SIEMENS

SIMATIC

Lista istruzioni (AWL) per S7-300/400

Manuale di riferimento

Il presente manuale fa parte del pacchetto di documentazione con il numero di ordinazione:

6ES7810-4CA10-8EW1

05/2010
A5E02790286-01

Avvertenze di legge

Concetto di segnaletica di avvertimento

Questo manuale contiene delle norme di sicurezza che devono essere rispettate per salvaguardare l'incolumità personale e per evitare danni materiali. Le indicazioni da rispettare per garantire la sicurezza personale sono evidenziate da un simbolo a forma di triangolo mentre quelle per evitare danni materiali non sono precedute dal triangolo. Gli avvisi di pericolo sono rappresentati come segue e segnalano in ordine descrescente i diversi livelli di rischio.

⚠PERICOLO
questo simbolo indica che la mancata osservanza delle opportune misure di sicurezza **provoca** la morte o gravi lesioni fisiche.

⚠AVVERTENZA
il simbolo indica che la mancata osservanza delle relative misure di sicurezza **può causare** la morte o gravi lesioni fisiche.

⚠CAUTELA
con il triangolo di pericolo indica che la mancata osservanza delle relative misure di sicurezza può causare lesioni fisiche non gravi.

CAUTELA
senza triangolo di pericolo indica che la mancata osservanza delle relative misure di sicurezza può causare danni materiali.

ATTENZIONE
indica che, se non vengono rispettate le relative misure di sicurezza, possono subentrare condizioni o conseguenze indesiderate.

Nel caso in cui ci siano più livelli di rischio l'avviso di pericolo segnala sempre quello più elevato. Se in un avviso di pericolo si richiama l'attenzione con il triangolo sul rischio di lesioni alle persone, può anche essere contemporaneamente segnalato il rischio di possibili danni materiali.

Personale qualificato

Il prodotto/sistema oggetto di questa documentazione può essere adoperato solo da **personale qualificato** per il rispettivo compito assegnato nel rispetto della documentazione relativa al compito, specialmente delle avvertenze di sicurezza e delle precauzioni in essa contenute. Il personale qualificato, in virtù della sua formazione ed esperienza, è in grado di riconoscere i rischi legati all'impiego di questi prodotti/sistemi e di evitare possibili pericoli.

Uso conforme alle prescrizioni di prodotti Siemens

Si prega di tener presente quanto segue:

⚠AVVERTENZA
I prodotti Siemens devono essere utilizzati solo per i casi d' impiego previsti nel catalogo e nella rispettiva documentazione tecnica. Qualora vengano impiegati prodotti o componenti di terzi, questi devono essere consigliati oppure approvati da Siemens. Il funzionamento corretto e sicuro dei prodotti presuppone un trasporto, un magazzinaggio, un' installazione, un montaggio, una messa in servizio, un utilizzo e una manutenzione appropriati e a regola d' arte. Devono essere rispettate le condizioni ambientali consentite. Devono essere osservate le avvertenze contenute nella rispettiva documentazione.

Marchio di prodotto

Tutti i nomi di prodotto contrassegnati con ® sono marchi registrati della Siemens AG. Gli altri nomi di prodotto citati in questo manuale possono essere dei marchi il cui utilizzo da parte di terzi per i propri scopi può violare i diritti dei proprietari.

Esclusione di responsabilità

Abbiamo controllato che il contenuto di questa documentazione corrisponda all'hardware e al software descritti. Non potendo comunque escludere eventuali differenze, non possiamo garantire una concordanza perfetta. Il contenuto di questa documentazione viene tuttavia verificato periodicamente e le eventuali correzioni o modifiche vengono inserite nelle successive edizioni.

Siemens AG
Industry Sector
Postfach 48 48
90026 NÜRNBERG
GERMANIA

A5E02790286-01
℗ 02/2010

Prefazione

Scopo del manuale

Questo manuale ha lo scopo di supportare l'utente nella creazione di programmi nel linguaggio di programmazione AWL.

Esso descrive gli elementi del linguaggio di programmazione AWL, la sua sintassi e il modo di funzionamento.

Requisiti di base

I destinatari di questo manuale sono i programmatori di programmi S7, chi li mette in servizio e il personale di assistenza. Vengono presupposte delle nozioni generali nel campo della tecnica dell'automazione.

È inoltre necessario disporre delle conoscenze operative sui computer o strumenti di lavoro simili ai PC (p. es. dispositivi di programmazione) in ambiente MS Windows XP, MS Windows Server 2003 o MS Windows 7.

Validità del manuale

Il presente manuale ha validità per il pacchetto software STEP 7 V5.5.

Adempimento delle norme secondo l'IEC 1131-3

AWL corrisponde al linguaggio "Lista istruzioni" (ingl. Instruction List) stabilito nella norma DIN EN-61131-3 (int. IEC 1131-3), ma per quanto riguarda le operazioni vi sono delle differenze sostanziali. Informazioni precise sull'adempimento delle norme possono essere consultate nella tabella di adempimento delle norme nel file NORM_TAB.RTF di STEP 7.

Presupposti

Il presente manuale di AWL presuppone che l'utente sia in possesso delle nozioni teoriche inerenti i programmi S7 che sono riportate nella Guida online a STEP 7. Poiché i pacchetti dei linguaggi si basano sul software di base STEP 7 l'utente dovrebbe già sapere come utilizzare il software di base STEP 7 e la relativa documentazione.

Il presente manuale è parte integrante del pacchetto di documentazione "Nozioni di riferimento di STEP 7".

La tabella seguente riporta un riepilogo della documentazione relativa a STEP 7.

Documentazione	Scopo	Numero di ordinazione
Nozioni fondamentali di STEP 7 mediante • Primi passi ed esercitazioni con STEP 7 • Programmazione con STEP 7 • Configurazione dell'hardware e progettazione di collegamenti con STEP 7 • Manuale di conversione: STEP 7, da S5 a S7	Conoscenze di base per il personale tecnico: procedure per la realizzazione di compiti di controllo con STEP 7 e S7-300/400	6ES7810-4CA10-8EW0
Nozioni di riferimento di STEP 7 con • Manuali KOP/FUP/AWL per S7-300/400 • Funzioni standard e di sistema per S7-300/400 Volume 1 e Volume 2	Nozioni di riferimento sui linguaggi di programmazione KOP, FUP , AWL, nonché sulle funzioni standard e di sistema; perfezionamento delle conoscenze di base di STEP 7.	6ES7810-4CA10-8EW1

Guide online	Scopo	Numero di ordinazione
Guida a STEP 7	Conoscenze di base per la programmazione e la configurazione hardware con STEP 7	Parte del pacchetto software STEP 7
Guide di riferimento a AWL/KOP/FUP Guida di riferimento a SFB/SFC Guida di riferimento ai blocchi organizzativi	Guida di riferimento sensibile al contesto	Parte del pacchetto software STEP 7

Guida online

Come completamento del manuale è possibile avvalersi in fase operativa della dettagliata guida online integrata nel software.

Il sistema della guida è integrato nel software mediante differenti interfacce.

- La Guida al contesto offre informazioni sul contesto attuale, p. es. su una finestra di dialogo aperta o su una finestra attiva. È richiamabile con il pulsante "?" o con il tasto F1.
- Nel menu **?** sono disponibili diversi comandi: **Argomenti della Guida** apre l'indice della guida di STEP 7.
- Glossario relativo a tutte le applicazioni STEP 7 (Pulsante "Glosario").

Il presente manuale è un estratto della Guida a AWL. Manuale e guida online hanno quasi l'identica articolazione; è facile quindi passare dall'uno all'altra.

Ulteriore supporto

Per tutte le domande sull'uso dei prodotti descritti nel manuale, che non trovano risposta nella documentazione, rivolgersi al rappresentante Siemens locale.

Sito Internet delle rappresentanze Siemens:

http://www.siemens.com/automation/partner

Per la guida alla documentazione tecnica dei singoli prodotti e sistemi SIMATIC, consultare il sito:

http://www.siemens.com/simatic-tech-doku-portal

Il catalogo in linea e il sistema di ordinazione in linea si trova al sito:

http://mall.automation.siemens.com/

Centro di addestramento

Per facilitare l'approccio al sistema di automazione SIMATIC S7, la Siemens organizza corsi specifici. Rivolgersi a questo proposito al centro di addestramento locale più vicino o al centro di addestramento centrale di Norimberga.

Internet: http://www.sitrain.com

Technical Support

Per tutti i prodotti Industry Automation and Drive Technology è possibile rivolgersi al Technical Support

- mediante il modulo Web per la Support Request
  http://www.siemens.com/automation/support-request

Per ulteriori informazioni sul Technical Support, consultare in Internet il sito
http://www.siemens.com/automation/service

Service & Support in Internet

Aggiuntivamente alla documentazione, mettiamo a disposizione della clientela diversi servizi in linea all'indirizzo sottoindicato.

http://www.siemens.com/automation/service&support

Su questo sito si possono trovare:

- la Newsletter con informazioni sempre aggiornate sui prodotti;
- i documenti appropriati relativi alla ricerca in Service & Support;
- il Forum, luogo di scambio di informazioni tra utenti e personale specializzato di tutto il mondo;
- il partner di riferimento locali di Industry Automation and Drive Technology;
- informazioni su Riparazioni, pezzi di ricambio e consulenza.

Indice

1 Operazioni logiche combinatorie di bit

1.1 Sommario delle operazioni logiche combinatorie di bit

Descrizione

Le operazioni logiche combinatorie a bit operano con due cifre: 1 e 0. Queste due cifre costituiscono la base di un sistema numerico denominato sistema binario. Le due cifre 1 e 0 vengono denominate cifre binarie o bit. Nel mondo dei contatti e delle bobine, 1 sta a significare attivato o eccitato, e 0 sta per disattivato o diseccitato.

Le operazioni logiche combinatorie a bit interpretano gli stati di segnale di 1 e 0, e li combinano secondo la logica booleana per eseguire una varietà di funzioni. Queste combinazioni producono un risultato di 1 o 0 che è chiamato ”risultato logico combinatorio” (RLC).

Per le combinazioni logiche con operandi a bit vi sono le seguenti operazioni di base:

- U AND
- UN AND negato
- O OR
- ON OR negato
- X OR esclusivo
- XN OR esclusivo negato

Si possono adoperare le seguenti operazioni per combinare espressioni a parentesi:

- U(AND con diramazione
- UN(AND negato con apertura parentesi
- O(OR con apertura parentesi
- ON(OR negato con apertura parentesi
- X(OR esclusivo con apertura parentesi
- XN(OR esclusivo negato con apertura parentesi
-) Chiusura parentesi

Si può terminare una stringa logica a bit usando una delle seguenti operazioni:

- = Assegnazione
- R Resetta
- S Imposta

Si può adoperare una delle seguenti operazioni per modificare il risultato logico combinatorio (RLC):

- NOT Nega RLC
- SET Imposta RLC (=1)
- CLR Resetta RLC (=0)
- SAVE Salva RLC nel registro BIE

Altre operazioni rispondono ad una transizione di fronte di salita:

- FN Fronte di discesa
- FP Fronte di salita

1.2 U AND

Formato

U <bit>

Operando	Tipo dati	Area memoria
<bit>	BOOL	E, A, M, L, D, T, Z

Descrizione dell'operazione

U interroga il bit indirizzato sullo stato di segnale "1", e combina l'esito dell'interrogazione con RLC tramite AND.

Interrogazione dello stato dei bit della parola di stato:

L'operazione **AND** consente di interrogare la parola di stato anche direttamente. Avvalersi a tal fine dei seguenti operandi: ==0, <>0, >0, <0, >=0, <=0, UO, BIE, OS, OV.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	x	x	x	1

Esempio

Programma AWL		Schema logico di relè	
		Sbarra collettrice	
U	**E 1.0**	E 1.0 Stato di segnale 1	Contatto aperto a riposo
U	**E 1.1**	E 1.1 Stato di segnale 1	Contatto aperto a riposo
=	**A 4.0**	A 4.0 Stato di segnale 1	Bobina
Indica che il contatto è chiuso.			

1.3 UN AND negato

Formato

UN <bit>

Operando	Tipo dati	Area memoria
<bit>	BOOL	E, A, M, L, D, T, Z

Descrizione dell'operazione

UN interroga il bit indirizzato sullo stato di segnale "0", e combina l'esito dell'interrogazione con RLC tramite AND.

Interrogazione dello stato dei bit della parola di stato:

L'operazione **AND negato** consente di interrogare la parola di stato anche direttamente. Avvalersi a tal fine dei seguenti operandi: ==0, <>0, >0, <0, >=0, <=0, UO, BIE, OS, OV.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	x	x	x	1

Esempio

Programma AWL		Schema logico di relè
		Sbarra collettrice
U	**E 1.0**	E 1.0 Stato di segnale 0 — Contatto normalm. aperto
UN	**E 1.1**	E 1.1 Stato di segnale 1 — Contatto normalm. chiuso
=	**A 4.0**	A 4.0 Stato di segnale 0 — Bobina

1.4 O OR

Formato

O <bit>

Operando	Tipo dati	Area memoria
<bit>	BOOL	E, A, M, L, D, T, Z

Descrizione dell'operazione

O interroga il bit indirizzato sullo stato di segnale "1", e combina l'esito dell'interrogazione con RLC tramite OR.

Interrogazione dello stato dei bit della parola di stato:

L'operazione **OR** consente di interrogare la parola di stato anche direttamente. Avvalersi a tal fine dei seguenti operandi: ==0, <>0, >0, <0, >=0, <=0, UO, BIE, OS, OV.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	0	x	x	1

Esempio

Programma AWL		Schema logico di relè
		Sbarra collettrice
O	**E 1.0**	E 1.0 Stato di segnale 1 Contatto aperto — E 1.1 Stato di segnale 0 Contatto aperto
O	**E 1.1**	
=	**A 4.0**	A 4.0 Stato di segnale 1 — Bobina
		Indicazione di contatto chiuso.

1.5 ON OR negato

Formato

ON <bit>

Operando	Tipo dati	Area memoria
<bit>	BOOL	E, A, M, L, D, T, Z

Descrizione dell'operazione

ON interroga il bit indirizzato sullo stato di segnale "0", e combina l'esito dell'interrogazione con RLC tramite OR.

Interrogazione dello stato dei bit della parola di stato:

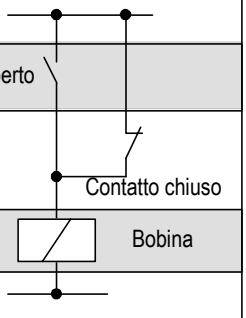
L'operazione **OR negato** consente di interrogare la parola di stato anche direttamente. Avvalersi a tal fine dei seguenti operandi: ==0, <>0, >0, <0, >=0, <=0, UO, BIE, OS, OV.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	0	x	x	1

Esempio

Programma AWL		Schema logico di relè
		Sbarra collettrice
O	**E 1.0**	E 1.0 Stato di segnale 0
ON	**E 1.1**	E 1.1 Stato di segnale 1
=	**A 4.0**	A 4.0 Stato di segnale 1



1.6 X OR esclusivo

Formato

X <bit>

Operando	Tipo dati	Area memoria
<bit>	BOOL	E, A, M, L, D, T, Z

Descrizione dell'operazione

X interroga il bit indirizzato sullo stato di segnale "1", e combina l'esito dell'interrogazione con RLC tramite OR esclusivo.

È possibile utilizzare la funzione OR esclusivo anche più volte di seguito. Il risultato logico combinatorio condiviso sarà pertanto "1", se un numero dispari di operandi interrogati dà come risultato dell'interrogazione "1".

Interrogazione dello stato dei bit della parola di stato:

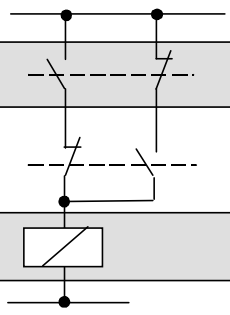
L'operazione **OR esclusivo** consente di interrogare la parola di stato anche direttamente. Avvalersi a tal fine dei seguenti operandi: ==0, <>0, >0, <0, >=0, <=0, UO, BIE, OS, OV.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	x	-	-	-	-	0	x	x	1

Esempio

Programma AWL		Schema logico di relè
		Sbarra collettrice
X	E 1.0	Contatto E 1.0
X	E 1.1	Contatto E 1.1
=	A 4.0	A 4.0 Bobina



1.7 XN OR esclusivo negato

Formato

XN <bit>

Operando	Tipo dati	Area memoria
<bit>	BOOL	E, A, M, L, D, T, Z

Descrizione dell'operazione

XN interroga il bit indirizzato sullo stato di segnale "0", e combina il risultato dell'interrogazione con RLC tramite OR esclusivo negato.

Interrogazione dello stato dei bit della parola di stato:

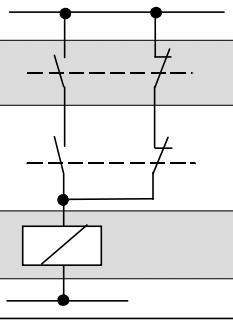
L'operazione **OR esclusivo negato** consente di interrogare la parola di stato anche direttamente. Avvalersi a tal fine dei seguenti operandi: ==0, <>0, >0, <0, >=0, <=0, UO, BIE, OS, OV.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	x	-	-	-	-	0	x	x	1

Esempio

Programma AWL		Schema logico di relè
		Sbarra collettrice
X	**E 1.0**	Contatto E 1.0
XN	**E 1.1**	Contatto E 1.1
=	**A 4.0**	A 4.0 Bobina



1.8 O AND prima di OR

Formato

O

Descrizione dell'operazione

L'operazione **O** esegue, secondo la regola AND prima di OR, le combinazioni OR su AND.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	x	1	-	x

Esempio

Programma AWL		Schema logico di relè
		Sbarra collettrice
U	**E 0.0**	E 0.0
U	**M 10.0**	M 10.0
		E 0.2
		M 10.1
O		
U	**E 0.2**	
U	**M 0.3**	
		M 0.3
O	**M 10.1**	
=	**A 4.0**	A 4.0 Bobina

1.9 U(AND con apertura parentesi

Formato

U(

Descrizione dell'operazione

U((AND con apertura parentesi) memorizza i bit RLC e OR nonché l'identificazione dell'operazione nello stack di parentesi. Lo stack può riportare un massimo di 7 registrazioni.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	0	1	-	0

Esempio

Programma AWL		Schema logico di relè
		Sbarra collettrice
U(		
O	**E 0.0**	E 0.0
O	**M 10.0**	M 10.0
)		
U(		
O	**E 0.2**	E 0.2
O	**M 10.3**	M 10.3
)		
U	**M 10.1**	M 10.1
=	**A 4.0**	A 4.0 Bobina

1.10 UN(AND negato con apertura parentesi

Formato

UN(

Descrizione dell'operazione

UN((AND negato con apertura parentesi) memorizza i bit RLC e OR nonché l'identificazione dell'operazione nello stack di parentesi. Lo stack può riportare un massimo di 7 registrazioni.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	0	1	-	0

1.11 O(OR con apertura parentesi

Formato

O(

Descrizione dell'operazione

O((OR con apertura parentesi) memorizza i bit RLC e OR nonché l'identificazione dell'operazione nello stack di parentesi. Lo stack può riportare un massimo di 7 registrazioni.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	0	1	-	0

1.12 ON(OR negato con apertura parentesi

Formato

ON(

Descrizione dell'operazione

ON((OR negato con apertura parentesi) memorizza i bit RLC e OR nonché l'identificazione dell'operazione nello stack di parentesi. Lo stack può riportare un massimo di 7 registrazioni.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	0	1	-	0

1.13 X(OR esclusivo con apertura parentesi

Formato

X(

Descrizione dell'operazione

X((OR esclusivo con apertura parentesi) memorizza i bit RLC, e OR, nonché l'identificazione dell'operazione nello stack di parentesi. Lo stack può riportare un massimo di 7 registrazioni.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	x	-	-	-	-	0	1	-	0

1.14 XN(OR esclusivo negato con apertura parentesi

Formato

XN(

Descrizione dell'operazione

XN((OR esclusivo negato con apertura parentesi) memorizza i bit RLC, e OR, nonché l'identificazione dell'operazione nello stack di parentesi. Lo stack può riportare un massimo di 7 registrazioni.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	x	-	-	-	-	0	1	-	0

1.15) Chiusura parentesi

Formato

)

Descrizione dell'operazione

) (Chiusura parentesi) cancella una registrazione dallo stack di parentesi, ripristina il bit OR, combina il risultato logico combinatorio (RLC) riportato nella registrazione stack con il risultato logico combinatorio attuale secondo l'identificazione dell'operazione, e trasmette il risultato a RLC. Se l'identificazione dell'operazione è AND oppure AND negato, viene considerato anche il bit OR.

Operazioni di diramazione:

- U(AND con apertura parentesi
- UN(AND negato con apertura parentesi
- ON(OR negato con apertura parentesi
- X(OR esclusivo con apertura parentesi
- XN(OR esclusivo negato con apertura parentesi

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	x	1	x	1

Esempio

Programma AWL		Schema logico di relè
		Sbarra collettrice
U(**O** **O** **)**	**E 0.0** **M 10.0**	E 0.0 M 10.0
U(**O** **O** **)**	**E 0.2** **M 10.3**	E 0.2 M 10.3
U	**M 10.1**	M 10.1
=	**A 4.0**	A 4.0 Bobina

1.16 = Assegna

Formato

= <bit>

Operando	Tipo dati	Area memoria
<bit>	BOOL	E, A, M, L, D

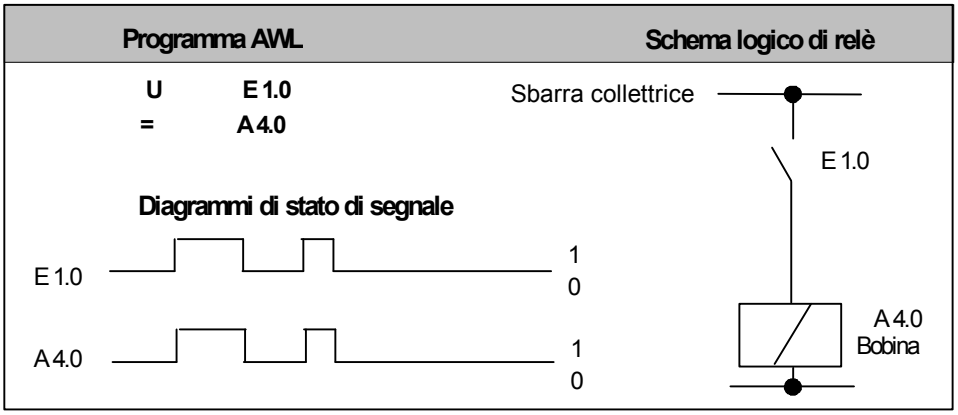
Descrizione dell'operazione

= <bit> se il Relè Master Control è attivato (MCR = 1), scrive RLC nel bit indirizzato. Se MCR = 0, il valore scritto nel bit indirizzato non è RLC, bensì "0".

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	0	x	-	0

Esempio



1.17 R Resetta

Formato

R <bit>

Operando	Tipo dati	Area memoria
<bit>	BOOL	E, A, M, L, D

Descrizione dell'operazione

R (Resetta) scrive il valore "0" nel bit indirizzato se RLC = 1, ed il Relè Master Control è attivato (MCR = 1). Se MCR = 0, il bit indirizzato resta inalterato.

Parola di stato

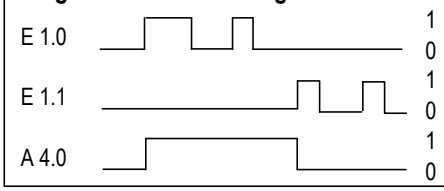
	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	0	x	-	0

Esempio

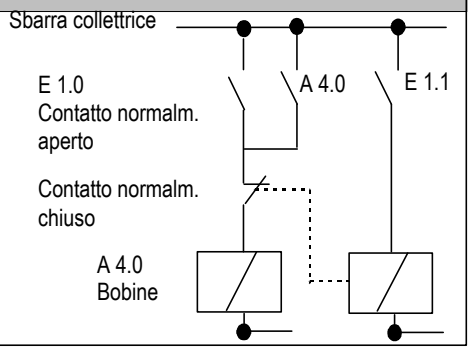
Programma AWL

```
U   E 1.0
S   A 4.0
U   E 1.1
R   A 4.0
```

Diagrammi di stato di segnale



Schema logico di relè



1.18 S Imposta

Formato

S <bit>

Operando	Tipo dati	Area memoria
<bit>	BOOL	E, A, M, L, D

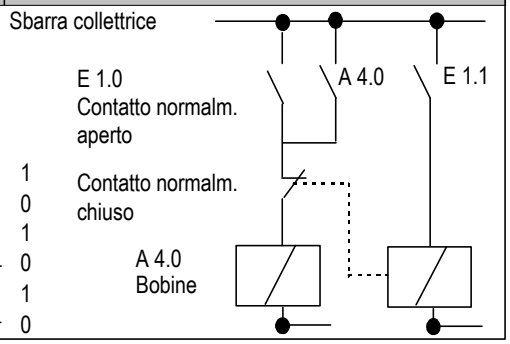
Descrizione dell'operazione

S (Imposta) scrive il valore "1" nel bit indirizzato, se RLC = 1 ed il Relè Master Control è attivato (MCR = 1). Se MCR = 0, il bit indirizzato resta inalterato.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	0	x	-	0

Esempio

Programma AWL	Schema logico di relè
U E 1.0 **S A 4.0** **U E 1.1** **R A 4.0** **Diagrammi di stato di segnale** E 1.0 E 1.1 A 4.0	

1.19 NOT Nega RLC

Formato

NOT

Descrizione dell'operazione

NOT nega il risultato logico combinatorio (RLC).

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	1	x	-

1.20 SET Imposta RLC (=1)

Formato

SET

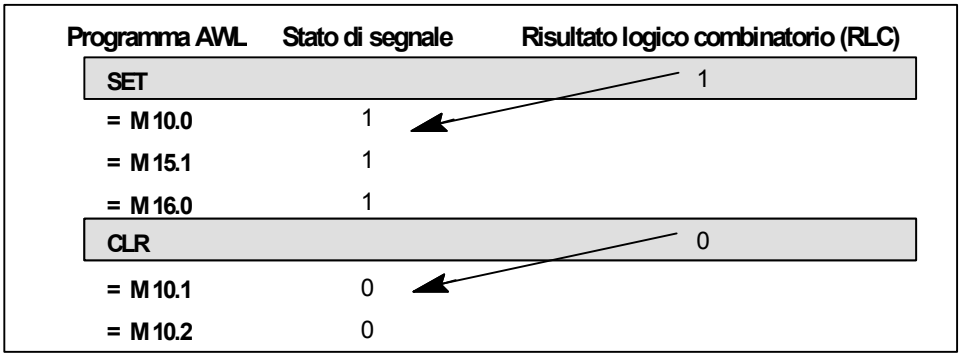
Descrizione dell'operazione

SET setta RLC allo stato di segnale "1".

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	0	1	1	0

Esempio


Programma AWL
Stato di segnale
Risultato logico combinatorio (RLC)
SET
1
= M 10.0
1
= M 15.1
1
= M 16.0
1
CLR
0
= M 10.1
0
= M 10.2
0

1.21 CLR Resetta RLC (=0)

Formato

CLR

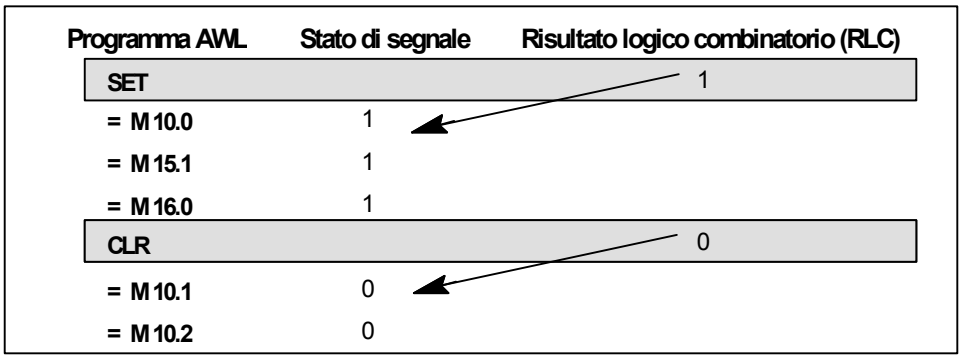
Descrizione dell'operazione

CLR setta RLC allo stato di segnale "0".

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	0	0	0	0

Esempio



1.22 SAVE Salva RLC nel registro BIE

Formato

SAVE

Descrizione dell'operazione

SAVE salva RLC nel bit BIE. Il bit di prima interrogazione/ER non viene resettato.
Per questo motivo, con una combinazione logica AND, anche lo stato del bit BIE viene combinato nel prossimo segmento.

L'impiego di SAVE e una successiva interrogazione del bit BIE nello stesso blocco o in blocchi sottostanti sono sconsigliati in quanto il bit BIE potrebbe subire una modifica dovuta alle numerose operazioni intermedie. È invece opportuno utilizzare l'operazione SAVE prima di abbandonare un blocco poiché, in questo modo, l'uscita ENO (=bit BIE) viene impostata sul valore del bit RLC permettendo all'utente di annettere una gestione degli errori del blocco.

Parola di stato

	BIE	**A1**	**A0**	**OV**	**OS**	**OR**	**STA**	**RLC**	**/ER**
Scrive:	x	-	-	-	-	-	-	-	-

1.23 FN Fronte di discesa

Formato

FN <bit>

Operando	Tipo dati	Area memoria	Descrizione
<bit>	BOOL	E, A, M, L, D	Merker del fronte, memorizza lo stato di segnale precedente di RLC

Descrizione dell'operazione

FN <bit> (fronte di discesa) rileva un fronte di discesa quando RLC cambia da "1" a "0", e indica tale transizione con RLC = 1.

Durante ogni ciclo di programma, lo stato di segnale del bit RLC viene confrontato con il bit RLC del ciclo precedente, per constatare eventuali variazioni dello stato. Per eseguire tale confronto, è necessario che lo stato del bit RLC precedente sia stato memorizzato nell'indirizzo del merker del fronte (**<bit>**). Se lo stato di segnale attuale del bit RLC differisce dallo stato precedente ("1") (rilevamento di un fronte negativo), il valore assunto dal bit RLC a conclusione dell'operazione è "1".

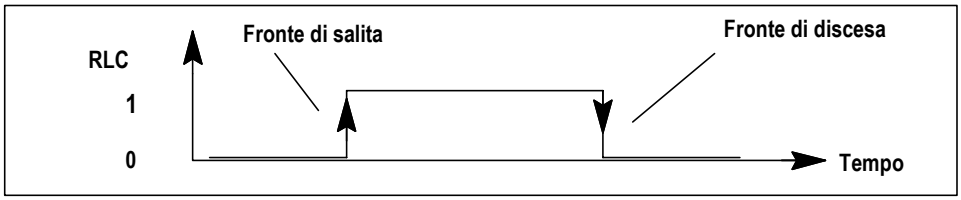
Nota

Non è logico eseguire questa operazione se il bit da controllare si trova nell'immagine di processo. I dati locali di un blocco, infatti, sono validi solo durante il suo tempo di esecuzione.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	0	x	x	1

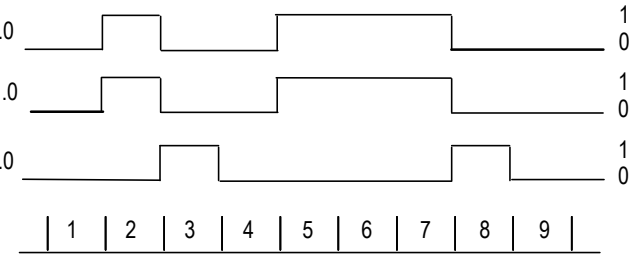
Definizione



Esempio

Quando il controllore programmabile rileva un fronte di discesa al contatto E 1.0, esso attiva l'uscita A 4.0 per un ciclo di OB1.

Programma AWL		Diagrammi di stato di segnale
U	**E 1.0**	E 1.0
FN	**M 1.0**	M 1.0
=	**A 4.0**	A 4.0
N. ciclo scansione OB1:		1 2 3 4 5 6 7 8 9



1.24 FP Fronte di salita

Formato

FP <bit>

Operando	Tipo dati	Area memoria	Descrizione
<bit>	BOOL	E, A, M, L, D	Merker del fronte, memorizza lo stato di segnale precedente di RLC

Descrizione dell'operazione

FP <bit> (Fronte di salita) rileva un fronte di salita quando RLC cambia da "0" a "1", ed indica tale transizione con RLC = 1.

Durante ogni ciclo di programma, lo stato di segnale del bit RLC viene confrontato con lo stato di segnale del bit RLC del ciclo precedente, per constatare eventuali variazioni di stato. Per eseguire tale confronto è necessario che lo stato RLC precedente sia stato memorizzato nell'indirizzo del merker del fronte (**<bit>**). Se lo stato di segnale attuale del bit RLC differisce dallo stato precedente ("0") (rilevamento di un fronte di salita), il valore assunto dal bit RLC a conclusione dell'operazione è "1".

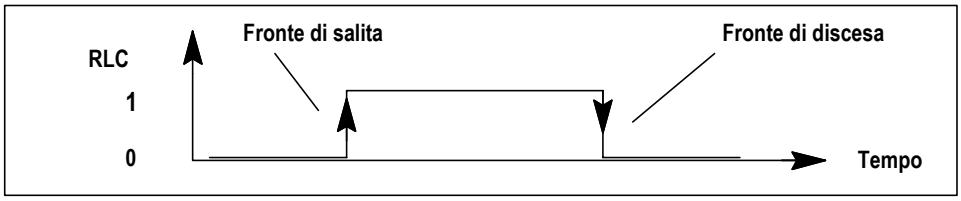
Nota

Non è logico eseguire questa operazione se il bit da controllare si trova nell'immagine di processo. I dati locali di un blocco, infatti, sono validi solo durante il suo tempo di esecuzione.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	0	x	x	1

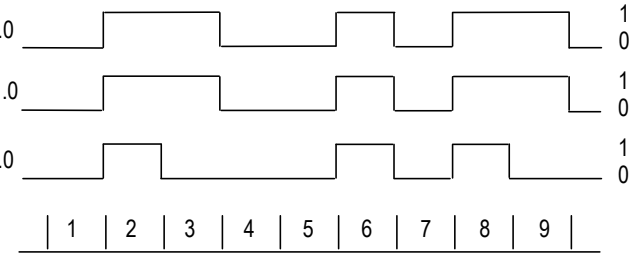
Definizione



Esempio

Quando il controllore programmabile rileva un fronte di discesa al contatto E 1.0, esso attiva l'uscita A 4.0 per un ciclo di OB1.

Programma AWL		Diagrammi di stato di segnale
U	**E 1.0**	E 1.0
FP	**M 1.0**	M 1.0
=	**A 4.0**	A 4.0
N. ciclo di scansione OB1:		1 2 3 4 5 6 7 8 9



2 Operazioni di confronto

2.1 Sommario delle operazioni di confronto

Descrizione

I valori numerici vengono caricati negli accumulatori 1 e 2. L'operazione di confronto confronta il valore di ACCU 2 con il valore di ACCU 1 secondo i criteri segueni:

== ACCU 1 è uguale a ACCU 2
<> ACCU 1 è diverso da ACCU 2
> ACCU 1 è maggiore di ACCU 2
< ACCU 1 è minore di ACCU 2
>= ACCU 1 è maggiore di o uguale a ACCU 2
<= ACCU 1 è minore di o uguale a ACCU 2

Il risultato del confronto è una cifra binaria, ovvero un 1 o uno 0. Un 1 indica che il risultato del confronto è vero; uno 0 indica che il risultato del confronto è falso. Questo risultato viene memorizzato nel bit di risultato logico combinatorio (bit RLC). I bit di stato A1 e A0 indicano le relazioni "minore di", "uguale a" o "maggiore di".

Sono disponibili le seguenti operazioni di confronta:

- ? I Confronta numeri interi (a 16 bit)
- ? D Confronta numeri interi (a 32 bit)
- ? R Confronta numeri in virgola mobile (a 32 bit)

2.2 ? I Confronta numeri interi (a 16 bit)

Formato

==I, <>I, >I, <I, >=I, <=I

Descrizione dell'operazione

Le operazioni Confronta numeri interi (a 16 bit) confrontano il contenuto dell'accumulatore 2-L con il contenuto dell'accumulatore 1-L. I contenuti degli accumulatori 2-L e 1-L vengono interpretati quali numeri interi (a 16 bit). Il risultato del confronto viene visualizzato dal RLC e dai bit rilevanti della parola di stato. RLC = 1 indica che il risultato del confronto è vero. RLC = 0 indica che il risultato del confronto è falso. I bit di stato A1 e A0 indicano la relazione "minore di", "uguale a" o"maggiore di".

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	x	x	0	-	0	x	x	1

Valori del RLC

Operazione di confronto eseguita	RLC se ACCU 2 > ACCU 1	RLC se ACCU 2 = ACCU 1	RLC se ACCU 2 < ACCU 1
==I	0	1	0
<>I	1	0	1
>I	1	0	0
<I	0	0	1
>=I	1	1	0
<=I	0	1	1

Esempio

```
AWL             Spiegazione
L     MW10      //Carica il contenuto di MW10 (numero intero a 16 bit).
L     EW24      //Carica il contenuto di EW24 (numero intero a 16 bit).
>I              //Confronta se ACCU 2-L (MW10) è maggiore (>) di ACCU 1-L (EW24).
=     M 2.0     //RLC = 1 se MW10 > EW24.
```

2.3 ? D Confronta numeri interi (a 32 bit)

Formato

==D, <>D, >D, <D, >=D, <=D

Descrizione dell'operazione

Le operazioni Confronta numeri interi (a 32 bit) confrontano il contenuto dell'accumulatore 2 con il contenuto dell'accumulatore 1. I contenuti degli accumulatori 2 e 1 vengono valutati come numeri interi (a 32 bit). Il risultato del confronto viene visualizzato dal RLC e dai bit rilevanti della parola di stato. RLC = 1 indica che il risultato del confronto è vero. RLC = 0 indica che il risultato del confronto è falso. I bit di stato A1 e A0 indicano le relazioni "minore di", "uguale a" o "maggiore di".

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	x	x	0	-	0	x	x	1

Valori del RLC

Operazione di confronto eseguita	RLC se ACCU 2 > ACCU 1	RLC se ACCU 2 = ACCU 1	RLC se ACCU 2 < ACCU 1
==D	0	1	0
<>D	1	0	1
>D	1	0	0
<D	0	0	1
>=D	1	1	0
<=D	0	1	1

Esempio

```
AWL             Spiegazione
L      MD10     //Carica il contenuto di MD10 (numero intero a 32 bit).
L      ED24     //Carica il contenuto di ED24 (numero intero a 32 bit).
>D              //Confronta se ACCU 2 (MD10) è maggiore (>) di ACCU 1 (ED24).
=      M 2.0    //RLC = 1, se MD10 > ED24.
```

2.4 ? R Confronta numeri in virgola mobile (a 32 bit)

Formato

==R, <>R, >R, <R, >=R, <=R

Descrizione dell'operazione

Le operazioni Confronta i numeri in virgola mobile (a 32 bit, IEEE 754) confrontano il contenuto dell'accumulatore 2 con il contenuto dell'accumulatore 1. I contenuti degli accumulatori 1 e 2 vengono interpretati quali numeri in virgola mobile (a 32 bit, IEEE 754). Il risultato del confronto viene indicato dal RLC e dai bit rilevanti della parola di stato. RLC = 1 indica che il risultato del confronto è vero. RLC = 0 indica che il risultato del confronto è sbagliato. I bit di stato A1 e A0 indicano le relazioni "minore di", "uguale a" o "maggiore di".

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	x	x	x	x	0	x	x	1

Valori del RLC

Operazione di confronto eseguita	RLC se ACCU 2 > ACCU 1	RLC se ACCU 2 = ACCU 1	RLC se ACCU 2 < ACCU 1
==R	0	1	0
<>R	1	0	1
>R	1	0	0
<R	0	0	1
>=R	1	1	0
<=R	0	1	1

Esempio

```
AWL             Spiegazione
L     MD10      //Carica il contenuto di MD10 (numero in virgola mobile).
L     1.359E+02 //Carica la costante 1.35E+02.
>R              //Confronta se ACCU 2 (MD10) è maggiore (>) di ACCU 1 (1,359E+02).
=     M 2.0     //RLC = 1 se MD10 > 1,359E+02.
```

3 Operazioni di conversione

3.1 Sommario delle operazioni di conversione

Descrizione

Per convertire numeri decimali in codice binario (BCD) e numeri interi in altri tipi di numeri si possono adoperare le seguenti operazioni:

- BTI Converti numero BCD in numero intero (a 16 bit)
- ITB Converti numero intero (a 16 bit) in numero BCD
- BTD Converti numero BCD in numero intero (a 32 bit)
- ITD Converti numero intero (a 16 bit) in numero intero (a 32 bit)
- DTB Converti numero intero (a 32 bit) in numero BCD
- DTR Converti numero intero (a 32 bit) in numero in virgola mobile (a 32 bit, IEEE 754)

Si possono adoperare le seguenti operazioni per formare i complementi dei numeri interi o per cambiare il segno di un numero in virgola mobile:

- INVI Complemento a 1 di numero intero (a 16 bit)
- INVD Complemento a 1 di numero intero (a 32 bit)
- NEGI Complemento a 2 di numero intero (a 16 bit)
- NEGD Complemento a 2 di numero intero (a 32 bit)
- NEGR Complemento a 2 di numero in virgola mobile (a 32 bit, IEEE 754)

L'utente può adoperare le seguenti operazioni per invertire l'ordine dei byte nella parola bassa di ACCU 1 o nell'intero accumulatore:

- TAW Cambia sequenza di byte in ACCU 1 (a 16 bit)
- TAD Cambia sequenza di byte in ACCU 1 (a 32 bit)

Per convertire un numero IEEE 754 in virgola mobile a 32 bit in ACCU 1 in un numero intero a 32 bit (doppio numero intero) si può adoperare una delle seguenti operazioni. Le singole operazioni differiscono l'una dall'altra nel metodo di arrotondamento:

- RND Arrotonda al numero intero
- TRUNC Arrotonda senza resto
- RND+ Arrotonda al numero intero superiore (a 32 bit)
- RND- Arrotonda al numero intero inferiore (a 32 bit)

3.2 BTI Converti numero BCD in numero intero (a 16 bit)

Formato

BTI

Descrizione dell'operazione

BTI (Converti un numero decimale in codice binario BDC a tre posizioni) interpreta il contenuto dell'accumulatore 1 quale numero decimale in codice binario (BCD) a tre posizioni, e lo converte in un numero intero (a 16 bit). Il risultato viene memorizzato nell'accumulatore 1-L. Gli accumulatori 1-H e 2 non vengono modificati.

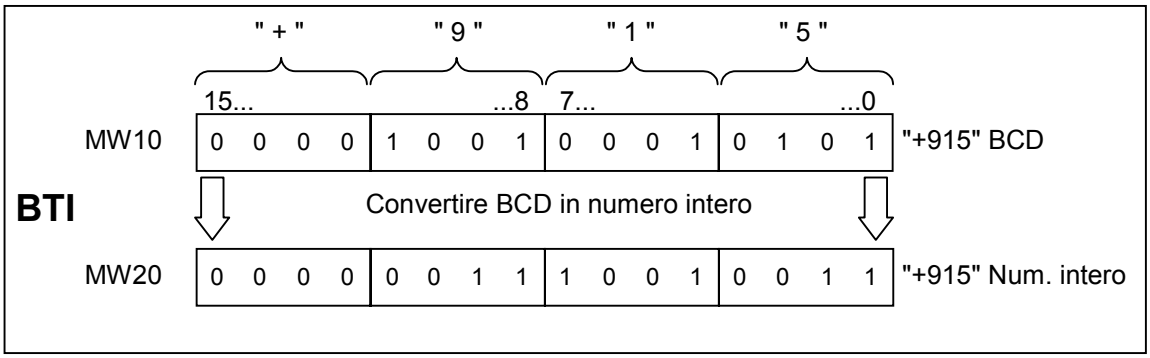
Numero BCD nell'accumulatore 1-L: sono consentiti valori compresi tra "-999" e "+999". I bit da 0 a 11 indicano il valore, ed il quindicesimo bit definisce il segno (0 = positivo, 1= negativo) del numero BCD. I bit da 12 a 14 non vengono utilizzati per la conversione. Se una posizione (4 bit) del numero BCD si trova nell'area non valida compresa tra 10 e 15, il tentativo di conversione determina un errore BCDF. In tal caso il controllore programmabile passa generalmente allo stato di funzionamento STOP. Avvalendosi di OB121 è tuttavia possibile programmare un'azione di risposta all'errore sincrono verificatosi.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio

```
AWL             Spiegazione
L     MW10      //Carica il numero BCD in ACCU 1-L.
BTI             //Converte il numero BCD in un numero intero, memorizza il risultato in
                //ACCU 1-L.
T     MW20      //Trasferisce il risultato (numero intero a 16 bit) a MW20.
```



3.3 ITB Converti numero intero (a 16 bit) in numero BCD

Formato

ITB

Descrizione dell'operazione

ITB (Converti numero intero (a 16 bit) in numero BCD) interpreta il contenuto dell'accumulatore 1-L come numero intero (a 16 bit) e lo converte in un numero decimale in codice binario a tre posizioni (BCD). Il risultato viene memorizzato in ACCU 1-L. I bit da 0 a 11 indicano il valore del numero BCD. I bit da 12 a 15 rappresentano lo stato del segno (0000 = positivo, 1111 = negativo) del numero BCD. Gli accumulatori 1-H e 2 non vengono modificati.

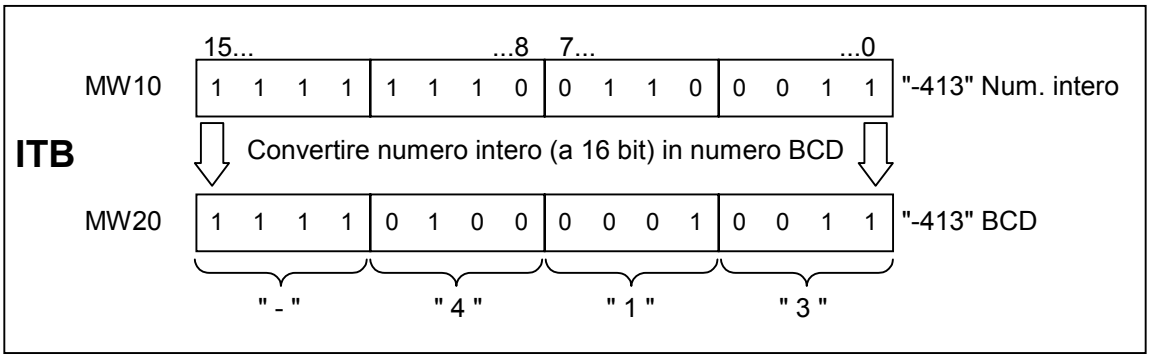
Il numero BCD può essere compreso tra "-999" e "+999". Se il numero non rientra nell'area di validità, i bit di stato OV e OS vengono settati a "1".

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	x	x	-	-	-	-

Esempio

```
AWL             Spiegazione
L      MW10     //Carica il numero intero in ACCU 1-L.
ITB             //Converte il numero intero (a 16 bit) in un numero BCD, memorizza il risultato
                //in ACCU 1-L.
T      MW20     //Trasferisce il risultato (numero BCD) a MW20.
```



3.4 BTD Converti numero BCD in numero intero (a 32 bit)

Formato

BTD

Descrizione dell'operazione

BTD (Converti un numero decimale in codice binario BCD a sette posizioni) interpreta il valore dell'accumulatore 1 come un numero decimale in codice binario a sette posizioni (BCD), e lo converte in un numero intero (a 32 bit). Il risultato viene memorizzato in ACCU 1. L'accumulatore 2 non viene modificato.

Numero BCD in ACCU 1: sono consentiti valori compresi tra "-9999999" e "+9999999". I bit da 0 a 27 indicano il valore e il trentunesimo bit definisce il segno (0 = positivo, 1 = negativo) del numero BCD. I bit compresi tra 28 e 30 non vengono utilizzati per la conversione.

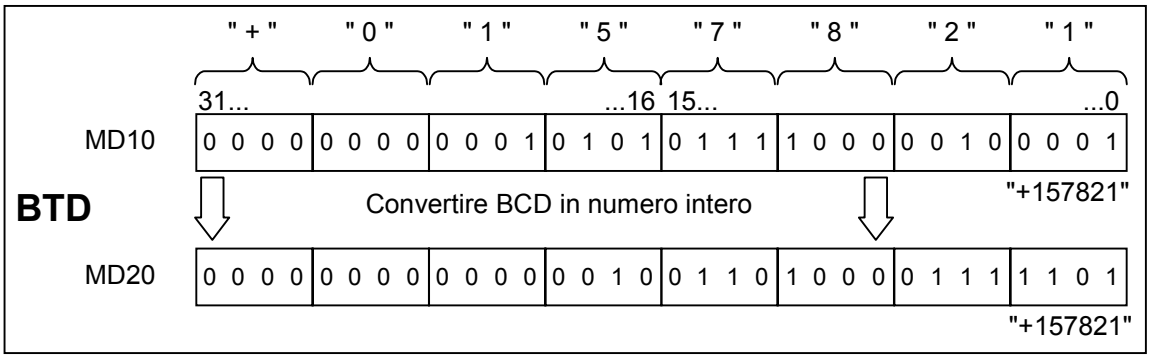
Se un numero decimale (nella rappresentazione BCD una tetrade di 4 bit) si trova nell'area non valida compresa tra 10 e 15, il tentativo di conversione determina un errore BCDF. In tal caso il controllore programmabile passa generalmente allo stato di funzionamento STOP. Avvalendosi di OB121 è tuttavia possibile programmare un'azione di risposta all'errore sincrono verificatosi.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio

```
AWL             Spiegazione
L     MD10      //Carica il numero BCD in ACCU 1.
BTD             //Converte il numero BCD in un numero intero, memorizza il risultato in
                //ACCU 1.
T     MD20      //Trasferisce il risultato (numero intero a 32 bit) a MD20.
```



3.5 ITD Converti numero intero (a 16 bit) in numero intero (a 32 bit)

Formato

ITD

Descrizione dell'operazione

ITD (Converti numero intero a 16 bit in numero intero a 32 bit) interpreta il contenuto dell'accumulatore 1-L come numero intero (a 16 bit), e lo converte in un numero intero (a 32 bit). Il risultato viene memorizzato nell'accumulatore 1. L'accumulatore 2 non viene modificato.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio

```
AWL             Spiegazione
L      MW12     //Carica il numero intero (a 16 bit) in ACCU 1.
ITD             //Converte il numero intero (a 16 bit) in un numero intero (a 32 bit), memorizza
                //il risultato in ACCU 1.
T      MD20     //Trasferisce il risultato (numero intero a 32 bit) a MD20.
```

Esempio: MW12 = "-10" (numero intero a 16 bit)

Contenuto	ACCU1-H				ACCU1-L			
Bit	31 . . .	. .	. .	. . . 16	15 . . .	. .	. .	. . . 0
prima dell'esecuzione di **ITD**	XXXX	XXXX	XXXX	XXXX	1111	1111	1111	0110
dopo l'esecuzione di **ITD**	1111	1111	1111	1111	1111	1111	1111	0110
	(X = 0 o 1, bit non necessari per la conversione)							

3.6 DTB Converti numero intero (a 32 bit) in numero BCD

Formato

DTB

Descrizione dell'operazione

DTB (Converti un numero intero (a 32 bit) in numero BCD) interpreta il contenuto dell'accumulatore 1 come numero intero (a 32 bit), e lo converte in un numero decimale in codice binario a sette posizioni. Il risultato viene memorizzato nell'accumulatore 1. I bit da 0 a 27 indicano il valore del numero BCD. I bit da 28 a 31 rappresentano lo stato del segno del numero BCD (0000 = positivo, 1111 = negativo). L'accumulatore 2 resta inalterato.

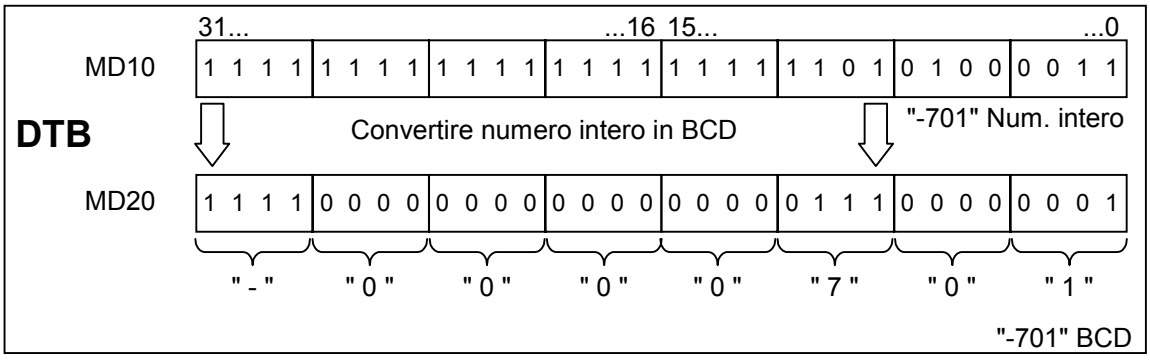
Il numero BCD può essere compreso tra "-9999999" e "+9999999". Se il numero non rientra nell'area di validità, i bit di stato OV e OS vengono settati a "1".

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	x	x	-	-	-	-

Esempio

```
AWL             Spiegazione
L     MD10      //Carica il numero intero (a 32 bit) in ACCU 1.
DTB             //Converte il numero intero (a 32 bit) in un numero BCD, memorizza il risultato
                //in ACCU 1.
T     MD20      //Trasferisce il risultato (numero BCD) a MD20.
```



3.7 DTR Converti numero intero (a 32 bit) in numero in virgola mobile (a 32 bit, IEEE 754)

Formato

DTR

Descrizione dell'operazione

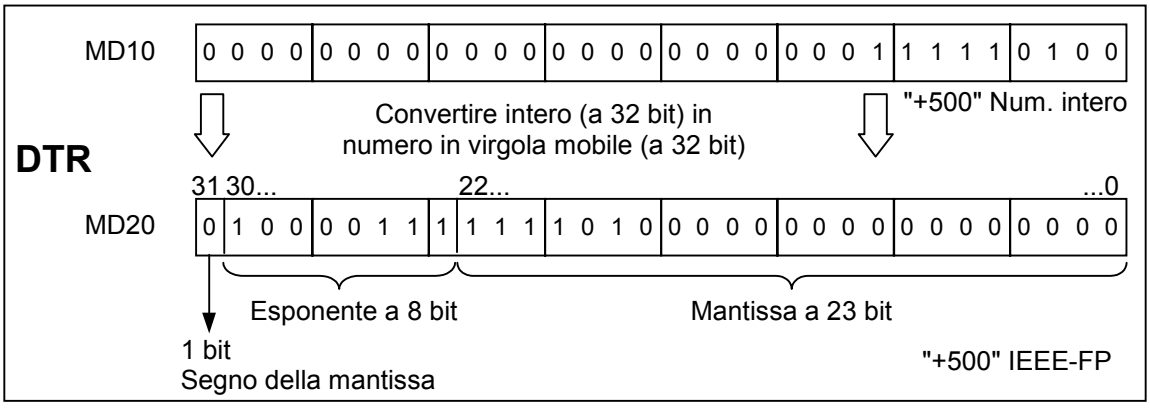
DTR (Converti un numero intero (a 32 bit) in numero in virgola mobile (a 32 bit, IEEE 754) interpreta il contenuto dell'accumulatore 1 come numero intero (a 32 bit), e lo converte in un numero in virgola mobile (a 32 bit, IEEE 754). Se necessario, l'operazione arrotonda il risultato (un numero intero a 32 bit presenta un maggior grado di precisione di un numero in virgola mobile a 32 bit, IEEE 754). Il risultato viene memorizzato nell'accumulatore 1.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio

```
AWL             Spiegazione
L     MD10      //Carica il numero intero (a 32 bit) in ACCU 1.
DTR             //Converte il numero intero (a 32 bit) in un numero in virgola mobile
                //(a 32 bit, IEEE- FP), memorizza il risultato in ACCU 1.
T     MD20      //Trasferisce il risultato (numero BCD) a MD20.
```



3.8 INVI Complemento a 1 di numero intero (a 16 bit)

Formato

INVI

Descrizione dell'operazione

INVI (Complemento a 1 di numero intero (a 16 bit) costituisce il complemento a uno del valore a 16 bit nell'accumulatore 1-L. La creazione del complemento a uno determina l'inversione di tutti I singoli bit: gli zeri vengono cioè sostituiti dagli uno, e gli uno dagli zeri. Il risultato viene memorizzato nell'accumulatore 1-L.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempi

```
AWL             Spiegazione
L      EW8      //Carica il valore in ACCU 1-L.
INVI            //Crea il complemento a uno (a 16 bit).
T      MW10     //Trasferisce il risultato in MW10.
```

Contenuto	ACCU1-L			
Bit	15 . . .	. .	. .	. . . 0
prima dell'esecuzione di **INVI**	0110	0011	1010	1110
dopo l'esecuzione di **INVI**	1001	1100	0101	0001

3.9 INVD Complemento a 1 di numero intero (a 32 bit)

Formato

INVD

Descrizione dell'operazione

INVD (Complemento a 1 di numero intero (a 32 bit)) costituisce il complemento a uno del valore a 32 bit nell'accumulatore 1. La creazione del complemento a uno determina l'inversione di ogni singolo bit: gli zeri vengono cioè sostituiti dagli uno, e gli uno dagli zeri. Il risultato viene memorizzato nell'accumulatore 1.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio

```
AWL             Spiegazione
L      ED8      //Carica il valore in ACCU 1.
INVD            //Crea il complemento a uno (a 32 bit).
T      MD10     //Trasferisce il risultato a MW10.
```

Contenuto	ACCU1-H				ACCU1-L			
Bit	31 . . .	. .	. .	. . . 16	15 . . .	. .	. .	. . . 0
prima dell'esecuzione di **INVD**	0110	1111	1000	1100	0110	0011	1010	1110
dopo l'esecuzione di **INVD**	1001	0000	0111	0011	1001	1100	0101	0001

3.10 NEGI Complemento a 2 di numero intero (a 16 bit)

Formato

NEGI

Descrizione dell'operazione

NEGI (Complemento a 2 di numero intero (a 16 bit)) costituisce il complemento a 2 del valore a 16 bit nell'accumulatore 1-L. La creazione del complemento a 2 determina l'inversione di tutti i singoli bit: gli 0 vengono cioè sostituiti dagli 1 ed gli 1 dagli 0. Viene quindi addizionato "1". Il risultato viene memorizzato nell'accumulatore 1-L. L'operazione Complemento a 2 di numero intero (a 16 bit) corrisponde ad una moltiplicazione per "-1". I bit di stato A1, A0, OS, e OV, vengono settati conformemente all'esito dell'operazione.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	x	x	x	x	-	-	-	-

Settaggio dei bit di stato	A1	A0	OV	OS
Risultato = 0	0	0	0	-
-32768 <= Risultato <= -1	0	1	0	-
32767 >= Risultato >= 1	1	0	0	-
Risultato = 2768	0	1	1	1

Esempio

```
AWL             Spiegazione
L     EW8       //Carica il valore in ACCU 1-L.
NEGI            //Crea il complemento a due (a 16 bit).
T     MW10      //Trasferisce il risultato a MW10.
```

Contenuto	ACCU1-L			
Bit	15 . . .	. .	. .	. . . 0
prima dell'esecuzione di **NEGI**	0101	1101	0011	1000
dopo l'esecuzione di **NEGI**	1010	0010	1100	1000

3.11 NEGD Complemento a 2 di numero intero (a 32 bit)

Formato

NEGD

Descrizione dell'operazione

NEGD (Complemento a 2 di numero intero (a 32 bit) costituisce il complemento a 2 del valore di 32 bit nell'accumulatore 1. La creazione del complemento a 2 determina l'inversione di tutti i singoli bit: gli 0 vengono cioè sostituiti dagli 1 e gli 1 dagli 0. Viene quindi addizionato "1". Il risultato viene memorizzato nell'accumulatore 1. L'operazione Complemento a 2 di numero intero (a 32 bit) corrisponde ad una moltiplicazione per -1. I bit di stato A1, A0, OS e OV vengono settati conformemente all'esito dell'operazione.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	x	x	x	x	-	-	-	-

Settaggio dei bit di stato	A1	A0	OV	OS
Risultato = 0	0	0	0	-
-2 147 483 647 <= Risultato <= -1	0	1	0	-
2 147 483 647 >= Risultato >= 1	1	0	0	-
Risultato = -2 147 483 648	0	1	1	1

Esempi

```
AWL             Spiegazione
L     ED8       //Carica il valore in ACCU 1.
NEGD            //Crea il complemento a due (a 32 bit).
T     MD10      //Trasferisce il risultato a MD10.
```

Contenuto	ACCU1-H				ACCU1-L			
Bit	31 . . .	. .	. .	. . . 16	15 . . .	. .	. .	. . . 0
prima dell'esecuzione di **NEGD**	0101	1111	0110	0100	0101	1101	0011	1000
dopo l'esecuzione di **NEGD**	1010	0000	1001	1011	1010	0010	1100	1000

3.12 NEGR Complemento a 2 di numero in virgola mobile (a 32 bit, IEEE 754)

Formato

NEGR

Descrizione dell'operazione

NEGR (Complemento a 2 di numero in virgola mobile (a 32 bit, IEEE 754)) nega il numero in virgola mobile (a 32 bit, IEEE 754) nell'accumulatore 1. L'operazione inverte lo stato del bit 31 nell'accumulatore 1 (segno della mantissa). Il risultato viene memorizzato nell'accumulatore 1.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio

```
AWL             Spiegazione
L     ED8       //Carica il valore in ACCU 1 (esempio: ED8 = 1.5E+02).
NEGR            //Nega il numero in virgola mobile (a 32 bit, IEEE 754), memorizza il risultato
                //in ACCU 1.
T     MD10      //Trasferisce il risultato a MD10 (esempio: risultato = −1.5E+02).
```

3.13 TAW Cambia sequenza di byte in ACCU 1 (a 16 bit)

Formato

TAW

Descrizione dell'operazione

TAW inverte la sequenza dei byte nell'accumulatore 1-L. Il risultato viene memorizzato nell'accumulatore 1-L. Gli accumulatori 1-H e 2 non vengono modificati.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio

```
AWL             Spiegazione
L      MW10     //Carica il valore di MW10 in ACCU 1.
TAW             //Inverte la sequenza dei byte in ACCU 1-L.
T      MW20     //Trasferisce il risultato a MW20.
```

Contenuto	ACCU 1-H-H	ACCU 1-H-L	ACCU 1-L-H	ACCU 1-L-L
prima dell'esecuzione di **TAW**	valore A	valore B	valore C	valore D
dopo l'esecuzione di **TAW**	valore A	valore B	valore D	valore C

3.14 TAD Cambia sequenza di byte in ACCU 1 (a 32 bit)

Formato

TAD

Descrizione dell'operazione

TAD inverte la sequenza dei byte nell'accumulatore 1. Il risultato viene memorizzato nell'accumulatore 1. L'accumulatore 2 resta inalterato.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio

```
AWL             Spiegazione
L     MD10      //Carica il valore di MD10 in in ACCU 1.
TAD             //Inverte la sequenza dei byte all'interno di accumulatore 1.
T     MD20      //Trasferisce il risultato a MD20.
```

Contenuto	ACCU 1-H-H	ACCU 1-H-L	ACCU 1-L-H	ACCU 1-L-L
prima dell'esecuzione di **TAD**	valore A	valore B	valore C	valore D
dopo l'esecuzione di **TAD**	valore D	valore C	valore B	valore A

3.15 RND Arrotonda al numero intero

Formato

RND

Descrizione dell'operazione

RND (Converti numero in virgola mobile a 32 bit, IEEE 754, in numero intero a 32 bit) interpreta il contenuto dell'accumulatore 1 come numero in virgola mobile (a 32 bit, IEEE 754). L'operazione converte il numero in virgola mobile (a 32 bit, IEEE 754) in un numero intero (a 32 bit), e arrotonda il risultato al numero intero superiore. Se il decimale del numero convertito è esattamente compreso tra un risultato pari ed uno dispari, l'operazione arrotonda al risultato pari. Se il numero non rientra nell'area di validità, i bit di stato OV e OS vengono settati a "1".

Qualora dovesse verificarsi un errore (presenza di NaN o numero in virgola mobile non rappresentabile quale numero intero a 32 bit), la conversione non viene eseguita, e viene indicato un overflow.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	x	x	-	-	-	-

Esempio

```
AWL             Spiegazione
C      MD10     //Carica il numero in virgola mobile (a 32 bit, IEEE 754) in ACCU 1-L.
RND             //Converte il numero in virgola mobile (a 32 bit, IEEE 754) in un numero intero
                (a 32 bit) e arrotonda il risultato. Memorizza il risultato in ACCU 1-L.
T      MD20     //Trasferisce il risultato (numero intero a 32 bit) a MD20.
```

Valore prima della conversione		Valore dopo la conversione
MD10 = "100.5"	=> RND =>	MD20 = "+100"
MD10 = "-100.5"	=> RND =>	MD20 = "-100"

3.16 TRUNC Arrotonda senza resto

Formato

TRUNC

Descrizione dell'operazione

TRUNC (Converti numero in virgola mobile a 32 bit, IEEE 754, in numero intero a 32 bit) interpreta il contenuto dell'accumulatore 1 come numero in virgola mobile (a 32 bit, IEEE 754). L'operazione converte il numero in virgola mobile (a 32 bit, IEEE 754) in un numero intero (a 32 bit). Il risultato è dato dalla parte intera del numero in virgola mobile (modo di arrotondamento IEEE 'Round to Zero'). Se il numero non rientra nell'area di validità, i bit di stato OV e OS vengono settati a "1". Il risultato viene memorizzato nell'in ACCU 1.

Qualora si dovesse verificare un errore (presenza di NaN o di numero in virgola mobile non rappresentabile quale numero intero a 32 bit), la conversione non viene eseguita e viene indicato un overflow.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	x	x	-	-	-	-

Esempio

```
AWL             Spiegazione
L     MD10      //Carica il numero in virgola mobile (a 32 bit, IEEE 754) in ACCU 1-L.
TRUNC           //Converte il numero in virgola mobile (a 32 bit, IEEE 754) in un numero intero
                (a 32 bit) e arrotonda il risultato. Memorizza il risultato in ACCU 1-L.
T     MD20      //Trasferisce il risultato (numero intero, 32 bit) a MD20.
```

Valore prima della conversione		Valore dopo la conversione
MD10 = "100.5"	=> TRUNC =>	MD20 = "+100"
MD10 = "-100.5"	=> TRUNC =>	MD20 = "-100"

3.17 RND+ Arrotonda al numero intero superiore (a 32 bit)

Formato

RND+

Descrizione dell'operazione

RND+ (Converti numero in virgola mobile a 32 bit, IEEE 754, in numero intero a 32 bit) interpreta il contenuto dell'accumulatore 1 come un numero in virgola mobile (a 32 bit, IEEE 754). L'operazione converte il numero in virgola mobile (a 32 bit, IEEE 754) in un numero intero (a 32 bit), ed arrotonda il risultato al numero intero più piccolo, maggiore od uguale al numero in virgola mobile convertito (modo di arrotondamento IEEE "Round to + infinity"). Se il numero non rientra nell'area di validità, i bit di stato OV e OS vengono settati a "1". Il risultato viene memorizzato nell'accumulatore 1.

Qualora dovesse verificarsi un errore (presenza di NaN o di numero in virgola mobile non rappresentabile come numero intero a 32 bit), la conversione non viene eseguita e viene visualizzato un overflow.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	x	x	-	-	-	-

Esempio

```
AWL             Spiegazione
L      MD10     //Carica il numero in virgola mobile (a 32 bit, IEEE 754)

                //in ACCU 1-L.
RND+            //Converte il numero in virgola mobile (a 32 bit, IEEE 754) in un numero intero
                //(a 32 bit) e arrotonda il risultato. Memorizza il risultato in ACCU 1-L.
T      MD20     //Trasferisce il risultato (numero intero a 32 bit) a MD20.
```

Valore prima della conversione		Valore dopo la conversione
MD10 = "100.5"	=> RND+ =>	MD20 = "+101"
MD10 = "-100.5"	=> RND+ =>	MD20 = "-100"

3.18 RND- Arrotonda al numero intero inferiore (a 32 bit)

Formato

RND-

Descrizione dell'operazione

RND- (Converti un numero in virgola mobile a 32 bit, IEEE 754, in numero intero a 32 bit) interpreta il contenuto dell'accumulatore 1 quale numero in virgola mobile (a 32 bit, IEEE 754). L'operazione converte il numero in virgola mobile (a 32 bit, IEEE 754) in un numero intero (a 32 bit), e arrotonda il risultato al numero intero minore od uguale al numero in virgola mobile convertito (modo di arrotondamento IEEE "Round to -infinity"). Se il numero non rientra nell'area consentita, i bit di stato OV e OS vengono settati a "1". Il risultato viene memorizzato in ACCU 1.

Qualora si dovesse verificare un errore (presenza di NaN o di un numero in virgola mobile che non può essere rappresentato come numero intero a 32 bit), la conversione non viene eseguita e viene visualizzato un overflow.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	x	x	-	-	-	-

Esempio

```
AWL             Spiegazione
L     MD10      //Carica il numero in virgola mobile (a 32 bit, IEEE 754) in ACCU 1-L.
RND-            //Converte il numero in virgola mobile (a 32 bit, IEEE 754) in un numero intero
                //(a 32 bit) e arrotonda il risultato. Memorizza il risultato in ACCU 1-L.
T     MD20      //Trasferisce il risultato (numero intero a 32 bit) a MD20.
```

Valore prima della conversione		Valore dopo la conversione
MD10 = "100.5"	=> RND- =>	MD20 = "+100"
MD10 = "-100.5"	=> RND- =>	MD20 = "-101"

4 Operazioni di conteggio

4.1 Sommario delle operazioni di conteggio

Descrizione

Un contatore è un elemento funzionale del linguaggio di programmazione STEP 7. I contatori hanno un'area di memoria propria nella CPU. Quest'area di memoria riserva una parola di 16 bit per ogni contatore. La programmazione con AWL supporta 256 contatori. Per sapere quanti contatori ha a disposizione la propria CPU, l'utente deve consultarne i dati tecnici. Le operazioni di conteggio sono le sole funzioni che hanno accesso all'area di memoria riservata al contatore.

Sono disponibili le seguenti operazioni di conteggio:

- FR Abilita contatore
- L Carica valore attuale di conteggio in ACCU 1 come numero intero
- LC Carica valore attuale di conteggio in ACCU 1 come BCD
- R Resetta contatore
- S Imposta valore iniziale di conteggio
- ZV Conta in avanti
- ZR Conta all'indietro

4.2 FR Abilita contatore

Formato

FR <contatore>

Operando	Tipo dati	Area memoria	Descrizione
<contatore>	COUNTER	Z	Contatore; l'area dipende dalla CPU.

Descrizione dell'operazione

FR <contatore> cancella il merker del fronte che setta il contatore indirizzato da ‘Conta in avanti' o ‘Conta all'indietro' quando RLC passa da "0" a "1". L'abilitazione del contatore non è necessaria qualora il contatore debba essere settato, o qualora venga eseguita una normale operazione di conteggio. Vale a dire che, nonostante un RLC costante di 1 alle istruzioni Imposta valore iniziale di conteggio, Conta in avanti o Conta all'indietro, queste operazioni vengono nuovamente eseguite dopo l'abilitazione.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	0	-	-	0

Esempio

```
AWL             Spiegazione
U     E 2.0     //Interroga lo stato di segnale all'ingresso E 2.0.
FR    Z3        //Abilita il contatore Z3 quando RLC passa da "0" a "1".
```

4.3 L Carica valore attuale di conteggio in ACCU 1 come numero intero

Formato

L <contatore>

Operando	Tipo dati	Area memoria	Descrizione
<contatore>	COUNTER	Z	Contatore; l'area dipende dalla CPU.

Descrizione dell'operazione

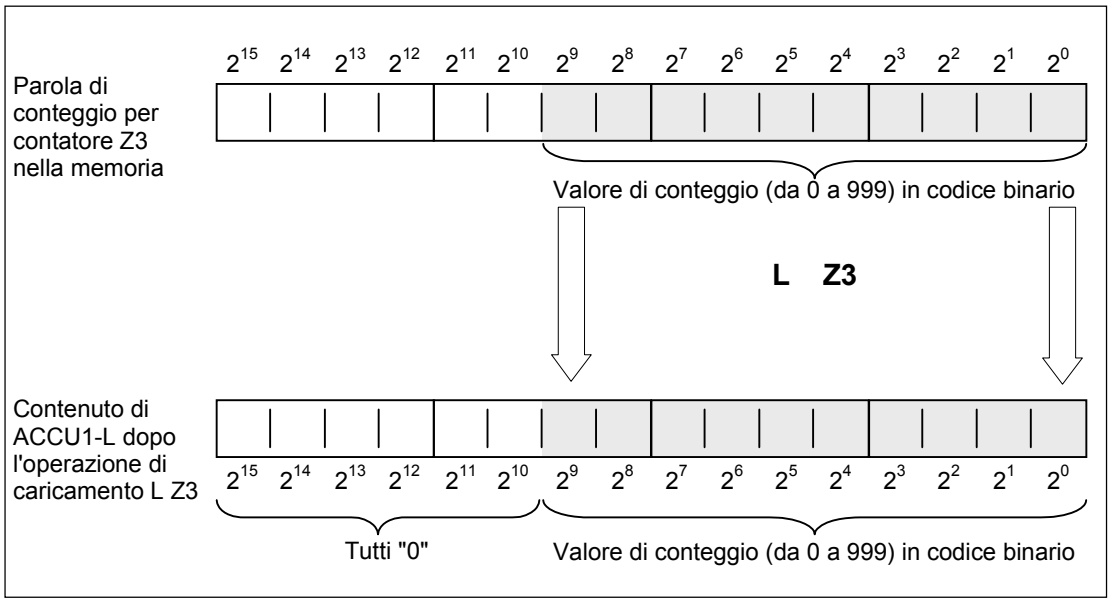
L <contatore> carica il valore di conteggio del contatore indirizzato con formato di numero intero in ACCU 1-L, successivamente al salvataggio del contenuto di ACCU 1 in ACCU 2.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio

```
AWL             Spiegazione
L     Z3        //Carica il valore di conteggio del contatore Z3 col formato binario in ACCU  1-L.
```



4.4 LC Carica valore attuale di conteggio in ACCU 1 come BCD

Formato

LC <contatore>

Operando	Tipo dati	Area memoria	Descrizione
<contatore>	COUNTER	Z	Contatore; l'area dipende dalla CPU.

Descrizione dell'operazione

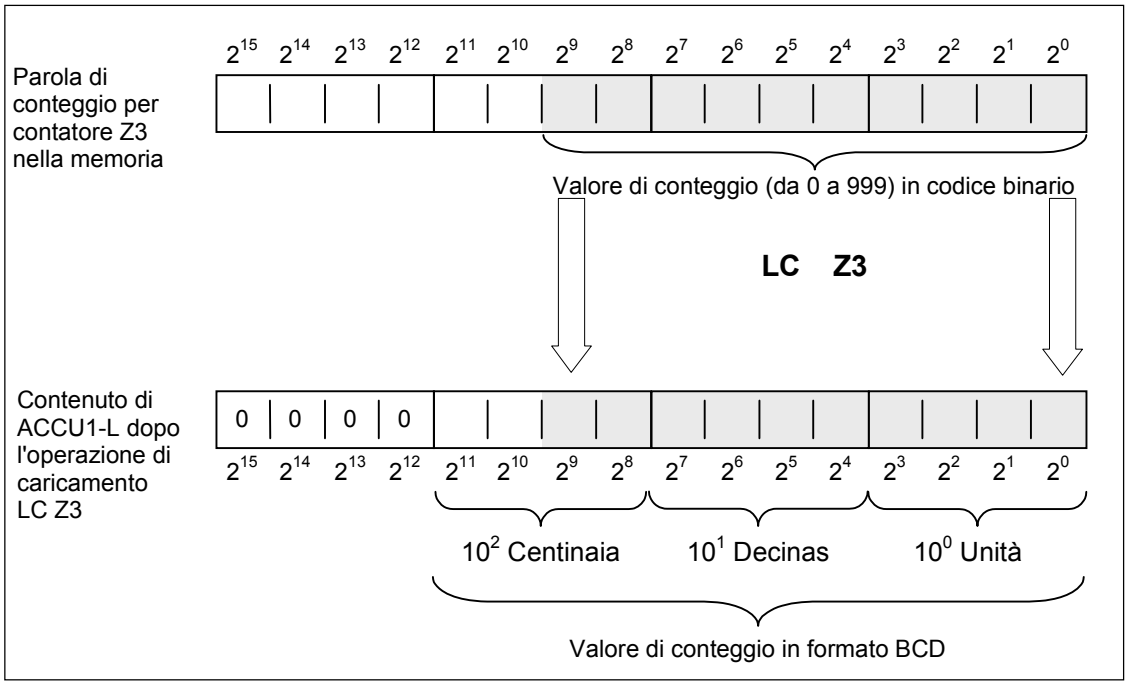
LC <contatore> carica il valore di conteggio del contatore indirizzato come numero BCD (decimale in codice binario) in ACCU 1, successivamente al salvataggio del contenuto di ACCU 1 in ACCU 2.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio

```
AWL              Spiegazione
LC    Z3         //Carica il valore di conteggio del contatore Z3 in formato BCD in ACCU  1- L.
```



4.5 R Resetta contatore

Formato

R <contatore>

Operando	Tipo dati	Area memoria	Descrizione
<contatore>	COUNTER	Z	Contatore da resettare; l'area dipende dalla CPU

Descrizione dell'operazione

R <contatore> carica il valore di conteggio "0" nel contatore indirizzato se RLC = 1.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	0	-	-	0

Esempio

```
AWL             Spiegazione
U     E 2.3     //Interroga lo stato di segnale all'ingresso E 2.3.
R     Z3        //Resetta il contatore Z3 a "0", quando RLC passa da "0" a "1".
```

4.6 S Imposta valore iniziale di conteggio

Formato

S <contatore>

Operando	Tipo dati	Area memoria	Descrizione
<contatore>	COUNTER	Z	Contatore da settare; l'area dipende dalla CPU

Descrizione dell'operazione

S <contatore> carica il valore di conteggio da ACCU 1-L nel contatore indirizzato, quando RLC passa da "0" a "1". Il valore di conteggio deve esistere in ACCU 1 con formato BCD, e deve essere compreso tra ”0" e "999".

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	0	-	-	0

Esempio

```
AWL             Spiegazione
U     E 2.3     //Interroga lo stato di segnale all'ingresso E 2.3.
L     C#3       //Carica il valore di conteggio 3 in ACCU 1-L.
S     Z1        //Setta il contatore Z1 al valore di conteggio quando RLC passa da "0" a "1".
```

4.7 ZV Conta in avanti

Formato

ZV <contatore>

Operando	Tipo dati	Area memoria	Descrizione
<contatore>	COUNTER	Z	Contatore; l'area dipende dalla CPU.

Descrizione dell'operazione

ZV <contatore> incrementa il valore di conteggio del contatore indirizzato di una unità quando RLC passa da "0" a "1", e il valore di conteggio è inferiore a "999". Quando il valore di conteggio raggiunge il valore limite superiore "999", il valore non viene più incrementato. Ulteriori transizioni di RLC rimangono prive d'effetto. Il bit di overflow (OV) non viene settato.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	0	-	-	0

Esempio

```
AWL             Spiegazione
U     E 2.1     //Interroga lo stato di segnale all'ingresso E 2.1.
ZV    Z3        //Incrementa il contatore Z3 di una unità quando RLC passa da "0" a "1".
```

4.8 ZR Conta all'indietro

Formato

ZR <contatore>

Operando	Tipo dati	Area memoria	Descrizione
<contatore>	COUNTER	Z	Contatore, l'area dipende dalla CPU

Descrizione dell'operazione

ZR <contatore> decrementa di un'unità il valore di conteggio del contatore indirizzato quando RLC passa da "0" a "1", ed il valore di conteggio è maggiore di "0". Quando il valore di conteggio raggiunge il valore limite inferiore "0", il valore non viene più decrementato. Ulteriori transizioni di RLC rimangono prive d'effetto, in quanto il contatore non opera con valori negativi.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	0	-	-	0

Esempio

```
AWL             Spiegazione
L     C#14      //Valore di conteggio presettato.
U     E 0.1     //Contatore presettato dopo il rilevamento di un fronte di salita nell'ingresso
                //E  0.1.
S     Z1        //Carica nel contatore Z1, se abilitato, i valori preimpostati.
U     E 0.0     //Decrementa di un'unità per ogni fronte di salita in E 0.0.
ZR    Z1        //Decrementa il contatore Z1 di un'unità quando il RLC dipendente dall'ingresso
                //E 0.0 passa da "0" a "1".
UN    Z1        //Individuazione degli zeri con bit Z1.
=     A 0.0     //Se il valore del contatore Z1 è uguale a "0", A 0.0 = 1.
```

5 Operazioni di blocchi di dati

5.1 Sommario delle operazioni del blocco dati

Descrizione

L'utente può adoperare l'operazione AUF (Apri blocco dati) per aprire un blocco dati, un blocco dati globale o un blocco dati di istanza.

Sono disponibili le seguenti operazioni del blocco dati:

- AUF Apri blocco dati
- TDB Scambia DB globale e DB di istanza
- L DBLG Carica lunghezza del DB globale in ACCU 1
- L DBNO Carica numero del DB globale in ACCU 1
- L DILG Carica lunghezza del DB di istanza in ACCU 1
- L DINO Carica numero del DB di istanza in ACCU 1

5.2 AUF Apri blocco dati

Formato

AUF <blocco di dati>

Operando	Tipo di blocco dati	Indirizzo sorgente
<blocco di dati>	DB, DI	da 1 a 65535

Descrizione dell'operazione

AUF <blocco di dati> apre un blocco di dati come blocco di dati globale o blocco di dati di istanza. Per ogni esecuzione è possibile aprire contemporaneamente un blocco di dati globale ed un blocco dati di istanza.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio

```
AWL             Spiegazione
AUF   DB10      //Apre il blocco di dati DB10 come blocco di dati globale.
L     DBW35     //Carica la parola dati DBW35 del blocco di dati aperto in ACCU1-L.
T     MW22      //Trasferisce il contenuto di ACCU-L alla MW22.
AUF   DI20      //Apre il blocco di dati DB20 come blocco di dati di istanza.
L     DIB12     //Carica il byte dati DIB12 del blocco di dati di istanza aperto in ACCU-1-L.
T     DBB37     //Trasferisce il contenuto di ACCU-L-L al byte dati DBB37 del blocco di dati
                globale aperto.
```

5.3 TDB Scambia DB globale e DB di istanza

Formato

TDB

Descrizione dell'operazione

TDB determina uno scambio dei registri dei blocchi di dati. Il blocco di dati globale diventa così un blocco di dati di istanza e viceversa.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

5.4 L DBLG Carica lunghezza del DB globale in ACCU 1

Formato

L DBLG

Descrizione dell'operazione

L DBLG (Carica lunghezza del DB globale) carica la lunghezza del blocco di dati globale nell'accumulatore 1; dopo che è stato memorizzato il contenuto dell'accumulatore 1 nell'accumulatore 2.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio

```
AWL          Spiegazione
AUF    DB10  //Apre il blocco di dati DB10 come blocco di dati globale.
L      DBLG  //Carica la lunghezza del blocco di dati globale (lunghezza di DB10).
L      MD10  //Valore di confronto per verificare se la lunghezza del blocco di dati è

             //sufficiente.
<D
SPB    ERRO  //Passa all'etichetta di salto ERRO se la lunghezza è inferiore al valore riportato
             //in MD10.
```

5.5 L DBNO Carica numero del DB globale in ACCU 1

Formato

L DBNO

Descrizione dell'operazione

L DBNO (Carica numero del DB globale) carica il numero del blocco di dati globale aperto nell'accumulatore 1-L, dopo che è stato memorizzato il contenuto dell'accumulatore 1 nell'accumulatore 2.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

5.6 L DILG Carica lunghezza del DB di istanza in ACCU 1

Formato

L DILG

Descrizione dell'operazione

L DILG (Carica lunghezza del DB di istanza) carica la lunghezza del blocco di dati di istanza nell'accumulatore 1, dopo che è stato memorizzato il contenuto dell'accumulatore 1 nell'accumulatore 2.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio

```
AWL             Spiegazione
AUF    DI20     //Apre il blocco di dati DB20 come blocco di dati di istanza.
L      DILG     //Carica la lunghezza del blocco di dati di istanza (lunghezza di DB20).
L      MW10     //Valore di confronto per verificare se la lunghezza del blocco di dati è
                sufficiente.
<I
SPB    ERRO     //Passa all'etichetta di salto ERRO se la lunghezza del blocco di dati è inferiore
                al valore riportato in MW10.
```

5.7 L DINO Carica numero del DB di istanza in ACCU 1

Formato

L DINO

Descrizione dell'operazione

L DINO (Carica numero del DB di istanza) carica il numero del blocco di dati di istanza aperto nell'accumulatore 1, dopo che è stato memorizzato il contenuto dell'accumulatore 1 nell'accumulatore 2.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

6 Operazioni di salto

6.1 Sommario delle operazioni di salto

Descrizione

L'utente può adoperare le operazioni di salto per controllare il flusso delle operazioni logiche del proprio programma. Queste operazioni abilitano il proprio programma ad interrompere il flusso lineare delle operazioni per riavviare la lettura in un punto diverso. L'utente può adoperare l'operazione LOOP per richiamare il segmento di programma più volte.

L'operando di un'operazione di salto o di loop è un'etichetta di salto. L'etichetta può essere data da un massimo di quattro caratteri, il primo dei quali deve essere una lettera dell'alfabeto. L'etichetta termina con due punti ":" e introduce l'istruzione nella riga.

Nota

Tenere conto del fatto che, per programmi per le CPU S7–300, in caso di operazioni di salto, la destinazione del salto è sempre l'**inizio** di una serie di combinazioni (non necessariamente con 318-2). La destinazione del salto non può trovarsi all'interno di una serie di com-binazioni.

L'utente può adoperare le seguenti operazioni di salto per interrompere in modo incondizionato un flusso normale di operazioni del programma:

- SPA Salto assoluto
- SPL Salta alle etichette

Le seguenti operazioni di salto interrompono l'esecuzione del programma sulla base del risultato logico combinatorio prodotto dall'istruzione di operazione antecedente:

- SPB Salta se RLC = 1
- SPBN Salta se RLC = 0
- SPBB Salta se RLC = 1 con BIE
- SPBNB Salta se RLC = 0 con BIE

Le seguenti operazioni di salto interrompono l'esecuzione del programma, sulla base dello stato di segnale di un bit nella parola di stato.

- SPBI Salta se BIE = 1
- SPBIN Salta se BIE = 0
- SPO Salta se OV = 1
- SPS Salta se OS = 1

Le seguenti operazioni di salto interrompono l'esecuzione del programma sulla base del risultato di un'operazione precedente:

- SPZ Salta se risultato = 0
- SPN Salta se risultato diverso da zero
- SPP Salta se risultato > 0
- SPM Salta se risultato < 0
- SPPZ Salta se risultato >= 0
- SPMZ Salta se risultato <= 0
- SPU Salta se operazione non ammessa

6.2 SPA Salto assoluto

Formato

SPA <etichetta di salto>

Operando	Descrizione
<etichetta di salto>	Nome simbolico della destinazione di salto

Descrizione dell'operazione

SPA <etichetta di salto> interrompe il flusso lineare delle operazioni logiche e salta alla destinazione di salto indipendentemente dallo stato. Il flusso lineare delle operazioni logiche viene ripreso alla destinazione di salto. La destinazione di salto viene indicata tramite un'etichetta. Il salto può essere eseguito sia in avanti che all'indietro. I salti sono eseguibili unicamente all'interno di un singolo blocco: l'istruzione di salto e la destinazione di salto devono cioè trovarsi nel medesimo blocco. Nel blocco può essere riportata un'unica destinazione di salto. L'ampiezza massima del salto è di -32768 / +32767 parole del codice programma. Il numero massimo di quante istruzioni possano effettivamente essere saltate dipende da come le istruzioni stesse sono state combinate internamente al programma (istruzioni ad una, due, tre parole).

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio

```
AWL                 Spiegazione
      U     E 1.0
      U     E 1.2
      SPB   DELE   //Salta all'etichetta DELE se RLC = 1.
      L     MB10
      INC   1
      T     MB10
      SPA   FORW   //Salto incondizionato all'etichetta FORW.
DELE: L     0
      T     MB10
FORW: U     E 2.1  //Il flusso delle operazioni logiche viene ripreso in questo punto dopo il
                   //salto all'etichetta FORW.
```

6.3 SPL Salta alle etichette

Formato

SPL <etichetta di salto>

Operando	Descrizione
<etichetta di salto>	Nome simbolico della destinazione di salto

Descrizione dell'operazione

SPL <etichetta di salto> (Salta tramite lista di salto) consente di programmare più salti. La lista delle destinazioni di salto, che può riportare un massimo di 255 registrazioni, inizia immediatamente dopo l'operazione SPL e finisce prima dell'etichetta indicata dall'operando SPL. Ogni destinazione di salto è costituita da un'operazione SPA. Il numero del salto di destinazione (da 0 a 255) viene rilevato dall'accumulatore 1-L-L.

Se il contenuto dell'accumulatore è inferiore al numero di destinazioni di salto tra l'istruzione SPL e l'etichetta di salto, l'operazione SPL salta ad una delle operazioni SPA. Se l'accumulatore 1-L-L = 0, il salto viene eseguito alla prima operazione SPA; se l'accumulatore 1-L-L = 1, il salto viene eseguito alla seconda operazione SPA, ecc. Se il numero di destinazioni di salto è invece maggiore, l'operazione SPL salta alla prima istruzione successiva all'ultima operazione SPA della lista delle destinazioni di salto.

Nella lista delle destinazioni di salto possono figurare soltanto SPA, specificate prima dell'etichetta che viene indicata dall'operando dell'istruzione SPL. All'interno della lista di salto non sono consentite altre operazioni.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio

```
AWL                 Spiegazione
      L     MB0     //Carica il numero di destinazioni di salto in ACCU 1-L-L.
      SPL   LSTX    //Destinazione di salto se ACCU 1-L-L > 3.
      SPA   SEG0    //Destinazione di salto se ACCU 1-L-L = 0.
      SPA   SEG1    //Destinazione di salto se ACCU 1-L-L = 1.
      SPA   COMM    //Destinazione di salto se ACCU 1-L-L = 2.
      SPA   SEG3    //Destinazione di salto se ACCU 1-L-L = 3.
LSTX: SPA   COMM
SEG0: *             //Istruzione ammessa.
      *
      SPA   COMM
SEG1: *             //Istruzione ammessa.
      *
      SPA   COMM
SEG3: *             //Istruzione ammessa.
      *
      SPA   COMM
COMM: *
      *
```

6.4 SPB Salta se RLC = 1

Formato

SPB <etichetta di salto>

Operando	Descrizione
<etichetta di salto>	Nome simbolico della destinazione di salto

Descrizione dell'operazione

Se RLC = 1, **SPB <etichetta di salto>** interrompe il flusso lineare delle operazioni logiche e salta alla destinazione di salto. Il flusso lineare delle operazioni logiche viene ripreso alla destinazione di salto. La destinazione di salto viene indicata tramite un'etichetta. Il salto può essere eseguito sia in avanti che all'indietro. I salti sono eseguibili unicamente all'interno di un singolo blocco: l'istruzione di salto e la destinazione di salto devono cioè trovarsi nel medesimo blocco. Nel blocco può essere riportata un'unica destinazione di salto. L'ampiezza massima del salto è di -32768 / +32767 parole del codice programma. Il numero massimo di quante istruzioni possano effettivamente essere saltate dipende da come le istruzioni stesse sono state combinate internamente al programma (istruzioni ad una, due, tre parole).

Se RLC = 0 il salto non viene eseguito. Il RLC viene settato a "1" e il flusso di operazioni logiche prosegue con l'istruzione seguente.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	0	1	1	0

Esempio

```
AWL                 Spiegazione
      U     E 1.0
      U     E 1.2
      SPB   JOVR  //Salta all'etichetta JOVR se RLC = 1.
      L     EW8   //Il flusso delle operazioni logiche viene ripreso in questo punto, se il
                  //salto non viene eseguito.
      T     MW22
JOVR: U     E 2.1 //Il flusso delle operazioni logiche viene ripreso in questo punto, dopo
                  //il salto all'etichetta JOVR.
```

6.5 SPBN Salta se RLC = 0

Formato

SPBN <etichetta di salto>

Operando	Descrizione
<etichetta di salto>	Nome simbolico della destinazione di salto

Descrizione dell'operazione

Se RLC = 0, **SPBN <etichetta di salto>** interrompe il flusso lineare delle operazioni logiche e salta alla destinazione di salto. Il flusso lineare delle operazioni logiche viene ripreso alla destinazione di salto. La destinazione di salto viene indicata tramite un'etichetta. Il salto può essere eseguito sia in avanti che all'indietro. I salti sono eseguibili unicamente all'interno di un singolo blocco: l'istruzione di salto e la destinazione di salto devono cioè trovarsi nel medesimo blocco. Nel blocco può essere riportata un'unica destinazione di salto. L'ampiezza massima del salto è di -32768 / +32767 parole del codice programma. Il numero massimo di quante istruzioni possano effettivamente essere saltate dipende da come le istruzioni stesse sono state combinate internamente al programma (istruzioni ad una, due, tre parole).

Se RLC = 1, il salto non viene eseguito. Il flusso delle operazioni logiche prosegue con l'istruzione seguente.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	0	1	1	0

Esempio

```
AWL                 Spiegazione
      U     E 1.0
      U     E 1.2
      SPBN  JOVR  //Salta all'etichetta JOVR se RLC = 0.
      L     EW8   //Il flusso delle operazioni logiche viene ripreso in questo punto se il
                  salto non viene eseguito.
      T     MW22
JOVR: U     E 2.1 //Il flusso delle operazioni logiche viene ripreso in questo punto dopo il
                  salto all'etichetta JOVR.
```

6.6 SPBB Salta se RLC = 1 con BIE

Formato

SPBB <etichetta di salto>

Operando	Descrizione
<etichetta di salto>	Nome simbolico della destinazione di salto

Descrizione dell'operazione

Se RLC = 1, **SPBB <etichetta di salto>** interrompe il flusso lineare delle operazioni logiche e salta alla destinazione di salto. Il flusso lineare delle operazioni logiche viene ripreso alla destinazione di salto. La destinazione di salto viene indicata tramite un'etichetta. Il salto può essere eseguito sia in avanti che all'indietro. I salti sono eseguibili unicamente all'interno di un singolo blocco: l'istruzione di salto e la destinazione di salto devono cioè trovarsi nel medesimo blocco. Nel blocco può essere riportata un'unica destinazione di salto. L'ampiezza massima del salto è di -32768 / +32767 parole del codice programma. Il numero massimo di quante istruzioni possano effettivamente essere saltate dipende da come le istruzioni stesse sono state combinate internamente al programma (istruzioni ad una, due, tre parole).

Se RLC = 0, il salto non viene eseguito. RLC viene settato a "1" e il flusso delle operazioni logiche prosegue con l'istruzione seguente.

Indipendentemente dal RLC, nella operazione **SPBB <etichetta di salto>** RLC viene copiato in BIE.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	x	-	-	-	-	0	1	1	0

Esempio

```
AWL                 Spiegazione
      U     E 1.0
      U     E 1.2
      SPBB  JOVR    //Salta all'etichetta JOVR se RLC = 1. Copia il contenuto del bit RLC nel bit
                    BIE.
      L     EW8     //Il flusso delle operazioni logiche viene ripreso in questo punto, se il
                    salto non viene eseguito.
      T     MW22
JOVR: U     E 2.1   //Il flusso delle operazioni logiche viene ripreso in questo punto, dopo
                    il salto all'etichetta JOVR.
```

6.7 SPBNB Salta se RLC = 0 con BIE

Formato

SPBNB <etichetta di salto>

Operando	Descrizione
<etichetta di salto>	Nome simbolico della destinazione di salto

Descrizione dell'operazione

Se RLC = 0, **SPBNB <etichetta di salto>** interrompe il flusso lineare delle operazioni logiche e salta alla destinazione di salto. Il flusso lineare delle operazioni logiche viene ripreso alla destinazione di salto. La destinazione di salto viene indicata tramite un'etichetta. Il salto può essere eseguito sia in avanti che all'indietro. I salti sono eseguibili unicamente all'interno di un singolo blocco: l'istruzione di salto e la destinazione di salto devono cioè trovarsi nel medesimo blocco. Nel blocco può essere riportata un'unica destinazione di salto. L'ampiezza massima del salto è di -32768 / +32767 parole del codice programma. Il numero massimo di quante istruzioni possano effettivamente essere saltate dipende da come le istruzioni stesse sono state combinate internamente al programma (istruzioni ad una, due, tre parole).

Se RLC = 1, il salto non viene eseguito. RLC viene settato a "1", e il flusso delle operazioni logiche prosegue con l'istruzione successiva.

Indipendentemente dal RLC, nell'operazione **SPBNB <etichetta di salto>** RLC viene copiato in BIE.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	x	-	-	-	-	0	1	1	0

Esempio

```
AWL                 Spiegazione
      U     E 1.0
      U     E 1.2
      SPBNB JOVR  //Salta all'etichetta JOVR se RLC = 0.  Copia in BIE il contenuto del bit
                  RLC
      L     EW    //Il flusso delle operazioni di programma viene ripreso in questo punto se
                  il salto non viene eseguito.
      T     MW22
JOVR: U     E 2.1 //Il flusso delle operazioni di programma viene ripreso in questo punto dopo
                  il salto all'etichetta JOVR.
```

6.8 SPBI Salta se BIE = 1

Formato

SPBI <etichetta di salto>

Operando	Descrizione
<etichetta di salto>	Nome simbolico della destinazione di salto

Descrizione dell'operazione

Se il bit di stato BIE = 1, **SPBI <etichetta di salto>** interrompe il flusso lineare delle operazioni logiche, e salta alla destinazione di salto. Il flusso lineare delle operazioni logiche viene ripreso alla destinazione di salto. La destinazione di salto viene indicata tramite un'etichetta. Il salto può essere eseguito sia in avanti che all'indietro. I salti sono eseguibili unicamente all'interno di un singolo blocco: l'istruzione di salto e la destinazione di salto devono cioè trovarsi nel medesimo blocco. Nel blocco può essere riportata un'unica destinazione di salto. L'ampiezza massima del salto è di -32768 / +32767 parole del codice programma. Il numero massimo di quante istruzioni possano effettivamente essere saltate dipende da come le istruzioni stesse sono state combinate internamente al programma (istruzioni ad una, due, tre parole).

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	0	1	-	0

6.9 SPBIN Salta se BIE = 0

Formato

SPBIN <etichetta di salto>

Operando	Descrizione
<etichetta di salto>	Nome simbolico della destinazione di salto

Descrizione dell'operazione

Se il bit di stato BIE = 0, **SPBIN <etichetta di salto>** interrompe il flusso lineare delle operazioni logiche e salta alla destinazione di salto. Il flusso lineare delle operazioni logiche viene ripreso alla destinazione di salto. La destinazione di salto viene indicata tramite un'etichetta. Il salto può essere eseguito sia in avanti che all'indietro. I salti sono eseguibili unicamente all'interno di un singolo blocco: l'istruzione di salto e la destinazione di salto devono cioè trovarsi nel medesimo blocco. Nel blocco può essere riportata un'unica destinazione di salto. L'ampiezza massima del salto è di -32768 / +32767 parole del codice programma. Il numero massimo di quante istruzioni possano effettivamente essere saltate dipende da come le istruzioni stesse sono state combinate internamente al programma (istruzioni ad una, due, tre parole).

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	0	1	-	0

6.10 SPO Salta se OV = 1

Formato

SPO <etichetta di salto>

Operando	Descrizione
<etichetta di salto>	Nome simbolico della destinazione di salto

Descrizione dell'operazione

Se il bit di stato OV = 1, **SPO <etichetta di salto>** interrompe il flusso lineare delle operazioni logiche e salta alla destinazione di salto. Il flusso lineare delle operazioni logiche viene ripreso alla destinazione di salto. La destinazione di salto viene indicata tramite un'etichetta. Il salto può essere eseguito sia in avanti che all'indietro. I salti sono eseguibili unicamente all'interno di un singolo blocco: l'istruzione di salto e la destinazione di salto devono cioè trovarsi nel medesimo blocco. Nel blocco può essere riportata un'unica destinazione di salto. L'ampiezza massima del salto è di -32768 / +32767 parole del codice programma. Il numero massimo di quante istruzioni possano effettivamente essere saltate dipende da come le istruzioni stesse sono state combinate internamente al programma (istruzioni ad una, due, tre parole).

Nel caso di operazioni aritmetiche combinate è necessario verificare che nessuna operazione aritmetica abbia provocato un'eccedenza, in modo tale da garantire che ogni risultato intermedio rientri nell'area di validità. In caso contrario è necessario avvalersi dell'operazione SPS.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio

```
AWL                 Spiegazione
      L     MW10
      L     3
      *I            //Moltiplicazione del contenuto di MW10 per "3".
      SPO   OVER    //Salta se il risultato eccede l'area massima (OV = 1).
      T     MW10    //Il flusso delle operazioni logiche viene ripreso in questo punto se il
                    //salto non viene eseguito.
      U     M 4.0
      R     M 4.0
      SPA   NEXT
OVER: UN    M 4.0   //Il flusso delle operazioni logiche viene ripreso in questo punto dopo il
                    //salto all'etichetta OVER.
      S     M 4.0
NEXT: NOP 0         //Il flusso delle operazioni logiche viene ripreso in questo punto dopo il
                    //salto all'etichetta NEXT.
```

6.11 SPS Salta se OS = 1

Formato

SPS <etichetta di salto>

Operando	Descrizione
<etichetta di salto>	Nome simbolico della destinazione di salto

Descrizione dell'operazione

Se il bit di stato OS = 1, **SPS <etichetta di salto>** interrompe il flusso lineare delle operazioni logiche e salta alla destinazione di salto. Il flusso lineare delle operazioni logiche viene ripreso alla destinazione di salto. La destinazione di salto viene indicata tramite un'etichetta. Il salto può essere eseguito sia in avanti che all'indietro. I salti sono eseguibili unicamente all'interno di un singolo blocco: l'istruzione di salto e la destinazione di salto devono cioè trovarsi nel medesimo blocco. Nel blocco può essere riportata un'unica destinazione di salto. L'ampiezza massima del salto è di -32768 / +32767 parole del codice programma. Il numero massimo di quante istruzioni possano effettivamente essere saltate dipende da come le istruzioni stesse sono state combinate internamente al programma (istruzioni ad una, due, tre parole).

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	0	-	-	-	-

Esempio

```
AWL                 Spiegazione
      L     EW10
      L     MW12
      *I
      L     DBW25
      +I
      L     MW14
      -I
      SPS   OVER   //Salta se si verifica eccedenza in una delle 3 operazioni precedente,
                   //OS = 1 (vedi nota).
      T     MW16   //Il flusso delle operazioni logiche viene ripreso in questo punto
                   //se il salto non viene eseguito.
      U     M 4.0
      R     M 4.0
      SPA   NEXT
OVER: UN    M 4.0  //Il flusso delle operazioni logiche viene ripreso in questo punto dopo
                   //il salto all'etichetta OVER.
      S     M 4.0
NEXT: NOP 0        //Il flusso delle operazioni logiche viene ripreso in questo punto dopo
                   //il salto all'etichetta NEXT.
```

Nota

In questo caso non è consentito avvalersi dell'operazione **SPO**. In caso di eccedenza, l'operazione **SPO** interrogherebbe solamente l'overflow della operazione -I precedente.

6.12 SPZ Salta se risultato = 0

Formato

SPZ <etichetta di salto>

Operando	Descrizione
<etichetta di salto>	Nome simbolico della destinazione di salto

Descrizione dell'operazione

Se i bit di stato sono A1 = 0 e A0 = 0, **SPZ <etichetta di salto>** interrompe il flusso lineare delle operazioni logiche e salta alla destinazione di salto. Il flusso lineare delle operazioni logiche viene ripreso alla destinazione di salto. La destinazione di salto viene indicata tramite un'etichetta. Il salto può essere eseguito sia in avanti che all'indietro. I salti sono eseguibili unicamente all'interno di un singolo blocco: l'istruzione di salto e la destinazione di salto devono cioè trovarsi nel medesimo blocco. Nel blocco può essere riportata un'unica destinazione di salto. L'ampiezza massima del salto è di -32768 / +32767 parole del codice programma. Il numero massimo di quante istruzioni possano effettivamente essere saltate dipende da come le istruzioni stesse sono state combinate internamente al programma (istruzioni ad una, due, tre parole).

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio

```
AWL                 Spiegazione
      L     MW10
      SRW   1
      SPZ   ZERO    //Salta all'etichetta ZERO se il bit traslato = 0.
      L     MW2     //Il flusso delle operazioni logiche viene ripreso in questo punto se il
                    //salto non viene eseguito.
      INC   1
      T     MW2
      SPA   NEXT
ZERO: L     MW4     //Il flusso delle operazioni logiche viene ripreso in questo punto dopo il
                    //salto all'etichetta di salto ZERO.
      INC   1
      T     MW4
NEXT: NOP 0         //Il flusso delle operazioni logiche viene ripreso in questo punto dopo il
                    //salto all'etichetta di salto NEXT.
```

6.13 SPN Salta se risultato diverso da zero

Formato

SPN <etichetta di salto>

Operando	Descrizione
<etichetta di salto>	Nome simbolico della destinazione di salto

Descrizione dell'operazione

Se il risultato indicato dai bit di stato A1 e A0 è maggiore o minore di zero (A1 = 0/A0 = 1 o A1 = 1/A0 = 0), **SPN <etichetta di salto>** (Salta se il risultato è <> 0) interrompe il flusso lineare delle operazioni logiche, e salta alla destinazione di salto. Il flusso lineare delle operazioni logiche viene ripreso alla destinazione di salto. La destinazione di salto viene indicata tramite un'etichetta. Il salto può essere eseguito sia in avanti che all'indietro. I salti sono eseguibili unicamente all'interno di un singolo blocco: l'istruzione di salto e la destinazione di salto devono cioè trovarsi nel medesimo blocco. Nel blocco può essere riportata un'unica destinazione di salto. L'ampiezza massima del salto è di -32768 / +32767 parole del codice programma. Il numero massimo di quante istruzioni possano effettivamente essere saltate dipende da come le istruzioni stesse sono state combinate internamente al programma (istruzioni ad una, due, tre parole).

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio

```
AWL                 Spiegazione
      L     EW8
      L     MW12
      XOW
      SPN   NOZE   //Salta se il contenuto di ACCU 1-L è diverso da zero.
      UN    M 4.0  //Il flusso delle operazioni logiche viene ripreso in questo punto se il
                   //salto non viene eseguito.
      S     M 4.0
      SPA   NEXT
NOZE: UN    M 4.1  //Il flusso delle operazioni logiche viene ripreso in questo punto dopo il
                   //salto all'etichetta NOZE.
      S     M 4.1
NEXT: NOP 0        //Il flusso delle operazioni logiche viene ripreso in questo punto dopo il
                   //salto all'etichetta NEXT.
```

6.14 SPP Salta se risultato > 0

Formato

SPP <etichetta di salto>

Operando	Descrizione
<etichetta di salto>	Nome simbolico della destinazione di salto

Descrizione dell'operazione

Se i bit di stato A1 = 1 e A0 = 0, **SPP <etichetta di salto>** interrompe il flusso lineare delle operazioni logiche e salta alla destinazione di salto. Il flusso lineare delle operazioni logiche viene ripreso alla destinazione di salto. La destinazione di salto viene indicata tramite un'etichetta. Il salto può essere eseguito sia in avanti che all'indietro. I salti sono eseguibili unicamente all'interno di un singolo blocco: l'istruzione di salto e la destinazione di salto devono cioè trovarsi nel medesimo blocco. Nel blocco può essere riportata un'unica destinazione di salto. L'ampiezza massima del salto è di -32768 / +32767 parole del codice programma. Il numero massimo di quante istruzioni possano effettivamente essere saltate dipende da come le istruzioni stesse sono state combinate internamente al programma (istruzioni ad una, due, tre parole).

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio

```
AWL                 Spiegazione
      L     EW8
      L     MW12
      -I            //Sottrazione del contenuto di MW12 dal contenuto di EW8.
      SPP   POS     //Salta se il risultato > 0 (vale a dire, ACCU 1 > 0).
      UN    M 4.0   //Il flusso delle operazioni logiche viene ripreso in questo punto se il
                    //salto non viene eseguito.
      S     M 4.0
      SPA   NEXT
POS:  UN    M 4.1   //Il flusso delle operazioni logiche viene ripreso in questo punto dopo il
                    //salto all'etichetta POS.
      S     M 4.1
NEXT: NOP 0         //Il flusso delle operazioni logiche viene ripreso in questo punto dopo il
                    //salto all'etichetta NEXT.
```

6.15 SPM Salta se risultato < 0

Formato

SPM <etichetta di salto>

Operando	Descrizione
<etichetta di salto>	Nome simbolico della destinazione di salto

Descrizione dell'operazione

Se il bit di stato A1 = 0 e il bit di stato A0 = 1, **SPM <etichetta di salto>** interrompe il flusso lineare delle operazioni logiche e salta alla destinazione di salto. Il flusso lineare delle operazioni logiche viene ripreso alla destinazione di salto. La destinazione di salto viene indicata tramite un'etichetta. I salti sono eseguibili unicamente all'interno di un singolo blocco: l'istruzione di salto e la destinazione di salto devono cioè trovarsi nel medesimo blocco. Nel blocco può essere riportata un'unica destinazione di salto. L'ampiezza massima del salto è di -32768 / +32767 parole del codice programma. Il numero massimo di quante istruzioni possano effettivamente essere saltate dipende da come le istruzioni stesse sono state combinate internamente al programma (istruzioni ad una, due, tre parole).

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio

```
AWL                 Spiegazione
      L     EW8
      L     MW12
      -I            //Sottrazione del contenuto di MW12 al contenuto di EW8.
      SPM   NEG     //Salta se il risultato < 0 (vale a dire, contenuto di ACCU 1 < 0).
      UN    M 4.0   //Il flusso delle operazioni logiche viene ripreso in questo punto se il
                    //salto non viene eseguito.
      S     M 4.0
      SPA   NEXT
NEG:  UN    M 4.1   //Il flusso delle operazioni logiche viene ripreso in questo punto dopo il
                    //salto all'etichetta NEG.
      S     M 4.1
NEXT: NOP 0         //Il flusso delle operazioni logiche viene ripreso in questo punto dopo il
                    //salto all'etichetta NEXT.
```

6.16 SPPZ Salta se risultato >= 0

Formato

SPPZ <etichetta di salto>

Operando	Descrizione
<etichetta di salto>	Nome simbolico della destinazione di salto

Descrizione dell'operazione

Se il risultato indicato dai bit di stato A1 e A0 è maggiore o uguale a zero (A1 = 0/A0 = 0 o A1 = 1/A0 = 0), **SPPZ <etichetta di salto>** (Salta se il risultato >= 0) interrompe il flusso lineare delle operazioni logiche e salta alla destinazione di salto. Il flusso lineare delle operazioni logiche viene ripreso alla destinazione di salto. La destinazione di salto viene indicata tramite un'etichetta. Il salto può essere eseguito sia in avanti che all'indietro. I salti sono eseguibili unicamente all'interno di un singolo blocco: l'istruzione di salto e la destinazione di salto devono cioè trovarsi nel medesimo blocco. Nel blocco può essere riportata un'unica destinazione di salto. L'ampiezza massima del salto è di -32768 / +32767 parole del codice programma. Il numero massimo di quante istruzioni possano effettivamente essere saltate dipende da come le istruzioni stesse sono state combinate internamente al programma (istruzioni ad una, due, tre parole).

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio

```
AWL                 Spiegazione
      L     EW8
      L     MW12
      -I            //Sottrazione del contenuto di MW12 dal contenuto di EW8.
      SPPZ  REG0    //Salta se il risultato >= 0 (vale a dire, se il contenuto di ACCU 1 >= 0).
      UN    M 4.0   //Il flusso delle operazioni logiche viene ripreso in questo punto se il
                    //salto non viene eseguito.
      S     M 4.0
      SPA   NEXT
REG0: UN    M 4.1   //Il flusso delle operazioni logiche viene ripreso in questo punto dopo il
                    //salto all'etichetta REG0.
      S     M 4.1
NEXT: NOP 0         //Il flusso delle operazioni logiche viene ripreso in questo punto dopo il
                    //salto all'etichetta NEXT.
```

6.17 SPMZ Salta se risultato <= 0

Formato

SPMZ <etichetta di salto>

Operando	Descrizione
<etichetta di salto>	Nome simbolico della destinazione di salto

Descrizione dell'operazione

Se il risultato indicato dai bit di stato A1 e A0 è minore o uguale a zero (A1 = 0/A0 = 0 oppure A1 = 0/A0 = 1), **SPMZ <etichetta di salto>** (Salta se il risultato <=0) interrompe il flusso lineare delle operazioni logiche e salta alla destinazione di salto. Il flusso lineare delle operazioni logiche viene ripreso alla destinazione di salto. La destinazione di salto viene indicata tramite un'etichetta. Il salto può essere eseguito sia in avanti che all'indietro. I salti sono eseguibili unicamente all'interno di un singolo blocco: l'istruzione di salto e la destinazione di salto devono cioè trovarsi nel medesimo blocco. Nel blocco può essere riportata un'unica destinazione di salto. L'ampiezza massima del salto è di -32768 / +32767 parole del codice programma. Il numero massimo di quante istruzioni possano effettivamente essere saltate dipende da come le istruzioni stesse sono state combinate internamente al programma (istruzioni ad una, due, tre parole).

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio

```
AWL                 Spiegazione
      L     EW8
      L     MW12
      -I            //Sottrazione del contenuto di MW12 dal contenuto di EW8.
      SPMZ  RGE0    //Salta se il risultato <= 0 (vale a dire, il contenuto di ACCU 1 <= 0).
      UN    M 4.0   //Il flusso delle operazioni logiche viene ripreso in questo punto se il
                    //salto non viene eseguito.
      S     M 4.0
      SPA   NEXT
RGE0: UN    M 4.1   //Il flusso delle operazioni logiche viene ripreso in questo punto dopo il
                    //salto all'etichetta RGE0.
      S     M 4.1
NEXT: NOP 0         //Il flusso delle operazioni logiche viene ripreso in questo punto dopo il
                    //salto all'etichetta NEXT.
```

6.18 SPU Salta se operazione non ammessa

Formato

SPU <etichetta di salto>

Operando	Descrizione
<etichetta di salto>	Nome simbolico della destinazione di salto

Descrizione dell'operazione

Se i bit di stato A1 = 1 e A0 = 0, **SPU <etichetta di salto>** interrompe il flusso lineare delle operazioni logiche e salta alla destinazione di salto. Il flusso lineare delle operazioni logiche viene ripreso alla destinazione di salto. La destinazione di salto viene indicata tramite un'etichetta. Il salto può essere eseguito sia in avanti che all'indietro. I salti sono eseguibili unicamente all'interno di un singolo blocco: l'istruzione di salto e la destinazione di salto devono cioè trovarsi nel medesimo blocco. Nel blocco può essere riportata un'unica destinazione di salto. L'ampiezza massima del salto è di -32768 / +32767 parole del codice programma. Il numero massimo di quante istruzioni possano effettivamente essere saltate dipende da come le istruzioni stesse sono state combinate internamente al programma (istruzioni ad una, due, tre parole).

Bit di stato A1 = 1 e A0 = 1, nel caso di

- divisione per zero
- utilizzo di operazioni non consentite o
- risultato "non valido" di un confronto di numeri in virgola mobile, qualora venga utilizzato un formato non valido.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio

```
AWL                 Spiegazione
      L     MD10
      L     ED2
      /D            //Divisione del contenuto di MD10 per il contenuto di ED2.
      SPU   ERRO    //Salta se divisone per zero (cioè ED2 = "0").
      T     MD14    //Il flusso delle operazioni logiche viene ripreso in questo punto se il
                    //salto non viene eseguito.
      U     M 4.0
      R     M 4.0
      SPA   NEXT
ERRO: UN    M 4.0   //Il flusso delle operazioni logiche viene ripreso in questo punto dopo il
                    //salto all'etichetta ERRO.
      S     M 4.0
NEXT: NOP 0         //Il flusso delle operazioni logiche viene ripreso in questo punto dopo il
                    salto all'etichetta NEXT.
```

6.19 LOOP Loop di programma

Formato

LOOP <etichetta di salto>

Operando	Descrizione
<etichetta di salto>	Nome simbolico della destinazione di salto

Descrizione dell'operazione

LOOP <etichetta di salto> (Decrementa l'accumulatore 1-L e salta, se l'accumulatore 1-L <> 0) semplifica la programmazione dei loop di programma. Il contatore loop è un numero intero (a 16 bit) senza segno, e si trova nell'accumulatore 1-L. L'istruzione salta alla destinazione di salto indicata. L'operazione di salto viene eseguita finché il contenuto dell'accumulatore 1-L è diverso da "0". Il flusso lineare delle operazioni logiche viene ripreso alla destinazione di salto. La destinazione di salto viene indicata tramite un'etichetta. Il salto può essere eseguito sia in avanti che all'indietro. I salti sono eseguibili unicamente all'interno di un singolo blocco: l'operazione Loop di programma e la destinazione di salto devono cioè trovarsi nel medesimo blocco. Nel blocco può essere riportata un'unica destinazione di salto. L'ampiezza massima del salto è di -32768 / +32767 parole del codice programma. Il numero massimo di quante istruzioni possano effettivamente essere saltate dipende da come le istruzioni stesse sono state combinate internamente al programma (istruzioni ad una, due, tre parole).

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio di calcolo della facoltá di 5 (5!)

```
AWL                 Spiegazione
      L     L#1     //Carica in ACCU 1 la costante di numero imtero (a  32 bit).
      T     MD20    //Trasferisce il contenuto di ACCU 1 in MD20 (inizializzazione).
      L     5       //Carica il numero di cicli loop in ACCU 1-L.
NEXT: T     MW10    //Etichetta = inizio del loop/trasferisce ACCU 1-L al contatore loop.
      L     MD20
      *     D       //Moltiplica il contenuto attuale di MD20 per il contenuto attuale di MB10.
      T     MD20    //Trasferisce il risultato della moltiplicazione in MD20.
      L     MW10    //Carica il contenuto del contatore loop in ACCU 1
      LOOP  NEXT    //Decrementa il contenuto di ACCU 1, e salta all'etichetta di salto NEXT,
                    //se ACCU1-L > 0.
      L     MW24    //Il flusso delle operazioni logiche viene ripreso in questo punto a conclusione
                    //del loop.
      L     200
      >I
```

7 Funzioni con numeri in virgola fissa

7.1 Sommario delle operazioni matematiche con i numeri interi

Descrizione

Le descrizioni delle funzioni riportate alla tabella 9-1 mostrano che le operazioni matematiche combinano il contenuto degli accumulatori 1 e 2. Il valore viene depositato in ACCU 1. Il contenuto precedente di ACCU 1 viene traslato in ACCU 2. Il contenuto di ACCU 2 rimane invariato.

Nelle CPU con 4 accumulatori infine viene copiato il contenuto di ACCU 3 in ACCU 2 e quello di ACCU 4 in ACCU 3. Il vecchio contenuto di ACCU 4 rimane invariato.

Con le operazioni matematiche con i numeri interi, le seguenti funzioni possono essere eseguite con **due** numeri interi (16 bit, 32 bit):

- +I Somma ACCU 1 e ACCU 2 come numeri interi (a 16 bit)
- -I Sottrai ACCU 1 da ACCU 2 come numeri interi (a 16 bit)
- *I Moltiplica ACCU 1 per ACCU 2 come numeri interi (a 16 bit)
- /I Dividi ACCU 2 per ACCU 1 come numeri interi (a 16 bit)
- + Somma costante di numero intero (a 16, 32 bit)

- +D Somma ACCU 1 e ACCU 2 come numeri interi (a 32 bit)
- -D Sottrai ACCU 1 da ACCU 2 come numeri interi (a 32 bit)
- *D Moltiplica ACCU 1 per ACCU 2 come numeri interi (a 32 bit)
- /D Dividi ACCU 2 per ACCU 1 come numeri interi (a 32 bit)
- MOD Resto della divisione di numero intero (a 32 bit)

Vedere anche Valutazione dei bit nella parola di stato con operazioni in virgola fissa.

7.2 Valutazione dei bit nella parola di stato con operazioni con numeri in virgola fissa

Descrizione

Le operazioni di calcolo di base influenzano i bit seguenti nella parola di stato:

- A1 e A0
- OV
- OS

Le tabelle seguente riportano lo stato di segnale dei bit della parola di stato per i risultati delle operazioni con i numeri in virgola fissa (16 bit e 32 bit):

Campo di valori valido	**A1**	**A0**	**OV**	**OS**
0 (zero)	0	0	0	*
16 bit: –32 768 < =risultato < 0 (numero negativo) 32 bit: –2 147 483 648 <=risultato < 0 (numero negativo)	0	1	0	*
16 bit: 32 767 > =risultato > 0 (numero positivo) 32 bit: 2 147 483 647 > = risultato > 0 (numero positivo)	1	0	0	*

* Il bit OS non è influenzato dal risultato dell'operazione.

Campo di valori non valido	**A1**	**A0**	**OV**	**OS**
Superamento negativo del campo con addizione 16 bit: risultato = –65536 32 bit: risultato = –4 294 967 296	0	0	1	1
Superamento negativo del campo con moltiplicazione 16 bit: risultato < –32 768 (numero negativo) 32 bit: risultato < –2 147 483 648 (numero negativo)	0	1	1	1
Superamento positivo del campo con addizione, sottrazione 16 bit: risultato > 32 767 (numero positivo) 32 bit: risultato > 2 147 483 647 (numero positivo)	0	1	1	1
Superamento positivo del campo con moltiplicazione, divisione 16 bit: risultato > 32 767 (numero positivo) 32 bit: risultato > 2 147 483 647 (numero positivo)	1	0	1	1
Superamento negativo del campo con addizione, sottrazione 16 bit: risultato < –32 768 (numero negativo) 32 bit: risultato < –2 147 483 648 (numero negativo)	1	0	1	1
Divisione per zero	1	1	1	1

Operazione	**A1**	**A0**	**OV**	**OS**
+D: risultato = –4 294 967 296	0	0	1	1
/D o MOD: divisione per 0	1	1	1	1

7.3 +I Somma ACCU 1 e ACCU 2 come numeri interi (a 16 bit)

Formato

+I

Descrizione dell'operazione

+I (Somma numeri interi, a 16 bit) addiziona il contenuto dell'accumulatore 1-L al contenuto dell'accumulatore 2-L, e memorizza il risultato nell'accumulatore 1-L. I contenuti degli accumulatori 1-L e 2-L vengono interpretati come numeri interi (a 16 bit). L'operazione viene eseguita senza considerare o influenzare RLC. I bit di stato A1, A0, OS e OV vengono settati conformemente all'esito dell'operazione. Qualora si verifichi un overflow o un underflow, il risultato dell'operazione non è dato da un numero intero a 32 bit, bensì da un numero intero a 16 bit.

Nelle CPU con due accumulatori, il contenuto di ACCU 2 rimane inalterato.

Nelle CPU con quattro accumulatori, viene copiato il contenuto di ACCU 3 in ACCU 2, e di ACCU 4 in ACCU 3. Il contenuto di ACCU 4 rimane inalterato.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	x	x	x	x	-	-	-	-

Settaggio dei bit di stato	A1	A0	OV	OS
Totale = 0	0	0	0	-
-32768 <= Totale < 0	0	1	0	-
32767 >= Totale > 0	1	0	0	-
Totale = -65536	0	0	1	1
65534 >= Totale > 32767	0	1	1	1
-65535 <= Totale < -32768	1	0	1	1

Esempio

```
AWL             Spiegazione
L     EW10      //Il valore di EW10 viene caricato in ACCU 1-L.
L     MW14      //Carica il contenuto di ACCU 1-L in ACCU 2-L. Carica il valore di MW14 in
                //ACCU 1-L.
+I              //Addiziona ACCU 2-L e ACCU 1-L, memorizza il risultato in ACCU 1-L.
T     DB1.DBW25 //Il contenuto di ACCU 1-L (risultato) viene trasferito a DBW25 in DB1.
```

7.4 -I Sottrai ACCU 1 da ACCU 2 come numeri interi (a 16 bit)

Formato

-I

Descrizione dell'operazione

-I (Sottrai numeri interi, a 16 bit) sottrae il contenuto dell'accumulatore 1-L dal contenuto dell'accumulatore 2-L, e memorizza il risultato in ACCU 1-L. I contenuti degli accumulatori 1-L e 2-L vengono interpretati come numeri interi (a 16 bit). L'operazione viene eseguita senza considerare o influenzare RLC. I bit di stato A1, A0, OS e OV vengono settati conformemente all'esito dell'operazione. Qualora si verifichi un overflow o un underflow, il risultato dell'operazione non è dato da un numero intero a 32 bit, bensì da un numero intero a 16 bit.

Nelle CPU con due accumulatori, il contenuto di ACCU 2 rimane inalterato.

Nelle CPU con quattro accumulatori, viene copiato il contenuto di ACCU 3 in ACCU 2, e di ACCU 4 in ACCU 3. Il contenuto di ACCU 4 rimane inalterato.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	x	x	x	x	-	-	-	-

Settaggio dei bit di stato	A1	A0	OV	OS
Differenza = 0	0	0	0	-
-32768 <= Differenza < 0	0	1	0	-
32767 >= Differenza > 0	1	0	0	-
65535 >= Differenza > 32767	0	1	1	1
-65535 <= Differenza < -32768	1	0	1	1

Esempio

```
AWL              Spiegazione
L      EW10      //Il valore di EW10 viene caricato in ACCU 1-L.
L      MW14      //Carica il contenuto di ACCU 1-L in ACCU 2-L. Carica il valore di MW14 in
                 //ACCU 1-L.
-I               //Sottrae ACCU 1-L da ACCU 2-L , memorizza il risultato in ACCU 1-L.
T      DB1.DBW25 //Il contenuto di ACCU 1-L (risultato) viene trasferito a DBW25 in DB1.
```

7.5 *I Moltiplica ACCU 1 per ACCU 2 come numeri interi (a 16 bit)

Formato

***I**

Descrizione dell'operazione

***I** (Moltiplica numeri interi, a 16 bit) moltiplica il contenuto dell'accumulatore 2-L per il contenuto dell'accumulatore 1-L. I contenuti degli accumulatori 1-L e 2-L vengono interpretati come numeri interi (a 16 bit). Il risultato viene memorizzato come numero intero (a 32 bit) nell'accumulatore 1. Se i bit di stato OV1 = 1 e OS = 1, il risultato non rientra nell'area di validità definita per i numeri interi (a 16 bit).

L'operazione viene eseguita senza considerare o influenzare RLC. I bit di stato A1, A0, OS e OV vengono settati conformemente all'esito dell'operazione.

Nelle CPU con due accumulatori, il contenuto di ACCU 2 rimane inalterato.

Nelle CPU con quattro accumulatori, viene copiato il contenuto di ACCU 3 in ACCU 2, e di ACCU 4 in ACCU 3. Il contenuto di ACCU 4 rimane inalterato.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	x	x	x	x	-	-	-	-

Settaggio dei bit di stato	A1	A0	OV	OS
Risultato = 0	0	0	0	-
-32768 <= Risultato < 0	0	1	0	-
32767 >= Risultato > 0	1	0	0	-
1,073,741,824 >= Risultato > 32767	1	0	1	1
-1,073,709,056 <= Risultato < -32768	0	1	1	1

Esempio

```
AWL             Spiegazione
L     EW10      //Il valore di EW10 viene caricato in ACCU 1-L.
L     MW14      //Carica il contenuto di ACCU 1-L in ACCU 2. Carica il valore di MW14 in
                //ACCU 1-L.
*I              //Moltiplica ACCU 2-L e ACCU 1-L, memorizza il risultato in ACCU 1.
T     DB1.DBD25 //Il contenuto di ACCU 1 (risultato) viene trasferito a DBD25 in DB1.
```

7.6 /I Dividi ACCU 2 per ACCU 1 come numeri interi (a 16 bit)

Formato

/I

Descrizione dell'operazione

/I (Dividi numeri interi, a 16 bit) divide il contenuto dell'accumulatore 2-L per il contenuto dell'accumulatore 1-L. I contenuti degli accumulatori 1-L e 2-L vengono interpretati come numeri interi (a 16 bit). Il risultato viene memorizzato nell'accumulatore 1, ed è costituito da due numeri interi (a 16 bit): dal quoziente e dal resto della divisione. Il quoziente viene memorizzato nell'accumulatore 1-L, ed il resto della divisione nell'accumulatore 1-H. L'operazione viene eseguita senza considerare o influenzare RLC. I bit di stato A1, A0, OS e OV vengono settati conformemente all'esito dell'operazione.

Nelle CPU con due accumulatori, il contenuto di ACCU 2 rimane inalterato.

Nelle CPU con quattro accumulatori, viene copiato il contenuto di ACCU 3 in ACCU 2, e di ACCU 4 in ACCU 3. Il contenuto di ACCU 4 rimane inalterato.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	x	x	x	x	-	-	-	-

Settaggio dei bit di stato	A1	A0	OV	OS
Quoziente = 0	0	0	0	-
-32768 <= Quoziente < 0	0	1	0	-
32767 >= Quoziente > 0	1	0	0	-
Quoziente = 32768	1	0	1	1
Divisione per zero	1	1	1	1

Esempio

```
AWL           Spiegazione
L     EW10    //Il valore di EW10 viene caricato in ACCU 1-L.
L     MW14    //Carica il contenuto di ACCU 1-L in ACCU 2-L. Carica il valore di MW14 in
              //ACCU 1-L.
/I            //Divide ACCU 2-L per ACCU 1-L, memorizza il risultato in ACCU 1: ACCU 1-L:
              quoziente, ACCU 1-H: resto della divisione
T     MD20    //Il contenuto di ACCU 1 (risultato) viene trasferito a MD20.
```

Esempio numerico: 13 diviso 4

Contenuto di ACCU 2-L prima dell'operazione (EW10): "13"

Contenuto di ACCU 1-L prima dell'operazione (MW14): "4"

Operazione /I (ACCU2-L / ACCU1-L): "13/4"

Contenuto di ACCU 1-L dopo l'operazione (quoziente): "3"

Contenuto di ACCU 1-H dopo l'operazione (resto della divisione): "1"

7.7 + Somma costante di numero intero (a 16, 32 bit)

Formato

+ <costante di numero intero>

Operando	Tipo dati	Descrizione
<costante di numero intero>	(numero intero a 16 o 32 bit)	Costante da addizionare

Descrizione dell'operazione

+ < Costante di numero intero> addiziona una costante di numero intero al contenuto dell'accumulatore 1; e memorizza il risultato nell'accumulatore 1. L'operazione viene eseguita senza considerare o influenzare i bit di stato.

+ < Costante di numero intero, a 16 bit> addiziona una costante di numero intero (a 16 bit) (area di validità compresa tra -32768 e +32767) al contenuto dell'accumulatore 1-L; e memorizza il risultato nell'accumulatore 1-L.

+ < Costante di numero intero, a 32 bit> addiziona una costante di numero intero (a 32 bit) (area di validità compresa tra -2,147,483,648 e 2,147,483,647) al contenuto dell'accumulatore 1, e memorizza il risultato nell'accumulatore 1.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempi 1

```
AWL             Spiegazione
L     EW10      //Carica il valore di EW10 in ACCU 1-L.
L     MW14      //Carica il contenuto di ACCU1-L in ACCU2-L. Carica il valore di MW14 in
                //ACCU 1-L.
+I              //Addiziona ACCU 2-L e ACCU 1-L, e memorizza il risultato in ACCU  1-L.
+     25        //Addiziona ACCU 1-L e '25', e memorizza il risultato in ACCU 1-L.
T     DB1.DBW25 //Trasferisce il contenuto da ACCU 1-L (risultato) a DBW25 di DB 1.
```

Esempi 2

```
AWL             Spiegazione
L     EW12
L     EW14
+     100       //Addiziona ACCU 1-L e '100', memorizza il risultato in ACCU 1.
>I              //Se ACCU 2 > ACCU 1, ovvero EW 12 > (EW14 + 100),
SPB   NEXT      //Passa all'etichetta di salto NEXT.
```

Esempi 3

```
AWL             Spiegazione
L     MD20
L     MD24
+D              //Addiziona ACCU 1 e 2, memorizza il risultato in ACCU 1.
+     L#-200    //Addiziona ACCU 1-L e -'200', memorizza il risultato in ACCU 1.
T     MD28
```

7.8 +D Somma ACCU 1 e ACCU 2 come numeri interi (a 32 bit)

Formato

+D

Descrizione dell'operazione

+D (Somma numeri interi, a 32 bit) addiziona il contenuto dell'accumulatore 1 al contenuto dell'accumulatore 2 e memorizza il risultato nell'accumulatore 1. I contenuti degli accumulatori 1 e 2 vengono interpretati come numeri interi (a 32 bit). L'operazione viene eseguita senza considerare o influenzare RLC. I bit di stato A1, A0, OS e OV vengono settati conformemente all'esito dell'operazione.

Nelle CPU con due accumulatori, il contenuto di ACCU 2 rimane inalterato.

Nelle CPU con quattro accumulatori, viene copiato il contenuto di ACCU 3 in ACCU 2, e di ACCU 4 in ACCU 3. Il contenuto di ACCU 4 rimane inalterato.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	x	x	x	x	-	-	-	-

Settaggio dei bit di stato	A1	A0	OV	OS
Totale = 0	0	0	0	-
-2,147,483,648 <= Totale < 0	0	1	0	-
2,147,483,647 >= Totale > 0	1	0	0	-
Totale = -4,294,967,296	0	0	1	1
4,294,967,294 >= Totale > 2,147,483,647	0	1	1	1
-4,294,967,295 <= Totale < -2,147,483,648	1	0	1	1

Esempio

```
L     ED10          //Il valore di ED10 viene caricato in ACCU.
L     MD14          //Carica il contenuto di ACCU 1 in ACCU 2. Carica il valore di MD14 in
                    //ACCU 1.
+D                  //Addiziona ACCU 2 e ACCU 1, memorizza il risultato in ACCU 1.
T     DB1.DBD25     //Il contenuto di ACCU 1 (risultato) viene trasferito a DBD25 in DB1.
```

7.9 -D Sottrai ACCU 1 da ACCU 2 come numeri interi (a 32 bit)

Formato

-D

Descrizione dell'operazione

-D (Sottrai numeri interi, a 32 bit) sottrae il contenuto dell'accumulatore 1 dal contenuto dell'accumulatore 2 e memorizza il risultato nell'accumulatore 1. I contenuti degli accumulatori 1 e 2 vengono interpretati come numeri interi (a 32 bit). L'operazione viene eseguita senza considerare o influenzare RLC. I bit di stato A1, A0, OS e OV vengono settati conformemente all'esito dell'operazione.

Nelle CPU con due accumulatori, il contenuto di ACCU 2 rimane inalterato.

Nelle CPU con quattro accumulatori, viene copiato il contenuto di ACCU 3 in ACCU 2, e di ACCU 4 in ACCU 3. Il contenuto di ACCU 4 rimane inalterato.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	x	x	x	x	-	-	-	-

Settaggio dei bit di stato	A1	A0	OV	OS
Differenza = 0	0	0	0	-
-2,147,483,648 <= Differenza < 0	0	1	0	-
2,147,483,647 >= Differenza > 0	1	0	0	-
4,294,967,295 >= Differenza > 2,147,483,647	0	1	1	1
-4,294,967,295 <= Differenza < -2,147,483,648	1	0	1	1

Esempio

```
AWL             Spiegazione
L      ED10     //Il valore di ED10 viene caricato in ACCU 1.
L      MD14     //Carica il contenuto di ACCU 1 in ACCU 2. Carica il valore di MD14 in
                //ACCU 1.
-D              //Sottrae ACCU 2 da ACCU 1, memorizza il risultato in ACCU 1.
T      DB1.DBD25 //Il contenuto di ACCU 1 (risultato) viene trasferito a DBD25 in DB1.
```

7.10 *D Moltiplica ACCU 1 per ACCU 2 come numeri interi (a 32 bit)

Formato

***D**

Descrizione dell'operazione

***D** (Moltiplica numeri interi, a 32 bit) moltiplica il contenuto di ACCU 1 per il contenuto dell'accumulatore 2. I contenuti degli accumulatori 1 e 2 vengono interpretati come numeri interi (a 32 bit). Il risultato viene memorizzato come numero intero (a 32 bit) nell'accumulatore 1. Se i bit di stato OV1 = 1 e OS = 1, il risultato non rientra nell'area di validità definita per i numeri interi (a 32 bit).

L'operazione viene eseguita senza considerare o influenzare RLC. I bit di stato A1, A0, OS e OV vengono settati conformemente all'esito dell'operazione.

Nelle CPU con due accumulatori, il contenuto di ACCU 2 rimane inalterato.

Nelle CPU con quattro accumulatori, viene copiato il contenuto di ACCU 3 in ACCU 2, e di ACCU 4 in ACCU 3. Il contenuto di ACCU 4 rimane inalterato.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	x	x	x	x	-	-	-	-

Settaggio dei bit di stato	A1	A0	OV	OS
Risultato = 0	0	0	0	-
-2,147,483,648 <= Risultato < 0	0	1	0	-
2,147,483,647 >= Risultato > 0	1	0	0	-
Risultato > 2,147,483,647	1	0	1	1
Risultato < -2,147,483,648	0	1	1	1

Esempio

```
AWL              Spiegazione
L     ED10       //Il valore di ED10 viene caricato in ACCU 1.
L     MD14       //Carica il contenuto di ACCU 1 in ACCU 2. Carica il valore di MD14 in
                 //ACCU 1.
*D               //Moltiplica ACCU 2 e 1, memorizza il risultato in ACCU  1.
T     DB1.DBD25  //Il contenuto di ACCU 1 (risultato) viene trasferito a DBD25 in DB1.
```

7.11 /D Dividi ACCU 2 per ACCU 1 come numeri interi (a 32 bit)

Formato

/D

Descrizione dell'operazione

/D (Dividi numeri interi, a 32 bit) divide il contenuto dell'accumulatore 2 per il contenuto dell'accumulatore 1. I contenuti degli accumulatori 1 e 2 vengono interpretati come numeri interi (a 32 bit). Il risultato viene memorizzato nell'accumulatore 1. Il risultato riporta solamente il quoziente e non il resto (per ottenere il resto, avvalersi dell'operazione MOD).

L'operazione viene eseguita senza considerare o influenzare RLC. I bit di stato A1, A0, OS e OV vengono settati conformemente all'esito dell'operazione.

Nelle CPU con due accumulatori, il contenuto di ACCU 2 rimane inalterato.

Nelle CPU con quattro accumulatori, viene copiato il contenuto di ACCU 3 in ACCU 2, e di ACCU 4 in ACCU 3. Il contenuto di ACCU 4 rimane inalterato.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	x	x	x	x	-	-	-	-

Settaggio dei bit di stato	A1	A0	OV	OS
Quoziente = 0 *)	0	0	0	-
-2147483648 <= Quoziente < 0	0	1	0	-
2147483647 >= Quoziente > 0	1	0	0	-
Quoziente = 2147483648	1	0	1	1
Divisione per zero	1	1	1	1

*) Quoziente = 0 si verifica se il dividendo è = 0 e il divisore è <> 0, o se il valore del dividendo è inferiore al valore del divisore.

Esempio

```
AWL          Spiegazione
L      ED10  //Il valore di ED10 viene caricato in ACCU 1.
L      MD14  //Carica il contenuto di ACCU 1 in ACCU 2. Carica il valore di MD14 in ACCU 1.
/D           //Divide ACCU 2 per ACCU 1, memorizza il risultato (quoziente) in ACCU 1.
T      MD20  //Il contenuto di ACCU 1 (risultato) viene trasferito a MD20.
```

Esempio numerico: 13 diviso 4

Contenuto di ACCU 2 prima dell'operazione (ED10): "13"

Contenuto di ACCU 1 dopo l'operazione (MD14): "4"

Operazione /D (ACCU 2 / ACCU 1): "13/4"

Contenuto di ACCU 1 dopo l'operazione (quoziente): "3"

7.12 MOD Resto della divisione di numero intero (a 32 bit)

Formato

MOD

Descrizione dell'operazione

MOD (Resto della divisione di numero intero, a 32 bit) divide il contenuto dell'accumulatore 2 per il contenuto dell'accumulatore 1. I contenuti degli accumulatori 1 e 2 vengono interpretati come numeri interi (a 32 bit). Il risultato viene memorizzato nell'accumulatore 1. Il risultato riporta solamente il resto della divisione e non il quoziente (per ottenere il quoziente, avvalersi dell'operazione /D).

L'operazione viene eseguita senza considerare o influenzare RLC. I bit di stato A1, A0, OS e OV vengono settati conformemente all'esito dell'operazione.

Nelle CPU con due accumulatori, il contenuto di ACCU 2 rimane inalterato.

Nelle CPU con quattro accumulatori, viene copiato il contenuto di ACCU 3 in ACCU 2, e di ACCU 4 in ACCU 3. Il contenuto di ACCU 4 rimane inalterato.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	x	x	x	x	-	-	-	-

Settaggio dei bit di stato	A1	A0	OV	OS
Resto = 0	0	0	0	-
-2147483648 <= Resto < 0	0	1	0	-
2147483647 >= Resto > 0	1	0	0	-
Divisione per zero	1	1	1	1

Esempio

```
AWL         Spiegazione
L     ED10  //Il valore di ED10 viene caricato in ACCU 1.
L     MD14  //Carica il contenuto di ACCU 1 in ACCU 2. Carica il valore di MD14 in ACCU 1.
MOD         //Divide ACCU 2 per ACCU 1, memorizza il risultato (resto della divisione) in ACCU 1.
T     MD20  //Il contenuto di ACCU 1 (risultato) viene trasferito a MD20.
```

Esempio numerico: 13 diviso 4

Contenuto di ACCU 2 prima dell'operazione (ED10): "13"

Contenuto di ACCU 1 prima dell'operazione (MD14): "4"

Operazione /D (ACCU 2 / ACCU 1): "13/4"

Contenuto di ACCU 1 dopo l'operazione (resto della divisione): "1"

8 Funzioni con numeri in virgola mobile

8.1 Sommario delle operazioni con numeri in virgola mobile

Descrizione

Le operazioni matematiche combinano il contenuto degli accumulatori 1 e 2. Il valore viene depositato in ACCU 1. Il contenuto precedente di ACCU 1 viene traslato in ACCU 2. Il contenuto di ACCU 2 rimane invariato.

Nelle CPU con 4 accumulatori infine viene copiato il contenuto di ACCU 3 in ACCU 2 e quello di ACCU 4 in ACCU 3. Il vecchio contenuto di ACCU 4 rimaneinvariato.

I numeri IEEE 754 in virgola mobile a 32 bit appartengono al tipo di dati denominato ”REAL”. Si possono adoperare le operazioni con numeri in virgola mobile per effettuare le seguenti operazioni adoperando **due** numeri IEEE 754 in virgola mobile a 32 bit:

- +R Somma ACCU 1 e ACCU 2
- -R Sottrai ACCU 1 da ACCU 2
- *R Moltiplica ACCU 1 per ACCU 2
- /R Dividi ACCU 2 per ACCU 1

Le seguenti funzioni possono essere eseguite con **un** numero in virgola mobile (32 bit, IEEE 754):

- ABS Valore assoluto
- EXP Formazione del valore esponenziale
- LN Formazione del logaritmo naturale
- SQR Formazione del quadrato
- SQRT Formazione della radice quadrata

- SIN Formazione del seno di un angolo
- COS Formazione del coseno
- TAN Formazione della tangente
- ASIN Formazione dell'arcoseno
- ACOS Formazione dell'arcocoseno
- ATAN Formazione dell'arcotangente

8.2 Valutazione dei bit nella parola di stato con operazioni in virgola mobile

Descrizione

Le operazioni di calcolo di base influenzano i seguenti bit nella parola di stato:

- A1 e A0
- OV
- OS

Le tabelle seguente riportano lo stato di segnale dei bit della parola di stato per i risultati delle operazioni con i numeri in virgola mobile (a 32 bit):

Campo di validità	**A1**	**A0**	**OV**	**OS**
+0, -0 (zero)	0	0	0	*
-3.402823E+38 < risultato < -1.175494E-38 (numero negativo)	0	1	0	*
+1.175494E-38 < risultato < +3.402823E+38 (numero positivo)	1	0	0	*

* Il bit OS non è influenzato dal risultato dell'operazione.

Campo di non validità	**A1**	**A0**	**OV**	**OS**
Superamento negativo di capacità -1.175494E-38 < risultato < -1.401298E-45 (numero negativo)	0	0	1	1
superamento negativo di capacità +1.401298E-45 < risultato < +1.175494E-38 (numero positivo)	0	0	1	1
Overflow Risultato < -3.402823E+38 (numero negativo)	0	1	1	1
Overflow Risultato > 3.402823E+38 (numero positivo)	1	0	1	1
Numero in virgola mobile non valido od operazione non permessa (valore di ingresso al di fuori del campo di validità dei valori)	1	1	1	1

8.3 Operazioni di base

8.3.1 +R Somma ACCU 1 e ACCU 2 come numeri in virgola mobile (a 32 bit)

Formato

+R

Descrizione dell'operazione

+R (Somma due numeri in virgola mobile, a 32 bit, IEEE 754) addiziona il contenuto dell'accumulatore 1 al contenuto dell'accumulatore 2, e memorizza il risultato nell'accumulatore 1. I contenuti degli accumulatori 1 e 2 vengono interpretati come numeri in virgola mobile (a 32 bit, IEEE 754). L'operazione viene eseguita senza considerare o influenzare il RLC. I bit di stato A1, A0, OS e OV vengono settati conformemente all'esito dell'operazione.

Nelle CPU con due accumulatori, il contenuto di ACCU 2 rimane inalterato.

Nelle CPU con quattro accumulatori, viene copiato il contenuto di ACCU 3 in ACCU 2, e di ACCU 4 in ACCU 3. Il contenuto di ACCU 4 rimane inalterato.

Risultato

Risultato in ACCU 1	A1	A0	OV	OS	Nota
+qNaN	1	1	1	1	
+infinito	1	0	1	1	Overflow
+normalizzato	1	0	0	-	
+denormalizzato	0	0	1	1	Underflow
+zero	0	0	0	-	
-zero	0	0	0	-	
-denormalizzato	0	0	1	1	Underflow
-normalizzato	0	1	0	-	
-infinito	0	1	1	1	Overflow
-qNaN	1	1	1	1	

Parola di statou

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	x	x	x	x	-	-	-	-

Esempio

```
AWL             Spiegazione
AUF   DB10
L     ED10      //Il valore di ED10 viene caricato in ACCU 1.
L     MD14      //Carica il contenuto di ACCU 1 in ACCU 2. Carica il valore di MD14 in ACCU 1.
+R              //Addiziona ACCU 2 e 1, memorizza il risultato in ACCU  1.
T     DBD25     //Il contenuto di ACCU 1 (risultato) viene trasferito a DBD25 in DB10.
```

8.3.2 -R Sottrai ACCU 1 da ACCU 2 come numeri in virgola mobile (a 32 bit)

Formato

-R

Descrizione dell'operazione

-R (Sottrai numeri in virgola mobile, a 32 bit, IEEE 754) sottrae il contenuto dell'accumulatore 1 dal contenuto dell'accumulatore 2 e memorizza il risultato nell'accumulatore 1. I contenuti degli accumulatori 1 e 2 vengono interpretati come numeri in virgola mobile (a 32 bit, IEEE 754). Il risultato viene memorizzato nell'accumulatore 1. L'operazione viene eseguita senza considerare o influenzare RLC. I bit di stato A1, A0, OS e OV vengono settati conformemente all'esito dell'operazione.

Nelle CPU con due accumulatori, il contenuto di ACCU 2 rimane inalterato.

Nelle CPU con quattro accumulatori, viene copiato il contenuto di ACCU 3 in ACCU 2, e di ACCU 4 in ACCU 3. Il contenuto di ACCU 4 rimane inalterato.

Risultato

Risultato in ACCU 1	A1	A0	OV	OS	Nota
+qNaN	1	1	1	1	
+infinito	1	0	1	1	Overflow
+normalizzato	1	0	0	-	
+denormalizzato	0	0	1	1	Underflow
+zero	0	0	0	-	
-zero	0	0	0	-	
-denormalizzato	0	0	1	1	Underflow
-normalizzato	0	1	0	-	
-infinito	0	1	1	1	Overflow
-qNaN	1	1	1	1	

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	x	x	x	x	-	-	-	-

Esempio

```
AWL             Spiegazione
AUF    DB10
L      ED10     //Il valore di ED10 viene caricato in ACCU 1.
L      MD14     //Carica il contenuto di ACCU 1 in ACCU 2. Carica il valore di MD14 in ACCU 1.
-R              //Sottrae ACCU 2 da ACCU 1, memorizza il risultato in ACCU 1.
T      DBD25    //Il contenuto di ACCU 1 (risultato) viene trasferito a DBD25 in DB10.
```

8.3.3 *R Moltiplica ACCU 1 per ACCU 2 come numeri in virgola mobile (a 32 bit)

Formato

***R**

Descrizione dell'operazione

***R** (Moltiplica numeri in virgola mobile, 32 bit, IEEE 754) moltiplica il contenuto dell'accumulatore 2 per il contenuto dell'accumulatore 1. I contenuti degli accumulatori 1 e 2 vengono interpretati come numeri in virgola mobile (a 32 bit, IEEE 754). Il risultato viene memorizzato nell'accumulatore 1 come numero in virgola mobile (a 32 bit, IEEE 754). L'operazione viene eseguita senza considerare o influenzare RLC. I bit di stato A1, A0, OS e OV vengono settati conformemente all'esito dell'operazione.

Nelle CPU con due accumulatori, il contenuto di ACCU 2 rimane inalterato.

Nelle CPU con quattro accumulatori, viene copiato il contenuto di ACCU 3 in ACCU 2, e di ACCU 4 in ACCU 3. Il contenuto di ACCU 4 rimane inalterato.

Risultato

Risultato in ACCU 1	A1	A0	OV	OS	Nota
+qNaN	1	1	1	1	
+ infinito	1	0	1	1	Overflow
+ normalizzato	1	0	0	-	
+ denormalizzato	0	0	1	1	Underflow
+ zero	0	0	0	-	
- zero	0	0	0	-	
- denormalizzato	0	0	1	1	Underflow
- normalizzato	0	1	0	-	
- infinito	0	1	1	1	Overflow
-qNaN	1	1	1	1	

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	x	x	x	x	-	-	-	-

Esempio

```
AWL             Spiegazione
AUF    DB10
L      ED10     //Il valore di ED10 viene caricato in ACCU 1.
L      MD14     //Carica il contenuto di ACCU 1 in ACCU 2. Carica il valore di MD14 in ACCU 1.
*R              //Moltiplica ACCU 2 e 1, memorizza il risultato in ACCU  1.
T      DBD25    //Il contenuto di ACCU 1 (risultato) viene trasferito a DBB25.
```

8.3.4 /R Dividi ACCU 2 per ACCU 1 come numeri in virgola mobile (a 32 bit)

Formato

/R

Descrizione dell'operazione

/R (Dividi numeri in virgola mobile, 32 bit, IEEE 754) divide il contenuto dell'accumulatore 2 per il contenuto dell'accumulatore 1. I contenuti degli accumulatori 1 e 2 vengono interpretati come numeri in virgola mobile (a 32 bit, IEEE 754). L'operazione viene eseguita senza considerare o influenzare RLC. I bit di stato A1, A0, OS e OV vengono settati conformemente all'esito dell'operazione.

Nelle CPU con due accumulatori, il contenuto di ACCU 2 rimane inalterato.

Nelle CPU con quattro accumulatori, viene copiato il contenuto di ACCU 3 in ACCU 2, e di ACCU 4 in ACCU 3. Il contenuto di ACCU 4 rimane inalterato.

Risultato

Risultato in ACCU 1	A1	A0	OV	OS	Nota
+qNaN	1	1	1	1	
+infinito	1	0	1	1	Overflow
+normalizzato	1	0	0	-	
+denormalizzato	0	0	1	1	Underflow
+zero	0	0	0	-	
-zero	0	0	0	-	
-denormalizzato	0	0	1	1	Underflow
-normalizzato	0	1	0	-	
-infinito	0	1	1	1	Overflow
-qNaN	1	1	1	1	

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	x	x	x	x	-	-	-	-

Esempio

```
AWL             Spiegazione
AUF    DB10
L      ED10     //Il valore di ED10 viene caricato in ACCU 1.
L      MD14     //Carica il contenuto di ACCU 1 in ACCU 2. Carica il valore di MD14 in ACCU 1.
/R              //Divide ACCU 2 per ACCU 1, memorizza il risultato in ACCU 1.
T      DBD20    //Il contenuto di ACCU 1 (risultato) viene trasferito a DBD20 in DB10.
```

8.3.5 ABS Valore assoluto di un numero in virgola mobile (a 32 bit, IEEE 754)

Formato

ABS

Descrizione dell'operazione

ABS (Valore assoluto di un numero in virgola mobile, a 32 bit, IEEE 754) definisce il valore assoluto di un numero in virgola mobile (a 32 bit, IEEE 754) nell'accumulatore 1. Il risultato viene memorizzato nell'accumulatore 1. L'operazione viene eseguita senza considerare o influenzare i bit di stato.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio

```
AWL             Spiegazione
L     ED8       //Carica il valore in ACCU 1 (esempio: ED8 = -1.5E+02).
ABS             //Definisce il valore assoluto, memorizza il risultato in ACCU 1.
T     MD10      //Trasferisce il risultato a MD10 (esempio: risultato = 1.5E+02).
```

8.4 Operazioni avanzate

8.4.1 SQR Formazione del quadrato di un numero in virgola mobile (a 32 bit)

Formato

SQR

Descrizione dell'operazione

SQR (Formazione del quadrato di un numero in virgola mobile, a 32 bit, IEEE 754) calcola appunto il quadrato di un numero in virgola mobile (a 32 bit, IEEE 754) in ACCU 1. Il risultato viene memorizzato in ACCU 1. L'operazione influenza i bit A1, A0, OV, e OS, della parola di stato.

Il contenuto di ACCU 2 (per le CPU con quattro accumulatori; anche il contenuto di ACCU 3 e ACCU 4) rimane inalterato.

Vedere anche Valutazione dei bit nella parola di stato con operazioni in virgola mobile.

Risultato

Il risultato in ACCU 1 è	A1	A0	OV	OS	Commento
+qNaN	1	1	1	1	
+infinito	1	0	1	1	overflow
+normalizzato	1	0	0	-	
+denormalizzato	0	0	1	1	underflow
+zero	0	0	0	-	
-qNaN	1	1	1	1	

Esempio

```
AWL                 Spiegazione
      AUF   DB17    //Apre blocco dati DB17.
      L     DBD0    //Il valore della doppia parola dati DBD0 viene caricato in ACCU 1. (Questo
                    //valore deve avere formato di numero in virgola mobile.)
      SQR           //Calcola il quadrato di un numero in virgola mobile (a 32 bit, IEEE 754)
                    //in ACCU 1. Il risultato viene memorizzato in ACCU 1.
      UN    OV      //Interroga sullo "0" il bit OV della parola di stato.
      SPB   OK      //Se non si sono verificati errori nell'operazione SQR, salta all'etichetta
                    //OK.
BEA                 //Fine blocco incondizionato, se si sono verificati errori nell'operazione
                    //SQR.
OK:   T     DBD4    //Trasferisce il risultato di ACCU 1 nella doppia parola dati DBD4.
```

8.4.2 SQRT Formazione della radice quadrata di un numero in virgola mobile (a 32 bit)

Formato

SQRT

Descrizione dell'operazione

SQRT (Formazione della radice quadrata di un numero in virgola mobile, a 32 bit, IEEE 754) calcola appunto la radice quadrata di un numero in virgola mobile (a 32 bit, IEEE 754) in ACCU 1. Il risultato viene memorizzato in ACCU 1. Il valore di ingresso deve essere maggiore o uguale a zero. In questo caso, il risultato è positivo. Unica eccezione: La radice quadrata di -0 è -0. L'operazione influenza i bit A1, A0, OV, e OS, della parola di stato.

Il contenuto di ACCU 2 (per le CPU con quattro accumulatori; anche il contenuto di ACCU 3 e ACCU 4) rimane inalterato.

Vedere anche Valutazione dei bit nella parola di stato con operazioni in virgola mobile.

Risultato

Il risultato in ACCU 1 è	A1	A0	OV	OS	Commento
+qNaN	1	1	1	1	
+infinito	1	0	1	1	overflow
+normalizzato	1	0	0	-	
+denormalizzato	0	0	1	1	underflow
+zero	0	0	0	-	
-zero	0	0	0	-	
-qNaN	1	1	1	1	

Esempio

```
AWL                 Spiegazione
      L     MD10    //Il valore della doppia parola dati MD10 viene caricato in ACCU 1.
                    //(Questo valore deve avere formato di numero in virgola mobile.)
      SQRT          //Calcola la radice quadrata di un numero in virgola mobile
                    //(a 32 bit, IEEE 754) in ACCU 1. Il risultato viene memorizzato in
                    //ACCU 1.
      UN    OV      //Interroga sullo "0" il bit OV della parola di stato.
      SPB   OK      //Se non si sono verificati errori nell'operazione SQRT, salta all'etichetta
                    //OK.
BEA                 //Fine blocco incondizionato, se si sono verificati errori nell'operazione
                    //SQRT.
OK:   T     MD20    //Trasferisce il risultato di ACCU 1 nella doppia parola dati MD20.
```

8.4.3 EXP Formazione del valore esponenziale di un numero in virgola mobile (a 32 bit)

Formato

EXP

Descrizione dell'operazione

EXP (Formazione del valore esponenziale di un numero in virgola mobile, a 32 bit, IEEE 754) calcola appunto il valore esponenziale (valore esponenziale di base e) di un numero in virgola mobile (a 32 bit, IEEE 754) in ACCU 1. Il risultato viene memorizzato in ACCU 1. L'operazione influenza i bit A1, A0, OV, e OS, della parola di stato.

Il contenuto di ACCU 2 (per le CPU con quattro accumulatori; anche il contenuto di ACCU 3 e ACCU 4) rimane inalterato.

Vedere anche Valutazione dei bit nella parola di stato con operazioni in virgola mobile.

Risultato

Il risultato in ACCU 1 è	A1	A0	OV	OS	Commento
+qNaN	1	1	1	1	
+infinito	1	0	1	1	overflow
+normalizzato	1	0	0	-	
+denormalizzato	0	0	1	1	underflow
+zero	0	0	0	-	
-qNaN	1	1	1	1	

Esempio

```
AWL                 Spiegazione
L           MD10    //Il valore della doppia parola merker MD10 viene caricato in ACCU 1. (Questo
                    //valore deve avere formato di numero in virgola mobile.)
EXP                 //Calcola il valore esponenziale di base e di un numero in virgola mobile
                    //(a 32 bit, IEEE 754) in ACCU 1. Il risultato viene memorizzato in ACCU 1.
UN          OV      //Interroga sullo "0" il bit OV della parola di stato.
SPB         OK      //Se non si sono verificati errori nell'operazione EXP, salta all'etichetta
                    //OK.
BEA                 //Fine blocco incondizionato, se si sono verificati errori nell'operazione
                    //EXP.
OK:   T     MD20    //Trasferisce il risultato di ACCU 1 nella doppia parola merker MD20.
```

8.4.4 LN Formazione del logaritmo naturale di un numero in virgola mobile (a 32 bit)

Formato

LN

Descrizione dell'operazione

LN (Formazione del logaritmo naturale di un numero in virgola mobile, a 32 bit, IEEE 754) calcola appunto il logaritmo naturale (logaritmo di base e) di un numero in virgola mobile (a 32 bit, IEEE 754) in ACCU 1. Il risultato viene memorizzato in ACCU 1. Il valore di ingresso deve essere maggiore di zero. L'operazione influenza i bit A1, A0, OV, e OS, della parola di stato.

Il contenuto di ACCU 2 (per le CPU con quattro accumulatori; anche il contenuto di ACCU 3 e ACCU 4) rimane inalterato.

Risultato

Il risultato in ACCU 1 è	A1	A0	OV	OS	Commento
+qNaN	1	1	1	1	
+infinito	1	0	1	1	overflow
+normalizzato	1	0	0	-	
+denormalizzato	0	0	1	1	underflow
+zero	0	0	0	-	
+infinito	1	0	1	1	
-zero	0	0	0	-	
-denormalizzato	0	0	1	1	underflow
-normalizzato	0	1	0	-	
-infinito	0	1	1	1	overflow
-qNaN	1	1	1	1	

Esempio

```
AWL                 Spiegazione
      L     MD10    //Il valore della doppia parola dati MD10 viene caricato in ACCU 1. (Questo
                    //valore deve avere formato di numero in virgola mobile.)
      LN            //Calcola il logaritmo naturale di un numero in virgola mobile (a 32 bit,
                    //IEEE 754) in ACCU 1. Il risultato viene memorizzato in ACCU 1.
      UN    OV      //Interroga sullo "0" il bit OV della parola di stato.
      SPB   OK      //Se non si sono verificati errori nell'operazione LN, salta all'etichetta
                    //OK.
BEA                 //Fine blocco incondizionato, se si sono verificati errori nell'operazione
                    //LN.
OK:   T     MD20    //Trasferisce il risultato di ACCU 1 nella doppia parola dati MD20.
```

8.4.5 SIN Formazione del seno di un angolo come numero in virgola mobile (a 32 bit)

Formato

SIN

Descrizione dell'operazione

SIN (Formazione del seno di un angolo come numero in virgola mobile, a 32 bit, IEEE 754) calcola appunto il seno di un angolo, indicato in radianti. L'angolo deve essere presente in ACCU1 come numero in virgola mobile. Il risultato viene memorizzato in ACCU 1. L'operazione influenza i bit A1, A0, OV, e OS, della parola di stato.

Il contenuto di ACCU 2 (per le CPU con quattro accumulatori; anche il contenuto di ACCU 3 e ACCU 4) rimane inalterato.

Risultato

Il risultato in ACCU 1 è	A1	A0	OV	OS	Commento
+qNaN	1	1	1	1	
+normalizzato	1	0	0	-	
+denormalizzato	0	0	1	1	overflow
+zero	0	0	0	-	
+infinito	1	0	1	1	
-zero	0	0	0	-	
-denormalizzato	0	0	1	1	underflow
-normalizzato	0	1	0	-	
-qNaN	1	1	1	1	

Vedere anche Valutazione dei bit nella parola di stato con operazioni in virgola mobile.

Esempio

```
AWL             Spiegazione
L     MD10      //Il valore della doppia parola merker MD10 viene caricato in ACCU 1. (Questo
                //valore deve avere formato di numero in virgola mobile.)
SIN             //Calcola il seno di un numero in virgola mobile (a 32 bit, IEEE 754) in ACCU 1.
                //Il risultato viene memorizzato in ACCU  1.
T     MD20      //Trasferisce il risultato di ACCU 1 nella doppia parola merker MD20.
```

8.4.6 COS Formazione del coseno di un angolo come numero in virgola mobile (a 32 bit)

Formato

COS

Descrizione dell'operazione

COS (Formazione del coseno di un angolo come numero in virgola mobile, a 32 bit, IEEE 754) calcola appunto il coseno di un angolo, indicato in radianti. L'angolo deve essere presente in ACCU1 come numero in virgola mobile. Il risultato viene memorizzato in ACCU 1. L'operazione influenza i bit A1, A0, OV, e OS, della parola di stato.

Il contenuto di ACCU 2 (per le CPU con quattro accumulatori; anche il contenuto di ACCU 3 e ACCU 4) rimane inalterato.

Risultato

Il risultato in ACCU 1 è	A1	A0	OV	OS	Commento
+qNaN	1	1	1	1	
+normalizzato	1	0	0	-	
+denormalizzato	0	0	1	1	overflow
+zero	0	0	0	-	
-zero	0	0	0	-	
-denormalizzato	0	0	1	1	underflow
-normalizzato	0	1	0	-	
-qNaN	1	1	1	1	

Esempio

```
AWL             Spiegazione
L     MD10      //Il valore della doppia parola merker MD10 viene caricato in ACCU 1. (Questo
                //valore deve avere formato di numero in virgola mobile.)
COS             //Calcola il coseno di un numero in virgola mobile (a 32 bit, IEEE 754) in ACCU 1.
                //Il risultato viene memorizzato in ACCU  1.
T     MD20      //Trasferisce il risultato di ACCU 1 nella doppia parola merker MD20.
```

8.4.7 TAN Formazione della tangente di un angolo come numero in virgola mobile (a 32 bit)

Formato

TAN

Descrizione dell'operazione

TAN (Formazione della tangente di un angolo come numero in virgola mobile, a 32 bit, IEEE 754) calcola appunto la tangente di un angolo, indicato in radianti. L'angolo deve essere presente in ACCU1 come numero in virgola mobile. Il risultato viene memorizzato in ACCU 1. L'operazione influenza i bit A1, A0, OV, e OS, della parola di stato.

Il contenuto di ACCU 2 (per le CPU con quattro accumulatori; anche il contenuto di ACCU 3 e ACCU 4) rimane inalterato.

Risultato

Il risultato in ACCU 1 è	A1	A0	OV	OS	Commento
+qNaN	1	1	1	1	
+infinito	1	0	1	1	overflow
+normalizzato	1	0	0	-	
+denormalizzato	0	0	1	1	underflow
+zero	0	0	0	-	
-zero	0	0	0	-	
-denormalizzato	0	0	1	1	underflow
-normalizzato	0	1	0	-	
-infinito	0	1	1	1	overflow
-qNaN	1	1	1	1	

Esempio

```
AWL                 Spiegazione
L          MD10     //Il valore della doppia parola merker MD10 viene caricato in ACCU 1. (Questo
                    valore deve avere formato di numero in virgola mobile.)
TAN                 //Calcola la tangente di un numero in virgola mobile (a 32 bit, IEEE 754)
                    in ACCU 1. Il risultato viene memorizzato in ACCU 1.
UN         OV       //Interroga sullo "0" il bit OV della parola di stato.
SPB        OK       //Se non si sono verificati errori nell'operazione TAN, salta all'etichetta
                    OK.
BEA                 //Fine blocco incondizionato, se si sono verificati errori nell'operazione
                    TAN.
OK:   T    MD20     //Trasferisce il risultato di ACCU 1 nella doppia parola merker MD20.
```

8.4.8 ASIN Formazione dell'arcoseno di un numero in virgola mobile (a 32 bit)

Formato

ASIN

Descrizione dell'operazione

ASIN (Formazione dell'arcoseno di un numero in virgola mobile, a 32 bit, IEEE 754) calcola appunto l'arcoseno di un numero in virgola mobile. Area di valori ammessa per il valore di ingresso:

-1 <= valore di ingresso <= +1

Il risultato è un angolo che viene indicato in radianti. Il valore rientra nella seguente area:

$-\pi / 2 <=$ arcoseno (ACCU 1) $<= +\pi / 2$, con $\pi = 3{,}14159...$

L'operazione influenza i bit A1, A0, OV, e OS, della parola di stato.

Il contenuto di ACCU 2 (per le CPU con quattro accumulatori; anche il contenuto di ACCU 3 e ACCU 4) rimane inalterato.

Risultato

Il risultato in ACCU 1 è	A1	A0	OV	OS	Commento
+qNaN	1	1	1	1	
+normalizzato	1	0	0	-	
+denormalizzato	0	0	1	1	overflow
+zero	0	0	0	-	
-zero	0	0	0	-	
-denormalizzato	0	0	1	1	underflow
-normalizzato	0	1	0	-	
-qNaN	1	1	1	1	

Esempio

```
AWL                 Spiegazione
L          MD10     //Il valore della doppia parola merker MD10 viene caricato in ACCU 1. (Questo
                    //valore deve avere formato di numero in virgola mobile.)
ASIN                //Calcola l'arcoseno di un numero in virgola mobile (a 32 bit, IEEE 754)
                    //in ACCU 1. Il risultato viene memorizzato in ACCU  1.
UN         OV       //Interroga sullo "0" il bit OV della parola di stato.
SPB        OK       //Se non si sono verificati errori nell'operazione ASIN, salta all'etichetta
                    //OK.
BEA                 //Fine blocco incondizionato, se si sono verificati errori nell'operazione
                    //ASIN.
OK:   T    MD20     //Trasferisce il risultato di ACCU 1 nella doppia parola merker MD20.
```

8.4.9 ACOS Formazione dell'arcocoseno di un numero in virgola mobile (a 32 bit)

Formato

ACOS

Descrizione dell'operazione

ACOS (Formazione dell'arcocoseno di un numero in virgola mobile, a 32 bit, IEEE 754) calcola appunto l'arcocoseno di un numero in virgola mobile. Area di valori ammessa per il valore di ingresso:

-1 <= valore di ingresso <= +1

Il risultato è un angolo che viene indicato in radianti. Il valore rientra nella seguente area:

0 <= Arcocoseno (AKKU 1) <= π, con π = 3,14159...

L'operazione influenza i bit A1, A0, OV, e OS, della parola di stato.

Il contenuto di ACCU 2 (per le CPU con quattro accumulatori; anche il contenuto di ACCU 3 e ACCU 4) rimane inalterato.

Risultato

Il risultato in ACCU 1 è	A1	A0	OV	OS	Commento
+qNaN	1	1	1	1	
+normalizzato	1	0	0	-	
+denormalizzato	0	0	1	1	overflow
+zero	0	0	0	-	
-zero	0	0	0	-	
-denormalizzato	0	0	1	1	underflow
-normalizzato	0	1	0	-	
-qNaN	1	1	1	1	

Esempio

```
AWL                 Spiegazione
L          MD10     //Il valore della doppia parola merker MD10 viene caricato in ACCU 1. (Questo
                    //valore deve avere formato di numero in virgola mobile.)
ACOS                //Calcola l'arcoseno di un numero in virgola mobile (a 32 bit, IEEE 754)
                    //in ACCU 1. Il risultato viene memorizzato in ACCU  1.
UN         OV       //Interroga sullo "0" il bit OV della parola di stato.
SPB        OK       //Se non si sono verificati errori nell'operazione ASIN, salta all'etichetta
                    //OK.
BEA                 //Fine blocco incondizionato, se si sono verificati errori nell'operazione
                    //ACOS.
OK:   T    MD20     //Trasferisce il risultato di ACCU 1 nella doppia parola merker MD20.
```

8.4.10 ATAN Formazione dell'arcotangente di un numero in virgola mobile (a 32 bit)

Formato

ATAN

Descrizione dell'operazione

ATAN (Formazione dell'arcotangente di un numero in virgola mobile, a 32 bit, IEEE 754) calcola appunto l'arcotangente di un numero in virgola mobile in ACCU 1. Il risultato è un angolo che viene indicato in radianti. Il valore rientra nella seguente area:

$-\pi / 2 <=$ Arcotangente (ACCU 1) $<= +\pi / 2$, con $\pi = 3{,}14159...$

L'operazione influenza i bit A1, A0, OV, e OS, della parola di stato.

Il contenuto di ACCU 2 (per le CPU con quattro accumulatori; anche il contenuto di ACCU 3 e ACCU 4) rimane inalterato.

Risultato

Il risultato in ACCU 1 è	A1	A0	OV	OS	Commento
+qNaN	1	1	1	1	
+normalizzato	1	0	0	-	
+denormalizzato	0	0	1	1	overflow
+zero	0	0	0	-	
-zero	0	0	0	-	
-denormalizzato	0	0	1	1	underflow
-normalizzato	0	1	0	-	
-qNaN	1	1	1	1	

Esempio

```
AWL                 Spiegazione
L          MD10     //Il valore della doppia parola merker MD10 viene caricato in ACCU 1. (Questo
                    //valore deve avere formato di numero in virgola mobile.)
ATAN                //Calcola l'arcotangente di un numero in virgola mobile (a 32 bit, IEEE 754)
                    //in ACCU 1. Il risultato viene memorizzato in ACCU 1.
UN         OV       //Interroga sullo "0" il bit OV della parola di stato.
SPB        OK       //Se non si sono verificati errori nell'operazione ATAN, salta all'etichetta
                    //OK.
BEA                 //Fine blocco incondizionato, se si sono verificati errori nell'operazione
                    //ATAN.
OK:   T    MD20     //Trasferisce il risultato di ACCU 1 nella doppia parola merker MD20.
```

9 Operazioni di caricamento e trasferimento

9.1 Sommario delle operazioni di caricamento e trasferimento

Descrizione

Le operazioni L (Carica) e T (Trasferisci) abilitano l'utente a programmare uno scambio di informazioni tra unità di ingresso e uscita e aree di memoria oppure tra aree di memoria. La CPU esegue queste operazioni in ogni ciclo di scansione come operazioni incondizionate, vale a dire che queste operazioni non sono influenzate dal risultato logico combinatorio di un'istruzione.

Sono disponibili le seguenti operazioni di caricamennto e trasferimento:

- L Carica
- L STW Carica la parola di stato in ACCU 1
- LAR1 Carica il registro di indirizzo 1 con il contenuto di ACCU 1
- LAR1 <D> Carica il registro di indirizzo 1 con numero intero a 32 bit
- LAR1 AR2 Carica registro di indirizzo 1 con contenuto del registro di indirizzo 2
- LAR2 Carica il registro di indirizzo 2 con il contenuto di ACCU 1
- LAR2 <D> Carica il registro di indirizzo 2 con numero intero a 32 bit

- T Trasferisci
- T STW Trasferisci ACCU 1 nella parola di stato
- TAR Scambia registro di indirizzo 1 con registro di indirizzo 2
- TAR1 Trasferisci registro di indirizzo 1 in ACCU 1
- TAR1 <D> Trasferisci registro di indirizzo 1 all'indirizzo destinazione (a 32 bit)
- TAR1 AR2 Trasferisci registro di indirizzo 1 nel registro di indirizzo 2
- TAR2 Trasferisci registro di indirizzo 2 in ACCU 1
- TAR2 <D> Trasferisci registro di indirizzo 2 all'indirizzo di destinazione

9.2 L Carica

Formato

L <operando>

Parametro	Tipo dati	Area memoria	Indirizzo di sorgente
<operando>	BYTE WORD DWORD	E, A, PE, M, L, D, puntatore, parametro	0...65535 0...65534 0...65532

Descrizione dell'operazione

L < operando> carica nell'accumulatore 1 il byte, la parola o la doppia parola indirizzati, dopo che il contenuto di ACCU 1 è stato memorizzato in ACCU 2, ed ACCU 1 è stato resettato a 0.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio

```
AWL             Spiegazione
L     EB10      //Carica il byte di ingresso EB10 in ACCU 1-L-L.
L     MB120     //Carica il byte di merker MB120 in ACCU 1-L-L.
L     DBB12     //Carica il byte di dati DBB12 in ACCU 1-L-L.
L     DIW15     //Carica la parola dati di istanza DIW15 in ACCU 1-L.
L     LD252     //Carica la doppia parola di dati locali LD252 in ACCU 1.
L     P# E 8.7  //Carica il puntatore in ACCU1
L     OTTO      //Carica il parametro "OTTO" in ACCU1
L     P# ANNA   //Carica il puntatore nel parametro indicato in ACCU1. Questo comando carica
                //il relativo offset di indirizzo del parametro indicato. Per individuare negli
                //FB di multi-istanza l'offset assoluto nel blocco dati di multi-istanza, a
                //questo valore va ancora sommato il contenuto del registro AR2.
```

Contenuto dell'accumulatore 1

Contenuto	ACCU1-H-H	ACCU1-H-L	ACCU1-L-H	ACCU1-L-L
prima dell'esecuzione di caricamento	XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX
dopo l'esecuzione di **L MB10** (L < byte >)	00000000	00000000	00000000	<MB10>
dopo l'esecuzione di **L MW10** (L < parola >)	00000000	00000000	<MB10>	<MB11>
dopo l'esecuzione di **L MD10** (L < doppia parola >)	<MB10>	<MB11>	<MB12>	<MB13>
dopo l'esecuzione di **L P# ANNA** (nell'FB)	<86>	< bit offset di ANNA relativo all'inizio dell'FB > Per individuare negli FB di multi-istanza l'offset assoluto nel blocco dati di multi-istanza, a questo valore va ancora sommato il contenuto del registro AR2.		
dopo l'esecuzione di **L P# ANNA** (nella FC)	< Un indirizzo valido per più aree della data che viene trasmessa ad ANNA >			
	X = "1" o "0"			

9.3 L STW Carica la parola di stato in ACCU 1

Formato

L STW

Descrizione dell'operazione

L STW (Operazione L con operando STW) carica nell'accumulatore 1 il contenuto della parola di stato. L'operazione viene eseguita senza considerare o influenzare i bit di stato.

Avvertenza

Per le CPU della serie S7-300, i bit della parola di stato /ER, STA e OR non vengono caricati dall'istruzione **L STW**. Soltanto i bit 1, 4, 5, 6, 7 e 8 vengono caricati nelle corrispondenti posizioni di bit della parola bassa di ACCU 1.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio

```
AWL              Spiegazione
L     STW        //Carica il contenuto della parola di stato in ACCU 1.
```

Contenuto dell'accumulatore 1 dopo l'esecuzione di **L STW**:

Bit	31-9	8	7	6	5	4	3	2	1	0
Contenuto	0	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER

9.4 LAR1 Carica il registro di indirizzo 1 con il contenuto di ACCU 1

Formato

LAR1

Descrizione dell'operazione

LAR1 carica nel registro di indirizzo AR 1 il contenuto dell'accumulatore 1 (a 32 bit). Gli accumulatori 1 e 2 non vengono modificati. L'operazione viene eseguita senza considerare o influenzare i bit di stato.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

9.5 LAR1 <D> Carica il registro di indirizzo 1 con numero intero a 32 bit

Formato

LAR1 <D>

Parametro	Tipo di dati	Area di memoria	Indirizzo di sorgente
<D>	DWORD Costante puntatore	D, M, L	0...65532

Descrizione dell'operazione

LAR1 <D> carica nel registro di indirizzo AR 1 il contenuto della doppia parola <D> indirizzata. Gli accumulatori 1 e 2 non vengono modificati. L'operazione viene eseguita senza considerare o influenzare i bit di stato.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio: Indirizzi diretti

```
AWL              Spiegazione
LAR1   DBD 20    //Carica in AR 1 il puntatore della doppia parola DBD20.
LAR1   DID 30    //Carica in AR 1 il puntatore della doppia parola di istanza DID30.
LAR1   LD 180    //Carica in AR 1 il puntatore della doppia parola di dati locali LD180.
LAR1   MD 24     //Carica in AR 1 il puntatore della doppia parola di merker MD24.
```

Esempio: Costante puntatore

```
AWL              Spiegazione
LAR1   P#M100.0  //Carica in AR 1 una costante di puntatore a 32 bit.
```

9.6 LAR1 AR2 Carica registro di indirizzo 1 con contenuto del registro di indirizzo 2

Formato

LAR1 AR2

Descrizione dell'operazione

LAR1 AR2 (Operazione LAR1 con operando AR2) carica nel registro di indirizzo AR 1 il contenuto del registro di indirizzo AR2. Gli accumulatori 1 e 2 non vengono modificati. L'operazione viene eseguita senza considerare o influenzare i bit di stato.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

9.7 LAR2 Carica il registro di indirizzo 2 con il contenuto di ACCU 1

Formato

LAR2

Descrizione dell'operazione

LAR2 carica nel registro di indirizzo AR2 il contenuto dell'accumulatore 1 (a 32 bit).

Gli accumulatori 1 e 2 non vengono modificati. L'operazione viene eseguita senza considerare o influenzare i bit di stato.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

9.8 LAR2 <D> Carica il registro di indirizzo 2 con numero intero a 32 bit

Formato

LAR2 <D>

Parametro	Tipo di dati	Area di memoria	Indirizzo di sorgente
<D>	DWORD Costante puntatore	D, M, L	0...65532

Descrizione dell'operazione

LAR2 <D> carica nel registro di indirizzo AR 2 il contenuto della doppia parola indirizzata <D>. Gli accumulatori 1 e 2 non vengono modificati. L'operazione viene eseguita senza considerare o influenzare i bit di stato.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio: Indirizzi diretti

```
AWL              Spiegazione
LAR2   DBD 20    //Carica in AR 2 il puntatore della doppia parola di dati DBD20.
LAR2   DID 30    //Carica in AR 2 il puntatore della doppia parola di istanza DID30.
LAR2   LD 180    //Carica in AR 2 il puntatore della doppia parola di dati locali LD180.
LAR2   MD 24     //Carica in AR 2 il puntatore della doppia parola di merker MD24.
```

Esempio: Indirizzo puntatore

```
AWL              Spiegazione
LAR2    P#M100.0 //Carica in AR 2 una costante di puntatore a 32 bit.
```

9.9 T Trasferisci

Formato

T <operando>

Parametro	Tipo di dati	Area di memoria	Indirizzo di sorgente
<operando>	BYTE	E, A, PA, M, L, D	0...65535
	WORD		0...65534
	DWORD		0...65532

Descrizione dell'operazione

T <operando> trasferisce (copia) il contenuto dell'accumulatore 1 nell'indirizzo di destinazione se il Relè Master Control è attivato (MCR = 1). Se MCR = 0, nell'indirizzo di destinazione viene scritto il valore "0". Il numero di byte copiati dall'accumulatore 1 dipende dalla grandezza specificata nell'indirizzo di destinazione. L'accumulatore 1 memorizza i dati anche dopo l'operazione di trasferimento. Il trasferimento all'area periferica diretta (area di memoria PA) determina anche il trasferimento del contenuto dell'accumulatore 1 o di "0" (se MCR = 0) all'indirizzo corrispondente nell'immagine di processo delle uscite (area di memoria A). L'operazione viene eseguita senza considerare o influenzare i bit di stato.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio

```
AWL             Spiegazione
T      AB10     //Trasferisce il contenuto di ACCU 1-L-L al byte di uscita AB10.
T      MW14     //Trasferisce il contenuto di ACCU 1-L alla parola di merker MW14.
T      DBD2     //Trasferisce il contenuto di ACCU 1 alla doppia parola di dati DBD2.
```

9.10 T STW Trasferisci ACCU 1 nella parola di stato

Formato

T STW

Descrizione dell'operazione

T STW (Operazione T con operando STW) trasferisce i bit da 0 a 8 dall'accumulatore 1 alla parola di stato.

L'operazione viene eseguita senza considerare i bit di stato.

Nota: nelle CPU appartenenti alla famiglia S7-300, i bit della parola di stato /ER, STA e OR, non vengono sovrascritti dall'istruzione T STW. Soltanto i bit 1, 4, 5, 6, 7 e 8 vengono sovrascritti seguendo l'assegnazione dei bit dell'AKKU1.

Parola di stato

	BIE	**A1**	**A0**	**OV**	**OS**	**OR**	**STA**	**RLC**	**/ER**
Scrive:	x	x	x	x	x	x	x	x	x

Esempio

```
AWL              Spiegazione
T      STW       //Trasferisce i bit da 0 a 8 da ACCU 1 alla parola di stato.
```

I bit dell'accumulatore 1 contengono i seguenti bit di stato:

Bit	**31-9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Contenuto	*)	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER

*) senza trasferimento dei bit.

9.11 TAR cambia registro di indirizzo 1 con registro di indirizzo 2

Formato

TAR

Descrizione dell'operazione

TAR (Scambia registri di indirizzo) scambia i contenuti dei registri di indirizzo AR1 e AR2. L'operazione viene eseguita senza considerare o influenzare i bit di stato.

Il contenuto del registro di indirizzo AR 1 viene traslato nel registro di indirizzo AR 2, ed il contenuto del registro di indirizzo AR 2 viene traslato nel registro di indirizzo AR 1.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

9.12 TAR1 Trasferisci registro di indirizzo 1 in ACCU 1

Formato

TAR1

Descrizione dell'operazione

TAR1 trasferisce il contenuto di AR 1 nell'accumulatore 1 (puntatore a 32 bit). Il contenuto dell'accumulatore 1 viene prima memorizzato nell'accumulatore 2. L'operazione viene eseguita senza considerare o influenzare i bit di stato.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

9.13 TAR1 <D> Trasferisci registro di indirizzo 1 all'indirizzo destinazione (a 32 bit)

Formato

TAR1 <D>

Parametro	Tipo dati	Area memoria	Indirizzo di sorgente
<D>	DWORD	D, M, L	0...65532

Descrizione dell'operazione

TAR1 <D> trasferisce il contenuto del registro di indirizzo AR 1 alla doppia parola indirizzata <D>. Fungono da possibili aree di destinazione le doppie parole di merker (MD), le doppie parole di dati locali (LD), le doppie parole di dati (DBD), e le doppie parole di istanza (DID).

Gli accumulatori 1 e 2 non vengono modificati. L'operazione viene eseguita senza considerare o influenzare i bit di stato.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempi

```
AWL             Spiegazione
TAR1   DBD20    //Trasferisce il contenuto di AR 1 alla doppia parola di dati DBD20.
TAR1   DID30    //Trasferisce il contenuto di AR 1 alla doppia parola di istanza DID30.
TAR1   LD18     //Trasferisce il contenuto di AR 1 alla doppia parola di dati locali LD18.
TAR1   MD24     //Trasferisce il contenuto di AR 1 alla doppia parola di merker MD24.
```

9.14 TAR1 AR2 Trasferisci registro di indirizzo 1 nel registro di indirizzo 2

Formato

TAR1 AR2

Descrizione dell'operazione

TAR1 AR2 (Operazione TAR1 con operando AR2) trasferisce il contenuto del registro di indirizzo AR1 al registro di indirizzo AR2.

Gli accumulatori 1 e 2 non vengono modificati. L'operazione viene eseguita senza considerare o influenzare i bit di stato.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

9.15 TAR2 Trasferisci registro di indirizzo 2 in ACCU 1

Formato

TAR2

Descrizione dell'operazione

TAR2 trasferisce il contenuto del registro di indirizzo AR 2 all'accumulatore 1 (a 32 bit). Il contenuto dell'accumulatore 1 viene precedentemente memorizzato nell'accumulatore 2. L'operazione viene eseguita senza considerare o influenzare i bit di stato.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

9.16 TAR2 <D> Trasferisci registro di indirizzo 2 all'indirizzo di destinazione

Formato

TAR2 <D>

Parametro	Tipo dati	Area memoria	Indirizzo di sorgente
<D>	DWORD	D, M, L	0...65532

Descrizione dell'operazione

TAR2 <D> trasferisce il contenuto del registro di indirizzo AR 2 alla doppia parola indirizzata <D>. Fungono da possibili aree di destinazione le doppie parole di merker (MD), le doppie parole di dati locali (LD), le doppie parole di dati (DBD), e le doppie parole di istanza (DID.

Gli accumulatori 1 e 2 non vengono modificati. L'operazione viene eseguita senza considerare o influenzare i bit di stato.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempi

```
AWL             Spiegazione
TAR2   DBD20    //Trasferisce il contenuto di AR 2 alla doppia parola di dati DBD20.
TAR2   DID30    //Trasferisce il contenuto di AR 2 alla doppia parola di istanza DID30.
TAR2   LD18     //Trasferisce il contenuto di AR 2 alla doppia parola di dati locali LD18.
TAR2   MD24     //Trasferisce il contenuto di AR 2 alla doppia parola di merker MD24.
```

10 Operazioni di controllo del programma

10.1 Sommario delle operazioni di controllo del programma

Descrizione

Sono disponibili le seguenti operazioni di controllo del programma:

- BE Fine blocco
- BEB Fine blocco condizionato
- BEA Fine blocco assoluto
- CALL Richiamo di blocco
- CC Richiamo condizionato di un blocco
- UC Richiamo incondizionato di un blocco

- Richiamo di FB
- Richiamo di FC
- Richiamo di SFB
- Richiamo di SFC
- Richiamo di multiistanze
- Richiamo di blocchi da una biblioteca

- Relè Master Control (MCR)
- Avvertenze importanti sulle funzionalità MCR
- MCR(Salva RLC nello stack MCR, inizio zona MCR
-)MCR Fine zona MCR
- MCRA Attiva zona MCR
- MCRD Disattiva zona MCR

10.2 BE Fine blocco

Formato

BE

Descrizione dell'operazione

BE (Fine blocco) interrompe il flusso delle operazioni logiche nel blocco attuale, e salta al blocco che ha richiamato quello terminato. Il flusso delle operazioni logiche viene ripreso alla prima istruzione successiva al richiamo del blocco. L'area di dati locali attuale viene abilitata, e l'area di dati locali precedente diventa quella attuale. I blocchi di dati, che erano aperti al richiamo del blocco, vengono nuovamente aperti. Viene inoltre ripristinata la dipendenza MCR del blocco che ha richiamato quello terminato, e RLC viene trasmesso dal blocco attuale al blocco che ha eseguito il richiamo. L'operazione BE viene eseguita a prescindere da eventuali condizioni. Se l'istruzione di fine blocco BE viene saltata, il flusso attuale delle operazioni logiche non viene terminato, bensì ripreso all'interno del blocco alla destinazione di salto indicata.

Questa operazione BE non coincide con l'operazione BE nel software S5. Per quanto riguarda l'hardware S7, l'operazione BE offre la medesima funzionalità dell'operazione BEA.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	0	0	1	-	0

Esempio

```
AWL                 Spiegazione
      U     E 1.0
      SPB   NEXT   //Salta all'etichetta NEXT se RLC = 1 (E 1.0 = 1).
      L     EW4    //Riprende in questo punto se il salto non viene eseguito.
      T     EW10
      U     E 6.0
      U     E 6.1
      S     M12.0
      BE           //Fine blocco.
NEXT: NOP 0        //Riprende in questo punto se il salto viene eseguito.
```

10.3 BEB Fine blocco condizionato

Formato

BEB

Descrizione dell'operazione

Se RLC = 1, **BEB** (Fine blocco condizionato) interrompe il flusso delle operazioni logiche nel blocco attuale e salta al blocco che ha richiamato quello terminato. Il flusso delle operazioni logiche viene ripreso alla prima istruzione successiva al richiamo del blocco. L'area di dati locali attuale viene abilitata e e l'area di dati locali precedente diventa quella attuale. I blocchi di dati, che erano aperti al momento del richiamo del blocco, vengono nuovamente aperti. La dipendenza MCR del blocco che ha richiamato quello terminato viene ripristinata.

RLC (= 1) viene trasferito dal blocco terminato a quello che ha eseguito il richiamo. Se RLC = 0, l'operazione BEB non viene eseguita. In questo caso, RLC viene settato a "1" e il flusso delle operazioni logiche viene ripreso all'istruzione di programmazione successiva.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	x	0	1	1	0

Esempio

```
AWL             Spiegazione
U     E 1.0     //Aggiorna RLC.
BEB             //Termina il blocco se RLC = 1 ist.
L     EW4       //Riprende in questo punto se l'istruzione BEB non viene eseguita (RLC = 0).
T     MW10
```

10.4 BEA Fine blocco assoluto

Formato

BEA

Descrizione dell'operazione

BEA (Fine blocco assoluto) interrompe il flusso delle operazioni logiche nel blocco attuale e salta al blocco che ha richiamato quello terminato. Il flusso delle operazioni logiche viene ripreso alla prima istruzione successiva al richiamo del blocco. L'area di dati locali attuale viene abilitata, e l'area di dati locali precedente diventa quella attuale. I blocchi di dati, che erano aperti al richiamo del blocco, vengono nuovamente aperti. Viene inoltre ripristinata la dipendenza MCR del blocco che ha richiamato quello terminato, e RLC viene trasmesso dal blocco attuale al blocco che ha eseguito il richiamo. L'operazione BEA viene eseguita a prescindere da eventuali condizioni. Se l'istruzione di fine blocco assoluto BEA viene saltata, il flusso attuale delle operazioni logiche non viene terminato, bensì ripreso all'interno del blocco alla destinazione di salto indicata.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	0	0	1	-	0

Esempio

```
AWL                 Spiegazione
       U     E 1.0
       SPB   NEXT   //Salta all'etichetta di salto NEXT se RLC = 1 (E 1.0 = 1).
       L     EW4    //Riprende in questo punto se il salto non viene eseguito
       T     EW10
       U     E 6.0
       U     E 6.1
       S     M12.0
       BEA          //Fine blocco assoluto.
NEXT:  NOP 0        //Riprende in questo punto se il salto viene eseguito.
```

10.5 CALL Richiamo di blocco

Formato

CALL < identificazione del blocco di codice >

Descrizione dell'operazione

CALL <Identificazione del blocco di codice> funge da richiamo per le funzioni (FC) ed i blocchi funzionali (FB) o anche per le funzioni standard (SFC) ed i blocchi di funzioni standard (SFB) fornite dalla Siemens. L'operazione CALL richiama FC e SFC, ovvero FB e SFB, immessi dall'utente come operandi, indipendentemente dal RLC o da altre condizioni. Se si richiama un FB o SFB con l'operazione CALL, occorre dotarlo di un blocco dati di istanza. A conclusione dell'elaborazione del blocco richiamato, viene elaborato il programma del blocco che ha eseguito il richiamo. L'identificazione del blocco di codice può essere indicata in modo assoluto o simbolico. I contenuti dei registri vengono riparati dopo un richiamo di SFB/SFC.

Esempio: CALL FB1, DB1, ovvero CALL FILLVAT1, RECIPE1

Blocco di codice	Tipo di blocco	Sintassi per il richiamo (indirizzo assoluto)
FC	Funzione	CALL FCn
SFC	Funzione di sistema	CALL SFCn
FB	Blocco funzionale	CALL FBn1,DBn2
SFB	Blocco funzionale di sistema	CALL SFBn1,DBn2

Nota

Se l'elaborazione viene eseguita con l'editor AWL, i dati (n, n1 oppure n2) della suddetta tabella devono essere preliminarmente riferiti a blocchi esistenti e validi. Anche i nomi simbolici devono essere definiti in anticipo.

Trasmissione di parametri (utilizza a tal fine il modo di elaborazione incrementale)

Il blocco che esegue il richiamo può scambiare i propri parametri con quelli del blocco richiamato tramite la lista di variabili. Tale lista viene aggiornata automaticamente nel programma AWL dopo ogni inserimento di una istruzione CALL valida.

Se viene richiamato un blocco funzionale oppure un blocco funzionale di sistema, o una funzione oppure una funzione di sistema, e la tabella di dichiarazione di variabili del blocco richiamato riporta dichiarazioni del tipo IN, OUT, e IN_OUT, le variabili in oggetto vengono incluse nel programma del blocco che esegue il richiamo come elenco di parametri formali.

Quando si richiamano le FC e SFC, bisogna assegnare ai parametri formali i parametri attuali del blocco di codice richiamante.

Quando si richiamano gli FB e SFB, si devono indicare soltanto i parametri attuali che si devono cambiare, rispetto all'ultimo richiamo. Dopo l'elaborazione di FB, i parametri attuali sono, infatti, memorizzati nel DB di istanza. Se il parametro attuale è un DB, è sempre necessario indicare l'indirizzo assoluto completo, ad es. DB1,DBW2.

I parametri IN possono essere specificati come costanti oppure come indirizzi assoluti o simbolici. I parametri OUT e IN_OUT devono essere indicati come indirizzi assoluti o simbolici. È necessario accertarsi che tutti gli indirizzi e le costanti siano compatibili con i tipi dei dati da trasferire.

L'operazione CALL memorizza l'indirizzo di ritorno (selettore e indirizzo relativo), i selettori di entrambi i blocchi di dati che sono stati aperti ,ed il bit MA nello stack B. L'operazione disattiva inoltre la dipendenza MCR e crea l'area dati locali per il blocco da richiamare.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	0	0	1	-	0

Esempio 1: Attribuzione dei parametri al richiamo della funzione FC6

```
CALL    FC6
        Parametro formale       Parametro attuale
        NO_OF_TOOL              := MW100
        TIME_OUT                := MW110
        FOUND                   := A 0.1
        ERROR                   := A 100.0
```

Esempio 2: Richiamo di una SFC senza parametri

```
AWL             Spiegazione
CALL    SFC43   //Richiama SFC43 per riavviare il monitoraggio del temporizzatore (senza
                parametri)
```

Esempio 3: Richiamo di FB99 con blocco dati di istanza DB1

```
CALL    FC99,DB1
        Parametro formale       Parametro attuale
        MAX_RPM                 := #RPM1_MAX
        MIN_RPM                 := #RPM1
        MAX_POWER               := #POWER1
        MAX_TEMP                := #TEMP1
```

Esempio 4: Richiamo di FB99 con blocco dati di istanza DB2

```
CALL    FC99,DB2
        Parametro formale       Parametro attuale
        MAX_RPM                 := #RPM2_MAX
        MIN_RPM                 := #RPM2
        MAX_POWER               := #POWER1
        MAX_TEMP                := #TEMP2
```

Nota

Per ogni singolo richiamo di blocchi funzionali o blocchi funzionali di sistema, deve esistere un blocco di dati di istanza. Nell'esempio illustrato, i blocchi DB1 e DB2 devono essere definiti preliminarmente.

10.6 Richiamo di FB

Formato

CALL FB n1, DB n1

Descrizione

L'operazione viene utilizzata per il richiamo di blocchi funzionali creati dall'utente (FB). L'operazione CALL richiama l'FB che è stato specificato come operando indipendentemente dall'RLC o da un'altra qualsiasi condizione. Se un FB viene richiamato con CALL, è necessario attribuirgli un blocco dati di istanza. Al termine dell'elaborazione del blocco richiamato, si procede con l'elaborazione del programma del blocco richiamante. L'identificazione del blocco di codice può essere specificata in modo assoluto o simbolico.

Trasferimento di parametri (utilizzare il modo di elaborazione incrementale)

Il blocco richiamante può scambiare parametri con il blocco richiamato tramite la lista delle variabili. Nel programma AWL, la lista delle variabili viene aggiornata automaticamente se viene specificata un'istruzione CALL valida.

Se viene richiamato un FB e la tabella di dichiarazione di variabili del blocco richiamato dispone di dichiarazioni di tipo IN, OUT e IN_OUT, queste variabili vengono aggiunte al programma del blocco richiamante come lista di parametri formali.

Per il richiamo degli FB basta specificare solo i parametri attuali che devono essere modificati rispetto all'ultimo richiamo in quanto dopo l'elaborazione dell'FB i parametri attuali sono memorizzati nel DB di istanza. Se il parametro attuale è un DB, deve essere sempre indicato l'indirizzo assoluto completo, ad es. DB1, DBW2.

I parametri IN possono essere indicati come costanti o come indirizzi assoluti o simbolici. I parametri OUT e IN_OUT devono invece essere specificati come indirizzi assoluti o simbolici. Verificare che tutti gli indirizzi e le costanti siano compatibili con i tipi dei dati che vengono trasmessi.

L'operazione CALL memorizza l'indirizzo di ritorno (selettore e indirizzo relativo), i selettori di entrambi i blocchi dati aperti e il bit MA nello stack B. Inoltre, l'operazione disattiva l'interazione MCR e genera l'area di dati locali del blocco che deve essere richiamato.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	0	0	1	-	0

Esempio 1: richiamo dell'FB99 con il blocco dati di istanza DB1

```
CALL    FB99,DB1
        Parametro formale       Parametro attuale
        MAX_RPM                 := #RPM1_MAX
        MIN_RPM                 := #RPM1
        MAX_POWER               := #POWER1
        MAX_TEMP                := #TEMP1
```

Esempio 2: richiamo dell'FB99 con il blocco dati di istanza DB2

```
CALL    FB99,DB2
        Parametro formale       Parametro attuale
        MAX_RPM                 := #RPM2_MAX
        MIN_RPM                 := #RPM2
        MAX_POWER               := #POWER2
        MAX_TEMP                := #TEMP2
```

Avvertenza

Ogni richiamo di un FB deve disporre di un blocco dati di istanza. Nell'esempio qui riportato, i blocchi DB1 e DB2 devono essere già presenti prima del richiamo.

10.7 Richiamo di FC

Formato

CALL FC n

Avvertenza

Se viene utilizzato l'editor AWL, le specificazioni effettuate devono riferirsi a blocchi validi già presenti. Anche i nomi simbolici devono essere stati definiti precedentemente.

Descrizione

L'operazione viene utilizzata per il richiamo di funzioni (FC) e di blocchi funzionali (FB) o per il richiamo di funzioni standard (SFC) e di blocchi funzionali standard (SFB), entrambi forniti dalla Siemens. L'operazione CALL richiama l'FC e l'SFC oppure l'FB e l'SFB che sono stati specificati come operandi indipendentemente dall'RLC o da un'altra qualsiasi condizione. Se un FB o un SFB viene richiamato con CALL, è necessario attribuirgli un blocco dati di istanza. Al termine dell'elaborazione del blocco richiamato, si procede con l'elaborazione del programma del blocco richiamante. L'identificazione del blocco di codice può essere specificata in modo assoluto o simbolico.

Trasferimento di parametri (utilizzare il modo di elaborazione incrementale)

Il blocco richiamante può scambiare parametri con il blocco richiamato tramite la lista delle variabili. Nel programma AWL, la lista delle variabili viene aggiornata automaticamente se viene specificata un'istruzione CALL valida.

Se viene richiamata una FC e la tabella di dichiarazione di variabili del blocco richiamato dispone di dichiarazioni di tipo IN, OUT e IN_OUT, queste variabili vengono aggiunte al programma del blocco richiamante come lista di parametri formali.

Con i richiami di FC, ai parametri formali devono essere attribuiti parametri attuali del blocco codice richiamante.

I parametri IN possono essere indicati come costanti o come indirizzi assoluti o simbolici. I parametri OUT e IN_OUT devono invece essere specificati come indirizzi assoluti o simbolici. Verificare che tutti gli indirizzi e le costanti siano compatibili con i tipi dei dati che vengono trasmessi.

L'operazione CALL memorizza l'indirizzo di ritorno (selettore e indirizzo relativo), i selettori di entrambi i blocchi dati aperti e il bit MA nello stack B. Inoltre, l'operazione disattiva l'interazione MCR e genera l'area di dati locali del blocco che deve essere richiamato.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	0	0	1	-	0

Esempio: attribuzione di parametri al richiamo della funzione FC6

```
CALL    FC6
        Parametro formale       Parametro attuale
        NO OF TOOL              := MW100
        TIME OUT                := MW110
        FOUND                   := A 0.1
        ERROR                   := A 100.0
```

10.8 Richiamo di SFB

Formato

CALL SFB n1, DB n2

Descrizione

L'operazione viene utilizzata per il richiamo di blocchi funzionali standard (SFB) che vengono entrambi forniti dalla Siemens. L'operazione CALL richiama gli SFB che sono stati specificati come operandi indipendentemente dall'RLC o da un'altra qualsiasi condizione. Se un SFB viene richiamato con CALL, è necessario attribuirgli un blocco dati di istanza. Al termine dell'elaborazione del blocco richiamato, si procede con l'elaborazione del programma del blocco richiamante. L'identificazione del blocco di codice può essere specificata in modo assoluto o simbolico.

Trasferimento di parametri (utilizzare il modo di elaborazione incrementale)

Il blocco richiamante può scambiare parametri con il blocco richiamato tramite la lista delle variabili. Nel programma AWL, la lista delle variabili viene aggiornata automaticamente se viene specificata un'istruzione CALL valida.

Se viene richiamato un SFB e la tabella di dichiarazione di variabili del blocco richiamato dispone di dichiarazioni di tipo IN, OUT e IN_OUT, queste variabili vengono aggiunte al programma del blocco richiamante come lista di parametri formali.

Per il richiamo degli SFB basta specificare solo i parametri attuali che devono essere modificati rispetto all'ultimo richiamo in quanto dopo l'elaborazione dell'SFB i parametri attuali sono memorizzati nel DB di istanza. Se il parametro attuale è un DB, deve essere sempre indicato l'indirizzo assoluto completo, ad es. DB1, DBW2.

I parametri IN possono essere indicati come costanti o come indirizzi assoluti o simbolici. I parametri OUT e IN_OUT devono invece essere specificati come indirizzi assoluti o simbolici. Verificare che tutti gli indirizzi e le costanti siano compatibili con i tipi dei dati che vengono trasmessi.

L'operazione CALL memorizza l'indirizzo di ritorno (selettore e indirizzo relativo), i selettori di entrambi i blocchi dati aperti e il bit MA nello stack B. Inoltre, l'operazione disattiva l'interazione MCR e genera l'area di dati locali del blocco che deve essere richiamato.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	0	0	1	-	0

Esempio

```
CALL    SFB4,DB4
        Parametro formale       Parametro attuale
        IN:                     E0.1
        PT:                     T#20s
        Q:                      M0.0
        ET:                     MW10
```

Avvertenza

Ogni richiamo di un SFB deve disporre di un blocco dati di istanza. Nell'esempio qui riportato, i blocchi SFB4 e DB4 devono essere già presenti prima del richiamo.

10.9 Richiamo di SFC

Formato

CALL SFC n

Avvertenza

Se viene utilizzato l'editor AWL, le specificazioni effettuate devono riferirsi a blocchi validi già presenti. Anche i nomi simbolici devono essere stati definiti precedentemente.

Descrizione

L'operazione viene utilizzata per il richiamo di funzioni standard (SFC), entrambi forniti dalla Siemens. L'operazione CALL richiama l'FC e l'SFC oppure l'FB e l'SFB che sono stati specificati come operandi indipendentemente dall'RLC o da un'altra qualsiasi condizione. Se un FB o un SFB viene richiamato con CALL, è necessario attribuirgli un blocco dati di istanza. Al termine dell'elaborazione del blocco richiamato, si procede con l'elaborazione del programma del blocco richiamante. L'identificazione del blocco di codice può essere specificata in modo assoluto o simbolico.

Trasferimento di parametri (utilizzare il modo di elaborazione incrementale)

Il blocco richiamante può scambiare parametri con il blocco richiamato tramite la lista delle variabili. Nel programma AWL, la lista delle variabili viene aggiornata automaticamente se viene specificata un'istruzione CALL valida.

Se viene richiamata una SFC e la tabella di dichiarazione di variabili del blocco richiamato dispone di dichiarazioni di tipo IN, OUT e IN_OUT, queste variabili vengono aggiunte al programma del blocco richiamante come lista di parametri formali.

Con i richiami di SFC, ai parametri formali devono essere attribuiti parametri attuali del blocco codice richiamante.

I parametri IN possono essere indicati come costanti o come indirizzi assoluti o simbolici. I parametri OUT e IN_OUT invece devono essere specificati come indirizzi assoluti o simbolici. Verificare che tutti gli indirizzi e le costanti siano compatibili con i tipi dei dati che vengono trasmessi.

L'operazione CALL memorizza l'indirizzo di ritorno (selettore e indirizzo relativo), i selettori di entrambi i blocchi dati aperti e il bit MA nello stack B. Inoltre, l'operazione disattiva l'interazione MCR e genera l'area di dati locali del blocco che deve essere richiamato.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	0	0	1	-	0

Esempio: richiamo di una SFC senza parametri

```
AWL              Spiegazione
CALL   SFC43     //Richiama SFC43, per avviare di nuovo il controllo di temporizzazione (senza
                 parametri).
```

10.10 Richiamo di multiistanze

Formato

CALL # Nome della variabile

Descrizione

Una multiistanza si crea mediante la dichiarazione di una variabile statica del tipo di dati di un blocco funzionale. Solo le multiistanze già dichiarate vengono riportate nel catalogo degli elementi del programma.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	0	0	X	X	X

10.11 Richiamo di blocchi da una biblioteca

Le biblioteche usate nel SIMATIC Manager vengono offerte all'utente per la selezione.

Da queste biblioteche è possibile selezionare blocchi

- integrati nel sistema operativo della CPU dell'utente (biblioteca "Standard Library"),
- depositati dall'utente stesso in biblioteche in quanto destinati ad un uso multiplo.

10.12 CC Richiamo condizionato di un blocco

Avvertenze importanti sulla funzionalità MCR

Formato

CC < identificazione del blocco di codice >

Descrizione dell'operazione

CC <identificazione del blocco di codice> (Richiamo condizionato di un blocco) richiama un blocco di codice se RLC = 1. L'operazione CC richiama i blocchi di codice, del tipo FC o SFC, senza i relativi parametri. Tale operazione è analoga all'operazione CALL, con la differenza che il programma richiamato non implica il trasferimento dei parametri. L'operazione memorizza l'indirizzo di rientro (selettore e relativo indirizzo), i selettori di entrambi i blocchi di dati attuali nonché il bit MA nello stack B, disattiva la dipendenza MCR, crea l'area dei dati locali del blocco da richiamare, e avvia l'esecuzione del blocco di codice richiamato. L'identificazione del blocco di codice può essere indicata in modo assoluto o simbolico.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	0	0	1	1	0

Esempio 1

```
AWL             Spiegazione
U     E 2.0     //Interroga lo stato di segnale all'ingresso E 2.0.
CC    FC6       //Richiama la funzione FC6 se E 2.0 = 1.
U     M3.0      //Istruzione eseguita al ritorno dalla funzione richiamata (se E 2.0 = 1) o
                subito dopo l'istruzione U E 2.0, se E 2.0 = 0.
```

Nota

Al richiamo di un blocco funzionale (FB) o di un blocco funzionale di sistema (SFB) tramite l'istruzione **CALL** è necessario indicare un blocco dati di istanza (Nr.DB) nell'istruzione. Con l'istruzione **CC** non è consentito utilizzare una variabile del tipo "BlockFB" o "BlockFC". Poiché non è possibile assegnare un blocco dati al richiamo con l'istruzione **CC** nell'operando dell'istruzione, è possibile utilizzare questa istruzione solo per blocchi dati senza parametri di blocco e dati locali statici.

A seconda della rete utilizzata, durante la compilazione del linguaggio di programmazione 'schema a contatti' nel linguaggio di programmazione 'lista di istruzioni', "KOP/AWL: Programmare blocco" genera in parte l'operazione **UC** e in parte l'operazione **CC**. Per evitare che si verifichino errori nei programmi creati dall'utente, si consiglia di utilizzare in generale l'operazione **CALL**.

10.13 UC Richiamo incondizionato di un blocco

Formato

UC < identificazione del blocco di codice >

Descrizione dell'operazione

UC <identificazione del blocco di codice> (Richiamo incondizionato di un blocco) richiama un blocco di codice del tipo FC o SFC. L'operazione UC è analoga all'operazione CALL, con la differenza che il programma richiamato non implica il trasferimento dei parametri. L'operazione memorizza l'indirizzo di ritorno (selettore e indirizzo relativo), i selettori di entrambi i blocchi di dati attuali nonché il bit MA nello stack B, disattiva la dipendenza MCR, crea l'area dei dati locali del blocco da richiamare, e avvia l'esecuzione del blocco di codice richiamato.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	0	0	1	-	0

Esempio 1

```
AWL             Spiegazione
UC      FC6     //Richiama la funzione FC6 (senza parametri).
```

Esempio 2

```
AWL             Spiegazione
UC      SFC43   //Richiama la funzione di sistema SFC43 (senza parametri).
```

Nota

Al richiamo di un FB o di un SFB tramite l'istruzione **CALL** è necessario indicare un blocco dati di istanza (Nr.DB) nell'istruzione. Con l'istruzione **UC** non è consentito utilizzare una variabile del tipo "BlockFB" o "BlockFC". Poiché non è possibile assegnare un blocco dati al richiamo con l'istruzione **UC** nell'operando dell'istruzione, è possibile utilizzare questa istruzione solo per blocchi dati senza parametri di blocco e dati locali statici.

A seconda della rete utilizzata, durante la compilazione del linguaggio di programmazione 'schema a contatti' nel linguaggio di programmazione 'lista di istruzioni', "KOP/AWL: Programmare blocco" genera in parte l'operazione **UC** e in parte l'operazione **CC**. Per evitare che si verifichino errori nei programmi creati dall'utente, si consiglia di utilizzare in generale l'operazione **CALL**.

10.14 Relè Master Control (MCR)

Avvertenze importanti sulla funzionalità MCR

Descrizione

Il Relè Master Control (MCR) viene utilizzato come commutatore principale di uno schema a contatti di rete per eccitare e diseccitare il flusso di corrente. Dipendono dal Relè Master Control le operazioni avviate dalle seguenti operazioni logiche combinatorie di bit e di trasferimento:

- **=** <bit>
- **S** <bit>
- **R** <bit>
- **T** <byte>, **T** <parola>, **T** <doppia parola>

L'operazione **T**, per la quale è possibile usare un byte, una parola o una doppia parola, scrive uno "0" nella memoria se MCR è "0". Le operazioni **S** e **R** non alterano il valore già esistente. L'operazione **=** scrive uno "0" nel bit indirizzato.

Operazioni dipendenti dallo stato del bit MCR

Stato di segnale di MCR	**= <bit>**	**S <bit>, R <bit>**	**T <byte>, T <parola> T <doppia parola>**
0 ("OFF ")	Scrive "0". (limita un relè che entra in stato di quiete quando viene tolta la tensione.)	Non scrive. (limita un relè che conserva lo stato attuale quando viene tolta la tensione.)	Scrive "0". (Imita una componente che in caso di caduta di tensione fornisce il valore "0".)
1 ("ON ")	Elaborazione normale.	Elaborazione normale.	Elaborazione normale.

MCR(- inizio della zona MCR /)MCR - fine della zona MCR

Il Relè Master Control viene controllato da uno stack, largo un bit e profondo otto bit. Il MCR è attivo finché tutti gli otto inserimenti sono uguali a "1". L'operazione MCR(copia il bit RLC nello stack MCR. L'operazione)MCR cancella l'ultima registrazione dallo stack, e setta la posizione priva di registrazione a "1". Le operazioni MCR(e)MCR devono essere sempre utilizzate in coppia. Un eventuale errore, vale a dire l'esistenza di più di otto operazioni MCR(sequenziali o il tentativo di esecuzione dell'operazione)MCR a stack vuoto, determina l'output del messaggio d'errore MCRF.

MCRA - Attiva zona MCR/ MCRD - Disattiva zona MCR

Le operazioni MCRA e MCRD devono sempre essere utilizzate in coppia. Le istruzioni programmate tra MCRA e MCRD dipendono dallo stato del bit MCR. Le istruzioni che non rientrano nella sequenza delimitata da MCRA e MCRD non dipendono dallo stato del bit MCR.

La dipendenza MCR delle funzioni (FC) e dei blocchi funzionali (FB) deve essere programmata nei relativi blocchi. Avvalersi a tal fine dell'operazione MCRA nel blocco richiamato.

Precauzione

Per escludere il potenziale danneggiamento di persone o cose, il Relè Master Control non deve mai essere utilizzato al posto di un Relè Master Control meccanico cablato, utilizzato per funzioni di stop d'emergenza.

10.15 Avvertenze importanti sulle funzionalità MCR

Attenzione ai blocchi nei quali il Relè Master Control è stato attivato con MCRA

- Se il Relè Master Control è disattivato, nelle sezioni di programma tra **MCR(** e **)MCR**, attraverso tutte le assegnazioni (T, =) viene scritto il valore 0!
- Il Relè Master Control è disattivato esattamente quando davanti a un comando **MCR(** il RLC era = 0.

Pericolo: STOP del sistema di automazione o comportamento di esecuzione indefinito!

Per il calcolo degli indirizzi il compilatore accede anche in scrittura ai dati locali dietro le variabili temporanee definite in VAR_TEMP. Per questo motivo le sequenze di comandi seguenti portano il PLC su STOP o provocano un comportamento di esecuzione indefinito.

Accessi a parametri formali

- Accessi a componenti di parametri FC complessi del tipo STRUCT, UDT, ARRAY, STRING
- Accessi a componenti di parametri FB complessi del tipo STRUCT, UDT, ARRAY, STRING dell'area IN_OUT in un blocco con multiistanza (versione blocchi 2)
- Accessi a parametri di un FB con multiistanza (versione blocchi 2) quando il loro indirizzo è maggiore di 8180.0
- L'accesso nell'FB con multiistanza (versione blocchi 2) a un parametro del tipo BLOCK_DB apre il DB 0. I successivi accessi ai dati portano la CPU su STOP. Con TIMER, COUNTER, BLOCK_FC, BLOCK_FB si utilizzano sempre anche T 0, Z 0, FC 0 e FB 0.

Trasferimento di parametri

- Richiami di blocco con i quali vengono trasmessi parametri.

KOP/FUP

- Diramazioni a T e connettori in KOP o FUP iniziano con RLC = 0.

Rimedio

Sciogliere i comandi indicati dalla dipendenza MCR:

1° disattivare il Relè Master Control con il comando MCRD prima dell'istruzione o della rete in questione

2° attivare il Relè Master Control con il comando MCRA dopo l'istruzione o la rete in questione.

10.16 MCR(Salva RLC nello stack MCR, inizio zona MCR

Avvertenze importanti sulla funzionalità MCR

Formato

MCR(

Descrizione dell'operazione

MCR((Inizia zona MCR) memorizza RLC nello stack MCR e apre una zona MCR.
Zona MCR = istruzioni comprese tra l'operazione **MCR(** e la corrispondente operazione **)MCR**.
L'utilizzo delle operazioni **MCR(** e **)MCR** è consentito solo in coppia.

Se RLC = 1, MCR è "attivato". Le istruzioni dipendenti da MCR comprese nella zona MCR vengono eseguite normalmente.

Se RLC = 0, MCR è "disattivato". Le istruzioni dipendenti da MCR comprese nella zona MCR vengono eseguite secondo quanto riportato nella seguente tabella.

Operazioni dipendenti dallo stato del bit MCR

Stato di segnale di MCR	**= <bit>**	**S <bit>, R <bit>**	**T <byte>, T <parola> T <doppia parola>**
0 ("OFF ")	Scrive "0". (limita un relè che entra in stato di quiete quando viene tolta la tensione.)	Non scrive. (limita un relè che conserva lo stato attuale quando viene tolta la tensione.)	Scrive "0". (Imita una componente che in caso di caduta di tensione fornisce il valore "0".)
1 ("ON ")	Elaborazione normale.	Elaborazione normale.	Elaborazione normale.

Le operazioni MCR(e)MCR possono essere annidate. La profondità massima di annidamento è di otto operazioni. Lo stack può quindi riportare un massimo di otto inserimenti. Se l'operazione MCR(viene eseguita a stack completo si verifica l'errore stack MCR (MCRF).

Parola di stato

	BIE	**A1**	**A0**	**OV**	**OS**	**OR**	**STA**	**RLC**	**/ER**
Scrive:	-	-	-	-	-	0	1	-	0

Esempio

```
AWL              Spiegazione
MCRA             //Attiva zona MCR.
U      E 1.0
MCR(             //Memorizza RLC nello stack MCR, apre una zona MCR. MCR è "ON" se RLC = 1 (E
                 1.0 = 1). MCR è "OFF" se RLC = 0 (E 1.0 = 0).
U      E 4.0
=      A 8.0     //Se MCR = "OFF", A 8.0 viene settata a "0", senza considerare E 4.0.
L      MW20
T      AW10      //Se MCR = "OFF", il valore "0" viene trasferito a AW10.
)MCR             //Fine zona MCR.
MCRD             //Disattiva la zona MCR.
U      E 1.1
=      A 8.1     //Queste istruzioni non rientrano nella zona MCR, e non dipendono dal bit MCR.
```

10.17)MCR Fine zona MCR

Avvertenze importanti sulla funzionalità MCR

Formato

)MCR

Descrizione dell'operazione

)MCR (Fine zona MCR) cancella una registrazione dallo stack MCR, e termina la zona MCR. L'ultima registrazione dello stack MCR viene abilitata e settata a "1". L'utilizzo delle operazioni MCR(e)MCR è consentito solo in coppia. Se l'operazione)MCR viene eseguita a stack vuoto, si verifica l'errore stack MCR (MCRF).

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	0	1	-	0

Esempio

```
AWL             Spiegazione
MCRA            //Attiva la zona MCR.
U     E 1.0
MCR(            //Memorizza RLC nello stack MCR, apre una zona MCR. MCR è "ON" se RLC = 1
                //(E 1.0 = 1). MCR è "OFF", se RLC = 0 (E 1.0 = 0).
U     E 4.0
=     A 8.0     //Se MCR = "OFF", A 8.0 viene settata a "0", senza considerare E 4.0.
L     MW20
T     AW10      //Se MCR = "OFF", "0" viene trasferito a AW10.
)MCR            //Termina la zona MCR.
MCRD            //Disattiva la zona MCR.
U     E 1.1
=     A 8.1     //Queste istruzioni non rientrano nella zona MCR, e non dipendono dal bit MCR.
```

10.18 MCRA Attiva zona MCR

Avvertenze importanti sulla funzionalità MCR

Formato

MCRA

Descrizione dell'operazione

MCRA (Attiva il Relè Master Control) attiva la dipendenza MCR per le istruzioni che seguono l'operazione. Le operazioni MCRA e MCRD (disattiva il Relé Master Control) devono sempre essere utilizzate in coppia. Le istruzioni programmate tra MCRA e MCRD dipendono dallo stato di segnale del bit MCR.

L'operazione viene eseguita senza considerare o influenzare i bit di stato.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio

```
AWL              Spiegazione
MCRA             //Attiva la zona MCR.
U      E 1.0
MCR(             //Memorizza RLC nello stack MCR, apre una zona MCR. MCR è "ON" se RLC = 1
                 //(E 1.0 = 1). MCR è "OFF" se RLC = 0 (E 1.0 = 0).
U      E 4.0
=      A 8.0     //Se MCR = "OFF", A 8.0 viene settata a "0", senza considerare E 4.0.
L      MW20
T      AW10      //Se MCR = "OFF", "0" viene trasferito a AW10.
)MCR             //Fine zona MCR.
MCRD             //Disattiva la zona MCR.
U      E 1.1
=      A 8.1     //Queste istruzioni non rientrano nella zona MCR, e non dipendono dal bit MCR.
```

10.19 MCRD Disattiva zona MCR

Avvertenze importanti sulla funzionalità MCR

Formato

MCRD

Descrizione dell'operazione

MCRD (Disattiva il Relè Master Control) disattiva la dipendenza MCR per le istruzioni che seguono dopo questa operazione. Le operazioni MCRD e MCRA (attiva il Relè Master Control) devono essere utilizzate sempre in coppia. Le istruzioni programmate tra MCRA e MCRD dipendono dallo stato di segnale del bit MCR.

L'operazione viene eseguita senza considerare o influenzare i bit di stato.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio

```
AWL             Spiegazione
MCRA            //Attiva la zona MCR.
U     E 1.0
MCR(            //Memorizza RLC nello stack MCR, apre una zona MCR. MCR è "ON" se RLC = 1
                //(E 1.0 = 1). MCR è "OFF" se RLC = 0 (E 1.0 = 0).
U     E 4.0
=     A 8.0     //Se MCR = "OFF", A 8.0 viene settata a "0", senza considerare E 4.0.
L     MW20
T     AW10      //Se MCR = "OFF", "0" viene trasferito a AW10.
)MCR            //Fine zona MCR.
MCRD            //Disattiva la zona MCR.
U     E 1.1
=     A 8.1     //Queste istruzioni non rientrano nella zona MCR, e non dipendono dal bit MCR.
```

11 Operazioni di scorrimento e rotazione

11.1 Operazioni di scorrimento

11.1.1 Sommario delle operazioni di scorrimento

Descrizione

L'utente ha la facoltà di adoperare le operazioni di scorrimento per traslare il contenuto della parola bassa di ACCU 1 o il contenuto dell'intero accumulatore, bit per bit, verso sinistra o verso destra (vedere Registri CPU). Decidendo di far scorrere verso sinistra, si moltiplica il contenuto dell'accumulatore alla potenza di 2^n; decidendo, al contrario, di far scorrere verso destra, si divide il contenuto dell'accumulatore alla potenza di 2^n. Per esempio, se l'utente fa scorrere a sinistra l'equivalente binario del valore decimale 3 nella misura di 3 bit, ottiene l'equivalente binario del valore decimale 24 nell'accumulatore. Se fa scorrere, invece, a destra l'equivalente binario del valore decimale 16 nella misura di 2 bit otterrà l'equivalente binario del valore decimale 4 nell'accumulatore.

Il numero che segue l'operazione di scorrimento od un valore nel byte basso della parola bassa di ACCU 2, sta ad indicare il numero di bit nella misura del quale si opera lo scorrimento. I posti che vengono lasciati vuoti dall'operazione di scorrimento vengono riempiti con degli zeri oppure con lo stato di segnale del bit del segno (0 sta per positivo, 1 sta per negativo). Il bit che è stato traslato per ultimo viene caricato nel bit A1 della parola di stato. I bit A0 e OV della parola di stato vengono resettati a 0. L'utente può usare le operazioni di salto per valutare il bit A1.

Le operazioni di scorrimento sono incondizionate, vale a dire che la loro esecuzione non dipende da alcuna condizione speciale. Esse non hanno influenza sul risultato logico combinatorio.

Sono disponibili le seguenti operazioni di scorrimento:

- SSI Fai scorrere numero intero con segno (a 16 bit)
- SSD Fai scorrere numero intero con segno (a 32 bit)
- SLW Fai scorere parola a sinistra (a 16 bit)
- SRW Fai scorrere parola a destra (a 16 bit)
- SLD Fai scorrere doppia parola a sinistra (a 32 bit)
- SRD Fai scorrere doppia parola a destra (a 32 bit)

11.1.2 SSI Fai scorrere numero intero con segno (a 16 bit)

Formati

SSI
SSI <numero>

Operando	Tipo dati	Descrizione
<numero>	Numero intero, senza segno	Numero di bit nella misura del quale si opera lo scorrimento; area da 0 a 15

Descrizione dell'operazione

SSI (Fai scorrere il numero intero bit con segno verso destra) fa scorrere l'intero contenuto dell'accumulatore 1-L verso destra, bit per bit. Nelle posizioni di bit che risultano vuote in seguito all'operazione di scorrimento, viene scritto lo stato di segnale del bit di segno (bit 15). Il bit traslato per ultimo viene caricato nel bit di stato A1. Il numero di bit nella misura del quale operare lo scorrimento viene indicato dall'operando <numero> o da un valore riportato nell'accumulatore 2-L-L.

SSI <numero>: il numero di scorrimento viene indicato dall'operando **<numero>**. Sono consentiti valori compresi tra 0 e 15. I bit di stato A0 e OV vengono resettati a "0", se **<numero>** è maggiore di zero. Se il numero di scorrimento è "0" l'operazione di scorrimento viene elaborata come un'operazione **NOP**.

SSI: il numero di scorrimento viene indicato dal valore riportato nell'accumulatore 2-L-L. Sono consentiti valori compresi tra 0 e 255. I numeri di scorrimento > 16 producono sempre il medesimo risultato: ACCU 1 = 16#0000, A1 = 0 oppure ACCU 1 = 16#FFFF, A1 = 1. Se il numero di scorrimento > 0, i bit di stato A0 e OV vengono resettati a "0". Se il numero di scorrimento è "0", l'operazione di scorrimento viene elaborata come un'operazione **NOP**.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	x	x	x	-	-	-	-	-

Esempi

Contenuto	ACCU1-H				ACCU1-L			
Bit	31 . . .	. .	. .	. . . 16	15 . . .	. .	. .	. . . 0
prima dell'esecuzione di **SSI 6**	0101	1111	0110	0100	1001	1101	0011	1011
dopo l'esecuzione di **SSI 6**	0101	1111	0110	0100	1111	1110	0111	0100

Esempio 1

```
AWL             Spiegazione
L     MW4       //Carica il valore in ACCU 1.
SSI   6         //Fa scorrere i bit in ACCU 1 di 6 posizioni verso destra.
T     MW8       //Trasferisce il risultato a MW8.
```

Esempio 2

```
AWL             Spiegazione
L     +3        //Carica il valore +3 in ACCU 1.
L     MW20      //Carica il contenuto di ACCU 1 in ACCU 2. Carica il valore di MW20 in ACCU 1.
SSI             //Il numero di scorrimento è dato dal valore di ACCU 2-L-L. => Fa scorrere i bit
                //in ACCU 1-L, considerando il segno, di 3 posizioni verso destra, e setta le
                //posizioni vuote sullo stato di segnale del bit di segno.
SPP   NEXT      //Passa all'etichetta di salto NEXT se il bit traslato per ultimo (A1) è = 1.
```

11.1.3 SSD Fai scorrere numero intero con segno (a 32 bit)

Formati

SSD
SSD <numero>

Operando	Tipo dati	Descrizione
<numero>	Numero intero, senza segno	Numero di bit, nella misura del quale si opera lo scorrimento; area da 0 a 32

Descrizione dell'operazione

SSD (Fai scorrere numero intero a 32 bit con segno a destra) fa scorrere l'intero contenuto dell'accumulatore 1 verso destra, bit per bit. Nelle posizioni di bit che risultano vuote in seguito all'operazione di scorrimento, viene scritto lo stato di segnale del bit di segno. Il bit traslato per ultimo viene caricato nel bit di stato A1. Il numero di bit nella misura del quale operare lo scorrimento viene indicato dall'operando **<numero>** o da un valore riportato nell'accumulatore 2-L-L.

SSD <numero>: il numero di scorrimento viene indicato dall'operando **<numero>**. Sono consentiti valori compresi tra 0 e 32. I bit di stato A0 e OV vengono resettati a "0", se **<numero>** è maggiore di zero. Se il numero di scorrimento è "0" l'operazione di scorrimento viene elaborata come un'operazione **NOP**.

SSD: il numero di scorrimento viene indicato dal valore riportato nell'accumulatore 2-L-L. Sono consentiti valori compresi tra 0 e 255. I numeri di scorrimento > 32 producono sempre il medesimo risultato: ACCU 1 = 32#00000000, A1 = 0, oppure ACCU 1 = 32#FFFFFFFF, A1 = 1. Se il numero di scorrimento > 0, i bit di stato A0 e OV vengono resettati a "0". Se il numero di scorrimento è "0", l'operazione di scorrimento viene elaborata come un'operazione **NOP**.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	x	x	x	-	-	-	-	-

Esempi

Contenu	ACCU1-H				ACCU1-L			
Bit	31 . . .	. .	. .	. . . 16	15 . . .	. .	. .	. . . 0
prima dell'esecuzione di **SSD 7**	1000	1111	0110	0100	0101	1101	0011	1011
dopo l'esecuzione di **SSD 7**	1111	1111	0001	1110	1100	1000	1011	1010

Esempio 1

```
AWL             Spiegazione
L     MD4       //Carica il valore in ACCU 1.
SSD   7         //Fa scorrere i bit in ACCU 1 di 7 posizioni verso destra.
T     MD8       //Trasferisce il risultato a MD8.
```

Esempio 2

```
AWL             Spiegazione
L     +3        //Carica il valore +3 in ACCU 1.
L     MD20      //Carica il contenuto di ACCU 1 in ACCU 2. Carica il valore di MD20 in ACCU 1.
SSD             //Il numero di scorrimento è dato dal valore di ACCU 2-L-L. => Fa scorrere i bit
                //in ACCU 1, considerando il segno, di 3 posizioni verso destra, setta le posizioni
                //vuote sullo stato di segnale del bit di segno.
SPP   NEXT      //Passa all'etichetta di salto NEXT se il bit traslato per ultimo (A1) è = 1.
```

11.1.4 SLW Fai scorere parola a sinistra (a 16 bit)

Formati

SLW
SLW <numero>

Operando	Tipo dati	Descrizione
<numero>	Numero intero, senza segno	Numero di bit, nella misura del quale si opera lo scorrimento; area da 0 a 15

Descrizione dell'operazione

SLW (Fai scorrere parola a sinistra) fa scorrere il contenuto di ACCU 1-L verso sinistra, bit per bit. Le posizioni dei bit, che risultano vuote in seguito all'operazione di scorrimento, vengono alimentate con zeri. Il bit traslato per ultimo viene caricato nel bit di stato A1. Il numero di bit nella misura del quale operare lo scorrimento viene indicato dall'operando <numero> oppure da un valore specificato nell'accumulatore 2-L-L.

SLW <numero>: il numero di bit nella misura del quale operare lo scorrimento viene indicato dall'operando <numero>. Sono consentiti valori compresi tra 0 e 15. I bit di stato A0 e OV vengono resettati a "0", se **<numero>** è maggiore di zero. Se **<numero>** è uguale a "0", l'operazione di scorrimento viene elaborata come un'operazione NOP.

SLW: il numero di scorrimento viene indicato dal valore in ACCU 2-L-L. Sono ammessi i valori da 0 a 255. Un numero di scorrimento > 16 richiama sempre lo stesso risultato: ACCU 1-L = 0, A1 = 0, A0 = 0, OV = 0. Se 0 < numero di scorrimento <= 16,i bit di stato A0 e OV vengono resettati a "0". Se il numero di scorrimento è "0", l'operazione di scorrimento viene elaborata come un'operazione **NOP**.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	x	x	x	-	-	-	-	-

Esempi

Contenu	ACCU1-H				ACCU1-L			
Bit	31 . . .	. .	. .	. . . 16	15 . . .	. .	. .	. . . 0
prima dell'esecuzione di **SLW 5**	0101	1111	0110	0100	0101	1101	0011	1011
dopo l'esecuzione di **SLW 5**	0101	1111	0110	0100	1010	0111	0110	0000

Esempio 1

```
AWL             Spiegazione
L     MW4       //Carica il valore in ACCU 1.
SLW   5         //Fa scorrere i bit in ACCU 1 di 5 posizioni verso sinistra.
T     MW8       //Trasferisce il risultato a MW8.
```

Esempio 2

```
AWL             Spiegazione
L     +3        //Carica il valore +3 in ACCU 1.
L     MW20      //Carica il contenuto di ACCU 1 in ACCU 2. Carica il valore di MW20 in ACCU 1.
SLW             //Il valore di scorrimento è dato dal valore di ACCU 2-L-L. => Fa scorrere i bit
                //in ACCU 1-L di 3 posizioni verso sinistra.
SPP   NEXT      //Passa all'etichetta di salto NEXT se il bit traslato da ultimo (A1) è = 1.
```

11.1.5 SRW Fai scorrere parola a destra (a 16 bit)

Formati

SRW
SRW <numero>

Operando	Tipo dati	Descrizione
<numero>	Numero intero, senza segno	Numero di bit, nella misura del quale si opera lo scorrimento; area da 0 a 15

Descrizione dell'operazione

SRW (Fai scorrere parola a destra) fa scorrere solo il contenuto dell'accumulatore 1-L verso destra, bit per bit. Le posizioni dei bit che risultano vuote in seguito all'operazione di scorrimento, vengono alimentate con zeri. Il bit traslato per ultimo viene caricato nel bit di stato A1. Il numero di bit nella misura del quale operare lo scorrimento viene indicato dall'operando <numero> oppure da un valore specificato nell'accumulatore 2-L-L.

SRW <numero>: il numero di bit nella misura del quale operare lo scorrimento viene indicato dall'operando **<numero>**. Sono consentiti valori compresi tra 0 e 15. I bit di stato A0 e OV vengono resettati a "0", se <numero> è maggiore di zero. Se il numero di scorrimento è "0", l'operazione di scorrimento viene elaborata come un'operazione **NOP**.

SRW: il numero di bit nella misura del quale operare lo scorrimento viene indicato dal valore riportato nell'accumulatore 2-L-L. Sono consentiti valori compresi tra 0 e 255. I numeri di scorrimento > 16 producono sempre il medesimo risultato: ACCU 1 = 0, A1 = 0, A0 = 0, OV = 0. Se 0 < numero di scorrimento <= 16, i bit di stato A0 e OV vengono resettati a "0". Se il numero di scorrimento è "0", l'operazione di scorrimento viene elaborata come un'operazione **NOP**.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	x	x	x	-	-	-	-	-

Esempi

Contenu	ACCU1-H				ACCU1-L			
Bit	31 . . .	. .	. .	. . . 16	15 . . .	. .	. .	. . . 0
prima dell'esecuzione di **SRW 6**	0101	1111	0110	0100	0101	1101	0011	1011
dopo l'esecuzione di **SRW 6**	0101	1111	0110	0100	0000	0001	0111	0100

Esempio 1

```
AWL             Spiegazione
L     MW4       //Carica il valore in ACCU 1.
SRW   6         //Fa scorrere i bit in ACCU 1 di 6 posizioni verso destra.
T     MW8       //Trasferisce il risultato a MW8
```

Esempio 12

```
AWL             Spiegazione
L     +3        //Carica il valore +3 in ACCU 1.
L     MW20      //Carica il contenuto di ACCU 1 in ACCU 2. Carica il valore di MW20 in ACCU 1.
SRW             //Il numero di scorrimento è dato dal valore di ACCU 2-L-L. => Fa scorrere i bit
                //in ACCU 1-L di 3 posizioni verso destra.
SPP   NEXT      //Passa all'etichetta di salto NEXT se il bit traslato per ultimo (A1) è = 1.
```

11.1.6 SLD Fai scorrere doppia parola a sinistra (a 32 bit)

Formati

SLD
SLD <numero>

Operando	Tipo dati	Descrizione
<numero>	Numero intero, senza segno	Numero di bit nella misura del quale si opera lo scorrimento; area da 0 a 32

Descrizione dell'operazione

SLD (Fai scorrere doppia parola a sinistra) fa scorrere l'intero contenuto dell'accumulatore 1 verso sinistra, bit per bit. Le posizioni bit, che risultano vuote in seguito all'operazione di scorrimento, vengono alimentate con zeri. Il bit traslato per ultimo viene caricato nel bit di stato A1. Il numero di bit nella misura del quale operare lo scorrimento, viene indicato dall'operando **<numero>** oppure da un valore riportato in ACCU 2-L-L.

SLD <numero>: il numero di bit nella misura del quale operare lo scorrimento viene indicato dall'operando **<numero>**. Sono consentiti valori compresi tra 0 e 32. I bit di stato A0 e OV vengono resettati a "0", se **<numero>** è maggiore di zero. Se il <numero> "0", l'operazione di scorrimento viene elaborata come un'operazione **NOP**.

SLD: il numero di bit nella misura del quale operare lo scorrimento viene indicato dal valore riportato in ACCU 2-L-L. Sono consentiti valori compresi tra 0 e 255. I numeri di scorrimento > 32 producono sempre il medesimo risultato: accumulatore 1 = 0, A1 = 0, A0 = 0; OV = 0. Se 0 < numero di scorrimento <= 32, i bit di stato A0 e OV vengono resettati "0". Se il numero di scorrimento è "0", l'operazione di scorrimento viene elaborata come un'operazione **NOP**.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	x	x	x	-	-	-	-	-

Esempi

Contenuto	ACCU1-H				ACCU1-L			
Bit	31 . . .	. .	. .	. . . 16	15 . . .	. .	. .	. . . 0
prima dell'esecuzione di **SLD 5**	0101	1111	0110	0100	0101	1101	0011	1011
dopo l'esecuzione di **SLD 5**	1110	1100	1000	1011	1010	0111	0110	0000

Esempio 1

```
AWL             Spiegazione
L     MD4       //Carica il valore in ACCU 1.
SLD   5         //Fa scorrere i bit in ACCU 1 di 5 posizioni verso sinistra.
T     MD8       //Trasferisce il risultato a MD8.
```

Esempio 2

```
AWL             Spiegazione
L     +3        //Carica il valore +3 in ACCU 1.
L     MD20      //Carica il contenuto di ACCU 1 in ACCU 2. Carica il valore di MD20 in ACCU 1.
SLD             //Il numero di scorrimento è dato dal valore di ACCU 2-L-L. => Fa scorrere i bit
                //in ACCU 1 di 3 posizioni verso sinistra
SPP   NEXT      //Passa all'etichetta di salto NEXT se l'ultimo bit traslato (A1) è = 1.
```

11.1.7 SRD Fai scorrere doppia parola a destra (a 32 bit)

Formati

SRD
SRD <numero>

Operando	Tipo dati	Descrizione
<numero>	Numero intero, senza segno	Numero di bit nella misura del quale si opera lo scorrimento; area da 0 a 32

Descrizione dell'operazione

SRD (Fai scorrere doppia parola a destra) fa scorrere l'intero contenuto di ACCU 1 verso destra, bit per bit. Le posizioni bit che risultano vuote in seguito all'operazione di scorrimento, vengono alimentate con zeri. Il bit traslato per ultimo viene caricato nel bit di stato A1. Il numero di bit nella misura del quale operare lo scorrimento viene indicato dall'operando <numero> oppure da un valore riportato nell'accumulatore 2-L-L.

SRD <numero>: il numero di bit nella misura del quale operare lo scorrimento viene indicato dall'operando <**numero**>. Sono consentiti valori compresi tra 0 e 32. I bit di stato A0 e OV vengono resettati a "0", se <numero> è maggiore di zero. Se il numero di scorrimento è "0", l'operazione di scorrimento viene elaborata come un'operazione **NOP**.

SRD: il numero di bit, nella misura del quale operare lo scorrimento, viene indicato dal valore riportato nell'accumulatore 2-L-L. Sono consentiti valori compresi tra 0 e 255. I numeri di scorrimento > 32 producono sempre il medesimo risultato: ACCU 1 = 0, A1 = 0, A0 = 0. Se 0 < numero di scorrimento <= 16, i bit di stato A0 e OV vengono resettati a "0". Se < numero di scorrimento <= 32, i bit di stato A0 e OV vengono resettati a "0". Se il numero di scorrimento è "0", l'operazione di scorrimento viene elaborata come un'operazione **NOP**.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	x	x	x	-	-	-	-	-

Esempi

Contenu	ACCU1-H				ACCU1-L			
Bit	31 . . .	. .	. .	. . . 16	15 . . .	. .	. .	. . . 0
prima dell'esecuzione di **SRD 7**	0101	1111	0110	0100	0101	1101	0011	1011
dopo l'esecuzione di **SRD 7**	0000	0000	1011	1110	1100	1000	1011	1010

Esempio 1

```
AWL             Spiegazione
L     MD4       //Carica il valore in ACCU 1.
SRD   7         //Fa scorrere i bit in ACCU 1 di 7 posizioni verso destra.
T     MD8       //Trasferisce il risultato a MD8.
```

Esempio 2

```
AWL             Spiegazione
L     +3        //Carica il valore +3 in ACCU 1.
L     MD20      //Carica il contenuto di ACCU 1 in ACCU 2. Carica il valore di MD20 in ACCU 1.
SRD             //Il numero di scorrimento è dato dal valore di ACCU 2-L-L. => Fa scorrere i bit
                //in ACCU 1 di 3 posizioni verso destra.
SPP   NEXT      //Passa all'etichetta di salto NEXT se il bit traslato per ultimo (A1) è = 1.
```

11.2 Operazioni di rotazione

11.2.1 Sommario delle operazioni di rotazione

Descrizione

L'utente può adoperare le operazioni di rotazione per far ruotare l'intero contenuto di ACCU 1 verso sinistra o verso destra (vedere Registri CPU), bit per bit. Le operazioni di rotazione avviano funzioni che sono analoghe alle funzioni di scorrimento. Comunque, le cifre bit rimaste libere vengono riempite con gli stati di segnale dei bit che vengono traslati fuori dall'accumulatore.

Il numero che segue l'operazione di rotazione o un valore nel byte basso della parola bassa di ACCU 2 sta ad indicare il numero di bit nella misura del quale si deve operare la rotazione.

In dipendenza dall'operazione, la rotazione avviene tramite il bit A1 della parola di stato. Il bit A0 della parola di stato viene resettato a 0.

Sono disponibili le seguenti operazioni di rotazione:

- RLD Fai ruotare doppia parola a sinistra (a 32 bit)
- RRD Fai ruotare doppia parola a destra (a 32 bit)
- RLDA Fai ruotare ACCU 1 a sinistra tramite A1 (a 32 bit)
- RRDA Fai ruotare ACCU 1 a destra tramite A1 (a 32 bit)

11.2.2 RLD Fai ruotare doppia parola a sinistra (a 32 bit)

Formati

RLD
RLD <numero>

Operando	Tipo dati	Descrizione
<numero>	Numero intero, senza segno	Numero di bit nella misura del quale si opera la rotazione; area da 0 a 32

Descrizione dell'operazione

RLD (Fai ruotare doppia parola a sinistra) fa ruotare l'intero contenuto dell'accumulatore 1 verso sinistra, bit per bit. I posti rimasti liberi vengono riempiti con gli stati di segnale dei bit, che vengono traslati fuori dall'accumulatore 1. Il bit traslato per ultimo viene caricato nel bit di stato A1. Il numero di bit, nella misura del quale operare la rotazione, viene indicato dall'operando **<numero>** oppure da un valore riportato nell'accumulatore 2-L-L.

RLD <numero>: il numero di posti per cui effettuare la rotazione viene indicato dall'operando <numero>. Sono consentiti valori compresi tra 0 e 32. I bit di stato A0 e OV vengono resettati a "0" se <numero> è maggiore di zero. Se tale numero è "0", l'operazione di rotazione viene elaborata come un'operazione **NOP**.

RLD: il numero di posti per cui effettuare la rotazione viene indicato dal valore riportato nell'accumulatore 2-L-L. Sono amessi i valori tra 0 e 255. I bit di stato A0 e OV vengono resettati a "0" se **<numero>** è maggiore di zero. Sono consentiti valori compresi tra 0 e 255. Se tale numero è "0", l'operazione di rotazione viene elaborata come un'operazione **NOP**.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	x	x	x	-	-	-	-	-

Esempi

Contenuto	ACCU1-H				ACCU1-L			
Bit	31 . . .	. .	. .	. . . 16	15 . . .	. .	. .	. . . 0
prima dell'esecuzione di **RLD 4**	0101	1111	0110	0100	0101	1101	0011	1011
dopo l'esecuzione di **RLD 4**	1111	0110	0100	0101	1101	0011	1011	0101

Esempio 1

```
AWL             Spiegazione
L     MD2       //Carica il valore in ACCU 1.
RLD   4         //Fa ruotare i bit in ACCU 1 di 4 posizioni.
T     MD8       //Trasferisce il risultato a MD8.
```

Esempio 2

```
AWL             Spiegazione
L     +3        //Carica il valore  +3 in ACCU 1.
L     MD20      //Carica il contenuto di ACCU 1 in ACCU 2 Carica il valore di MD20 in ACCU 1.
RLD             //Il numero di posti per cui effettuare la rotazione è dato dal valore di
                //ACCU 2-L-L. => Ruota i bit in ACCU 1 di 3 posizioni verso sinistra.
SPP   NEXT      //Passa all'etichetta di salto NEXT se l'ultimo bit traslato (A1) è = 1.
```

11.2.3 RRD Fai ruotare doppia parola a destra (a 32 bit)

Formati

RRD
RRD <numero>

Operando	Tipo dati	Descrizione
<numero>	Numero intero, senza segno	Numero di bit nella misura del quale si opera la rotazione; area da 0 a 32

Descrizione dell'operazione

RRD (Fai ruotare la doppia parola a destra) fa ruotare l'intero contenuto dell'accumulatore 1 verso destra, bit per bit. I posti rimasti liberi vengono riempiti con gli stati di segnale dei bit che vengono traslati fuori dall'accumulatore 1. Il bit traslato per ultimo viene caricato nel bit di stato A1. Il numero di bit nella misura del quale operare la rotazione viene indicato dall'operando **<numero>**, oppure da un valore riportato in ACCU 2-L-L.

RRD: il numero di posti per cui effettuare la rotazione viene indicato dall'operando **<numero>.** Sono consentiti valori compresi tra 0 e 255. I bit di stato A0 e OV vengono resettati a "0", se **<numero>** è maggiore di zero. Se tale numero è "0", l'operazione di rotazione viene elaborata come un'operazione NOP.

RRD: il numero di posti per cui effettuare la rotazione viene indicato dal valore riportato nell'accumulatore 2-L-L. Sono consentiti valori compresi tra 0 e 255. I bit di stato A0 e OV vengono resettati a "0" se **<numero>** è maggiore di zero. Se tale numero è "0", l'operazione di rotazione viene elaborata come un'operazione **NOP**.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	x	x	x	-	-	-	-	-

Esempi

Contenu	ACCU1-H				ACCU1-L			
Bit	31 . . .	. .	. .	. . . 16	15 . . .	. .	. .	. . . 0
prima dell'esecuzione di **RRD 4**	0101	1111	0110	0100	0101	1101	0011	1011
dopo l'esecuzione di **RRD 4**	1011	0101	1111	0110	0100	0101	1101	0011

Esempio 1

```
AWL             Spiegazione
L     MD2       //Carica il valore in ACCU 1.
RRD   4         //Fa ruotare i bit di ACCU 1 di 4 posizioni verso destra.
T     MD8       //Trasferisce il risultato a MD8.
```

Esempio 2

```
AWL             Spiegazione
L     +3        //Carica il valore +3 in ACCU 1.
L     MD20      //Carica il contenuto di ACCU 1 in ACCU 2. Carica il valore di MD20 in ACCU 1.
RRD             //Il numero di bit per cui operare la rotazione è dato dal valore di ACCU 2-L-L.
                //=> Fa ruotare i bit in ACCU 1 di tre posizioni verso destra.
SPP   NEXT      //Passa all'etichetta di salto NEXT se l'ultimo bit traslato (A1) è = 1.
```

11.2.4 RLDA Fai ruotare ACCU 1 a sinistra tramite A1 (a 32 bit)

Formato

RLDA

Descrizione dell'operazione

RLDA (Fai ruotare doppia parola a sinistra tramite A1) fa ruotare l'intero contenuto dell'accumulatore 1 verso sinistra di una posizione bit tramite il bit di visualizzazione A1. I bit di stato A0 e OV vengono resettati a "0".

Parola di stato

	BIE	**A1**	**A0**	**OV**	**OS**	**OR**	**STA**	**RLC**	**/ER**
Scrive:	-	x	0	0	-	-	-	-	-

Esempi

Contenuto	**A1**	**ACCU1-H**				**ACCU1-L**			
Bit		31 . . .	. .	. .	. . . 16	15 . . .	. .	. .	. . . 0
prima dell'esecuzione di **RLDA**	**X**	**0**101	1111	0110	0100	0101	1101	0011	1011
dopo l'esecuzione di **RLDA**	**0**	1011	1110	1100	1000	1011	1010	0111	011**X**
	(X = 0 o 1, stato di segnale precedente di A1)								

```
AWL             Spiegazione
L     MD2       //Carica il valore in ACCU 1.
RLDA            //Fa ruotare i bit in ACCU 1 di una posizione verso sinistra tramite A1.
SPP   NEXT      //Passa all'etichetta di salto NEXT, se il bit traslato da ultimo (A1)
                //è  = 1.
```

11.2.5 RRDA Fai ruotare ACCU 1 a destra tramite A1 (a 32 bit)

Formato

RRDA

Descrizione dell'operazione

RRDA (Fai ruotare doppia parola a destra tramite A1) fa ruotare di una posizione bit l'intero contenuto dell'accumulatore 1 verso destra. I bit di stato A0 e OV vengono resettati a "0".

Parola di stato

	BIE	**A1**	**A0**	**OV**	**OS**	**OR**	**STA**	**RLC**	**/ER**
Scrive:	-	x	0	0	-	-	-	-	-

Esempi

Contenuto	**A1**	**ACCU1-H**				**ACCU1-L**			
Bit		31 . . .	. .	. .	. . . 16	15 . . .	. .	. .	. . . 0
prima dell'esecuzione di **RRDA**	**X**	0101	1111	0110	0100	0101	1101	0011	101**1**
dopo l'esecuzione di **RRDA**	**1**	**X**010	1111	1011	0010	0010	1110	1001	1101
	(X = 0 o 1, stato di segnale di A1 precedente)								

```
AWL             Spiegazione
L     MD2       //Carica il valore in ACCU 1.
RRDA            //Fa ruotare i bit in ACCU 1 di una posizione verso destra tramite A1.
SPP   NEXT      //Passa all'etichetta di salto NEXT se il bit traslato per ultimo (A1) è = 1.
```

12 Operazioni di temporizzazione

12.1 Sommario delle operazioni di temporizzazione

Descrizione

Al paragrafo "Aree di memoria e componenti di un temporizzatore" si trovano informazioni e la selezione dei temporizzatori.

Sono disponibili le seguenti operazioni di temporizzatore:

- FR Abilita temporizzatore
- L Carica valore attuale di conteggio in ACCU 1 come numero intero
- LC Carica valore attuale di conteggio in ACCU 1 come BCD
- R Resetta temporizzatore
- SI Avvia temporizzatore come impulso
- SV Avvia temporizzatore come impulso prolungato
- SE Avvia temporizzatore come ritardo all'inserzione
- SS Avvia temporizzatore come ritardo all'inserzione con memoria
- SA Avvia temporizzatore come ritardo alla disinserzione

12.2 Aree di memoria e componenti di un temporizzatore

Area di memoria

I temporizzatori hanno un'area riservata nella memoria della CPU. Quest'area di memoria riserva una parola a 16 bit per ogni operando del temporizzatore. Il set di operazioni logiche KOP supporta 256 temporizzatori. Le parole di tempo rizzazione a disposizione per la CPU utilizzata sono riportate nei relativi dati tecnici.

Le seguenti funzioni hanno accesso all'area di memoria del temporizzatore:

- Operazioni di temporizzazione
- Aggiornamento di parole di temporizzazione mediante generatore di clock. Questa funzione della CPU nello stato di funzionamento RUN decrementa un determinato valore di un'unità in intervalli definiti dalla base di tempo finché il valore temporale non è uguale a zero.

Valore di tempo

I bit da 0 a 9 della parola di temporizzazione contengono il valore temporale in codice binario. Questo valore specifica un numero di unità. L'aggiornamento del tempo decrementa il valore di un'unità in intervalli definiti dalla base di tempo. Il decremento continua finché il valore temporale non è uguale a zero. Si può caricare un valore di tempo in formato binario, esadecimale o decimale codificato in binario (BCD).

Si può precaricare un valore di tempo in uno dei seguenti formati:

- W#16#txyz

- laddove t = base di tempo (ossia, l'intervallo di tempo o risoluzione)
- laddove xyz = valore in formato BCD

- S5T#a**H**_b**M**_c**S**_d**MS**

- laddove H = ore, M = minuti, S = secondi, MS = millisecondi;
  a, b, c, d vengono definiti dall'utente
- La base di tempo viene selezionata automaticamente e il valore viene arrotondato al numero immediatamente inferiore rispetto ad essa.

Il valore di tempo massimo che si può immettere è 9.990 secondi, o 2H_46M_30S. Esempi:

S5TIME#4S --> 4 secondi
s5t#2h_15m --> 2 ore e 15 minuti
S5T#1H_12M_18S --> 1 ora, 12 minuti e 18 secondi

Base di tempo

I bit 12 e 13 della parola di temporizzazione contengono la base di tempo in codice binario. La base di tempo definisce l'intervallo di decremento di un'unità. La base di tempo più piccola è 10 ms; la più grande è 10 s.

Base di tempo	Codice binario per la base di tempo
10 ms	00
100 ms	01
1 s	10
10 s	11

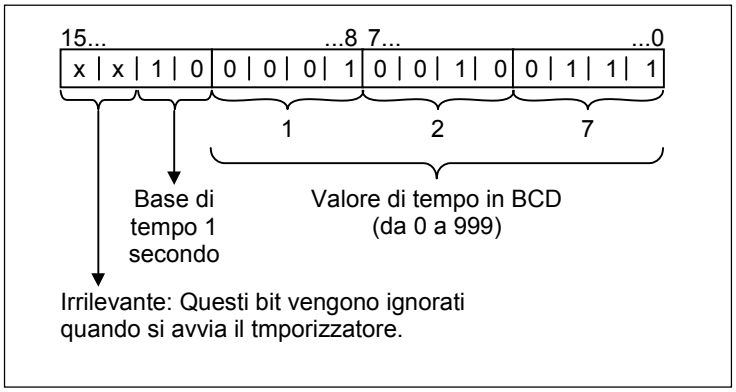
I valori non devono superare 2H_46M_30S. I valori con un'area o una risoluzione troppo grande (ad es. 2H_10MS) vengono arrotondati così da corrispondere alla tabella per l'area e la risoluzione. Il formato generale per il tipo di dati S5TIME ha i seguenti valori limite per l'area e la risoluzione:

Risoluzione	Area
0,01 secondi	Da 10MS a 9S_990MS
0,1 secondi	Da 100MS a 1M_39S_900MS
1 secondo	Da 1S a 16M_39S
10 secondi	Da 10S a 2H_46M_30S

Configurazione dei bit nella cella di tempo

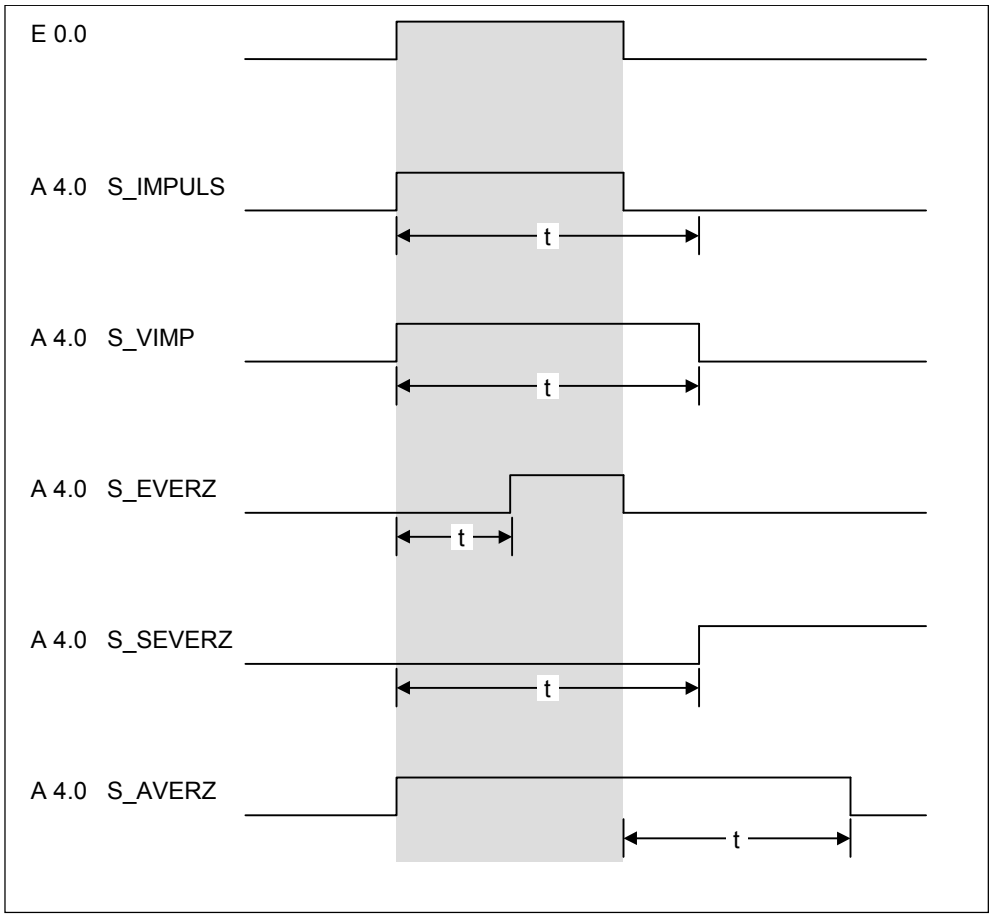
Quando si avvia un temporizzatore, il contenuto della cella di tempo viene utilizzato come valore di tempo. I bit da 0 a 11 della cella di tempo registrano il valore di tempo in formato decimale codificato in binario (formato BCD: ogni serie di quattro bit contiene il codice binario di un valore decimale). I bit 12 e 13 registrano la base di tempo in codice binario.

La figura mostra il contenuto della cella di tempo caricato con il valore di temporizzazione 127 e una base di tempo di 1 secondo:



Scelta del giusto temporizzatore

La figura illustra cinque tipi di temporizzatori che sono stati descritti in questo capitolo. Questo riepilogo vuole essere d'aiuto all'utente nella scelta del temporizzatore più adeguato ai suoi fini di utilizzo.



Temporizzatore	Descrizione
S_IMPULS Avvia temporizzatore come impulso	Il tempo massimo in cui il segnale di uscita resta a 1, è uguale al valore di tempo programmato t. Il segnale di uscita resta a 1 per un tempo più breve se il segnale di ingresso passa a 0.
S_VIMP Avvia temporizzatore come impulso prolungato	Il segnale di uscita resta a 1 per la durata programmata, indipendentemente dal tempo che il segnale di ingresso resta a 1.
S_EVERZ Avvia temporizzatore come ritardo all'inserzione	Il segnale di uscita è 1 solo quando è trascorso il tempo programmato e il segnale di ingresso è ancora 1.
S_SEVERZ Avvia temporizzatore come ritardo all'inserzione con memoria	Il segnale di uscita passa da 0 a 1 solo quando è trascorso il tempo programmato, indipendentemente dal tempo in cui il segnale di ingresso resta a 1.
S_AVERZ Avvia temporizzatore come ritardo alla disinserzione	Il segnale di uscita è 1 quando il segnale di ingresso è 1. Il segnale di uscita resta a 1 per la durata programmata. Il tempo viene avviato quando il segnale di ingresso cambia da 1 a 0.

12.3 FR Abilita temporizzatore

Formato

FR <temporizzatore>

Operando	Tipo dati	Area memoria	Descrizione
<temporizzatore>	TIMER	T	Numero del temporizzatore; l'area dipende dalla CPU

Descrizione dell'operazione

FR <temporizzatore> cancella il merker del fronte utilizzato per avviare il temporizzatore indirizzato quando RLC passa da "0" a "1". Il passaggio del bit RLC da "0" a "1" prima dell'esecuzione dell'operazione FR abilita il temporizzatore.

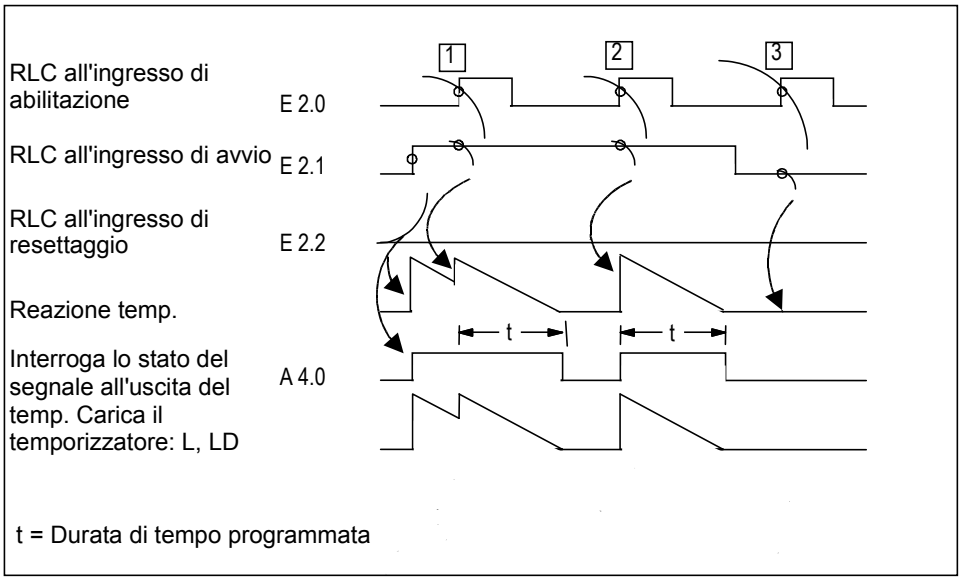
L'operazione di abilitazione del temporizzatore non costituisce una premessa indispensabile all'avviamento del temporizzatore. Il suo impiego è obbligatorio solamente nei casi in cui sia necessario riavviare un temporizzatore già in funzione. Il riavvio è, infatti, eseguibile solo se l'operazione di avviamento continua ad essere elaborata con RLC = 1.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	0	-	-	0

Esempio

```
AWL             Spiegazione
U     E 2.0
FR    T1        //Abilita il temporizzatore T1.
U     E 2.1
L     S5T#10s   //Predispone una preimpostazione di 10 secondi nell'accumulatore 1.
SI    T1        //Avvia il temporizzatore T1 come impulso.
U     E 2.2
R     T1        //Resetta il temporizzatore T1.
U     T1        //Interroga lo stato di segnale del temporizzatore T1.
=     A 4.0
L     T1        //Carica il valore di tempo attuale del temporizzatore T1 sotto forma di cifra
                //binaria.
T     MW10
```



(1) Passaggio dell'RLC da "0" a "1" all'ingresso di abilitazione. Durante il trascorrimento del tempo viene riavviato il temporizzatore. La durata di tempo programmata corrisponde al tempo attuale per il riavviamento. Il passaggio da "1" a "0" dell'RLC all'ingresso di abilitazione non incide.

(2) Passaggio dell'RLC da "0" a "1" all'ingresso di abilitazione. Il tempo non trascorre e quando l'RLC all'ingresso di avvio è "1", il temporizzatore viene avviato come impulso con il valore di tempo programmato.

(3) Passaggio dell'RLC da "0" a "1" all'ingresso di abilitazione. Quando l'RLC all'ingresso di avvio è "0" non si ha alcuna variazione del temporizzatore.

12.4 L Carica valore attuale di conteggio in ACCU 1 come numero intero

Formato

L <temporizzatore>

Operando	Tipo dati	Area memoria	Descrizione
<temporizzatore>	TIMER	T	Numero del temporizzatore; l'area dipende dalla CPU

Descrizione dell'operazione

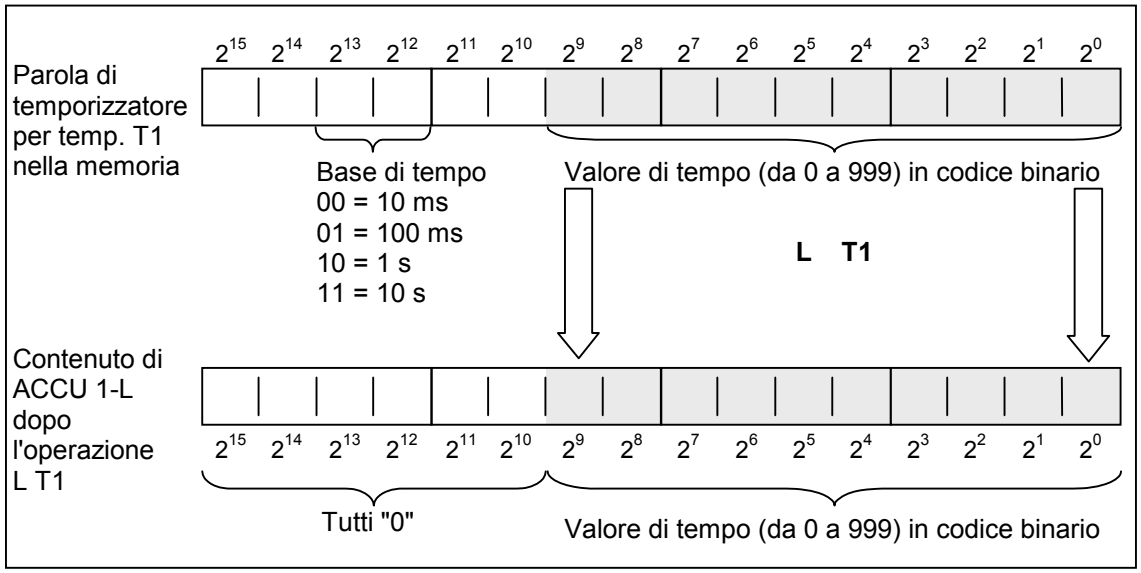
L <temporizzatore> carica il valore attuale di conteggio dalla parola di temporizzazione indirizzata in formato binario intero nell'accumulatore 1-L, dopo che il contenuto dell'accumulatore 1 è stato caricato nell'accumulatore 2.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio

```
AWL             Spiegazione
L     T1        //Carica nell'accumulatore 1-L il valore attuale di conteggio del temporizzatore
                //T1 in codice binario.
```



Nota

L <temporizzatore> carica soltanto il codice binario del val. di tempo attuale nell'acc. 1-L e non la base di decrementato del tempo trascorso a partire dall'avvio della funzione di temporizzazione.tempo. Il valore di tempo caricato rappresenta il valore iniziale del temporizzatore,

12.5 LC Carica valore attuale di conteggio in ACCU 1 come BCD

Formato

LC <temporizzatore>

Operando	Tipo dati	Area memoria	Descrizione
<temporizzatore>	TIMER	T	Numero del temporizzatore; l'area dipende dalla CPU

Descrizione dell'operazione

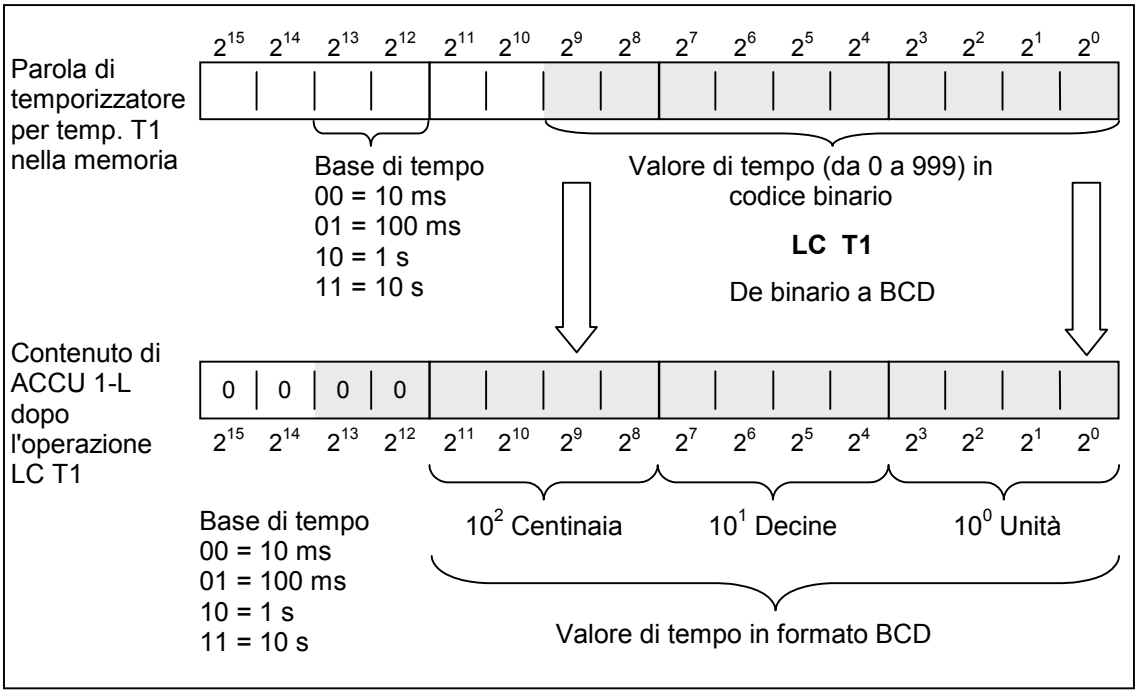
LC <temporizzatore> carica il valore e la base di tempo dalla parola di temporizzazione indirizzata in formato BCD (decimale in codice binario) nell'accumulatore 1, dopo che il contenuto dell'accumulatore 1 è stato caricato nell'accumulatore 2.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio

```
AWL              Spiegazione
LC    T1         //Carica nell'accumulatore 1-L la base ed il valore di tempo attuale del
                 //temporizzatore T1 in formato BCD.
```



12.6 R Resetta temporizzatore

Formato

R <temporizzatore>

Operando	Tipo dati	Area memoria	Descrizione
<temporizzatore>	TIMER	T	Numero del temporizzatore; l'area dipende dalla CPU

Descrizione dell'operazione

R <temporizzatore> termina la funzione di temporizzazione attuale e cancella il valore e la base di tempo della parola di temporizzazione indirizzata, se RLC passa da 0 a 1.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	0	-	-	0

Esempio

```
AWL             Spiegazione
U     E 2.1
R     T1        //Interroga lo stato di segnale dell'ingresso E 2.1. Se RLC passa da 0 a 1,
                resetta il temporizzatore T1.
```

12.7 SI Avvia temporizzatore come impulso

Formato

SI <temporizzatore>

Operando	Tipo dati	Area memoria	Descrizione
<temporizzatore>	TIMER	T	Numero del temporizzatore; l'area dipende dalla CPU

Descrizione dell'operazione

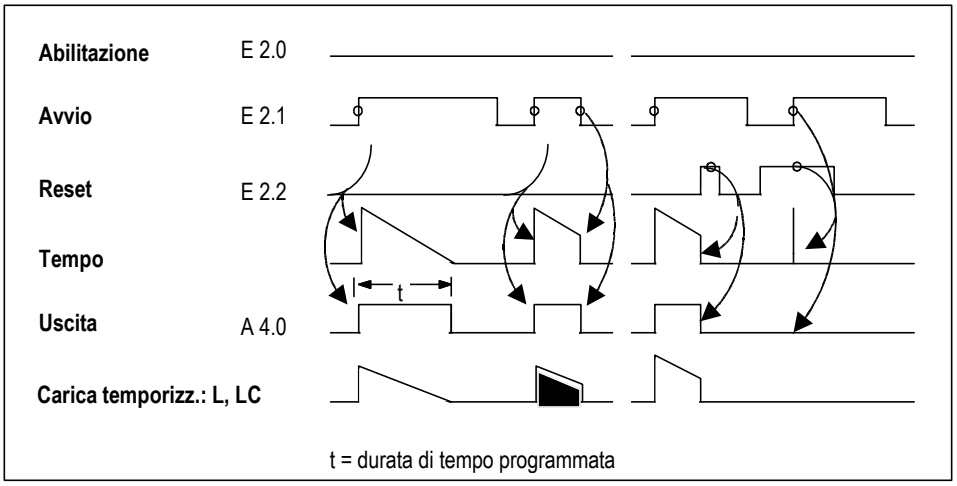
SI <temporizzatore> avvia il temporizzatore indirizzato quando RLC passa da "0" a "1". Il count-down della durata programmata continua finché RLC = 1. Il passaggio di RLC a "0" determina l'arresto del temporizzatore. Per eseguire l'operazione di avviamento del temporizzatore, il valore e la base di tempo devono essere stati memorizzati in formato BCD nell'accumulatore 1-L.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	0	-	-	0

Esempio

```
AWL             Spiegazione
U     E 2.0
FR    T1        //Abilita il temporizzatore T1.
U     E 2.1
L     S5T#10s   //Predispone una preimpostazione di 10 secondi nell'accumulatore 1.
SI    T1        //Avvia il temporizzatore T1 come impulso.
U     E 2.2
R     T1        //Resetta il temporizzatore T1.
U     T1        //Interroga lo stato di segnale del temporizzatore T1.
=     A 4.0
L     T1        //Carica il valore di tempo attuale del temporizzatore T1 in formato binario.
T     MW10
LC    T1        //Carica il valore di tempo attuale del temporizzatore T1 in formato BCD.
T     MW12
```



12.8 SV Avvia temporizzatore come impulso prolungato

Formato

SV <temporizzatore>

Operando	Tipo dati	Area memoria	Descrizione
<temporizzatore>	TIMER	T	Numero del temporizzatore; l'area dipende dalla CPU

Descrizione dell'operazione

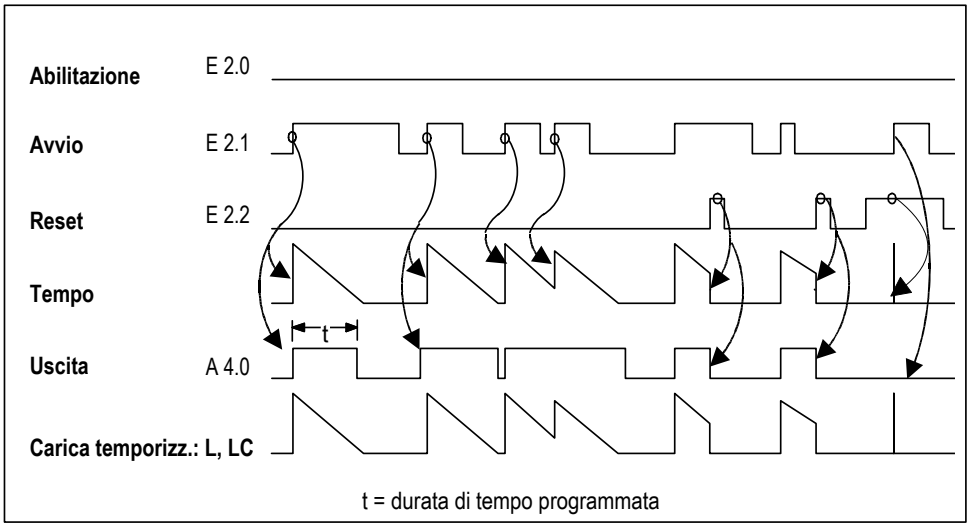
SV <temporizzatore> avvia il temporizzatore indirizzato quando RLC passa da "0" a "1". Il count-down del tempo programmato prosegue anche se nel frattempo RLC ritorna a "0". Il passaggio di RLC da "0" a "1" prima della scadenza del tempo programmato determina il riavvio della durata di tempo programmata. Per eseguire l'operazione di avviamento del temporizzatore, il valore e la base di tempo devono essere stati memorizzati in formato BCD nell'accumulatore 1-L.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	0	-	-	0

Esempio

```
AWL             Spiegazione
U     E 2.0
FR    T1        //Abilita il temporizzatore T1.
U     E 2.1
L     S5T#10s   //Predispone una preimpostazione di 10 secondi nell'accumulatore 1.
SV    T1        //Avvia il temporizzatore T1 come impulso ritardato.
U     E 2.2
R     T1        //Resetta il temporizzatore T1.
U     T1        //Interroga lo stato di segnale del temporizzatore T1.
=     A 4.0
L     T1        //Carica il valore di tempo attuale del temporizzatore T1 in formato binario
T     MW10
LC    T1        //Carica il valore di tempo attuale del temporizzatore T1 in formato BCD.
T     MW12
```



12.9 SE Avvia temporizzatore come ritardo all'inserzione

Formato

SE <temporizzatore>

Operando	Tipo dati	Area memoria	Descrizione
<temporizzatore>	TIMER	T	Numero del temporizzatore; l'area dipende dalla CPU

Descrizione dell'operazione

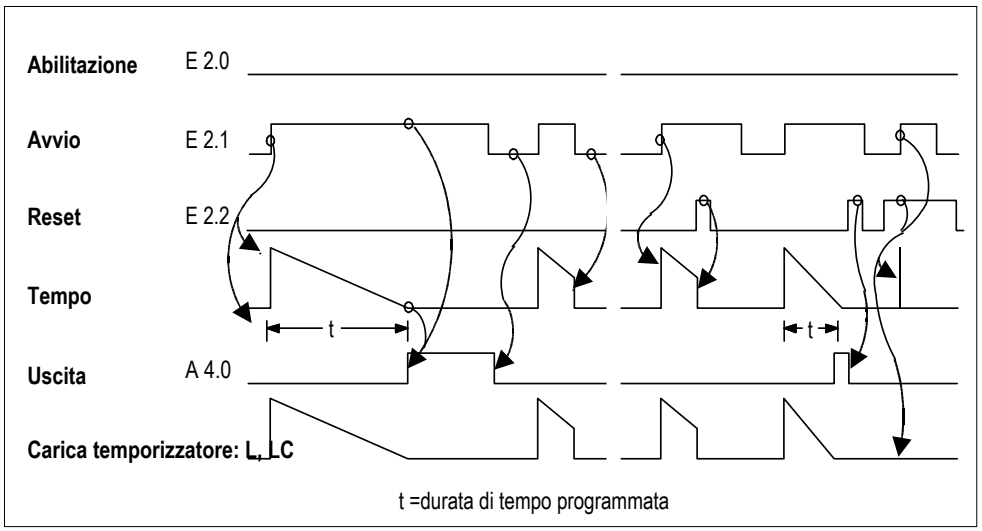
SE <temporizzatore> avvia il temporizzatore indirizzato quando RLC passa da "0" a "1". Il count-down del tempo programmato viene eseguito finché RLC = 1. Un passaggio di RLC a "0" prima della scadenza della durata di tempo programmata determina l'arresto del temporizzatore. Per eseguire l'operazione di avviamento del temporizzatore, il valore e la base di tempo devono essere stati memorizzati in formato BCD nell'accumulatore 1-L.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	0	-	-	0

Esempio

```
AWL             Spiegazione
U     E 2.0
FR    T1        //Abilita il temporizzatore T1.
U     E 2.1
L     S5T#10s   //Predispone una preimpostazione di 10 secondi nell'accumulatore 1.
SE    T1        //Avvia il temporizzatore T1 come ritardo all'inserzione.
U     E 2.2
R     T1        //Resetta il temporizzatore T1.
U     T1        //Interroga il segnale di stato del temporizzatore T1.
=     A 4.0
L     T1        //Carica il valore di tempo attuale del temporizzatore T1 in formato binario.
T     MW10
LC    T1        //Carica il valore di tempo attuale del temporizzatore T1 in formato BCD.
T     MW12
```



12.10 SS Avvia temporizzatore come ritardo all'inserzione con memoria

Formato

SS <temporizzatore>

Operando	Tipo dati	Area memoria	Descrizione
<temporizzatore>	TIMER	T	Numero del temporizzatore; l'area dipende dalla CPU

Descrizione dell'operazione

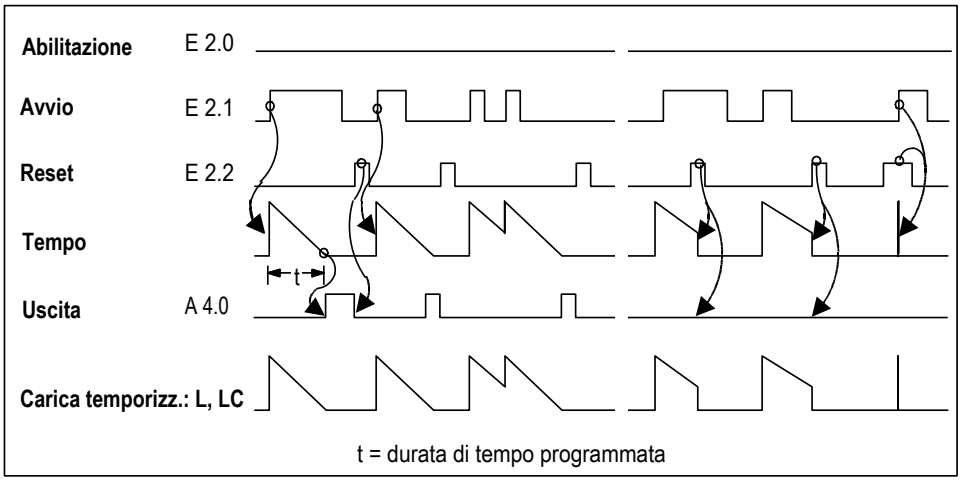
SS <temporizzatore> (Avvia temporizzatore come ritardo all'inserzione con memoria) avvia il temporizzatore indirizzato quando RLC passa da "0" a "1". Il count-down della durata programmata prosegue anche se nel frattempo RLC passa a "0". Il passaggio di RLC da "0" a "1" prima della scadenza del tempo programmato determina il riavvio della durata programmata. Per eseguire l'operazione di avviamento del temporizzatore, il valore e la base di tempo devono essere stati memorizzati in formato BCD nell'accumulatore 1-L.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	0	-	-	0

Esempio

```
AWL             Spiegazione
U     E 2.0
FR    T1        //Abilita il temporizzatore T1.
U     E 2.1
L     S5T#10s   //Predispone una preimpostazione di 10 secondi nell'accumulatore 1.
SS    T1        //Avvia il temporizzatore T1 come ritardo all'inserzione con memoria.
U     E 2.2
R     T1        //Resetta il temporizzatore T1.
U     T1        //Interroga lo stato di segnale del temporizzatore T1.
=     A 4.0
L     T1        //Carica il valore di tempo attuale del temporizzatore T1 in formato binario.
T     MW10
LC    T1        //Carica il valore di tempo attuale del temporizzatore T1 in formato BCD.
T     MW12
```



12.11 SA Avvia temporizzatore come ritardo alla disinserzione

Formato

SA <temporizzatore>

Operando	Tipo dati	Area memoria	Descrizione
<temporizzatore>	TIMER	T	Numero del temporizzatore; l'area dipende dalla CPU

Descrizione dell'operazione

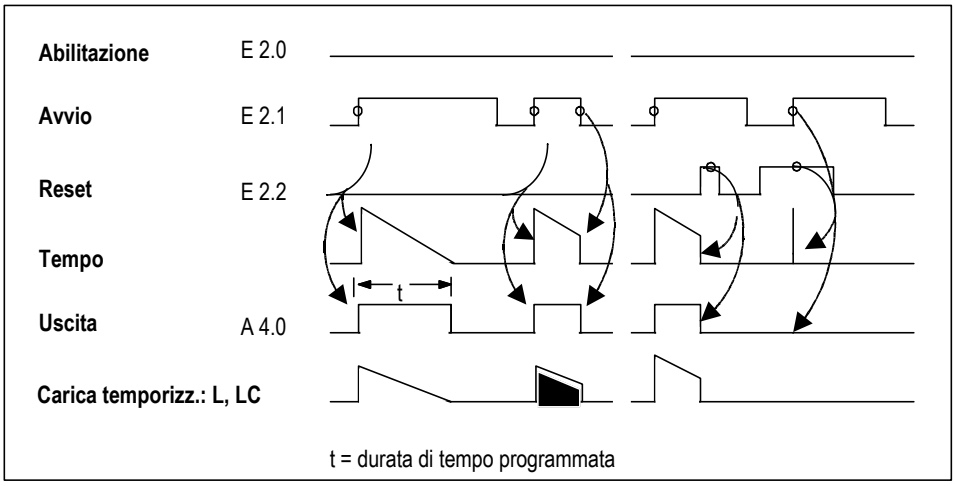
SA <temporizzatore> avvia il temporizzatore indirizzato quando RLC passa da "1" a "0". Il count-down del tempo programmato prosegue finché RLC = 0. Il passaggio di RLC a "1" prima della scadenza della durata di tempo programmata determina l'arresto del temporizzatore. Per eseguire l'operazione di avviamento del temporizzatore, il valore e la base di tempo devono essere stati memorizzati in formato BCD nell'accumulatore 1-L.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	0	-	-	0

Esempio

```
AWL             Spiegazione
U     E 2.0
FR    T1        //Abilita il temporizzatore T1.
U     E 2.1
L     S5T#10s   //Predispone una preimpostazione di 10 secondi nell'accumulatore 1.
SA    T1        //Avvia il temporizzatore T1 come ritardo alla disinserzione.
U     E 2.2
R     T1        //Resetta il temporizzatore T1.
U     T1        //Interroga lo stato di segnale del temporizzatore T1.
=     A 4.0
L     T1        //Carica il valore di tempo attuale del temporizzatore T1 in formato binario.
T     MW10
LC    T1        //Carica il valore di tempo attuale del temporizzatore T1 in formato BCD.
T     MW12
```



13 Operazioni logiche a parola

13.1 Sommario delle operazioni logiche a parola

Descrizione

Le operazioni logiche a parola combinano coppie di parole (a 16 bit) o doppie parole (a 32 bit) bit per bit, in conformità alla logica booleana. Ognuna delle due parole o doppie parole deve trovarsi in uno dei due accumulatori.

Per le parole, il contenuto della parola bassa di ACCU 2 viene combinato con il contenuto della parola bassa di ACCU 1. Il risultato della combinazione viene memorizzato nella parola bassa di ACCU 1, sovrascrivendo il contenuto precedente.

Per le doppie parole, il contenuto di ACCU 2 viene combinato con il contenuto di ACCU 1. Il risultato della combinazione viene memorizzato in ACCU 1, sovrascrivendo il contenuto precedente.

Se il risultato logico combinatorio è 0, il bit A1 della parola di stato viene resettato a 0. Se il risultato logico combinatorio non è 0, A1 viene settato a 1.

Sono disponibili le seguenti operazioni logiche a parola:

- UW AND a parola (a 16 bit)
- OW OR a parola (a 16 bit)
- XOW OR esclusivo a parola (a 16 bit)
- UD AND a doppia parola (a 32 bit)
- OD OR a doppia parola (a 32 bit)
- XOD OR esclusivo a doppia parola (a 32 bit)

13.2 UW AND a parola (a 16 bit)

Formati

UW
UW <costante>

Operando	Tipo dati	Descrizione
<costante>	WORD, costante (a 16 bit)	Configurazione binaria da combinare a ACCU 1-L mediante AND.

Descrizione dell'operazione

UW (AND a parola) combina il contenuto dell'accumulatore 1-L con il contenuto dell'accumulatore 2-L, oppure con una costante (a 16 bit), bit per bit, secondo l'operazione logica combinatoria AND. Il bit della parola di risultato è "1" solamente se i corrispondenti bit delle parole combinate sono entrambi "1". Il risultato viene memorizzato nell'accumulatore 1-L. Gli accumulatori 1-H e 2 (e per le CPU con quattro accumulatori anche ACCU 3 e ACCU 4) rimangono inalterati. Il bit di stato A1 viene impostato conformemente all'esito dell'operazione (A1=1 se il risultato è diverso da zero). I bit di stato A0 e OV vengono resettati a "0".

UW: combina il contenuto dell'accumulatore 1-L con il contenuto dell'accumulatore 2-L.

UW <costante>: combina il contenuto dell'accumulatore 1-L con una costante (a 16 bit).

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	x	0	0	-	-	-	-	-

Esempi

Bit	15 . .	. .	. .	. . . 0
ACCU 1-L prima dell'esecuzione di **UW**	0101	1001	0011	1011
ACCU 2-L o costante (a 16 bit):	1111	0110	1011	0101
Risultato (ACCU 1-L) dopo l'esecuzione di **UW**	0101	0000	0011	0001

Esempio 1

```
AWL             Spiegazione
L     EW20      //Carica il contenuto di EW20 in ACCU 1-L.
L     EW22      //Carica il contenuto di ACCU 1 in ACCU 2. Carica il contenuto di EW22 in ACCU 1.
UW              //Combina i bit di ACCU 1-L ai bit di ACCU 2-L mediante AND, memorizza il
                //risultato in ACCU 1-L.
T     MW 8      //Trasferisce il risultato a MW8.
```

Esempio 2

```
AWL             Spiegazione
L     EW20      //Carica il contenuto di EW20 in ACCU 1-L.
UW    W#16#0FFF //Combina i bit di ACCU 1-L con la configurazione binaria della costante
                //(a 16 bit) (0000_1111_1111_1111) mediante AND, memorizza il risultato in
                //ACCU 1-L.
SPP   NEXT      //Passa all'etichetta di salto NEXT, se il risultato è diverso da zero (A1=1).
```

13.3 OW OR a parola (a 16 bit)

Formati

OW
OW <costante>

Operando	Tipo dati	Descrizione
<costante>	WORD, costante (a 16 bit)	Configurazione binaria da combinare a ACCU 1-L mediante OR.

Descrizione dell'operazione

OW (OR a parola) combina il contenuto dell'accumulatore 1-L con il contenuto dell'accumulatore 2-L oppure con una costante (a 16 bit), bit per bit, secondo l'operazione logica combinatoria OR. Il bit della parola di risultato è "1" se almeno uno dei bit corrispondenti di entrambe le parole è "1". Il risultato viene memorizzato nell'accumulatore 1-L. ACCU 1-H e 2 (e per le CPU con quattro accumulatori ACCU 3 e ACCU 4) rimangono inalterati. Il bit di stato A1 viene impostato conformemente all'esito dell'operazione (A1=1, se il risultato è diverso da zero). I bit di stato A0 e OV vengono resettati a "0".

OW: combina i contenuti degli accumulatori 1-L e 2-L.

OW <costante>: combina il contenuto dell'accumulatore 1-L con una costante (a 16 bit).

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	x	0	0	-	-	-	-	-

Esempi

Bit	**15 . .**	**. .**	**. .**	**. . . 0**
ACCU 1-L prima dell'esecuzione di **OW**	0101	0101	0011	1011
ACCU 2-L o costante (a 16 bit):	1111	0110	1011	0101
Risultato (ACCU 1-L) dopo l'esecuzione di **OW**	1111	0111	1011	1111

Esempio 1

```
AWL             Spiegazione
L     EW20      //Carica il contenuto di EW20 in ACCU 1-L.
L     EW22      //Carica il contenuto di ACCU 1 in ACCU 2. Carica il contenuto di EW22 in ACCU 1.
OW              //Combina i bit di ACCU 1-L con i bit di ACCU 2-L mediante OR, memorizza il
                //risultato in ACCU 1-L.
T     MW8       //Trasferisce il risultato a MW 8.
```

Esempio 2

```
AWL             Spiegazione
L     EW20      //Carica il contenuto di EW20 in ACCU 1-L.
OW    W#16#0FFF //Combina i bit di ACCU 1-L con la configurazione binaria della costante
                //(a 16 bit) (0000_1111_1111_1111) mediante OR, memorizza il risultato in
                //ACCU 1-L.
SPP   NEXT      //Passa all'etichetta di salto NEXT, se il risultato è diverso da zero
                //(A1 = 1).
```

13.4 XOW R esclusivo a parola (a 16 bit)

Formati

XOW
XOW <costante>

Operando	Tipo dati	Descrizione
<costante>	WORD, costante (a 16 bit)	Configurazione binaria da combinare a ACCU 1-L mediante OR esclusivo.

Descrizione dell'operazione

XOW (OR esclusivo a parola) combina il contenuto dell'accumulatore 1-L con il contenuto dell'accumulatore 2-L, oppure con una costante (a 16 bit), bit per bit, secondo l'operazione logica combinatoria OR esclusivo. Il bit della parola di risultato è "1" se almeno uno dei corrispondenti bit delle parole da combinare è "1". Il risultato viene memorizzato nell'accumulatore 1-L. Gli accumulatori 1-L e 2 (e per le CPU con quattro accumulatori ACCU 3 e ACCU 4) rimangono inalterati. Il bit di stato A1 viene settato conformemente all'esito dell'operazione (A1 = 1, se il risultato è diverso da zero). I bit di stato A0 e OV vengono resettati a "0".

È possibile utilizzare la funzione OR esclusivo anche più volte di seguito. Il risultato logico combinatorio condiviso sarà pertanto "1", se un numero dispari di operandi interrogati dà come risultato dell'interrogazione "1".

XOW: combina il contenuto degli accumulatori 1-L e 2-L.

XOW <costante>: combina il contenuto dell'accumulatore 1-L ad una costante (a 16 bit).

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	x	0	0	-	-	-	-	-

Esempi

Bit	**15 . .**	**. .**	**. .**	**. . . 0**
ACCU 1 prima dell'esecuzione di **XOW**	0101	0101	0011	1011
ACCU 2-L o costante (a 16 bit):	1111	0110	1011	0101
Risultato (ACCU 1) dopo l'esecuzione di **XOW**	1010	0011	1000	1110

Esempio 1

```
AWL             Spiegazione
L     EW20      //Carica il contenuto di EW20 in ACCU 1-L.
L     EW22      //Carica il contenuto di ACCU 1 in ACCU 2. Carica il contenuto di ED24 in ACCU 1.
XOW             //Combina i bit di ACCU 1-L con i bit di ACCU 2-L mediante OR esclusivo, memorizza
                //il risultato in ACCU 1-L.
T     MW8       //Trasferisce il risultato a MW8.
```

Esempio 2

```
AWL             Spiegazione
L     EW20      //Carica il contenuto di EW20 in ACCU 1-L.
XOW   16#0FFF   //Combina i bit di ACCU 1-L con la configurazione binaria della costante
                //(a 16 bit) (0000_1111_1111_1111) mediante OR esclusivo, memorizza il risultato
                //in ACCU 1-L.
SPP   NEXT      //Passa all'etichetta di salto NEXT; se il risultato è diverso da zero (A1 = 1).
```

13.5 UD AND a doppia parola (a 32 bit)

Formato

UD
UD <costante>

Operando	Tipo dati	Descrizione
<costante>	DWORD, costante (a 32 bit)	Configurazione binaria da combinare a ACCU 1-L mediante AND

Descrizione dell'operazione

UD (AND a doppia parola) combina il contenuto dell'accumulatore 1 con il contenuto dell'accumulatore 2 oppure con una costante (a 32 bit), bit per bit, in conformità all'operazione logica combinatoria AND. Il bit della doppia parola di risultato è "1" solamente se i corrispondenti bit delle doppie parole combinate sono entrambi "1". Il risultato viene memorizzato nell'accumulatore 1. ACCU 2 (e per le CPU con quattro accumulatori ACCU 3 e ACCU 4) rimane inalterato. Il bit di stato A1 viene impostato conformemente all'esito dell'operazione (A1=1 se il risultato è diverso da zero). I bit di stato A0 e OV vengono resettati a "0".

UD: combina i contenuti degli accumulatori 1 e 2.

UD <costante>: combina il contenuto dell'accumulatore 1 con una costante (a 32 bit).

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	x	0	0	-	-	-	-	-

Esempi

Bit	**31 . .**	**. .**	**. .**	**. .**	**. .**	**. .**	**. .**	**. . . 0**
ACCU 1 prima dell'esecuzione di **UD**	0101	0000	1111	1100	1000	1001	0011	1011
ACCU 2 o costante (a 32 bit):	1111	0011	1000	0101	0111	0110	1011	0101
Risultato (ACCU 1) dopo l'esecuzione di **UD**	0101	0000	1000	0100	0000	0000	0011	0001

Esempio 1

```
AWL                 Spiegazione
L       ED20        //Carica il contenuto di ED20 in ACCU 1.
L       ED24        //Carica il contenuto di ACCU 1 in ACCU 2. Carica il contenuto di ED24 in
                    //ACCU 1.
UD                  //Combina i bit di ACCU 1 con i bit di ACCU 2 avvalendosi dell'operazione
                    //logica combinatoria AND, memorizza il risultato in ACCU 1.
T       MD8         //Trasferisce il risultato a MD8.
```

Esempio 2

```
AWL                 Spiegazione
L       ED 20       //Carica il contenuto di ED20 in ACCU 1.
UD      DW#16#0FFF_ //Combina i bit di ACCU 1 con la configurazione binaria della costante
        EF21        //(a 32 bit) (0000_1111_1111_1111_1110_1111_0010_0001) mediante AND,
                    //memorizza il risultato in ACCU 1.
SPP     NEXT        //Passa all'etichetta di salto NEXT se il risultato è diverso da zero
                    //(A1 = 1).
```

13.6 OD OR a doppia parola (a 32 bit)

Formati

OD
OD <costante>

Operando	Tipo dati	Descrizione
<costante>	DWORD, costante (a 32 bit)	Configurazione binaria da combinare con ACCU 1-L mediante OR

Descrizione dell'operazione

OD (OR doppia parola) combina il contenuto dell'accumulatore 1 con il contenuto dell'accumulatore 2, oppure con una costante (a 32 bit), bit per bit, in conformità all'operazione logica combinatoria OR. Il bit della doppia parola di risultato è "1" solamente se i corrispondenti bit delle doppie parole combinate sono entrambi "1". Il risultato viene memorizzato nell'accumulatore 1. ACCU 2 (per le CPU con quattro accumulatori anche ACCU 3 e ACCU 4) rimane inalterato. Il bit di stato A1 viene impostato conformemente all'esito dell'operazione (A1=1 se il risultato è diverso da zero). I bit di stato A0 e OV vengono resettati a "0".

OD: combina il contenuto dell'accumulatore 1 con il contenuto dell'accumulatore 2.

OD <costante>: combina il contenuto dell'accumulatore 1 con una costante (a 32 bit).

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	x	0	0	-	-	-	-	-

Esempi

Bit	**31 . .**	**. .**	**. .**	**. .**	**. .**	**. .**	**. .**	**. . . 0**
ACCU 1 prima dell'esecuzione di **OD**	0101	0000	1111	1100	1000	0101	0011	1011
ACCU 2 o costante (a 32 bit):	1111	0011	1000	0101	0111	0110	1011	0101
Risultato (ACCU 1) dopo l'esecuzione di **OD**	1111	0011	1111	1101	1111	0111	1011	1111

Esempio 1

```
AWL                     Spiegazione
L     ED20              //Carica il contenuto di ED20 in ACCU 1.
L     ED24              //Carica il contenuto di ACCU 1 in ACCU 2. Carica il contenuto di ED24
                        //in ACCU 1.
OD                      //Combina i bit di ACCU 1 con i bit di ACCU 2 mediante OR, memorizza
                        //il risultato in ACCU 1.
T     MD8               //Trasferisce il risultato a MD8.
```

Esempio 2

```
AWL                     Spiegazione
L     ED20              //Carica il contenuto di ED20 in ACCU 1.
OD    DW#16#0FFF_EF21   //Combina i bit di ACCU 1 con la configurazione binaria della costante
                        //(a 32 bit) (0000_1111_1111_1111_1110_1111_0010_0001) mediante OR,
                        //memorizza il risultato in ACCU 1.
SPP   NEXT              //Passa all'etichetta di salto NEXT, se il risultato è diverso da zero
                        //(A1 = 1).
```

13.7 XOD OR esclusivo a doppia parola (a 32 bit)

Formati

XOD
XOD <costante>

Operando	Tipo dati	Descrizione
<costante>	DWORD, costante (a 32 bit)	Configurazione binaria da combinare con ACCU 1 mediante OR esclusivo.

Descrizione dell'operazione

XOD (OR esclusivo a doppia parola) combina il contenuto dell'accumulatore 1 con il contenuto dell'accumulatore 2, oppure con una costante (a 32 bit), bit per bit, secondo l'operazione logica combinatoria OR esclusivo. Il bit della doppia parola di risultato è "1" se almeno uno dei corrispondenti bit delle doppie parole da combinare è "1". Il risultato viene memorizzato nell'accumulatore 1. ACCU 2 (e per le CPU con quattro accumulatori ACCU 3 e ACCU 4) rimane inalterato. Il bit di stato A1 viene settato conformemente all'esito dell'operazione (A1 = 1, se il risultato è diverso da zero). I bit di stato A0 e OV vengono resettati a "0".

È possibile utilizzare la funzione OR esclusivo anche più volte di seguito. Il risultato logico combinatorio condiviso sarà pertanto "1", se un numero dispari di operandi interrogati dà come risultato dell'interrogazione "1".

XOD: combina i contenuti degli accumulatori 1 e 2.

XOD <costante>: combina il contenuto dell'accumulatore 1 ad una costante (a 32 bit).

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	x	0	0	-	-	-	-	-

Esempi

Bit	**31 . .**	**. .**	**. .**	**. .**	**. .**	**. .**	**. .**	**. . . 0**
ACCU 1 prima dell'esecuzione di **XOD**	0101	0000	1111	1100	1000	0101	0011	1011
ACCU 2 o costante (a 32 bit):	1111	0011	1000	0101	0111	0110	1011	0101
Risultato (ACCU 1) dopo l'esecuzione di **XOD**	1010	0011	0111	1001	1111	0011	1000	1110

Esempio 1

```
AWL                     Spiegazione
L      ED20             //Carica il contenuto di ED20 in ACCU 1.
L      ED24             //Carica il contenuto di ACCU 1 in ACCU 2. Carica il contenuto di ED24
                        //in ACCU 1.
XOD                     //Combina i bit di ACCU 1 con i bit di ACCU 2 mediante OR esclusivo,
                        //memorizza il risultato in ACCU 1.
T      MD8              //Trasferisce il risultato a MD8.
```

Esempio 2

```
AWL                     Spiegazione
L      ED20             //Carica il contenuto di ED20 in ACCU 1.
XOD    DW#16#0FFF_EF21  //Combina i bit di ACCU 1 con la configurazione binaria della costante
                        //(a 32 bit) (0000_1111_1111_1111_1111_1110_0010_0001) mediante OR,
                        //memorizza il risultato in ACCU 1.
SPP    NEXT             //Passa all'etichetta di salto NEXT, se il risultato è diverso da zero
                        //(A1   = 1).
```

14 Operazioni per il funzionamento dell'accumulatore

14.1 Sommario delle operazioni per il funzionamento degli accumulatori e istruzioni del registro d'indirizzo

Descrizione

Le seguenti operazioni possono essere adoperate dall'utente per elaborare il contenuto di uno o più accumulatori oppure dei registri d'indirizzo:

- TAK Scambio di accumulatore 1 con accumulatore 2
- PUSH CPU con due accumulatori
- PUSH CPU con quattro accumulatori
- POP CPU con due accumulatori
- POP CPU con quattro accumulatori

- ENT Immissione stack accumulatore
- LEAVE Uscire da stack accumulatore
- INC Incremento di accumulatore 1
- DEC Decremento di accumulatore 1

- +AR1 Sommare ACCU 1 al registro di indirizzi 1
- +AR2 Sommare ACCU 1 al registro di indirizzi 2
- BLD Comando di visualizzazione del programma
- NOP 0 Nessuna operazione 0
- NOP 1 Nessuna operazione 1
-
- Vedere anche
- TAW Cambia sequenza di byte in ACCU 1 (a 16 bit)
- TAD Cambia sequenza di byte in ACCU 1 (a 32 bit)

14.2 TAK Scambia ACCU 1 con ACCU 2

Formato

TAK

Descrizione dell'operazione

TAK (Scambia ACCU 1 con ACCU 2) scambia il contenuto di ACCU 1 con quello di ACCU 2. L'operazione viene eseguita senza considerare o influenzare i bit di stato. Gli accumulatori 3 e 4 rimangono inalterati (per CPU con quattro accumulatori).

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio: Sottrarre il valore inferiore dal valore maggiore

```
AWL                 Spiegazione
       L     MW10   //Carica il contenuto di MW10 in ACCU 1-L.
       L     MW12   //Carica il contenuto di ACCU 1-L in ACCU 2-L. Carica il contenuto di MW12
                    in ACCU 1-L.
       >I           //Controlla se ACCU 2-L (MW10) è maggiore di ACCU  1-L (MW12).
       SPB   NEXT   //Passa all'etichetta di salto NEXT, se ACCU 2 (MW10) è maggiore di ACCU 1
                    (MW12).
       TAK          //Scambia i contenuti degli accumulatori 1 e 2.
NEXT:  -I           //Sottrae il contenuto di ACCU 1-L dal contenuto di ACCU 2-L.
       T     MW14   //Trasferisce il risultato (= il valore maggiore meno il valore minore) a
                    MW14.
```

Contenuto degli accumulatori 1 e 2

Contenuto	ACCU 1	ACCU 2
Prima dell'esecuzione di **TAK**	<MW12>	<MW10>
Dopo l'esecuzione di **TAK**	<MW**10**>	<MW**12**>

14.3 PUSH CPU con due accumulatori

Formato

PUSH

Descrizione dell'operazione

PUSH (accumulatore 1 nell'accumulatore 2) copia l'intero contenuto di ACCU 1 in ACCU 2. ACCU 1 resta inalterato. L'operazione viene eseguita senza considerare o influenzare i bit di stato.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio

```
AWL             Spiegazione
L     MW10      //Carica il contenuto di MW10 nell'accumulatore 1.
PUSH            //Copia l'intero contenuto di ACCU 1 nell'accumulatore 2.
```

Contenuto degli accumulatori 1 e 2

Contenuto	ACCU 1	ACCU 2
Prima dell'esecuzione di **PUSH**	<MW10>	<X>
Dopo l'esecuzione di **PUSH**	<MW10>	<**MW10**>

14.4 PUSH CPU con quattro accumulatori

Formato

PUSH

Descrizione dell'operazione

PUSH (CPU con quattro accumulatori) copia il contenuto di ACCU 3 in ACCU 4, il contenuto di ACCU 2 in ACCU 3; ed il contenuto di ACCU 1 in ACCU 2. ACCU 1 resta inalterato. L'operazione viene eseguita senza considerare o influenzare i bit di stato.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio

```
AWL             Spiegazione
C      MW10     //Carica il contenuto di MW10 in ACCU 1.
PUSH            //Copia l'intero contenuto di ACCU 1 in ACCU 2, il contenuto di ACCU 2 in ACCU 3,
                //ed il contenuto di ACCU 3 in ACCU 4.
```

Contenuto degli accumulatori da 1 a 4

Contenuto	ACCU 1	ACCU 2	ACCU 3	ACCU 4
Prima dell'esecuzione di **PUSH**	valore A	valore B	valore C	valore D
Dopo l'esecuzione di **PUSH**	valore A	valore A	valore B	valore C

14.5 POP CPU con due accumulatori

Formato

POP

Descrizione dell'operazione

POP (CPU con due accumulatori) copia l'intero contenuto di ACCU 2 in ACCU 1. ACCU 2 resta inalterato. L'operazione viene eseguita senza considerare o influenzare i bit di stato.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio

```
AWL             Spiegazione
T     MD10      //Trasferisce il contenuto da ACCU 1 (= valore A) a MD10.
POP             //Copia l'intero contenuto di ACCU 2 in ACCU 1.
T     MD14      //Trasferisce il contenuto di ACCU 1 (= valore B) a MD14.
```

Contenuto degli accumulatori 1 e 2

Contenuto	ACCU 1	ACCU 2
Prima dell'esecuzione di **POP**	valore A	valore B
Dopo l'esecuzione di **POP**	valore B	valore B

14.6 POP CPU con quattro accumulatori

Formato

POP

Descrizione dell'operazione

POP (CPU con quattro accumulatori) copia il contenuto di ACCU 2 in ACCU 1, di ACCU 3 in ACCU 2, e di ACCU 4 in ACCU 3. ACCU 4 resta inalterato. L'operazione viene eseguita senza considerare o influenzare i bit di stato.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio

```
AWL             Spiegazione
T       MD10    //Trasferisce il contenuto da ACCU 1 (= valore A) a MD10.
POP             //Copia l'intero contenuto di ACCU 2 in ACCU 1, di ACCU 3 in ACCU 2, e di ACCU 4
                //in ACCU 3.
T       MD14    //Trasferisce il contenuto di ACCU 1 (= valore B) a MD14.
```

Contenuto degli accumulatori da 1 a 4

Contenuto	ACCU 1	ACCU 2	ACCU 3	ACCU 4
Prima dell'esecuzione di **POP**	valore A	valore B	valore C	valore D
Dopo l'esecuzione di **POP**	valore B	valore C	valore D	valore D

14.7 ENT Immetti stack accumulatore

Formato

ENT

Descrizione dell'operazione

ENT (Immetti stack accumulatore) copia il contenuto di ACCU 3 in ACCU 4, ed il contenuto di ACCU 2 in ACCU 3. Se l'utente programma l'operazione ENT direttamente prima di una operazione di caricamento, si può salvare un risultato intermedio in ACCU 3.

Esempio

```
AWL             Spiegazione
L     DBD0      //Carica in ACCU 1 il valore della doppia parola dati DBD0. (Tale valore deve
                //avere formato di numero in virgola mobile.)
L     DBD4      //Copia in ACCU 2 il valore di ACCU 1. Carica in ACCU 1 il valore della doppia
                //parola dati DBD4. (Tale valore deve avere formato di numero in virgola mobile.)
+R              //Somma i contenuti di ACCU 1 e 2 come numeri in virgola mobile (IEEE 754,
                //a 32 bit), e memorizza il risultato in ACCU 1.
L     DBD8      //Copia in ACCU 2 il valore di ACCU 1. Carica in ACCU 1 il valore della doppia
                //parola dati DBD8.
ENT             //Copia il contenuto di ACCU 3 in ACCU 4. Copia il contenuto di ACCU 2 (risultato
                //intermedio) in ACCU 3.
L     DBD12     //Carica in ACCU 1 il valore della doppia parola dati DBD12.
-R              //Sottrae il contenuto di ACCU 1 dal contenuto di ACCU 2, e memorizza il risultato
                //in ACCU 1. Copia il contenuto di ACCU 3 in ACCU 2; ed il contenuto di ACCU 4
                //in ACCU 3.
/R              //Divide il contenuto di ACCU 2 (DBD0 + DBD4) per il contenuto di ACCU 1
                //(DBD8 - DBD12), e memorizza il risultato in ACCU 1.
T     DBD16     //Transferisce il risultato (ACCU 1) nella doppia parola dati DBD16
```

14.8 LEAVE Esci da stack accumulatore

Formato

LEAVE

Descrizione dell'operazione

LEAVE (Esci da stack accumulatore) copia il contenuto di ACCU 3 in ACCU 2, il contenuto di ACCU 4 in ACCU 3. Se l'utente programma la operazione LEAVE direttamente prima di una operazione di scorrimento e rotazione, che combina gli accumulatori, l'operazione LEAVE funziona come una operazione matematica. Rimangono invariati i contenuti di ACCU 1 e ACCU 4.

14.9 DEC Decrementa ACCU 1

Formato

DEC <numero intero, 8 bit>

Operando	Tipo dati	Descrizione
<numero intero, a 8 bit >	Costante (numero intero, 8 bit)	Costante che viene sottratta da ACCU1-L-L; campo da 0 a 255

Descrizione dell'operazione

DEC <numero intero, a 8 bit> (Decrementa ACCU 1-L-L) sottrae un numero intero (a 8 bit) dal contenuto di ACCU 1-L-L, e memorizza il risultato in ACCU 1-L-L. Gli accumulatori 1-L-H, 1-H, e 2 rimangono inalterati. L'operazione viene eseguita senza considerare o influenzare i bit di stato.

Avvertenza

Le suddette operazioni non sono adatte alla matematica a 16 bit o a 32 bit, in quanto non viene effettuato nessun riporto dal byte basso della parola bassa di ACCU 1 al byte alto della parola bassa di ACCU 1. Per la matematica a 16 bit o a 32 bit si deve usare rispettivamente l'operazione +I o l'operazione +D.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio

```
AWL             Spiegazione
L     MB250     //Carica il valore di MB250.
DEC   1         //Decrementa ACCU 1-L-L di 1, memorizza il risultato in ACCU 1-L-L.
T     MB250     //Trasferisce nuovamente il contenuto di ACCU 1-L-L (risultato) a MB250.
```

14.10 INC Incrementa ACCU 1

Formato

INC <numero intero, a 8 bit>

Operando	Tipo dati	Descrizione
<numero intero, a 8 bit >	Costante (numero intero, 8 bit)	Costante che viene sumata a ACCU1-L-L; campo da 0 a 255

Descrizione dell'operazione

INC <numero intero, a 8 bit> (Incrementa ACCU 1-L-L) addiziona un numero intero (a 8 bit) al contenuto di ACCU 1-L-L, e memorizza il risultato in ACCU 1-L-L. Gli accumulatori1-L-H, 1-H, e 2 restano inalterati. L'operazione viene eseguita senza considerare o influenzare i bit di stato.

Avvertenza

Le suddette operazioni non sono adatte alla matematica a 16 bit o a 32 bit, in quanto non viene effettuato nessun riporto dal byte basso della parola bassa di ACCU 1 al byte alto della parola bassa di ACCU 1. Per la matematica a 16 bit o a 32 bit si deve usare rispettivamente l'operazione +I o l'operazione +D.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio

```
AWL             Spiegazione
L     MB22      //Carica il valore di MB22.
INC   1         //Incrementa ACCU 1 (MB 22) di 1, memorizza il risultato in ACCU 1-L-L.
T     MB22      //Trasferisce nuovamente il contenuto di ACCU 1-L-L (risultato) a MB22.
```

14.11 +AR1 Somma ACCU 1 al registro di indirizzo 1

Formati

+AR1
+AR1 <P#Byte.Bit>

Operando	Tipo dati	Descrizione
<P#Byte.Bit>	Costante puntatore	Indirizzo che viene sommato a AR 1.

Descrizione dell'operazione

+AR1 (Somma ACCU 1 a AR1) aggiunge al contenuto di AR1 uno spostamento, indicato nell'istruzione od in ACCU 1-L. Il numero intero (a 16 bit) viene dapprima esteso a 24 bit col segno appropriato, poi aggiunto ai 24 bit bassi di AR1 (parte dell'indirizzo di AR1). Non viene modificata la parte della identificazione di area di AR1 (bit 24, 25, e 26). L'operazione viene eseguita senza considerare o influenzare i bit di stato.

+AR1: il numero intero (a 16 bit) che deve essere aggiunto al contenuto di AR1 viene indicato col valore di ACCU 1-L. Sono ammessi i valori da -32768 a +32767.

+AR1 <P#Byte.Bit>: lo spostamento che deve essere aggiunto viene indicato con l'operando <P#Byte.Bit>.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio 1

```
AWL              Spiegazione
L      +300      //Carica il valore in ACCU 1-L.
+AR1             //Somma ACCU 1-L (numero intero, a 16 bit) a AR 1.
```

Esempio 2

```
AWL              Spiegazione
+AR1   P#300.0   //Somma lo spostamento 300.0 a AR 1.
```

14.12 +AR2 Somma ACCU 1 al registro di indirizzo 2

Formati

+AR2
+AR2 <P#Byte.Bit>

Operando	Tipo dati	Descrizione
<P#Byte.Bit>	Costante puntatore	Indirizzo che viene aggiunto a AR2.

Descrizione dell'operazione

+AR2 (Somma ACCU 1 a AR2) aggiunge al contenuto di AR1 uno spostamento, indicato nell'istruzione od in ACCU 1-L. Il numero intero (a 16 bit) viene dapprima esteso a 24 bit col segno appropriato, poi aggiunto ai 24 bit bassi di AR2 (parte dell'indirizzo di AR2). Non viene modificata la parte della identificazione di area di AR2 (bit 24, 25, e 26). L'operazione viene eseguita senza considerare o influenzare i bit di stato.

+AR2: il numero intero (a 16 bit) che deve essere aggiunto al contenuto di AR1 viene indicato col valore di ACCU 1-L. Sono ammessi i valori da -32768 a +32767.

+AR2 <P#Byte.Bit>: lo spostamento che deve essere aggiunto viene indicato con l'operando **<P#Byte.Bit>**.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

Esempio 1

```
AWL             Spiegazione
L     +300      //Carica il valore in ACCU 1-L.
+AR2            //Aggiunge ACCU 1-L (numero intero, a 16 bit) a AR 2.
```

Esempio 2

```
AWL             Spiegazione
+AR2  P#300.0   //Aggiunge lo spostamento 300.0 a AR 2.
```

14.13 BLD Comando di visualizzazione del programma (NOP)

Formato

BLD <numero>

Operando	Descrizione
<numero>	Numero di identificazione dell'operazione BLD; area da 0 a 255

Descrizione dell'operazione

BLD <numero> (comando di visualizzazione del programma, comando di nessuna operazione) non esegue alcuna funzione e non influenza i bit di stato. L'operazione serve soltanto per i dispositivi di programmazione (PG), per la rappresentazione grafica dell'immagine. Essa viene creata automaticamente quando viene visualizzato in AWL un programma KOP o FUP. L'operando <numero> rappresenta il numero di identificazione dell'operazione BLD, e viene generato dal dispositivo di programmazione.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

14.14 NOP 0 Nessuna operazione 0

Formato

NOP 0

Descrizione dell'operazione

NOP 0 (operazione NOP con operando "0") non esegue alcuna funzione e non influisce sui bit di stato. L'identificazione dell'operazione è data da un modello binario a 16 zeri. L'operazione assume rilievo per il dispositivo di programmazione (PG) soltanto se viene visualizzato un programma.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

14.15 NOP 1 Nessuna operazione 1

Formato

NOP 1

Descrizione dell'operazione

NOP 1 (operazione NOP con operando "1") non esegue alcuna funzione e non influisce sui bit di stato. L'identificazione dell'operazione è data da un modello binario a 16 uno. L'operazione assume rilievo per il dispositivo di programmazione (PG) soltanto se viene visualizzato un programma.

Parola di stato

	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER
Scrive:	-	-	-	-	-	-	-	-	-

A Sommario di tutte le operazioni AWL

A.1 Operazioni AWL ordinate secondo il set mnemonico tedesco (SIMATIC)

Mnemonico tedesco	Mnemonico inglese	Catalogo elementi del programma	Descrizione
=	=	Operazione logica combinatoria a bit	Assegnazione
)	)	Operazione logica combinatoria a bit	Chiusura parentesi
*D	*D	Funzione in virgola fissa	Moltiplica ACCU 1 per ACCU 2 come numeri interi (a 32 bit)
*I	*I	Funzione in virgola fissa	Moltiplica ACCU 1 per ACCU 2 come numeri interi (a 16 bit)
*R	*R	Funzione in virgola mobile	Moltiplica ACCU 1 per ACCU 2 come numeri in virgola mobile (a 32 bit, IEEE 754)
/D	/D	Funzione in virgola fissa	Dividi ACCU 2 per ACCU 1 come numeri interi (a 32 bit)
/I	/I	Funzione in virgola fissa	Dividi ACCU 2 per ACCU 1 come numeri interi (a 16 bit)
/R	/R	Funzione in virgola mobile	Dividi ACCU 2 per ACCU 1 come numeri in virgola mobile (a 32 bit, IEEE 754)
? D	? D	Comparatori	Confronta numeri interi (a 32 bit)
? I	? I	Comparatori	Confronta numeri interi (a 16 bit)
? R	? R	Comparatori	Confronta numeri in virgola mobile (a 32 bit)
+	+	Funzione in virgola fissa	Somma costante di numero intero (a 16, 32 bit)
+AR1	+AR1	Accumulatori	Somma accumulatore 1 al registro di indirizzo 1
+AR2	+AR2	Accumulatori	Somma accumulatore 1 al registro di indirizzo 2
+D	+D	Funzione in virgola fissa	Somma ACCU 1 e ACCU 2 come numeri interi (a 32 bit)
+I	+I	Funzione in virgola fissa	Somma ACCU 1 e ACCU 2 come numeri interi (a 16 bit)
+R	+R	Funzione in virgola mobile	Somma ACCU 1 e ACCU 2 come numeri in virgola mobile (a 32 bit, IEEE 754)
ABS	ABS	Funzione in virgola mobile	Valore assoluto di un numero in virgola mobile (a 32 bit, IEEE 754)
ACOS	ACOS	Funzione in virgola mobile	Formazione dell'arcoseno di un numero in virgola mobile (a 32 bit)
ASIN	ASIN	Funzione in virgola mobile	Formazione dell'arcoseno di un numero in virgola mobile (a 32 bit)
ATAN	ATAN	Funzione in virgola mobile	Formazione dell'arcotangente di un numero in virgola mobile (a 32 bit)
AUF	OPN	Blocco dati	Apri blocco dati
BE	BE	Comando del programma	Fine blocco
BEA	BEU	Comando del programma	Fine blocco assoluto
BEB	BEC	Comando del programma	Fine blocco condizionato
BLD	BLD	Accumulatori	Comando di visualizzazione del programma (NOP)
BTD	BTD	Convertitori	Converti numero BCD in numero intero (a 32 bit)

Mnemonico tedesco	Mnemonico inglese	Catalogo elementi del programma	Descrizione
BTI	BTI	Convertitori	Converti numero BCD in numero intero (a 16 bit)
CALL	CALL	Comando del programma	Richiamo di blocco
CALL	CALL	Comando del programma	Richiamo di una multi-istanza
CALL	CALL	Comando del programma	Richiamo di blocchi da una biblioteca
CC	CC	Comando del programma	Richiamo condizionato di un blocco
CLR	CLR	Operazione logica combinatoria a bit	Resetta RLC (=0)
COS	COS	Funzione in virgola mobile	Formazione del coseno di un angolo come numero in virgola mobile (a 32 bit)
–D	–D	Funzione in virgola fissa	Sottrai ACCU 1 da ACCU 2 come numeri interi (a 32 bit)
DEC	DEC	Accumulatori	Decrementa accumulatore 1
DTB	DTB	Convertitori	Converti numero intero (a 32 bit) in un numero BCD
DTR	DTR	Convertitori	Converti numero intero (a 32 bit) in numero in virgola mobile (a 32 bit, IEEE 754)
ENT	ENT	Accumulatori	Immetti stack accumulatore
EXP	EXP	Funzione in virgola mobile	Formazione del valore esponenziale di un numero in virgola mobile (a 32 bit)
FN	FN	Operazione logica combinatoria a parola	Fronte di discesa
FP	FP	Operazione logica combinatoria a parola	Fronte di salita
FR	FR	Contatori	Abilita contatore
FR	FR	Temporizzatori	Abilita temporizzatore
–I	–I	Funzione in virgola fissa	Sottrai ACCU 1 da ACCU 2 come numeri interi (a 16 bit)
INC	INC	Accumulatori	Incrementa accumulatore 1
INVD	INVD	Convertitori	Complemento a 1 di numero intero (a 32 bit)
INVI	INVI	Convertitori	Complemento a 1 di numero intero (a 16 bit)
ITB	ITB	Convertitori	Converti numero intero (a 16 bit) in numero BCD
ITD	ITD	Convertitori	Converti numero intero (a 16 bit) in numero intero (a 32 bit)
L	L	Caricamento/Trasferimento	Carica
L STW	L STW	Caricamento/Trasferimento	Carica parola di stato in ACCU 1
L	L	Temporizzatori	Carica valore attuale di conteggio in ACCU 1 come numero intero (il valore attuale di conteggio può essere un numero compreso tra 0 e 255, per esempio: L T 32)
L	L	Contatori	Carica valore attuale di conteggio in ACCU 1 come numero intero (il valore di conteggio attuale può essere un numero compreso tra 0 e 255, per esempio: L Z 15)
L DBLG	L DBLG	Blocco dati	Carica lunghezza del DB globale in ACCU 1
L DBNO	L DBNO	Blocco dati	Carica numero del DB globale in ACCU 1
L DILG	L DILG	Blocco dati	Carica lunghezza del DB di istanza in ACCU 1
L DINO	L DINO	Blocco dati	Carica numero del DB di istanza in ACCU 1
LAR1	LAR1	Caricamento/Trasferimento	Carica il registro di indirizzo 1con il contenuto di ACCU 1
LAR1	LAR1	Caricamento/Trasferimento	Carica il registro di indirizzo 1 con numero intero a 32 bit
LAR1	LAR1	Caricamento/Trasferimento	Carica il registro di indirizzo 1 con il contenuto del registro di indirizzo 2
LAR2	LAR2	Caricamento/Trasferimento	Carica il registro di indirizzo 2 con il contenuto di ACCU 1

Mnemonico tedesco	Mnemonico inglese	Catalogo elementi del programma	Descrizione
LAR2	LAR2	Caricamento/Trasferimento	Carica il registro di indirizzo 2 con numero intero (a 32 Bit)
LC	LC	Contatori	Carica valore attuale di conteggio in ACCU 1 come BCD (il valore attuale di conteggio può essere un numero compreso tra 0 e 255, per esempio: LC Z 15)
LC	LC	Temporizzatori	Carica valore attuale di conteggio in ACCU 1 come BCD (il valore attuale di conteggio può essere un numero compreso tra 0 e 255, per esempio: LC T 32)
LEAVE	LEAVE	Accumulatori	Esci da stack accumulatore
LN	LN	Funzione in virgola mobile	Formazione del logaritmo naturale di un numero in virgola mobile (a 32 bit)
LOOP	LOOP	Salti	Loop di programma
MCR(	MCR(	Comando del programma	Salva RLC nello stack MCR, inizio zona MCR
)MCR	)MCR	Comando del programma	Fine zona MCR
MCRA	MCRA	Comando del programma	Attiva zona MCR
MCRD	MCRD	Comando del programma	Disattiva zona MCR
MOD	MOD	Funzione in virgola fissa	Resto della divisione di numero intero (a 32 bit)
NEGD	NEGD	Convertitori	Complemento a 2 di numero intero (a 32 bit)
NEGI	NEGI	Convertitori	Complemento a 2 di numero intero (a 16 bit)
NEGR	NEGR	Convertitori	Complemento a 2 di numero in virgola mobile (a 32 bit, IEEE 754)
NOP 0	NOP 0	Accumulatori	Nessuna operazione 0
NOP 1	NOP 1	Accumulatori	Nessuna operazione 1
NOT	NOT	Operazione logica combinatoria a bit	Nega RLC
O	O	Operazione logica combinatoria a bit	OR
O(	O(	Operazione logica combinatoria a bit	OR con apertura parentesi
OD	OD	Operazione logica combinatoria a parola	OR a doppia parola (a 32 bit)
ON	ON	Operazione logica combinatoria a bit	OR negato
ON(	ON(	Operazione logica combinatoria a bit	OR negato con apertura parentesi
OW	OW	Operazione logica combinatoria a parola	OR a parola (a 16 bit)
POP	POP	Accumulatori	POP CPU con due accumulatori
POP	POP	Accumulatori	POP CPU con quattro accumulatori
PUSH	PUSH	Accumulatori	PUSH CPU con due accumulatori
PUSH	PUSH	Accumulatori	PUSH CPU con quattro accumulatori
R	R	Operazione logica combinatoria a bit	Resetta
R	R	Contatori	Resetta contatore (il contatore attuale può essere un numero compreso tra 0 e 255, per esempio: R Z 15)
R	R	Temporizzatori	Resetta temporizzatore (il temporizzatore attuale può essere un numero compreso tra 0 e 255, per esempio: R T 32)
–R	–R	Funzione in virgola mobile	Sottrai ACCU 1 da ACCU 2 come numeri in virgola mobile (a 32 bit, IEEE 754)

Mnemonico tedesco	Mnemonico inglese	Catalogo elementi del programma	Descrizione
RLD	RLD	Scorrimento/rotazione	Fai ruotare doppia parola a sinistra (a 32 bit)
RLDA	RLDA	Scorrimento/rotazione	Fai ruotare ACCU 1 a sinistra tramite A1 (a 32 bit)
RND	RND	Convertitori	Arrotonda al numero intero
RND–	RND–	Convertitori	Arrotonda al numero intero inferiore (a 32 bit)
RND+	RND+	Convertitori	Arrotonda al numero intero superiore (a 32 bit)
RRD	RRD	Scorrimento/rotazione	Fai ruotare doppia parola a destra (a 32 bit)
RRDA	RRDA	Scorrimento/rotazione	Fai ruotare ACCU 1 a destra tramite A1 (a 32 bit)
S	S	Operazione logica combinatoria a bit	Imposta
S	S	Contatori	Imposta valore iniziale di conteggio (il contatore attuale può essere un numero compreso tra 0 e 255, per esempio: S Z 15)
SA	SF	Temporizzatori	Avvia temporizzatore come ritardo alla disinserzione
SAVE	SAVE	Operazione logica combinatoria a bit	Salva RLC nel registro BIE
SE	SD	Temporizzatori	Avvia temporizzatore come ritardo all'inserzione
SET	SET	Operazione logica combinatoria a bit	Imposta RLC (=1)
SI	SP	Temporizzatori	Avvia temporizzatore come impulso
SIN	SIN	Funzione in virgola mobile	Formazione del seno di un angolo come numero in virgola mobile (a 32 bit)
SLD	SLD	Scorrimento/rotazione	Fai scorrere doppia parola a sinistra (a 32 bit)
SLW	SLW	Scorrimento/rotazione	Fai scorrere parola a sinistra (a 16 bit)
SPA	JU	Salti	Salto assoluto
SPB	JC	Salti	Salta se RLC = 1
SPBB	JCB	Salti	Salta se RLC = 1 con BIE
SPBI	JBI	Salti	Salta se BIE = 1
SPBIN	JNBI	Salti	Salta se BIE = 0
SPBN	JCN	Salti	Salta se RLC = 0
SPBNB	JNB	Salti	Salta se RLC = 0 con BIE
SPL	JL	Salti	Salta alle etichette
SPM	JM	Salti	Salta se risultato < 0
SPMZ	JMZ	Salti	Salta se risultato <= 0
SPN	JN	Salti	Salta se risultato diverso da zero
SPO	JO	Salti	Salta se OV = 1
SPP	JP	Salti	Salta se risultato > 0
SPPZ	JPZ	Salti	Salta se risultato >= 0
SPS	JOS	Salti	Salta se OS = 1
SPU	JUO	Salti	Salta se operazione non ammessa
SPZ	JZ	Salti	Salta se risultato = 0
SQR	SQR	Funzione in virgola mobile	Formazione del quadrato di un numero in virgola mobile (a 32 bit)
SQRT	SQRT	Funzione in virgola mobile	Formazione della radice quadrata di un numero in virgola mobile (a 32 bit)
SRD	SRD	Scorrimento/rotazione	Fai scorrere doppia parola a destra (a 32 bit)
SRW	SRW	Scorrimento/rotazione	Fai scorrere parola a destra (a 16 bit)
SS	SS	Temporizzatori	Avvia temporizzatore come ritardo all’inserzione con memoria
SSD	SSD	Scorrimento/rotazione	Fai scorrere numero intero con segno (a 32 bit)

Mnemonico tedesco	Mnemonico inglese	Catalogo elementi del programma	Descrizione
SSI	SSI	Scorrimento/rotazione	Fai scorrere numero intero con segno (a 16 bit)
SV	SE	Temporizzatori	Avvia temporizzatore come impulso prolungato
T	T	Caricamento/Trasferimento	Trasferisci
T STW	T STW	Caricamento/Trasferimento	Trasferisci ACCU 1 nella parola di stato
TAD	CAD	Convertitori	Cambia sequenza di byte in ACCU 1 (a 32 bit)
TAK	TAK	Accumulatori	Scambia ACCU 1 con ACCU 2
TAN	TAN	Funzione in virgola mobile	Formazione della tangente di un angolo come numero in virgola mobile (a 32 bit)
TAR	CAR	Caricamento/Trasferimento	Scambia registro di indirizzo 1 con registro di indirizzo 2
TAR1	TAR1	Caricamento/Trasferimento	Trasferisci registro di indirizzo 1 in ACCU 1
TAR1	TAR1	Caricamento/Trasferimento	Trasferisci registro di indirizzo 1 all'indirizzo di destinazione
TAR1	TAR1	Caricamento/Trasferimento	Trasferisci registro di indirizzo 1 nel registro di indirizzo 2
TAR2	TAR2	Caricamento/Trasferimento	Trasferisci registro di indirizzo 2 in ACCU 1
TAR2	TAR2	Caricamento/Trasferimento	Trasferisci registro di indirizzo 2 all'indirizzo di destinazione
TAW	CAW	Convertitori	Cambia sequenza di byte in ACCU 1 (a 16 bit)
TDB	CDB	Blocco dati	Scambia DB globale e DB di istanza
TRUNC	TRUNC	Convertitori	Arrotonda senza resto
U	A	Operazione logica combinatoria a bit	AND
U(	A(	Operazione logica combinatoria a bit	AND con apertura parentesi
UC	UC	Comando del programma	Richiamo incondizionato di un blocco
UD	AD	Operazione logica combinatoria a parola	AND a doppia parola (a 32 bit
UN	AN	Operazione logica combinatoria a bit	AND negato
UN(	AN(	Operazione logica combinatoria a bit	AND negato con apertura parentesi
UW	AW	Operazione logica combinatoria a parola	AND a parola (a 16 bit
X	X	Operazione logica combinatoria a bit	OR esclusivo
X(	X(	Operazione logica combinatoria a bit	OR esclusivo con apertura parentesi
XN	XN	Operazione logica combinatoria a bit	OR esclusivo negato
XN(	XN(	Operazione logica combinatoria a bit	OR esclusivo negato con apertura parentesi
XOD	XOD	Operazione logica combinatoria a parola	OR esclusivo a doppia parola (a 32 bit)
XOW	XOW	Operazione logica combinatoria a parola	OR esclusivo a parola (a 16 bit)
ZR	CD	Contatori	Conta all'indietro
ZV	CU	Contatori	Conta in avanti

A.2 Operazioni AWL ordinate secondo il set mnemonico inglese (internazionale)

Mnemonico inglese	Mnemonico tedesco	Catalogo elementi del programma	Descrizione
=	=	Operazione logica combinatoria a bit	Assegnazione
)	)	Operazione logica combinatoria a bit	Chiusura parentesi
*D	*D	Funzione in virgola fissa	Moltiplica ACCU 1 per ACCU 2 come numeri interi (a 32 bit)
*I	*I	Funzione in virgola fissa	Moltiplica ACCU 1 per ACCU 2 come numeri interi (a 16 bit)
*R	*R	Funzione in virgola mobile	Moltiplica ACCU 1 per ACCU 2 come numeri in virgola mobile (a 32 bit, IEEE 754)
/D	/D	Funzione in virgola fissa	Dividi ACCU 2 per ACCU 1 come numeri interi (a 32 bit)
/I	/I	Funzione in virgola fissa	Dividi ACCU 2 per ACCU 1 come numeri interi (a 16 bit)
/R	/R	Funzione in virgola mobile	Dividi ACCU 2 per ACCU 1 come numeri in virgola mobile (a 32 bit, IEEE 754)
? D	? D	Comparatori	Confronta numeri interi (a 32 bit)
? I	? I	Comparatori	Confronta numeri interi (a 16 bit)
? R	? R	Comparatori	Confronta numeri in virgola mobile (a 32 bit)
+	+	Funzione in virgola fissa	Somma costante di numero intero (a 16, 32 bit)
+AR1	+AR1	Accumulatori	Somma accumulatore 1 al registro di indirizzo 1
+AR2	+AR2	Accumulatori	Somma accumulatore 1 al registro di indirizzo 2
+D	+D	Funzione in virgola fissa	Somma ACCU 1 e ACCU 2 come numeri interi (a 32 bit)
+I	+I	Funzione in virgola fissa	Somma ACCU 1 e ACCU 2 come numeri interi (a 16 bit)
+R	+R	Funzione in virgola mobile	Somma ACCU 1 e ACCU 2 come numeri in virgola mobile (a 32 bit, IEEE 754)
A	U	Operazione logica combinatoria a bit	AND
A(	U(	Operazione logica combinatoria a bit	AND con apertura parentesi
ABS	ABS	Funzione in virgola mobile	Valore assoluto di un numero in virgola mobile (a 32 bit, IEEE 754)
ACOS	ACOS	Funzione in virgola mobile	Formazione dell'arcoseno di un numero in virgola mobile (a 32 bit)
AD	UD	Operazione logica combinatoria a parola	AND a doppia parola (a 32 bit
AN	UN	Operazione logica combinatoria a bit	AND negato
AN(	UN(	Operazione logica combinatoria a bit	AND negato con apertura parentesi
ASIN	ASIN	Funzione in virgola mobile	Formazione dell'arcoseno di un numero in virgola mobile (a 32 bit)
ATAN	ATAN	Funzione in virgola mobile	Formazione dell'arcotangente di un numero in virgola mobile (a 32 bit)
AW	UW	Operazione logica combinatoria a parola	AND a parola (a 16 bit
BE	BE	Comando del programma	Fine blocco
BEC	BEB	Comando del programma	Fine blocco condizionato

Mnemonico inglese	Mnemonico tedesco	Catalogo elementi del programma	Descrizione
BEU	BEA	Comando del programma	Fine blocco assoluto
BLD	BLD	Accumulatori	Comando di visualizzazione del programma (NOP)
BTD	BTD	Convertitori	Converti numero BCD in numero intero (a 32 bit)
BTI	BTI	Convertitori	Converti numero BCD in numero intero (a 16 bit)
CAD	TAD	Convertitori	Cambia sequenza di byte in ACCU 1 (a 32 bit)
CALL	CALL	Comando del programma	Richiamo di blocco
CALL	CALL	Comando del programma	Richiamo di una multi-istanza
CALL	CALL	Comando del programma	Richiamo di blocchi da una biblioteca
CAR	TAR	Caricamento/Trasferimento	Scambia registro di indirizzo 1 con registro di indirizzo 2
CAW	TAW	Convertitori	Cambia sequenza di byte in ACCU 1 (a 16 bit)
CC	CC	Comando del programma	Richiamo condizionato di un blocco
CD	ZR	Contatori	Conta all'indietro
CDB	TDB	Blocco dati	Scambia DB globale e DB di istanza
CLR	CLR	Operazione logica combinatoria a bit	Resetta RLC (=0)
COS	COS	Funzione in virgola mobile	Formazione del coseno di un angolo come numero in virgola mobile (a 32 bit)
CU	ZV	Contatori	Conta in avanti
–D	–D	Funzione in virgola fissa	Sottrai ACCU 1 da ACCU 2 come numeri interi (a 32 bit)
DEC	DEC	Accumulatori	Decrementa accumulatore 1
DTB	DTB	Convertitori	Converti numero intero (a 32 bit) in un numero BCD
DTR	DTR	Convertitori	Converti numero intero (a 32 bit) in numero in virgola mobile (a 32 bit, IEEE 754)
ENT	ENT	Accumulatori	Immetti stack accumulatore
EXP	EXP	Funzione in virgola mobile	Formazione del valore esponenziale di un numero in virgola mobile (a 32 bit)
FN	FN	Operazione logica combinatoria a parola	Fronte di discesa
FP	FP	Operazione logica combinatoria a parola	Fronte di salita
FR	FR	Contatori	Abilita contatore
FR	FR	Temporizzatori	Abilita temporizzatore
–I	–I	Funzione in virgola fissa	Sottrai ACCU 1 da ACCU 2 come numeri interi (a 16 bit)
INC	INC	Accumulatori	Incrementa accumulatore 1
INVD	INVD	Convertitori	Complemento a 1 di numero intero (a 32 bit)
INVI	INVI	Convertitori	Complemento a 1 di numero intero (a 16 bit)
ITB	ITB	Convertitori	Converti numero intero (a 16 bit) in numero BCD
ITD	ITD	Convertitori	Converti numero intero (a 16 bit) in numero intero (a 32 bit)
JBI	SPBI	Salti	Salta se BIE = 1
JC	SPB	Salti	Salta se RLC = 1
JCB	SPBB	Salti	Salta se RLC = 1 con BIE
JCN	SPBN	Salti	Salta se RLC = 0
JL	SPL	Salti	Salta alle etichette
JM	SPM	Salti	Salta se risultato < 0

Mnemonico inglese	Mnemonico tedesco	Catalogo elementi del programma	Descrizione
JMZ	SPMZ	Salti	Salta se risultato <= 0
JN	SPN	Salti	Salta se risultato diverso da zero
JNB	SPBNB	Salti	Salta se RLC = 0 con BIE
JNBI	SPBIN	Salti	Salta se BIE = 0
JO	SPO	Salti	Salta se OV = 1
JOS	SPS	Salti	Salta se OS = 1
JP	SPP	Salti	Salta se risultato > 0
JPZ	SPPZ	Salti	Salta se risultato >= 0
JU	SPA	Salti	Salto assoluto
JUO	SPU	Salti	Salta se operazione non ammessa
JZ	SPZ	Salti	Salta se risultato = 0
L	L	Caricamento/Trasferimento	Carica
L STW	L STW	Caricamento/Trasferimento	Carica parola di stato in ACCU 1
L	L	Temporizzatori	Carica valore attuale di conteggio in ACCU 1 come numero intero (il valore attuale di conteggio può essere un numero compreso tra 0 e 255, per esempio: L T 32)
L	L	Contatori	Carica valore attuale di conteggio in ACCU 1 come numero intero (il valore di conteggio attuale può essere un numero compreso tra 0 e 255, per esempio: L Z 15)
L DBLG	L DBLG	Blocco dati	Carica lunghezza del DB globale in ACCU 1
L DBNO	L DBNO	Blocco dati	Carica numero del DB globale in ACCU 1
L DILG	L DILG	Blocco dati	Carica lunghezza del DB di istanza in ACCU 1
L DINO	L DINO	Blocco dati	Carica numero del DB di istanza in ACCU 1
LAR1	LAR1	Caricamento/Trasferimento	Carica il registro di indirizzo 1con il contenuto di ACCU 1
LAR1	LAR1	Caricamento/Trasferimento	Carica il registro di indirizzo 1 con numero intero a 32 bit
LAR1	LAR1	Caricamento/Trasferimento	Carica il registro di indirizzo 1 con il contenuto del registro di indirizzo 2
LAR2	LAR2	Caricamento/Trasferimento	Carica il registro di indirizzo 2 con il contenuto di ACCU 1
LAR2	LAR2	Caricamento/Trasferimento	Carica il registro di indirizzo 2 con numero intero (a 32 Bit)
LC	LC	Contatori	Carica valore attuale di conteggio in ACCU 1 come BCD (il valore attuale di conteggio può essere un numero compreso tra 0 e 255, per esempio: LC Z 15)
LC	LC	Temporizzatori	Carica valore attuale di conteggio in ACCU 1 come BCD (il valore attuale di conteggio può essere un numero compreso tra 0 e 255, per esempio: LC T 32)
LEAVE	LEAVE	Accumulatori	Esci da stack accumulatore
LN	LN	Funzione in virgola mobile	Formazione del logaritmo naturale di un numero in virgola mobile (a 32 bit)
LOOP	LOOP	Salti	Loop di programma
MCR(	MCR(	Comando del programma	Salva RLC nello stack MCR, inizio zona MCR
)MCR	)MCR	Comando del programma	Fine zona MCR
MCRA	MCRA	Comando del programma	Attiva zona MCR
MCRD	MCRD	Comando del programma	Disattiva zona MCR
MOD	MOD	Funzione in virgola fissa	Resto della divisione di numero intero (a 32 bit)
NEGD	NEGD	Convertitori	Complemento a 2 di numero intero (a 32 bit)
NEGI	NEGI	Convertitori	Complemento a 2 di numero intero (a 16 bit)

Mnemonico inglese	Mnemonico tedesco	Catalogo elementi del programma	Descrizione
NEGR	NEGR	Convertitori	Complemento a 2 di numero in virgola mobile (a 32 bit, IEEE 754)
NOP 0	NOP 0	Accumulatori	Nessuna operazione 0
NOP 1	NOP 1	Accumulatori	Nessuna operazione 1
NOT	NOT	Operazione logica combinatoria a bit	Nega RLC
O	O	Operazione logica combinatoria a bit	OR
O(	O(	Operazione logica combinatoria a bit	OR con apertura parentesi
OD	OD	Operazione logica combinatoria a parola	OR a doppia parola (a 32 bit)
ON	ON	Operazione logica combinatoria a bit	OR negato
ON(	ON(	Operazione logica combinatoria a bit	OR negato con apertura parentesi
OPN	AUF	Blocco dati	Apri blocco dati
OW	OW	Operazione logica combinatoria a parola	OR a parola (a 16 bit)
POP	POP	Accumulatori	POP CPU con due accumulatori
POP	POP	Accumulatori	POP CPU con quattro accumulatori
PUSH	PUSH	Accumulatori	PUSH CPU con due accumulatori
PUSH	PUSH	Accumulatori	PUSH CPU con quattro accumulatori
R	R	Operazione logica combinatoria a bit	Resetta
R	R	Contatori	Resetta contatore (il contatore attuale può essere un numero compreso tra 0 e 255, per esempio: R Z 15)
R	R	Temporizzatori	Resetta temporizzatore (il temporizzatore attuale può essere un numero compreso tra 0 e 255, per esempio: R T 32)
–R	–R	Funzione in virgola mobile	Sottrai ACCU 1 da ACCU 2 come numeri in virgola mobile (a 32 bit, IEEE 754)
RLD	RLD	Scorrimento/rotazione	Fai ruotare doppia parola a sinistra (a 32 bit)
RLDA	RLDA	Scorrimento/rotazione	Fai ruotare ACCU 1 a sinistra tramite A1 (a 32 bit)
RND	RND	Convertitori	Arrotonda al numero intero
RND–	RND–	Convertitori	Arrotonda al numero intero inferiore (a 32 bit)
RND+	RND+	Convertitori	Arrotonda al numero intero superiore (a 32 bit)
RRD	RRD	Scorrimento/rotazione	Fai ruotare doppia parola a destra (a 32 bit)
RRDA	RRDA	Scorrimento/rotazione	Fai ruotare ACCU 1 a destra tramite A1 (a 32 bit)
S	S	Operazione logica combinatoria a bit	Imposta
S	S	Contatori	Imposta valore iniziale di conteggio (il contatore attuale può essere un numero compreso tra 0 e 255, per esempio: S Z 15)
SAVE	SAVE	Operazione logica combinatoria a bit	Salva RLC nel registro BIE
SD	SE	Temporizzatori	Avvia temporizzatore come ritardo all'inserzione
SE	SV	Temporizzatori	Avvia temporizzatore come impulso prolungato
SET	SET	Operazione logica combinatoria a bit	Imposta RLC (=1)

Mnemonico inglese	Mnemonico tedesco	Catalogo elementi del programma	Descrizione
SF	SA	Temporizzatori	Avvia temporizzatore come ritardo alla disinserzione
SIN	SIN	Funzione in virgola mobile	Formazione del seno di un angolo come numero in virgola mobile (a 32 bit)
SLD	SLD	Scorrimento/rotazione	Fai scorrere doppia parola a sinistra (a 32 bit)
SLW	SLW	Scorrimento/rotazione	Fai scorrere parola a sinistra (a 16 bit)
SP	SI	Temporizzatori	Avvia temporizzatore come impulso
SQR	SQR	Funzione in virgola mobile	Formazione del quadrato di un numero in virgola mobile (a 32 bit)
SQRT	SQRT	Funzione in virgola mobile	Formazione della radice quadrata di un numero in virgola mobile (a 32 bit)
SRD	SRD	Scorrimento/rotazione	Fai scorrere doppia parola a destra (a 32 bit)
SRW	SRW	Scorrimento/rotazione	Fai scorrere parola a destra (a 16 bit)
SS	SS	Temporizzatori	Avvia temporizzatore come ritardo all'inserzione con memoria
SSD	SSD	Scorrimento/rotazione	Fai scorrere numero intero con segno (a 32 bit)
SSI	SSI	Scorrimento/rotazione	Fai scorrere numero intero con segno (a 16 bit)
T	T	Caricamento/Trasferimento	Trasferisci
T STW	T STW	Caricamento/Trasferimento	Trasferisci ACCU 1 nella parola di stato
TAK	TAK	Accumulatori	Scambia ACCU 1 con ACCU 2
TAN	TAN	Funzione in virgola mobile	Formazione della tangente di un angolo come numero in virgola mobile (a 32 bit)
TAR1	TAR1	Caricamento/Trasferimento	Trasferisci registro di indirizzo 1 in ACCU 1
TAR1	TAR1	Caricamento/Trasferimento	Trasferisci registro di indirizzo 1 all'indirizzo di destinazione
TAR1	TAR1	Caricamento/Trasferimento	Trasferisci registro di indirizzo 1 nel registro di indirizzo 2
TAR2	TAR2	Caricamento/Trasferimento	Trasferisci registro di indirizzo 2 in ACCU 1
TAR2	TAR2	Caricamento/Trasferimento	Trasferisci registro di indirizzo 2 all'indirizzo di destinazione
TRUNC	TRUNC	Convertitori	Arrotonda senza resto
UC	UC	Comando del programma	Richiamo incondizionato di un blocco
X	X	Operazione logica combinatoria a bit	OR esclusivo
X(	X(	Operazione logica combinatoria a bit	OR esclusivo con apertura parentesi
XN	XN	Operazione logica combinatoria a bit	OR esclusivo negato
XN(	XN(	Operazione logica combinatoria a bit	OR esclusivo negato con apertura parentesi
XOD	XOD	Operazione logica combinatoria a parola	OR esclusivo a doppia parola (a 32 bit)
XOW	XOW	Operazione logica combinatoria a parola	OR esclusivo a parola (a 16 bit)

B Esempi di programmazione

B.1 Sommario

Applicazione pratiche

Tutte le operazioni AWL inizializzano un'operazione specifica. Combinando queste operazioni in un programma, è possibile eseguire numerose e diversificate operazioni di automazione. Questa appendice contiene i seguenti esempi di applicazioni pratiche:

- Controllo di un nastro trasportatore utilizzando le operazioni logiche combinatorie a bit
- Rilevazione della direzione di movimento di un nastro trasportatore utilizzando le operazioni logiche combinatorie a bit
- Generazione di un impulso di clock utilizzando le operazioni di temporizzazione
- Registrazione dello spazio di memoria avvalendosi delle operazioni di conteggio e confronto
- Soluzione di un problema utilizzando le operazioni matematiche con numeri interi
- Impostazione della durata di riscaldamento di un forno

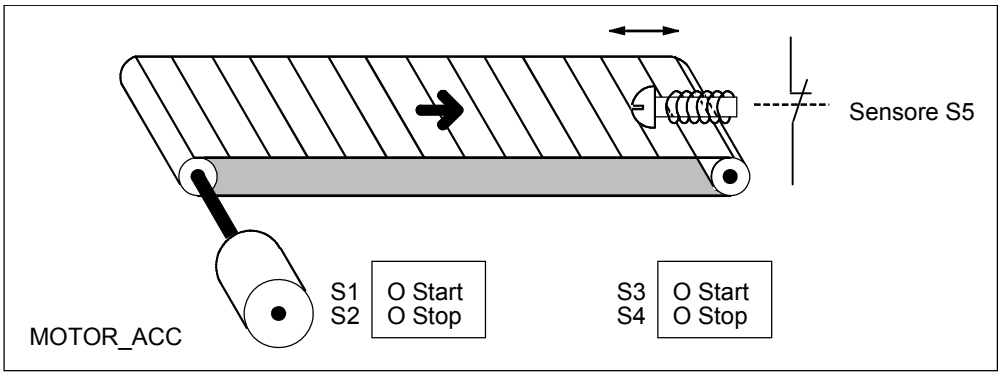
Operazioni utilizzate

Mnemonico	Operazione	Descrizione
UW	Operazione logica a parola	AND a parola
OW	Operazione logica a parola	OR a parola
ZV, ZR	Contatori	Conta in avanti, Conta all'indietro
S, R	Operazione logica a bit	Imposta, Resetta
NOT	Operazione logica a bit	Nega RLC
FP	Operazione logica a bit	Fronte di salita
+I	Funzione in virgola fissa	Somma ACCU1 ad ACCU2 come numeri interi
/I	Funzione in virgola fissa	Dividi ACCU1 per ACCU2 come numeri interi
*I	Funzione in virgola fissa	Moltiplica ACCU1 per ACCU2 come numeri interi
>=I, <=I	Comparatori	Confronta numeri interi
U, UN	Operazione logica a bit	AND, AND negato
O, ON	Operazione logica a bit	O, OR negato
=	Operazione logica a bit	Assegna
INC	Accumulatori	Incrementa ACCU
BE, BEB	Comando del programma	Fine blocco, Fine blocco condizionato
L, T	Caricamento/Trasferimento	Carica, Trasferisci
SV	Temporizzatori	Avvia temporizzatore come impulso prolungato

B.2 Esempi: Operazioni logiche combinatorie a bit

Esempio 1: Controllo di un nastro trasportatore

La figura mostra un nastro trasportatore che può essere attivato elettricamente. Alla partenza del nastro sono presenti due interruttori a pulsante: S1 per START e S2 per STOP. Anche alla fine del nastro sono presenti due interruttori: S3 per START e S4 per STOP. E' quindi possibile avviare e arrestare il nastro da entrambi i suoi capi. La presenza di un sensore S5 permette di arrestare il nastro quando un elemento trasportato raggiunge il punto finale.



Programmazione con valori assoluti e **a simboli**

È possibile scrivere un programma destinato al controllo di un nastro trasportatore, avvalendosi di **valori assoluti** oppure di **simboli** che rappresentano i diversi componenti del sistema di trasporto.

I simboli scelti vengono correlati nella tabella dei simboli con i valori assoluti (consultare la Guida online di STEP 7).

Componente del sistema	Indirizzo assoluto	Simbolo	Tabella dei simboli
Pulsante Start	E 1.1	S1	E 1.1 S1
Pulsante Stop	E 1.2	S2	E 1.2 S2
Pulsante Start	E 1.3	S3	E 1.3 S3
Pulsante Stop	E 1.4	S4	E 1.4 S4
Sensore	E 1.5	S5	E 1.5 S5
Motore	A 4.0	MOTORE_ACC	A 4.0 MOTORE_ACC

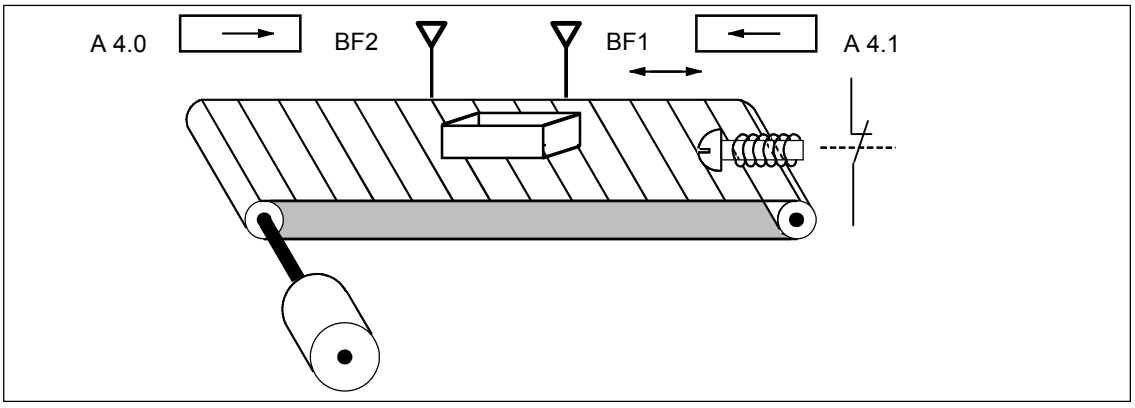
Programmazione assoluta		Programmazione simbolica	
O	E 1.1	O	S1
O	E 1.3	O	S3
S	A 4.0	S	MOTOR_ACC
O	E 1.2	O	S2
O	E 1.4	O	S4
ON	E 1.5	ON	S5
R	A 4.0	R	MOTOR_ACC

Operazione AWL per il controllo del nastro trasportatore

```
AWL             Spiegazione
O     E 1.1     //Premendo uno dei due interruttori di avvio, si accende il motore.
O     E 1.3
S     A 4.0
O     E 1.2     //Premendo uno dei due interruttori di arresto, ovvero aprendo il contatto
                //normalmente chiuso al termine del nastro, si spegne il motore.
O     E 1.4
ON    E 1.5
R     A 4.0
```

Esempio 2: Rilevamento della direzione di marcia di un nastro trasportatore

La seguenta figura mostra un nastro trasportatore che dispone di due barriere a fotocellula (BF1 e BF2) il cui scopo è quello di rilevare la direzione di movimento del materiale sul nastro. Ognuna delle due barriere a fotocellula elettrica funziona come un contatto normalmente aperto.



Programmazione con valori assoluti e a simboli

È possibile scrivere un programma destinato al controllo di un nastro trasportatore, avvalendosi di **valori assoluti** oppure di **simboli** che rappresentano i diversi componenti del sistema di trasporto.

I simboli scelti vengono correlati nella tabella dei simboli con i valori assoluti (consultare la Guida online di STEP 7).

Componente del sistema	Indirizzo assoluto	Simbolo	Tabella dei simboli
Barriera a fotocellula elettrica 1	E 0.0	BF 1	E 0.0 BF 1
Barriera a fotocellula elettrica 2	E 0.1	BF 2	E 0.1 BF 2
Visualizzatore per il senso di marcia a destra	A 4.0	DESTRA	A 4.0 DESTRA
Visualizzatore per il senso di marcia a sinistra	A 4.1	SINISTRA	A 4.1 SINISTRA
Bit 1 della memoria d'impulso	M 0.0	MI1	M 0.0 MI 1
Bit 2 della memoria d'impulso	M 0.1	MI 2	M 0.1 MI 2

Programmazione assoluta		Programmazione simbolica	
U	E 0.0	U	BF1
FP	M 0.0	FP	MI1
UN	E 0.1	UN	BF 2
S	A 4.1	S	SINISTRA
U	E 0.1	U	BF 2
FP	M 0.1	FP	MI 2
UN	E 0.0	UN	MI 1
S	A 4.0	S	DESTRA
UN	E 0.0	UN	BF 1
UN	E 0.1	UN	BF 2
R	A 4.0	R	DESTRA
R	A 4.1	R	SINISTRA

Operazione AWL per il rilevamento della direzione di marcia di un nastro trasportatore

```
AWL                 Spiegazione
U       E 0.0       //Se vi è un cambio nello stato di segnale da 0 a 1 (fronte di salita) all'ingresso
                    //E 0.0 e, contemporaneamente, lo stato di segnale all'ingresso E 0.1 è 0, il
                    //pacco sul nastro si muove verso sinistra.
FP      M 0.0
UN      E 0.1
S       A 4.1
U       E 0.1       //Se vi è un cambio nello stato di segnale da 0 a 1 (fronte di salita) all'ingresso
                    //E 0.1 e, contemporaneamente, lo stato di segnale all'ingresso E 0.0 è 0, il
                    //pacco sul nastro si muove verso destra. Se una delle barriere di luce
                    //fotoelettrica risulta spezzata (discontinua), ciò significa che vi è un pacco
                    //tra le due barriere.
FP      M 0.1
UN      E 0.0
S       A 4.0
UN      E 0.0       //Se nessuna delle barriere di luce fotoelettrica risulta spezzata non vi è
                    //alcun pacco tra le due barriere. Il puntatore di direzione si disinserisce.
UN      E 0.1
R       A 4.0
R       A 4.1
```

B.3 Esempio: Operazioni di temporizzazione

Generatore d'impulso di clock

È possibile utilizzare un generatore di impulsi di clock o un relè di lampeggio per poter produrre un segnale che si ripete periodicamente. Un generatore di impulsi di clock è alquanto comune in un sistema di segnalazione che controlla il lampeggio delle spie.

Quando si utilizza S7-300, si può implementare la funzione di generazione d'impulsi avvalendosi di un'elaborazione temporizzata in blocchi di organizzazione speciali.

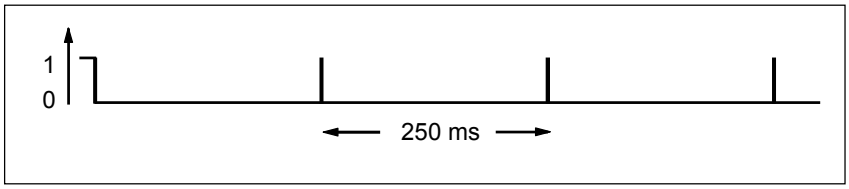
Operazione AWL per l'attivazione delle spie su un pannello visualizzatore (fattore di impulso 1:1)

```
AWL               Spiegazione
UN     T1         //Se è scaduto il temporizzatore T1,
L      S5T#250ms  //carica il valore di tempo 250ms in T1
SV     T1         //ed avvia T1 come un generatore di impulso prolungato.
NOT               //Nega (inverte) il risultato logico combinatorio.

BEB               //Se il tempo è in corso, termina il blocco attuale.
L      MB100      //Se il temporizzatore è scaduto, carica il contenuto del byte del merker MB100,
INC    1          //incrementa il contenuto di 1 e
T      MB100      //trasferisce il risultato al byte di merker MB100.
```

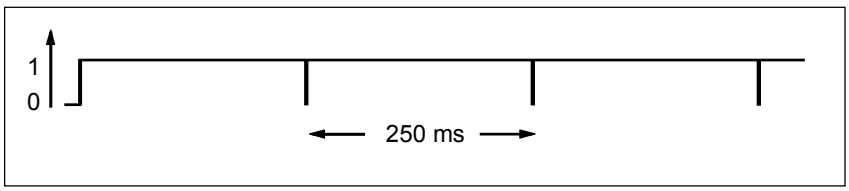
Interrogazione di segnale

Un'interrogazione di segnale del temporizzatore T1 produce il risultato logico combinatorio.



Appena trascorso il tempo, il temporizzatore viene riavviato. L'interrogazione di segnale effettuata dall'istruzione UN T1 produce soltanto brevemente uno stato di segnale di 1.

Il bit di negazione RLC (invertito):



Ogni 250 ms il bit RLC negato è 0. Tuttavia, l'istruzione BEB non termina l'elaborazione del blocco. Invece, il contenuto del byte di merker MB100 viene incrementato di 1.

Il contenuto del byte di merker MB100 cambia ogni 250 ms nel modo seguente:

0 -> 1 -> 2 -> 3 -> ... -> 254 -> 255 -> 0 -> 1 ...

Ottenimento di una frequenza spezifica

Con i bit dei merker MB100 è possibile ottenere le seguenti frequenze:

Bit di MB100	Frequenza in Hertz	Durata	
M 100.0	2.0	0.5 s	(250 ms on / 250 ms off)
M 100.1	1.0	1 s	(0.5 s on / 0.5 s off)
M 100.2	0.5	2 s	(1 s on / 1 s off)
M 100.3	0.25	4 s	(2 s on / 2 s off)
M 100.4	0.125	8 s	(4 s on / 4 s off)
M 100.5	0.0625	16 s	(8 s on / 8 s off)
M 100.6	0.03125	32 s	(16 s on / 16 s off)
M 100.7	0.015625	64 s	(32 s on / 32 s off)

Operazione AWL

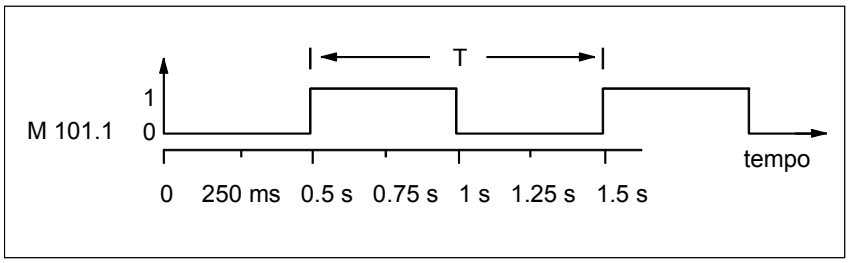
```
AWL             Spiegazione
U     M10.0     //M10.0 = 1 quando interviene un errore. La spia di errore lampeggia alla frequenza
                di 1 Hz quando interviene un errore.
U     M100.1
=     A 4.0
```

Stati dei segnali dei bit del byte di merker MB101

Ciclo	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Valore di tempo in ms
0	0	0	0	0	0	0	**0**	0	250
1	0	0	0	0	0	0	**0**	1	250
2	0	0	0	0	0	0	1	0	250
3	0	0	0	0	0	0	1	1	250
4	0	0	0	0	0	1	**0**	0	250
5	0	0	0	0	0	1	**0**	1	250
6	0	0	0	0	0	1	1	0	250
7	0	0	0	0	0	1	1	1	250
8	0	0	0	0	1	0	**0**	0	250
9	0	0	0	0	1	0	**0**	1	250
10	0	0	0	0	1	0	1	0	250
11	0	0	0	0	1	0	1	1	250
12	0	0	0	0	1	1	**0**	0	250

Stato del segnale del bit 1 di MB101 (M 101.1)

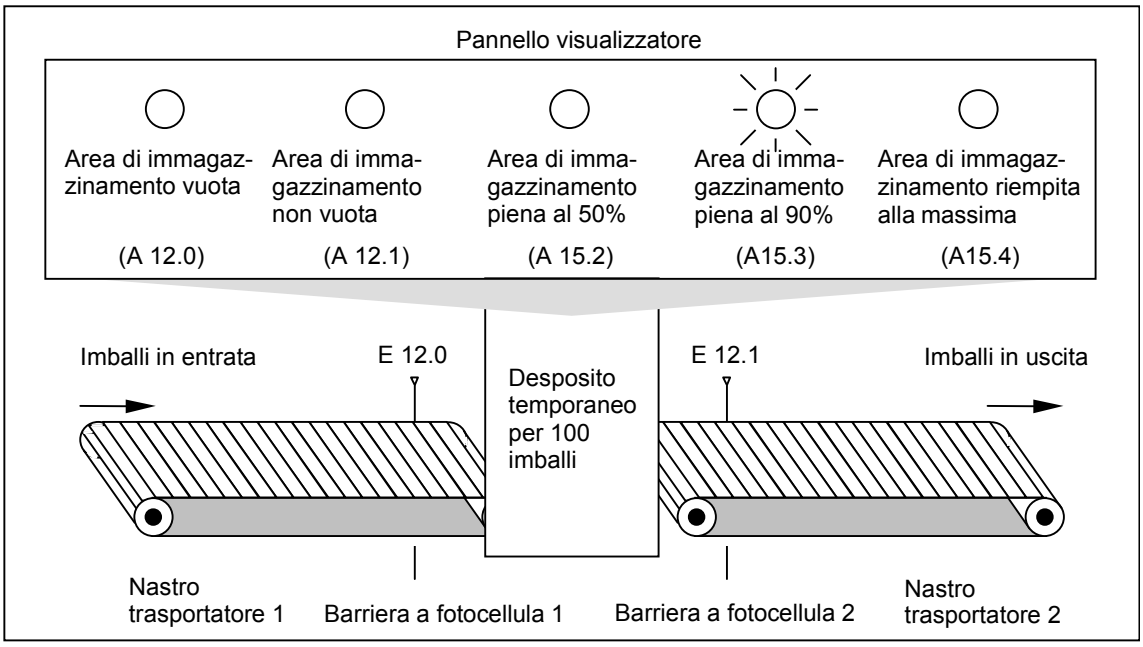
Frequenza = 1/T = 1/1 s = 1 Hz



B.4 Esempio: Operazioni di conteggio e confronto

Area di immagazzinamento con Contatore e Confrontatore

La seguenta figura mostra un sistema con due nastri trasportatori e un'area di immagazzinamento temporaneo tra i due sistemi di trasporto. Il nastro trasportatore 1 invia il materiale nell'area di immagazzinamento. Una barriera a fotocellula alla fine del nastro 1 in prossimità dell'area di immagazzinamento determina quanti imballi sono stati trasportati nell'area di immagazzinamento. Il nastro 2 trasporta gli imballi dall'area di immagazzinamento temporaneo fino ad una piattaforma di carico dove degli autocarri sono pronti a ricevere il materiale da consegnare al cliente. Una barriera a fotocellula alla fine del nastro 2 in prossimità dell'area di immagazzinamento registra il numero degli imballi che escono dall'area di immagazzinamento per essere trasportati verso la piattaforma di carico. Un pannello visualizzatore dispone di cinque spie che segnalano il livello di riempimento dell'area di immagazzinamento.



Operazione AWL per l'attivazione delle spie su un pannello visualizzatore

```
AWL             Spiegazione
U      E 0.0    //Ogni impulso generato dalla barriera fotoelettrica 1
ZV     Z1       //incrementa il valore di conteggio del contatore Z1 di uno, contando contemporaneam.
                //il numero di pacchi che entrano nell'area di stoccaggio.
                //
U      E 0.1    //Ogni impulso generato dalla barriera fotoelettrica 2
ZR     Z1       //decrementa il valore di conteggio del contatore Z1 di uno, contando
                //contemporaneamente il numero di pacchi che escono dall'area di stoccaggio.
                //
UN     Z1       //Se il valore di conteggio non è 0,
=      A 4.0    //si accende la spia indicatrice di "Area di stoccaggio vuota".
                //
U      Z1       //Se il valore di conteggio non è 0,
=      A 4.1    //si accende la spia indicatrice di "Area di stoccaggio non vuota".
                //
L      50
L      Z1
<=I             //Se il valore di conteggio è minore o uguale a 50,
=      A 4.2    //si accende la spia indicatrice di "Area di stoccaggio piena al 50%".
                //
L      90
>=I             //Se il valore di conteggio è maggiore o uguale a 90,
=      A 4.3    //si accende la spia indicatrice di "Area di stoccaggio piena al 90%".
                //
L      Z1
L      100
>=I             //Se il valore di conteggio è maggiore o uguale a 100,
=      A 4.4    //si accende la spia indicatrice di "Area di stoccaggio completamente piena".
                //(Si potrebbe utilizzare anche l'uscita A 4.4 per bloccare il nastro
                //trasportatore 1).
```

B.5 Esempio: Operazioni matematiche con i numeri interi

Soluzione di un problema matematico

Il seguente programma di esempio mostra come utilizzare tre operazioni matematiche con numeri interi per conseguire lo stesso risultato che si ottiene dalla seguente equazione:

MD4 = ((EW0 + DBW3) x 15) / MW2

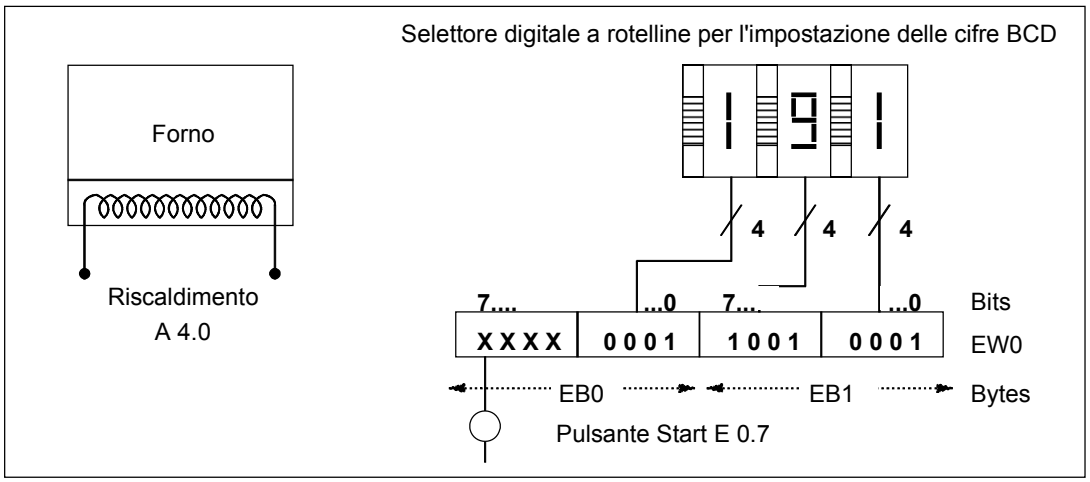
Operazione AWL

```
AWL             Spiegazione
L     EW0       //Carica il valore della parola di ingresso EW0 in ACCU 1.
L     DB5.DBW3  //Carica il valore della parola di dati globali DBW3 dal DB5 in ACCU 1. Il contenuto
                //precedente di ACCU 1 viene fatto scorrere in ACCU 2.
+I    E 0.1     //Somma il contenuto delle parole basse degli accumulatori 1 e 2. Il risultato
                //viene memorizzato nella parola bassa di ACCU 1. Il contenuto di ACCU 2 e la
                //parola alta di ACCU 1 restano inalterati.
L     +15       //Carica il valore della costante +15 in ACCU 1. Il contenuto precedente di
                //ACCU 1 viene fatto scorrere in ACCU 2.
*I              //Moltiplica il contenuto della parola bassa di ACCU 2 per il contenuto della
                //parola bassa di ACCU 1. Il risultato viene memorizzato in ACCU 1. Il contenuto
                //di ACCU 2 resta inalterato.
L     MW2       //Carica il valore della parola di merker MW2 in ACCU 1. Il contenuto precedente
                //dell'ACCU 1 viene fatto scorrere in ACCU 2.
/I              //Divide il contenuto della parola bassa di ACCU 2 per il contenuto della parola
                //bassa di ACCU 1. Il risultato viene memorizzato in ACCU 1. Il contenuto di
                //ACCU 2 resta inalterato.
T     MD4       //Trasferisce il risultato finale alla doppia parola di merker MD4. Il contenuto
                //di ambedue gli accumulatori resta inalterato.
```

B.6 Esempio: Operazioni logiche combinatorie a parola

Riscaldamento di un forno

L'operatore di un forno avvia il riscaldamento del forno premendo il pulsante Start. L'operatore può impostare la durata del tempo di riscaldamento avvalendosi di selettori a rotella. Il valore che l'operatore imposta indica i secondi in formato BCD (decimali codificati in binario).



Componente del sistema	Indirizzo assoluto
Pulsante Start	E 0.7
Rotellina delle unità	da E 1.0 a E 1.3
Rotellina delle decine	da E 1.4 a E 1.7
Rotellina delle centinaia	da E 0.0 a E 0.3
Avvio riscaldamento	A 4.0

Operazione AWL

```
AWL             Spiegazione
U     T1        //Se il temporizzatore è operante, esso accende il riscaldamento.
=     A 4.0
BEB             //Se il temporizzatore è operante, esso termina qui l'elaborazione.
                //Ciò impedisce al temporizzatore di essere riavviato, se viene premuto il
                //pulsante.
L     EW0
UW    W#16#0FFF //Maschera i bit di ingresso da E 0.4 a E 0.7 (vale a dire, li resetta a 0).
                //Il valore di tempo in secondi si trova nella parola bassa di ACCU 1 in formato
                //di cifra decimale in codice binario.
OW    W#16#2000 //Assegna la base di tempo come secondi nei bit 12 e 13 della parola
                //bassa di ACCU 1.
U     E 0.7
SV    T1        //Avvia il temporizzatore T1 come impulso prolungato se viene premuto il pulsante.
```

C Esempio di attributi del sistema per parametri

Definizione del problema

In un funzionamento di tecnica di processo vanno in tutto controllati quattro segnali in tre parti dell'impianto: uno in ciascuna delle parti dell'impianto A e B e due nella parte dell'impianto C. Sia con fronte di salita che di discesa deve essere generata una segnalazione a tutti i sistemi di servizio e supervisione attivati. La prova del cambio di segnale deve essere effettuata ogni 100 ms.

Va progettato un programma di controllo tale che tutti i dati concernenti una parte dell'impianto siano depositati in un blocco dati.

Soluzione

Il controllo segnale avviene con l'ausilio dell'SFB 33 "ALARM". Vengono programmati due blocchi funzionali. L'FB "BLOCCO SEGNALAZIONE" richiama l'SFB 33 "ALARM". Esso controlla un segnale e viene utilizzato per le parti A e B dell'impianto. Va fatta attenzione affinché si assegnino alla variabile di ingresso "Numero di segnalazione" gli attributi del sistema rilevanti per la generazione della segnalazione. Si deve inoltre tenere in considerazione che al richiamo dell'SFB 33 "ALARM" i dati di istanza propri sono nel DB di istanza dell'FB "BLOCCO SEGNALAZIONE" (multi-istanza).

Nell'FB "GRUPPO SEGNALAZIONE" viene richiamato due volte l'FB "BLOCCO SEGNALAZIONE", e cioè in modo che i dati di istanza propri siano nel DB di istanza dell'FB "GRUPPO SEGNALAZIONE" (multi-istanza). Serve al controllo della parte C dell'impianto.

Gli FB "BLOCCO SEGNALAZIONE" e "GRUPPO SEGNALAZIONE" vengono richiamati dall' OB35 (OB schedulazione orologio con reticolo temporale 100 ms).

Indice analitico

D

E

F

G

I

L

M

N

O

P

R

S

T

U

V

X

Z