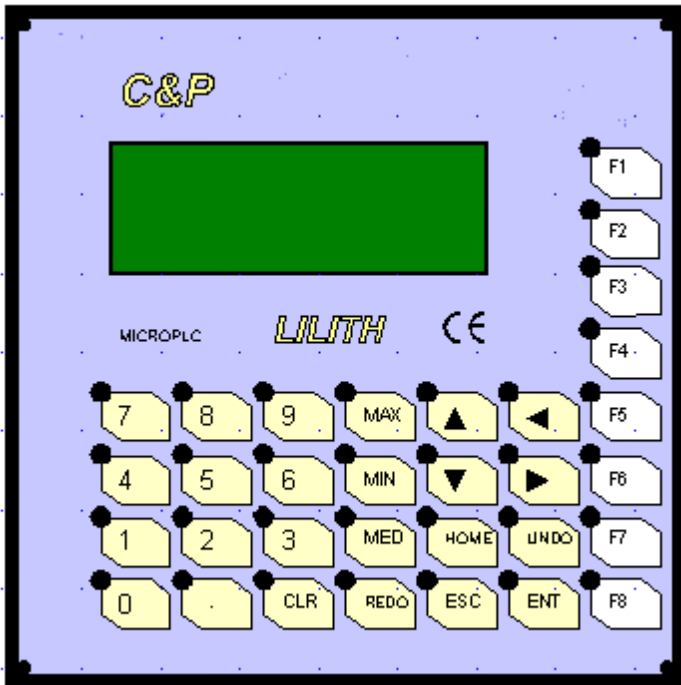


MANUALE D'USO
MICROPLC
LILITH

CONTROLORE A LOGICA



PROGRAMMABILE

Manuale V.1.0
Settembre 1999

C&P di Coppi Angelo.

ELETRONICA 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

- ♣) Tutte le informazioni presenti in questo manuale d'uso sono proprietà della ditta ***C&P di Coppi Angelo***
- ♣) Questo documento, o qualsiasi parte di esso, non essere copiato , riprodotto o ridistribuito in qualsiasi forma.
- ♣) Il software in dotazione non può essere copiato
riprodotto
o ridistribuito in qualsiasi forma . Il software è di solo utilizzo personale.

Questo manuale fa riferimento al MicroPLC Mod. LILITH 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 LILITH.

WINDOWS è un marchio registrato da Microsoft Corporation

• **INTRODUZIONE**

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

Contenuto della confezione:

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

• PREFAZIONE

Il **MicroPLC Lilith** è la evoluzione del microPLC merlino per applicazioni fronte quadro ove sia richiesto la impostazione di parametri a mezzo tastiera e visibilità mioglorata dei dati sul display. Per questo il microPLC lilith mantiene piena compatibilità software con il PLC merlino. Migliora solo ove richiesto prestazioni e funzioni.

• Caratteristiche tecniche:

Caratteristiche Generali

Tensione di alimentazione	12-30 VDC. (protezione contro l'inversione di polarità). 9..18 VAC
Corrente di alimentazione	400 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	0..5VDC LIVELLO TTL	
	Tensione di ON	4VDC
	Tensione di OFF	2VDC
Resistenza di ingresso	Circa 100KΩ	

Caratteristiche delle uscite

Tipo di uscita	Collettore Aperto
Massimo Carico	30 ma 24VDC

Caratteristiche degli ingressi analogici

Range	0-5 VDC
Impedenza ingresso	>=100koms
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 assoluto oppure 2 contatori separati
Frequenza Max impulsi	500 khz
Sampling	Definibile dall'utente da 25hz a 5Khz
Impedenza ingresso	1Kohms livello TTL

Caratteristiche delle porte di comunicazione

Standard	RS-485 a 4 fili differenziale
Massima distanza	1Km su cavo twisted
Velocità di comunicazione	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 μ P, ripartenza automatica

Caratteristiche generali del sistema

Software	<p>Linguaggio Interpretato appositamente studiato per realizzare 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)</p> <p>Implementazione di 1500 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.</p> <p>Possibilità 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</p>
-----------------	---

• INSTALLAZIONE HARDWARE

Descrizione dei Connettori

Il MicroPLC **LILITH** Viene fornito in o come scheda nuda o su un contenitore modulare standard 144x144 frontale. In entrambi i casi i morsetti per le connessioni sono estraibili.

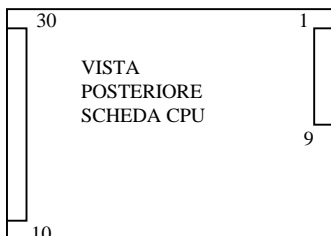


TABELLA CONNESSIONI LATO LCD:

N°	DESCRIZIONE	N°	DESCRIZIONE
1	ALIMENTAZIONE	30	INPUT ANALOGICO N° 8
2	SCHERMO DI TERRA	29	INPUT ANALOGICO N° 7
3	ALIMENTAZIONE	28	INPUT ANALOGICO N° 6
4	USCITA ANALOGICA 0	27	INPUT ANALOGICO N° 5
5	USCITA ANALOGICA 1	26	INPUT ANALOGICO N° 4
6	COM 2 RX +	25	INPUT ANALOGICO N° 3
7	COM 2 RX -	24	INPUT ANALOGICO N° 2
8	COM 2 TX -	23	INPUT ANALOGICO N° 1
9	COM 2 TX +	22	GND
		21	ENCODER INPUT A
		20	GND
		19	ENCODER INPUT B
		18	COM 1 RX +
		17	COM 1 RX -
		16	COM 1 TX -
		15	COM 1 TX +
		14	COM 0 RX +
		13	COM 0 RX -
		12	COM 0 TX -
		11	COM 0 TX +
		10	GND

Display LCD

All'accensione 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 pagine definite nel sistema operativo. Queste pagine sono organizzate in menu. Selezionare con i tasti su e giù il menu interessato poi premere enter. Comparire la pagina interessata.

Premendo enter si passa da una variabile all'altra. Qualora si voglia modificare un qualsiasi valore lo si può impostare a mezzo dei tasti numerici e poi premere enter per confermare. Premendo esc si esce dalla modifica della variabile. In questa fase i tasti F1..F8 non hanno nessuna funzionalità.

Istallazione

Connettere ai morsetti di alimentazione N° 1 e N°3 una tensione o continua da **12VDC a 30 VDC MAX**

O alternata da **9 VAC a 20 VAC** con una corrente di almeno 400 ma separata galvanicamente dal resto del circuito. Collegare il Polo 2 a terra per poter scaricare eventuali disturbi 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

Tutti 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.

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.

Connessioni verso Personal Computer o PLC master

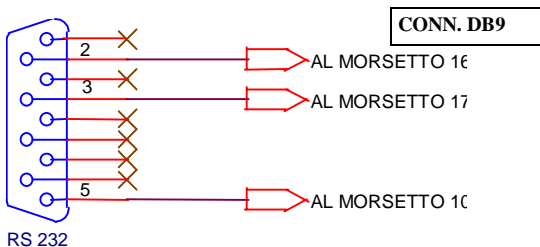
Collegare i cavi di connessione seriale nel seguente modo e controllare la disposizione dei segnali sul vostro convertitore RS-232 >> RS-485



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

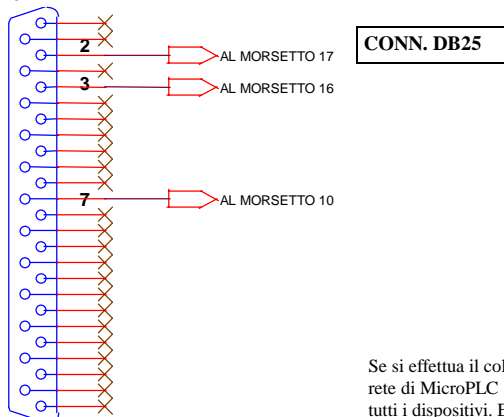
Connessioni in standar RS-232

Il *microPLC lilith* 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)



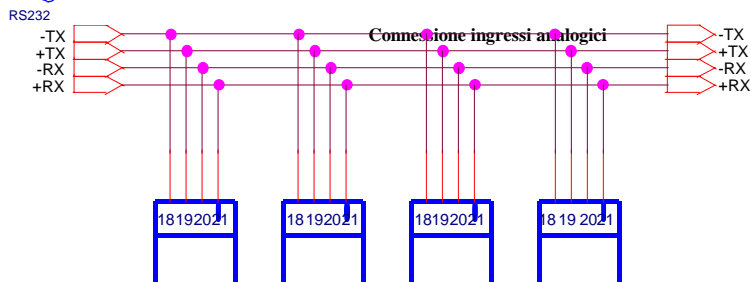
DB9	Morsetto
2	16
3	17
5	10

RS 232



DB25	Morsetto
2	17
3	16
7	10

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



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.

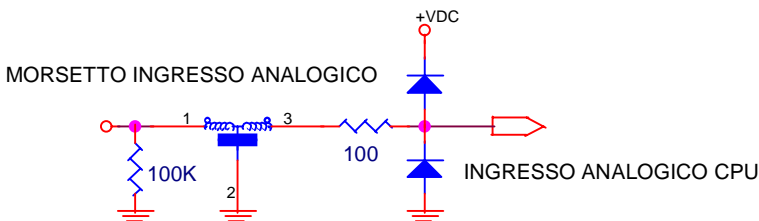
E' buona norma generalmente mettere degli isolatori di segnale agli ingressi analogici, con tali dispositivi non sono necessarie altri accorgimenti,visto che eventuali anomalie del segnale sono a carico di questi dispositivi.Nel caso che si intenda collegare direttamente un segnale all'ingresso analogico bisogna assicurarsi che in alcun modo la tensione di ingresso possa avere valori superiori ai 6 volt o inferiori a -0,5Volt

ATTENZIONE

Gli ingressi analogici sono canali ad alta impedenza($\geq 100K$) 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 installare un separatore galvanico di segnale,almeno procedere alla protezione del canale analogico tramite soppressori di segnale a 5VDC.

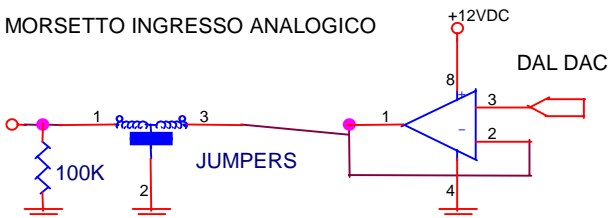
Segue schema di ingresso dei primi tre canali analogici:



Connessioni uscite analogiche.

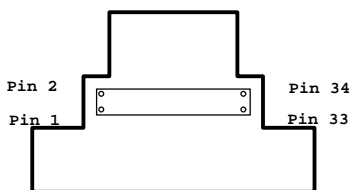
Il *microPLC Lilith* mette a disposizione del programmatore anche due uscite analogiche a 8 bit,quando si imposta l' uscita analogica 0 (1) si perde il canale analogico 6 (7). Qualora non si usi le uscite analogiche ai capi di queste uscite è presente copia del segnale presente sugli ingressi 6(7)

Segue schema di uscita canali analogici:



Il connettore a 34 poli serve per le espansioni. Le connessioni sono le seguenti:

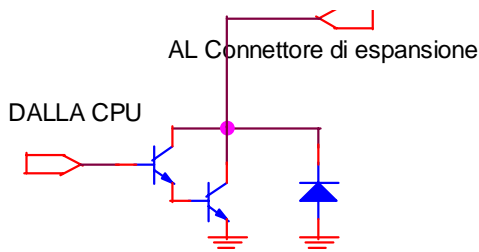
Pin	Qualifica
1	N.C.
2	N.C.
3	N.C.
4	N.C.
5	Uscita a collettore aperto n°9
6	N.C.
7	Uscita a collettore aperto n°10
8	Ingresso TTL n°16
9	Uscita a collettore aperto n°11
10	Ingresso TTL n°15
11	Uscita a collettore aperto n°12
12	Ingresso TTL n°14
13	Uscita a collettore aperto n°13
14	Ingresso TTL n°13
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	N.C.
26	Ingresso TTL n°18
27	N.C.
28	N.C.
29	N.C.
30	N.C.
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 aperto. 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".

Dip-switch:

Sul lato posteriore è presente un dip switch a 8 posizioni , du ci solo 5 sono attive , con le seguenti funzioni

<i>Dip switch</i>	
1	ON: forza le impostazioni di comunicazione seriale del PLC al default: PLC NUMERO 1 ,COM 0 e COM1 115Kbaud , COM 2 9600baud
2	ON: all'accensione imposta per il Test della Tastiera
3	ON : all'accensione imposta per il reset del PLC
4	DISPONIBILE
5	OFF: COM 2 impostata su Funzioni SYS ascii ricezione trasmissione ON : COM 2 impostata su protocollo MODBUS ASCII
6	Se tutti e 3 questi dip son ON allora il PLC non esegue alcun test all'accensione, non visualiz
7	za nessuna scritta e si mettein RUN anche se la memoria interna è impostata in stop
8	(funzionamento anche senza batteria tampone)

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 lilith* è un PLC basato su un microcontrollore a 16/32 bit dell'ultima generazione in tecnologia flash. Ottimizzato per eseguire una istruzione per ogni ciclo di clock 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(21,7 Mhz, 40 ns istruzione , istruzione maxima 2 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 possono essere così suddivisi:

- Processo principale
- Comunicazione Seriale canale 0, 1 e 2
- Timer di sistema
- Gestione Encoder
- Gestione Errori Hardware
- Gestione Interrupt Utente
- Programmazione Memoria Flash

Questi processi sono in parte gestiti tramite interrupt o tramite DMA automatico, 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,tutte le istruzioni sono chiamate a funzioni ottimizzate e in genere non occupano più di 1 microsecondo 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.

- **Linguaggio Assembler**

Il *microPLC lilith* mette a disposizione per il programmatore 1500 registri generali a 32 bit come memorie di sistema, tutti accessibili, 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 è quella a 32 bit, tale dimensione è definita "long" e viene identificata con la estensione ".l". Ogni dato long è costituito da due dati interi che sono definiti come "low word" per i 16 Bit meno significativi e "high word" per i 16 Bit più significativi. Questi dati sono rispettivamente identificati dall'estensione ".lw" e ".hw". Il dato a 16 bit low word è 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 più significativi sono definiti come "low high byte" e vengono individuati dalla estensione ".lhb". Anche il dato a 16 bit high word è 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 ".hbw", mentre gli 8 bit più significativi sono definiti come "high high byte" e vengono individuati dalla estensione ".hhb".

Riassumendo, possiamo schematizzare la seguente tabella:

31				long				0			
31				high word				0			
31				hbw				0			
16				15				low word			
16				15				lhb			
8				7				llb			
8				7				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 più 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 è necessario un registro con la funzione d'accumulatore per le operazioni a bit questo è 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 più importanti

Sono tre:

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

Inoltre poiché un'istruzione ha fino a massimo di tre dati si possono avere una o più 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 spiazamento sul programma dell'indirizzo della stringa sommato a 0x18000. L'area di memoria ram del sistema è di 8Kbyte 2000 long .Quest'area è solo in parte dedicata ai registri di sistema(1500)la restante area è dedicata allo stack e alle variabili nascoste del sistema operativo. 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 0xffe000 sino a 0xffff7f.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 0xffe000+n*4.

Mapa del sistema operativo:

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

Sistema operativo	
RAM	Address
Registro 0	0xFFE000
Registro 1500	0xFFFF7f

***ATTENZIONE** questa è l'unica differenza rispetto al microPLC Merlino:ma se ne occupa in modo trasparente il compilatore assembler.

Tramite le istruzioni di I/O si può 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 H8S2357F20 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 (*.asm) ha poche direttive che
sono così riassunte:

DB	Definisce un byte nell'area programma es: DB 10,0xfe,0b11110101
DW	Le virgole indicano una successiva definizione Definisce un intero a 16 bit nell'area programma es: DW 10,0xfe45,0b11110101001111
DM	Le virgole indicano una successiva definizione Definisce una stringa di caratteri DM "Hello World",10,13
DL	Le virgole indicano o una successiva dichiarazione o la presenza di una definizione di byte 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 123456789,0x1ffffff,r(300),r(300)+2 DL 0f1234.2345
EQU	Le virgole indicano una successiva dichiarazione. 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" Etichette	Prima di elaborare il file corrente ingloba il file <def> 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: jnz 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 lilith* 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 semplicissimo, appoggiare su un rele interno questo contatto e usare i contatti del rele 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 e deve essere utilizzata ogni qualvolta si usa contatti che possono essere variabili (eventi asincroni) 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.

*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

Dimensione formato

HHB	HLB	LHB	LLB	Byte
HW		LW		Word
L				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:

Prima dell' eseg. della istruz. ADD.LLB #100, R(300) Dopo l'esecuzione della istruz.	HHB HLB LHB LLB	<table border="1"> <tbody> <tr> <td>x</td> <td>x</td> <td>x</td> <td>100</td> <td>Contenuto R(300)</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>x</td> <td>x</td> <td>x</td> <td>200</td> <td>Contenuto R(300)</td> </tr> </tbody> </table>	x	x	x	100	Contenuto R(300)						x	x	x	200	Contenuto R(300)
x	x	x	100	Contenuto R(300)													
x	x	x	200	Contenuto R(300)													
Prima dell' eseg. della istruz. ADD.HW #01200, R(320) Dopo l'esecuzione della istruz.	HW LW	<table border="1"> <tbody> <tr> <td>1000</td> <td>X</td> <td>Contenuto R(320)</td> </tr> <tr> <td></td> <td></td> <td></td> </tr> <tr> <td>2200</td> <td>X</td> <td>Contenuto R(320)</td> </tr> </tbody> </table>	1000	X	Contenuto R(320)				2200	X	Contenuto R(320)						
1000	X	Contenuto R(320)															
2200	X	Contenuto R(320)															
Prima dell' eseg. della istruz. ADD.L R(310), R(320) Dopo l'esecuzione della istruz.	R(310) R(320)	<table border="1"> <tbody> <tr> <td>1800</td> <td>3200</td> </tr> <tr> <td></td> <td></td> </tr> <tr> <td>1800</td> <td>5000</td> </tr> </tbody> </table>	1800	3200			1800	5000									
1800	3200																
1800	5000																
Prima dell' eseg. della istruz. Contenuto di R(300) ADD.L R(310), @R(300) Dopo l'esecuzione della istruz.	R(310) R(320)	<table border="1"> <tbody> <tr> <td>135000</td> <td>1000</td> </tr> <tr> <td>320</td> <td></td> </tr> <tr> <td></td> <td></td> </tr> <tr> <td>135000</td> <td>136000</td> </tr> </tbody> </table>	135000	1000	320				135000	136000							
135000	1000																
320																	
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

Dimensione formato

HHB	HLB	LHB	LLB	
HW	LW			Byte
L				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.

ESEMPLI:

Prima dell' l' esec. della istruz.

AND.LLB #0xF0, R(300)

Dopo l'esecuzione della istruz.

HHB	HLB	LHB	LLB	
x	x	x	0F	Contenuto R(300)
x	x	x	00	Contenuto R(300)
HW		LW		
FFFF		X		Contenuto R(320)
FFF0		X		Contenuto R(320)

Prima dell' l' esec. della istruz.

AND.HW #0xFFFF0, R(320)

Dopo l'esecuzione della istruz.

R(310)	R(320)
0FF0FF	FFFFFF
0FF0FF	0FF0FF

Prima dell' l' esec. della istruz.

AND.L R(310), R(320)

Dopo l'esecuzione della istruz.

R(310)	R(320)
00000F	FFFFFF
320	
00000F	00000F

*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' eseg. della istruz.
DIV.HW #01200, R(320)
Dopo l'esecuzione della istruz.
Registro R(0)

4800	x	Contenuto R(320)
4	x	Contenuto R(320)
0	0	Nessun resto

Prima dell' eseg. della istruz.
DIV.L R(310), R(320)
Dopo l'esecuzione della istruz.
Contenuto di R(320)
Registro R(0)

R(310)	R(320)	
16	7800	
HW	LW	
0	487	
0	5	Resto

Prima dell' eseg. della istruz.
Contenuto di R(300)
DIV.L R(310), @R(300)
Dopo l'esecuzione della istruz.
Registro R(0)

R(310)	R(320)	
1000	13400	
320		
0	13	
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.

ESEMPLI:

;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)
dec.l.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
dec.l.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 più 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. EXT.W R(320)	x	x	x	01010000	Contenuto R(320)
Dopo l'esecuzione della istruz.	x			0000000001010000	Contenuto R(320)

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

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

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.LR(300),R(301),LABEL
GTR.*	#Valore	@R(nnn)	Label	GTR.HW 1500 ,R(300),LABEL
GTR.*	R(nnn)	@R(nnn)	Label	GTR.LR(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.

ESEMPLI:

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:

```


*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.

ESEMPLI:

```

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.

ESEMPLI:

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:

*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
		LW	
L			

Byte

Word

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

Dimensione formato

HHB	HLB	LHB	LLB	Byte
HW		LW		Word
L				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.

ESEMPLI:

	HHB	HLB	LHB	LLB	
MOV.LLB #100, R(300)	x	x	x	100	Contenuto R(300)
MOV.HLB #100, R(300)	x	100	x	x	Contenuto R(300)

	HW		LW		
MOV.LW #0xFFFF, R(320)	x		FFFF		Contenuto R(320)

	300		41500		
MOV.HW #41500, @R(450)					Contenuto R(450)
Dopo l'esecuzione della istruz.			x		Contenuto R(300)

	R(301)				R(300)			
MOV.HHB R(301),R(300)	AA	55	FF	22	43	x	x	x
Dopo l'esecuzione della istruz	AA	55	FF	22	AA	x	x	x

	R(310)		R(320)		R(315)	
MOV.L R(310),@R(320)	0x1200FFFF		315		0xFFFFFFFF	
Dopo l'esecuzione della istruz.	0x1200FFFF		315		0x1200FFFF	

	R(400)		R(401)		R(870)	
MOV.L @R(400),R(870)	401		0x22334455		0xFFFFFFFF	
Dopo l'esecuzione della istruz.	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.

ESEMPLI:

Prima dell'exec. 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	x	100	Contenuto R(300)

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

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

Prima dell'exec. 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

Dimensione formato

HHB	HLB	LHB	LLB	Byte
HW		LW		Word
L				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.

ESEMPLI:

Prima dell' l'esec. della istruz.
NEG.LLB R(300)
Dopo l'esecuzione della istruz.

HHB	HLB	LHB	LLB	
x	x	x	7Fh	Contenuto R(300)
x	x	x	81h	Contenuto R(300)

Prima dell' l'esec. della istruz.
NEG.HW R(320)
Dopo l'esecuzione della istruz.

HW	LW	
3FFFh	x	Contenuto R(320)
C001h	x	Contenuto R(320)

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

R(310)	R(320)
320	80000h
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

Dimensione formato

HHB	HLB	LHB	LLB	Byte
		LW		Word
L				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.

ESEMPLI:

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

Dimensione formato

HHB	HLB	LHB	LLB	
HW		LW		Byte
L				Word
				Long

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.

ESEMPLI:

	HHB	HLB	LHB	LLB	
Prima dell' l'esec. della istruz. NOT.LLB R(300)	x	x	x	00h	Contenuto R(300)
Dopo l'esecuzione della istruz.	x	x	x	FFh	Contenuto R(300)
	HW	LW			
Prima dell' l'esec. Della istruz. NOT.HW R(320)	AA55h	x			Contenuto R(320)
Dopo l'esecuzione della istruz.	55AAh	x			Contenuto R(320)
	R(310)	R(320)			
Prima dell' l'esec. Della istruz. NOT.L @ R(310)	320	000FFFFh			
Dopo l'esecuzione della istruz.	320	FFF0000h			

*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

Dimensione formato

HHB	HLB	LHB	LLB	
HW	LW			Byte
L				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' eseg. della istruz. OR.LLB #0xF0, R(300)	x	x	x	0F	Contenuto R(300)
Dopo l'esecuzione della istruz.	x	x	x	FF	Contenuto R(300)

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

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

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

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
 BIT ACC R(009).0
 Dopo l'esec. Della istruz.

CONTENUTO BITSTACK = 1
ACCUMULATORE R(009).0 = 0
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.

ESEMPLI:

Prima dell' l'esec. della istruz.

POP.LW R(320)

Cont. WORD STACK

Dopo l'esecuzione della istruz.

HW	LW	
4800	x	Contenuto R(320)
5200		
4800	5200	Contenuto R(320)

Prima dell' l'esec. della istruz.

Cont. LONG STACK

POP.L @R(350)

Dopo l'esecuzione della istruz.

Contenuto di R(320)

R(350)	R(320)
320	7800
137000	
HW	LW
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(mnn) , 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

ESEMPLI:

POPB
 BIT ACC R(009).0
 Dopo l'exec. Della istruz.

CONTENUTO BIT 0 STACK = 1
ACCUMULATORE R(009).0 = 0
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

Dimensione formato

HW	LW	Word
L		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:

Prima dell' l'esec. della istruz.
PUSH.LW R(320)
Cont. WORD STACK
Dopo l'esecuzione della istruz.

HW	LW	
x	4800	Contenuto R(320)
xxxxxxxxxxxxxxxx		
4800		Contenuto STACK

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

R(350)	R(320)
320	7800
xxxxxxxxxxxxxxxx	
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

Dopo l'esec. Della istruz.

Cima dello STACK

ACCUMULATORE R(009).0 = 1
VALORE A 32 BIT DEL REGISTRO R(009)

*Leggi da un indirizzo assoluto***RD**

Istruzione	Sorgente	Destinazione	Esempio
RD.*	@Valore	R(nnn)	RD.B @0xFFE10,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 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 azzera 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

Dimensione formato

HHB	HLB	LHB	LLB	Byte
HW		LW		Word
L				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.

ESEMPLI:

	HHB	HLB	LHB	LLB	
Prima dell' l'esec. della istruz. ROTL.LLB R(300)	x	x	x	80	Contenuto R(300)
Dopo l'esecuzione della istruz.	x	x	x	01	Contenuto R(300)
	HW		LW		
Prima dell' l'esec. della istruz. ROTL.HW R(320)	00000001		x		Contenuto R(320)
Dopo l'esecuzione della istruz.	00000010		x		Contenuto R(320)
	HW		LW		
Prima dell' l'esec. della istruz. ROTL.LW R(310)	x		10000000		Contenuto R(310)
Dopo l'esecuzione della istruz.	x		00000001		Contenuto R(310)
Prima dell' l'esec. della istruz. Contenuto di R(300) ROTL.L @R(300)	1000100010001000				Contenuto R(320)
Dopo l'esecuzione della istruz.	320				Contenuto R(300)
	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	
HW		LW		Byte
L				Word
				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.

ESEMPLI:

	HHB	HLB	LHB	LLB	
Prima dell' eseg. della istruz. ROTR.LLB R(300)	x	x	x	01	Contenuto R(300)
Dopo l'esecuzione della istruz.	x	x	x	80	Contenuto R(300)
	HW		LW		
Prima dell' eseg. della istruz. ROTR.HW R(320)	00000010		x		Contenuto R(320)
Dopo l'esecuzione della istruz.	00000001		x		Contenuto R(320)
	HW		LW		
Prima dell' eseg. della istruz. ROTR.LW R(310)	x		00000001		Contenuto R(310)
Dopo l'esecuzione della istruz.	x		10000000		Contenuto R(310)
Prima dell' eseg. della istruz. Contenuto di R(300) ROTR.L @R(300)	0001000100010001				Contenuto R(320)
Dopo l'esecuzione della istruz.	320				Contenuto R(300)
	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
```


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:	#VALORE di
	1 START su HH-MM	LHB: va da 0 a 7 e
	2 START su HH-MM STOP su HH-MM	definisce il numero
	3 START su ME-GG-HH-MM	di orologio
	4 START su ME-GG STOP su ME-GG	selezionato.
	5 START su GG giorno della settimana	
	6 START su GG giorno settimana stop GG settimana	

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 modulo 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)
```


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 cortocircuitare 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 è aggiornato tenendo conto del segno (senso 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 precedente.

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 lettura 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 una 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 risultato 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.

ESEMPLI:

```
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:	R(nnn) =0-900	
	0 1/10 DI SECONDO		
	1 SECONDI		
	2 MINUTI		
	3 ORE		
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 inseribile)
- 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

Dimensione formato

HHB	HLB	LHB	LLB	Byte
HW		LW		Word
L				Long

Descrizione:

Questa istruzione esegue uno spostamento a sinistra di un bit nel registro selezionato.

Il bit piú 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.

ESEMPLI:

	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

Dimensione formato

HHB	HLB	LHB	LLB	Byte
HW		LW		Word
L				Long

Descrizione:

Questa istruzione esegue uno 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.

ESEMPLI:

	HHB	HLB	LHB	LLB	
Prima dell' esecuzione della istruzione SHFR.LLB R(300)	x	x	x	80	Contenuto R(300)
Dopo l'esecuzione della istruzione	x	x	x	40	Contenuto R(300)
	HW		LW		
Prima dell' esecuzione della istruzione SHFR.HW R(320)	00000101		x		Contenuto R(320)
Dopo l'esecuzione della istruzione	00000010		x		Contenuto R(320)
	HW		LW		
Prima dell' esecuzione della istruzione SHFR.LW R(310)	x	11000000			Contenuto R(310)
Dopo l'esecuzione della istruzione	x	01100000			Contenuto R(310)
Prima dell' esecuzione della istruzione SHFR.L @R(300)	1000100010001000				Contenuto R(320)
Contenuto di R(300)	320				Contenuto R(300)
Dopo l'esecuzione della istruzione SHFR.L @R(300)	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).

ESEMPLI:

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'accumulatore = 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

Prima dell'esec della istruz

Dopo l'esec della istruz.

ACCUMULATORE R(009).0 = 1
Contatore N° 1 = xxxx
Contatore N°1= xxxx+1

STC R(301)

Contenuto del registro R(301)

Prima dell'esec della istruz

Dopo l'esec. Della istruz.

ACCUMULATORE R(009).0 = 1
23
Contatore N° 23 = xxxx
Contatore N° 23 = xxxx+1

STC @R(300)

Contenuto del registro R(300)

Prima dell'esec della istruz

Dopo l'esec. Della istruz.

ACCUMULATORE R(009).0 = 0
1
Contatore N° 2 = xxxx
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'accumulatore = 1 l'orologio impostato dalla istruzione SETCLOCK si avvierà con la modalità prevista.

ESEMPI:

STK #0

Prima dell'esec della istruz

Dopo l'esec della istruz.

ACCUMULATORE R(009).0 = 1
Orologio N° 1 disabilitato
Orologio N° 1 abilitato

STK R(301)

Contenuto del registro R(301)

Prima dell'esec della istruz

Dopo l'esec. Della istruz.

ACCUMULATORE R(009).0 = 1
7
Orologio N° 7 disabilitato STOP
Orologio N° 7 abilitato RUN

STK @R(300)

Contenuto del registro R(300)

Prima dell'esec della istruz

Dopo l'esec. Della istruz.

ACCUMULATORE R(009).0 = 0
1
Se Orologio N°2 abilitato RUN, altrimenti STOP
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'accumulatore = 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.

ESEMPLI:

```

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)
INCL 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'exec. 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'exec. 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'exec. della istruz.
SUBC.L R(310), R(320)
Dopo l'esecuzione della istruz.

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

Prima dell'exec. 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

ESEMPLI:

Prima dell' l'esec. della istruz.

SWAP.LW R(300)

Dopo l'esecuzione della istruz.

HHB HLB LHB LLB

x	x	35	200	Contenuto R(300)
x	x	200	35	Contenuto R(300)

Prima dell' l'esec. della istruz.

SWAP.L R(320)

Dopo l'esecuzione della istruz.

HW LW

2200	4300	Contenuto R(320)
4300	2200	Contenuto R(320)

Prima dell' l'esec. della istruz.

SWAP.L @R(400)

Dopo l'esecuzione della istruz.

HW LW

1800	4000	R(300)
R(400)=300		
4000	1800	R(300)

Prima dell' l'esec. della istruz.

Contenuto di R(300)

SWAP.HW @R(300)

Dopo l'esecuzione della istruz.

HHB HLB LHB LLB

50	32	x	x	R(320)
320				
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(mnn)	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).

ESEMPLI:

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)

*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	
HHB				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' eseg. della istruz. XOR.LLB #0xFF, R(300)	x	x	x	0F	Contenuto R(300)
Dopo l'esecuzione della istruz.	x	x	x	F0	Contenuto R(300)

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

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

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

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
 BIT ACC R(009).0
 Dopo l'esec. Della istruz.

CONTENUTO BITSTACK = 1
ACCUMULATORE R(009).0 = 0
ACCUMULATORE R(009).0 = 1

- **COMUNICAZIONE SERIALE**

La comunicazione seriale del **MicroPLC Lilith** 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 Lilith** ha 3 porte di comunicazione seriale **Com 0** e **Com 1** e **Com 2**. Con il PLC in STOP, macchina queste porte sono impostate:

Com 0	57600 baud 8 dati 1 Stop Nessuna parità
Com 1	115200 baud 8 dati 1 Stop Nessuna parità
Com 2	9600 baud 8 dati 1 Stop Nessuna parità

Ad ogni **MicroPLC Lilith** 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** o la **Com 2** (Protocollo MODBUS). Inoltre ogni **MicroPLC Lilith** funziona 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 DI208**, oltre naturalmente ai **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 6 slave. In questa tipologia si può direttamente comandare altri 40 I/O senza nessuna aggiunta hardware oltre i 6 slave. Per default lo slave numero 1 deve essere (se richiesto) una **EXP8IO** che funziona come input-output per i primi 8 I/O del PLC. Considerando inoltre la presenza della interfaccia di comunicazione RS-485 si può distribuire la rete in modalità a stella con un raggio di 1 Km. La porta di comunicazione **Com 1** e **2** funzionano solo da slave la porta **Com 0** può funzionare anche da master.

- **protocollo di comunicazione:**

Ogni **MicroPLC Lilith** 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 identico al **MicroPLC Merlino** e uso lo stesso modo di formattazione dei messaggi. Lo scambio di messaggi tra un computer di supervisione o tra **MicroPLC lilith** 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	
Carattere di START alla checksum esclusa.	

***Riguardo al funzionamento del protocollo per la COM 2 (ascii, modbus etc) vedere capitolo seguente**

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 <i>MicroPLC Merlino</i> numero 1
0x05	lunghezza messaggio 5 bytes
0x00	Comando di test presenza scheda (non prevede nessuno scambio di dati)
0x05	somma di tutti i bytes precedenti.

Ogni volta che un *MicroPLC Lilith* 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 comunicazione il *MicroPLC Lilith* ha 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*. Il *MicroPLC lilith* può ricevere e trasmettere contemporaneamente da ambedue i canali di comunicazione senza impegnare la cpu infatti sia la trasmissione che la ricezione avvengono con l'ausilio di canali DMA ad alta velocità. 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 lilith* 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 e Merlino, 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 0x03¹

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

Messaggio di interrogazione

START	PLC	LEN	WRITE_INPUT	hbb	hbl	lhb	llb	CHK
-------	-----	-----	-------------	-----	-----	-----	-----	-----

Messaggio di risposta

START	PLC	LEN	WRITE_INPUT+128					CHK
-------	-----	-----	-----------------	--	--	--	--	-----

nel *MicroPLC Lilith* 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 Lilith* sta eseguendo un programma.

- **Comando WRITE_OUTPUT** valore 0x04

Quando un *MicroPLC Lilith* 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 Lilith* vi è un programma in RUN questi reimposterà autonomamente le uscite in base alla logica di programma. Se invece il *MicroPLC Lilith* 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	hbb	hbl	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 EXP810* .Per questo usa una forma del messaggio compatibile con questi apparecchi

Messaggio di interrogazione

START	PLC	LEN	WR_O_RD_I	hbb	hbl	lhb	llb	CHK
-------	-----	-----	-----------	-----	-----	-----	-----	-----

Messaggio di risposta

START	PLC	LEN	WR_O_RD_I +128	hbb	hbl	lhb	llb	CHK
-------	-----	-----	----------------	-----	-----	-----	-----	-----

- **Comando WR_O_RD_O** valore 0x06

Questo comando imposta lo stato delle uscite e ritorna la lettura di queste.

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

E' stato inserito per il funzionamento del *MicroPLC Lilith* come scheda remota su computer. Con questo comando infatti si ha automaticamente verifica di quanto scritto.

<i>Messaggio di interrogazione</i>				Impostazione uscite				
START	PLC	LEN	WR_O_RD_O	hbb	hlb	lhb	llb	CHK
<i>Messaggio di risposta</i>				Stato delle uscite				
START	PLC	LEN	WR_O_RD_O +128	hbb	hlb	lhb	llb	CHK

• 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 Lilith* ,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 Merlin*o 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.

<i>Messaggio di interrogazione</i>				Impostazione uscite				
START	PLC	LEN	WR_O_RD_IO	Hhb	Hlb	lhb	llb	CHK
<i>Messaggio di risposta</i>				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 .

<i>Messaggio di interrogazione</i>								
START	PLC	LEN	READ_ANI					CHK
<i>Messaggio di risposta</i>				ANI0		ANI1		
START	PLC	LEN	READ_ANI +128	Hb	Lb	hb	lb	
				ANI4		ANI5		
Hb	lb	Hb	lb	Hb	Lb	hb	lb	
				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.

<i>Messaggio di interrogazione</i>								
START	PLC	LEN	READ_ENC					CHK
<i>Messaggio di risposta</i>				Stato Encoder				
START	PLC	LEN	READ_ENC +128	Hhb	hlb	lhb	llb	
				Stato Contatore 0				
hbb	hlb	Lhb	llb	Hhb	hlb	lhb	llb	
				Impostazione rate				
hbb	hlb	Lhb	llb	Hhb	hlb	lhb	llb	
				Posizione 1				
hbb	hlb	Lhb	llb	Hhb	hlb	lhb	llb	
				Posizione 3				
hbb	hlb	Lhb	llb	Hhb	hlb	lhb	llb	
				Posizione 4				
hbb	hlb	Lhb	llb	Hhb	hlb	lhb	llb	
				Posizione 5				
				Posizione 6				

• **Comando READ_SLAVE1** valore 0x0e

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

<i>Messaggio di interrogazione</i>								
START	PLC	LEN	READ_SLAVE1	Stato Uscite				CHK
<i>Messaggio di risposta</i>								
START	PLC	LEN	READ_SLAVE1+128	Hhb	hlb	lhb	llb	
				Stato Ingressi				
hbb	hllb	llhb	llb	hbb	hllb	llhb	llb	
				Stato bobine contatori				
hbb	hllb	llhb	llb	hbb	hllb	llhb	llb	
				Stato bobine rele uscita				
hbb	hllb	llhb	llb	hbb	hllb	llhb	llb	
				Memoria di configurazione				
hbb	hllb	llhb	llb	hbb	hllb	llhb	llb	
				Mem. EXCHOS1+1				
hbb	hllb	llhb	llb	hbb	hllb	llhb	llb	
				Mem. EXCHOS1+2				
hbb	hllb	llhb	llb	hbb	hllb	llhb	llb	
				Mem. EXCHOS1+3				
hbb	hllb	llhb	llb	hbb	hllb	llhb	llb	
				Mem. EXCHIS1+2				
hbb	hllb	llhb	llb	hbb	hllb	llhb	llb	
				Mem. EXCHIS1+3				
hbb	hllb	llhb	llb	hbb	hllb	llhb	llb	CHK

• **Comando READ_SLAVE2** valore 0x0f

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

<i>Messaggio di interrogazione</i>								
START	PLC	LEN	READ_SLAVE2	Stato Uscite				CHK
<i>Messaggio di risposta</i>								
START	PLC	LEN	READ_SLAVE2+128	hbb	hllb	llhb	llb	
				Stato Ingressi				
hbb	hllb	llhb	llb	hbb	hllb	llhb	llb	
				Stato bobine timer				
hbb	hllb	llhb	llb	hbb	hllb	llhb	llb	
				Stato bobine rele uscita				
hbb	hllb	llhb	llb	hbb	hllb	llhb	llb	
				Memoria di configurazione				
hbb	hllb	llhb	llb	hbb	hllb	llhb	llb	
				Mem. EXCHOS2+1				
hbb	hllb	llhb	llb	hbb	hllb	llhb	llb	
				Mem. EXCHOS2+2				
hbb	hllb	llhb	llb	hbb	hllb	llhb	llb	
				Mem. EXCHOS2+3				
hbb	hllb	llhb	llb	hbb	hllb	llhb	llb	
				Mem. EXCHIS2+2				
hbb	hllb	llhb	llb	hbb	hllb	llhb	llb	
				Mem. EXCHIS2+3				
hbb	hllb	llhb	llb	hbb	hllb	llhb	llb	CHK

• **Comando SET_FLAG** valore 0x10

Imposta lo stato del flag delle uscite.

<i>Messaggio di interrogazione</i>	Dato impostazione
------------------------------------	-------------------

START	PLC	LEN	SET_FLAG	8 bit	CHK
<i>Messaggio di risposta</i>				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

<i>Messaggio di interrogazione</i>					
START	PLC	LEN	READ_SLAVE3		CHK
<i>Messaggio di risposta</i>				Stato Uscite	
START	PLC	LEN	READ_SLAVE3 +128	hbb	hbb
			Stato Ingressi	hbb	hbb
hbb	hbb	hbb	Stato bobine timer	hbb	hbb
hbb	hbb	hbb	Stato bobine rele ingresso	hbb	hbb
hbb	hbb	hbb	Mem. EXCHOS3	hbb	hbb
hbb	hbb	hbb	Mem. EXCHOS3+1	hbb	hbb
hbb	hbb	hbb	Mem. EXCHOS3+2	hbb	hbb
hbb	hbb	hbb	Mem. EXCHIS3	hbb	hbb
hbb	hbb	hbb	Mem. EXCHIS3+2	hbb	hbb
hbb	hbb	hbb	Mem. EXCHIS3+3	hbb	hbb
hbb	hbb	hbb		hbb	hbb
					CHK

• **Comando READ_SLAVE4** valore 0x12

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

<i>Messaggio di interrogazione</i>					
START	PLC	LEN	READ_SLAVE4		CHK
<i>Messaggio di risposta</i>				Stato Uscite	
START	PLC	LEN	READ_SLAVE4 +128	hbb	hbb
			Stato Ingressi	hbb	hbb
hbb	hbb	hbb	Stato bobine timer	hbb	hbb
hbb	hbb	hbb	Stato bobine rele ingresso	hbb	hbb
hbb	hbb	hbb	Mem. EXCHOS4	hbb	hbb
hbb	hbb	hbb	Mem. EXCHOS4+1	hbb	hbb
hbb	hbb	hbb	Mem. EXCHOS4+2	hbb	hbb
hbb	hbb	hbb	Mem. EXCHIS4	hbb	hbb
hbb	hbb	hbb	Mem. EXCHIS4+2	hbb	hbb
hbb	hbb	hbb	Mem. EXCHIS4+3	hbb	hbb
hbb	hbb	hbb		hbb	hbb
					CHK

• **Comando WR_S_RD_S** valore 0x13

Questo comando è usato per l'interscambio dati tra *lilith e Merlino-lilith* 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			PLC SLAVE 1	
Offset	Offset		Offset	Offset
R(85)	EXCHOS1		R(105)	EXCHIS1
R(86)	EXCHO1S1		R(106)	EXCHI1S1
R(87)	EXCHO2S1		R(107)	EXCHI2S1
R(88)	EXCHO3S1		R(108)	EXCHI3S1
R(105)	EXCHIS1		R(85)	EXCHOS1
R(106)	EXCHI1S1		R(86)	EXCHO1S1
R(107)	EXCHI2S1		R(87)	EXCHO2S1
R(108)	EXCHI3S1		R(88)	EXCHO3S1
PLC MASTER			PLC SLAVE 2	
Offset	Offset		Offset	Offset
R(89)	EXCHOS2		R(105)	EXCHIS1
R(90)	EXCHO1S2		R(106)	EXCHI1S1
R(91)	EXCHO2S2		R(107)	EXCHI2S1
R(92)	EXCHO3S2		R(108)	EXCHI3S1
R(109)	EXCHIS2		R(85)	EXCHOS1
R(110)	EXCHI1S2		R(86)	EXCHO1S1
R(111)	EXCHI2S2		R(87)	EXCHO2S1
R(112)	EXCHI3S2		R(88)	EXCHO3S1
PLC MASTER			PLC SLAVE 3	
Offset	Offset		Offset	Offset
R(93)	EXCHOS3		R(105)	EXCHIS1
R(94)	EXCHO1S3		R(106)	EXCHI1S1
R(95)	EXCHO2S3		R(107)	EXCHI2S1
R(96)	EXCHO3S3		R(108)	EXCHI3S1
R(113)	EXCHIS3		R(85)	EXCHOS1
R(114)	EXCHI1S3		R(86)	EXCHO1S1
R(115)	EXCHI2S3		R(87)	EXCHO2S1
R(116)	EXCHI3S3		R(88)	EXCHO3S1
PLC MASTER			PLC SLAVE 4	
Offset	Offset		Offset	Offset
R(97)	EXCHOS2		R(105)	EXCHIS1
R(98)	EXCHO1S2		R(106)	EXCHI1S1
R(99)	EXCHO2S2		R(107)	EXCHI2S1
R(100)	EXCHO3S2		R(108)	EXCHI3S1
R(117)	EXCHIS2		R(85)	EXCHOS1
R(118)	EXCHI1S2		R(86)	EXCHO1S1
R(119)	EXCHI2S2		R(87)	EXCHO2S1
R(120)	EXCHI3S2		R(88)	EXCHO3S1
PLC MASTER			PLC SLAVE 5	
Offset	Offset		Offset	Offset
R(100)	EXCHOS5		R(105)	EXCHIS1
R(101)	EXCHO1S5		R(106)	EXCHI1S1
R(102)	EXCHO2S5		R(107)	EXCHI2S1
R(103)	EXCHO3S5		R(108)	EXCHI3S1
R(121)	EXCHIS2		R(85)	EXCHOS1
R(122)	EXCHI1S2		R(86)	EXCHO1S1
R(123)	EXCHI2S2		R(87)	EXCHO2S1
R(124)	EXCHI3S2		R(88)	EXCHO3S1

<i>Messaggio di interrogazione</i>				EXCHOSn			
START	PLC	LEN	WR_S_RD_S	hhb	hlb	lhb	llb
						EXCHO1Sn	
hhb	hlb	lhb	llb	hhb	hlb	lhb	llb
						EXCHO2Sn	
						EXCHO3Sn	
hhb	hlb	lhb	llb				CHK
<i>Messaggio di risposta</i>				EXCHISn			
START	PLC	LEN	WR_S_RD_S +128	hhb	hlb	lhb	llb
						EXCHI1Sn	
hhb	hlb	lhb	llb	hhb	hlb	lhb	llb
						EXCHI2Sn	
						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 lilith* e non cambia velocità seriale. Si ottiene comunque un messaggio di risposta valido nella vecchia velocità seriale.

Nota:

Quando il *MicroPLC lilith* 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 comunicazione seriale il comando non viene acquisito il PLC e non cambia velocità seriale mentre il computer esegue interrogazioni alla nuova velocità, bisogna impostare un timeout oltre il quale reimpostare la vecchia velocità di comunicazione.

<i>Messaggio di interrogazione</i>				Nuova impostazione			
START	PLC	LEN	CHANGE_SPEED			Da 0 a 8	CHK
<i>Messaggio di risposta</i>				Nuova impostazione			
START	PLC	LEN	CHANGE_SPEED +128			Da 0 a 8	CHK

• **Comando LOAD_PRO valore 0x15**

Caricare un programma sul *MicroPLC lilith* 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.

<i>Messaggio di interrogazione</i>				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 lilith* disabilita le uscite azzerata 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 gruppi di 32 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 lilith* 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 lilith* 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 azzerata 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	CHK
-------	-----	-----	----------	---------	---------	-----

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, il 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 lilith* 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
-------	-----	-----	----------	-----	-----	-----	-----	-----

Manuale Utente

MicroPLC Lilith

• **Comando STOP** **valore 0x18**

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

Messaggio di interrogazione

START	PLC	LEN	STOP					CHK
<i>Messaggio di risposta</i>				DA SYSSTATOPLC R(5)=0				
START	PLC	LEN	STOP+128	h h	h l	l h	l l	CHK

• **Comando TRACE** **valore 0x19**

Impostata lo stato di debug remoto per il plc.

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

Messaggio di interrogazione

START	PLC	LEN	TRACE					CHK
<i>Messaggio di risposta</i>				DA SYSSTATOPLC R(5)=6				
START	PLC	LEN	TRACE+128	h h	h l	l h	l l	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
<i>Messaggio di risposta</i>				DA SYSSTATOPLC R(5)=6				
START	PLC	LEN	TRACE_RUN+128	h h	h l	l h	l l	
				Da SYSACC r(9)				
H h	H l	L h	l l	Da SYSPC r(10)				
				Da SYSCS r(11)				
H h	H l	L h	l l	Da SYSDEBUG r(14)				
				Da SYSDEBUG1 r(15)				
H h	H l	L h	l l	Da SYSBREKP1 r(16)				
				Da SYSBREKP2 r(17)				
H h	H l	L h	l l	Da SYSIND r(18)				
				Da SYSIND1 r(19)				
H h	H l	L h	l l	h h	h l	l h	l l	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

SYSDIND	della memoria del parametro sorgente
SYSSERRPLC	E' uguale al contenuto della memoria indiretta puntata dal contenuto della memoria del parametro destinazione 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	Stato Uscite				CHK
<i>Messaggio di risposta</i>								
START	PLC	LEN	READ_SLAVE5 +128	hbb	h1b	l1b	l1b	
Stato Ingressi				Stato bobine contatori				
Hbb	H1b	L1b	l1b	hbb	h1b	l1b	l1b	
Stato bobine timer				Stato bobine rele uscita				
Hbb	H1b	L1b	l1b	hbb	h1b	l1b	l1b	
Stato bobine rele ingresso				Memoria di configurazione				
Hbb	H1b	l1b	l1b	hbb	h1b	l1b	l1b	
Mem. EXCHOS5				Mem. EXCHOS5+1				
Hbb	H1b	l1b	l1b	hbb	h1b	l1b	l1b	
Mem. EXCHOS5+2				Mem. EXCHOS5+3				
Hbb	H1b	l1b	l1b	hbb	h1b	l1b	l1b	
Mem. EXCHIS5				Mem. EXCHIS5+2				
Hbb	H1b	l1b	l1b	hbb	h1b	l1b	l1b	
Mem. EXCHIS5+2				Mem. EXCHIS5+3				
Hbb	H1b	l1b	l1b	hbb	h1b	l1b	l1b	CHK

• **Comando READ_MEMORY1 valore 0x1c**

Legge il contenuto di un qualsiasi registro del PLC

Messaggio di interrogazione

START	PLC	LEN	READ_MEMORY1	ADDR hb	ADDR lb	CHK
<i>Messaggio di risposta</i>						
START	PLC	LEN	READ_MEMORY1+128	ADDR hb	ADDR lb	
Contenuto della memoria			l1b	l1b	CHK	

• **Comando WRITE_MEMORY1 valore 0x1d**

Scrive il contenuto di un qualsiasi registro del PLC

Messaggio di interrogazione

START	PLC	LE	WRITE_MEMORY1	ADDR hb	ADDR lb	CHK
<i>Messaggio di risposta</i>						
START	PLC	LE	WRITE_MEMORY1+128	ADDR hb	ADDR lb	CHK
Contenuto della memoria			l1b	l1b	CHK	
Address						

• **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

<i>Messaggio di interrogazione</i>				EXCHISn				
START	PLC	LEN	WR_S_RD_I	hbb	hlb	lhb	llb	
	EXCHI1Sn					EXCHI2Sn		
hbb	hhlb	lhb	llb	hbb	hhlb	lhb	llb	
	EXCHI3Sn							
hbb	hhlb	lhb	llb				CHK	
<i>Messaggio di risposta</i>				Ingressi PLC				
START	PLC	LEN	WR_S_RD_I+128	hbb	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

<i>Messaggio di interrogazione</i>				EXCHISn				
START	PLC	LEN	WR_S_RD_T	hbb	hlb	lhb	llb	
	EXCHI1Sn					EXCHI2Sn		
hbb	hhlb	lhb	Llb	hbb	hhlb	lhb	llb	
	EXCHI3Sn							
hbb	hhlb	lhb	Llb				CHK	
<i>Messaggio di risposta</i>				Bobine timer PLC				
START	PLC	LEN	WR_S_RD_T+128	hbb	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

<i>Messaggio di interrogazione</i>				Uscite del PLC				
START	PLC	LEN	WR_O_RD_T	hbb	hlb	lhb	llb	CHK
<i>Messaggio di risposta</i>				Bobine timer PLC				
START	PLC	LEN	WR_O_RD_T+128	hbb	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

<i>Messaggio di interrogazione</i>				EXCHISn				
START	PLC	LEN	WR_S_RD_C	hbb	hlb	lhb	llb	
	EXCHI1Sn					EXCHI2Sn		
hbb	Hlb	lhb	Llb	hbb	hhlb	lhb	llb	
	EXCHI3Sn							
hbb	Hlb	lhb	Llb				CHK	
<i>Messaggio di risposta</i>				Bobine Contatori PLC				
START	PLC	LEN	WR_S_RD_C+128	hbb	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

<i>Messaggio di interrogazione</i>				Uscite del PLC				
START	PLC	LEN	WR_O_RD_C	hhb	h1b	l1b	llb	CHK
<i>Messaggio di risposta</i>				Bobine contatori PLC				
START	PLC	LEN	WR_O_RD_C+128	hhb	h1b	l1b	llb	CHK

• ³**Comando WRITE_TIMER** **valore 0x29**

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

<i>Messaggio di interrogazione</i>				Stato a cui impostare le bobine dei Timer				
START	PLC	LEN	WRITE_TIMER	hhb	h1b	l1b	llb	CHK
<i>Messaggio di risposta</i>								
START	PLC	LEN	WRITE_TIMER +128					CHK

• **Comando WRITE_COUNTER** **valore 0x2A**

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

<i>Messaggio di interrogazione</i>				Stato a cui impostare le bobine dei Contatori				
START	PLC	LE N	WRITE_COUNTER	hhb	h1b	l1b	llb	CHK
<i>Messaggio di risposta</i>								
START	PLC	LE N	WRITE_COUNTER +128					CHK

• **Comando WRITE_RELE** **valore 0x2b**

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

<i>Messaggio di interrogazione</i>				Blocco 1 R(46)				
START	LEN	PLC	LEN	hhb	h1b	l1b	llb	
	Blocco 2 R(47)				Blocco 3 R(48)			
hhb	H1b	l1b	llb	hhb	h1b	l1b	llb	
	Blocco 4 R(49)				Blocco 5 R(50)			
hhb	H1b	l1b	llb	hhb	h1b	l1b	llb	
	Blocco 6 R(51)				Blocco 7 R(52)			
hhb	H1b	l1b	llb	hhb	h1b	l1b	llb	
	Blocco 8 R(53)							
hhb	H1b	l1b	llb					CHK
<i>Messaggio di risposta</i>								
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.

<i>Messaggio di interrogazione</i>							
START	PLC	LE	WRITE_TIME	ORA	MIN	SEC	GG
		N		Char	Char	Char	Char
MM	AA	NumG					
Char	Char	Char					CHK
<i>Messaggio di risposta</i>							
START	PLC	LEN	WRITE_TIME+128				CHK

• **⁴Comando 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.

<i>Messaggio di interrogazione</i>					Nuovo Numero di PLC
START	PLC	LE	WRITE_NUM_PLC		CHK
		N		Char	
<i>Messaggio di risposta</i>					
START	PLC	LE	WRITE_NUM_PLC		CHK
		N	+128		

• **⁵Comando 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: <ul style="list-style-type: none"> • 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						
<i>Messaggio di interrogazione</i>							
START	PLC	LE	READ_ST_TIMER	Numero di timer	CHK		
		N		Char			
<i>Messaggio di risposta</i>							
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: <ul style="list-style-type: none"> • 1 Conta su • 2 Conta giu • 3 Auto su (come timer veloce) • 4 Auto giu(come timer veloce)
7	qualsiasi altro valore indica una impostazione errata del contatore byte che indica se il contatore è abilitato o disabilitato

<i>Messaggio di interrogazione</i>				Numero di Contatore			
START	PLC	LE	READ_ST_COUNTER	Char		CHK	
		N					
<i>Messaggio di risposta</i>							
START	PLC	LE	READ_ST_COUNTER +128	Char	Char	Char	Char
		N					
Char	Char	Char					CHK

• **6Comando 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 <ul style="list-style-type: none"> 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 abilitato o disabilitato

<i>Messaggio di interrogazione</i>				Numero di Orologio			
START	PLC	LE	READ_ST_CLOCK	Char	Char	Char	CHK
		N					

<i>Messaggio di risposta</i>							
START	PLC	LE	READ_ST_CLOCK +128	Char	Char	Char	Char
		N					
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).

<i>Messaggio di interrogazione</i>				Numero di timer			
START	PLC	LEN	WRITE_ST_TIMER	Char	Char	Char	Char
Char	Char	Char					CHK

<i>Messaggio di risposta</i>							
START	PLC	LE	WRITE_ST_TIMER+128				CHK
		N					

• 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).

<i>Messaggio di interrogazione</i>				Numero di timer			
START	PLC	LE	WRITE_ST_COUNTER	Char	Char	Char	Char
		N					
Char	Char	Char					CHK

<i>Messaggio di risposta</i>							
START	PLC	LE	WRITE_ST_COUNTER+128				CHK
		N					

• Comando WRITE_ST_CLOCK valore 0x38

Con questo comando si imposta un l'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 dell'orologio (vedi READ_ST_CLOCK).

<i>Messaggio di interrogazione</i>				Numero di timer			
START	PLC	LEN	WRITE_ST_CLOCK	Char	Char	Char	Char
Char	Char	Char					CHK

<i>Messaggio di risposta</i>							
START	PLC	LE	WRITE_ST_CLOCK+128				CHK
		N					

• Comando RET_RUN valore 0x39

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

Messaggio di interrogazione

START	PLC	LE N	RET_RUN	CHK
-------	-----	---------	---------	-----

Messaggio di risposta

START	PLC	LE N	RET_RUN+128	CHK
-------	-----	---------	-------------	-----

• **Comando READ_MEMORY** **valore 0x3b**

Permette di leggere 16 registri di sistema contemporaneamente in tutto il range dei registri 0..1500
Nel messaggio di interrogazione nel campo dati bisogna specificare l'indice dei registri da leggere

Messaggio di interrogazione

START	PLC	LEN	READ_MEMORY	High byte	Low byte	CHK
-------	-----	-----	-------------	-----------	----------	-----

Messaggio di risposta

START	PLC	LE N	READ_MEMORY +128	High byte	Low byte	
		R [ADDR]			R [ADDR+1]	
h hb	H lb	l hb	L lb	h hb	h lb l hb	L lb
		R [ADDR+2]			R [ADDR+3]	
h hb	H lb	l hb	l lb	h hb	h lb l hb	L lb
		R [ADDR+4]			R [ADDR+5]	
h hb	H lb	l hb	l lb	h hb	h lb l hb	L lb
		R [ADDR+6]			R [ADDR+7]	
h hb	H lb	l hb	l lb	h hb	h lb l hb	L lb
		R [ADDR+8]			R [ADDR+9]	
h hb	H lb	l hb	l lb	h hb	h lb l hb	L lb
		R [ADDR+10]			R [ADDR+11]	
h hb	h lb	l hb	l lb	h hb	h lb l hb	l lb
		R [ADDR+12]			R [ADDR+13]	
h hb	h lb	l hb	l lb	h hb	h lb l hb	l lb
		R [ADDR+14]			R [ADDR+15]	
h hb	h lb	l hb	l lb	h hb	h lb l hb	l lb CHK

• **Comando WRITE_MEMORY** **valore 0x3c**

Permette di scrivere 16 registri di sistema contemporaneamente in tutto il range dei registri 0..1500
Nel messaggio di interrogazione nel campo dati bisogna specificare l'indice dei registri da scrivere, oltre ovviamente ai dati da impostare

Messaggio di interrogazione

START	PLC	LE N	WRITE_MEMORY	High byte	Low byte	
-------	-----	---------	--------------	-----------	----------	--

		R [ADDR]			R [ADDR+1]	
H hb	h lb	l hb	l lb	h hb	h lb l hb	l lb
		R [ADDR+2]			R [ADDR+3]	
H hb	h lb	l hb	l lb	h hb	h lb l hb	l lb
		R [ADDR+4]			R [ADDR+5]	
H hb	h lb	l hb	l lb	h hb	h lb l hb	l lb
		R [ADDR+6]			R [ADDR+7]	
h hb	h lb	l hb	l lb	h hb	h lb l hb	l lb
		R [ADDR+8]			R [ADDR+9]	
h hb	h lb	l hb	l lb	h hb	h lb l hb	l lb
		R [ADDR+10]			R [ADDR+11]	
h hb	h lb	l hb	l lb	h hb	h lb l hb	l lb

	R [ADDR+12]			R [ADDR+13]			
hbb	hbb	hbb	hbb	hbb	hbb	hbb	hbb
	R [ADDR+14]			R [ADDR+15]			
hbb	hbb	hbb	hbb	hbb	hbb	hbb	CHK
<i>Messaggio di risposta</i>							
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 20 caratteri.

<i>Messaggio di interrogazione</i>							
START	PLC	LEN	WRITE_ROW	X	Y	Char	Char
Char	Char	Char	0 (fine stringa)				CHK
<i>Messaggio di risposta</i>							
START	PLC	LE	WRITE_ROW+128				CHK
		N					

• **Comando READ_ROW** **valore 0x44**

Permette di leggere una riga sul display LCD.

<i>Messaggio di interrogazione</i>								Numero di riga 0..1
START	PLC	LEN	READ_ROW			Char	Char	CHK
Char	Char	Char	0 (fine stringa)					
<i>Messaggio di risposta</i>								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	Char	Char	Char	Char
			Stringa di 20 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 80 caratteri che compongono il messaggio.

<i>Messaggio di interrogazione</i>								40 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	Char	Char	Char	Char	Char	Char	Char
<i>Messaggio di risposta</i>								Numero di riga 0..1
START	PLC	LE	WRITE_LCD +128			Char	Char	CHK
		N						

• **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

<i>Messaggio di interrogazione</i>					Posizione di start		
START	PLC	LEN	READ_LCD		0:primi 40	1 secondi 40	CHK
<i>Messaggio di risposta</i>							
START	PLC	LE N	READ_LCD +128			0,1	
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
Char	Char	Char	Char	Char	Char	Char	Char
40 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.

Per mantenere piena compatibilità con il PLC Merlino rispecchia fedelmente le impostazioni di questo PLC

⁸ Riferimento	Addr	Offset	Aiuto
SYSADDRSTART	0	R(0)	<ul style="list-style-type: none"> • 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)	<ul style="list-style-type: none"> • 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: <ul style="list-style-type: none"> • 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) <ul style="list-style-type: none"> • 0 : STOP • 1 : RUN • 2 : STEP • 3 : BREAK • 4 : ERASE

⁸ Con Riferimento si intende la etichetta così come è definita nel file Merlino.def

			<ul style="list-style-type: none"> • 5 : LOAD • 6 : DEBUG REMOTO
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)
SYSYS	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 <ul style="list-style-type: none"> • 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 <ul style="list-style-type: none"> • 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 <ul style="list-style-type: none"> • 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 <ul style="list-style-type: none"> • 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 :

OROL	180	R(45)	<ul style="list-style-type: none"> • 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. <p>32 BOBINE OROLOGI INTERNI</p> <ul style="list-style-type: none"> • 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'accensione)
R2	188	R(47)	32 BOBINE RELE 2 INTERNI (impostati a 0 all'accensione)
R3	192	R(48)	32 BOBINE RELE 2 INTERNI (impostati a 0 all'accensione)
R4	196	R(49)	32 BOBINE RELE 3 INTERNI (impostati a 0 all'accensione)
R5	200	R(50)	32 BOBINE RELE 4 INTERNI (stato ritenuto all'accensione)
R6	204	R(51)	32 BOBINE RELE 5 INTERNI (stato ritenuto all'accensione)
R7	208	R(52)	32 BOBINE RELE 6 INTERNI (stato ritenuto all'accensione)
R8	212	R(53)	32 BOBINE RELE 7 INTERNI (stato ritenuto all'accensione)
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....hbb
TIM1	280	R(70)	Temporizzazioni 30SEC,2MIN,5MIN,10MIN llb....hbb
TIM2	284	R(71)	Orologio di sistema llb..hbb <ul style="list-style-type: none"> • LLB = Decimi di secondo • LHB = Secondi • HLB = Minuti • HHB = Ore
TIM3	288	R(72)	Orologio di sistema llb..hbb <ul style="list-style-type: none"> • 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: <ul style="list-style-type: none"> • 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

			<ul style="list-style-type: none"> • 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 Merlinio 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:
SI2	304	R(76)	<ul style="list-style-type: none"> • Bit 0 ... Bit 23 bobina istantanea timer 0..23 Contatti Speciali blocco 3: <ul style="list-style-type: none"> • 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: <ul style="list-style-type: none"> • 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: <ol style="list-style-type: none"> 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 RS11(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 EXCH1S1(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
			EXCHIS5 del
			MERLINO slave 5

EXCHO2S5	412	R(103)	INTRACOM A MERLINO SLAVE 5
EXCHO3S5	416	R(104)	INTRACOM A 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:
			<ul style="list-style-type: none"> • LLB Posizione attuale di scrittura copia display • LHB contatore per visualizzazione pagina stato • HLB Posizione cursore X • HHB Posizione Cursore Y
LASTPAGE	824	R(206)	Ultima pagina visualizzata.Contiene il puntatore al descrittore
			di pagina.Viene usata dal molte SYS 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 : <ul style="list-style-type: none"> • LLB Com 0 se 0 modo binario se 1 modo ascii • LHB Com 1 se 0 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. <ul style="list-style-type: none"> • 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 hw = valore attuale
TIMIMP1	864	R(216)	Timer 1 lw = Set impostazione hw = valore attuale
TIMIMP2	868	R(217)	Timer 2 lw = Set impostazione hw = valore attuale
TIMIMP3	872	R(218)	Timer 3 lw = Set impostazione hw = valore attuale
TIMIMP4	876	R(219)	Timer 4 lw = Set impostazione hw = valore attuale
TIMIMP5	880	R(220)	Timer 5 lw = Set impostazione hw = valore attuale
TIMIMP6	884	R(221)	Timer 6 lw = Set impostazione hw = valore attuale
TIMIMP7	888	R(222)	Timer 7 lw = Set impostazione hw = valore attuale
TIMIMP8	892	R(223)	Timer 8 lw = Set impostazione hw = valore attuale
TIMIMP9	896	R(224)	Timer 9 lw = Set impostazione hw = valore attuale
TIMIMP10	900	R(225)	Timer 10 lw = Set impostazione hw = valore attuale
TIMIMP11	904	R(226)	Timer 11 lw = Set impostazione hw = valore attuale
TIMIMP12	908	R(227)	Timer 12 lw = Set impostazione hw = valore attuale
TIMIMP13	912	R(228)	Timer 13 lw = Set impostazione hw = valore attuale
TIMIMP14	916	R(229)	Timer 14 lw = Set impostazione hw = valore attuale
TIMIMP15	920	R(230)	Timer 15 lw = Set impostazione hw = valore attuale
TIMIMP16	924	R(231)	Timer 16 lw = Set impostazione hw = valore attuale
TIMIMP17	928	R(232)	Timer 17 lw = Set impostazione hw = valore attuale
TIMIMP18	932	R(233)	Timer 18 lw = Set impostazione hw = valore attuale
TIMIMP19	936	R(234)	Timer 19 lw = Set impostazione hw = valore attuale
TIMIMP20	940	R(235)	Timer 20 lw = Set impostazione hw = valore attuale
TIMIMP21	944	R(236)	Timer 21 lw = Set impostazione hw = valore attuale
TIMIMP22	948	R(237)	Timer 22 lw = Set impostazione hw = valore attuale
TIMIMP23	952	R(238)	Timer 23 lw = Set impostazione hw = valore attuale
TIMMODE0	956	R(239)	llb modo timer 0..... hbb modo timer 3
TIMMODE1	960	R(240)	llb modo timer 4..... hbb modo timer 7
TIMMODE2	964	R(241)	llb modo timer 8..... hbb modo timer 11
TIMMODE3	968	R(242)	llb modo timer 12..... hbb modo timer 15
TIMMODE4	972	R(243)	llb modo timer 16..... hbb modo timer 19
TIMMODE5	976	R(244)	llb modo timer 20..... hbb modo timer 23

CONIMP0	980	R(245)	Counter 0 lw = Set impostazione hw = valore attuale
CONIMP1	984	R(246)	Counter 1 lw = Set impostazione hw = valore attuale
CONIMP2	988	R(247)	Counter 2 lw = Set impostazione hw = valore attuale
CONIMP3	992	R(248)	Counter 3 lw = Set impostazione hw = valore attuale
CONIMP4	996	R(249)	Counter 4 lw = Set impostazione hw = valore attuale
CONIMP5	1000	R(250)	Counter 5 lw = Set impostazione hw = valore attuale
CONIMP6	1004	R(251)	Counter 6 lw = Set impostazione hw = valore attuale
CONIMP7	1008	R(252)	Counter 7 lw = Set impostazione hw = valore attuale
CONIMP8	1012	R(253)	Counter 8 lw = Set impostazione hw = valore attuale
CONIMP9	1016	R(254)	Counter 9 lw = Set impostazione hw = valore attuale
CONIMP10	1020	R(255)	Counter 10 lw = Set impostazione hw = valore attuale
CONIMP11	1024	R(256)	Counter 11 lw = Set impostazione hw = valore attuale
CONIMP12	1028	R(257)	Counter 12 lw = Set impostazione hw = valore attuale
CONIMP13	1032	R(258)	Counter 13 lw = Set impostazione hw = valore attuale
CONIMP14	1036	R(259)	Counter 14 lw = Set impostazione hw = valore attuale
CONIMP15	1040	R(260)	Counter 15 lw = Set impostazione hw = valore attuale
CONIMP16	1044	R(261)	Counter 16 lw = Set impostazione hw = valore attuale
CONIMP17	1048	R(262)	Counter 17 lw = Set impostazione hw = valore attuale
CONIMP18	1052	R(263)	Counter 18 lw = Set impostazione hw = valore attuale
CONIMP19	1056	R(264)	Counter 19 lw = Set impostazione hw = valore attuale
CONIMP20	1060	R(265)	Counter 20 lw = Set impostazione hw = valore attuale
CONIMP21	1064	R(266)	Counter 21 lw = Set impostazione hw = valore attuale
CONIMP22	1068	R(267)	Counter 22 lw = Set impostazione hw = valore attuale
CONIMP23	1072	R(268)	Counter 23 lw = Set impostazione hw = valore attuale
CONMODE0	1076	R(269)	llb modo Counter 0..... hbb modo Counter 3
CONMODE1	1080	R(270)	llb modo Counter 4..... hbb modo Counter 7
CONMODE2	1084	R(271)	llb modo Counter 8..... hbb modo Counter 11
CONMODE3	1088	R(272)	llb modo Counter 12..... hbb modo Counter 15
CONMODE4	1092	R(273)	llb modo Counter 16..... hbb modo Counter 19
CONMODE5	1096	R(274)	llb modo Counter 20..... hbb 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 (hbb)
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(1500) 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 lilith* 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 tre 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 Merlinoo Lilith* 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 Lilith* 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 doppiini 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 inilizzazione 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 inilizzazione (prima di istruzione start)
2	Controllo e impostazione Velocita di comunicazione Seriale: Queste impostazioni vengono effettuate posizionando un blocco init sullo schema elettrico in modo da impostare i parametri di comunicazione
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

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

CONF _n	
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)
- 9 Il messaggio scambiato tra il merlino e D1208 e WR_S_RD_S .
- 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 passati attraverso 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 lista è 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 50simsi 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

Scrivere , 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

Scrivere , 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

Scrivere , 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 pagina. 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 visualizzare
Return	No

Scriva sul display LCD la pagina puntata dal puntatore **ptr**. Imposta una copia del puntatore alla pagina sul registro r(206). Una pagina è definita in due modi:

Compatibile MERLINO : da due puntatori ai messaggi per le righe 1 e 2 e da un array di puntatori ai valori delle variabili da visualizzare.

Oppure nel nuovo formato : un numero che indica il numero delle righe 0..255, un apuntatore ad un array di stringhe delle righe e da un array di puntatori ai valori delle variabili da visualizzare.

Esempio

```
mov.l #miapagina,r(300)
sys #13,r(300)
```

COMPATIBILE MERLINO

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 di puntatori ai valori deve terminare con un puntatore nullo DL 0.

I puntatori ai valori indicano l'indirizzo della struttura descrittiva della variabile da visualizzare e sono così definiti:

Valore1:

```
DL r(300) ;(indirizzo variabile) indirizzo della variabile da visualizzare
DL limit1 ;(limite della variabile) tipo di visualizzazione e trattamento del puntatore
```

Valore2:

```
DL r(301) ;indirizzo della variabile da visualizzare
DL limit1 ;tipo di visualizzazione 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 così 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
```

LILITH

miapagina:

```
DL 5 ;numero delle righe
DL ArrayMsg ;puntatore ai messaggi
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
```

ArrayMsg::

DL Msg1

DL Msg2

DL Msg3

DL Msg4

DL Msg5

Msg1:

DM "R(300)=

“

Msg2:

DM "R(301)=

“

Msg3:

DM "R(301)=

“

Msg4:

DM "R(301)=

“

Msg5:

DM "R(301)=

“

Per le variabili usa la stessa sintassi del merlino con la sola eccezione che accetta posizioni Y fino al numero di righe meno uno.

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 i **24 TASTI NUMERICI**. 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

- **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 uscirne 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 errore 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 essenzialmente 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 = 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 controllo 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 essenzialmente 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 controllo 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 indirizzo 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	Velocità (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	Velocità di comunicazione
Return	No

Imposta la porta di comunicazione **0** alla velocità 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	Velocità di comunicazione
Return	No

Imposta la porta di comunicazione **1** alla velocità 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 consecutivi

• **void WriteCom(register char porta,register char num,register char mode)**

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 quantità *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

```
absctlb 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 impulsi 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 ignorato e impostato a 7353

Se tick è maggiore di 65535 viene ignorato e impostato a 65535.

Il rate è calcolato come impulsi 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 2 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 2 la stringa immagazzinata nel buffer di uscita seriale 2

- **Float ToFloat(register long t)**

Offset	47
Parametro 1	Dato da convertire in floating point
Return	Si

Converte il dato in ingresso nel formato floating point. Ad esempio:

```
mov.l #123456,r(300) ;r300 =long
sys #47,r(300) ;tfloat
mov.l r(0),r(300) ;r300= rappresentazione floating point del numero 123456
```

- **long FloatToLong(register float t)**

Offset	48
Parametro 1	Dato da convertire in long
Return	Si

Converte il dato in ingresso nel formato long. Bisogna puntualizzare che ogni registro è adatto a contenere un dato floating 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 convertire 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 convertire 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)
Jnz b condizionale ;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 *FloatEq*(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 del registro r(9) accumulatore a bit.

- **void *FloatNeg*(register float a, register float b)**

Offset	58
Parametro 1	Dato
Parametro 2	Dato
Return	No

Se $a \neq b$ è vero viene impostato a 1 il bit 0 del 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 del 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 del 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 floating 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 No
 I
 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 Indirizzo di memoria da 0 a 4192,(ultimi 4Kb memoria di programma da 1F000 a 1FFFF)
 I Dato da programmare (1 Byte)
 Parametro 2
 Return No

Scriva 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

<i>Offset</i>	83
<i>Parametro</i>	Puntatore alla stringa (in RAM)
<i>1</i>	
<i>Parametro</i>	Dato da formattare
<i>2</i>	
<i>Return</i>	No

Formatta una stringa con un valore in formato dato dalla rappresentazione binaria dei primi 8 bit del dato

• StrBin16

<i>Offset</i>	84
<i>Parametro</i>	Puntatore alla stringa (in RAM)
<i>1</i>	
<i>Parametro</i>	Dato da formattare
<i>2</i>	
<i>Return</i>	No

Formatta una stringa con un valore in formato dato dalla rappresentazione binaria dei primi 16 bit del dato

• StrBin32

<i>Offset</i>	85
<i>Parametro</i>	Puntatore alla stringa (in RAM)
<i>1</i>	
<i>Parametro</i>	Dato da formattare
<i>2</i>	
<i>Return</i>	No

Formatta una stringa con un valore in formato dato dalla rappresentazione binaria dei primi 32 bit del dato

• StrNum

<i>Offset</i>	86
<i>Parametro</i>	Puntatore alla stringa (in RAM)
<i>1</i>	
<i>Parametro</i>	Dato da formattare
<i>2</i>	
<i>Parametro</i>	Formato da usare per il numero in dato. Identico ai formati usati per la visualizzazione sul display LCD
<i>3</i>	
<i>Return</i>	No

Formatta una stringa con un valore in RAM

• AscTime

<i>Offset</i>	87
<i>Parametro</i>	No
<i>1</i>	
<i>Return</i>	No

Spedisce dalla porta di comunicazione 2 un messaggio ascii con la indicazione della ora e data.ES.
Alle 10:00:00 del 11/8/97 ::

• TxNStr

Offset **88**
Parametro Puntatore alla stringa
1
Return No
Trasmette direttamente la stringa puntata sulla porta di comunicazione 2

• TxStrNum

Offset **89**
Parametro Puntatore alla stringa
1
Parametro Dato da formattare
2
Parametro Formato del dato
3
Return No
Trasmette direttamente la stringa puntata sulla porta di comunicazione 2 con il dato formattato come specificato dal terzo parametro. Appende alla fine della stringa i caratteri \r\n

• StrLen

Offset **90**
Parametro Puntatore alla stringa
1
Return Lunghezza della stringa
Ritorna la lunghezza della stringa

• StrCpy

Offset **91**
Parametro Puntatore alla stringa destinazione in ram
1
Parametro Puntatore alla stringa sorgente
2
Return No
Copia la stringa sorgente nella destinazione

• StrSet

Offset **92**
Parametro Puntatore alla stringa
1
Parametro Dato
2
Return No
Imposta tutta la stringa con il carattere specificato dal secondo parametro

• RetPtrStr

Offset **93**
Parametro Puntatore alla stringa
1
Parametro Indirizzo dove finisce la stringa
Return Ritorna l'indirizzo dove finisce una stringa

• InitCom0

Offset **94**
Parametro Dati 7-8
1
Parametro Stop 1-2
2
Parametro Parità 0,1,2
3
Return No
 Imposta la porta di comunicazione 0 nel modo specificato. La velocità di comunicazione è specificata dalla parte bassa della MEM[7]

• StrCmp

Offset **95**
Parametro Puntatore Stringa 1
1
Parametro Puntatore Stringa 2
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 Puntatore area confronto 1
1
Parametro Puntatore area confronto 2
2
Parametro Lunghezza area
3
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 Puntatore inizio tabella puntatori alle stringhe
1
Parametro Indice della stringa da trasmettere
2
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 considerazione 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..1500
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..1500
Return	No

Monitorizza la ricezione sulla porta di comunicazione 2. 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:

<u>REG[0]</u>	Stato del posizionamento 1 arrivato 0 in corso
<u>REG[1]</u>	Punto di Arrivo asse X
<u>REG[2]</u>	Punto di Arrivo asse Y
<u>REG[3]</u>	Punto di Arrivo asse Z
<u>REG[4]</u>	Punto di Partenza asse X
<u>REG[5]</u>	Punto di Partenza asse Y
<u>REG[6]</u>	Punto di Partenza asse Z
<u>REG[7]</u>	Massima distanza tra i punti
<u>REG[8]</u>	Variabile temporanea che va da 0 a REG[7]
<u>REG[9]</u>	Distanza X Attuale
<u>REG[10]</u>	Distanza Y Attuale
<u>REG[11]</u>	Distanza Z Attuale
<u>REG[12]</u>	Errore X Attuale
<u>REG[13]</u>	Errore Y Attuale
<u>REG[14]</u>	Errore Z Attuale
<u>REG[15]</u>	Incremento X
<u>REG[16]</u>	Incremento Y
<u>REG[17]</u>	Incremento Z
<u>REG[18]</u>	Stato delle uscite
<u>REG[19]</u>	Se vi è buffer di lavorazione posizione sul buffer
<u>REG[20]</u>	Se vi è buffer di lavorazione lunghezza del buffer
<u>REG[21]</u>	Se vi è buffer di lavorazione indirizzo in ram del buffer primario
<u>REG[22]</u>	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. $F=21873000/\text{divisore}$	

• CNCXYZ	
<i>Offset</i>	115
<i>Parametro</i>	Puntatore alla struttura di posizionamento
<i>I</i>	
<i>Return</i>	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 occupano 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 vertici, 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 uniformità di lavorazione eliminando tempi morti dovuti alla comunicazione dei vertici di lavorazione.	
• CNCXY	
<i>Offset</i>	116
<i>Parametro</i>	Puntatore alla struttura di posizionamento
<i>I</i>	
<i>Return</i>	No
Come funzione 115 ma solo XY	
• CNCXZ	
<i>Offset</i>	117
<i>Parametro</i>	Puntatore alla struttura di posizionamento
<i>I</i>	
<i>Return</i>	No
Come funzione 115 ma solo XZ	
• CNCYZ	
<i>Offset</i>	118
<i>Parametro</i>	Puntatore alla struttura di posizionamento
<i>I</i>	
<i>Return</i>	No
Come funzione 115 ma solo YZ	
• EnableUser1	
<i>Offset</i>	119
<i>Parametro</i>	No
<i>I</i>	
<i>Return</i>	No
Abilita l'interrupt timer utente 1	
• EnableUser2	
<i>Offset</i>	120
<i>Parametro</i>	No
<i>I</i>	
<i>Return</i>	No
Abilita l'interrupt timer utente 2	
• DisableUser1	

<i>Offset</i>	121
<i>Parametro</i>	No
<i>I</i>	
<i>Return</i>	No

Disabilita l'interrupt timer utente 1

- **DisableUser2**

<i>Offset</i>	122
<i>Parametro</i>	No
<i>I</i>	
<i>Return</i>	No

Disabilita l'interrupt timer utente 2

- **EnableEncoder**

<i>Offset</i>	123
<i>Parametro</i>	No
<i>I</i>	
<i>Return</i>	No

Abilita la gestione dell'encoder default

- **DisableEncoder**

<i>Offset</i>	124
<i>Parametro</i>	No
<i>I</i>	
<i>Return</i>	No

Non esegue la gestione dell'encoder

- **SweepTimerUno**

<i>Offset</i>	125
<i>Parametro</i>	Puntatore alla Struttura di posizionamento
<i>I</i>	
<i>Return</i>	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 velocità e precisamente

R[POS+23]=Velocità minima

R[POS+24]=Velocità massima

R[POS+25]=lunghezza in passi accelerazione-decelerazione

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 accelerazione a trapezio di cui la memoria ad offset 25 rappresenta in passi la distanza di accelerazione mentre la memoria ad offset 26 rappresenta l'accelerazione fino ad ottenere la velocità specificata nella memoria ad offset 24.

La frequenza è ottenuta con

$F_{min} = 14753000 / \text{velocità minima}$

$F_{min} = 14753000 / \text{velocità massima}$

- **SweepCNCXYZ**

Offset	126
Parametro	Puntatore alla struttura di posizionamento
I	
Return	No
Come la funzione 115 ma gestisce anche il profilo della velocità (CNCXYZ+SweepTimerUno)	

• **SweepCNCXY**

Offset	127
Parametro	Puntatore alla struttura di posizionamento
I	
Return	No
Come funzione 126 ma solo XY	

• **SweepCNCXZ**

Offset	128
Parametro	Puntatore alla struttura di posizionamento
I	
Return	No
Come funzione 126 ma solo XZ	

• **SweepCNCYZ**

Offset	129
Parametro	Puntatore alla struttura di posizionamento
I	
Return	No
Come funzione 126 ma solo YZ	

• **StopZSweepCNCXYZ**

Offset		130
Parametro	Puntatore alla struttura di posizionamento	
I		
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 ple 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.

SOLO microPLC Lilith

• Regolatore P.I.D

Offset	130
Parametro	Puntatore alla struttura di regolazione
I	
Return	No

La struttura di regolazione ha il seguente schema:

sia n il parametro 1 allora

R[n]= SEGNO USCITA 1 (bit 0) in salita, 2 (bit 1) in discesa, 0 su setpoint

R[n+1]= USCITA DEL REGOLATORE

R[n+2]=SCALA DELL' INGRESSO - ZERO SCALA

R[n+3]=SCALA DELL' INGRESSO - SPAN SCALA

R[n+4]=VARIABILE DI INGRESSO

R[n+5]=SETPOINT RISPETTO ALLA SCALA DEFINITA

R[n+6]=KOSTANTE PROPORZIONALE NUMERATORE

R[n+7]=KOSTANTE PROPORZIONALE DENOMINATORE

R[n+8]=KOSTANTE DERIVATIVA NUMERATORE

R[n+9]=KOSTANTE DERIVATIVA DENOMINATORE

R[n+10]=KOSTANTE INTEGRATIVA NUMERATORE

R[n+11]=KOSTANTE INTEGRATIVA DENOMINATORE

R[n+12]=ACCUMULATORE

R[n+13]=ERR1

R[n+14]=ERR2

R[n+15]=ERR3

R[n+16]=ERR4

R[n+17]=VECCHIO VALORE INGRESSO

R[n+18]=VECCHIO VALORE USCITA

Il regolatore P.I.D. esegue il calcolo in floatting Point.

• Regolatore Proporzionale

Offset	131
Parametro	Puntatore alla struttura di regolazione
I	
Return	No

Della struttura di regolazione definita sulla precedente funzione si usa solo la parte che riguarda ingresso ,uscita e Kostante proporzionale

Il regolatore Proporzionale esegue il calcolo in floatting Point.

• Visualizza menu

Offset**131****Parametro**

Puntatore alla struttura del menu

1**Parametro**

Numero del registro indice

2**Return**

No

I menu sul microPLC merlino sono stati implementati attraverso il linguaggio senza ricorso a funzioni di sistema. Allo scopo di ottimizzare questa proprietà e' stata implementata la questa funzione.

La struttura del menu è così definita:

DL numero Item

DL Msgitem 1

DL Msgitem 2

....

..

.

Msgitem1:

DM "prova",0,

In base al contenuto del registro indice vengono visualizzati i 4 messaggi partendo da quello indicato .

GARANZIA

Il MicroPLC LILITH 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
- D) difetti causati da una programmazione insufficiente o errata

L'assistenza tecnica si intende franco nostra sede.

Trasporto a carico dell'acquirente