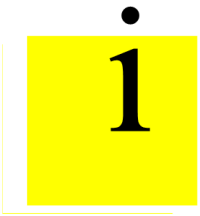
Manuale utente

Controller per motore passo-passo
Codice RS: 434540
Codice RS: 434542
Codice RS: 434543

Precauzioni e note nel testo:

Attenzione
Diversi tipi di pericolo, che possono provocare danni a cose o lesioni alle persone.



Informazioni
Consigli, suggerimenti o riferimenti ad un altro documento.

Punti salienti e formattazione di frammenti di testo:

- Etichetta o marchio a scelta

1) Alcune azioni devono essere eseguite in ordine sequenziale

— Enumerazioni comuni

Informazioni del produttore

RS Components aderisce alla linea di sviluppo continuo e si riserva il diritto di apportare modifiche e miglioramenti al design e al software del prodotto senza preavviso.
Le informazioni contenute in questo manuale sono soggette a modifiche in qualsiasi momento e senza preavviso.

Indice

1. SCOPO DEL CONTROLLER

Il controller è progettato per funzionare con motori passo-passo. Il dispositivo può essere controllato da un PLC (tramite RS-485 Modbus ASCII/RTU) o può operare in modalità standalone secondo il programma di esecuzione preimpostato.

Il controller fornisce un funzionamento a passo intero o microstepping fino a 1/256. Il controller garantisce un movimento fluido con un basso livello di vibrazioni e un'elevata precisione di posizionamento.

Il controller offre le seguenti modalità di controllo:

- Modalità di controllo a programma – funzionamento autonomo secondo l'algoritmo di esecuzione preimpostato o controllo in tempo reale da un PC o PLC tramite comandi impartiti tramite il protocollo Modbus.
- Controllo analogico della velocità – la velocità di rotazione del motore viene regolata dal potenziometro sul lato anteriore del controller.
- Modalità di controllo STEP/DIR – controlla la posizione del motore tramite segnali logici a impulsi.

Il controller dispone di 8 ingressi logici e 10 uscite (2 di queste uscite sono ad alta tensione). Lo stato di I/O può essere letto o impostato da un programma utente in esecuzione o direttamente tramite comandi Modbus. Un programma di esecuzione interno può essere scaricato e caricato sul controller tramite USB o RS-485.

Forniamo un software per la regolazione, l'assemblaggio o la modifica dei programmi di esecuzione del controller e per il caricamento del programma di esecuzione nella memoria del controller.

Il controller dispone di una protezione da surriscaldamento.

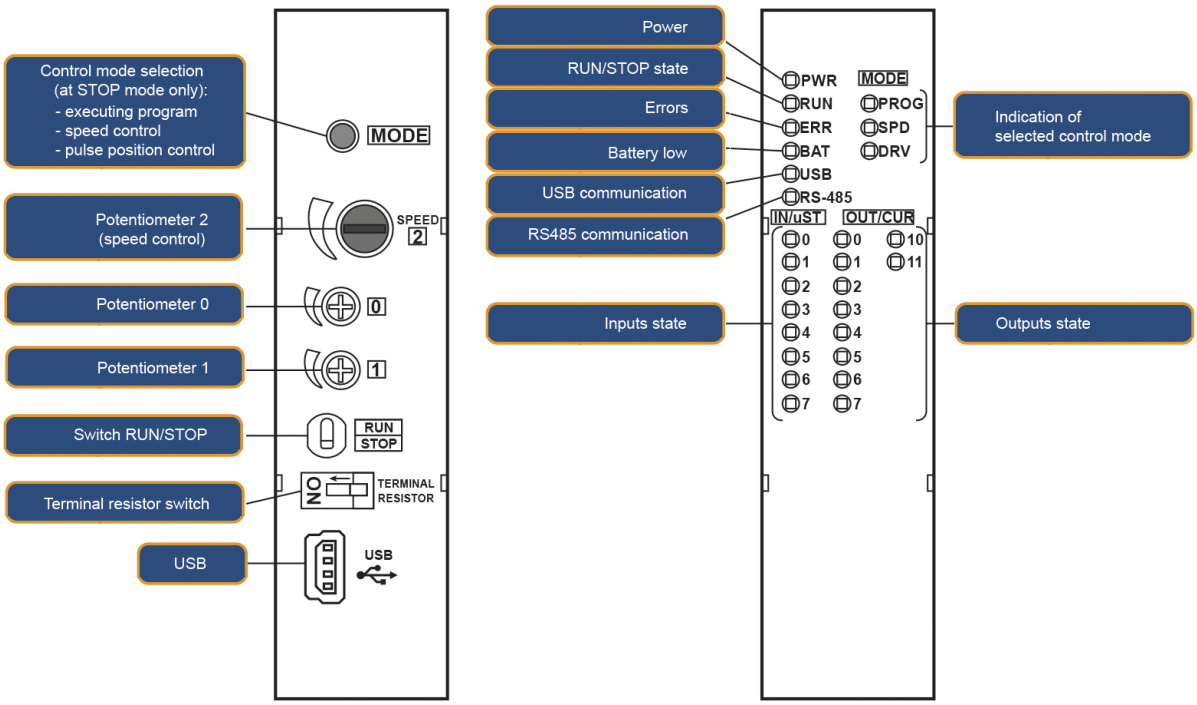


Fig. 1. Scopo degli elementi di controllo

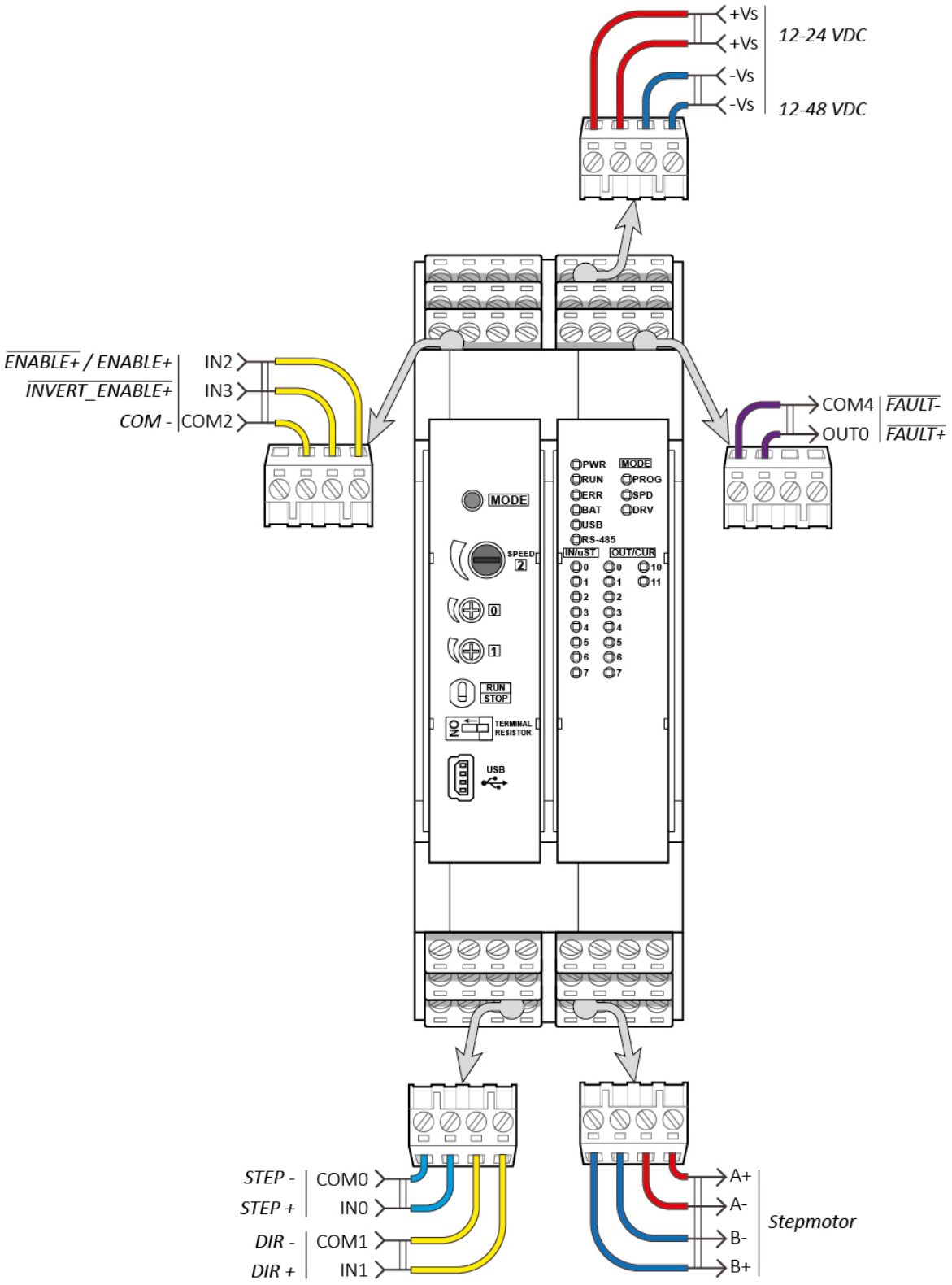


Fig. 2a. Assegnazione dei terminali – modalità di controllo del driver

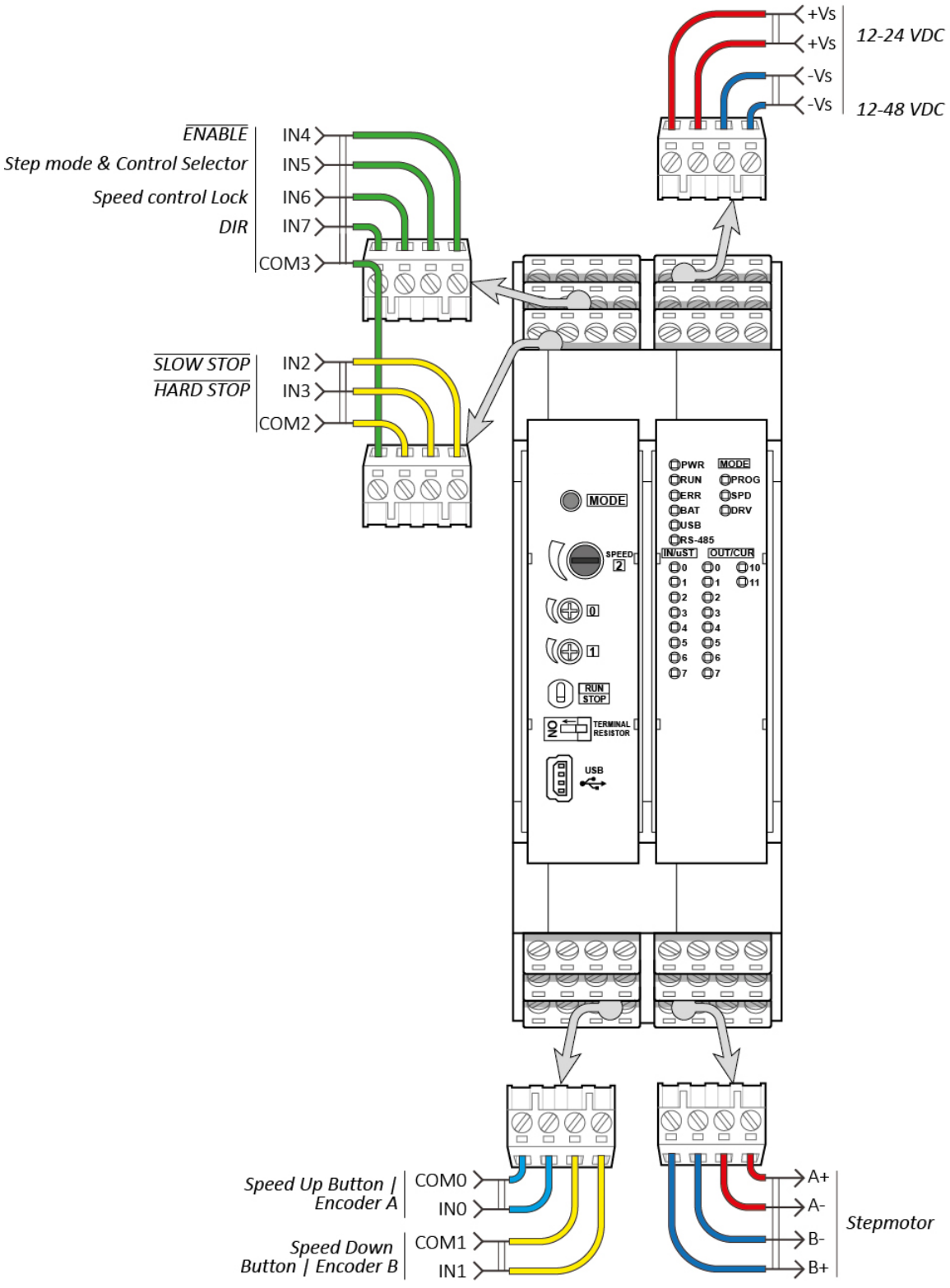


Fig. 3b. Assegnazione dei terminali – modalità di controllo della velocità

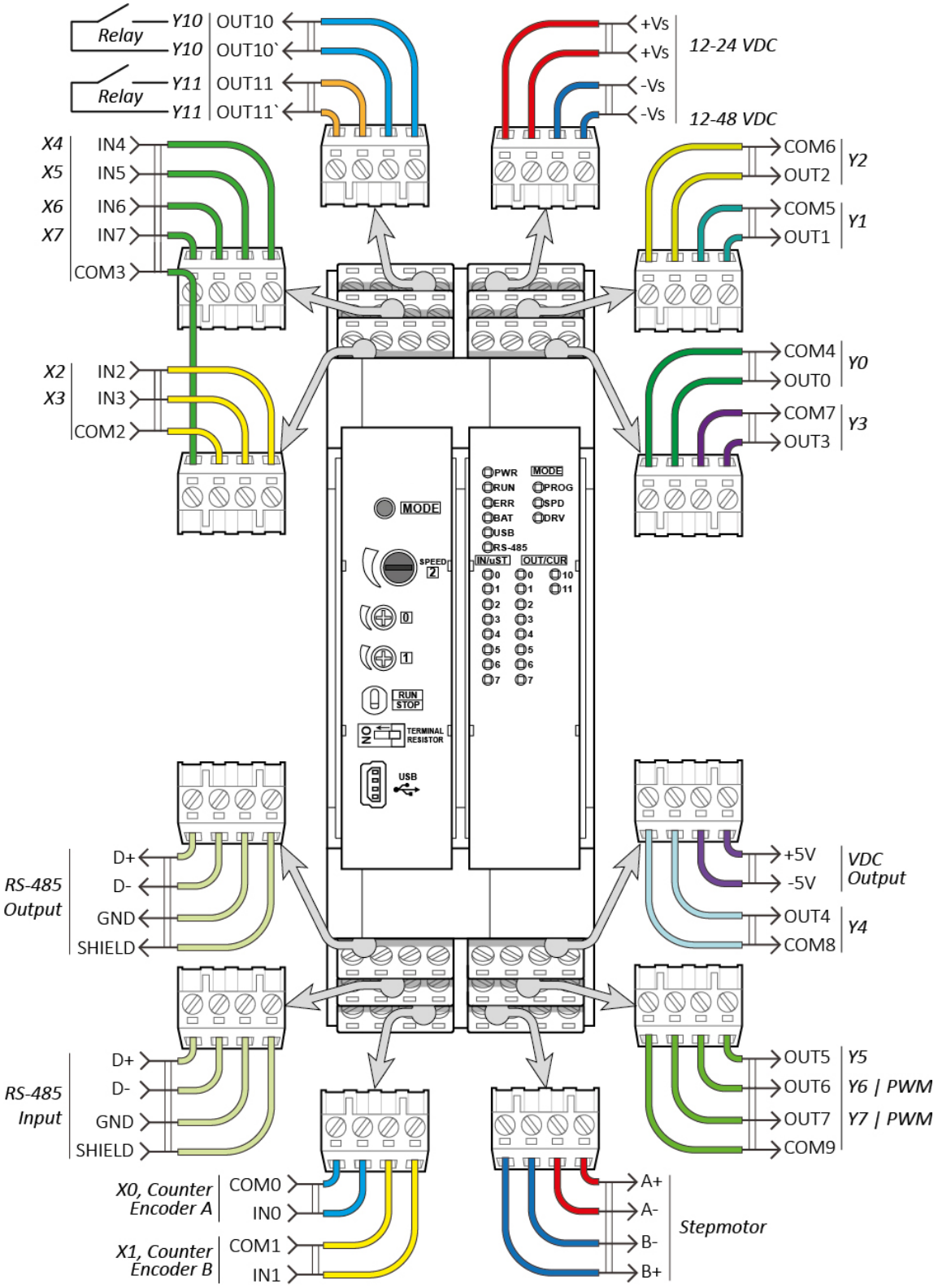


Fig. 4c. Assegnazione dei terminali – modalità di controllo del programma utente

Fig. 1 mostra il pannello frontale del controller con gli elementi di controllo e di indicazione. L'indicatore **PWR** indica la presenza della tensione di alimentazione. **RUN** indica lo stato attuale del controller (RUN o STOP). In modalità **PROG** (esecuzione di un programma utente) e in modalità **SPD** (controllo della velocità), lo stato attivo dell'indicatore **RUN** indica l'esecuzione del programma, lo stato inattivo indica lo stato di stop. In modalità **DRV** (modalità Step/Dir), lo stato inattivo di **RUN**

indica la possibilità di impostare i parametri del driver tramite gli elementi di controllo del controller. Lo stato attivo di **RUN** indica l'ingresso nella modalità operativa, le modifiche dei parametri sono disabilitate. La possibilità di passare tra le modalità PROG / SPD / DRV è disabilitata nello stato **RUN**. Nello stato **STOP**, la commutazione della modalità di controllo può essere effettuata tramite il pulsante **MODE**. **ERR** indica la presenza di errori. **BAT** indica batteria scarica all'interno dell'unità. **USB** e **RS485** indicano il processo di un frame Modbus trasmesso tramite USB e RS-485. In modalità **PROG** e **SPD**, i LED **IN0 ...IN7** indicano la presenza di un livello alto di un segnale logico all'ingresso corrispondente. Gli stati attivi degli indicatori **OUT0 ...OUT11** indicano lo stato aperto dell'uscita a transistor (vedere Fig. 5– Fig. 8). In modalità **DRV** (modalità Step/Dir) i LED degli ingressi **IN0 ...IN7** indicano l'impostazione del microstepping, le uscite **OUT0 ...OUT3** visualizzano l'impostazione della corrente di funzionamento, **OUT4 ...OUT7** visualizzano l'impostazione della corrente di mantenimento.

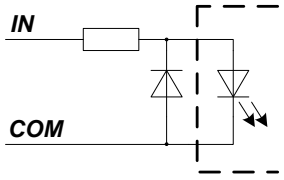


Fig. 5. Ingressi IN0 e IN1

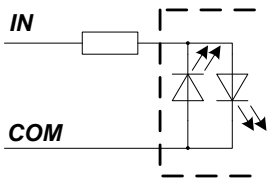


Fig. 6. Ingressi IN2 – IN7

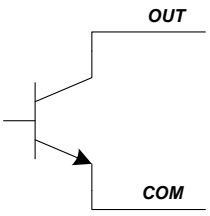


Fig. 7. Uscite OUT0 – OUT7

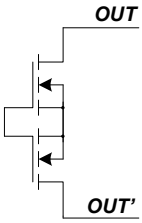


Fig. 8. Uscite OUT10 e OUT11

La posizione dei potenziometri **0, 1, 2 (SPEED)** viene convertita in valori a 12 bit, accessibili per un ulteriore utilizzo in un programma in esecuzione e leggibili tramite un comando Modbus. Nella modalità **SPD** (controllo della velocità), il potenziometro **2 (SPEED)** viene utilizzato per impostare la velocità di rotazione, il potenziometro 0 (velocità di accelerazione e decelerazione, 1 (corrente di lavoro del motore. Nella modalità **DRV** (Pulse/Dir), il potenziometro 2 viene utilizzato per impostare il microstepping, 0 – corrente di lavoro del motore, 1 - corrente di mantenimento.

L'interruttore **RUN/STOP** viene utilizzato per avviare e arrestare l'esecuzione del programma nelle modalità di controllo **PROG** e **SPD**. Viene utilizzato per passare dalle impostazioni dei parametri al funzionamento nella modalità di controllo **DRV**.

Fig. 2 (a–c) mostra i terminali del controller e il loro scopo a seconda della modalità di controllo.

2. SPECIFICHE TECNICHE

Caratteristica		Valore	
		min	max
Tensione di alimentazione, V_{DC}	RS 434540	12	36
	RS 434542 e RS 434543	12	49
Max. corrente di uscita per fase, A	RS 434540	0.15	1.5
	RS 434542	1.0	4.2
	RS 434543	2.8	8.0
Alto livello degli ingressi logici, VDC			2.4
Livello basso degli ingressi logici, VDC		0,7	
Tensione consigliata (miglior tempo di risposta) degli ingressi logici, VDC			5
Max. livello di tensione degli ingressi logici, VDC			24(1)
Tensione di uscita del transistor logico, VDC			80
Corrente massima dell'uscita del transistor logico, mA			50
Tensione di uscita del relè logico, VCA/VCC			350
Corrente massima dell'uscita relè logica DC (AC/DC), mA			250 (~120)
Corrente massima dell'uscita aggiuntiva +5VDC, mA			200
Durata del livello di tensione alto/basso dei segnali	Segnale STEP in modalità di controllo Step/Dir (DRV), ns	250(2)	
	segnali agli ingressi IN0 e IN1, ns	70(2)	
	segnali agli ingressi IN2…IN7, µs	5(2)	
Frequenza di generazione del segnale PWM, Hz		0,3	5000
Tempo di istruzione di base, µs(3)		20	

(1) – Quando si utilizza una tensione superiore a 8V per gli ingressi IN2.. .IN7 e superiore a 12V per gli ingressi IN0.. .IN1, è necessario installare resistori di limitazione della corrente.
– Resistenze consigliate per gli ingressi IN2.. .IN7: non inferiori a 270 Ohm quando utilizzate per una sorgente di segnale a 12V e non inferiori a 1 kOhm quando utilizzate per 24V.
– La resistenza consigliata per gli ingressi IN0 e IN1 è di 9,1 kOhm quando si utilizza una sorgente di segnale logico a 24V.

(2) a condizione di un livello di alta tensione di 5VDC

(3) - senza tener conto del ritorno alla linea zero, dell'impostazione delle uscite e della lettura degli ingressi

Informazioni aggiuntive

№	Caratteristica	Valore
1	Possibili velocità di trasmissione per la trasmissione dati RS-485	1200, 2400, 4800, 9600, 14400, 19200, 38400, 57600, 115200, 128000, 256000
2	Possibili impostazioni di microstepping	1/1, 1/2, 1/4, 1/8, 1/16, 1/32, 1/128, 1/256
3	Protocollo di comunicazione	Modbus RTU, Modbus ASCII
4	Linguaggio di programmazione	IL (Lista Istruzioni), LD* (Diagramma Ladder)

* *È necessario un software speciale per convertire LD in IL prima di scrivere il programma nel controller*

3. SEQUENZA OPERATIVA

La sequenza di funzionamento del controller è la seguente:

- lettura dei dispositivi esterni (ingressi logici, Modbus Coils);
- elaborazione del programma utente;
- impostazione dei nuovi stati dei dispositivi di uscita (uscite logiche, ingressi discreti Modbus, esecuzione del movimento).

Il programma utente è costituito da una sequenza di istruzioni di controllo (comandi) che determinano la funzionalità finale. Il controller esegue i comandi in sequenza, uno per uno. Il passaggio completo del programma viene ripetuto continuamente. Il tempo richiesto per un passaggio del programma è chiamato tempo di ciclo e i passaggi del programma sono chiamati scansione ciclica.

I controller possono operare in modalità tempo reale e possono essere utilizzati sia per la costruzione di nodi di automazione locale che di sistemi I/O distribuiti con l'organizzazione dello scambio dati tramite interfaccia RS-485 con protocollo Modbus.

Offriamo un software speciale per l'assemblaggio e il debug dei programmi utente, che non richiede risorse informatiche significative ed è uno strumento semplice per gli utenti. Vengono utilizzati due linguaggi di programmazione: LD (logica a contatti ladder o diagrammi ladder) e IL (elenco di istruzioni).

3.1. Il principio di funzionamento dei circuiti a relè e dei diagrammi a ladder nel controller

Il linguaggio dei diagrammi ladder è un derivato del diagramma a contatti di relè in una rappresentazione semplificata. I circuiti a contatti di relè nel controllore hanno una serie di componenti di base, come: contatto normalmente aperto, contatto normalmente chiuso, bobina (uscita), timer, contatore, ecc., nonché istruzioni applicate: funzioni matematiche, comandi di controllo motore, elaborazione dati e un gran numero di funzioni e comandi speciali. Possiamo supporre che il controllore sia costituito da decine o centinaia di relè, contatori, timer e memoria separati. Tutti questi contatori, timer, ecc. fisicamente non esistono, ma sono modellati dal processore e sono progettati per scambiare dati tra funzioni integrate, contatori, timer, ecc.

Il linguaggio logico a contatti di relè nel controllore è molto simile ai circuiti elettrici a contatti di relè di base se confrontiamo i simboli grafici utilizzati. Nei circuiti a contatti di relè possono esserci due tipi di logica: combinata, ovvero circuiti costituiti da frammenti indipendenti l'uno dall'altro, e logica sequenziale, dove tutti i passaggi del programma sono interconnessi e il circuito non può essere parallelizzato.

Logica combinata

Il primo segmento del circuito è costituito da un contatto normalmente aperto X0 e da una bobina Y0, che determina lo stato dell'uscita Y0. Quando lo stato del contatto X0 è aperto (logico "0"), anche lo stato dell'uscita Y0 è aperto (logico "0"). Quando il contatto X0 è chiuso, anche l'uscita Y0 cambia il suo stato in chiuso (logico "1").

Il secondo segmento del circuito è costituito da un contatto normalmente chiuso X1 e da una bobina Y1, che determina lo stato dell'uscita Y1. Nello stato normale del contatto X1, l'uscita Y1 sarà chiusa (logico "1"). Quando lo stato del contatto X1 cambia in aperto, anche l'uscita Y1 cambia il suo stato in aperto.

Al terzo segmento del circuito, lo stato dell'uscita Y2 dipende da una combinazione degli stati dei tre contatti di ingresso X2, X3 e X4. L'uscita Y2 è chiusa quando X2 è disattivato e X4 è attivato, oppure quando X3 e X4 sono attivati.
Lo schema generale è una combinazione di tre segmenti, che operano indipendentemente l'uno dall'altro.

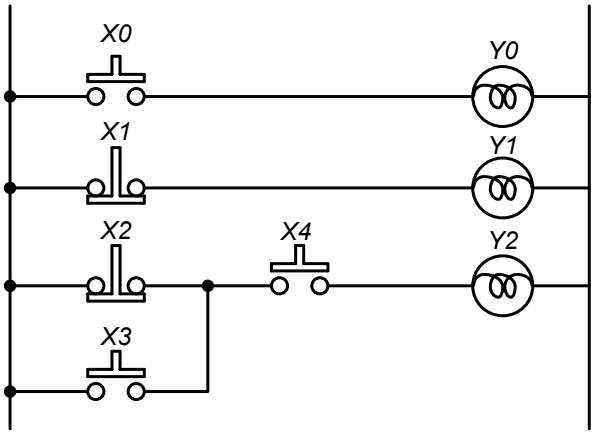


Fig. 9. Circuito a relè

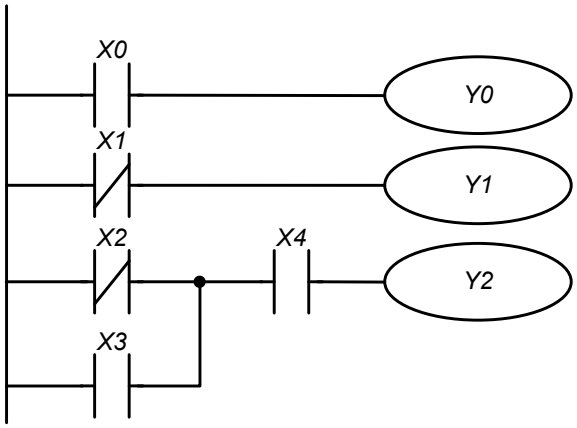


Fig. 10. Diagramma ladder nel controller

Logica sequenziale
Nei circuiti a logica sequenziale, il risultato dell'esecuzione in un passaggio precedente è una condizione di ingresso per il passaggio successivo.In altre parole, un'uscita al passaggio precedente è un ingresso al passaggio successivo.

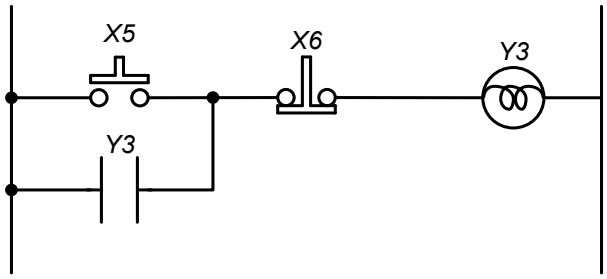


Fig. 11. Circuito a relè

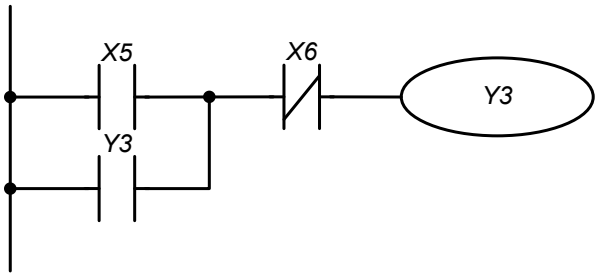


Fig. 12. Diagramma ladder nel controller

Quando il contatto X5 è chiuso, l'uscita Y3 cambia il suo stato in chiuso. Tuttavia, quando X5 viene riaperto, Y3 mantiene il suo stato chiuso fino al momento in cui X6 viene aperto. In questo circuito, l'uscita Y3 è autobloccante.

3.2. Differenze tra la logica dei circuiti a contatti di relè reali e i diagrammi ladder nel controllore

Tutti i processi di controllo specificati vengono eseguiti simultaneamente (in parallelo) nei circuiti elettrici a contatti di relè convenzionali. Ogni cambiamento nello stato dei segnali di ingresso influisce immediatamente sullo stato dei segnali di uscita.

Una modifica dello stato dei segnali di ingresso verificatasi durante il passaggio corrente del programma nel controllore viene riconosciuta solo al ciclo di programma successivo. Questo comportamento viene attenuato grazie al breve tempo di ciclo.

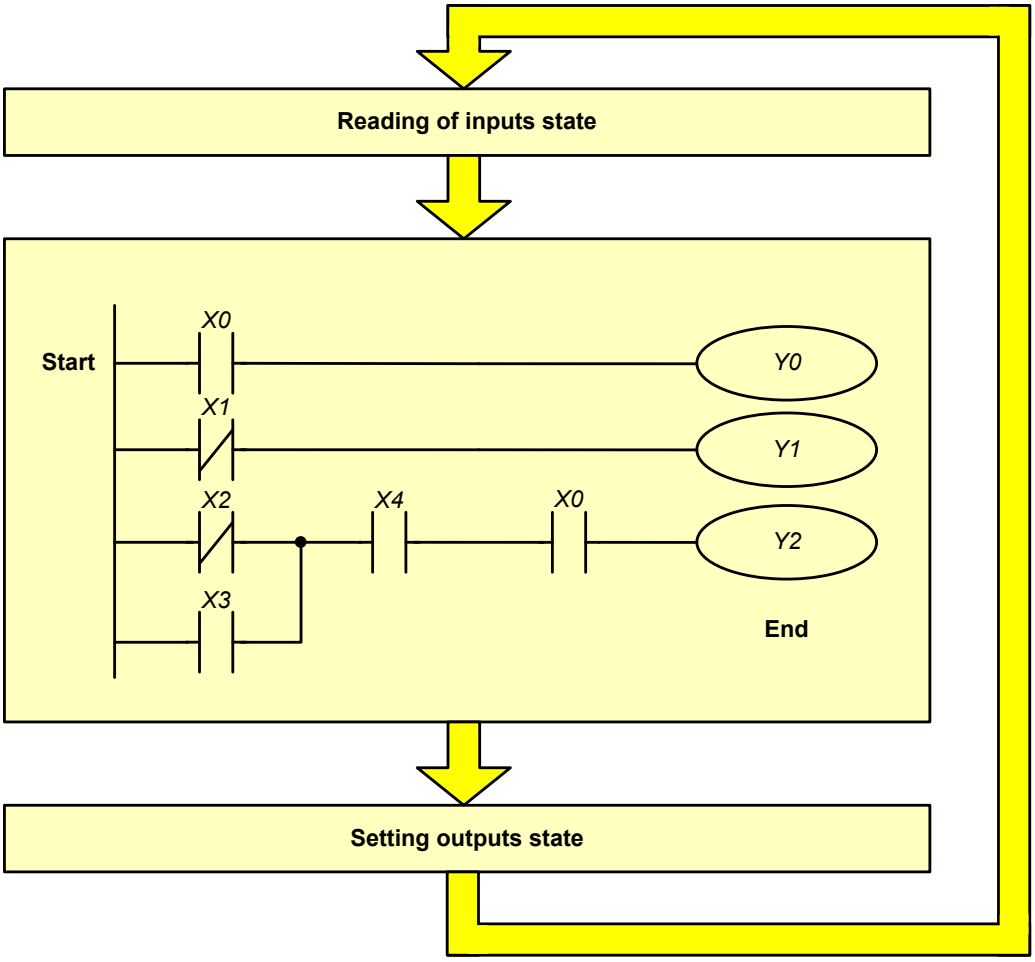


Fig. 13. Sequenza operativa nel controllore

Durante il funzionamento, il controllore legge continuamente lo stato corrente degli ingressi e modifica lo stato delle uscite (on/off) a seconda del programma utente.

La Fig. 13 mostra un diagramma di flusso di un ciclo di programma.

Nella prima fase, il controllore legge lo stato degli ingressi fisici e virtuali (i cui stati sono impostati tramite Modbus Coils) e li memorizza nel buffer nella memoria interna del controllore.

Nella seconda fase, lo stato degli ingressi memorizzati nel buffer viene elaborato e lo stato delle uscite nella memoria del controllore cambia in base a un determinato programma utente. Pertanto, tutte le modifiche delle uscite avvengono senza modificare la loro condizione fisica. Durante questa fase, lo stato degli ingressi fisici e virtuali può cambiare, ma la successiva memorizzazione nel buffer dello stato aggiornato avverrà nella prima fase del ciclo successivo del programma utente.

Nella terza fase, il controllore modifica lo stato delle uscite fisiche e virtuali.

Un'altra differenza tra la logica a contatti di relè del controllore e i circuiti elettrici a contatti di relè convenzionali è che i programmi utente vengono eseguiti in righe solo da sinistra a destra e

dall'alto verso il basso. Ad esempio, un circuito con una direzione di corrente inversa (sezione a-b in) provocherà un errore durante la compilazione nel controllore.

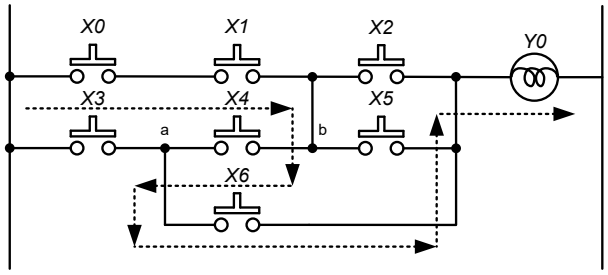


Fig. 14. Circuito dei contatti del relè elettrico

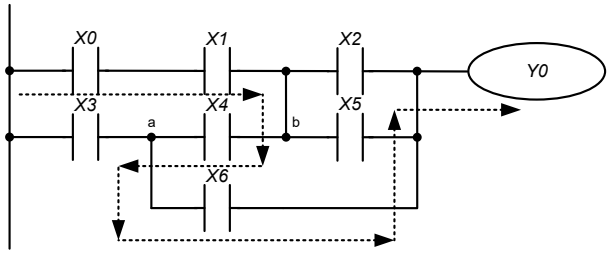


Fig. 15. Circuito dei contatti a relè del controllore

Un errore nella 3ªriga

3.3. Operandi

Tutti gli oggetti interni (dispositivi) del controllore – operandi – sono suddivisi in alcuni tipi diversi e hanno indirizzi. Ogni tipo ha la sua designazione e il suo formato, che determina lo spazio che occupa nella memoria del controllore. Così, ad esempio, i relè di ingresso sono denominati "X" e hanno un formato a 1 bit, mentre i registri dati di uso generale sono denominati "D" e hanno un formato a 16 bit (1 word) o 32 bit (2 word).

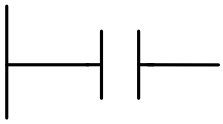
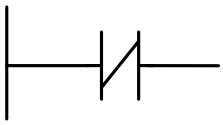
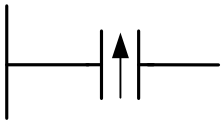
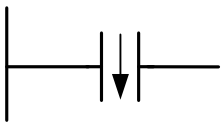
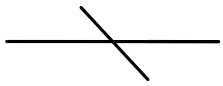
Tipo e nome dell'operando		Descrizione
Ingresso	X	Relè di ingresso. Determinano lo stato dei dispositivi bit esterni, che sono collegati ai terminali di ingresso del controller, e lo stato degli ingressi virtuali, il cui stato può essere impostato tramite protocollo Modbus. Questi operandi possono assumere uno dei due possibili valori: 0 o 1. L'indirizzamento è ottale: X0, X1.. . X7, X10, X11, ...
Uscita	Y	Relè di uscita. Determinano lo stato dei terminali di uscita del controllore, a cui è collegato il carico, e lo stato delle uscite virtuali, il cui stato può essere letto tramite il protocollo Modbus. Nel programma possono essere contatti o bobine. Questi operandi possono assumere uno dei due possibili valori: 0 o 1. L'indirizzamento è ottale: Y0, Y1.. .Y7, Y10, Y11, ...
Merker	M	Relè ausiliari. Si tratta di una memoria per risultati intermedi binari. Nel programma utente possono essere contatti o bobine. Questi operandi possono assumere uno dei due possibili valori: 0 o 1. L'indirizzamento è decimale: M0, M1.. .M7, M8, M9,...

Temporizzatore	T	Relè temporizzato. Il programma può essere utilizzato per la memorizzazione del valore corrente del timer ed ha un formato a 16 bit. Inoltre, questi operandi possono essere utilizzati come contatti e assumono uno dei due stati: 0 o 1. L'indirizzamento è decimale: T0, T1 ...T64
Contatore	C	Un contatore viene utilizzato per implementare il conteggio. Il programma può essere utilizzato per la memorizzazione del valore corrente del contatore ed ha un formato a 16-bit o 32-bit, e può anche essere utilizzato come un contatto ed assumere uno dei due possibili significati: 0 o 1. L'indirizzamento è decimale: C0, C1.. .C66
Costante decimale	K	Determina un numero in decimale
Costante esadecimale	H	Determina un numero esadecimale
Costante a virgola mobile	F	Determina un numero a virgola mobile
Registro dati	D	Memorizzazione dati. Formato a 16-bit o 32-bit. L'indirizzamento è decimale: D0, D1,..., D391. Per i dati a 32-bit, un elemento occupa due registri. Ad esempio, per la lettura di dati a 32-bit dal registro D10, i dati vengono letti dai registri D10 e D11.
Registro indice	A	Memorizzazione dati per risultati intermedi e identificazione indice. Formato a 16 bit. Indirizzamento: A0 – A7, B0 – B7 decimale
	B	
Puntatore	P	Un indirizzo per la chiamata del sottoprogramma. Decimale.
Puntatore di interrupt	I	Indirizzo di gestione degli interrupt. Decimale.

3.4. Simboli grafici delle istruzioni di controllo in un diagramma ladder

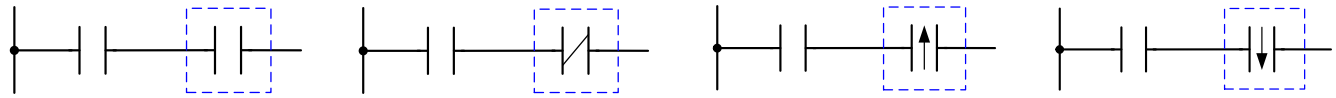
Un circuito a contatti di relè è costituito da una linea verticale a sinistra e da linee orizzontali che si estendono a destra. La linea verticale a sinistra è una linea bus, le linee orizzontali sono linee di comando = passi. Ci sono simboli di condizioni di ingresso sulle linee di comando che portano ai comandi (istruzioni) situati a destra. Le combinazioni logiche di queste condizioni di ingresso determinano quando e come vengono eseguiti i comandi a destra.

I seguenti simboli vengono utilizzati nei circuiti a contatti di relè:

Simbolo	Descrizione	Comando	Operandi
	Contatto di ingresso – normalmente aperto	LD	X, Y, M, T, C
	Contatto di ingresso – normalmente chiuso	LDI	X, Y, M, T, C
	Contatto impulso di ingresso – fronte di salita	LDP	X, Y, M, T, C
	Contatto impulso di ingresso – fronte di discesa	LDF	X, Y, M, T, C
	Segnale di uscita (bobina)	OUT	Y, M
	Istruzioni di base e applicative	Fare riferimento al capitolo 7. Istruzioni per l'applicazione	Fare riferimento al capitolo 7. Istruzioni per l'applicazione
	Inversione logica	INV	-

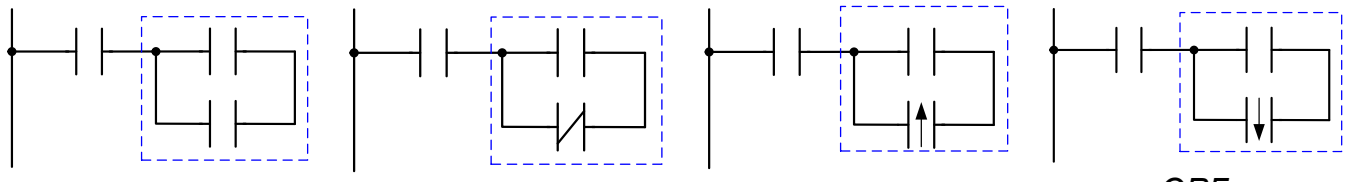
I contatti di ingresso possono essere combinati in blocchi seriali e paralleli:

Connessioni seriali:



AND *ANI* *ANDP* *ANDF*

Connessioni parallele:



OR *ORI* *ORP* *ORF*

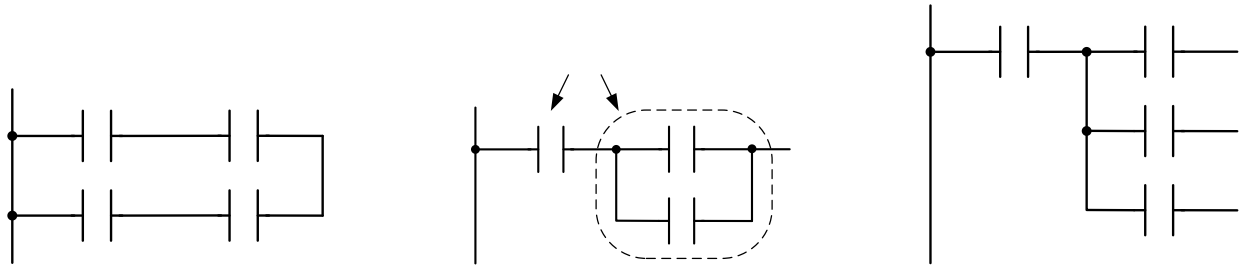


Fig. 16. Simboli grafici delle istruzioni di controllo

La scansione del programma inizia dall'angolo in alto a sinistra del diagramma e termina nell'angolo in basso a destra. Il seguente esempio illustra la sequenza di un programma:

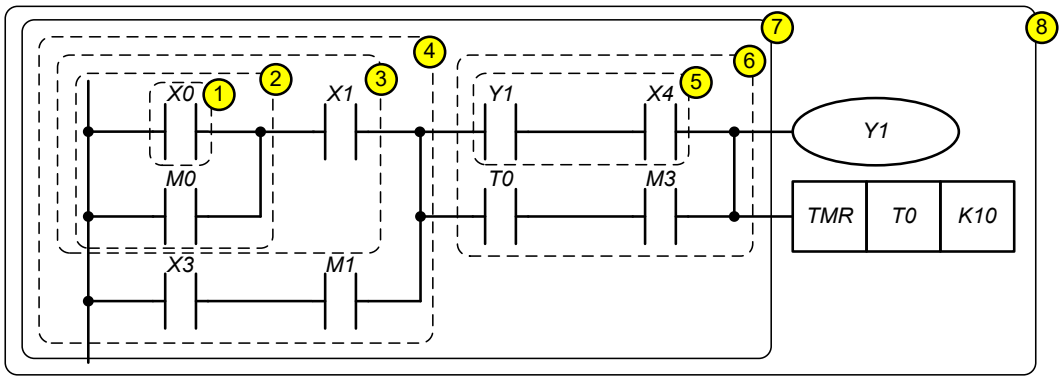
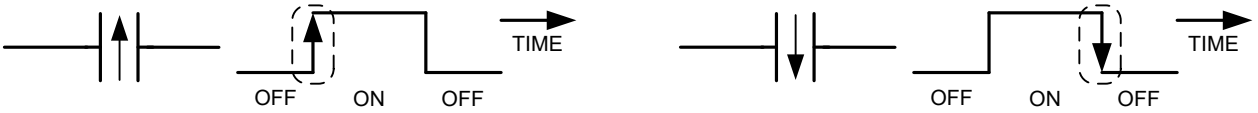


Fig. 17. Sequenza di un programma

1	LD	X0	
2	OR	M0	
3	AND	X1	
4	LD	X3	
	AND	M3	
	ORB		
5	LD	Y1	
	AND	X4	
6	LD	T0	
	AND	M3	
	ORB		
7	ANB		
8	OUT	Y1	
	TMR	T0	K10

I simboli dei segnali di ingresso con fronte di salita (quando un segnale passa da 0 a 1) e con fronte di discesa (quando un segnale passa da 1 a 0) sono spiegati di seguito:



Contatto a fronte di salita Contatto sul fronte di discesa

Fig. 18. Filtraggio del fronte

I comandi di blocco logico ANB e ORB non corrispondono a condizioni specifiche sul circuito a contatti di relè, ma descrivono la relazione tra i blocchi. Il comando ANB esegue l'operazione **AND LOGICO** sulle condizioni di esecuzione prodotte da due blocchi logici.

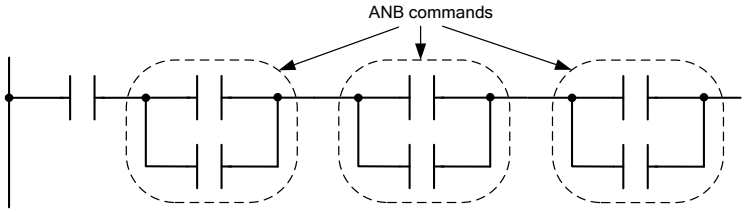


Fig. 19. Istruzione ANB

Il comando ORB esegue un'operazione **OR LOGICO** sulle condizioni di esecuzione prodotte da due blocchi logici.

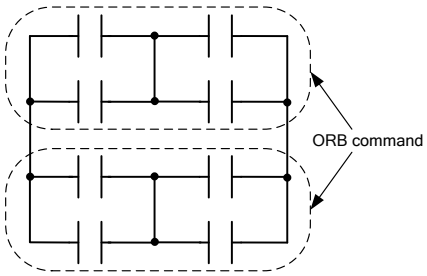


Fig. 20. Istruzione ORB

3.5. Convertire i circuiti dei contatti a relè (LD) in codice mnemonico (IL)

La figura seguente mostra un programma presentato sotto forma di simboli di contatti di relè (LD) e un elenco di istruzioni - codice mnemonico (IL). La figura mostra la sequenza di conversione del diagramma ladder (LD) nel codice eseguito dal controller (IL).

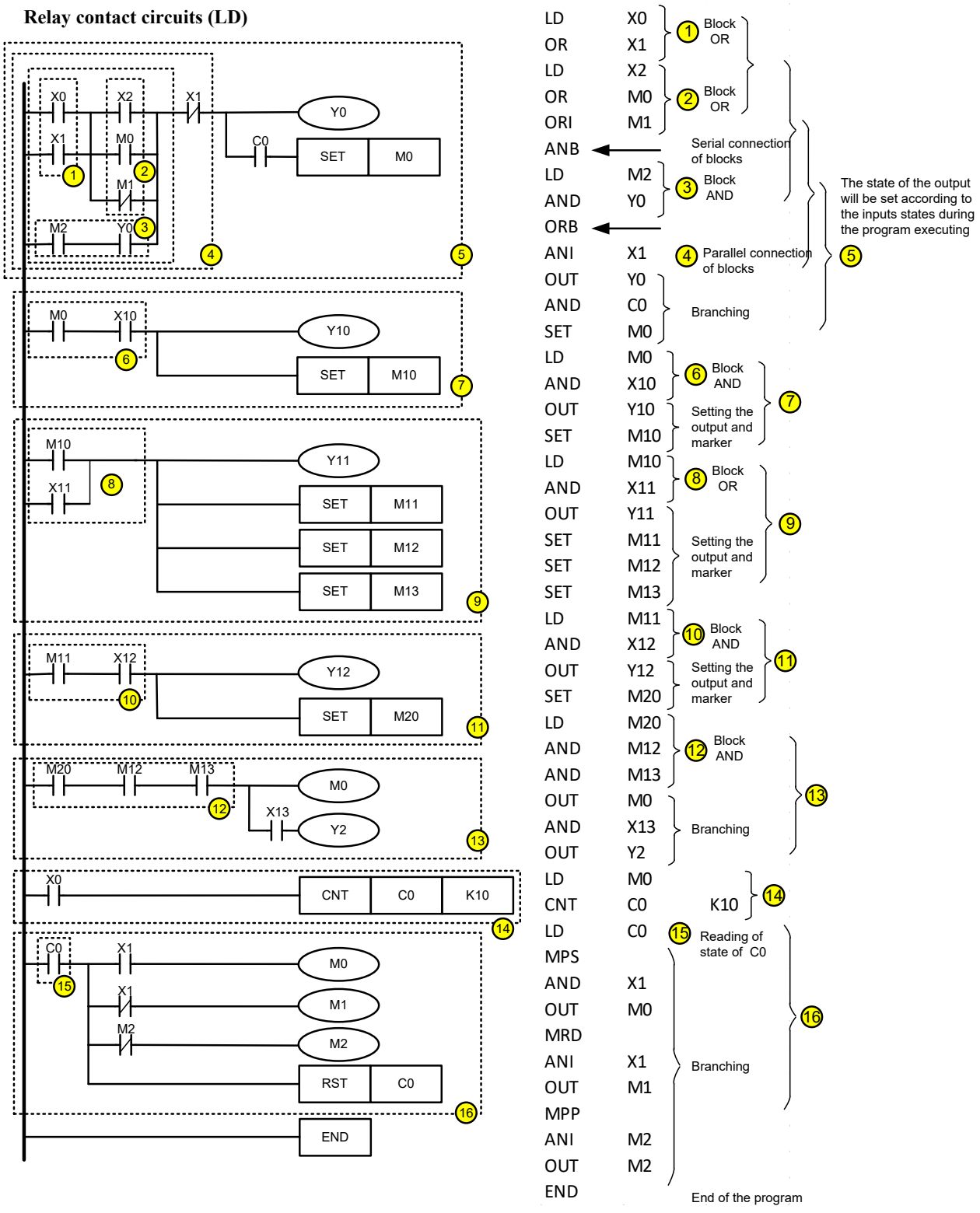


Fig. 21.

Conversione di LD in IL

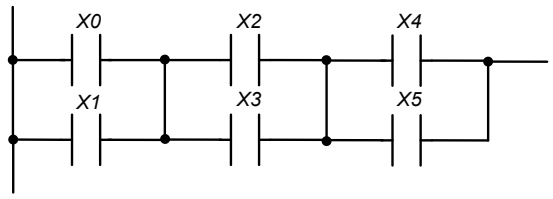
L'elaborazione del circuito a contatti di relè inizia nell'angolo in alto a sinistra e termina in basso a destra, tuttavia, potrebbero esserci eccezioni e varie opzioni per la conversione in codice mnemonico, come mostrato nei seguenti esempi:

Esempio 1

Il seguente diagramma ladder può essere convertito in un elenco di istruzioni in due modi diversi, ma il risultato sarà identico (Fig. 22).

Il primo metodo di codifica è preferibile in quanto il numero di blocchi logici è illimitato.

Il secondo metodo è limitato dal numero massimo di blocchi logici (il numero massimo di blocchi è 8).

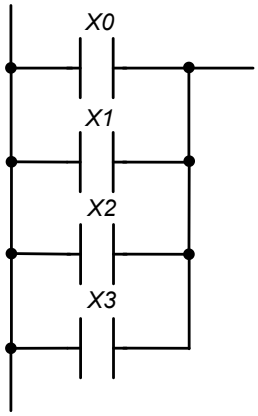


Metodo 1		Metodo 2	
LD	X0	LD	X0
OR	X1	OR	X1
LD	X2	LD	X2
OR	X3	OR	X3
ANB		LD	X4
LD	X4	OR	X5
OR	X5	ANB	
ANB		ANB	

Fig. 22. Diversi metodi di utilizzo delle istruzioni ANB

Esempio 2

Di seguito sono mostrati diversi metodi di codifica dei contatti collegati in parallelo (Fig. 23).



Metodo 1		Metodo 2	
LD	X0	LD	X0
OR	X1	LD	X1
OR	X2	LD	X2
OR	X3	LD	X3
		ORB	
		ORB	
		ORB	

Fig. 23. Diversi metodi di utilizzo delle istruzioni ORB

Il primo metodo di conversione del diagramma ladder in un elenco di istruzioni è il più preferibile dal punto di vista dell'utilizzo della RAM del controller.

4. Funzionalità del controller

4.1. Panoramica degli operandi

Tipo	Operando			Intervallo di indirizzi		Funzione
Relè (memoria a 1 bit)	**X**	Relè di ingresso esterni	Ingressi fisici	X0…X7	Max. 128 punti	Ingressi del controller
			Ingressi virtuali (Bobina Modbus)	X10…X177		
	Y	Relè di uscita esterni	Uscite fisiche	Y0…Y7	Max. 128 punti	Uscite del controller
			Uscite virtuali (Ingressi Discreti Modbus)	Y10…Y177		
	M	Relè interni (marcatori)	Uso generale	M0…M99, M111…M127	Max. 128 punti	Memoria binaria intermedia. Corrisponde ai relè intermedi nei circuiti elettrici.
			Scopo speciale	M100…M110		
	T	Timer	Risoluzione 100 ms	T0…T47 (T46, T47 – cumulativo)	Max. 64 punti	Utilizzati come contatti (T), che si chiudono quando il timer corrispondente raggiunge il valore impostato (comando TMR)
			Risoluzione 10 ms	T48…T63 (T62, T63 – cumulativo)		
	C	Contatori	Incrementale generico	C0…C63	Max. 66 punti	Utilizzati come contatti (C), che si chiudono quando il contatore corrispondente raggiunge il valore impostato (comando CNT)
			Impulsi esterni	C64, C65		
Registri (memoria a 16 bit)	**T**	Valore corrente del timer		64 punti (T0…T63)		Registri per la memorizzazione dei valori attuali del timer
	C	Valore corrente del contatore		66 – Contatori a 32 bit		Registri per la memorizzazione dei valori correnti del contatore
	D	Registri dati	Uso generale	D0…D319, D385…D391		Utilizzato per memorizzare i dati. Registri speciali

			Non volatile(1)	D320…D351	Max. 384 punti	configurano il controller e ne visualizzano lo stato.
			Speciale	D352…D384		
	A	Registri indice	Minori a 16 bit	A0…A7	Max. 16 punti	Può essere utilizzato per l'indicazione di indice
	B		Maggiori 16 bit	B0…B7		
Puntatori	**P**	Puntatori per le istruzioni CALL, CJ		32 punti (P0…P31)		Etichette per le istruzioni di transizioni e sottoprogrammi
	I	Interruzioni	Comunicazione	I0	Max. 15 punti	Etichette per il sottoprogramma per la gestione delle interruzioni
			Temporizzato	I1…I100 (Max. 4 punti)		
			Esterno	I1000…I1007		
			Driver	I2000, I2001		
Costanti	**K**	Costanti decimali		K-32768 ...K32767 (funzioni a 16-bit) K-2147483648 ...K2147483647 (funzioni a 32-bit)		
	H	Costanti esadecimali		H0000...HFFFF (funzioni a 16 bit) H00000000...HFFFFFFFF (funzioni a 32-bit)		
	F	Costante a virgola mobile		F±1.175494351 E-38… 3.402823466 E+38 (Funzioni a 32 bit solamente)		

(1) – La memorizzazione dei dati è fornita dall'alimentatore interno CR2032.

4.2. Indirizzamento e funzioni degli ingressi [X] e delle uscite [Y]

Gli ingressi e le uscite nel programma utente sono rappresentati da operandi. Specificando l'indirizzo dell'operando, è possibile fare riferimento agli ingressi e alle uscite fisiche e virtuali del controller durante la programmazione.

Gli ingressi/uscite discreti sono indirizzati nel sistema ottale, il che significa che i numeri 8 e 9 non vengono utilizzati per ingressi e uscite.

Funzione dei relè di ingresso X

I relè di ingresso X leggono lo stato dei dispositivi fisici esterni (pulsanti, interruttori, contatti di relè, ecc.) direttamente collegati ai terminali di ingresso del controller.Ogni ingresso X può essere utilizzato nel programma un numero illimitato di volte.

Funzione dei relè di uscita Y

I relè di uscita Y controllano lo stato dei contatti di uscita fisici del controller e quindi i dispositivi di carico (lampade, bobine di relè, ecc.) direttamente collegati ai terminali di uscita del controller.

Ogni uscita Y può essere utilizzata nel programma un numero illimitato di volte, ma si consiglia di utilizzare la bobina di uscita Y nel programma non più di una volta, perché quando la bobina Y viene utilizzata più volte, lo stato di uscita è determinato dall'ultima Y nella scansione.

Lo stato dei segnali I/O può essere letto nel programma da diverse istruzioni.

Il processo di gestione dei segnali I/O nel controller:

Ingressi:

1. Il controller legge lo stato dei dispositivi di ingresso esterni e lo memorizza all'inizio di ogni ciclo di scansione.
2. Le modifiche nello stato di ingresso durante il ciclo non verranno accettate se l'impulso di ingresso è molto breve (inferiore al tempo di una scansione).

Programma:

3. Il controller esegue il programma a partire dalla riga 0 e memorizza lo stato di tutti gli operandi nella memoria dell'oggetto.

Uscite:

4. Dopo aver eseguito l'istruzione END, lo stato del relè di uscita Y viene scritto nella memoria delle uscite e gli stati dei contatti di uscita verranno modificati.

4.3. Indirizzamento e funzione dei relè interni [M]

Per memorizzare i risultati binari dei collegamenti logici (stati del segnale "0" o "1"), viene utilizzata una memoria intermedia (relè interno) all'interno del programma. Corrispondono ai relè intermedi nei sistemi di controllo basati sulla logica a relè.

Due tipi di relè interni sono utilizzati nel controller:

1. Uso generale, che non viene salvato quando l'alimentazione viene spenta;
2. Scopo speciale, che fornisce all'utente funzionalità aggiuntive.

I relè interni sono programmati come uscite. Possono essere utilizzati nel programma un numero illimitato di volte. L'indirizzamento dei relè interni è in formato decimale.

Assegnazione di merker speciali:

Merker	Funzione
M100…M107	Questi relè ausiliari vengono utilizzati solo in combinazione con le interruzioni dagli ingressi esterni I1000 ... I1007, rispettivamente. Il valore del merker corrisponde allo stato dell'ingresso fisico (IN0 ... IN7) nel momento in cui l'interruzione è stata elaborata (I1000 ... I1007). I valori X0 ... X7 vengono aggiornati solo all'inizio della successiva scansione del programma utente. Ad esempio, dopo essere entrati nel gestore di interrupt I1004, è possibile determinare lo stato dell'ingresso IN4 richiedendo lo stato di M104 (LD M104) (il valore di X4 non è rilevante in questo caso).

M108	Il fronte di salita di questo relè ausiliario indica il completamento dell'inizializzazione delle periferiche del controller. Durante il successivo funzionamento, il merker mantiene un valore alto. Il reset del merker reinizializza il controller. Ad esempio, dopo aver ridefinito le uscite dai generatori di segnale PWM e gli ingressi dai contatori di impulsi, è necessaria una reinizializzazione. In questo caso, è necessario resettare M108.
M109	Impostando il merker si attiva l'indicazione “ERR” sul pannello frontale del controller, resettandolo si disattiva.
M110	Impostando questo merker e poi resettando M108 si otterrà un riavvio completo del controller.

4.4. Indirizzamento e funzione dei timer [T]

Alcuni processi di controllo richiedono un relè temporizzato. Molti sistemi controllati da relè utilizzano relè temporizzati che si attivano con un ritardo. Il controller utilizza elementi di memoria interni per questi scopi, chiamati timer. Le caratteristiche dei timer possono essere determinate nel programma.

L'indirizzamento dei timer è decimale.

T	Timer	Risoluzione 100 ms	T0…T47 (T46, T47 – cumulativi)	Max. 64 punti
		Risoluzione 10 ms	T48…T63 (T62, T63 – cumulativi)	

L'impostazione del tempo richiesta è determinata da una costante decimale K, che indica il numero di passi temporali contati (discreti).

Esempio: un timer con risoluzione di 100 ms impostato come K5, il valore effettivo dell'impostazione sarà 5 x 100 = 500 ms.

Il timer funziona con un ritardo all'eccitazione. Si attiva con lo stato del contatto = 1. Dopo aver contato il valore di tempo impostato, il timer imposta il contatto di ingresso T corrispondente sullo stato "1". Il timer torna allo stato di disattivazione e reimposta il suo valore corrente quando il suo contatto di ingresso viene impostato su "0".

L'impostazione del tempo può essere eseguita anche indirettamente tramite un numero decimale registrato in precedenza nel registro dati D.

Nei controller, il timer inizia a contare immediatamente quando esegue il comando TMR.

Spiegazione del funzionamento di due tipi di timer:

Timer per uso generale

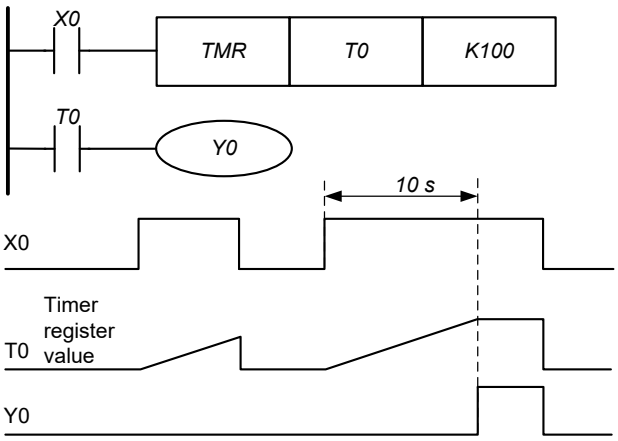


Fig. 24. Principio di funzionamento del timer generico

Quando l'ingresso X0 assume lo stato "1", inizia il conteggio del tempo impostato. Dopo aver raggiunto i 10 secondi programmati, l'uscita Y0 assume lo stato "1". Il timer si spegne e il registro T0 viene azzerato non appena l'ingresso X0 assume lo stato "0".

Timer cumulativo

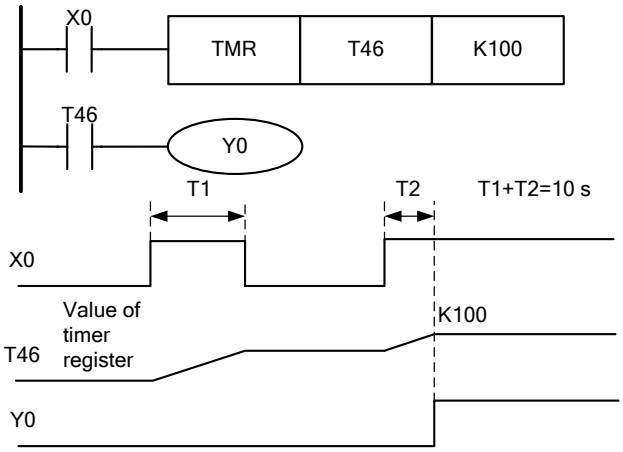


Fig. 25. Principio di funzionamento del timer cumulativo

Oltre ai timer generici, il controller dispone di timer accumulativi che, dopo la disattivazione del collegamento logico di controllo, salvano il valore del tempo accumulato.

4.5. Indirizzamento e funzione dei contatori [C]

È necessario contare gli impulsi (aggiungere o sottrarre) in alcuni processi di controllo. Molti sistemi controllati da relè utilizzano contatori di impulsi per questo scopo. Il controller utilizza due tipi di elementi di memoria interni (contatori).

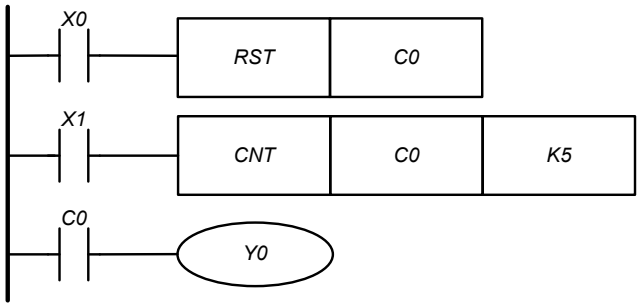
L'indirizzamento dei timer è decimale.

C	Contatori	Incrementale per uso generale	C0…C63	Max. 66 punti
		Impulsi esterni (hardware)	C64, C65	

Funzione dei contatori:

Quando il segnale di ingresso del contatore cambia il suo stato da 0 a 1, il valore corrente del contatore C aumenta di uno. Quando diventa uguale al valore impostato (set point), il contatto di lavoro del contatore si attiva.

```
LD      X0
RST     C0
LD      X1
CNT     C0      K5
LD      C0
OUT     Y0
```



Il contatore viene resettato quando X0=1: il valore corrente del registro C0 = 0, il contatto C0 è aperto.

Dopo aver commutato X1 da 0 a 1, il valore di C) incrementa di uno.

Quando il valore del registro C0 = 5, i contatti C0 e Y0 sono chiusi e tutti gli impulsi successivi in ingresso non vengono contati.

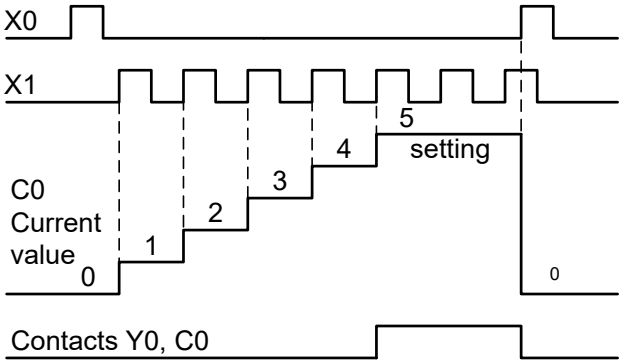
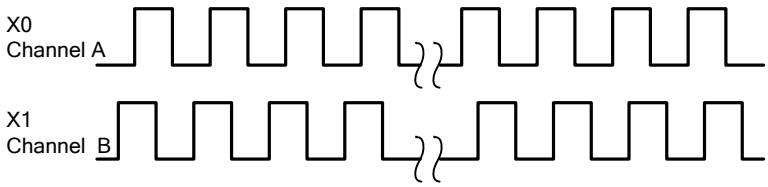


Fig. 26. Principio di funzionamento del contatore

I contatori per uso generale non contano oltre una soglia, a differenza dei contatori hardware, che non contano sul segnale di ingresso, ma solo sullo stato fisico dell'ingresso discreto a cui si riferiscono. A seconda del valore del registro di configurazione D355, i contatori possono essere configurati nel modo seguente:

Valore del registro D355	Configurazione
0	Il valore predefinito. X0 e X1 (IN0 e IN1) funzionano come ingressi digitali.
1	L'ingresso discreto X0 si riferisce al contatore C64, che conta i fronti di salita degli impulsi. X1 funziona come un ingresso discreto.
2	L'ingresso discreto X0 si riferisce al contatore C64, che conta i fronti di discesa degli impulsi. X1 funziona come un ingresso discreto.
3	L'ingresso discreto X0 si riferisce al contatore C64, che conta sia i fronti di salita che i fronti di discesa degli impulsi. X1 funziona come un ingresso discreto.

4	X0 funziona come un ingresso discreto. L'ingresso discreto X1 si riferisce al contatore C65, che conta i fronti di salita degli impulsi.
5	X0 funziona come un ingresso discreto. L'ingresso discreto X1 si riferisce al contatore C65, che conta i fronti di discesa degli impulsi.
6	Gli ingressi discreti X0 e X1 si riferiscono rispettivamente ai contatori C64 e C65. I contatori contano i fronti di salita degli impulsi.
7	Gli ingressi discreti X0 e X1 si riferiscono rispettivamente ai contatori C64 e C65. C64 conta i fronti di discesa degli impulsi, C65 conta i fronti di salita degli impulsi.
8	Gli ingressi discreti X0 e X1 si riferiscono rispettivamente ai contatori C64 e C65. C64 conta sia i fronti di salita che i fronti di discesa degli impulsi, C65 conta i fronti di salita degli impulsi.
9	Gli ingressi discreti X0 e X1 fanno riferimento ai contatori C64 e C65. C64 conta i fronti di salita degli impulsi, C65 conta i fronti di discesa degli impulsi.
10	Gli ingressi discreti X0 e X1 si riferiscono rispettivamente ai contatori C64 e C65. I contatori contano i fronti di discesa degli impulsi.
11	Gli ingressi discreti X0 e X1 si riferiscono rispettivamente ai contatori C64 e C65. C64 conta sia i fronti di salita che i fronti di discesa degli impulsi, C65 conta i fronti di discesa degli impulsi.
12	Gli ingressi discreti X0 e X1 fanno riferimento al contatore C64 e operano come un encoder. Un segnale in quadratura viene applicato agli ingressi. 

4.6. Indirizzamento e funzione dei registri [D], [A], [B]

Registri dati [D]

I registri rappresentano la memoria dati all'interno del controller. I registri possono memorizzare valori numerici e informazioni binarie uno dopo l'altro.

I dati vengono memorizzati in un registro a 16 bit (D0, ecc.), che può memorizzare un numero da -32768 a +32767. L'unione di due registri a 16 bit fornisce un "doppio registro" a 32 bit (D0, D1, ecc.), che può memorizzare un numero da -2147483648 a +2147483647.

L'indirizzamento dei registri dati è decimale. Per i doppi registri (32 bit), l'indirizzamento inizia con il registro inferiore a 16 bit.

D	Registri dati	Uso generale	D0…D319, D385…D391	Max. 384 punti

		Non volatile	D320…D351	
		Speciale	D352…D384	

Esistono i seguenti tipi di registri dati:

Registri dati per uso generale:

Questi registri vengono utilizzati durante l'esecuzione del programma utente, i dati non vengono salvati quando l'alimentazione viene spenta.

Registri dati non volatili:

I dati in questi registri vengono salvati nella memoria del controller quando l'alimentazione viene spenta. L'alimentazione della memoria è fornita da una fonte interna, CR2032.

Registri indice:

Questo registro viene utilizzato per memorizzare i risultati intermedi e per indicare gli operandi.

Registri speciali:

Questi registri vengono utilizzati per configurare il controller e per accedere ad alcune funzionalità speciali. I numeri dei registri speciali sono indicati nella tabella seguente:

Registro	Funzione	Valori
D352	Il registro contiene i dati sulla posizione del potenziometro “0” sul pannello frontale del controller.	0…4095
D353	Il registro contiene i dati sulla posizione del potenziometro “1” sul pannello frontale del controller.	0…4095
D354	Il registro contiene i dati sulla posizione del potenziometro “2”, “Speed”.	0…4095
D355	Il registro configura i tipi di ingresso IN0 e IN1, per maggiori dettagli, fare riferimento alla sezione 4.5	0…12

<table>
<tr>
<td>D356</td>
<td>Il registro configura i tipi di uscite OUT6 e OUT7 per l'istruzione PWM (vedere 7. Istruzioni applicative, istruzione PWM).
<table>
<tr><th rowspan="2">Valore</th><th rowspan="2">Tempo di discretizzazione, mks</th><th colspan="2">Funzione dell'uscita</th></tr>
<tr><th>OUT6</th><th>OUT7</th></tr>
<tr><td>0</td><td>–</td><td>uscita</td><td>uscita</td></tr>
<tr><td>1</td><td>100</td><td>Generatore PWM</td><td>uscita</td></tr>
<tr><td>2</td><td>10</td><td>Generatore PWM</td><td>uscita</td></tr>
<tr><td>3</td><td>100</td><td>uscita</td><td>Generatore PWM</td></tr>
<tr><td>4</td><td>10</td><td>uscita</td><td>Generatore PWM</td></tr>
<tr><td>5</td><td>100</td><td>Generatore PWM</td><td>Generatore PWM</td></tr>
<tr><td>6</td><td>10</td><td>Generatore PWM</td><td>Generatore PWM</td></tr>
</table>
Fare riferimento alla sezione 7. Istruzioni applicative(istruzioni PWM) per informazioni dettagliate sulla generazione del segnale PWM.</td>
<td>0…6</td>
</tr>
<tr>
<td>D357…D384</td>
<td>Registri di impostazione della modalità e di stato del driver del motore passo-passo. Vedere la sezione 8. «Istruzioni per il controllo del driver del motore passo-passo» per maggiori dettagli.</td>
<td>–</td>
</tr>
</table>

4.7. Registri indice [A], [B]

I registri indice vengono utilizzati per indicizzare gli indirizzi degli operandi e modificare i valori costanti.

I registri indice sono registri a 16 bit.

Nelle istruzioni a 32 bit, i registri indice A e B vengono utilizzati in combinazione. A contiene 16 bit di ordine inferiore e B contiene 16 bit di ordine superiore. Il registro indice A viene utilizzato come indirizzo di destinazione.

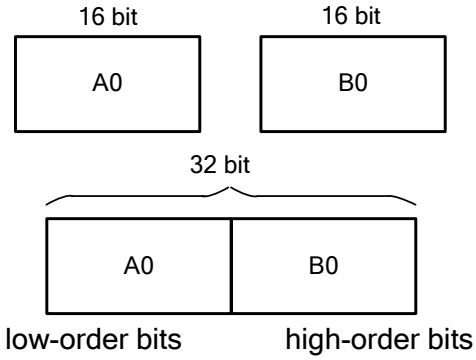
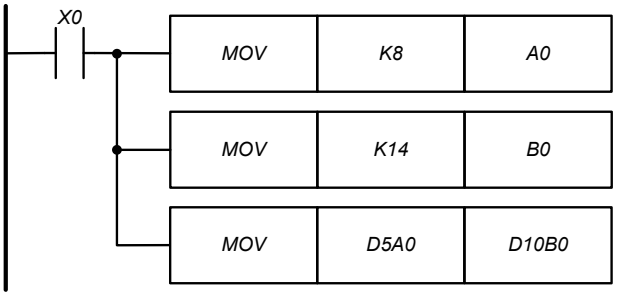


Fig. 27. Struttura del registro indice

Esempio di trasferimento dati dal registro dati D5A0 al registro dati D10B0:



Quando X0 = 1: A0 = 8, B = 14

- L'indirizzo di origine del trasferimento è D5A0 = 5 + 8 = D13
- L'indirizzo di destinazione è D10B0 = 10 + 14 = 24.
- In questo modo, i dati vengono trasferiti dal registro D13 al registro dati D24

Fig. 28. Trasferimento dati tramite registri indice

I registri indice possono essere utilizzati per operazioni di trasferimento dati e confronto in combinazione con operandi byte e operandi bit.

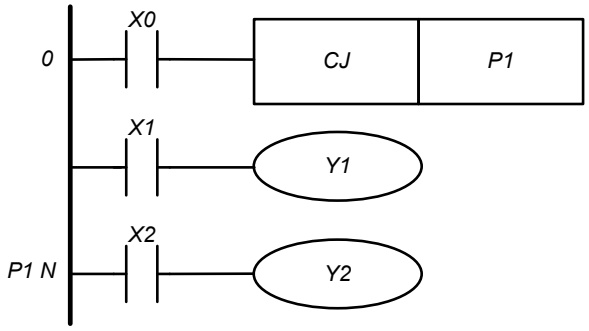
È anche possibile indicizzare le costanti allo stesso modo. Quando si indicizzano le costanti, è necessario utilizzare il simbolo "@". Ad esempio: MOV K10 @ A0 D0B0.

4.8. Puntatori [P], [I]

P	Puntatori di istruzione CALL, CJ		32 punti (P0…P31)		Etichette o contrassegni per comandi di transizione o chiamata di sottoprogrammi
I	Interruzioni	Comunicazione	I0	Max. 15 punti	Etichette per il sottoprogramma per la gestione delle interruzioni
		Temporizzato	I1…I100 (Max. 4 punti)		
		Esterno	I1000…I1007		
		Driver	I2000, I2001		

Puntatori (P) sono utilizzati in combinazione con le istruzioni CJ (transizioni) o CALL (sottoprogrammi). Questi puntatori sono indirizzi di posizioni di luoghi o sottoprogrammi, che sono stati contrassegnati.

Un esempio di esecuzione di un'istruzione di salto CJ:

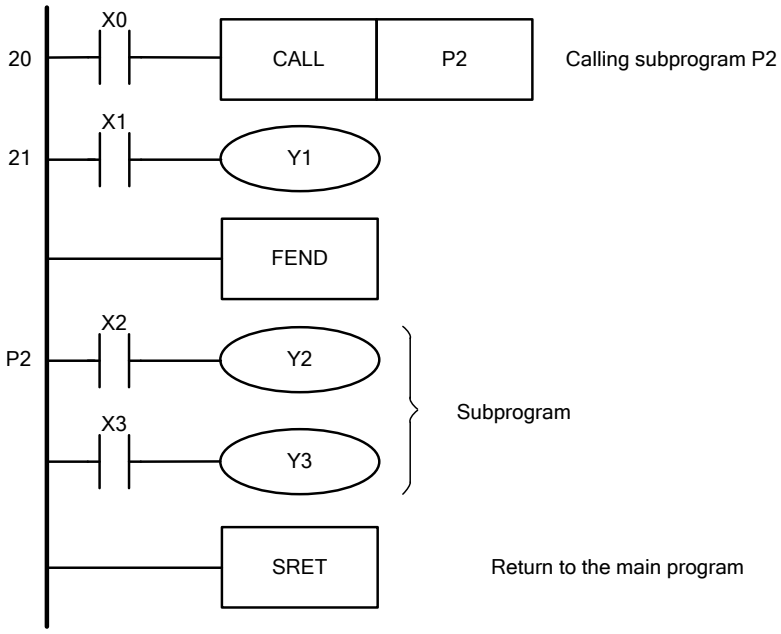


Quando X0 = 1, dopo l'esecuzione della riga 0, il programma va immediatamente alla riga con il puntatore P1, e le righe situate tra di loro non vengono eseguite.

Se X0 = 0, il programma viene eseguito normalmente passo dopo passo.

Fig. 29. Implementazione dell'istruzione CJ

Un esempio di utilizzo di sottoprogrammi:



Quando X0 = 1 alla riga 20, l'esecuzione del programma va direttamente alla riga contrassegnata con P2, il sottoprogramma viene eseguito e, dopo il comando SRET, l'esecuzione del programma ritorna alla riga 21.

Fig. 30. Implementazione dell'istruzione CALL

Puntatori di interruzione (I)sono utilizzati con le istruzioni EI, DI e IRET per interrompere l'esecuzione del programma principale. Esistono i seguenti tipi di interruzioni:

1. Interruzione di comunicazione: se il controller riceve un frame broadcast tramite protocollo Modbus, immediatamente (indipendentemente dal ciclo di scansione) passa al sottoprogramma di gestione delle interruzioni, contrassegnato dal puntatore I0. Ritorna al programma principale dopo l'esecuzione dell'istruzione IRET.
2. Interruzione temporizzata: il sottoprogramma di gestione delle interruzioni viene eseguito automaticamente a intervalli di tempo specificati da 10 a 1000 ms con incrementi di 10 ms. In totale, è possibile avere fino a 4 interruzioni temporizzate. Ad esempio, le interruzioni con puntatori I10, I50, I80 e I100 verranno eseguite rispettivamente una volta ogni 100 ms, 500 ms, 800 ms e 1 s. L'esecuzione ritorna al programma principale dopo l'istruzione IRET.
3. Interruzioni esterne: quando il segnale all'ingresso IN0 ...IN7 passa da 0 a 1 o da 1 a 0, il controller passa immediatamente all'esecuzione del sottoprogramma di gestione delle interruzioni con il puntatore I

corrispondente (IN0 → I1000, IN1 → I1001, ecc.). Il ritorno al programma principale avviene dopo l'esecuzione dell'istruzione IRET.

4. Interruzione driver: quando si verifica un errore durante la commutazione delle fasi del motore, rappresentato dallo speciale registro D381 (ERROR_CODE, maggiori dettagli nella sezione 8. “Istruzioni per il controllo del driver del motore passo-passo”), il controller passa all'esecuzione del sottoprogramma con il puntatore I2000. Quando lo stato del driver del motore passo-passo cambia, registro D371 (MOTOR_STATUS, per maggiori dettagli fare riferimento alla sezione 8. “Istruzioni per il controllo del driver del motore passo-passo”), il controller passa all'esecuzione del sottoprogramma con il puntatore I2001.Il controller ritorna al programma principale dopo l'esecuzione dell'istruzione IRET.

5. Codici di errore

Se il LED "ERR" è acceso dopo aver caricato ed eseguito il programma utente, significa che il programma utente contiene un errore: un errore grammaticale o un errore di operando errato. Ogni errore che si verifica nel controller viene registrato in un apposito registro (vengono registrati il numero di passo e il codice di errore). Queste informazioni possono essere lette utilizzando un PC o un PLC. La tabella seguente contiene un elenco di codici di errore e descrizioni.

Indirizzo	Tipo	Dimensioni	Descrizione
E004	Registri di Ingresso	16-bit	Codice di errore durante l'esecuzione di un programma utente.
E084	Registri di Ingresso	16-bit	Riga del programma utente in cui è stato rilevato l'errore.
E004	Bobine		Flag di errore durante l'esecuzione del programma utente.

Codice di errore	Descrizione
2012h	Comando sconosciuto
1007h	Errore interno, il tipo di collisione del segnale non è identificato.
1005h	Errore interno, il tipo di segnale in caso di collisione di livello non è identificato.
1006h	Errore interno, tipo di segnale non identificato in caso di collisione di livello invertito.
2002h	Istruzione LD, overflow dello stack.
2001h	Il tipo di segnale principale non è identificato.
2000h	Elaborazione dei comandi di tipo LD, il codice di istruzione è cambiato.
1001h	Errore interno, tipo sconosciuto di singola collisione.
1000h	Errore interno, singola collisione al valore corrente. Il tipo di segnale dell'operando è sconosciuto.
200Bh	Elaborazione del comando di tipo AND durante la rimozione del segnale dallo stack di output. Tipo di collisione sconosciuto.
1004h	Errore interno di collisione di gruppo. Tipo di collisione sconosciuto.
1002h	Errore interno di collisione di gruppo. Il tipo di segnale operando per la collisione di livello non è definito.
1003h	Errore interno di collisione di gruppo. Il tipo di segnale operando per la collisione a livello inverso non è definito.
200Ch	Elaborazione del comando di tipo AND, il codice di istruzione è cambiato.

Codice di errore	Descrizione
200Dh	Elaborazione del comando di tipo OR, il codice di istruzione è cambiato.
2010h	Non ci sono voci nello stack principale quando viene applicata l'istruzione ANB.
200Fh	Errore nell'applicazione dell'istruzione ANB. Non ci sono voci nello stack di output e c'è solo una voce nello stack principale.
200Eh	Segnale sconosciuto nello stack di output con il comando ANB.
2011h	L'assenza di almeno due elementi nello stack principale per l'applicazione dell'istruzione ORB.
2013h	Stack overflow di diramazione, istruzione MPS.
2016h	Lo stack di diramazione è vuoto, istruzioni MRD, MPP.
2015h	Overflow dello stack principale, istruzioni MRD, MPP.
2014h	Il tipo di segnale non viene riconosciuto durante l'assegnazione del selettore, delle istruzioni MRD e MPP.
2017h	Stack overflow, istruzione NEXT.
3012h	Pre-scan – indice P fuori intervallo.
3014h	Pre-scansione – l'indice I è fuori intervallo.
3013h	Prescan. Impossibile creare una nuova interruzione temporizzata, il limite di quantità è stato superato.
201Dh	Tipo di operando non corretto, istruzioni CJ/CJP.
201Ch	Operando fuori intervallo, istruzioni CJ/CJP.
2024h	Tipo di operando non corretto, istruzioni CALL/CALLP.
2023h	Operando fuori intervallo, istruzioni CALL/CALLP.
2025h	Non ci sono punti di ritorno nello stack, istruzione SRET.
2004h	Un comando END/FEND è stato ricevuto durante l'elaborazione di un'interruzione.
202Ah	Un comando IRET è stato ricevuto nel programma principale.
2056h	Istruzione END, lo stack principale non è vuoto.
2057h	Istruzione END, lo stack di diramazione non è vuoto.
2058h	Istruzione END, lo stack dei cicli non è vuoto.
2059h	Istruzione END, la pila dei sottoprogrammi non è vuota.
2026h	Istruzione IRET, lo stack principale non è vuoto.
2027h	Istruzione IRET, lo stack di diramazione non è vuoto.

Codice di errore	Descrizione
2028h	Istruzione IRET, lo stack dei cicli non è vuoto.
2029h	Istruzione IRET, lo stack del sottoprogramma non è vuoto.
2020h	Istruzione CALL/CALLP, operando indice sconosciuto.
201Eh	Istruzione CALL/CALLP, l'operando indice A è fuori intervallo.
201Fh	Istruzione CALL/CALLP, l'operando indice B è fuori intervallo.
201Ah	Istruzione CJ/CJP, operando indice sconosciuto.
2018h	Istruzione CJ/CJP, l'operando indice A è fuori intervallo.
2019h	Istruzione CJ/CJP, l'operando indice B è fuori intervallo.
201Bh	Istruzione CJ/CJP, il puntatore richiesto non esiste.
2022h	Istruzione CALL/CALLP, il riferimento richiesto non esiste.
2021h	Istruzione CALL/CALLP, stack overflow.
2003h	Operando errato, istruzione OUT.
200Ah	Operando errato, istruzione SET/RST.
2005h	Il SET di istruzioni non può essere applicato all'operando C.
2006h	L'insieme di istruzioni SET non può essere applicato all'operando T.
2007h	Il SET di istruzioni non può essere applicato all'operando D.
2008h	Il SET di istruzioni non può essere applicato all'operando A.
2009h	Il SET di istruzioni non può essere applicato all'operando B.
202Dh	Istruzione INV, tipo di segnale sconosciuto.
202Bh	Nell'istruzione TMR, il primo argomento non è tipico.
202Ch	Istruzione CNT, il primo argomento non è tipico.
202Eh	Istruzione INC/DEC, operando non corretto.
2037h	Istruzione ADD/SUB/MUL/DIV/WAND/WOR/WXOR, il tipo del terzo operando non è corretto.
2038h	Istruzione NEG/ABS, tipo di operando non corretto.
2030h	Istruzione CMP, il tipo del terzo operando non è corretto.
2031h	Istruzione ZCP, il tipo di operando 3d non è corretto.
202Fh	Istruzione MOV/BMOV/FMOV, tipo di operando di destinazione non corretto.
2039h	Istruzione XCH, il tipo di dato del 1ºoperando non è corretto.

Codice di errore	Descrizione
203Ah	Istruzione XCH, il tipo di dato del secondo operando non è corretto. .
203Bh	Istruzione ROR/ROL, il tipo di dati del 1° operando non è corretto.
2033h	Istruzione ZRST, gli operandi non sono dello stesso tipo.
2032h	Istruzione ZRST, il tipo di operando non è corretto.
2036h	Istruzione DIV, divisione per zero di un intero.
2046h	Istruzione DECO, il tipo del secondo operando non è corretto.
2047h	Istruzione ENCO, il tipo del secondo operando non è corretto.
2048h	Istruzione SUM, il tipo del secondo operando non è corretto.
2049h	Istruzione BON, il tipo del secondo operando non è corretto.
204Bh	Istruzione SQR, il tipo di operando 2d non è corretto.
204Ah	Istruzione SQR, valore negativo.
204Ch	Istruzione POW, il tipo del terzo operando non è corretto.
203Ch	Istruzione FLT, il tipo del secondo operando non è corretto.
203Dh	Istruzione INT, il tipo di operando 2d non è corretto.
203Eh	Istruzione PWM, il terzo operando non è applicabile per l'uscita del segnale PWM.
203Fh	Istruzione PWM, il tipo del terzo operando non è corretto.
2041h	Istruzione DECMP, il tipo del primo operando non è corretto.
2040h	Istruzione DECMP, il tipo del secondo operando non è corretto.
2042h	Istruzione DECMP, il tipo del terzo operando non è corretto.
2045h	Istruzione DEZCP, il tipo di terzo operando non è corretto.
2044h	Istruzione DEZCP, il tipo del primo operando non è corretto.
2043h	Istruzione DEZCP, il tipo di operando 2d non è corretto.
2050h	Istruzione DEADD/DESUB/DEMUL/DEDIV/DEPOW, il tipo del terzo operando non è corretto.
204Fh	Istruzione DEADD/DESUB/DEMUL/DEDIV/DEPOW, il tipo del primo operando non è corretto.
204Eh	Istruzione DEADD/DESUB/DEMUL/DEDIV/DEPOW, il tipo del secondo operando non è corretto.
204Dh	Istruzione DEDIV, divisione per zero.

Codice di errore	Descrizione
3015h	Errore di prescansione, comando sconosciuto rilevato.
2053h	Istruzione DESQR, tipo del 1° operando non corretto.
2052h	Istruzione DESQR, il tipo di operando 2d non è corretto.
2051h	Valore negativo dell'istruzione DESQR.
2035h	Istruzione LD#, stack overflow.
2034h	Istruzione LD#, tipo di segnale principale non riconosciuto.
4000h	Overflow della coda delle interruzioni degli switch.
4001h	Superamento della coda delle interruzioni temporizzate TIM0.
4002h	Superamento della coda delle interruzioni temporizzate TIM1.
4003h	Superamento della coda delle interruzioni temporizzate TIM2.
4004h	Superamento della coda delle interruzioni temporizzate TIM3.
4005h	Overflow della coda degli interrupt esterni IN0.
4006h	Overflow della coda degli interrupt esterni IN1.
4007h	Overflow della coda degli interrupt esterni IN2.
4008h	Overflow coda di interruzioni esterne IN3.
4009h	Overflow coda di interruzioni esterne IN4.
400Ah	Overflow coda di interruzioni esterne IN5.
400Bh	Overflow coda di interruzioni esterne IN6.
400Ch	Overflow coda di interruzioni esterne IN7.
400Dh	Overflow della coda delle interruzioni del driver.
400Eh	Overflow della coda di interruzione per cambio di stato del motore.
2054h	Istruzione TRD, tipo di operando non corretto.
2055h	Istruzione TWR, tipo di operando non corretto.
3000h	Stack vuoti, nessun valore di segnale.
3001h	Lo stack principale è vuoto, nessun valore di segnale.
3003h	Il valore del registro indice è fuori intervallo.
3004h	Indice dell'operando X fuori intervallo.
3005h	Indice dell'operando Y fuori intervallo.

Codice di errore	Descrizione
3006h	Indice dell'operando M fuori intervallo.
3007h	Indice dell'operando C fuori intervallo.
3008h	Indice dell'operando T fuori intervallo.
3009h	Indice dell'operando A/B fuori intervallo.
300Ah	Indice dell'operando D fuori intervallo.
300Bh	Indice dell'operando P fuori intervallo.
300Ch	Indice dell'operando I fuori intervallo.
300Dh	Tipo di operando sconosciuto
300Fh	Impossibile ottenere il valore dell'operando.
300Eh	Il numero FLOAT viene utilizzato con un'istruzione a 16 bit.
3010h	Ottenimento del valore dell'operando - tipo di operando errato.
3011h	Ottenimento del token dell'operando - tipo di operando non corretto.
5000h	Alimentazione + 5V - corto circuito
205Ah	Istruzione TWR, formato orario errato.
205Bh	Istruzione MOD, divisione per zero.
205Ch	Comando DWR, formato data non valido.
205Dh	Stack overflow delle istruzioni di tipo contatto (LD#*)
205Eh	Stack overflow delle istruzioni di tipo contatto (AND#*)
205Fh	Stack overflow delle istruzioni di tipo contatto (OR#*)

6. Istruzioni di base

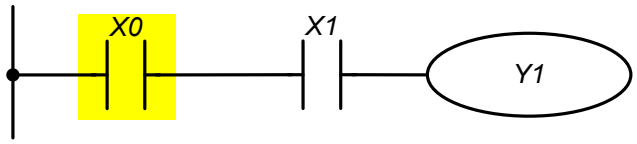
Istruzioni	Funzione
LD	Contatto normalmente aperto

Operando	X	Y	M	T	C	A	B	D
	•	•	•	•	•			

Descrizione:

L'istruzione LD viene utilizzata come contatto normalmente aperto per la programmazione dell'inizio delle catene logiche. Si trova a sinistra nello schema dei contatti ed è collegata direttamente alla linea del bus di alimentazione.

Utilizzo:



LD	X0
AND	X1
OUT	Y1

L'istruzione LD X0 "contatto normalmente aperto X0" avvia la connessione logica sequenziale. Se agli ingressi X0 e X1 è presente contemporaneamente un segnale "1", allora l'uscita Y1 verrà impostata sullo stato "1".

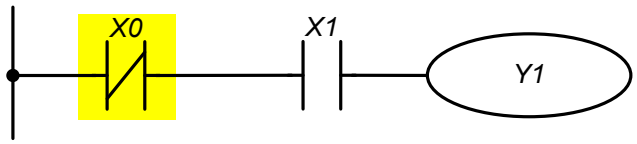
Istruzioni	Funzione
LDI	Contatto normalmente chiuso

Operando	X	Y	M	T	C	A	B	D
	•	•	•	•	•			

Descrizione:

L'istruzione LDI viene utilizzata come contatto normalmente chiuso per la programmazione dell'inizio delle catene logiche. Si trova a sinistra nello schema dei contatti ed è collegata direttamente alla linea del bus di alimentazione.

Utilizzo:



LDI	X0
AND	X1
OUT	Y1

L'istruzione LDI X0 "contatto normalmente chiuso X0" avvia la connessione logica sequenziale. Se agli ingressi X0 e X1 è presente contemporaneamente un segnale "1", allora l'uscita Y1 verrà impostata sullo stato "1".

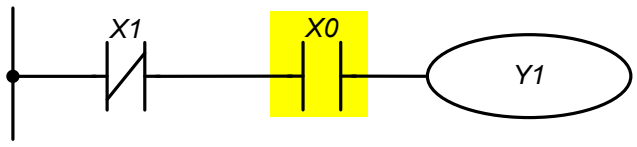
Istruzioni	Funzione
AND	Collegamento in serie - contatto normalmente aperto (AND logico)

Operando	X	Y	M	T	C	A	B	D
	•	•	•	•	•			

Descrizione:

L'istruzione AND viene utilizzata come un contatto normalmente aperto collegato in serie per la programmazione dell'operazione di moltiplicazione logica (AND). L'istruzione rappresenta un'operazione logica e, pertanto, non può essere programmata all'inizio della sequenza. Per le istruzioni di inizio sequenza, si deve usare LD o LDI.

Utilizzo:




```
LDI    X1
AND    X0
OU     Y1
```

L'istruzione AND X0 "Connessione in serie - contatto normalmente aperto X0" crea una connessione logica in serie con il contatto X1 e viene utilizzata per eseguire l'operazione di moltiplicazione logica. Se c'è “0” all'ingresso X1 e “1” a X0, l'uscita Y1 passa allo stato “1”.

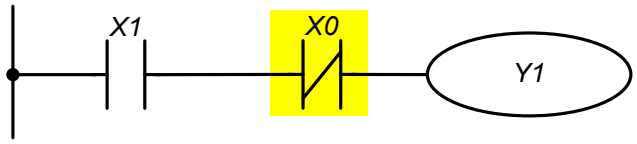
Istruzioni	Funzione
ANI	Collegamento in serie - contatto normalmente chiuso (NAND logico)

Operando	X	Y	M	T	C	A	B	D
	•	•	•	•	•			

Descrizione:

L'istruzione ANI viene utilizzata come contatto normalmente chiuso collegato in serie per la programmazione dell'operazione logica NAND (AND NOT). L'istruzione rappresenta un'operazione logica e, pertanto, non può essere programmata all'inizio della sequenza. Per le istruzioni di inizio sequenza, è necessario utilizzare LD o LDI.

Utilizzo:




```
LD     X1
ANI    X0
OUT    Y1
```

L'istruzione "Connessione in serie - contatto normalmente chiuso X0" crea una connessione logica in serie con il contatto X1 e viene utilizzata per eseguire l'operazione logica NAND. Se c'è “1” all'ingresso X1 e “0” a X0, l'uscita Y1 passa allo stato “1”.

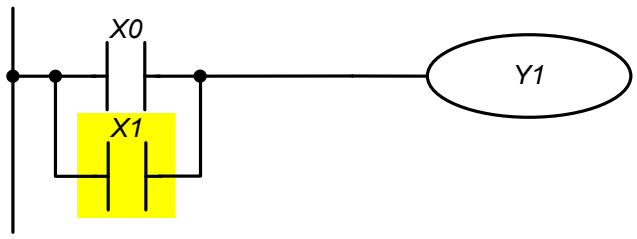
Istruzioni	Funzione
OR	Collegamento in parallelo – contatto normalmente aperto (OR logico)

Operando	X	Y	M	T	C	A	B	D
	•	•	•	•	•			

Descrizione:

L'istruzione OR viene utilizzata come un contatto normalmente aperto collegato in parallelo per la programmazione dell'addizione logica (OR). L'istruzione rappresenta un'operazione logica e, pertanto, non può essere programmata all'inizio della sequenza. Per le istruzioni di inizio sequenza, è necessario utilizzare LD o LDI.

Utilizzo:




```
LD    X0
OR    X1
OUT   Y1
```

L'istruzione "Connessione in parallelo – contatto normalmente aperto X1" crea una connessione logica in parallelo con il contatto X0 e viene utilizzata per eseguire l'operazione di addizione logica. Se almeno uno degli ingressi X0 o X1 è "1", l'uscita Y1 passa allo stato "1".

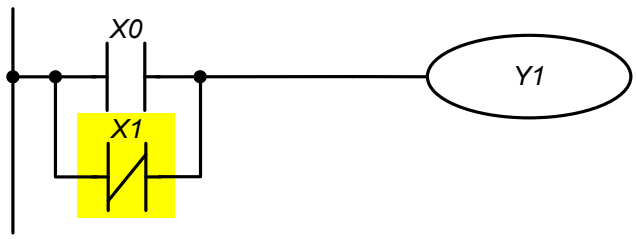
Istruzioni	Funzione
ORI	Collegamento in parallelo – contatto normalmente chiuso (NOR logico)

Operando	X	Y	M	T	C	A	B	D
	•	•	•	•	•			

Descrizione:

L'istruzione ORI viene utilizzata come contatto normalmente chiuso collegato in parallelo per la programmazione dell'operazione logica NOR (OR NOT). L'istruzione rappresenta un'operazione logica e, pertanto, non può essere programmata all'inizio della sequenza. Per le istruzioni di inizio sequenza, è necessario utilizzare LD o LDI.

Utilizzo:




```
LD    X0
ORI   X1
OUT   Y1
```

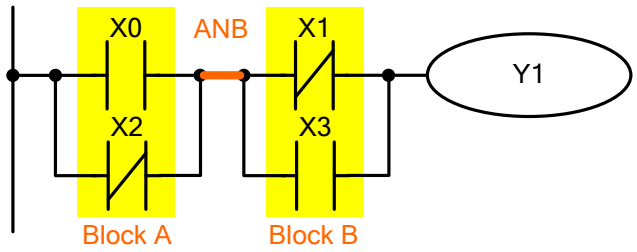
L'istruzione "Connessione in parallelo – contatto normalmente chiuso X1" crea una connessione logica in parallelo con il contatto X0 e viene utilizzata per eseguire l'operazione di istruzione logica NOR (OR NOT). Se l'ingresso X0 è "1" o l'ingresso X1 è "0" (una o entrambe le condizioni contemporaneamente), l'uscita Y1 passa allo stato "1".

Istruzioni	Funzione
ANB	Blocco «AND»: collegamento in serie dei blocchi

Descrizione:

- L'istruzione ANB viene utilizzata per la connessione in serie di due catene (blocchi) logiche.Blocchi separati di elementi collegati in parallelo vengono inseriti nel programma separatamente.Per collegare questi blocchi in serie, viene programmata un'istruzione ANB dopo ogni blocco.
- Inizio della diramazione programmata utilizzando le istruzioni LD o LDI.
- L'istruzione ANB è indipendente e non richiede operandi.
- L'istruzione ANB all'interno dell'intero programma utente può essere utilizzata un numero illimitato di volte.
- L'istruzione ANB è mostrata come una connessione in serie in un diagramma a contatti.L'istruzione ANB in un elenco di istruzioni in linguaggio IL può essere mostrata in un circuito a contatti come un ponticello.
- Se è necessario collegare alcuni blocchi separati uno dopo l'altro, il numero di istruzioni LD/LDI e anche il numero di istruzioni ANB deve essere limitato a 8.

Utilizzo:




```
LD    X0
ORI   X2
LDI   X1
OR    X3
ANB
OUT   Y1
```

L'istruzione ANB crea una serie di connessioni logiche tra due blocchi logici (Blocco A e Blocco B).

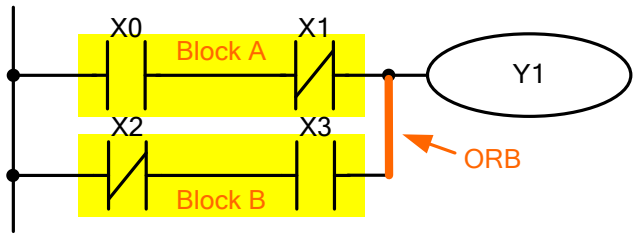
Istruzioni	Funzione
ORB	Blocco «OR»: collegamento in parallelo dei blocchi

Descrizione:

- L'istruzione ORB viene utilizzata per il collegamento in parallelo di due o più contatti o blocchi collegati in serie.Se più blocchi collegati in serie sono collegati in parallelo, è necessario aggiungere un'istruzione ORB dopo ogni blocco.
- L'inizio della diramazione viene programmato utilizzando le istruzioni LD o LDI.
- L'istruzione ORB è indipendente e non richiede operandi.
- L'istruzione ORB all'interno del programma utente può essere utilizzata un numero illimitato di volte.
- Se più blocchi separati vengono programmati direttamente uno dopo l'altro, è necessario limitare il numero di istruzioni LD e LDI e anche il numero di istruzioni ORB a 8.

— L'istruzione ORB viene visualizzata come una connessione parallela in un diagramma a contatti.L'istruzione ORB in un elenco di istruzioni in linguaggio IL può essere visualizzata in un circuito a contatti come un ponticello.

Utilizzo:




```
LD    X0
ANI   X1
LDI   X2
AND   X3
OR
OUT   Y1
```

L'istruzione ORB crea una serie di connessioni logiche tra due blocchi logici (Blocco A e Blocco B).

Istruzioni	Funzione
MPS	Offset verso il basso nello stack

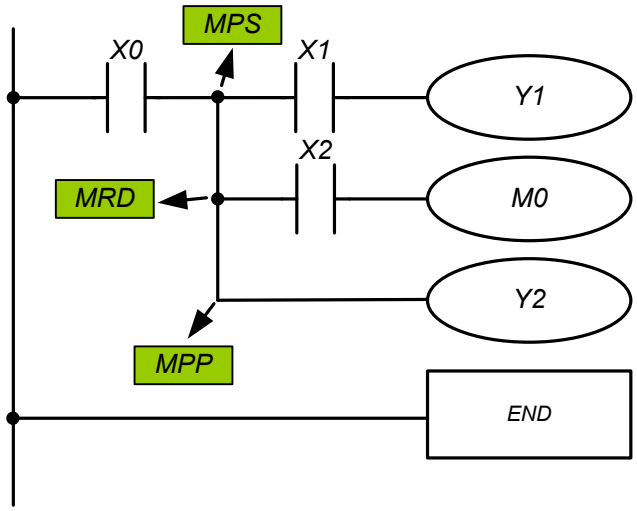
Istruzioni	Funzione
MRD	Leggere il valore dallo stack

Istruzioni	Funzione
MPP	Uscita dallo stack

Descrizione:

— Le istruzioni MPS, MRD e MPP vengono utilizzate per creare livelli di connessioni logiche (ad esempio, dopo un'espressione logica iniziale, creare diverse espressioni logiche in uscita, ovvero attivare diverse bobine di uscita).
— Utilizzando l'istruzione MPS, il risultato precedente delle connessioni logiche (elaborazione di un'espressione logica) viene memorizzato.
— Utilizzando l'istruzione MRD, è possibile creare diverse diramazioni indipendenti tra l'inizio (MPS) e la fine (MPP) della diramazione.Il risultato dell'elaborazione di un'espressione logica nel punto MPS viene preso in considerazione in ogni diramazione.
— L'ultima diramazione viene creata dall'istruzione MPP.
— La diramazione aperta con l'istruzione MPS deve essere sempre chiusa dall'istruzione MPP.
— Le istruzioni MPS, MRD e MPP non necessitano di operandi.
— Queste istruzioni non vengono visualizzate nel diagramma a contatti.Se la programmazione viene eseguita in un circuito a contatti, le diramazioni vengono utilizzate come di consueto. Quando si converte un programma utente da un diagramma ladder (LD) a un elenco di istruzioni (IL), le istruzioni MPS, MRD e MPP devono essere aggiunte a IL.

Utilizzo:




```
LD      X0
MPS
AND     X1
OUT     Y1
MPD
AND     X2
OUT     M0
MPP
OUT     Y2
END
```

MPS

Un risultato intermedio (valore X0) al 1° livello di connessioni logiche è elencato al 1° posto nella memoria stack delle connessioni intermedie. Viene eseguita la moltiplicazione logica di X1 con X0 e viene impostato l'output Y1.

MRD

Prima di eseguire l'istruzione successiva, viene letto un risultato intermedio nella prima posizione della memoria delle connessioni logiche. Viene eseguita la moltiplicazione logica di X2 con X0 e viene impostato l'output di M0.

MPP

Prima di eseguire l'istruzione successiva, viene letto un risultato intermedio nella prima posizione della memoria delle connessioni logiche. L'uscita Y2 è impostata. L'operazione al 1° livello di risultati intermedi è completata e la memoria delle connessioni logiche viene cancellata.

Istruzioni	Funzione
OUT	Bobina di uscita

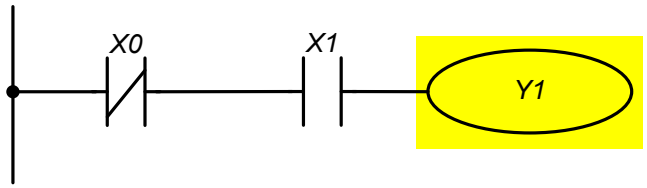
Operando	X	Y	M	T	C	A	B	D
		•	•					

Descrizione:

- L'istruzione OUT viene utilizzata per impostare una bobina di uscita in base al risultato delle connessioni logiche (il risultato dell'elaborazione dell'espressione logica da parte del controller).
- Utilizzando l'istruzione OUT, è possibile terminare la programmazione di una riga (espressione logica).
- È anche possibile programmare diverse istruzioni OUT come risultato dell'elaborazione di un'espressione logica.

- Il risultato delle connessioni logiche rappresentato dall'istruzione OUT può essere applicato nei passaggi successivi del programma come stato del segnale di ingresso, ovvero può essere letto più volte in più espressioni logiche.
- Il risultato delle connessioni logiche rappresentato dall'istruzione OUT è attivo (on) finché le condizioni per la sua attivazione sono valide.
- Quando si programma la doppia registrazione delle stesse uscite (i loro indirizzi), possono sorgere problemi durante l'esecuzione del programma.Evitare la doppia registrazione delle uscite, poiché ciò può causare interferenze durante l'esecuzione del programma.

Utilizzo:




```
LDI   X0
AND   X1
OUT   Y1
```

Se X0 = 0 e X1 = 1, l'istruzione OUT Y1 imposta lo stato dell'uscita Y1 = "1".

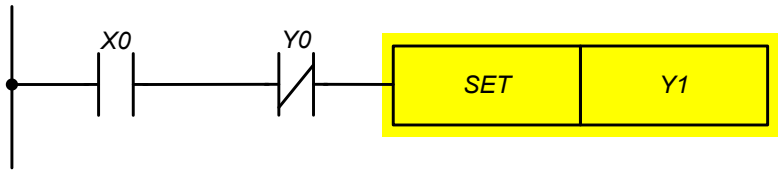
Istruzioni	Funzione
SET	Attivazione dell'uscita latchata

Operando	X	Y	M	T	C	A	B	D
		•	•					

Descrizione:

- Lo stato dell'operando può essere impostato direttamente dall'istruzione SET.
- Gli operandi Y e M possono essere attivati dall'istruzione SET.
- Non appena viene stabilita la condizione di ingresso per l'istruzione SET (segnale "1"), il corrispondente operando si attiva.
- Se le condizioni di ingresso per l'istruzione SET non sono più soddisfatte, il corrispondente operando rimane attivo.

Utilizzo:




```
LD    X0
ANI   Y0
SET   Y1
```

L'uscita Y1 si attiva quando le condizioni di ingresso (X0, Y0) sono soddisfatte. Dopodiché, l'uscita Y1 non dipende più dalle condizioni di ingresso. L'unico modo per disattivare l'uscita Y1 è utilizzare l'istruzione RST o disattivare l'alimentazione del controller.

Istruzioni	Funzione
RST	Ripristino dello stato dell'operando

Operando	X	Y	M	T	C	A	B	D
		●	●	●	●	●	●	●

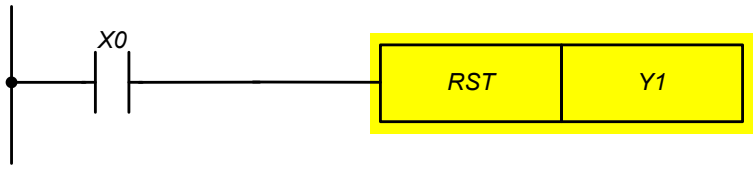
Descrizione:

Lo stato di un operando può essere ripristinato direttamente.

L'istruzione RST disattiva i corrispondenti operandi. Significa:

- Uscite Y, contatti M sono disattivati (stato del segnale "0").
- I valori attuali di timer e contatori, i valori dei registri D, A e B vengono azzerati a "0".
- Non appena si verifica la condizione di ingresso per l'istruzione RST (segnale "1"), l'operando corrispondente si disattiva.
- Se le condizioni di ingresso per l'istruzione RST non sono più soddisfatte, l'operando corrispondente rimane disattivato.

Utilizzo:




```
LD    X0
RST   Y1
```

L'uscita Y1 si disattiva quando la condizione X1 è soddisfatta e rimane disattivata anche quando la condizione X0 non è soddisfatta.

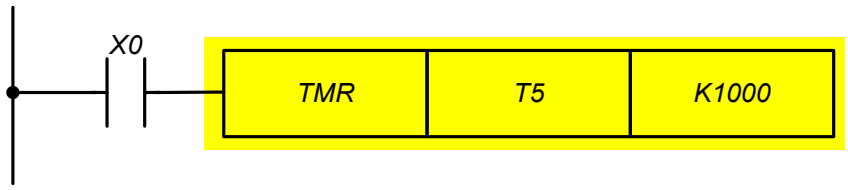
Istruzioni	Funzione
TMR	Timer (16-bit)

Operandi	K	H	F	X	Y	M	T	C	A	B	D
S1							●				
S2	●	●					●	●	●	●	●

Descrizione:

- L'istruzione TMR viene utilizzata per impostare lo stato di un segnale (attivazione/disattivazione) a seconda del risultato delle connessioni logiche dopo un periodo di tempo specificato nell'istruzione.
- Utilizzando l'istruzione TMR, è possibile terminare la programmazione di una riga (espressione logica).
- Il risultato delle connessioni logiche rappresentate dall'istruzione TMR può essere utilizzato nei successivi passaggi del programma come stato del segnale di ingresso, ovvero può essere letto più volte in più espressioni logiche.
- Il risultato delle connessioni logiche rappresentate dall'istruzione TMR è attivo (acceso) finché le condizioni di ingresso sono valide.

Utilizzo:




```
LD    X0
TMR   T5   K1000
```

Alla condizione X0 = 1, l'istruzione TMR T5 conta fino a quando il valore nel registro T5 raggiunge il valore di K1000 (100 sec). Se X0 = 0, l'esecuzione dell'istruzione TMR si interrompe e T5 si azzera a "0".

Istruzioni		Funzione
CNT	S1 S2	Contatore (16-bit)
DCNT		Contatore (32-bit)

Operandi	K	H	F	X	Y	M	T	C	A	B	D
S1								•			
S2	•	•					•	•	•	•	•

Informazioni

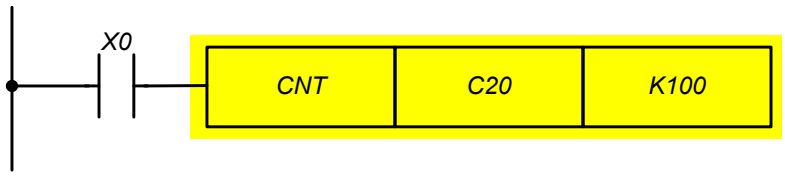
D Di solito, per utilizzare istruzioni a 32 bit, viene aggiunto il prefisso "D" al nome dell'istruzione. D – esiste solo una versione a 32 bit dell'istruzione.

P Per le istruzioni impulsive con una "durata" di una singola scansione, viene aggiunto il suffisso "**P**". Il concetto di*singola scansione*deve essere attribuito all'operando utilizzato. Ad esempio, l'operando **M0** alla riga 7 del programma principale è stato impostato da "0" a "1". Ora, per tutte le istruzioni successive, prima di FEND o END, l'operando **M0** ha una componente impulsiva sul fronte di salita (il risultato per LDP M0 sarà “1”), così come per le istruzioni che partono dalla riga 0ªalla 6ªdurante la scansione successiva l'operando **M0** avrà una componente impulsiva. Quando si passa alla riga 7 (o inferiore se viene utilizzato un comando CJ), **M0** avrà un livello di segnale alto senza componenti impulsive. Quindi, è stato eseguito un ciclo lungo il corpo del programma - una scansione, spostata sulla riga di modifica dell'operando. In caso di interruzioni che si verificano prima di raggiungere la riga 7, l'operando **M0** mantiene la componente impulsiva fino al ritorno al programma principale. Se l'operando **M0** è stato modificato in un'interruzione o in un sottoprogramma, allora il punto in cui l'operando viene modificato è considerato la riga dalla quale è stata eseguita la transizione al sottoprogramma o la riga del programma principale, prima dell'elaborazione della quale è stato chiamato il gestore di interrupt.

Descrizione:

- L'istruzione CNT viene utilizzata per riepilogare il numero di chiusure del contatto di ingresso e assegnare lo stato del segnale (attivare l'uscita) quando il valore corrente del contatore raggiunge il valore impostato.
- Utilizzando l'istruzione CNT, è possibile terminare la programmazione di una riga (espressione logica).
- - Il risultato delle connessioni logiche rappresentate dall'istruzione CNT può essere applicato nei successivi passaggi del programma come stato del segnale di ingresso, ovvero può essere letto più volte in più espressioni logiche.
- Per azzerare il valore corrente di un contatore, utilizzare l'istruzione RST.
- **Attenzione:**i contatori hardware contano al di sopra della soglia e funzionano indipendentemente dalla presenza di un segnale di ingresso

Utilizzo:




```
LD      X0
CNT     C20     K100
```

Quando X0 passa da “0” a “1”, il valore del registro C20 aumenta di 1. Si ripete fino a quando il valore del registro C20 raggiunge K100 (100 impulsi). Dopo di che, il conteggio si interrompe e il contatto C20 si attiva. Per azzerare il valore del registro C20, utilizzare l'istruzione RST C20.

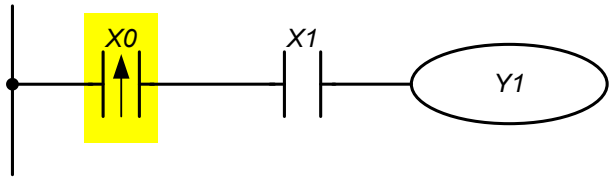
Istruzioni	Funzione
LDP	Inizio dell'espressione logica con un fronte di salita (impulso)

Operando	X	Y	M	T	C	A	B	D
	•	•	•	•	•			

Descrizione:

- L'istruzione LDP viene utilizzata per programmare l'avvio a impulsi di una connessione logica.
- L'istruzione LDP deve essere programmata all'inizio del circuito.
- L'istruzione LDP viene utilizzata anche in combinazione con le istruzioni ANB e ORB per avviare la ramificazione.
- L'istruzione LDP dopo un fronte di salita viene memorizzata per la durata del ciclo di programma (scansione).

Utilizzo:




```
LDP     X0
AND     X1
OUT     Y1
```

L'istruzione "LDP X0" avvia la connessione logica in serie. Se l'ingresso X0 passa da "0" a "1" (e X1 = 1), l'uscita Y1 mantiene lo stato "1" durante una scansione.

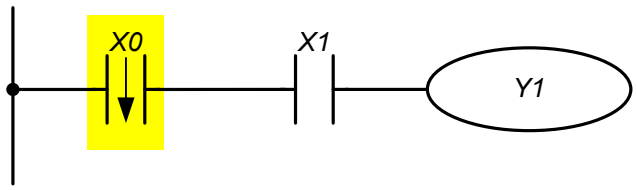
Istruzioni	Funzione
LDF	Inizio di un'espressione logica con un fronte di discesa (impulso)

Operando	X	Y	M	T	C	A	B	D
	•	•	•	•	•			

Descrizione:

- L'istruzione LDF viene utilizzata per programmare l'avvio a impulsi di una connessione logica.
- L'istruzione LDF deve essere programmata all'inizio del circuito.
- L'istruzione LDF viene utilizzata anche in combinazione con le istruzioni ANB e ORB per avviare la ramificazione.
- L'istruzione LDF dopo un fronte di discesa viene memorizzata per la durata del ciclo di programma (scansione).

Utilizzo:




```
LDF   X0   3
AND   X1
OUT   Y1
```

L'istruzione "LDF X0 avvia la connessione logica in serie. Se l'ingresso X0 passa da "1" a "0" (e X1 = 1), l'uscita Y1 mantiene lo stato "1" durante una scansione.

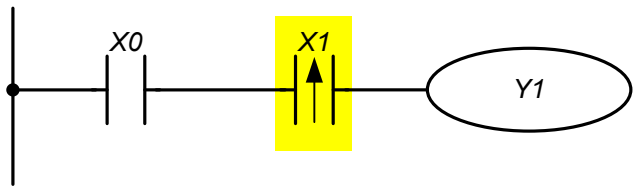
Istruzioni	Funzione
ANDP	«AND» con polling (impulso) sul fronte di salita

Operando	X	Y	M	T	C	A	B	D
	•	•	•	•	•			

Descrizione:

- L'istruzione ANDP viene utilizzata per programmare un contatto a impulsi collegato in serie con polling del fronte di salita (impulso).

Utilizzo:




```
LD     X0
ANDP   X1
OUT    Y1
```

L'istruzione "ANDP X1" crea una connessione logica in serie. Se l'ingresso X1 passa da "0" a "1" (e X0 = 1), l'uscita Y1 mantiene lo stato "1" durante una scansione.

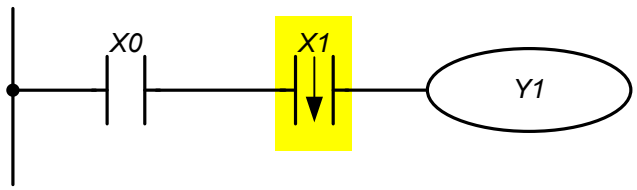
Istruzioni	Funzione
ANDF	«AND» con polling sul fronte di discesa (impulso)

Operando	X	Y	M	T	C	A	B	D
	•	•	•	•	•			

Descrizione:

— L'istruzione ANDF viene utilizzata per programmare un contatto a impulsi collegato in serie con polling del fronte di discesa (impulso).

Utilizzo:




```
LD    X0
ANDF  X1
OUT   Y1
```

L'istruzione "ANDF X1" crea una connessione logica in serie. Se l'ingresso X1 passa da "1" a "0" (e X0 = 1), l'uscita Y1 mantiene lo stato "1" durante una scansione.

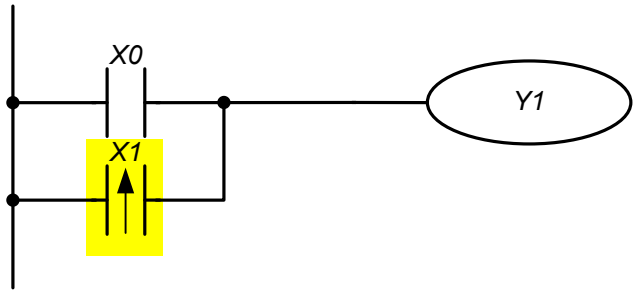
Istruzioni	Funzione
ORP	«OR» con polling del fronte di salita (impulso)

Operando	X	Y	M	T	C	A	B	D
	•	•	•	•	•			

Descrizione:

— L'istruzione ORP viene utilizzata per la programmazione di un contatto a impulsi collegato in parallelo con polling del fronte di salita (impulso).

Utilizzo:




```
LD    X0
ORP   X1
OUT   Y1
```

L'istruzione "ORP X1" crea una connessione logica parallela. L'uscita Y1 manterrà lo stato "1" durante una scansione se l'ingresso X1 cambia da "0" a "1" o X0 = 1.

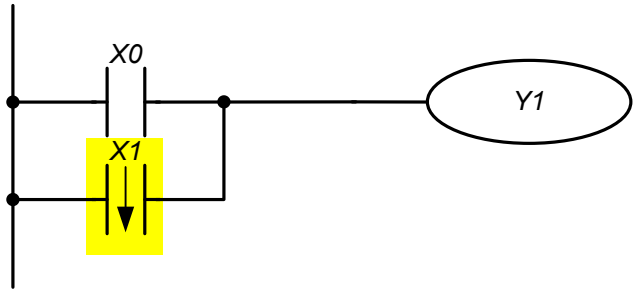
Istruzioni	Funzione
ORF	«OR» con polling sul fronte di discesa (impulso)

Operando	X	Y	M	T	C	A	B	D
	•	•	•	•	•			

Descrizione:

— L'istruzione ORF viene utilizzata per programmare un contatto a impulsi collegato in parallelo con polling del fronte di discesa (impulso).

Utilizzo:




```
LD    X0
ORF   X1
OUT   Y1
```

L'istruzione "ORF X1" crea una connessione logica parallela. L'uscita Y1 manterrà lo stato "1" durante una scansione se l'ingresso X1 cambia da "1" a "0" o X0 = 1.

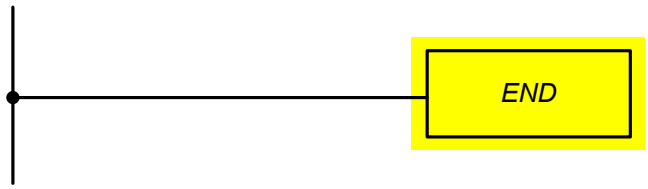
Istruzioni	Funzione
END	Fine programma

Descrizione:

Fine di un programma utente e passaggio all'inizio del programma (passo 0).

— Ogni programma del controller deve terminare con un'istruzione END.
— Se viene programmata un'istruzione END, a questo punto l'elaborazione del programma termina.Le aree successive del programma non vengono più prese in considerazione.Dopo l'elaborazione dell'istruzione END, le uscite vengono impostate e il programma si avvia (passo 0).

Utilizzo:




```
END
```

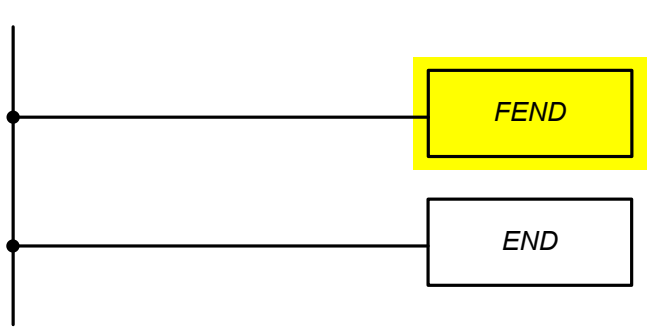
Istruzioni	Funzione
FEND	Fine del programma principale

Descrizione:

Fine del programma utente principale e passaggio all'inizio del programma (passo 0). Le principali differenze rispetto all'istruzione END sono le seguenti:

- L'elaborazione non termina con il comando FEND.L'istruzione FEND separa il programma principale dai sottoprogrammi e dai gestori di interruzione, che si trovano nell'area tra le istruzioni FEND ed END e sono inquadrati da P e SRET, I e IRET.
- Se i sottoprogrammi e le interruzioni non vengono utilizzati in un programma utente, l'istruzione FEND non è necessaria.
- L'istruzione FEND può essere utilizzata una sola volta.

Utilizzo:




```
FEND
END
```

Istruzioni	Funzione
NOP	Riga vuota nel programma

Descrizione:

Una riga vuota senza funzioni logiche può essere successivamente utilizzata per qualsiasi istruzione, ad esempio durante l'assemblaggio di un programma o per il debug.

- Dopo aver assemblato correttamente un programma, le istruzioni NOP devono essere eliminate, altrimenti estendono inutilmente il tempo del ciclo del programma.
- Il numero di istruzioni NOP in un programma non è limitato.

Utilizzo:

```
LD      X0
NOP
OUT     Y0
```

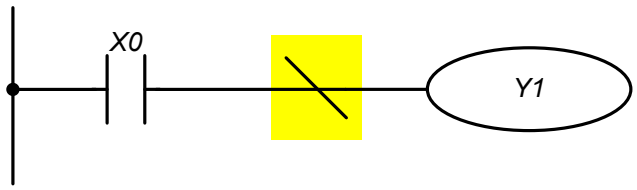
Le istruzioni NOP non vengono visualizzate nei diagrammi a contatti.

Istruzioni	Funzione
INV	**Inversione - sostituzione del risultato delle connessioni logiche con l'opposto**

Descrizione:

- L'istruzione INV inverte lo stato del segnale risultante delle istruzioni precedenti.
- Il risultato delle connessioni logiche “1” prima dell'istruzione INV diventa “0” dopo di essa.
- Il risultato delle connessioni logiche “0” prima dell'istruzione INV diventa “1” dopo di essa.
- L'istruzione INV può essere applicata come istruzione AND o ANI.
- L'istruzione INV può essere utilizzata per invertire il segnale risultante di un circuito complesso.
- L'istruzione INV può essere utilizzata per invertire il risultato del segnale delle istruzioni a impulsi LDP, LDF, ANP, ecc.

Utilizzo:




```
LD    X0
INV
OUT   Y1
```

Se l'ingresso X0 = 0, l'uscita Y1 = 1. Se l'ingresso X0 = 1, l'uscita Y1 = 0.

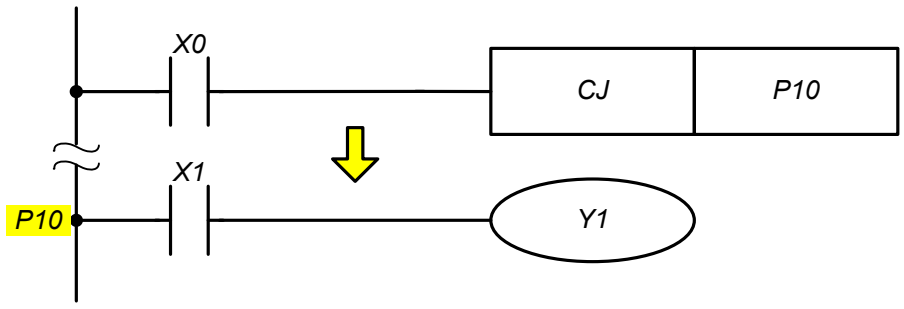
Istruzioni	Funzione
P	Indirizzamento di un punto di salto in un programma o sottoprogramma

Operando	0…31

Descrizione:

- L'istruzione P viene utilizzata per indicare un punto di transizione per le istruzioni CJ, CALL.
- Il numero del punto nel programma non deve essere ripetuto.

Utilizzo:




```
LD    X0
CJ    P10
.
.
P     10
LD    X1
OUT   Y1
```

Il punto P10 indica l'indirizzo di transizione per l'esecuzione dell'istruzione CJ P10.

Istruzioni	Funzione
I	Indirizzamento di un punto di interruzione

Operando	0…100, 1000…1007, 2000, 2001

Descrizione:

L'istruzione I viene utilizzata per indicare il punto di transizione al gestore di interruzione. Globalmente, le interruzioni vengono abilitate dall'istruzione EN e disabilitate dall'istruzione DS.

In totale, il controller può avere 15 interruzioni.

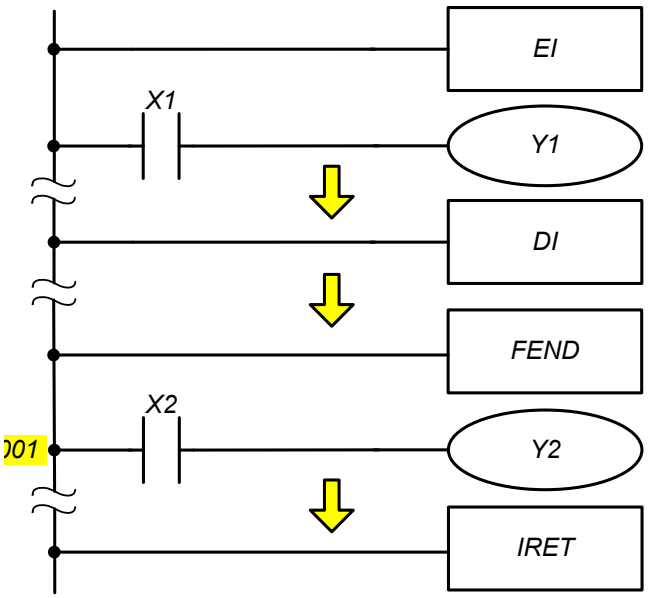
Un'interruzione che si verifica quando un frame Modbus (broadcast o indirizzato al controller) viene ricevuto tramite RS-485 è contrassegnata come I0.

È possibile organizzare fino a 4 interruzioni temporizzate nel controller In, dove n è il periodo di chiamata del gestore di interruzione di 10 ms e può avere un valore da 1 a 100. Quindi, $n = \frac{T}{10} ms$, dove T è il periodo di chiamata desiderato del gestore (misurato in ms).

8 interruzioni esterne **I1000**…**I1007** corrispondono agli ingressi discreti 0..7. L'interruzione si verifica quando il livello del segnale di ingresso cambia.

2 interruzioni del driver: **I2000**– si verifica in caso di errore e **I2001**– si verifica quando lo stato del motore cambia (fare riferimento alla sezione 8. “Istruzioni per il controllo del driver del motore passo-passo” per maggiori dettagli).

Utilizzo:




```
EI              Abilitazione interruzioni
LD    X1        Contatto NO X1
OUT   Y1        Uscita Y1
…
DI              Disabilitazione delle interruzioni
…
FEND            Fine del programma principale
I     1001      Punto di ingresso per il gestore di interrupt.
LD    X2        Contatto NO X2
OUT   Y2        Uscita Y2
…
```

IRET	Fine del gestore di interrupt

7. Istruzioni per l'applicazione

Istruzioni	Operandi	Varianti associate	Funzione
CJ	S	P	Salto condizionato

S	I puntatori P vengono utilizzati come operandi. Gli operandi possono essere indicizzati (A, B)

Descrizione:

Utilizzando l'istruzione CJ, è possibile saltare una parte del programma. Quando si applica questa istruzione, è possibile ridurre il tempo di esecuzione del programma. Ad esempio, saltando una sezione del programma allocata per l'inizializzazione delle periferiche del controller, l'attivazione delle interruzioni, ecc. (fare riferimento alla sezione 4.8 per maggiori dettagli).

CALL	S	P	Chiamata del sottoprogramma

S	I puntatori P vengono utilizzati come operandi. Gli operandi possono essere indicizzati (A, B)

Descrizione:

L'istruzione CALL viene utilizzata per chiamare un sottoprogramma.

- Un sottoprogramma è contrassegnato con i punti P e può essere richiamato da un'istruzione CALL.
- L'istruzione SRET deve essere posizionata alla fine del sottoprogramma.
- Un sottoprogramma deve essere posizionato dopo l'istruzione FEND e prima dell'istruzione END.
- Quando l'istruzione CALL viene eseguita, il controller va al punto contrassegnato.Dopo l'esecuzione dell'istruzione SRET, il controller ritorna al programma principale all'istruzione che segue CALL.
- I punti possono essere utilizzati con un numero illimitato di istruzioni CALL.
- I sottoprogrammi possono essere richiamati da altri sottoprogrammi. Max. 8 livelli di annidamento possibili.

SRET			Fine sottoprogramma

Descrizione:

L'istruzione SRET definisce la fine di un sottoprogramma (fare riferimento alla sezione 4.8per maggiori dettagli).

- Ogni sottoprogramma deve terminare con l'istruzione SRET.
- Il programma ritorna all'istruzione successiva all'istruzione CALL dopo aver elaborato SRET.
- L'istruzione SRET può essere utilizzata solo insieme all'istruzione CALL.

Nota:questa istruzione non richiede una condizione di ingresso (i contatti non sono necessari).

IRET			Fine del gestore di interrupt

Descrizione:

L'istruzione IRET definisce la fine dell'elaborazione dell'interruzione (fare riferimento alla sezione 4.8per maggiori dettagli).

Nota:questa istruzione non richiede una condizione di ingresso (i contatti non sono necessari).

EI			Abilitazione interrupt globali

Descrizione:

L'istruzione EI abilita l'elaborazione delle interruzioni (fare riferimento alla sezione 4.8per maggiori dettagli).

Nota:questa istruzione non richiede una condizione di ingresso (i contatti non sono necessari).

DI			Disabilitazione degli interrupt globali

Descrizione:
L'istruzione DI disabilita l'elaborazione delle interruzioni (fare riferimento alla sezione 4.8per maggiori dettagli).
Nota: questa istruzione non richiede una condizione di ingresso (i contatti non sono necessari).

Chiamata di un sottoprogramma di gestione delle interruzioni

- Durante l'elaborazione di un'interruzione, viene effettuata una transizione dal programma principale al gestore delle interruzioni.
- Dopo l'interruzione, l'elaborazione è completata, il controller ritorna al programma principale.
- L'inizio del sottoprogramma di interruzione è determinato dall'impostazione della marcatura (punto di interruzione).
- La fine del sottoprogramma di interruzione è determinata dall'istruzione IRET.
- Il sottoprogramma di interruzione deve essere programmato alla fine del programma utente dopo l'istruzione FEND e prima dell'istruzione END.

Nota: Se nessuna delle due istruzioni EI o DI è programmata, la modalità di interruzione non è attivata, ovvero nessuno dei segnali di interruzione verrà elaborato.

Esecuzione di un sottoprogramma di interruzione

Diversi sottoprogrammi di interruzione che si susseguono vengono elaborati in sequenza alla loro chiamata. Se più sottoprogrammi di interruzione vengono chiamati contemporaneamente, il programma di interruzione con l'indirizzo di punto più basso viene elaborato per primo.

FOR	S		Inizio di un ciclo FOR-NEXT

	K	H	F	X	Y	M	T	C	A	B	D
S	•	•					•	•	•	•	•

Nota:questa istruzione non richiede una condizione di ingresso (i contatti non sono necessari).

NEXT			Fine di un ciclo FOR-NEXT

Nota:questa istruzione non richiede una condizione di ingresso (i contatti non sono necessari).

Cicli

Le istruzioni FOR/NEXT vengono utilizzate per programmare ripetizioni cicliche di parti di programma (ciclo di programma).

Descrizione:

- La parte del programma tra le istruzioni FOR e NEXT viene ripetuta "n" volte.Dopo aver completato l'istruzione FOR, il programma procede al passo del programma dopo l'istruzione NEXT.
- Il valore "n" può essere compreso tra +1 e +32 767.Se viene impostato un valore compreso tra 0 e -32 768, il ciclo FOR-NEXT viene eseguito una sola volta.
- Sono possibili fino a 8 livelli di annidamento di cicli FOR-NEXT.
- Le istruzioni FOR e NEXT possono essere utilizzate solo in coppia.Ogni istruzione FOR deve corrispondere all'istruzione NEXT.

Fonte di errori

Gli errori appaiono nel programma nei seguenti casi:

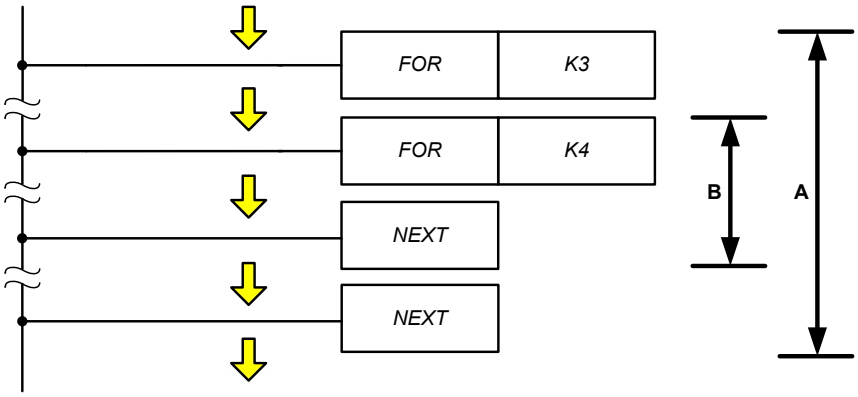
- L'istruzione successiva precede l'istruzione FOR.
- Il numero di istruzioni NEXT è diverso dal numero di istruzioni FOR.
- Un numero elevato di ripetizioni di "n" può aumentare significativamente il tempo di esecuzione di un programma.

Un esempio di utilizzo delle istruzioni FOR/NEXT:

L'esempio seguente mostra due cicli FOR-NEXT, uno dentro l'altro.

La parte del programma A viene eseguita 3 volte (K3 significa numero decimale 3).

La parte del programma B viene eseguita 4 volte all'interno di ogni ripetizione della parte A (K4 significa numero decimale 4).



CMP	S1 S2 D	D P	Confronto dei dati numerici

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
D					•	•					

Nota: l'operando D occupa 3 indirizzi.

Descrizione:

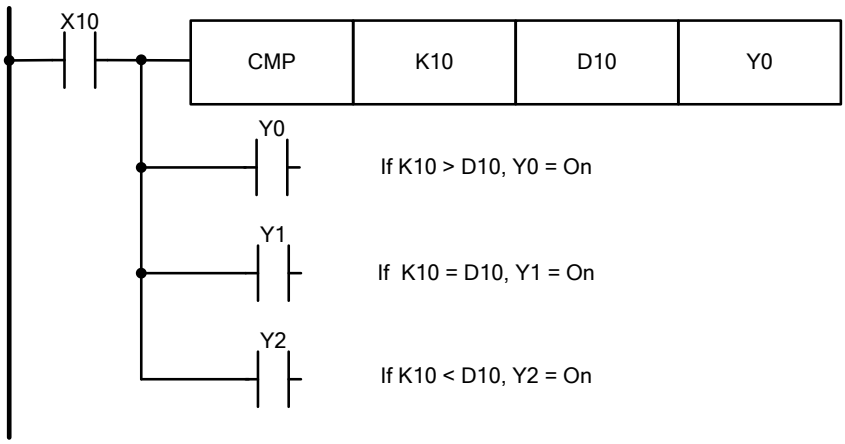
Confronto di due valori numerici (maggiore, minore, uguale).

- I dati in entrambe le sorgenti (S1) e (S2) vengono confrontati.
- Il risultato del confronto (maggiore, minore, uguale) viene visualizzato (indicato) attivando il relè M o l'uscita Y. Quale dei contatti all'operando di destinazione (D) è attivo, è determinato dal risultato del confronto:
  (S1) > (S2) → (D)

(S1) = (S2) → (D+1)
(S1) < (S2) → (D+2)

- I dati in S1 e S2 vengono elaborati come dati interi con segno.

Esempio:



Y0: si attiva se K10 > registro dati D10, Y1 e Y2 sono disattivati.

Y1: si attiva se K10 = registro dati D10, Y0 e Y2 sono disattivati.

Y2: si attiva se K10 < registro dati D10, Y0 e Y1 sono disattivati.

Y0, Y1, Y2 non vengono modificati se la condizione di ingresso X10=0.

Per resettare i risultati del confronto, utilizzare le istruzioni RST, ZRST.

ZCP	S1 S2 S D	D P	Confronto di zona dei dati numerici

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
S	•	•					•	•	•	•	•
D					•	•					

Nota:

- L'operando D occupa 3 indirizzi.
- L'operando S1 deve essere minore di S2.

Descrizione:

- Confronto di valori numerici con aree dati numeriche (maggiore, minore, uguale)
- I dati nell'origine (S) vengono confrontati con i dati in entrambe le origini (S1) e (S2)
- Il risultato del confronto (maggiore, minore, uguale) viene visualizzato (indicato) attivando il relè M o l'uscita Y. Quale dei contatti all'operando di destinazione (D) è attivo, è determinato dal risultato del confronto.

(S) < (S1) → (D)
(S1) ≤ (S) ≤ (S2) → (D + 1)
(S) > (S2) → (D + 2)

— Se il valore in (S1) è maggiore del valore in (S2), tutti i contatti nell'operando (D) vengono resettati.

Per resettare i risultati del confronto, utilizzare le istruzioni RST, ZRST.

MOV	S D	D P	Trasferimento dati

	K	H	F	X	Y	M	T	C	A	B	D
S	•	•	•	•	•	•	•	•	•	•	•
D					•	•	•	•	•	•	•

Descrizione:

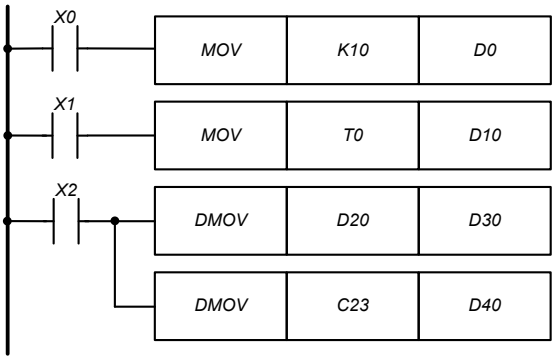
- L'istruzione MOV viene utilizzata per trasferire dati da un'origine dati (S) a una destinazione (D).Il valore dell'origine (S) non cambia.
- I dati nell'origine dati (S) vengono letti come valori binari durante l'esecuzione dell'istruzione MOV.
- Gli operandi bit occupano il numero di indirizzi corrispondente al tipo di istruzione - 16 o 32 indirizzi.In questo caso, è possibile combinare i tipi di operandi per l'origine e la destinazione.Ad esempio, come risultato dell'esecuzione del comando MOV D3 M0, i relè M0 ...M15 visualizzano il valore del registro D3 in forma binaria.

Esempio:

Se la condizione di ingresso X0 è attivata, il valore del registro D0 = 10. Se X0 è disattivato, il valore del registro D0 non viene modificato.

Se la condizione di ingresso X1 è attivata, il valore corrente del timer T0 viene trasferito al registro dati D10. Se X1 è disattivato, il valore del registro D10 non viene modificato.

Se la condizione di ingresso X2 è attivata, il valore dei registri D20 e D21 viene trasferito ai registri dati D30 e D31; il valore corrente del contatore C23 viene trasferito ai registri dati D40 e D41.



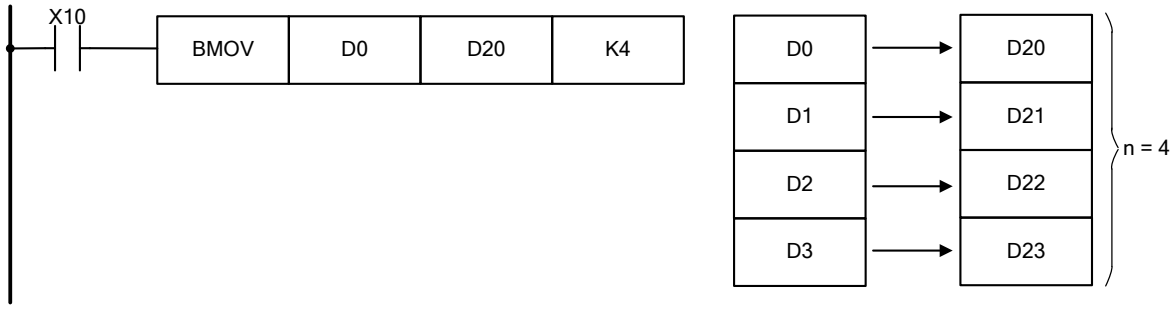
BMOV	S D n	D P	Trasferimento dati a blocchi

	K	H	F	X	Y	M	T	C	A	B	D
S	•	•	•	•	•	•	•	•	•	•	•
D					•	•	•	•	•	•	•
n	•	•					•	•	•	•	•

Descrizione:

Copia il pacchetto di dati. Lo spostamento durante l'operazione viene eseguito sia per l'operando di origine (S) che per l'operando di destinazione (D) a (n) elementi di blocco, a seconda dell'istruzione (16 bit o 32 bit).

Esempio:



Se X10 è attivato, i valori dei registri D0 – D3 vengono trasferiti ai registri D20 – D23.

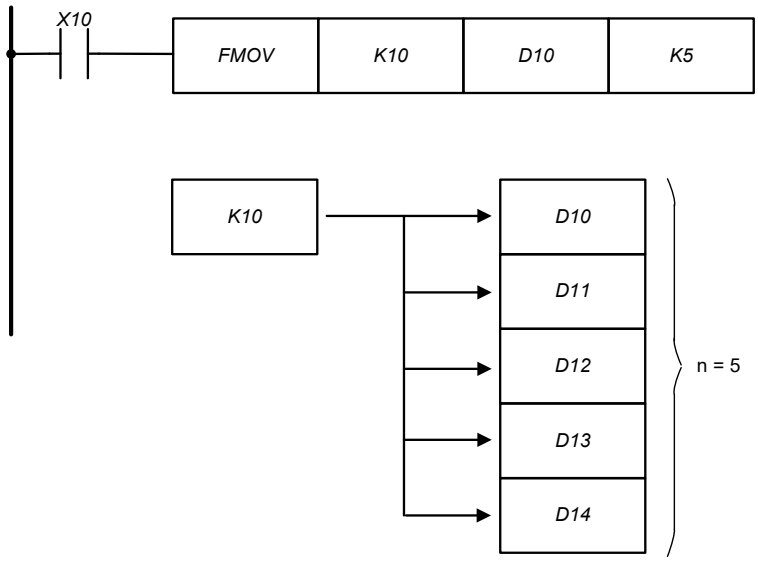
FMOV	S D n	D P	Trasferimento dati a indirizzi multipli

	K	H	F	X	Y	M	T	C	A	B	D
S	•	•	•	•	•	•	•	•	•	•	•
D					•	•	•	•	•	•	•
n	•	•					•	•	•	•	•

Descrizione:

Il valore dell'operando di origine (S) viene copiato su (n) operandi di destinazione (D) dello stesso tipo.

Esempio:



L'istruzione FMOV copia il valore "10" nei registri dati D10.. .D14.

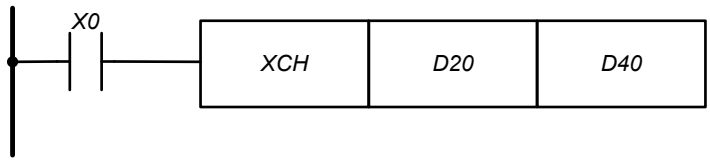
XCH	D1 D2	D P	Scambio dati

	K	H	F	X	Y	M	T	C	A	B	D
D1							•	•	•	•	•
D2							•	•	•	•	•

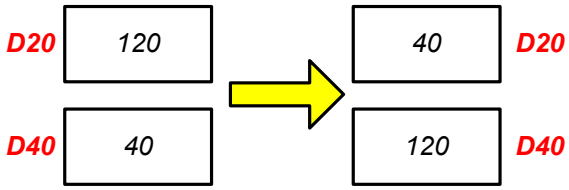
Descrizione:

I valori degli operandi (D1) e (D2) vengono scambiati.

Esempio:



Se X0 = 1, viene eseguito lo scambio di dati:



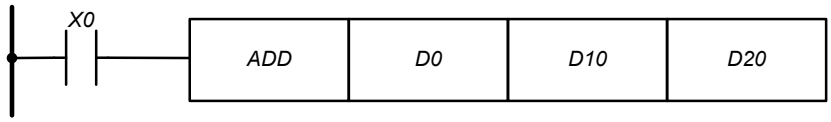
ADD	S1 S2 D	D P	Addizione di dati numerici

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
D							•	•	•	•	•

Descrizione:

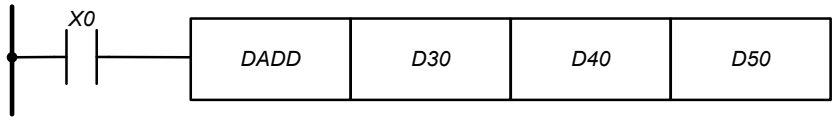
- I dati binari negli operandi sorgente (S1) e (S2) vengono sommati.Il risultato dell'addizione viene memorizzato nell'operando di destinazione (D). L'operazione viene eseguita su tipi di dati interi con segno.
  (S1) + (S2) = (D)
- Il bit più significativo contiene il segno del risultato: 0 – segno di un numero positivo, 1 – segno di un numero negativo.
- Quando si esegue un'istruzione a 32 bit, i 16 bit inferiori devono essere indicati nell'operando. Il seguente registro dati contiene i 16 bit più significativi.

Esempi:



(D0) + (D10) = (D20)

Se X0 è attivato, i valori dei registri dati D0 e D10 vengono sommati, il risultato viene salvato nel registro dati D20.



(D31, D30) + (D41, D40) = (D51, D50)

Se X0 è attivato, il risultato della somma dei valori dei registri (D31, D30) e (D41, D40) viene salvato nei registri dati (D51, D50).

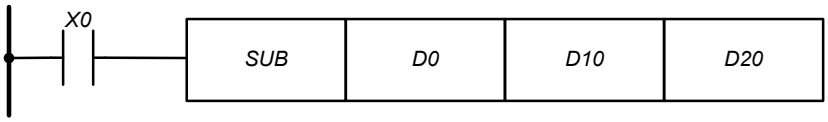
SUB	S1 S2 D	D P	Sottrazione di dati numerici

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
D							•	•	•	•	•

Descrizione:

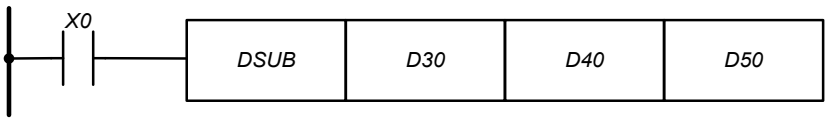
- Il valore dei dati in (S2) viene sottratto dal valore dei dati (S1). Il risultato della sottrazione viene memorizzato nell'operando di destinazione (D). L'operazione viene eseguita su tipi di dati interi con segno.
  (S1) – (S2) = (D)
- Il bit più significativo contiene il segno del risultato: 0 – segno di un numero positivo, 1 – segno di un numero negativo.
- Quando si esegue un'istruzione a 32 bit, i 16 bit inferiori devono essere indicati nell'operando. Il seguente registro dati contiene i 16 bit più significativi.

Esempi:



(D0) - (D10) = (D20)

Se X0 è attivato, viene calcolata la differenza tra i valori di dati nei registri D0 e D10. Il risultato viene salvato nel registro dati D20.



(D31, D30) - (D41, D40) = (D51, D50)

Se X0 è attivato, viene calcolata la differenza tra i valori di dati nei registri (D31, D30) e (D41, D40). Il risultato viene salvato nei registri dati (D51, D50).

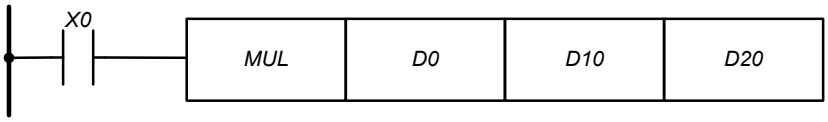
MUL	S1 S2 D	D P	Moltiplicazione di dati numerici

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
D							•	•	•	•	•

Descrizione:

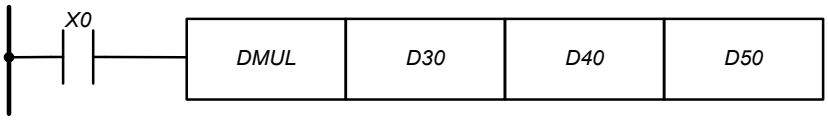
- I dati negli operandi (S1) e (S2) vengono moltiplicati tra loro.Il risultato viene memorizzato nell'operando di destinazione (D). L'operazione viene eseguita su tipi di dati interi con segno.
  (S1) x (S2) = (D)
- Il bit più significativo contiene il segno del risultato: 0 – segno di un numero positivo, 1 – segno di un numero negativo.
- Quando si esegue un'istruzione a 32 bit, i 16 bit inferiori devono essere indicati nell'operando. Il seguente registro dati contiene i 16 bit più significativi.

Esempi:



(D0) * (D10) = (D20)

Se X0 è attivato, i valori nei registri dati D0 e D10 vengono moltiplicati tra loro. Il risultato viene salvato nel registro dati D20.



(D31, D30) * (D41, D40) = (D51, D50)

Se X0 è attivato, i valori nei registri (D31, D30) e (D41, D40) vengono moltiplicati tra loro. Il risultato viene salvato nei registri dati (D51, D50).

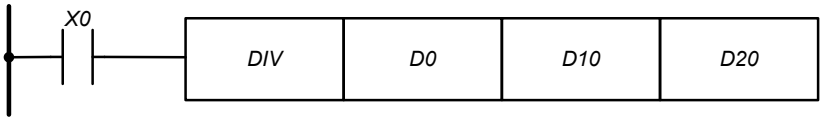
DIV	S1 S2 D	D P	Divisione di dati numerici

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
D							•	•	•	•	•

Descrizione:

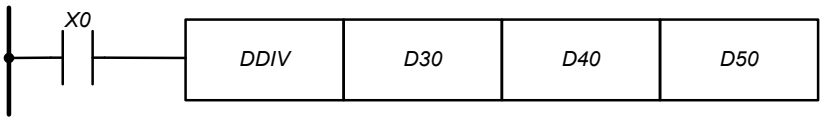
- Il valore dell'operando sorgente (S1) viene diviso per il valore dei dati dell'operando sorgente (S2). La parte intera del risultato della divisione viene memorizzata nell'operando di destinazione (D). L'operazione viene eseguita su tipi di dati interi con segno.
  (S1) / (S2) = (D)
- Il bit più significativo contiene il segno del risultato: 0 – segno di un numero positivo, 1 – segno di un numero negativo.
- Quando si esegue un'istruzione a 32 bit, i 16 bit inferiori devono essere indicati nell'operando. Il seguente registro dati contiene i 16 bit più significativi.
- La divisione per zero genera un errore.

Esempi:



(D0) / (D10) = (D20)

Se X0 è attivato, viene eseguita la divisione dei valori di dati nei registri D0 e D10. Il risultato viene salvato nel registro dati D20.



(D31, D30) / (D41, D40) = (D51, D50)

Se X0 è attivo, viene eseguita la divisione dei valori di dati nei registri (D31, D30) e (D41, D40). Il risultato viene salvato nei registri dati (D51, D50).

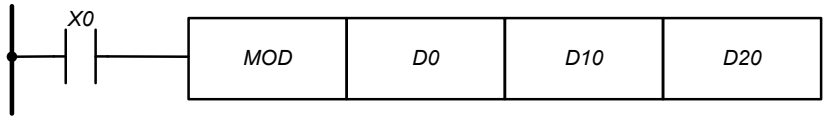
MOD	S1 S2 D	D P	Calcolo del resto della divisione

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
D							•	•	•	•	•

Descrizione:

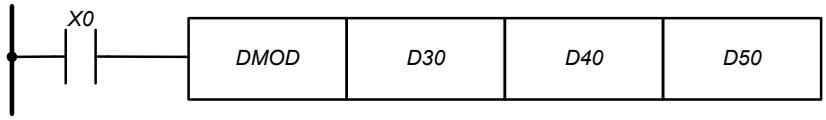
- Il valore dell'operando sorgente (S1) viene diviso per il valore dei dati dell'operando sorgente (S2). Il resto della divisione viene memorizzato nell'operando di destinazione (D). L'operazione viene eseguita su tipi di dati interi con segno.
  (S1) % (S2) = (D)
- Il bit più significativo contiene il segno del risultato: 0 – segno di un numero positivo, 1 – segno di un numero negativo.
- Quando si esegue un'istruzione a 32 bit, i 16 bit inferiori devono essere indicati nell'operando. Il seguente registro dati contiene i 16 bit più significativi.
- La divisione per zero genera un errore.

Esempi:



(D0) % (D10) = (D20)

Se X0 è attivato, viene eseguita la divisione dei valori di dati nei registri D0 e D10. Il risultato (resto della divisione) viene salvato nel registro dati D20.



(D31, D30) % (D41, D40) = (D51, D50)

Se X0 è attivo, viene eseguita la divisione dei valori di dati nei registri (D31, D30) e (D41, D40). Il risultato (resto della divisione) viene salvato nei registri dati (D51, D50).

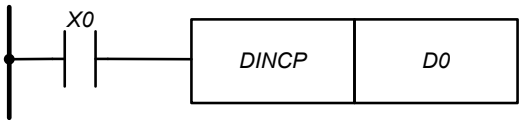
INC	D	D P	Incrementa dati numerici

	K	H	F	X	Y	M	T	C	A	B	D
D							•	•	•	•	•

Descrizione:

Il valore nell'operando (D) viene incrementato di 1.

Esempio:



Il valore nei registri dati (D1, D0) viene incrementato di 1 se la condizione di ingresso X0 è attivata. L'istruzione viene attivata grazie alla funzione a impulsi collegata in modo che il processo di somma non venga eseguito in ogni ciclo del programma.

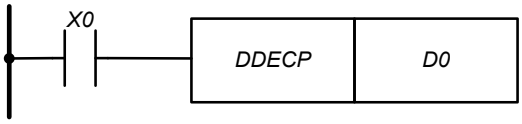
DEC	D	D P	Decrementa i dati numerici

	K	H	F	X	Y	M	T	C	A	B	D
D							•	•	•	•	•

Descrizione:

Il valore nell'operando (D) viene decrementato di 1.

Esempio:



Il valore nei registri dati (D1, D0) viene decrementato di 1 se la condizione di ingresso X0 è attivata. L'istruzione viene attivata grazie alla funzione a impulsi collegata in modo che il decremento non venga eseguito in ogni ciclo del programma.

WAND	S1 S2 D	D P	Moltiplicazione logica di dati numerici (operazione "AND")

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
D							•	•	•	•	•

Nota: WANDè un'istruzione a 16 bit, DANDè un'istruzione a 32 bit.

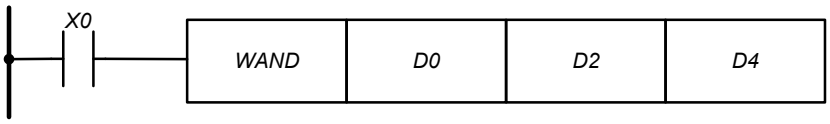
Descrizione:

— L'operazione "AND logico" per i dati numerici è un'operazione bit per bit (eseguita bit per bit).

— I valori degli operandi sorgente (S1) e (S2) vengono moltiplicati bit per bit. Il risultato viene memorizzato nell'operando di destinazione (D).
— La tabella di verità della moltiplicazione logica:

(S1)	(S2)	(D)
1	1	1
1	0	0
0	1	0
0	0	0

Esempio:



Se X0 = 1, viene eseguita la moltiplicazione logica dei valori dei registri dati D0 e D2. Il risultato viene salvato nel registro dati D4.

		b15														b00
(S1)	D0	1	1	1	1	1	0	0	0	0	0	1	1	0	0	0	1
									WAND							
(S2)	D2	0	0	1	1	0	1	1	0	0	1	1	0	1	1	0	0
		⇩														
(D)	D4	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0

WOR	S1 S2 D	D P	Addizione logica di dati numerici (operazione OR)

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
D							•	•	•	•	•

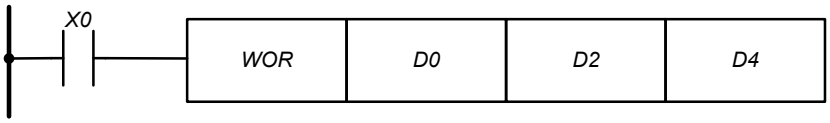
Nota: WORè un'istruzione a 16 bit, DORè un'istruzione a 32 bit.

Descrizione:

— L'operazione "OR logico" per i dati numerici è un'operazione bit per bit (eseguita bit per bit).
— I valori degli operandi sorgente (S1) e (S2) vengono sommati bit per bit. Il risultato viene memorizzato nell'operando di destinazione (D).
— La tabella di verità dell'addizione logica:

(S1)	(S2)	(D)
1	1	1
1	0	1
0	1	1
0	0	0

Esempio:



Se X0 = 1, viene eseguita l'addizione logica dei valori dei registri dati D0 e D2. Il risultato viene salvato nel registro dati D4.

		b15														b00
(S1)	D0	1	1	1	1	1	0	0	0	0	0	1	1	0	0	0	1
								WOR								
(S2)	D2	0	0	1	1	0	1	1	0	0	1	1	0	1	1	0	0
		⇩														
(D)	D4	1	1	1	1	1	1	1	0	0	1	1	1	1	1	0	1

WXOR	S1 S2 D	D P	Operazione logica "OR esclusivo"

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
D							•	•	•	•	•

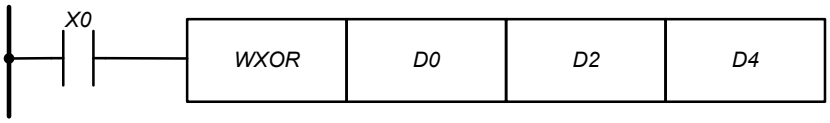
Nota: WXORè un'istruzione a 16 bit, DXORè un'istruzione a 32 bit.

Descrizione:

- L'operazione "XOR logico" (OR esclusivo) per i dati numerici è un'operazione bit per bit (eseguita bit per bit).
- I valori degli operandi sorgente (S1) e (S2) vengono elaborati bit per bit. Il risultato viene memorizzato nell'operando di destinazione (D).
- La tabella di verità dell'OR esclusivo logico:

(S1)	(S2)	(D)
1	1	0
1	0	1
0	1	1
0	0	0

Esempio:



Se X0 = 1, l'operazione “OR esclusivo” viene eseguita con i valori nei registri dati D0 e D2. Il risultato dell'operazione viene salvato nel registro dati D4.

		b15														b00
(S1)	D0	1	1	1	1	1	0	0	0	0	0	1	1	0	0	0	1
								WXOR								
(S2)	D2	0	0	1	1	0	1	1	0	0	1	1	0	1	1	0	0
		⇩														
(D)	D4	1	1	0	0	1	1	1	0	0	1	0	1	1	1	0	1

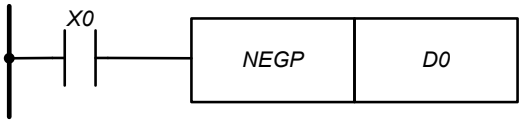
NEG	D	D P	Negazione logica

	K	H	F	X	Y	M	T	C	A	B	D
D							•	•	•	•	•

Descrizione:

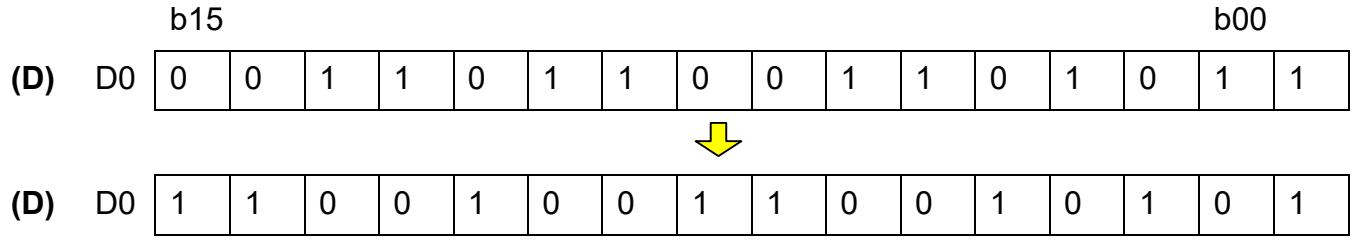
Operazione di negazione logica (inversione di tutti i bit in forma binaria e addizione con 1) per dati numerici.

Esempio:

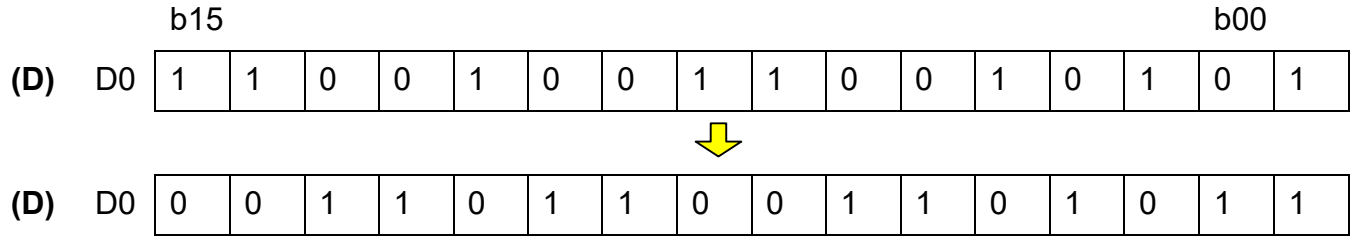


Se X0 = 1, l'operazione di negazione logica e modifica viene eseguita con il valore nell'operando D0.

13931 → -13931 + 1 = -13931



-13931 → 13930 + 1 = 13931



ROL	D n	D P	Spostamento ciclico a sinistra

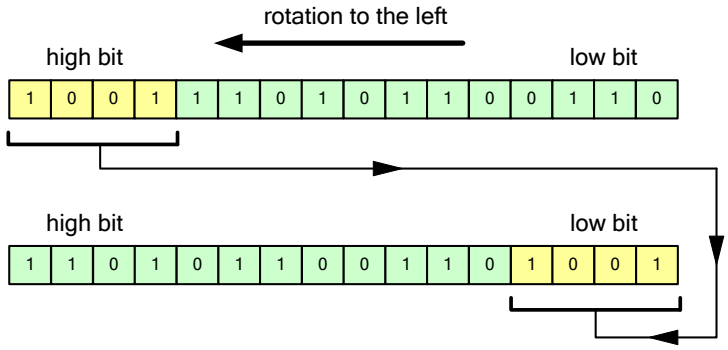
	K	H	F	X	Y	M	T	C	A	B	D
D							•	•	•	•	•
n	•	•					•	•	•	•	•

Descrizione:

Rotazione dei bit di (n) posizioni a sinistra.

Esempio:

Se X0 = 1, i bit del valore nel registro dati D10 ruotano di 4 bit a sinistra e il valore viene modificato.



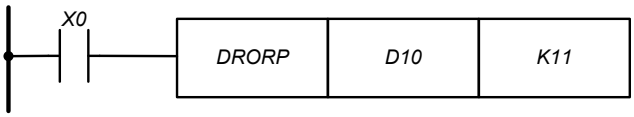
ROR	D n	P	Rotazione ciclica a destra

	K	H	F	X	Y	M	T	C	A	B	D
D							•	•	•	•	•
n	•	•					•	•	•	•	•

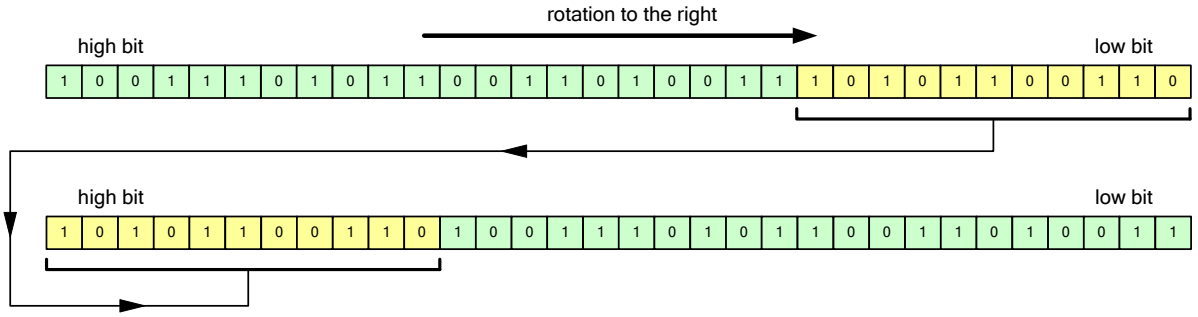
Descrizione:

Rotazione dei bit di (n) posizioni a destra.

Esempio:



Se X0 = 1, i bit del valore nel registro dati D10 ruotano di 11 bit a destra e il valore viene modificato.



ZRTS	D1 D2	D P	Ripristino di gruppo degli operandi

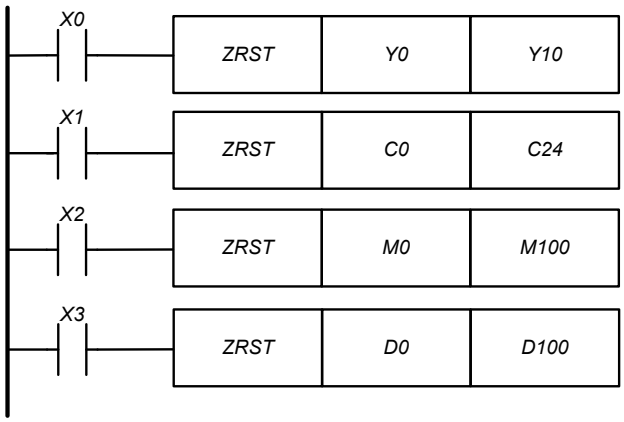
	K	H	F	X	Y	M	T	C	A	B	D
D1					•	•	•	•	•	•	•
D2					•	•	•	•	•	•	•

Descrizione:

I valori di più operandi consecutivi (regione di operandi) possono essere azzerati dall'istruzione ZRST. I contatti bit vengono disattivati, i registri vengono impostati sul valore "0".

- Gli operandi (D1) e (D2) determinano la regione da azzerare.
- Gli operandi in (D1) e (D2) devono essere dello stesso tipo.
- (D1) – il primo indirizzo della regione, (D2) – l'ultimo indirizzo della regione.
- (D1) deve essere minore di (D2).

Esempio:



Quando le condizioni di ingresso sono soddisfatte, gli operandi bit Y0 ... Y10, M0 ... M100 vengono disattivati (passano allo stato "0"). Gli operandi numerici C0 ... C24, D0 ... D100 vengono portati al valore effettivo "0". Le bobine e i contatti corrispondenti vengono disattivati.

DECO	S D n	P	Decoder da 8 → 256 bit

	K	H	F	X	Y	M	T	C	A	B	D
S	•	•		•	•	•	•	•	•	•	•
D					•	•	•	•	•	•	•
n	•	•					•	•	•	•	•

Nota: Se (D) è un operando a bit: (n) = 1…8. Se (D) è un operando numerico: (n) = 1…4. Se (n) è al di fuori dell'intervallo possibile, l'istruzione viene eseguita con il massimo (n) possibile a seconda di (D).

Descrizione:

Decodifica dei dati. I dati in (n) operandi vengono decodificati a partire dall'indirizzo iniziale, specificato in (S). L'operando (D) determina l'indirizzo iniziale della destinazione (dove viene scritto il risultato della decrittografia).

(n) è il numero di operandi i cui dati devono essere decodificati. Quando si specifica l'operando bit in (D), è necessario osservare quanto segue: 1 ≤ (n) ≤ 8. Quando si specifica l'operando numerico in (D), è necessario osservare quanto segue: 1 ≤ (n) ≤ 4.

(S) è l'indirizzo iniziale degli operandi i cui dati devono essere decodificati.

2^n– numero di operandi da decodificare.

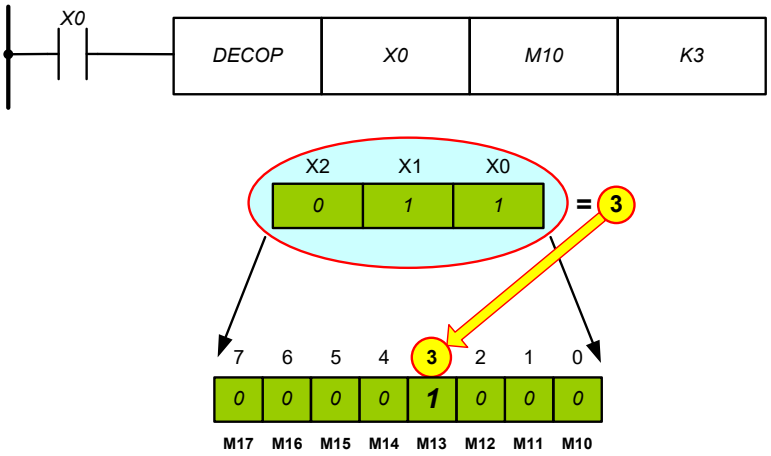
(D) è un indirizzo iniziale per l'operando di destinazione.

Attenzione! L'istruzione non viene eseguita se (n) = 0.

Di conseguenza, l'output rimane attivo se le condizioni di input alla fine dell'azione si disattivano nuovamente.

Esempio:

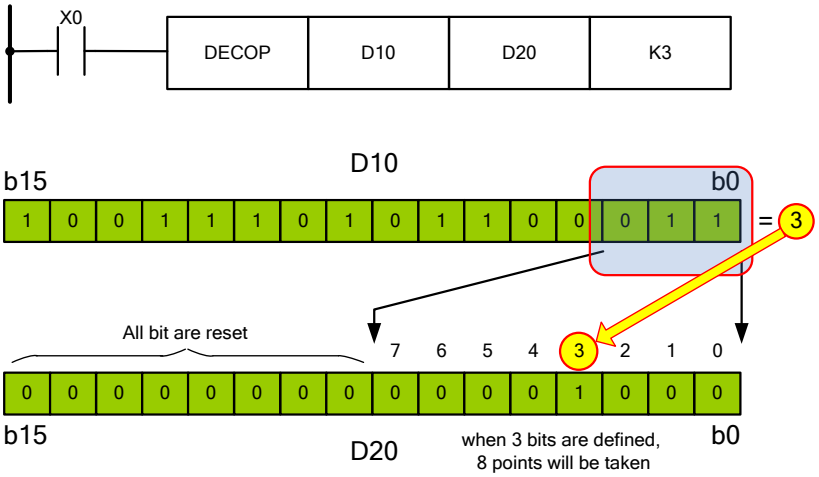
Utilizzo di un'istruzione DECO con operandi bit in (D).



Se (n) = 3, gli operandi di input X0, X1 e X2 vengono elaborati. $2^n = 2^3 = 8$ indirizzi vengono utilizzati come destinazione M10…M17.

Il valore degli operandi di input è 1 + 2 = 3. Quindi il terzo indirizzo di destinazione, ovvero il relè M13, viene attivato. Se viene elaborato il valore dell'operando di input "0", viene attivato il relè M10.

Utilizzo di un'istruzione DECO con operandi numerici in (D).



I 3 bit meno significativi del registro dati D10 vengono decodificati. Il risultato della decodifica 1 + 2 = 3 viene trasferito al registro dati D20. Il 3° bit viene attivato in questo registro dati.

Se il valore di (n) <3, allora tutti i bit non necessari di un numero maggiore negli indirizzi di destinazione vengono impostati a zero.

ENCO	S D n	P	Codificatore a 256 → 8 bit

	K	H	F	X	Y	M	T	C	A	B	D
S	•	•		•	•	•	•	•	•	•	•
D					•	•	•	•	•	•	•
n	•	•					•	•	•	•	•

Nota: Se (D) è un operando a bit: (n) = 1…8. Se (D) è un operando numerico: (n) = 1…4. Se (n) è al di fuori dell'intervallo possibile, l'istruzione viene eseguita con il massimo (n) possibile a seconda di (D).

Descrizione:

Codifica dei dati. I dati in 2^noperandi vengono codificati a partire dall'indirizzo iniziale, che è specificato in (S). L'operando (D) determina l'indirizzo iniziale della destinazione (dove viene scritto il risultato della decrittografia).

2^nè il numero di operandi i cui dati devono essere codificati.

(n) – il numero di operandi di destinazione.

Quando si specifica l'operando bit in (S), è necessario osservare quanto segue: $1 \leq (n) \leq 8$. Quando si specifica l'operando numerico in (S), è necessario osservare quanto segue: $1 \leq (n) \leq 4$.

(S) è l'indirizzo iniziale degli operandi i cui dati devono essere codificati.

(D) è l'indirizzo iniziale dell'operando di destinazione.

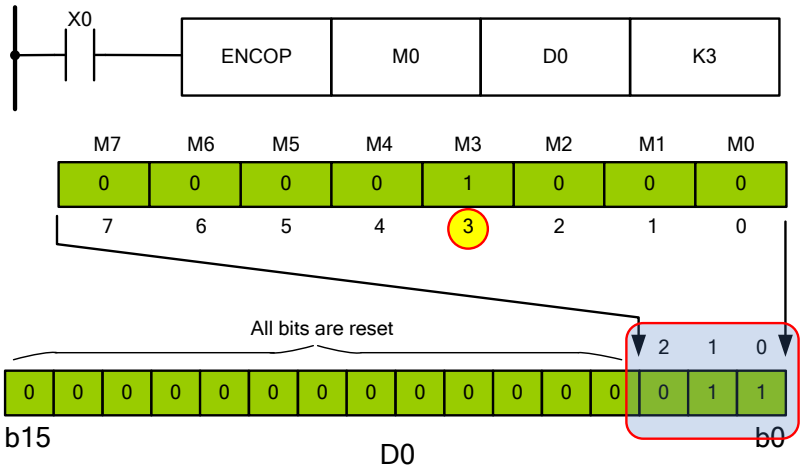
Se più operandi specificati in (S) hanno il valore “1”, allora viene elaborato solo il bit meno significativo.

Attenzione! L'istruzione non viene eseguita se (n) = 0.

Di conseguenza, l'output rimane attivo se le condizioni di input alla fine dell'azione si disattivano nuovamente.

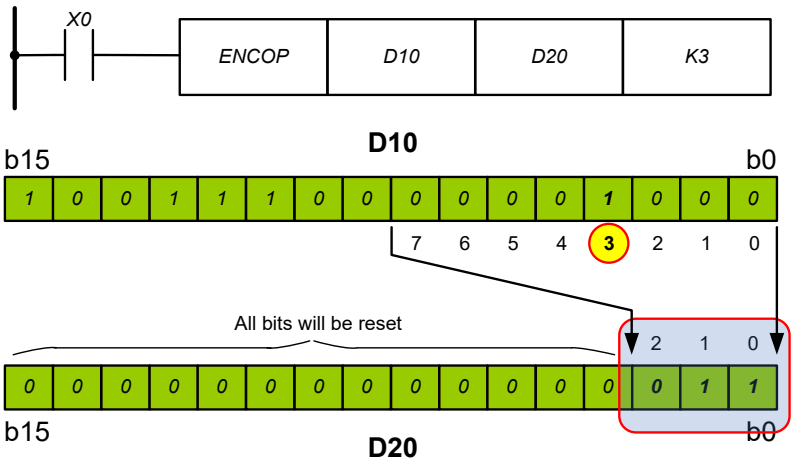
Esempio:

Utilizzo di un'istruzione ENCO con operandi bit in (S).



Se 2n = 23 = 8, allora gli indirizzi dei relè di uscita sono M0.. .M7. Quando l'uscita 3d è attivata (ovvero, M3 = 1), il valore 3 viene scritto nel registro dati D0.

Utilizzo di un'istruzione ENCO con operandi numerici in (S).



Il 3° bit è attivo nel registro dati D10. Quindi il valore 3 viene codificato e salvato nel registro dati D20.

SUM	S D	D P	Somma dei singoli bit

	K	H	F	X	Y	M	T	C	A	B	D
S	•	•					•	•	•	•	•
D							•	•	•	•	•

Descrizione:

— Determinazione del numero di bit attivi in una parola dati. L'istruzione conta i bit attivati in (S).

— Il valore risultante viene scritto in (D).

Se viene elaborata un'istruzione a 32 bit, i 16 bit più significativi (D + 1) degli operandi di destinazione (D) vengono impostati a zero, poiché il numero massimo di bit attivati in (S) è 32.

BON	S D n	D P	Verificare lo stato del bit

	K	H	F	X	Y	M	T	C	A	B	D
S	•	•					•	•	•	•	•
D					•	•					
n	•	•					•	•	•	•	•

Nota: La condizione necessaria: (n) = 0…15 (16 bit), (n) = 0…31 (32 bit).

Descrizione:

Un singolo bit viene controllato all'interno della parola dati. Se il bit (n) è attivato in (S), allora il bit corrispondente in (D) viene attivato.

SQR	S D	D P	Calcolo della radice quadrata

	K	H	F	X	Y	M	T	C	A	B	D
S	•	•					•	•	•	•	•
D											•

Descrizione:

Calcolo della radice quadrata$(D) = \sqrt{(S)}$

Viene calcolata la radice quadrata del valore nell'operando (S), il risultato viene arrotondato a un intero e scritto nell'operando (D). L'operazione viene eseguita su tipi di dati interi con segno.

Attenzione! La radice quadrata di un numero negativo porta sempre a un errore.

FLT	S D	D P	Converti intero in virgola mobile

	K	H	F	X	Y	M	T	C	A	B	D
S	•	•					•	•	•	•	•
D											•

Descrizione:

L'istruzione FLT converte un numero intero con segno in formato a virgola mobile.

- L'intero nell'operando (S) viene convertito in un numero a virgola mobile. Il risultato viene salvato nell'operando (D).
- Il risultato della conversione è sempre un numero a 32 bit.

PWM	S1 S2 D		Uscita impulso PWM

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
D					•						

Nota:Il valore dell'operando (S1) deve essere minore o uguale al valore dell'operando (S2).

Descrizione:

(S1) – durata dell'impulso, t.

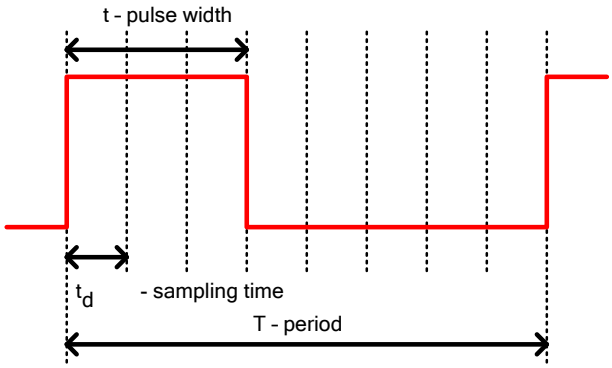
(S2) – durata del periodo, T.

(D) – indirizzo di uscita, Y6 o Y7.

I valori validi per (S1) e (S2) sono da 1 a 32767. (S1) e (S2) sono il numero di intervalli di campionamento. Il periodo di campionamento viene impostato simultaneamente per entrambi i canali, 100 µs o 10 µs, dal registro speciale D356 (vedere la sezione 4.6per maggiori dettagli).

Un segnale PWM è presente all'uscita finché il segnale all'ingresso dell'istruzione PWM è attivo.

Esempio:



Sia il periodo di campionamento 100 µs. Quando la condizione di ingresso X0 = 1 è soddisfatta, il segnale PWM con un periodo di 8 x 100 µs = 0.8 ms e una durata dell'impulso di 3 x 100 µs = 0.3 ms appare all'uscita Y6.

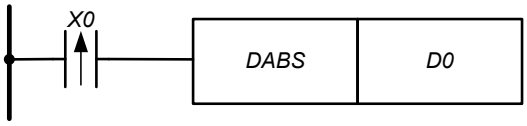
ABS	D	D P	Valore assoluto

	K	H	F	X	Y	M	T	C	A	B	D
D							•	•	•	•	•

Descrizione:

L'istruzione ABS scrive il valore assoluto di un numero nell'operando (D). Se il valore in (D) è negativo, dopo l'esecuzione dell'istruzione ABS, il segno "-" viene eliminato e il numero diventa positivo. Se (D) ha un valore positivo, non si verificano modifiche.

Esempio:



Quando la condizione di ingresso è soddisfatta, viene determinato il modulo del numero nel registro (D1, D0).

POW	S1 S2 D	D P	Elevamento a potenza

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
D											•

Descrizione:

Elevamento a potenza: $(D) = (S1)^{(S2)}$.

L'istruzione POW eleva il valore nell'operando (S1) alla potenza di (S2). Il risultato viene salvato nell'operando (D). L'operazione viene eseguita su tipi di dati interi con segno.

DECMP	S1 S2 D	D P	Confronto di numeri a virgola mobile

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•	•								•
S2	•	•	•								•
D					•	•					

Nota:

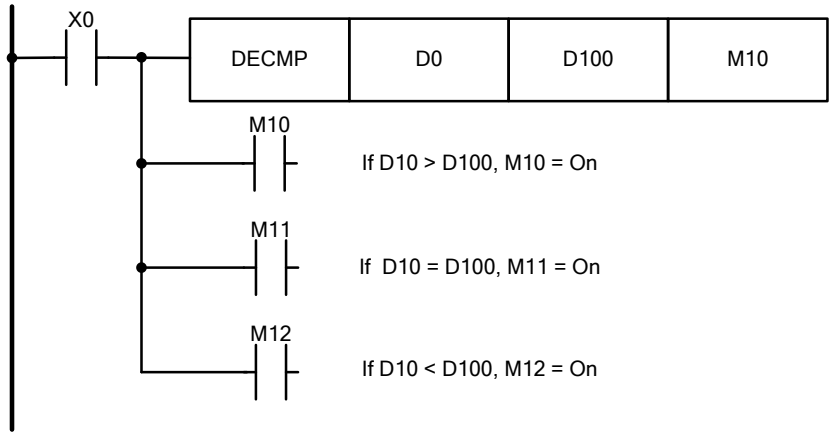
- Istruzione a 32 bit.
- L'operando (D) occupa 3 indirizzi consecutivi.
- I tipi K e H non vengono convertiti in F, ma vengono proiettati su un'area di memoria. Per convertire i tipi di dati interi in virgola mobile, utilizzare l'istruzione **FLT**.

Descrizione:

L'istruzione DECMP esegue il confronto di due numeri binari a virgola mobile e restituisce il risultato del confronto.

- L'istruzione DECMP confronta il numero a virgola mobile in (S1) con il numero a virgola mobile in (S2).
- Il risultato del confronto viene memorizzato in 3 indirizzi successivi.
- Se il numero in (S2) è minore del numero in (S1), l'operando bit (D) viene attivato.
- Se il numero in (S2) è uguale al numero in (S1), l'operando bit ((D) +1) viene attivato.
- Se il numero in (S2) è maggiore del numero in (S1), l'operando bit ((D) +2) viene attivato.
- Gli operandi di uscita interrogati rimangono attivati dopo la disattivazione delle condizioni di ingresso dell'istruzione DECMP.

Esempio:



Quando il contatto X0 viene attivato, il numero a virgola mobile specificato in D100 (S2) viene confrontato con il numero a virgola mobile specificato in D0 (S1). Se il numero in D100 è minore del numero D0, il relè M10 viene attivato. Se il numero in D100 è uguale al numero D0, il relè M11 viene attivato. Se il numero in D100 è maggiore del numero D0, il relè M12 viene attivato.

Per ottenere i risultati del confronto nella forma: ≤ ≥ ≠, è possibile utilizzare le combinazioni di contatti paralleli M10 - M12. Per resettare il risultato, è possibile utilizzare le istruzioni RST, ZRST.

DEZCP	S1 S2 S D	D P	Confronto a virgola mobile di zona

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•	•								•
S2	•	•	•								•
S	•	•	•								•
D					•	•					

Nota:

- Istruzione a 32 bit.
- L'operando (D) occupa 3 indirizzi consecutivi.
- I tipi K e H non vengono convertiti in F, ma vengono proiettati su un'area di memoria. Per convertire i tipi di dati interi in virgola mobile, utilizzare l'istruzione **FLT**.

Descrizione:

Confronto del numero in virgola mobile con l'area selezionata (indicata) e l'output del risultato del confronto.

- L'istruzione DEZCP confronta il numero in virgola mobile nell'operando (S1) con l'area compresa tra (S1) e (S2).
- Il risultato del confronto viene memorizzato in 3 indirizzi successivi.
- Se il numero nell'operando (S) è minore dei numeri negli operandi (S1) e (S2), allora l'operando bit (D) viene attivato.
-
- Se il numero nell'operando (S) è uguale ai numeri compresi tra (S1) e (S2), allora l'operando bit ((D) +1) viene attivato.
- Se il numero nell'operando (S) è maggiore dei numeri compresi tra (S1) e (S2), allora l'operando bit ((D) +2) viene attivato.
- Gli operandi di output interrogati rimangono attivi dopo che le condizioni di ingresso dell'istruzione DEZCP sono disabilitate.
- Se il numero nell'operando (S1) è maggiore del numero nell'operando (S2), allora tutti i bit in (D), (D+1) e (D+2) verranno resettati.

DEADD	S1 S2 D	D P	Addizione di numeri a virgola mobile

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•	•								•
S2	•	•	•								•
D											•

Nota:

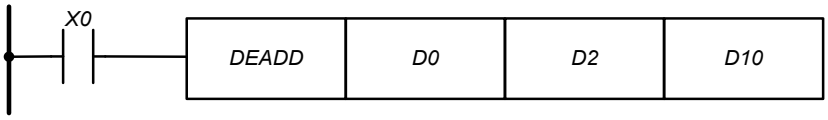
- Istruzione a 32 bit.
- I tipi K e H non vengono convertiti in F, ma vengono proiettati su un'area di memoria. Per convertire i tipi di dati interi in virgola mobile, utilizzare l'istruzione**FLT**.

Descrizione:

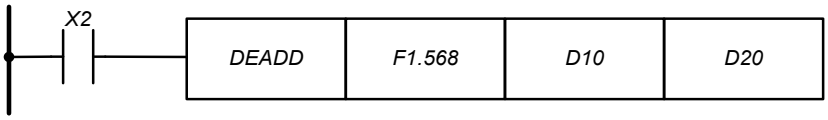
Calcolo della somma di due numeri in formato binario a virgola mobile.

- I numeri in virgola mobile specificati in (S1) e (S2) vengono sommati.Il risultato dell'addizione viene memorizzato nell'operando di destinazione (D).
- Due registri consecutivi vengono utilizzati per ciascun operando.
- Lo stesso operando può essere utilizzato sia come sorgente che come destinazione. In questo caso, il risultato calcolato viene nuovamente memorizzato nell'operando di origine e può essere utilizzato per il calcolo successivo. Questo processo viene ripetuto in ogni ciclo del programma.

Esempi:



Quando l'ingresso X0 è attivo, il numero in virgola mobile in (D3, D2) viene aggiunto al numero in virgola mobile in (D1, D0). Il risultato viene salvato in (D11, D10).



Quando l'input X2 è attivato, il numero a virgola mobile in (D11, D10) viene aggiunto alla costante F1.568. Il risultato viene salvato in (D21, D20).

DESUB	S1 S2 D	D P	Sottrazione di numeri a virgola mobile

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•	•								•
S2	•	•	•								•
D											•

Nota:

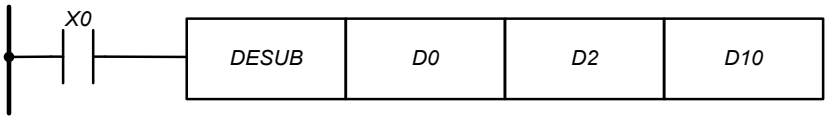
- Istruzione a 32 bit.
- I tipi K e H non vengono convertiti in F, ma vengono proiettati su un'area di memoria. Per convertire i tipi di dati interi in virgola mobile, utilizzare l'istruzione **FLT**.

Descrizione:

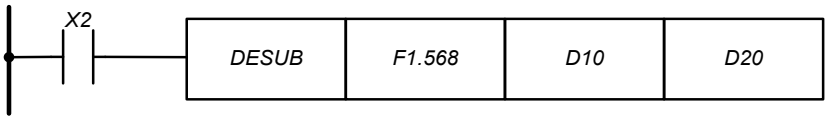
L'istruzione DESUB calcola la differenza di due numeri in formato binario a virgola mobile.

— Il numero in virgola mobile specificato in (S2) viene sottratto dal numero in virgola mobile specificato in (S1). Il risultato viene salvato in (D).
— Due registri consecutivi vengono utilizzati per ciascun operando.
— Lo stesso operando può essere utilizzato sia come sorgente che come destinazione. In questo caso, il risultato calcolato viene nuovamente memorizzato nell'operando di origine e può essere utilizzato per il calcolo successivo. Questo processo viene ripetuto in ogni ciclo del programma.

Esempi:



Quando l'ingresso X0 è attivo, il numero in virgola mobile in (D3, D2) viene sottratto dal numero in virgola mobile in (D1, D0). Il risultato viene salvato in (D11, D10).



Quando l'ingresso X2 è attivato, il numero a virgola mobile in (D11, D10) viene sottratto dalla costante F1.568. Il risultato viene salvato in (D21, D20).

DEMUL	S1 S2 D	D P	Moltiplicazione di numeri a virgola mobile

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•	•								•
S2	•	•	•								•
D											•

Nota:

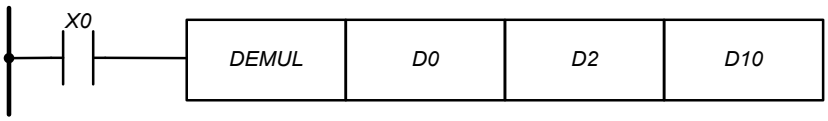
— Istruzione a 32 bit.
— I tipi K e H non vengono convertiti in F, ma vengono proiettati su un'area di memoria. Per convertire i tipi di dati interi in virgola mobile, utilizzare l'istruzione **FLT**.

Descrizione:

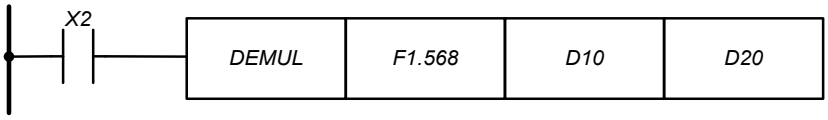
Moltiplicazione di due numeri in formato binario a virgola mobile.

— Il numero in virgola mobile specificato nell'operando (S1) viene moltiplicato per il numero in virgola mobile nell'operando (S2). Il risultato viene salvato nell'operando (D).
— Due registri consecutivi vengono utilizzati per ciascun operando.
— Lo stesso operando può essere utilizzato sia come sorgente che come destinazione. In questo caso, il risultato calcolato viene nuovamente memorizzato nell'operando di origine e può essere utilizzato per il calcolo successivo. Questo processo viene ripetuto in ogni ciclo del programma.

Esempi:



Quando l'ingresso X0 è attivo, il numero in virgola mobile in (D1, D0) viene moltiplicato per il numero in virgola mobile in (D3, D2). Il risultato viene salvato in (D11, D10).



Quando l'ingresso X2 è attivo, la costante F1.568 viene moltiplicata per il numero in virgola mobile in (D11, D10). Il risultato viene salvato in (D21, D20).

DEDIV	S1 S2 D	D P	Divisione di numeri a virgola mobile

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•	•								•
S2	•	•	•								•
D											•

Nota:

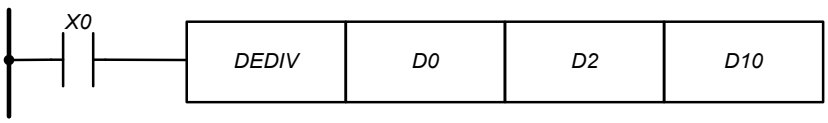
- Istruzione a 32 bit.
- I tipi K e H non vengono convertiti in F, ma vengono proiettati su un'area di memoria. Per convertire i tipi di dati interi in virgola mobile, utilizzare l'istruzione **FLT**.

Descrizione:

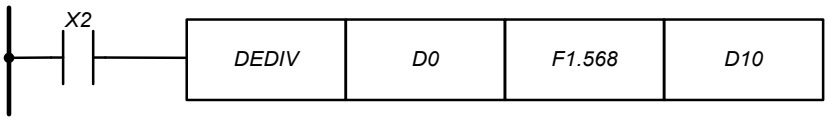
Calcolo del quoziente della divisione di due numeri in formato binario a virgola mobile.

- Il numero in virgola mobile specificato nell'operando (S1) viene diviso per il numero in virgola mobile specificato nell'operando (S2). Il risultato viene salvato in (D).
- Due registri consecutivi vengono utilizzati per ciascun operando.
- Lo stesso operando può essere utilizzato sia come sorgente che come destinazione. In questo caso, il risultato calcolato viene nuovamente memorizzato nell'operando di origine e può essere utilizzato per il calcolo successivo. Questo processo viene ripetuto in ogni ciclo del programma.
- L'operando (S2) non può essere zero perché la divisione per zero non è consentita.

Esempi:



Quando l'ingresso X0 è attivo, il numero in virgola mobile in (D1, D0) viene diviso per il numero in virgola mobile in (D3, D2). Il risultato viene salvato in (D11, D10).



Quando l'ingresso X2 è attivo, il numero in virgola mobile in (D1, D0) viene diviso per la costante F1.568. Il risultato viene salvato in (D11, D10).

DESQR	S D	D P	Radice quadrata in formato a virgola mobile

	K	H	F	X	Y	M	T	C	A	B	D
S	•	•	•								•
D											•

Nota:

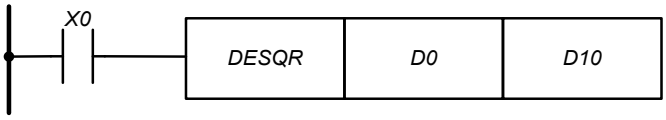
- Istruzione a 32 bit.
- I tipi K e H non vengono convertiti in F, ma vengono proiettati su un'area di memoria. Per convertire i tipi di dati interi in virgola mobile, utilizzare l'istruzione **FLT**.
- Condizione necessaria: (S) ≥ 0

Descrizione:

Calcolo della radice quadrata di un numero binario a virgola mobile.

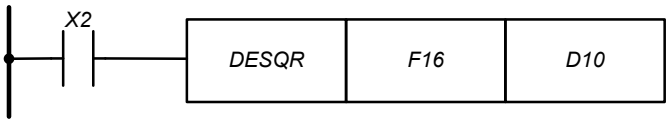
- La radice quadrata viene calcolata dal numero in virgola mobile specificato nell'operando (S). Il risultato viene salvato in (D).
- Due registri consecutivi vengono utilizzati per ciascun operando.
- Lo stesso operando può essere utilizzato sia come sorgente che come destinazione. In questo caso, il risultato calcolato viene nuovamente memorizzato nell'operando di origine e può essere utilizzato per il calcolo successivo. Questo processo viene ripetuto in ogni ciclo del programma.

Esempi:



$$\sqrt{(D1, D0)} \rightarrow (D11, D10)$$

Quando l'ingresso X0 è attivo, viene calcolata la radice quadrata del numero in virgola mobile in (D1, D0). Il risultato viene salvato in (D11, D10).



Quando l'ingresso X0 è attivato, viene calcolata la radice quadrata della costante F16. Il risultato viene salvato in (D11, D10).

DEPOW	S1 S2 D	D P	Elevamento a potenza in formato a virgola mobile

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•	•								•
S2	•	•	•								•
D											•

Nota:

- Istruzione a 32 bit.
- I tipi K e H non vengono convertiti in F, ma vengono proiettati su un'area di memoria. Per convertire i tipi di dati interi in virgola mobile, utilizzare l'istruzione **FLT**.

Descrizione:

Elevamento a potenza di un numero in formato binario a virgola mobile.

- Il numero specificato in (S1) viene elevato alla potenza (S2). Il risultato viene salvato nell'operando (D). $(S1)^{(S2)} = (D)$.
- Due registri consecutivi vengono utilizzati per ciascun operando.
- Lo stesso operando può essere utilizzato sia come sorgente che come destinazione. In questo caso, il risultato calcolato viene nuovamente memorizzato nell'operando di origine e può essere utilizzato per il calcolo successivo. Questo processo viene ripetuto in ogni ciclo del programma.

INT	S D	D P	Conversione di un numero a virgola mobile in un intero

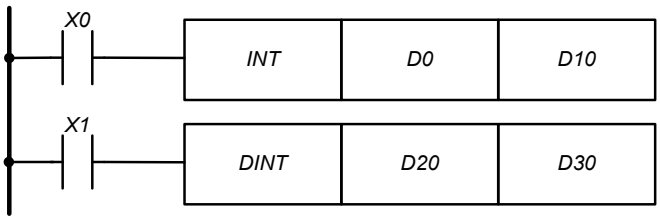
	K	H	F	X	Y	M	T	C	A	B	D
S	.										•
D							•	•	•	•	•

Descrizione:

L'istruzione INT converte i numeri a virgola mobile in numeri interi, arrotondati al numero intero più vicino.

- Il numero a virgola mobile specificato in (S) viene arrotondato al valore intero più vicino e salvato in (D).
- L'operando sorgente è sempre una doppia parola.
- Quando si utilizza l'istruzione INT, l'operando parola è l'operando di destinazione.
- Quando si utilizza l'istruzione DINT, l'operando di destinazione è un operando a doppia parola.
- L'istruzione INT è la funzione inversa dell'istruzione FLT.

Esempio:



Quando l'ingresso X0 è attivato, il numero a virgola mobile in (D0, D1) viene arrotondato al valore intero inferiore più vicino. Il risultato viene salvato in D10.

Quando l'ingresso X1 è attivato, il numero a virgola mobile in (D20, D21) viene arrotondato al valore intero inferiore più vicino. Il risultato viene salvato in (D30, D31).

TRD	D	P	Tempo di lettura dei dati

	K	H	F	X	Y	M	T	C	A	B	D
D											•

Nota: L'operando D occupa 3 indirizzi consecutivi.

Descrizione:

Lettura del valore corrente dell'orologio in tempo reale.

- Utilizzando l'istruzione TRD, vengono letti i dati in tempo reale (ore, minuti, secondi).
- Questi dati vengono salvati in 3 indirizzi operandi (D) consecutivi.

Esempio:

Quando l'input X0 è attivato, i dati in tempo reale vengono letti e salvati nei registri D0 ... D2

Registro	Funzione	Valore	Esempio	
D0	Secondi	0…59	20	12:36:20
D1	Minuti	0…59	36	
D2	Ore	0…23	12	

TWR	S	P	Registrazione dei dati temporali

	K	H	F	X	Y	M	T	C	A	B	D
S											•

Nota: L'operando (S) occupa 3 indirizzi consecutivi.

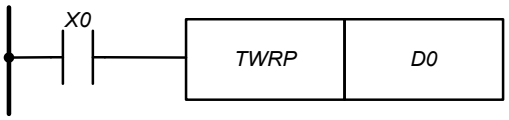
Descrizione:

L'istruzione TWR viene utilizzata per modificare i dati in tempo reale (ore, minuti, secondi).

I dati vengono prelevati da 3 indirizzi consecutivi, specificati in (S).

Se i valori in (S) superano l'intervallo di valori consentito, si verifica un errore.

Esempio:



Quando la condizione di ingresso è soddisfatta, l'orologio in tempo reale del controller viene impostato sui valori indicati nei registri D0 ... D2.

Registro	Funzione	Valore	Esempio	
D0	Secondi	0…59	42	03:11:42
D1	Minuti	0…59	11	
D2	Ore	0…23	3	

DRD	D	P	Lettura dei dati di data

	K	H	F	X	Y	M	T	C	A	B	D
D											•

Nota: L'operando D occupa 3 indirizzi consecutivi.

Descrizione:

Lettura del valore corrente della data.

- Utilizzando l'istruzione DRD, viene letta la data corrente (giorno, mese, anno).
- Questi dati vengono salvati in 3 indirizzi operandi (D) consecutivi.

Esempio:

Quando l'input X0 è attivato, i dati in tempo reale vengono letti e salvati nei registri D0 ... D2.

Registro	Funzione	Valore	Esempio	
D0	Giorno	1…31	26	26.01.22
D1	Mese	1…12	1	
D2	Anno	21…99	22	

DWR	S	P	Registrazione dei dati di data

	K	H	F	X	Y	M	T	C	A	B	D
S											•

Nota: L'operando (S) occupa 3 indirizzi consecutivi.

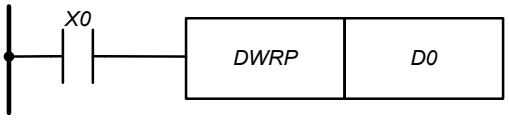
Descrizione:

L'istruzione DWR viene utilizzata per modificare i dati della data (giorno, mese, anno).

I dati vengono prelevati da 3 indirizzi consecutivi, specificati in (S).

Se i valori in (S) superano l'intervallo di valori consentito, si verifica un errore.

Esempio:



Quando la condizione di ingresso è soddisfatta, l'orologio in tempo reale del controller viene impostato sui valori indicati nei registri D0 ... D2.

Registro	Funzione	Valore	Esempio	
D0	Giorno	1…31	4	04.02.22
D1	Mese	1…12	2	
D2	Anno	21…99	22	

LD#	S1 S2	D	Operazioni logiche di tipo contatto

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•		•	•	•	•	•	•	•	•
S2	•	•		•	•	•	•	•	•	•	•

Nota:

- Il simbolo **#** è &, |, ^.
- Gli operandi di bit vengono presi a 16 o 32, a seconda del tipo di istruzione, e vengono convertiti in un tipo di dati intero per l'ulteriore elaborazione.

Descrizione:

Esecuzione dell'operazione logica "AND", "OR" ed "EXCLUSIVE OR" sugli operandi (S1) e (S2) e attivazione del contatto LD, a seconda del risultato dell'operazione.

Le istruzioni LD# nel programma si trovano a sinistra e aprono una connessione logica o sono condizioni per l'esecuzione dei comandi a destra.

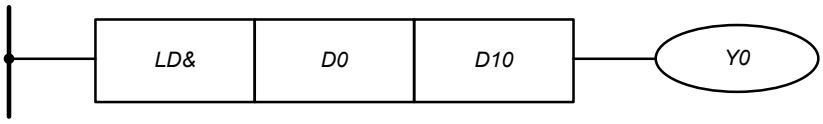
Istruzioni a 16 bit	Istruzioni a 32 bit	Contatto chiuso se:	Contatto aperto se:
LD&	DLD&	(S1) & (S2) ≠ 0	(S1) & (S2) = 0
LD|	DLD|	(S1) | (S2) ≠ 0	(S1) | (S2) = 0
LD^	DLD	(S1) ^ (S2) ≠ 0	(S1) ^ (S2) = 0

&: moltiplicazione logica (AND)

|: addizione logica (OR)

^: OR esclusivo (XOR)

Esempio:



AND#	S1 S2	D	Operazioni logiche di tipo contatto Connessione seriale

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•		•	•	•	•	•	•	•	•
S2	•	•		•	•	•	•	•	•	•	•

Nota:

- Il simbolo **#** è &, |, ^.
- Gli operandi di bit vengono presi a 16 o 32, a seconda del tipo di istruzione, e vengono convertiti in un tipo di dati intero per l'ulteriore elaborazione.

Descrizione:

Esecuzione dell'operazione logica "AND", "OR", "Exclusive OR" sugli operandi (S1) e (S2) e attivazione del contatto AND a seconda del risultato dell'operazione.

Le istruzioni AND# nel programma si trovano dopo i comandi LD e creano una connessione logica AND.

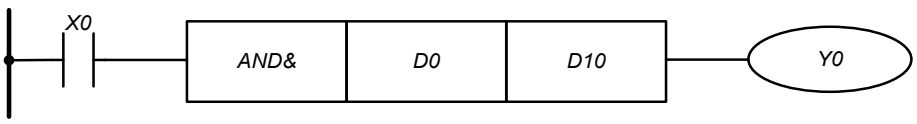
Istruzioni a 16 bit	Istruzioni a 32 bit	Contatto chiuso se:	Contatto aperto se:
AND&	DAND&	(S1) & (S2) ≠ 0	(S1) & (S2) = 0
AND|	DAND|	(S1) | (S2) ≠ 0	(S1) | (S2) = 0
AND^	DAND^	(S1) ^ (S2) ≠ 0	(S1) ^ (S2) = 0

&: moltiplicazione logica (AND)

|: addizione logica (OR)

^: OR esclusivo (XOR)

Esempio:



OR#	S1 S2	D	Operazioni logiche di tipo contatto Collegamento parallelo

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•		•	•	•	•	•	•	•	•
S2	•	•		•	•	•	•	•	•	•	•

Nota:

- Il simbolo **#** è &, |, ^.
- Gli operandi di bit vengono presi a 16 o 32, a seconda del tipo di istruzione, e vengono convertiti in un tipo di dati intero per l'ulteriore elaborazione.

Descrizione:

Esecuzione dell'operazione logica "AND", "OR", "Exclusive OR" sugli operandi (S1) e (S2) e attivazione del contatto OR a seconda del risultato dell'operazione.

Le istruzioni OR# nel programma si trovano a sinistra in parallelo all'istruzione LD e creano una connessione logica OR.

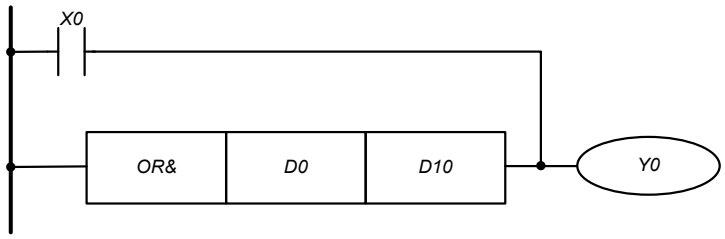
Istruzioni a 16 bit	Istruzioni a 32 bit	Contatto chiuso se:	Contatto aperto se:
OR&	DOR&	(S1) & (S2) ≠ 0	(S1) & (S2) = 0
OR|	DOR|	(S1) | (S2) ≠ 0	(S1) | (S2) = 0
OR^	DOR^	(S1) ^ (S2) ≠ 0	(S1) ^ (S2) = 0

&: moltiplicazione logica (AND)

|: addizione logica (OR)

^: OR esclusivo (XOR)

Esempio:



LD*	S1 S2	D	Operazioni di confronto del tipo di contatto

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•		•	•	•	•	•	•	•	•
S2	•	•		•	•	•	•	•	•	•	•

Nota:

- Il simbolo **#** è =, >, <, <>, ≤, ≥.
- Gli operandi di bit vengono presi a 16 o 32, a seconda del tipo di istruzione, e vengono convertiti in un tipo di dati intero per l'ulteriore elaborazione.

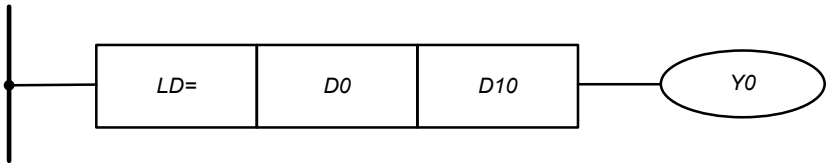
Descrizione:

Confronto dei valori degli operandi (S1) e (S2) e attivazione di un contatto LD, a seconda del risultato dell'operazione.

- Le istruzioni LD * nel programma si trovano a sinistra e iniziano una connessione logica o sono condizioni per l'esecuzione delle istruzioni a destra.
- Se il risultato del confronto è vero, il contatto LD viene attivato.
- Se il risultato del confronto è falso, il contatto LD viene disattivato.

Istruzioni a 16 bit	Istruzioni a 32 bit	Contatto chiuso se:	Contatto aperto se:
LD=	DLD=	(S1) = (S2)	(S1) ≠ (S2)
LD>	DLD>	(S1) > (S2)	(S1) ≤ (S2)
LD<	DLD<	(S1) < (S2)	(S1) ≥ (S2)
LD<>	DLD<>	(S1) ≠ (S2)	(S1) = (S2)
LD<=	DLD<=	(S1) ≤ (S2)	(S1) > (S2)
LD>=	DLD>=	(S1) ≥ (S2)	(S1) < (S2)

Esempio:



AND*	S1 S2	D	Operazioni di confronto del tipo di contatto Connessione seriale

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•		•	•	•	•	•	•	•	•
S2	•	•		•	•	•	•	•	•	•	•

Nota:

- Il simbolo **#** è =, >, <, <>, ≤, ≥.
- Gli operandi di bit vengono presi a 16 o 32, a seconda del tipo di istruzione, e vengono convertiti in un tipo di dati intero per l'ulteriore elaborazione.

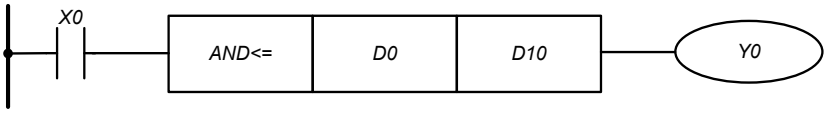
Descrizione:

Confronto dei valori degli operandi S1 e S2 e attivazione del contatto AND, a seconda del risultato dell'operazione.

- Le istruzioni AND* nel programma si trovano dopo i comandi LD e creano una connessione logica AND.
- Se il risultato del confronto è vero, il contatto AND viene attivato.
- Se il risultato del confronto è falso, il contatto AND viene disattivato.

Istruzioni a 16 bit	Istruzioni a 32 bit	Contatto chiuso se:	Contatto aperto se:
AND=	DAND=	(S1) = (S2)	(S1) ≠ (S2)
AND>	DAND>	(S1) > (S2)	(S1) ≤ (S2)
AND<	DAND<	(S1) < (S2)	(S1) ≥ (S2)
AND<>	DAND<>	(S1) ≠ (S2)	(S1) = (S2)
AND<=	DAND<=	(S1) ≤ (S2)	(S1) > (S2)
AND>=	DAND>=	(S1) ≥ (S2)	(S1) < (S2)

Esempio:



OR*	S1 S2	D	Operazioni di confronto del tipo di contatto Collegamento parallelo

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•		•	•	•	•	•	•	•	•
S2	•	•		•	•	•	•	•	•	•	•

Nota:

- Il simbolo **#** è =, >, <, <>, ≤, ≥.
- Gli operandi di bit vengono presi a 16 o 32, a seconda del tipo di istruzione, e vengono convertiti in un tipo di dati intero per l'ulteriore elaborazione.

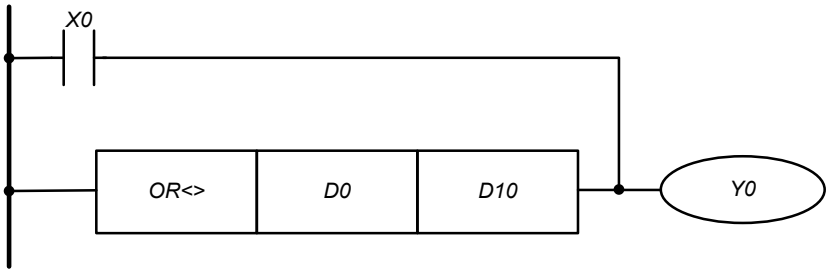
Descrizione:

Confronto dei valori degli operandi S1 e S2 e attivazione del contatto OR, a seconda del risultato dell'operazione.

- Le istruzioni OR* nel programma si trovano a sinistra in parallelo all'istruzione LD e creano una connessione OR logica.
- Se il risultato del confronto è vero, il contatto OR viene attivato.
- Se il risultato del confronto è falso, il contatto OR viene disattivato.

Istruzioni a 16 bit	Istruzioni a 32 bit	Contatto chiuso se:	Contatto aperto se:
OR=	DOR=	(S1) = (S2)	(S1) ≠ (S2)
OR>	DOR>	(S1) > (S2)	(S1) ≤ (S2)
OR<	DOR<	(S1) < (S2)	(S1) ≥ (S2)
OR<>	DOR<>	(S1) ≠ (S2)	(S1) = (S2)
OR<=	DOR<=	(S1) ≤ (S2)	(S1) > (S2)
OR>=	DOR>=	(S1) ≥ (S2)	(S1) < (S2)

Esempio:



8. Istruzioni per il controllo del driver del motore passo-passo

Il driver del motore passo-passo è controllato da comandi che specificano i parametri di rotazione o movimento. Tutti i comandi sono divisi in due gruppi: **RUN** e **MOVE**. Il gruppo RUN è progettato per controllare la velocità corrente dell'azionamento e MOVE per controllare il movimento. Per avviare una rotazione dopo aver selezionato un comando e impostato i parametri del driver, viene chiamata l'istruzione **SPIN**.

SPIN		P	Eseguire il movimento specificato
Indirizzo	**Tipo di oggetto**	**Descrizione dell'oggetto Modbus**	
5100h	**Bobine**	Scrivere “1” in questo oggetto (anche se non è resettato) attiva un movimento parametrizzato, il reset viene ignorato.	

Descrizione:

L'istruzione avvia una rotazione parametrizzata. I parametri di movimento sono impostati nei registri di servizio, che, come le istruzioni del driver del motore passo-passo, sono accessibili tramite il protocollo Modbus in modalità RUN. L'istruzione SPIN ha una priorità inferiore rispetto a xSTOP e xHIZ. Per evitare errori, si consiglia di controllare i flag BUSY_MOVE e BUSY_RUN prima di chiamare l'istruzione SPIN. Di seguito è riportata una descrizione dettagliata dei registri di servizio del driver.

Indirizzo	Tipo di oggetto	Registro di servizio		Dimensione (bit)	Descrizione
		Numero	Nome		
5000h	Registri di Mantenimento	D357	SPEED	32	La velocità (target) del motore impostata (in impulsi al secondo, pps) per i comandi del gruppo RUN e la velocità massima per i comandi del gruppo MOVE. Il limite inferiore è 8 pps, il limite superiore è 120000 pps nella condizione di FS_SW_EN resettato. Se FS_SW_EN è impostato, il limite superiore è $SPEED_{max}=120000*2^{U\_STEP}$.

Indirizzo	Tipo di oggetto	Registro di servizio		Dimensione (bit)	Descrizione
		Numero	Nome		
5002h	Registri di Mantenimento	D359	MIN_SPEED	32	La velocità minima di rotazione per i comandi del gruppo RUN. Per il gruppo MOVE, è anche la velocità minima se il flag CMIN_SPD_EN non è impostato. Se CMIN_SPD_EN è impostato, la velocità minima ottimale viene calcolata automaticamente.
5004h	Registri di Mantenimento	D361	ACC	16	Accelerazione, pps^2.
5005h	Registri di Mantenimento	D362	DEC	16	Decelerazione, pps^2.
5006h	Registri di Mantenimento	D363	ABS	32	Posizione attuale. L'unità di valore è pari allo spostamento di un microstep.
5008h	Registri di Input	D365	U_POS	16	La posizione corrente del microstep è in quattro passi completi. Indica la posizione del rotore del motore rispetto ai poli dello statore. Il registro è di sola lettura.

<table>
<tr><th rowspan="2">Indirizzo</th><th rowspan="2">Tipo di oggetto</th><th colspan="2">Registro di servizio</th><th rowspan="2">Dimensione (bit)</th><th rowspan="2">Descrizione</th></tr>
<tr><th>Numero</th><th>Nome</th></tr>
<tr><td>5009h</td><td>Registri di Mantenimento</td><td>D366</td><td>U_STEP</td><td>16</td><td>Micropassi<table><tr><th>Valore del registro</th><th>Micropass</th></tr><tr><td>0</td><td>1/1</td></tr><tr><td>1</td><td>1/2</td></tr><tr><td>2</td><td>1/4</td></tr><tr><td>3</td><td>1/8</td></tr><tr><td>4</td><td>1/16</td></tr><tr><td>5</td><td>1/32</td></tr><tr><td>7</td><td>1/128</td></tr><tr><td>8</td><td>1/256</td></tr></table>Un valore di “6” non è valido.</td></tr>
<tr><td>500Ah</td><td>Registri di Mantenimento</td><td>D374</td><td>DIR</td><td>16</td><td rowspan="2">Direzione di rotazione:
"1" – avanti,
"0" – indietro.</td></tr>
<tr><td>500Ah</td><td>Bobine</td><td colspan="3">DIR</td></tr>
<tr><td>500Bh</td><td>Registri di Mantenimento</td><td>D377</td><td>FS_SPD_THR</td><td>32</td><td>Il valore di soglia per il passaggio dalla modalità microstepping alla modalità full-step, misurato in passi interi al secondo.</td></tr>
<tr><td>500Dh</td><td>Registri di Mantenimento</td><td>D379</td><td>FS_SW_EN</td><td>16</td><td rowspan="2">Impostando l'oggetto su “1” si attiva la funzione morphing: il controller passa alla modalità full-step dopo aver raggiunto la velocità specificata in FS_SPD_THR. Questa funzione consente di ottenere una coppia maggiore alle alte velocità. Reimpostando l'oggetto su “0” si disattiva questa funzione (morphing/aumento di coppia).</td></tr>
<tr><td>500Dh</td><td>Bobine</td><td colspan="3">FS_SW_EN</td></tr>
</table>

Indirizzo	Tipo di oggetto	Registro di servizio		Dimensione (bit)	Descrizione
		Numero	Nome		
500Eh	Registri di Mantenimento	D372	TARGET_POS	32	La posizione target da raggiungere. L'unità di valore è pari allo spostamento di un microstep.
5010h	Registri di Mantenimento	D376	CMD	16	Un comando di movimento per il driver (fare riferimento alla tabella sottostante).
5011h	Registri di Mantenimento	D375	SW_INPUT	16	Il numero di input per la connessione del sensore – per i comandi GOUNTIL_... e …RELEASE. Valori 0…7. Attenzione: La dichiarazione di interruzione nel programma principale è necessaria per gli input utilizzati. La gestione dell'interruzione può essere lasciata vuota. Esempio: Un sensore è collegato all'ingresso X3 (IN3), il programma utente deve includere la parte seguente: *FEND* *I 1003* *IRET* *END*
5012h	Registri di Mantenimento	D367	ACC_CUR	16	Corrente di accelerazione, mA. Intervallo di valori validi: RS 434540 - 150…1500; RS 434542 – 1000…5000 RS 434543 – 2800…10000.

Indirizzo	Tipo di oggetto	Registro di servizio		Dimensione (bit)	Descrizione
		Numero	Nome		
5013h	Registri di Mantenimento	D368	DEC_CUR	16	Corrente di decelerazione, mA. Intervallo di valori validi: RS 434540 - 150…1500; RS 434542 – 1000…5000 RS 434543 – 2800…10000.
5014h	Registri di Mantenimento	D369	STEADY_CUR	16	Corrente a velocità costante, mA. Intervallo di valori validi: RS 434540 - 150…1500; RS 434542 – 1000…4200 RS 434543 – 2800…8000
5015h	Registri di Mantenimento	D370	HOLD_CUR	16	Corrente di mantenimento, mA. Intervallo di valori validi: RS 434540 - 150…1500; RS 434542 – 1000…4200 RS 434543 – 2800…8000
5016h	Registri di Mantenimento	D382	CMIN_SPD_EN	16	"1" - utilizza il calcolo automatico della velocità iniziale e finale di movimento per i comandi del gruppo MOVE. "0" - usa MIN_SPEED come velocità iniziale e finale.
5017h	Registri di Mantenimento	D380	ERROR_SET_HIZ	16	I bit di questo registro determinano quali errori del driver devono portare alla disattivazione del motore (l'albero ruota liberamente) - lo stato HiZ.

Indirizzo	Tipo di oggetto	Registro di servizio		Dimensione (bit)	Descrizione
		Numero	Nome		
5017h	Bobine	TERMAL_OVER_CURRENT_ ERROR_SET_HIZ			Bit 0 del registro D380. Se il bit è impostato, un errore TERMAL_ERROR (surriscaldamento del circuito del driver – registro D381) provoca la disattivazione del motore (stato HiZ).
5018h	Bobine	SOFTWARE_ERROR_SET_HIZ			1 bit del registro D380. Se il bit è impostato, un errore SOFTWARE_ERROR (l'errore interno del controller – il registro D381) causa la disattivazione del motore (stato HiZ).
5019h	Bobine	CMD_ERROR_SET_HIZ			Bit 2 del registro D380. Se il bit è impostato, un errore CMD_ERROR non è in grado di elaborare un comando in arrivo – il registro D381) causa la disattivazione del motore (stato HiZ).
501Ah	Bobine	DATA_ERROR_SET_HIZ			3-bit del registro D380. Se il bit è impostato, un errore DATA_ERROR (inserimento dati errato in ACC, DEC, U_STEP – il registro D381) provoca la disattivazione del motore (stato HiZ).
501Bh	Bobine	OUT_OF_LIM_MIN_SPD_ ERROR_SET_HIZ			4 bit del registro D380. Se il bit è impostato, un errore OUT_OF_LIM_MIN_SPD_E RROR_SET_HIZ (velocità impostata inferiore al minimo – registro D381) provoca la disattivazione del motore (stato HiZ).

Indirizzo	Tipo di oggetto	Registro di servizio		Dimensione (bit)	Descrizione
		Numero	Nome		
501Ch	Bobine	OUT_OF_LIM_MAX_SPD_ ERROR_SET_HIZ			5° bit del registro D380. Se il bit è impostato, un errore OUT_OF_LIM_MAX_SPD_ERROR_SET_HIZ (superamento della velocità massima possibile – il registro D381) causa la disattivazione del motore (stato HiZ).
501Dh	Bobine	UNREACHABLE_FS_SPD_ ERROR_SET_HIZ			6-bit del registro D380. Se il bit è impostato, un errore UNREACHABLE_FS_SPD_ ERROR_SET_HIZ (impossibile raggiungere la soglia di fullstepspeed in modalità torque boost – il registro D381) causa la disattivazione del motore (stato HiZ).
501Eh	Bobine	NOT_APP_FS_PARAM_ ERROR_SET_HIZ			Bit 7 del registro D380. Se il bit è impostato, un errore NOT_APP_FS_PARAM_ERROR_SET_HIZ (la transizione da aumento di coppia non è possibile mentre il motore è in rotazione – il registro D381) provoca la disattivazione del motore (stato HiZ).
5027h	Registri di Mantenimento	D381	ERROR_CODE	16	Codice di errore. Si veda di seguito la descrizione dei bit di registro (errori).

Indirizzo	Tipo di oggetto	Registro di servizio		Dimensione (bit)	Descrizione
		Numero	Nome		
5027h	Bobine	TERMAL_OVER_CURRENT_ ERROR			Bit 0 del registro D381 TERMAL_OVER_ ERRORE_CORRENTE – surriscaldamento del circuito del driver o corrente eccessiva negli avvolgimenti del motore.
5028h	Bobine	SOFT_ERROR			1 bit del registro D381 SOFTWARE_ERROR – errore interno del controller.
5029h	Bobine	CMD_ERROR			Bit 2 del registro D381 CMD_ERROR – impossibile elaborare un comando in ingresso.
502Ah	Bobine	DATA_ERROR			3-bit del registro D381 ERRORE_DATI – Inserimento dati errato in ACC, DEC, U_STEP
502Bh	Bobine	OUT_OF_LIM_MIN_SPD_ERROR			4 bit del registro D381 FUORI_LIMITE_ MIN_SPD_ERROR – Lavelocità impostata è inferiore al minimo.
502Ch	Bobine	OUT_OF_LIM_MAX_SPD_ERROR			5° bit del registro D381 FUORI_LIMITE_ MAX_SPD_ERROR – superamento della velocità massima possibile.
502Dh	Bobine	UNREACHABLE_FS_SPD_ERROR			6-bit del registro D381 IRRAGGIUNGIBILE _FS_SPD_ERROR – Impossibile raggiungere la soglia di velocità massima (full stepspeed) in modalità aumento di coppia (torque boost).

Indirizzo	Tipo di oggetto	Registro di servizio		Dimensione (bit)	Descrizione
		Numero	Nome		
502Eh	Bobine	NOT_APP_FS_PARAM_ERROR			Bit 7 del registro D381 . NOT_APP_FS_ PARAM_ERROR – transizione da incremento di coppia non possibile mentre il motore è in rotazione.
502F	Bobine	OVLO/UVLO_INTERNAL_ PROTECTION_ERROR			Errore: la tensione dei circuiti di alimentazione interni non rientra nell'intervallo standard.
5030	Bobine	VS_OUT_OF_RANGE_ERROR			Errore: la tensione di alimentazione non è compresa nell'intervallo consentito.
5037h	Registri di Input	D371	MOTOR_STATUS	16	Il registro mostra lo stato attuale del motore e del sistema di controllo. La descrizione dei bit del registro è riportata di seguito.
5037h	Ingressi Discreti	HIZ			Bit 0 del registro D371. Stato HiZ del motore (le fasi sono de-energizzate).
5038h	Ingressi Discreti	STOP			1 bit del registro D371. Modalità di mantenimento.
5039h	Ingressi Discreti	ACCELERATING			Bit 2 del registro D371. Accelerazione.
503Ah	Ingressi Discreti	DECELERATING			Bit 3 del registro D371. Decelerazione.
503Bh	Ingressi Discreti	STEADY			4 bit del registro D371. Rotazione a velocità costante.
503Ch	Ingressi Discreti	BUSY_MOVE			5-bit del registro D371. Il flag dell'impossibilità di applicare i comandi del gruppo MOVE.

Indirizzo	Tipo di oggetto	Registro di servizio		Dimensione (bit)	Descrizione
		Numero	Nome		
503Dh	Ingressi Discreti	BUSY_RUN			6-bit del registro D371. Il flag dell'impossibilità di usare i comandi del gruppo RUN.
5047h	Registri di Input	D383	CURRENT_SPD	32	Velocità attuale, pps. (Si raccomanda di utilizzare il flag STEADY come evento di raggiungimento di una determinata velocità.)
5048h	Registri di Mantenimento	D385	EMERGENCY_DEC	32	Decelerazione di emergenza, pps^2.
5100h	Bobine	SPIN (APPLY_CMD)			Impostando l'oggetto a “1” o applicando l'istruzione SPIN si attiva l'esecuzione del comando specificato nel registro CMD (D376), con i parametri specificati.
5101h	Bobine	TORQUE (APPLY_CURRENT)			Impostare un oggetto su “1” o applicare un'istruzione TORQUE applica i valori correnti ACC_CUR, DEC_CUR, RUN_CUR e HOLD_CUR per il motore.
5102h	Bobine	HSTOP (HARD_STOP)			Impostare un oggetto su “1” o applicare l'istruzione HSTOP arresta immediatamente il motore e lo porta in modalità di mantenimento.
5103h	Bobine	HHIZ (HARD_HIZ)			Impostare un oggetto su “1” o applicare l'istruzione HHIZ porta immediatamente il motore allo stato HiZ.

Indirizzo	Tipo di oggetto	Registro di servizio		Dimensione (bit)	Descrizione
		Numero	Nome		
5104h	Bobine	SSTOP (SLOW_STOP)			Impostare un oggetto su “1” o applicare l'istruzione SSTOP arresta il motore in base al DEC, quindi lo porta in modalità di mantenimento.
5105h	Bobine	SHIZ (SLOW_HIZ)			Impostare un oggetto su “1” o applicare l'istruzione SHIZ arresta il motore in base al DEC e poi passa allo stato HiZ.

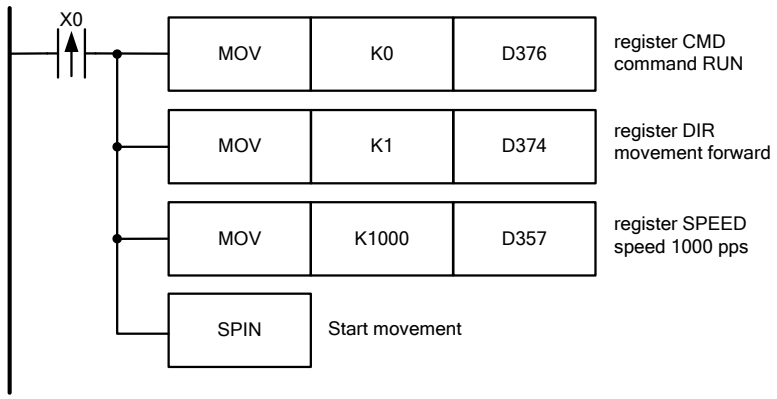
Comando di movimento (registro CMD)

Valore	Gruppo	Nome	Descrizione
0	ESEGUI	RUN	Rotazione in base alla velocità impostata SPEED, accelerazione ACC, decelerazione DEC, direzione DIR.
1	SPOSTA	MOVE	Sposta di un numero di passi specificato TARGET_POS con i parametri di movimento dati SPEED, ACC, DEC e DIR.
2	SPOSTA	GOTO	Spostarsi nella posizione specificata TARGET_POS con i parametri di movimento dati SPEED, ACC, DEC. DIR dipende dalla posizione corrente, il valore fornito non viene preso in considerazione.
3	SPOSTA	GOTO_DIR	Spostarsi nella posizione specificata TARGET_POS con i parametri di movimento dati SPEED, ACC, DEC e DIR.
4	SPOSTA	GOHOME	Spostarsi nella posizione "0" con i parametri di movimento impostati su VELOCITÀ, ACCELERAZIONE, DECELERAZIONE. Il comando è equivalente a GOTO "0".

5	ESEGUI	GOUNTIL_SLOWSTOP	Movimento con una velocità impostata SPEED, accelerazione ACC, direzione DIR fino a quando il sensore SW_INPUT non viene attivato su un fronte di salita con un controllo iniziale del livello di ingresso seguito da decelerazione e arresto secondo un DEC impostato.
6	ESEGUI	GOUNTIL_FRONT_SLOWSTOP	Movimento a velocità impostata SPEED, accelerazione ACC, direzione DIR fino a quando il sensore SW_INPUT non si attiva su un fronte di salita, con la seguente decelerazione e arresto secondo un DEC impostato.
7	ESEGUI	GOUNTIL_HARDSTOP	Movimento con velocità impostata SPEED, accelerazione ACC, direzione DIR fino a quando il sensore SW_INPUT non viene attivato su un fronte di salita con un controllo iniziale del livello di ingresso, quindi passa alla modalità di mantenimento.
8	ESEGUI	GOUNTIL_FRONT_HARDSTOP	Movimento con velocità impostata SPEED, accelerazione ACC, direzione DIR fino a quando il sensore SW_INPUT non viene attivato su un fronte di salita, quindi passa alla modalità di mantenimento.
9	ESEGUI	RELEASE	Movimento con velocità impostata SPEED, accelerazione ACC e direzione DIR fino a quando il sensore SW_INPUT si attiva sul fronte di discesa, con un controllo iniziale del livello di ingresso, e poi passa alla modalità di mantenimento.

10	ESEGUI	FRONT_RELEASE	Movimento con una velocità impostata SPEED, accelerazione ACC, direzione DIR fino a quando il sensore SW_INPUT si attiva sul fronte di discesa e quindi passa alla modalità di mantenimento.

Esempio:



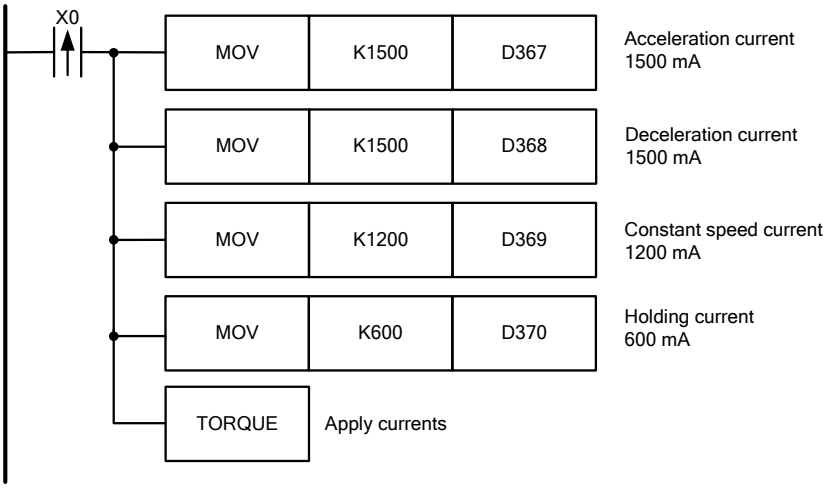
Quando la condizione di ingresso è soddisfatta, il comando di movimento, la direzione e la velocità di rotazione vengono impostati nei registri di servizio.La seguente istruzione SPIN avvia il movimento.

TORQUE		P	Applicare le correnti impostate al motore
5101h	**Bobine**	Scrivere “1” in un oggetto (anche se non è resettato) applica al motore i valori di corrente, che sono impostati nei registri di servizio.Il reset (valore = “0”) viene ignorato.	

Descrizione:

L'applicazione di questa istruzione imposta le correnti di funzionamento del motore indicate nei registri ACC_CUR (D367), DEC_CUR (D368), RUN_CUR (D369) e HOLD_CUR (D370).

Esempio:

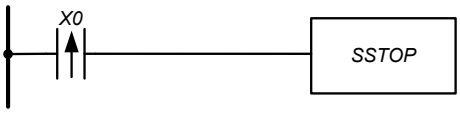


SSTOP		P	Il motore si ferma in base al parametro DEC e poi entra in modalità di mantenimento.
5104h	Bobine	Scrivere “1” in un oggetto (anche se non è resettato) applica la frenatura a motore secondo DEC e poi passa alla modalità di mantenimento, il reset viene ignorato.	

Descrizione:

Applicazione della frenata secondo DEC e successivo passaggio alla modalità di mantenimento. Questa istruzione sovrascrive l'operazione SPIN, ha la stessa priorità di SHIZ, ma può essere sovrascritta dalle istruzioni HSTOP e HHIZ.

Esempio:

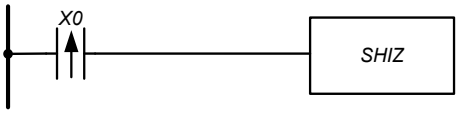


SHIZ		P	Il motore si ferma in base al parametro DEC e poi passa alla modalità HiZ.
5105h	Bobine	Scrivere “1” in un oggetto (anche se non è resettato) applica la frenatura motore secondo DEC e poi passa alla modalità HiZ, il reset viene ignorato.	

Descrizione:

Applicazione della frenata secondo DEC, seguita dalla disattivazione degli avvolgimenti. Questa istruzione sovrascrive l'operazione SPIN, ha la stessa priorità di SSTOP, ma può essere sovrascritta dalle istruzioni HSTOP e HHIZ.

Esempio:

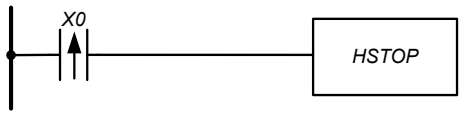


HSTOP		P	Arresta immediatamente il motore e poi passa alla modalità di mantenimento.
5102h	Bobine	Scrivere “1” in un oggetto (anche se non è resettato) arresta immediatamente il motore e poi passa alla modalità di mantenimento, il reset viene ignorato.	

Descrizione:

Arresta immediatamente il motore e poi passa alla modalità di mantenimento. Questa istruzione sovrascrive SPIN, SSTOP, SHIZ e ha la stessa priorità di HHIZ.

Esempio:

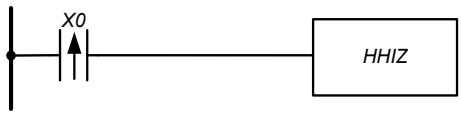


HHIZ		P	Arresta immediatamente il motore e poi passa alla modalità HiZ.
5103h	**Bobine**	Scrivere “1” in un oggetto (anche se non è resettato) arresta immediatamente il motore e poi passa alla modalità HiZ, il reset viene ignorato.	

Descrizione:

Arresta immediatamente il motore e poi passa alla modalità HiZ (il motore è disattivato, l'albero ruota liberamente). Questa istruzione sovrascrive SPIN, SSTOP, SHIZ e ha la stessa priorità di HSTOP.

Esempio:



9. Parametri di comunicazione

Il controller dispone di interfacce USB e RS-485, entrambe con lo stesso accesso ai registri e agli operandi bit. L'interfaccia USB è una porta COM virtuale (VCP), è principalmente destinata alla configurazione del controller e alla registrazione dei programmi utente, pertanto ha parametri di comunicazione fissi: Modbus ASCII, ID 1, 115200 baud, 7, Pari, 1. Le variazioni dei parametri per RS-485 sono indicate nell'Appendice Anella sezione "Parametri di comunicazione dell'interfaccia RS-485". Parametri di comunicazione di fabbrica per RS-485: Modbus RTU, ID 1, 9600 baud, 8, pari, 1.

9.1. Modifica delle impostazioni di comunicazione per RS-485

Impostare i parametri di comunicazione richiesti secondo la sezione "Parametri di comunicazione dell'interfaccia RS-485" dell'Appendice A. Affinché le modifiche abbiano effetto, riavviare il dispositivo. Questo può essere fatto spegnendo e riaccendendo l'alimentazione o impostando l'oggetto Coils 8101h (Reset).

Esempio:

È necessario modificare i parametri di comunicazione con i seguenti: Modbus RTU, ID 100, 128000 baud, 8, Dispari, 1. Tutte le possibili combinazioni di parametri di comunicazione sono presenti nell'Appendice A

Sequenza di azioni:

1) Scrittura del valore 100d nei Holding Registers 8103h – cambia l'indirizzo del dispositivo (ID) a 100.
2) Impostazione di Coils 8100h – selezione del protocollo RTU.
3) Scrittura del valore 128000d nei Holding Registers 8100h – impostazione della velocità di trasferimento dati 128000 baud.
4) Scrittura del valore 1d nei Holding Registers 8102h – selezione del tipo di parità Dispari.
5) Impostazione di Coils 8101h – riavvia il controller.

9.2. Protocollo Modbus

Si consiglia vivamente di leggere le specifiche del protocollo sul sito. http://modbus.org/. Le funzioni di protocollo supportate sono presentate nella tabella seguente:

			Funzione	Codice
Accesso ai dati	Ingressi Discreti	1-bit	Leggi Ingressi Discreti	02(02h)
	Bobine		Lettura Bobine	01(01h)
			Scrivi bobina singola	05(05h)
			Scrittura Multipla Bobine	15(0Fh)
	Registri di Input	16-bit	Leggi Registro di Input	04(04h)
	Registri di Mantenimento		Lettura Registri di Mantenimento	03(03h)
			Scrittura Registro Singolo	06(06h)
			Scrittura di Registri Multipli	16(10h)
			Lettura/Scrittura Registri Multipli	23(17h)
			Registro Scrittura Maschera	22(16h)

I codici di errore del protocollo sono presentati nella tabella seguente:

Codice di errore	Descrizione
01(01h)	*FUNZIONE NON VALIDA* Il codice funzione non può essere elaborato.
02(02h)	*INDIRIZZO DATI NON VALIDO* L'indirizzo del registro specificato nella richiesta non è disponibile.
03(03h)	*VALORE DATI NON VALIDO* Il valore contenuto nel campo dati della richiesta non è valido.
04(04h)	*GUASTO DEL DISPOSITIVO SERVER* Si è verificato un errore irreversibile durante l'esecuzione dell'azione richiesta.

I codici di errore registrati durante l'elaborazione dei pacchetti di protocollo sono presentati nelle tabelle seguenti.

Indirizzo	Tipo	Dimensione	Descrizione
E003h	Registri di Input	16-bit	Codice di errore durante l'elaborazione del frame Modbus.
E003h	Bobine	1-bit	Segnalazione di un errore durante lo scambio tramite protocollo Modbus.

Codice di errore	Descrizione
0001h	Errore di allocazione della memoria.
0002h	Errore di checksum.
0003h	Si è verificato un errore durante la ricezione e l'elaborazione di un pacchetto broadcast.
0004h	Dimensioni del frame non corrispondenti.
0005h	Errore di funzione (0Fh).Non tutti i bit sono stati sovrascritti. .
0006h	Errore di funzione (10h). Non tutti i registri sono stati sovrascritti. .
0007h	Errore di funzione (17h). Non tutti i registri sono stati sovrascritti. .
0008h	Frame perso a causa di un errore DMA.
0009h	Frame perso a causa dello stack di elaborazione dei frame in overflow.

Se il dispositivo si trova all'estremità della linea di comunicazione RS-485, collegare un resistore terminale attivando l'interruttore a levetta accanto al connettore RS-485, come mostrato in Fig. 31.

Fig. 31. Collegamento del resistore terminale.

10. Impostazione dell'orologio in tempo reale

Il controller dispone di un orologio in tempo reale, alimentato da una fonte interna (batteria CR2032), che ne garantisce il funzionamento anche quando l'alimentazione principale è spenta. La stessa batteria viene utilizzata per il funzionamento dei registri non volatili e per le impostazioni di sicurezza dei parametri di comunicazione del controller. L'indicatore **BAT** si accende in caso di assenza o imminente guasto della batteria interna CR2032. L'orologio in tempo reale può essere impostato tramite il programma utente utilizzando l'istruzione TWR o tramite il protocollo Modbus nel seguente ordine:

1) Disabilitare la sovrascrittura automatica dei registri Holding 8110h…8112h resettando i Coil 8110h.
2) Registrazione del valore dell'ora corrente nei registri Holding 8110h, 8111h, 8112h rispettivamente secondi, minuti, ore (fare riferimento alla sezione «Impostazione dell'orologio» in «Appendice A. Registri del controller»).
3) Impostare un nuovo valore di tempo impostando i Coil 8111h.
4) Abilitare la sovrascrittura automatica dei registri Holding 8110h…8112h impostando i Coil 8110h.

11. Un programma utente - caricamento e lettura dal controller

11.1. Procedura di caricamento/scaricamento del programma utente

Il controller dispone di due aree per il download dei programmi: una generica e una speciale.

L'area generica è destinata al caricamento di un programma utente con una lunghezza massima di 59752 righe (l'area è vuota per impostazione predefinita). La lunghezza massima dell'area speciale è di 1926 righe. Quest'area contiene un programma per il controllo della velocità di un motore passo-passo tramite un potenziometro, pulsanti e un encoder. Se necessario, quest'area può essere sovrascritta.

Di seguito è riportato un esempio di programma utente. L'elenco dei registri coinvolti in queste operazioni è riportato in «Appendice A. Registri del controllerAppendice A» nella sezione «Lavorare con la ROM».

Programma utente in formato LD:

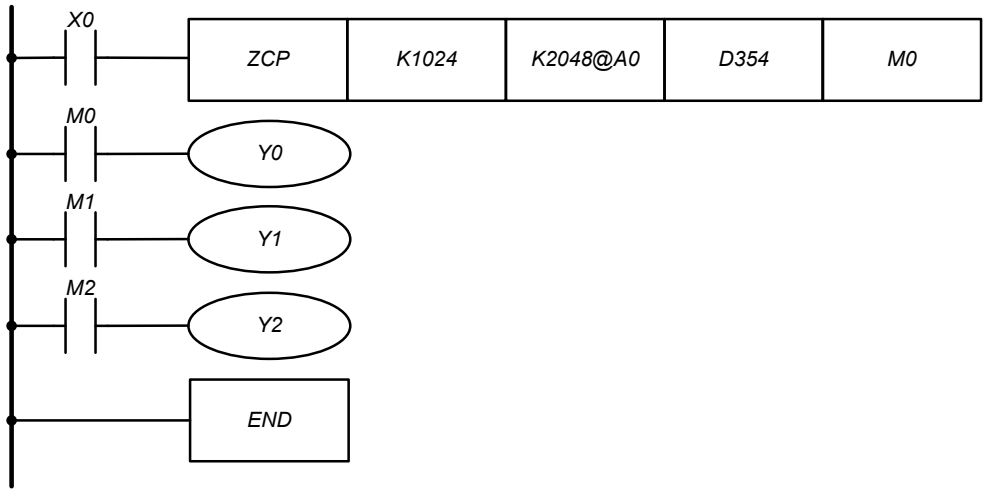


Fig. 32. Programma utente.

Il programma utente convertito in IL:

```
LD      X0                          ;condizione di ingressoper l'operazione di confronto di zona
ZCP     K1024  K2048@A0  D354  M0   ;confronto di zona, determinazione della posizione del; potenziometro SPEED (2)
LD      M0                          ;se il valore nel registro D354 è inferiore a 1024, allora Y0 viene attivato, altrimenti – disattivato
OUT     Y0
LD      M1                          ;se il valore nel registro D354 è maggiore o uguale a 1024 e minore o uguale alla somma di 2048 e il valore di A0, allora Y1 viene attivato, altrimenti – disattivato.
OUT     Y1
LD      M2                          Se il valore nel registro D354 è maggiore della somma di 2048 e il valore di A0, allora Y2 viene attivato,
OUT     Y2
END                                 ;fine del programma
```

Tutti i codici di istruzione supportati dal controller sono presentati in «Appendice B. Elenco delle istruzioni». Utilizzarlo durante l'assemblaggio di un programma utente o utilizzare il software per PC fornito per la programmazione del controller.

1) Assicurarsi che il controller sia in modalità STOP. La modifica del programma utente in modalità RUN è impossibile. Per verificare lo stato RUN/STOP del controller, leggere il valore degli Ingressi Discreti F001h. È resettato in modalità STOP, è impostato in modalità RUN.
2) Per leggere il programma dal controller: verificare che non sia protetto da lettura prima di leggere un programma dal controller. Ci sono due oggetti di protezione dalla lettura nel controller: i Coil F001h (per la protezione di un programma utente) e F002h (per la protezione di un programma di servizio). È impossibile leggere il programma se la protezione è impostata per il programma. Se la protezione non è impostata, andare al passaggio 5 per leggere il programma dal controller.
3) Prima di scrivere un nuovo programma nel controller, è necessario cancellare quello precedente. Impostare i Coil F003h per cancellare il programma utente o i Coil F004h per cancellare un programma di servizio. In questo esempio, il programma principale sta scrivendo, quindi è necessario impostare i Coil F003h.
4) Dopo aver impostato i Coil F003h (o F004h), attendere che gli Ingressi Discreti F000h siano resettati, il che indicherà il completamento della procedura di cancellazione e la disponibilità della ROM per ulteriori operazioni.
5) Dopo aver cancellato il programma precedente, è necessario impostare il tipo di operazione: lettura o scrittura. Per scrivere un nuovo programma, impostare i Coil F005h, per leggere il programma dal controller, resettare i Coil F005h.
6) Selezionare il tipo di programma. Per il programma utente, resettare i Coil F006h, per il programma di servizio, impostare i Coil F006h.
7) Impostare il numero di riga per la scrittura/lettura del programma utilizzando i Holding Register F100h. La numerazione inizia da 0. Per l'operazione di lettura, andare al passaggio 10.Per scrivere un nuovo programma, impostare il suo valore a 0.
8) Riempire il settore di download F300h ...F314 secondo l'esempio seguente:

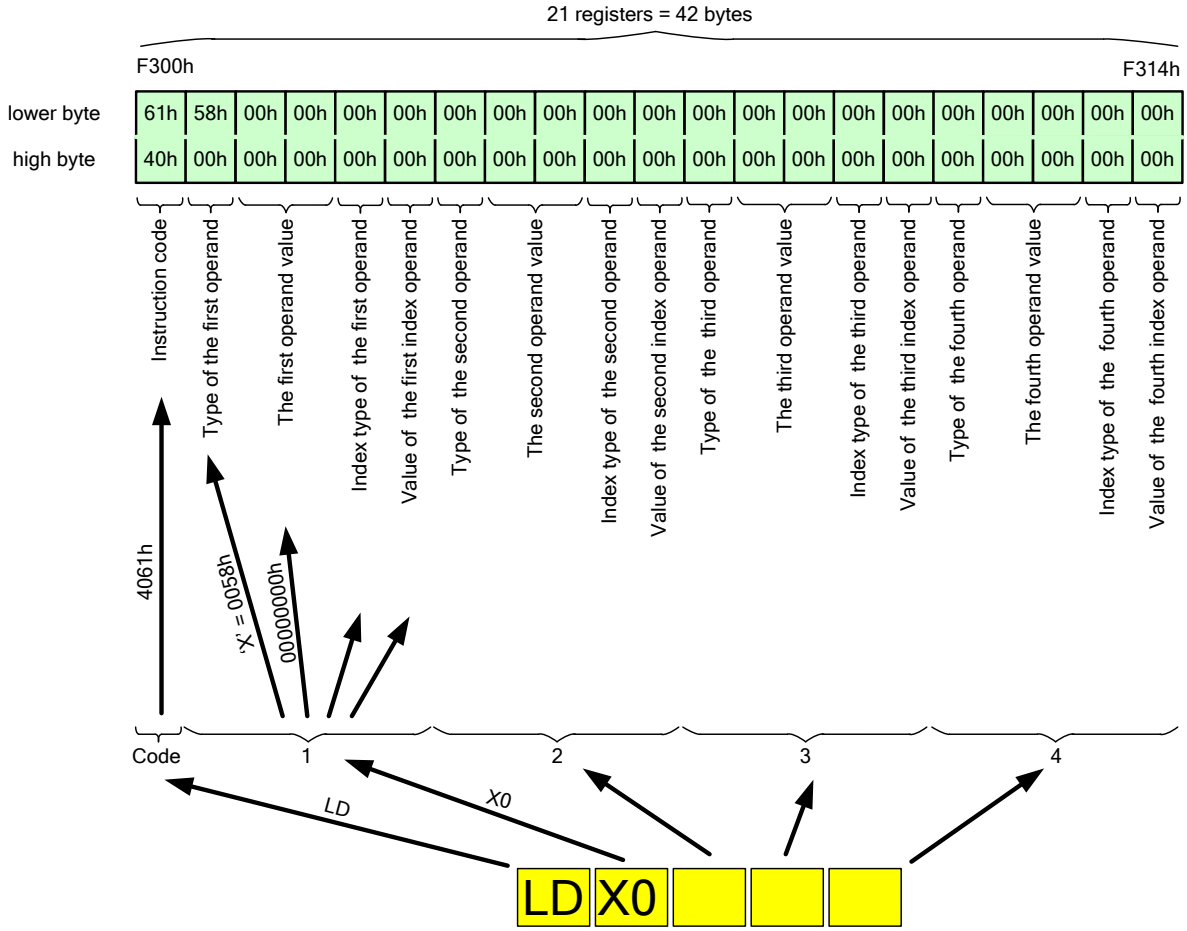


Fig. 33. Proiezione di un'istruzione con operandi nello spazio di indirizzamento del protocollo Modbus.

9) L'impostazione delle Coil F000h avvia l'operazione parametrizzata nelle Coil F005h e F006h. In questo esempio, la scrittura della prima riga nella ROM del controller. Pertanto, ripetendo i passaggi 7-9, spostando il programma verso il basso fino alla fine, incrementando i Holding Register F100h, il programma viene registrato nel controller.
Come esempio, di seguito è riportata la formazione del settore di download dalla seconda riga del programma.

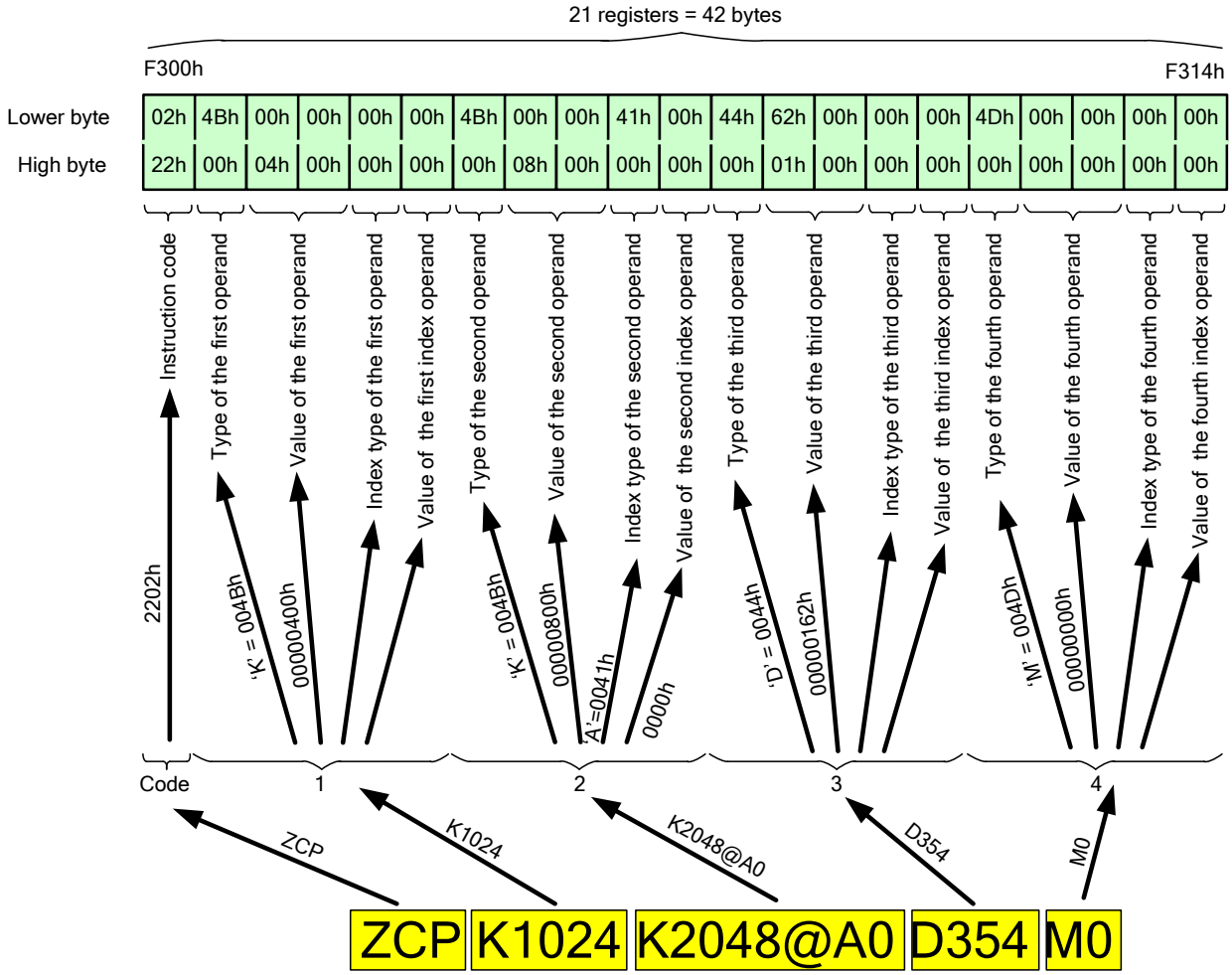


Fig. 34. Proiezione di un'istruzione con operandi nello spazio di indirizzamento del protocollo Modbus.

10) Per leggere il programma, è necessaria l'operazione opposta. La differenza è che il settore di upload ha l'indirizzo Input Register F200h ...F214h e l'operazione parametrizzata viene eseguita prima impostando le Coil F000h, quindi leggendo il settore e infine incrementando il numero di riga fino all'arrivo dell'istruzione END.

I codici tipo operando sono mostrati nella tabella sottostante:

Operando	Codice
K	4Bh
H	48h
F	46h
X	58h
Y	59h
M	4Dh
T	54h
C	43h
A	41h
B	42h
D	44h
P	50h
I	49h

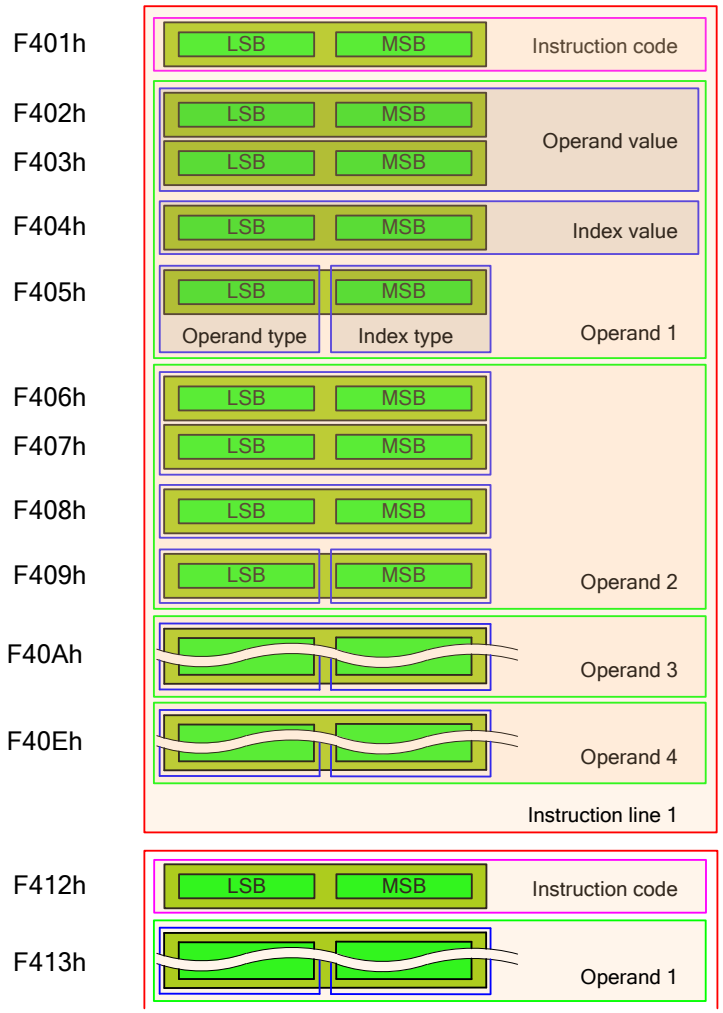
11.2. Caricamento/scaricamento a blocchi di un programma utente

Un programma utente può essere letto e scritto più velocemente se si utilizza il caricamento/scaricamento a blocchi. La procedura di caricamento/scaricamento a blocchi di un programma utente è simile alla sezione «Procedura di caricamento/scaricamento del programma utente», ma con le seguenti differenze:

Per scrivere un programma utente	
Inizio della procedura	Utilizzare i Coil F007h invece di F000h.
Download del settore	Gli indirizzi del settore di download sono i registri Holding F401h…F4FFh, il settore può contenere da 1 fino a 15 righe di comando (istruzioni).Il numero di righe da scrivere è indicato nei registri Holding F400h.
Per leggere un programma utente	
Inizio della procedura	Utilizzare i Coil F007h invece di F000h.

Caricamento del settore	Gli indirizzi del settore di caricamento sono i registri Holding F401h…F4FFh, il settore può contenere da 1 fino a 15 righe di comando (istruzioni).Il numero di righe da leggere è indicato nei registri Holding F400h.Al termine della procedura, F400h visualizzerà il numero effettivo di righe lette. .
Riga di comando	Per ogni riga di comando nel settore di download/upload vengono allocati 34 byte.

La struttura del settore di caricamento/scaricamento a blocchi è la seguente:



. .
. .
. .

11.3. Codici di errore che si verificano durante l'utilizzo della ROM

Indirizzo	Tipo	Dimensione	Descrizione
E002h	Registri di Input	16-bit	Codice di errore quando si lavora con la ROM
E002h	Bobine		Flag di un errore durante l'elaborazione con la ROM.

Codice di errore	Descrizione
0001h	La protezione in lettura per il programma principale non è stata impostata.
0002h	La protezione in lettura per il programma di servizio non è stata impostata.
0003h	Impossibile cancellare il settore del programma principale.
0004h	Impossibile cancellare il settore del programma di servizio.
0005h	Impossibile scrivere l'istruzione nel programma principale.
0006h	Impossibile scrivere l'istruzione nel programma di servizio.

12. Modalità di controllo della velocità

Questa modalità è destinata al controllo della velocità di rotazione di un motore passo-passo utilizzando il potenziometro integrato "SPEED" (2), i pulsanti o un encoder.

Per accedere alla modalità di controllo della velocità, impostare il controller sullo stato STOP, quindi utilizzare il pulsante di selezione della modalità per impostare la modalità SPD. Assemblare e collegare al controller il circuito mostrato in Fig. 35. – Collegamento degli elementi di controllo.

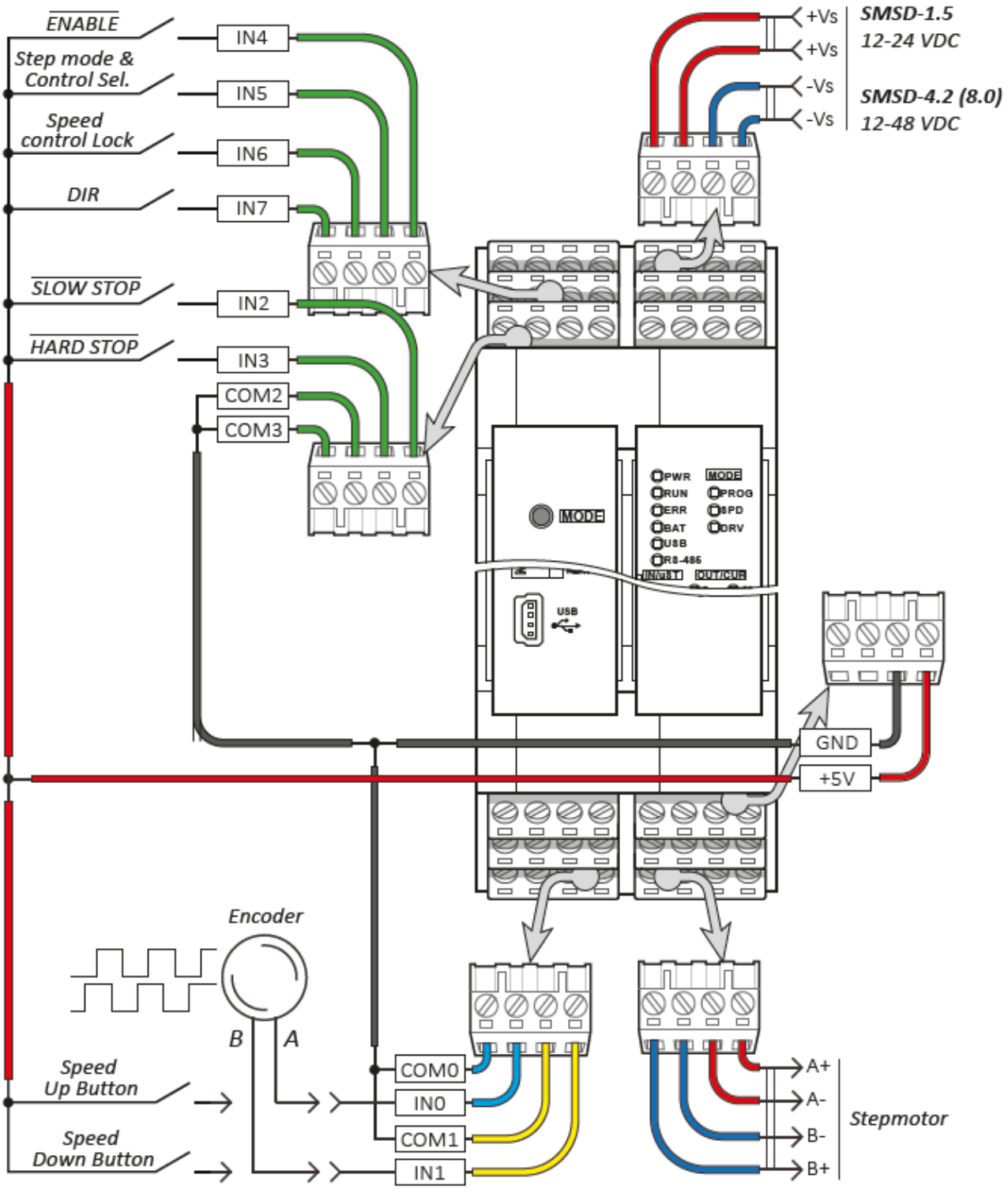


Fig. 35. Collegamento degli elementi di controllo.

Attenzione

Ruotando il controller in modalità **RUN** mentre gli interruttori SLOW STOP e HARD STOP sono chiusi e l'interruttore ENABLE è aperto, si**provocherà la rotazione del motore**. Per evitare rotazioni incontrollate, ruotare il potenziometro “SPEED” nella posizione minima o cambiare la posizione di uno qualsiasi degli interruttori sopra menzionati in quella opposta indicata nel diagramma.

Nello stato **RUN**, selezionare il microstepping richiesto premendo il pulsante corrispondente. Il metodo di controllo della velocità viene selezionato dall'ingresso IN5, maggiori dettagli nella tabella sottostante.

Indicazione LED OUT0…7	**Micropassi**
OUT0	1/1
OUT1	1/2
OUT2	1/4
OUT3	1/8
OUT4	1/16
OUT5	1/32
OUT6	1/128
OUT7	1/256

Indicazione LED OUT10…11	**Sorgente di controllo della velocità**
Entrambi sono disattivati	La velocità viene regolata dal potenziometro «SPEED».
OUT10	La velocità viene regolata dai pulsanti “Aumenta” e “Diminuisci” (IN0 e IN1). L'incremento di variazione della velocità è impostato dal potenziometro «SPEED».
OUT11	La velocità viene regolata da un encoder, collegato agli ingressi IN0 e IN1. L'incremento di velocità per ogni evento dell'encoder viene impostato tramite il potenziometro «SPEED».

Quando il blocco di modifica della velocità è attivato, il controller smette di rispondere ai controlli della velocità. Questa opzione è progettata per prevenire impatti meccanici accidentali sul potenziometro, sull'encoder e sui pulsanti.

DIR– cambia la direzione di rotazione del motore.

ENABLE– controlla l'alimentazione delle fasi del motore passo-passo.

$\overline{\text{HARD STOP}}$– l'apertura del circuito interrompe immediatamente il funzionamento e porta il motore in modalità di mantenimento. La corrente di mantenimento è pari al 50% della corrente di lavoro. Il valore della corrente di lavoro viene impostato tramite il potenziometro (**1**) dal valore minimo al valore massimo per il modello.

$\overline{\text{SLOW STOP}}$– l'apertura del circuito provoca l'arresto del motore secondo la decelerazione impostata dal potenziometro (0) (anche il valore di accelerazione viene impostato dal potenziometro (0)).

Il codice del programma di servizio è fornito in Appendice D. Codice del programma di servizio "Controllo della velocità del motore passo-passo". Il codice può essere modificato per soddisfare requisiti specifici.

13. Modalità di controllo della posizione degli impulsi Step/Dir

Il controller fornisce la modalità di controllo della posizione degli impulsi tramite i segnali di passo degli impulsi STEP (gli ingressi STEP +, STEP-) e il segnale di direzione DIR (gli ingressi DIR +, DIR-). L'ingresso ENABLE + controlla l'alimentazione delle fasi del motore. INVERT_ENABLE + inverte il segnale ENABLE. L'uscita FAULT indica stati di allarme: sovracorrente e surriscaldamento o passi mancanti a causa di questi due motivi (Fig. 2).

Per attivare la modalità di controllo della posizione degli impulsi del controller, prima portarlo nello stato **STOP** e poi utilizzare il pulsante di selezione della modalità per passare alla modalità **DRV**. In questo stato, le fasi del motore sono disattivate. Selezionare il microstepping, la corrente di lavoro e la corrente di mantenimento necessari. (Il passaggio alla modalità viene eseguito un secondo dopo il rilevamento dell'ultimo segnale Step sul fronte di salita dell'impulso).

Il microstepping viene impostato dal potenziometro SPEED, i valori della corrente di lavoro - dal potenziometro (0), la corrente di mantenimento - dal potenziometro (1). I parametri impostati vengono visualizzati sul pannello LED, maggiori dettagli nelle tabelle sottostanti:

OUT0	OUT1	OUT2	OUT3	Corrente di funzionamento		
OUT4	OUT5	OUT6	OUT7	Corrente di mantenimento		
				RS 434540	RS 434542	RS 434543
•				150 мА	1000 мА	2800 мА
	•			245 мА	1285 мА	3310 мА
•	•			340 мА	1570 мА	3830 мА
		•		440 мА	1860 мА	4340 мА
•		•		535 мА	2140 мА	4860 мА
	•	•		630 мА	2430 мА	5370 мА
•	•	•		730 мА	2710 мА	5885 мА
			•	825 мА	3000 мА	6400 мА
•			•	920 мА	3285 мА	6910 мА
	•		•	1020 мА	3570 мА	7430 мА
•	•		•	1115 мА	3860 мА	7940 мА
		•	•	1210 мА	4140 мА	8460 мА
•		•	•	1310 мА	4430 мА	8970 мА
	•	•	•	1400 мА	4710 мА	9485 мА
•	•	•	•	1500 мА	5000 мА	10000 мА

Indicazione LED IN0…7	Micropassi
IN0	1/1
IN1	1/2
IN2	1/4
IN3	1/8
IN4	1/16
IN5	1/32
IN6	1/128
IN7	1/256

Quando il controller è nello stato **RUN**, i parametri di cui sopra sono fissi e vengono salvati dopo lo spegnimento. Utilizzare gli ingressi e le uscite del controller secondo la tabella di assegnazione dei pin (Fig. 2).

14. Modalità di controllo del programma utente

Il controller fornisce la modalità di controllo secondo un programma utente ed è controllato dai comandi Modbus. Il controller indica questa modalità di controllo tramite l'indicatore LED **PROG**. Un programma utente può essere inviato alla memoria del controller quando il controller è nello stato **STOP**. Dopo il passaggio allo stato **RUN**, il controller inizia a eseguire il programma utente. È anche possibile controllare lo stato del controller, del programma utente, delle uscite fisiche, del driver del motore passo-passo e monitorare lo stato degli ingressi fisici tramite l'interfaccia RS-485 utilizzando il protocollo Modbus.

Esempi di programmi utente che dimostrano la funzionalità di base del controller sono descritti nell'Appendice C. Esempi di programmi utente.

Appendice A. Registri del controller

Indirizzo	Tipo	Dimensione	Descrizione
Parametri di comunicazione dell'interfaccia RS-485			
0x8100	Bobine	-	Selezione del protocollo di comunicazione Ripristina — Modbus ASCII. Impostato — Modbus RTU. Le modifiche hanno effetto dopo il riavvio del controller.
0x8100	Registri di Mantenimento	32-bit	Velocità di trasmissione. Valori consentiti: 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 57600, 115200, 128000, 256000. Le modifiche hanno effetto dopo il riavvio del controller. .
0x8102	Registri di Mantenimento	16-bit	Parità: 0 – NESSUNO 1 – DISPARI 2 – PARI
• 1			*Sono disponibili i seguenti parametri di trasferimento dati. :* *Modbus RTU:* • *8-bit / PARI / 1 STOP* • *8-bit / DISPARI / 1 STOP* • *8-bit / NESSUNO / 2 STOP* *Modbus ASCII:* - *7 bit / PARI / 1 STOP* - *7 bit / DISPARI / 1 STOP* *I parametri del bit di stop vengono impostati automaticamente.*
0x8103	Registri di Mantenimento	16-bit	ID del controller (indirizzo del dispositivo). Valori consentiti: 1.. .247.
Impostazione dell'orologio			
0x8110	Registri di Mantenimento	16-bit	Secondi. Valori consentiti: 0.. .59.
0x8111	Registri di Mantenimento	16-bit	Minuti. Valori validi: 0.. .59.
0x8112	Registri di Mantenimento	16-bit	Ore. Valori validi: 0.. .23.

Indirizzo	Tipo	Dimensione	Descrizione
0x8110	Bobine	-	Aggiornamento automatico del registro. Imposta — nei registri 0x8110 - 0x8112 il valore effettivo del tempo. Reset — l'aggiornamento automatico dei dati è disabilitato, la registrazione dei valori utente è consentita.
0x8111	Bobine	-	L'impostazione dell'oggetto imposta l'ora dai registri 0x8110 - 0x8112. È consentito riattivare l'aggiornamento automatico dopo aver impostato l'oggetto.
Impostazione data			
0x8113	Patrimonio Registri	16-bit	Giorno. Valori validi: 1…31
0x8114	Patrimonio Registri	16-bit	Mese Valori validi: 1…12
0x8115	Patrimonio Registri	16-bit	Anno Valori validi: 21…99
		L'impostazione della data è simile all'impostazione dell'ora con un reset preliminare delle Bobine 8110h (fare riferimento alla sezione 10).	
0x8112	Bobine	-	L'impostazione dell'oggetto imposta la data dai registri 0x8110 - 0x8112. È consentito riattivare l'aggiornamento automatico dopo aver impostato l'oggetto.
Ulteriori			
0x8101	Bobine	-	L'impostazione dell'oggetto causa un riavvio del controller.
0xF001	Registri di Mantenimento	16-bit	Modalità operative del controller: programma utente (PROG), programma di servizio (SPD), modalità driver (DRV). La modifica della modalità operativa del controller è possibile solo nello stato di STOP. 0 - Programma utente. 1 - Programma di servizio. 2 - Modalità driver.

Indirizzo	Tipo	Dimensione	Descrizione
	Lavorare con la ROM		
0xF001	Ingressi Discreti	-	Stato dell'interruttore a levetta RUN/STOP. Le operazioni ROM non sono possibili quando il controller è nello stato RUN. Reset — stato di STOP. Imposta lo stato —RUN.
0xF000	Ingressi Discreti	-	Indicazione dello stato della ROM. Reset — La ROM è pronta per il funzionamento. Impostato — La ROM è occupata.
0xF000	Bobine	-	Oggetto di controllo per la lettura/scrittura riga per riga di un programma utente. L'impostazione dell'oggetto avvia l'operazione parametrizzata negli oggetti 0xF005 e 0xF006.
0xF001	Bobine	-	Protezione in lettura del programma principale (utente). Impostato — non protetto. Ripristina — la protezione in lettura è impostata. Il tentativo di impostare l'oggetto causa la cancellazione del programma principale.
0xF002	Bobine	-	Protezione in lettura del programma di servizio. Impostato — non protetto. Ripristina — la protezione in lettura è impostata. Il tentativo di impostare l'oggetto causa la cancellazione del programma di servizio.
0xF003	Bobine	-	Cancellazione del programma principale. L'impostazione dell'oggetto avvia la procedura di cancellazione del programma principale. L'azzeramento dell'oggetto viene ignorato.
0xF004	Bobine	-	Cancellazione del programma di servizio. L'impostazione dell'oggetto avvia la procedura di cancellazione del programma di servizio. L'azzeramento dell'oggetto viene ignorato.
0xF005	Bobine	-	Selezione del tipo di operazione. Reset — lettura. Imposta — scrivi.

Indirizzo	Tipo	Dimensione	Descrizione
0xF006	Bobine	-	Selezione programma (principale/servizio) Azzera — principale. Imposta — servizio.
0xF007	Bobine	-	Oggetto di controllo per la lettura/scrittura a blocchi di un programma utente (fare riferimento alla sezione 11.2). L'impostazione dell'oggetto avvia l'operazione parametrizzata negli oggetti 0xF005 e 0xF006.
0xF100	Registri di Mantenimento	16-bit	Il numero di riga nel programma da leggere o sovrascrivere (0 ... 59753 - per il programma principale, 0 ... 1927 - per il servizio).
Lettura del settore ROM riga per riga			
0xF200	Registri di Input	16-bit	Codice istruzioni
0xF201	Registri di Input	16-bit	Tipo del primo operando
0xF202	Registri di Input	32-bit	Valore del primo operando
0xF204	Registri di Input	16-bit	Tipo di indice del primo operando
0xF205	Registri di Input	16-bit	Valore dell'operando del primo indice
0xF206	Registri di Input	16-bit	Tipo del secondo operando
0xF207	Registri di Input	32-bit	Valore del secondo operando
0xF209	Registri di Input	16-bit	Tipo di indice del secondo operando
0xF20A	Registri di Input	16-bit	Valore del secondo operando di indice
0xF20B	Registri di Input	16-bit	Tipo del terzo operando
0xF20C	Registri di Input	32-bit	Valore del terzo operando
0xF20E	Registri di Input	16-bit	Tipo di indice del terzo operando
0xF20F	Registri di Input	16-bit	Valore del terzo operando di indice
0xF210	Registri di Input	16-bit	Tipo del quarto operando
0xF211	Registri di Input	32-bit	Valore del quarto operando
0xF213	Registri di Input	16-bit	Tipo di indice del quarto operando
0xF214	Registri di Input	16-bit	Valore del quarto operando di indice
Settore di scrittura ROM riga per riga			
0xF300	Registri di Mantenimento	16-bit	Codice istruzioni
0xF301	Registri di Mantenimento	16-bit	Tipo del primo operando

Indirizzo	Tipo	Dimensione	Descrizione
0xF302	Registri di Mantenimento	32-bit	Valore del primo operando
0xF304	Registri di Mantenimento	16-bit	Tipo di indice del primo operando
0xF305	Registri di Mantenimento	16-bit	Valore del primo operando di indice
0xF306	Registri di Mantenimento	16-bit	Tipo del secondo operando
0xF307	Registri di Mantenimento	32-bit	Valore del secondo operando
0xF309	Registri di Mantenimento	16-bit	Tipo di indice del secondo operando
0xF30A	Registri di Mantenimento	16-bit	Valore del secondo operando di indice
0xF30B	Registri di Mantenimento	16-bit	Tipo del terzo operando
0xF30C	Registri di Mantenimento	32-bit	Valore del terzo operando
0xF30E	Registri di Mantenimento	16-bit	Tipo di indice del terzo operando
0xF30F	Registri di Mantenimento	16-bit	Valore del terzo operando di indice
0xF310	Registri di Mantenimento	16-bit	Tipo del quarto operando
0xF311	Registri di Mantenimento	32-bit	Valore del quarto operando
0xF313	Registri di Mantenimento	16-bit	Tipo di indice del quarto operando
0xF314	Registri di Mantenimento	16-bit	Valore del quarto operando di indice
Settore di lettura/scrittura per il caricamento/scaricamento a blocchi di un programma utente ***(15 righe di comando, 17 registri per riga)***			
Riga 1 del settore di lettura/scrittura			
0xF401	Registri di Mantenimento	16-bit	Codice istruzioni della riga 1 del settore di lettura/scrittura
Operando 1 della riga 1 del settore di lettura/scrittura			
0xF402	Registri di Mantenimento	32-bit	Valore dell'operando 1 della riga 1 del settore di lettura/scrittura

Indirizzo	Tipo	Dimensione	Descrizione
0xF404	Registri di Mantenimento	16-bit	Valore dell'indice dell'operando 1 della riga 1 del settore di lettura/scrittura
0xF405	Registri di Mantenimento	16-bit	8 bit LSB – Tipo operando 1 – riga 1 del settore di lettura/scrittura 8 bit MSB – Tipo di indice dell'operando 1 del settore di lettura/scrittura
. . .	. . .	. . .	. . .
Operando 4 della riga 1 del settore di lettura/scrittura			
0xF40E	Registri di Mantenimento	32-bit	Valore dell'operando 4 della riga 1 del settore di lettura/scrittura
0xF404	Registri di Mantenimento	16-bit	Valore dell'operando 4 indice della riga 1 del settore di lettura/scrittura
0xF405	Registri di Mantenimento	16-bit	8 bit LSB – Tipo operando 4 – riga 1 del settore di lettura/scrittura 8 bit MSB – Tipo di indice dell'operando 4 del settore di lettura/scrittura
. . .	. . .	. . .	. . .
Riga 15 del settore di lettura/scrittura			
0xF4EF	Registri di Mantenimento	16-bit	Codice istruzioni della riga 15 del settore di lettura/scrittura
Operando 1 della riga 15 del settore di lettura/scrittura			
0xF4F0	Registri di Mantenimento	32-bit	Valore dell'operando 1 della riga 15 del settore di lettura/scrittura
0xF4F2	Registri di Mantenimento	16-bit	Valore dell'indice dell'operando 1 della riga 15 del settore di lettura/scrittura
0xF4F3	Registri di Mantenimento	16-bit	8 bit LSB – Tipo operando 1 – riga 15 del settore di lettura/scrittura 8 bit MSB – Tipo di indice dell'operando 1 del settore di lettura/scrittura
. . .	. . .	. . .	. . .
Operando 4 della riga 15 del settore di lettura/scrittura			
0xF4FC	Registri di Mantenimento	32-bit	Valore dell'operando 4 della riga 15 del settore di lettura/scrittura
0xF4FE	Registri di Mantenimento	16-bit	Valore dell'operando 4 indice della riga 15 del settore di lettura/scrittura
0xF4FF	Registri di Mantenimento	16-bit	8 bit LSB – Tipo operando 4 – riga 15 del settore di lettura/scrittura 8 bit MSB – Tipo di indice dell'operando 4 del settore di lettura/scrittura.

Indirizzo	Tipo	Dimensione	Descrizione
	Errori		
0xE000	Bobine	-	L'impostazione dell'oggetto reimposta tutti i tipi di errori validi per lo stato corrente del controller.
0xE000	Ingressi Discreti	-	Errore generale. Viene sempre impostato quando compare almeno uno dei tipi di errore da 0xE001 a 0xE004.
0xE001	Ingressi Discreti	-	Lo stato attivo dell'oggetto indica una batteria CR2032 scarica all'interno del controller. È necessaria la sostituzione.
0xE002	Ingressi Discreti	-	Lo stato attivo dell'oggetto indica un errore durante l'operazione ROM. Il codice di errore è specificato nei Registri di Input 0xE002.
0xE003	Ingressi Discreti	-	Lo stato impostato dell'oggetto indica un errore durante il processo di scambio utilizzando il protocollo Modbus. Il codice di errore è specificato nei Registri di Input 0xE003.
0xE004	Ingressi Discreti	-	Lo stato attivo dell'oggetto indica un errore durante l'esecuzione del programma utente. Il codice di errore è specificato nei Registri di Input 0xE004. La riga di programma che ha causato l'errore è indicata in 0xE084.
0xE002	Registri di Input	16-bit	Codice di errore di funzionamento della ROM.
0xE003	Registri di Input	16-bit	Codice di errore del protocollo Modbus.
0xE004	Registri di Input	16-bit	Codice di errore del programma utente.
0xE084	Registri di Input	16-bit	Il numero di riga che ha causato l'errore nel programma utente (la numerazione inizia da 0, vedere la descrizione delle Bobine 0xF100 sopra).
	Accesso agli operandi del programma		
	Uscite discrete		
0x1000	Ingressi Discreti	-	Uscita discreta Y0
0x1001	Ingressi Discreti	-	Uscita discreta Y1
. . .	. . .	. . .	. . .
0x107F	Ingressi Discreti	-	Uscita discreta Y177

Indirizzo	Tipo	Dimensione	Descrizione
Stato degli ingressi fisici discreti			
0x2000	Ingressi Discreti	-	Ingresso discreto X0
. . .	. . .	. . .	. . .
0x2007	Ingressi Discreti	-	Ingresso discreto X7
Ingressi discreti			
0x2008	Bobine	-	Ingresso discreto X10
0x2009	Bobine	-	Ingresso discreto X11
. . .	. . .	. . .	. . .
0x207F	Bobine	-	Ingresso discreto X177
Registri dati di uso generale D192.. .D255			
0x3000	Registri di Input	16-bit	Registro D192
0x3001	Registri di Input	16-bit	Registro D193
. . .	. . .	. . .	. . .
0x303F	Registri di Input	16-bit	Registro D255
Registri dati di uso generale D256.. .D319			
0x4000	Registri di Mantenimento	16-bit	Registro D256
0x4001	Registri di Mantenimento	16-bit	Registro D257
. . .	. . .	. . .	. . .
0x403F	Registri di Mantenimento	16-bit	Registro D319
Registri dati non volatili D320.. .D327			
0x3100	Registri di Input	16-bit	Registro D320
. . .	. . .	. . .	. . .
0x3107	Registri di Input	16-bit	Registro D327
Registri dati non volatili D328.. .335			
0x4100	Registri di Mantenimento	16-bit	Registro D328

Indirizzo	Tipo	Dimensione	Descrizione
. . .	. . .	. . .	. . .
0x4107	Registri di Mantenimento	16-bit	Registro D335
Contatori hardware			
0x4200	Registri di Mantenimento	32-bit	Contatore C64
0x4202	Registri di Mantenimento	32-bit	Contatore C65
Convertitori analogico-digitali			
0x3200	Registri di Input	16-bit	Registro D352, dati dal potenziometro «0»
0x3201	Registri di Input	16-bit	Registro D353, dati dal potenziometro «1»
0x3202	Registri di Input	16-bit	Registro D354, dati dal potenziometro «SPEED»
Versioni hardware e software			
0x8001	Registri di Input	16-bit	Versione hardware principale
0x8002	Registri di Input	16-bit	Versione hardware minore
0x8003	Registri di Input	16-bit	Versione principale del software
0x8004	Registri di Input	16-bit	Versione minore del software
0x8005	Registri di Input	16-bit	Versione principale del bootloader
0x8006	Registri di Input	16-bit	Versione minore del bootloader
Controllo motore passo-passo			
0x5000	Registri di Mantenimento	32-bit	Registro D357 – SPEED.
0x5002	Registri di Mantenimento	32-bit	Registro D359 – MIN_SPEED.
0x5004	Registri di Mantenimento	16-bit	Registro D361 – ACC.
0x5005	Registri di Mantenimento	16-bit	Registro D362 – DEC.
0x5006	Registri di Mantenimento	32-bit	Registro D363 – ABS.
0x5008	Registri di Input	16-bit	Registro D365 – U_POS.
0x5009	Registri di Mantenimento	16-bit	Registro D366 – U_STEP.

Indirizzo	Tipo	Dimensione	Descrizione
0x500A	Registri di Mantenimento	16-bit	Registro D374 – DIR.
0x500A	Bobine	-	Registro D374 – DIR.
0x500B	Registri di Mantenimento	32-bit	Registro D377 – FS_SPD_THR.
0x500D	Registri di Mantenimento	16-bit	Registro D379 – FS_SW_EN.
0x500D	Bobine	-	Registro D379 – FS_SW_EN.
0x500E	Registri di Mantenimento	32-bit	Registro D372 – TARGET_POS.
0x5010	Registri di Mantenimento	16-bit	Registro D376 – CMD.
0x5011	Registri di Mantenimento	16-bit	Registro D375 – SW_INPUT.
0x5012	Registri di Mantenimento	16-bit	Registro D367 – ACC_CUR.
0x5013	Registri di Mantenimento	16-bit	Registro D368 – DEC_CUR.
0x5014	Registri di Mantenimento	16-bit	Registro D369 – RUN_CUR.
0x5015	Registri di Mantenimento	16-bit	Registro D370 – HOLD_CUR.
0x5016	Registri di Mantenimento	16-bit	Registro D382 – CMIN_SPD_EN.
0x5017	Registri di Mantenimento	16-bit	Registro D380 – ERROR_SET_HIZ.
0x5017	Bobine	-	TERMAL_ERROR_SET_HIZ
0x5018	Bobine	-	SOFTWARE_ERROR_SET_HIZ
0x5019	Bobine	-	CMD_ERROR_SET_HIZ
0x501A	Bobine	-	DATA_ERROR_SET_HIZ
0x5027	Registri di Mantenimento	16-bit	Registro D381 – ERROR_CODE.
0x5027	Bobine	-	TERMAL_ERROR_OVER_CURRENT
0x5028	Bobine	-	SOFT_ERROR
0x5029	Bobine	-	CMD_ERROR
0x502A	Bobine	-	DATA_ERROR
0x502F	Bobine	-	OVLO/UVLO_INTERNAL_PROTECTION_ERROR (RS 434542 and RS 434543)

Indirizzo	Tipo	Dimensione	Descrizione
0x5030	Bobine	-	VS_OUT_OF_RANGE_ERROR
0x5037	Registri di Input	16-bit	Registro D371 – STATO_MOTORE.
0x5037	Ingressi Discreti	-	HIZ
0x5038	Ingressi Discreti	-	STOP
0x5039	Ingressi Discreti	-	ACCELERATING
0x503A	Ingressi Discreti	-	DECELERATING
0x503B	Ingressi Discreti	-	STEADY
0x503C	Ingressi Discreti	-	BUSY_MOVE
0x503D	Ingressi Discreti	-	BUSY_RUN
0x5048	Patrimonio Registri	32-bit	Registro D385 – EMERGENCY_DEC.
0x5100	Bobine	-	Istruzione SPIN – APPLY_CMD.
0x5101	Bobine	-	Istruzione TORQUE – APPLY_CURRENT.
0x5102	Bobine	-	Istruzione HSTOP – HARD_STOP.
0x5103	Bobine	-	Istruzione HHIZ – HARD_HIZ.
0x5104	Bobine	-	Istruzione SSTOP – SLOW_STOP.
0x5105	Bobine	-	Istruzione SHIZ – SLOW_HIZ.

Appendice B. Elenco delle istruzioni

Istruzione		Descrizione
API	Codice	
Istruzioni di base		
LD	0x4061	Contatto normalmente aperto
LDI	0x4001	Contatto normalmente chiuso
AND	0x4065	Collegamento in serie - contatto normalmente aperto (AND logico)
ANI	0x4005	Collegamento in serie - contatto normalmente chiuso (NAND logico)
OR	0x4066	Collegamento in parallelo – contatto normalmente aperto (OR logico)
ORI	0x4046	Collegamento in parallelo – contatto normalmente chiuso (NOR logico)
LDP	0x4821	Inizio dell'espressione logica con polling del fronte di salita (impulso)
LDF	0x4841	Inizio di un'espressione logica con polling sul fronte di discesa (impulso)
ANDP	0x4825	«AND» con un polling (impulso) a fronte di salita
ANDF	0x4845	«AND» con un impulso (polling) sul fronte di discesa
ORP	0x4806	«OR» con polling (impulso) sul fronte di salita
ORF	0x4826	«OR» con polling sul fronte di discesa (impulso)
TMR	0x2014	Timer (16 bit)
CNT	0x2015	Contatore (16-bit)
DCNT	0x3015	Contatore (32-bit)
INV	0x4016	Inversione - sostituzione del risultato delle connessioni logiche con l'opposto
ANB	0x4007	Blocco «AND»: collegamento in serie dei blocchi
ORB	0x4008	Blocco «OR»: collegamento in parallelo dei blocchi
MPS	0x4009	Offset verso il basso nello stack
MRD	0x402A	Leggere il valore dallo stack
MPP	0x400A	Uscita dallo stack
SET	0x2024	Attivazione dell'uscita latchata (impostazione del valore logico “1”)
RST	0x2004	Ripristino dello stato dell'operando
OUT	0x2002	Bobina di uscita - assegnazione all'uscita del risultato di un'espressione logica

FEND	0x6003	Fine del programma principale
NOP	0x8011	Riga vuota nel programma
P	0x6051	Indirizzamento di un punto di salto in un programma o sottoprogramma
I	0x6031	Indirizzamento di un punto di interruzione
END	0x6023	Fine del programma
Istruzioni per cicli, transizioni, sottoprogrammi		
CJ	0x200D	Salto condizionato - vai alla riga di programma specificata
CJP	0x280D	Salto condizionato - vai alla riga di programma specificata con polling del fronte di salita (impulso)
CALL	0x200E	Chiamata del sottoprogramma
CALLP	0x280E	Chiamata del sottoprogramma con polling del fronte di salita (impulso)
SRET	0x600F	Fine del sottoprogramma
FOR	0x400B	Inizio di un ciclo FOR-NEXT
NEXT	0x400C	Fine di un ciclo FOR-NEXT
Interruzioni		
IRET	0x6010	Fine del gestore di interrupt
EI	0x8012	Abilitazione delle interruzioni globali
DI	0x8013	Disabilitazione degli interrupt globali
Trasferimento e confronto dati		
CMP	0x2201	Confronto di dati numerici
CMPP	0x2A01	Confronto di dati numerici con polling a fronte di salita (impulso)
DCMP	0x3201	Confronto di dati numerici, istruzione a 32 bit
DCMPP	0x3A01	Confronto di dati numerici, istruzione a 32 bit con polling sul fronte di salita (impulso)
ZCP	0x2202	Confronto di zona dei dati numerici
ZCPP	0x2A02	Confronto di zona di dati numerici con polling a fronte di salita (impulso)
DZCP	0x3202	Confronto di zona di dati numerici, istruzione a 32 bit
DZCPP	0x3A02	Confronto di zona di dati numerici, istruzione a 32 bit con polling (impulso) sul fronte di salita
MOV	0x2018	Trasferimento dati

MOVP	0x2818	Trasferimento dati con polling (impulso) sul fronte di salita
DMOV	0x3018	Trasferimento dati, istruzione a 32 bit
DMOVP	0x3818	Trasferimento dati, istruzione a 32 bit con polling (impulso) sul fronte di salita
BMOV	0x2038	Trasferimento dati a blocchi
BMOVP	0x2838	Trasferimento dati a blocchi con polling (impulso) sul fronte di salita
DBMOV	0x3038	Trasferimento dati a blocchi, istruzione a 32 bit
DBMOVP	0x3838	Trasferimento dati a blocchi, istruzione a 32 bit con polling (impulso) sul fronte di salita
FMOV	0x2058	Trasferimento di dati a indirizzi multipli
FMOVP	0x2858	Trasferimento dati a indirizzi multipli con polling (impulso) sul fronte di salita
DFMOV	0x3058	Trasferimento dati a indirizzi multipli, istruzione a 32 bit
DFMOVP	0x3858	Trasferimento dati a indirizzi multipli, istruzione a 32 bit con polling (impulso) sul fronte di salita
XCH	0x220A	Scambio dati
XCHP	0x2A0A	Scambio dati con polling (impulso) sul fronte di salita
DXCH	0x320A	Scambio dati, istruzione a 32 bit
DXCHP	0x3A0A	Scambio dati, istruzione a 32 bit con polling (impulso) sul fronte di salita
Operazioni aritmetiche (numeri interi)		
ADD	0x2208	Addizione di dati numerici
ADDP	0x2A08	Addizione di dati numerici con polling (impulso) sul fronte di salita
DADD	0x3208	Addizione di dati numerici, istruzione a 32 bit
DADDP	0x3A08	Addizione di dati numerici, istruzione a 32 bit con polling (impulso) sul fronte di salita
SUB	0x2228	Sottrazione di dati numerici
SUBP	0x2A28	Sottrazione di dati numerici con polling (impulso) sul fronte di salita
DSUB	0x3228	Sottrazione di dati numerici, istruzione a 32 bit
DSUBP	0x3A28	Sottrazione di dati numerici, istruzione a 32 bit con polling (impulso) sul fronte di salita
MUL	0x2248	Moltiplicazione di dati numerici
MULP	0x2A48	Moltiplicazione di dati numerici con polling (impulso) sul fronte di salita

DMUL	0x3248	Moltiplicazione di dati numerici, istruzione a 32 bit
DMULP	0x3A48	Moltiplicazione di dati numerici, istruzione a 32 bit con polling (impulso) sul fronte di salita
DIV	0x2268	Divisione di dati numerici
DIVP	0x2A68	Divisione dei dati numerici con polling (impulso) a fronte di salita
DDIV	0x3268	Divisione di dati numerici, istruzione a 32 bit
DDIVP	0x3A68	Divisione di dati numerici, istruzione a 32 bit con polling (impulso) sul fronte di salita
MOD	0x22E8	Resto della divisione
MODP	0x2AE8	Resto della divisione con polling (impulso) sul fronte di salita
DMOD	0x32E8	Resto della divisione, istruzione a 32 bit
DMODP	0x3AE8	Resto della divisione, istruzione a 32 bit con polling (impulso) sul fronte di salita
INC	0x2037	Incrementa dati numerici (aumenta di 1)
INCP	0x2837	Incrementa i dati numerici (aumenta di 1) con polling del fronte di salita (impulso)
DINC	0x3037	Incrementa dati numerici (aumenta di 1), istruzione a 32 bit
DINCP	0x3837	Incrementa dati numerici (aumenta di 1), istruzione a 32 bit con polling del fronte di salita (impulso)
DEC	0x2017	Decrementa i dati numerici (diminuisci di 1)
DECP	0x2817	Decrementa i dati numerici (diminuisci di 1) con polling del fronte di salita (impulso)
DDEC	0x3017	Decrementa dati numerici (diminuisci di 1), istruzione a 32 bit
DDECP	0x3817	Decrementa dati numerici (diminuisci di 1), istruzione a 32 bit con polling sul fronte di salita (impulso)
WAND	0x2288	Moltiplicazione logica di dati numerici (operazione "AND")
WANDP	0x2A88	Moltiplicazione logica di dati numerici (operazione "AND") con polling del fronte di salita (impulso)
DAND	0x3288	Moltiplicazione logica di dati numerici (operazione "AND"), istruzione a 32 bit
DANDP	0x3A88	Moltiplicazione logica di dati numerici (operazione "AND"), istruzione a 32 bit con polling sul fronte di salita (impulso)
WOR	0x22A8	Addizione logica di dati numerici (operazione OR)
WORP	0x2AA8	Addizione logica di dati numerici (operazione OR) con polling del fronte di salita (impulso)

DOR	0x32A8	Addizione logica di dati numerici (operazione OR), istruzione a 32 bit
DORP	0x3AA8	Addizione logica di dati numerici (operazione OR), istruzione a 32 bit con polling sul fronte di salita (impulso)
WXOR	0x22C8	Operazione logica "OR esclusivo"
WXORP	0x2AC8	Operazione logica "OR esclusivo" con polling (impulso) del fronte di salita
DXOR	0x32C8	Operazione logica "OR esclusivo", istruzione a 32 bit
DXORP	0x3AC8	Operazione logica "OR esclusivo", istruzione a 32 bit con polling (impulso) sul fronte di salita
NEG	0x2209	Negazione logica
NEGP	0x2A09	Negazione logica con polling del fronte di salita (impulso)
DNEG	0x3209	Negazione logica, istruzione a 32 bit
DNEGP	0x3A09	Negazione logica, istruzione a 32 bit con polling (impulso) sul fronte di salita
ABS	0x2229	Valore assoluto
ABSP	0x2A29	Valore assoluto con polling del fronte di salita (impulso)
DABS	0x3229	Valore assoluto, istruzione a 32 bit
DABSP	0x3A29	Valore assoluto, istruzione a 32 bit con polling sul fronte di salita (impulso)
SQR	0x2215	Calcolo della radice quadrata
SQRP	0x2A15	Calcolo della radice quadrata con polling del fronte di salita (impulso)
DSQR	0x3215	Calcolo della radice quadrata, istruzione a 32 bit
DSQRP	0x3A15	Calcolo della radice quadrata, istruzione a 32 bit con polling del fronte di salita (impulso)
POW	0x2216	Elevamento a potenza
POWP	0x2A16	Elevamento a potenza con polling (impulso) sul fronte di salita
DPOW	0x3216	Elevamento a potenza, istruzione a 32 bit
DPOWP	0x3A16	Elevamento a potenza, istruzione a 32 bit con polling del fronte di salita (impulso)
Operazioni di scorrimento		
ROR	0x220B	Rotazione ciclica a destra
RORP	0x2A0B	Rotazione ciclica a destra con polling (impulso) del fronte di salita
DROR	0x320B	Spostamento ciclico a destra, istruzione a 32 bit

DRORP	0x3A0B	Rotazione ciclica a destra, istruzione a 32 bit con polling del fronte di salita (impulso)
ROL	0x222B	Spostamento ciclico a sinistra
ROLP	0x2A2B	Rotazione ciclica a sinistra con polling (impulso) del fronte di salita
DROL	0x322B	Rotazione ciclica a sinistra, istruzione a 32 bit
DROLP	0x3A2B	Rotazione ciclica a sinistra, istruzione a 32 bit con polling del fronte di salita (impulso)
Elaborazione dati		
ZRST	0x2203	Azzeramento di gruppo degli operandi in un intervallo dato
ZRSTP	0x2A03	Azzeramento di gruppo degli operandi in un intervallo dato con polling (impulso) sul fronte di salita
DECO	0x2211	Decoder 8 → 256-bit
DECOP	0x2A11	Decoder 8 → 256-bit con polling (impulso) sul fronte di salita
ENCO	0x2212	Codificatore a 256 → 8 bit
ENCOP	0x2A12	Encoder a 256 → 8-bit con polling (impulso) sul fronte di salita
SUM	0x2213	Somma dei singoli bit nel registro
SUMP	0x2A13	Somma dei singoli bit nel registro con polling (impulso) sul fronte di salita
DSUM	0x3213	Somma dei singoli bit nel registro, istruzione a 32 bit
DSUMP	0x3A13	Somma dei singoli bit nel registro, istruzione a 32 bit con polling (impulso) sul fronte di salita
BON	0x2214	Verifica lo stato di un bit impostando un'uscita
BONP	0x2A14	Verifica lo stato di un bit impostando un'uscita con polling del fronte di salita (impulso)
DBON	0x3214	Verifica lo stato di un bit con impostazione di un'uscita, istruzione a 32 bit
DBONP	0x3A14	Verifica lo stato di un bit impostando un'uscita, istruzione a 32 bit con polling del fronte di salita (impulso)
FLT	0x220C	Converti intero in virgola mobile
FLTP	0x2A0C	Converti intero in virgola mobile con polling del fronte di salita (impulso)
DFLT	0x320C	Converti intero in virgola mobile, istruzione a 32 bit
DFLTP	0x3A0C	Converti intero in virgola mobile, istruzione a 32 bit con polling sul fronte di salita (impulso)

Operazioni in virgola mobile		
DECMP	0x220F	Confronto di numeri a virgola mobile
DECMPP	0x2A0F	Confronto di numeri a virgola mobile con polling a fronte di salita (impulso)
DEZCP	0x2210	Confronto a virgola mobile di zona
DEZCPP	0x2A10	Confronto a virgola mobile di zona con polling a fronte di salita (impulso)
DEADD	0x2217	Addizione di numeri a virgola mobile
DEADDP	0x2A17	Addizione di numeri a virgola mobile con polling (impulso) sul fronte di salita
DESUB	0x2237	Sottrazione di numeri a virgola mobile
DESUBP	0x2A37	Sottrazione di numeri a virgola mobile con polling (impulso) sul fronte di salita
DEMUL	0x2257	Moltiplicazione di numeri a virgola mobile
DEMULP	0x2A57	Moltiplicazione di numeri a virgola mobile con polling (impulso) a fronte di salita
DEDIV	0x2277	Divisione di numeri a virgola mobile
DEDIVP	0x2A77	Divisione di numeri a virgola mobile con polling (impulso) sul fronte di salita
DESQR	0x2218	Radice quadrata in formato a virgola mobile
DESQRP	0x2A18	Radice quadrata in formato a virgola mobile con polling sul fronte di salita (impulso)
DEPOW	0x2297	Elevamento a potenza in formato a virgola mobile
DEPOWP	0x2A97	Elevamento a potenza in virgola mobile con polling sul fronte di salita (impulso)
INT	0x220D	Conversione di un numero a virgola mobile in un numero intero
INTP	0x2A0D	Conversione di un numero a virgola mobile in un intero con polling del fronte di salita (impulso)
DINT	0x320D	Conversione di un numero a virgola mobile in un intero, istruzione a 32 bit
DINTP	0x3A0D	Conversione di un numero a virgola mobile in un intero, istruzione a 32 bit con polling del fronte di salita (impulso)
Tempo e PWM		
TRD	0x2219	Lettura del valore corrente dell'orologio in tempo reale

TRDP	0x2A19	Lettura del valore corrente dell'orologio in tempo reale con polling (impulso) sul fronte di salita
TWR	0x221A	Modifica del valore di un orologio in tempo reale
TWRP	0x2A1A	Modifica del valore di un orologio in tempo reale con polling del fronte di salita (impulso)
PWM	0x220E	Uscita a modulazione di larghezza d'impulso (PWM)
Data		
DRD	0x2239	Lettura del valore della data corrente
DRDP	0x2A39	Lettura del valore della data corrente con polling (impulso) sul fronte di salita
DWR	0x223A	Cambiare il valore della data
DWRP	0x2A3A	Cambiare il valore della data con polling del fronte di salita (impulso)
Operazioni logiche di tipo contatto		
LD&	0x4204	Il contatto è chiuso se S1 & S2 ≠ 0
DLD&	0x5204	Il contatto è chiuso se S1 & S2 ≠ 0, istruzione a 32 bit
LD|	0x4224	Il contatto è chiuso se S1 | S2 ≠ 0
DLD|	0x5224	Il contatto è chiuso se S1 | S2 ≠ 0, istruzione a 32 bit
LD^	0x4244	Il contatto è chiuso se S1 ^ S2 ≠ 0
DLD^	0x5244	Il contatto è chiuso se S1 ^ S2 ≠ 0, istruzione a 32 bit
AND&	0x4205	Contatto seriale chiuso se S1 & S2 ≠ 0
DAND&	0x5205	Contatto seriale chiuso se S1 & S2 ≠ 0, istruzione a 32 bit
AND|	0x4225	Contatto seriale chiuso se S1 | S2 ≠ 0
DAND|	0x5225	Contatto seriale chiuso se S1 | S2 ≠ 0, istruzione a 32 bit
AND^	0x4245	Contatto seriale chiuso se S1 ^ S2 ≠ 0
DAND^	0x5245	Contatto seriale chiuso se S1 ^ S2 ≠ 0, istruzione a 32 bit
OR&	0x4206	Contatto parallelo chiuso se S1 & S2 ≠ 0
DOR&	0x5206	Contatto parallelo chiuso se S1 & S2 ≠ 0, istruzione a 32 bit
OR|	0x4226	Contatto parallelo chiuso se S1 | S2 ≠ 0
DOR|	0x5226	Contatto parallelo chiuso se S1 | S2 ≠ 0, istruzione a 32 bit
OR^	0x4246	Contatto parallelo chiuso se S1 ^ S2 ≠ 0
DOR^	0x5246	Contatto parallelo chiuso se S1 ^ S2 ≠ 0, istruzione a 32 bit

Operazioni di confronto del tipo di contatto		
LD=	0x4264	Il contatto è chiuso se S1 = S2
DLD=	0x5264	Il contatto è chiuso se S1 = S2, istruzione a 32 bit
LD>	0x4284	Il contatto è chiuso se S1 > S2
DLD>	0x5284	Il contatto si chiude se S1 > S2, istruzione a 32 bit
LD<	0x42A4	Il contatto è chiuso se S1 < S2
DLD<	0x52A4	Il contatto si chiude se S1 < S2, istruzione a 32 bit
LD<>	0x42C4	Il contatto è chiuso se S1 ≠ S2
DLD<>	0x52C4	Il contatto è chiuso se S1 ≠ S2, istruzione a 32 bit
LD<=	0x42E4	Il contatto è chiuso se S1 ≤ S2
DLD<=	0x52E4	Il contatto è chiuso se S1 ≤ S2, istruzione a 32 bit
LD>=	0x4304	Il contatto è chiuso se S1 ≥ S2
DLD>=	0x5304	Il contatto si chiude se S1 ≥ S2, istruzione a 32 bit
AND=	0x4265	Contatto seriale chiuso se S1 = S2
DAND=	0x5265	Contatto seriale chiuso se S1 = S2, istruzione a 32 bit
AND>	0x4285	Contatto seriale chiuso se S1 > S2
DAND>	0x5285	Contatto seriale chiuso se S1 > S2, istruzione a 32 bit
AND<	0x42A5	Contatto seriale chiuso se S1 < S2
DAND<	0x52A5	Contatto seriale chiuso se S1 < S2, istruzione a 32 bit
AND<>	0x42C5	Contatto seriale chiuso se S1 ≠ S2
DAND<>	0x52C5	Contatto seriale chiuso se S1 ≠ S2, istruzione a 32 bit
AND<=	0x42E5	Contatto seriale chiuso se S1 ≤ S2
DAND<=	0x52E5	Contatto seriale chiuso se S1 ≤ S2, istruzione a 32 bit
AND>=	0x4305	Contatto seriale chiuso se S1 ≥ S2
DAND>=	0x5305	Contatto seriale chiuso se S1 ≥ S2, istruzione a 32 bit
OR=	0x4266	Contatto parallelo chiuso se S1 = S2
DOR=	0x5266	Contatto parallelo chiuso se S1 = S2, istruzione a 32 bit
OR>	0x4286	Contatto parallelo chiuso se S1 > S2
DOR>	0x5286	Contatto parallelo chiuso se S1 > S2, istruzione a 32 bit
OR<	0x42A6	Contatto parallelo chiuso se S1 < S2
DOR<	0x52A6	Contatto parallelo chiuso se Contatto parallelo chiuso se S1 < S2, istruzione a 32 bit

OR<>	0x42C6	Contatto parallelo chiuso se S1 ≠ S2
DOR<>	0x52C6	Contatto parallelo chiuso se S1 ≠ S2, istruzione a 32 bit
OR<=	0x42E6	Contatto parallelo chiuso se S1 ≤ S2
DOR<=	0x52E6	Contatto parallelo chiuso se S1 ≤ S2, istruzione a 32 bit
OR>=	0x4306	Contatto parallelo chiuso se S1 ≥ S2
DOR>=	0x5306	Contatto parallelo chiuso se S1 ≥ S2, istruzione a 32 bit
Controllo motore passo-passo		
SPIN	0x2207	Avvia movimento preimpostato
SPINP	0x2A07	Avvia il movimento preimpostato con il polling del fronte di salita (impulso)
TORQUE	0x2227	Applicare le correnti impostate al motore
TORQUEP	0x2A27	Applicare le correnti impostate al motore con polling (impulso) sul fronte di salita
HSTOP	0x2247	Passa immediatamente alla modalità hold
HSTOPP	0x2A47	Passa immediatamente alla modalità hold con polling del fronte di salita (impulso)
HHIZ	0x2267	Disenergizzare immediatamente le fasi del motore (l'albero ruota liberamente)
HHIZP	0x2A67	Disenergizzare le fasi del motore immediatamente (l'albero ruota liberamente) con polling (impulso) a fronte di salita
SSTOP	0x2287	Decelerare fino all'arresto completo e passare alla modalità di mantenimento
SSTOPP	0x2A87	Decelerare fino all'arresto completo e passare alla modalità di mantenimento con polling del fronte di salita (impulso)
SHIZ	0x22A7	Decelerare fino all'arresto completo e diseccitare le fasi del motore (l'albero ruota liberamente)
SHIZP	0x2AA7	Decelerare fino all'arresto completo e diseccitare le fasi del motore (l'albero ruota liberamente) con polling (impulso) sul fronte di salita

Appendice C. Esempi di programmi utente

Esempio 1. Utilizzo del comando RUN

LDP	X0			***;cattura il fronte di salita dell'impulso all'ingresso X0 (pulsante)***
DMOV	K8	D359		*;imposta velocità minima 8 pps*
DMOV	K120000	D357		*;imposta la velocità massima 120000 pps*
FMOV	K30000	D361	K2	*;imposta accelerazione e decelerazione 30000 pps^2*
MOV	K3	D366		*;microstepping 1/8 (fare riferimento alla descrizione dell'istruzione SPIN)*
MOV	K1	D374		*;direzione – avanti*
DMOV	K6000	D377		*;imposta la velocità fullstep a 6000 pps/sec*
MOV	K1	D379		*;abilitare il passaggio alla modalità fullstep al raggiungimento della velocità fullstep*
MOV	K0	D376		*;comando ESEGUI*
FMOV	K1500	D367	K2	*;correnti di accelerazione e decelerazione 1500 mA*
MOV	K1200	D369		*;corrente a velocità costante 1200 mA*
MOV	K600	D370		*;mantenendo la corrente a 600 mA*
TORQUE				*;applica i valori correnti*
FMOV	K0	D380	K3	*;nessuna risposta di errore, errori ripristinati, usare ; MIN_SPEED*
SPIN				*;avvia movimento*
LDP	X1			*;cattura il fronte di salita dell'impulso all'ingresso X1 (pulsante)*
SSTOP				*;arresto in base al DEC preimpostato e passaggio alla modalità di mantenimento*
LDP	X2			*;cattura il fronte di salita dell'impulso all'ingresso X2 (pulsante)*
SHIZ				*;arresto in base al DEC preimpostato e passaggio alla modalità HiZ*
LDP	X3			*;cattura il fronte di salita dell'impulso all'ingresso X3 (pulsante)*
HSTOP				*;passare immediatamente alla modalità di attesa*
LDP	X4			*;cattura il fronte di salita dell'impulso all'ingresso X4 (pulsante)*
HHIZ				*;passa immediatamente alla modalità HiZ*
END				*;fine del programma*

Esempio 2. Utilizzo dei comandi MOVE, GOTO, GOHOME

LD	M0			***;per saltare la sezione di inizializzazione, verificare la condizione M0***
CJ	P1			*;e salta alla riga contrassegnata con P1*
LDP	M108			*;M108 fronte di salita solo dopo l'inizializzazione*
DMOV	K120000	D357		*;imposta la velocità massima a 120000 pps*
FMOV	K30000	D361	K2	*;imposta l'accelerazione e la decelerazione a 30000pps^2*

MOV	K3	D366		*;microstepping 1/8 (fare riferimento alla descrizione dell'istruzione SPIN)*
DMOV	K6000	D377		*;imposta la velocità fullstep a 6000 pps/sec*
MOV	K1	D379		*;abilitare il passaggio alla modalità fullstep al raggiungimento della velocità fullstep*
FMOV	K1500	D367	K2	*;correnti di accelerazione e decelerazione 1500 mA*
MOV	K1200	D369		*;corrente a velocità costante 1200 mA*
MOV	K600	D370		*;mantenendo la corrente a 600 mA*
TORQUE				*;applica i valori correnti*
FMOV	K0	D380	K2	*;nessuna risposta di errore, gli errori vengono ripristinati*
MOV	K1	D382		*;usa il calcolo automatico della velocità iniziale e finale*
DMOV	K0	D363		*;azzerare la posizione corrente*
SET	M0			*;attiva la condizione di bypass dell'inizializzazione del driver*
P	1			*;punto di transizione*
LDP	X0			*;cattura il fronte di salita dell'impulso all'ingresso X0 (pulsante)*
AND&	D371	K3		*;solo se il motore è in modalità HiZ o Hold*
DMOV	K10000	D372		*;sposta10000 micro passi*
MOV	K1	D374		*;in avanti*
MOV	K1	D376		*;vieneeseguito dal comando MOVE*
SPIN				*;avvia movimento*
LDP	X1			*;cattura il fronte di salita dell'impulso all'ingresso X1 (pulsante)*
AND&	D371	K3		*;solo se il motore è in modalità HiZ o Hold*
DMOV	K100000	D372		*;spostamento a una posizione con coordinata 100000*
MOV	K2	D376		*;vieneeseguito dal comando GOTO*
SPIN				*;avvia movimento*
LDP	X2			*;cattura il fronte di salita dell'impulso all'ingresso X2 (pulsante)*
AND&	D371	K3		*;solo se il motore è in modalità HiZ o Hold*
MOV	K0	D374		*;movimento alla posizione "0" in direzione inversa*
MOV	K4	D376		*;vieneeseguito dal comando GOHOME*
SPIN				*;avvia movimento*
LDP	X3			*;cattura il fronte di salita dell'impulso all'ingresso X3 (pulsante)*
SSTOP				*;arresto in base al DEC preimpostato e passaggio alla modalità di mantenimento*
LDP	X4			*;cattura il fronte di salita dell'impulso all'ingresso X4 (pulsante)*
SHIZ				*;arresto in base al DEC preimpostato e passaggio alla modalità HiZ*
LDP	X5			*;cattura il fronte di salita dell'impulso all'ingresso X5 (pulsante)*
HSTOP				*;passare immediatamente alla modalità di attesa*
LDP	X6			*;cattura il fronte di salita dell'impulso all'ingresso X6 (pulsante)*

```
HHIZ                    ;passa immediatamente alla modalità HiZ
END                     ;fine del programma
```

Esempio 3. Utilizzo dei comandi GOUNTIL_SLOWSTOP e RELEASE

Utilizzo dei comandi GOUNTIL_SLOWSTOP e RELEASE come esempio di spostamento nella posizione di origine lungo il finecorsa positivo (vedere Fig. 36).

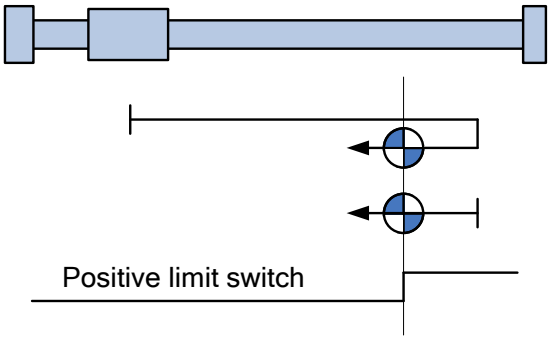


Fig. 36. Spostamento all'origine.

LD	M0			***;per saltare la sezione di inizializzazione, verificare la condizione M0***
CJ	P1			*;e salta alla riga contrassegnata con P1*
LDP	M108			*;M108 fronte di salita solo dopo l'inizializzazione*
FMOV	K30000	D361	K2	*;imposta l'accelerazione e la decelerazione a 30000pps²*
MOV	K3	D366		*;microstepping 1/8 (fare riferimento alla descrizione dell'istruzione SPIN)*
DMOV	K6000	D377		*;imposta la velocità fullstep a 6000 pps/sec*
MOV	K1	D379		*;abilitare il passaggio alla modalità fullstep al raggiungimento della velocità fullstep*
FMOV	K1500	D367	K2	*;correnti di accelerazione e decelerazione 1500 mA*
MOV	K1200	D369		*;corrente a velocità costante 1200 mA*
MOV	K600	D370		*;mantenendo la corrente a 600 mA*
TORQUE				*;applica i valori correnti*
FMOV	K0	D380	K2	*;nessuna risposta di errore, gli errori vengono ripristinati*
MOV	K1	D382		*;usa il calcolo automatico della velocità iniziale e finale*
MOV	K7	D375		*;il limite dell'interruttore è collegato all'ingresso IN7*
SET	M0			*;attiva la condizione di bypass dell'inizializzazione del driver*
P	1			*;punto di transizione*
LDP	X0			*;cattura il fronte di salita dell'impulso all'ingresso X0 (pulsante)*
AND&	D371	K3		*;solo se il motore è in modalità HiZ o Hold*
DMOV	K20000	D357		*;imposta la velocità massima a 20000 pps*
MOV	K5	D376		*;comando GOUNTIL_SLOWSTOP*
MOV	K1	D374		*;direzione – avanti*
SPIN				*;avvia movimento*
SET	M1			*;imposta il flag per avviare la prima fase*
LD	M1			*;attendere l'attivazione del finecorsa alla prima fase*
AND&	D371	K2		*;e il motore si ferma*
RST	M1			*;reimposta il flag della prima fase*

DMOV	K1000	D357	*;diminuisci la velocità*
MOV	K9	D376	*;movimento nella direzione opposta fino al finecorsa*
MOV	K0	D374	*;apre*
SPIN			*;avvia movimento*
SET	M2		*;e passare alla seconda fase*
LD	M2		*;in attesa che il finecorsa si apra e il motore si arresti*
AND&	D371	K2	*;nella seconda fase*
RST	M2		*;reimposta il flag della seconda fase*
DMOV	K0	D363	*;e ripristina la posizione corrente, che diventa l'origine*
LDP	X1		*;cattura il fronte di salita dell'impulso all'ingresso X1 (pulsante)*
SSTOP			*;arresto in base al DEC preimpostato e passaggio alla modalità di mantenimento*
LDP	X2		*;cattura il fronte di salita dell'impulso all'ingresso X2 (pulsante)*
SHIZ			*;arresto in base al DEC preimpostato e passaggio alla modalità HiZ*
LDP	X3		*;cattura il fronte di salita dell'impulso all'ingresso X3 (pulsante)*
HSTOP			*;passare immediatamente alla modalità di attesa*
LDP	X4		*;cattura il fronte di salita dell'impulso all'ingresso X4 (pulsante)*
HHIZ			*;passa immediatamente alla modalità HiZ*
FEND			*;fine del programma principale*
I	1007		*;gestore di interrupt per l'ingresso IN7 (richiesto per i comandi GOUNTIL _... e ... RELEASE, può essere lasciato vuoto)*
IRET			*;ritorno al programma principale*
END			*;fine del programma*

Appendice D. Codice del programma di servizio “Controllo della velocità del motore passo-passo”

LD	M0			***;condizione di bypass dell'inizializzazione***
CJ	P1			
LDP	M108			*;parte di inizializzazione*
MPS				
LD=	D320	K6		*;D320 memorizza il valore del microstepping*
OR>	D320	K8		
OR<	D320	K0		
ANB				
MOV	K0	D320		
MRD				
MOV	D320	D366		
MOV	D320	D0		*;D0 –il registro di servizio per la visualizzazione del microstepping*
MOV	K0	D2		
MRD				
AND>	D0	K6		*;poiché il controller non supporta il microstepping 1/64,*
DEC	D0			*;salta questo valore*
MRD				
DECO	D0	Y0	K3	*;visualizzazione del microstepping sulla scala delle uscite*
MOV	K0	D379		*;configurazione iniziale del driver del motore passo-passo*
MOV	K0	D376		
MOV	K250	D359		
MOV	K0	D380		
MRD				
LD<	D321	K0		*;D321 memorizza i dati del metodo di controllo:*
OR>	D321	K2		*;potenziometro/pulsanti/encoder*
ANB				
MOV	K0	D321		
MRD				
SET	M0			
MOV	D321	Y10		*;visualizzazione del metodo di controllo*
MRD				
AND<>	A0	D321		
MPS				
AND=	D321	K2		*;se è selezionato un**encoder**, allora le periferiche*
MOV	K12	D355		*;del controllore devono essere impostati di conseguenza*
MOV	D321	A0		
RST	M108			*;e riavviare il programma*
MPP				
AND<>	D321	K2		
MOV	K0	D355		
MOV	D321	A0		

RST	M108			
MRD				
MUL	D353	K10	D4	*;dati del potenziometro**1***
DIV	D4	K27	D4	
FMOV	D4	D367	K3	*;imposta corrente di accelerazione, decelerazione e velocità costante*
DIV	D4	K2	D370	*;mantenimento della corrente – 50% della corrente di lavoro*
TORQUE				
MRD				
AND	X7			
MOV	K1	D374		
MRD				
ANI	X7			
MOV	K0	D374		
MRD				
DLD<	D322	K250		*;D322, D323 velocità impostata dai pulsanti*
DOR>	D322	K120000		
ANB				
DMOV	K250	D322		
MRD				
DMOV	D322	D5		
MRD				
DLD<	D324	K250		*;D324, D325 velocità impostata dall'encoder*
DOR>	D324	K120000		
ANB				
DMOV	K250	D324		
MRD				
DMOV	D324	D15		
MRD				
DMOV	K0	C64		
DMOV	C64	D7		
MPP				
ANI	X4			
HSTOP				
LD	X2			
OUT	M102			
EI				*;*abilita*le interruzioni, la fine del blocco di inizializzazione*
P	1			
LD	X4			
AND&	D371	HFE		
HHIZ				
LDI	X4			
MPS				
AND	M102			
AND	X3			
AND&	D371	K3		

CALL	P10A0			
SPIN				
MPP				
LDI	X3			
ANB				
AND&	D371	HFD		
HSTOP				
LDI	M102			
AND	X3			
ANI	X4			
AND&	D371	H15		
SSTOP				
LD	M109			*;se si è verificato un errore e l'indicatore ERR era attivo -*
TMR	T0	K10		*;avvia il timer per spegnerlo*
AND	T0			
RST	M109			
LD<>	A0	K1		
CJ	P19			
LD=	A0	K1		
MPS				
ANDP	X0			
DADD	D5	D354	D5	
MPS				
DAND>	D5	K120000		
DMOV	K120000	D5		
MPP				
DMOV	D5	D322		
MPP				
ANDP	X1			
DSUB	D5	D354	D5	
MPS				
DAND<	D5	K250		
DMOV	K250	D5		
MPP				
DMOV	D5	D322		
P	19			
LD<>	A0	K2		
CJ	P20			
LD=	A0	K2		
MPS				
DSUB	C64	D7	D9	
DADD	D7	D9	D7	
DCMP	D9	K0	M1	
MRD				
AND	M1			
MOV	D354	D1		
DMUL	D9	D1	D11	

DADD	D11	D15	D15	
MPP				
AND	M3			
MOV	D354	D1		
ABS	D9			
DMUL	D9	D1	D11	
DADD	D15	D11	D15	
LD	M1			
OR	M3			
MPS				
DAND<	D15	K250		
DMOV	K250	D15		
MRD				
DAND>	D15	K120000		
DMOV	K120000	D15		
MPP				
DMOV	D15	D324		
P	20			
FEND				*;fine del programma principale*
P	10			*;sottoprogramma di impostazione della velocità tramite potenziometro*
LD	M108			
MOV	K0	D2		
MOV	D354	D1		
MPS				
AND=	D1	K0		
MOV	K1	D1		
MRD				
DMUL	D1	K29	D13	
MRD				
DAND<	D13	K250		
DMOV	K250	D13		
MPP				
DMOV	D13	D357		
CALL	P0			
SRET				
P	11			*; sottoprogramma di impostazione della velocità tramite pulsanti*
LD	M108			
DMOV	D322	D357		
CALL	P0			
SRET				
P	12			*; sottoprogramma di impostazione della velocità tramite encoder*
LD	M108			
DMOV	D324	D357		
CALL	P0			
SRET				

P	0			*;sottoprogramma di aggiornamento accelerazione e decelerazione*
LD	M108			
MOV	D352	D3		
MPS				
AND=	D3	K0		
MOV	K1	D3		
MPP				
MUL	D3	K14	D361	
MOV	D361	D362		
SRET				
I	50			*;Interruzione di 500 ms per l'aggiornamento delle correnti*
LD	M108			
MUL	D353	K10	D4	
DIV	D4	K27	D4	
FMOV	D4	D367	K3	
DIV	D4	K2	D370	
TORQUE				
IRET				
I	10			*;interruzione con un periodo di 100 ms per aggiornare la velocità*
LDI	X6			
AND	X2			
AND	M102			
AND	X3			
ANI	X4			
BON	D371	M60	K6	
ANI	M60			
CALL	P10A0			
SPIN				
IRET				
I	1005			*;interruzione dall'ingresso IN5*
LD	M105			
INC	D320			
MPS				
AND=	D320	K6		
INC	D320			
MRD				
AND=	D320	K9		
MOV	K0	D320		
INC	D321			
MPS				
AND=	D321	K3		
MOV	K0	D321		
MRD				
MOV	D321	Y10		
MOV	D321	A0		

```
MRD
AND=    D321    K2
MOV     K12     D355
RST     M108
MPP
AND<>   D321    K2
MOV     K0      D355
RST     M108
MRD
MOV     D320    D0
MOV     D320    D366
MRD
AND>    D0      K6
DEC     D0
MPP
DECO    D0      Y0      K3
IRET
I       1004                    ;interruzione dall'ingresso IN4
LD      M104
HHIZ
LDI     M104
MPS
AND     X2
AND     X3
AND&    D371    K3
CALL    P10A0
SPIN
MPP
LDI     X2
ORI     X3
ANB
HSTOP
IRET
I       1003                    ;interruzione dall'ingresso IN3
LDI     M103
ANI     X4
HSTOP
LD      M103
AND     X2
ANI     X4
AND&    D371    K3
CALL    P10A0
SPIN
IRET
I       1002                    ;interruzione dall'ingresso IN2
LDI     M102
AND     X3
ANI     X4
```

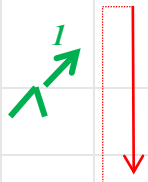
```
SSTOP
LD      M102
AND     X3
ANI     X4
AND&    D371    K3
CALL    P10A0
SPIN
IRET
I       1007                    ;interruzione dall'ingresso IN7
LD&     D371    H1C
MPS
AND     M107
AND=    D374    K0
SSTOP
MPP
ANI     M107
AND=    D374    K1
SSTOP
LD      M107
MOV     K1      D374
AND     X2
AND     X3
ANI     X4
AND&    D371    K3
CALL    P10A0
SPIN
LDI     M107
MOV     K0      D374
AND     X2
AND     X3
ANI     X4
AND&    D371    K3
CALL    P10A0
SPIN
IRET
I       2000                    ;interruzione quando si verifica un errore del driver
LD&     D381    K1
MOV     K0      D381
SET     M109
MOV     K0      T0
IRET
END
```

Appendice E. La durata dei fronti degli operandi M e Y

Tutte le informazioni di seguito sono valide per gli operandi M e Y, sia per i fronti di salita che di discesa.

Esempio 1

Si consideri la durata del fronte di salita dell'operando M0, con passaggio garantito del punto di inizio vita alla scansione successiva.

Riga	Istruzione	Operandi, numero d'ordine		Durata M0 frontale, scansione		Spiegazione
		1	2	1	2	
1	LDP	M108				Il fronte di salita di M108 esiste solo alla prima scansione del programma
2	ZRST	D0	D2			Azzeramento di D0…D2
3	LD	M108				
4	OUT	M0				L'inizio della vita del fronte di salita dell'operando M0 è nella prima scansione e la fine è nella seconda
5	P	1				
6	LD	M0				
7	CALL	P0				Lo stato corrente M0 viene salvato per ogni salto alla subroutine P0.
8	LDP	M0				Funzionerà solo una volta, poiché la condizione di ripassaggio attraverso il punto iniziale del fronte dell'operando M0 nella successiva scansione del programma è soddisfatta.
9	INC	D0				Il risultato del programma è D0 = 1.
10	FEND					
11	I	0				La prima interruzione si verifica dopo la riga 2 alla prima scansione.In questo momento il fronte di salita di M0 è assente. La seconda interruzione viene elaborata dopo la riga 6. La presenza di un fronte di salita a M0 viene trasmessa all'elaborazione dell'interruzione, quindi il risultato del lavoro è D2 = 1.
13	LDP	M0				
14	INC	D2				
15	IRET					
16	P	0				

17	LDP	M0				La condizione viene soddisfatta alla prima scansione. La condizione non viene soddisfatta alla seconda scansione.
18	INC	D1				Il risultato è D1 = 1.
19	SRET					
20	END					

- interruzione

- esistenza di un fronte dell'operando

Esempio 2

Si consideri la durata del fronte di salita dell'operando M0, in assenza del passaggio del punto di inizio vita alla scansione successiva.

Riga	Istruzione	Operandi, numero d'ordine		Durata M0 frontale, scansione			Spiegazione
		1	2	1	2	3	
1	LDP	M0					
2	CJ	P1					Aggirare il punto di inizio vita del fronte.
3	LDP	M108					Il fronte di salita di M108 esiste solo alla prima scansione del programma.
4	ZRST	D0	D2				Azzeramento di D0…D2.
5	LD	M108					
6	OUT	M0					L'inizio della vita del fronte di salita dell'operando M0 nella prima scansione.Il passaggio successivo di questo punto sarà solo nella terza scansione.
7	P	1					
8	LDP	M0					Funzionerà due volte, poiché il punto di inizio della durata del fronte è stato saltato dal gestore dei comandi nella seconda scansione e la durata è stata aumentata fino alla ricezione del comando END / FEND.
9	INC	D0					Il risultato del programma è D0 = 2.

10	END						La durata del fronte di salita dell'operando M0 viene estesa fino alla fine della scansione a causa dell'assenza del passaggio del punto di inizio della durata del fronte M0.

↓ - esistenza di un fronte dell'operando

Se è richiesto un singolo passaggio tra i punti 7 ... 9, allora, ad esempio, può essere utilizzato il contatto relè opzionale M1. Il programma verrà modificato come segue:

```
1   LDP     M0
2   CJ      P1
3   LDP     M108
4   ZRST    D0          D2
5   ZRST    M1                  ;Inizializzazione M1
6   LD      M108
7   OUT     M0
8   P       1
9   LDP     M0
10  ANI     M1                  ;Condizione aggiuntiva
11  INC     D0                  ;Il risultato del programma èD0 = 1
12  SET     M1                  ;Blocco
13  END
```

Appendice F. Debug del programma utente

La modalità di debug consente all'utente di:

- impostare quattro breakpoint per l'esecuzione del programma utente (breakpoint),
- visualizzare e modificare gli operandi,
- mettere in pausa e riprendere l'esecuzione del programma utente.

Di seguito è riportato l'elenco dei registri del debugger:

Indirizzo	Tipo	Dimensione	Descrizione
Controllo dell'esecuzione del programma utente			
0x6100	Registri di Input	16-bit	Indice corrente (numero di riga di comando) del programma utente
0x6100	Bobine	-	Impostare l'oggetto attiva la modalità di debug, il ripristino la disattiva. Inoltre, la modalità di debug viene disattivata quando l'interruttore a levetta RUN/STOP viene portato nello stato STOP.
0x6100	Ingressi Discreti	-	Indicazione della modalità di debug. Impostato – il controller è in modalità debug Azzera - il controller non è in modalità debug
0x6101	Bobine	-	L'impostazione dell'oggetto sospende l'esecuzione del programma utente. L'impostazione del registro sospenderà l'esecuzione del programma utente all'indice corrente. Reset - riprende l'esecuzione.
0x6101	Ingressi Discreti	-	Indicazione di sospensione di un programma utente. Impostato – il programma utente è sospeso Reset - il programma utente è in esecuzione
0x6102	Bobine	-	Impostando l'oggetto si attiva il debug passo-passo. Quando si tenta di riprendere il programma utente resettando le Bobine 6101h, l'esecuzione verrà automaticamente interrotta all'indice successivo.
Punti di interruzione			
● 1			*Oltre al debug passo-passo, quando l'esecuzione di un programma utente si sospende ad ogni riga di comando successiva, è possibile specificare quattro punti di interruzione in corrispondenza dei quali l'esecuzione del programma verrà sospesa.*

Indirizzo	Tipo	Dimensione	Descrizione
Punto di interruzione 1			
0x6200	Bobine	-	Impostando l'oggetto si attiva il breakpoint 1. L'esecuzione del programma utente verrà sospesa all'indice, che è specificato nei Registri di Mantenimento 6200h.
0x6200	Registri di Mantenimento	16-bit	Indice (numero di riga di comando) del punto di interruzione 1.
Punto di interruzione 2			
0x6201	Bobine	-	Impostando l'oggetto si attiva il breakpoint 2. L'esecuzione del programma utente verrà sospesa all'indice, che è specificato nei Registri di Mantenimento 6201h.
0x6201	Registri di Mantenimento	16-bit	Indice (numero di riga di comando) del punto di interruzione 2.
Punto di interruzione 3			
0x6202	Bobine	-	Impostando l'oggetto si attiva il breakpoint 3. L'esecuzione del programma utente verrà sospesa all'indice, che è specificato nei Holding Register 6202h.
0x6202	Registri di Mantenimento	16-bit	Indice (numero di riga di comando) del punto di interruzione 3.
Punto di interruzione 4			
0x6203	Bobine	-	Impostando l'oggetto si attiva il breakpoint 4. L'esecuzione del programma utente verrà sospesa all'indice, che è specificato nei Holding Register 6203h.
0x6203	Patrimonio Registri	16-bit	Indice (numero di riga di comando) del punto di interruzione 4.
Monitoraggio e modifica degli operandi			
0x6000	Bobine	-	Richiesta di lettura dati impostando il registro. Il ripristino avviene automaticamente. La risposta alla richiesta indica la disponibilità dei dati richiesti relativi all'operando.

Indirizzo	Tipo	Dimensione	Descrizione
0x6001	Bobine	-	Richiesta di scrittura dati impostando il registro. Il ripristino avviene automaticamente. La risposta alla richiesta indica che i dati sono stati scritti nell'operando.
0x6002	Bobine	-	Dimensione dei dati del registro di lettura/scrittura Azzera – 16-bit Set – 32-bit.
0x6003	Bobine	-	L'impostazione del registro annulla la modifica del valore dell'operando quando la bobina 6001h è impostata – per gli operandi “C” e “T”.
0x6004	Bobine	-	L'impostazione del registro annulla la modifica del segnale dell'operando quando la bobina 6001h è impostata – per gli operandi “C” e “T”.
0x6000	Ingressi Discreti	-	L'oggetto viene impostato quando un'operazione di lettura o scrittura di un operando fallisce. Il ripristino viene eseguito automaticamente quando viene richiesta un'operazione di lettura o scrittura.
Parametrizzazione degli operandi			
0x6000	Registri di Mantenimento	16-bit	Tipo di operando. X (0x58), Y (0x59), M (0x4D), T (0x54), C (0x43), A (0x41), B (0x42), D (0x44).
0x6001	Registri di Mantenimento	16-bit	Indice dell'operando.
Monitoraggio degli operandi			
0x6002	Registri di Input	32-bit	Questo registro contiene il valore dell'operando (se disponibile) parametrizzato per la lettura. La dimensione è impostata dalle Bobine 6002h.
0x6004	Registri di Input	16-bit	Questo registro contiene il segnale dell'operando (se disponibile), parametrizzato per la lettura. Valori possibili: 0x00 – basso livello, 0x03 – alto livello, con un fronte di salita 0x02 – alto livello, 0x04 – basso livello, con un fronte di discesa.

Indirizzo	Tipo	Dimensione	Descrizione
Modifica operandi			
0x6002	Registri di Mantenimento	32-bit	Questo registro contiene il valore dell'operando (se disponibile) parametrizzato per la scrittura. La dimensione è impostata dalle Bobine 6002h.
0x6004	Registri di Mantenimento	16-bit	Questo registro contiene il segnale dell'operando (se disponibile) parametrizzato per la scrittura. Valori possibili: 0x00 – basso livello, 0x03 – alto livello, con un fronte di salita 0x02 – alto livello, 0x04 – basso livello, con un fronte di discesa.

Fornitura in set completi

Driver per motore passo-passo RS 434540 / RS 434542 / RS 434543 1 pz

Informazioni sul produttore

RS Components aderisce alla linea di sviluppo continuo e si riserva il diritto di apportare modifiche e miglioramenti al design e al software del prodotto senza preavviso.

Le informazioni contenute in questo manuale sono soggette a modifiche in qualsiasi momento e senza preavviso.

Garanzia

Qualsiasi riparazione o modifica deve essere eseguita dal produttore o da un'azienda autorizzata.

Il produttore garantisce il funzionamento senza guasti del controller per 12 mesi dalla data di vendita, a condizione che vengano rispettate le condizioni operative.

Indirizzo del reparto vendite del produttore:

RS Components Ltd, Birchington Rd, Corby, NN17 9RS, United Kingdom, rs-online.com
RS Components GmbH, Mainzer Landstrasse 180, 60327 Frankfurt/Main, Germany, rs-online.com

Ultima modifica: 07.2025