti gli slave. I segnali di ricezione di un master (RX+,RX-) siano connessi ai segnali di trasmissione (TX+,TX-) di tutti gli slave

Dal punto di vista hardware non sono necessari ulteriori fasi se non il controllo dei doppini di collegamento.**Per collegamenti a lunga distanza (>100 m) si rende necessario l'uso di apparati per il controllo dei transienti** .Controllare che il cavo usato per il collegamento sia sempre dello stesso tipo.Generalmente un cavo multipolare twistato telefonico è più che adeguato allo scopo.

- **Software tra Master e Slave**

Dal punto di vista software una connessione tra master e slave viene effettuata con una serie di istruzioni di inizilizzazione che informano il sistema che il PLC ha una rete di slave.Mentre se il PLC è uno slave non vi è bisogno di nessuna aggiunta software.
Procedura impostazione slave:

1	Fase di inizializzazione (prima di istruzione start)
2	Controllo e impostazione Velocita di comunicazione Seriale: Per default la Comunicazione della COM 0 è impostata a57600 baud che è la massima vellocità ammessa per il ***microPLC D1208*** e la ***EXP8IO*** .Se si vuole impostare una velocita diversa usare la sys Com0 (24).
3	Impostare sulle memorie CONF1..5 il valore che identifica l'apparato connesso alla rete.Alla memoria CONF1 corrisponde il PLC che ha numero 1, Alla memoria CONF2 corrisponde il PLC che ha numero 2 etc.
4	Inizio programma (istruzione start)

I messaggi scambiati tra master e slave variano a seconda del tipo di plc presente nella rete:

CONFn	
0	Nessuno comunicazione con l'apparato n
1	Il modulo slave è una EXP8IO,sono scritte 8 uscite e letti 8 ingressi .Le uscite sono impostate dal byte LLB della memoria OUT1(2..5) gli ingressi sono copiati nel byte LLB INP1(2..5) il messaggio scambiato tra il merlino e EXP8IO e WR_O_RD_I .
2	Il modulo slave è un PLC D1208 usato come EXP8IO,sono scritte 8 uscite e letti 8 ingressi .Le uscite sono impostate dal byte LLB della memoria OUT1(2..5) gli ingressi sono copiati nel byte LLB INP1(2..5) Il messaggio scambiato tra il merlino e D1208 e WR_O_RD_I .
3	Il modulo slave è un PLC D1208 sono scritte 8 uscite e ritorna lo Stato degli 8 timer interni il PLC .Le uscite sono impostate dal byte LLB della memoria OUT1(2..5) lo stato dei timer viene copiato nel byte LLB della memoria TIMER1(2…5) Il messaggio scambiato tra il merlino e D1208 e WR_O_RD_T.
4	Il modulo slave è un PLC D1208 sono scritte 8 uscite e ritorna lo Stato degli 8 contatori interni il PLC .Le uscite sono impostate dal byte LLB della memoria OUT1(2..5) lo stato dei contatori viene copiato nel byte LLB della memoria COUNT1(2…5) Il messaggio scambiato tra il merlino e D1208 e WR_O_RD_C.
5	Il modulo slave è un PLC D1208 sono scritti 8 rele interni e ritorna lo stato degli 8 ingressi il PLC .Lo stato dei rele da scrivere è impostato dal byte LLB della memoria RSO1(2..5) lo stato degli ingressi viene copiato nel byte LLB della memoria INPUT1(2…5) Il messaggio scambiato tra il merlino e D1208 e WR_S_RD_I .
6	Il modulo slave è un PLC D1208 sono scritti 8 rele interni e ritorna lo stato degli 8 timer del PLC .Lo stato dei rele da scrivere è impostato dal byte LLB della memoria RSO1(2..5) lo stato dei timer viene copiato nel byte LLB della memoria TIMER1(2…5) Il messaggio scambiato tra il merlino e D1208 e WR_S_RD_T .
7	Il modulo slave è un PLC D1208 sono scritti 8 rele interni e ritorna lo stato degli 8 Contatori del PLC .Lo stato dei rele da scrivere è impostato dal byte LLB della memoria RSO1(2..5) lo stato dei Contatori viene copiato nel byte LLB della memoria COUNT1(2…5) Il messaggio scambiato tra il merlino e D1208 e WR_S_RD_C .

	Il modulo slave è un PLC D1208 sono scritti 8 rele interni e ritorna lo stato di 8 rele interni del PLC .Lo stato dei rele da scrivere è impostato dal byte LLB della memoria RSO1(2..5) lo stato dei rele letti viene copiato nel byte LLB della memoria RSI1(2…5) Il messaggio scambiato tra il merlino e D1208 e WR_S_RD_S .
9	Il modulo slave è un PLC MERLINO (modo intracom). Sono passati messaggi a pacchetti di 16 byte.I byte di invio a Merlino Slave iniziano alla memoria EXCHO1S1(2…5).I byte ricevuti da merlino Slave sono copiati a partire dalla memoria EXCHI1S1(2..5). E' lasciato al programmatore l'arbitraggio dei messaggi ,suggerimento il primo byte è un contatore che identifica il messaggio ,quando nei dati di ingresso ritorna lo stesso contatore si può cambiare messaggio. Il messaggio scambiato tra il merlino e merlino e WR_S_RD_S .Controllare nella Comunicazione seriale il funzionamento di questo messaggio per lato Master o lato slave

- **Funzioni di Sistema**

Il programmatore ha a disposizione , attraverso l'istruzione ***SYS*** collegamenti alle funzioni del sistema operativo o funzioni dedicate .
Queste funzioni sono ottimizzate per essere eseguite alla massima velocità permessa dal processore agevolando lo svolgimento di programmi anche complessi.
I parametri alle funzioni vengono passsati attaverso registri in modo diretto ,si possono passare fino a quattro parametri in sequenza,il formato dei parametri deve essere ***long***.Ad esempio se la funzione Func richiede tre parametri bisogna nel programma impostare le seguenti linee di codice:

- Mov.l #parametro1,r(300)
- Mov.l #parametro2,r(301)
- Mov.l #parametro3,r(302)
- Sys #***Func***,r(300)

In base al tipo di funzione si può avere un risultato oppure no.Quando vi è un risultato questo viene memorizzato nella memoria **r(0).**

Nella seguente listaè presente l'elenco delle funzioni,per la rappresentazione delle quali viene presa in considerazione una sintassi Ansi C.

- **void *WaitDelay*(register int *tick*)**

Offset	0
Parametro	Numero in 50simi di secondo di attesa
Return	No

Questa funzione congela lo svolgimento del programma per un tempo prestabilito dalla formula Ta=20ms*tick

- **void *CtrlDsp*(register char *data*,register char *control*)**

Offset	1
Parametro 1	Dato da inviare direttamente al display LCD
Parametro 2	0=dato 1=Comando
Return	No

Questa funzione scrive direttamente nel display un carattere che può essere o un dato di controllo o un carattere da scrivere.Il display LCD è un modello standard LM016 con controller HD44780, vedere nei data sheet di questo prodotto tutti i comandi disponibili.

- **void *PutChar*(register char *data*)**

Offset	2
Parametro	Dato da scrivere sul display
Return	No

Questa funzione scrive il carattere ***data*** nel display LCD alla posizione corrente.Imposta inoltre una copia per potere leggere via seriale la scritta corrente del display LCD.

- **void *ClearLcd*(void)**

Offset	3
Parametro	No
Return	No

Questa funzione scancella tutto il contenuto del display LCD e della copia per la lettura seriale.Inoltre imposta la posizione del Cursore alla posizione XY 0,0 e il cursore in modo nascosto.

- **void *PosXY*(register char *X*,register char *Y*)**

Offset	4
Parametro 1	Posizione X 0..15
Parametro 2	Posizione Y 0...2
Return	No

Questa funzione posiziona il cursore alla colonna X della riga Y sul display LCD

- **void *WrMsg*(register char **ptr*)**

Offset	5
Parametro 1	Puntatore alla stringa da visualizzare,deve terminare con 0
Return	No

Scrive , partendo dalla posizione corrente la stringa puntata dal parametro **ptr**.Nel linguaggio assembler un puntatore è rappresentato con l'indirizzo della memoria dove si trova l'oggetto puntato.Ad esempio per la funzione ***WrMsg*** si può avere il seguente frammento di codice:

```
        mov.l   #stringa,r(300)
        sys     #5,r(300)
        .
        .
stringa:
    DM  "Salve Mondo",0
```

In questo caso su R(300) viene immesso l'indirizzo fisico della memoria ove risiede il messaggio da visualizzare (ad indirizzo **stringa)**

- **void *MemMsg*(register char **ptr*,register char *num*)**

Offset	6
Parametro 1	Puntatore alla stringa da visualizzare
Parametro 2	Numero dei caratteri da scrivere
Return	No

Scrive , partendo dalla posizione corrente la stringa puntata dal parametro **ptr**,per una lunghezza in caratteri data dal dato **num**.

- **void *WrVal*(register long *valore*,register char *formato*)**

Offset	7
Parametro 1	Valore da visualizzare
Parametro 2	Formato con cui viene visualizzato sul display
Return	No

Scrive , partendo dalla posizione corrente il dato **valore** nel modo stabilito dal parametro **formato** .
I formati possibili sono i seguenti:

formato	**Risultato**	
16	0..9	1 carattere
17	0..99	2 caratteri
18	0..999	3 caratteri
19	0..9999	4 caratteri
20	0..99999	5 caratteri
21	0..999999	6 caratteri
22	0..9999999	7 caratteri
23	0..99999999	8 caratteri
24	+0..999999999	9 caratteri con segno
25	+0..9999999999	10 caratteri con segno
32	99.9 (valore=999)	3 caratteri con punto decimale
33	99.99 (valore=9999)	4 caratteri con punto decimale
34	999.9 (valore=9999)	4 caratteri con punto decimale
35	999.99 (valore=99999)	5 caratteri con punto decimale
48	1111111111111111	16 bit modo binario
49	11111111	8 bit modo binario
50	1111	4 bit modo binario
51	□□□□□□□□□□□□□□□□	16 bit modo casella
52	□□□□□□□□	8 bit modo casella
53	FF	8 bit modo esadecimale
54	FFFF	16 bit modo esadecimale
64	Impostazione vel comunicazione seriale (valore 0..8)	

65	0..255	Se è un tasto nome o num
80	0..1	Modo a bit 0-1
81	No..Si	Modo a bit No-Si
82	Off..On	Modo a bit Off-On
83	Dis…En	Modo a bit Dis-En.
96	0..7	Giorni della settimana
97	0..9	Modo di configurazione
98	0..4	Modo conf. Timer
99	0..4	Modo conf. Counter
100	0..6	Modo conf. Orologi
101	0..5	Modo stato plc

- **void *WrBit*(register char *val*,register char *formato*, register char *mask*)**

Offset	8
Parametro 1	Valore da visualizzare
Parametro 2	Formato con cui viene visualizzato sul display
Parametro 3	Maschera per And con cui confrontare il bit desiderato
Return	No

Scrive , partendo dalla posizione corrente ,nel **formato** specificato il risultato dell'operazione val and mask.

Formato	**Risultato**	
80	0..1	Modo a bit 0-1
81	No..Si	Modo a bit No-Si
82	Off..On	Modo a bit Off-On
83	Dis…En	Modo a bit Dis-En.

- **void *ConfDisplay*(void)**

Offset	9
Parametro 1	No
Return	No

Reimposta il display cancellandone il contenuto ,riprogramma i primi 8 caratteri e imposta il formato 2 righe 16 caratteri.

- **void *FillDsp*(register void **ptr*)**

Offset	10
Parametro 1	Puntatore alle stringe da visualizzare
Parametro 2	Numero dei caratteri da scrivere
Return	No

Scrive due messaggi uno per ogni riga del display.Il puntatore **ptr** punta ad un array di puntatori ai messaggi .Esempio:

```
                mov.l   #arrayptr,r(300)
                sys     #10,r(300)
                .
                .
 arrayptr:
DL      Stringa1
DL      Stringa2
Stringa1:
DM "1°riga   utente "
Stringa1:
DM "2°riga   utente "
```

- **void *VisDati*(register void **ptr*)**

Offset	11
Parametro 1	Puntatore ai dati di una pagina LCD
Return	No

Rinfresca i valori delle variabili di una pagina.Non scrive le stringhe della pagine.Vedi la funzione VisPagine più avanti

- **void *VisPagInd*(register int *indice*)**

Offset	12
Parametro 1	Indica la pagina di memoria da visualizzare
Return	No

Scrive sul display LCD la pagina di memoria corrispondente all'indice.Sono disponibili 96 pagine differenti già preimpostate e sono quelle accessibili quando il PLC è in stop.

0	Visualizzazione Input 0..15
1	Visualizzazione Input 16..31
2	Visualizzazione Output 0..15
3	Visualizzazione Output 16..31
4	Visualizzazione Comunicazione
5	Visualizzazione Plc Slave 1
6	Visualizzazione Plc Slave 2
7	Visualizzazione Plc Slave 3
8	Visualizzazione Plc Slave 4
9	Visualizzazione Plc Slave 5
10	Visualizzazione Timer 1
11	Visualizzazione Timer 2
12	Visualizzazione Timer 3
13	Visualizzazione Timer 4
14	Visualizzazione Timer 5
15	Visualizzazione Timer 6
16	Visualizzazione Timer 7
17	Visualizzazione Timer 8
18	Visualizzazione Timer 9
19	Visualizzazione Timer 10
20	Visualizzazione Timer 11
21	Visualizzazione Timer 12
22	Visualizzazione Timer 13
23	Visualizzazione Timer 14
24	Visualizzazione Timer 15

25	Visualizzazione Timer	16
26	Visualizzazione Timer	17
27	Visualizzazione Timer	18
28	Visualizzazione Timer	19
29	Visualizzazione Timer	20
30	Visualizzazione Timer	21
31	Visualizzazione Timer	22
32	Visualizzazione Timer	23
33	Visualizzazione Timer	24
34	Visualizzazione Contatori	1
35	Visualizzazione Contatori	2
36	Visualizzazione Contatori	3
37	Visualizzazione Contatori	4
38	Visualizzazione Contatori	5
39	Visualizzazione Contatori	6
40	Visualizzazione Contatori	7
41	Visualizzazione Contatori	8
42	Visualizzazione Contatori	9
43	Visualizzazione Contatori	10
44	Visualizzazione Contatori	11
45	Visualizzazione Contatori	12
46	Visualizzazione Contatori	13
47	Visualizzazione Contatori	14
48	Visualizzazione Contatori	15
49	Visualizzazione Contatori	16
50	Visualizzazione Contatori	17
51	Visualizzazione Contatori	18
52	Visualizzazione Contatori	19
53	Visualizzazione Contatori	20
54	Visualizzazione Contatori	21
55	Visualizzazione Contatori	22
56	Visualizzazione Contatori	23
57	Visualizzazione Contatori	24
58	Visualizzazione Ani 1..2	
59	Visualizzazione Ani 3..4	
60	Visualizzazione Ani 5..6	
61	Visualizzazione Ani 7..8	
62	Visualizzazione Dac 1..2	
63	Visualizzazione Encoder	
64	Visualizzazione Contatore Hard.1 ,	
65	Visualizzazione Contatore Hard.2 ,	
66	Visualizzazione Posizione	1
67	Visualizzazione Posizione	2
68	Visualizzazione Posizione	3
69	Visualizzazione Posizione	4
70	Visualizzazione Posizione	5
71	Visualizzazione Posizione	6
72	Visualizzazione Posizione	7
73	Visualizzazione Posizione	8
74	Visualizzazione Ora Sistema	
75	Visualizzazione Data Sistema	
76	Visualizzazione Orologio	1
77	Visualizzazione Orologio	2
78	Visualizzazione Orologio	3
79	Visualizzazione Orologio	4
80	Visualizzazione Orologio	5

81	Visualizzazione Orologio 6
82	Visualizzazione Orologio 7
83	Visualizzazione Orologio 8
84	Visualizzazione Numero Plc
85	Visualizzazione Errori Plc 1
86	Visualizzazione Errori Plc 2
87	Visualizzazione Stato Plc
88	Visualizzazione Rele blocco 1
89	Visualizzazione Rele blocco 2
90	Visualizzazione Rele blocco 3
91	Visualizzazione Rele blocco 4
92	Visualizzazione Rele blocco 5
93	Visualizzazione Rele blocco 6
94	Visualizzazione Rele blocco 7
95	Visualizzazione Rele blocco 8
96	Visualizzazione Stato

- **void *VisPagina*(register Pagina *ptr)**

Offset	13
Parametro 1	Puntatore alla struttura descrittiva la pagina da visulaizzare
Return	No

Scrive sul display LCD la pagina puntata dal puntatore **ptr**.Imposta una copia del puntatore alla pagina sul registro r(206).Una pagina è definita da due puntatori ai messagggi per le righe 1 e 2 e da un array di puntatori ai valori delle variabili da visualizzare.Esempio

```
                mov.l   #miapagina,r(300)
                sys     #13,r(300)
                .
                .
miapagina:
DL   Msg1               ;puntatore al messaggio della 1° riga
DL   Msg2               ;puntatore al messaggio della 2° riga
DL   valore1            ;puntatore alla struttura valore1
DL   valore2            ;puntatore alla struttura valore2
DL   0                  ;puntatore nullo indica che non ci sono più variabili da visualizzare

Msg1:
DM  "R(300)=        "
Msg2:
DM  "R(301)=        "
```

L' array deipuntatori ai valori deve terminare con un puntatore nullo DL 0.
I puntatori ai valori indicano l'indirizzo della struttura descrittiva la variabile da visualizzare e sono cosi definiti:

```
Valore1:
DL   r(300)             ;(indirizzo variabile) indirizzo della variabile da visualizzare
DL   limit1             ;(limite della variabile) tipo di visualizzane e trattamento del puntatore
Valore2:
DL   r(301)             ;indirizzo della variabile da visualizzare
DL   limit1             ;tipo di visualizzane e trattamento del puntatore
```

I descrittori della variabile sono strutture che indicano il formato di visualizzazione,come trattare il puntatore al dato il minimo e il massimo permesso alla variabile se si tratta di una variabile che può essere modificata e sono cos'ì definiti:

```
limit1:
DB   16             ;(formato) formato di visualizzazione come sys WrVal
DB   0              ;(tipo ptr) PTR come tratto il puntatore al dato vedi tabella
DB   5              ;(pos X) colonna dove viene visualizzato il dato
DB   1              ;(pos Y) riga dove viene visualizzato il dato
DL   0              ;(minimo) valore minimo ammesso per questa variabile
DL   1000           ;(massimo)valore massimo ammesso per questa variabile
```

Il secondo parametro della struttura (**tipo ptr**)può assumere i seguenti valori:

0…3	Il puntatore viene trattato come puntatore a carattere a 8 bit
4..5	Il puntatore viene trattato come puntatore a intero a 16 bit
6	Il puntatore viene trattato come puntatore a long a 32 bit
7	Il puntatore è un puntatore a un carattere a 8 bit ,il parametro minimo indica il valore per attuare la maschera and che da il risultato per la visualizzazione
128	Sommando ai parametri precedenti il valore 128 ogni variabile mantiene lo stesso formato di visualizzazione ma non è modificabile .(**lock**)

Questo tipo di implementazione può a prima vista sembrare complessa ma questo modo permette di avere un descrittore completo della pagina anche quando si usa la routine di rinfresco o di aggiornamento della pagina stessa senza dovere aggiungere dati.Inoltre implementando limiti separati dalle variabili si può usare la stessa struttura per più variabili dello stesso tipo.

- **void *UserInputVal*(register int indice)**

Offset	14	
Parametro 1	Indice che indica la memoria che sarà utilizzata da questa funzione	
Return	Si	

Questa funzione permette di modificare le variabili della pagina.Se una variabile non è modificabile viene saltata durante la scansione delle variabili.I tasti presi in considerazione dalla funzione sono **up,down, esc ,enter**.La funzione assume che nel registro 206 vi sia un indirizzo a una pagina valida e tratta le variabili della pagina con i questi descrittori di variabile.Ad ogni scansione di programma questa funzione controlla il tasto premuto e agisce di conseguenza.Il parametro in ingresso definisce la variabile da usare per mantenere memoria dello stato.Si possono definire due parametri entro questa variabile:

HLB	0..255	Per quante scansioni di programma il tasto deve essere premuto prima di essere preso in considerazione
LLB	Xx	Variabile che viene presa in considerazione dalla funzione

Si possono avere i seguenti codi di ritorno:

256	Il registro 206 non contiene un indirizzo di pagina valido
512	Il registro indice di ingresso non punta un registro valido (>900)
1024	L'indice della variabile sotto modifica è maggiore delle variabili presenti nella pagina
2048	Il puntatore della variabile sotto esame è nullo
1	Premuto tasto UP variabile aggiornata OK
3	Premuto tasto ESC+UP variabile aggiornata OK
2	Premuto tasto DW variabile aggiornata OK
4	Premuto tasto ESC+DW variabile aggiornata OK
8	Premuto ESC rilascio modifica
16	Premuto ENTER fine modifica

- **void *LastPag*(void)**

Offset	15
Parametro 1	No
Return	No

Questa funzione ridisegna completamente l'ultima pagina visualizzata.Il registro r(206) deve avere un puntatore valido alla pagina da visualizzare

- **void *RefreshPag*(void)**

Offset	16
Parametro 1	No
Return	No

Questa funzione rinfresca le variabili visualizzate nella pagina puntata dal registro r(206)

- **void *TestKey*(void)**

Offset	17
Parametro 1	No
Return	No

Controlla lo stato della tastiera .Se un tasto è premuto o comunque il valore degli ingressi dei tasti è diverso da 255 compare un messaggio di tasto premuto.Questo test viene effettuato alla partenza del PLC.Nel caso il PLC sia entrato in modo visualizzazione messaggio per uscirte premere UP+DW+ESC+ENTER= RESET

- **void *TestRam*(void)**

Offset	18
Parametro 1	No
Return	No

Esegue il test della RAM .Se una memoria da un errrore viene visualizzato un messaggio corrispondente

- **void *SysShiftLeft*(register int *mem*,register int *limit*,register int *val*)**

Offset	19
Parametro 1	Offset della memoria su cui eseguire l'operazione
Parametro 2	Numero massimo dello shift
Parametro 3	Numero di shift da effettuare
Return	No

Serve essenzalmente per lo shift di un singolo bit su una memoria eseguendo nello stesso tempo il controllo per non perdere il valore .Ad esempio se limit e 9 e mem>512 mem=512 mentre se limit =8 mem=64 e val=3 dopo l'esecuzione della funzione mem=256.Permette di implementare rele che si attivano in sequenza facendo nello stesso tempo il conmtrollo massimo conteggio (contatore Jonhson in avanti)

- **void *SysShiftRight*(register int *mem*,register int *limit*,register int *val*)**

Offset	20
Parametro 1	Offset della memoria su cui eseguire l'operazione
Parametro 2	Numero massimo dello shift
Parametro 3	Numero di shift da effettuare
Return	No

Serve essenzalmente per lo shift di un singolo bit su una memoria eseguendo nello stesso tempo il controllo per non perdere il valore .Ad esempio se limit e 2 e mem<4 mem=4 mentre se limit =8 mem=512 e val=3 dopo l'esecuzione della funzione mem=256.Permette di implementare rele che si attivano in sequenza facendo nello stesso tempo il conmtrollo massimo conteggio (contatore Jonhson indietro)

- **void *MemCpy*(register char **dest*,register char* *srg*,register int *len*)**

Offset	21
Parametro 1	Puntatore indirizzo destinazione
Parametro 2	Puntatore inidirizzo sorgente
Parametro 3	Lunghezza in byte da copiare
Return	No

Copia per la lunghezza in byte **len** i dati da **srg** a **dest**.Se si vuole copiare dei registri bisogna ricordare che un registro ha lunghezza 4.

- **void *MemSet*(register char **dest*,register char *val*,register int *len*)**

Offset	22
Parametro 1	Puntatore indirizzo destinazione
Parametro 2	Valore a cui impostare la memoria
Parametro 3	Lunghezza in byte da copiare
Return	No

Imposta per la lunghezza in byte **len** i dati di **dest** al valore **val**.

- **void *InitComSerial*(int *porta*,int *data*,int *stop*,long *vel*,char *parity*)**

Offset	23
Parametro 1	Porta seriale da impostare 0,1
Parametro 2	Dati impostazione 7,8

Parametro 3	Bit di stop 0,1,2
Parametro 4	Velocita (600,1200,2400,4800,9600,19200,28800,38400,57600,115200)
Parametro 5	Parità N,E,O
Return	No

Imposta la porta di comunicazione specificata nel modo voluto

- **void *Com0*(register char *vel*)**

Offset	24
Parametro 1	Velocita di comunicazione
Return	No

Imposta la porta di comunicazione **0** alla velocita specificata da **vel** (0 .. 8) 8 bit dati 1 stop nessuna parità .Il parametro vel ha il seguente significato:

0	115200 baud
1	600 baud
2	1200 baud
3	2400 baud
4	4800 baud
5	9600 baud
6	19200 baud
7	38400 baud
8	57600 baud

- **void *Com1*(register char *vel*)**

Offset	25
Parametro 1	Velocita di comunicazione
Return	No

Imposta la porta di comunicazione **1** alla velocita specificata da **vel** (0 .. 8) 8 bit dati 1 stop nessuna parità .Il parametro vel ha il seguente significato:

0	115200 baud
1	600 baud
2	1200 baud
3	2400 baud
4	4800 baud
5	9600 baud
6	19200 baud
7	38400 baud
8	57600 baud

- **void *WaitCom*(register int *porta*)**

Offset	26
Parametro 1	Porta di comunicazione
Return	No

Controlla se per la porta specificata vi sono messaggi in uscita.Se Vi sono messaggi aspetta fino a quando il messaggio non è stato spedito in modo da non fare sovrapporre due messaggi consegutivi

- **void *WriteCom*(register char *porta*,register char *num*,register char modo)**

Offset	27
Parametro 1	Porta di comunicazione

Parametro 2	Numero dei caratteri da spedire
Parametro 3	Modo di comunicazione 0=binario 1=utente
Return	No

Imposta il processore per effettuare la trasmissione di un messaggio.Se mode è 0 la comunicazione è in formato standard e il numero di caratteri da spedire è nel messaggio altrimenti prende in considerazione il parametro num.La funzione non spedisce il messaggio ma imposta e genera gli interrupt necessari a per spedirlo.

- **void *InitCode*(void)**

Offset	28
Parametro 1	No
Return	No

Esegue un reset Software del PLC

- **void *UserReadAll*(register int porta)**

Offset	29
Parametro 1	Descrittore porta comunicazione
Return	No

Imposta e spedisce un messaggio lettura stato PLC.Vedere al capitolo comunicazione seriale comando READALL.

- **void *UserReadAni*(register int porta)**

Offset	30
Parametro 1	Descrittore porta comunicazione
Return	No

Imposta e spedisce un messaggio lettura stato PLC.Vedere al capitolo comunicazione seriale comando READANI.

- **void *UserReadAux*(register int porta)**

Offset	31
Parametro 1	Descrittore porta comunicazione
Return	No

Imposta e spedisce un messaggio lettura stato PLC.Vedere al capitolo comunicazione seriale comando READAUX.

- **void *ReadHardInput*(void)**

Offset	32
Parametro 1	No
Return	No

Legge lo stato degli ingressi fisici e lo copia nella variabile di sistema INPUT

- **void *CopyOutput*(void)**

Offset	33
Parametro 1	No
Return	No

Se il registro di abilitazione delle uscite è diverso da zero copia il registro delle uscite nelle uscite fisiche.

- **void *UserAddVal*(register long *inc*, register long *min*, register long *max*)**

Offset	34
Parametro 1	Quantita da sommare al registro INPUT VAL
Parametro 2	Limite minimo
Parametro 3	Limite massimo
Return	No

Quando si immette un nuovo valore viene usata questa funzione che incrementa della quantita ***inc*** il registro dell'ultimo valore immesso.Viene eseguito un controllo perché la variabile sia nei range ***min e max*** altrimenti viene reimpostata al limite più vicino

- **void *SysHome*(void)**

Offset	35
Parametro 1	No
Return	No

Esegue il comando HOME per LCD.La posizione del cursore è 0,0 il cursore è nascosto

- **void *SysCurLeft*(void)**

Offset	36
Parametro 1	No
Return	No

Sposta il cursore a sinistra di una posizione.Non modifica lo stato di visualizzazione del cursore

- **void *SysCurRight*(void)**

Offset	37
Parametro 1	No
Return	No

Sposta il cursore a destra di una posizione.Non modifica lo stato di visualizzazione del cursore

- **void *SysLeft*(void)**

Offset	38
Parametro 1	No
Return	No

Sposta la riga corrente di una posizione facendo uno shift left di tutta la riga

- **void *SysRight*(void)**

Offset	39
Parametro 1	No
Return	No

Sposta la riga corrente di una posizione facendo uno shift Right di tutta la riga

- **void *SysOffCur*(void)**

Offset	40
Parametro 1	No
Return	No

Imposta lo stato di visualizzazione del cursore come nascosto

- **void *SysOnCur*(void)**

Offset 41
Parametro 1 No
Return No
Imposta lo stato di visualizzazione del cursore come attivo

- **void *SetUserSvrCom1*(register long *addr*)**

Offset 42
Parametro 1 Indirizzo della routine utente da eseguire come interrupt seriale COM 1
Return No
Imposta la routine descritta dall'indirizzo addr come routine di interrupt seriale per la porta di comunicazione numero 1.Ad esempio:

```
        Mov.l   #mysvrcom,r(300)
        Sys     #42,r(300)
        .
        .
mysvrcom:                       ;questa è una routine di interrupt che viene eseguita
        abscplb     r(20).0     ;ogni volta che la COM1 riceve un carattere
        reti
```

Quando viene settato questo interrupt la ricezione standard viene esclusa e il carattere ricevuto è immagazzinato nel byte LLB del registro r(210) COMINP1

- **void *SetUserSvrTimer1*(register long *addr* ,register long *tick*)**

Offset 43
Parametro 1 Indirizzo della routine utente da eseguire come interrupt timer 1
Parametro 2 Parametro di impostazione interrupt
Return No
Imposta la routine descritta dall'indirizzo addr come routine di interrupt timer canale 1 al rate descritto dal parametro ***tick***.Ad esempio:

```
        Mov.l   #mysvrtim,r(300)
        Mov.l   #10000,r(301)
        Sys     #43,r(300)
        .
        .
mysvrtim:                       ;questa è una routine di interrupt
        abscplb     r(20).0
        reti
```

Il rate descritto dalla variabile ***tick:***
Se tick è minore di 7353 viene ignorato e impostato a 7353 .
Se tick è maggiore di 65535 viene ignorato e impostato a 65535.
Il rate è calcolato come inpulsi di clock ovvero impostando a 50000 si ha un rate pari a 50000*(1/14757000) in secondi.

- **void *SetUserSvrTimer2*(register long *addr* ,register long *tick*)**

Offset 44
Parametro 1 Indirizzo della routine utente da eseguire come interrupt timer 1
Parametro 2 Parametro di impostazione interrupt
Return No

Imposta la routine descritta dall'indirizzo addr come routine di interrupt timer canale 2 al rate descritto dal parametro ***tick***.Ad esempio:

```
        Mov.l   #mysvrtim2,r(300)
        Mov.l   #10000,r(301)
        Sys     #44,r(300)
        .
        .
mysvrtim2:                          ;questa è una routine di interrupt
        abscplb     r(20).0
        reti
```

Il rate descritto dalla variabile ***tick:***

Se tick è minore di 7353 viene ignorat o e impostato a 7353

Se tick è maggiore di 65535 viene ignorato e impostato a 65535.

Il rate è calcolato come inpulsi di clock ovvero impostando a 50000 si ha un rate pari a 50000*(1/14757000) in secondi.

- **void *TxChar*(register char *c*)**

Offset	45
Parametro 1	Carattere da trasmettere immediatamente
Return	No

Trasmette dalla porta di comunicazione 1 il carattere c.Non viene usato l'interrupt di trasmissione il carattere viene trasmesso immediatamente e il programma viene congelato fino a che il dato non è completamente trasmesso.Applicazione tipica : Eco dei caratteri ricevuti per routine di interrupt seriale

- **void *TxStr*(void)**

Offset	46
Parametro 1	No
Return	No

Trasmette dalla porta di comunicazione 1 la stringa immagazzinata nel buffer di uscita seriale 1

- **Float *ToFloat*(register long *t*)**

Offset	47
Parametro 1	Dato da covertire in floatting point
Return	Si

Converte il dato in ingresso nel formato floatting point.Ad esempio:

```
mov.l   #123456,r(300)   ;r300 =long
sys     #47,r(300)       ;tofloat
mov.l   r(0),r(300)      ;r300= rappresentazione floatting point del numero 123456
```

- **long *FloatTolong*(register float *t*)**

Offset	48
Parametro 1	Dato da covertire in long
Return	Si

Converte il dato in ingresso nel formato long.Bisogna puntualizzare che ogni registro è adatto a contenere un dato floatting point (16 bit esponente + 16 bit mantissa) per cui il parametro in ingresso è un qualsiasi registro .Ad esempio

```
mov.l   #0f123.456,r(300)    ;r300 =float
sys     #48,r(300)           ;floatTolong
mov.l   r(0),r(300)          ;r300= 123 long
```

- **Int *FloatToInt*(register float *t*)**

Offset	49
Parametro 1	Dato da covertire in int (16 bit)
Return	Si

Converte il dato in ingresso nel formato intero a 16 bit (LOW WORD)

- **Char *FloatToChar*(register float *t*)**

Offset	50
Parametro 1	Dato da covertire in int (8 bit)
Return	Si

Converte il dato in ingresso nel formato intero a 8 bit (LOW LOW BYTE)

- **float *AddFloat*(register float *a*, register float *b*)**

Offset	51
Parametro 1	Dato
Parametro 2	Dato
Return	Si

Esegue la somma a+b in modo float .I registri passati alla funzione devono contenere dati in formato float.Ad esempio:

```
mov.l   #0f123456.12334,r(300)
mov.l   #0f0.0002345,r(301)
sys     #51,r(300)
risultato in r(0) in forma floating point
```

- **float *SubcFloat*(register float *a*, register float *b*)**

Offset	52
Parametro 1	Dato
Parametro 2	Dato
Return	Si

Esegue la differenza a-b in modo float

- **float *DivFloat*(register float *a*, register float *b*)**

Offset	53
Parametro 1	Dato
Parametro 2	Dato
Return	Si

Esegue la divisione a/b in modo float

- **float *MultFloat*(register float *a*, register float *b*)**

Offset	54
Parametro 1	Dato
Parametro 2	Dato
Return	Si

Esegue la moltiplicazione a*b in modo float

- **void *FloatGtr*(register float *a*, register float *b*)**

Offset	55
Parametro 1	Dato
Parametro 2	Dato
Return	No

Se a>b è vero viene impostato a 1 il bit 0 dell registro r(9).Ad esempio

```
Mov.l   #123456 , r(300)
Mov.l   #0.5    , r(301)
Sys     #55,r(300)
Jnzb    condizione      ;salto condizionale in base al contenuto dell'accumulatore bit 0 r(9)
```

- **void *FloatGeq*(register float *a*, register float *b*)**

Offset	56
Parametro 1	Dato
Parametro 2	Dato
Return	No

Se a>=b è vero viene impostato a 1 il bit 0 dell registro r(9) accumulatore a bit.

- **void *FloatEql*(register float *a*, register float *b*)**

Offset	57
Parametro 1	Dato
Parametro 2	Dato
Return	No

Se a==b è vero viene impostato a 1 il bit 0 dell registro r(9) accumulatore a bit.

- **void *FloatNeq*(register float *a*, register float *b*)**

Offset	58
Parametro 1	Dato
Parametro 2	Dato
Return	No

Se a!=b è vero viene impostato a 1 il bit 0 dell registro r(9) accumulatore a bit.

- **void *FloatLeq*(register float *a*, register float *b*)**

Offset	59
Parametro 1	Dato
Parametro 2	Dato
Return	No

Se a<=b è vero viene impostato a 1 il bit 0 dell registro r(9) accumulatore a bit.

- **void *FloatLes*(register float *a*, register float *b*)**

Offset	60
Parametro 1	Dato
Parametro 2	Dato
Return	No

Se a<b è vero viene impostato a 1 il bit 0 dell registro r(9) accumulatore a bit.

- **Float *IntPowf*(register float *a*, register long *b*)**

Offset	61
Parametro 1	Dato
Parametro 2	Dato
Return	Si

Viene eseguita la potenza del numero a all'intero b. Se A=10.0 e b=3 risultato=1000.

- **long *Fattoriale*(register long *a*)**

Offset	62
Parametro 1	Dato
Return	Si

Viene eseguito il calcolo n! per n come intero.

- **Float *Sin*(register float *a*)**

Offset	63
Parametro 1	Dato
Return	Si

Se a esprime in radianti un angolo tra 0 e 90 ° questa funzione ritorna il valore del seno .Viene calcolata con una serie su 8 iterazioni.

- **Float *Cos*(register float *a*)**

Offset	64
Parametro 1	Dato
Return	Si

Se a esprime in radianti un angolo tra 0 e 90 ° questa funzione ritorna il valore del coseno .Viene calcolata con una serie su 8 iterazioni.

- **Float *Tan*(register float *a*)**

Offset	65
Parametro 1	Dato
Return	Si

Se a esprime in radianti un angolo tra 0 e 90 ° questa funzione ritorna il valore della tangente .Viene calcolata con una serie su 8 iterazioni.

- **Float *Cotan*(register float *a*)**

Offset	66
Parametro 1	Dato
Return	Si

Se a esprime in radianti un angolo tra 0 e 90 ° questa funzione ritorna il valore del cotangente .Viene calcolata con una serie su 8 iterazioni.

- **Float *IntMPowf*(register float *a*, register long *b*)**

Offset	67
Parametro 1	Dato
Parametro 2	Dato
Return	Si

Viene eseguita la potenza del numero a all'intero b.Il numero b viene considerato come negativo.Se a=10.0 e b=3 risultato 0.001

- **void *VisNumFloat*(register float *a*, register long *b*, register long *c*)**

Offset	68
Parametro 1	Dato da visualizzare
Parametro 2	Numero di cifre parte intera
Parametro 3	Numero di cifre parte frazionaria
Return	No

Visualizza sul display dalla posizione corrente un numero floatting point.Nel parametro b vi è il numero di cifre da visualizzare come parte intera e viene riempito con zero se la parte intera è minore.Il massimo parametro per b è 38 altrimenti viene visualizzato un messaggio di errore.Se il numero di cifre è minore di quello richiesto viene visualizzato un messaggio di errore.Il parametro c sono i numeri decimali che vengono visualizzati.

- **Float *SquareRoot*(register float *a*)**

Offset	69
Parametro 1	Dato
Return	No

Calcola la radice quadrata del numero a

- **void *UserReadAll1*(register int porta)**

Offset	70
Parametro 1	Descrittore porta comunicazione
Return	No

Imposta e invia un messaggio lettura stato PLC slave 1.Vedere al capitolo comunicazione seriale comando READALL1.

- **void *UserReadAll2*(register int porta)**

Offset	71
Parametro 1	Descrittore porta comunicazione
Return	No

Imposta e invia un messaggio lettura stato PLC slave 2.Vedere al capitolo comunicazione seriale comando READALL2.

- **void *UserReadAll3*(register int porta)**

Offset	72
Parametro 1	Descrittore porta comunicazione
Return	No

Imposta e invia un messaggio lettura stato PLC slave 3.Vedere al capitolo comunicazione seriale comando READALL3.

- **void *UserReadAll4*(register int porta)**

Offset	73
Parametro 1	Descrittore porta comunicazione
Return	No

Imposta e invia un messaggio lettura stato PLC slave 4.Vedere al capitolo comunicazione seriale comando READALL4.

- **void *UserReadAll5*(register int porta)**

Offset	74
Parametro 1	Descrittore porta comunicazione
Return	No

Imposta e invia un messaggio lettura stato PLC slave 5.Vedere al capitolo comunicazione seriale comando READALL5.

- **void *ExtOut*(void)**

Offset	75
Parametro 1	No
Return	No

Imposta gli ingressi della espansione come uscite .Questa funzione non im posta lo stato ma il solo modo di funzionamento

- **void *WrExtOut*(register int val)**

Offset	76
Parametro 1	Valore a cui impostare le uscite (i primi 10 bit)
Return	No

Se attraverso la funzione ExtOut si impostano gli ingressi delle espansioni come uscite bisogna usare questa funzione per scrivere un valore in tali uscite.Viene comunque monitorato lo stato del flag delle uscite.

.

Le funzioni che seguono sono state implementate sui PLC con OP.SYS. V2.00

- **WrClkOra**

Offset	**77**
Parametro 1	HHB=ORA HLB=MINUTI LHB=SEC
Return	No

Per impostare l'orologio bisogna chiamare questa funzione pre scrivere nei registri Hardware del circuito integrato dell'orologio.I decimi di secondo sono impostati a zero

- **WrClkData**

Offset	**78**
Parametro 1	HHB=NUM.DEL GIORNO HLB=ANNO LHB=MESE LLB=GIORNO
Return	No

Per impostare l'orologio bisogna chiamare questa funzione pre scrivere nei registri Hardware del circuito integrato dell'orologio.

- **UserErase**

Offset	**79**
Parametro 1	No
Return	No

Permette di cancellare la memoria di programma ,negli ultimi 4 kb disponibili.Risulta utile per implementare registrazioni nella memoria flash senza ricorrere ai registri RAM.

- **USerPgm**

Offset	**80**
Parametro 1	Indirizzo di memoria da 0 a 4192,(ultimi 4Kb memroia di programma da 1F000 a 1FFFF)
Parametro 2	Dato da programmare (1 Byte)
Return	No

Scrive il byte del secondo parametro all'indirizzo specificato

- **Matricola**

Offset **81**

Funzione riservata per la matricolazione dei PLC

- **StrBin4**

Offset	**82**
Parametro 1	Puntatore alla stringa (in RAM)
Parametro 2	Dato da formattare
Return	No

Formatta una stringa con un valore in formato dato dalla rappresentazione binaria dei primi 4 bit del dato

- **StrBin8**

Offset	**83**
Parametro 1	Puntatore alla stringa (in RAM)
Parametro 2	Dato da formattare
Return	No

Formatta una stringa con un valore in formato dato dalla rappresentazione binaria dei primi 8 bit del dato

- **StrBin16**

Offset	**84**
Parametro 1	Puntatore alla stringa (in RAM)
Parametro 2	Dato da formattare
Return	No

Formatta una stringa con un valore in formato dato dalla rappresentazione binaria dei primi 16 bit del dato

- **StrBin32**

Offset	**85**
Parametro 1	Puntatore alla stringa (in RAM)
Parametro 2	Dato da formattare
Return	No

Formatta una stringa con un valore in formato dato dalla rappresentazione binaria dei primi 32 bit del dato

- **StrNum**

Offset	**86**
Parametro 1	Puntatore alla stringa (in RAM)
Parametro 2	Dato da formattare
Parametro 3	Formato da usare per il numero in dato.Identico ai formati usati per la visulaizzazzione sul display LCD
Return	No

Formatta una stringa con un valore in RAM

- **AscTime**

Offset	**87**
Parametro 1	No
Return	No

Spedisce dalla porta di comunicazione un messaggio ascii con la indicazione della ora e data.ES.
Alle 10:00:00 del 11/8/97 ::

- **TxNStr**

Offset	**88**
Parametro 1	Puntatore alla stringa
Return	No

Trasmette direttamente la stringa puntata sulla porta di comunicazione 1

- **TxStrNum**

Offset	**89**
Parametro 1	Puntatore alla stringa
Parametro 2	Dato da formattare
Parametro 3	Formato del dato
Return	No

Trasmette direttamente la stringa puntata sulla porta di comunicazione 1 con il dato formattato come specificato dal terzo parametro.Appende alla fine della stringa i caratteri \r\n

- **StrLen**

Offset	**90**
Parametro 1	Puntatore alla stringa
Return	Lunghezza della stringa

Ritorna la lunghezza della stringa

- **StrCpy**

Offset	**91**
Parametro 1	Puntatore alla stringa destinazione in ram
Parametro 2	Puntatore alla stringa sorgente
Return	no

Copia la stringa sorgente nella destinazione

- **StrSet**

Offset	**92**
Parametro 1	Puntatore alla stringa
Parametro 2	dato
Return	no

Imposta tutta la stringa con il carattere specificato dal secondo parametro

- **RetPtrStr**

Offset	**93**
Parametro 1	Puntatore alla stringa
Return	Indirizzo doce finisce la stringa

Ritorna l'indirizzo dove finisce una stringa

- **InitCom0**

Offset	**94**
Parametro 1	Dati 7-8
Parametro 2	Stop 1-2
Parametro 3	Parita 0,1,2
Return	No

Imposta la porta di comunicazione 0 nel modo specificato.La velocita di comunicazione è specificata dalla parte bassa della MEM[7]

- **StrCmp**

Offset	**95**
Parametro 1	Puntatore Stringa 1
Parametro 2	Puntatore Stringa 2
Return	0= diverse 1=uguali

Confronta le due stringhe e ritorna 0 se sono diverse e 1 se sono uguali

- **MemCmp**

Offset	**96**
Parametro 1	Puntatore area confronto 1
Parametro 2	Puntatore area confronto 2
Parametro 3	Lunghezza area
Return	0= diverse 1=uguali

Confronta le due aree di memoria e ritorna 0 se sono diverse e 1 se sono uguali

- **TxTabStr**

Offset	**97**
Parametro 1	Puntatore inizio tabella puntatori alle stringhe
Parametro 2	Indice della stringa da trasmettere
Return	No

Trasmette da una tabella di puntatori la stringa puntata dal valore PTR[ind].
La tabella dei puntatori si costruisce nel seguente modo:

```
TABPTR:
DL   msg1
DL   msg2
DL   msg3
DL   msg4
msg1:
DM “sono la stringa 1”,0
msg2:
DM “sono la stringa 1”,0
msg3:
DM “sono la stringa 1”,0
msg4:
DM “sono la stringa 1”,0
```

- **TxTabStrNum**

Offset	**98**
Parametro 1	Puntatore inizio tabella Strutture
Parametro 2	Indice della stringa da trasmettere
Return	0= diverse 1=uguali

Trasmette da una tabella di strutture dei messaggi formattati con una stringa e un valore.
La tabella delle strutture è così composta:

```
TABSTRUTT:
DL  Rec1
DL  Rec2
DL  Rec3
DL  Rec4
;prima struttura
Rec1:
DL  Msg1:
DL  #R(300)    ;trasmette il valore di R(300)
DL  20         ;formato 0….99999
Msg1:
DM  “la prima stringa”,0
Etc…
```

- **RegVar**

Offset	**99**
Parametro 1	Indice del registro da prendere in cosiderazione 0…900
Return	NO

Confronta la parte bassa e la parte alta del registro,se sono diverse mette a 1 il bit 0 della MEM[9]
.Permette la implementazione di un contatto che si chiude se varia la parte bassa del registro.

- **UserAscOra**

Offset	**100**
Parametro 1	No
Return	No

Spedisce dalla porta di comunicazione seriale un messaggio con l’ora attuale .Es. 10:00:10

- **UserAscData**

Offset	**101**
Parametro 1	No
Return	No

Spedisce dalla porta di comunicazione seriale un messaggio con la data attuale .Es. 10/09/98

- **RxStr**

Offset **102**
Parametro 1 Puntatore alla stringa
Parametro 2 Indice Registro di uscita 0..900
Return No
Monitorizza la ricezione sulla porta di comunicazione 1 .Quando viene ricevuto il carattere CR (13) confronta i caratteri ricevuti con la stringa e se sono uguali imposta il registro di uscita a 1

- **RxStr**

Offset **103**
Parametro 1 Puntatore alla tabella delle stringhe
Parametro 2 Indice Registro di uscita 0..900
Return No
Monitorizza la ricezione sulla porta di comunicazione 1 .Quando viene ricevuto il carattere CR (13) confronta i caratteri ricevuti con le stringhe della tabella e se sono uguali imposta il registro di uscita con il bit a 1 dato dal numero della stringa uguale.

- **XYZ**

Offset **104**
Parametro 1 Puntatore alla Struttura per posizionamento
Return No
Esegue un posizionamento ad anello aperto per il pilotaggio di motori passo passo su tre assi ad interpolazione lineare.La struttura di posizionamento è una area di memoria RAM composta da 23 registri ,il parametro deve specificare l'indirizzo del primo registro,con le seguenti funzioni:

REG[0] Stato del posizionamento 1 arrivato 0 in corso
REG[1] Punto di Arrivo asse X
REG[2] Punto di Arrivo asse Y
REG[3] Punto di Arrivo asse Z
REG[4] Punto di Partenza asse X
REG[5] Punto di Partenza asse Y
REG[6] Punto di Partenza asse Z
REG[7] Massima distanza tra i punti
REG[8] Variabile temporanea che va da 0 a REG[7]
REG[9] Distanza X Attuale
REG[10] Distanza Y Attuale
REG[11] Distanza Z Attuale
REG[12] Errore X Attuale
REG[13] Errore Y Attuale
REG[14] Errore Z Attuale
REG[15] Incremento X
REG[16] Incremento Y
REG[17] Incremento Z
REG[18] Stato delle uscite
REG[19] Se vi è buffer di lavorazione posizione sul buffer
REG[20] Se vi è buffer di lavorazione lunghezza del buffer
REG[21] Se vi è buffer di lavorazione indirizzo in ram del buffer primario
REG[22] Se vi è buffer di lavorazione indirizzo in ram del buffer secondario

Questa struttura è comune per tutte le funzioni di posizionamento da 1 a 3 assi in modo da poter passare da una funzione all'altra.
La funzione imposta direttamente le uscite da 9 a 16 in modo da implementare per ogni asse i segnali ENABLE ,DIR,CLOCK rispettivamente per X(OUT 9,10,11),Y,Z
Viene calcolata e generata una uscita ad ogni chiamata alla funzione per cui per un controllo ottimale della velocità è conveniente usare un interrupt utente a timer per chiamare la funzione.
Per portarsi ad un punto basta scrivere le coordinate di Arrivo X,Y,Z .

- **XY**

Offset **105**
Parametro 1 Puntatore alla Struttura per posizionamento
Return No
Come la funzione 104 ma solo XY

- **XZ**

Offset **106**
Parametro 1 Puntatore alla Struttura per posizionamento
Return No
Come la funzione 104 ma solo XZ

- **YZ**

Offset **107**
Parametro 1 Puntatore alla Struttura per posizionamento
Return No
Come la funzione 104 ma solo YZ

- **X**

Offset **108**
Parametro 1 Puntatore alla Struttura per posizionamento
Return No
Come la funzione 104 ma solo X

- **Y**

Offset **109**
Parametro 1 Puntatore alla Struttura per posizionamento
Return No
Come la funzione 104 ma solo Y

- **Z**

Offset **110**
Parametro 1 Puntatore alla Struttura per posizionamento
Return No
Come la funzione 104 ma solo Z

- **AnalogDigital**

Offset **111**
Parametro 1 No
Return No
E la funzione con cui il sistema operativo in automatico legge dal convertitore analogico i valori della conversione.Solo una conversione viene effettuata per chiamata e quindi occorrono 8 chiamate per convertire tutti e 8 gli ingressi analogici.
Si può decidere di effettuare in modo manuale la chiamata a tale funzione impostando il Bit 13 della memoria MEM[214] a 1,mentre se è zero questa funzione viene chiamata dal sistema operativo ogni 20 ms.

- **ChADC**

Offset **112**
Parametro 1 Numero del canale analogico 0..7 da convertire
Return No
Per conversioni analogiche veloci fino a 5 khz si può usare un interrupt utente a timer che chiama questa funzione con il parametro che indica il canale da convertire.Questa funzione imposta a 1 il bit 13 della memoria MEM[214] per informare il sistema operativo che non deve più eseguire le conversioni analogiche.

- **TickUser1**

Offset **113**
Parametro 1 Divisore
Return No
Permette di variare la frequenza dell'interrupt a timer utente 1 senza interromperne l'utilizzo.Il parametro specifica il divisore per generare la frequenza. F=14753000/divisore

- **TickUser2**

Offset **114**
Parametro 1 Divisore
Return No
Permette di variare la frequenza dell'interrupt a timer utente 2 senza interromperne l'utilizzo.Il parametro specifica il divisore per generare la frequenza. F=14753000/divisore

- **CNCXYZ**

Offset **115**
Parametro 1 Puntatore alla struttura di posizionamento
Return No

Come la funzione 104 ma implementa il passaggio di vertici tramite buffere primario e secondario.
Per usare la funzione bisogna specificare nel registro 20 della struttura di posizionamento la lunghezza del buffer contenente i vertici (poiché ogni vertice è sempre formato da 3 coordinate XYZ la lunghezza è in terni ovvero 10 specifica 10 vertici che ocuupano 30 registri)e nel registro 21 l'indirizzo del buffer contenente i vertici.
La funzione si posiziona a su tutti i vertici contenuti.Alla fine della esecuzione dei vertici se il registro 22 contiene un indirizzo non nullo che punta a un buffer contenente lo stesso numero di vertic,questo buffer viene copiato nel buffer primario, e il posizionamento continua su questi vertici.Il registro 21 viene quindi azzerato.Nel tempo di lavorazione quindi un computer supervisore può impostare un nuovo buffer secondario.
In questo modo si ottiene una buona uniformita di lavorazione eliminando tempi morti dovuti alla comunicazione dei vertici di lavorazione.

- **CNCXY**

Offset **116**
Parametro 1 Puntatore alla struttura di posizionamento
Return No

Come funzione 115 ma solo XY

- **CNCXZ**

Offset **117**
Parametro 1 Puntatore alla struttura di posizionamento
Return No

Come funzione 115 ma solo XZ

- **CNCYZ**

Offset **118**
Parametro 1 Puntatore alla struttura di posizionamento
Return No

Come funzione 115 ma solo YZ

- **EnableUser1**

Offset **119**
Parametro 1 No
Return No

Abilita l'interrupt timer utente 1

- **EnableUser2**

Offset **120**
Parametro 1 No
Return No

Abilita l'interrupt timer utente 2

- **DisableUser1**

Offset	**121**
Parametro 1	No
Return	No

Disabilita l'interrupt timer utente 1

- **DisableUser2**

Offset **122**
Parametro 1 No
Return No
Disabilita l'interrupt timer utente 2

- **EnableEncoder**

Offset **123**
Parametro 1 No
Return No
Abilita la gestione dell'encoder default

- **DisableEncoder**

Offset **124**
Parametro 1 No
Return No
Non esegue la gestione dell'encoder

- **SweepTimerUno**

Offset **125**
Parametro 1 Puntatore alla Struttura di posizionamento
Return No
Con questa funzione alla posizione di OFFSET 23,24,25,26 rispetto l'inizio della struttura di posizionamento vi sono le variabili per la modulazione della velocita e precisamente

R[POS+23]=Velocità minima

R[POS+24]=Velocità massima

R[POS+25]=lunghezza in passi accellerazione-decellerazione

R[POS+26]=Incremento unitario

Alla prima chiamata della funzione viene impostato un rate per la gestione dell'interrupt utente uno dato dal numero posto nella memoria ad offset 23.Ad ogni chiamata dell'interrupt viene modulato la durata dell'interrupt stesso per ottenere un profilo di accellerazione a trapezio di cui la memoria ad offset 25 rappresenta in passi la distanza di accellerazione mentre la memoria ad offset 26 rappresenta l'accellerazione fino ad ottenere la velocità specificata nella memoria ad offset 24.
La frequenza è ottenuta con
Fmin=14753000/velocità minima
Fmin=14753000/velocità massima

- **SweepCNCXYZ**

Offset **126**
Parametro 1 Puntatore alla struttura di posizionamento
Return No
Come la funzione 115 ma gestisce anche il profilo della velocita (CNCXYZ+SweepTimerUno)

- **SweepCNCXY**

Offset **127**
Parametro 1 Puntatore alla struttura di posizionamento
Return No
Come funzione 126 ma solo XY

- **SweepCNCXZ**

Offset **128**
Parametro 1 Puntatore alla struttura di posizionamento
Return No
Come funzione 126 ma solo XZ

- **SweepCNCYZ**

Offset **129**
Parametro 1 Puntatore alla struttura di posizionamento
Return No
Come funzione 126 ma solo YZ

- **StopZSwepCNCXYZ**

Offset 130
Parametro 1 Puntatore alla struttura di posizionamento
Return No

Grazie a questa funzione è possibile fare in modo che l'interpolazione venga fermata ogni qual volta il punto di fine Z è diverso da zero.Quando si verifica QUESTA CONDIZIONE la memoria che indica lo stato del posizionamento viene impostata a 2.Da programma utente a questo punto si può monitorare questo stato con un contatto generico bit 1 e far partire un ciclo, ad esempio attivare una uscita .Quando il ciclo finisce per far riavviare il posizionamento bisogna impostare la memoria dello stato a 4 ad esempio con una bobina generica bit 2.
Esempio Applicativo:
Supponiamo di dover realizzare una macchina che riempie bottiglie .Ogni volta che l'asse Zeta si abbassa per portarsi entro la bottiglia ,il posizionamento viene fermato e parte il ciclo di riempimento.Il plc attiva l'uscita di riempimento e monitorizza il livello del liquido tramite l'ingresso analogico.Quando il livello è nel range prefissato viene impostato a 4 la memoria dello stato del posizionamento.L'asse Zeta viene alzato e la macchina si sposta sulla prossima bottiglia

- **Esempio Applicativo**

Riteniamo utile per il programmatore che si appresta a studiare il linguaggio assembler del ***microPLC Merlino*** eseguire un esempio passo passo per potersi addentrare nel sistema.
Per valutare il compilatore ladder che esegue in modo automatico la trasformazione da schema elettrico a linguaggio assembler vere l'apposito manuale software.
Tra le varie strade possibili abbiamo scelto di programmare come applicazione un termoregolatore Proporzinale.L'applicazione di persè complessa viene suddivisa in vari parti in modo da entrare per passi nella programmazione.

- **Lezione 1**

Supponiamo all'inizio che l'apparecchiatura debba accettare una temperatura da 0 a 100 gradi da un ingresso analogico avere un set-point inpostato in una memoria come valore 1000 punti (0..100.0C°) ed in uscita dare un segnale proporzionale all'ingresso e avere due uscite digitali ,una per riscaldamento e l'altra per il raffrescamento.Eseguiamo quindi le seguenti scelte

Ingresso sonda temperatura	Ingresso analogico 1
Uscita analogica di pilotaggio	Uscita analogica Zero su Morsetto Ingr.An.Zero
Abilitazione risacaldamento	OUTPUT 1
Abilitazione raffrescamento	OUTPUT 2

Ed iniziamo ad implementare il programma.
Impostiamo le seguenti definizioni comode per avere un accesso diretto e univoco ai registri interessati:

```
Sez A   SETPOINT:   EQU   300   ;R(301) contiene il set-point di regolazione
        BMORTA:     EQU   301   ;R(302) range temp.banda morta
        TEMP:       EQU   302   ;R(303) temperatura virgola mobile
        NUMPLC:     EQU   3     ;identifica memoria impostazione Numero PLC
        ACC:        EQU   9     ;accumulatore per operazioni a BIT
        SYSCOM0:    EQU   24    ;funzione impostazione vel.comunicazione COM 0
        SYSCOM1:    EQU   25    ;funzione impostazione vel.comunicazione COM 1
        ANI:        EQU   64    ;registro lettura canale analogico 0 e 1
        K:          EQU   400   ;registro uso generale
        OUT:        EQU   20    ;registro delle uscite
        FLAG:       EQU   6     ;flag delle uscite
        DAC:        EQU   68    ;registro uscite analogiche
```

La prima parte del programma serve per inizializzare il sistema passiamo quindi a scrivere le seguenti righe di codice

```
Sez.B   Mov.lw    #1,R(NUMPLC)        ;imposto apparato come numero 1 sulla rete
        Mov.l     #8,R(ACC)           ;preparo per chiamata a funzione di sistema
        Sys       #SYSCOM0,R(ACC)     ;imposto COM 0 a 57600baud
        Mov.l     #0,R(ACC)           ;preparo per chiamata a funzione di sistema
        Sys       #SYSCOM1,R(ACC)     ;imposto COM 1 a 115200baud
        Mov.l     #1,R(FLAG)          ;abilito le uscite
        Mov.l     #0,R(TEMP)          ;reset valore temperatura prima della lettura
        Setdac    #0                  ;Imposto ingresso analogico 6 come uscita an
```

Il programma inizia con Start e finisce con End.Queste due istruzioni identificano il **Corpo** del programma. Come prima fase leggiamo l' ingresso analogico e trasformiamolo nella scala opportuna

```
Sez.C   Start                       ;inizio scansione di programma
        Mov.lw    #0,R(K)           ;lw=0
        Mov.hw    R(ANI),R(K)       ;prendo valore canale analogico 1
        Swap.l    R(K)              ;porto in forma normalizzata lungo R(K)
        Mult.l    #1000,R(K)        ;R(K)* fondo scala
        Div.l     #1024,R(K)        ;R(K)/1024 temperatura 0…1000 0..100.0 C°
        Mov.l     R(K),R(TEMP)      ;salvo temperatura letta
```

Controllo raffrescamento riscaldamento

```
Sez.D   Mov.l     R(SETPOINT),R(K)  ;prendo setpoint
        Subc.l    R(BMORTA),R(K)    ;SETPOINT-BMORTA
        Les.l     R(TEMP),R(K),riscalda ;TEMP<( SETPOINT-BMORTA)
        Mov.l     R(SETPOINT),R(K)  ;prendo setpoint
        Add.l     R(BMORTA),R(K)    ;SETPOINT+BMORTA
        Gtr.l     R(TEMP),R(K),raffres. ;TEMP>( SETPOINT+ BMORTA)
```

Se il programma non passa il controllo ne a riscalda ne a raffres. Situazione di stallo disabilito uscite abilitazione risc.e raffr. Reset uscita analogica e termine scansione

```
Sez.E   ABSRESB   R(OUT).0          ;reset uscita 0
        ABSRESB   R(OUT).1          ;reset uscita 1
        Mov.lw    #0,R(DAC)         ;reset uscita analogica 0
        End
```

Se vi è bisogno di riscaldamento il flusso di programma passa alla etichetta riscalda

```
Sez.F   Riscalda:                   ;riscaldamento
        ABSSETB   R(OUT).0          ;accendo uscita zero riscaldamento ON
        ABSRESB   R(OUT).1          ;interblocco con raffrescamento
        Mov.l     R(SETPOINT),R(K)  ;prendo SETPOINT
```

```
        Subc.l      R(TEMP),R(K)          ;SETPOINT-TEMP
        Sub         CALCOUT               ;chiama routine che in base alla diff. da l'uscita
        End
```

Se vi è bisogno di raffrescamento il flusso di programma passa alla etichetta raffres.

```
Sez.G   Raffres:                          ;raffrescamento
        ABSSETB     R(OUT).1              ;accendo uscita zero raffrescamento ON
        ABSRESB     R(OUT).0              ;interblocco con riscaldamento
        Mov.l       R(TEMP),R(K)          ;prendo TEMP
        Subc.l      R(SETPOINT),R(K)      ; TEMP- SETPOINT
        Sub         CALCOUT               ;chiama routine che in base alla diff. da l'uscita
        End
```

La routine Calcout calcola in base alla differenza dal setpoint l'uscita analogica

```
Sez.F   Calcout:                          ;calcolo uscita range 0..255 ingresso su R(K)
        Shfl.l      R(K)                  ;se R(K) =R(K)*2
        Shfl.l      R(K)                  ;se R(K) =R(K)*4  amplifico per 4
        Gtr.lw      #255,R(K),Noclip      ;se R(K).lw >255 allora R(K)=255
        Mov.l       #255,R(k)             ;max out
        Noclip:
        Mov.lw      R(K),R(DAC)           ;imposto uscita analogica zero
        Ret
```

Provare il precedente programma.Per adesso non vi è nessun riferimento alla visualizzazione dei dati Impostando sulla memoria 300 un setpoint valido nel range 0..1000 e nella bandamorta - r(301) – un dato valido esempio 10 (±1C°) e inviando un segnale analogico nel range 0..5v al morsetto 23 si apre e chiude i relè della uscita 0 e 1.Dal morsetto 22 esce un segnale analogico proporzionale alla differenza. Quando il PLC è dentro il range della bandamorta le uscite sono spente e l'uscita analogica è zeo.

- **Lezione 2**

L'applicazione sopra descritta sino a questo punto descrive più un termostato che un regolatore ma da già un'idea della applicazione.In questo paragrafo iniziamo a impostare delle pagine di visualizzazione Vogliamo che alla accenzione della macchina dopo l'inizializzazione venga visualizzato un messaggio con informazioni generali.Aggiungiamo alla sezione A delle definizioni le seguenti righe di codice

```
Sez A   FILLDSP:    EQU     10      ;Funzione di sistema visualizzazione messaggi
        TIMER:      EQU     48      ;contatti dei timer
```

Alla fine della sezione B aggiungiamo le seguenti linee di codice

```
Sez.B   Mov.l       #paginastart,R(K)   ;punto ai messaggi della pagina iniziale
        Sys         #FILLDSP,R(K)       ;eseguo funziona che visulaizza le stringhe
```

Rimane adesso da definire l'array di puntatori alle stringhe da visualizzare e la definizione delle stringhe.Alla fine della sezione F aggiungere

```
Sez.G   DL          0                   ;forza allineamento della memoria come lungo
        Paginastart:                    ;indirizzo pagina start
        DL          Str1                ;punta a stringa 1
        DL          Str2                ;punta a stringa 2
        Str1:                           ;indirizzo Str1
        DM          " TERMOTRONIC INC."             ;1° riga (16 char)
        DM          " V.1.00        "               ;2° riga (16 char)
```

Eseguite le precedenti modifiche compilate e provate il programma .Funziona tutto come prima ma compaiono i messaggi impostati.Vogliamo che il precedente messaggio compaia sul display per 5 secondi dopo di che vada nel loop principale e visualizzi altri messaggi.Tra i vari modi possibili abbiamo scelto il seguente,che reimposta lo start program a fine dei 5 secondi.Alla fine della sezione A aggiungere la seguente parte di codice

```
Sez.B   SetTimer    #1,#50,#0           ;Timer0 a 50 decimi di secondo
        Start                           ;questo start è valido solo nei primi 5 secondi
        Mov.llb     #1,r(ACC)           ;ACC a BIT =1
        Stt         #0                  ;abilito timer
        Ldb         R(TIMER).0          ;leggo stato del timer
        Jnz         Loop                ;arrivato vado al loop principale
        End                             ;fine ciclo iniziale
```

All'inizio della sezione C aggiungere

```
Sez.C   Loop                            ;rientro e nuovo start successivo a questa etichetta
```

Compilare e provare il programma ottenuto.Adesso campaiono i messaggi sul display e la regolazione funziona come nella lezione 1.

- **Lezione 3**

Adesso vogliamo che trascorsi i primi 5 secondi compaia una pagina sul display lcd che visualizzi la temperatura misurata all'ingresso analogico 1,il setpoint di regolazione,l'uscita attuale e se il sistema è in raffrescamento o riscaldamento.
Innanzi tutto bisogna definire una pagina di sistema con due variabili, la temperatura e il setpoint, perchè l'uscita analogica si vuole visualizzare con una barra.

Alla fine della sezione G aggiungere il seguente codice

```
Sez.G   DL          0                   ;allineamento
        PaginaNorm:                     ;definisco la pagina di visualizzazione normale
        DL          Mp1                 ;messaggio riga 1
        DL          Mp2                 ;messaggio riga 2
        DL          Var1                ;puntatore struttura descrittore variabile 1
        DL          Var2                ;puntatore struttura descrittore variabile 2
        DL          0                   ;fine struttura
        Mp1:
        DM          "S(  )      "       ;mesg riga 1 (16 char)
        Mp2:
        DM          "           "       ;mesg riga 2 (16 char)
        Var1:
        DL          R(SETPOINT)         ;indirizzo 1° variabile
        DL          Limit1p1            ;limite per questa variabile
        Var2:
        DL          R(TEMP)             ;indirizzo 1° variabile
        DL          Limit2p1            ;limite per questa variabile
        Limit1p1:
        DB          32,22,2,0           ;Vis. 99.9 ptr a long *lock X=2 Y=0
        DL          0,1000              ;min=0 max=1000
        Limit2p1:
        DB          32,22,2,1           ;Vis. 99.9 ptr a long *lock X=2 Y=1
        DL          0,1000              ;min=0 max=1000
        MsgCaldo:                       ;se richiesto riscaldamento
        DM          "---->>>>",0        ;stringa
        MsgFreddo:                      ;se richiesto raffrescamento
        DM          "<<<<----",0        ;stringa
        MsgStop:                        ;se entro banda morta
        DM          "<<<<>>>>",0        ;stringa
        MsgCanc:
        DM          "       ",0         ;cancella la barra
```

Alla fine della sezione A aggiungere la seguente definizione

```
Sez A   VISPAG:     EQU     13          ;Funzione di sistema visualizz. Pagina utente
        POSXY:      EQU     4           ;posiziona il cursore sul display
        PUTCH:      EQU     2           ;scrive sul display un carattere
        WRMSG:      EQU     5           ;scrive una stringa
```

All'inizio della sezione C dopo l'etichetta Loop aggiungere

```
Sez.C   Sub         Paginadefault       ;routine di visualizzazione pagina default
```

Alla fine della sezione F aggiungere il codice di questa routine

```
Sez.F   Paginadefault:
        Mov.l       #paginaNorm,R(K)    ;prendo indirizzo pagina
        Sys         #VISPAG,R(K)        ;e visualizzo
        Sub         VisOut              ;visualizzo uscita analogica
        Ret
        VisBarra:                       ;routine visulizzazione barra
        Mov.lw      #8,R(K)             ;imposto X=8
        Mov.lw      #0,R(K+1)           ;imposto Y=0
```

```
Sys             #POSXY,R(K)         ;vado alla posizione XY per disegno barra
Mov.l           #MsgCanc,R(K)       ;cancello la barra precedente
Sys             #WRMSG,R(K)
Mov.lw          #8,R(K)             ;imposto X=8
Mov.lw          #0,R(K+1)           ;imposto Y=0
Sys             #POSXY,R(K)         ;vado alla posizione XY per disegno barra
Mov.lw          R(DAC),R(K)         ;prendo uscita segnale analogico
Mov.hw          #0,R(K)             ;reset parte alta
Div.l           #32,R(K)            ;ottengo res da 0 a 7
Eql.lw          #0,R(K),FineBarra   ;se 0 no disegno nulla
ContaBarra:
Mov.l           #4,R(K+1)           ;carattere 4 quadrato pieno
Sys             PUTCH,R(K+1)        ;scrivo carattere
Djnz            R(K),ContaBarra     ;ciclo per R(K)
FineBarra
Sub             RisRaf              ;visualizzo stato riscaldamento raffrescamento
Ret
RisRaf:
Mov.lw          #8,R(K)             ;imposto X=8
Mov.lw          #1,R(K+1)           ;imposto Y=1
Sys             POSXY,R(K)          ;vado alla posizione XY per disegno barra
Ldb             R(OUT).0            ;richiesta caldo
Jbz             Visfreddo           ;non visualizzo msg per caldo
Mov.l           #MsgCaldo,R(K)      ;visualizzo messaggio caldo
Sys             #WRMSG,R(K)
Ret
Visfreddo:
Ldb             R(OUT).1            ;richiesta freddo
Jbz             VisSet
Mov.l           #MsgFreddo,R(K)     ;visualizzo messaggio freddo
Sys             #WRMSG,R(K)
Ret
VisSet:                             ;entro banda morta
Mov.l           #MsgStop,R(K)       ;visualizzo messaggio caldo
Sys             #WRMSG,R(K)
Ret
```

Eseguite le modifiche compilare e caricare il programma quindi provare le modifiche.

- **Lezione 4**

Abbiamo ottenuto la pagina di introduzione e la pagina dei valori adesso dobbiamo rinfrescare continuamente questi valori per dare un dato aggiornato a chi legge il display.Per fare questo dobbiamo collegarci ad una funzione di sistema e a dei temporizzatori.
Alla fine della sezione a aggiungere

```
Sez A   REFRPAG:    EQU   11      ;Funzione di rinfresco dati della pagina
        SPEC0:      EQU   74      ;registro contatti speciali 0
        LAST:       EQU   206     ;registro che contiene l'ultima pagina visualizzata
```

Nel loop principale dopo lo start della sezione C aggiungiamo il seguente codice

```
Sez.C   Ldb     R(SPEC0).0          ;bit 0 inverte lo stato ogni 1/10 sec
        REDGE   #0                  ;con in serie fronte
        Jbz     Norefresh           ;se risultato =0 niente rinfresco
        Sys     #REFRPAG,R(LAST)    ;rinfresco dati
        Sub     VisBarra            ;e visualizzo barra + stato
        Norefresh:
```

Compilare il programma caricarlo e provarlo. Adesso il display si è animato ed in parte già da una prima idea dell'apparato.

- **Lezione 5**

Scopo di questa lezione è quello di implementare 2 pagine utente in cui impostare il SetPoint e la banda morta,aggungere alla fine della sezione a:

```
Sez A   RELE1:      EQU     46      ;Memoria rele blocco 1 (va bene quals. Memoria)
        KEY:        EQU     78      ;tasti di sistema
```

Alla fine della sezione G aggiungere il seguente codice

```
Sez.G   DL          0               ;allineamento
        pagSetPoint:                ;definizione pagina impostazione SetePoint
        DL          msgpag2r1
        DL          msgpag2r2
        DL          Varpag2V1
        DL          0
        msgpag2r1:
        DM          "SetPoint   ",0
        msgpag2r2:
        DM          "Regolazione    ",0
        DL          0
        varpag2V1:
        DL          r(300)
        DL          Limit2Var1
        limit2Var1:
        DB          32,6,11,0
        DL          0,999
        pagBMorta:
        DL          Msgpag3r1
        DL          Msgpag3r2
        DL          Varpag3V1
        DL          0
        msgpag3r1:
        DM          "B. Morta   ",0
        msgpag3r2:
        DM          "Regolazione    ",0
        DL          0
        varpag3V1:
        DL          r(301)
        DL          Limit3Var1
        limit3Var1:
        DB          32,6,11,0
        DL          0,50
```

Assumiamo che quando si vuole fare una impostazione avviamo un ciclo di tempo prestabilito oltre il quale il sistema ritorna nella pagina di default.Per fare questo riusiamo il timer 0 con una nuova impostazione e costruiamo un circuito che imposta la bobina di R0 e a fine tempo la rilascia .In questo modo R0 funziona come flag di impostazione in corso.Questo circuito è realizzato con il parallelo tra i contatti dei tasti f1e f2 con in parallelo un contatto NA di R0 per l'autointrattenimento con in serie il contatto NC di timer 0 che abilita sia R0 che timer 0.Dopo la etichetta Loop inserire la nuova impostazione del timer:

```
Sez.C   SetTimer    #1,#200,#0      ;reimposto timer 0 a 20 secondi
```

Dopo l' etichetta Norefresh inserire il circuito rilevatore di impostazione

```
Sez.C   Ldb         R(KEY).26       ;contatto tasto F1
        Orb         R(KEY).25       ;parallelo tasto F2
        Orb         R(RELE1).0      ;autointrattenimento R0
        Andb        !(R(TIMER).0)   ;serie NC Timer 0
        Stb         R(RELE1).0      ;risultato su R0
```

```
        Stt         R(TIMER).0          ;e timer 0
```

Dopo questo circuito il relè R0 è attivo ogni volta che si preme f1 o f2 per 20 secondi.
Dobbiamo adesso implementare i circuiti che visualizzano le pagine di impostazione.Di seguito al codice precedente aggiungere

```
Sez.C   Ldb         R(RELE1).0          ;R0 attivo stato di  impostazione
        Andb        R(KEY).26           ;serie tasto F1
        Rsedge      #1                  ;su rilevatore id fronte 1
        Jbz         NoSetPoint          ;res =0 non visualizzo impostazione setpoint
        Mov.l       #pageSetPoint,R(K)  ;altrimenti carico pagina impostazione SetPoint
        Sys         #VISPAG,R(K)        ;e visualizzo
        Jmp         NoBMorta            ;se f1 è premuto insieme a F2
        NoSetPoint:
        Ldb         R(RELE1).0          ;R0 attivo stato di  impostazione
        Andb        R(KEY).25           ;serie tasto F2
        Rsedge      #2                  ;su rilevatore id fronte 2
        Jbz         NoBMorta            ;res =0 non visualizzo impostazione Bmorta
        Mov.l       #pageBmorta,R(K)    ;altrimenti carico pagina impostazione SetPoint
        Sys         #VISPAG,R(K)        ;e visualizzo
        NoBMorta:
```

Adesso bisogna che a fine timeout di impostazione (20 Sec) sia reimpostata la pagina di default. Per fare questo aggiungere sempre di seguito al codice precedente le seguenti linee di programma.Questa parte del circuito rivela la caduta di R0 e quindi visualizza la pagina di default

```
        Ldb         !(R(RELE1).0)       ;stato NC di R0
        RsEdge      #3                  ;variazione
        Jbz         NoRefresh1          ;non è caduto
        SUB         Paginadefault       ;visualizzo pagina default
        NoRefresh1:
```

Bisogna adesso modificare la routine VisBarra perché ad ogni rinfresco dei dati della pagina visualizza la barra di stato anche quando siamo nelle pagine di impostazione.
Per questo alla sezione F si ha:

```
Sez.F   VisBarra:                       ;routine visulizzazione barra
        Ldb         R(RELE1).0          ;prendo stato impostazione
        Jbz         VisBarra1           ;non siamo nello stato impostazione
        Ret                             ;si niente barra
        VisBarra1:
```

Compilare caricare e provare il programma .Premento F1 compare la pagina SetPoint.Premendo F2 compare la pagina BamdaMorta.Dopo 20 secondi ritorna la pagina di impostazione.

- **Lezione 6**

Scopo di questo paragrafo è implementare una routine che, quando sono visualizzate le pagine di impostazione, permetta l'immissione dei dati e di modificare la visualizzazione delle pagine di impostazione, in modo che premendo il tasto ESC si torni subito alla pagina di default senza attendere i 20Sec. Inoltre il tempo di timeout deve iniziare a scorrere dall'ultima impostazione dei tasti UP, DOWN, ENTER. Aggiungere:

```
Sez A   MODPAG:     EQU     14          ;funzione che modifica i dati di una pagina
        TIMVAL0:    EQU     215         ;dati del timer 0 HW=val LW=set
        MOD:        EQU     303         ;memoria in uso a funzione MODPAG
```

Subito dopa la etichetta Loop aggiungere

```
Sez C   Mov.l       #0x00640000,R(MOD)      ;imposta divisore per funzione MODPAG a 100 cicli
```

Sempre nella sezione C bisogna mettere in serie al contatto NC del timer 0, che resetta lo stato di impostazione dati, il contatto NC del tasto ESC in modo che, premendo questo tasto, si esca immediatamente da QUESTO STATO.

```
Sez C   Andb        !(R(KEY).30)            ;reset stato impostazione
```

Adesso bisogna impostare la routine del rinfresco dati di pagina quando siamo in fase impostazione.Subito dopo l' etichettta Norefresh aggiungere

```
Sez C   Ldb         R(RELE1).0              ;siamo in stato impostazione
        Jbz         NOMod                   ;no
        Mov.l       #MOD,R(K)               ;si passo puntatore memoria in uso
        Sys         #MODPAG,R(K)            ;fiunzione modifica dati
        NoMod:
```

Nella medesima sezione subito dopo la etichetta NoRefresh1 aggiugere

```
Sez C   Ldb         R(KEY).28               ;tasto UP in parallelo a
        Ldb         R(KEY).29               ;tasto DOWN in parallelo a
        Ldb         R(KEY).31               ;tasto ENTER
        Jbz         NoTimerOut              ;non sono premuti
        Mov.hw      #0,R(TIMVAL0)           ;se uno è premuto,a zero stato valore timer 0
        NoTimerOut:
```

Questo logica vede se o UP o DOWN o ENTER sono premuti e imposta il numero intero che contiene il conteggio del timer, a zero, facendolo ripartire da zero.
A questo punto compilare ed eseguire il programma.Accertarsi che siano stati raggiunti gli scopi prefissati.

- **Lezione 7**

Dobbiamo adesso migliorare il loop di regolazione per questo è innanzi tutto necessario scegliere una base dei tempi adeguata. Supponiamo di avere una regolazione che dia una uscita stabile ogni 1,6 secondi, implementiamo poi una media delle misure di temperatura tra una regolazione e l'altra.
Aggiungiamo le seguenti definizioni:

```
Sez A | MEDIA:    EQU   304   ;indice per media misure
      | BTEMP:    EQU   305   ;Buffer di 15 registri per copia dati
```

Nella sezione C aggiungere sotto alla etichetta Loop la impostazione per i timer 1 e 2

```
Sez C | SetTimer  #1,#16,#1     ;regolazione ogni 1,6 sec timer 1
      | SetTimer  #1,#1,#2      ;timer lettura esecuzione valiri media Timer 2
```

Subito dopo lo start aggiungere un circuito che reimposta il timer 1 e 2 facendoli diventare ciclici

```
Sez C | Ldb       R(RELE1).1    ;prendo NA rele 1
      | Stt       #1            ;abilita timer 1
      | AbsSetB   R(RELE1).1    ;ogni ciclo abilita timer reset su regolazione
      | Ldb       R(RELE1).2    ;prendo NA rele 2
      | Stt       #2            ;abilita timer 2
      | AbsSetB   R(RELE1).2    ;ogni ciclo abilita timer reset su regolazione
```

Sotto l'etichetta Notimeout sostituire il codice con il seguente

```
Sez C | Ldb       R(TIMER).2          timer 2 arrivato
      | Jbz       NoMisura
      | Mov.lw    #0,R(K)             ;si prendo misura lw=0
      | Mov.hw    R(ANI),R(K)         ;prendo valore canale analogico 1
      | Swap.l    R(K)                ;porto in forma normalizzata lungo R(K)
      | Mult.l    #1000,R(K)          ;R(K)* fondo scala + decimi +centesimi
      | Div.l     #1024,R(K)          ;R(K)/1024 temperatura 0...1000 0..100.0 C°
      | AbsResb   R(RELE1).2          ;reset rele abilitazione timer 2
      | mov.l     #BTEMP,R(K+1)       ;Dove metto il valore
      | Add.l     R(MEDIA),R(K+1)     sommo indice
      | mov.l     R(K),@R(K+1)        ;immagazzino valore nel buffer
      | inc1.l    R(MEDIA)            ;incremento indice
      | gtr.lw    #15,R(MEDIA),Nomisura ;fine indice
      | mov.l     #15,R(MEDIA)        ;eseguo le media dei 15 valori
      | mov.l     #BTEMP,R(K+1)       ;puntatore
      | mov.l     #0,R(K)             ;reset stato registro
      | Ciclomedia:
      | mov.l     @R(K+1),R(K+2)
      | add.l     R(K+2),R(K
      | inc1.l    R(K+1)              ;incremento puntatore ai dati
      | djnz      R(MEDIA),ciclomedia ;loop
      | DiV.l     #15,R(K)            ;somma dei valori diviso 15
      | Mov.l     R(K),R(TEMP)        ;salvo temperatura letta
      | mov.l     #0,R(MEDIA)         ;reset indice
      | Nomisura:
      | ldb       R(TIMER).1
      | JBnz      Regolazione
      | End
      | Regolazione:
```

Ad ogni uscita dalla regolazione prima della istruzione End in riscalda, raffres, etc bisogna resettare il rele 1 per poter reimpostare il timer per la regolazione.

```
Sez.C | AbsResb   R(RELE1).1     Reset per reimpostaziobne timer 1
```

Eseguire le modifiche e provare il programma. Adesso la lettura dei dati è più ritardata e anche gli scambi per bande morte piccole sono molto più lenti.Questo tipo di applicazione è solo una dimostrazione a scopo didattico ma la filosofia di programmazione è senz'altro valida . Per una

regolazione ancora migliorata si può implementare una regolazione PID che la macchina la può gestire tranquillamente sia eseguendo calcoli in virgola mobile sia come dati long. In senso più generale non è consigliabile utilizzare un plc per sostituire un solo termoregolatore, ma diventa sicuramente vantaggioso se i loop da controllare sono già due. Segue il listato completo di questa appllicazione.

```
SETPOINT:       EQU     300     ;R(300) contiene il set-point di regolazione
BMORTA  :       EQU     301     ;R(301) range temp.banda morta
TEMP    :       EQU     302     ;R(302) temperatura virgola mobile
MOD     :       EQU     303     ;R(303) memoria per le modifiche
NUMPLC  :       EQU     3       ;identifica memoria impostazione Numero PLC
ACC     :       EQU     9       ;accumulatore per operazioni a BIT
SYSCOM0 :       EQU     24      ;funzione impostazione vel.comunicazione
SYSCOM1 :       EQU     25      ;funzione impostazione vel.comunicazione
ANI     :       EQU     64      ;registro lettura canale analogico 0 e 1
K       :       EQU     400     ;registro uso generale
OUT     :       EQU     20      ;registro delle uscite
FLAG    :       EQU     6       ;flag delle uscite
DAC     :       EQU     68      ;uscita analogica
FILLDSP :       EQU     10      ;funzione visualizzazione messaggi
TIMER   :       EQU     38      ;contatti dei timer
VISPAG  :       EQU     13      ;Funzione di sistema visualizz. Pagina utente
POSXY   :       EQU     4       ;Posiziona sul display il cursore
PUTCH   :       EQU     2       ;visualizza sul display un carattere
WRMSG   :       EQU     5       ;funzione di sistema visualizza una stringa
REFRPAG :       EQU     11 ;funzione rinfresca dati pagina
SPEC0   :       EQU     74      ;registro contatti speciali 0
LAST    :       EQU     206     ;ultima pagina visualizzata
KEY     :       EQU     78      ;memoria tasti
RELE1   :       EQU     46      ;1° blocco 32 rele
MODPAG  :       EQU     14      ;funzione sistema modifica dati pagina
TIMVAL0 :       EQU     215     ;parametri del timer zero
MEDIA   :       EQU     304     ;R(304) contatore media
BTEMP   :       EQU     305     ;R(305) BUFFER PER MEDIA MISURE 16

Mov.lw          #1,R(NUMPLC)    ;imposto apparato come numero 1 sulla rete
Mov.l           #8,R(ACC)       ;preparo per chiamata a funzione di sistema
Sys             #SYSCOM0,R(ACC) ;imposto COM 0 a 57600baud
Mov.l           #0,R(ACC)       ;preparo per chiamata a funzione di sistema
Sys             #SYSCOM1,R(ACC) ;imposto COM 1 a 115200baud
mov.l           #1,R(FLAG)      ;abilito uscite
mov.l           #0,R(TEMP)      ;reset valore temperatura
SETDAC          #0              ;Imposto ani6 come uscita DAC0
Mov.l           #paginastart,R(K) ;punta a messaggi della pagina iniziale
Sys             #FILLDSP,R(K)   ;scrivo messaggi
SetTimer        #1,#50,#0       ;timer 0 a 5 secondi
Start                           ;start per i primi 5 sec
Mov.llb         #1,R(ACC)       ;ACC BIT =1
Stt             #0              ;Abilitazione Timer 0
ldb             R(TIMER).0      ;prendo contatto timer
jbnz            Loop
End
Loop:
mov.l   #0x00640000,R(MOD)  ;divisore impostazione dati ogni 100 cicli
mov.l   #0,R(MEDIA)         ;reset puntatore indice per media misure
SETTIMER        #1,#200,#0  ;Riuso timer 0 timeout impostazione dati
SETTIMER        #1,#16,#1   ;timer 1 controllo regolazione
```

```
SETTIMER        #1,#1,#2            ;timer 2 media della misura
Sub             Paginadefault       ;routine di visualizzazione pagina default
Start                               ;inizio scansione di programma
ldb             R(RELE1).1          ;Rele 1 abilitato
Stt             #1                  ;res in timer 1 abilitazione
abssetb         R(RELE1).1          ;abilito R1 disabilito a uscita regolazione
ldb             R(RELE1).2          ;Rele 2 abilitato
Stt             #2                  ;res in timer 2 abilitazione
abssetb         R(RELE1).2          ;abilito R2 disabilito a uscita media della mis,
ldb             R(SPEC0).0          ;contatto a 1/10 sec
RSedge          #0                  ;in serie a rilevatore di fronte
jbz             Norefresh           ;res=0 non rinfresco
sys             #REFRPAG,R(LAST)    ;rinfresco dati
sub             VisBarra            ;e la barra + stato
Norefresh:
ldb             R(RELE1).0          ;siamo in fase di impostazione
jbz             NoMod               ;no
mov.l           #MOD,R(K)           ;si assegno memoria per modifiche
sys             #MODPAG,R(K)        ;funzione modifica dati
NoMod:
LDB             R(KEY).26           ;Controllo premuto tasto f1
ORB             R(KEY).25           ;parallelo tasto tasto f2
ORB             R(RELE1).0          ;Autointrattenimento con R0
ANDB            !(R(TIMER).0)       ;In Serie NC Timer 0
ANDB            !(R(KEY).30)
STB     R(RELE1).0          ;res su R0
STT             #0                  ;res su Timer dopo 20 secondi reset automatico
ldb             R(RELE1).0          ;R0 attivo un tasto premuto
andb            R(KEY).26           ;quale tasto f1 o f2
RSEDGE          #1                  ;con rilevatore di fronte 1
jbz             NoSetPoint ;res=0 non visualizzo
mov.l           #pagSetPoint,R(K)
sys             #VISPAG,R(K)        ;visualizzo Setpoint
jmp     NoBMorta            ;interblocco su f1premuto e F2 premuto
NoSetPoint:
ldb             R(RELE1).0          ;R0 attivo un tasto premuto
andb            R(KEY).25           ;quale tasto f1 o f2
RSEDGE          #2                  ;con rilevatore di fronte 2
jbz             NoBmorta            ;res=0 non visualizzo
mov.l           #pagBmorta,R(K)
sys             #VISPAG,R(K)        ;visualizzo pagina imp banda Morta
NoBMorta:
LDB             !(R(RELE1).0)       ;se NC R0 pagina default
RSEDGE          #3                  ;fronte visualizzazione pagina default
JBZ             NoRefresh1 ;non è arrivato
SUB             paginadefault       ;pagina default
NoRefresh1:
ldb             R(KEY).28           ;tasto UP +
Orb             R(KEY).29           ;tasto DOWN +
Orb             R(KEY).31           ;tasto ENTER
jbz             NoTimeout
mov.hw          #0,R(TIMVAL0)       ;riassetto valore timer 0 per timeout
NoTimeout:
ldb             R(TIMER).2          ;timer 2 arrivato
jbz             NoMisura
Mov.lw          #0,R(K)             ;si prendo misura lw=0
```

```
Mov.hw          R(ANI),R(K)          ;prendo valore canale analogico 1
Swap.l          R(K)                 ;porto in forma normalizzata lungo R(K)
Mult.l          #1000,R(K) ;R(K)* fondo scala + decimi +centesimi
Div.l           #1024,R(K) ;R(K)/1024 temperatura 0...1000 0..100.0 C°
AbsResb         R(RELE1).2           ;reset rele abilitazione timer 2
mov.l           #BTEMP,R(K+1)        ;Dove metto il valore
Add.l           R(MEDIA),R(K+1)      ;sommo indice
mov.l           R(K),@R(K+1)         ;immagazzino valore nel buffer
inc1.l          R(MEDIA)             ;incremento indice
gtr.lw          #15,R(MEDIA),Nomisura  ;fine indice
mov.l           #15,R(MEDIA)         ;eseguo le media dei 15 valori
mov.l           #BTEMP,R(K+1)        ;puntatore
mov.l           #0,R(K)              ;reset stato registro
ciclomedia:
mov.l           @R(K+1),R(K+2)
add.l           R(K+2),R(K)
inc1.l          R(K+1)               ;incremento puntatore ai dati
djnz            R(MEDIA),ciclomedia  ;loop
DiV.l           #15,R(K)             ;somma dei valori diviso 15
Mov.l           R(K),R(TEMP)         ;salvo temperatura letta
mov.l           #0,R(MEDIA)          ;reset indice
Nomisura:
ldb             R(TIMER).1
jbnz            Regolazione
End
Regolazione:
Mov.l           R(SETPOINT),R(K)     ;prendo setpoint
Subc.l          R(BMORTA),R(K)       ;SETPOINT-BMORTA
Les.l           R(TEMP),R(K),riscalda          ;TEMP<( SETPOINT-BMORTA)
Mov.l           R(SETPOINT),R(K)     ;prendo setpoint
Add.l           R(BMORTA),R(K)       ;SETPOINT+BMORTA
Gtr.l           R(TEMP),R(K),raffres ;TEMP>( SETPOINT+ BMORTA)
ABSRESB         R(OUT).0             ;reset uscita 0
ABSRESB         R(OUT).1             ;reset uscita 1
Mov.lw          #0,R(DAC)            ;reset uscita analogica 0
absresb         R(RELE1).1           ;disabilito regolazione effettuata
End
Riscalda:
ABSSETB         R(OUT).0             ;accendo uscita zero riscaldamento ON
ABSRESB         R(OUT).1             ;Sicurezza uscita uno raffrescamento a OFF
Mov.l           R(SETPOINT),R(K)     ;prendo SETPOINT
Subc.l          R(TEMP),R(K)         ;SETPOINT-TEMP
Sub             CALCOUT ;chiama routine che in base alla diff. da l'uscita
absresb         R(RELE1).1           ;disabilito regolazione effettuata
End
Raffres:                             ;raffrescamento
ABSSETB         R(OUT).1             ;accendo uscita zero raffrescamento ON
ABSRESB         R(OUT).0             ;sicurezza uscita zero riscaldamento a OFF
Mov.l           R(TEMP),R(K)         ;prendo TEMP
Subc.l          R(SETPOINT),R(K)     ; TEMP- SETPOINT
Sub             CALCOUT ;chiama routine che in base alla diff. da l'uscita
absresb         R(RELE1).1           ;disabilito regolazione effettuata
End
Calcout:
Shfl.l    R(K)          ;R(K)=R(K)*2
Shfl.l    R(K)          ;R(K)=R(K)*2 Amplificazione =4
```

```
GTR.lw          #255,R(K),Noclip     ;se R(K) >255 allora R(K)=255
Mov.l           #255,R(K)            ;max out
Noclip:
Mov.lw          R(K),R(DAC)          ;imposto uscita analogica zero
Ret
Paginadefault:
Mov.l           #paginaNorm,R(K)     ;prendo indirizzo pagina
Sys             #VISPAG,R(K)         ;e visualizzo
Sub             VisBarra             ;segnale uscita
ret
VisBarra:                            ;barra di 8 quadrati max
ldb             R(RELE1).0           ;R0 attivo se impostazione in corso
jbz      VisBarra1           ;
ret
Visbarra1:
mov.lw          #08,R(K)             ;pos x=8
mov.lw          #0,R(K+1)            ;pos y=0
sys             #POSXY,R(K)          ;in posizione disegno barra
Mov.l           #MsgCanc,R(K)        ;cancello la barra
Sys             #WRMSG,R(K)          ;
mov.lw          #08,R(K)             ;pos x=8
mov.lw          #0,R(K+1)            ;pos y=0
sys             #POSXY,R(K)          ;in posizione disegno barra
mov.lw          R(DAC),R(K+2)        ;prendo uscita segnale analogico
mov.hw          #0,R(K+2)            ;azzero parte alta
div.l           #32,R(K+2) ;da 0 a 7
eql.lw          #0,R(K+2),finebarra  ;res 0 niente vis
ContBarra:
mov.l           #4,R(K+1)            ;carattere  di sistema quadrato
sys             #PUTCH,R(K+1)        ;scrivo
djnz            R(K+2),ContBarra     ;
finebarra:
Sub             RisRaf               ;visualizzo stato riscaldamento raffrescamento
Ret
RisRaf:
Mov.lw          #8,R(K)              ;imposto X=8
Mov.lw          #1,R(K+1)            ;imposto Y=1
Sys             #POSXY,R(K)          ;vado alla posizione XY per disegno barra
Ldb             R(OUT).0             ;richiesta caldo
Jbz             Visfreddo            ;non visualizzo msg per caldo
Mov.l           #MsgCaldo,R(K)       ;visaulizzo messaggio caldo
Sys             #WRMSG,R(K)
Ret
Visfreddo:
Ldb             R(OUT).1             ;richiesta freddo
Jbz             VisSet
Mov.l           #MsgFreddo,R(K)      ;visaulizzo messaggio caldo
Sys             #WRMSG,R(K)
Ret
VisSet:
Mov.l           #MsgStop,R(K)        ;visaulizzo messaggio caldo
Sys             #WRMSG,R(K)
Ret
DL              0                                ;allineamento memoria come lungo
paginastart:
DL              Str1
```

```
DL                  Str2
Str1:
DM                  "TERMOTRONIC INC."
Str2:
DM                  "V.1.00        "
DL          0                                   ;allineamento
PaginaNorm:                                     ;definisco la pagina di visualizzazione normale
DL          Mp1                                 ;messaggio riga 1
DL          Mp2                                 ;messaggio riga 2
DL          Var1                                ;puntatore struttura descrittore variabile 1
DL          Var2                                ;puntatore struttura descrittore variabile 2
DL          0                                   ;fine struttura
Mp1:
DM          "S(   )         "                   ;mesg riga 1 (16 char)
Mp2:
DM          "               "                   ;mesg riga 2 (16 char)
Var1:
DL          R(SETPOINT)                         ;indirizzo 1° variabile
DL          Limit1p1                            ;limite per questa variabile
Var2:
DL          R(TEMP)                             ;indirizzo 1° variabile
DL          Limit2p1                            ;limite per questa variabile
Limit1p1:
DB          32,22,2,0                           ;Vis. 99.9 ptr a long *lock X=2 Y=0
DL          0,1000                              ;min=0 max=1000
Limit2p1:
DB          32,22,2,1                           ;Vis. 99.9 ptr a long *lock X=2 Y=1
DL          0,1000                              ;min=0 max=1000
MsgCaldo:
DM          "---->>>>",0                        ;riscaldamento
MsgFreddo:
DM          "<<<<----",0                        ;raffrescamento
MsgStop:
DM          "<<<<>>>>",0                        ;sulla banda morta
MsgCanc:
DM          "       ",0                         ;cancella la barra
DL          0                                   ;allineamneto
pagSetPoint:
DL      msgpag2r1
DL      msgpag2r2
DL      varpag2V1
DL      0               ;FINE PAGINA
msgpag2r1:
DM      "SetPoint       ",0
msgpag2r2:
DM      "Regolazione    ",0
DL    0
varpag2V1:
DL      r(300)
DL      limit2Var1
limit2Var1:
DB      32,6,11,0
DL      0,999
pagBMorta:
DL      msgpag3r1
DL      msgpag3r2
```

```
DL      varpag3V1
DL      0
msgpag3r1:
DM      "B. Morta      ",0
msgpag3r2:
DM      "Regolazione   ",0
DL    0
varpag3V1:
DL      r(301)
DL      limit3Var1
limit3Var1:
DB      32,6,11,0
DL      0,50
```

Nel disco dove sono i programmi sono presenti tutti i listati del programma dalla lezione 1 alla lezione 7.

GARANZIA

Il MicroPLC D1208 è garantito contro difetti di funzionamento Hardware per un periodo di 6 mesi dalla data di acquisto della fattura.
La C&P di Coppi Angelo si impegna esclusivamente alla riparazione o alla sostituzione gratuita di MicroPLC in cui sia provato un difetto in un normale uso del prodotto.La sostituzione in garanzia è a totale discrezione della C&P di Coppi Angelo.

CONDIZIONI DI GARANZIA

L'utente perde il diritto di garanzia:
A)se dovessero essere riscontrate riparazioni da persone non autorizzate
B)se dovessero venire riscontrate manomissioni
la garanzia non si applica nei seguenti casi:
A)avarie o rotture causate dal trasporto
B)***errata o cattiva installazione del prodotto***
C)insufficienza o anomalia degli impianti elettrici
D)trascuratezza negligenza o incapacità nell'uso del prodotto
E)l'Utente non è il primo acquirente
F)Cause non dipendenti dalla **C&P**
G)Interventi per vizi presunti o per verifiche di comodo
H)la **C&P di Coppi Angelo** non è responsabile dei danni derivati all'Utente da un mancato o imperfetto funzionamento del prodotto o causati alle cose da un funzionamento difettoso
I)Difetti causati da una programmazione insufficiente o errata

L'assistenza tecnica si intende franco nostra sede.
Trasporto a carico dell'acquirente