

SIMATIC

S7-SCL V5.3 per S7-300/400

Manuale

Prefazione, Indice

Presentazione del prodotto

1

Installazione

2

Ideazione di un programma
S7-SCL

3

Come utilizzare S7-SCL

4

Concetti di base S7-SCL

5

Struttura del programma
S7-SCL

6

Tipi di dati

7

Dichiarazione di variabili locali
e parametri

8

Convenzioni di costanti e
etichette di salto

9

Dati globali

10

Espressioni, operazioni e
operandi

11

Istruzioni

12

Contatori e temporizzatori

13

Funzioni standard di S7-SCL

14

Descrizione del linguaggio
di programmazione

15

Suggerimenti e strategie

16

Glossario, Indice analitico

Istruzioni tecniche di sicurezza

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 decrescente i diversi livelli di rischio.



Pericolo

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



Avvertenza

il simbolo indica che la mancata osservanza delle relative misure di sicurezza la morte o gravi lesioni fisiche.



Cautela

indica che la mancata osservanza delle relative misure di sicurezza può causare lesioni fisiche non gravi.

Cautela

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

L'apparecchio/sistema in questione deve essere installato e messo in servizio solo rispettando le indicazioni contenute in questa documentazione. La messa in servizio e l'esercizio di un apparecchio/sistema devono essere eseguiti solo da **personale qualificato**. Con riferimento alle indicazioni contenute in questa documentazione in merito alla sicurezza, come personale qualificato si intende quello autorizzato a mettere in servizio, eseguire la relativa messa a terra e contrassegnare le apparecchiature, i sistemi e i circuiti elettrici rispettando gli standard della tecnica di sicurezza.

Uso regolamentare delle apparecchiature/dei sistemi:

Si prega di tener presente quanto segue:



Avvertenza

L'apparecchiatura può essere destinata solo agli impieghi previsti nel catalogo e nella descrizione tecnica e può essere utilizzata solo insieme a apparecchiature e componenti di Siemens o di altri costruttori raccomandati o omologati dalla Siemens.

Per garantire un funzionamento ineccepibile e sicuro del prodotto è assolutamente necessario che le modalità di trasporto, di immagazzinamento, di installazione e di montaggio siano corrette, che l'apparecchiatura venga usata con cura e che si provveda ad una manutenzione appropriata.

Marchio di prodotto

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.

Copyright Siemens AG 2005 All rights reserved

La duplicazione e la cessione di questa documentazione sono vietate, come pure l'uso improprio del suo contenuto salvo espressa autorizzazione. Qualsiasi violazione comporta un risarcimento danni. Tutti i diritti sono riservati, in particolare quelli relativi ai brevetti e ai marchi registrati.

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.

Prefazione

Scopo del manuale

Il presente manuale descrive la programmazione con S7-SCL e ha l'obiettivo di supportare l'utente nell'installazione e la messa in servizio del software. Vi vengono quindi illustrati la procedura di creazione dei programmi, la struttura dei programmi utente e gli elementi del linguaggio.

Il manuale si rivolge agli operatori che programmano in S7-SCL e al personale che si occupa di progettazione, messa in servizio e assistenza dei sistemi di automazione.

Si consiglia di iniziare dall'esempio illustrato nel capitolo 2 "Progettazione di un programma SCL" che consente di apprendere le basi della programmazione in S7-SCL.

Formazione richiesta

Per poter comprendere il manuale è necessario disporre di conoscenze generali nel settore della tecnica d'automazione.

Sono inoltre richieste conoscenze sull'utilizzo dei computer o di dispositivi simili ai PC (ad es. i dispositivi di programmazione) eseguibili in MS Windows 2000 Professional e MS Windows XP Professional. Poiché SCL viene installato in STEP 7 è necessario disporre di conoscenze anche su questo software di base e consultare il manuale "Programmazione con STEP 7 V5.3".

Validità del manuale

Il manuale vale per il pacchetto S7-SCL V5.3 a partir dal Service Pack 1.

Pacchetti di documentazione per S7-SCL e il software di base STEP 7

La seguente tabella fa un riepilogo della documentazione disponibile su STEP 7 e S7-SCL.

Manuali	Scopo	Numero di ordinazione
Informazioni di base e di riferimento su S7SCL con <ul style="list-style-type: none"> S7-SCL per S7-300/400: Programmazione di blocchi 	Informazioni di base e di riferimento relative alla procedura di creazione dei programmi, alla struttura dei programmi utente e agli elementi del linguaggio.	Il manuale è disponibile su CD, Manual Collection ed in Internet. Non è possibile ordinarlo separatamente.
Informazioni di base su STEP 7 con <ul style="list-style-type: none"> Primi passi ed esercitazioni con STEP 7 V5.3 Programmazione con STEP 7 V5.3 Configurazione dell'hardware e progettazione dei collegamenti con STEP 7 V5.3 Manuale di conversione STEP 7, da S5 a S7 	Fornisce informazioni di base per il personale tecnico e descrive la procedura di realizzazione dei compiti di controllo con STEP 7 e S7-300/400.	6ES7810-4CA07-8EW0
Informazioni di riferimento su STEP 7 con <ul style="list-style-type: none"> Manuali KOP/FUP/AWL per S7-300/400 Funzioni standard e di sistema per S7-300/400 	Il manuale di riferimento è un testo di consultazione che descrive i linguaggi di programmazione KOP, FUP e AWL e le funzioni standard e di sistema e che integra il manuale di base di STEP 7.	6ES7810-4CA07-8EW1

Guide online	Scopo	Numero di ordinazione
Guida a S7-SCL	Manuale d'uso e di riferimento di S7-SCL disponibile come Guida online	Fa parte del pacchetto software S7-SCL
Guida a STEP 7	Manuale d'uso per la programmazione e la configurazione dell'hardware con STEP 7 disponibile come Guida online	Fa parte del pacchetto software STEP 7
Guide di riferimento di AWL/KOP/FUP Guida di riferimento di SFB/SFC Guida di riferimento dei blocchi organizzativi Guida di riferimento delle funzioni IEC Guida di riferimento degli attributi di sistema	Manuale di riferimento sensibile al contesto	Fanno parte del pacchetto software STEP 7

Guida online

La Guida online è sempre disponibile e consente di ottenere rapidamente le informazioni necessarie senza doverle cercare nel manuale. La Guida online contiene:

- **Argomenti della Guida:** consentono di accedere in vari modi alle informazioni della Guida .
- **Guida al contesto** (tasto F1): visualizza informazioni sull'oggetto selezionato oppure sul campo o la finestra di dialogo attiva ecc.
- **Introduzione:** breve riepilogo dell'utilizzo, delle caratteristiche principali e delle funzioni dell'applicazione.
- **Primi passi:** riassume le prime operazioni necessarie per eseguire le operazioni di base.
- **Uso della Guida:** descrive le modalità di ricerca delle informazioni della Guida.
- **Informazioni su:** indica la versione attuale dell'applicazione.

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.

Telefono: +49 (911) 895-3200.

Internet: <http://www.sitrain.com>

Technical Support

Per tutti i prodotti A&D è possibile rivolgersi al Technical Support

- mediante il modulo Web per la Support Request
<http://www.siemens.com/automation/support-request>
- Telefono: + 49 180 5050 222
- Fax: + 49 180 5050 223

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 Automation & Drives;
- informazioni su assistenza tecnica sul posto, riparazioni, parti di ricambio e maggiori dettagli alla voce "Service".

Indice

1	Presentazione del prodotto	1-1
1.1	Campo di applicazione di S7-SCL	1-1
1.2	Vantaggi di S7-SCL.....	1-2
1.3	Modalità operativa di S7-SCL	1-4
1.4	Nuove funzioni della versione V5.3 SP1	1-6
2	Installazione	2-1
2.1	Automation License Manager	2-1
2.1.1	Concessione della licenza d'utilizzo mediante Automation License Manager	2-1
2.1.2	Installazione dell'Automation License Manager	2-3
2.1.3	Regole per l'utilizzo delle chiavi di licenza	2-4
2.2	Installazione	2-5
2.2.1	Requisiti della installazione	2-5
2.2.2	Installazione di S7-SCL	2-5
3	Ideazione di un programma S7-SCL	3-1
3.1	Esempio introduttivo "Rilevazione di un valore di misura"	3-1
3.2	Impostazione del problema	3-2
3.3	Realizzazione di un programma strutturato con S7-SCL.....	3-4
3.4	Definizione dei compiti parziali.....	3-6
3.5	Definizione delle interfacce fra i blocchi	3-8
3.6	Definizione dell'interfaccia d'ingresso/uscita	3-10
3.7	Definizione della sequenza dei blocchi nella sorgente	3-11
3.8	Definizione di simboli.....	3-11
3.9	Come creare la funzione QUADRATO.....	3-12
3.10	Come creare il blocco funzionale ANALISI	3-13
3.10.1	Diagramma di flusso di ANALISI	3-13
3.10.2	Parte dichiarazioni dell'FB ANALISI	3-14
3.10.3	Parte istruzioni dell'FB ANALISI.....	3-15
3.11	Come creare il blocco funzionale RILEVAZIONE	3-17
3.11.1	Diagramma di flusso di Rilevazione	3-17
3.11.2	Parte dichiarazioni dell'FB RILEVAZIONE	3-18
3.11.3	Parte istruzioni dell'FB RILEVAZIONE.....	3-20
3.12	Come creare il blocco organizzativo CICLO	3-23
3.13	Dati di test	3-25
4	Come utilizzare S7-SCL.....	4-1
4.1	Avvio del software S7-SCL.....	4-1
4.2	Superficie operativa.....	4-2
4.3	Adattamento della superficie operativa	4-3
4.4	Come creare e gestire le sorgenti S7-SCL	4-4
4.4.1	Generazione di una nuova sorgente S7-SCL	4-4
4.4.2	Apertura di una sorgente S7-SCL	4-5
4.4.3	Chiusura di una sorgente S7-SCL	4-5
4.4.4	Apertura di blocchi.....	4-6
4.4.5	Definizione delle proprietà di oggetti	4-6
4.4.6	Generazione di sorgenti S7-SCL con un editor standard	4-6
4.4.7	Protezione dei blocchi	4-7

4.5	Regole per le sorgenti S7-SCL	4-8
4.5.1	Regole generali per le sorgenti S7-SCL.....	4-8
4.5.2	Ordine dei blocchi.....	4-8
4.5.3	Utilizzo di indirizzi simbolici	4-9
4.6	Come editare nelle sorgenti S7-SCL.....	4-10
4.6.1	Annullamento dell'ultima azione di editazione	4-10
4.6.2	Ripristino di un'azione di editazione	4-10
4.6.3	Ricerca e sostituzione di oggetti di testo.....	4-10
4.6.4	Selezione di oggetti di testo	4-11
4.6.5	Copia di oggetti di testo.....	4-11
4.6.6	Taglio di oggetti di testo	4-12
4.6.7	Cancellazione di oggetti di testo	4-12
4.6.8	Posizionamento del cursore su una determinata riga	4-13
4.6.9	Allineamento delle righe in base alla sintassi	4-14
4.6.10	Impostazione dello stile e del colore del carattere	4-15
4.6.11	Posizionamento dei segnalibro nel testo sorgente	4-16
4.6.12	Come inserire le variabili.....	4-17
4.6.12.1	Inserimento di richiami di blocco	4-17
4.6.12.2	Inserimento di modelli per i blocchi	4-17
4.6.12.3	Inserimento di modelli per i commenti.....	4-17
4.6.12.4	Inserimento di modelli per i parametri.....	4-18
4.6.12.5	Inserimento di strutture di controllo	4-18
4.7	Come compilare i programmi S7-SCL	4-19
4.7.1	Istruzioni fondamentali per la compilazione	4-19
4.7.2	Impostazione del compilatore	4-20
4.7.3	Compilazione del programma	4-21
4.7.4	Creazione di un file di compilazione.....	4-22
4.7.5	Eliminazione di errori dopo la compilazione.....	4-22
4.8	Come salvare e stampare le sorgenti S7-SCL.....	4-23
4.8.1	Salvataggio di una sorgente S7-SCL	4-23
4.8.2	Impostazione del formato di pagina	4-23
4.8.3	Stampa di una sorgente S7-SCL	4-24
4.8.4	Impostazione delle opzioni di stampa	4-25
4.9	Come caricare i programmi.....	4-26
4.9.1	Cancellazione totale della memoria CPU.....	4-26
4.9.2	Caricamento di programmi utente nella CPU.....	4-26
4.10	Come testare i programmi.....	4-28
4.10.1	Funzioni di test di S7-SCL.....	4-28
4.10.2	Informazioni importanti sulla funzione di test "Controlla"	4-29
4.10.3	Informazioni importanti sul test con punti d'arresto/in modalità passo singolo	4-30
4.10.4	Fasi del controllo	4-31
4.10.4.1	Definizione di un ambiente di richiamo del blocco	4-32
4.10.5	Fasi del test con punti di arresto	4-33
4.10.5.1	Definizione dei punti d'arresto	4-33
4.10.5.2	Avvio del test con i punti d'arresto.....	4-33
4.10.5.3	Conclusione del test con i punti d'arresto	4-34
4.10.5.4	Attivazione, disattivazione e cancellazione dei punti d'arresto	4-34
4.10.5.5	Definizione di un ambiente di richiamo per punti d'arresto	4-35
4.10.5.6	Test in modalità passo singolo	4-36
4.10.6	Funzioni di test di STEP 7	4-37
4.10.6.1	Creazione o visualizzazione dei dati di riferimento	4-37
4.10.6.2	Controllo e comando di variabili	4-38
4.10.6.3	Verifica coerenza blocchi	4-38

4.11	Visualizzazione e modifica delle proprietà della CPU	4-40
4.11.1	Visualizzazione e modifica dello stato di funzionamento della CPU	4-40
4.11.2	Visualizzazione e impostazione di data e ora della CPU	4-40
4.11.3	Visualizzazione dei dati della CPU	4-41
4.11.4	Lettura del buffer di diagnostica della CPU	4-41
4.11.5	Visualizzazione/compressione della memoria utente della CPU	4-41
4.11.6	Visualizzazione del tempo di ciclo della CPU	4-42
4.11.7	Visualizzazione delle caratteristiche dell'orologio della CPU	4-42
4.11.8	Visualizzazione dei blocchi della CPU	4-42
4.11.9	Visualizzazione delle informazioni per la comunicazione con la CPU	4-42
4.11.10	Visualizzazione degli stack della CPU	4-43
5	Concetti di base S7-SCL	5-1
5.1	Interpretazione dei diagrammi sintattici	5-1
5.2	Set di caratteri	5-4
5.3	Parole riservate	5-5
5.4	Identificatori	5-6
5.5	Identificatori standard	5-7
5.6	Identificatori di blocchi	5-7
5.7	Identificatori di operandi	5-9
5.8	Identificatori di temporizzatori	5-10
5.9	Identificatori di contatori	5-10
5.10	Numeri	5-11
5.11	Stringhe di caratteri	5-13
5.12	Simbolo	5-14
5.13	Blocco di commenti	5-15
5.14	Commento di una riga	5-16
5.15	Variabili	5-17
6	Struttura del programma S7-SCL	6-1
6.1	Blocchi nelle sorgenti S7-SCL	6-1
6.2	Ordine dei blocchi	6-2
6.3	Struttura generica del blocco	6-3
6.4	Inizio e fine di un blocco	6-3
6.5	Attributi di blocco	6-5
6.6	Commento al blocco	6-7
6.7	Attributi di sistema per i blocchi	6-7
6.8	Parte convenzioni	6-8
6.9	Attributi di sistema per i parametri	6-9
6.10	Parte istruzioni	6-10
6.11	Istruzioni	6-11
6.12	Struttura di un blocco funzionale (FB)	6-12
6.13	Struttura di una funzione (FC)	6-14
6.14	Struttura di un blocco organizzativo (OB)	6-16
6.15	Struttura di un blocco dati (DB)	6-17
6.16	Struttura di un tipo di dati definito dall'utente	6-20
6.17	Opzioni di compilazione nelle sorgenti S7-SCL	6-22

7	Tipi di dati	7-1
7.1	Sommario dei tipi di dati in S7-SCL	7-1
7.2	Tipi di dati semplici	7-3
7.2.1	Tipi di dati a bit	7-3
7.2.2	Tipi di dati a caratteri	7-3
7.2.3	Tipi di dati numerici	7-3
7.2.4	Tipi di temporizzatori	7-4
7.3	Tipi di dati composti	7-5
7.3.1	Tipo di dati DATE_AND_TIME	7-5
7.3.2	Tipo di dati STRING	7-7
7.3.3	Tipo di dati ARRAY	7-9
7.3.4	Tipo di dati STRUCT	7-11
7.4	Tipi di dati definiti dall'utente	7-13
7.4.1	Tipo di dati definito dall'utente (UDT)	7-13
7.5	Tipi di dati per parametri	7-15
7.5.1	Tipi di dati TIMER e COUNTER	7-15
7.5.2	Tipo di dati BLOCK	7-16
7.5.3	Tipo di dati POINTER	7-16
7.6	Tipo di dati ANY	7-18
7.6.1	Esempio di un tipo di dati ANY	7-19
8	Dichiarazione di variabili locali e parametri	8-1
8.1	Variabili locali e parametri di blocco	8-1
8.2	Sintassi generale di una dichiarazione di variabili o parametri	8-2
8.3	Inizializzazione	8-3
8.4	Dichiarazione di accessi a campi di variabili	8-5
8.5	Uso di multiistanze	8-6
8.6	Dichiarazione di istanze	8-6
8.7	Flag (Flag OK)	8-7
8.8	Componenti di dichiarazione	8-8
8.8.1	Sommario delle sezioni di dichiarazione	8-8
8.8.2	Variabili statiche	8-9
8.8.3	Variabili temporanee	8-10
8.8.4	Parametri di blocchi	8-11
9	Convenzioni di costanti e etichette di salto	9-1
9.1	Costanti	9-1
9.1.1	Convenzione per nomi simbolici di costanti	9-2
9.1.2	Tipi di dati delle costanti	9-3
9.1.3	Modo di scrittura per costanti	9-4
9.1.3.1	Costanti a bit	9-6
9.1.3.2	Costante di numero intero	9-7
9.1.3.3	Costante di numero reale	9-8
9.1.3.4	Costante Char (carattere singolo)	9-9
9.1.3.5	Costante String	9-11
9.1.3.6	Costante di data	9-13
9.1.3.7	Costante di intervallo	9-13
9.1.3.8	Costante ora del giorno	9-16
9.1.3.9	Costante di data e ora	9-17
9.2	Convenzione di etichette di salto	9-17

10	Dati globali.....	10-1
10.1	Dati globali.....	10-1
10.2	Aree di memoria della CPU.....	10-2
10.2.1	Aree di memoria della CPU.....	10-2
10.2.2	Accesso simbolico alle aree di memoria della CPU.....	10-3
10.2.3	Accesso indicizzato alle aree di memoria della CPU.....	10-4
10.2.4	Accesso assoluto alle aree di memoria della CPU.....	10-5
10.3	Blocchi dati.....	10-7
10.3.1	Blocchi dati.....	10-7
10.3.2	Accesso assoluto ai blocchi dati.....	10-8
10.3.3	Accesso indicizzato ai blocchi dati.....	10-10
10.3.4	Accesso strutturato ai blocchi dati.....	10-11
11	Espressioni, operazioni e operandi.....	11-1
11.1	Espressioni, operazioni e operandi.....	11-1
11.2	Operazioni.....	11-2
11.3	Operandi.....	11-3
11.4	Sintassi di un'espressione.....	11-5
11.5	Espressione semplice.....	11-7
11.6	Espressioni aritmetiche.....	11-8
11.7	Espressioni logiche.....	11-10
11.8	Espressioni di confronto.....	11-11
12	Istruzioni.....	12-1
12.1	Assegnazione di valori.....	12-1
12.1.1	Assegnazione di valori con variabili di tipo di dati semplici.....	12-2
12.1.2	Assegnazione di valori con variabili di tipo STRUCT e UDT.....	12-3
12.1.3	Assegnazione di valori con variabili di tipo ARRAY.....	12-5
12.1.4	Assegnazione di valori con variabili di tipo STRING.....	12-7
12.1.5	Assegnazione di valori con variabili di tipo DATE_AND_TIME.....	12-8
12.1.6	Assegnazione di valori con variabili assolute per aree di memoria.....	12-9
12.1.7	Assegnazione di valori con variabili globali.....	12-10
12.2	Istruzioni di controllo.....	12-12
12.2.1	Istruzioni di controllo.....	12-12
12.2.2	Condizioni.....	12-13
12.2.3	Istruzione IF.....	12-14
12.2.4	Istruzione CASE.....	12-15
12.2.5	Istruzione FOR.....	12-17
12.2.6	Istruzione WHILE.....	12-20
12.2.7	Istruzione REPEAT.....	12-21
12.2.8	Istruzione CONTINUE.....	12-22
12.2.9	Istruzione EXIT.....	12-23
12.2.10	Istruzione GOTO.....	12-24
12.2.11	Istruzione RETURN.....	12-25
12.3	Richiamo di funzioni e blocchi funzionali.....	12-26
12.3.1	Richiamo e trasferimento di parametri.....	12-26
12.3.2	Richiamo di blocchi funzionali.....	12-28
12.3.2.1	Richiamo di blocchi funzionali (FB o SFB).....	12-28
12.3.2.2	Assegnazione dei parametri FB.....	12-30
12.3.2.3	Assegnazione d'ingresso (FB).....	12-31
12.3.2.4	Assegnazione di ingresso/uscita (FB).....	12-32
12.3.2.5	Lettura di valori iniziali (richiamo FB).....	12-33
12.3.2.6	Esempio di richiamo come istanza globale.....	12-33
12.3.2.7	Esempio di richiamo come istanza locale.....	12-35

12.3.3	Richiamo di funzioni	12-36
12.3.3.1	Richiamo di funzioni (FC)	12-36
12.3.3.2	Valore di ritorno (FC)	12-37
12.3.3.3	Parametri FC	12-38
12.3.3.4	Assegnazione d'ingresso (FC)	12-39
12.3.3.5	Assegnazione di uscita e di ingresso/uscita (FC)	12-40
12.3.3.6	Esempio di richiamo di funzione	12-41
12.3.4	Parametri con definizione implicita	12-42
12.3.4.1	Parametro di ingresso EN	12-42
12.3.4.2	Parametro di uscita ENO	12-43
13	Contatori e temporizzatori	13-1
13.1	Contatori	13-1
13.1.1	Funzioni di conteggio	13-1
13.1.2	Richiamo di funzioni di conteggio	13-1
13.1.3	Assegnazione di parametri nelle funzioni di conteggio	13-3
13.1.4	Introduzione ed analisi del valore del contatore	13-4
13.1.5	Conteggio in avanti (S_CU)	13-5
13.1.6	Conteggio all'indietro (S_CD)	13-5
13.1.7	Conteggio in avanti e all'indietro (S_CUD)	13-6
13.1.8	Esempio di funzione di conteggio	13-6
13.2	Temporizzatori	13-8
13.2.1	Funzioni temporali	13-8
13.2.2	Richiamo di funzioni temporali	13-8
13.2.3	Assegnazione di parametri nelle funzioni temporali	13-10
13.2.4	Introduzione ed analisi del valore temporale	13-11
13.2.5	Avvia temporizzatore come impulso (S_PULSE)	13-13
13.2.6	Avvia temporizzatore come impulso prolungato (S_PEXT)	13-14
13.2.7	Avvia temporizzatore come ritardo all'inserzione (S_ODT)	13-15
13.2.8	Avvia temporizzatore come ritardo all'inserzione con memoria (S_ODTS)	13-16
13.2.9	Avvia temporizzatore come ritardo alla disinserzione (S_OFFDT)	13-17
13.2.10	Esempio di funzione di temporizzazione	13-18
13.2.11	Selezione del temporizzatore giusto	13-19
14	Funzioni standard di S7-SCL	14-1
14.1	Funzioni di conversione dei tipi di dati	14-1
14.1.1	Conversione dei tipi di dati	14-1
14.1.2	Conversione implicita del tipo di dati	14-2
14.1.2.1	Funzioni di conversione di classe A	14-2
14.1.3	Funzioni standard per la conversione esplicita di tipi di dati	14-3
14.1.3.1	Funzioni di conversione di classe B	14-3
14.1.3.2	Funzioni per arrotondare e tagliare	14-6
14.1.3.3	Esempi di conversione con le funzioni standard	14-7
14.2	Funzioni aritmetiche standard	14-9
14.2.1	Funzioni aritmetiche standard generiche	14-9
14.2.2	Funzioni logaritmiche	14-9
14.2.3	Funzioni trigonometriche	14-10
14.2.4	Esempi di funzioni numeriche standard	14-10
14.3	Funzioni standard su stringhe di bit	14-11
14.3.1	Esempi di funzioni standard su stringhe di bit	14-12
14.4	Funzioni per l'elaborazione di stringe	14-13
14.4.1	Funzioni per la gestione delle stringhe	14-13
14.4.2	Funzioni per il confronto delle stringhe	14-17
14.4.3	Funzioni per la conversione del formato dei dati	14-19
14.4.4	Esempio di elaborazione di stringhe di caratteri	14-21
14.5	Funzioni di selezione dei valori	14-23
14.5.1	Funzioni di selezione dei valori	14-23

14.6	SFC, SFB e biblioteca standard.....	14-27
14.6.1	Funzioni, blocchi funzionali di sistema e biblioteca standard.....	14-27
14.6.2	Interfaccia di trasferimento agli OB	14-29
15	Descrizione del linguaggio di programmazione.....	15-1
15.1	Descrizione formale.....	15-1
15.1.1	Diagrammi sintattici	15-1
15.1.2	Regole	15-2
15.1.3	Terminali delle regole lessicali	15-4
15.1.4	Caratteri di formattazione, separatori e operazioni	15-5
15.1.5	Parole chiave e identificatori predefiniti.....	15-8
15.1.6	Identificazioni di operandi e parole chiave di blocchi	15-11
15.1.7	Non-terminali	15-12
15.1.8	Token.....	15-12
15.1.9	Identificatori	15-13
15.1.10	Assegnazione di nomi in S7-SCL.....	15-14
15.1.11	Costanti predefinite e flag.....	15-17
15.2	Regole lessicali	15-18
15.2.1	Regole lessicali	15-18
15.2.2	Identificatori	15-18
15.2.3	Costanti	15-20
15.2.4	Indirizzamento assoluto.....	15-25
15.2.5	Commenti	15-27
15.2.6	Attributi di blocco	15-28
15.2.7	Opzioni di compilazione	15-29
15.3	Regole sintattiche.....	15-30
15.3.1	Regole sintattiche.....	15-30
15.3.2	Suddivisione delle sorgenti S7-SCL.....	15-31
15.3.3	Struttura della parte convenzioni.....	15-33
15.3.4	Tipi di dati in S7-SCL.....	15-37
15.3.5	Parte istruzioni.....	15-40
15.3.6	Assegnazione di valori	15-42
15.3.7	Richiamo di funzioni e blocchi funzionali.....	15-45
15.3.8	Istruzioni di controllo.....	15-47
16	Suggerimenti e strategie.....	16-1

Glossario

Indice analitico

1 Presentazione del prodotto

1.1 Campo di applicazione di S7-SCL

L'S7-SCL (Structured Control Language) è un linguaggio avanzato simile al PASCAL concepito per la programmazione di controllori a logica programmabile mediante SIMATIC S7.

Certificato PLCopen

Il linguaggio S7-SCL risponde ai requisiti del linguaggio ST (Testuale Strutturato) definito nella norma IEC 61131-3 ed è stato predisposto per la certificazione relativa al Reusability Level.

Campo di applicazione

S7-SCL rappresenta uno strumento ottimale per la programmazione dei controllori a logica programmabile e contiene sia elementi di linguaggio del linguaggio di programmazione PASCAL sia elementi PLC tipici quali ingressi, uscite, temporizzatori e contatori.

S7-SCL è particolarmente indicato per assolvere alle seguenti funzioni:

- programmazione di algoritmi complessi
- programmazione di funzioni matematiche
- gestione di dati e ricette
- ottimizzazione del processo

1.2 Vantaggi di S7-SCL

S7-SCL offre tutti i vantaggi dei linguaggi di programmazione avanzati e presenta delle caratteristiche concepite espressamente per il supporto della programmazione strutturata come descritto di seguito.

Biblioteche dei blocchi

I blocchi predefiniti vengono depositati in biblioteche, ad es.:

- funzioni di sistema
- funzioni IEC
- funzioni di conversione

Una finestra di dialogo supporta la navigazione nella biblioteca. Selezionando un blocco lo schema dei parametri della funzione viene copiato automaticamente nel file elaborato. A questo punto occorre soltanto inserire i parametri desiderati.

Modelli per il programma

L'editor S7-SCL offre diversi modelli da inserire che devono solo essere compilati:

- modelli per blocchi (ad es. blocchi funzionali, blocchi dati) e i relativi richiami
- modelli per commenti di blocco, parametri di blocco e costanti
- modelli per strutture di controllo (IF, CASE, FOR, WHILE, REPEAT)

Elementi del linguaggio contenuti nella programmazione avanzata

Creazione semplice, rapida e difficilmente fallibile di programmi grazie all'impiego di costrutti linguistici efficaci, ad es.:

- loop di esecuzione
- diramazioni alternative (IF ... THEN ... ELSE)
- salti

Immediatezza del programma

Le seguenti caratteristiche relative all'elaborazione facilitano la leggibilità del programma:

- programmazione basata interamente su simboli,
- commenti,
- tipi di dati semplici e autodefiniti,
- visualizzazione di riferimenti incrociati,
- formattazione automatica dell'inserimento tramite allineamento,
- colorazione degli elementi di linguaggio in base alla sintassi,

Debugger a livello di linguaggio avanzato

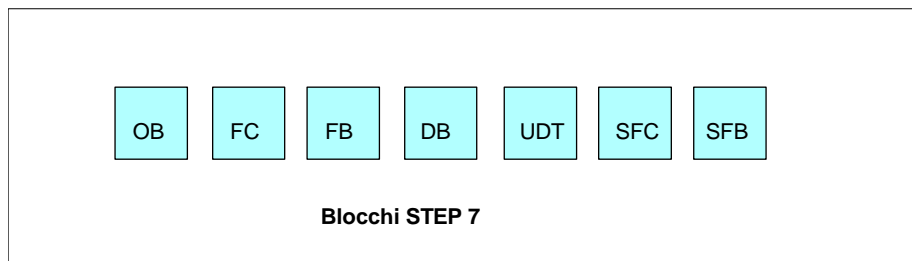
Il debugger consente di testare il programma in un modo molto semplice, offrendo le seguenti funzionalità:

- controllo regolare dello svolgimento del programma,
- controllo per fasi con l'ausilio di punti d'arresto impostabili individualmente,
- funzionalità Step-in (possibilità di accedere ai blocchi richiamati durante l'esecuzione del test)

1.3 Modalità operativa di S7-SCL

Integrazione in STEP 7

S7-SCL supporta il concetto di blocco di STEP 7.



È possibile creare i seguenti blocchi di STEP 7 mediante S7-SCL:

- OB
- FC
- FB
- DB
- UDT

In un programma S7 i blocchi di S7-SCL possono essere combinati anche con blocchi di altri linguaggi di programmazione STEP 7. Questi blocchi si possono richiamare reciprocamente. È possibile salvare i blocchi di S7-SCL anche nelle biblioteche e utilizzarli, recuperandoli da quella sede, in altri linguaggi.

Poiché i programmi di S7-SCL sono programmati come sorgenti ASCII, sono facilmente importabili ed esportabili.

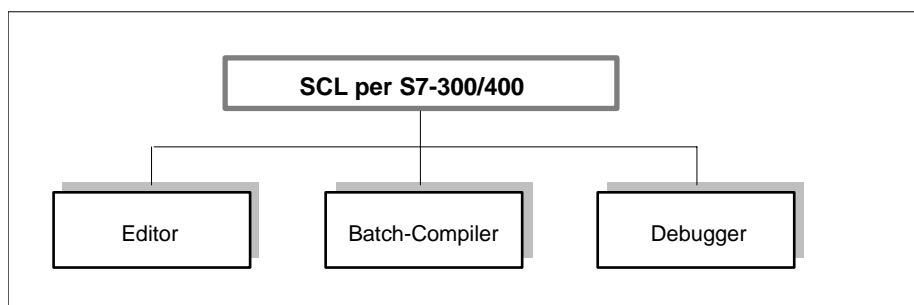
I blocchi di S7-SCL possono essere ricompilati nel linguaggio di programmazione di STEP 7 AWL (lista istruzioni). Occorre tuttavia tenere presente che, dopo un'unica operazione di salvataggio in AWL, non è più possibile eseguire elaborazioni in S7-SCL.

Ambiente di sviluppo

Per un pratico impiego S7-SCL offre un ambiente di sviluppo dalle ottime prestazioni, adatto sia alle caratteristiche specifiche di S7-SCL sia a STEP 7. I componenti di tale ambiente sono:

- Un **editor** per la programmazione di programmi costituiti da funzioni (FC), blocchi funzionali (FB), blocchi organizzativi (OB), blocchi dati (DB) e tipi di dati definiti dall'utente (UDT). Il lavoro del programmatore è supportato da alcune potenti funzioni.
- Un **batch compiler** per la compilazione del programma editato in codice macchina MC7. Il codice MC7 creato è eseguibile in tutte le CPU del sistema di automazione S7-300/400 successive alla CPU 314.
- Un **debugger** per la ricerca degli errori logici di programmazione, che consente una compilazione senza errori. La ricerca degli errori viene eseguita nel linguaggio sorgente.

La figura seguente presenta una panoramica dei componenti dell'ambiente di sviluppo:



1.4 Nuove funzioni della versione V5.3 SP1

Ampliamenti nell'ambito del linguaggio

In S7-SCL V5.3 SP1 sono stati potenziati gli strumenti del linguaggio che definiscono la norma IEC 61131-3:

- Funzioni per l'elaborazione dei valori numerici come funzioni interne S7-SCL (SEL, MAX, MIN, LIMIT, MUX);
- Supporto della rappresentazione BCD di numeri interi mediante funzioni di conversione (BCD_TO_INT, INT_TO_BCD, etc.);
- Operatore di assegnazione => per i parametri di uscita delle funzioni;
- Inizializzazione di array con parentesi;
- Nuove funzioni di conversione (BYTE_TO_INT; INT_TO_BYTE, etc.).

Impostazioni del compilatore nella sorgente

Le impostazioni del compilatore possono essere memorizzate nelle sorgenti S7-SCL oppure nei file di compilazione. In questo modo è possibile memorizzare le proprietà di una compilazione specificatamente per la sorgente.

Funzioni di test ampliate

- È possibile determinare e correggere le incoerenze dei blocchi e i conflitti di data e ora nei blocchi di S7-SCL tramite la funzione di test di STEP 7 "Verifica coerenza blocchi". Questa funzione di test è disponibile a partire dalla versione 5.3 SP2 di STEP 7;
- La funzione di test "Controlla" può essere impostata nel punto desiderato mediante la definizione dell'ambiente di richiamo;
- L'area di controllo relativa alla funzione di test "Controlla" può essere delimitata direttamente selezionando una sezione nella sorgente.

Stampa a colori

Le sorgenti S7-SCL possono essere stampate a colori.

Funzione di ricerca ampliata

S7-SCL ora consente anche di cercare la posizione del cursore verso l'alto e di effettuare ricerche all'interno di una selezione.

Posizionamento dei segnalibro nel testo sorgente

Mediante i segnalibro è possibile navigare rapidamente all'interno di una sorgente.

Creazione di blocchi di S7-SCL con set di caratteri derivanti da lingue straniere

È possibile che le sorgenti S7-SCL contengano set di caratteri derivanti da lingue straniere. In questo modo è possibile creare blocchi per il mercato internazionale nei quali i componenti principali visibili all'utente appaiano con set di caratteri in lingua straniera (ad es. nomi simbolici di blocchi, attributi e commenti).

Per ulteriori informazioni relative ai set di caratteri in lingua straniera, consultare il file Leggimi.

2 Installazione

2.1 Automation License Manager

2.1.1 Concessione della licenza d'utilizzo mediante Automation License Manager

Automation License Manager

Per l'utilizzo del software di programmazione è necessaria una chiave di licenza (licenza d'utilizzo) specifica per il prodotto, la cui installazione, a partire della versione V5.3, viene eseguita mediante Automation License Manager.

L'Automation License Manager è un software Siemens valido per tutti i sistemi operativi e necessario per la gestione delle chiavi di licenza (rappresentanti tecnici delle licenze).

L'Automation License Manager si trova:

- sul supporto di installazione de STEP 7
- come download da scaricare dalla pagina Internet A&D Customer Support della Siemens.

Nell'Automation License Manager è integrata una Guida online che, dopo l'installazione, può essere richiamata mediante il tasto F1 (per la parte sensibile al contesto), oppure mediante il comando di menu **? > Guida a License Manager**. Tale Guida contiene informazioni dettagliate sull'Automation License Manager.

Licenze

Per l'utilizzo dei pacchetti software di STEP 7 è necessario disporre della relativa licenza. La licenza concede il diritto d'uso di tali pacchetti ed è rappresentata dai seguenti elementi:

- CoL (**C**ertificate **o**f **L**icense) e
- Chiavi di licenza.

Certificate of License (CoL)

Il "Certificate of License" compreso nella fornitura dei prodotti software è la dimostrazione giuridica del diritto di utilizzo. Il prodotto può essere usato solo dal possessore del CoL o da una persona da lui incaricata.

Chiave di licenza

La chiave di licenza è il rappresentante tecnico della licenza (timbro di licenza elettronico).

Per ogni software protetto da licenza la Siemens assegna una chiave di licenza. Quando il software viene avviato nel computer, viene verificata la presenza di una chiave di licenza valida e il software può essere utilizzato nel rispetto delle condizioni di licenza e di utilizzo.

Avvertenze

- È possibile testare brevemente la superficie operativa e le funzioni del software anche senza chiave di licenza.
- L'utilizzo senza limitazioni, nel rispetto delle condizioni di licenza, è possibile ed ammesso solo in presenza di una chiave di licenza installata.
- Se la chiave di licenza **non** è installata, l'utente viene invitato ad intervalli regolari a provvedere all'installazione.

Le chiavi di licenza possono essere memorizzate e trasferite mediante i seguenti supporti:

- dischetti contenenti le chiavi di licenza
- dischi rigidi locali
- drive di rete.

Per ulteriori informazioni sulla gestione delle chiavi di licenza consultare la Guida online relativa all'Automation License Manager.

Tipi di licenze

Per i prodotti software della Siemens si distinguono i seguenti tipi di licenze orientate all'applicazione. Il comportamento del software è determinato dalle chiavi di licenza relative ai diversi tipi di licenze. Il tipo di utilizzo deriva dallo specifico Certificate of License.

Tipo di licenza	Descrizione
Single License	Il diritto di utilizzo del software vale per un periodo illimitato su un qualsiasi computer.
Floating License	Diritto di utilizzo in rete (utilizzo "remoto") illimitato nel tempo.
Trial License	Il diritto di utilizzo del software è limitato: <ul style="list-style-type: none"> • ad un massimo di 14 giorni, • al numero di giorni dopo il primo utilizzo indicato nel contratto, • a test e per provarne la validità (esclusione di responsabilità).
Rental License	L'utilizzo del software è vincolato a: <ul style="list-style-type: none"> • una validità di max.50 giorni • un determinato numero di ore di utilizzo
Upgrade License	Per un aggiornamento possono essere richiesti determinati requisiti di sistema: <ul style="list-style-type: none"> • una licenza di aggiornamento permette l'aggiornamento dalla "vecchia" versione x alla nuova versione >x+... • l'aggiornamento della licenza può essere p. es. necessario in caso di ampliamento della funzionalità.

2.1.2 Installazione dell'Automation License Manager

L'Automation License Manager viene installato mediante un setup. Il software di installazione dell'Automation License Manager si trova sul CD ROM di STEP 7.

L'Automation License Manager può essere installato contestualmente a S7-SCL oppure in un secondo momento.

Avvertenze

- Per informazioni dettagliate sul modo di procedere nell'installazione dell'Automation License Manager, consultare il file Leggimi.wri dell'Automation License Manager.
 - La Guida online dell'Automation License Manager contiene informazioni complete sul funzionamento e la gestione delle chiavi di licenza.
-

Installazione della chiave di licenza a posteriori

Se si avvia S7-SCL e non sono presenti chiavi di licenza, il programma visualizza un messaggio.

Avvertenze

- È possibile testare brevemente la superficie operativa e le funzioni del software di base STEP 7 anche senza chiave di licenza.
 - L'utilizzo senza limitazioni, nel rispetto delle condizioni di licenza, è possibile ed ammesso solo in presenza di una chiave di licenza installata.
 - Se la chiave di licenza **non** è installata, l'utente viene invitato ad intervalli regolari a provvedere all'installazione.
-

È possibile installare la chiave di licenza a posteriori nei seguenti modi:

- da dischetti
- scaricandole dal Web (previa ordinazione)
- utilizzando chiave di licenza "Floating" presenti in rete.

Per informazioni complete sul modo di procedere, consultare la Guida online dell'Automation License Manager che, dopo l'installazione, può essere richiamata mediante il tasto F1 (per la parte sensibile al contesto), oppure mediante il comando di menu ? > **Guida a License Manager**.

Avvertenze

- In Windows 2000/XP le chiavi di licenza funzionano solo se salvate su un disco rigido accessibile in scrittura.
 - Le Floating License possono essere utilizzate anche in rete (ossia in modalità remota).
-

2.1.3 Regole per l'utilizzo delle chiavi di licenza



Attenzione

Si tengano presenti le avvertenze sull'utilizzo delle chiavi di licenza contenute nella Guida online e nel file Leggimi.wri dell'Automation License Manager. Diversamente, le chiavi di licenza potrebbero andare perdute irrimediabilmente.

La Guida online dell'Automation License Manager può essere richiamata mediante il tasto F1 (per la parte sensibile al contesto), oppure mediante il comando di menu **? > Guida a License Manager**. Tale Guida contiene informazioni dettagliate sull'Automation License Manager.

2.2 Installazione

2.2.1 Requisiti della installazione

Requisiti del sistema

Il pacchetto opzionale S7-SCL V5.3 SP1 è eseguibile in un PG/PC con l'installazione del pacchetto di base STEP 7 V5.3 o versione superiore.

I requisiti relativi al sistema operativo sono reperibili nel file Leggimi.wri.

Requisiti hardware

I requisiti necessari per poter utilizzare S7-SCL sono gli stessi del pacchetto base di STEP 7. Lo spazio di memoria richiesto dal pacchetto opzionale S7-SCL V5.3 SP1 è indicato nel file Leggimi.wri.

2.2.2 Installazione di S7-SCL

Avvio del programma di installazione

S7-SCL contiene un programma che esegue l'installazione automaticamente. Sullo schermo compaiono richieste di introduzione dati che guidano l'utente passo passo nel corso dell'intera installazione.

Procedura

1. In Windows avviare l'installazione del software facendo doppio clic sull'icona "Installazione applicazioni" del Pannello di controllo.
2. Fare clic su "Installa".
3. Inserire il supporto dati e fare clic su "Avanti". Windows cerca il programma di installazione "Setup.exe".
4. Seguire passo passo le istruzioni visualizzate dal programma di installazione.

Installazione delle chiavi di licenza

Durante l'installazione viene verificata la presenza sul disco rigido della chiave di licenza necessaria. Se non viene individuata alcuna chiave di licenza valida, un messaggio informa che il software può essere utilizzato solo in presenza della chiave di licenza. Questa può essere installata subito oppure a posteriori una volta completata l'installazione; nel primo caso introdurre al prompt il dischetto contenente la chiave di licenza compresa nella fornitura.

3 Ideazione di un programma S7-SCL

3.1 Esempio introduttivo "Rilevazione di un valore di misura"

Obiettivo dell'esempio

Questo esempio introduttivo spiega come utilizzare S7-SCL nel modo più efficiente. Le domande più frequenti all'inizio possono essere:

- Come si progetta un programma con S7-SCL?
- Quali strumenti del linguaggio S7-SCL sono utilizzabili per la soluzione del problema?
- Di quali funzioni di test posso disporre?

Queste ed altre domande vengono trattati in questo capitolo.

Strumenti del linguaggio S7-SCL utilizzati

Nell'esempio riportato vengono descritti fra l'altro i seguenti elementi del linguaggio S7-SCL:

- Struttura e impiego dei vari tipi di blocchi S7-SCL
- Richiamo dei blocchi con trasferimento e analisi dei parametri
- Vari formati d'ingresso/uscita
- Programmazione con tipi di dati semplici e campi
- Inizializzazione delle variabili
- Strutture del programma con impiego di diramazioni e loop

Requisiti hardware

Il programma esempio può essere eseguito con SIMATIC S7-300 o SIMATIC S7-400; a tal fine è necessaria la seguente periferia:

- un'unità d'ingresso con 16 canali
- un'unità di uscita con 16 canali

Funzioni di test utilizzate

Il programma è concepito in modo tale da poter essere rapidamente testato tramite gli interruttori sull'unità d'ingresso e gli elementi di visualizzazione sull'unità di uscita. Per l'esecuzione di un test dettagliato, utilizzare le funzioni di test di S7-SCL.

Oltre ai vantaggi del linguaggio, l'utente ha a disposizione di tutte le funzioni che offre il pacchetto STEP 7.

3.2 Impostazione del problema

Panoramica

I valori di misura vengono rilevati tramite un'unità di ingresso e quindi disposti ed elaborati mediante un programma S7-SCL. I risultati vengono visualizzati tramite un'unità di uscita,

Rilevazione dei valori di misura

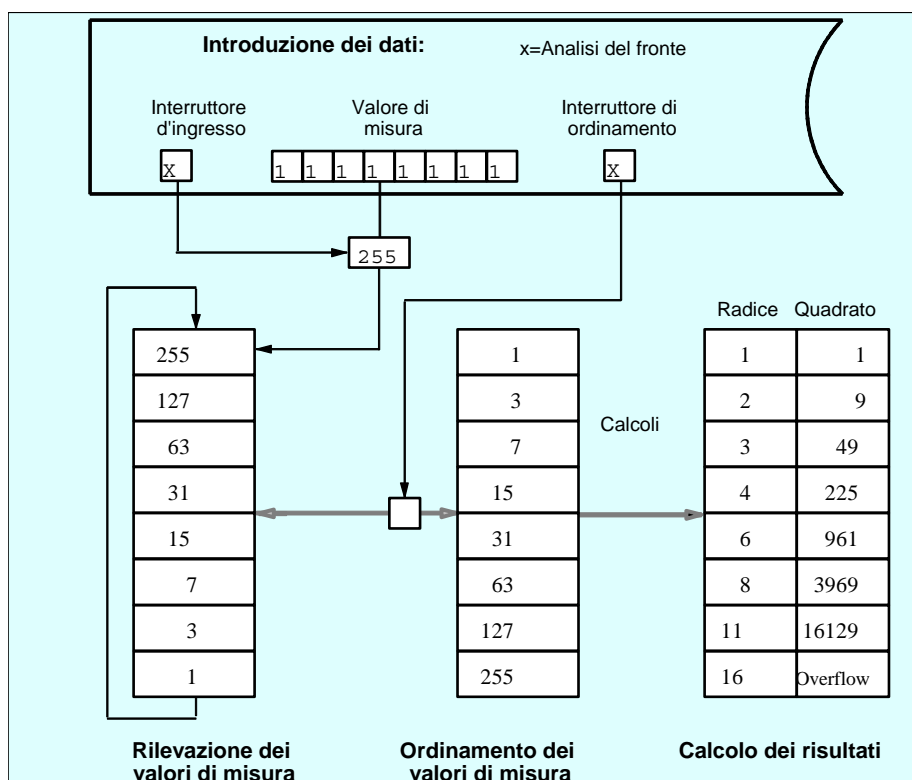
Il valore di misura viene impostato mediante 8 interruttori d'ingresso. Quando in un interruttore di ingresso viene rilevato un fronte, il valore viene trasferito in un campo apposito della memoria (vedi la figura seguente).

Il campo dei valori di misura è compreso fra 0 e 255. L'immissione richiede quindi un byte.

Elaborazione dei valori di misura

Il campo dei valori di misura deve essere organizzato come buffer circolare con 8 registrazioni al massimo.

Quando su un interruttore di ordinamento viene riconosciuto un fronte, i valori memorizzati nel campo valori di misura devono essere ordinati in ordine ascendente. Quindi, per ciascun valore si devono calcolare la radice e il quadrato. Le funzioni di elaborazione occupano una parola.



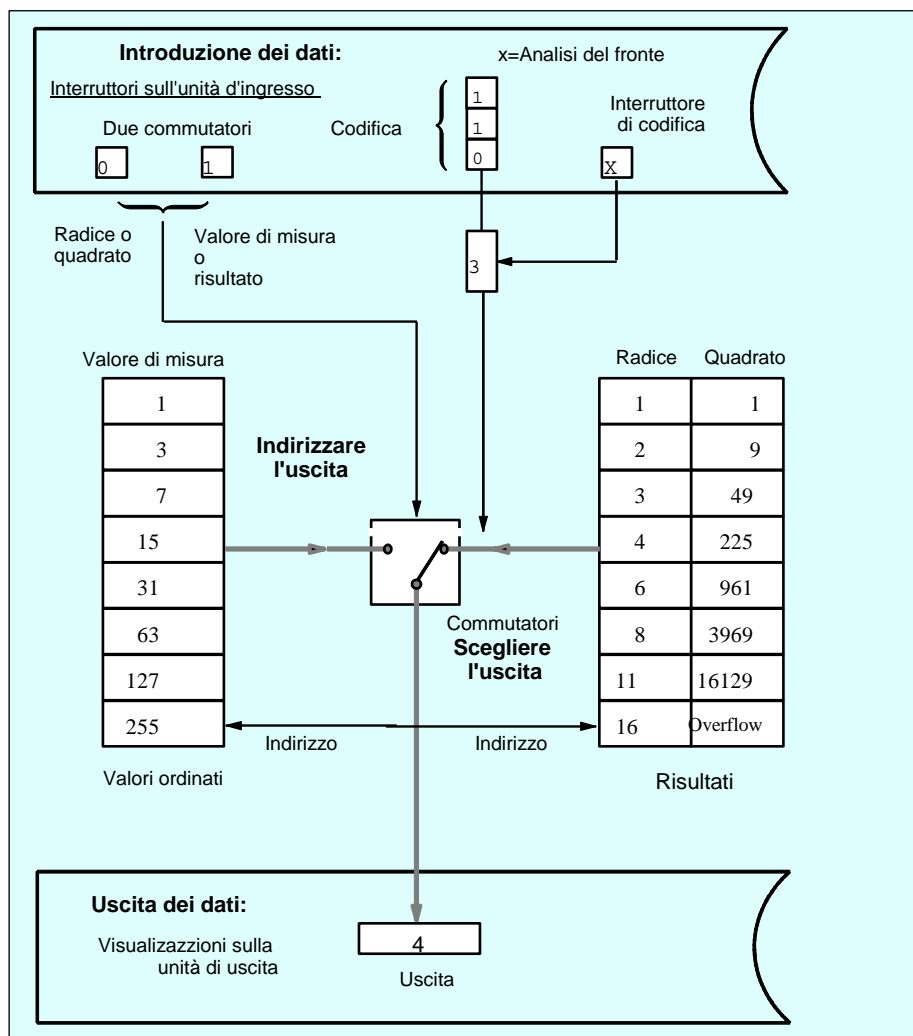
Uscite impostabili

Poiché nell'unità di uscita è possibile visualizzare solo un valore per volta sono ammesse le seguenti possibilità:

- selezione dell'elemento di una lista
- selezione di un valore di misura, una radice o un quadrato.

La selezione del valore visualizzato viene effettuata nel seguente modo:

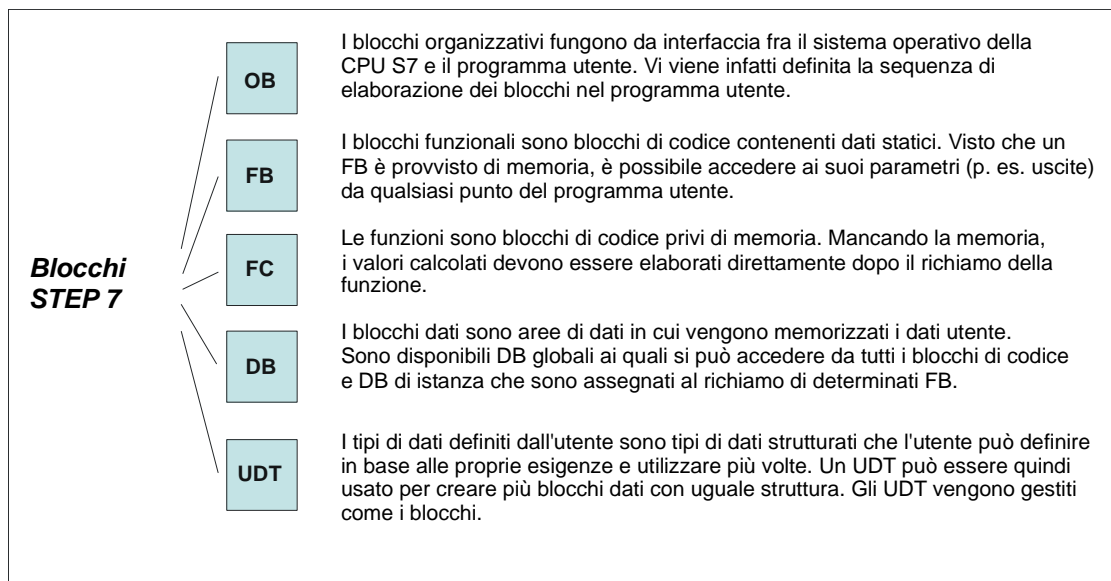
- Tramite tre interruttori viene impostata una codifica che viene confermata se su un quarto interruttore, l'interruttore di codifica, viene riconosciuto un fronte. Viene quindi calcolato l'indirizzo usato per indirizzare l'uscita.
- Con lo stesso indirizzo vengono preparati per l'uscita tre valori: valore di misura, radice e quadrato. Per poter scegliere uno dei tre valori, si devono predisporre due commutatori.



3.3 Realizzazione di un programma strutturato con S7-SCL

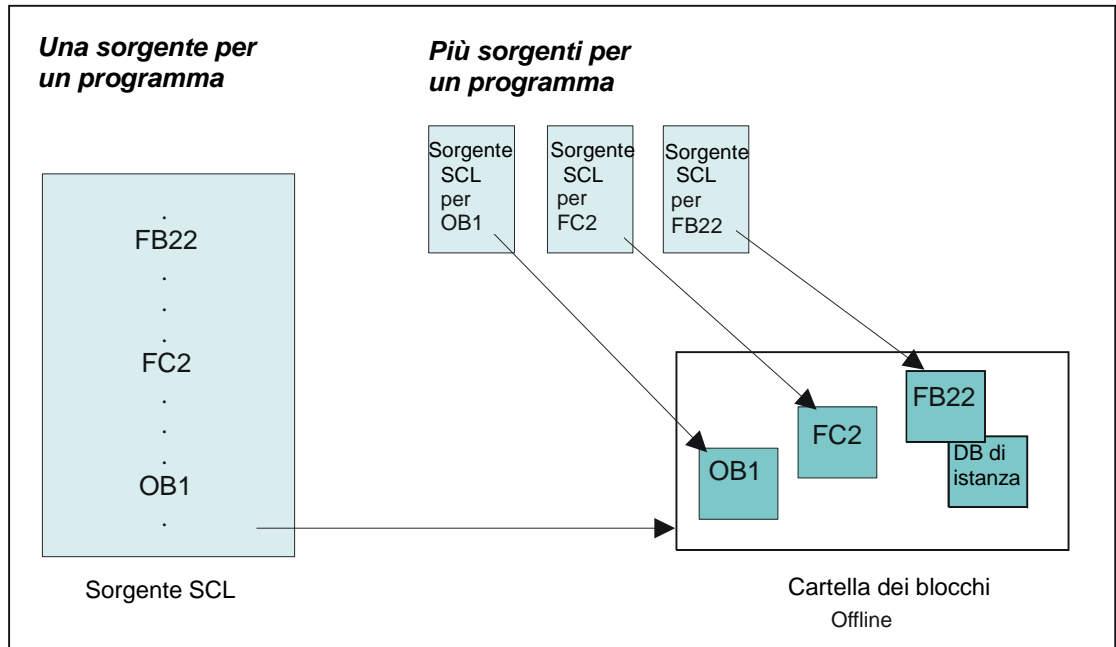
Tipi di blocco

Il modo migliore per risolvere il problema descritto è quello di utilizzare un **programma S7-SCL strutturato**. Questo programma ha una struttura modulare, cioè è suddiviso in blocchi che eseguono uno o più compiti parziali. In S7-SCL, così come negli altri linguaggi di programmazione di STEP 7, l'utente ha la scelta tra numerosi tipi di blocchi.



Disposizione dei blocchi nelle sorgenti S7-SCL

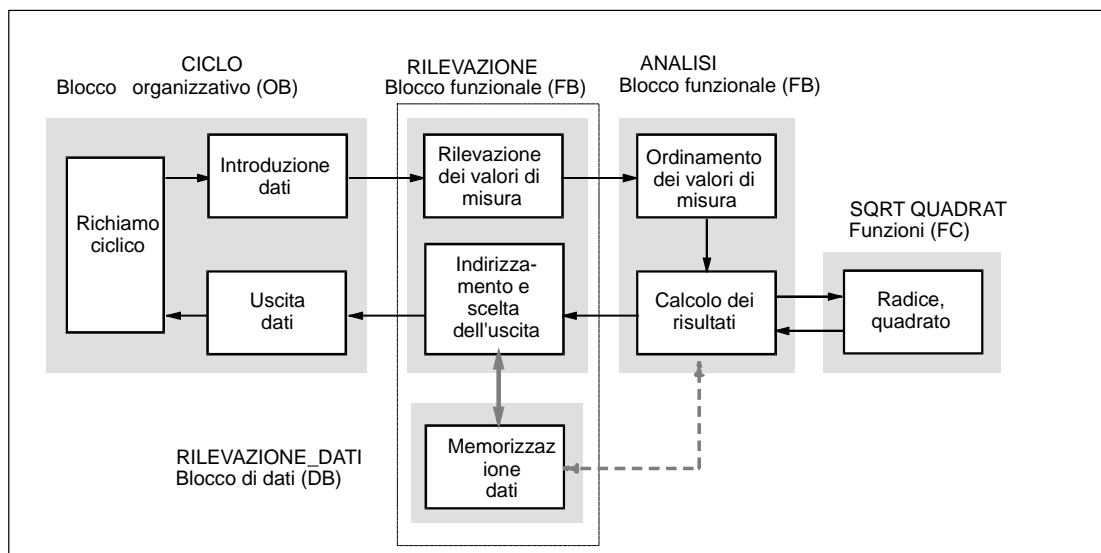
I programmi S7-SCL sono costituiti da una o più sorgenti S7-SCL. Una sorgente può contenere un unico blocco o un programma completo composto da più blocchi.



3.4 Definizione dei compiti parziali

Compiti parziali

I compiti parziali sono illustrati con caselle nella figura seguente. Le aree quadrate su sfondo grigio rappresentano i blocchi. La disposizione dei blocchi di codice da sinistra verso destra corrisponde alla sequenza di richiamo.



Scelta e attribuzione dei blocchi dati ai compiti parziali

In seguito vengono descritti i criteri con i quali è stata eseguita la scelta dei blocchi:

Funzione		Nome del blocco
I programmi utente possono essere avviati solo in OB. Poiché i valori devono essere presenti e rilevati in continuazione, è necessario un OB per il <i>richiamo ciclico</i> (OB1). Una parte dell'elaborazione viene programmata nell'OB: <i>Introduzione dati e Uscita dati</i> .	⇒	OB "Ciclo"
Per il compito parziale <i>Rilevazione valori di misura</i> è necessario un blocco con memoria cioè un FB, poiché determinati dati dei blocchi locali (p. es. il buffer circolare) devono essere conservati da un ciclo di programma all'altro. Il luogo per la <i>Memorizzazione dati</i> , detto anche memoria, è il blocco dati di istanza RILEVZIONE_DATI. Lo stesso FB può anche eseguire il compito parziale <i>Indirizzamento e scelta dell'uscita</i> , poiché esso dispone dei dati necessari.	⇒	FB "Rilevazione"
Per la scelta del tipo di blocco per risolvere i compiti parziali <i>Ordinamento dei valori di misura</i> e <i>Calcolo dei risultati</i> si deve tener presente che è necessario creare un buffer d'uscita, il quale per ogni valore di misura contiene i risultati dei calcoli della radice e del quadrato. Perciò, come blocco si può utilizzare solo un FB. Poiché l'FB viene richiamato da un FB sovraordinato, esso non ha bisogno di un DB proprio. I suoi dati di istanza possono essere depositati nel blocco dati di istanza dell'FB richiamante.	⇒	FB "Analisi"
Per risolvere il compito parziale <i>Calcolo della radice o del quadrato</i> la soluzione migliore è una FC poiché il ritorno del risultato può aver luogo come valore della funzione. Inoltre, per il calcolo non sono necessari dati che devono essere conservati più a lungo di un ciclo di elaborazione del programma. Per il calcolo della radice si può utilizzare <code>SQRT</code> , la funzione standard in S7-SCL. Per il calcolo del quadrato si deve creare una funzione <code>QUADRAT</code> , la quale deve eseguire anche un controllo dei valori limite del campo di valori.	⇒ ⇒	FC "SQRT" (radice quadrata) e FC "Quadrato"

3.5 Definizione delle interfacce fra i blocchi

Panoramica

L'interfaccia di un blocco viene realizzata mediante parametri ai quali è possibile accedere da altri blocchi.

I parametri dichiarati nel blocco sono "segnaposto" i cui valori vengono definiti quando il blocco viene utilizzato (richiamato). Questi segnaposti vengono denominati parametri formali, mentre i valori assegnati durante il richiamo del blocco vengono denominati parametri attuali. Quando viene richiamato un blocco, gli vengono trasferiti i dati d'ingresso come parametri attuali. Dopo il ritorno al blocco richiamante, i dati di uscita vengono preparati per la conferma. Una funzione (FC) può restituire il proprio risultato come valore di funzione.

I parametri di blocco possono essere suddivisi nelle seguenti categorie.

Parametri del blocco	Significato	Dichiarazione con
parametri d'ingresso	I parametri d'ingresso assumono i valori d'ingresso attuali durante il richiamo del blocco. Essi possono essere solo letti.	VAR_INPUT
Parametri d'uscita	I parametri d'uscita trasferiscono i valori d'uscita attuali al blocco richiamante. Essi possono essere letti e scritti.	VAR_OUTPUT
Parametri di ingresso/uscita	Al momento del richiamo del blocco, i parametri di ingresso/uscita assumono il valore attuale di una variabile, lo elaborano e infine depositano i risultati ottenuti nella stessa variabile.	VAR_IN_OUT

OB Ciclo

L'OB CICLO non possiede propri parametri formali. Esso richiama l'FB RILEVAZIONE e gli trasferisce il valore di misura e i dati di comando nei suoi parametri formali.

FB Rilevazione

Nome di parametro	Tipo di dati	Tipo di dichiarazione	Descrizione
valoremisura_inserito	INT	VAR_INPUT	Valore di misura
valorenuovo	BOOL	VAR_INPUT	Interruttore per confermare il valore di misura nel buffer circolare
ordinamentonuovo	BOOL	VAR_INPUT	Interruttore per confermare il valore di misura nel buffer circolare
sceltafunzione	BOOL	VAR_INPUT	Commutatore per scegliere radice o quadrato
scelta	WORD	VAR_INPUT	Codifica per scegliere il valore di uscita
sceltanuova	BOOL	VAR_INPUT	Interruttore per confermare la codifica
risultato_out	DWORD	VAR_OUTPUT	Uscita del risultato calcolato
valoremisura_out	DWORD	VAR_OUTPUT	Uscita del corrispondente valore di misura

Analisi

L'FB RILEVAZIONE richiama l'FB ANALISI. Entrambi hanno come dati in comune il campo valori di misura da ordinare. Perciò, esso viene dichiarato come parametro di ingresso/uscita. Per i risultati del calcolo della radice e del quadrato viene creato un campo strutturato come parametro di uscita. La seguente tabella illustra i parametri formali:

Nome	Tipo di dati	Tipo di dichiarazione	Descrizione
bufferordinamento	ARRAY[..] OF REAL	VAR_IN_OUT	Campo valori di misura, corrisponde al buffer circolare
buffercalcolo	ARRAY[..]OF STRUCT	VAR_OUTPUT	Campo per i risultati: Struttura con i componenti "radice" e "quadrato" del tipo INT

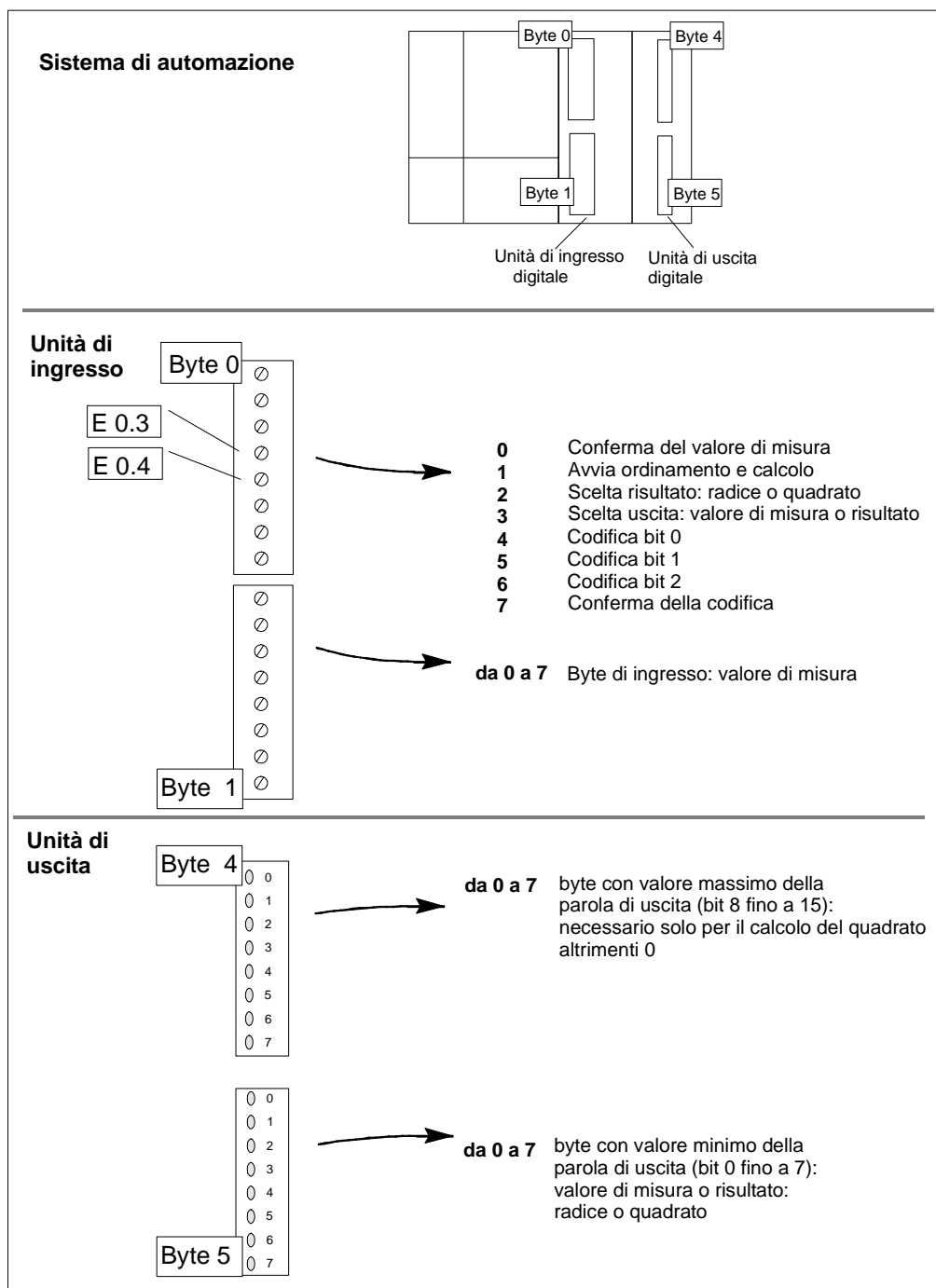
SQRT e Quadrato

Le funzioni vengono richiamate da ANALISI. Esse hanno bisogno di un valore d'ingresso e forniscono il loro risultato come valore della funzione.

Nome	Tipo di dati	Tipo di dichiarazione	Descrizione
valore	REAL	VAR_INPUT	Ingresso per SQRT
SQRT	REAL	Valore della funzione	Radice del valore d'ingresso
valore	INT	VAR_INPUT	Ingresso per QUADRATO
QUADRATO	INT	Valore della funzione	Quadrato del valore d'ingresso

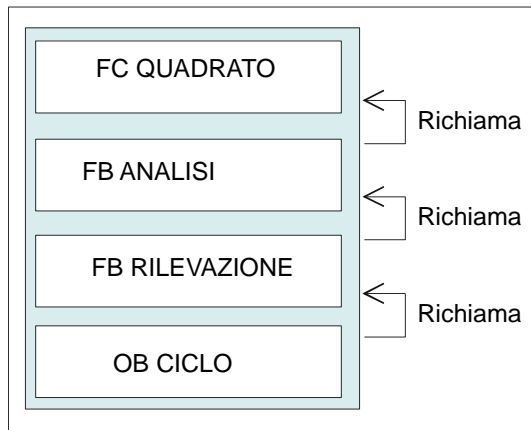
3.6 Definizione dell'interfaccia d'ingresso/uscita

La figura seguente illustra l'interfaccia d'ingresso/uscita. Si prega di tener presente che, per l'ingresso/uscita byte per byte il byte superiore ha valore minimo e il byte inferiore ha valore massimo. Si ha invece il contrario per l'ingresso/uscita parola per parola.



3.7 Definizione della sequenza dei blocchi nella sorgente

Riguardo ai blocchi della sorgente S7-SCL si deve tener conto del fatto che un blocco, per poter essere utilizzato (ovvero richiamato da un altro blocco), deve essere innanzitutto disponibile. I blocchi della sorgente S7-SCL devono essere quindi disposti nel seguente modo:



3.8 Definizione di simboli

La comprensione di un programma migliora notevolmente se per gli indirizzi dei blocchi e per i blocchi vengono assegnati nomi simbolici. A tal fine è necessario trascrivere le relative registrazioni nella tabella dei simboli.

La seguente figura rappresenta la tabella dei simboli dell'esempio in cui vengono stabiliti i nomi simbolici, presupposto indispensabile per la compilazione delle sorgenti.

	Simbolo	Indirizzo	Tipo di dati
1	ANALISI	FB 20	FB 20
2	CICLO	OB 1	OB 1
3	Codifica	EW 0	WORD
4	Ingresso 0.0	E 0.0	BOOL
5	Interruttore di codifica	E 0.7	BOOL
6	Interruttore di funzione	E 0.2	BOOL
7	Interruttore di ordinam.	E 0.1	BOOL
8	Interruttore di uscita	E 0.3	BOOL
9	Introduzione	EB 1	BYTE
10	quadrato	FC 41	FC 41
11	RILEVAZIONE	FB 10	FB 10
12	RILEVAZIONE_DATI	DB 10	FB 10
13	Uscita	AW 4	INT

3.9 Come creare la funzione QUADRATO

Parte istruzioni

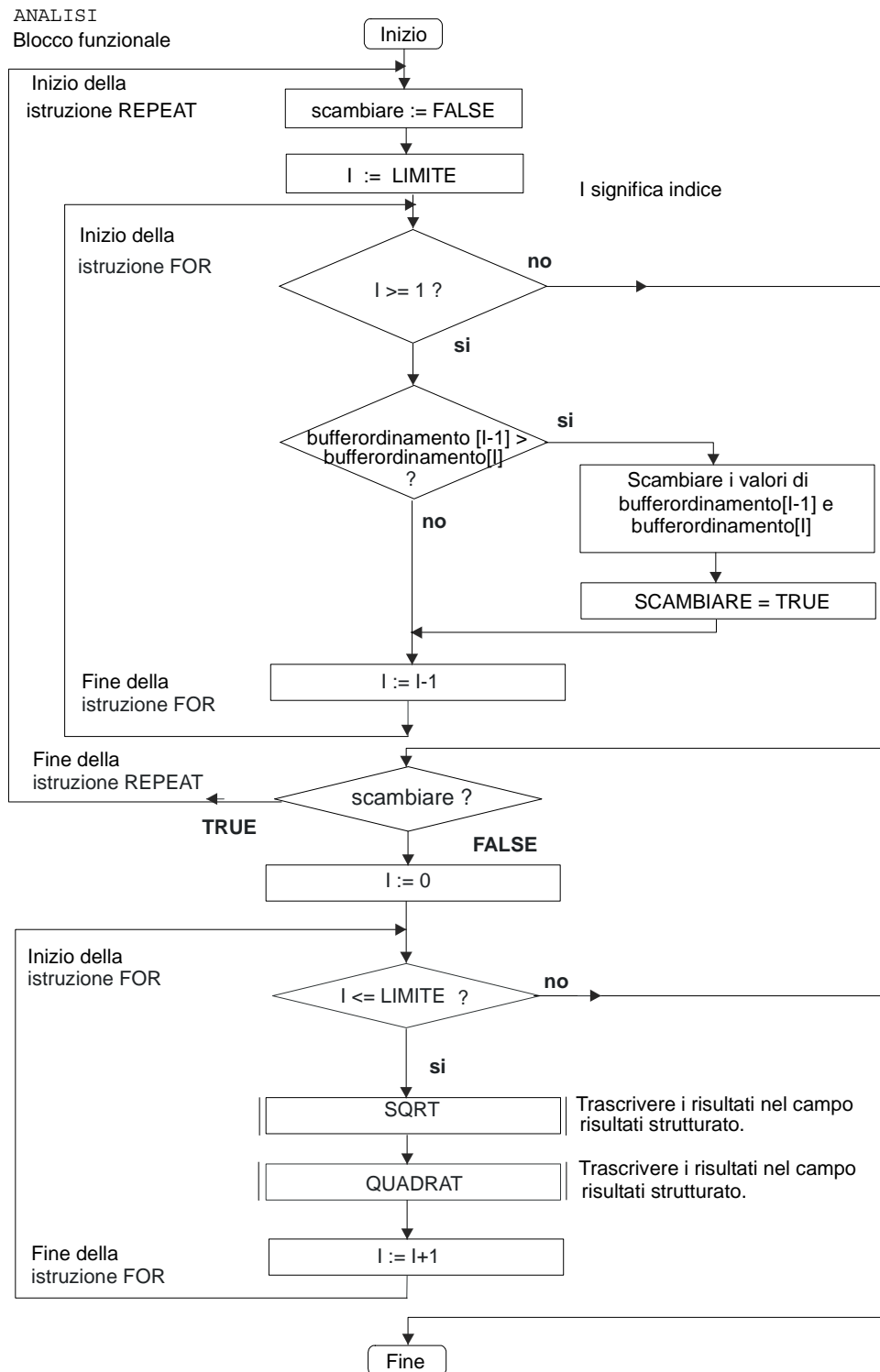
Dapprima si verifica se il valore d'ingresso supera il limite massimo consentito per il campo di conteggio di numeri interi. In tal caso, viene trascritto il valore massimo per i numeri interi. Altrimenti viene eseguita la funzione di calcolo del quadrato. Il risultato viene trasferito come valore della funzione.

```
FUNCTION QUADRATO : INT
(*****
Questa funzione fornisce come valore il quadrato del valore di
ingresso e in caso di overflow, il valore massimo rappresentabile
come Integer.
*****)
VAR_INPUT
    valore : INT;
END_VAR
BEGIN
IF valore <= 181 THEN
    quadrato := valore * valore; //Calcolo del valore della
funzione
ELSE
    quadrato := 32_767; // in caso di overflow impostare il valore
massimo
END_IF;
END FUNCTION
```

3.10 Come creare il blocco funzionale ANALISI

3.10.1 Diagramma di flusso di ANALISI

La figura illustra l'algoritmo in forma di un diagramma di flusso:



3.10.2 Parte dichiarazioni dell'FB ANALISI

Struttura della parte dichiarazioni

In questo blocco, la parte dichiarazioni si compone delle seguenti parti:

- Dichiarazione delle costanti: fra CONST e END_CONST
- Parametri di ingresso/uscita: fra VAR_IN_OUT e END_VAR
- Parametri di uscita: fra VAR_OUTPUT e END_VAR
- Dichiarazione delle variabili temporanee: fra VAR_TEMP e END_VAR

```
CONST
    LIMITE := 7;
END_CONST

VAR_IN_OUT
    bufferordinamento : ARRAY[0..LIMITE] OF INT;
END_VAR

VAR_OUTPUT
    buffercalcolo      : ARRAY[0..LIMITE] OF
        STRUCT
            radice      : INT;
            quadrato    : INT;
        END_STRUCT;
END_VAR

VAR_TEMP
    scambio             : BOOL;
    index, ausil        : INT;
    valore, risultato   : REAL; END_VAR
```

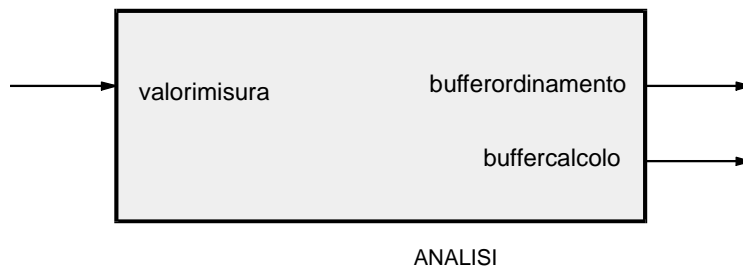

3.10.3 Parte istruzioni dell'FB ANALISI

Esecuzione del programma

Il parametro di ingresso/uscita "bufferordinamento" viene associato al buffer circolare "valorimisura", cioè il contenuto originale del buffer viene sovrascritto con l'ordinamento dei valori di misura.

Per i risultati di calcolo viene creato il campo "buffercalcolo" come parametro di uscita. I suoi elementi sono strutturati in modo da contenere radice e quadrato per ogni valore di misura.

La figura seguente descrive il nesso esistente fra i campi descritti.



Questa interfaccia illustra il nucleo dello scambio di dati per l'elaborazione dei valori di misura. I valori vengono memorizzati nel blocco dati di istanza `RILEVAZIONE_DATI`, poiché nell'FB `RILEVAZIONE` richiamante è stata creata un'istanza locale per l'FB `ANALISI`.

Parte istruzioni di ANALISI

Dapprima vengono ordinati i valori di misura nel buffer circolare e quindi vengono eseguiti i calcoli:

- Metodo dell'algoritmo per l'ordinamento
Per ordinare il buffer dei valori di misura viene impiegato il metodo di scambio permanente di valori, cioè due valori attigui vengono confrontati fra loro e scambiati fino ad ottenere la sequenza desiderata. Il buffer utilizzato è il parametro di ingresso/uscita "bufferordinamento".
- Avvio del calcolo
Dopo aver completato l'ordinamento, per effettuare i calcoli viene eseguito un loop in cui vengono richiamate la funzione `QUADRATO` per calcolare il quadrato e la funzione `SQRT` per calcolare la radice. I risultati ottenuti vengono memorizzati nel campo strutturato "buffercalcolo".

Parte istruzioni di ANALISI

La parte istruzioni del blocco di codice si presenta nel seguente modo:

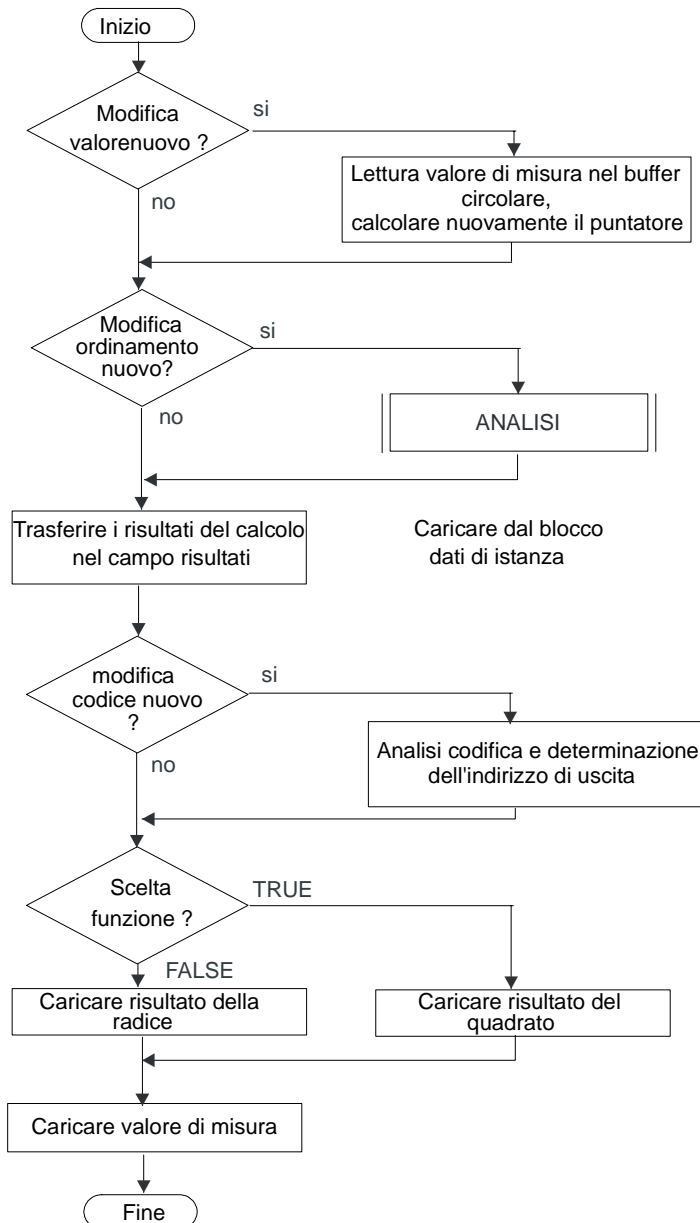
```
BEGIN
(*****
Parte 1 Ordinamento : in base al metodo "Bubble Sort" scambiare in
coppia i valori fino all'ordinamento del buffer valori di misura.
*****)
REPEAT
    scambiare := FALSE;
    FOR index := LIMITE TO 1 BY -1 DO
        IF bufferordinamento[index-1] > bufferordinamento[index]
            THEN ausil
:=bufferordinamento[index];
        bufferordinamento[index] :=
bufferordinamento[index-1];
        bufferordinamento[index-1] := ausil;
        scambiare := TRUE;
            END_IF;
        END_FOR;
    UNTIL NOT scambiare
END_REPEAT;
(*****
Parte 2 Calcolo : Calcolo della radice con la funzione standard
SQRT e calcolo del quadrato con la funzione QUADRATO.
*****)
FOR index := 0 TO LIMITE BY 1 DO
    valore := INT_TO_REAL(bufferordinamento[index]);
    risult := SQRT(valore);
    buffercalcolo[index].radice := REAL_TO_INT(risult);
    buffercalcolo[index].quadrato :=
QUADRATO(bufferordinamento[index]);
END_FOR;
END FUNCTION BLOCK
```

3.11 Come creare il blocco funzionale RILEVAZIONE

3.11.1 Diagramma di flusso di Rilevazione

La figura seguente illustra l'algoritmo in forma di un diagramma di flusso:

RILEVAZIONE
Blocco funzionale



Con l'operazione MOD viene realizzato il buffer ciclico: quando viene raggiunto il limite ricominciare da capo.

Ordinamento del buffer circolare ed esecuzione dei calcoli (creare a tal fine un campo risultati).

Spostare dapprima i bit rilevanti verso il margine destro, mascherare quindi con AND le posizioni non necessarie.

Caricare:
Scrivere gli elementi della lista con l'indirizzo di uscita nei parametri di uscita e visualizzare quindi i loro valori.

3.11.2 Parte dichiarazioni dell'FB RILEVAZIONE

Struttura della parte dichiarazioni

In questo blocco, la parte dichiarazioni si compone dei seguenti elementi:

- Dichiarazione delle costanti: fra CONST e END_CONST
- Parametro d'ingresso: fra VAR_INPUT e END_VAR
- Parametri di uscita: fra VAR_OUTPUT e END_VAR
- Variabili statiche: fra VAR e END_VAR
Fra queste vi è anche la dichiarazione dell'istanza locale per il blocco ANALISI.

```

CONST
    LIMITE := 7;
    NUMERO := LIMITE + 1;
END_CONST
VAR_INPUT
    valoremisura_in      : INT ; // Nuovo valore di misura
    valorenuevo          : BOOL; // Valore di misura nel
                                // buffer circolare "valorimisura"
    ordinamentonuovo     : BOOL; // Ordinamento dei valori di misura
    sceltafunzione       : BOOL; // Scelta della funzione
                                // di calcolo Radice/Quadrato
    sceltanuova          : BOOL; // Immissione indirizzo di uscita
    scelta               : WORD; // Indirizzo di uscita
END_VAR
VAR_OUTPUT
    risultato_out        : INT; // Valore calcolato
    valoremisura_out     : INT; // Corrispondente valore di misura
END_VAR
VAR
    valorimisura         : ARRAY[0..LIMITE] OF INT := 8(0);
    bufferrisultato      : ARRAY[0..LIMITE] OF
    STRUCT
        radice          : INT;
        quadrato        : INT;
    END_STRUCT;
    puntatore           : INT := 0;
    valore_vecchio      : BOOL := TRUE;
    ordinamentovecchio  : BOOL := TRUE;
    sceltavecchia       : BOOL := TRUE;
    indirizzo            : INT := 0; //Indirizzo di uscita
                                //convertito
    istanza_analisi: ANALISI; //Dichiarazione istanza
                                //locale
END VAR

```

Variabili statiche

Il tipo di blocco FB è stato scelto poiché esistono dati che devono essere salvati da un ciclo di programma all'altro. Si tratta delle variabili statiche che vengono dichiarate nella parte di dichiarazione "VAR, END_VAR".

Le variabili statiche sono variabili locali i cui valori vengono conservati per tutte le esecuzioni dei blocchi. Esse hanno la funzione di memorizzare i valori di un blocco funzionale, e vengono depositate nel blocco dati di istanza.

Inizializzazione delle variabili

Si prega di osservare i valori di impostazione, i quali vengono registrati nelle variabili durante l'inizializzazione del blocco (dopo il caricamento nella CPU). Nella parte di dichiarazione "VAR, END_VAR" viene dichiarata anche l'istanza locale per ANALISI_FB. Il nome viene utilizzato in seguito per il richiamo e per i parametri di uscita. Come memoria dati viene utilizzata l'istanza globale RILEVAZIONE_DATI.

Nome	Tipo di dati	Predefinizione	Descrizione
valorimisura	ARRAY [..] OF INT	8(0)	Buffer circolare per valori di misura
bufferrisultati	ARRAY[.. OF STRUCT	-	Campo per strutture con i componenti "radice" e "quadrato" del tipo INT
puntatore	INT	0	Indice per buffer circolare, registrazione del prossimo valore di misura
valorevecchio	BOOL	FALSE	Valore precedente per conferma valore di misura con "valore nuovo"
ordinamentovecchio	BOOL	FALSE	Valore precedente per ordinamento con "ordinamentonuovo"
sceltavecchia	BOOL	FALSE	Valore precedente per conferma della codifica con "sceltanuova"
indirizzo	INT	0	Indirizzo per uscita valore di misura o risultati
analisi_istanza	istanza locale	-	Istanza locale per ANALISI_FB

3.11.3 Parte istruzioni dell'FB RILEVAZIONE

Struttura della parte istruzioni

La parte istruzioni RILEVAZIONE è suddivisa in 3 parti:

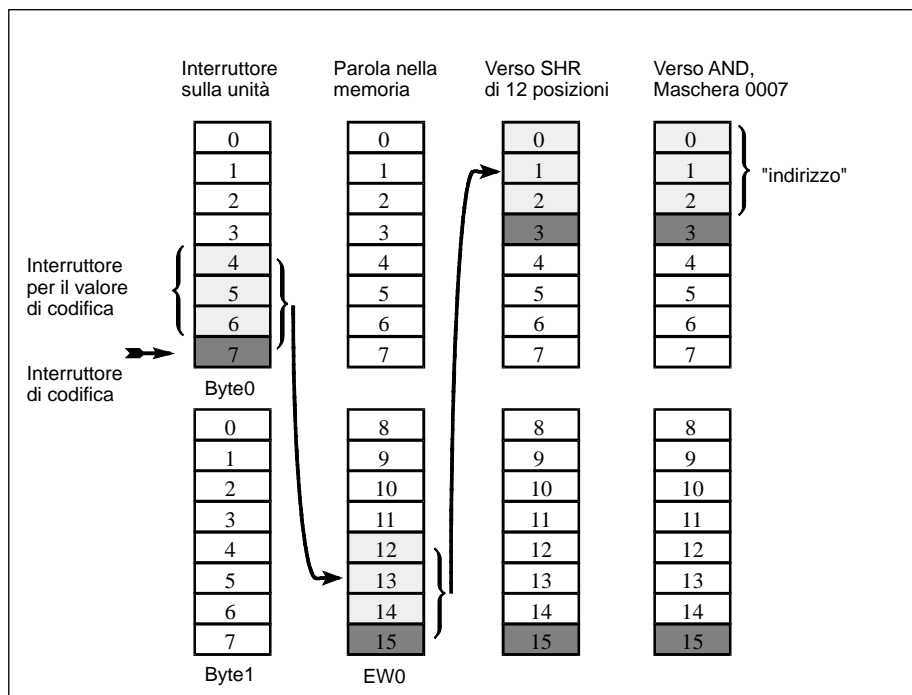
- Rilevazione dei valori di misura
Se il parametro d'ingresso "valorenuovo" è stato modificato rispetto al "valorevecchio", nel buffer circolare viene letto un nuovo valore di misura.
- Avvio ordinamento e calcolo
Mediante richiamo della funzione ANALISI, se il parametro d'ingresso "ordinamentonuovo" è stato modificato rispetto a "ordinamentovecchio".
- Analisi codifica e preparazione uscita
La codifica viene letta parola per parola: conformemente alle convenzioni SIMATIC, ciò significa che il gruppo interruttori superiore (Byte0) contiene gli 8 bit con valore massimo della parola d'ingresso, mentre il gruppo interruttori inferiore (Byte1) contiene i bit con valore minimo. La figura seguente illustra le posizioni in cui si trovano gli interruttori tramite i quali si regola la codifica:

Calcolo dell'indirizzo

La figura seguente illustra il calcolo dell'indirizzo: nei bit 12 fino a 14, la parola d'ingresso EW0 contiene la codifica che viene confermata quando, sull'interruttore di codifica (bit 15), viene riconosciuto un fronte. "indirizzo" viene determinato mediante spostamento verso destra con la funzione standard SHR e mascheramento dei bit rilevanti con una maschera AND.

Con questo indirizzo, gli elementi di campo (risultato del calcolo e relativo valore di misura) vengono scritti nel parametro d'uscita. Se viene emessa la radice o il quadrato dipende da "scelta funzione".

Un fronte sull'interruttore di codifica viene riconosciuto dal fatto che "sceltanuova" è cambiata rispetto a "sceltavecchia".



Parte istruzioni

La parte istruzioni del blocco di codice si presenta nel seguente modo:

```

BEGIN
  (*****
  Parte 1 : Rilevazione dei valori di misura. In caso di modifica del
  "valorenuovo"
  viene introdotto il valore di misura. Con l'operazione MOD si
  realizza un buffer circolare per i valori di misura.
  *****)
  IF valorenuovo <> valorevecchio THEN
    puntatore           := puntatore MOD NUMERO;
    valorimisura[puntatore] := valoremisura_in;
    puntatore           := puntatore + 1;
  END_IF;
  valorevecchio := valorenuovo;
  (*****
  Parte 2 : Avvio ordinamento e calcolo
  In caso di modifica di "ordinamentonuovo" avvio
  dell'ordinamento del buffer circolare ed esecuzione
  di calcoli con i valori di misura. I risultati vengono memorizzati
  in un campo nuovo "Buffer di calcolo".
  *****)
  IF ordinamentonuovo <> ordinamentovecchio THEN
    puntatore := 0;           //Resettare il puntatore buffer
    circolare
    istanza_analisi(bufferordinamento := valorimisura); //Richiamo di
    ANALISI
  END_IF;
  ordinamentovecchio := ordinamentonuovo;
  bufferrisultati := istanza_analisi.buffercalcolo; //quadrato e
  radice

  (*****
  Parte 3 : Analisi codifica e preparazione uscita: In caso di
  modifica
  di "Sceltanuova" viene determinata nuovamente la codifica per
  l'indirizzamento
  dell'elemento di campo per l'uscita: I bit rilevanti di "Scelta"
  vengono
  mascherati e convertiti in numeri interi.
  A seconda della posizione dell'interruttore "sceltafunzione" viene
  preparato
  per l'uscita "radice" o "quadrato".
  *****)
  )
  IF sceltanuova <> sceltavecchia THEN
    indirizzo := WORD_TO_INT(SHR(IN := scelta, N := 12) AND
    16#0007);
  END_IF;
  sceltavecchia := sceltanuova;
  IF sceltafunzione THEN
    risultato_out := bufferrisultato[indirizzo].quadrato;
  ELSE
    risultato_out := bufferrisultato[indirizzo].radice;
  END_IF;
  valoremisura_out := valorimisura[indirizzo]; //Visualizzazione
  valori di misura
  END_FUNCTION_BLOCK

```


3.12 Come creare il blocco organizzativo CICLO

Compiti dell'OB CICLO

È stato scelto un OB poiché esso viene richiamato ciclicamente. Con esso vengono realizzati i seguenti compiti per il programma:

- Richiamo e alimentazione del blocco funzionale RILEVAZIONE con dati d'ingresso e dati di controllo
- Immissione dei dati di risultato del blocco funzionale RILEVAZIONE
- Emissione dei valori per la visualizzazione

All'inizio della parte dichiarazioni è presente il campo dati temporaneo con 20 byte di "dati di sistema".

Codice di programma dell'OB CICLO

```

ORGANIZATION_BLOCK CICLO
(*****
CICLO corrisponde a OB1, cioè viene richiamato ciclicamente dal
sistema S7.
Parte 1 : Richiamo del blocco funzionale e conferma dei
valori d'ingresso Parte 2 : Immissione dei valori di uscita
ed emissione con commutazione uscita
*****)
VAR_TEMP
    dati di sistema : ARRAY[0..20] OF BYTE; // Area per OB1
END_VAR
BEGIN
(* Parte 1 : *****)
RILEVAZIONE.RILEVAZIONE_DATI(
    valoremisura_in      := WORD_TO_INT(Ingresso),
    valorenuovo          := "Ingresso 0.0", //Interruttore
d'ingresso come indicatore di segnale
    ordinamentonuovo     := Interruttore di ordinamento,
    sceltafunzione       := Interruttore di funzione,
    sceltanuova          := Interruttore di codifica,
    scelta               := Codifica);

(* Parte 2 : *****)
IF Interruttore di uscita THEN
//Commutazione uscita
    Uscita := RILEVAZIONE_DATI.risultato_out; //Radice o
quadrato
ELSE
    Uscita := RILEVAZIONE_DATI.valoremisura_out; //Valore di
misura
END_IF;
END_ORGANIZATION_BLOCK

```

Conversione dei tipi di dati

Il valore di misura è presente all'ingresso come tipo BYTE. Esso deve essere convertito in INT. A tal fine esso deve essere convertito da WORD in INT, mentre la precedente conversione da BYTE in WORD avviene automaticamente tramite il compilatore. Non è invece necessaria alcuna conversione per l'uscita, poiché quest'ultima è stata dichiarata come INT nella tabella dei simboli.

3.13 Dati di test

Presupposti

Per il test sono necessari un'unità d'ingresso sull'indirizzo 0 e un'unità di uscita sull'indirizzo 4.

Prima del test, disporre gli 8 interruttori del gruppo superiore verso sinistra ("0") e gli 8 interruttori del gruppo inferiore verso destra ("1").

Poiché vengono testati anche i blocchi iniziali delle variabili, ricaricare i blocchi nella CPU.

Fasi del test

Le fasi del test sono illustrati nella tabella.

Test	Azione	Conseguenza
1	Impostare la codifica su "111" (E0.4, E0.5 e E0.6) e confermare con l'interruttore di codifica (E0.7).	Tutti i LED dell'unità di uscita corrispondenti al byte meno significativo si accendono, indicando che il buffer è stato riorganizzato e che il valore di misura è stato spostato al primo posto.
2	Visualizzare la radice posizionando l'interruttore di uscita (E0.3) su "1".	Viene visualizzato all'uscita il valore binario "10000" (=16).
3	Visualizzare il quadrato posizionando l'interruttore di funzione (E0.2) su "1".	I 15 LED all'uscita si accendono, indicando un overflow: 255 x 255 dà un risultato troppo alto per il campo di numeri interi.
4a	Posizionare l'interruttore di uscita (E0.3) nuovamente su "0".	I LED dell'unità di uscita indicano nuovamente il valore di misura, corrispondente al byte meno significativo.
4b	Impostare all'ingresso il nuovo valore di misura 3 (il valore binario "11").	L'uscita non viene ancora modificata.
5a	Controllo del caricamento del valore di misura: impostare la codifica su "000" e confermare con l'interruttore di codifica (E0.7), in modo da poter controllare il caricamento del valore impostato.	L'unità di uscita si trova ancora sullo 0: nessun LED si accende.
5b	Per caricare il valore impostato nella fase 4, commutare l'interruttore d'ingresso 0.0 (E0.0). Viene letto il valore della quarta fase di test.	Sull'unità di uscita viene visualizzato il valore 3 (valore binario "11").
6	Avviare l'ordinamento ed il calcolo commutando l'interruttore di ordinamento (E0.1).	Sull'unità di uscita viene visualizzato nuovamente 0, poiché in seguito all'operazione di ordinamento il valore di misura è stato ulteriormente spostato verso l'alto.
7	Visualizzazione del valore di misura dopo l'ordinamento: impostare la codifica su "110" (E0.6 = 1, E0.5 = 1, E0.4 = 0 di EB0, corrispondenti a bit 14, bit 13, bit 12 di EW0) e confermare commutando l'interruttore di codifica.	Sull'unità di uscita viene nuovamente visualizzato il valore di misura "11", poiché questo è il secondo valore in ordine di grandezza.
8a	Visualizzare i risultati corrispondenti: commutando l'interruttore d'uscita (E0.3) viene visualizzato il quadrato del valore di misura della settima fase del test.	Il valore d'uscita è 9 (valore binario "1001").
8b	Commutando l'interruttore di funzione (E0.2) si ottiene la radice.	Viene visualizzato il valore d'uscita 2 (valore binario "10").

Test supplementare

Le spiegazioni degli interruttori sull'unità di ingresso delle tabelle seguenti e gli esempi dei test per quadrato e radice consentono all'utente di definire con facilità i propri passi di test:

- L'introduzione dei dati avviene tramite interruttori: tramite gli 8 interruttori superiori viene eseguito il comando, mentre tramite gli 8 interruttori inferiori viene regolato il valore di misura.
- L'emissione dei dati avviene tramite le visualizzazioni: sul gruppo superiore appare il byte di uscita con valore superiore, mentre sul gruppo inferiore appare il byte con valore inferiore.

Interruttori di comando	Nome	Spiegazione
Canale 0	Interruttore d'ingresso	Commutazione per confermare il valore di misura
Canale 1	Interruttore di ordinamento	Commutazione su ordinamento/analisi
Canale 2	Interruttore di funzione	Interruttore verso sinistra ("0"): radice Interruttore verso destra ("1"): quadrato
Canale 3	Interruttore di uscita	Interruttore verso sinistra ("0"): valore di misura Interruttore verso destra ("1"): risultato
Canale 4	Codifica	Indirizzo di uscita bit 0
Canale 5	Codifica	Indirizzo di uscita bit 1
Canale 6	Codifica	Indirizzo di uscita bit 2
Canale 7	Interruttore di codifica	Commutazione per confermare la codifica

Gli esempi dei test della tabella seguente usano tutti gli 8 valori di misura in una sequenza già prestabilita.

Introdurre i valori in una sequenza a piacere. Combinare i relativi bit ed immettere ogni valore mediante commutazione dell'interruttore d'ingresso. Dopo l'introduzione di tutti i valori, avviare l'ordinamento e l'analisi mediante commutazione dell'interruttore di ordinamento.

Dopodiché si possono visualizzare i valori di misura ordinati o i risultati - radice o quadrato.

Valore di misura	Radice	Quadrato
0000 0001 = 1	0, 0000 0001 = 1	0000 0000, 0000 0001 = 1
0000 0011 = 3	0, 0000 0010 = 2	0000 0000, 0000 1001 = 9
0000 0111 = 7	0, 0000 0011 = 3	0000 0000, 0011 0001 = 49
0000 1111 = 15	0, 0000 0100 = 4	0000 0000, 1110 0001 = 225
0001 1111 = 31	0, 0000 0110 = 6	0000 0011, 1100 0001 = 961
0011 1111 = 63	0, 0000 1000 = 8	0000 1111, 1000 0001 = 3969
0111 1111 = 127	0, 0000 1011 = 11	0011 1111, 0000 0001 = 16129
1111 1111 = 255	0, 0001 0000 = 16	0111 111, 1111 1111 = Visualizzazione di overflow!

4 Come utilizzare S7-SCL

4.1 Avvio del software S7-SCL

Avvio della superficie operativa di Windows

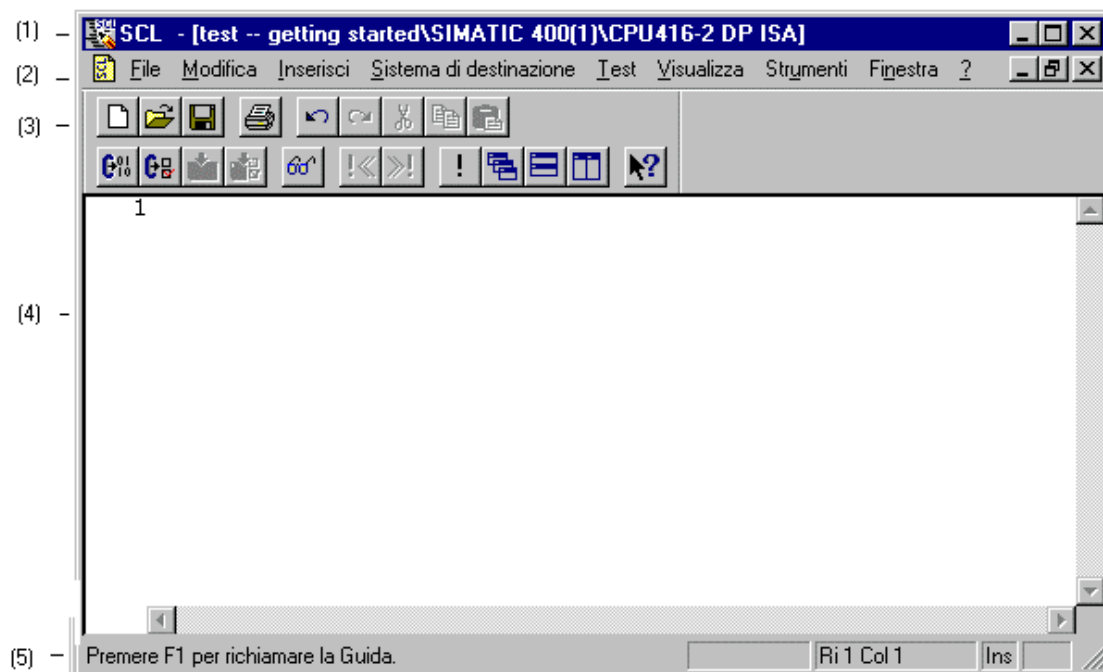
Dopo aver installato il software S7-SCL sul proprio PC, si può avviare S7-SCL anche tramite il pulsante "Start" sulla barra degli strumenti in Windows (Registrazione sotto "SIMATIC / STEP 7").

Avvio dal SIMATIC Manager

Il modo più rapido per avviare S7-SCL è quello di posizionare il puntatore del mouse su una sorgente S7-SCL nel SIMATIC Manager e fare doppio clic.

4.2 Superficie operativa

Le finestre S7-SCL si compongono dei seguenti elementi standard:



1. **Barra del titolo:**
contiene il titolo della finestra e i simboli per il comando della finestra.
2. **Barra dei menu:**
contiene i menu disponibili nella finestra.
3. **Barra degli strumenti:**
contiene icone che consentono di eseguire rapidamente i comandi usati più frequentemente.
4. **Area di lavoro:**
contiene le finestre nelle quali si può editare il testo del programma oppure leggere informazioni sulla compilazione e il test.
5. **Barra di stato:**
visualizza lo stato e ulteriori informazioni sull'oggetto selezionato.

4.3 Adattamento della superficie operativa

Impostazione dell'editor

Per eseguire impostazioni nell'editor selezionare il comando di menu **Strumenti > Impostazioni** e fare clic sulla scheda "Editor" della finestra di dialogo "Impostazioni". Effettuare le seguenti impostazioni.

Opzioni della scheda "Editor"	Significato
Caratteri	Imposta il tipo di carattere del file sorgente.
Tabulazione	Imposta i tabulatori del file sorgente.
Visualizza numero di riga	Visualizza i numeri di riga all'inizio delle righe.
Salva prima di compilare	Prima della compilazione chiede se si desidera salvare il file sorgente.
Conferma prima di salvare	Chiede di confermare se si vuole eseguire il salvataggio.

Impostazione del tipo e del colore del carattere

Per impostare il tipo e il colore del carattere dei diversi elementi del linguaggio, selezionare il comando di menu **Strumenti > Impostazioni** e fare clic sulla scheda "Formato" della finestra di dialogo "Impostazioni". Eseguire quindi le seguenti impostazioni.

Opzioni della scheda "Formato"	Significato
Parole chiave in maiuscolo	Formatta in lettere maiuscole le parole chiave di S7-SCL quali FOR, WHILE, FUNCTION_BLOCK, VAR o END_VAR.
Allinea alle parole chiave	Allinea le righe delle componenti di dichiarazione e delle istruzioni di controllo IF, CASE, FOR, WHILE e REPEAT.
Allineamento automatico	Quando si va a capo, la nuova riga si allinea automaticamente sotto la linea precedente, rispettando lo stesso margine. Questa impostazione vale solo per le righe immesse.
Stile/Colore	Definisce lo stile e il colore dei diversi elementi del linguaggio

Le impostazioni di questa scheda diventano attive solo se nella scheda "Formato" è stata impostata l'opzione "Utilizza i seguenti formati:" e se sono state effettuate le impostazioni desiderate.

Barra delle funzioni, barra dei punti d'arresto, barra di stato

Le barre delle funzioni, dei punti d'arresto e di stato sono attivabili e disattivabili separatamente con il corrispondente comando del menu **Visualizza**. Un segno di spunta prima del comando di menu indica che il comando è attivato.

Finestra "Risultati"

La finestra "Risultati" mostra errori e messaggi di avvertimento che compaiono durante la compilazione di un file sorgente. È possibile attivarla e disattivarla con il comando di menu **Visualizza > Risultati**.

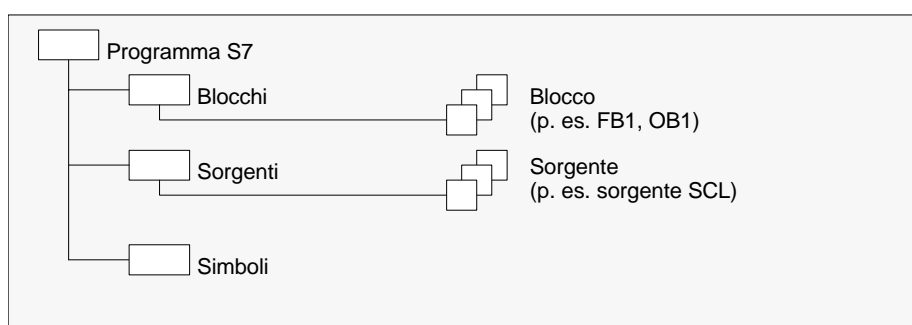
4.4 Come creare e gestire le sorgenti S7-SCL

4.4.1 Generazione di una nuova sorgente S7-SCL

Prima di poter scrivere un nuovo programma S7-SCL è necessario generare una nuova sorgente S7-SCL. Questa sorgente si genera nel contenitore Sorgenti sotto un programma S7.

Struttura di un programma S7 nel SIMATIC Manager

Le sorgenti generate con S7-SCL possono essere integrate nella struttura di un programma S7 nel modo seguente:



Procedere nel modo seguente:

1. Aprire la finestra di dialogo "Nuovo"
 - facendo clic sul simbolo "Nuovo" nella barra degli strumenti o
 - selezionando il comando di menu **File > Nuovo**.
2. Nella finestra di dialogo "Nuovo", selezionare
 - un progetto
 - il filtro "Sorgente S7-SCL" e
 - il contenitore Sorgenti nell'ambito del programma S7.
3. Registrare il nome dell'oggetto sorgente nella corrispondente casella di testo. Il nome può essere lungo max. 24 caratteri.
4. Confermare con "OK".

L'oggetto sorgente viene generato sotto il nome indicato dall'utente e visualizzato in una finestra di lavoro per la sua ulteriore elaborazione.

Nota

Per generare una sorgente S7-SCL si può utilizzare anche il SIMATIC Manager, selezionando un contenitore sorgente e il comando di menu **Inserisci > Software S7 > Sorgente S7-SCL**.

4.4.2 Apertura di una sorgente S7-SCL

Si può aprire una sorgente S7-SCL per sottoporla a compilazione o editazione.

Procedere nel modo seguente:

1. Aprire la finestra di dialogo "Apri"
 - facendo clic sul simbolo "Apri" oppure
 - selezionando il comando di menu **File > Apri**.
2. Selezionare nella finestra di dialogo:
 - il progetto desiderato,
 - il programma S7 desiderati e
 - il corrispondente contenitore sorgente.
3. Selezionare la sorgente S7-SCL.
4. Fare clic sul pulsante "OK".

Nota

Una sorgente S7-SCL può essere aperta anche con SIMATIC Manager facendo doppio clic sul suo simbolo o scegliendo il comando di menu **Modifica > Apri oggetto** dopo aver selezionato l'oggetto desiderato.

4.4.3 Chiusura di una sorgente S7-SCL

Procedere nel modo seguente:

- selezionare il comando di menu **File > Chiudi** oppure
- fare clic sul simbolo "Chiudi" nella barra dei simboli della finestra.

Nota

Se la sorgente è stata modificata, il sistema chiede se si desidera salvare la sorgente prima della chiusura. Se non si salva la sorgente, le modifiche apportate vanno perse.

4.4.4 Apertura di blocchi

Con l'applicazione S7-SCL non è possibile aprire blocchi. È consentito aprire sempre solo la relativa sorgente. I blocchi generati con S7-SCL possono però essere aperti con l'editor KOP/AWL/FUP e quindi visualizzati ed elaborati con il linguaggio di programmazione AWL. Non apportare modifiche al blocco in AWL perché:

- i comandi MC7 visualizzati non rappresentano necessariamente un blocco AWL valido,
- una compilazione corretta con il compilatore AWL richiede delle modifiche per le quali è necessario conoscere bene sia AWL che S7-SCL
- il blocco compilato con AWL avrà il codice di linguaggio AWL e non S7-SCL
- la sorgente S7-SCL e il codice MC7 non saranno più coerenti.

Per ulteriori informazioni si rimanda alla Guida online di STEP 7.

Nota

Un metodo molto semplice per gestire i propri programmi CPU è quello di apportare le eventuali modifiche nella sorgente S7-SCL e di ricompilare quest'ultima.

4.4.5 Definizione delle proprietà di oggetti

Si possono definire le proprietà di oggetti assegnando un attributo ai blocchi. Le proprietà della sorgente S7-SCL (ad es. l'autore) possono essere definite nella finestra di dialogo "Proprietà".

Procedere nel modo seguente:

1. Selezionare il comando di menu **File > Proprietà**.
2. Registrare le opzioni nella finestra di dialogo "Proprietà".
3. Confermare con "OK".

4.4.6 Generazione di sorgenti S7-SCL con un editor standard

Per editare la sorgente S7-SCL si può utilizzare anche un Editor ASCII standard. In tal caso però non sono disponibili le numerose funzioni di editazione e la Guida online integrata di S7-SCL.

Dopo aver creato e salvato la sorgente, la si deve importare nel contenitore sorgenti di un programma utente con l'ausilio del SIMATIC Manager (vedere documentazione STEP 7). Infine, si può aprire la sorgente in S7-SCL per sottoporla a ulteriore elaborazione o compilazione.

4.4.7 Protezione dei blocchi

Per i blocchi è possibile impostare una protezione del blocco indicando l'attributo `KNOW_HOW_PROTECT` durante la programmazione del blocco nella sorgente.

La protezione del blocco comporta le seguenti conseguenze:

- Quando in seguito si apre un blocco compilato con l'Editor incrementale AWL, non si può vedere la parte istruzioni del blocco.
- Nella parte convenzioni del blocco vengono visualizzate solo le variabili dei tipi di dichiarazione `VAR_IN`, `VAR_OUT` e `VAR_IN_OUT`. Non vengono invece visualizzate le variabili dei blocchi di convenzione `VAR` e `VAR_TEMP`.

Durante l'introduzione della protezione del blocco vale quanto segue:

- La parola chiave è `KNOW_HOW_PROTECT`. Essa viene introdotta prima di tutti gli altri attributi del blocco.
- In tal modo è possibile proteggere OB, FB FC e DB.

4.5 Regole per le sorgenti S7-SCL

4.5.1 Regole generali per le sorgenti S7-SCL

Le sorgenti S7-SCL devono rispettare le regole seguenti:

- In una sorgente S7-SCL si può editare un numero qualsiasi di blocchi di codice (FB, FC, OB), blocchi dati (DB) e tipi di dati definiti dall'utente (UDT).
- Ogni tipo di blocco ha una propria struttura tipica.
- Ogni istruzione e ogni dichiarazione di variabile termina con un punto e virgola (;).
- Non viene fatta distinzione tra lettere maiuscole e minuscole.
- I commenti servono solo per la documentazione del programma. Essi non influenzano l'esecuzione del programma.
- I blocchi dati di istanza vengono generati automaticamente durante il richiamo di un blocco funzionale. Essi non devono essere editati.
- Il DB 0 è preassegnato. Non è perciò possibile generare un DB con questo nome.

4.5.2 Ordine dei blocchi

Per quanto concerne l'ordine dei blocchi, durante la creazione di una sorgente S7-SCL si deve osservare quanto segue:

- I blocchi richiamati devono essere situati prima dei blocchi richiamanti.
- I tipi di dati definiti dall'utente (UDT) sono situati prima dei blocchi in cui essi vengono utilizzati.
- I blocchi dati ai quali è stato assegnato un tipo di dati definiti dall'utente (UDT) sono situati dopo l'UDT.
- I blocchi dati sono situati prima di tutti i blocchi dati che accedono ad essi.

4.5.3 Utilizzo di indirizzi simbolici

In un programma S7-SCL si lavora con vari operandi come segnali E/A, merker, contatori, temporizzatori e blocchi. Nel programma è possibile indirizzare questi operandi in modo assoluto (ad es. E1.1, M2.0, FB11), ma il grado di leggibilità delle sorgenti S7-SCL aumenta notevolmente se per l'indirizzamento si ricorre a dei simboli (ad es. Motore_ACC). In quest'ultimo caso, nel programma utente per indirizzare un determinato operando si può utilizzare il simbolo assegnato.

Simboli locali e globali

- I simboli globali vengono impiegati per aree di memoria della CPU e per nomi di blocchi. Essi sono noti nell'intero programma utente e devono perciò essere identificati in modo univoco. La tabella dei simboli può essere creata con STEP 7.
- I simboli locali sono noti solo nel blocco nella cui parte convenzioni essi sono stati definiti dall'utente. Si possono assegnare nomi per variabili, parametri, costanti e etichette di salto, e in blocchi diversi è possibile utilizzare gli stessi nomi per scopi diversi.

Nota

Fare attenzione affinché i nomi simbolici siano univoci e non siano identici con parole chiavi.

4.6 Come editare nelle sorgenti S7-SCL

4.6.1 Annullamento dell'ultima azione di editazione

Il comando di menu **Modifica > Annulla** consente di annullare una o più azioni.

Non tutti gli operandi possono essere annullati. Ad es. il comando di menu **File > Salva** non può essere annullato.

4.6.2 Ripristino di un'azione di editazione

Una volta annullate, le azioni possono essere ripristinate con il comando di menu **Modifica > Ripristina**.

4.6.3 Ricerca e sostituzione di oggetti di testo

Se si desidera rielaborare o modificare una sorgente S7-SCL, spesso si può risparmiare molto tempo prezioso effettuando la ricerca e se necessario la sostituzione degli oggetti di testi desiderati. Fra gli obiettivi dell'operazione di ricerca vi possono essere per es. parole chiave, identificatori assoluti, identificatori simbolici, ecc.

Procedere nel modo seguente:

1. Selezionare il comando di menu **Modifica > Trova/Sostituisci...**
2. Registrare le opzioni nella finestra di dialogo "Trova/Sostituisci".
3. Avviare la ricerca
 - facendo clic sul pulsante "Trova", per cercare e selezionare un oggetto di testo o
 - facendo clic sul pulsante "Sostituisci" oppure "Sostituisci tutto", per cercare un testo e sostituirlo con il testo contenuto nella casella di testo "Sostituisci con".

4.6.4 Selezione di oggetti di testo

Per selezionare gli oggetti di testo tenere premuto il tasto del mouse e trascinare il puntatore del mouse fino all'area del testo desiderata.

Inoltre, si può:

- selezionare l'intero testo di una sorgente scegliendo il comando di menu **Modifica > Seleziona tutto**.
- selezionare una parola facendo doppio clic su di essa, oppure
- selezionare un'intera riga facendo clic sul margine a sinistra vicino alla riga.

La selezione può essere annullata con il comando di menu **Modifica > Annulla selezione**.

4.6.5 Copia di oggetti di testo

Con questa funzione si possono copiare sezioni di programmi o interi programmi. Il testo copiato viene depositato negli appunti e da qui può essere inserito a piacere in altre posizioni.

Procedere nel modo seguente:

1. Selezionare l'oggetto del testo che si desidera copiare.
2. Copiare l'oggetto
 - facendo clic sul simbolo "Copia" nella barra degli strumenti oppure
 - selezionando il comando di menu **Modifica > Copia**.
3. Spostare il cursore nella posizione in cui si desidera inserire l'oggetto (nella stessa applicazione o in un'altra applicazione).
4. Inserire l'oggetto
 - facendo clic sul simbolo "Incolla" nella barra degli strumenti oppure
 - selezionando il comando di menu **Modifica > Incolla**.

4.6.6 Taglio di oggetti di testo

Questa funzione consente di spostare negli appunti il testo selezionato. Normalmente questo comando di menu viene utilizzato assieme al comando di menu **Modifica > Incolla** che inserisce il contenuto degli appunti nella posizione attuale del cursore.

Procedere nel modo seguente:

1. Selezionare l'oggetto che deve essere tagliato.
2. Tagliare l'oggetto
 - facendo clic sul simbolo "Taglia" nella barra degli strumenti o
 - selezionando il comando di menu **Modifica > Taglia**.

Nota

- L'oggetto selezionato non può essere tagliato se il comando di menu **Modifica > Taglia** non è stato attivato (su sfondo grigio).
 - Con il comando di menu **Modifica > Incolla** si può inserire il testo tagliato in un'altra posizione (o dello stesso programma o di un altro programma).
 - Il contenuto degli appunti rimane inalterato finché non viene eseguito uno dei comando di menu **Modifica > Taglia** o **Modifica > Copia**.
-

4.6.7 Cancellazione di oggetti di testo

Un oggetto di testo selezionato può essere eliminato dal testo sorgente.

Procedere nel modo seguente:

1. Selezionare il testo da cancellare.
2. Selezionare il comando di menu **Modifica > Cancella**.

Il testo cancellato non viene inserito negli appunti. Il testo cancellato può essere nuovamente inserito con il comando di menu **Modifica > Annulla** oppure **Modifica > Ripristina**.

4.6.8 Posizionamento del cursore su una determinata riga

Le seguenti funzioni consentono di posizionare il cursore nel punto desiderato.

Posizionamento su un determinato numero di riga

È possibile posizionare il cursore all'inizio di una determinata riga:

1. Selezionare il comando di menu **Modifica > Vai a Riga**.

La finestra di dialogo "Vai a" si apre.

2. Registrare il numero di riga nella finestra di dialogo "Vai a".
3. Confermare con "OK".

Posizionamento sul segnalibro successivo/precedente

Se nella sorgente sono stati impostati i segnalibro, tra di essi è consentito navigare:

- Selezionare il comando di menu **Modifica > Vai a > Segnalibro successivo/precedente**.

Posizionamento sull'errore successivo/precedente nel codice di programma

Nella finestra di dialogo "Risultati" vengono visualizzati, in seguito alla compilazione, tutti gli errori sintattici comprensivi del numero di riga e di colonna.

S7-SCL consente di navigare tra i singoli errori all'interno del programma per eliminare tutti quelli originati durante la precedente compilazione:

1. Posizionare il cursore in qualsiasi punto del testo sorgente.
2. Selezionare il comando di menu **Modifica > Vai a > Errore precedente/errore successivo**.

4.6.9 Allineamento delle righe in base alla sintassi

Le seguenti funzioni consentono di definire la struttura delle sorgenti S7-SCL impostando l'allineamento delle righe.

- **Allineamento automatico**

Quando la funzione è attiva, ogni volta che si va a capo, la nuova linea viene automaticamente allineata sotto la linea precedente, rispettando lo stesso margine.

- **Allinea parole chiave**

Se questa funzione è attiva, le righe delle componenti di dichiarazione e delle strutture di controllo IF, CASE, FOR, WHILE e REPEAT vengono allineate.

Procedere nel modo seguente:

1. Selezionare il comando di menu **Strumenti > Impostazioni**.
2. Nella finestra di dialogo visualizzata scegliere la scheda "Formato".
3. Verificare che l'opzione "Utilizza i seguenti formati:" sia attiva.
4. Attivare l'opzione "Allineamento automatico" o "Allinea parole chiave".

4.6.10 Impostazione dello stile e del colore del carattere

Attribuendo stili e colori diversi agli elementi del linguaggio si ottiene una sorgente S7-SCL più leggibile e professionale. Il testo sorgente può essere formattato con le seguenti funzioni:

- **Parole chiave in maiuscolo:**
Quando questa funzione è attiva alcune parole chiave quali FOR, WHILE, FUNCTION_BLOCK, VAR o END_VAR vengono scritte in lettere maiuscole.
- **Definizione dello stile e del colore del carattere:**
Per default gli elementi del linguaggio quali le operazioni, i commenti e le costanti hanno un proprio stile e colore. L'impostazione di default è tuttavia modificabile. Sono disponibili i seguenti colori:

Colore del carattere	Elemento del linguaggio	Esempio
Blu	Parole chiave	ORGANIZATION_BLOCK
	Tipi di dati predefiniti	INT
	Identificatori predefiniti	ENO
	Funzioni standard	BOOL_TO_WORD
Ocra	Operazioni	NOT
Rosa	Costanti	TRUE
Smeraldo	Commento	//... o (*...*)
Violetto	Simboli globali (tabella dei simboli) tra virgolette	"Motore"
Nero	Testo normale	Variabili

Procedere nel modo seguente:

1. Selezionare il comando di menu **Strumenti > Impostazioni**.
2. Selezionare la scheda "Formato" della finestra visualizzata.
3. Verificare che l'opzione "Utilizza i seguenti formati per la stampa" sia attiva.
4. Eseguire le impostazioni desiderate. Per ottenere informazioni sulla finestra di dialogo fare clic sul pulsante "?" nella finestra aperta.

4.6.11 Posizionamento dei segnalibro nel testo sorgente

Con l'ausilio dei segnalibro è possibile navigare rapidamente all'interno di una sorgente. Questi si rivelano utili ad esempio per apportare modifiche che si riflettono in diversi punti di una sorgente. È possibile inserire i segnalibro in qualsiasi punto di una sorgente. Se sono presenti più segnalibro si può navigare in avanti e all'indietro tra ognuno di essi.

Validità

I segnalibro sono validi finché rimane aperta la sorgente ma non verranno salvati insieme ad essa.

Inserimento dei segnalibro

1. Posizionare il cursore sulla riga che si intende selezionare.
2. Selezionare il comando di menu **Modifica > Attiva/disattiva segnalibro**.

Navigazione tra diversi segnalibro

Selezionare il comando di menu **Modifica > Vai a > Segnalibro precedente/successivo**.

Cancellazione dei segnalibro

Selezionare il comando di menu **Modifica > Cancella tutti i segnalibro**.

Nota

Le funzioni necessarie per operare con i segnalibro sono immediatamente disponibili tramite la barra segnalibro visualizzabile con il comando di menu **Visualizza > Barra segnalibro**.

4.6.12 Come inserire le variabili

4.6.12.1 Inserimento di richiami di blocco

S7-SCL offre un supporto per la programmazione di richiami di blocco. Sono blocchi richiamabili:

- blocchi funzionali di sistema (SFB) e funzioni di sistema (SFC) delle biblioteche SIMATIC,
- blocchi funzionali e funzioni creati con S7-SCL,
- blocchi funzionali e funzioni creati con altri linguaggi di STEP 7.

Procedere nel modo seguente:

1. Selezionare il comando di menu **Inserisci > Richiamo di blocco**.
2. Selezionare nella successiva finestra di dialogo il blocco SFC, SFB, FC o FB desiderato e confermare con "OK".
S7-SCL copia automaticamente il blocco chiamato nel programma S7 e registra il richiamo di blocco e i parametri formali del blocco con una sintassi corretta nella sorgente.
3. Se è stato richiamato un blocco funzionale, indicare anche il DB di istanza.
4. Inserire quindi i parametri attuali per alimentare il blocco. Per facilitare la selezione di un parametro attuale, S7-SCL indica il tipo di dati richiesto in forma di commento.

4.6.12.2 Inserimento di modelli per i blocchi

Una delle funzioni di editazione di S7-SCL è l'inserimento di modelli per i gli OB, FB, FC, DB, DB di istanza, DB da UDT e UDT. Questi facilitano l'immissione e il rispetto delle regole sintattiche.

Procedere nel modo seguente:

1. collocare il cursore nel punto in cui si desidera inserire il modello per il blocco.
2. Selezionare il corrispondente comando di menu **Inserisci > Modello per blocco > OB/FB/FC/DB/IDB/DB da UDT/UDT**.

4.6.12.3 Inserimento di modelli per i commenti

Una delle funzioni di editazione di S7-SCL è l'inserimento di modelli per i blocchi di commenti che facilitano l'immissione e il rispetto delle regole sintattiche.

Procedere nel modo seguente:

1. Posizionare il cursore dopo l'intestazione del blocco desiderato.
2. Selezionare il corrispondente comando di menu **Inserisci > Modello per blocco > Commento**.

4.6.12.4 Inserimento di modelli per i parametri

Una delle funzioni di editazione di S7-SCL è l'inserimento di modelli per i parametri. Questi facilitano l'immissione e il rispetto delle regole sintattiche. I parametri possono essere definiti in blocchi funzionali e in funzioni.

Procedere nel modo seguente:

1. Posizionare il cursore nella parte convenzioni di un FB o di una FC.
2. Selezionare il comando di menu **Inserisci > Modello per blocco > Parametro**.

4.6.12.5 Inserimento di strutture di controllo

Una delle funzioni di editazione di S7-SCL è l'inserimento di modelli di strutture di controllo per i blocchi di codice che facilitano l'immissione e il rispetto delle regole sintattiche.

Procedere nel modo seguente:

1. Collocare il cursore nella posizione in cui si desidera inserire il modello.
2. Selezionare il corrispondente comando di menu **Inserisci > Struttura di controllo > IF/CASE/FOR/WHILE/REPEAT**.

4.7 Come compilare i programmi S7-SCL

4.7.1 Istruzioni fondamentali per la compilazione

Prima di poter eseguire o testare un programma, esso deve essere compilato. Il compilatore viene attivato con l'avvio del processo di compilazione. Il compilatore ha le seguenti caratteristiche:

- Durante un ciclo di compilazione la sorgente S7-SCL può essere elaborata sia come entità unica, che come blocchi singoli.
- Tutti gli errori sintattici riscontrati nel corso della compilazione vengono visualizzati in seguito in un'apposita finestra.
- Ad ogni richiamo di un blocco funzionale viene generato un corrispondente blocco dati di istanza, a condizione che tale blocco non sia già esistente.
- L'utente ha anche la possibilità di compilare più sorgenti S7-SCL in una volta, tramite la creazione di un file di compilazione S7-SCL.
- Con il comando di menu **Strumenti > Impostazioni** si possono impostare le opzioni per il compilatore.

Dopo aver generato e compilato un programma utente senza errori, si può partire dal presupposto che tale programma sia corretto. Tuttavia, se il programma viene eseguito sotto PLC si possono verificare degli errori. Per identificare tali errori si possono utilizzare le funzioni di test di S7-SCL.

4.7.2 Impostazione del compilatore

Esiste la possibilità di adattare la fase di compilazione alle esigenze individuali dell'utente.

Procedere nel modo seguente:

1. Selezionare il comando di menu **Strumenti > Impostazioni**, per aprire la finestra di dialogo "Impostazioni".
2. Selezionare la scheda "Compilatore" o "Crea blocco".
3. Registrare le opzioni nella scheda.

Opzioni della scheda "Compilatore"

Crea codice dell'oggetto	Questa opzione consente di indicare se si vuole creare un codice eseguibile. Senza questa opzione la compilazione serve unicamente per eseguire un controllo della sintassi.
Ottimizza codice oggetto	Selezionando questa opzione si fa in modo che il fabbisogno di memoria e il tempo di esecuzione dei blocchi nel PLC vengano ottimizzati. È consigliabile impostarla perché non compromette in alcun modo la funzionalità dei blocchi.
Controlla limiti array	Se si seleziona questa opzione, durante l'esecuzione del programma S7 viene controllato se l'indice dell'array è compreso entro i limiti stabiliti per il campo (ARRAY). Se l'indice dell'array supera i limiti ammessi il flag OK viene impostato su FALSE.
Crea informazioni debug	Questa opzione consente di eseguire un ciclo di test con il debugger dopo aver compilato e caricato il programma nella CPU. Essa determina tuttavia un incremento del fabbisogno di memoria e dei tempi di esecuzione del programma nel PLC.
Attiva flag OK	Questa opzione consente di interrogare il flag OK nei testi sorgente S7-SCL.
Autorizza commenti annidati	Questa opzione consente di annidare più commenti uno nell'altro nella sorgente S7-SCL.
Lunghezza max. stringa	Consente di ridurre la lunghezza standard del tipo di dati STRING che è impostata per default su 254 caratteri. Questa impostazione si riferisce a tutti i parametri di uscita e di ingresso/uscita e ai valori di ritorno delle funzioni. Si noti che il valore impostato non deve essere inferiore alle variabili STRING utilizzate nel programma.

Opzioni della scheda "Crea blocco"

Sovrascrivi blocchi	Se nel corso della compilazione vengono creati blocchi con lo stesso nome, questa opzione sovrascrive i blocchi omonimi della cartella "Blocchi" di un programma S7. Anche durante il caricamento, i blocchi con nome uguale, già presenti nel sistema di destinazione, vengono sovrascritti. Se l'opzione non è stata selezionata, per sovrascrivere il blocco si deve confermare un messaggio.
Visualizza avvisi	Stabilisce se dopo un ciclo di compilazione verranno segnalati, oltre agli errori, anche degli avvisi.
Visualizza errori e poi avvisi	Stabilisce che nella finestra di emissione compaiano prima gli errori e poi gli avvisi.
Crea dati di riferimento	Selezionare questa opzione per fare in modo che quando si crea un blocco vengano creati automaticamente i dati di riferimento. Con il comando di menu Strumenti > Dati di riferimento è tuttavia possibile creare o aggiornare i dati di riferimento successivamente.
Considera attributo di sistema "S7_server"	Questa opzione fa in modo che, quando si crea un blocco, venga considerato anche l'attributo di sistema per il parametro "S7-server". Questo attributo viene assegnato se il parametro è rilevante per la progettazione dei collegamenti e delle messaggi e contiene il numero del collegamento o del messaggio. Questa opzione allunga la procedura di compilazione.

4.7.3 Compilazione del programma

Prima di poter testare o eseguire un programma, esso deve essere compilato. Per essere sicuri di avere sempre la versione più recente della sorgente S7-SCL, si raccomanda di selezionare il comando di menu **Strumenti > Impostazioni** e nella scheda "Editor" di fare clic sull'opzione "Salva prima di compilare". Il comando di menu **File > Compila** memorizza implicitamente la sorgente S7-SCL.

Procedere nel modo seguente:

1. Salvare la sorgente S7-SCL da compilare.
2. Per poter generare un programma operabile, è assolutamente necessario selezionare l'opzione "Codice macchina" nella finestra di dialogo "Impostazioni" nella scheda "Compilatore".
3. Modificare eventualmente altre impostazioni del compilatore.
4. Determinare se la corrispondente tabella dei simboli si trova nella stessa directory di programma.
5. Per avviare il processo di compilazione esistono due possibilità:
 - Il comando di menu **File > Compila** compila il file sorgente completo.
 - Il comando di menu **File > Compilazione parziale** visualizza una finestra di dialogo in cui si possono selezionare i singoli blocchi da compilare.
6. Nella finestra di dialogo "Risultati" vengono visualizzati tutti gli errori sintattici e i messaggi di avviso che si sono verificati nel corso della compilazione del programma. Dopo la fase di compilazione, correggere gli errori eventualmente segnalati e ripetere la procedura sopra descritta.

4.7.4 Creazione di un file di compilazione

Un file di compilazione offre all'utente la possibilità di compilare più sorgenti S7-SCL in una volta all'interno di un contenitore di sorgente (SO). Nel file di compilazione vanno riportati i nomi delle sorgenti S7-SCL nell'ordine in cui esse devono essere compilate.

Procedere nel modo seguente:

1. Aprire la finestra di dialogo "Nuovo" selezionando il comando di menu **File > Nuovo**.
2. Nella finestra di dialogo "Nuovo", selezionare
 - un contenitore di sorgente all'interno di un programma S7 e
 - il tipo di oggetto "File di compilazione S7-SCL".
3. Riportare il nome del file di controllo nel campo di testo opportuno (al massimo 24 caratteri) e confermare con "OK".
4. Il file viene creato e visualizzato, per un'ulteriore elaborazione, in una finestra di lavoro. Introdurre nella finestra il nome delle sorgenti S7-SCL da compilare nell'ordine desiderato e salvare il file.
5. Avviare quindi la procedura di compilazione selezionando il comando di menu **File > Compila**.

4.7.5 Eliminazione di errori dopo la compilazione

Alla fine della fase di compilazione, tutti gli errori sintattici e i messaggi di avviso che si sono verificati durante la compilazione vengono visualizzati nella finestra di dialogo "Risultati". La presenza di un errore impedisce la generazione del rispettivo blocco, mentre se si verificano dei messaggi di avviso viene generato ugualmente un blocco operabile. Tuttavia, in fase di esecuzione del PLC si possono verificare errori.

Per eliminare gli errori dopo la compilazione.

1. Selezionare l'errore e premere il tasto F1, per visualizzare una descrizione dell'errore e degli interventi per eliminarlo.
2. Nel caso che nel messaggio venga indicato un numero di riga (Rxx) e di colonna (Cxx), è possibile localizzare il punto errato nel testo sorgente
 - facendo clic sul messaggio di errore nella finestra di dialogo "Risultati", richiamando il menu contestuale con il tasto destro del mouse e selezionando il comando di menu **Visualizza errore**
 - facendo doppio clic sul messaggio di errore per posizionare il cursore sulla posizione segnalata (riga, colonna).
3. Per la corretta sintassi si rimanda alla descrizione del linguaggio.
4. Correggere in modo opportuno il testo sorgente.
5. Salvare la sorgente.
6. Compilare nuovamente la sorgente.

4.8 Come salvare e stampare le sorgenti S7-SCL

4.8.1 Salvataggio di una sorgente S7-SCL

Con il termine "Salvare" in S7-SCL si intende sempre il salvataggio di sorgenti. In S7-SCL, i blocchi vengono generati durante la compilazione della sorgente e memorizzati automaticamente nella corrispondente directory di programma. Si può salvare in qualsiasi momento una sorgente S7-SCL allo stato attuale. L'oggetto non viene compilato. Vengono salvati anche gli eventuali errori sintattici.

Procedere nel modo seguente:

- Selezionare il comando di menu **File > Salva** o fare clic sul simbolo "Salva" nella barra degli strumenti.
La sorgente S7-SCL viene aggiornata.
- Se si desidera creare una copia della sorgente S7-SCL attuale, selezionare il comando di menu **File > Salva con nome**. Appare la finestra di dialogo "Salva con nome", in cui si può indicare il nome e il percorso del duplicato.

4.8.2 Impostazione del formato di pagina

L'aspetto della stampa può essere modificato nel seguente modo:

- Con il comando di menu **File > Imposta pagina** è possibile stabilire il formato e l'allineamento della pagina da stampare.
- Le righe di intestazione e le righe a piè di pagina per i documenti da stampare possono essere impostate selezionando prima il comando di menu **File > Imposta pagina** e poi nella successiva finestra di dialogo la scheda "Campi di scrittura".
- Con il comando di menu **File > Imposta stampante** è possibile effettuare ulteriori impostazioni di stampa.

Nota

I dati relativi all'allineamento della pagina devono essere inseriti nella finestra di dialogo "Imposta pagina". Invece quelli contenuti nella finestra di dialogo "Imposta stampante" non sono rilevanti per la stampa delle sorgenti S7-SCL.

- Con il comando di menu **File > Anteprima di stampa** è possibile verificare le impostazioni eseguite nell'anteprima di stampa prima di inviare il documento alla stampante.

4.8.3 Stampa di una sorgente S7-SCL

Viene stampata la sorgente S7-SCL della finestra di lavoro attiva, ovvero per poter stampare una sorgente S7-SCL la si deve dapprima aprire.

Procedere nel modo seguente:

1. Attivare la finestra di lavoro della sorgente S7-SCL, il cui contenuto si desidera stampare.
2. Aprire la finestra di dialogo "Stampa"
 - facendo clic sul simbolo "Stampa" o
 - selezionando il comando di menu **File > Stampa**.
3. Selezionare nella finestra di dialogo "Stampa" le opzioni di stampa, come p. es. l'area di stampa o il numero di copie.
4. Confermare con "OK".

4.8.4 Impostazione delle opzioni di stampa

Sono disponibili le seguenti opzioni per la formattazione della stampa:

- **Nuova pagina all'inizio del blocco/alla fine del blocco**
Quando questa funzione è attiva ogni blocco viene stampato in una pagina nuova o si conclude con una nuova pagina.
- **Stampa con numero di riga**
Quando questa funzione è attiva all'inizio di ogni riga viene stampato un numero.
- **Stampa a colori**
Quando questa funzione è attiva la stampa riporterà fedelmente i colori utilizzati nella sorgente.
- **Carattere di stampa**
Per default è impostato il carattere Courier New con una grandezza di 8 punti. Questo carattere di stampa ha una risoluzione ottimale.
- **Stile**
È possibile definire stili diversi per i singoli elementi del linguaggio. I seguenti elementi sono selezionabili singolarmente:

Elemento del linguaggio	Esempio
Testo normale	
Parole chiave	ORGANIZATION_BLOCK
Identificatore di tipi di dati predefiniti	INT
Identificatori predefiniti	ENO
Identificatori di funzioni standard	BOOL_TO_WORD
Operazioni	NOT
Costanti	TRUE
Blocco di commenti	(* *)
Commento a una riga	//...
Simboli globali (tabella dei simboli) fra virgolette	"Motore"

Procedere nel modo seguente:

1. Selezionare il comando di menu **Strumenti > Impostazioni**.
2. Nella finestra di dialogo visualizzata scegliere la scheda "Stampa".
3. Verificare che l'opzione "Utilizza i seguenti formati:" sia attiva.
4. Effettuare le impostazioni desiderate. Per ottenere informazioni sulla finestra di dialogo fare clic sul pulsante "?" della finestra aperta.

4.9 Come caricare i programmi

4.9.1 Cancellazione totale della memoria CPU

Con la funzione Cancellazione totale su può cancellare online un intero programma utente in una CPU.

Procedere nel modo seguente:

1. Selezionare il comando di menu **Sistema di destinazione > Stato di funzionamento e** commutare la CPU su STOP.
2. Selezionare il comando di menu **Sistema di destinazione > Cancellazione totale**.
3. Confermare l'azione nella finestra di dialogo che appare subito dopo.



Attenzione

- La CPU viene resettata.
 - Tutti i dati utente vengono cancellati.
 - La CPU interrompe tutti i collegamenti esistenti.
 - Se è inserita una memory card, dopo la cancellazione totale la CPU copia il contenuto della Memory Card nella memoria di caricamento interna.
-

4.9.2 Caricamento di programmi utente nella CPU

Presupposti

Durante la compilazione di una sorgente S7-SCL, dalla sorgente vengono generati i blocchi, i quali vengono salvati nella directory "Blocchi" del programma S7.

I blocchi richiamati dai blocchi S7-SCL nel primo livello vengono cercati, copiati automaticamente nella directory "Blocchi" e ripresi nella lista di caricamento.

Gli altri blocchi del programma utente possono essere caricati dal PG nella CPU con SIMATIC Manager.

Per poter eseguire il caricamento è necessario innanzitutto collegare il PG e la CPU. All'unità CPU deve essere assegnato un programma utente online nel SIMATIC Manager.

Procedura

Una volta compilata la sorgente si può avviare il caricamento in uno dei seguenti modi:

- con il comando di menu **File > Carica** che consente di caricare tutti i blocchi della sorgente e i blocchi richiamati nel primo livello
- con il comando di menu **File > Caricamento parziale** che apre una finestra di dialogo in cui è possibile selezionare i singoli blocchi da caricare.

I blocchi vengono trasferiti alla CPU. Se un blocco è già presente nella RAM della CPU, alla domanda del sistema si deve confermare o non se si desidera sovrascrivere il blocco.

Nota

Il caricamento di un programma utente allo stato di funzionamento STOP presenta determinati vantaggi poiché durante la sovrascrittura di un vecchio programma allo stato di funzionamento RUN si possono verificare errori.

4.10 Come testare i programmi

4.10.1 Funzioni di test di S7-SCL

Le funzioni di test di S7-SCL offrono la possibilità di controllare l'esecuzione di un programma nella CPU e di localizzare in tal modo probabili errori. Durante la compilazione vengono visualizzati gli errori sintattici. Anche gli errori del tempo di esecuzione del programma vengono visualizzati tramite allarmi di sistema; con le funzioni di test si possono identificare eventuali errori logici di programmazione.

Funzioni di test di S7-SCL

In S7-SCL sono disponibili le seguenti funzioni di test:

- La funzione Controlla
Con questa funzione è possibile visualizzare nella sorgente S7-SCL i nomi e i valori attuali delle variabili. Durante l'esecuzione del test, i valori delle variabili e dei parametri di quest'area vengono visualizzati in sequenza cronologica e aggiornati ciclicamente.
- Test con punti d'arresto e in modalità passo singolo
Questa funzione consente di impostare dei punti d'arresto ed eseguire un ciclo di test in passi singoli. Si può eseguire l'algoritmo del programma p. es. istruzione per istruzione e controllare il cambiamento dei valori delle variabili elaborate.



Attenzione

L'esecuzione di un test durante il funzionamento dell'impianto può causare gravi danni a persone e cose in caso di disturbi di funzionamento o errori di programma!

Prima di eseguire le funzioni di test ci si deve perciò assicurare che non si possano verificare stati pericolosi!

Presupposti per lo svolgimento del test

- Il programma è stato compilato con le opzioni "Crea codice dell'oggetto" e "Crea informazioni debug".
Queste opzioni si possono selezionare nella scheda "Compilatore" della finestra di dialogo "Impostazioni". Quest'ultima può essere aperta con il comando di menu **Strumenti > Impostazioni**.
- Sono a disposizione i dati di riferimento attuali.
I dati di riferimento vengono generati automaticamente con la compilazione se è stata attivata l'opzione "Crea informazioni debug".
- Esiste un collegamento online tra il PG/PC e la CPU.
- Il programma è stato caricato nella CPU. Esso può essere eseguito con il comando di menu **Sistema di destinazione > Carica**.

4.10.2 Informazioni importanti sulla funzione di test "Controlla"

Funzione di test "Controlla"

Durante il controllo continuo di un programma è possibile testare un gruppo di istruzioni. Questo gruppo di istruzioni viene denominato "area di controllo". Durante l'esecuzione del test, i valori delle variabili e dei parametri di quest'area vengono visualizzati in sequenza cronologica e aggiornati ciclicamente. Se l'area di controllo si trova in una parte di programma che viene eseguita in ogni ciclo, in genere i valori delle variabili non possono essere rilevati da cicli che si susseguono.

I valori che nell'ultimo ciclo hanno subito modifiche e quelli che sono rimasti invariati sono distinti da colori diversi.

Area di controllo

L'ambiente delle istruzioni controllabili dipende dalla potenza di elaborazione della CPU collegata.

Poiché le istruzioni S7-SCL del codice sorgente vengono convertite in numerose istruzioni in codice macchina, la lunghezza dell'area di controllo è variabile e viene determinata e selezionata dal debugger S7-SCL quando viene selezionata la prima istruzione dell'area di controllo desiderata.

È inoltre possibile definire un'area di controllo nel punto desiderato. Selezionare a tale scopo le istruzioni da controllare nella sorgente.

Modalità test

Nella maggior parte dei casi, l'interrogazione di queste informazioni di test comporta un incremento dei tempi di ciclo. Per poter ridurre tale aumento dei tempi di ciclo, S7-SCL offre due diverse modalità.

Modo di funzionamento	Spiegazione
Test	Nel modo di funzionamento "Test", l'area di controllo viene limitata solo dalla potenza di elaborazione della CPU collegata. Le funzioni di test sono utilizzabili senza alcuna limitazione. Si può verificare un incremento del tempo di ciclo della CPU perché, ad esempio, lo stato delle istruzioni dei loop programmati viene rilevato in ogni ciclo.
Processo	Nel modo di funzionamento "Processo", il debugger S7-SCL limita l'area di controllo massima in modo che i tempi di ciclo durante l'esecuzione del test non superino, o superino solo in modo irrilevante, i tempi reali di esecuzione del processo.

Limitazioni

Nell'ambito della funzione "Controlla" occorre prestare attenzione alle seguenti limitazioni:

- Le variabili di un tipo di dati superiore non possono essere visualizzate come un tutt'uno. Gli elementi di tali variabili, se appartengono ad un tipo semplice di dati, sono controllabili;
- Le variabili del tipo DATE_AND_TIME e STRING così come i parametri del tipo BLOCK_FB, BLOCK_FC, BLOCK_DB, BLOCK_SDB, TIMER e COUNTER non vengono visualizzati;
- Gli accessi ai blocchi dati aventi forma: <Symbol>.<absoluteAdresse> non vengono visualizzati (ad es. dati.DW4).

4.10.3 Informazioni importanti sul test con punti d'arresto/in modalità passo singolo

Durante il test con punti d'arresto il test viene eseguito in singoli passi. Si può eseguire l'algoritmo del programma istruzione per istruzione e controllare il cambiamento dei valori delle variabili elaborate.

Dopo l'impostazione di punti d'arresto, si può dapprima far eseguire il programma fino ad un determinato punto d'arresto e a partire da questo punto cominciare con il controllo passo per passo.

Funzioni di esecuzione passo singolo

Una volta avviata la funzione di test "Test con punti d'arresto", è possibile eseguire i seguenti passi.

- Esegui istruzione successiva
Viene eseguita l'istruzione selezionata attualmente.
- Continua
Continua fino al successivo punto d'arresto attivo.
- Esecuzione fino al punto selezionato
Continua fino ad un punto predefinito della sorgente.
- Esegui richiamo
Salto in un blocco sottostante o richiamo di un blocco S7-SCL sottostante.

Punti d'arresto

I punti d'arresto possono essere definiti in qualsiasi punto nella parte istruzioni del testo sorgente.

I punti d'arresto vengono trasferiti al sistema di automazione e attivati solo con la selezione del comando di menu **Test > Punti d'arresto attivi**. Il programma viene quindi eseguito fino al raggiungimento del primo punto di arresto.

Il numero massimo dei punti d'arresto attivati dipende dalla CPU.

Presupposti

La sorgente aperta non deve essere stata modificata nell'editor.

4.10.4 Fasi del controllo

Dopo aver caricato il programma compilato nel sistema di destinazione, è possibile testarlo nel modo "Controlla".

Nota

L'ambiente di istruzioni da testare (area massima di controllo) è determinato dalla potenza di elaborazione della CPU collegata.

Procedere nel modo seguente:

1. Assicurarsi che i presupposti per lo svolgimento del test siano stati soddisfatti e che la CPU si trovi nello stato di funzionamento RUN o RUN-P.
2. Selezionare la finestra che contiene la sorgente del programma da testare.
3. Se si desidera modificare il modo di funzionamento preimpostato (processo) selezionare il comando di menu **Test > Funzionamento > Test**.
4. Definire l'area di controllo scegliendo una di queste due possibilità:
 - Posizionare il cursore sulla riga del testo sorgente che contiene la prima istruzione dell'area da testare. S7-SCL seleziona un'area massima di controllo a partire dal punto in cui è stato posizionato il cursore.
 - Selezionare nel testo sorgente solo quelle istruzioni che si desidera controllare.
5. Accertarsi che non si possano verificare stati pericolosi.
6. Selezionare il comando di menu **Test > Controlla**.
7. Disattivare il comando di menu **Test > Controlla** per interrompere l'esecuzione del test.
8. Selezionare il comando di menu **Test > Concludi test** per terminare il test.

Risultato

L'area di controllo viene definita e visualizzata da una barra grigia sul margine sinistro della finestra. Quest'ultima si suddivide in due e, sulla destra, vengono visualizzati riga per riga i nomi e i valori attuali delle variabili contenute nell'area di controllo.

Adattamento della funzione di controllo

Esistono le seguenti possibilità di adattamento della funzione di controllo a seconda delle diverse esigenze:

- Definire un ambiente di richiamo del blocco da controllare.
- Selezionare il comando di menu **Visualizza > Rappresentazione simbolica** per attivare oppure disattivare nel programma la visualizzazione dei nomi simbolici;
- Selezionare il comando di menu **Strumenti > Impostazioni** ed impostare il colore dei valori nella scheda "Formato".

4.10.4.1 Definizione di un ambiente di richiamo del blocco

Ambiente di richiamo

Per definire in modo ancora più mirato l'area di controllo si può determinare un ambiente di richiamo del blocco da controllare. In tal modo si stabilisce che un blocco venga controllato solo in una delle seguenti condizioni:

Condizione	Possibilità di selezione
Il blocco viene richiamato da un blocco sovraordinato specifico.	Percorso di richiamo
Il blocco viene richiamato insieme ad un blocco dati specifico.	Blocco dati globale e/o blocco dati di istanza

Per definire un percorso di richiamo procedere come segue:

1. Selezionare il comando di menu **Test > Ambiente di richiamo del blocco**:
la finestra di dialogo successiva visualizzerà una lista dei blocchi esistenti;
2. Selezionare un blocco dalla lista;
3. Attivare l'opzione "Attiva percorso di richiamo":
nella parte inferiore della finestra verranno rappresentati graficamente i percorsi di richiamo possibili;
4. Selezionare il percorso di richiamo desiderato.

Per definire un blocco dati procedere come segue:

1. Selezionare il comando di menu **Test > Ambiente di richiamo del blocco**:
la finestra di dialogo successiva visualizzerà una lista dei blocchi esistenti;
2. Selezionare un blocco dalla lista;
3. Attivare l'opzione "Blocchi dati aperti";
4. Indicare il numero del DB desiderato.

Nota

Dopo aver definito un ambiente di richiamo, procedere nel modo seguente per avviare il controllo:

1. Posizionare il cursore nel blocco da controllare, per cui è stato definito l'ambiente di richiamo.
 2. Selezionare il comando di menu **Test > Controlla**.
Viene avviata la funzione di controllo. Il blocco verrà controllato se saranno soddisfatte tutte le condizioni di richiamo definite.
-

4.10.5 Fasi del test con punti di arresto

4.10.5.1 Definizione dei punti d'arresto

Per impostare e definire i punti d'arresto procedere nel seguente modo:

1. Aprire la sorgente da testare.
2. Visualizzare la barra degli strumenti per i punti d'arresto con il comando di menu **Visualizza > Barra dei punti d'arresto**.
3. Impostare il cursore nel punto desiderato e selezionare il comando di menu **Test > Definisci punto d'arresto** o il corrispondente pulsante della barra dei punti d'arresto. I punti di arresto vengono rappresentati con un cerchio rosso sul margine sinistro della finestra.
4. Se lo si desidera, selezionare il comando di menu **Test > Modifica punti d'arresto** e definire un ambiente di richiamo. L'ambiente di richiamo stabilisce se un punto d'arresto verrà attivato solo quando il blocco in cui è contenuto:
 - viene richiamato da un blocco sovraordinato e/o
 - viene richiamato assieme ad un blocco dati/blocco dati di istanza.

4.10.5.2 Avvio del test con i punti d'arresto

Dopo aver caricato il programma compilato nel sistema di destinazione e aver definito i punti d'arresto è possibile testare il programma con la funzione "Test con punti di arresto".

Procedere nel modo seguente:

1. Aprire la sorgente S7-SCL del programma da testare.
2. Assicurarsi che non si verifichino stati pericolosi e che la CPU si trovi nello stato RUN-P.
3. Selezionare il comando di menu **Test > Punti d'arresto attivi**.
Risultato: la finestra viene suddivisa in due in senso verticale. Il programma gira fino al punto di arresto successivo. Una volta raggiunto, la CPU passa allo stato di funzionamento ALT e il punto di arresto rosso viene evidenziato con una freccia gialla.
4. Proseguire come indicato di seguito.
 - Selezionando il comando di menu **Test > Continua** oppure facendo clic sul pulsante "Continua".
 La CPU passerà allo stato RUN. Quando raggiungerà il successivo punto d'arresto attivo, si fermerà nuovamente visualizzando il punto di arresto nella parte sinistra della finestra.
 - Selezionando il comando di menu **Test > Esegui istruzione successiva** o facendo clic sul simbolo "Esegui istruzione successiva".
 La CPU commuta allo stato di funzionamento RUN. Dopo aver elaborato l'istruzione selezionata, essa si arresta nuovamente per visualizzare i contenuti delle variabili momentaneamente in corso di elaborazione sul lato destro della finestra.
 - Selezionando il comando di menu **Test > Esegui fino alla selezione** oppure facendo clic sul simbolo "Esegui fino alla selezione".
 La CPU passerà allo stato RUN. Dopo aver raggiunto il cursore essa si arresta nuovamente.

- Selezionare il comando di menu **Test > Esegui richiamo**, quando il programma viene fermato in una riga contenente un richiamo di blocco.
Se il blocco sottostante è stato creato con S7-SCL, esso viene aperto ed elaborato durante la funzione di test. Dopo l'elaborazione, il programma ritorna al punto di richiamo.
Se il blocco è stato creato in un'altra lingua, tale richiamo viene ignorato e viene selezionata la riga di programma successiva.

Nota

I comandi di menu **Test > Esegui istruzione successiva** o **Test > Esegui fino alla selezione** impostano e attivano implicitamente un punto d'arresto. Quando si seleziona questa funzione ci si deve assicurare che non venga raggiunto il numero massimo di punti d'arresto attivi specifico per ogni CPU.

4.10.5.3 Conclusione del test con i punti d'arresto

Per tornare al normale ciclo del programma procedere nel seguente modo:

- disattivare il comando di menu **Test > Punti d'arresto attivi** per interrompere l'esecuzione del test, oppure
- selezionare il comando di menu **Test > Concludi test** per terminare il test.

4.10.5.4 Attivazione, disattivazione e cancellazione dei punti d'arresto

I punti d'arresto possono essere attivati, disattivati e cancellati singolarmente.

1. Selezionare il comando di menu **Test > Modifica punti d'arresto**.
2. Nella successiva finestra di dialogo è possibile
 - attivare o disattivare i punti d'arresto desiderati, munendoli di un segno di spunta o ricancellando il segno di spunta a seconda dei casi.
 - cancellare singoli punti d'arresto.

Per cancellare tutti i punti d'arresto selezionare il comando di menu **Test > Cancella tutti i punti d'arresto**.

4.10.5.5 Definizione di un ambiente di richiamo per punti d'arresto

Ambiente di richiamo

Definendo un ambiente di richiamo si stabilisce che un punto d'arresto è valido solo in una delle seguenti condizioni:

Condizione	Possibilità di selezione
Il blocco in cui si trova il punto d'arresto viene richiamato da un blocco sovraordinato specifico.	Percorso di richiamo
Il blocco in cui si trova il punto d'arresto viene richiamato insieme ad un blocco dati specifico.	Blocco dati globale e/o blocco dati di istanza

Per definire un percorso di richiamo procedere come segue:

1. Selezionare il comando di menu **Test > Modifica punti d'arresto**:
la finestra di dialogo successiva visualizzerà una lista dei punti d'arresto esistenti;
2. Selezionare un punto d'arresto dalla lista;
3. Attivare l'opzione "Attiva percorso di richiamo":
nella parte inferiore della finestra verranno rappresentati graficamente i percorsi di richiamo possibili;
4. Selezionare il percorso di richiamo desiderato.

Per definire un blocco dati procedere come segue:

1. Selezionare il comando di menu **Modifica punti d'arresto**:
la finestra di dialogo successiva visualizzerà una lista dei punti d'arresto esistenti;
2. Selezionare un punto d'arresto dalla lista;
3. Attivare l'opzione "Blocchi dati aperti";
4. Indicare il numero del blocco dati desiderato.

4.10.5.6 Test in modalità passo singolo

Procedere nel modo seguente:

1. Impostare un punto d'arresto prima della riga di istruzione dalla quale si vuole iniziare il test del programma in modalità passo singolo.
2. Selezionare il comando di menu **Test > Punti d'arresto attivi**.
3. Eseguire il programma fino al punto definito.
4. Per eseguire l'istruzione successiva selezionare il comando di menu **Test > Esegui istruzione successiva**.
 - Se viene richiamato un blocco, il richiamo viene elaborato e si salta alla prima istruzione successiva al richiamo.
 - Il comando di menu **Test > Esegui richiamo** consente di saltare nel blocco da dove si può continuare il test a passo singolo o impostare ulteriori punti d'arresto. Al termine del blocco si passa nuovamente all'istruzione successiva al richiamo del blocco.

4.10.6 Funzioni di test di STEP 7

4.10.6.1 Creazione o visualizzazione dei dati di riferimento

È possibile creare e valutare dati di riferimento per facilitare l'esecuzione del test e delle modifiche del programma utente.

Possono essere visualizzati i dati di riferimento seguenti:

- la struttura del programma utente
- l'elenco dei riferimenti incrociati
- la tabella di occupazione
- l'elenco degli operandi non utilizzati
- l'elenco degli operandi senza simbolo

Creazione dei dati di riferimento

Per la creazione dei dati di riferimento è possibile scegliere tra le seguenti possibilità:

- Mediante il comando di menu **Strumenti > Dati di riferimento** è possibile eventualmente creare o aggiornare e visualizzare i dati.
- Impostando un filtro è possibile limitare la quantità di dati visualizzati e rendere molto più rapida la creazione e la visualizzazione dei dati. Per eseguire questa operazione selezionare il comando di menu **Strumenti > Dati di riferimento > Filtra**.
- Mediante il comando di menu **Strumenti > Impostazioni** è possibile stabilire che vengano generati automaticamente dati di riferimento durante la compilazione di una sorgente. Selezionare a tal fine nella scheda "Crea blocco" la voce "Crea dati di riferimento".

La creazione automatica dei dati di riferimento allunga tuttavia l'operazione di compilazione.

4.10.6.2 Controllo e comando di variabili

Durante il test con la funzione "Controlla e comanda variabili" è possibile:

- fare visualizzare (controllare) i valori attuali di blocchi dati dal proprio programma utente .
- assegnare (comandare) valori fissi alle variabili del programma utente.

Procedere nel modo seguente:

- Selezionare il comando di menu **Sistema di destinazione> Controlla e comanda variabili**.
- Creare la tabella delle variabili (VAT) nella finestra subito dopo visualizzata. Nel caso si desideri comandare variabili, specificare inoltre i valori desiderati.
- Stabilire i punti e le condizioni di trigger.



Attenzione

Un'eventuale modifica dei valori delle variabili durante il funzionamento dell'impianto può causare disturbi di funzionamento o errori di programma con conseguenti gravi danni a persone e cose! Prima di eseguire questa funzione ci si deve assicurare che non si possono verificare stati pericolosi!

4.10.6.3 Verifica coerenza blocchi

Nota

Questa funzione è disponibile a partire dalla versione STEP 7 V5.3 SP2.

Se una sorgente S7-SCL viene modificata, i blocchi a cui vi si fa eventualmente riferimento devono essere anch'essi adattati, altrimenti possono verificarsi incoerenze all'interno del programma. Anche se la data e l'ora delle sorgenti e quelle del blocco non coincidono, è possibile che si originino incoerenze comunque.

Tramite la funzione di STEP 7 "Verifica coerenza blocchi" è possibile individuare tali incoerenze ed eliminarle rapidamente.

In seguito a modifiche del programma è importante avviare una verifica di coerenza di tutte le sorgenti S7-SCL presenti nel progetto. Nel caso di incoerenze che non si riesce ad eliminare automaticamente, occorre eseguire la funzione direttamente nelle posizioni da modificare nella sorgente. Qui si effettuano le modifiche e, passo dopo passo, verranno eliminate tutte le incoerenze riscontrate.

Presupposti

- Sulla propria apparecchiatura deve essere installata la versione STEP 7 V5.3 SP2 o superiore.
- Una sorgente coinvolta nella funzione "Verifica coerenza blocchi" deve essere stata sottoposta al processo di compilazione almeno una volta mediante S7-SCL V5.3 SP1 o versione superiore.
- S7-SCL deve essere installata sull'apparecchiatura su cui si esegue la verifica di coerenza.

Procedere come segue

1. Aprire SIMATIC Manager;
2. Selezionare il contenitore "Blocchi";
3. Selezionare il comando di menu **Modifica > Verifica coerenza blocchi**;
4. Selezionare il comando di menu **Visualizza > Visualizza riferimenti file sorgente S7-SCL**.

Risultato

STEP 7 controlla la data e l'ora e le interfacce di tutti i blocchi e delle relative sorgenti nel contenitore attuale per poi segnalare i seguenti conflitti:

- Conflitti di data e ora tra le sorgenti S7-SCL e i blocchi.
- Riferimenti tra i diversi blocchi e conflitti che ne derivano tra le interfacce.

Nel caso in cui venga individuata un'incoerenza, viene avviata una nuova compilazione. Se la sorgente contiene più sorgenti di blocco, tutti i blocchi che ne derivano verranno generati ex novo. Per la compilazione sono valide le opzioni corrispondenti attualmente impostate.

Nota

Per ulteriori informazioni relative a questa funzione consultare la Guida a STEP 7.

4.11 Visualizzazione e modifica delle proprietà della CPU

4.11.1 Visualizzazione e modifica dello stato di funzionamento della CPU

Si può interrogare e modificare lo stato di funzionamento attuale di una CPU. Il presupposto indispensabile è l'esistenza di un collegamento con la CPU.

Procedere nel modo seguente:

1. Selezionare il comando di menu **Sistema di destinazione > Stato di funzionamento**.
2. Selezionare nella seguente finestra di dialogo uno dei seguenti stati di funzionamento:
 - Nuovo avviamento (avviamento a caldo)
 - Avviamento a freddo
 - Riavviamento
 - STOP



Attenzione

Un'eventuale modifica dei valori delle variabili durante il funzionamento dell'impianto può causare disturbi di funzionamento o errori di programma con conseguenti gravi danni a persone e cose!

Prima di eseguire questa funzione ci si deve assicurare che non si possono verificare stati pericolosi!

4.11.2 Visualizzazione e impostazione di data e ora della CPU

Si può interrogare e modificare l'ora momentanea di una CPU. Il presupposto indispensabile è l'esistenza di un collegamento con la CPU.

Procedere nel modo seguente:

1. Selezionare il comando di menu **Sistema di destinazione > Imposta data e ora**
2. Introdurre nella seguente finestra di dialogo la data e l'ora per l'orologio hardware della CPU.

Se la CPU non dispone di un orologio hardware, nella finestra di dialogo vengono visualizzati "00:00:00" per l'ora e "00.00.00" per la data. Non si può apportare alcuna modifica.

4.11.3 Visualizzazione dei dati della CPU

Si possono visualizzare le seguenti informazioni su una CPU:

- Famiglia di sistema, tipo di CPU, numero di ordinazione e versione della CPU.
- Dimensione della memoria di lavoro e della memoria di caricamento e massima configurabilità possibile della memoria di caricamento.
- Numero e area di indirizzamento ingressi e uscite, temporizzatori, contatori e merker.
- Numero dei dati locali con i quali questa CPU può lavorare.
- Indicazione, se si tratta di una CPU con multiprocessore o non (dipende dalla CPU).

Il presupposto indispensabile è l'esistenza di un collegamento con la CPU.

Procedere nel modo seguente:

1. Selezionare il comando di menu **Sistema di destinazione > Stato dell'unità**.
2. Selezionare nella seguente finestra di dialogo la scheda "Informazioni generali".

4.11.4 Lettura del buffer di diagnostica della CPU

Tramite la lettura del buffer di diagnostica si può stabilire la causa dello stato di funzionamento STOP oppure si possono analizzare le cause che hanno creato determinati eventi di diagnostica.

Il presupposto indispensabile è l'esistenza di un collegamento con la CPU.

Procedere nel modo seguente:

1. Selezionare il comando di menu **Sistema di destinazione > Stato dell'unità**.
2. Selezionare nella seguente finestra di dialogo la scheda "Buffer di diagnostica".

4.11.5 Visualizzazione/compressione della memoria utente della CPU

Questa funzione consente di visualizzare l'utilizzo della memoria CPU ed eventualmente di riorganizzare la memoria CPU. Ciò si rende necessario quando la massima area di memoria contigua non è sufficiente per caricare un nuovo oggetto dal PG alla CPU.

Il presupposto indispensabile è l'esistenza di un collegamento con la CPU.

Procedere nel modo seguente:

1. Selezionare il comando di menu **Sistema di destinazione > Stato dell'unità**.
2. Selezionare nella seguente finestra di dialogo la scheda "Memoria".

4.11.6 Visualizzazione del tempo di ciclo della CPU

I tempi seguenti vengono visualizzati nell'ambito dei due valori limiti parametrizzabili:

- Durata del ciclo più lungo dall'ultimo passaggio da STOP a RUN.
- Durata del ciclo più breve dall'ultimo passaggio da STOP a RUN.
- Durata dell'ultimo ciclo.

Quando la durata dell'ultimo ciclo si avvicina al tempo di controllo, può capitare che quest'ultimo venga superato e la CPU commuti nello stato di funzionamento STOP. Si può avere un incremento del tempo di ciclo per esempio quando si testano dei blocchi allo stato di programma. Il presupposto indispensabile per la visualizzazione dei tempi di ciclo del programma utente è l'esistenza di un collegamento con la CPU.

Procedere nel modo seguente:

1. Selezionare il comando di menu **Sistema di destinazione > Stato dell'unità**.
2. Selezionare nella seguente finestra di dialogo la scheda "Tempo di ciclo".

4.11.7 Visualizzazione delle caratteristiche dell'orologio della CPU

La finestra "Caratteristiche orologio" relativa all'unità centrale (CPU) comprende informazioni sull'orologio interno e sulla sincronizzazione degli orologi interni di molte CPU.

Il presupposto indispensabile è l'esistenza di un collegamento con la CPU.

Procedere nel modo seguente:

1. Selezionare il comando di menu **Sistema di destinazione > Stato dell'unità**.
2. Selezionare nella seguente finestra di dialogo la scheda "Caratteristiche orologio".

4.11.8 Visualizzazione dei blocchi della CPU

È possibile visualizzare i blocchi disponibili online per la CPU.

Il presupposto indispensabile è l'esistenza di un collegamento con la CPU.

Procedere nel modo seguente:

1. Selezionare il comando di menu **Sistema di destinazione > Stato dell'unità**.
2. Nella finestra di dialogo visualizzata selezionare la scheda "Dati utili".

4.11.9 Visualizzazione delle informazioni per la comunicazione con la CPU

Per ciascuna CPU, si possono visualizzare online le informazioni su baudrate impostato e massimo, collegamenti e carico di comunicazione.

Il presupposto indispensabile è l'esistenza di un collegamento con la CPU.

Procedere nel modo seguente:

1. Selezionare il comando di menu **Sistema di destinazione > Stato dell'unità**.
2. Selezionare nella seguente finestra di dialogo la scheda "Comunicazione".

4.11.10 Visualizzazione degli stack della CPU

Con questa scheda si possono visualizzare online per ciascuna unità centrale (CPU) le informazioni sul contenuto degli stack. La CPU deve trovarsi allo stato di funzionamento STOP o deve aver raggiunto un punto d'arresto.

La visualizzazione degli stack rappresenta un valido aiuto per localizzare gli errori, p. es. durante il test dei blocchi. Se la CPU si trova nello stato di funzionamento STOP, nello stack delle interruzioni (U-Stack) si possono visualizzare il punto di interruzione con le visualizzazioni attuali e i contenuti degli accumulatori, per poter identificare la causa (p. es. di errori di programmazione).

Il presupposto indispensabile è l'esistenza di un collegamento con la CPU.

Procedere nel modo seguente:

1. Selezionare il comando di menu **Sistema di destinazione > Stato dell'unità**.
2. Selezionare nella seguente finestra di dialogo la scheda "Stack".

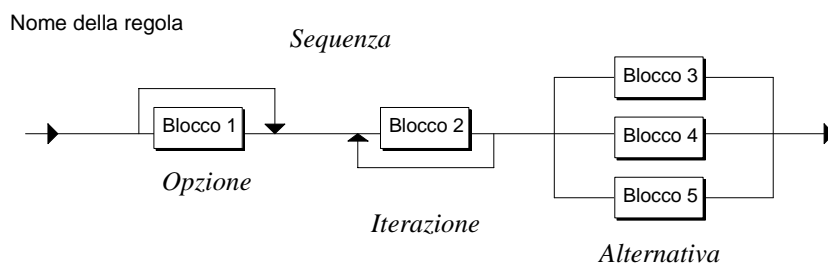
5 Concetti di base S7-SCL

5.1 Interpretazione dei diagrammi sintattici

La base per la descrizione del linguaggio è costituita dai diagrammi sintattici. Essi offrono una chiara panoramica della struttura sintattica di S7-SCL. Il capitolo "Descrizione del linguaggio" contiene un riepilogo di tutti i diagrammi con gli elementi linguistici.

Che cos'è un diagramma sintattico?

Il diagramma sintattico è una rappresentazione grafica della struttura del linguaggio. La struttura è costituita da una sequenza gerarchica di regole. Ogni regola a sua volta è presente come blocco in una regola sottostante.

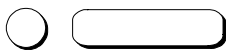


Il diagramma sintattico va letto da destra verso sinistra tenendo conto delle seguenti strutture:

- Sequenza: sequenza di blocchi
- Opzione: ramo saltabile
- Iterazione: ripetizione di rami
- Alternativa: diramazione

Quali tipi di blocchi esistono?

Un blocco è un elemento base oppure un elemento che a sua volta si compone di vari blocchi. La figura seguente mostra i tipi di simboli corrispondenti ai vari blocchi:



Elemento base che non deve essere ulteriormente descritto. Si tratta di caratteri stampabili e di caratteri speciali, di parole chiave e di identificatori predefiniti. Le indicazioni su questi blocchi devono essere accettate senza alcuna modifica.



Elemento composto che viene descritto da ulteriori diagrammi sintattici.

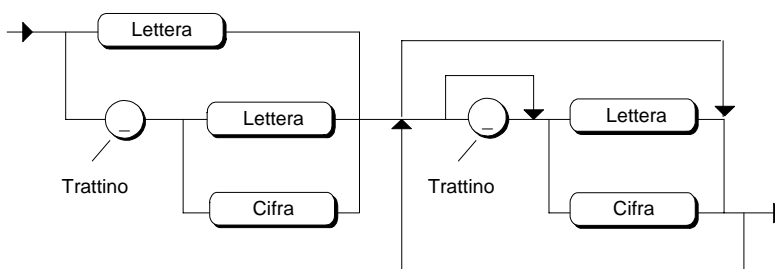
Che cosa significa libertà di formato?

Quando si immettono i testi sorgente si devono osservare sia le **regole sintattiche** che le **regole lessicali**.

Entrambi i gruppi vengono descritti nel capitolo "Descrizione del linguaggio". La libertà di formato significa che fra i blocchi vuoti si possono inserire caratteri di formattazione come spazi, tabulatori, cambi pagina e commenti.

Con le regole lessicali ciò non è consentito. Quando si utilizza una regola lessicale la si deve assumere senza apportarvi modifiche.

Regola lessicale



In base alla regola illustrata sono esempi validi:

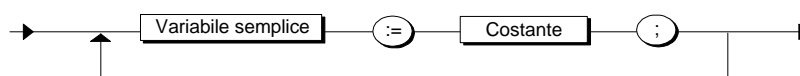
```
R_REGOLATORE3
_A_CAMPO
_100_3_3_10
```

In base alla regola illustrata non sono validi gli esempi:

```
1_1AB
RR__20
*#AB
```

Esempio di regola sintattica

Nelle regole sintattiche l'utente ha libertà di formato.



In base alla regola illustrata, sono esempi validi:

```
VARIABILE_1      := 100;      INTERRUETTORE:=FALSE;  
VARIABILE_2      := 3.2;
```

5.2 Set di caratteri

Lettere e cifre

Dall'area parziale del set di caratteri ASCII, S7-SCL utilizza:

- le lettere (maiuscole e minuscole) dalla A alla Z,
- i numeri arabi da 0 a 9,
- gli spazi - lo spazio stesso (valore ASCII 32) e tutti i caratteri di controllo (ASCII 0-31) compreso il carattere di fine riga (ASCII 13).

Altri caratteri

In S7-SCL questi caratteri hanno un significato preciso:

+	-	*	/	=	<	>	[]	()
:	;	\$	#	"	'	{	}	%	.	,

Nota

Il capitolo "Descrizione del linguaggio" riporta un elenco dettagliato dei caratteri utilizzabili e ne descrive l'utilizzo in S7-SCL.

5.3 Parole riservate

Le parole riservate sono parole chiave che devono essere usate esclusivamente nel modo prescritto. Non viene fatta alcuna distinzione fra lettere maiuscole e minuscole.

Parole chiave in S7-SCL

AND	END_CASE	ORGANIZATION_BLOCK
ANY	END_CONST	POINTER
ARRAY	END_DATA_BLOCK	PROGRAM
AT	END_FOR	REAL
BEGIN	END_FUNCTION	REPEAT
BLOCK_DB	END_FUNCTION_BLOCK	RETURN
BLOCK_FB	END_IF	S5TIME
BLOCK_FC	END_LABEL	STRING
BLOCK_SDB	END_TYPE	STRUCT
BLOCK_SFB	END_ORGANIZATION_BLOCK	THEN
BLOCK_SFC	END_REPEAT	TIME
BOOL	END_STRUCT	TIMER
BY	END_VAR	TIME_OF_DAY
BYTE	END_WHILE	TO
CASE	ENO	TOD
CHAR	EXIT	TRUE
CONST	FALSE	TYPE
CONTINUE	FOR	VAR
COUNTER	FUNCTION	VAR_TEMP
DATA_BLOCK	FUNCTION_BLOCK	UNTIL
DATE	GOTO	VAR_INPUT
DATE_AND_TIME	IF	VAR_IN_OUT
DINT	INT	VAR_OUTPUT
DIV	LABEL	VOID
DO	MOD	WHILE
DT	NIL	WORD
DWORD	NOT	XOR
ELSE	OF	Nomi delle funzioni standard
ELSIF	OK	
EN	OR	

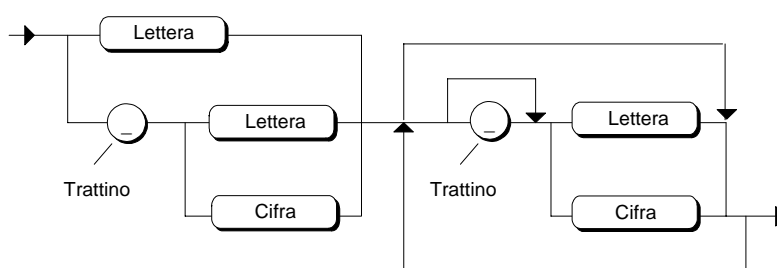
5.4 Identificatori

Definizione

Un identificatore è un nome che l'utente può assegnare ad un oggetto linguistico di S7-SCL, cioè ad una costante, una variabile e un blocco.

Regole

Gli identificatori possono essere composti da un massimo di 24 lettere o cifre in una sequenza qualsiasi in cui solo il primo carattere deve essere una lettera o un trattino. Sono ammesse sia lettere maiuscole che minuscole. Anche in questo caso non viene fatta una distinzione tra maiuscole e minuscole (Anna e AnNa, p. es., saranno identici).



Esempi

I nomi seguenti sono alcuni esempi di identificatori standard.

X	y12
Somma	Temperatura
Nome	Superficie
Regolatore	Tabelle

I nomi seguenti non sono identificatori validi per i motivi indicati.

4	//Il primo carattere deve essere una lettera //o un trattino.
Array	//ARRAY è una parola chiave
Valore S	//Gli spazi non sono consentiti (si deve //ricordare che uno spazio è un carattere).

Nota:

- Assicurarsi che il nome non sia già occupato da parole chiave o da identificatori standard.
- Si consiglia di scegliere nomi univoci ed espressivi per facilitare la comprensione del testo sorgente.

5.5 Identificatori standard

In S7-SCL è già stata definita una serie di identificatori definiti "identificatori standard". Questi identificatori standard sono:

- gli identificatori di blocchi,
- gli identificazioni di operandi per indirizzare le aree di memoria della CPU,
- gli identificatori di temporizzatori, e
- gli identificatori di contatori.

5.6 Identificatori di blocchi

Definizione

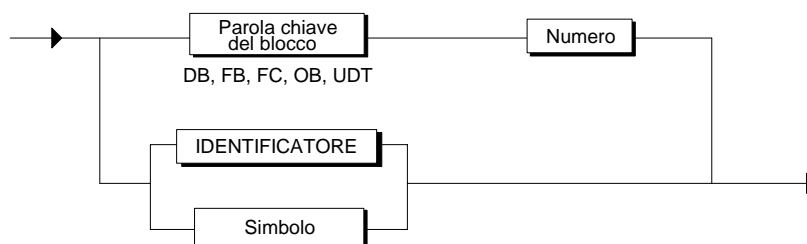
Questi identificatori standard vengono utilizzati per l'indirizzamento assoluto dei blocchi.

Regole

La tabella è ordinata in base al mnemonico SIMATIC e indica anche il corrispondente mnemonico IEC. La lettera x qui viene usata come carattere sostitutivo per un numero compreso fra 0 e 65533 oppure tra 0 e 65535 per temporizzatori e contatori.

Mnemonico (SIMATIC)	Mnemonico IEC	identifica
DBx	DBx	Blocco dati (Data-Block) L'identificatore DB0 è riservato da S7-SCL.
FBx	FBx	Blocco funzionale (Function-Block)
FCx	FCx	Funzione (Function)
OBx	OBx	Blocco organizzativo (Organization-Block)
SDBx	SDBx	Blocco dati di sistema (System-Data-Block)
SFCx	SFCx	Funzione di sistema (System-Function)
SFBx	SFBx	Blocco funzionale di sistema (System-Function-Block)
Tx	Tx	Temporizzatore (Timer)
UDTx	UDTx	Tipo di dati globali o definiti dall'utente (Userdefined Data-Type)
Zx	Cx	Contatori (Counter)

S7-SCL offre diverse possibilità per l'introduzione del nome del blocco. Come numero del blocco si può introdurre un numero intero decimale.



Esempio

Sono identificatori validi:

FB10
DB100
T141

5.7 Identificatori di operandi

Definizione

Le aree di memoria di una CPU possono essere indirizzate da qualsiasi posizione del programma utilizzando il relativo identificatore di operando.

Regole

La tabella è ordinata in base al mnemonico SIMATIC e indica anche il corrispondente mnemonico IEC. Gli identificatori degli operandi per i blocchi dati sono validi solo assieme all'indicazione del blocco dati.

Mnemonico (SIMATIC)	Mnemonico (IEC)	Elemento indirizzato	Tipo di dati
Ax.y	Qx.y	Uscita (tramite l'immagine di processo)	Bit
ABx	QBx	Uscita (tramite l'immagine di processo)	Byte
ADx	QDx	Uscita (tramite l'immagine di processo)	Doppia parola
AWx	QWx	Uscita (tramite l'immagine di processo)	Parola
AXx.y	QXx.y	Uscita (tramite l'immagine di processo)	Bit
Dx.y	Dx.y	Blocco dati	Bit
DBx	DBx	Blocco dati	Byte
DDx	DDx	Blocco dati	Doppia parola
DWx	DWx	Blocco dati	Parola
DXx.y	DXx.y	Blocco dati	Bit
Ex.y	Ix.y	Ingresso (tramite l'immagine di processo)	Bit
EBx	IBx	Ingresso (tramite l'immagine di processo)	Byte
EDx	IDx	Ingresso (tramite l'immagine di processo)	Doppia parola
EWx	IWx	Ingresso (tramite l'immagine di processo)	Parola
EXx.y	IXx.y	Ingresso (tramite l'immagine di processo)	Bit
Mx.y	Mx.y	Merker	Bit
MBx.y	MBx.y	Merker	Byte
MDx	MDx	Merker	Doppia parola
MWx.y	MWx	Merker	Parola
MXx	MXx	Merker	Bit
PABx	PQBx	Uscita (periferia diretta)	Byte
PADx	PQDx	Uscita (periferia diretta)	Doppia parola
PAWx	PQWx	Uscita (periferia diretta)	Parola
PEBx	PIBx	Ingresso (periferia diretta)	Byte
PEDx	PIDx	Ingresso (periferia diretta)	Doppia parola
PEWx	PIWx	Ingresso (periferia diretta)	Parola

x = Numero compreso fra 0 e 65535 (indirizzo assoluto)

y = Numero compreso fra 0 e 7 (numero di bit)

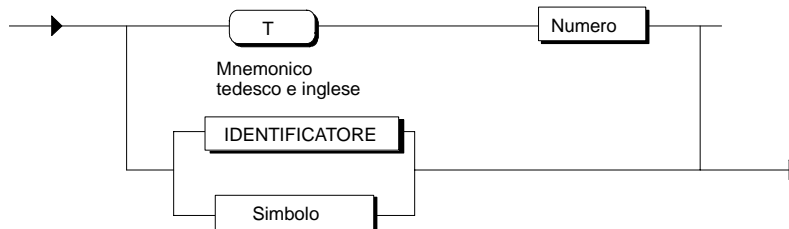
Esempio

E1.0 MW10 PAW5 DB20.DW3

5.8 Identificatori di temporizzatori

Regole

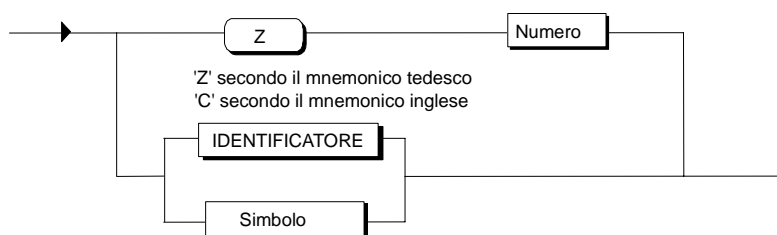
S7-SCL offre diverse possibilità per l'introduzione di un temporizzatore. Come numero del temporizzatore si può introdurre un numero intero decimale.



5.9 Identificatori di contatori

Regole

S7-SCL offre diverse possibilità per l'introduzione di un contatore. Come numero del contatore si può introdurre un numero intero decimale.



5.10 Numeri

S7-SCL supporta diverse modalità di scrittura dei numeri. Le seguenti osservazioni valgono per tutti i numeri:

- Un numero può avere, a scelta, un segno, un punto decimale e un esponente.
- Un numero non può contenere né virgole né spazi.
- Per indicare una separazione si può utilizzare un tratto di sottolineatura (_).
- Un numero può essere preceduto a scelta da un segno positivo (+) o negativo (-). Se non è preceduto da un segno viene considerato positivo.
- I numeri non devono essere superare determinati valori massimi e minimi.

Numeri interi

I numeri interi non contengono né virgole (punti) decimali né esponenti e sono una semplice sequenza di cifre che può essere preceduta da un segno matematico. In S7-SCL vengono utilizzati 2 tipi di numeri interi caratterizzati dai campi di valori INT e DINT.

Seguono alcuni numeri interi validi:

0	1	+1	-1
743	-5280	600_00	-32_211

Per le ragioni sopra descritte **errati**:

123,456	Non sono ammesse virgole.
36.	Un numero intero non può contenere un punto decimale.
10 20 30	Non sono consentiti spazi.

S7-SCL consente di rappresentare i numeri interi in diversi sistemi numerici ponendo prima del numero la parola chiave specifica del sistema scelto. Ad esempio 2# indica il sistema binario, 8# il sistema ottale e 16# il sistema esadecimale.

Seguono alcuni numeri interi validi per il numero decimale 15:

2#1111 8#17 16#F

Numeri in virgola mobile

Un numero in virgola mobile deve contenere un punto decimale o un esponente (o entrambi). Poiché il punto deve trovarsi fra due cifre, i numeri in virgola mobile non possono iniziare e terminare con un punto.

Seguono alcuni numeri in virgola mobile validi:

0.0	1.0	-0.2	827.602
50000.0	-0.000743	12.3	-315.0066

I seguenti numeri reali sono **errati**:

1.	Il punto decimale deve essere situato fra due cifre.
1,000.0	Non sono ammesse virgole.
.3333	Il punto decimale deve essere situato fra due cifre.

Per definire la posizione del punto decimale si può indicare un esponente. Se il punto non è presente si presuppone che sia situato a destra della cifra. L'esponente deve essere un numero intero positivo o negativo. La base 10 viene sostituita con la lettera E.

La grandezza 3×10 esponente 10 può essere rappresentata in S7-SCL mediante i seguenti numeri reali:

3.0E+10	3.0E10	3e+10	3E10
0.3E+11	0.3e11	30.0E+9	30e9

I seguenti numeri reali sono **errati**:

3.E+10	Il punto decimale deve essere situato fra due cifre.
8e2.3	L'esponente deve essere un numero intero.
.333e-3	Il punto decimale deve essere situato fra due cifre.
30 E10	Non sono consentiti spazi.

5.11 Stringhe di caratteri

Definizione

Una stringa di caratteri è una sequenza di caratteri (cioè lettere, cifre e caratteri speciali) racchiusa fra virgolette.

Seguono alcune stringhe di caratteri valide :

'ROSSO' '76181 Karlsruhe' '270-32-3456'
'DM19.95' 'La risposta esatta è:'

Regole

I caratteri speciali di formattazione, le virgolette (') e il carattere \$ possono essere specificati utilizzando il simbolo \$.

Testo sorgente	Dopo la compilazione
'SEGNALE\$'ROSSO\$''	SEGNALE'ROSSO'
'50.0\$\$'	50.0\$
'VALORE\$P'	VALORE <i>Salto pagina</i>
'REG-\$L'	REG-A <i>capo automatico</i>
'REGOLATORE\$R	REGOLATORE <i>Ritorno</i> <i>carrello</i>
'FASE\$T'	FASE <i>Tabulatore</i>

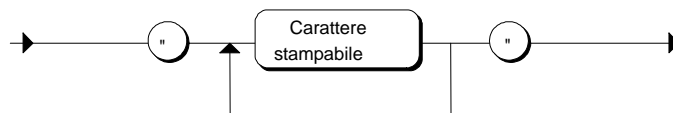
Per i caratteri non stampabili si deve ricorrere al formato esadecimale utilizzando la formula \$hh, dove hh indica il valore esadecimale del carattere ASCII.

Per interrompere una stringa di caratteri (per commenti che non devono essere stampati o visualizzati) in S7-SCL sono disponibili i caratteri \$> e \$< che consentono l'interruzione di una stringa.

5.12 Simbolo

In S7-SCL i simboli devono essere specificati secondo la sintassi descritta di seguito. Le virgolette sono necessarie solo se il simbolo non è conforme alla regola IDENTIFICATORE.

Sintassi:



5.13 Blocco di commenti

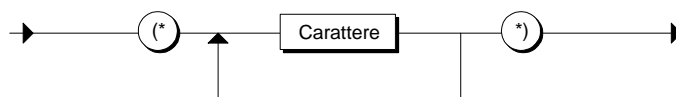
Regole

- Il blocco di commenti può comprendere più righe e deve iniziare con "(" e terminare con ")".
- Per default l'annidamento dei blocchi di commento è ammesso, ma può essere disattivato modificando la relativa impostazione.
- Un commento non deve interrompere né un nome simbolico né una costante. Però è consentita l'interruzione di stringhe.

Sintassi

Il blocco di commenti viene rappresentato mediante il seguente diagramma sintattico:

Blocco di commento



Esempio

(* Questo è un esempio di blocco di commenti
che può comprendere più righe.*)

```
INTERRUPTORE := 3 ;  
END_FUNCTION_BLOCK
```

5.14 Commento di una riga

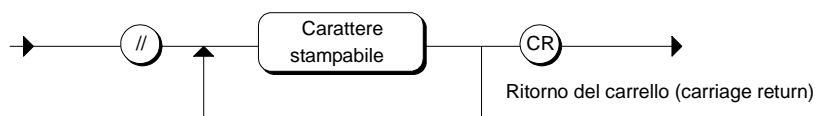
Regole

- Il commento di una riga è preceduto dai caratteri "//" e si estende fino alla fine della riga.
- La lunghezza del commento non deve superare i 254 caratteri compresi i caratteri introduttivi "//".
- Un commento non deve interrompere né un nome simbolico né una costante. Però è consentita l'interruzione di stringhe.

Sintassi

Il commento di una riga viene rappresentato con il seguente diagramma sintattico:

Commento a una riga



Esempio

```
VAR  
    INTERRUETTORE : INT ; // Commento di una riga  
END_VAR
```

Nota:

- I commenti della parte convenzioni preceduti da // vengono inseriti nell'interfaccia del blocco e possono essere quindi visualizzati anche negli editor KOP/AWL/FUP.
 - Per i caratteri stampabili si rimanda al capitolo "Descrizione del linguaggio".
 - All'interno del commento di una riga i caratteri "(" e ")" non hanno alcuna rilevanza.
-

5.15 Variabili

Un identificatore il cui valore può essere modificato durante l'esecuzione del programma viene denominato Variabile. Ogni variabile deve essere dichiarata (ossia definita) singolarmente prima di poter essere utilizzata in un blocco di codice o in un blocco dati. Nella dichiarazione delle variabili si definisce che un identificatore è una variabile (e non una costante, ecc.) e si specifica il tipo di variabile mediante associazione ad un tipo di dati.

A seconda della validità delle variabili si distinguono:

- dati locali
- dati utente globali
- variabili consentite e predefinite (aree di memoria nella CPU)

Dati locali

I dati locali sono dati utilizzati nell'ambito di un blocco di codice (FC, FB, OB) e sono validi solo per questo blocco di codice. Si tratta in particolare dei dati seguenti:

Variabili	Significato
Variabili statiche	Le variabili statiche sono variabili locali i cui valori rimangono inalterati in tutte le esecuzioni dei blocchi (memoria dei blocchi). Tale variabile serve per memorizzare i valori di un blocco funzionale.
Variabili temporanee	Le variabili temporanee fanno parte di un blocco di codice e non occupano alcuna area di memoria statica. Il loro valore rimane costante solo durante l'esecuzione di un blocco. Al di fuori del blocco in cui sono state dichiarate le variabili non è possibile accedere alle variabili temporanee.
Parametri del blocco	Parametri del blocco sono parametri formali di un blocco funzionale o di una funzione. Si tratta di variabili locali che servono per trasferire i parametri attuali indicati al momento del richiamo.

Dati utente globali

I dati utente globali sono dati o aree di dati utilizzabili in qualsiasi parte del programma. A tal fine si devono creare blocchi dati (DB).

Quando si crea un DB la sua struttura viene definita in un'apposita convenzione. Al posto di una convenzione di struttura si può anche utilizzare un tipo di dati definito dall'utente (UDT). La sequenza in cui l'utente introduce i componenti della struttura determina la sequenza dei dati nel DB.

Aree di memoria di una CPU

Alle aree di memoria di una CPU si può accedere tramite gli identificatori di operandi direttamente da un punto qualsiasi del programma, senza dover prima definire queste variabili.

Si ha inoltre la possibilità di indirizzare queste aree di memoria anche simbolicamente. In tal caso l'assegnazione dei simboli avviene in modo globale nella tabella dei simboli in STEP 7.

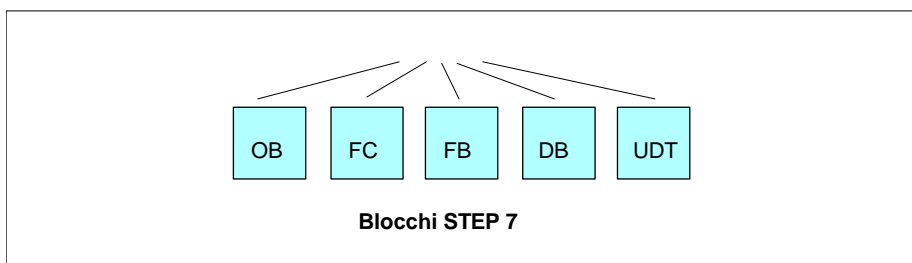
6 Struttura del programma S7-SCL

6.1 Blocchi nelle sorgenti S7-SCL

In un file sorgente S7-SCL si possono programmare da 1 a n blocchi. Per la loro funzione, struttura o tipo d'impiego, i blocchi STEP7 sono parti ben definite di un programma utente.

Tipi di blocco

Sono disponibili i seguenti tipi di blocco:



Blocchi predefiniti

L'utente non deve programmare ogni singola funzione ma ha la possibilità di utilizzare anche blocchi predefiniti. Essi sono presenti nel sistema operativo dell'unità centrale o nelle librerie (*S7lib*) del pacchetto base STEP 7 e possono essere utilizzati p. es. per programmare le funzioni di comunicazione.

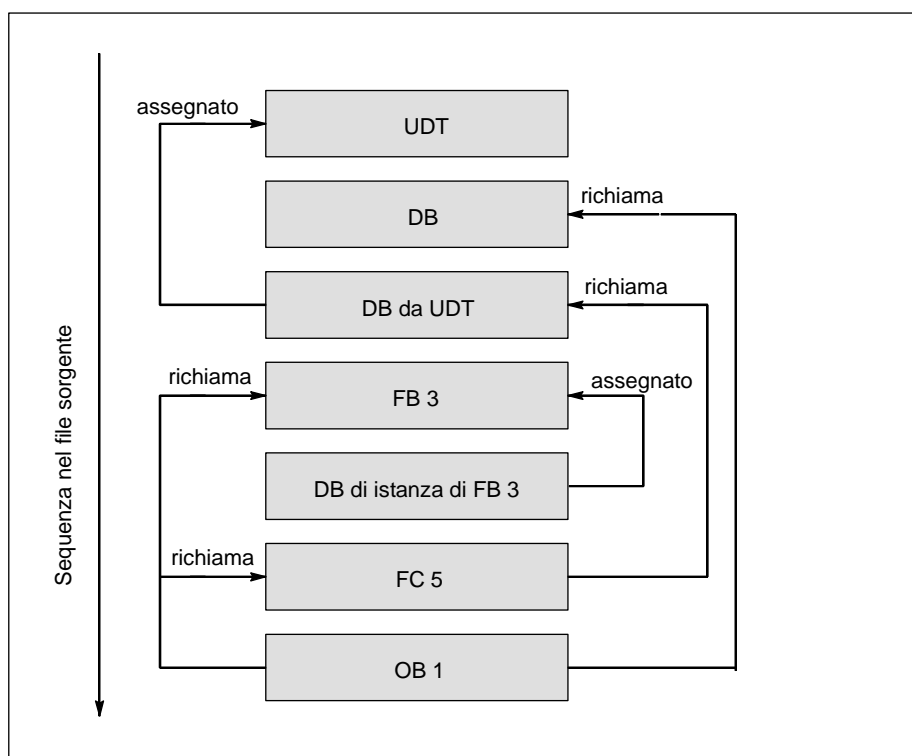
6.2 Ordine dei blocchi

In generale vale la seguente regola:

i blocchi richiamati si trovano prima dei blocchi richiamanti.

Più precisamente:

- I tipi di dati definiti dall'utente (UDT) sono situati prima dei blocchi in cui vengono utilizzati.
- I blocchi dati ai quali è stato assegnato un tipo di dati definito dall'utente (UDT) sono situati dopo l'UDT.
- I blocchi dati a cui è possibile accedere da tutti i blocchi di codice sono situati prima di tutti i blocchi dati che accedono ad essi.
- I blocchi dati con blocco funzionale assegnato sono situati dietro il blocco funzionale.
- Il blocco organizzativo OB1 il quale richiama altri blocchi, è situato alla fine dei blocchi. A loro volta, i blocchi che vengono richiamati da blocchi richiamati dall'OB1 devono essere situati prima di questi ultimi blocchi.
- I blocchi che vengono richiamati nel file sorgente ma che non vengono programmati nello stesso file sorgente devono essere già presenti al momento della compilazione del file nel relativo programma utente.
- Oltre ai blocchi, le sorgenti S7-SCL possono contenere anche indicazioni relative alle impostazioni di compilazione con cui i singoli blocchi devono essere compilati. Le opzioni di compilazione si trovano al di fuori dei limiti del blocco.



6.3 Struttura generica del blocco

I blocchi sono costituiti dalle seguenti parti:

- Inizio blocco, contrassegnato da una parola chiave e da un numero o un nome simbolico, p. es. "BLOCCO_ORGANIZZATIVO OB1" per un blocco organizzativo
Per le funzioni viene indicato anche il tipo che determina il tipo di dati del valore di ritorno. Se non è previsto un valore di ritorno si deve indicare la parola chiave VOID.
- Titolo di blocco opzionale introdotto dalla parola chiave "TITLE ="
- Commento al blocco opzionale. Il commento al blocco può comprendere più righe ognuna delle quali deve iniziare con "//".
- Attributi di blocco (opzionali)
- Attributi di sistema per i blocchi (opzionali)
- Parte convenzioni (diversa a seconda del tipo di blocco)
- Parte istruzioni nei blocchi di codice o assegnazione di valori attuali in blocchi dati (opzionale)
- Nei blocchi di codice: istruzioni
- Fine del blocco, contrassegnata da END_ORGANIZATION_BLOCK, END_FUNCTION_BLOCK o END_FUNCTION.

6.4 Inizio e fine di un blocco

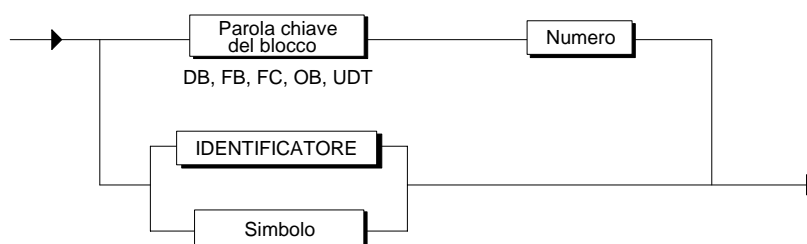
In funzione del tipo di blocco, il testo sorgente di un blocco viene contrassegnato con un identificatore standard di inizio blocco e da un nome. Il testo viene concluso con un identificatore di fine blocco.

La tabella seguente mostra la sintassi dei tipi di blocchi:

Nome del blocco	Tipo di blocco	Sintassi
Blocco funzionale	FB	FUNCTION_BLOCK fb_name ... END_FUNCTION_BLOCK
Funzione	FC	FUNCTION fc_name : tipo di funzione ... END_FUNCTION
Blocco organizzativo	OB	ORGANIZATION_BLOCK ob_name ... END_ORGANIZATION_BLOCK
Blocco dati	DB	DATA_BLOCK db_name ... END_DATA_BLOCK
Tipo di dati globale	UDT	TYPE udt_name ... END_TYPE

Nome del blocco

Nella tabella *xx_nome* indica il nome del blocco conformemente alla sintassi seguente:



Il numero del blocco può essere compreso fra 0 e 65533, il nome di blocco DB0 è riservato.

Si noti inoltre che gli identificatori e i simboli devono essere definiti nella tabella dei simboli di STEP 7.

Esempio

```
FUNCTION_BLOCK FB10
FUNCTION_BLOCK Bloccoregolatore
FUNCTION_BLOCK "Regolatore.B1&U2"
```

6.5 Attributi di blocco

Definizione

Gli attributi definiscono le proprietà del blocco quali il titolo, la versione e l'autore. Le proprietà possono essere visualizzate in una finestra delle proprietà di STEP 7 durante la selezione di blocchi per il caso di applicazione specifico.

Possono essere assegnati gli attributi seguenti:

Parola chiave / Attributo	Significato	Esempi
TITLE='carattere stampabile'	Titolo del blocco	TITLE='ORDINAMENTO'
VERSION : 'Sequenza di cifre decimali. Sequenza di cifre decimali'	Numero di versione del blocco (0 ..15) Nota: Con blocchi dati (DB), l'attributo VERSION non viene indicato tra virgolette.	VERSION : '3.1' //Con DB: VERSION : 3.1
KNOW_HOW_PROTECT	Protezione del blocco; un blocco che è stato compilato con questa opzione non può essere aperto con STEP 7.	KNOW_HOW_PROTECT
AUTHOR :	Nome dell'autore: nome della ditta, sigla dell'ufficio o altri nomi (IDENTIFICATORI e caratteri stampabili)	AUTHOR : Siemens AUTHOR : 'A&D AS'
NAME :	Nome del blocco (IDENTIFICATORI e caratteri stampabili)	NAME : PID NAME : 'A&D AS'
FAMILY :	Nome della famiglia di blocchi: p. es. motori. Salva il blocco in un gruppo di blocchi, per consentirne il rapido reperimento (IDENTIFICATORI e caratteri stampabili).	FAMILY : Esempio FAMILY : 'A&D AS'

Regole

- Gli attributi vengono impostati mediante parole chiave subito dopo l'istruzione di inizio blocco.
- Gli identificatori possono avere una lunghezza massima di 8 caratteri.

Gli attributi devono essere conformi alla seguente sintassi:

Titolo



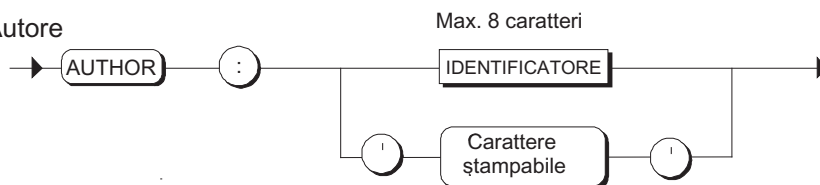
Versione



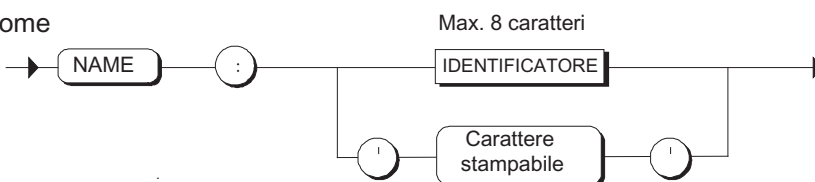
Protezione del blocco



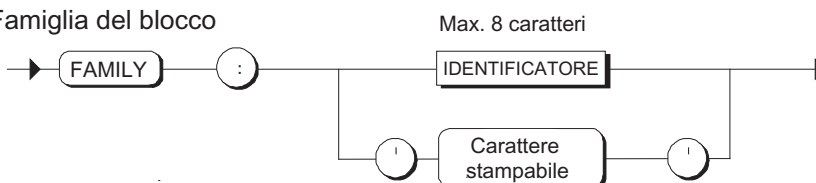
Autore



Nome



Famiglia del blocco



Esempi

```

FUNCTION_BLOCK FB10
TITLE = 'Valore medio'
VERSION : '2.1'
KNOW_HOW_PROTECT
AUTHOR : AUT_1
  
```


6.6 Commento al blocco

I commenti relativi all'intero blocco possono essere specificati nell'intestazione dopo la riga "TITLE:" selezionando il commento di riga. Se il commento comprende più righe ogni riga deve iniziare con //.

Il commento al blocco compare, ad esempio, nella finestra Proprietà del blocco e negli editor KOP/AWL/FUP.

Esempio

```
FUNCTION_BLOCK FB15
TITLE=MIO_BLOCCO
//Questo è un commento al blocco.
//Viene immesso dopo i commenti di riga e può
//essere ad es. visualizzato in SIMATIC Manager.
AUTHOR...
FAMILY...
```

6.7 Attributi di sistema per i blocchi

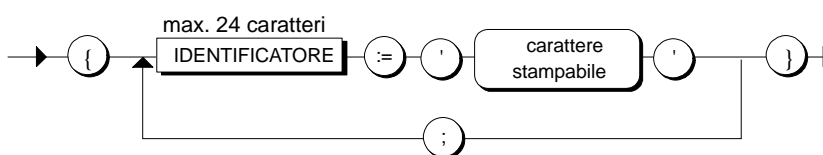
Definizione

Gli attributi di sistema sono attributi validi in più applicazioni del sistema. Gli attributi di sistema per i blocchi valgono per tutto il blocco.

Regole

- Questi attributi vengono impostati direttamente dopo l'istruzione di inizio blocco.
- Attenersi alla seguente sintassi:

Attributi di sistema per blocchi



Esempi

```
FUNCTION_BLOCK FB10
{S7_m_c := 'true' ;
S7_blockview := 'big'}
```

6.8 Parte convenzioni

Definizione

La parte convenzioni serve per definire le convenzioni per variabili, parametri, costanti ed etichette di salto locali.

- Le variabili, i parametri, le costanti e le etichette di salto locali che devono essere validi solo in un blocco vengono definiti nella parte convenzioni del blocco di codice.
- Le aree di dati che devono poter essere indirizzate da qualsiasi blocco di codice vengono definite nella parte convenzioni del blocco dati.
- Nella parte convenzioni di un UDT viene definito un tipo di dati definito dall'utente.

Struttura

Una parte convenzioni è suddivisa in vari blocchi convenzioni, i quali sono contrassegnati da una propria coppia di parole chiavi. Ogni blocco contiene un elenco dichiarazioni per dati simili. La sequenza dei blocchi è irrilevante. La tabella seguente mostra i possibili blocchi convenzioni:

Dati	Sintassi	FB	FC	OB	DB	UDT
Costanti	CONST; Elenco dichiarazioni END CONST	X	X	X		
Etichette di salto	LABEL; Elenco dichiarazioni END LABEL	X	X	X		
Variabili temporanee	VAR_TEMP; Elenco dichiarazioni END VAR	X	X	X		
Variabili statiche	VAR; Elenco dichiarazioni END VAR	X	X *)		X **)	X **)
Parametri d'ingresso	VAR_INPUT; Elenco dichiarazioni END VAR	X	X			
Parametri d'uscita	VAR_OUTPUT; Elenco dichiarazioni END VAR	X	X			
Parametri di ingresso/uscita	VAR_IN_OUT; Elenco dichiarazioni END VAR	X	X			

*) La dichiarazione di variabili all'interno della coppia di parole chiave VAR e END_VAR è ammessa anche per le funzioni; tuttavia, con la compilazione di una sorgente, le dichiarazioni vengono spostate nell'area temporanea.

**) Nei DB e UDT le parole chiave VAR e END_VAR vengono sostituite da STRUCT e END_STRUCT.

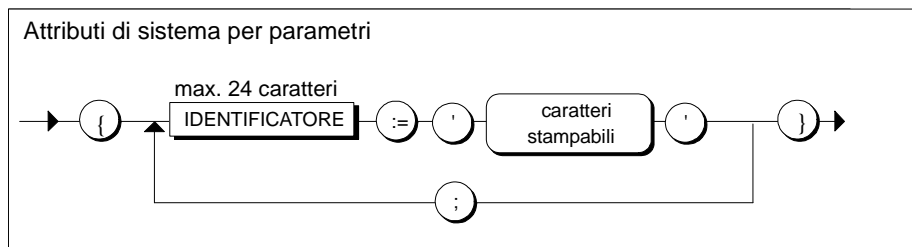
6.9 Attributi di sistema per i parametri

Definizione

Gli attributi di sistema sono attributi validi in più applicazioni del sistema e vengono utilizzati, ad esempio, per la progettazione di messaggi e collegamenti. Gli attributi di sistema per i parametri valgono solo per il parametro che si sta progettando. Ai parametri di ingresso, di uscita e di ingresso/uscita possono essere assegnati attributi di sistema.

Regole

- Gli attributi di sistema per i parametri vengono assegnati nei blocchi convenzioni Parametri di ingresso, Parametri di uscita o Parametri di ingresso/uscita.
- Un identificatore deve essere lungo max. 24 caratteri.
- Attenersi alla seguente sintassi:



Esempio

```

VAR_INPUT
    in1      {S7_server:='alarm_archiv';
              S7_a_type:='ar_send'}: DWORD ;
END_VAR
  
```

La Guida agli attributi di sistema può essere richiamata nella documentazione online di S7-SCL selezionando il capitolo "Richiamo delle Guide di riferimento".

6.10 Parte istruzioni

Definizione

La parte istruzioni contiene istruzioni che devono essere eseguite dopo il richiamo di un blocco di codice. Queste istruzioni servono per definire dati o indirizzi.

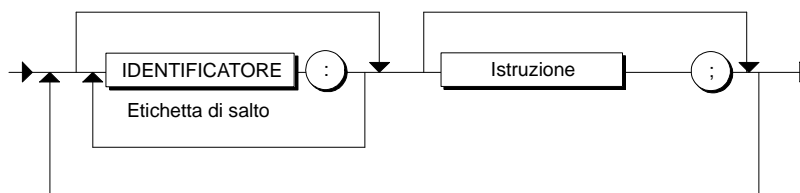
La parte istruzioni di un blocco dati contiene istruzioni per la preimpostazione delle proprie variabili.

Regole

- La parte istruzioni inizia con la parola chiave BEGIN. Nei blocchi dati questa parola chiave deve essere sempre specificata. La parte istruzioni termina con la parola chiave di fine blocco.
- Tutte le istruzioni terminano con un punto e virgola.
- Tutti gli identificatori utilizzati nella parte istruzioni devono essere stati definiti in precedenza.
- Ogni istruzione può essere preceduta in opzione da etichette di salto.

Attenersi alla seguente sintassi:

Parte istruzioni



Esempio

```
BEGIN
    VALORE INIZIALE      := 0;
    VALORE FINALE := 200;
.
.
SALVARE:  RISULTATO      := VALORE NOMINALE;
.
.
END_FUNCTION_BLOCK
```

6.11 Istruzioni

Definizione

Un'istruzione è la più piccola unità indipendente di un programma utente. Essa rappresenta una norma di lavoro per il processore.

In S7-SCL vengono utilizzati i seguenti tipi di istruzioni:

- Assegnazioni di valori che servono per assegnare ad una variabile il risultato di un'espressione o il valore di un'altra variabile.
- Istruzioni di controllo che servono per ripetere istruzioni o gruppi di istruzioni oppure per realizzare diramazioni in un programma.
- Elaborazione di sottoprogrammi che serve per richiamare funzioni e blocchi funzionali.

Regole

Attenersi alla seguente sintassi:



Esempio

Gli esempi seguenti hanno lo scopo di illustrare le diverse varianti delle istruzioni:

```
// Esempio di assegnazione di valori
VALPREMISURA:= 0 ;

// Esempio di elaborazione di sottoprogramma
FB1.DB11 (TRASFERIMENTO:= 10) ;

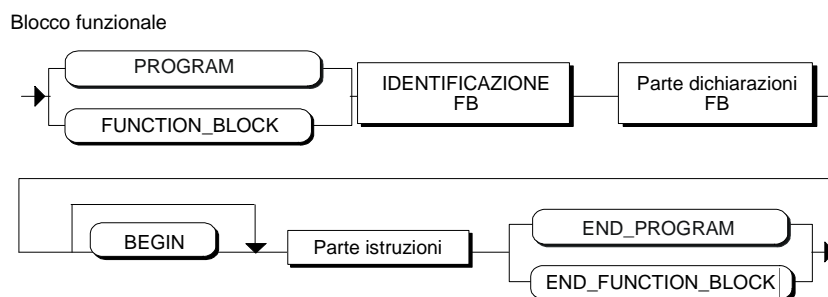
// Esempio di istruzione di controllo
WHILE CONTATORE < 10 DO..
.
.
END_WHILE;
```

6.12 Struttura di un blocco funzionale (FB)

Definizione

Un blocco funzionale (FB) è un blocco di codice contenente una parte di un programma e che dispone di un'area di memoria assegnata. Ogni volta che viene richiamato un FB vi si deve assegnare un DB di istanza. L'utente determina la struttura di questo DB di istanza mediante definizione della parte convenzioni dell'FB.

Sintassi



Nome dell'FB

Dopo la parola chiave FUNCTION_BLOCK o PROGRAM indicare come nome dell'FB la parola chiave FB e quindi il numero o il nome simbolico del blocco. Il numero del blocco può avere un valore compreso fra 0 e 65533.

Esempi:

```
FUNCTION_BLOCK FB10  
FUNCTION_BLOCK MOTORE1
```

Parte convenzioni dell'FB

La parte convenzioni dell'FB serve per definire i dati specifici del blocco. Per i blocchi convenzioni consentiti si rimanda al capitolo "Parte convenzioni". Si deve tener presente che la parte convenzioni determina anche la struttura del DB d'istanza assegnato.

Esempio

Il seguente esempio illustra il codice sorgente per un blocco funzionale. In questo caso i parametri di ingresso e uscita (qui V1, V2) sono stati impostati con valori iniziali.

```
FUNCTION_BLOCK FB11
VAR_INPUT
    V1 : INT := 7 ;
END_VAR
VAR_OUTPUT
    V2 : REAL ;
END_VAR
VAR
    FX1, FX2, FY1, FY2 : REAL ;

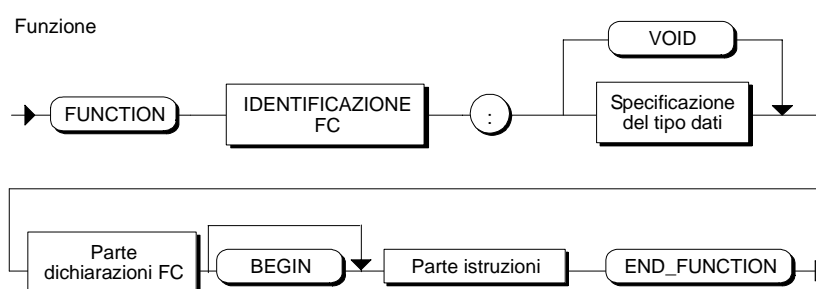
END_VAR
BEGIN
    IF V1 = 7 THEN
        FX1 := 1.5 ;
        FX2 := 2.3 ;
        FY1 := 3.1 ;
        FY2 := 5.4 ;
        //Richiamo della funzione FC11 con immissione dei
        //parametri mediante le variabili statiche.
        V2 := FC11 (X1:= FX1, X2 := FX2, Y1 := FY1, Y2 :=
FY2) ;
    END_IF ;
END_FUNCTION_BLOCK
```

6.13 Struttura di una funzione (FC)

Definizione

Una funzione (FC) è un blocco di codice al quale non è assegnata alcuna area di memoria. Essa non necessita di alcun DB di istanza. Contrariamente ad un FB, tale funzione può restituire il risultato di una funzione (valore di ritorno) al punto di richiamo. Perciò, tale funzione può essere utilizzata come una variabile in un'espressione. Le funzioni del tipo VOID non hanno alcun valore di ritorno.

Sintassi



Nome della FC

Dopo la parola chiave "FUNCTION" indicare come nome dell'FC la parola chiave FC e quindi il numero o il nome simbolico della funzione. Il numero del blocco può essere compreso fra 0 e 65533.

Esempio:

```

FUNCTION FC17 : REAL
FUNCTION FC17 : VOID

```

Specificazione dei tipi di dati

La specificazione del tipo di dati determina il tipo di dati del valore di ritorno. Sono consentiti tutti i tipi di dati ad eccezione di STRUCT e ARRAY. L'indicazione del tipo di dati non è necessaria se con una funzione tipo VOID si rinuncia al valore di ritorno.

Parte convenzioni della FC

La parte convenzioni della FC consente di dichiarare i dati locali (variabile temporale, parametri di ingresso, di uscita e di ingresso/uscita, costanti, etichette di salto).

Parte istruzioni della FC

Nella parte istruzioni al nome della funzione deve essere assegnato il risultato della funzione. Le funzioni del tipo VOID non hanno questa assegnazione. Segue un esempio di istruzione valida nell'ambito di una funzione con la denominazione FC31, ad es.:

```
FC31 := VALORE;
```


Esempio

```
FUNCTION FC11: REAL
VAR_INPUT
    x1: REAL ;
    x2: REAL ;
    x3: REAL ;
    x4: REAL ;
END_VAR
VAR_OUTPUT
    Q2: REAL ;
END_VAR
BEGIN
    // Ritorno del valore della funzione
    FC11:= SQRT( (x2 - x1)**2 + (x4 - x3) **2 ) ;
    Q2:= x1 ;
END_FUNCTION
```

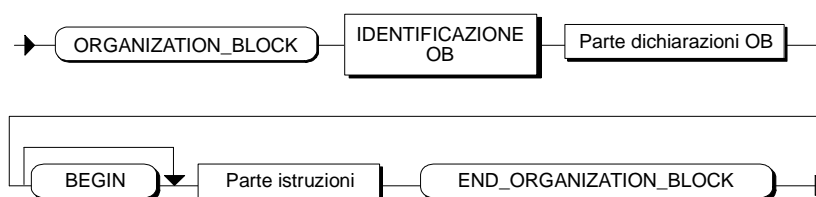
6.14 Struttura di un blocco organizzativo (OB)

Definizione

Come gli FB e le FC, il blocco organizzativo è una parte del programma utente e viene richiamato dal sistema operativo a intervalli ciclici o quando si verificano determinati eventi. Esso costituisce l'interfaccia fra programma utente e sistema operativo.

Sintassi

Blocco organizzativo



Nome dell'OB

Dopo la parola chiave "ORGANIZATION_BLOCK" indicare come nome dell'OB la parola chiave OB e quindi il numero o il nome simbolico del blocco. Il numero del blocco può essere compreso fra 1 e 65533.

Esempi:

```
ORGANIZATION_BLOCK OB1
ORGANIZATION_BLOCK ALLARME
```

Parte convenzioni dell'OB

La parte convenzioni dell'OB consente di dichiarare i dati locali (variabili temporanee, costanti, etichette di salto).

In linea di massima, ogni OB necessita per l'esecuzione di 20 byte di dati locali per il sistema operativo (BESY). Si deve perciò definire un campo con un identificatore a piacere. Se si inserisce il modello di blocco per gli OB, questa convenzione è già inclusa.

Esempio

```
ORGANIZATION_BLOCK OB1
VAR_TEMP
    HEADER : ARRAY [1..20] OF BYTE ; //20 byte per sistema
operativo
END_VAR
BEGIN
    FB17.DB10 (V1 := 7) ;
END_ORGANIZATION_BLOCK
```

6.15 Struttura di un blocco dati (DB)

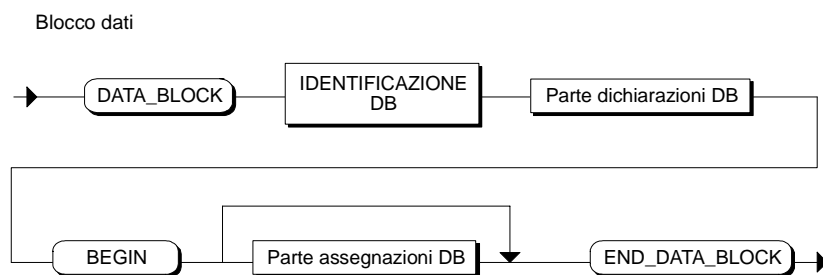
Definizione

I dati globali specifici dell'utente che devono poter accedere a tutti i blocchi di un programma, vengono depositati in blocchi dati. Ogni FB, FC o OB può leggere o sovrascrivere questi blocchi dati.

Esistono due tipi di blocchi dati:

- **Blocchi dati**
Blocchi dati che possono accedere a tutti i blocchi dati del programma CPU. Ogni FB, FC o OB è in grado di leggere o scrivere i dati contenuti in questi blocchi dati.
- **Blocchi dati che sono assegnati ad un FB (DB di istanza)**
I blocchi dati di istanza sono blocchi dati assegnati ad un determinato blocco funzionale (FB). Essi contengono i dati locali per questo blocco funzionale assegnato. Questi blocchi dati vengono generati automaticamente dal compilatore S7-SCL non appena l'FB viene richiamato nel programma utente.

Sintassi



Nome del DB

Dopo la parola chiave "DATA_BLOCK" indicare come nome del DB la parola chiave DB e quindi il numero o il nome simbolico del blocco. Il numero del blocco può essere compreso fra 1 e 65533.

Esempi:

```
DATA_BLOCK DB20
DATA_BLOCK CAMPO_MISURA
```

Parte convenzioni del DB

Nella parte convenzioni del DB viene definita la struttura dati del DB. Si può procedere nei due modi descritti di seguito.

- **Tramite assegnazione di un tipo di dati definito dall'utente**

Qui si può indicare l'identificazione di un tipo di dati definito dall'utente. Il blocco dati rileva quindi la struttura di questo UDT. I valori iniziali per le variabili possono essere assegnati nella parte assegnazioni del DB.

- **Tramite definizione di un tipo di dati STRUCT**

Nell'ambito delle specifiche del tipo di dati STRUCT vengono definiti il tipo di dati per ciascuna variabile da salvare nel DB e se necessario anche i valori iniziali.

Parte dichiarazioni DB



Esempio:

```

DATA_BLOCK DB20
  STRUCT // Parte convenzioni
    VALORE:ARRAY [1..100] OF INT;
  END_STRUCT

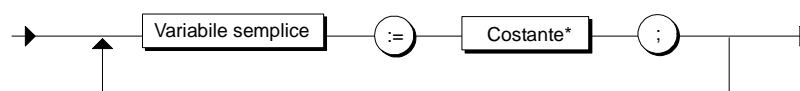
BEGIN // Inizio parte assegnazioni
:
END_DATA_BLOCK // Fine blocco dati
  
```

Parte assegnazioni DB

Nella parte assegnazioni, i dati che sono stati definiti nella parte convenzioni possono essere adattati per applicazioni specifiche con singoli valori specifici del DB.

La parte assegnazioni inizia con la parola chiave: BEGIN e si compone quindi di una sequenza di assegnazioni di valori.

Parte assegnazioni DB



* modalità di scrittura AWL

Quando si assegnano dei valori iniziali (inizializzazione), si indicano degli attributi e dei commenti, si deve utilizzare la sintassi di AWL. Le informazioni relative al modo di scrivere le costanti, gli attributi e i commenti possono essere trovate nella Guida online per AWL o nella documentazione di STEP 7.

Esempio

```
// Blocco dati con tipo di dati STRUCT
DATA_BLOCK DB10
    STRUCT // Convenzione di dati con configurazione predefinita
        VALORE :      ARRAY [1..100] OF INT := 100 (1) ;
        INTERRUETTORE:  BOOL      := TRUE ;
        S_WORD :      WORD      := W#16#FFAA ;
        S_BYTE :      BYTE      := B#16#FF ;
        S_TIME :      S5TIME     := S5T#1h30m10s ;
    END_STRUCT

BEGIN // Parte assegnazioni
    // Assegnazione valori per determinati elementi di campo
    VALORE [1] := 5;
    VALORE [5] := -1 ;

END_DATA_BLOCK

// Blocco dati con tipo di dati definito dall'utente
DATA_BLOCK DB11
    UDT 51
BEGIN
END_DATA_BLOCK
```

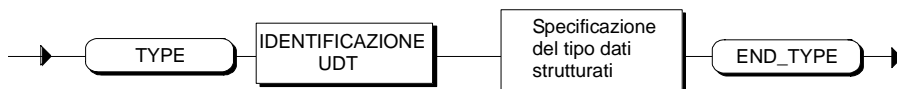
6.16 Struttura di un tipo di dati definito dall'utente

I tipi di dati definiti dall'utente (UDT) sono strutture speciali di dati generati dall'utente. Poiché i tipi di dati definiti dall'utente possiedono un nome, possono essere impiegati più volte. Dopo la loro definizione possono essere utilizzati nell'intero programma utente e sono quindi tipi di dati globali. Perciò questi tipi di dati possono essere utilizzati:

- nel blocco come tipi di dati semplici o tipi di dati composti oppure
- come modello per la creazione di blocchi dati con la stessa struttura.

Durante l'introduzione dei tipi di dati definiti dall'utente si deve tener presente che nella sorgente S7-SCL essi sono situati prima dei blocchi nei quali vengono utilizzati.

Tipo di dati definito dall'utente



Identificazione UDT

Dopo la parola chiave **TYPE** indicare la parola chiave **UDT** e quindi il numero o il nome simbolico dell'UDT. Il numero del blocco può essere compreso fra 0 e 65533.

Esempi:

```
TYPE UDT10
TYPE BLOCCO_DI_ASSEGNAZIONE
```

Specificazione dei tipi di dati

La specificazione del tipo di dati avviene sempre con l'ausilio di una specificazione di **tipi di dati STRUCT**. Il tipo di dati UDT può essere utilizzato nei blocchi convenzioni dei blocchi di codice oppure assegnato ai DB.

Esempio di definizione di UDT

```
TYPE VALORIMISURA
STRUCT
// Definizione di UDT con nome simbolico
    BIPOL_1 : INT := 5;
    BIPOL_2 : WORD := W#16#FFAA ;
    BIPOL_3 : BYTE := B#16#F1 ;
    BIPOL_4 : WORD := B#(25,25) ;
    MISURA : STRUCT
        BIPOLAR_10V : REAL ;
        UNIPOLAR_4_20MA : REAL ;
    END_STRUCT ;
END_STRUCT ;
END_TYPE

//Utilizzo dell'UDT in un FB
FUNCTION_BLOCK FB10
VAR
    CAMPO_MISURA : VALOREMISURA;
END_VAR
BEGIN
    // . . .
    CAMPO_MISURA.BIPOL_1 := -4 ;
    CAMPO_MISURA.MISURA.UNIPOLARE_4_20MA := 2.7 ;
    // . . .
END_FUNCTION_BLOCK
```

6.17 Opzioni di compilazione nelle sorgenti S7-SCL

Definizione

Oltre ai blocchi le sorgenti S7-SCL possono contenere anche indicazioni relative alle impostazioni di compilazione con cui i singoli blocchi devono essere compilati.

Le opzioni di compilazione controllano l'esecuzione della compilazione dei singoli blocchi o dell'intera sorgente indipendentemente dalle impostazioni nella scheda "Compilatore (impostazioni)".

Le opzioni di compilazione possono essere utilizzate nelle sorgenti S7-SCL o nei file di compilazione.

Validità

Le opzioni sono valide solo per la compilazione della sorgente per cui sono state definite. La validità di un'opzione di compilazione ha inizio con la relativa denominazione e termina alla fine della sorgente o del file di compilazione. Per diverse opzioni di compilazione identiche vale l'ultima in ordine di tempo.

Se sono state definite opzioni di compilazione per un blocco, queste sono prioritarie rispetto alle impostazioni della scheda "Compilatore (impostazioni)". Le impostazioni nella scheda restano tuttavia globalmente valide.

Regole

Per le opzioni di compilazione valgono le seguenti regole:

- Le opzioni si trovano nella sorgente al di fuori dei limiti del blocco;
- Le opzioni si trovano in una riga propria;
- Non si fa distinzione tra maiuscolo e minuscolo.

Opzioni disponibili

La tabella mostra le opzioni disponibili:

Opzione	Valore	Significato
[Scl_]ResetOptions	Non è possibile indicare alcun valore	Creazione della preimpostazione di compilazione (impostazioni dalla finestra di dialogo)
[Scl_]OverwriteBlocks	'y[es]' oppure 'n[o]'	Sovrascrittura dei blocchi
[Scl_]GenerateReferenceData	'y[es]' oppure 'n[o]'	Creazione dei dati di riferimento
[Scl_]S7ServerActive	'y[es]' oppure 'n[o]'	Considerazione dell'attributo di sistema "S7_server"
[Scl_]CreateObjectCode	'y[es]' oppure 'n[o]'	Creazione del codice dell'oggetto
[Scl_]OptimizeObjectCode	'y[es]' oppure 'n[o]'	Ottimizzazione del codice dell'oggetto
[Scl_]MonitorArrayLimits	'y[es]' oppure 'n[o]'	Controllo dei limiti array
[Scl_]CreateDebugInfo	'y[es]' oppure 'n[o]'	Creazione informazioni debug
[Scl_]SetOKFlag	'y[es]' oppure 'n[o]'	Impostazione del flag OK
[Scl_]SetMaximumStringLength	'Da 1 a 254'	Lunghezza massima della stringa

Esempio

```
{SCL OverwriteBlocks := 'y' ; SCL_CreateDebugInfo := 'y'}
{SetOKFlag := 'y' ; OptimizeObjectCode := 'y'}
```


7 Tipi di dati

7.1 Sommario dei tipi di dati in S7-SCL

Un tipo di dati è la riunione di campi di valori e operazioni per formare delle unità.

I tipi di dati determinano:

- il tipo e il significato degli elementi di dati
- i campi consentiti degli elementi di dati
- la quantità consentita delle operazioni che possono essere eseguite con un operando di un tipo di dati
- la modalità di scrittura delle costanti del tipo di dati in oggetto.

Tipi di dati semplici

I tipi di dati semplici definiscono la struttura dei dati che non possono essere suddivisi in unità inferiori. Essi sono conformi alla definizione della norma DIN EN1131-3. Un tipo di dati semplice descrive un'area di memoria di lunghezza fissa e sta per bit, numero intero, numero reale, durata, ora e dimensione dei caratteri. I seguenti tipi di dati sono stati predefiniti in S7-SCL.

Gruppo	Tipi di dati	Significato
Tipi di dati a bit	BOOL BYTE WORD DWORD	I dati di questo tipo occupano 1 bit, 8 bit, 16 bit o 32 bit
Tipi a caratteri	CHAR	I dati di questo tipo occupano esattamente 1 carattere del set di caratteri ASCII
Tipi di dati numerici	INT DINT REAL	I dati di questo tipo sono disponibili per l'elaborazione di valori numerici.
Tipo di dati di temporizzazione	TIME DATE TIME_OF_DAY S5TIME	I dati di questo tipo rappresentano svariati valori di tempo/data nell'ambito di STEP 7.

Tipi di dati composti

S7-SCL supporta i seguenti tipi di dati composti:

Tipo di dati	Significato
DATE AND TIME DT	Definisce un campo di 64 bit (8 byte). Questo tipo di dati memorizza la data e l'ora (in formato decimale in codice binario) ed è già predefinito in S7-SCL.
STRING	Definisce un campo per una sequenza di max. 254 caratteri (tipo di dati CHAR).
ARRAY	Definisce un campo con elementi di un tipo di dati (semplici o composti).
STRUCT	Definisce un raggruppamento di tipi di dati con qualunque combinazione. Si possono definire campi di strutture oppure strutture di strutture e campi.

Tipi di dati definiti dall'utente

Il programmatore può creare un UDT (tipo di dati definito dall'utente) utilizzando la dichiarazione dei tipi di dati. Gli UDT possiedono un nome proprio e possono essere utilizzati più volte. In tal modo, un tipo di dati definito dall'utente può essere utilizzato per generare diversi blocchi dati con struttura identica.

Tipi di parametri

I tipi di parametri sono tipi di dati speciali per temporizzatori, contatori e blocchi che possono essere utilizzati come parametri formali.

Tipo di dati	Significato
TIMER	Serve per definire come parametro le funzioni temporali
COUNTER	Serve per definire come parametro le funzioni di conteggio
BLOCK_xx	Serve per definire come parametro FC, FB, DB e SDB
ANY	Serve per poter utilizzare come parametro un operando con qualsiasi tipo di dati
POINTER	Serve per poter utilizzare come parametro un'area di memoria

Tipo di dati ANY

In S7-SCL è possibile utilizzare le variabili con tipo di dati ANY come parametro formale di un blocco. È inoltre possibile creare variabili temporanee di questo tipo e utilizzarle nelle assegnazioni di valori.

7.2 Tipi di dati semplici

7.2.1 Tipi di dati a bit

I dati di questo tipo sono combinazioni di bit che possono occupare 1 bit (tipi di dati BOOL), 8 bit, 16 bit o 32 bit. Per i tipi di dati byte, parola e doppia parola non è possibile introdurre un campo di valori numerico. Si tratta di combinazioni di bit con cui è possibile creare solo espressioni booleane.

Tipo	Parola chiave	Numero di bit	Allineamento	Campo di valori
Bit	BOOL	1 bit	inizia dal bit meno significativo del byte	0, 1 oppure FALSE, TRUE
Byte	BYTE	8 bit	inizia dal bit meno significativo del byte	-
Parola	WORD	16 bit	inizia a parola	-
Doppia parola	DWORD	32	inizia a parola	-

7.2.2 Tipi di dati a caratteri

I dati di questo tipo occupano esattamente 1 carattere del set di caratteri ASCII.

Tipo	Parola chiave	Numero di bit	Campo di valori
Carattere singolo	CHAR	8	Set di caratteri ASCII ampliato

7.2.3 Tipi di dati numerici

Questo tipo di dati serve per l'elaborazione di valori numerici (p. es. per calcolare espressioni aritmetiche).

Tipo	Parola chiave	Numero di bit	Allineamento	Campo di valori
Integer (Numero intero)	INT	16	inizia a parola	-32_768 fino a 32_767
Intero doppio	DINT	32	inizia a parola	-2_147_483_648 fino a 2_147_483_647
Numero in virgola mobile (numero in virgola mobile IEEE)	REAL	32	inizia a parola	-3.402822E+38 fino a -1.175495E-38 +/- 0 1.175495E-38 fino a 3.402822E+38

7.2.4 Tipi di temporizzatori

I dati di questo tipo rappresentano i vari valori di ora/e data nell'ambito di STEP 7 (p. es. per impostare la data e introdurre i valori di tempo).

Tipo	Parola chiave	Numero di bit	Allineamento	Campo di valori
Tempo S5	S5TIME S5T	16	inizia a parola	T#0H_0M_0S_10MS fino a T#2H_46M_30S_0MS
Durata: tempo IEC a intervalli di 1 ms.	TIME T	32	inizia a parola	-T#24D_20H_31M_23S_647MS fino a T#24D_20H_31M_23S_647MS
Data: data IEC a intervalli di 1 giorno.	DATE D	16	inizia a parola	D#1990-01-01 fino a D#2168-12-31
Ora del giorno: ora a intervalli di 1 ms.	TIME_OF_DAY TOD	32	inizia a parola	TOD#0:0:0.0 fino a TOD#23:59:59.999

Per le variabili del tipo di dati S5TIME la risoluzione è limitata, ciò significa che sono disponibili solo le basi dei tempi 0,01s, 0,1s, 1s, 10s. Il compilatore arrotonda i valori in modo opportuno. Quando il valore impostato è maggiore di quanto non consenta il campo di valori, viene allora utilizzato il valore limite superiore.

7.3 Tipi di dati composti

7.3.1 Tipo di dati DATE_AND_TIME

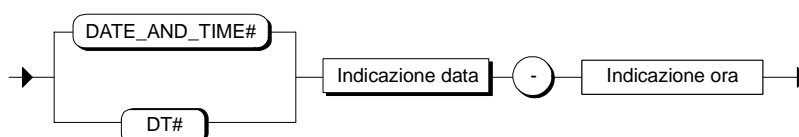
Definizione

Questo tipo di dati definisce un'area di 64 bit (8 byte) per l'indicazione della data e dell'ora. Nell'area di dati vengono memorizzate le seguenti informazioni:

anno, mese, giorno, ore, minuti, secondi, millisecondi.

Sintassi

DATE_AND_TIME



La sintassi precisa per l'indicazione della data e dell'ora è descritta nel paragrafo "Convenzione per le costanti".

Campo di valori

Tipo	Parola chiave	Numero di bit	Allineamento	Campo di valori
Data e ora	DATE_AND_TIME DT	64	Inizia e termina a parola	DT#1990-01-01-0:0:0.0 fino a DT#2089-12-31-23:59:59.999

Il tipo di dati Date_And_Time viene salvato in formato BCD:

Byte	Contenuto	Area
0	Anno	1990 ... 2089
1	Mese	01 ... 12
2	Giorno	1 ... 31
3	Ore	0 ... 23
4	Minuti	0 ... 59
5	Secondi	0 ... 59
6	2 MSD (most significant decade) di ms	00 ... 99
7 (4 MSB)	LSD (least significant decade) di ms	0 ... 9
7 (4 LSB)	Giorno della settimana	1 ... 7 (1 = domenica)

Esempio

Una definizione valida per il 20.10.1995 alle ore 12.20 min. 30 sec. e 10 millisecondi è la seguente:

```
DATE_AND_TIME#1995-10-20-12:20:30.10  
DT#1995-10-20-12:20:30.10
```

Nota

Per accedere direttamente ai componenti DATE o TIME sono disponibili funzioni standard (nella biblioteca STEP 7).

7.3.2 Tipo di dati STRING

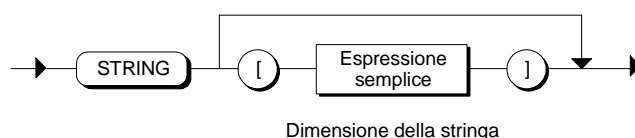
Definizione

Un tipo di dati STRING definisce una stringa di max. 254 caratteri. L'area standard riservata per una stringa di caratteri si compone di 256 byte. Quest'area di memoria è necessaria per memorizzare 254 caratteri e un'intestazione di 2 byte.

Si può ridurre lo spazio necessario per una stringa di caratteri definendo il numero max. di caratteri che devono essere memorizzati nella stringa. Una stringa zero, ossia una stringa senza contenuto, rappresenta il valore minimo possibile.

Sintassi

Specificazione del tipo dati STRING



L'espressione semplice indica il numero max. di caratteri di STRING. In una stringa di caratteri sono consentiti tutti i caratteri del codice ASCII. Una stringa può inoltre contenere caratteri speciali, ad esempio caratteri di controllo e caratteri non stampabili. Per specificarli utilizzare la sintassi \$hh dove hh sostituisce il valore del carattere ASCII espresso in esadecimale (esempio: '\$0D\$0Atesto')

Quando si dichiara lo spazio di memoria riservato alle stringhe si può specificare il numero massimo di caratteri memorizzabili nella stringa. Se non si specifica alcun valore per la lunghezza massima viene creata una stringa di 254 caratteri.

Esempio:

```

VAR
    Testot1    : String [123];
    Testo2     : String;
END_VAR

```

Nella dichiarazione della variabile "Testo1" la costante "123" indica il numero massimo di caratteri della stringa. Nella variabile "Testo2" viene riservata una lunghezza pari a 254 caratteri.

Nota

Per i parametri di uscita e di ingresso/uscita nonché per i valori di ritorno delle funzioni l'area riservata standard di 254 caratteri può essere ridotta per poter meglio usufruire delle risorse della CPU. Selezionare allora il comando di menu **Strumenti > Impostazioni** e, nella finestra di dialogo successiva, la scheda "Compilatore".

Indicare quindi nell'opzione "Lunghezza max. stringa" il numero di caratteri desiderato. Questa impostazione si riferisce a tutte le variabili STRING della sorgente. Il valore impostato non deve essere quindi inferiore alle variabili STRING utilizzate nel programma.

Inizializzazione delle stringhe di caratteri

Le variabili di tipo stringa, come del resto le altre variabili, possono essere inizializzate con sequenze di caratteri costanti durante la dichiarazione dei parametri dei blocchi funzionali (FB). Nei parametri delle funzioni (FC) l'inizializzazione non è possibile.

Se la stringa di caratteri preimpostata è più breve di quella dichiarata le posizioni in eccesso restano vuote. Durante l'elaborazione delle variabili vengono prese in considerazione solo le posizioni effettivamente occupate da caratteri.

Esempio:

```
x : STRING[7]:='Indirizzo';
```

Se si usano variabili di tipo String, ad esempio per memorizzare temporaneamente dei risultati, prima di utilizzarle è necessario descriverle con una costante String nella dichiarazione delle variabili o in una successiva assegnazione di valori con un valore di default.

Nota

Se la funzione prelevata da una biblioteca standard fornisce un valore di ritorno con tipo di dati STRING al quale si deve assegnare una variabile temporanea, prima di procedere è necessario inizializzare tale variabile.

Esempio:

```
FUNCTION Test : STRING[45]
VAR_TEMP
  x : STRING[45];
END_VAR
x := 'a';
x := concat (in1 := x, in2 := x);
Test := x;
END_FUNCTION
```

Senza l'inizializzazione **x := 'a'**; la funzione fornirebbe un risultato errato.

Allineamento

Le variabili di tipo STRING iniziano e terminano a parola.

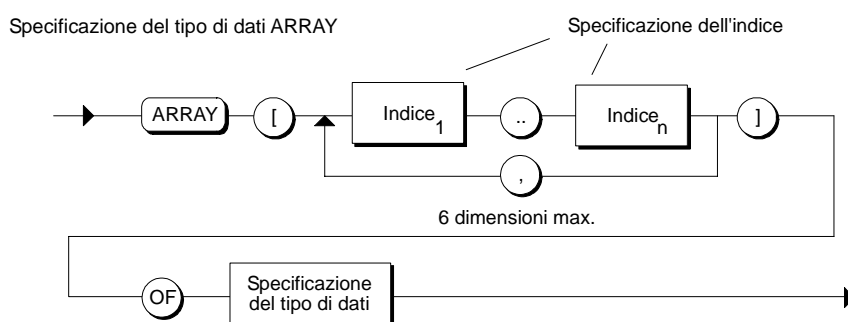
7.3.3 Tipo di dati ARRAY

Definizione

Gli ARRAY hanno un numero fisso di componenti di un solo tipo di dati. S7-SCL distingue tra:

- il tipo ARRAY a una dimensione. Si tratta di una lista di elementi di dati ordinati in ordine crescente,
- il tipo ARRAY a due dimensioni. Si tratta di una tabella di dati composta di righe e colonne. La prima dimensione si riferisce al numero di riga e la seconda al numero di colonna,
- il tipo di ARRAY di dimensione superiore. Si tratta di un'estensione del tipo di ARRAY a due dimensioni che viene ampliato con ulteriori dimensioni. Il numero max. di dimensioni consentite è 6.

Sintassi



Specificazione dell'indice

Descrive le dimensioni del tipo di dati ARRAY come indicato di seguito.

- Con l'indice minimo e massimo possibile (area indice) per ogni dimensione. L'indice può essere un numero intero qualsiasi (da -32768 fino a 32767).
- I limiti devono essere separati da due punti. Le singole aree di indice devono essere separate mediante virgole.
- Le specificazioni di indice complete vengono racchiuse fra parentesi quadre.

Specificazione dei tipi di dati

Con la specificazione dei tipi di dati si dichiara il tipo di dati dei componenti. Sono consentiti tutti i tipi di dati per la specificazione. Il tipo di dati di un Array può a sua volta essere un tipo ARRAY o un tipo STRUCT; in altre parole: i tipi di parametri non possono essere utilizzati come tipo di elemento per un campo.

Esempio

```
VAR
    REGOLATORE1 :
        ARRAY[1..3,1..4] OF INT:=  -54,  736,  -83,  77,
        -1289,      10362,      385,  2,
        60,  -37,  -7,  103 ;
    REGOLATORE2 : ARRAY[1..10] OF REAL ;
END_VAR
```

Allineamento

Le variabili di tipo ARRAY vengono create riga per riga. Le dimensioni di una variabile di tipo BOOL, BYTE o CHAR terminano a byte, le altre a parola.

7.3.4 Tipo di dati STRUCT

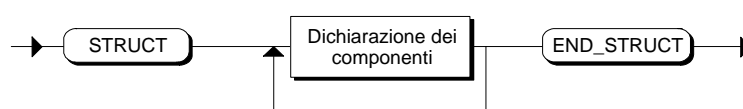
Definizione

Il tipo di dati STRUCT descrive un'area composta da un numero fisso di componenti, i quali possono essere di qualsiasi tipo. Questi elementi di dati vengono indicati subito dopo la parola chiave STRUCT nella dichiarazione dei componenti.

In particolare, un elemento di dati del tipo STRUCT può essere a sua volta del tipo composto. In altre parole: è consentito l'annidamento del tipo di dati STRUCT.

Sintassi

Specificazione del tipo di dati STRUCT

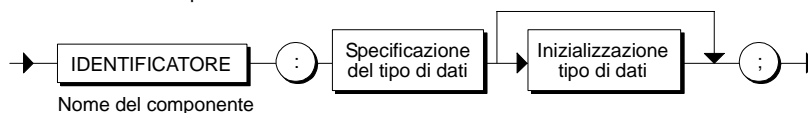


Dichiarazione componenti

La dichiarazione dei componenti è un elenco dei vari componenti del tipo di dati STRUCT. Essa si compone di:

- da 1 fino a n identificatori con il corrispondente tipo di dati e
- di un'impostazione opzionale con valori iniziali.

Dichiarazione dei componenti



Nome del componente

Nomi dei componenti nell'ambito di strutture

L'identificatore è il nome di un elemento di struttura per la seguente specificazione di dati.

I dati possono essere specificati con qualsiasi tipo, ad eccezione dei tipi di parametri.

Come opzione, un singolo elemento della struttura può essere impostato con un valore iniziale conformemente alla specificazione dei dati. Questa assegnazione avviene tramite un'assegnazione di valori.

Esempio

```
VAR
    MOTORE : STRUCT
        DATI : STRUCT
            CORRENTE_CARICO : REAL ;
            TENSIONE       : INT  := 5 ;
        END_STRUCT ;
    END_STRUCT ;
END_VAR
```

Allineamento

Le variabili di tipo STRUCT iniziano e terminano a parola.

Attenzione

In caso si definisca una struttura che non termina a parola, S7-SCL procede al completamento automatico dei byte mancanti e adatta in tal modo le dimensioni della struttura.

Il processo di adattamento delle dimensioni della struttura può portare a conflitti nella fase di accesso ai tipi di dati con lunghezza di byte dispari.

7.4 Tipi di dati definiti dall'utente

7.4.1 Tipo di dati definito dall'utente (UDT)

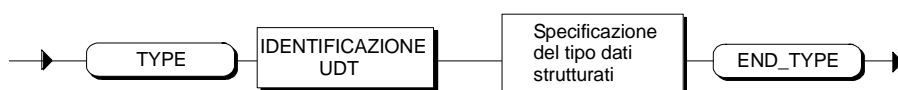
Definizione

Gli UDT vengono definiti come blocchi. Un tale tipo di dati, in base alla sua definizione può essere utilizzato nell'intero programma utente e è quindi un tipo di dati globale. Questi tipi di dati possono essere utilizzati con la loro identificazione UDT UDTx (x indica il numero) oppure con un nome simbolico assegnato nella parte dichiarazioni di un blocco o di un blocco dati.

Il tipo di dati definito può essere impiegato per definire variabili, parametri, blocchi dati e altri tipi di dati definiti dall'utente e consente inoltre di impostare i componenti di campi o strutture.

Sintassi

Tipo di dati definito dall'utente



Identificazione degli UDT

Una convenzione di un tipo di dati definito dall'utente viene indicata dalla parola chiave TYPE, seguita dal nome del tipo di dati definito dall'utente (identificatore UDT). Il nome del tipo di dati definito dall'utente può essere assoluto, cioè indicato con un nome standard in forma UDTx (x è in numero), oppure può essere indicato con un nome simbolico.

Esempi:

```
TYPE UDT10
TYPE VALORIMISURA
```

Specificazione dei tipi di dati

Dopo l'identificazione degli UDT segue la specificazione del tipo di dati. Qui è consentita solo la specificazione del tipo di dati STRUCT:

```
STRUCT
:
END_STRUCT
```

Nota

All'interno di un tipo di dati definito dall'utente viene applicata la sintassi di AWL. Ciò riguarda, ad esempio, la modalità di scrittura di costanti e l'assegnazione di valori iniziali (inizializzazione). Informazioni sul modo di scrittura delle costanti possono essere desunte dalla Guida online ad AWL.

Esempio

```
// Definizione di UDT con nome simbolico
TYPE
VALORIMISURA:    STRUCT
                    BIPOL_1 : INT := 5;
                    BIPOL_2 : WORD := W#16#FFAA ;
                    BIPOL_3 : BYTE := B#16#F1 ;
                    BIPOL_4 : WORD := W#16#1919 ;
                    MISURA : STRUCT
                        BIPOLARE_10V : REAL ;
                        UNIPOLARE_4_20MA : REAL ;
                    END_STRUCT;
                END_STRUCT;

END_TYPE

//Utilizzo dell'UDT in un FB
FUNCTION_BLOCK FB10
VAR
    CAMPO_MISURA : VALORIMISURA;
END_VAR
BEGIN
    // . . .
    CAMPO_MISURA.BIPOL_1 := -4 ;
    CAMPO_MISURA.MISURA.UNIPOLAR_4_20MA := 2.7 ;
    // . . .
END_FUNCTION_BLOCK
```


7.5 Tipi di dati per parametri

Per la definizione dei parametri di blocchi formali per FB e FC, oltre ai tipi di dati già descritti, si possono impiegare i cosiddetti tipi di parametro.

Parametri	Dimensione	Descrizione
TIMER	2 byte	Contrassegna un determinato temporizzatore che viene utilizzato dal programma nel blocco di codice richiamato. Parametro attuale: ad es. T1
COUNTER	2 byte	Contrassegna un determinato contatore che viene utilizzato dal programma nel blocco codice richiamato. Parametro attuale: ad es. Z10
BLOCK_FB BLOCK_FC BLOCK_DB BLOCK_SDB	2 byte	Contrassegna un determinato blocco che deve essere utilizzato da un blocco codice AWL richiamato. Parametro attuale: ad es. FC101, DB42 Note: È possibile accedere al tipo di dati BLOCK_DB (myDB.dw10) in modo assoluto. È parimenti possibile elaborarlo ulteriormente tramite BLOCK_DB_TO_WORD(). I tipi di dati BLOCK_SDB, BLOCK_FB e BLOCK_FC non possono essere valutati dai programmi S7-SCL. È possibile utilizzarli unicamente per assegnare parametri di questo tipo richiamando i blocchi AWL.
ANY	10 byte	Viene impiegato nei casi in cui come tipo di dati del parametro attuale è consentito un tipo di dati qualsiasi.
POINTER	6 byte	Contrassegna una determinata area di memoria che deve essere impiegata dal programma. Parametro attuale: ad es. M50.0

7.5.1 Tipi di dati TIMER e COUNTER

Viene definito un determinato temporizzatore (TIMER) o contatore (COUNTER) che si desidera utilizzare per l'elaborazione di un blocco. I tipi di dati TIMER e COUNTER sono consentiti solo per parametri d'ingresso (VAR_INPUT).

7.5.2 Tipo di dati BLOCK

Viene definito un determinato blocco, che si desidera utilizzare come parametro d'ingresso. La dichiarazione del parametro d'ingresso determina il tipo di blocco (FB, FC, DB). Durante l'assegnazione dei parametri viene indicato l'identificatore del blocco. Sono consentiti sia l'indicatore assoluto che l'indicatore simbolico.

Al tipo di dati BLOCK_DB è possibile accedere in modo assoluto (`myDB.dw10`). Per gli altri tipi di dati del blocco S7-SCL non mette a disposizione alcuna operazione. Durante il richiamo di blocchi si possono assegnare possibili parametri di questo tipo. Nelle funzioni non è consentito il trasferimento di un parametro d'ingresso.

In S7-SCL si possono correlare ai seguenti tipi di dati gli operandi quali parametri attuali:

- Blocchi funzionali senza parametri formali.
- Funzioni senza parametri formali e valore di ritorno (funzioni VOID).
- Blocchi dati e blocchi dati di sistema.

7.5.3 Tipo di dati POINTER

È possibile assegnare delle variabili al tipo di dati POINTER dopo averle definite come parametri formali di un blocco. Richiamando il blocco si possono assegnare ai parametri degli operandi con qualsiasi tipo di dati (ad eccezione di ANY).

S7-SCL offre tuttavia solo un'istruzione per l'elaborazione del tipo di dati POINTER: il trasferimento a blocchi subordinati.

È possibile assegnare come parametri attuali i seguenti tipi di operandi:

- Indirizzi assoluti
- Nomi simbolici
- Operandi con tipo di dati POINTER:
Questo è solo possibile se l'operando è un parametro formale di tipo compatibile.
- Costante NIL:
Si specifica un puntatore a zero.

Limitazioni:

- Il tipo di dati POINTER è consentito per i parametri di ingresso, i parametri di ingresso/uscita di FB e FC e per i parametri di uscita di FC. Le costanti non sono consentite come parametri attuali o sulla destra di un'attribuzione di valori (ad eccezione della costante NIL).
- Se nel richiamare un FB o una FC a un parametro formale del tipo POINTER viene assegnata una variabile temporanea, non è possibile assegnare questo parametro a un altro blocco. Le variabili temporanee perdono la loro validità durante il passaggio.
- Richiamando un FC o un FB è possibile assegnare ingressi di processo (%PEW) solo a parametri formali di tipo Pointer, se il parametro formale è stato definito come parametro di ingresso.
- Richiamando un FB è possibile assegnare uscite di processo (%PAW) solo a parametri formali di tipo Pointer, se il parametro formale è stato definito come parametro di uscita.

Esempio

```
FUNCTION FC100 : VOID
VAR_IN_OUT
    N_out : INT;
    out   : POINTER;
END_VAR
VAR_TEMP
    ret   : INT;
END_VAR
BEGIN
    // ...
    ret := SFC79(N := N_out, SA := out);
    // ...
END_FUNCTION

FUNCTION_BLOCK FB100
VAR
    ii   : INT;
    aa : ARRAY[1..1000] OF REAL;
END_VAR

BEGIN
    // ...
    FC100( N_out := ii, out := aa);
    // ...
END_FUNCTION_BLOCK
```

7.6 Tipo di dati ANY

In S7-SCL è possibile definire variabili del tipo di dati ANY nel modo seguente.

- Come parametri formali di un blocco a cui, al richiamo del blocco, possono essere assegnati parametri attuali di un tipo di dati qualsiasi.
- Come variabili temporanee alle quali è possibile assegnare valori di un tipo di dati qualsiasi.

I seguenti dati possono essere utilizzati come parametri attuali oppure sulla destra di un'attribuzione di valori.

- Variabili locali e globali
- Variabili nel DB (indirizzamento assoluto o simbolico)
- Variabili nell'istanza locale (indirizzamento assoluto o simbolico)
- Costante NIL:
Si specifica un puntatore a zero.
- Tipo di dati ANY
- Temporizzatori, contatori e blocchi:
si deve indicare la relativa identificazione (p. es. T1, Z20 o FB6).

Limitazioni:

- Il tipo di dati ANY è consentito per i parametri di ingresso, i parametri di ingresso/uscita di FB e FC e per i parametri di uscita di FC. Le costanti non sono consentite come parametri attuali o sulla destra di un'attribuzione di valori (ad eccezione della costante NIL).
- Se nel richiamare un FB o una FC a un parametro formale del tipo ANY viene assegnata una variabile temporanea, non è possibile assegnare questo parametro a un altro blocco. Le variabili temporanee perdono la loro validità durante il passaggio.
- Variabili di questo tipo non possono essere utilizzate come tipo di componenti in una struttura o come tipo di elementi per un campo.
- Richiamando un FC o un FB è possibile assegnare ingressi di processo (%PEW) solo a parametri formali di tipo ANY, se il parametro formale è stato definito come parametro di ingresso.
- Richiamando un FB è possibile assegnare uscite di processo (%PAW) solo a parametri formali di tipo ANY, se il parametro formale è stato definito come parametro di uscita.

7.6.1 Esempio di un tipo di dati ANY

```
VAR_INPUT
    iANY : ANY;
END_VAR

VAR_TEMP
    pANY : ANY;
END_VAR

CASE ii OF
1:
    pANY := MW4;          // pANY contiene l'indirizzo di MW4

3..5:
    pANY:= aINT[ii];      // pANY contiene l'indirizzo dell'ii-esimo
                        // elemento del campo aINT;

100:
    pANY := iANY;         // pANY contiene il valore della
                        // variabile di ingresso iANY ELSE
    pANY := NIL;          // pANY contiene il valore
                        // del puntatore NIL
END_CASE;

SFCxxx(IN := pANY);
```


8 Dichiarazione di variabili locali e parametri

8.1 Variabili locali e parametri di blocco

Categorie di variabili

Le variabili locali possono essere suddivise nelle seguenti categorie conformemente alla tabella:

Variabili	Significato
Variabili statiche	Le variabili statiche sono variabili locali il cui valore rimane immutato ad ogni esecuzione del blocco (memoria del blocco). Esse hanno la funzione di memorizzare i valori di un blocco funzionale, e vengono depositate nel blocco dati di istanza.
Variabili temporanee	Le variabili temporanee fanno parte di un blocco di codice e non occupano alcuna area di memoria statica poiché esse vengono deposte nello stack della CPU. Il loro valore rimane costante solo durante l'esecuzione di un blocco. Al di fuori del blocco in cui sono state dichiarate le variabili non è possibile accedere alle variabili temporanee.

Categorie di parametri di blocco

I parametri di blocco sono segnaposti che vengono definiti solo al momento dell'impiego concreto (richiamo) del blocco. I segnaposti presenti nel blocco vengono denominati parametri formali, mentre i valori assegnati durante il richiamo del blocco vengono denominati parametri attuali. I parametri formali di un blocco possono essere considerati come variabili locali.

I parametri di blocco possono essere suddivisi nelle seguenti categorie.

Parametri del blocco	Significato
Parametri d'ingresso	I parametri d'ingresso assumono i valori d'ingresso attuali durante il richiamo del blocco. Essi possono essere solo letti.
Parametri d'uscita	I parametri d'uscita trasferiscono i valori d'uscita attuali al blocco richiamante. Essi possono essere letti e scritti.
Parametri di ingresso/uscita	Al richiamo di un blocco i parametri di ingresso/uscita assumono i valori di ingresso attuali, quindi, dopo l'elaborazione del valore, assumono il risultato e lo restituiscono al blocco richiamante.

Flag (Flag OK)

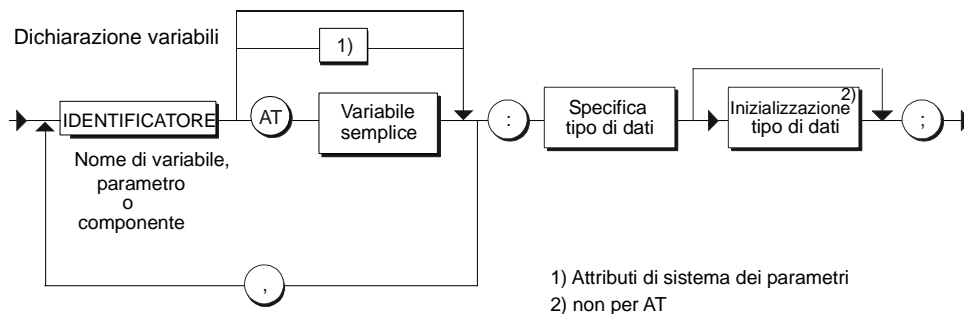
Il compilatore S7-SCL mette a disposizione un flag che consente di localizzare eventuali errori durante l'esecuzione dei programmi nella CPU. Si tratta di una variabile locale di tipo BOOL con il nome predefinito "OK".

8.2 Sintassi generale di una dichiarazione di variabili o parametri

Per poter essere utilizzati in un blocco di codice o un blocco dati, le variabili e i parametri di blocco devono essere dichiarati uno per uno. La dichiarazione stabilisce che un identificatore venga utilizzato come parametro di blocco o come variabile e gli assegna un tipo di dati.

Una dichiarazione di variabili o parametri si compone di un identificatore liberamente scelto e dell'indicazione del tipo di dati. Il diagramma sintattico illustra la forma generale.

Sintassi di una dichiarazione di variabili o parametri



Esempi

```
VALORE1      :      REAL;
se esistono diverse variabili dello stesso tipo:

VALORE2, VALORE3, VALORE4, ....: INT;
CAMPO        :      ARRAY[1..100, 1..10] OF REAL;
RECORD       :      STRUCT
                                CAMPOMISURA:ARRAY[1..20] OF REAL;
                                INTERRUETTORE:BOOL;
END_STRUCT
```

Nota

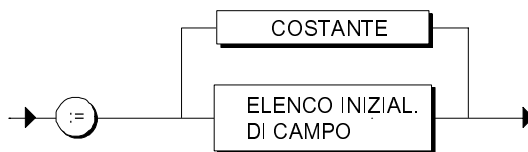
Per poter utilizzare le parole riservate con funzione di identificatore, le si deve far precedere dal carattere "#" (ad es. #FOR).

8.3 Inizializzazione

Nella dichiarazione, le variabili statiche, i parametri d'ingresso di un FB e i parametri di uscita di un FB possono essere impostati con un determinato valore. Anche i parametri di ingresso/uscita possono essere predefiniti, ma solo se sono di tipo di dati elementare. Nelle variabili semplici, questa impostazione avviene mediante assegnazione ($:=$) di una costante dopo l'indicazione del tipo di dati.

Sintassi

INIZIALIZZAZIONE



Esempio

```
VALORE      :REAL := 20.25;
```

Nota

L'inizializzazione di una lista di variabili (A1, A2, A3,...: INT:=...) non è possibile. In tal caso le variabili devono essere inizializzate singolarmente.

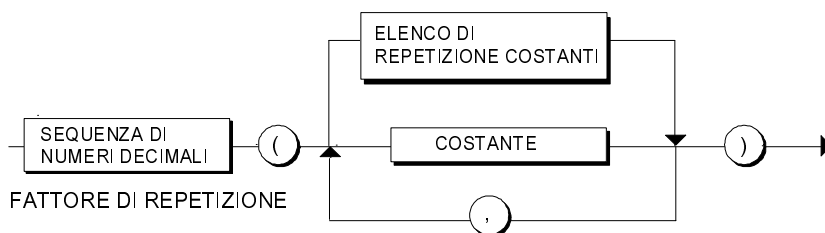
Inizializzazione di campo

Nell'inizializzazione di ARRAY, per ciascun componente di campo si può indicare un valore separato da virgole oppure si possono inizializzare varie componenti con lo stesso valore facendo precedere un valore di ripetizione (Integer).

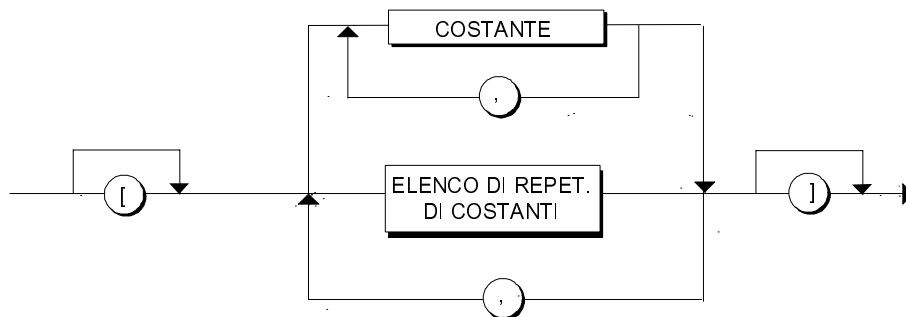
I valori iniziali possono essere inseriti in una parentesi. Anche nel caso di campi multidimensionali viene indicata solo una coppia di parentesi.

Sintassi per l'inizializzazione di campo

ELENCO DI REPETIZIONE COSTANTI



LISTA INIZIALIZZAZIONE DI CAMPO



Esempi

```
// Inizializzazione di variabili statiche:
INDEX1 : INT := 3 ;
//Inizializzazione di campo:
REGOLATORE1 : ARRAY [1..2, 1..2] OF INT := -54, 736, -83, 77;
REGOLATORE2 : ARRAY[1..10] OF REAL := 10(2.5);
REGOLATORE1 : ARRAY [1..2, 1..2] OF INT := [-54, 736, -83, 77];
REGOLATORE2 : ARRAY[1..10] OF REAL := [10(2.5)];

//Inizializzazione della struttura:
GENERATORE: STRUCT
    DAT1 : REAL := 100.5;
    A1 : INT := 10 ;
    A2 : STRING[6] := 'FATTORE';
    A3 : ARRAY[1..12] OF REAL := 0.0, 10(100.0), 1.0;
END_STRUCT ;
END_VAR
```

8.4 Dichiarazione di accessi a campi di variabili

Per accedere con un diverso tipo di dati ad una variabile dichiarata si possono definire degli accessi alla variabile o a campi interni alla variabile utilizzando la parola chiave "AT". Gli accessi sono visibili solo localmente nel blocco e non vengono trasferiti nell'interfaccia. Possono essere utilizzati nel blocco come qualsiasi altra variabile ed ereditano le proprietà della variabile a cui puntano, solo il tipo di dati è nuovo.

Esempio

Nel seguente esempio vengono realizzati più accessi ad un parametro di ingresso:

```
VAR_INPUT
    Buffer : ARRAY[0..255] OF BYTE;
    Telegramma1 AT Buffer : UDT100 ;
    Telegramma2 AT Buffer : UDT200 ;
END_VAR
```

Il blocco richiamante alimenta il parametro Buffer ma non vede i nomi Telegramma1 e Telegramma2. Ha quindi tre possibilità per interpretare i dati: come campo con il nome Buffer o con una diversa struttura con i nomi Telegramma1 o Telegramma2.

Regole

- La dichiarazione di un altro accesso ad una variabile deve essere effettuata dopo la dichiarazione della variabile a cui punterà e nello stesso blocco di dichiarazione.
- Non è possibile alcuna inizializzazione.
- Il tipo di dati dell'accesso deve essere compatibile con quello della variabile. La variabile definisce la dimensione dell'area di memoria. La memoria richiesta dall'accesso deve essere uguale o inferiore. Valgono inoltre le seguenti regole:

		Tipo di dati dell'accesso:	Tipo di dati della variabile:		
			Semplici	Composti	ANY/POINTER
FB	Dichiarazione di un accesso in VAR, VAR_TEMP, VAR_IN o VAR_OUT	Semplici Composti ANY/POINTER	x x	x x x (1)	x (1)
	Dichiarazione di un accesso in VAR_IN_OUT	Semplici Composti ANY/POINTER	x	x	
FC	Dichiarazione di un accesso in VAR o VAR_TEMP	Semplici Composti ANY/POINTER	x x	x x x	x
	Dichiarazione di un accesso in VAR_IN, VAR_OUT o VAR_IN_OUT	Semplici Composti ANY/POINTER	x	x	

(1) Any_Pointer in VAR_OUT non ammesso.

Elementari = BOOL, BYTE, WORD, DWORD, INT, DINT, DATE, TIME, S5TIME, CHAR

Composti = ARRAY, STRUCT, DATE_AND_TIME, STRING

8.5 Uso di multiistanze

È possibile che a causa dei dati utili (ad es. spazio di memoria) della CPU S7 utilizzata si possa o voglia mettere a disposizione solo un numero limitato di blocchi dati per i dati di istanza. Se nel programma utente vengono richiamati in un FB ulteriori blocchi funzionali già presenti (gerarchia di richiamo di FB), si potranno richiamare tali ulteriori blocchi funzionali senza dei propri (ovvero supplementari) DB di istanza.

Si offre la soluzione seguente.

- Inserire gli FB da richiamare come variabili statiche nella dichiarazione delle variabili dell'FB da richiamare.
- In questo blocco funzionale si richiamano ulteriori blocchi funzionali senza DB di istanza propri.
- Ciò consentirà di raggruppare i dati di istanza in un unico blocco dati di istanza e di utilizzare tutti i DB disponibili nel modo più vantaggioso.

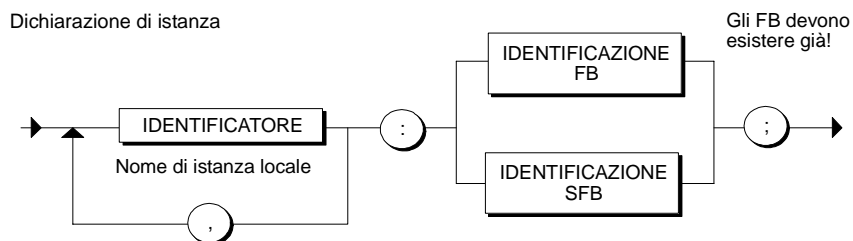
8.6 Dichiarazione di istanze

Nei blocchi funzionali, nel blocco convenzione per variabili statiche (VAR; END_VAR), oltre alle variabili con tipi di dati semplici, composti o definiti dall'utente, si possono definire anche variabili di tipo FB o SFB. Queste variabili vengono denominate istanze locali dell'FB o dell'SFB.

I dati di istanza locali vengono memorizzati nel blocco dati di istanza del blocco funzionale richiamante. Un'inizializzazione locale, in base alle specifiche, dell'istanza non è possibile.

I blocchi richiamati come istanze locali non possono avere lunghezza 0 e l'utente vi deve dichiarare almeno una variabile statica o un parametro.

Sintassi



Esempio

```

Assegnazione1      : FB10;
Assegnazione2, Assegnazione3, Assegnazione4      FB100;
Motore1            : Motore ;
  
```

Motore è un simbolo registrato nella tabella dei simboli per un FB.

8.7 Flag (Flag OK)

Il flag OK serve per prendere nota dell'esecuzione corretta o errata di un blocco. Si tratta di una variabile locale del tipo BOOL con il nome predefinito "OK".

All'inizio dell'esecuzione di un programma il flag OK ha il valore TRUE. Tale valore può essere interrogato oppure impostato su TRUE / FALSE in qualsiasi posizione di un blocco con istruzioni S7-SCL. Se si verifica un errore durante l'esecuzione di un'operazione (ad es. una divisione per zero), il flag OK viene impostato su FALSE. Quando si esce dal blocco, il valore del flag OK viene memorizzato nel parametro di uscita ENO e può quindi essere analizzato dal blocco richiamaente.

Dichiarazione

Il flag OK è una variabile definita dal sistema. Non è necessaria alcuna convenzione. Tuttavia, se si desidera utilizzare un flag OK nel proprio programma utente è necessario selezionare l'opzione di compilazione "Imposta flag OK" prima della compilazione.

Esempio

```
// Impostare il flag OK su TRUE
// per poter controllare se l'azione
// viene eseguita correttamente.
OK:= TRUE;
Divisione:= 1 / IN;
IF OK THEN
    // La divisione è stata eseguita correttamente.
    // :
    // :
ELSE
    // La divisione è stata eseguita in modo errato.
    // :
END_IF;
```

8.8 Componenti di dichiarazione

8.8.1 Sommario delle sezioni di dichiarazione

Ad ogni categoria di variabili o parametri locali viene assegnato un proprio blocco convenzione, il quale è contrassegnato da una coppia di parole chiave. Ogni blocco contiene le dichiarazioni consentite per questo blocco convenzione. La sequenza dei blocchi è irrilevante.

La tabella seguente mostra quali tipi di variabili o parametri si possono elaborare nei vari blocchi di codice.

Dati	Sintassi	FB	FC	OB
Variabili come: Variabili statiche	VAR ... END_VAR	X	X *)	
Variabili temporanee	VAR_TEMP ... END_VAR	X	X	X
Parametri di blocchi come: Parametri d'ingresso	VAR_INPUT ... END_VAR	X	X	
Parametri d'uscita	VAR_OUTPUT ... END_VAR	X	X	
Parametri di ingresso/uscita	VAR_IN_OUT ... END_VAR	X	X	

*) La convenzione delle variabili all'interno della coppia di parole chiave VAR e END_VAR è ammessa anche nelle funzioni, ma in seguito alla compilazione di una sorgente le convenzioni vengono create nell'area temporanea.

8.8.2 Variabili statiche

Le variabili statiche sono variabili locali i cui valori rimangono inalterati in tutte le esecuzioni dei blocchi (memoria dei blocchi). Esse servono per memorizzare il valore di un blocco funzionale e vengono memorizzate nel corrispondente blocco dati di istanza.

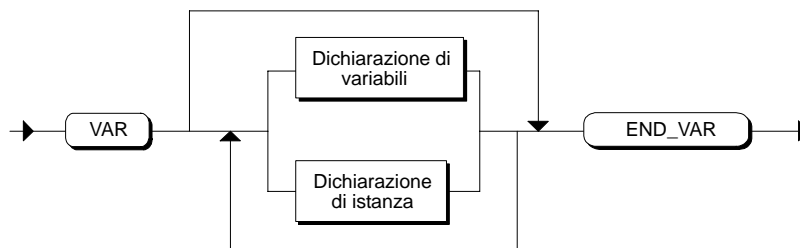
Sintassi

Le variabili statiche vengono dichiarate nella sezione di dichiarazione VAR / END_VAR. Questo blocco convenzioni fa parte della parte dichiarazioni dell'FB. Dopo la compilazione, insieme con i blocchi per i parametri del blocco, questo blocco determina la struttura del blocco dati di istanza assegnato.

In questo blocco è possibile effettuare quanto segue:

- creare variabili, assegnarvi un tipo di dati e inizializzarle
- dichiarare come variabile statica un FB da richiamare per poterlo richiamare come istanza locale nell'FB attuale.

Blocco di variabili statiche



Esempio

```
VAR
ESECUZIONE           :INT;
CAMPO_MISURA         :ARRAY [1..10] OF REAL;
INTERRUTTORE          :BOOL;
MOTORE_1,MOTORE_2     :FB100;    //Dichiarazione di un'istanza

END_VAR
```

Accesso

L'accesso alle variabili avviene nella parte istruzioni:

- **Accesso all'interno del blocco:** nella parte istruzioni dell'FB nella cui parte convenzioni è stata dichiarata una variabile è possibile accedere alla variabile. La procedura dettagliata è descritta nel capitolo "Assegnazione di valori".
- **Accesso dall'esterno tramite il DB di istanza:** è possibile indirizzare una variabile da altri blocchi utilizzando un accesso indicizzato ad esempio *DBx.variabale*.

8.8.3 Variabili temporanee

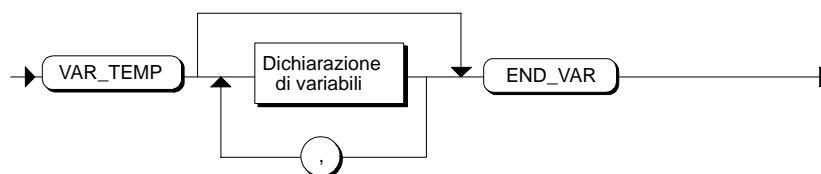
Le variabili temporanee fanno parte di un blocco di codice e non occupano alcuna area di memoria statica. Esse vengono depositate nello stack della CPU. Il loro valore rimane costante solo durante l'esecuzione di un blocco. Al di fuori del blocco in cui sono state dichiarate le variabili non è possibile accedere alle variabili temporanee. All'inizio dell'esecuzione di un OB, FB o FC il valore dei dati temporanei non è definito. Non è possibile alcuna inizializzazione.

Si consiglia di definire dei dati come dati temporanei se essi sono necessari per memorizzare risultati intermedi durante l'elaborazione dei propri OB, FB o FC.

Sintassi

La convenzione delle variabili temporanee viene effettuata nella sezione di dichiarazione VAR_TEMP / END_VAR. Questo blocco convenzioni fa parte di un FB, una FC o un OB. Conformemente al capitolo 1.2 nell'ambito della dichiarazione di variabili vengono indicati i nomi delle variabili e i tipi di dati.

Blocco di variabili temporanee



Non è possibile alcuna inizializzazione

Esempio

```
VAR_TEMP
  BUFFER 1 : ARRAY [1..10] OF INT ;
  AUSIL1, AUSIL2 : REAL; END_VAR
```

Accesso

L'accesso alle variabili avviene sempre nella parte istruzioni del blocco di codice nella cui parte dichiarazioni è stata dichiarata la variabile (accesso dall'interno), vedere capitolo "Assegnazione di valori".

8.8.4 Parametri di blocchi

I parametri sono dei segnaposti che vengono definiti solo in caso di utilizzo effettivo (richiamo) del blocco. I segnaposti presenti (definiti) nel blocco vengono denominati parametri formali, mentre i valori assegnati durante il richiamo del blocco vengono denominati parametri attuali. I parametri costituiscono quindi un meccanismo che consente lo scambio di informazioni fra i blocchi.

Tipi di parametri di blocchi:

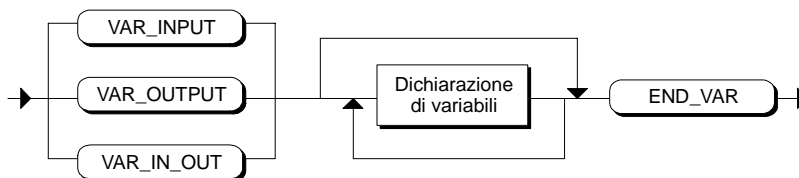
- Parametri formali d'ingresso accettano i valori d'ingresso attuali.
(Flusso di dati dall'esterno verso l'interno)
- I parametri formali di uscita servono per il trasferimento di valori di uscita.
(Flusso di dati dall'interno verso l'esterno)
- I parametri formali di ingresso/uscita svolgono sia la funzione di parametri d'ingresso che di parametri di uscita.

Sintassi

La definizione dei parametri formali avviene nella parte convenzioni di un blocco funzionale o di una funzione, separatamente per ciascun tipo di parametro nei tre blocchi convenzione per i parametri. Nell'ambito della dichiarazione di variabili vengono definiti i nomi dei parametri e il tipo di dati. Una inizializzazione può essere eseguita solo con i parametri di ingresso e di uscita di un FB.

Nella convenzione dei parametri formali, oltre ai tipi di dati semplici, composti e definiti dall'utente, si possono utilizzare anche i tipi di dati per parametri.

Blocco di parametri



Inizializzazione possibile solo per VAR_INPUT e VAR_OUTPUT

Esempio

```
VAR_INPUT          // Parametro d'ingresso
    MIO_DB : BLOCK_DB ;
    REGOLATORE : DWORD ;
    ORA : TIME_OF_DAY ;
END_VAR

VAR_OUTPUT          // Parametro di uscita
    VALNOMINALI: ARRAY [1..10] of INT ;
END_VAR

VAR_IN_OUT          // Parametro di ingresso/uscita
    IMPOSTAZIONE : INT ;
END_VAR
```

Accesso

L'accesso ai parametri di blocco avviene nella parte istruzioni di un blocco di codice:

- **Accesso dall'interno:** cioè nella parte istruzioni del blocco nella cui parte dichiarazioni è stato dichiarato il parametro. Ciò viene spiegato nel capitolo "Assegnazione di valori" e nel capitolo "Espressioni, operatori e operandi".
- **Accesso dall'esterno tramite il DB di istanza:** si può accedere ai parametri di blocchi funzioni tramite i DB di istanza assegnati.

9 Convenzioni di costanti e etichette di salto

9.1 Costanti

Le costanti sono dati con determinati valori fissi, i quali non possono essere modificati durante l'esecuzione del programma.

S7-SCL utilizza i seguenti gruppi di costanti.

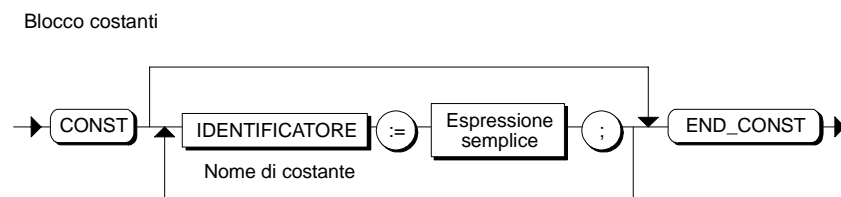
- Costanti a bit
- Costanti numeriche
 - Costanti di numero intero
 - Costanti di numero reale
- Costanti di caratteri
 - Costanti Char
 - Costanti String
- Indicazione della data
 - Costanti di data
 - Costanti di intervallo
 - Costanti dell'ora del giorno
 - Costanti di data e ora

9.1.1 Convenzione per nomi simbolici di costanti

Non è necessario una convenzione di costanti. Sussiste, comunque, la possibilità di assegnare dei nomi simbolici alle costanti nella parte dichiarazioni.

La convenzione per i nomi simbolici di costanti avviene con l'istruzione CONST nella parte convenzioni del blocco di codice. Essa viene raccomandata per tutte le costanti di un blocco. In tal modo si consegue una migliore leggibilità ed una più semplice gestione del blocco in caso di modifica di valori costanti.

Sintassi



Nelle espressioni semplici sono possibili solo le operazioni aritmetiche di base (*, /, +, -, **, DIV, MOD).

Esempio

```

CONST
  Numero      := 10 ;
  ORA1        := TIME#1D_1H_10M_22S_2MS ;
  NOME        := 'SIEMENS' ;
  NUMERO2     := 2 * 5 + 10 * 4 ;
  NUMERO3     := 3 + NUMERO2 ;
END_CONST
    
```

9.1.2 Tipi di dati delle costanti

L'assegnazione dei tipi di dati alle costanti è diversa dalla procedura prevista in AWL:

la costante viene definita con il tipo di dati solo in seguito alla combinazione aritmetica o logica in cui viene utilizzata, ad esempio:

```
Int1:=Int2 + 12345    //viene attribuito a "12345" il tipo
                      //di dati INT
Real1:=Real2 + 12345  //viene attribuito a "12345"
                      //il tipo di dati REAL
```

Alla costante viene assegnato il tipo di dati il cui campo di valori è appena sufficiente per poter memorizzare la costante senza alcuna perdita di valore. Ad esempio la costante "12345" non ha sempre il tipo di dati INT come in AWL, ma la classe di tipi di dati ANY_NUM in funzione dell'utilizzo che ne viene fatto, ovvero INT, DINT o REAL.

Costanti tipizzate

La modalità tipizzata di scrittura delle costanti consente di specificare anche esplicitamente un tipo di dati per i seguenti tipi di dati numerici.

Esempi:

Tipo di dati	Modalità di scrittura tipizzata	
BOOL	BOOL#1 Bool#false	bool#0 BOOL#TRUE
BYTE	BYTE#0 Byte#'ä'	B#2#101 b#16#f
WORD	WORD#32768 W#2#1001_0100	word#16#f WORD#8#177777
DWORD	DWORD#16#f000_0000 DW#2#1111_0000_1111_0000	dword#32768 DWord#8#3777777777
INT	INT#16#3f_ff Int#2#1111_0000	int#-32768 inT#8#77777
DINT	DINT#16#3fff_ffff DInt#2#1111_0000	dint#-65000 dinT#8#1777777777
REAL	REAL#1 real#2e4	real#1.5 real#3.1
CHAR	CHAR#A	CHAR#49

9.1.3 Modo di scrittura per costanti

Per il valore di una costante esiste una determinata modalità di scrittura (formato) a seconda del tipo e del formato dei dati: Il tipo e il valore delle costanti risultano direttamente dalla modalità di scrittura e non devono essere quindi dichiarati.

Esempi:

15	VALORE 15	come costante di numero intero in rappresentazione decimale
2#1111	VALORE 15	come costante di numero intero in rappresentazione binaria
16#F	VALORE 15	come costante di numero intero in rappresentazione esadecimale

Possibili modalità di scrittura

Tipo di dati	Descrizione	Esempi in S7-SCL	Esempi in AWL, se diversi
BOOL	1 bit	FALSE TRUE BOOL#0 BOOL#1 BOOL#FALSE BOOL#TRUE	
BYTE	Numero esadecimale di 8 bit	B#16#00 B#16#FF BYTE#0 B#2#101 Byte#'ä' b#16#f	
CHAR	8 bit (1 carattere ASCII)	'A' CHAR#49	
STRING	Max. 254 caratteri ASCII	'indirizzo'	
WORD	Numero esadecimale di 16 bit Numero ottale di 16 bit Numero binario di 16 bit	W#16#0000 W#16#FFFF word#16#f WORD#8#177777 8#177777 W#2#1001_0100 WORD#32768	
DWORD	Numero esadecimale di 32 bit Numero ottale di 32 bit Numero binario di 32 bit	DW#16#0000_0000 DW#16#FFFF_FFFF Dword#8#3777777777 8#3777777777 DW#2#1111_0000_1111_0000 dword#32768	

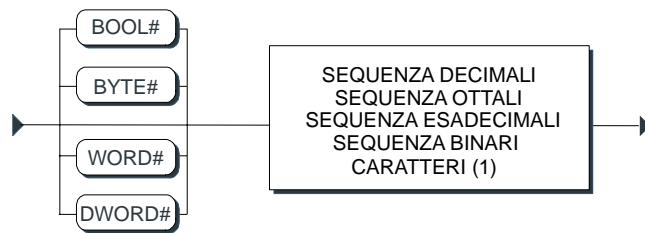
Tipo di dati	Descrizione	Esempi in S7-SCL	Esempi in AWL, se diversi
INT	Numero in virgola mobile di 16 bit	-32768 +32767 INT#16#3f_ff int#-32768 Int#2#1111_0000 inT#8#77777	
DINT	Numero in virgola mobile di 32 bit	-2147483648 +2147483647 DINT#16#3fff_ffff dint#-65000 Dint#2#1111_0000 dinT#8#1777777777	L#-2147483648 L#+2147483647
REAL	Numero in virgola mobile di 32 bit	Rappresentazione decimale 123.4567 REAL#1 real#1.5 Rappresentazione esponenziale real#2e4 +1,234567E+02	
S5TIME	Valore di tempo di 16 bit in formato SIMATIC	T#0ms TIME#2h46m30s T#0.0s TIME#24.855134d	S5T#0ms S5TIME#2h46m30s
TIME	Valore di tempo di 32 bit in formato IEC	-T#24d20h31m23s647ms TIME#24d20h31m23s647ms T#0.0s TIME#24.855134d	
DATE	Valore della data di 16 bit	D#1990-01-01 DATE#2168-12-31	
TIME_OF_DAY	Ora del giorno di 32 bit	TOD#00:00:00 TIME_OF_DAY#23:59:59.999	
DATE_AND_TIME	Valori di data e ora	DT#1995-01-01-12:12:12.2	

9.1.3.1 Costanti a bit

Le costanti a bit contengono valori con lunghezza di 1 bit, 8 bit, 16 bit o 32 bit. Nel programma S7-SCL, questi ultimi possono essere assegnati (a seconda della lunghezza) a variabili con i tipi di dati BOOL, BYTE, WORD e DWORD.

Sintassi

COSTANTE DI BIT



(1) solo nel tipo di dati BYTE

Sequenza di cifre decimali

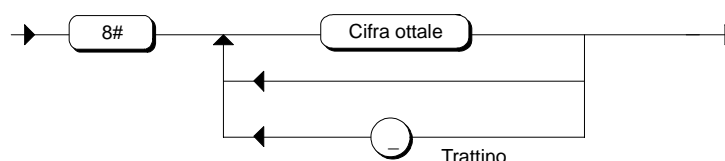
Il punto decimale all'interno delle costanti si compone di una sequenza di cifre che, come opzione, possono essere separate da trattini. I trattini hanno lo scopo di migliorare la leggibilità di grandi numeri. Gli esempi seguenti contengono valide modalità di scrittura per sequenze di cifre decimali all'interno delle costanti:

```
DW#2#1111_0000_1111_0000
dword#32768
```

Valori binari, ottali ed esadecimali

Per indicare una costante di numero intero in un sistema numerico diverso da quello decimale si devono specificare i prefissi **2#**, **8#** o **16#** prima del numero nel formato del sistema scelto. Questo viene illustrato nella figura seguente con un esempio di una sequenza di cifre per un numero ottale:

Sequenza di cifre ottali



Esempio

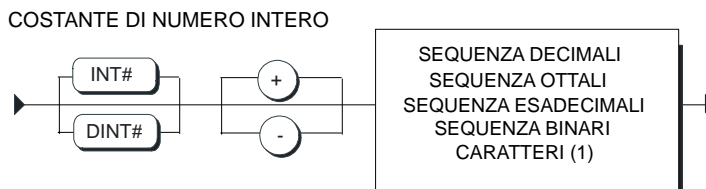
I seguenti esempi sono possibili modalità di scrittura delle costanti a bit:

```
Bool#false
8#177777
DW#16#0000_0000
```


9.1.3.2 Costante di numero intero

Le costanti di numero intero contengono valori di numero intero con una lunghezza di 16 o 32 bit. Nel programma S7-SCL questi ultimi possono essere assegnati (a seconda della lunghezza) a variabili con i tipi di dati INT o DINT

Sintassi



(1) solo nel tipo di dati INT

Sequenza di cifre decimali

Il punto decimale all'interno delle costanti si compone di una sequenza di cifre che, come opzione, possono essere separate da trattini. I trattini hanno lo scopo di migliorare la leggibilità di grandi numeri. Gli esempi seguenti contengono valide modalità di scrittura per sequenze di cifre decimali all'interno delle costanti:

1000

1_120_200

666_999_400_311

Valori binari, ottali, esadecimali

Per indicare una costante di numero intero in un sistema numerico diverso da quello decimale si devono specificare i prefissi **2#**, **8#** o **16#** prima del numero nel formato del sistema scelto.

Esempio

I seguenti esempi sono possibili modalità di scrittura delle costanti di numero intero:

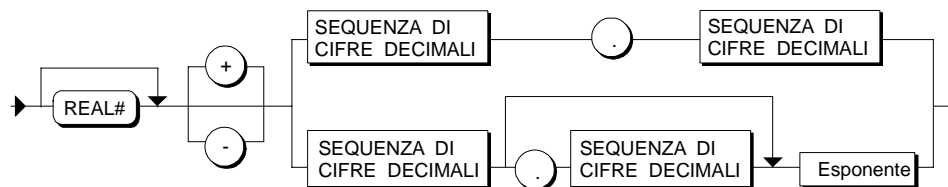
```
Valore_2:=2#0101;      // numero binario, valore decimale 5
Valore_3:=8#17;        // numero ottale, valore decimale 14
Valore_4:=16#F;        // numero esadecimale, valore decimale 15
Valore_5:=INT#16#3f_ff // numero esadecimale, modalità
                        // di scrittura tipizzata
```

9.1.3.3 Costante di numero reale

Le costanti di numero reale sono valori con decimali. Essi possono essere assegnati a variabili del tipo di dati REAL.

Sintassi

COSTANTE DI NUMERO REALE



L'indicazione del segno è opzionale. Se non viene indicato alcun segno, il numero viene interpretato come numero positivo.

Il punto decimale all'interno delle costanti si compone di una sequenza di cifre che, come opzione, possono essere separate da trattini. I trattini hanno lo scopo di migliorare la leggibilità di grandi numeri. Gli esempi seguenti contengono valide modalità di scrittura per sequenze di cifre decimali all'interno delle costanti:

```
1000
1_120_200
666_999_400_311
```

Esponente

Nel modo di scrittura esponenziale, per l'introduzione di numeri in virgola mobile si può utilizzare un esponente. L'indicazione dell'esponente avviene premettendo la lettera "E" o "e" seguita da un valore di numero intero.

In S7-SCL il valore 3×10^{10} può essere rappresentato con i seguenti numeri in virgola mobile:

3.0E+10	3.0E10	3e+10	3E10
0.3E+11	0.3e11	30.0E+9	30e9

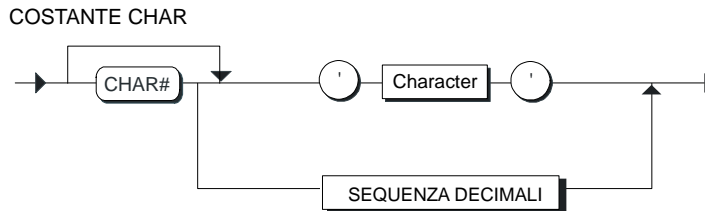
Esempi

```
NUMERO4:= -3.4 ;
NUMERO5:= 4e2 ;
NUMERO6:= real#1.5;
```

9.1.3.4 Costante Char (carattere singolo)

La costante Char contiene un solo carattere racchiuso fra virgolette semplici (''). Questo tipo di costanti non è utilizzabile nelle espressioni.

Sintassi



Esempio

```

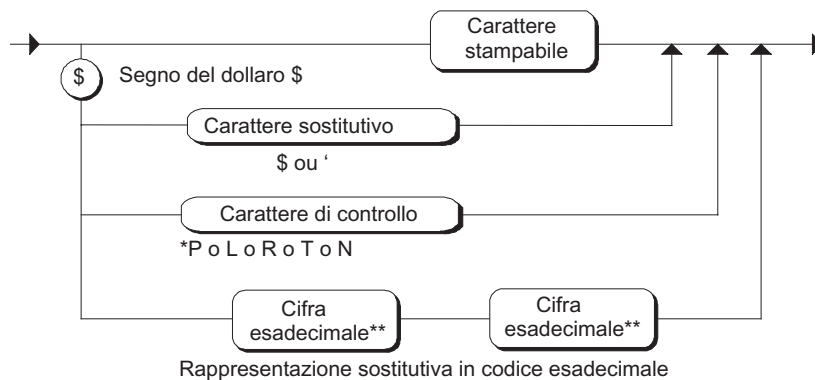
Carattere_1 := 'B';
Carattere_2 := char#43;
Carattere_3 := char#'B';
  
```

Sintassi per i caratteri

Il carattere può appartenere al set di caratteri ASCII ampliato. Con il simbolo \$ si possono introdurre speciali caratteri di formattazione, le virgolette (') o un carattere \$.

Si possono introdurre anche caratteri non stampabili del set di caratteri ASCII ampliato. A tal fine si deve utilizzare la rappresentazione sostitutiva in codice esadecimale.

Carattere



* P=avanzamento pagina
L=avancamento riga
R=ritorno carella
T=tabulatore
N=nuova riga

** \$00 non consentito

Esempio per l'introduzione di un carattere in codice esadecimale:

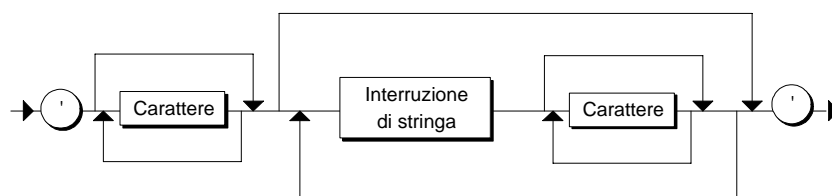
```
CARATTERE      := '$41' ; //corrisponde al carattere 'A'
Spazio vuoto    := '$20';  //corrisponde al carattere  
```

9.1.3.5 Costante String

Una costante String è una stringa di max. 254 caratteri. I caratteri sono racchiusi fra virgolette semplici. Le costanti String non sono utilizzabili nelle espressioni.

Sintassi

COSTANTE STRING

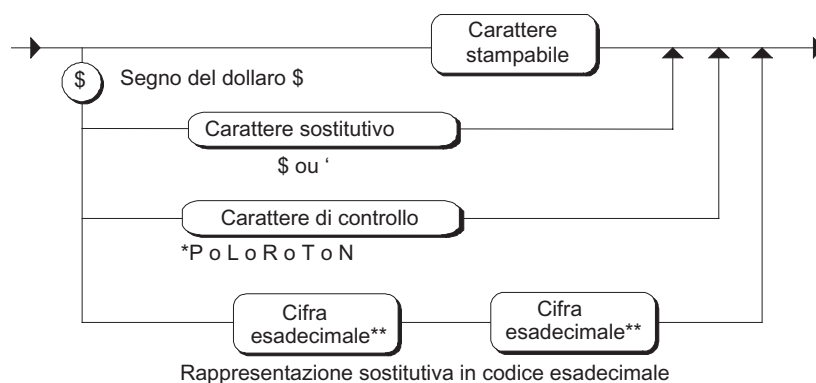


Sintassi per i caratteri

Il carattere può far parte del set di caratteri ampliato ASCII. Con il simbolo \$ si possono introdurre speciali caratteri di formattazione, le virgolette (') o un carattere \$.

Si possono introdurre anche caratteri non stampabili del set di caratteri ASCII ampliato. A tal fine si deve utilizzare la rappresentazione sostitutiva in codice esadecimale.

Carattere



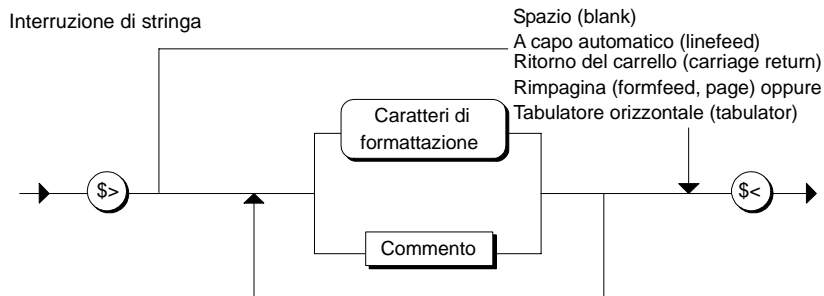
* P=avanzamento pagina
L=avancamento riga
R=ritorno carella
T=tabulatore
N=nuova riga

** \$00 non consentito

Interruzione di stringa

Le costanti String possono essere interrotte e proseguite più volte.

Una stringa è situata in una riga di un blocco S7-SCL oppure viene ripartita su più righe mediante identificazioni speciali. Per interrompere una stringa utilizzare \$>, per proseguirla nella riga successiva utilizzare \$<. Lo spazio tra queste due indicazioni può comprendere più righe e contenere, oltre agli spazi vuoti, solo commenti.



Esempi

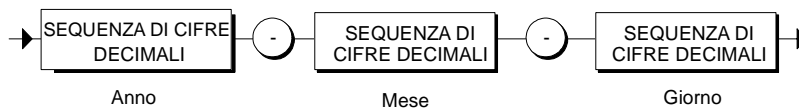
```
// Costante String:
NOME:= 'SIEMENS';
//Interruzione di una costante String
MESSAGGIO1:= 'MOTORE- $>
$< controllo';
// Stringa in formato esadecimale:
MESSAGGIO1:= '$41$4E' (*Sequenza di caratteri AN*);
```

9.1.3.6 Costante di data

Una data viene preannunciata con i prefissi DATE# o D#. L'indicazione della data avviene tramite numeri interi per la cifra dell'anno (di 4 cifre), indicazione della data e del giorno, i quali devono essere separati mediante trattini.

Sintassi

Indicazione della data



Esempio

```

VARTEMPORALE1 := DATE#1995-11-11 ;
VARTEMPORALE2 := D#1995-05-05 ;
    
```

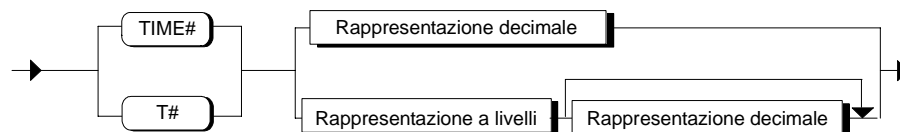
9.1.3.7 Costante di intervallo

Un intervallo viene preannunciato dai prefissi TIME# o T#. L'indicazione dell'intervallo può aver luogo in due modi:

- rappresentazione decimale
- rappresentazione a livelli

Sintassi

DURATA



- Ogni unità di tempo (p. es. ore, minuti) può essere indicata una sola volta.
- La sequenza - giorni, ore, minuti, secondi, millisecondi - deve essere rispettata

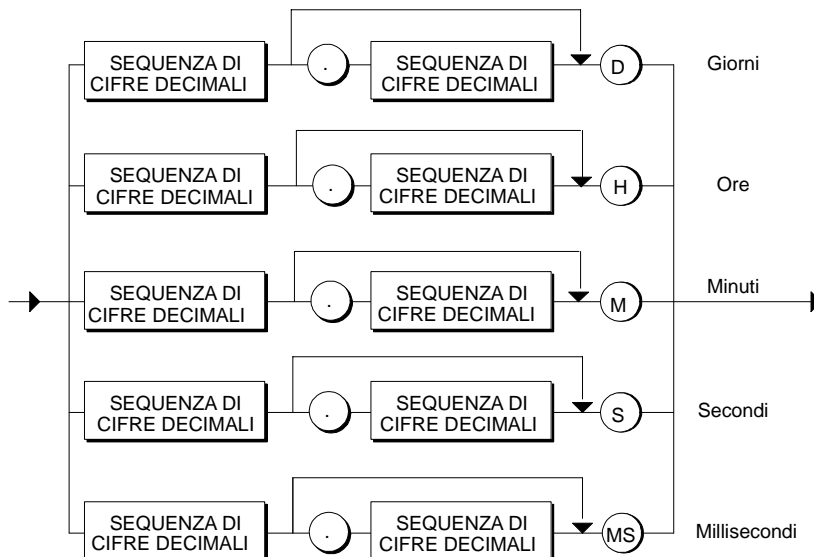
Il passaggio dalla rappresentazione a livelli alla rappresentazione decimale è possibile solo per le unità di tempo che non sono state ancora introdotte.

Dopo i prefissi iniziali T# o TIME# si deve introdurre almeno un'unità di tempo.

Rappresentazione decimale

La rappresentazione decimale viene utilizzata quando si desidera introdurre come numero intero una componente di tempo come per esempio ore o minuti.

Rappresentazione decimale

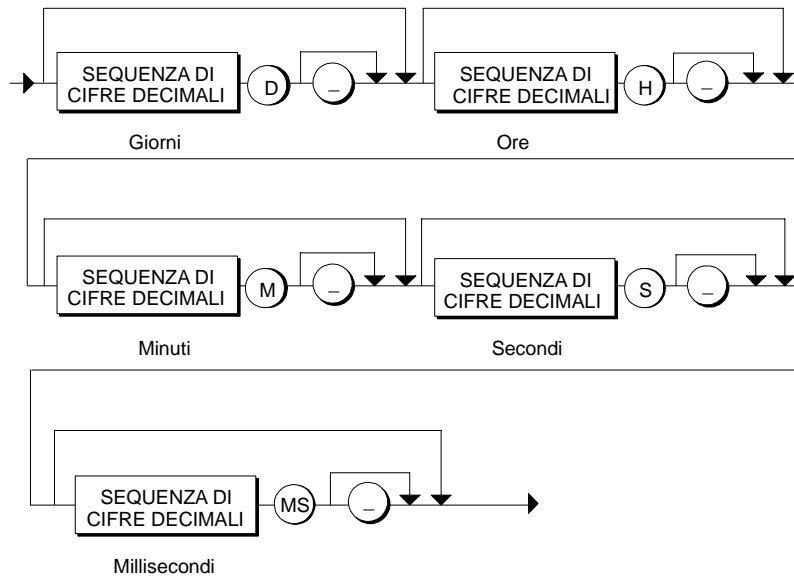


L'accesso alla rappresentazione decimale è possibile solo con unità di tempo non ancora definite.

Rappresentazione a livelli

La rappresentazione a livelli è una sequenza dei singoli componenti di tempo. Vengono introdotti dapprima i giorni, seguiti dalle ore, ecc. e separati da trattini; è consentita l'omissione di componenti. Si deve però indicare almeno una componente.

Rappresentazione a livelli



Esempio

```
// Rappresentazione decimale
Intervallo1:= TIME#10.5S ;

// Rappresentazione a livelli
Intervallo2:= T#3D_2S_3MS ;

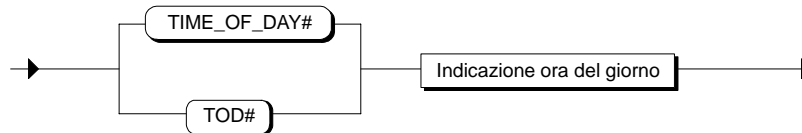
// Rappresentazione a livelli e decimale
Intervallo3 := T#2D_2.3s ;
```

9.1.3.8 Costante ora del giorno

L'indicazione dell'ora del giorno viene preannunciata dai prefissi TIME_OF_DAY# o TOD#.

Sintassi

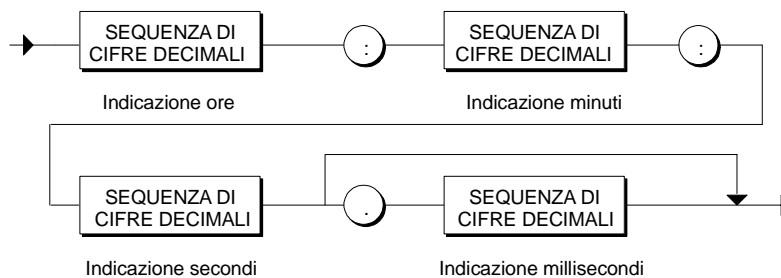
ORA DEL GIORNO



L'indicazione dell'ora avviene introducendo ore, minuti e secondi, i quali devono essere separati mediante doppio punto. L'indicazione dei millisecondi è opzionale. Essa viene separata dagli altri componenti tramite un punto.

Indicazione dell'ora

Indicazione ora del giorno



Esempio

```

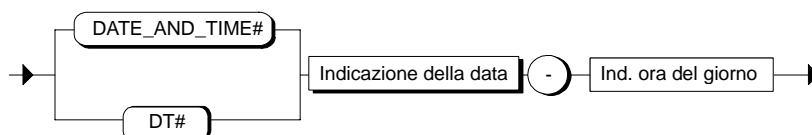
ORA1:= TIME_OF_DAY#12:12:12.2 ;
ORA2:= TOD#11:11:11 ;
  
```

9.1.3.9 Costante di data e ora

L'indicazione della data e dell'ora viene preannunciata dai prefissi DATE_AND_TIME# o DT#. Si tratta di una costante composta dalle indicazioni di data e ora.

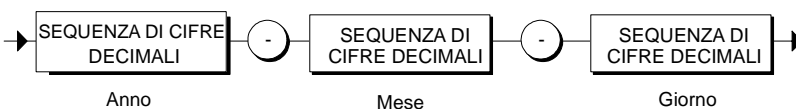
Sintassi

DATA E ORA



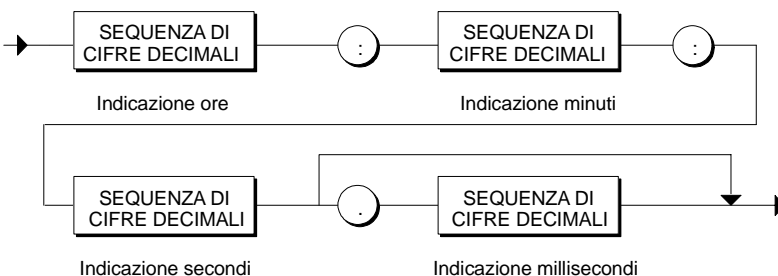
Indicazione della data

Indicazione della data



Indicazione dell'ora

Indicazione ora del giorno



Esempio

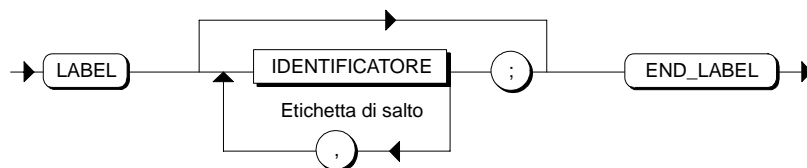
```
ORA1 := DATE_AND_TIME#1995-01-01-12:12:12.2 ;
ORA2 := DT#1995-02-02-11:11:11;
```

9.2 Convenzione di etichette di salto

Le etichette di salto (Labels) servono per definire l'obiettivo di una istruzione GOTO. Esse vengono definite nella parte convenzioni di un blocco di codice con i loro nomi simbolici.

Sintassi

Blocco etichette di salto



Esempio

```
LABEL  
    ETICHETTA1, ETICHETTA2, ETICHETTA3;  
END_LABEL
```

10 Dati globali

10.1 Dati globali

In S7-SCL si ha la possibilità di accedere ai dati globali. Esistono due tipi di dati globali:

- **Aree di memoria della CPU**

Queste aree di memoria sono dati definiti dal sistema: p. es. ingressi, uscite e merker. La dimensione delle aree di memoria a disposizione dell'utente dipende dalla CPU impiegata.

- **Dati utente globali come blocchi dati caricabili**

Queste aree di dati sono situate all'interno di blocchi dati. Per poterle utilizzare occorre dapprima creare i blocchi dati e definire i propri dati in tali blocchi. Nel caso di blocchi dati di istanza, essi vengono derivati da blocchi funzionali e generati automaticamente.

Accesso ai dati globali

L'accesso ai dati globali può aver luogo nei modi seguenti:

- **assoluto:** mediante identificazione di operando e indirizzo assoluto.
- **simbolico:** mediante un simbolo che l'utente ha definito in precedenza nella tabella dei simboli.
- **indicizzato:** mediante identificazione di operando e indice di campo.
- **strutturato:** mediante una variabile.

Tipo di accesso	Aree di memoria della CPU	Blocchi dati
assoluto	sì	sì
simbolico	sì	sì
indicizzato	sì	sì
strutturato	no	sì

10.2 Aree di memoria della CPU

10.2.1 Aree di memoria della CPU

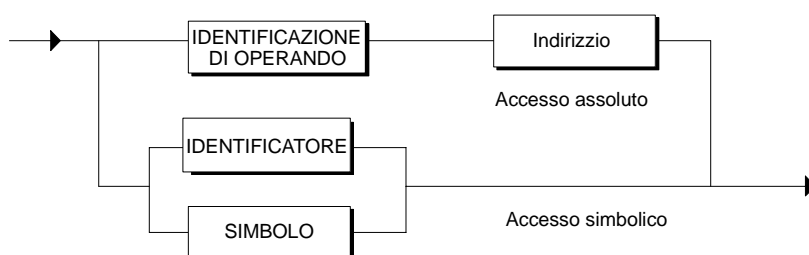
Le aree di memoria di una CPU sono aree definite nel sistema. Perciò l'utente non deve definire queste aree nel blocco di codice dell'utente. Ogni CPU mette a disposizione le seguenti aree di memoria con una propria area di indirizzamento:

- Ingressi / Uscite nell'immagine di processo (p. es. A1.0)
- Ingressi / Uscite di periferia (p. es. PA1.0)
- Merker (p. es. M1.0)
- Temporizzatori, contatori (C1)

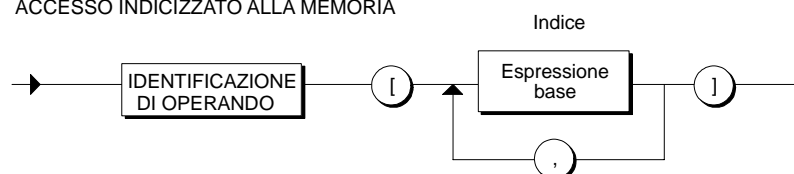
Sintassi per l'accesso:

- L'accesso ad un'area di memoria della CPU avviene mediante un'assegnazione di valori nella parte istruzioni di un blocco di codice: con un semplice accesso, che può essere del tipo assoluto o simbolico, oppure
- con un accesso indicizzato.

SEMPLICE ACCESSO ALLA MEMORIA



ACCESSO INDICIZZATO ALLA MEMORIA



10.2.2 Accesso simbolico alle aree di memoria della CPU

L'indirizzamento simbolico consente di utilizzare dei nomi simbolici al posto di identificatori assoluti per indirizzare le aree di memoria della CPU.

L'assegnazione dei nomi simbolici per i rispettivi operandi del programma utente avviene nella tabella dei simboli. Con il comando di menu **Strumenti > Tabella dei simboli** essa può essere aperta in qualsiasi momento in S7-SCL per aggiungervi ulteriori simboli.

Come tipi di dati sono consentiti tutti i tipi di dati semplici purché essi siano in grado di recepire la larghezza di accesso. Un esempio di tabella dei simboli è illustrato di seguito.

Simbolo	Indirizzo assoluto	Tipo di dati	Commento
Contatto_motore_1	E 1.7	BOOL	Interruttore a contatto 1 per motore A
Ingresso1	EW 10	INT	Parola di stato

Accesso

L'accesso si svolge tramite assegnazione di valore a una variabile dello stesso tipo con il simbolo convenuto.

Esempio

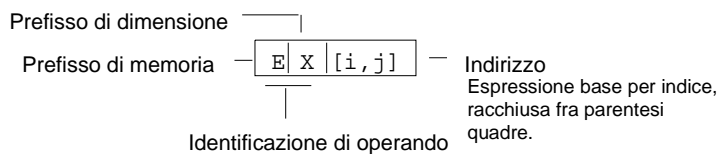
```
VALOREMISURA_1 := contatto motore 1;  
stato_motore1 := ingresso1 ;
```

10.2.3 Accesso indicizzato alle aree di memoria della CPU

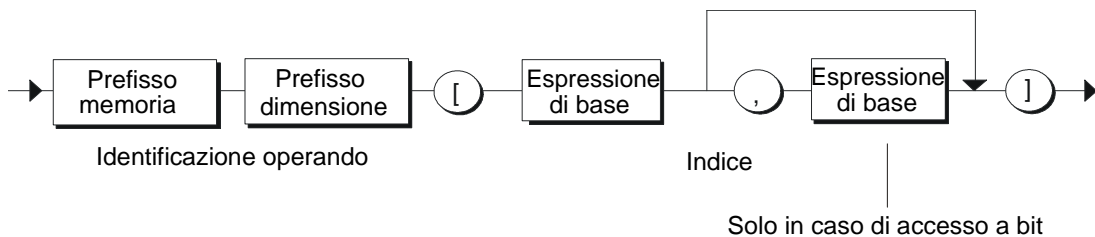
Esiste anche la possibilità di accedere con metodo indicizzato a specifiche aree della CPU. Rispetto all'indirizzamento assoluto, questo metodo offre il vantaggio di poter indirizzare dinamicamente gli indirizzi mediante impiego di indici variabili. Per es. come indirizzo si può utilizzare la variabile di esecuzione di un loop FOR.

L'accesso indicizzato ad un'area di memoria è simile all'accesso assoluto. L'unica differenza consiste nell'indicazione dell'indirizzo. Al posto dell'indirizzo viene specificato un indice che può essere una costante, una variabile o un'espressione aritmetica.

Nell'accesso indicizzato, l'identificatore assoluto si compone dell'identificazione di operando (prefisso di memoria e prefisso di lunghezza) nonché di un'espressione base per l'indicizzazione.



Sintassi dell'identificatore assoluto



L'indicizzazione (espressione) deve essere conforme alle regole seguenti:

- Ogni indice deve essere un'espressione aritmetica del tipo di dati INT
- Per un accesso con tipo di dati BYTE, WORD o DWORD si deve utilizzare esattamente un indice. L'indice viene interpretato come indirizzo a byte. La larghezza di accesso viene definita tramite il prefisso di lunghezza.
- In caso di accesso con tipo di dati BOOL si devono utilizzare due indici. Il primo indice specifica l'indirizzo a byte, mentre il secondo indice specifica la posizione del bit all'interno del byte.

Esempio

```
VALOREMISURA_1    :=EW [CONTATORE] ;
MARCHIO            :=E [NBYTE, NBIT] ;
```


10.2.4 Accesso assoluto alle aree di memoria della CPU

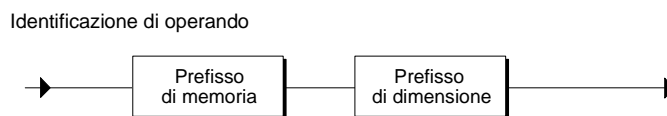
L'accesso assoluto alle aree di memoria della CPU avviene, ad esempio, mediante assegnazione di valore di un identificatore assoluto ad una variabile dello stesso tipo.

```
STATUS_2 := EB10;
```

| |
Variabile dello stesso tipo Accesso assoluto

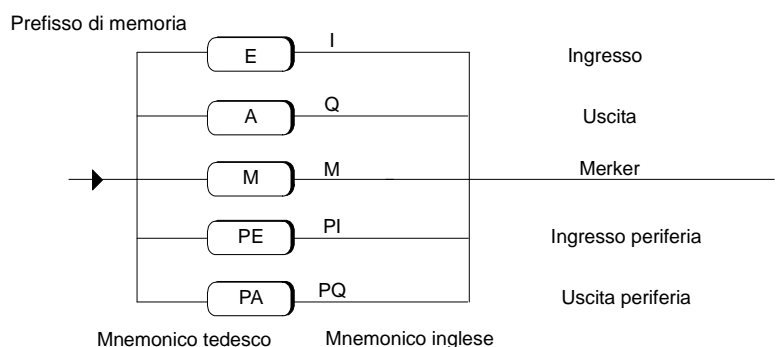
L'identificatore assoluto fa riferimento ad un'area di memoria all'interno della CPU. L'utente può specificare quest'area indicando l'identificazione di operando (qui **EB**) seguita dall'indirizzo (qui **10**).

Sintassi dell'identificatore assoluto



Prefisso di memoria

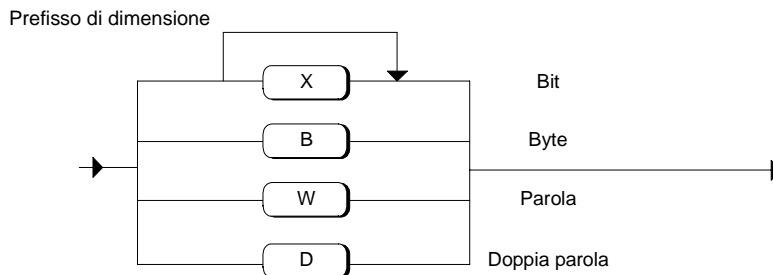
Tramite il prefisso di memoria si definisce il tipo di aree di memoria da indirizzare.



In funzione del mnemonico impostato, l'identificazione di operando SIMATIC o IEC assume un significato riservato.

Prefisso di lunghezza

Tramite l'indicazione del prefisso di lunghezza si specifica la lunghezza dell'area di memoria che si desidera leggere dalla periferia. Si può leggere p. es. un byte o una parola. Se si desidera specificare solo un bit, l'indicazione del prefisso di lunghezza è opzionale.



Indirizzo

L'indicazione dell'indirizzo avviene specificando dapprima l'indirizzo assoluto a byte e quindi, separati da un punto, l'indirizzo a bit del byte interessato. L'indicazione dell'indirizzo a bit è opzionale.

Indirizzo

Esempi

```
STATUSBYTE :=EB10;
STATUS_3   :=E1.1;
VALOREMISURA :=EW20;
```

10.3 Blocchi dati

10.3.1 Blocchi dati

Nell'ambito di blocchi dati, per la propria applicazione si possono memorizzare ed elaborare tutti i dati il cui campo di validità fa riferimento all'intero programma o all'intero progetto. Ogni blocco di codice può accedere in lettura o scrittura ai dati utente globali.

Accesso

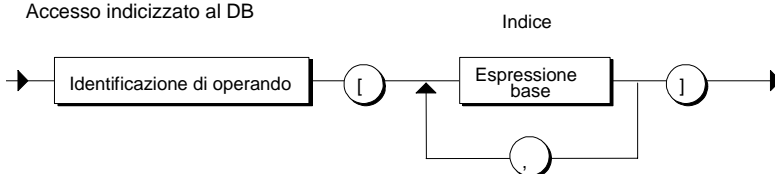
È possibile accedere ai dati del blocco dati globale nei seguenti modi:

- assoluto o semplice,
- strutturato,
- indicizzato.

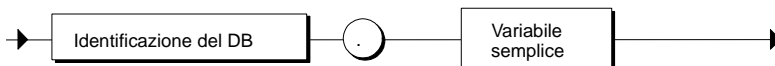
Accesso assoluto al DB



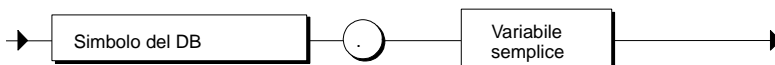
Accesso indicizzato al DB



Accesso strutturato al DB

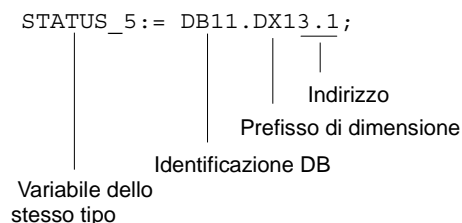


Accesso simbolico al DB



10.3.2 Accesso assoluto ai blocchi dati

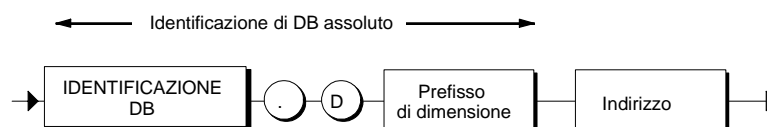
L'accesso assoluto a un blocco dati avviene in modo analogo all'accesso alle aree di memoria della CPU mediante assegnazione di un valore ad una variabile dello stesso tipo. Dopo l'introduzione dell'identificazione DB segue la parola chiave "D" con indicazione del prefisso di lunghezza (p. es. X per bit) e l'indirizzo a byte (p. es. 13.1).



Sintassi

L'accesso viene specificato indicando l'identificazione DB insieme con il prefisso di dimensione e l'indirizzo.

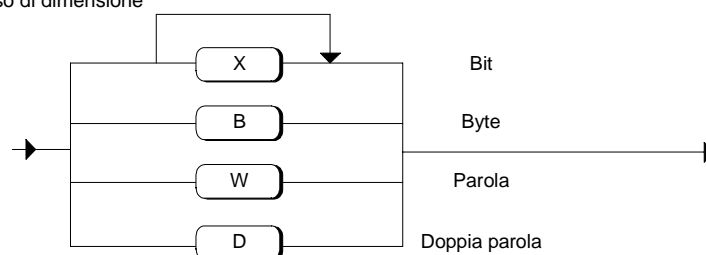
Accesso assoluto al DB



Prefisso di lunghezza

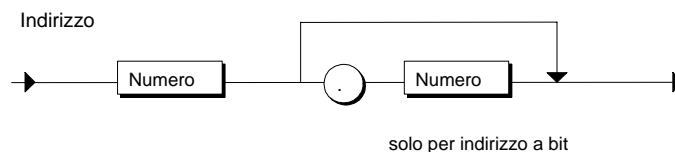
Il prefisso di lunghezza indica la lunghezza dell'area di memoria nel blocco dati che si desidera indirizzare. Si può p. es. leggere un byte o una parola dal DB. Se si desidera specificare solo un bit, l'indicazione del prefisso di lunghezza è opzionale.

Prefisso di dimensione



Indirizzo

L'introduzione dell'indirizzo avviene indicando dapprima l'indirizzo a byte assoluto e quindi, separati da un punto, l'indirizzo a bit (solo nel caso di accesso a bit) del byte interessato.



Esempio

Seguono alcuni esempi di accesso assoluto ad un blocco dati. Nella prima parte il blocco dati viene indicato in modo assoluto, nella seconda parte in modo simbolico:

```

STATUSBYTE      :=DB101.DB10;
STATUS_3        :=DB30.D1.1;
VALOREMISURA   :=DB25.DW20;

STATUSBYTE      :=datidistato.DB10;
STATUS_3        :="Nuovi dati".D1.1;
VALOREMISURA   :=datidimisura.DW20.DW20;

STATUS_1        :=WORD_TO_BLOCK_DB (INDEX).DW10;

```

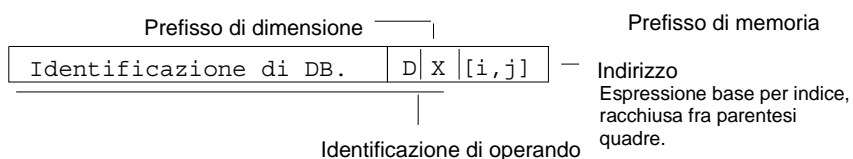
10.3.3 Accesso indicizzato ai blocchi dati

Esiste anche la possibilità di accedere con metodo indicizzato. Rispetto all'indirizzamento simbolico questo metodo offre il vantaggio di consentire l'indirizzamento degli operandi il cui indirizzo diventa stabile solo durante l'esecuzione. Per es. come indirizzo si può utilizzare la variabile di esecuzione di un loop FOR.

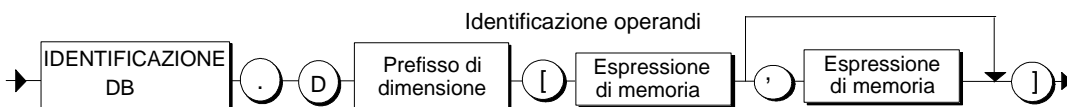
L'accesso indicizzato ad un blocco dati avviene in modo analogo all'accesso assoluto. L'unica differenza consiste nell'indicazione dell'indirizzo.

Al posto dell'indirizzo viene specificato un indice che può essere una costante, una variabile o un'espressione aritmetica.

Nell'accesso indicizzato, l'identificatore assoluto si compone dell'identificazione di operando (parola chiave "D" e prefisso di lunghezza) nonché di un'espressione base per l'indicizzazione.



Sintassi



L'indicizzazione deve essere conforme alle regole seguenti:

- Per un accesso con tipo di dati BYTE, WORD o DWORD si deve utilizzare esattamente un indice. L'indice viene interpretato come indirizzo a byte. La larghezza di accesso viene definita tramite il prefisso di lunghezza.
- In caso di accesso con tipo di dati BOOL si devono utilizzare due indici. Il primo indice specifica l'indirizzo a byte, mentre il secondo indice specifica la posizione del bit all'interno del byte.
- Ogni indice deve essere un'espressione aritmetica del tipo di dati INT (0 - 32767)

Esempio

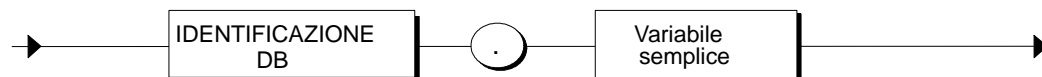
```
STATUS_1 := DB11.DW[CONTATORE];
STATUS_2 := DB12.DX[WNR, BITNR];
STATUS_1 := database1.DW[CONTATORE];
STATUS_2 := database2.DX[WNR, BITNR];
STATUS_1 := WORD_TO_BLOCK_DB(INDEX).DW[CONTATORE];
```

10.3.4 Accesso strutturato ai blocchi dati

L'accesso strutturato avviene tramite l'identificatore della variabile definita nel blocco dati. Si può assegnare la variabile ad ogni variabile dello stesso tipo.

Per fare riferimento alla variabile nel blocco dati si deve indicare il nome del DB e, separato da un punto, il nome della variabile semplice.

Sintassi



Per variabile semplice qui si intende una variabile alla quale nella convenzione DB è stato assegnato un tipo di dati semplice o composto.

Se un parametro di tipo BLOCK_DB o il risultato di una funzione di conversione WORD_TO_BLOCK_DB vengono utilizzati come introduzione per l'accesso ad un blocco dati, non è possibile accedere in modo strutturato, ma solo in modo assoluto o indicizzato.

Esempio

```

Nella parte convenzioni dell'FB10:
VAR
Risultato:    STRUCT RIS1 : INT;
RIS2 : WORD;
END_STRUCT
END_VAR
  
```

```

Tipo di dati definito dall'utente UDT1
TYPE UDT1    STRUCT RIS1 : INT;
RIS2 : WORD;
END_STRUCT
  
```

```

DB20 con tipo di dati definito dall'utente:
DB20
UDT1
BEGIN ...
  
```

```

DB30 senza tipo di dati definito dall'utente:
DB30    STRUCT RIS1 : INT;
RIS2 : WORD;
END_STRUCT
BEGIN ...
  
```

```

Blocco funzionale con gli accessi:
..
FB10.DB10();
PAROLARIS_A  := DB10.risultato.RIS2;
PAROLARIS_B  := DB20.RIS2;
PAROLARIS_C  := DB30.RIS2;
  
```


11 Espressioni, operazioni e operandi

11.1 Espressioni, operazioni e operandi

Un'espressione indica un valore che viene calcolato in fase di compilazione o durante l'esecuzione del programma e si compone di operandi (p. es. costanti, variabili o richiami di funzioni) ed operazioni (p. es. *, /, + oppure -).

I tipi di dati degli operandi e le operazioni impiegate determinano il tipo dell'espressione. S7-SCL distingue tra:

- espressioni aritmetiche
- espressioni di confronto
- espressioni logiche

L'analisi dell'espressione avviene con una sequenza ben determinata. Tale sequenza viene definita:

- dalla priorità delle operazioni interessate e
- dalla sequenza sinistra-destra o
- da operazioni della stessa priorità dalla parentesi usata.

Il risultato di un'espressione può essere:

- assegnato ad una variabile.
- utilizzato come condizione per una istruzione di controllo.
- utilizzato come parametro per il richiamo di una funzione o di un blocco funzionale.

11.2 Operazioni

Le espressioni sono costituite da operazioni e operandi. La maggior parte delle operazioni di S7-SCL sono una combinazione di due operandi e vengono perciò denominati *binarie*. Le altre operazioni lavorano con un solo operando e vengono perciò denominati *unarie*.

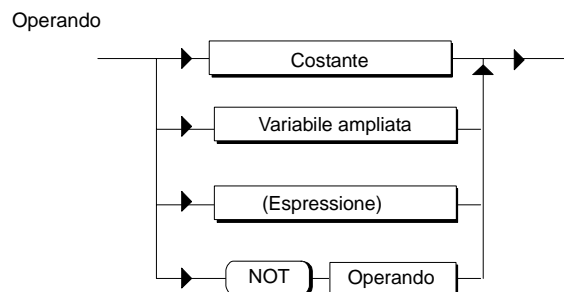
Le operazioni binarie vengono scritte fra gli operandi (ad es. A + B). B. A + B). Le operazioni unarie si trovano sempre subito prima il relativo operando (ad es. -B).

La priorità delle operazioni descritta nella tabella seguente regola la sequenza dei calcoli. La cifra "1" corrisponde alla massima priorità.

Classe	Operazione	Rappresentazione	Priorità
Operazione di assegnazione:	Assegnazione	: =	11
Operazioni aritmetiche:	Potenza	**	2
	Operazioni unarie		
	Più unario	+	3
	Meno unario	-	3
	Operazioni aritmetiche di base		
	Moltiplicazione	*	4
	Divisione	/	4
	Funzione modulo	MOD	4
	Divisione numeri interi	DIV	4
	Addizione	+	5
	Sottrazione	-	5
Operazioni di confronto:	Minore di	<	6
	Maggiore di	>	6
	Minore o uguale	<=	6
	Maggiore o uguale	>=	6
	Uguaglianza	=	7
	Disuguaglianza	<>	7
Operazioni logiche:	Negazione	NOT	3
	Operazioni logiche di base		
	And	AND o &	8
	Or esclusivo	XOR	9
	OR	OR	10
Caratteri di parentesi:	Parentesi	()	1

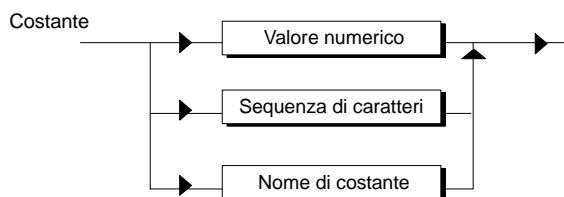
11.3 Operandi

Gli operandi sono oggetti con i quali si può generare un'espressione. Per crearli si possono utilizzare i seguenti elementi:



Costanti

Si possono utilizzare costanti con il loro valore numerico, con un nome simbolico o con una stringa di caratteri.



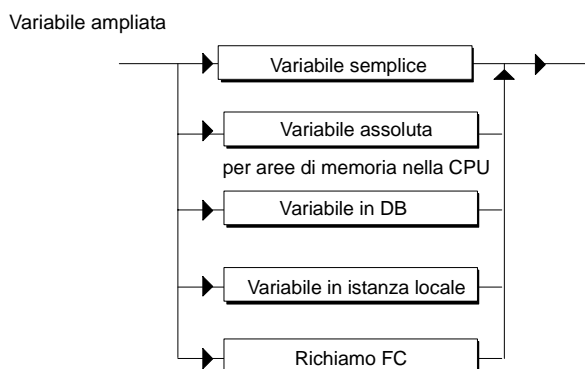
Seguono alcuni esempi di costanti valide:

```

4_711
4711
30.0
FATTORE
'CARATTERE'
  
```

Variabile ampliata

La variabile ampliata è un termine generico per indicare una serie di variabili che vengono descritte più dettagliatamente nel capitolo "Assegnazione di valori".



Seguono alcuni esempi di variabili valide:

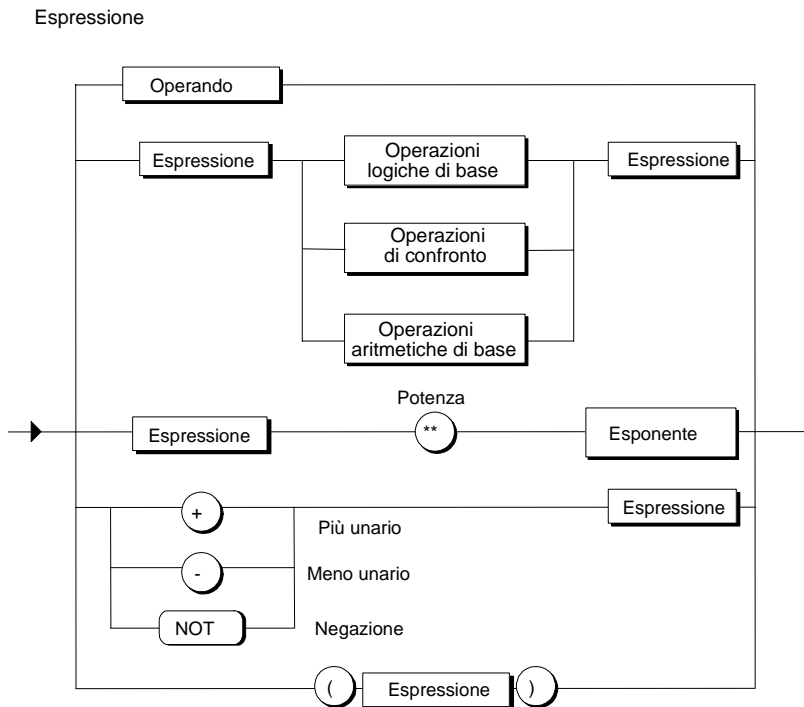
VALNOM	variabile semplice
EW10	variabile assoluta
E100.5	variabile assoluta
DB100.DW [INDICE]	variabile nel DB
GIRI.MOTORE	variabile nell'istanza locale
SQR (20)	funzione standard
FC192 (VALNOM)	richiamo della funzione

Nota

In un richiamo della funzione, il risultato calcolato, ossia il valore di ritorno, viene inserito nell'espressione al posto del nome della funzione. Per tale motivo, le funzioni VOID che non possiedono alcun valore di ritorno non sono ammesse come operandi di un'espressione.

11.4 Sintassi di un'espressione

Sintassi



Risultato di un'espressione

Il risultato di un'espressione può essere:

- assegnato ad una variabile
- utilizzato come condizione per una istruzione di controllo
- utilizzato come parametro per il richiamo di una funzione o di un blocco funzionale.

Sequenza di analisi

La sequenza di analisi di un'espressione dipende:

- dalla priorità delle operazioni interessate e
- dalla sequenza sinistra-destra o
- dalla parentesi usata (per operazioni della stessa priorità).

Regole

L'elaborazione delle espressioni avviene conformemente alle seguenti regole:

- Un operando fra due operazioni di diversa priorità è sempre legato a quella di gerarchia superiore.
- Le operazioni vengono elaborate conformemente alla loro sequenza gerarchica.
- Le operazioni della stessa priorità vengono elaborate da sinistra verso destra.
- L'inserimento del segno meno davanti a un identificatore equivale ad una moltiplicazione per -1.
- Le operazioni aritmetiche non devono essere disposte in successione. Perciò, l'espressione $a * - b$ non è valida, mentre invece $a * (-b)$ è consentita.
- L'inserimento di coppie di parentesi può disattivare la priorità delle operazioni, cioè le parentesi hanno la massima priorità.
- Le espressioni fra parentesi vengono considerate come un operando singolo e vengono analizzate sempre per prime.
- Il numero di parentesi sinistre deve corrispondere al numero di parentesi destre.
- Le operazioni aritmetiche non possono essere utilizzate assieme a caratteri o dati logici. Perciò, le espressioni come 'A' + 'B' e $(n \leq 0) + (m > 0)$ sono errate.

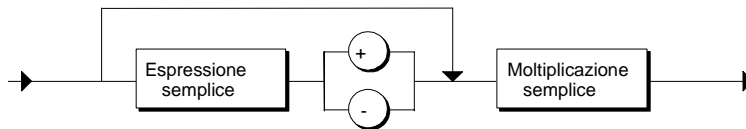
Esempi di espressioni

EB10	// operando
A1 AND (A2)	// espressione logica
(A3) < (A4)	// espressione di confronto
3+3*4/2	// espressione aritmetica

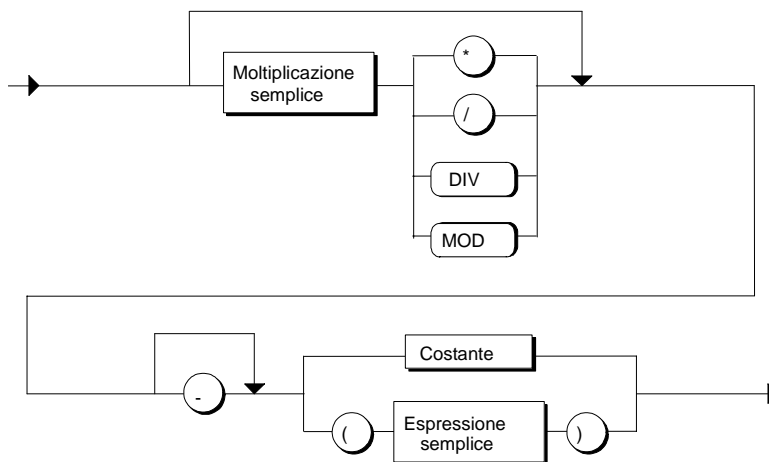
11.5 Espressione semplice

Per espressione semplice in S7-SCL si intendono le espressioni aritmetiche semplici. Si ha la possibilità di moltiplicare o dividere a coppie i valori costanti e combinare queste coppie mediante addizione o sottrazione.

Sintassi di un'espressione semplice:



Sintassi di una moltiplicazione semplice:



Esempio:

```
ESPRESSIONE_SEMPL= A * B + D / C - 3 * VALORE1;
```

11.6 Espressioni aritmetiche

Un'espressione aritmetica è un'espressione formata con operazioni aritmetiche. Queste espressioni consentono l'elaborazione di tipi di dati numerici.

La tabella seguente contiene un riepilogo delle operazioni possibili e mostra a quale tipo si deve assegnare il risultato in funzione degli operandi. Vengono utilizzate le seguenti abbreviazioni:

ANY_INT	per i tipi di dati	INT, DINT
ANY_NUM	per i tipi di dati	ANY_INT e REAL

Operazione	Operatore	1° operando	2° operando	Risultato	Priorità
Potenza	**	ANY_NUM	ANY_NUM	REAL	2
Più unario	+	ANY_NUM	-	ANY_NUM	3
		TIME	-	TIME	3
Meno unario	-	ANY_NUM	-	ANY_NUM	3
		TIME	-	TIME	3
Moltiplicazione	*	ANY_NUM	ANY_NUM	ANY_NUM	4
		TIME	ANY_INT	TIME	4
Divisione	/	ANY_NUM	ANY_NUM	ANY_NUM	4
		TIME	ANY_INT	TIME	4
Divisione di interi	DIV	ANY_INT	ANY_INT	ANY_INT	4
		TIME	ANY_INT	TIME	4
Divisione modulo	MOD	ANY_INT	ANY_INT	ANY_INT	4
Addizione	+	ANY_NUM	ANY_NUM	ANY_NUM	5
		TIME	TIME	TIME	5
		TOD	TIME	TOD	5
		DT	TIME	DT	5
Sottrazione	-	ANY_NUM	ANY_NUM	ANY_NUM	5
		TIME	TIME	TIME	5
		TOD	TIME	TOD	5
		DATE	DATE	TIME	5
		TOD	TOD	TIME	5
		DT	TIME	DT	5
		DT	DT	TIME	5

Nota

Il tipo di dati del risultato di una divisione (/) è determinato dal tipo di dati dell'operando avente valore massimo.

Se per es. due operandi sono divisi dal tipo di dati INT il risultato deriva anche dal tipo di dati INT (vale a dire $10/3=3$ mentre $10.0/3=3.33$).

Regole

Le operazioni delle espressioni aritmetiche vengono trattate in base alla loro priorità.

- Per una maggiore chiarezza, si raccomanda di racchiudere fra parentesi i numeri negativi anche nei casi in cui ciò non è necessario.
- Nelle divisioni con due operandi interi del tipo di dati INT, le operazioni "DIV" e "/" forniscono lo stesso risultato (vedere esempio seguente).
- Le operazioni di divisione ("/", MOD e "DIV") richiedono che il secondo operando sia diverso da zero.
- Se un operando è del tipo INT (numero intero) e l'altro operando è del tipo REAL (numero in virgola mobile), il risultato sarà sempre di tipo REAL.
- Sottraendo i dati di tipo DATE_AND_TIME e TIME l'operando che contempla il tipo di dati TIME deve collocarsi sempre a destra dell'operatore "-".

Esempi

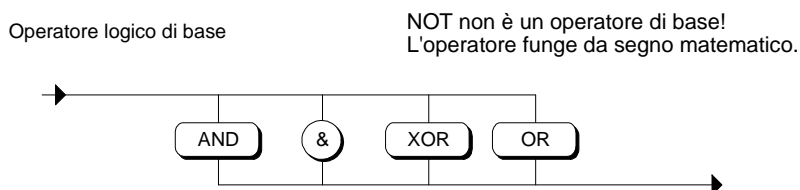
```
// Il risultato (11) dell'espressione aritmetica viene
// assegnato alla variabile "VALORE"
VALORE1 := 3 + 3 * 4 / 2 - (7+3) / (-5) ;
// Il VALORE dell'espressione successiva è 1
VALORE2 := 9 MOD 2 ;
```

11.7 Espressioni logiche

Un'espressione logica è un'espressione generata con operazioni logiche.

Operazioni logiche di base

Con le operazioni AND, &, XOR e OR si possono combinare operandi logici (tipo BOOL) o variabili del tipo di dati BYTE, WORD o DWORD per poter generare espressioni logiche. Per la negazione di un operando logico viene utilizzata l'operazione NOT.



Operazioni logiche

Il risultato dell'espressione è TRUE o FALSE in caso di combinazione logica di operandi booleani oppure una stringa di bit in caso di combinazione logica di bit fra i due operandi.

La tabella seguente contiene informazioni sulle espressioni logiche disponibili e sui tipi di dati per il risultato e gli operandi. Vengono utilizzate le seguenti abbreviazioni:

ANY_BIT	per i tipi di dati	BOOL, BYTE, WORD, DWORD
---------	--------------------	-------------------------

Operazione	Operando	1° operando	2° operando	risultato	Priorità
Negazione	NOT	ANY_BIT	-	ANY_BIT	3
Congiunzione	AND	ANY_BIT	ANY_BIT	ANY_BIT	8
Disgiunzione esclusiva	XOR	ANY_BIT	ANY_BIT	ANY_BIT	9
Disgiunzione	OR	ANY_BIT	ANY_BIT	ANY_BIT	10

Risultato

Il risultato di un'espressione logica può essere

- 1 (*true*) o 0 (*false*) in una combinazione di operandi booleani
- oppure una stringa di bit dopo un'operazione logica combinatoria a bit fra i due operandi.

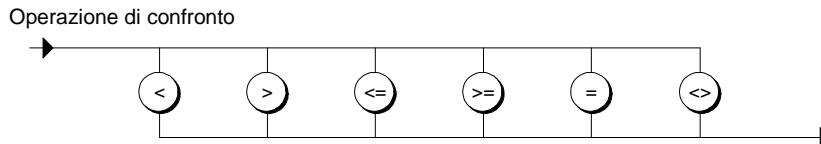
Esempi

```
// Il risultato dell'espressione di confronto viene negato.
    IF NOT (CONTATORE > 5) THEN . . . ;
// Il risultato della prima espressione di confronto viene negato e
// coniugato con il risultato della seconda
    A := NOT (CONTATORE1 = 4) AND (CONTATORE2 = 10) ;
// Disgiunzione di due espressioni di confronto
    WHILE (A >= 9) OR (INTERROGAZIONE <> "n") DO.... ;
// Mascheramento di un byte di ingresso (combinazione di bit)
    Risultato := EB10 AND 2#11110000 ;
```

11.8 Espressioni di confronto

Le operazioni di confronto confrontano i valori di due operandi e forniscono un valore booleano. Il risultato è TRUE se il confronto è positivo e FALSE se il confronto è negativo.

Sintassi



Regole

La seguente tabella mostra i tipi di dati paragonabili e le regole valide per il confronto:

Tipo di dati	=	<>	>0	<0	>	<	Regole
INT	✓	✓	✓	✓	✓	✓	All'interno della classe dei tipi di dati numerici sono ammessi tutti gli operatori di confronto. L'operatore avente tipo più forte determina il tipo di operazione.
DINT	✓	✓	✓	✓	✓	✓	
REAL	✓	✓	✓	✓	✓	✓	
BOOL	✓	✓					All'interno della classe dei tipi di dati di bit sono ammessi solo UGUALE A e DIVERSO DA come operatori di confronto. L'operatore avente tipo più forte determina il tipo di operazione.
BYTE	✓	✓					
WORD	✓	✓					
DWORD	✓	✓					
CHAR	✓	✓	✓	✓	✓	✓	Per i caratteri e le relative stringhe si portano a confronto la lunghezza delle variabili e il valore numerico di ogni carattere ASCII. Il confronto di STRING viene eseguito internamente mediante le funzioni EQ_STRNG, GE_STRNG, LE_STRNG, GT_STRNG e LT_STRNG della biblioteca IEC. I seguenti operandi non sono ammessi per queste funzioni: <ul style="list-style-type: none"> • Parametri di un FC. • Parametri IN_OUT di un FB di tipo STRUCT o ARRAY.
STRING	✓	✓	✓	✓	✓	✓	

Tipo di dati	=	<>	>0	<0	>	<	Regole
DATE	✓	✓	✓	✓	✓	✓	
TIME	✓	✓	✓	✓	✓	✓	
DT	✓	✓	✓	✓	✓	✓	<p>Il confronto di DT viene eseguito internamente mediante le funzioni EQ_DT, GE_DT, LE_DT, GT_STRNG e LT_DT della biblioteca IEC.</p> <p>I seguenti operandi non sono ammessi per queste funzioni:</p> <ul style="list-style-type: none"> • Parametri di un FC. • Parametri IN_OUT di un FB di tipo STRUCT o ARRAY.
TOD	✓	✓	✓	✓	✓	✓	
S5-TIME							Le variabili S5 TIME non sono consentite come operandi di confronto. È necessario convertire esplicitamente da S5TIME in TIME mediante le funzioni IEC.

Combinazione di confronti

Le espressioni di confronto possono essere combinate in base alla logica booleana per poter realizzare interrogazioni del tipo "se $a < b$ e $b < c$ allora ...".

Le singole operazioni vengono eseguite in base alla loro priorità che può essere modificata con l'uso delle parentesi.

Esempi

```
// Confronto di 3 INFERIORE UGUALE A 4. Il risultato
// è "TRUE"
      A := 3 <= 4
// Confronto di 7 DIVERSO DA 7. Il risultato
// è "FALSE"
      7 <> 7
// Valutazione di un'espressione di confronto in
// una diramazione IF
      IF CONTATORE < 5 THEN ....
// Combinazione di due espressioni di confronto
Valore_A > 20 AND Valore_B < 20
// Combinazione di due espressioni di confronto con parentesi
A<>(B AND C)
```

12 Istruzioni

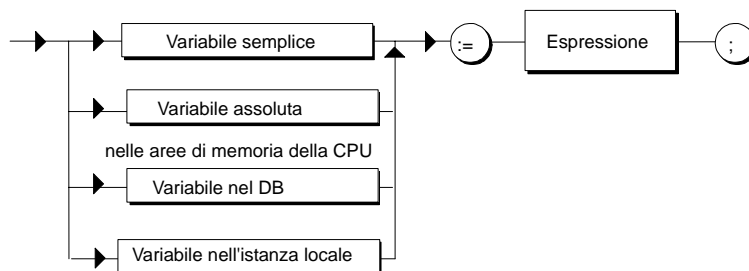
12.1 Assegnazione di valori

Le assegnazioni di valori sostituiscono il valore momentaneo di una variabile con un nuovo valore che viene introdotto tramite un'espressione. Questa espressione può contenere anche identificatori di funzioni, che vengono così attivate e ritornano valori corrispondenti (valore di ritorno).

Come mostra la figura sottostante, l'espressione viene memorizzata sul lato destro dell'assegnazione di valore e il risultato nella variabile il cui nome è situato sul lato sinistro del segno di assegnazione. La figura illustra le variabili consentite.

Sintassi delle assegnazioni di valori:

Assegnazione di valore



Il tipo di un'espressione di assegnazione è il tipo dell'operando sinistro. La variabile semplice può essere una variabile del tipo di dati semplice o strutturato.

12.1.1 Assegnazione di valori con variabili di tipo di dati semplici

Ogni espressione e ogni variabile con tipo di dati semplici possono essere assegnate ad un'altra variabile dello stesso tipo.

```
Identificatore := espressione;  
Identificatore := variabile ;
```

Esempio

```
FUNCTION_BLOCK FB12  
VAR  
    INTERRUETTORE_1      : INT ;  
    INTERRUETTORE_2      : INT ;  
    VALNOM_1             : REAL ;  
    VALNOM_2             : REAL ;  
    INTERROG_1           : BOOL ;  
    ORA_1                : S5TIME ;  
    ORA_2                : TIME ;  
    DATA_1              : DATE ;  
    ORAGIORNO_1          : TIME_OF_DAY ;  
END_VAR  
BEGIN  
  
    // Assegnazione di una costante ad una variabile  
    INTERRUETTORE_1      := -17 ;  
    VALNOM_1             := 100.1 ;  
    INTERROG_1           := TRUE ;  
    ORA_1                := T#1H_20M_10S_30MS ;  
    ORA_2                := T#2D_1H_20M_10S_30MS ;  
    DATA_1              := D#1996-01-10 ;  
  
    // Assegnazione di una variabile ad una variabile  
    VALNOM_1 := VALNOM_2 ;  
    INTERRUETTORE_2 := INTERRUETTORE_1 ;  
  
    // Assegnazione di un'espressione ad una variabile  
    INTERRUETTORE_2      := INTERRUETTORE_1 * 3 ;  
END_FUNCTION_BLOCK
```

12.1.2 Assegnazione di valori con variabili di tipo STRUCT e UDT

Le variabili di tipo STRUCT e UDT sono variabili strutturate che rappresentano una struttura completa o una componente di questa struttura.

Alcune indicazioni valide per una variabile di struttura potrebbero essere:

```
Immagine           //identificatore per una struttura
Immagine.elemento  //identificatore per le componenti
                  //di una struttura
Immagine.campo     //identificatore di un campo semplice
                  //all'interno di una struttura
Immagine.campo[2,5] //identificatore della componente di un campo
                  //all'interno di una struttura
```

Assegnazione di una struttura completa

Una struttura completa può essere assegnata ad un'altra struttura solo se le componenti della struttura corrispondono sia nel tipo di dati che nel nome. Sono assegnazioni valide:

```
structname_1:=structname_2;
```

Assegnazione di componenti di struttura:

Ad ogni componente di struttura può essere assegnata una variabile di tipo compatibile o un'altra componente di struttura.

Ad una componente di struttura si può fare riferimento indicando l'identificatore della struttura e l'identificatore della componente di struttura separati da un punto. Sono assegnazioni valide:

```
structname_1.element1      := valore ;
structname_1.element1      := 20.0 ;
structname_1.element1      := structname_2.element1 ;
structname_1.nomedicampo1   := structname_2.nomedicampo2 ;
structname_1.nomedicampo[10] := 100 ;
```

Esempio

```
FUNCTION_BLOCK FB3
VAR
  VARAUSIL : REAL ;
  VALOREMISURA : STRUCT //Struttura di destinazione
      TENSIONE:REAL ;
      RESISTENZA:REAL ;
      CAMPOSEMPLICE : ARRAY [1..2, 1..2]
      OF INT ;
      END_STRUCT ;
  VALEFFET : STRUCT //Struttura sorgente
      TENSIONE : REAL ;
      RESISTENZA:REAL ;
      CAMPOSEMPLICE: ARRAY [1..2, 1..2]
      OF INT ;
      END_STRUCT ;
END_VAR

BEGIN
  //Assegnazione di una struttura completa ad una struttura completa
  VALOREMISURA := VALEFFET ;
  //Assegnazione del componente di una struttura alla componente
  //di una struttura
  VALOREMISURA.TENSIONE := VALEFFET.TENSIONE ;
  //Assegnazione del componente di una struttura ad una variabile
  //dello stesso tipo
  VARAUSIL := VALEFFET.RESISTENZA;
  //Assegnazione di una costante alla componente di una struttura
  VALOREMISURA.RESISTENZA:= 4.5;
  //Assegnazione di una costante ad un elemento di campo semplice
  VALOREMISURA.CAMPOSEMPLICE[1,2] := 4;
END_FUNCTION_BLOCK
```


12.1.3 Assegnazione di valori con variabili di tipo ARRAY

Un campo si compone di 1 fino a max. 6 dimensioni e contiene elementi di campo tutti dello stesso tipo. Per l'assegnazione di campi ad una variabile esistono due varianti di accesso: Si può fare riferimento a campi completi oppure a campi parziali.

Assegnazione di un campo completo

Un campo completo può essere assegnato ad un altro campo se corrispondono sia il tipo di dati delle componenti che i limiti di campo (indici di campo più piccoli e più grandi possibili). In caso positivo, dopo il segno di assegnazione si deve indicare l'identificatore del campo. Sono assegnazioni valide:

```
nomedicampo_1:=nomedicampo_2;
```

Nota

Non deve essere assegnata alcuna costante a campi completi.

Assegnazione di una componente di campo

Un singolo elemento di campo viene indirizzato con il nome del campo seguito da valori indice idonei racchiusi fra parentesi quadre. Per ogni dimensione è disponibile un indice. Essi vengono separati mediante virgole e sono racchiusi fra parentesi quadre. Un indice deve essere un'espressione aritmetica con tipo di dati INT.

Per realizzare un'assegnazione di valori per un campo parziale, nelle parentesi quadre si devono omettere indici dietro il nome del campo a partire da destra. In tal modo si indirizza un'area parziale il cui numero di dimensione equivale al numero di indici omessi. Sono assegnazioni valide:

```
nomedicampo_1[ i ] := nomedicampo_2[ j ] ;  
nomedicampo_1[ i ] := espressione;  
identificatore_1 := nomedicampo_1[ i ] ;
```

Esempio

```
FUNCTION_BLOCK FB3
VAR
  VALNOMINALI      :ARRAY [0..127] OF INT ;
  VALEFFETTIVI     :ARRAY [0..127] OF INT ;
  // Convenzione per una matrice(=campo bidimensionale)
  // di 3 righe e 4 colonne
  REGOLATORE : ARRAY [1..3, 1..4] OF INT ;
  // Convenzione per un vettore (=campo a una dimensione)
  // con 4 componenti
  REGOLATORE_1 : ARRAY [1..4] OF INT ;
END_VAR

BEGIN
  // Assegnazione di un campo completo ad un campo
  VALNOMINALI := VALEFFETTIVI ;
  // Assegnazione di un vettore alla seconda riga del campo REGOLATORE
  REGOLATORE[2] := REGOLATORE_1 ;
  // Assegnazione di una componente del campo ad una componente
  // del campo REGOLATORE
  REGOLATORE [1,4] := REGOLATORE_1 [4] ;
END_FUNCTION_BLOCK
```

12.1.4 Assegnazione di valori con variabili di tipo STRING

Una variabile con tipo di dati STRING contiene una stringa di caratteri contenente max. 254 caratteri. Ad ogni variabile con tipo di dati STRING si può assegnare un'altra variabile dello stesso tipo. Sono assegnazioni valide:

```
variabilestringa_1 := costante String;  
variabilestringa_1 := variabilestringa_2 ;
```

Esempio

```
FUNCTION_BLOCK FB3  
VAR  
    VISUALIZ_1    : STRING[50] ;  
    STRUTTURA1   : STRUCT  
        VISUALIZ_2 : STRING[100] ;  
        VISUALIZ_3 : STRING[50] ;  
    END_STRUCT ;  
END_VAR  
  
BEGIN  
    // Assegnazione di una costante ad una variabile STRING  
    VISUALIZ_1 := 'Errore nel modulo 1' ;  
    // Assegnazione di una componente della struttura ad una  
    // variabile STRING.  
    VISUALIZ_1 := STRUTTURA1.VISUALIZ_3 ;  
    // Assegnazione di una variabile STRING ad una variabile STRING  
    If VISUALIZ_1 <> STRUTTURA1.VISUALIZ_3 THEN  
        VISUALIZ_1 := STRUTTURA1.VISUALIZ_3 ;  
    END_IF;  
END_FUNCTION_BLOCK
```

12.1.5 Assegnazione di valori con variabili di tipo DATE_AND_TIME

Il tipo di dati DATE_AND_TIME definisce un'area di 64 bit (8 byte) per l'indicazione della data e dell'ora. Ad ogni variabile con tipo di dati DATE_AND_TIME si può assegnare un'altra variabile o costante dello stesso tipo. Sono assegnazioni valide:

```
dtvariabile_1 := costante di data e ora;  
dtvariabile_1 := dtvariabile_2 ;
```

Esempio

```
FUNCTION_BLOCK FB3  
VAR  
    DATA_ORA_1      : DATE_AND_TIME ;  
    STRUTTURA1      : STRUCT  
        DATA_ORA_2 : DATE_AND_TIME ;  
        DATA_ORA_3 : DATE_AND_TIME ;  
    END_STRUCT ;  
END_VAR  
  
BEGIN  
    // Assegnazione di una costante ad una variabile DATE_AND_TIME  
    DATA_ORA_1 := DATE_AND_TIME#1995-01-01-12:12:12.2 ;  
    STRUTTURA1.TEMPO_3 := DT#1995-02-02-11:11:11 ;  
    // Assegnazione di una componente di struttura ad una  
    // variabile DATE_AND_TIME  
    DATA_ORA_1 := STRUTTURA1.DATA_ORA_2 ;  
    // Assegnazione di una variabile DATE_AND_TIME ad una  
    // variabile DATE_AND_TIME  
    If DATA_ORA_1 < STRUTTURA1.DATA_ORA_3 THEN  
        DATA_ORA_1 := STRUTTURA1.DATA_ORA_3 ;  
    END_IF ;  
END_FUNCTION_BLOCK
```

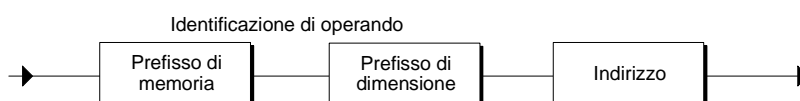
12.1.6 Assegnazione di valori con variabili assolute per aree di memoria

La variabile assoluta fa riferimento alle valide aree di memoria globali della CPU. Per indirizzare queste aree esistono tre possibilità:

- Accesso assoluto
- Accesso indicizzato
- Accesso simbolico

Sintassi delle variabili assolute

Variabili assolute



Esempio

```

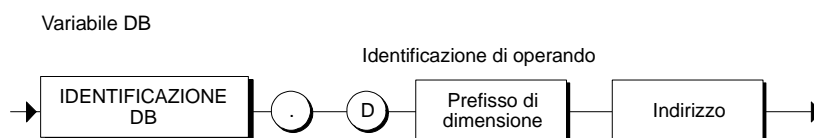
FUNCTION_BLOCK FB3
VAR
    PAROLADISTATO1      : WORD ;
    PAROLADISTATO2      : BOOL  ;
    PAROLADISTATO3      : BYTE  ;
    PAROLADISTATO4      : BOOL  ;
    INDIRIZZO           : INT   ;
END_VAR
BEGIN
    INDIRIZZO := 10 ;
    // Assegnazione di una parola di ingresso ad una variabile
    // (accesso semplice)
    PAROLADISTATO1 := EW4 ;
    // Assegnazione di una variabile ad un bit di uscita
    // (accesso semplice)
    a1.1 := PAROLADISTATO2 ;
    // Assegnazione di un byte di ingresso ad una variabile
    // (accesso indicizzato)
    PAROLADISTATO3 := EB[INDIRIZZO] ;
    // Assegnazione di un bit di ingresso ad una variabile
    // (accesso indicizzato)
    FOR INDIRIZZO := 0 TO 7 BY 1 DO
        PAROLADISTATO4 := e[1, INDIRIZZO] ;
    END_FOR ;
END_FUNCTION_BLOCK
  
```

12.1.7 Assegnazione di valori con variabili globali

Anche l'accesso ai dati nei blocchi dati globali avviene tramite un'assegnazione di valori a variabili dello stesso tipo o viceversa. Ad ogni variabile globale si può assegnare una variabile o un'espressione dello stesso tipo. Per indirizzare questi dati esistono tre possibilità:

- Accesso strutturato
- Accesso assoluto
- Accesso indicizzato

Sintassi di una variabile di DB



Esempio

```

FUNCTION_BLOCK FB3
VAR
    REGOLATORE_1      : ARRAY [1..4] OF INT ;
    PAROLADISTATO1     : WORD ;
    PAROLADISTATO2     : ARRAY [0..10] OF WORD ;
    PAROLADISTATO3     : INT ;
    PAROLADISTATO4     : WORD ;
    INDIRIZZO          : INT ;
END_VAR
VAR_INPUT
    PAROLAINDIR        : WORD ;
END_VAR
BEGIN
    // Assegnazione della parola 1 del DB11
    // ad una variabile (accesso semplice)
    PAROLADISTATO1 := DB11.DW1 ;
    // Alla componente di campo nella riga 1 e
    // nella colonna 1 della matrice viene assegnato il valore
    // della variabile "NUMERO" (accesso strutturato):
    REGOLATORE_1[1] := DB11.NUMERO ;
    // Assegnazione della componente della struttura "NUMERO2"
    // 1" alla variabile paroladistato3
    PAROLADISTATO3 := DB11.NUMERO1.NUMERO2 ;
    // Assegnazione di una parola con indice INDIRIZZO
    // da DB11 ad una variabile (accesso indicizzato)
    FOR
        INDIRIZZO := 1 TO 10 BY 1 DO
            PAROLADISTATO2[INDIRIZZO] := DB11.DW[INDIRIZZO] ;
            // Qui il parametro di ingresso PAROLAINDIR
            // viene utilizzato come numero del DB e l'indice
            // INDIRIZZO viene utilizzato per indicare
            // l'indirizzo a parola all'interno del DB.
            PAROLADISTATO4 :=
WORD_TO_BLOCK_DB(PAROLAINDIR).DW[INDIRIZZO] ;
        END_FOR ;
    END_FUNCTION_BLOCK

```

12.2 Istruzioni di controllo

12.2.1 Istruzioni di controllo

Istruzioni di selezione

Le istruzioni di selezione consentono di far proseguire il flusso del programma in diverse sequenze di istruzioni in funzione di determinate condizioni.

Tipo di diramazione	Funzione
Istruzione IF	Con l'istruzione IF si può far proseguire l'esecuzione del programma da una diramazione scelta fra due alternative in funzione di un determinata condizione che può essere TRUE o FALSE.
Istruzione CASE	Con l'istruzione CASE si può far proseguire l'esecuzione del programma in base ad una diramazione di tipo 1:n, assegnando ad una variabile un determinato valore scelto fra n valori possibili.

Elaborazione di loop

L'elaborazione di loop può essere controllata con l'ausilio di istruzioni di ripetizione. Un'istruzione di ripetizione indica quali parti di un programma devono essere ripetute in funzione di determinate condizioni.

Tipo di diramazione	Funzione
Istruzione FOR	Serve per ripetere una sequenza di istruzioni finché la variabile d'esecuzione rimane all'interno del campo di valori indicato.
Istruzione WHILE	Serve per ripetere una sequenza di istruzioni finché è soddisfatta una determinata condizione d'esecuzione.
Istruzione REPEAT	Serve per ripetere una sequenza di istruzioni finché non venga soddisfatta una determinata condizione d'interruzione.

Salto di programma

Un salto di programma causa il salto immediato a un determinato obiettivo di salto e quindi ad un'istruzione nell'ambito dello stesso blocco.

Tipo di diramazione	Funzione
Istruzione CONTINUE	Serve per interrompere la momentanea esecuzione del loop.
Istruzione EXIT	Serve per uscire da un loop in qualsiasi momento e indipendentemente dal fatto che sia soddisfatta o meno una determinata condizione d'interruzione.
Istruzione GOTO	Causa il salto immediato ad un'etichetta di salto indicata.
Istruzione RETURN	Causa l'uscita da un blocco momentaneamente in corso di elaborazione.

12.2.2 Condizioni

L'espressione può essere un'espressione di confronto, un'espressione logica o un'espressione aritmetica. Essa è del tipo BOOL e può assumere i valori TRUE o FALSE. Le espressioni aritmetiche sono TRUE se esse assumono un valore diverso da zero e FALSE se esse assumono un valore uguale a zero. La tabella seguente illustra alcuni esempi di condizioni:

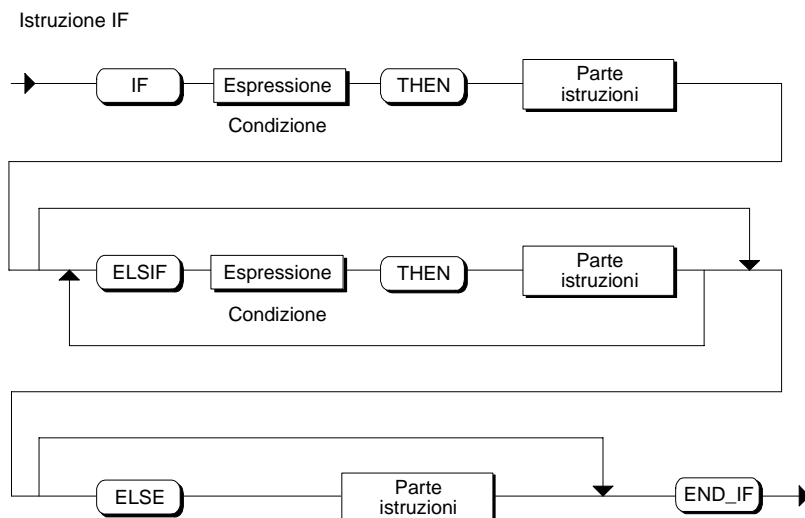
Tipo	Esempio
Espressione di confronto	TEMP > 50 CONTATORE <= 100 CHAR1 < 'S'
Espressione di confronto e logica	(ALPHA <> 12) AND NOT BETA
Operando booleano	E1.1
Espressione aritmetica	ALPHA = (5 + BETA)

12.2.3 Istruzione IF

L'istruzione IF è un'istruzione condizionata. Essa consente una o più opzioni e seleziona una delle sue parti istruzioni (se necessario nessuna) per l'esecuzione.

L'esecuzione delle istruzioni condizionate causa la valutazione dell'espressione logica indicata. Se il valore di un'espressione è TRUE, la condizione viene considerata soddisfatta, se il valore è FALSE la condizione non è soddisfatta.

Sintassi



L'istruzione IF viene elaborata in base alle seguenti regole:

- viene eseguita la prima sequenza di istruzioni la cui espressione logica = TRUE. Le altre sequenze di istruzioni non vengono eseguite.
- Se nessuna espressione booleana = TRUE, viene eseguita la sequenza di istruzioni contenenti ELSE (oppure nessuna sequenza di istruzioni, se non è presente alcun ramo ELSE).
- Vi può essere un numero qualsiasi di istruzioni ELSIF.

Nota

Rispetto ad una sequenza di istruzioni IF, l'impiego di uno o più rami ELSIF presenta il vantaggio che le espressioni logiche che seguono un'espressione valida non vengono più analizzate. Ciò consente di ridurre il tempo d'esecuzione di un programma.

Esempio

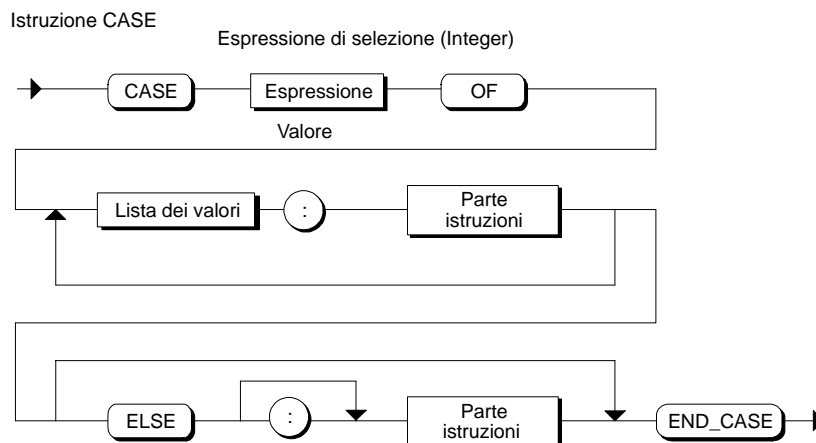
```

IF  E1.1 THEN
    N := 0 ;
    SUM := 0 ;
    OK := FALSE ; // Imposta il flag OK su FALSE
ELSIF START = TRUE THEN
    N := N + 1 ;
    SUM := SUM + N ;
ELSE
    OK := FALSE ;
END_IF ;
  
```

12.2.4 Istruzione CASE

L'istruzione CASE consente di selezionare una di n parti di programma alternative. Questa selezione si basa sul valore corrente di un'espressione di selezione.

Sintassi

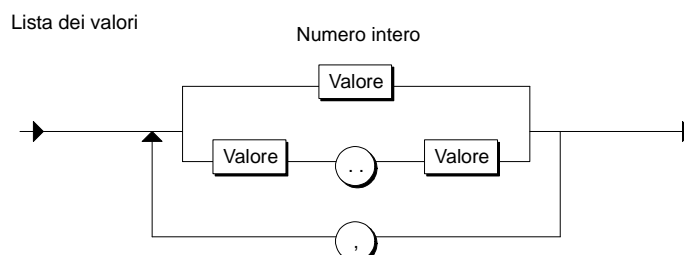


L'istruzione CASE viene elaborata conformemente alle regole seguenti:

- L'espressione di selezione deve fornire un valore del tipo INTEGER.
- Durante l'elaborazione dell'istruzione CASE si verifica se il valore dell'espressione di selezione è contenuto in una determinata lista di valori. Se il confronto ha esito positivo, viene eseguita la parte istruzioni assegnata alla lista.
- Se il confronto non ha esito positivo, viene eseguita la parte istruzioni che segue ELSE oppure nessuna istruzione se il ramo ELSE non è presente.

Lista dei valori

La lista dei valori contiene i valori consentiti per l'espressione di selezione.

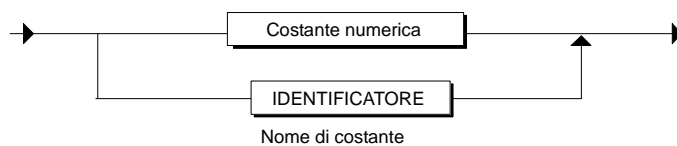


Valgono le regole seguenti:

- Ogni lista di valori inizia con una costante, una lista di costanti o un'area di costanti.
- I valori nell'ambito della lista di valori devono essere del tipo INTEGER.
- Ogni valore può essere presente una sola volta.

Valore

Il valore segue la sintassi sottostante:



Esempio

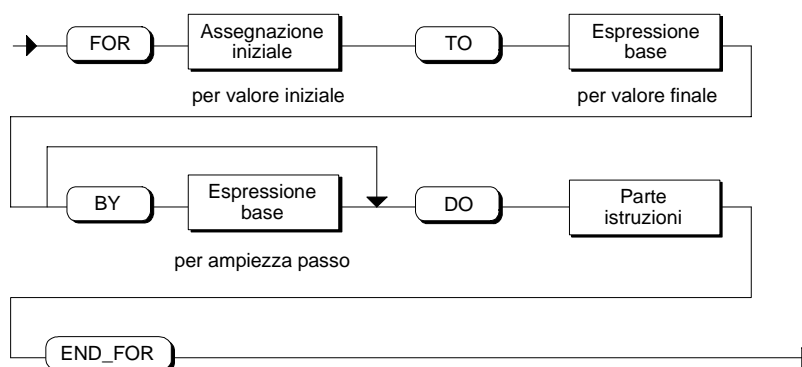
```
CASE TW OF
  1 :      DISPLAY:= OVEN_TEMP;
  2 :      DISPLAY:= MOTOR_SPEED;
  3 :      DISPLAY:= GROSS_TARE;
          AW4:= 16#0003;
  4..10:   DISPLAY:= INT_TO_DINT (TW);
          AW4:= 16#0004;
  11,13,19: DISPLAY:= 99;
          AW4:= 16#0005;
ELSE:
  DISPLAY:= 0;
  TW_ERROR:= 1;
END_CASE ;
```

12.2.5 Istruzione FOR

Un'istruzione FOR serve per ripetere una sequenza di istruzioni finché una variabile d'esecuzione si trova nell'ambito dell'area di valori indicata. La variabile d'esecuzione deve essere l'identificatore di una variabile del tipo INT o DINT. La definizione di un loop con FOR include anche la definizione di un valore iniziale e di un valore finale. Entrambi i valori devono essere dello stesso tipo della variabile d'esecuzione.

Sintassi

Istruzione FOR



L'istruzione FOR viene elaborata conformemente alle regole seguenti:

- Quando viene eseguito il loop, la variabile d'esecuzione viene impostata sul valore iniziale (assegnazione iniziale) e ad ogni passaggio del loop viene incrementata dell'ampiezza di passo indicata (ampiezza di passo positiva) o diminuita (ampiezza di passo negativa) finché non venga raggiunto il valore finale.
- Dopo ogni passaggio si verifica se la condizione (valore finale raggiunto) è soddisfatta. In caso positivo, la sequenza di istruzioni viene eseguita, mentre in caso negativo il loop, e quindi la sequenza di istruzioni, viene saltato.

Regole

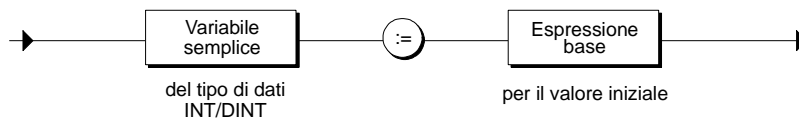
Quando si impiega l'istruzione FOR si devono osservare le regole seguenti:

- La variabile d'esecuzione deve essere solo del tipo di dati INT o DINT.
- L'indicazione di BY [ampiezza di passo] può essere omessa. Se l'ampiezza di passo non viene specificata, essa assume il valore +1.

Assegnazione iniziale

La generazione del valore iniziale delle variabili d'esecuzione deve essere conforme alla sintassi seguente: La variabile semplice che si trova sulla sinistra dell'assegnazione deve essere del tipo di dati INT o DINT.

Assegnazione iniziale

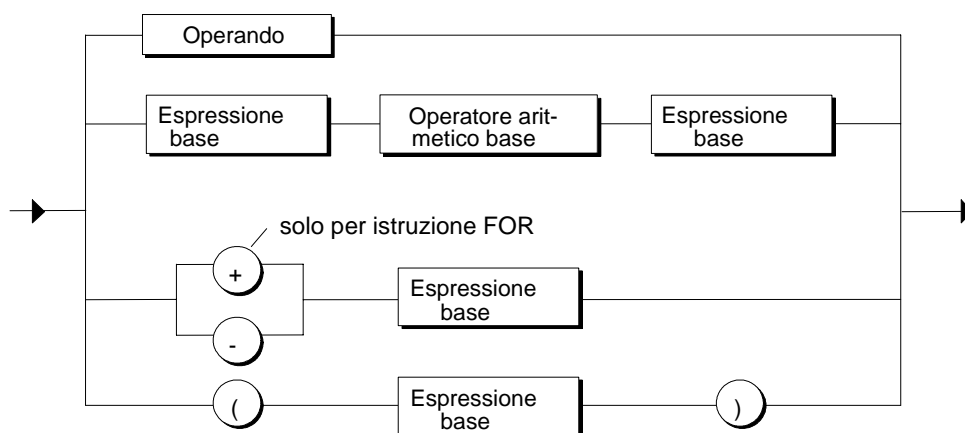


Esempi di assegnazioni iniziali corrette sono:

```
FOR I := 1 TO 20
FOR I := 1 TO (INIZIO + J)
```

Valore finale e ampiezza di passo

Per la formazione del valore finale e della ampiezza di passo desiderata si può formare un'espressione base. La generazione dell'espressione base deve essere conforme alla sintassi seguente:



- L'indicazione di *BY [ampiezza di passo]* può essere omessa. Se l'ampiezza di passo non viene specificata, essa assume il valore +1.
- Valore iniziale, valore finale e ampiezza di passo sono espressioni (vedere il capitolo "Espressioni, operazioni, operandi"). L'analisi avviene una sola volta all'inizio dell'esecuzione dell'istruzione FOR.
- Durante l'esecuzione del loop non è consentita alcuna modifica del valore finale e dell'ampiezza di passo.

Esempio

```
FUNCTION_BLOCK FOR_BSP
VAR
    INDICE: INT ;
    PASSWORD: ARRAY [1..50] OF STRING;
END_VAR
BEGIN
    FOR INDICE := 1 TO 50 BY 2 DO
        IF PASSWORD [INDICE] = 'KEY' THEN
            EXIT;
        END_IF;
    END_FOR;

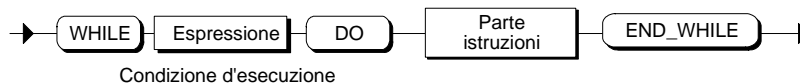
    END_FUNCTION_BLOCK
```

12.2.6 Istruzione WHILE

L'istruzione WHILE consente l'esecuzione iterativa di una sequenza di istruzioni sotto il controllo di una condizione di esecuzione. La condizione di esecuzione viene generata in base alle regole di una espressione logica.

Sintassi

Istruzione WHILE



L'istruzione WHILE viene elaborata conformemente alle regole seguenti:

- La condizione di esecuzione viene analizzata prima di ogni esecuzione della parte istruzioni (loop condizionato).
- La parte istruzioni che segue DO viene ripetuta finché la condizione di esecuzione fornisce il valore TRUE.
- Se si verifica il valore FALSE, il loop viene saltato e viene eseguita l'istruzione che segue il loop.

Esempio

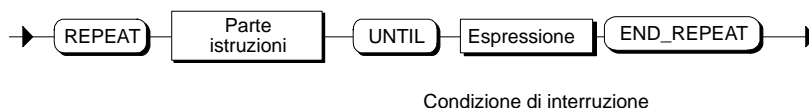
```
FUNCTION_BLOCK WHILE_BSP
VAR
    INDICE: INT ;
    PASSWORD: ARRAY [1..50] OF STRING ;
END_VAR
BEGIN
    INDICE := 1 ;
    WHILE INDICE <= 50 AND PASSWORD[INDICE] <> 'KEY' DO
        INDICE := INDICE + 2 ;
    END_WHILE ;
END_FUNCTION_BLOCK
```


12.2.7 Istruzione REPEAT

Un'istruzione REPEAT causa l'esecuzione iterativa di una sequenza di istruzioni compresa fra REPEAT e UNTIL finché non si verifica una condizione di interruzione. La condizione di interruzione viene generata in base alle regole di una espressione logica.

Sintassi

Istruzione REPEAT



La condizione viene verificata dopo ogni esecuzione del nucleo delle istruzioni. Ciò significa che il nucleo delle istruzioni viene eseguito almeno una volta, anche se la condizione di interruzione è soddisfatta fin dall'inizio.

Esempio

```
FUNCTION_BLOCK REPEAT_BSP
VAR
    INDICE: INT ;
    PASSWORD: ARRAY [1..50] OF STRING ;
END_VAR

BEGIN
    INDICE := 0 ;
    REPEAT
        INDICE := INDICE + 2 ;
    UNTIL INDICE > 50 OR PASSWORD[INDICE] = 'KEY'
    END_REPEAT ;

END_FUNCTION_BLOCK
```

12.2.8 Istruzione CONTINUE

L'istruzione CONTINUE serve per interrompere l'esecuzione momentanea di un loop di un'istruzione di ripetizione (FOR, WHILE o REPEAT).

Sintassi

Istruzione CONTINUE



L'istruzione CONTINUE viene elaborata conformemente alle regole seguenti:

- Questa istruzione causa l'immediata interruzione dell'esecuzione della sequenza di istruzioni.
- A seconda che la condizione per la ripetizione del loop è soddisfatta o meno, il nucleo viene eseguito ancora una volta oppure si esce dall'istruzione di ripetizione e viene eseguita l'istruzione immediatamente successiva.
- In un'istruzione FOR, immediatamente dopo l'istruzione CONTINUE la variabile d'esecuzione viene incrementata del passo di ampiezza indicato.

Esempio

```

FUNCTION_BLOCK CONTINUE_BSP
VAR
    INDICE :INT ;
    CAMPO  :ARRAY[1..100] OF INT ;
END_VAR

BEGIN
    INDEX := 0 ;
    WHILE INDICE <= 100 DO
        INDICE := INDICE + 1 ;
        // Se CAMPO[INDICE] è uguale a INDICE,
        // CAMPO [INDICE] non viene modificato:
        IF CAMPO[INDICE] = INDICE THEN
            CONTINUE ;

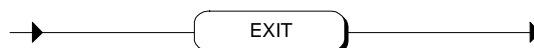
            END_IF ;
            CAMPO[INDICE] := 0 ;
        // Altre istruzioni
    END_WHILE ;
END_FUNCTION_BLOCK
  
```

12.2.9 Istruzione EXIT

Un'istruzione EXIT serve per uscire da un loop (FOR, WHILE o REPEAT) in qualsiasi momento e indipendentemente dal fatto che la condizione di interruzione sia stata soddisfatta o meno.

Sintassi

Istruzione EXIT



L'istruzione EXIT viene elaborata conformemente alle regole seguenti:

- Questa istruzione causa l'uscita immediata da quella istruzione di ripetizione, che circonda immediatamente l'istruzione EXIT.
- L'esecuzione del programma viene continuata dopo la fine del loop di ripetizione (ad es. dopo END_FOR).

Esempio

```

FUNCTION_BLOCK EXIT_BSP
VAR
    INDICE_1      : INT ;
    INDICE_2      : INT ;
    INDICE_CERCATO : INT ;
    PASSWORD      : ARRAY[1..51] OF STRING ;
END_VAR

BEGIN
    INDICE_2      := 0 ;
    FOR INDICE_1   := 1 TO 51 BY 2 DO
        // Uscire dal loop FOR se
        // PASSWORD[INDICE_1] è uguale a 'KEY':
        IF PASSWORD[INDICE_1] = 'KEY' THEN
            INDICE_2 := INDICE_1 ;
            EXIT ;
        END_IF ;
    END_FOR ;
    // La seguente assegnazione di valore viene effettuata
    // dopo l'esecuzione di EXIT o dopo la conclusione
    // regolare del loop FOR:
    INDICE_CERCATO := INDICE_2 ;
END_FUNCTION_BLOCK
  
```

12.2.10 Istruzione GOTO

Con una istruzione GOTO si può realizzare un salto di programma. Essa causa il salto immediato ad un'etichetta di salto indicata e quindi ad un'altra istruzione nell'ambito dello stesso blocco.

Le istruzioni GOTO dovrebbero essere utilizzate solo in casi speciali, p. es. per il trattamento degli errori. In base alle regole della programmazione strutturata, l'istruzione GOTO non dovrebbe essere utilizzata affatto.

Sintassi



L'etichetta di salto identifica un'etichetta nel blocco convenzioni LABEL/END_LABEL. Questa etichetta precede l'istruzione che deve essere eseguita per prima dopo GOTO.

Quando si impiega l'istruzione GOTO si devono osservare le regole seguenti:

- La destinazione di un'istruzione di salto deve trovarsi all'interno dello stesso blocco.
- La destinazione di salto deve essere univoca.
- Un salto in un blocco di loop non è consentito. È possibile, invece, un salto da un blocco di loop.

Esempio

```
FUNCTION_BLOCK GOTO_BSP
VAR
    INDICE          : INT ;
    A                : INT ;
    B                : INT ;
    C                : INT ;
    PASSWORD         : ARRAY[1..51] OF STRING ;
END_VAR
LABEL
    ETICHETTA1, ETICHETTA2, ETICHETTA3 ;
END_LABEL

BEGIN
    IF A > B THEN
        GOTO ETICHETTA1 ;
    ELSIF A > C THEN
        GOTO ETICHETTA2 ;
    END_IF ;
    // . . .
    ETICHETTA1: INDICE := 1 ;
                GOTO ETICHETTA3 ;
    ETICHETTA2: INDICE := 2 ;
    // . . .
    ETICHETTA3:
    // . . .
```

12.2.11 Istruzione RETURN

Un'istruzione RETURN causa l'uscita dal blocco momentaneamente in elaborazione (OB, FB, FC) e il ritorno al blocco richiamante o al sistema operativo quando si esce da un OB.

Sintassi

Istruzione RETURN



Nota

Un'istruzione RETURN alla fine della parte di un blocco di codice o della parte convenzioni di un blocco dati è ridondante poiché essa viene eseguita automaticamente.

12.3 Richiamo di funzioni e blocchi funzionali

12.3.1 Richiamo e trasferimento di parametri

Richiamo di FC e FB

Per garantire la leggibilità e la manutenzione dei programmi utente, il complesso di funzioni di un programma viene suddiviso in vari compiti parziali, i quali vengono svolti da blocchi funzionali (FB) e da funzioni (FC). Da un blocco S7-SCL si possono richiamare altri FC o FB. Sono blocchi richiamabili:

- i blocchi funzionali e le funzioni generati in S7-SCL,
- i blocchi funzionali e le funzioni generati in altri linguaggi STEP 7 (KOP, FUP, AWL),
- Funzioni di sistema (SFC) e blocchi funzionali di sistema (SFB) disponibili nel sistema operativo della CPU.

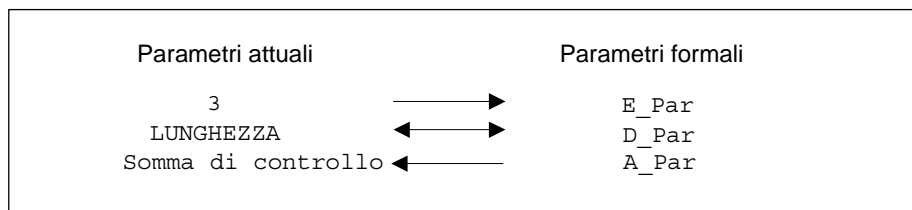
Principio del trasferimento di parametri

Durante il richiamo di funzioni o blocchi funzionali avviene uno scambio di dati fra il blocco richiamante e quello richiamato. Nell'interfaccia del blocco richiamato sono definiti i parametri con cui il blocco lavora. Tali parametri vengono chiamati parametri formali e sono semplicemente dei "Segnaposti" per i parametri che vengono trasferiti al blocco durante il richiamo. I parametri passati al blocco durante il richiamo vengono chiamati parametri attuali.

Sintassi del trasferimento dei parametri

I parametri che devono essere trasferiti devono essere indicati nel richiamo come lista di parametri. I parametri vengono inseriti tra parentesi. Più parametri vengono separati mediante virgole.

Nel seguente esempio di un richiamo di una funzione vengono indicati un parametro d'ingresso, un parametro di ingresso/uscita e un parametro d'uscita.



L'indicazione dei parametri avviene in forma di assegnazione di valori. Mediante questa assegnazione di valori si assegnano determinati valori (parametri attuali) ai vari parametri che sono stati definiti nella parte dichiarazioni del blocco richiamato (parametri formali).

Richiamo assoluto:

```
FB10.DB20 (X1:=5,X2:=78,.....);
```

Richiamo simbolico:

```
ANTRIEB.EIN (X1:=5,X2:=78,.....);
```



Assegnazione di parametri

12.3.2 Richiamo di blocchi funzionali

12.3.2.1 Richiamo di blocchi funzionali (FB o SFB)

Durante il richiamo di un blocco funzionale si possono utilizzare sia blocchi dati di istanza globale che aree di istanza locali del blocco dati di istanze attuale.

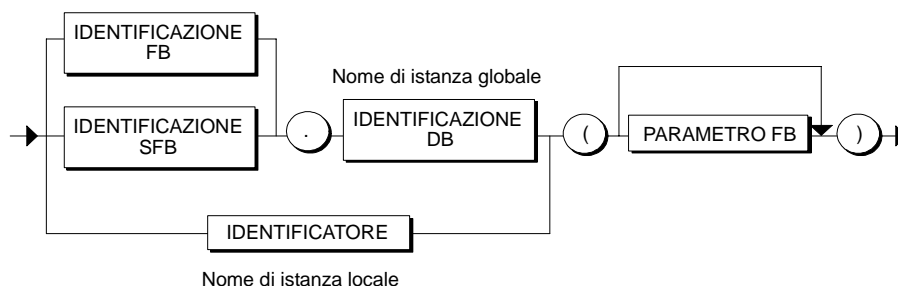
Il richiamo di un FB come istanza locale si distingue dal richiamo come istanza globale per il modo in cui vengono memorizzati i dati. In questo caso i dati non vengono depositati in un DB speciale, ma nel blocco dati istanza dell'FB richiamante.

Sintassi

Richiami di FB

FB: Blocco funzionale

SFB: Blocco funzionale di sistema

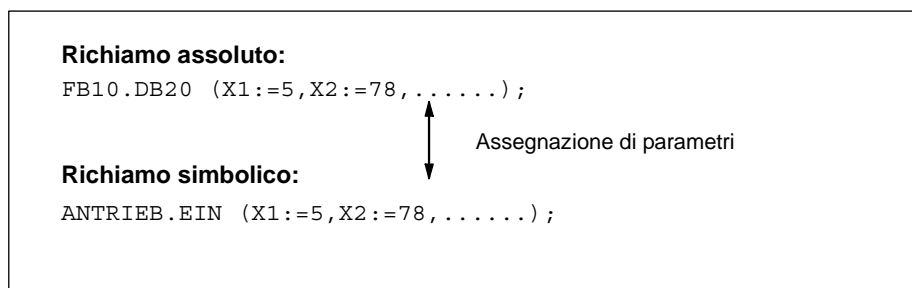


Richiamo come istanza globale

Il richiamo avviene in un'istruzione di richiamo con indicazione:

- del nome del blocco funzionale o del blocco funzionale di sistema (identificazione di FB o SFB),
- del blocco dati di istanza (identificazione DB),
- dell'assegnazione dei parametri (parametri FB).

Un richiamo di un'istanza globale può essere definito in modo assoluto o simbolico.



Richiamo come istanza locale

Il richiamo avviene in un'istruzione di richiamo con indicazione:

- del nome di istanza locale (IDENTIFICATORE),
- dell'assegnazione dei parametri (parametri FB),

Un richiamo di un'istanza locale è sempre simbolico. Il nome simbolico deve essere definito nella parte convenzioni del blocco richiamante.

```
MOTORE (X1:=5,X2:=78, . . . . .) ;
```



Alimentazione dei parametri

12.3.2.2 Assegnazione dei parametri FB

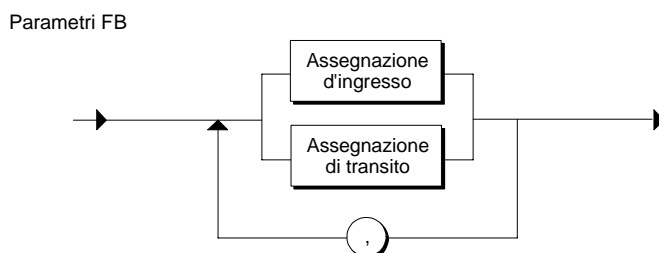
Nel richiamo di un blocco funzionale come istanza globale o locale l'utente deve passare i seguenti parametri nella lista dei parametri:

- parametri d'ingresso
- parametri di ingresso/uscita

I parametri di uscita non vengono indicati durante il richiamo di un FB, bensì dopo il richiamo di esso.

Sintassi di assegnazioni di valori per la definizione dei parametri FB:

La sintassi dell'indicazione dei parametri di FB è identica per istanze globali e locali.



Per l'assegnazione dei parametri valgono le seguenti regole:

- La sequenza delle assegnazioni è irrilevante.
- I tipi di dati dei parametri formali e dei parametri attuali devono coincidere.
- Le singole assegnazioni sono separate fra loro mediante virgole.
- Nei richiami FB non sono consentite assegnazioni d'uscita. Il valore di un parametro d'uscita definito in precedenza è memorizzato nei dati di istanza. Ad esso si può accedere da qualsiasi FB. Per poterlo leggere da un FB, l'utente deve definire un apposito accesso.
- Osservare le particolarità dei parametri di tipo di dati ANY e di tipo di dati POINTER.

Risultato dopo l'esecuzione del blocco

Dopo l'esecuzione del blocco

- i parametri d'ingresso attuali trasferiti sono immutati
- i parametri d'ingresso trasferiti ma modificati sono aggiornati. Fanno eccezione i parametri di ingresso/uscita con tipo di dati semplice.
- i parametri d'uscita del blocco richiamante possono essere letti dal blocco dati di istanza locale.

Esempio

Un richiamo con un'assegnazione di un parametro d'ingresso e di un parametro di ingresso/uscita potrebbe avere p. es. l'aspetto seguente:

```
FB31.DB77 (E_Par:=3, D_Par:=LUNGHEZZA) ;
```

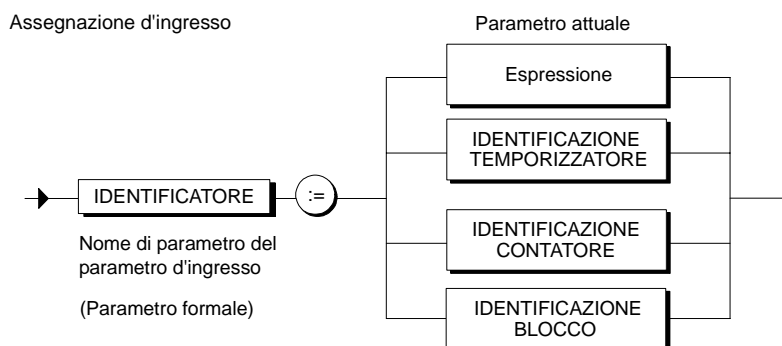
12.3.2.3 Assegnazione d'ingresso (FB)

Mediante assegnazioni d'ingresso vengono assegnati dei valori (parametri attuali) ai parametri d'ingresso formali. L'FB non può modificare questi parametri attuali. L'assegnazione di parametri d'ingresso attuali è opzionale. Se non viene indicato alcun parametro attuale, i valori dell'ultimo richiamo rimangono immutati.

Possibili parametri attuali:

Parametro attuale	Spiegazione
Espressione	<ul style="list-style-type: none"> Espressione aritmetica, logica o di confronto Costante Variabile ampliata
Identificazione di TEMPORIZZATORI/CONTATORI	Definiscono un determinato temporizzatore o un determinato contatore che deve essere utilizzato per l'elaborazione di un blocco.
Nome del blocco	<p>Definiscono un determinato blocco, il quale viene utilizzato come parametro d'ingresso. Il tipo di blocco (FB, FC, DB) viene definito nella dichiarazione del parametro d'ingresso.</p> <p>Nell'assegnazione di parametri si deve indicare il numero del blocco. Quest'ultimo può essere indicato in modo assoluto o simbolico.</p>

Sintassi



12.3.2.4 Assegnazione di ingresso/uscita (FB)

Le assegnazioni di ingresso/uscita vengono usate per assegnare parametri attuali ai parametri di ingresso/uscita formali dell'FB richiamato. L'FB richiamante può modificare i parametri di ingresso/uscita. Il nuovo valore del parametro creato durante l'elaborazione dell'FB viene registrato nuovamente nel parametro attuale. Il valore originale viene sovrascritto.

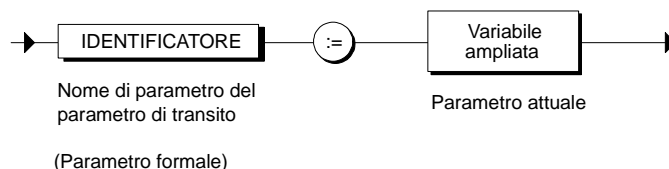
Se nell'FB richiamato sono stati definiti dei parametri di ingresso/uscita, questi devono essere assegnati durante il primo richiamo. Nelle esecuzioni successive l'indicazione dei parametri attuali è opzionale. Tuttavia, per i parametri di ingresso/uscita di un tipo di dati semplice non viene eseguita alcuna attualizzazione del parametro attuale se durante il richiamo i parametri formali non vengono assegnati.

Poiché il parametro attuale assegnato può essere modificato durante l'esecuzione dell'FB, esso deve essere una variabile.

Parametro attuale	Spiegazione
Variabile ampliata	<p>Sono disponibili i seguenti tipi di variabili ampliate:</p> <ul style="list-style-type: none"> • variabili e parametri semplici • accesso a variabili assolute • accesso a blocchi dati • richiamo di funzioni

Sintassi

Assegnazione di transito



Nota

- Nell'assegnazione dei tipi di dati ANY e POINTER vengono applicate regole particolari.
- Nei parametri di ingresso/uscita di un tipo di dati non semplici, non sono consentiti come parametri attuali:
 - Parametri di ingresso/uscita di FB
 - Parametri FC

12.3.2.5 Lettura di valori iniziali (richiamo FB)

Dopo l'elaborazione del blocco richiamato, i parametri d'uscita possono essere letti dal blocco di istanza globale o dall'area di istanza locale tramite una assegnazione di valori.

Esempio

```
RISULTATO := DB10.CONTROL ;
```

12.3.2.6 Esempio di richiamo come istanza globale

Un blocco funzionale con un loop FOR potrebbe avere l'aspetto seguente. Negli esempi riportati si presuppone che nella tabella dei simboli per FB17 sia stato definito il simbolo TEST.

Blocco funzionale

```
FUNCTION_BLOCK TEST

VAR_INPUT
    VALORE FINALE:      INT;  //Parametro d'ingresso
END_VAR
VAR_IN_OUT
    IQ1      :          REAL; //Parametro di ingresso/uscita
END_VAR
VAR_OUTPUT
    CONTROL:          BOOL; //Parametro d'uscita
END_VAR
VAR
    INDEX:      INT;
END_VAR

BEGIN
    CONTROL      :=FALSE;
    FOR INDEX    := 1 TO VALORE FINALE DO
        IQ1      :=IQ1*2;
        IF IQ1 > 10000 THEN
            CONTROL      := TRUE;
        END_IF;
    END_FOR;
END_FUNCTION_BLOCK
```

Richiamo

Per richiamare questo FB si può scegliere una delle varianti seguenti. Si presuppone che VARIABILE1 sia stata definita come variabile REAL nel blocco richiamante.

```
// Richiamo assoluto, istanza globale:
FB17.DB10 (VALORE FINALE:=10, IQ1:=VARIABILE1);

//Richiamo simbolico, istanza globale:
TEST.TEST_1 (VALORE FINALE:= 10, IQ1:= VARIABILE1) ;
```

Risultato

Dopo l'esecuzione del blocco, in VARIABILE1 è disponibile il valore calcolato per il parametro di ingresso/uscita IQ1.

Lettura del valore d'uscita

I due esempi sottostanti servono ad illustrare le alternative possibili per leggere il parametro d'uscita CONTROL:

```
// L'accesso al parametro d'uscita  
//viene effettuato mediante:  
    RISULTATO:= DB10.CONTROL;
```

```
//Il parametro di uscita può però essere anche utilizzato  
//in un altro richiamo di FB per assegnare  
//direttamente un parametro di ingresso:  
    FB17.DB12 (EIN_1:= DB10.CONTROL);
```

12.3.2.7 Esempio di richiamo come istanza locale

Un blocco funzionale con un semplice loop FOR potrebbe essere programmato come illustrato nell'esempio "Richiamo come istanza locale" in cui si presuppone che nella tabella dei simboli sia stato definito il simbolo TEST per l'FB17.

Questo FB può essere richiamato come indicato nell'esempio seguente a condizione che nel blocco richiamante VARIABILE1 sia stata dichiarata come REAL:

Richiamo

```
FUNCTION_BLOCK CALL
VAR
// Convenzione per l'istanza locale
    TEST_L : TEST ;
    VARIABILE1 : REAL ;
    RISULTATO : BOOL ;
END_VAR
BEGIN
. . .

// Richiamo dell'istanza locale:
TEST_L (VALORE FINALE:= 10, IQ1:= VARIABILE1) ;
```

Lettura del parametro d'uscita

```
Il parametro d'uscita CONTROL può essere letto come segue:
// L'accesso al parametro d'uscita
// viene effettuato mediante:
RISULTATO := TEST_L.CONTROL ;
END_FUNCTION_BLOCK
```

12.3.3 Richiamo di funzioni

12.3.3.1 Richiamo di funzioni (FC)

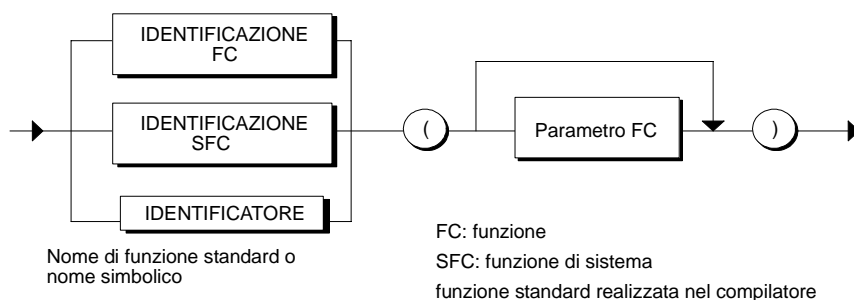
Il richiamo di una funzione avviene indicando il nome della funzione (IDENTIFICAZIONE FC, o IDENTIFICAZIONE SFC o IDENTIFICATORE) nonché la lista dei parametri. Il nome della funzione, che identifica il valore di ritorno, può essere indicato come nome assoluto o simbolico:

```
FC31 (X1:=5, Q1:=Somma di controllo)      // Assoluto  
DISTANZA (X1:=5, Q1:=Somma di controllo) // Simbolico
```

Dopo il richiamo, i risultati sono a disposizione della funzione come valore di ritorno o come parametri d'uscita e di ingresso/uscita (parametri attuali).

Sintassi

Richiamo di funzione



Nota

Se in S7-SCL viene richiamata una funzione il cui valore di ritorno non è stato assegnato, ciò può condurre ad un'esecuzione errata del programma utente:

- In una funzione programmata in S7-SCL ciò può accadere quando il valore di ritorno è stato assegnato ma non viene eseguita la relativa istruzione.
- In una funzione programmata in AWL/KOP/FUP ciò può avvenire quando la funzione è stata programmata senza assegnazione del valore di ritorno oppure non viene eseguita la relativa istruzione.

12.3.3.2 Valore di ritorno (FC)

Contrariamente ai blocchi funzionali, le funzioni forniscono quale risultato il valore di ritorno. Per tale motivo, le funzioni possono essere trattate come operandi (fanno eccezione le funzioni del tipo VOID).

La funzione calcola il valore di ritorno, che ha lo stesso nome della funzione, e lo ritorna al blocco richiamante. Qui, il valore sostituisce il richiamo della funzione.

Nella seguente assegnazione di valore viene richiamata p. es. la funzione `DISTANZA` e il risultato viene assegnato alla variabile `LUNGHEZZA`:

```
LUNGHEZZA:= DISTANZA (X1:=-3, Y1:=2);
```

Il valore di ritorno può essere utilizzato nei seguenti elementi di un FC o FB:

- in un'assegnazione di valori,
- in un'espressione logica, aritmetica o di confronto oppure
- come parametro per un ulteriore richiamo di blocco funzionale/funzione.

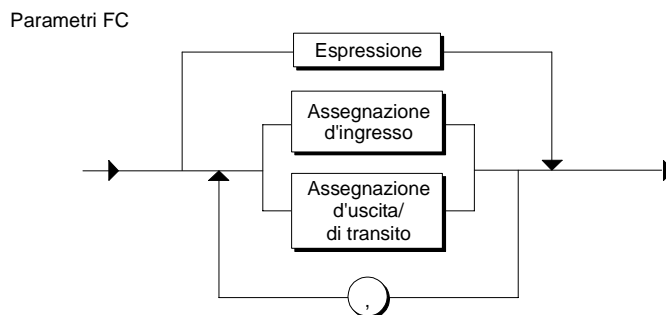
Nota

- Nelle funzioni con il valore di ritorno ANY, almeno un parametro d'ingresso e di ingresso/uscita deve essere del tipo ANY. Se sono stati definiti più parametri ANY, questi devono essere assegnati con parametri attuali della stessa classe tipo (p. es. INT, DINT e REAL). In tal caso il valore di ritorno appartiene al più grande tipo di questa classe.
 - La lunghezza massima del tipo di dati STRING può essere ridotta da 254 caratteri a un numero qualunque di caratteri.
-

12.3.3.3 Parametri FC

Contrariamente ai blocchi funzionali, le funzioni non possiedono alcuna memoria in cui poter memorizzare i valori dei parametri. Durante l'esecuzione della funzione, i dati locali vengono memorizzati solo temporaneamente. Per questo motivo, durante il richiamo di una funzione si devono assegnare dei parametri attuali a tutti i parametri d'ingresso, di ingresso/uscita e d'uscita, che sono stati definiti nella parte convenzioni di una funzione.

Sintassi



Regole

Per l'assegnazione dei parametri valgono le seguenti regole:

- La sequenza delle assegnazioni è irrilevante.
- I tipi di dati dei parametri formali e dei parametri attuali devono coincidere.
- Le singole assegnazioni sono separate fra loro mediante virgole.

Esempio

Un richiamo con assegnazioni dei parametri di ingresso, uscita e di ingresso/uscita potrebbe avere p. es. l'aspetto seguente:

```
FC32 (E_Param1:=5,D_Param1:=LUNGHEZZA,  
      A_Param1:=Somma di controllo)
```

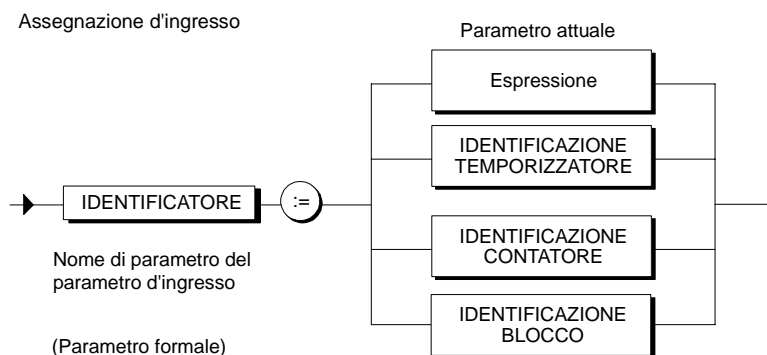
12.3.3.4 Assegnazione d'ingresso (FC)

Mediante assegnazioni d'ingresso vengono assegnati dei valori (parametri attuali) ai parametri d'ingresso formali della FC richiamata. La FC può lavorare con questi parametri, ma non può modificarli. Contrariamente al richiamo FB, nel richiamo FC questa assegnazione non è opzionale.

Nelle assegnazioni d'ingresso si possono assegnare i seguenti parametri attuali:

Parametro attuale	Spiegazione
Espressione	Un'espressione rappresenta un determinato valore e si compone di operandi e operazioni. Sono disponibili i seguenti tipi di espressioni: <ul style="list-style-type: none"> • Espressione aritmetica, logica o di confronto • Costante • Variabile ampliata
Identificatore di TEMPORIZZATORI/CONTATORI	Definiscono un determinato temporizzatore o un determinato contatore che deve essere utilizzato per l'elaborazione di un blocco.
Nome del blocco	Definiscono un determinato blocco, il quale viene utilizzato come parametro d'ingresso. Il tipo di blocco (FB, FC, DB) viene definito nella dichiarazione del parametro d'ingresso. Nell'assegnazione dei parametri si deve indicare l'indirizzo del blocco. Quest'ultimo può essere indicato in modo assoluto o simbolico.

Sintassi



Nota

Nei parametri formali d'ingresso di tipo di dati non semplici, i parametri di ingresso/uscita FB e i parametri FC non sono consentiti come parametri attuali. Osservare le particolarità dei tipi di dati ANY e POINTER.

12.3.3.5 Assegnazione di uscita e di ingresso/uscita (FC)

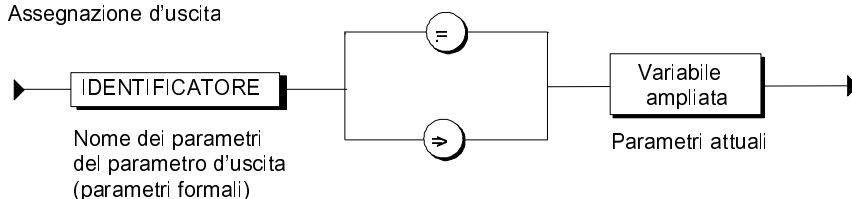
In un'assegnazione d'uscita si definisce in quale variabile del blocco richiamante devono essere scritti i valori d'uscita che vengono generati durante l'elaborazione di una funzione. Con un'assegnazione di ingresso/uscita si assegna un valore attuale ad un parametro di ingresso/uscita.

I parametri attuali nelle assegnazioni d'uscita e di ingresso/uscita devono essere delle variabili poiché l'FC deve scrivere dei valori in tali parametri. Per tale motivo, i parametri d'ingresso non possono essere assegnati in assegnazioni di ingresso/uscita (in tal caso non sarebbe possibile scrivere il valore). Perciò, nelle assegnazioni d'uscita e di ingresso/uscita si può assegnare solo la variabile ampliata:

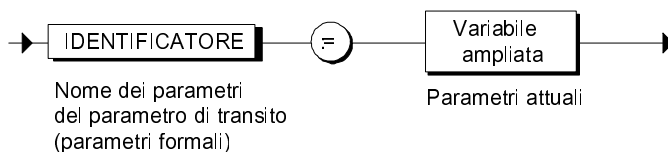
Parametro attuale	Spiegazione
Variabile ampliata	<p>Sono disponibili i seguenti tipi di variabili ampliate:</p> <ul style="list-style-type: none"> • variabili e parametri semplici • accesso a variabili assolute • accesso a blocchi dati • richiamo di funzioni

Sintassi

Assegnazione d'uscita



Assegnazione di transito



Nota

I parametri attuali seguenti non sono ammessi nei parametri formali di uscita e di ingresso/uscita:

- parametri di ingresso FC/FB
- parametri di ingresso/uscita FB di tipo di dati non semplici
- parametri FC di ingresso/uscita e di uscita di tipi di dati non semplici
- Osservare le particolarità dei tipi di dati ANY e POINTER.
- La lunghezza massima del tipo di dati STRING può essere ridotta da 254 caratteri a un numero qualunque di caratteri.

12.3.3.6 Esempio di richiamo di funzione

Funzione da richiamare

Una funzione DISTANZA per il calcolo della distanza di due punti (X1,Y1) e (X2,Y2) nel piano con l'impiego di un sistema di coordinate cartesiane potrebbe avere l'aspetto seguente (negli esempi riportati si presuppone sempre che il simbolo DISTANZA sia stato definito in una tabella dei simboli per FC37).

```
FUNCTION DISTANZA: REAL // simbolico
VAR_INPUT
    X1: REAL;
    X2: REAL;
    Y1: REAL;
    Y2: REAL;
END_VAR
VAR_OUTPUT
    Q2: REAL;
END_VAR
BEGIN
    DISTANZA:= SQRT( (X2-X1)**2 + (Y2-Y1)**2 );
    Q2:= X1+X2+Y1+Y2;
END_FUNCTION
```

Blocco richiamante

Per l'ulteriore utilizzo di un valore di una funzione sono disponibili fra l'altro le seguenti possibilità:

```
FUNCTION_BLOCK CALL
VAR
    LUNGHEZZA : REAL ;
    SOMMACONTROLLO : REAL ;
    RAGGIO : REAL;
    Y : REAL;
END_VAR
BEGIN
    . . .
    // Richiamo in un'assegnazione di valori:
        LUNGHEZZA := DISTANZA (X1:=3, Y1:=2, X2:=8.9, Y2:= 7.4,
    Q2:=Somma di controllo) ;
    //Richiamo in un'espressione aritmetica o logica, ad es.
        Y := RAGGIO + DISTANZA (X1:=-3, Y1:=2, X2:=8.9, Y2:=7.4,
    Q2:=Somma di controllo)
    //Utilizzo di un altro blocco richiamato nell'assegnazione dei
    parametri
        FB32.DB32 (DISTANZA:= DISTANZA (X1:=-3, Y1:=2, X2:=8.9,
    Y2:=7.4), Q2:=Somma di controllo)
    . . .
END_FUNCTION_BLOCK
```

12.3.4 Parametri con definizione implicita

12.3.4.1 Parametro di ingresso EN

Ogni blocco funzionale e ogni funzione possiede il parametro d'ingresso EN definito implicitamente. Il parametro EN è del tipo BOOL e è depositato nell'area dei dati di blocco temporanei. Se EN è uguale a TRUE, il blocco richiamato viene eseguito, in caso contrario esso non viene eseguito. L'assegnazione del parametro EN è opzionale. Si deve però tener presente che esso non può essere dichiarato nella parte convenzioni di un blocco o di una funzione.

Poiché EN è un parametro d'ingresso, EN non può essere modificato nell'ambito di un blocco.

Nota

Il valore di ritorno di una funzione non è definito, se la funzione non è stata richiamata (EN : FALSE).

Esempio

```
FUNCTION_BLOCK FB57
VAR
    MIA_ENABLE: BOOL ;
    Risultato : REAL;
END_VAR
// . . .
BEGIN
    // . . .
    MIA_ENABLE:= FALSE ;

    // Richiamo di una funzione con assegnazione del parametro EN:
    Risultato := FC85 (EN:= MIA_ENABLE, PAR_1:= 27) ;
    // FC85 non è stato eseguito perché MIA_ENABLE precedente è
    impostato
    uguale a FALSE....

END_FUNCTION_BLOCK
```

12.3.4.2 Parametro di uscita ENO

Ogni blocco funzionale e ogni funzione possiede il parametro d'uscita ENO definito implicitamente, il quale è un parametro del tipo BOOL e è depositati nell'area dei dati di blocco temporanei. Alla fine dell'esecuzione di un blocco, il valore attuale del flag OK viene scritto in ENO.

Subito dopo il richiamo di un blocco, in base al valore di ENO si può verificare se tutte le operazioni del blocco sono state eseguite correttamente o se si sono verificati errori.

Esempio

```
// Richiamo di un blocco funzionale:  
FB30.DB30 ([assegnazione parametri]);  
  
// Verificare se nel blocco richiamato tutto  
// si è svolto correttamente:  
IF ENO THEN  
// tutto corretto  
// . . .  
ELSE  
// Si sono verificati errori per cui si deve  
// attivare la gestione errori  
// . . .  
END_IF;
```


13 Contatori e temporizzatori

13.1 Contatori

13.1.1 Funzioni di conteggio

STEP 7 mette a disposizione una serie di funzioni standard di conteggio. Questi contatori possono essere utilizzati nel programma utente senza doverli prima definire. Essi devono solo essere alimentati con i necessari parametri. STEP 7 offre le seguenti funzioni di conteggio:

Funzione di conteggio	Significato
S_CU	Contatori di conteggio in avanti (Counter Up)
S_CD	Contatori di conteggio all'indietro (Counter Down)
S_CUD	Conteggio in avanti e all'indietro (Counter Up Down)

13.1.2 Richiamo di funzioni di conteggio

Le funzioni di conteggio vengono richiamate in modo analogo alle funzioni. L'identificazione di funzione può essere impiegata ovunque al posto di un operando in un'espressione, a condizione che il tipo di valore della funzione sia compatibile con il tipo dell'operando sostituito.

Il valore della funzione (valore di ritorno) che viene restituito al punto di richiamo è il valore di conteggio attuale (formato BCD) del tipo WORD.

Richiamo assoluto o dinamico

Nel richiamo è possibile specificare come numero di contatore un valore assoluto quale B. C_NO:=Z10). Tale valore non è tuttavia modificabile durante l'esecuzione.

Al posto del numero assoluto del contatore, si può introdurre anche una variabile o costante con tipo di dati INT. Ciò presenta il vantaggio di poter eseguire un richiamo dinamico del contatore assegnando alla variabile o costante un altro numero assoluto ad ogni richiamo.

Un'altra possibilità di eseguire un richiamo dinamico consiste nell'indicare un variabile di tipo COUNTER.

Esempi

```
//Esempio di richiamo assoluto:
S_CUD (C_NO:=Z12,
      CD:=E0.0,
      CU:=E0.1,
      S:=E0.2 & E0.3,
      PV:=120,
      R:=FALSE,
      CV:=binVal,
      Q:=actFlag);

//Esempio di richiamo dinamico: ad ogni esecuzione di
//un loop FOR viene richiamato un contatore diverso:
FUNCTION_BLOCK CONT
VAR_INPUT
    Cont: ARRAY [1..4] of STRUCT
        C_NO: INT;
        PV  : WORD;
    END_STRUCT;
.
.
END_VAR
.
.
FOR I:= 1 TO 4 DO
    S_CD(C_NO:=cont[I].C_NO, S:=true, PV:= cont[I].PV);
END_FOR;

//Esempio di richiamo dinamico in caso
//di utilizzo di una variabile
//con tipo di dati COUNTER:
FUNCTION_BLOCK CONTATORE
VAR_INPUT
    Miocontatore:COUNTER;
END_VAR
.
.
CurrVal:=S_CD (C_NO:=Miocontatore,.....);
```

Nota

I nomi delle funzioni e dei parametri sono uguali nei mnemonici tedesco e inglese. Dipendente dal mnemonico è solo la denominazione del contatore (ad esempio tedesco: Z, inglese: C).

13.1.3 Assegnazione di parametri nelle funzioni di conteggio

La tabella seguente mostra un sommario dei parametri per funzioni di conteggio:

Parametri	Tipo di dati	Descrizione
C_NO	COUNTER INT	Numero di identificazione del contatore (IDENTIFICAZIONE CONTATORE); l'area dipende dalla CPU.
CD	BOOL	Ingresso CD: Conteggio all'indietro
CU	BOOL	Ingresso CU: Conteggio in avanti
S	BOOL	Ingresso per impostazione del contatore
PV	WORD	Valore nel campo 0 - 999 per l'impostazione del contatore (introdotto come 16#<Valore>, con valore in formato BCD)
R	BOOL	Ingresso di reset
Q	BOOL	Uscita: Stato del contatore
CV	WORD	Uscita: stato del contatore binario
RET_VAL	WORD	Risultato in formato BCD

Regole

Poiché i valori dei parametri (per es. CD:=E0.0) sono memorizzati in modo globale, in determinati casi la loro indicazione è opzionale. Queste regole di carattere generale devono essere rispettate in fase di assegnazione dei parametri:

- Il parametro per l'identificazione del contatore C_NO deve essere alimentato durante il richiamo. Al posto del numero assoluto del contatore (ad es. Z12) nel richiamo si può specificare anche una variabile o una costante con il tipo di dati INT o un parametro di ingresso di tipo COUNTER.
- Si deve alimentare il parametro CU (contatore in avanti) o il parametro CD (contatore all'indietro).
- L'indicazione dei parametri PV (valore di preimpostazione) e S (impostazione) può essere effettuata a coppie.
- Il valore del risultato in formato BCD è sempre il valore della funzione.

Esempio

```

FUNCTION_BLOCK FB1
VAR
    CurrVal, binVal: word;
    actFlag: bool;
END_VAR

BEGIN
    CurrVal := S_CD (C_NO:= Z10, CD:=TRUE, S:=TRUE, PV:=100, R:=FALSE,
                    CV:=binVal, Q:=actFlag);
    CurrVal := S_CU (C_NO:= Z11, CU:=M0.0, S:=M0.1,
                    PV:=16#110, R:=M0.2,
                    CV:=binVal, Q:=actFlag);
    CurrVal := S_CUD (C_NO:= Z12, CD:=E0.0, CU:=E0.1,
                    S:=E0.2 &E0.3, PV:=120, R:=FALSE,
                    CV:=binVal, Q:=actFlag);
    CurrVal := S_CD (C_NO:= Z10, CD:=FALSE,
                    S:=FALSE, PV:=100, R:=TRUE,
                    CV:=binVal, Q:=actFlag);
END_FUNCTION_BLOCK

```

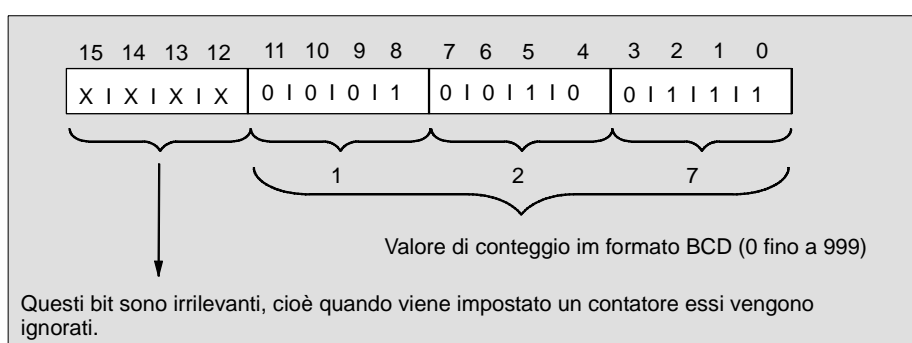
13.1.4 Introduzione ed analisi del valore del contatore

Per l'introduzione del valore di preimpostazione o per l'analisi del risultato della funzione è necessaria la rappresentazione interna del valore del contatore. Il valore del contatore ha un tipo di dati WORD, in cui i bit 0-11 contengono il valore del contatore in codice BCD. I bit 12-15 non sono rilevanti.

Quando si imposta il contatore, il valore definito dall'utente viene scritto nel contatore. Il campo di valori è compreso fra 0 e 999. Si può modificare il valore del contatore nell'ambito di questo campo indicando le operazioni di conteggio in avanti/all'indietro (S_CUD), conteggio in avanti (S_CU) e conteggio all'indietro (S_CD).

Formato

La figura seguente illustra la configurazione di bit del valore di conteggio:



Ingresso

- Decimale come valore integer: p. es. 295, purché questo valore corrisponda ad un valido formato BCD.
- In formato BCD (introduzione come costante esadecimale): p. es. 16#127

Analisi:

- Come risultato della funzione (tipo WORD): in formato BCD
- Come parametro d'uscita CV (tipo WORD): binario

13.1.5 Conteggio in avanti (S_CU)

Con il contatore Counter Up (S_CU) si possono eseguire solo operazioni di conteggio in avanti. La tabella illustra il modo di funzionamento del contatore:

Operazione	Modo di funzionamento
Conteggio in avanti	Il valore del contatore viene aumentato di "1", quando lo stato del segnale all'ingresso CU passa da "0" a "1" e il valore di conteggio è minore di 999.
Imposta contatore	Quando lo stato del segnale all'ingresso S passa da "0" a "1", il contatore viene impostato con il valore dell'ingresso PV . Un tale cambio di segnale è sempre necessario per poter impostare un contatore.
Resetta	Il contatore viene resettato quando l'ingresso R = 1. Il reset del contatore imposta il valore di conteggio a "0".
Interrogazione contatore	Un'interrogazione dello stato del segnale all'uscita Q risulta "1" se il valore di conteggio è maggiore di "0". L'interrogazione risulta "0" se il valore di conteggio è uguale a "0".

13.1.6 Conteggio all'indietro (S_CD)

Con il contatore Counter Down (S_CD) si possono eseguire solo operazioni di conteggio all'indietro. La tabella illustra il modo di funzionamento del contatore:

Operazione	Modo di funzionamento
Conteggio all'indietro	Il valore del contatore viene diminuito di "1", quando lo stato del segnale all'ingresso CU passa da "0" a "1" (fronte di salite) e il valore di conteggio è maggiore di 999.
Imposta contatore	Quando lo stato del segnale all'ingresso S passa da "0" a "1", il contatore viene impostato con il valore dell'ingresso PV . Un tale cambio di segnale è sempre necessario per poter impostare un contatore.
Resetta	Il contatore viene resettato quando l'ingresso R = 1. Il reset del contatore imposta il valore di conteggio a "0".
Interrogazione contatore	Un'interrogazione dello stato del segnale all'uscita Q risulta "1" se il valore di conteggio è maggiore di "0". L'interrogazione risulta "0" se il valore di conteggio è uguale a "0".

13.1.7 Conteggio in avanti e all'indietro (S_CUD)

Con i contatori Counter Up Down (S_CUD) si possono eseguire operazioni di conteggio sia in avanti che all'indietro. In caso di simultaneità degli impulsi di conteggio in avanti e all'indietro, vengono eseguite entrambe le operazioni. Il valore di conteggio rimane immutato. La tabella illustra il modo di funzionamento del contatore:

Operazione	Modo di funzionamento
Conteggio in avanti	Il valore del contatore viene aumentato di "1", quando lo stato del segnale all'ingresso CU passa da "0" a "1" e il valore di conteggio è minore di 999.
Conteggio all'indietro	Il valore del contatore viene diminuito di "1", quando lo stato del segnale all'ingresso CU passa da "0" a "1" e il valore di conteggio è maggiore di 0.
Imposta contatore	Quando lo stato del segnale all'ingresso S passa da "0" a "1", il contatore viene impostato con il valore dell'ingresso PV . Un tale cambio di segnale è sempre necessario per poter impostare un contatore.
Resetta	Il contatore viene resettato quando l'ingresso R = 1. Il reset del contatore imposta il valore di conteggio a "0".
Interrogazione contatore	Un'interrogazione dello stato del segnale all'uscita Q risulta "1" se il valore di conteggio è maggiore di "0". L'interrogazione risulta "0" se il valore di conteggio è uguale a "0".

13.1.8 Esempio di funzione di conteggio

Occupazione dei parametri

La tabella illustra l'occupazione dei parametri della funzione riportata nell'esempio S_CD.

Parametri	Descrizione
C_NO	MIOCONTATORE
CD	INGRESSO E0.0
S	IMPOSTARE
PV	PREIMPOSTAZIONE 16#0089
R	RESET
Q	A0.7
CV	VALORE BIN

Esempio

```

FUNCTION_BLOCK CONTEGGIO
VAR_INPUT
    MIOCONTATORE          : COUNTER ;
END_VAR
VAR_OUTPUT
    RISULTATO              : INT ;
END_VAR
VAR
    IMPOSTARE              : BOOL ;
    RESETTARE              : BOOL ;
    VALORE_BCD              : WORD ; // Stato contatore codice BCD
    VALORE_BIN              : WORD ; // Stato contatore binario
    PREIMPOSTAZIONE        : WORD ;
END_VAR
BEGIN
    A0.0                   := 1 ;
    IMPOSTARE               := E0.2 ;
    RESETTARE               := E0.3 ;
    PREIMPOSTAZIONE         := 16#0089 ;
    //Conteggio all'indietro
    VALORE_BCD := S_CD (C_NO := MIOCONTATORE,
        CD:= E0.0 ,
        S               := IMPOSTARE,
        PV:= PREIMPOSTAZIONE,
        R               := RESETTARE,
        CV:= VALORE_BIN,
        Q               := A0.7) ;
    //Ulteriore elaborazione come parametro d'uscita
    RISULTATO           := WORD_TO_INT (VALORE_BIN) ;
    AW4                 := VALORE_BCD ;
END_FUNCTION_BLOCK

```

13.2 Temporizzatori

13.2.1 Funzioni temporali

I temporizzatori sono elementi funzionali nel programma utente che eseguono e sorvegliano eventi comandati a tempo. STEP 7 mette a disposizione dell'utente una serie di funzionali temporali standard alle quali si può accedere tramite S7-SCL:

Funzione di temporizzazione	Significato
S_PULSE	Avvia temporizzatore come impulso
S_PEXT	Avvia temporizzatore come impulso prolungato
S_ODT	Avvia temporizzatore come ritardo all'inserzione
S_ODTS	Avvia temporizzatore come ritardo all'inserzione con memoria
S_OFFDT	Avvia temporizzatore come ritardo alla disinserzione

13.2.2 Richiamo di funzioni temporali

Le funzioni temporali vengono richiamate in modo analogo alle funzioni. L'identificazione di funzione può essere impiegata ovunque al posto di un operando in un'espressione, a condizione che il tipo di risultato della funzione sia compatibile con il tipo dell'operando sostituito.

Il valore della funzione (valori di ritorno) che viene restituito al punto di richiamo è un valore temporale con tipo di dati S5TIME.

Richiamo assoluto o dinamico

Nel richiamo si può specificare un valore assoluto come numero della funzione di temporizzazione (ad es. T_NO:=T10) con tipo di dati TIMER. Tale valore non è tuttavia modificabile durante l'esecuzione.

Al posto del numero assoluto, si può introdurre anche una variabile o costante con tipo di dati INT. Ciò presenta il vantaggio di poter eseguire un richiamo dinamico assegnando alla variabile o costante un altro numero assoluto ad ogni richiamo.

Un'altra possibilità di eseguire un richiamo dinamico consiste nell'indicare un variabile di tipo TIMER.

Esempi

```
//Esempio di richiamo assoluto:
CurrTime:=S_ODT (T_NO:=T10,
                S:=TRUE,
                TV:=T#1s,
                R:=FALSE,
                BI:=biVal,
                Q:=actFlag);

//Esempio di richiamo dinamico: ad ogni esecuzione di un
//loop FOR viene richiamata un'altra funzione di temporizzazione:
FUNCTION_BLOCK TEMPORIZZATORE
VAR_INPUT
    MIOTEMPORIZZATORE: ARRAY [1..4] of STRUCT
        T_NO: INT;
        TV  : WORD;
    END_STRUCT;
.
.
END_VAR
.
.
FOR I:= 1 TO 4 DO
    CurrTime:= S_ODT(T_NO:=MIOTEMPORIZZATORE[I].T_NO, S:=true, TV:=
    MIOTEMPORIZZATORE[I].TV);
END_FOR;

//Esempio di richiamo dinamico in caso
//di utilizzo di una variabile
//con tipo di dati TIMER:
FUNCTION_BLOCK TEMPORIZZATORE
VAR_INPUT
    miotemporizzatore:TIMER;
END_VAR
.
.
CurrTime:=S_ODT (T_NO:=miotemporizzatore,.....);
```

Nota

I nomi delle funzioni sono uguali nei mnemonici tedesco e inglese.

13.2.3 Assegnazione di parametri nelle funzioni temporali

La tabella seguente contiene un sommario dei parametri per funzioni temporali:

Parametri	Tipo di dati	Descrizione
T_NO	TIMER INTEGER	Identificazione di tempo; l'area dipende dalla CPU
S	BOOL	Ingresso di avvio
TV	S5TIME	Preimpostazione valore di tempo (formato BCD)
R	BOOL	Ingresso di reset
Q	BOOL	Stato del temporizzatore
BI	WORD	Valore di tempo residuo (binario)
RET_VAL	S5TIME	Valore del timer

Regole

Poiché i valore dei parametri sono memorizzati in modo globale, in determinati casi la loro indicazione è opzionale. Per l'assegnazione di parametri si devono osservare le seguenti regole generali:

- Il parametro per l'identificazione del timer T_NO deve essere alimentato durante il richiamo. Al richiamo invece del numero assoluto del temporizzatore (ad es. T10) si può indicare anche una variabile con tipo di dati INT o un parametro di ingresso con tipo di dati TIMER.
- L'indicazione dei parametri PV (valore di preimpostazione) e S (impostazione) può essere effettuata a coppie.
- Il valore del risultato in formato S5TIME è sempre il valore della funzione.

Esempio

```

FUNCTION_BLOCK FB2
VAR
    CurrTime      : S5time;
    BiVal         : word;
    ActFlag       : bool;
END_VAR

BEGIN
    CurrTime      :=S_ODT   (T_NO:= T10, S:=TRUE, TV:=T#1s, R:=FALSE,
                             BI:=biVal,Q:=actFlag);
    CurrTime      :=S_ODTS  (T_NO:= T11, S:=M0.0, TV:= T#1s, R:=M0.1,
                             BI:=biVal,Q:=actFlag);
    CurrTime      :=S_OFFDT (T_NO:= T12, S:=E0.1 & actFlag, TV:= T#1s,
                             R:=FALSE, BI:=biVal,Q:=actFlag);
    CurrTime      :=S_PEXT  (T_NO:= T13, S:=TRUE, TV:= T#1s, R:=E0.0,
                             BI:=biVal,Q:=actFlag);
    CurrTime      :=S_PULSE (T_NO:= T14, S:=TRUE, TV:= T#1s, R:=FALSE,
                             BI:=biVal,Q:=actFlag);
END_FUNCTION_BLOCK

```

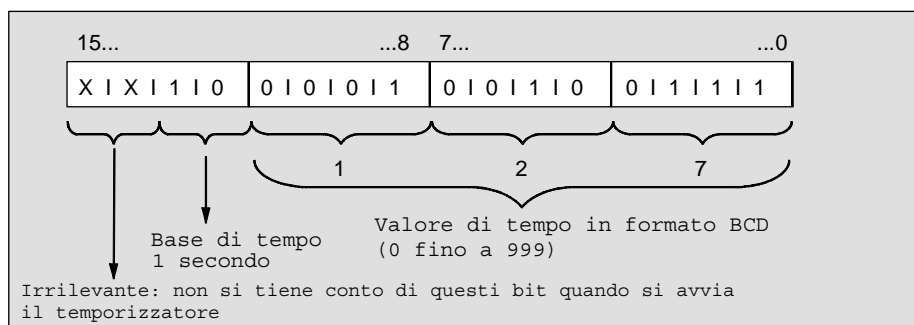
13.2.4 Introduzione ed analisi del valore temporale

Per l'introduzione del valore di preimpostazione o per l'analisi dei risultati della funzione in codice BCD è necessaria la rappresentazione interna del valore di tempo (vedere figura). Il valore del contatore ha un tipo di dati WORD, in cui i bit 0-11 contengono il valore di conteggio in formato BCD e i bit 12 e 13 contengono la base di temporizzazione. I bit 14 e 15 non sono rilevanti.

Ogni aggiornamento del tempo riduce il valore di tempo di un'unità in un intervallo che dipende dalla base del tempo. Il valore di tempo viene ridotto finché esso non sia uguale a "0". La gamma di tempo comprende da 0 fino a 9.990 secondi.

Formato

La figura seguente illustra la configurazione di bit del valore di tempo:



Ingresso

Con i seguenti formati si può caricare un valore di tempo predefinito:

- in rappresentazione a livelli
- in rappresentazione decimale

In entrambi i casi la base di temporizzazione viene scelta automaticamente e il valore viene arrotondato al numero immediatamente inferiore con questa base di temporizzazione.

Analisi

Il risultato può essere analizzato in due formati diversi:

- come risultato della funzione (tipo S5TIME): in formato BCD
- come parametro d'uscita (valore di tempo senza base di tempo del tipo WORD): binario

Base di temporizzazione per valori di tempo

Per l'introduzione e l'analisi del valore di tempo è necessaria la base di temporizzazione (bit 12 e 13 del parola del Timer). La base di temporizzazione definisce l'intervallo in cui il valore di tempo viene ridotto di un'unità (vedi tabella). La minima base di temporizzazione è 10 ms; la massima è 10 s.

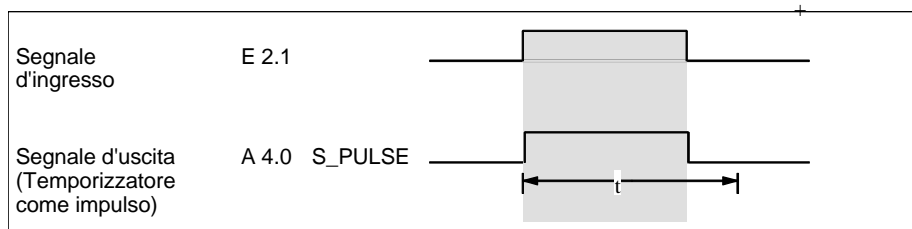
Base di temporizzazione	Codice binario per base di tempo
10 ms	00
100 ms	01
1 s	10
10 s	11

Nota

Poiché i valori di tempo vengono memorizzati solo in un determinato intervallo di tempo, i valori che non corrispondono ad un multiplo esatto dell'intervallo di tempo vengono tagliati. I valori la cui risoluzione è troppo alta per l'area desiderata vengono arrotondati in modo da raggiungere l'area desiderata, ma non la risoluzione desiderata.

13.2.5 Avvia temporizzatore come impulso (S_PULSE)

Il tempo massimo durante il quale il segnale d'uscita rimane ad "1" equivale al valore di tempo programmato. Se durante il tempo di esecuzione del temporizzatore sull'ingresso si verifica lo stato di segnale 0, il temporizzatore viene impostato ad "0". Ciò significa che il tempo di esecuzione termina prima del tempo previsto.



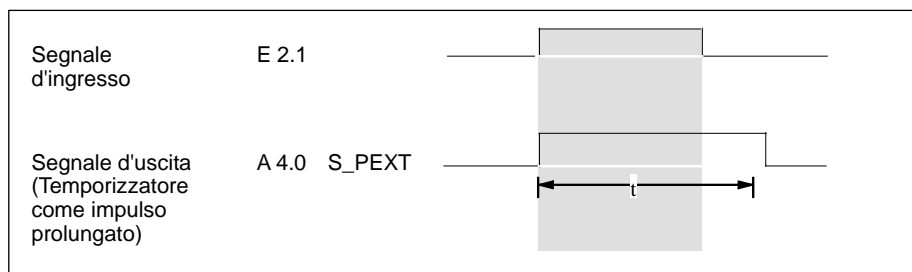
Modo di funzionamento

La tabella illustra il modo di funzionamento della funzione "Avvia temporizzatore come impulso":

Operazione	Modo di funzionamento
Avviare il tempo	L'operazione "Avvia temporizzatore come impulso" avvia un tempo indicato quando lo stato del segnale all'ingresso di avvio (S) passa da "0" a "1". Per abilitare il temporizzatore è sempre necessario un cambio del segnale.
Definisci il tempo di esecuzione	Il temporizzatore continua a scorrere con il valore indicato all'ingresso TV , fino allo scadere del tempo programmato e l'ingresso S = 1.
Fine del tempo di esecuzione prima del previsto	Se l'ingresso S passa da "1" a "0" prima che il valore del tempo sia scaduto, il temporizzatore viene interrotto.
Resetta	Il temporizzatore viene resettato quando l'ingresso di reset (R) passa da "0" a 1", durante lo scorrimento del temporizzatore. In seguito a questo cambio, anche il valore del tempo e la base di tempo vengono resettate a zero. Lo stato del segnale "1" all'ingresso R non è rilevante se il temporizzatore non è in funzione.
Interroga stato del segnale	Finché il temporizzatore è in funzione, un'interrogazione dello stato del segnale "1" all'ingresso Q fornisce il risultato "1". In caso di fine del tempo di esecuzione prima del tempo previsto, un'interrogazione dello stato del segnale all'ingresso Q fornisce il risultato "0".
Interroga valore del tempo attuale	Il valore del tempo attuale può essere interrogato all'uscita BI tramite il valore della funzione S_PULSE.

13.2.6 Avvia temporizzatore come impulso prolungato (S_PEXT)

Il segnale d'uscita per il tempo programmato (t) rimane su "1" indipendentemente dal tempo in cui il segnale d'ingresso rimane su "1". Una nuova risoluzione dell'impulso di avvio causando un nuovo scorrimento del tempo, ritarda anche l'impulso d'uscita (post-attivazione).



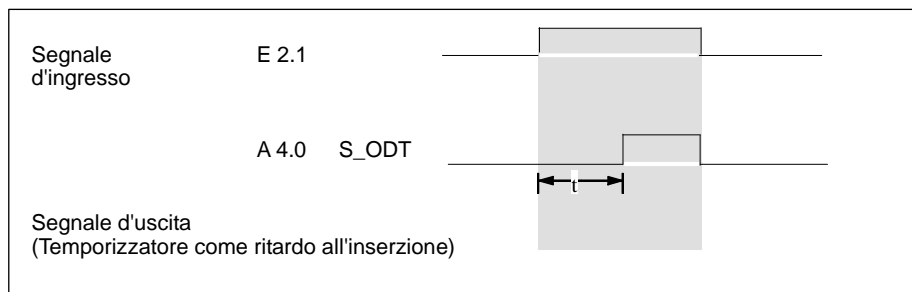
Modo di funzionamento

La tabella illustra il modo di funzionamento della funzione "Avvia temporizzatore come impulso prolungato":

Operazione	Modo di funzionamento
Avviare il tempo	L'operazione "Avvia temporizzatore come impulso prolungato" (S_PEXT) avvia un tempo indicato quando lo stato del segnale all'ingresso di avvio (S) passa da "0" a "1". Per abilitare il temporizzatore è sempre necessario un cambio del segnale.
Avvia nuovamente il tempo di esecuzione	Se lo stato del segnale all'ingresso S durante il tempo di esecuzione passa nuovamente a "1", il tempo viene avviato nuovamente con il valore del tempo indicato.
Preimposta tempo di esecuzione	Il temporizzatore continua a funzionare con il valore indicato all'ingresso TV , fino allo scadere del tempo programmato.
Resetta	Il temporizzatore viene resettato quando l'ingresso di reset (R) passa da "0" a "1", durante lo scorrimento del temporizzatore. In seguito a questo cambio, anche il valore del tempo e la base di tempo vengono resettate a zero. Lo stato del segnale "1" all'ingresso R non è rilevante se il temporizzatore non è in funzione.
Interroga stato del segnale	Finché il temporizzatore è in funzione, un'interrogazione dello stato del segnale "1" all'ingresso Q fornisce il risultato "1", indipendentemente dalla lunghezza del segnale d'ingresso.
Interroga valore del tempo attuale	Il valore del tempo attuale può essere interrogato all'uscita BI tramite il valore della funzione S_PEXT.

13.2.7 Avvia temporizzatore come ritardo all'inserzione (S_ODT)

Il segnale d'uscita passa da "0" a "1" quando il tempo programmato è scaduto e il segnale d'ingresso è ancora "1". vale a dire che l'uscita viene inserita con ritardo. I segnali d'ingresso la cui durata del tempo è più breve del programmato, non appaiono all'uscita.



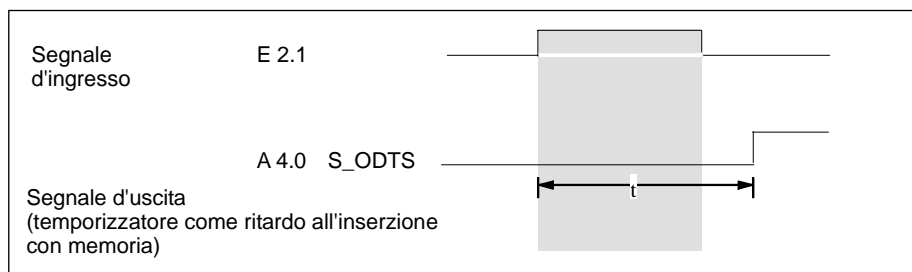
Modo di funzionamento

La tabella illustra il modo di funzionamento della funzione "Avvia temporizzatore come ritardo all'inserzione":

Operazione	Modo di funzionamento
Avviare il tempo	L'operazione "Avvia temporizzatore come ritardo all'inserzione" avvia un tempo indicato quando lo stato del segnale all'ingresso di avvio (S) passa da "0" a "1". Per abilitare il temporizzatore è sempre necessario un cambio del segnale.
Interrompi temporizzatore	Il tempo viene interrotto se lo stato del segnale all'ingresso S passa da "1" a "0" mentre il temporizzatore è in funzione.
Definisci il tempo di esecuzione	Il temporizzatore continua a scorrere con il valore indicato all'ingresso TV finché lo stato del segnale all'ingresso S = 1.
Resetta	Il temporizzatore viene resettato quando l'ingresso di reset (R) passa da "0" a "1", durante lo scorrimento del temporizzatore. In seguito a questo cambio, anche il valore del tempo e la base di tempo vengono resettate a zero. Lo stato del segnale "1" all'ingresso R non è rilevante se il temporizzatore non è in funzione.
Interroga stato del segnale	Un'interrogazione dello stato del segnale "1" all'uscita Q risulta "1" se il temporizzatore è scaduto senza errori e l'ingresso S è ancora "1". Se il temporizzatore è stato interrotto, un'interrogazione dello stato del segnale "1" risulta sempre "0". Un'interrogazione dello stato del segnale "1" all'uscita Q risulta sempre "0" anche se il temporizzatore non è in funzione e lo stato di segnale all'ingresso S è sempre "1".
Interroga valore del tempo attuale	Il valore del tempo attuale può essere interrogato all'uscita BI tramite il valore della funzione S_ODT.

13.2.8 Avvia temporizzatore come ritardo all'inserzione con memoria (S_ODTS)

Il segnale d'uscita passa da "0" a "1" allo scadere del temporizzatore programmato, indipendentemente dal tempo in cui il segnale d'ingresso rimane su "1".



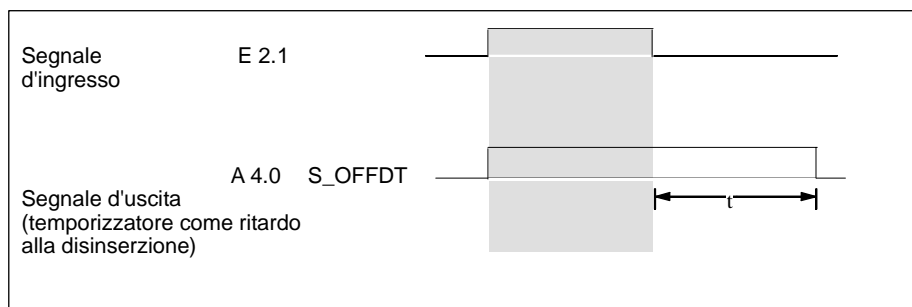
Modo di funzionamento

La tabella illustra il modo di funzionamento della funzione "Avvia temporizzatore come ritardo all'inserzione con memoria":

Operazione	Modo di funzionamento
Avviare il tempo	L'operazione "Avvia temporizzatore come ritardo all'inserzione con memoria" avvia un tempo indicato quando lo stato del segnale all'ingresso di avvio (S) passa da "0" a "1". Per abilitare il temporizzatore è sempre necessario un cambio del segnale.
Riavvia temporizzatore	Il temporizzatore viene riavviato con il valore indicato quando l'ingresso S passa da "0" a "1" mentre il temporizzatore è in funzione.
Definisci il tempo di esecuzione	Il temporizzatore continua a scorrere con il valore indicato all'ingresso TV , anche se lo stato del segnale all'ingresso S passa a "0" prima dello scadere del tempo.
Resetta	Se l'ingresso di reset (R) passa da "0" a "1", il temporizzatore viene resettato indipendentemente dal stato del segnale all'ingresso S .
Interroga stato del segnale	Un'interrogazione dello stato del segnale "1" all'uscita Q risulta "1" allo scadere del temporizzatore indipendentemente dallo stato del segnale all'ingresso S .
Interroga valore del tempo attuale	Il valore del tempo attuale può essere interrogato all'uscita BI tramite il valore della funzione S_ODTS.

13.2.9 Avvia temporizzatore come ritardo alla disinserzione (S_OFFDT)

In caso di passaggio dello stato del segnale da "0" a "1" all'ingresso S, all'uscita Q appare lo stato "1". Se all'ingresso di avvio lo stato passa da "1" a "0", il temporizzatore viene attivato. L'uscita del segnale assume lo stato "0" solo allo scadere del tempo. L'uscita viene avviata con ritardo all'inserzione.



Modo di funzionamento

La tabella illustra il modo di funzionamento della funzione "Avvia temporizzatore come ritardo alla disinserzione":

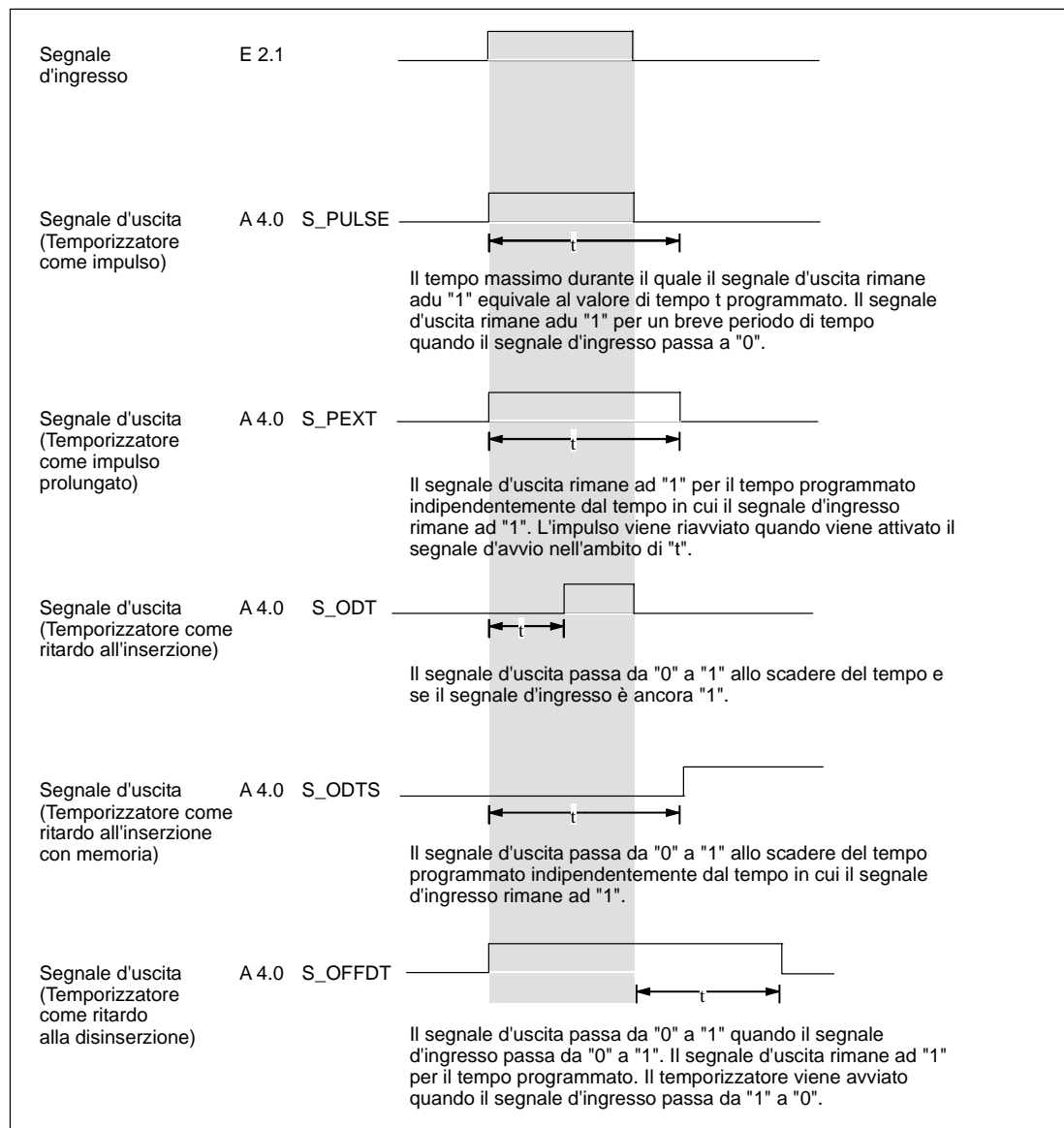
Operazione	Modo di funzionamento
Avviare il tempo	L'operazione "Avvia temporizzatore come ritardo alla disinserzione" avvia un tempo indicato quando lo stato del segnale all'ingresso di avvio (S) passa da "0" a "1". Per abilitare il temporizzatore è sempre necessario un cambio del segnale.
Riavvia temporizzatore	Il temporizzatore viene riavviato quando lo stato del segnale all'ingresso S passa nuovamente da "1" a "0" (p. es. dopo il reset).
Definisci il tempo di esecuzione	Il temporizzatore continua a scorrere con il valore indicato all'ingresso TV .
Resetta	Se l'ingresso di reset (R) passa da "0" a "1", mentre il temporizzatore è in funzione, il temporizzatore viene resettato.
Interroga stato del segnale	Un'interrogazione dello stato del segnale "1" all'uscita Q risulta "1", se lo stato del segnale all'ingresso S = 1 o se il temporizzatore è in funzione.
Interroga valore del tempo attuale	Il valore del tempo attuale può essere interrogato all'uscita BI tramite il valore della funzione S_OFFDT.

13.2.10 Esempio di funzione di temporizzazione

```
FUNCTION_BLOCK TEMPORIZZATORE
VAR_INPUT
    miotemporizzatore : TIMER ;
END_VAR
VAR_OUTPUT
    risultato : S5TIME ;
END_VAR
VAR
    impostare      : BOOL ;
    resettare      : BOOL ;
    valoreBcd      : S5TIME ; //Base di temporizzazione e valore
                                //residuo in codice BCD
    valoreBin      : WORD ; //Valore di tempo binario
    preimpostaz.   : S5TIME ;
END_VAR
BEGIN
    A0.0 := 1;
    preimpostaz. := E0.0 ;
    resettare := E0.1;
    preimpostaz. := T#25S ;
    valoreBcd := S_PEXT (T_NO := miotemporizzatore ,
        S := impostare ,
        TV := preimpostazione,
        R := resettare ,
        BI := valoreBin ,
        Q := A0.7) ;
    //Ulteriore elaborazione come parametro d'uscita
    risultato := valoreBcd ;
    //All'uscita per visualizzazione
    AW4 := valoreBin ;
END_FUNCTION_BLOCK
```

13.2.11 Selezione del temporizzatore giusto

La figura seguente offre una panoramica dei cinque temporizzatori descritti in questo paragrafo. Questa panoramica intende essere un aiuto per facilitare la scelta dei temporizzatori più adatti alle esigenze dell'utente.



14 Funzioni standard di S7-SCL

14.1 Funzioni di conversione dei tipi di dati

14.1.1 Conversione dei tipi di dati

Se si combinano due operandi in un'operazione si deve verificare che i rispettivi tipi di dati siano compatibili. Se gli operandi hanno tipo di dati diverso si deve effettuare una conversione. S7-SCL consente i seguenti tipi di conversione dei tipi di dati:

- Conversione implicita di tipi di dati

I tipi di dati sono suddivisi in classi all'interno delle quali S7-SCL effettua una conversione implicita. Le funzioni utilizzate dal compilatore per tale conversione sono riassunte nel capitolo "Funzioni di conversioni di classe A".

- Conversione esplicita dei tipi di dati

Negli operandi che non appartengono alla stessa classe è necessario richiamare esplicitamente una funzione di conversione. Per la conversione esplicita del tipo di dati S7-SCL mette a disposizione diverse funzioni standard suddivise nelle seguenti classi:

- Funzioni di conversione di classe B
- Funzioni per arrotondamento e taglio

14.1.2 Conversione implicita del tipo di dati

Nell'ambito delle classi di tipi di dati definite nella tabella, il compilatore esegue una conversione implicita dei tipi di dati in base alla sequenza indicata. Come formato comune di due operandi viene definito il più grande tra i due tipi di dati: per es. il formato comune di BYTE e WORD - WORD.

Si prega di osservare che in una conversione di tipi di dati nell'ambito della classe ANY_BIT i bit iniziali vengono impostati su 0.

Classi	Sequenza di conversione
ANY_BIT	BOOL > BYTE > WORD > DWORD
ANY_NUM	INT > DINT > REAL

Esempio di conversione implicita del tipo di dati

```
VAR
    REGOLATORE_PID_1 : BYTE ;
    REGOLATORE_PID_2 : WORD ;
END_VAR
BEGIN
    IF (REGOLATORE_PID_1 <> REGOLATORE_PID_2) THEN ...
    (* nella precedente istruzione IF il REGOLATORE_PID_1 viene
    implicitamente convertito da BYTE in WORD. *)
```

14.1.2.1 Funzioni di conversione di classe A

La tabella illustra le funzioni di conversione di tipi di dati della classe A. Queste funzioni vengono attivate implicitamente dal compilatore, ma possono anche essere attivate esplicitamente dall'utente. Il risultato è sempre definito.

Nome della funzione	Regola di conversione
BOOL_TO_BYTE	Integrazione con zeri iniziali
BOOL_TO_DWORD	Integrazione con zeri iniziali
BOOL_TO_WORD	Integrazione con zeri iniziali
BYTE_TO_DWORD	Integrazione con zeri iniziali
BYTE_TO_WORD	Integrazione con zeri iniziali
CHAR_TO_STRING	Trasformazione in una stringa (di lunghezza 1) contenente lo stesso carattere.
DINT_TO_REAL	Trasformazione in REAL secondo la norma IEEE. Il valore può essere soggetto a modifiche a causa della diversa precisione di REAL.
INT_TO_DINT	La parola più significativa del valore della funzione viene riempita con 16#FFFF in caso di parametro d'ingresso negativo, negli altri casi viene riempita con zeri. Il valore rimane lo stesso.
INT_TO_REAL	Trasformazione in REAL secondo la norma IEEE. Il valore rimane lo stesso.
WORD_TO_DWORD	Integrazione con zeri iniziali

14.1.3 Funzioni standard per la conversione esplicita di tipi di dati

La descrizione generale del richiamo di funzioni viene descritta sotto "Richiamo di funzioni".

Durante il richiamo delle funzioni di conversione si deve osservare quanto segue:

- Parametri d'ingresso:
Ogni funzione per la conversione del tipo di dati possiede esattamente un parametro d'ingresso con il nome IN. Poiché si tratta di una funzione con un solo parametro, quest'ultimo non deve essere indicato.
- Valore della funzione
Il risultato è sempre il valore della funzione.
- Assegnazione di nomi
Poiché i tipi di dati del parametro d'ingresso e del valore della funzione risultano dal rispettivo nome della funzione, essi non sono elencati nei sommari (classe A e classe B):
p. es. nella funzione BOOL_TO_BYTE, il tipo di dati del parametro d'ingresso è BOOL e il tipo di dati del valore della funzione è BYTE.

14.1.3.1 Funzioni di conversione di classe B

La tabella illustra le funzioni di conversione di tipi di dati della classe B. Queste funzioni devono essere attivate esplicitamente dall'utente. Il risultato può anche essere indefinito se la dimensione del tipo di dati di destinazione non è sufficiente.

L'utente può controllare da sé un caso del genere attivando il controllo di valori limite oppure può far eseguire il controllo dal sistema selezionando l'opzione "flag OK" prima della compilazione. Nei casi in cui il risultato è indefinito, il sistema imposta su FALSE il flag OK.

Nome della funzione	Regola di conversione	OK
BOOL_TO_INT	WORD_TO_INT(BOOL_TO_WORD(x))	N
BOOL_TO_DINT	DWORD_TO_DINT(BOOL_TO_DWORD(x))	N
BYTE_TO_BOOL	Copiatura del bit meno significativo	S
BYTE_TO_CHAR	Immissione della stringa di bit	N
BYTE_TO_INT	WORD_TO_INT(BYTE_TO_WORD(x))	N
BYTE_TO_DINT	DWORD_TO_DINT(BYTE_TO_DWORD(x))	N
CHAR_TO_BYTE	Immissione della stringa di bit	N
CHAR_TO_INT	La stringa di bit presente nel parametro d'ingresso viene registrata nel byte meno significativo del valore della funzione. Il byte più significativo viene riempito con zeri.	N
DATE_TO_DINT	Immissione della stringa di bit	N
DINT_TO_DATE	Immissione della stringa di bit	S
DINT_TO_DWORD	Immissione della stringa di bit	N
DINT_TO_INT	Copiatura dei bit per il carattere. Il valore presente nel parametro d'ingresso viene interpretato nel tipo di dati INT. Se il valore è minore di -32_768 o maggiore di 32_767, la variabile OK viene impostata su FALSE.	S
DINT_TO_TIME	Immissione della stringa di bit	N
DINT_TO_TOD	Immissione della stringa di bit	S
DINT_TO_BOOL	DWORD_TO_BOOL(DINT_TO_DWORD(x))	S
DINT_TO_BYTE	DWORD_TO_BYTE(DINT_TO_DWORD(x))	S
DINT_TO_STRING	DI_STRNG	N
DINT_TO_WORD	DWORD_TO_WORD(DINT_TO_DWORD(x))	S

Nome della funzione	Regola di conversione	OK
DWORD_TO_BOOL	Copiatura del bit meno significativo	S
DWORD_TO_BYTE	Copiatura degli 8 bit meno significativi	S
DWORD_TO_DINT	Immissione della stringa di bit	N
DWORD_TO_REAL	Immissione della stringa di bit	N
DWORD_TO_WORD	Copiatura dei 16 bit meno significativi	S
DWORD_TO_INT	DINT_TO_INT (DWORD_TO_DINT(x))	S
INT_TO_CHAR	Immissione della stringa di bit	S
INT_TO_WORD	Immissione della stringa di bit	N
INT_TO_BOOL	WORD_TO_BOOL(INT_TO_WORD(x))	S
INT_TO_BYTE	WORD_TO_BYTE(INT_TO_WORD(x))	S
INT_TO_DWORD	WORD_TO_DWORD(INT_TO_WORD(x))	N
INT_TO_STRING	I_STRNG(x)	N
REAL_TO_DINT	Arrotondamento del valore IEEE-REAL a DINT. Se il valore è minore di -2_147_483_648 o maggiore di 2_147_483_647, la variabile OK viene impostata su FALSE.	S
REAL_TO_DWORD	Immissione della stringa di bit	N
REAL_TO_INT	Arrotondamento del valore IEEE-REAL a INT. Se il valore è minore di -32_768 o maggiore di 32_767, la variabile OK viene impostata su FALSE.	S
REAL_TO_STRING	R_STRNG(x)	N
STRING_TO_CHAR	Copiatura del primo carattere della stringa. Se la STRING non ha una lunghezza di 1, la variabile OK viene impostata su FALSE.	S
STRING_TO_INT	STRNG_I(x)	N
STRING_TO_DINT	STRNG_DI(x)	N
STRING_TO_REAL	STRNG_R(x)	N
TIME_TO_DINT	Immissione della stringa di bit	N
TOD_TO_DINT	Immissione della stringa di bit	N
WORD_TO_BOOL	Copiatura del bit meno significativo	S
WORD_TO_BYTE	Copiatura degli 8 bit meno significativi	S
WORD_TO_INT	Immissione della stringa di bit	N
WORD_TO_DINT	INT_TO_DINT(WORD_TO_INT(x))	N
WORD_TO_BLOCK_DB	La stringa di bit di WORD viene interpretata come numero di blocco dati.	N
BLOCK_DB_TO_WORD	Il numero di blocco dati viene interpretato da WORD come stringa di bit.	N
BCD_TO_INT(x) WORD_BCD_TO_INT(x)	L'espressione x è di tipo WORD e viene assunta come valore codificato BCD collocato tra -999 e +999. Il risultato si trova dopo la conversione come numero intero (rappresentazione binaria) di tipo INT. Se durante la conversione si verifica un errore il sistema di automazione si imposta su STOP. È possibile elaborare la causa che ha portato allo stop in OB121.	N
INT_TO_BCD(x) INT_TO_BCD_WORD(x)	L'espressione x è di tipo INT e viene assunta come numero intero avente valore individuato tra -999 e +999. Il risultato si trova dopo la conversione come numero codificato BCD di tipo WORD. Al di fuori del campo di valori il risultato è indefinito. Selezionando l'opzione "Imposta flag OK" il flag OK ottiene in questo caso il valore FALSE.	S

Nome della funzione	Regola di conversione	OK
BCD_TO_DINT(x) DWORD_BCD_TO_DINT(x)	L'espressione x è di tipo DWORD e viene assunta come valore codificato BCD collocato tra -9999999 e +9999999. Il risultato si trova dopo la conversione come numero intero (rappresentazione binaria) di tipo DINT. Se durante la conversione si verifica un errore il sistema di automazione si imposta su STOP. È possibile elaborare la causa che ha portato allo stop in OB121.	N
DINT_TO_BCD(x) DINT_TO_BCD_DWORD(x)	L'espressione x è di tipo DINT e viene assunta come numero intero avente valore individuato tra -9999999 e +9999999. Il risultato si trova dopo la conversione come numero codificato BCD di tipo DWORD. Al di fuori dell'area di valori il risultato è indefinito. Selezionando l'opzione "Imposta flag OK" il flag OK ottiene in questo caso il valore FALSE.	S

Attenzione

Se una costante di un tipo di dati più significativo viene convertita in un tipo meno significativo, durante la compilazione viene registrato un messaggio di errore in caso la costante si trovi al di fuori del campo del tipo di dati meno significativo.

Esempi:

```
M0.0 :=WORD_TO_BOOL(W#16#FFFF);
MW0  :=DINT TO INT(35000);
```

Nota

Si ha inoltre la possibilità di sfruttare altre funzioni IEC per la conversione dei tipi di dati. Informazioni relative a tali funzioni possono essere reperite nel manuale di riferimento di STEP 7 "Funzioni standard e di sistema S7-300/400".

14.1.3.2 Funzioni per arrotondare e tagliare

Le conversioni di tipi di dati comprendono anche le funzioni per arrotondare e tagliare dei numeri. La tabella illustra i nomi, i tipi di dati (per il parametro d'ingresso e il valore della funzione) e i compiti di queste funzioni:

Nome della funzione	Tipo di dati parametri d'ingresso	Tipo di dati valore della funzione	Compito
ROUND	REAL	DINT	Arrotondamento (generazione di un numero DINT) Secondo DIN EN 61131-3 viene arrotondato sempre al successivo valore intero pari, ovvero sia 1,5 che 2,5 vengono arrotondati a 2.
TRUNC	REAL	DINT	Taglio (generazione di un numero DINT)

Nota

Si ha inoltre la possibilità di sfruttare altre funzioni IEC per la conversione dei tipi di dati. Informazioni su tali funzioni possono essere reperite nel manuale di riferimento di STEP 7 "Funzioni di sistema e standard S7-300/400".

Esempio

```
// Qui viene arrotondato per difetto (Risultato: 3)
    ROUND (3.14) ;

// Qui viene arrotondato per eccesso (Risultato: 4)
    ROUND (3.56) ;

// Qui viene tagliato (Risultato: 3)
    TRUNC (3.14) ;

// Qui viene tagliato (Risultato: 3)
    TRUNC (3.56) ;
```

14.1.3.3 Esempi di conversione con le funzioni standard

Nell'esempio seguente è necessaria una conversione esplicita poiché il tipo di dati di destinazione è meno importante del tipo di dati di origine.

```
FUNCTION_BLOCK FB10
VAR
    INTERRUTTORE : INT;
    REGOLATORE   : DINT;
END_VAR

(* INT è meno importante di DINT *)
INTERRUTTORE := DINT_TO_INT (REGOLATORE) ;
// . . .
END_FUNCTION_BLOCK
```

Nell'esempio seguente viene applicata una conversione esplicita visto che il tipo di dati REAL non è consentito per un'espressione aritmetica con operazione MOD.

```
FUNCTION_BLOCK FB20
VAR
    INTERRUTTORE : REAL
    VALOREINT     : INT := 17;
    CONV2         : INT ;
END_VAR

(* MOD può essere utilizzato solo per i dati di tipo INT o DINT *)
CONV2 := VALOREINT MOD REAL_TO_INT (INTERRUTTORE);
// . . .
END_FUNCTION_BLOCK
```

Nell'esempio seguente è necessaria una conversione poiché non è presente il tipo di dati corretto per un'operazione logica. È consentito utilizzare l'operazione NOT solo con dati del tipo BOOL, BYTE, WORD o DWORD.

```
FUNCTION_BLOCK FB30
VAR
    VALOREINT : INT := 17;
    CONV1     : WORD ;
END_VAR

(* NOT non deve essere utilizzato per i dati di tipo INT *)
CONV1 := NOT INT_TO_WORD(VALOREINT);
// . . .
END_FUNCTION_BLOCK
```

L'esempio seguente illustra la conversione nel caso di operazioni di ingresso e uscita in periferia.

```
FUNCTION_BLOCK FB40
VAR
    Raggio_on      : WORD ;
    Raggio          : INT;
END_VAR

    Raggio_on      := %EB0;
    Raggio         := WORD_TO_INT (raggio_on);
(* conversione in caso di passaggio in un'altra classe di dati. Il
valore deriva dall'ingresso e viene convertito per eseguire
ulteriori calcoli.*)

    Raggio         := raggio (superficie:= daticerchio.superficie)
    %AB0          :=WORD_TO_BYTE (INT_TO_WORD(RAGGIO));
(*Il raggio viene ricavato dalla superficie e fornito come numero
intero. Per l'uscita, il valore viene dapprima convertito in
un'altra classe di dati (INT_TO_WORD) e quindi nuovamente convertito
in un tipo meno importante (WORD_TO_BYTE).*)
// . . .
END_FUNCTION_BLOCK
```

14.2 Funzioni aritmetiche standard

14.2.1 Funzioni aritmetiche standard generiche

Si tratta di funzioni per il calcolo del valore assoluto, del quadrato o della radice quadrata di un valore.

Il tipo di dati ANY_NUM indica INT, DINT o REAL. Prestare anche attenzione al fatto che i parametri di ingresso del tipo INT o DINT vengono convertiti internamente in variabili REAL, se il valore della funzione è di tipo REAL.

Nome della funzione	Tipo di dati parametri d'ingresso	Tipo di dati valore della funzione	Descrizione
ABS	ANY_NUM	ANY_NUM	Valore assoluto
SQR	ANY_NUM	REAL	Quadrato
SQRT	ANY_NUM	REAL	Radice

Nota

Si ha inoltre la possibilità di sfruttare altre funzioni IEC per la conversione dei tipi di dati. Informazioni relative a tali funzioni possono essere reperite nel manuale di riferimento di STEP 7 "Funzioni di sistema e standard S7-300/400".

14.2.2 Funzioni logaritmiche

Si tratta di funzioni per il calcolo di un valore esponenziale o del logaritmo di un valore.

Il tipo di dati ANY_NUM indica INT, DINT o REAL. Si osservi che i parametri d'ingresso del tipo ANY_NUM vengono internamente convertiti in variabili REAL.

Nome della funzione	Tipo di dati parametri d'ingresso	Tipo di dati valore della funzione	Descrizione
EXP	ANY_NUM	REAL	e elevato IN
EXPD	ANY_NUM	REAL	10 elevato IN
LN	ANY_NUM	REAL	Logaritmo naturale
LOG	ANY_NUM	REAL	Logaritmo a base 10

Nota

Si ha inoltre la possibilità di sfruttare altre funzioni IEC per la conversione dei tipi di dati. Informazioni relative a tali funzioni possono essere reperite nel manuale di riferimento di STEP 7 "Funzioni di sistema e standard S7- 300/400".

14.2.3 Funzioni trigonometriche

Le funzioni trigonometriche rappresentate nella tabella presuppongono e calcolano dimensioni in angoli e archi.

Il tipo di dati ANY_NUM indica INT, DINT o REAL. Si osservi che i parametri d'ingresso del tipo ANY_NUM vengono internamente convertiti in variabili REAL.

Nome della funzione	Tipo di dati parametri d'ingresso	Tipo di dati valore della funzione	Descrizione
ACOS	ANY_NUM	REAL	Arco-Coseno
ASIN	ANY_NUM	REAL	Arco-Seno
ATAN	ANY_NUM	REAL	Arco-Tangente
COS	ANY_NUM	REAL	Coseno
SIN	ANY_NUM	REAL	Seno
TAN	ANY_NUM	REAL	Tangente

Nota

Si ha inoltre la possibilità di sfruttare altre funzioni IEC per la conversione dei tipi di dati. Informazioni relative a tali funzioni possono essere reperite nel manuale di riferimento di STEP 7 "Funzioni di sistema e standard S7-300/400".

14.2.4 Esempi di funzioni numeriche standard

Richiamo	Risultato
RISULTATO := ABS (-5) ;	// 5
RISULTATO := SQRT (81.0);	// 9
RISULTATO := SQR (23);	// 529
RISULTATO := EXP (4.1);	// 60.340 ...
RISULTATO := EXPD (3);	// 1_000
RISULTATO := LN (2.718 281) ;	// 1
RISULTATO := LOG (245);	// 2.389_166 ...
PI := 3.141592;	
RISULTATO := SIN (PI / 6) ;	// 0.5
RISULTATO := ACOS (0.5);	// 1.047_197 (=PI / 3)

14.3 Funzioni standard su stringhe di bit

Ogni funzione standard di stringa di bit possiede due parametri d'ingresso, identificati da IN o N. Il risultato è sempre il valore della funzione. La tabella seguente illustra i nomi delle funzioni e i tipi dei dati dei due parametri d'ingresso e del valore della funzione. Essi significano:

- parametro d'ingresso IN: buffer in cui vengono eseguite le operazioni di scorrimento bit. Il tipo di dati del parametro di ingresso determina il tipo di dati del valore della funzione.
- parametro d'ingresso N: numero delle rotazioni per le funzioni del buffer circolare ROL e ROR e numero delle posizioni di bit da far scorrere in SHL e SHR.

La tabella illustra le possibili funzioni standard su stringhe di bit:

Nome della funzione	Tipo di dati parametri d'ingresso IN	Tipo di dati parametri d'ingresso N	Tipo di dati valore della funzione	Compito
ROL	BOOL BYTE WORD DWORD	INT INT INT INT	BOOL BYTE WORD DWORD	Il valore presente nel parametro IN viene fatto ruotare verso sinistra di un numero di posizioni di bit indicato nel contenuto del parametro N.
ROR	BOOL BYTE WORD DWORD	INT INT INT INT	BOOL BYTE WORD DWORD	Il valore presente nel parametro IN viene fatto ruotare verso destra di un numero di posizioni di bit indicato nel contenuto del parametro N.
SHL	BOOL BYTE WORD DWORD	INT INT INT INT	BOOL BYTE WORD DWORD	Il valore presente nel parametro IN viene fatto scorrere verso sinistra mentre sul lato destro i bit vengono sostituiti con 0 conformemente al contenuto del parametro N.
SHR	BOOL BYTE WORD DWORD	INT INT INT INT	BOOL BYTE WORD DWORD	Il valore presente nel parametro IN viene fatto scorrere verso destra mentre sul lato sinistro i bit vengono sostituiti con 0 conformemente al contenuto del parametro N.

Nota

Si ha inoltre la possibilità di sfruttare altre funzioni IEC per la conversione dei tipi di dati. Informazioni relative a tali funzioni possono essere reperite nel manuale di riferimento di STEP 7 "Funzioni di sistema e standard S7-300/400".

14.3.1 Esempi di funzioni standard su stringhe di bit

Richiamo	Risultato
RISULTATO := ROL (IN:=BYTE#2#1101_0011, N:=5);	// 2#0111_1010 // (= 122 decimale)
RISULTATO := ROR (IN:=BYTE#2#1101_0011, N:=2);	// 2#1111_0100 // (= 244 decimale)
RISULTATO := SHL (IN:=BYTE#2#1101_0011, N:=3);	// 2#1001_1000 // (= 152 decimale)
RISULTATO := SHR (IN:=BYTE#2#1101_0011, N:=2);	// 2#0011_0100 // (= 52 decimale)

14.4 Funzioni per l'elaborazione di stringe

14.4.1 Funzioni per la gestione delle stringhe

LEN

La funzione LEN (FC 21) fornisce come valore di ritorno la lunghezza attuale della stringa di caratteri (numero dei caratteri validi). Una stringa vuota (") ha lunghezza pari a zero. La funzione non segnala errori.

Esempio: `LEN (S:= 'XYZ')`

Parametri	Dichiarazione	Tipo di dati	Area di memoria	Descrizione
S	INPUT	STRING	D, L	Variabile di ingresso in formato STRING
Valore di ritorno		INT	E, A, M, D, L	Numero di caratteri attuali

CONCAT

La funzione CONCAT riunisce un massimo di 32 variabili STRING in una stringa di caratteri. Se la stringa di caratteri risultante è più lunga della variabile creata nel parametro di uscita viene limitata alla lunghezza massima. Utilizzando la funzione CONCAT di S7-SCL, viene richiamato implicitamente FC2 dalla biblioteca "Funzioni IEC".

Esempio: `CONCAT (IN1:= 'Valvola', IN2:= 'Aperta')`

Parametri	Dichiarazione	Tipo di dati	Area di memoria	Descrizione
IN	INPUT	STRING CHAR	D, L	Variabile di ingresso in formato STRING o CHAR
IN2	INPUT	STRING CHAR	D, L	Variabile di ingresso in formato STRING o CHAR
INn	INPUT	STRING CHAR	D, L	Variabile di ingresso in formato STRING o CHAR
Valore di ritorno		STRING CHAR	D, L	Stringa di caratteri concatenata

LEFT o RIGHT

Le funzioni LEFT e RIGHT (FC 20 e FC 32) forniscono i primi e gli ultimi caratteri L di una stringa. Se L è maggiore della lunghezza attuale delle variabili STRING viene fornita la stringa completa. Se L = 0 viene fornita una stringa vuota. Se L è negativo viene emessa una stringa vuota.

Esempio: LEFT (IN:= 'Valvola', L:= 4)

Parametri	Dichiarazione	Tipo di dati	Area di memoria	Descrizione
IN	INPUT	STRING	D, L	Variabile di ingresso in formato STRING
L	INPUT	INT	E, A, M, D, L, cost.	Lunghezza della stringa di caratteri sinistra
Valore di ritorno		STRING	D, L	Variabile di uscita in formato STRING

MID

La funzione MID (FC 26) fornisce una parte di una stringa di caratteri. L è la lunghezza della stringa che deve essere letta, P è la posizione del primo carattere da leggere.

Se la somma di L e (P-1) supera la lunghezza attuale della variabile STRING, viene fornita una stringa che si estende dal carattere P-esimo fino alla fine del valore di ingresso. In tutti gli altri casi (P non è compreso nella lunghezza attuale, P e/o L sono pari a zero o negativi) viene emessa una stringa vuota.

Esempio: MID (IN:= 'Temperatura', L:= 2, P:= 3)

Parametri	Dichiarazione	Tipo di dati	Area di memoria	Descrizione
IN	INPUT	STRING	D, L	Variabile di ingresso in formato STRING
L	INPUT	INT	E, A, M, D, L, cost.	Lunghezza della stringa di caratteri centrale
P	INPUT	INT	E, A, M, D, L, cost.	Posizione del primo carattere
Valore di ritorno		STRING	D, L	Variabile di uscita in formato STRING

INSERT

La funzione INSERT (FC 17) inserisce una stringa di caratteri del parametro IN2 nella stringa del parametro IN1 dopo il P-esimo carattere. Se P è uguale a zero, la seconda stringa viene inserita davanti alla prima. Se P è maggiore della lunghezza attuale della prima stringa di caratteri, la seconda stringa viene inserita dopo la prima. Se P è negativo viene emessa una stringa vuota. Se la stringa risultante è più lunga della variabile indicata nel parametro di uscita; la stringa risultante viene limitata alla lunghezza massima impostata.

Esempio: INSERT (IN1:= 'Partecipante presente', IN2:='Martini',
P:= 11)

Parametri	Dichiarazione	Tipo di dati	Area di memoria	Descrizione
IN1	INPUT	STRING	D, L	Variabile STRING in cui viene effettuato l'inserimento
IN2	INPUT	STRING	D, L	Variabile STRING da inserire
P	INPUT	INT	E, A, M, D, L, cost.	Posizione di inserimento
Valore di ritorno		STRING	D, L	Stringa di caratteri risultante

DELETE

La funzione DELETE (FC 4) cancella dei caratteri da una stringa L a partire dal P-esimo carattere (compreso). Se L e/o P sono pari a zero o P è maggiore della lunghezza attuale della stringa di caratteri di ingresso, viene restituita la stringa di caratteri di ingresso. Se la somma di L e P è maggiore della stringa di caratteri di ingresso vengono cancellati tutti i caratteri fino alla fine della stringa. Se L e/o P sono negativi viene emessa una stringa vuota.

Esempio: DELETE (IN:= 'Temperatura ok', L:= 6, P:= 5)

Parametri	Dichiarazione	Tipo di dati	Area di memoria	Descrizione
IN	INPUT	STRING	D, L	Variabile STRING in cui vengono cancellati i caratteri
L	INPUT	INT	E, A, M, D, L, KONST	Caratteri da cancellare
P	INPUT	INT	E, A, M, D, L, KONST	Posizione del primo carattere da cancellare
Valore di ritorno		STRING	D, L	Stringa di caratteri risultante

REPLACE

La funzione REPLACE (FC 31) sostituisce con la seconda stringa (IN2) L i caratteri della prima stringa (IN1) a partire dal P-esimo carattere (compreso). Se L è pari a zero, viene restituita la prima stringa. Se P è uguale a zero o a uno la sostituzione inizia dal primo carattere (compreso). Se P non si trova nella prima stringa la seconda stringa viene unita alla prima. Se L e/o P sono negativi viene emessa una stringa vuota. Se la stringa risultante è più lunga della variabile indicata nel parametro di uscita; la stringa risultante viene limitata alla lunghezza massima impostata.

Esempio: REPLACE (IN1:= 'Temperatura', IN2:= 'alta', L:= 6, P:= 5)

Parametri	Dichiarazione	Tipo di dati	Area di memoria	Descrizione
IN1	INPUT	STRING	D, L	Variabile STRING in cui viene effettuata la sostituzione
IN2	INPUT	STRING	D, L	Variabile STRING da utilizzare per la sostituzione
L	INPUT	INT	E, A, M, D, L, cost.	Numero di caratteri da sostituire
P	INPUT	INT	E, A, M, D, L, cost.	Posizione del primo carattere sostituito
Valore di ritorno		STRING	D, L	Stringa di caratteri risultante

FIND

La funzione FIND (FC 11) fornisce la posizione della seconda stringa (IN2) all'interno della prima (IN1). La ricerca inizia da sinistra e viene segnalata la prima stringa individuata. Se la seconda stringa non è presente nella prima viene segnalato zero. La funzione non segnala errori.

Esempio: FIND (IN1:= 'Stazione di elaborazione', IN2:='Stazione')

Parametri	Dichiarazione	Tipo di dati	Area di memoria	Descrizione
IN1	INPUT	STRING	D, L	Variabile STRING in cui viene effettuata la ricerca
IN2	INPUT	STRING	D, L	Variabile STRING da ricercare
Valore di ritorno		INT	E, A, M, D, L	Posizione della prima stringa trovata

14.4.2 Funzioni per il confronto delle stringhe

I confronti in cui le stringhe vengono utilizzate come operandi sono possibili con le operazioni di confronto S7-SCL ==, <>, <, >, <= e >=. Il compilatore richiama automaticamente la funzione corrispondente. Le seguenti funzioni sono elencate solo per completezza.

EQ_STRNG e NE_STRNG

La funzione EQ_STRNG (FC 10) o (FC 29) esegue un confronto di uguale (FC10) o diverso (FC29) fra il contenuto di due variabili in formato STRING ed emette il risultato sotto forma di valore di ritorno. Quest'ultimo ha lo stato "1" se la stringa del parametro S1 è uguale/diversa dalla stringa del parametro S2. Questa funzione non segnala errori.

Parametri	Dichiarazione	Tipo di dati	Area di memoria	Descrizione
S1	INPUT	STRING	D, L	Variabile di ingresso in formato STRING
S2	INPUT	STRING	D, L	Variabile di ingresso in formato STRING
Valore di ritorno		BOOL	E, A, M, D, L	Risultato del confronto

GE_STRNG e LE_STRNG

La funzione GE_STRNG (FC 13) e (FC 19) esegue un confronto di maggiore/inferiore o uguale tra il contenuto di due variabili in formato STRING ed emette il risultato sotto forma di valore di ritorno. Quest'ultimo ha lo stato "1" se la stringa del parametro S1 è maggiore/inferiore o uguale alla stringa del parametro S2. I caratteri vengono confrontati da sinistra verso destra in relazione alla loro codifica ASCII (ad es. 'a' maggiore di 'A'). Il primo carattere diverso determina il risultato del confronto. Se la parte sinistra della stringa più lunga è uguale alla stringa più corta, la stringa più lunga viene considerata maggiore. Questa funzione non segnala errori.

Parametri	Dichiarazione	Tipo di dati	Area di memoria	Descrizione
S1	INPUT	STRING	D, L	Variabile di ingresso in formato STRING
S2	INPUT	STRING	D, L	Variabile di ingresso in formato STRING
Valore di ritorno		BOOL	E, A, M, D, L	Risultato del confronto

GT_STRNG e LT_STRNG

La funzione GT_STRNG (FC 15) e (FC 24) effettua un confronto di maggiore/inferiore fra il contenuto di due variabili in formato STRING ed emette il risultato sotto forma di valore di ritorno. Quest'ultimo ha lo stato "1" se la stringa del parametro S1 è maggiore/inferiore alla stringa del parametro S2. I caratteri vengono confrontati da sinistra verso destra in relazione alla loro codifica ASCII (ad es. 'a' maggiore di 'A'). Il primo carattere diverso determina il risultato del confronto. Se la parte sinistra della stringa più lunga è uguale alla stringa più corta, la stringa più lunga viene considerata maggiore. Questa funzione non segnala errori.

Parametri	Dichiarazione	Tipo di dati	Area di memoria	Descrizione
S1	INPUT	STRING	D, L	Variabile di ingresso in formato STRING
S2	INPUT	STRING	D, L	Variabile di ingresso in formato STRING
Valore di ritorno		BOOL	E, A, M, D, L	Risultato del confronto

14.4.3 Funzioni per la conversione del formato dei dati

INT_TO_STRING e STRING_TO_INT

Le funzioni INT_TO_STRING und STRING_TO_INT convertono una variabile in formato INT in una stringa di caratteri e viceversa. In modo implicito si utilizzano le funzioni I_STRNG (FC16) o STRNG_I (FC38) dalla biblioteca "Funzioni IEC" fornita. La stringa è preceduta da un carattere introduttivo. Se la variabile specificata nel parametro di ritorno è troppo corta, la conversione non viene effettuata.

Parametri	Dichiarazione	Tipo di dati	Area di memoria	Descrizione
INT_TO_STRING				
I	INPUT	INT	E, A, M, D, L, cost.	Valore di ingresso
Valore di ritorno		STRING	D, L	Stringa di caratteri risultante
STRING_TO_INT				
S	INPUT	STRING	D, L	Stringa di caratteri di ingresso
Valore di ritorno		INT	E, A, M, D, L	Risultato

DINT_TO_STRING e STRING_TO_DINT

Le funzioni DINT_TO_STRING und STRING_TO_DINT convertono una variabile in formato DINT in una stringa di caratteri e viceversa. In modo implicito si utilizzano le funzioni DI_STRNG (FC5) o STRNG_DI (FC37) dalla biblioteca "Funzioni IEC" fornita. La stringa è preceduta da un carattere introduttivo. Se la variabile specificata nel parametro di ritorno è troppo corta, la conversione non viene effettuata.

Parametri	Dichiarazione	Tipo di dati	Area di memoria	Descrizione
DINT_TO_STRING				
I	INPUT	DINT	E, A, M, D, L, cost.	Valore di ingresso
Valore di ritorno		STRING	D, L	Stringa di caratteri risultante
STRING_TO_DINT				
S	INPUT	STRING	D, L	Stringa di caratteri di ingresso
Valore di ritorno		DINT	E, A, M, D, L	Risultato

REAL_TO_STRING e STRING_TO_REAL

Le funzioni REAL_TO_STRING und STRING_TO_REAL convertono una variabile in formato REAL in una stringa di caratteri o viceversa. In modo implicito si utilizzano le funzioni R_STRNG (FC30) o STRNG_R (FC39) dalla biblioteca "Funzioni IEC" fornita.

La stringa di caratteri deve essere presente nel seguente formato:

$\pm v.nnnnnnnE\pm xx$ (\pm = Segno, v = Posizione prima della virgola, n = Posizioni dopo la virgola, x = Posizioni esponenti)

Se la lunghezza della stringa di caratteri è minore di 14, oppure se non è strutturata come descritto sopra, allora non avviene alcuna conversione.

Se la variabile indicata nel valore di ritorno è troppo corta o se nel parametro IN non è presente un numero in virgola mobile valido, la conversione non viene effettuata e il flag OK viene impostato a "0".

Parametri	Dichiarazione	Tipo di dati	Area di memoria	Descrizione
REAL_TO_STRING				
IN	INPUT	REAL	E, A, M, D, L, cost.	Valore d'ingresso
Valore di ritorno		STRING	D, L	Stringa di caratteri risultante
STRING_TO_REAL				
S	INPUT	STRING	D, L	Stringa di caratteri di ingresso
Valore di ritorno		REAL	E, A, M, D, L	Risultato

14.4.4 Esempio di elaborazione di stringhe di caratteri

Composizione di testi di segnalazione

```
//Composizione e salvataggio comandati dal processo di testi di
segnalazione

////////////////////////////////////
//Il blocco contiene i necessari testi di segnalazione e gli      //
//ultimi 20 messaggi generati                                     //
////////////////////////////////////

DATA_BLOCK Testi_di_segnalazione

STRUCT
    Indice          : INT;
    Buffer_di_testo  : ARRAY[0..19] OF STRING[34];
    HW              : ARRAY[1..5]   OF STRING[16]; //5 diversi
                                                    //dispositivi
    stati           : ARRAY[1..5]   OF STRING[12]; //5 diversi
                                                    //stati
END_STRUCT

BEGIN
    Indice :=0;
    HW[1]  := 'motore ';
    HW[2]  := 'valvola ';
    HW[3]  := 'pressa ';
    HW[4]  := 'staz. saldatura';
    HW[5]  := 'bruciatore ';
    Stati[1] := ' guasto';
    Stati[2] := ' avviato';
    Stati[3] := ' temperatura';
    Stati[4] := ' riparato';
    Stati[5] := ' attesa';
END_DATA_BLOCK

////////////////////////////////////
//La funzione compone testi di segnalazione e li registra nel DB //
//dei testi di segnalazione.                                     //
//I testi di segnalazione vengono salvati in un buffer circolare. //
//Anche il successivo indice libero del buffer di testo si trova //
//nel DB dei testi di segnalazione e viene aggiornato dalla      //
//funzione.                                                       //
////////////////////////////////////

FUNCTION generatore_di_testi : bool
VAR_INPUT
    unit   : int; //Indice del dispositivo
    nr     : int; //N. ID del dispositivo
    stato  : int;
    valore  : int;
END_VAR
VAR_TEMP
    testo : string[34];
    i     : int;
END_VAR
```

```
//inizializzazione della variabile temporanea
testo := '';
generatore_di_testi := true;
Case unit of
  1..5 : case stato of
    1..5 : testo := concat( in1 := Testi_di_segnalazione.HW[unit],
                           in2 := right(l:=2,in:=I_STRNG(nr)));
    testo := concat( in1 := testo,
                     in2 := Testi_di_segnalazione.stati[stato]);
    if valore<> 0 then
      testo := concat( in1 := testo,
                       in2 := I_STRNG(valore));
    end_if;
  else generatore_di_testi := false;
end_case;
else generatore_di_testi := false;
end_case;
i := Testi_di_segnalazione.Indice;
Testi_di_segnalazione.Buffer_di_testo[i] := testo;
Testi_di_segnalazione.indice := (i+1) mod 20;
END_FUNCTION

/////////////////////////////////////////////////////////////////
//La funzione viene richiamata nel programma ciclico al cambio di //
//fronte in %M10.0 e determina la registrazione di un messaggio //
//in seguito alla modifica di un parametro. //
/////////////////////////////////////////////////////////////////

ORGANIZATION_BLOCK Ciclo
Var_temp
  Sistoperativo_ifx : array [0..20] of byte;
  errori: BOOL;
End_var;

/////////////////////////////////////////////////////////////////
//Il seguente richiamo determina la registrazione di "Motore12 //
//avviato" nel buffer di testo del DB dei testi di segnalazione, //
//operazione durante la quale %MW0 assegna un 1, %EW2 un 12 e %MW2//
//un 2. *) //
/////////////////////////////////////////////////////////////////

if %M10.0 <> %M10.1 then
  errori := generatore_di_testi (unit := word_to_int(%MW0),
                                nr := word_to_int(%EW2),
                                stato := word_to_int(%MW2),
                                valore := 0);

  %M10.1:=M10.0;
end_if;
end_organization_block
```

14.5 Funzioni di selezione dei valori

14.5.1 Funzioni di selezione dei valori

Sono disponibili le seguenti funzioni di selezione dei valori come funzioni S7-SCL interne conformi alla norma IEC 61131-3.

Nota

Alcune delle funzioni sono reperibili anche nella biblioteca standard di STEP 7. Le funzioni della biblioteca tuttavia non corrispondono in toto ai parametri indicati dalla norma IEC.

SEL

La funzione SEL seleziona uno dei due valori di ingresso.

Come valori di ingresso sono ammessi tutti i tipi di dati ad eccezione dei tipi ARRAY e STRUCT nonché dei tipi di dati dei parametri. Tutte le variabili parametrizzate devono appartenere al tipo di dati della stessa classe.

Esempio:

```
A := SEL (G := SELECT, IN0 := X, IN1 := Y);
```

Parametri	Dichiarazione	Tipo di dati	Area di memoria	Descrizione
G	INPUT	BOOL	E, A, M, D, L	Criterio di selezione
IN0	INPUT	Tutti i tipi di dati ad eccezione di ARRAY e STRUCT	E, A, M, D, L	Primo valore di ingresso
IN1	INPUT	Tutti i tipi di dati ad eccezione di ARRAY e STRUCT	E, A, M, D, L	Secondo valore di ingresso
Valore di ritorno	OUTPUT	Tutti i tipi di dati ad eccezione di ARRAY e STRUCT	E, A, M, D, L	Valore di ingresso selezionato (facoltativo)

MAX

La funzione MAX seleziona, da un numero di valori relativi alle variabili, il più alto.

Come valori di ingresso sono ammessi tipi di dati numerici e temporali. Tutte le variabili parametrizzate devono appartenere al tipo di dati della stessa classe. L'espressione assume il tipo di dati con valore più alto.

Esempio: A:= MAX (IN1:=a, IN2:=b, IN3:=c, IN4:=d);

Parametri	Dichiarazione	Tipo di dati	Area di memoria	Descrizione
IN1	INPUT	ANY_NUM Tipi di dati temporali ad eccezione di S5TIME	E, A, M, D, L	Primo valore di ingresso
IN2	INPUT	ANY_NUM Tipi di dati temporali ad eccezione di S5TIME	E, A, M, D, L	Secondo valore di ingresso
INn (n=3...32)	INPUT	ANY_NUM Tipi di dati temporali ad eccezione di S5TIME	E, A, M, D, L	Ultimo valore di ingresso (facoltativo)
Valore di ritorno	OUTPUT	ANY_NUM Tipi di dati temporali ad eccezione di S5TIME	E, A, M, D, L	Il valore d'ingresso più alto (facoltativo)

MIN

La funzione MIN seleziona, da un numero di valori relativi alle variabili, il più basso. Come valori di ingresso sono ammessi tipi di dati numerici e temporali. Tutte le variabili parametrizzate devono appartenere al tipo di dati della stessa classe. L'espressione assume il tipo di dati con valore più alto.

Esempio: A:= MIN (IN1:=a, IN2:=b, IN3:=c, IN4:=d);

Parametri	Dichiarazione	Tipo di dati	Area di memoria	Descrizione
IN1	INPUT	ANY_NUM Tipi di dati temporali ad eccezione di S5TIME	E, A, M, D, L	Primo valore di ingresso
IN2	INPUT	ANY_NUM Tipi di dati temporali ad eccezione di S5TIME	E, A, M, D, L	Secondo valore di ingresso
INn (n=3...32)	INPUT	ANY_NUM Tipi di dati temporali ad eccezione di S5TIME	E, A, M, D, L	Ultimo valore di ingresso (facoltativo)
Valore di ritorno	OUTPUT	ANY_NUM Tipi di dati temporali ad eccezione di S5TIME	E, A, M, D, L	Valore d'ingresso più basso (facoltativo)

LIMIT

La funzione LIMIT delimita il valore numerico di una variabile a seconda dei valori limite parametrizzabili. Come valori di ingresso sono ammessi tipi di dati numerici e temporali. Tutte le variabili parametrizzate devono appartenere al tipo di dati della stessa classe. L'espressione assume il tipo di dati con valore più alto. Il valore limite inferiore (MN) non deve superare il valore limite superiore (MX).

Esempio: `A:= LIMIT (MN:=5, IN:= fasi di elaborazione, MX:= 10);`

Parametri	Dichiarazione	Tipo di dati	Area di memoria	Descrizione
MN	INPUT	ANY_NUM Tipi di dati temporali ad eccezione di S5TIME	E, A, M, D, L	Limite inferiore
IN	INPUT	ANY_NUM Tipi di dati temporali ad eccezione di S5TIME	E, A, M, D, L	Variabile d'ingresso
MX	INPUT	ANY_NUM Tipi di dati temporali ad eccezione di S5TIME	E, A, M, D, L	Limite superiore
Valore di ritorno	OUTPUT	ANY_NUM Tipi di dati temporali ad eccezione di S5TIME	E, A, M, D, L	Variabile di uscita delimitata (facoltativo)

MUX

La funzione MUX seleziona un valore di ingresso da un numero di valori di ingresso. La selezione avviene in base al parametro di ingresso K. Come valori di ingresso sono ammessi tutti i tipi di dati. L'espressione assume il tipo di dati con valore più alto.

Esempio:

`A:= MUX (K:=SELECT, IN0:= Steps, IN1:=Number, IN2:=Total);`

Parametri	Dichiarazione	Tipo di dati	Area di memoria	Descrizione
K	INPUT	INT	E, A, M, D, L	Criterio di selezione
IN0	INPUT	Tutti i tipi di dati ad eccezione di ARRAY e STRUCT	E, A, M, D, L	Primo valore di ingresso
IN1	INPUT	Tutti i tipi di dati ad eccezione di ARRAY e STRUCT	E, A, M, D, L	Secondo valore di ingresso
INn (n=2...31)	INPUT	Tutti i tipi di dati ad eccezione di ARRAY e STRUCT	E, A, M, D, L	Ultimo valore di ingresso (facoltativo)

Parametri	Dichiarazione	Tipo di dati	Area di memoria	Descrizione
INELSE	INPUT	Tutti i tipi di dati ad eccezione di ARRAY e STRUCT	E, A, M, D, L	<p>Valore di ingresso alterantivo (facoltativo)</p> <p>Se K si colloca al di fuori del campo da 0 a n, il valore attuale verrà utilizzato da INELSE.</p> <p>Se INELSE non è occupato, il valore attuale verrà utilizzato da INO.</p>
Valore di ritorno	OUTPUT	Tutti i tipi di dati ad eccezione di ARRAY e STRUCT	E, A, M, D, L	Valore selezionato (facoltativo)

14.6 SFC, SFB e biblioteca standard

14.6.1 Funzioni, blocchi funzionali di sistema e biblioteca standard

I sistemi operativi delle CPU S7 contengono funzioni di sistema e funzioni standard che l'utente può usare per la programmazione in S7-SCL. Si tratta in particolare dei dati seguenti:

- Blocchi organizzativi (OB)
- Funzioni di sistema (SFC)
- Blocchi funzionali di sistema (SFB)

Interfaccia di richiamo (SFC/SFB)

I blocchi possono essere indirizzati in modo simbolico o assoluto. A tal fine è necessario il nome simbolico, definito nella tabella dei simboli, o il numero per l'identificazione assoluta del blocco.

Durante il richiamo occorre assegnare ai parametri formali, i cui nomi e tipi di dati sono stati definiti durante la generazione dei blocchi parametrizzabili, quei parametri attuali i cui valori vengono utilizzati dal blocco durante l'esecuzione del programma utente.

S7-SCL ricerca il blocco da richiamare nelle seguenti directory e biblioteche:

- la cartella "Programmi"
- le biblioteche standard Simatic (tedesco)
- la biblioteca standard IEC (inglese)

Quando S7-SCL trova un blocco lo copia nel programma utente. Fanno eccezione i blocchi che, per il loro nome, devono essere richiamati con (" ... ") e i blocchi richiamati in modo assoluto. Questi nomi vengono cercati solo nella tabella dei simboli del programma utente. Queste funzioni devono essere copiate manualmente nel programma utente dal SIMATIC Manager.

Richiamo condizionato (SFB/SFC)

Per un richiamo condizionato, si deve impostare a 0 il parametro d'ingresso EN predefinito (p. es. tramite l'ingresso E0.3); in questo caso il blocco non viene richiamato. Se EN viene occupato con 1, la funzione viene richiamata. Anche in questo caso il parametro d'uscita ENO viene impostato su "1" (altrimenti su "0") se durante l'elaborazione del blocco non si verificano errori.

È sconsigliabile richiamare le SFC in modo condizionato perché, se la funzione non viene richiamata, la variabile che deve assumere il valore di ritorno della funzione non viene definita.

Nota

Se nel programma si utilizzano queste operazioni per i tipi di dati TIME, DATE_AND_TIME e STRING, S7-SCL richiama implicitamente i relativi blocchi standard.

Per questo motivo i simboli e i numeri di questi blocchi standard sono riservati e non sono utilizzabili per altri blocchi. Poiché S7-SCL non verifica sempre se questa regola è stata rispettata si possono verificare errori di compilazione.

La seguente tabella elenca le funzioni standard IEC utilizzate implicitamente da S7-SCL.

Operazione	DATE_AND_TIME	STRING
==	EQ_DT (FC9)	EQ_STRING (FC10)
<>	NE_DT (FC28)	NE_STRING (FC29)
>	GT_DT (FC14)	GT_STRING (FC15)
>=	GE_DT (FC12)	GE_STRING (FC13)
<=	LE_DT (FC18)	LE_STRING (FC19)
<	LT_DT (FC23)	LT_STRING (FC24)
DATE_AND_TIME + TIME	AD_DT_TM (FC1)	
DATE_AND_TIME + TIME	SB_DT_TM (FC35)	
DATE_AND_TIME + DATE_AND_TIME	SB_DT_DT (FC34)	
TIME_TO_S5TIME(TIME)	TIM_S5TI (FC40)	
S5TIME_TO_TIME(S5TIME)	S5TI_TIM (FC33)	

Tutte le altre informazioni sugli SFB, SFC e OB disponibili e la descrizione dettagliata dell'interfaccia sono riportate nel manuale di riferimento di STEP 7 "Funzioni di sistema e standard S7-300/400".

14.6.2 Interfaccia di trasferimento agli OB

Blocchi organizzativi

I blocchi organizzativi costituiscono l'interfaccia fra il sistema operativo della CPU e il programma utente. Con l'ausilio di OB si possono eseguire determinate parti di un programma:

- all'avviamento della CPU
- con esecuzione a frequenza ciclica o temporale
- a determinate ore o in determinati giorni
- allo scadere di una durata di tempo predefinita
- se si verificano degli errori
- se si verificano interrupt di processo o allarmi di comunicazione

I blocchi organizzativi vengono elaborati in base alla priorità loro assegnata.

OB disponibili

Non tutte le CPU sono in grado di elaborare tutti gli OB disponibili in S7. Per una descrizione degli OB disponibili si rimanda ai dati tecnici della CPU usata.

15 Descrizione del linguaggio di programmazione

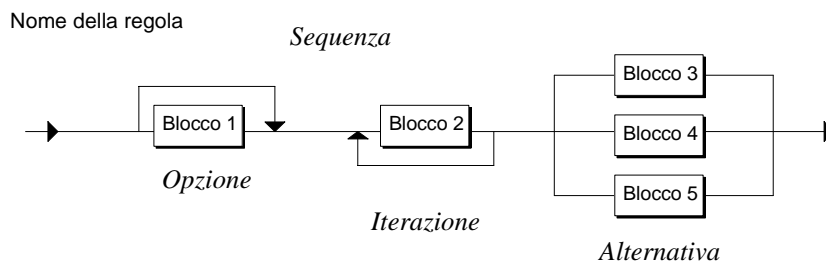
15.1 Descrizione formale

15.1.1 Diagrammi sintattici

La base per la descrizione del linguaggio è costituita dai diagrammi sintattici. Essi offrono una chiara panoramica della struttura sintattica di S7-SCL. L'elenco completo dei diagrammi e dei relativi elementi è riportato nei capitoli "Regole lessicali" e "Regole sintattiche".

Che cos'è un diagramma sintattico?

Il diagramma sintattico è una rappresentazione grafica della struttura del linguaggio. La struttura è costituita da una sequenza gerarchica di regole. Ogni regola a sua volta è presente come blocco in una regola sottostante.

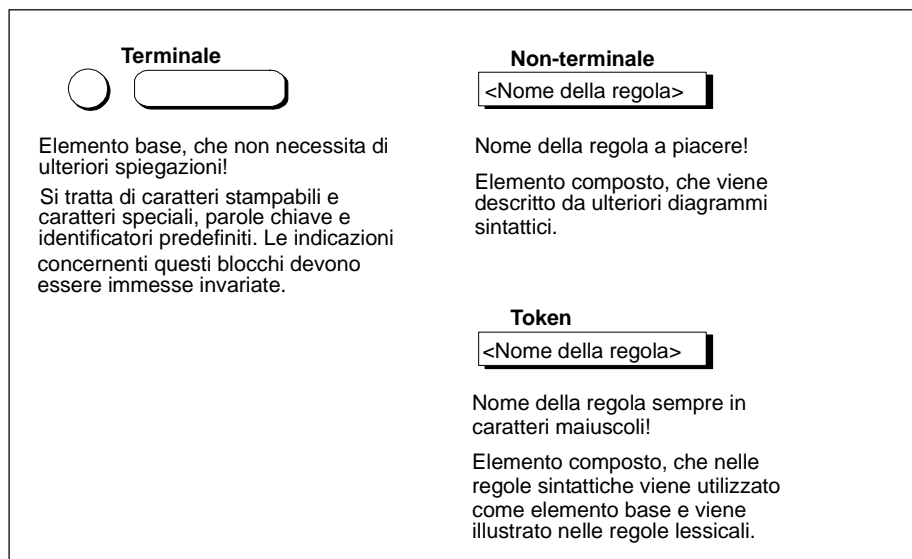


Il diagramma sintattico viene letto da destra verso sinistra. Si devono osservare le seguenti strutture delle regole:

- Sequenza: sequenza di blocchi
- Opzione: ramo saltabile
- Iterazione: ripetizione di rami
- Alternativa: diramazione

Quali tipi di blocchi esistono?

Un blocco è un elemento base oppure un elemento che a sua volta si compone di vari blocchi. La figura seguente mostra i tipi di simboli corrispondenti ai vari blocchi:



15.1.2 Regole

Le regole che possono essere utilizzate per creare il programma utente S7-SCL sono suddivise in **regole lessicali** e **regole sintattiche**.

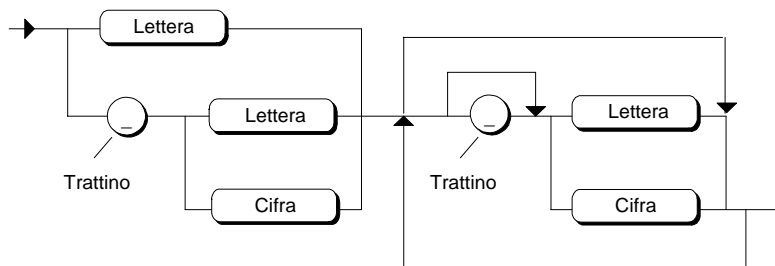
Regole lessicali

Le regole lessicali descrivono la struttura degli elementi (token) che vengono elaborati durante l'analisi lessicale del compilatore. Perciò, la modalità di scrittura non è in formato libero, cioè le regole devono essere rigorosamente rispettate. Ciò significa in modo particolare che:

- non è consentito inserire caratteri di formattazione
- non è consentito inserire commenti al blocco e alla riga
- non è consentito inserire attributi per gli identificatori

L'esempio illustra la regola lessicale IDENTIFICATORE. Essa descrive la struttura di un identificatore (nome), p. es.:

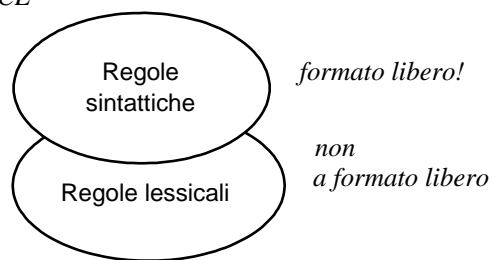
CAMPO_MISURA_12
VALNOM_B_1



Regole sintattiche

Sulla base delle regole lessicali, le regole sintattiche descrivono la struttura di S7-SCL. Nell'ambito di tali regole l'utente può creare il proprio programma S7-SCL in formato libero:

Programma SCL



Formalità

Ogni regola possiede un nome che viene preposto. Se la regola viene utilizzata in una regola sovraordinata, questo nome appare in un rettangolo.

Se il nome della regola è scritto in caratteri maiuscoli, si tratta di un token che viene descritto nelle regole lessicali.

Semantica

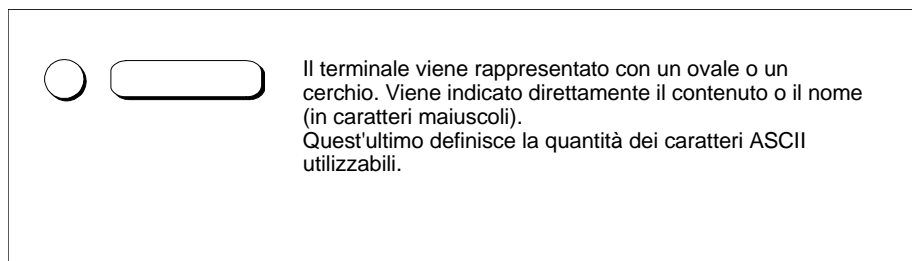
Nelle regole può essere descritta solo la struttura formale del linguaggio. Esse non sempre descrivono il significato, cioè la semantica. Perciò, nei punti più importanti, accanto alle regole vengono scritte sempre informazioni supplementari. Seguono alcuni esempi di queste informazioni.

- In caso di elementi dello stesso tipo, ma con significato diverso viene indicato un nome supplementare: p. es. nella regola dell'indicazione della data nella SEQUENZA DI CIFRE DECIMALI anno, mese o giorno. Il nome indica il tipo d'impiego.
- Le restrizioni importanti vengono annotate accanto alle regole: p. es. accanto alla regola Simbolo viene indicato che i simboli devono essere definiti nella tabella dei simboli.

15.1.3 Terminali delle regole lessicali

Definizione

Un terminale è un elemento base che non viene spiegato con un'ulteriore regola, ma in modo verbale. Nei diagrammi sintattici esso viene indicato con il simbolo seguente:



Nelle tabelle seguenti, i terminali sono specificati mediante indicazione della quantità degli elementi del set di caratteri ASCII.

Lettere e cifre

Sono i caratteri principalmente utilizzati. L'IDENTIFICATORE si compone p. es. di lettere, cifre e del trattino.

Caratteri	Sottogruppo	Elementi del set di caratteri
Lettera	Carattere maiuscolo Carattere minuscolo	A.. Z a.. z
Cifra	Cifra decimale	0.. 9
Cifra ottale	Cifra ottale	0.. 7
Cifra esadecimale	Cifra esadecimale	0.. 9, A.. F, a.. f
Bit	Cifra binaria	0, 1

Caratteri stampabili e caratteri speciali

Il set di caratteri ASCII ampliato e completo può essere utilizzato in stringhe, commenti e simboli.

Caratteri	Sottogruppo	Elementi del set di caratteri
Carattere stampabile	Dipende dal codice di carattere utilizzato. Per esempio nel codice ASCII a partire dall'equivalente decimale , senza DEL e senza i seguenti caratteri sostitutivi:	Tutti i caratteri stampabili
Carattere sostitutivo	Dollaro Apostrofo	\$ '
Carattere di controllo	\$P o \$p \$L o \$l \$R o \$r \$T o \$t \$N o \$n	Salto pagina (formfeed, page) Salto riga (linefeed) Ritorno carrello (carriage return) Tabulatore Nuova riga
Rappresentazione sostitutiva in codice esadecimale	\$hh	Qualsiasi carattere, rappresentabile in codice esadecimale (hh)

15.1.4 Caratteri di formattazione, separatori e operazioni

Nelle regole lessicali

La tabella seguente descrive l'impiego di singoli caratteri del set di caratteri ASCII come caratteri di formattazione, separatori e operatori nell'ambito delle regole lessicali.

Caratteri	Descrizione
:	Separatori fra ore, minuti e secondi Attributi
.	Separatori per indirizzamento assoluto e rappresentazione di numeri in virgola mobile e intervalli di tempo
' '	Caratteri e stringhe di caratteri
" "	Caratteri introduttivi per simboli in base alle regole della tabella dei simboli
_ Trattino	Separatore per valori numerici in costanti, può essere presente in IDENTIFICATORI
\$	Simbolo del dollaro per indicare caratteri di controllo o caratteri sostitutivi
\$> \$<	Simbolo di interruzione stringa nel caso in cui una stringa non rientri in una riga o se si intendono inserire dei commenti.

Per costanti

La tabella seguente descrive l'uso di singoli caratteri e stringhe di caratteri per costanti nell'ambito delle regole lessicali. La tabella vale per il mnemonico tedesco e inglese.

Prefisso	Contrassegno per	Regola lessicale
BOOL#	Costante tipizzata di tipo BOOL	Costante BIT
BYTE#	Costante tipizzata di tipo BYTE	Costante BIT
WORD#	Costante tipizzata di tipo WORD	Costante BIT
DWORD#	Costante tipizzata di tipo DWORD	Costante BIT
INT#	Costante tipizzata di tipo INT	Costante di numero intero
DINT#	Costante tipizzata di tipo DINT	Costante di numero intero
REAL#	Costante tipizzata di tipo REAL	Costante REAL
CHAR#	Costante tipizzata di tipo CHAR	Costante CHAR
2#	Costante numerica	Sequenza di cifre binarie
8#	Costante numerica	Sequenza di cifre ottali
16#	Costante numerica	Sequenza di cifre esadecimali
D#	Indicazione della data	DATA
DATE#	Indicazione della data	DATA
DATE_AND_TIME#	Indicazione della data	DATA E ORA
DT#	Indicazione della data	DATA E ORA
E	Separatori per le costanti di numero reale	Esponente
e	Separatori per le costanti di numero reale	Esponente
D	Separatore per intervallo di tempo (Day)	Giorni (Regola: rappresentazione a livelli)
H	Separatore per intervallo di tempo (Hour)	Ore: (Regola: rappresentazione a livelli)
M	Separatore per intervallo di tempo (Minutes)	Minuti : (Regola: rappresentazione a livelli)

Prefisso	Contrassegno per	Regola lessicale
MS	Separatore per intervallo di tempo (Milliseconds)	Millisecondi: (Regola: rappresentazione a livelli)
S	Separatore per intervallo di tempo (Seconds)	Secondi: (Regola: rappresentazione a livelli)
T#	Indicazione della data	INTERVALLO
TIME#	Indicazione della data	INTERVALLO
TIME_OF_DAY#	Indicazione della data	ORA DEL GIORNO
TOD#	Indicazione della data	ORA DEL GIORNO

Nelle regole sintattiche

La tabella seguente descrive l'impiego di singoli caratteri del set di caratteri ASCII come caratteri di formattazione e separatori nell'ambito delle regole sintattiche, nei commenti e negli attributi.

Caratteri	Descrizione	Regola sintattica, commento o attributo
:	Separatore per indicazione del tipo, con istruzione dopo etichetta di salto	Dichiarazione di variabile, dichiarazione di istanza, funzione, parte istruzioni, istruzione CASE
;	Conclusione di una convenzione o istruzione	Dichiarazione di costante e di variabile, parte istruzioni, parte assegnazioni, blocco costanti, blocco etichette di salto
,	Separatore per liste e blocchi di etichette di salto	Dichiarazione di variabile, specificazione tipo di dati array, lista inizializzazione di campo, parametro di FB, parametro di FC, lista di valori, dichiarazione di istanza
..	Indicazione di campo	Specificazione tipo di dati array, lista di valori
.	Separatore per nome di FB e DB, indirizzamento assoluto	Richiamo di FB, variabile strutturata
()	Richiamo di funzione e di blocco funzionale, parentesi in espressioni, lista di inizializzazione per array	Richiamo di funzione, richiamo di FB, espressione, lista inizializzazione di campo, moltiplicazione semplice, espressione di potenza
[]	Dichiarazione di array, variabile strutturata parte array, indicizzazione nelle variabili globali e nelle stringhe	Specificazione tipo di dati array, specificazione tipo di dati STRING
(* *)	Blocco di commenti	Vedere "Regole lessicali"
//	Commento a una riga	vedere "Regole lessicali"
{ }	Blocco di attributo	Specificazione di attributi
%	Elemento introduttivo per identificatori diretti	Per programmare in conformità alla norma IEC utilizzare %M4.0 invece di M4.0.
#	Carattere indicante che l'elemento successivo non è una parola chiave	Indica che l'identificatore non è una parola chiave ad es. #FOR.

Operazioni

Nella tabella seguente sono descritte tutte le operazioni S7-SCL, la parole chiave, p. es. AND e le consuete operazioni come caratteri singoli. La tabella vale per il mnemonico tedesco e inglese.

Operazione	Descrizione	Esempio di regola sintattica
:=	Operazione di assegnazione, assegnazione iniziale, inizializzazione del tipo di dati	Assegnazione di valori, parte assegnazioni DB, blocco costanti, assegnazione d'uscita e di ingresso/uscita, assegnazione d'ingresso, assegnazione di ingresso/uscita
+, -	Operazioni aritmetiche: operazioni unarie, segno	Espressione, espressione semplice, espressione di potenza
+, -, *, / MOD; DIV	Operazioni aritmetiche di base	Operazione aritmetica di base, moltiplicazione semplice
**	Operazioni aritmetiche: operazione di potenza	Espressione
NOT	Operazioni logiche: negazione	Espressione
AND, &, OR; XOR,	Operazioni logiche di base	Operazione logica di base
<, >, <=, >=, =, <>	Operazione di confronto	Operazione di confronto

15.1.5 Parole chiave e identificatori predefiniti

La tabella seguente contiene parole chiave e identificatori predefiniti di S7-SCL, elencati in ordine alfabetico. Vengono inoltre riportate descrizioni e regole sintattiche, in cui esse vengono utilizzate come terminali. In generale, le parole chiave non dipendono dal mnemonico.

Parole chiave	Descrizione	Esempio di regola sintattica
AND	Operazione logica	Operazione logica di base
ANY	Identificazione per il tipo di dati ANY	Specificazione tipo di dati parametro
ARRAY	Elemento introduttivo della specificazione di un array, segue la lista indice fra "[" e "]"	Specificazione tipo di dati array
AT	Dichiara un accesso ad una variabile	Dichiarazione di variabili e parametri
BEGIN	Elemento introduttivo della parte istruzioni nel blocco di codice o nella parte inizializzazione del blocco dati	Blocco organizzativo, funzione, blocco funzionale, blocco dati
BLOCK_DB	Identificazione per il tipo di dati BLOCK_DB	Specificazione tipo di dati parametro
BLOCK_FB	Identificazione per il tipo di dati BLOCK_FB	Specificazione tipo di dati parametro
BLOCK_FC	Identificazione per il tipo di dati BLOCK_FC	Specificazione tipo di dati parametro
BLOCK_SDB	Identificazione per il tipo di dati BLOCK_SDB	Specificazione tipo di dati parametro
BOOL	Tipo di dati semplice per dati binari	Tipo di dati bit
BY	Elemento introduttivo dell'ampiezza di passo	Istruzione FOR
BYTE	Tipo di dati semplice	Tipo di dati bit
CASE	Elemento introduttivo dell'istruzione di controllo per la selezione	Istruzione CASE
CHAR	Tipo di dati semplice	Tipo del carattere
CONST	Elemento introduttivo per la definizione di costanti	Blocco costanti
CONTINUE	Istruzione di controllo per loop FOR, WHILE e REPEAT	Istruzione CONTINUE
COUNTER	Tipo di dati per contatori, utilizzabile solo nel blocco parametri	Specificazione tipo di dati parametro
DATA_BLOCK	Elemento introduttivo del blocco dati	Blocco dati
DATE	Tipo di dati semplice per data	Tipo di tempo
DATE_AND_TIME	Tipo di dati composto per data e ora	DATE_AND_TIME
DINT	Tipo di dati semplice per numero intero (Integer) doppia precisione	Tipo di dati numerici
DIV	Operazione per divisione	Operazione aritmetica di base, moltiplicazione semplice
DO	Elemento introduttivo della parte istruzioni nell'istruzione FOR	Istruzione FOR, istruzione WHILE
DT	Tipo di dati semplice per data e ora	DATE_AND_TIME
DWORD	Tipo di dati semplice di doppia parola	Tipo di dati bit
ELSE	Elemento introduttivo nel caso non venga soddisfatta alcuna condizione	Istruzione IF, istruzione CASE
ELSIF	Elemento introduttivo della condizione alternativa	Istruzione IF
EN	Flag per abilitazione blocco	
ENO	Flag di errore del blocco	
END_CASE	Fine dell'istruzione CASE	Istruzione CASE

Parole chiave	Descrizione	Esempio di regola sintattica
END_CONST	Fine della definizione di costanti	Blocco costanti
END_DATA_BLOCK	Fine del blocco dati	Blocco dati
END_FOR	Fine dell'istruzione FOR	Istruzione FOR
END_FUNCTION	Fine della funzione	Funzione
END_FUNCTION_BLOCK	Fine del blocco funzionale	Blocco funzionale
END_IF	Fine dell'istruzione IF	Istruzione IF
END_LABEL	Fine della dichiarazione di un blocco etichette di salto	Blocco etichette di salto
END_TYPE	Fine dell'UDT	Tipo di dati definito dall'utente
END_ORGANIZATION_BLOCK	Fine del blocco organizzativo	Blocco organizzativo
END_REPEAT	Fine dell'istruzione REPEAT	Istruzione REPEAT
END_STRUCT	Fine della specificazione di una struttura	Specificazione tipo di dati struttura
END_VAR	Fine del blocco di dichiarazione	Blocco variabili temporanee, blocco variabili statiche, blocco parametri
END_WHILE	Fine dell'istruzione WHILE	Istruzione WHILE
EXIT	Uscita diretta dall'elaborazione di loop	EXIT
FALSE	Costante booleana predefinita: condizione logica non soddisfatta, valore uguale a 0	
FOR	Elemento introduttivo dell'istruzione di controllo per l'elaborazione di loop	Istruzione FOR
FUNCTION	Elemento introduttivo della funzione	Funzione
FUNCTION_BLOCK	Elemento introduttivo del blocco funzionale	Blocco funzionale
GOTO	Istruzione per l'esecuzione di un salto ad un'etichetta di salto	Salto di programma
IF	Elemento introduttivo dell'istruzione di controllo per la selezione	Istruzione IF
INT	Tipo di dati semplice per numero intero (Integer), semplice precisione	Tipo di dati numerici
LABEL	Elemento introduttivo di un blocco etichette di salto	Blocco etichette di salto
MOD	Operazione aritmetica per resto della divisione	Operazione aritmetica di base, moltiplicazione semplice
NIL	Puntatore zero	
NOT	Operazione logica appartenente alle operazioni unarie	Espressione
OF	Elemento introduttivo della specificazione del tipo di dati	Specificazione tipo di dati array, istruzione CASE
OK	Flag indicante se le istruzioni di un blocco sono state elaborate senza errori	
OR	Operazione logica	Operazione logica di base
ORGANIZATION_BLOCK	Elemento introduttivo del blocco organizzativo	Blocco organizzativo
POINTER	Tipo di dati puntatore, consentito solo nella dichiarazione parametri del blocco parametri; non viene elaborato in S7-SCL	vedere capitolo "Dati globali".
PROGRAM	Elemento introduttivo della parte istruzioni di un FB (sinonimo di FUNCTION_BLOCK)	Blocco funzionale
REAL	Tipo di dati semplice	Tipo di dati numerici

Parole chiave	Descrizione	Esempio di regola sintattica
REPEAT	Elemento introduttivo dell'istruzione di controllo per l'elaborazione di loop	Istruzione REPEAT
RET_VAL	Valore di ritorno di una funzione	Funzione
RETURN	Istruzione di controllo per il ritorno dal sottoprogramma	Istruzione RETURN
S5TIME	Tipo di dati semplice per indicazioni di tempo, formato speciale S5	Tipo di tempo
STRING	Tipo di dati per stringa di caratteri	Specificazione tipo di dati STRING
STRUCT	Elemento introduttivo della specificazione di una struttura, segue la lista dei componenti	Specificazione tipo di dati struttura
THEN	Elemento introduttivo delle azioni seguenti, se è soddisfatta la condizione	Istruzione IF
TIME	Tipo di dati semplice per indicazioni di tempo	Tipo di tempo
TIMER	Tipo di dati per temporizzatore, utilizzabile solo nel blocco parametri	Specificazione tipo di dati parametro
TIME_OF_DAY	Tipo di dati semplice per ora del giorno	Tipo di tempo
TO	Elemento introduttivo del valore finale	Istruzione FOR
TOD	Tipo di dati semplice per ora del giorno	Tipo di tempo
TRUE	Costante booleana predefinita: condizione logica soddisfatta, valore diverso da 0	
TYPE	Elemento introduttivo dell'UDT	Tipo di dati definito dall'utente
VAR	Elemento introduttivo del blocco di dichiarazione	Blocco variabili statiche
VAR_TEMP	Elemento introduttivo del blocco di dichiarazione	Blocco variabili temporanee
UNTIL	Elemento introduttivo della condizione d'interruzione per l'istruzione REPEAT	Istruzione REPEAT
VAR_INPUT	Elemento introduttivo del blocco di dichiarazione	Blocco parametri
VAR_IN_OUT	Elemento introduttivo del blocco di dichiarazione	Blocco parametri
VAR_OUTPUT	Elemento introduttivo del blocco di dichiarazione	Blocco parametri
WHILE	Elemento introduttivo dell'istruzione di controllo per l'elaborazione dei loop	Istruzione WHILE
WORD	Tipo di dati semplice di parola	Tipo di dati bit
VOID	Nessun valore di ritorno in un richiamo di funzione	Funzione
XOR	Operazione logica	Operazione logica di base

15.1.6 Identificazioni di operandi e parole chiave di blocchi

Dati globali del sistema

La tabella seguente contiene il mnemonico tedesco delle identificazioni di operando di S7-SCL con descrizione in ordine alfabetico:

- Indicazione dell'identificazione di operando:
 Prefisso di memoria (A, E, M, PA, PE) o blocco dati (D)
- Indicazione della dimensione dell'elemento dati:
 Prefisso di dimensione (opzionale o B, D, W, X)

Il mnemonico rappresenta una combinazione fra l'identificazione di operando (prefisso di memoria o D per blocco dati) e prefisso di dimensione. Entrambi sono regole lessicali. La tabella è ordinata in base al mnemonico tedesco, viene inoltre indicata il corrispondente mnemonico inglese.

Mnemonico tedesco	Mnemonico inglese	Prefisso di memoria o blocco dati	Prefisso di lunghezza
A	Q	Uscita (tramite immagine di processo)	Bit
AB	QB	Uscita (tramite immagine di processo)	Byte
AD	QD	Uscita (tramite immagine di processo)	Doppia parola
AW	QW	Uscita (tramite immagine di processo)	Parola
AX	QX	Uscita (tramite immagine di processo)	Bit
D	D	Blocco dati	Bit
DB	DB	Blocco dati	Byte
DD	DD	Blocco dati	Doppia parola
DW	DW	Blocco dati	Parola
DX	DX	Blocco dati	Bit
E	I	Ingresso (tramite immagine di processo)	Bit
EB	IB	Ingresso (tramite immagine di processo)	Byte
ED	ID	Ingresso (tramite immagine di processo)	Doppia parola
EW	IW	Ingresso (tramite immagine di processo)	Parola
EX	IX	Ingresso (tramite immagine di processo)	Bit
M	M	Merker	Bit
MB	MB	Merker	Byte
MD	MD	Merker	Doppia parola
MW	MW	Merker	Parola
MX	MX	Merker	Bit
PAB	PQB	Uscita (periferia diretta)	Byte
PAD	PQD	Uscita (periferia diretta)	Doppia parola
PAW	PQW	Uscita (periferia diretta)	Parola
PEB	PIB	Ingresso (periferia diretta)	Byte
PED	PID	Ingresso (periferia diretta)	Doppia parola
PEW	PIW	Ingresso (periferia diretta)	Parola

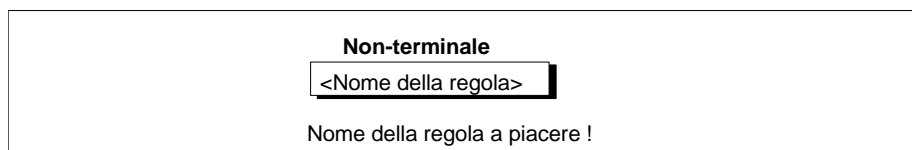
Parole chiave di blocchi

Vengono utilizzate per l'indirizzamento assoluto di blocchi. La tabella è ordinata in base al mnemonico tedesco, viene inoltre indicato il corrispondente mnemonico inglese.

Mnemonico tedesco	Mnemonico inglese	Prefisso di memoria o blocco dati
DB	DB	Blocco dati (Data-Block)
FB	FB	Blocco funzionale (Function Block)
FC	FC	Funzione (Function)
OB	OB	Blocco organizzativo (Organization Block)
SDB	SDB	Blocco dati di sistema (System Data Block)
SFC	SFC	Funzione di sistema (System Function)
SFB	SFB	Blocco funzionale di sistema (System Function Block)
T	T	Temporizzatori (Timer)
UDT	UDT	Tipo di dati globale o definito dall'utente (Userdefined Data Type)
Z	C	Contatori (Counter)

15.1.7 Non-terminali

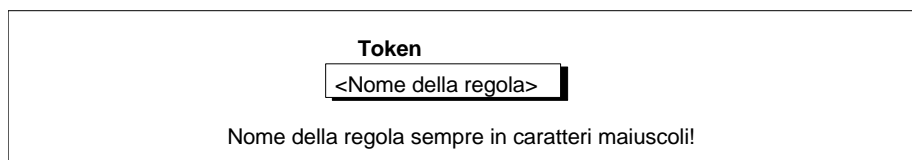
Un non-terminale è un elemento composto che viene descritto da un'ulteriore regola. Il non-terminale viene rappresentato da quadrato. Il nome nel quadrato corrisponde al nome della regola successiva.



L'elemento è presente nelle regole lessicali e nelle regole sintattiche.

15.1.8 Token

Un token è un elemento composto impiegato come elemento base nelle regole sintattiche e viene dichiarato nelle regole lessicali. Il token viene rappresentato da un rettangolo. Il NOME, in caratteri maiuscoli, corrisponde al nome della regola lessicale seguente (senza rettangolo).



I token definiti rappresentano identificatori che sono stati determinati come risultato delle regole lessicali. Questi token descrivono:

- Identificatori
- Assegnazione di nomi in S7-SCL
- Costanti predefinite e flag

15.1.9 Identificatori

Identificatori

Gli identificatori consentono di far riferimento agli oggetti linguistici di S7-SCL. La tabella seguente contiene informazioni sulle classi di identificatori.

Tipo di identificatore	Note, esempi
Parole chiave	Per es. Istruzioni di controllo <code>BEGIN</code> , <code>DO</code> , <code>WHILE</code>
Nomi predefiniti	Nomi di <ul style="list-style-type: none"> • tipi di dati standard (p. es. <code>BOOL</code>, <code>BYTE</code>, <code>INT</code>) • funzioni standard predefinite p. es. <code>ABS</code> • costanti standard <code>TRUE</code> e <code>FALSE</code>
Identificazioni di operandi negli identificatori assoluti	Per dati globali del sistema e blocchi dati: p. es. <code>E1.2</code> , <code>MW10</code> , <code>FC20</code> , <code>T5</code> , <code>DB30</code> , <code>DB10.D4.5</code>
Nomi selezionabili liberamente secondo la regola IDENTIFICATORE	Nomi di <ul style="list-style-type: none"> • variabili dichiarate • componenti di strutture • parametri • costanti dichiarate • etichette di salto
Simboli della tabella dei simboli	Soddisfano la regola lessicale IDENTIFICATORE o la regola lessicale simbolo, cioè racchiuse fra virgolette, p. es. <code>"xyz"</code>

Maiuscole/minuscole

Per quanto riguarda le parole chiave, le minuscole e le maiuscole non sono rilevanti. Dalla versione 4.0 di S7-SCL non viene più fatta distinzione tra maiuscole e minuscole, né nei nomi predefiniti né in quelli a libera scelta, p. es. le variabili, e neppure nei simboli della tabella dei simboli. La tabella seguente ne fornisce una visione di insieme.

Tipo di identificatore	minuscole e maiuscole rilevanti?
Parole chiave	no
Nomi predefiniti nei tipi di dati standard	no
Nomi nelle funzioni standard predefinite	no
Nomi predefiniti nelle costanti standard	no
Identificazioni di operandi negli identificatori assoluti	no
Nomi liberi	no
Simboli della tabella dei simboli	no

I nomi delle funzioni standard, p. es. `BYTE_TO_WORD` e `ABS`, possono così essere scritti anche a lettere minuscole. Allo stesso modo i parametri per le funzioni di temporizzazione e di conteggio, p. es. `SV`, se o `ZV`.

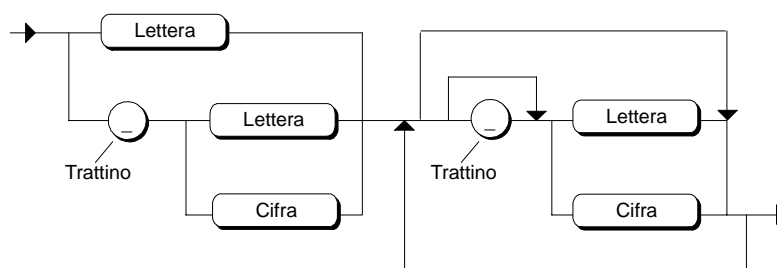
15.1.10 Assegnazione di nomi in S7-SCL

Assegnazione di nomi selezionabili

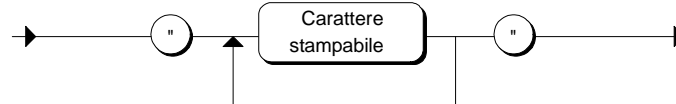
In genere per l'assegnazione di nomi si hanno due possibilità:

- Si possono assegnare dei nomi nell'ambito di S7-SCL. Questi nomi devono corrispondere alla regola IDENTIFICATORE. La regola IDENTIFICATORE può essere usata per ogni nome in S7-SCL.
- Inoltre, si possono inserire nomi tramite STEP 7 con l'ausilio della tabella dei simboli. Anche per questi nomi la regola è IDENTIFICATORE o Simbolo come ulteriore possibilità. Grazie alle virgolette, il simbolo può venire formato da tutti i caratteri stampabili (p. es. spazio).

IDENTIFICATORE



SIMBOLO



I simboli devono essere definiti nella tabella dei simboli.

Regole per l'assegnazione dei nomi

Si prega di tener presente quanto segue:

- Al momento dell'assegnazione dei nomi, si raccomanda di scegliere nomi univoci ed esplicativi, che facilitano la comprensione del programma.
- Si deve verificare dapprima se il nome è già occupato dal sistema, p. es. da identificatori dei tipi di dati o delle funzioni standard.
- Campo di validità: per i nomi che hanno validità globale, il campo di validità si estende per l'intero programma. I nomi con validità locale sono validi solo all'interno di un blocco. In tal modo si ha la possibilità di utilizzare lo stesso nome in diversi blocchi. La tabella seguente informa l'utente sulle varie alternative.

Restrizioni

Durante l'assegnazione di nomi si devono osservare le seguenti restrizioni.

I nomi devono essere univoci nel loro campo di validità, cioè i nomi che sono già stati assegnati all'interno di un blocco non possono essere più assegnati nello stesso blocco. Inoltre, i seguenti nomi occupati dal sistema non possono essere utilizzati:

- Nomi di parole chiavi: p. es. CONST, END_CONST, BEGIN
- Nomi di operazioni: p. es. AND, XOR
- Nomi di identificatori predefiniti: p. es. nomi per tipi di dati come BOOL, STRING, INT
- Nomi di costanti predefinite TRUE e FALSE
- Nomi di funzioni standard: p. es. ABS, ACOS, ASIN, COS, LN
- Nomi di identificazioni assolute o di operandi per dati globali del sistema: p. es. EB, EW, ED, AB, AW, AD MB, MD

Impiego di IDENTIFICATORE

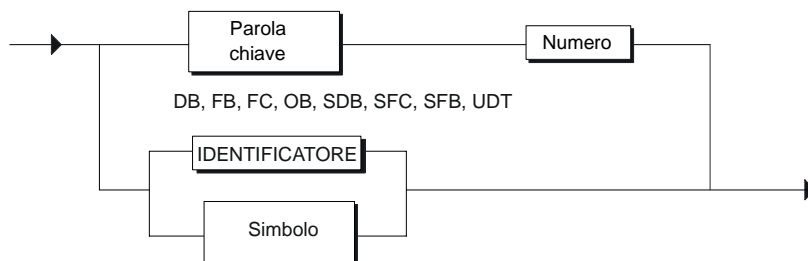
La tabella seguente indica in quali casi l'utente può indicare nomi corrispondenti alla regola IDENTIFICATORE.

IDENTIFICATORE	Descrizione	Regola
Nome del blocco	Nome simbolico per blocchi	IDENTIFICAZIONE BLOCCO, richiamo di funzione
Nomi per temporizzatori e contatori	Nome simbolico per temporizzatori e contatori	IDENTIFICAZIONE TEMPORIZZATORE, IDENTIFICAZIONE CONTATORE
Nome di attributo	Nome per un attributo	Assegnazione di attributo
Nome di costante	Dichiarazione costante simbolica, impiego	Blocco costanti, Costante
Etichetta di salto	Dichiarazione di etichetta di salto, impiego etichetta di salto	Parte istruzioni del blocco etichette di salto, istruzione GOTO
Nome di variabile	Dichiarazione di variabile temporanea o statica	Dichiarazione di variabile, Variabile semplice, Variabile strutturata
Nome di istanza locale	Convenzione per istanze locali	Dichiarazione di istanza, nome di richiamo di FB

IDENTIFICAZIONE BLOCCO

La regola IDENTIFICAZIONE BLOCCO è un caso in cui IDENTIFICATORE e simbolo possono essere impiegati in modo alternativo:

IDENTIFICATORE STANDARD



Le regole IDENTIFICAZIONE TEMPORIZZATORE e IDENTIFICAZIONE CONTATORE sono analoghe a IDENTIFICAZIONE BLOCCO. Per tali regole valgono gli stessi criteri.

15.1.11 Costanti predefinite e flag

Entrambe le tabelle valgono per il mnemonico tedesco e inglese.

Costanti

Mnemonico	Descrizione
FALSE	Costante booleana predefinita (costante standard) con valore 0. Relativamente alla logica significa che la condizione non è soddisfatta.
TRUE	Costante booleana predefinita (costante standard) con valore 1. Relativamente alla logica significa che la condizione non è soddisfatta.

Flag

Mnemonico	Descrizione
EN	Flag per l'abilitazione del blocco
ENO	Flag di errore del blocco
OK	Viene impostato su FALSE quando un'istruzione è stata eseguita in modo errato.

15.2 Regole lessicali

15.2.1 Regole lessicali

Le regole lessicali descrivono la struttura degli elementi (token) che vengono elaborati durante l'analisi lessicale del compilatore. Perciò, la modalità di scrittura non è in formato libero, cioè le regole devono essere rigorosamente rispettate. Ciò significa in modo particolare:

- non è consentito inserire caratteri di formattazione
- non è consentito inserire commenti al blocco e alla riga
- non è consentito inserire attributi per gli identificatori

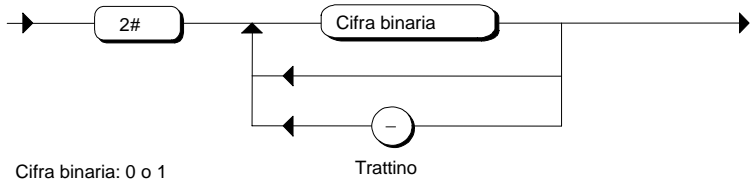
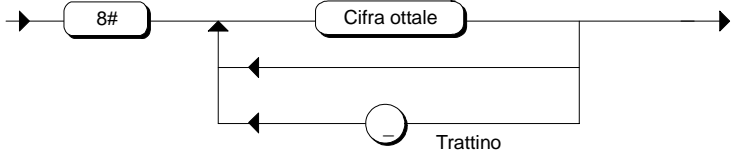
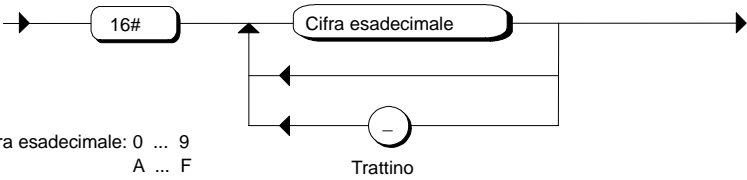
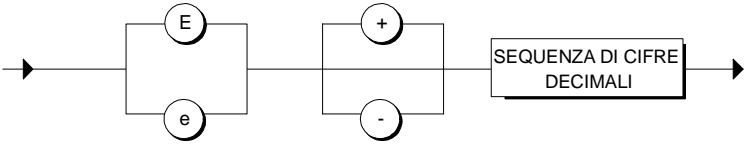
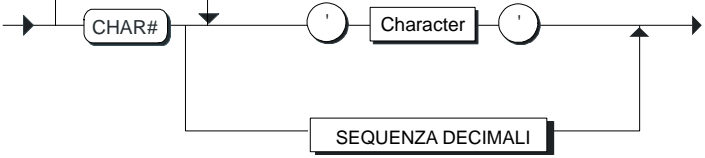
15.2.2 Identificatori

Regola	Diagramma sintattico
Identificatore	<p>IDENTIFICATORE</p>
Nome del blocco	<p>IDENTIFICATORE STANDARD</p>

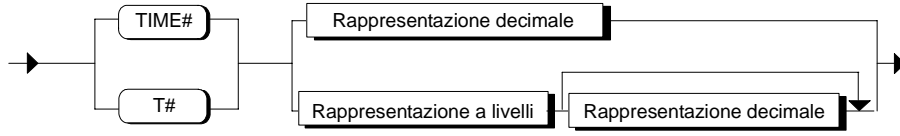
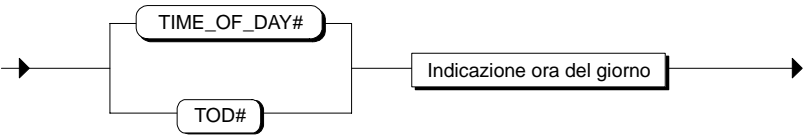
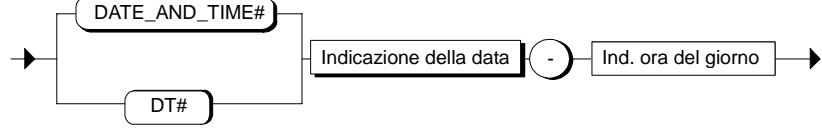
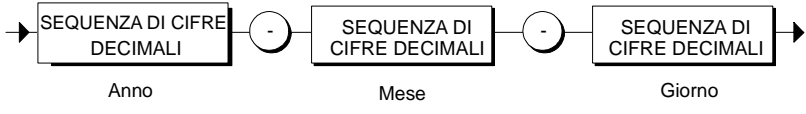
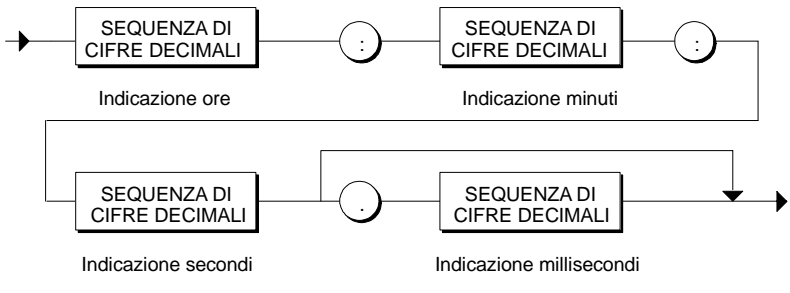
Regola	Diagramma sintattico
IDENTIFICAZIONE TEMPORIZZATORE	
IDENTIFICAZIONE CONTATORE	
Parola chiave del blocco	
Simbolo	
Numero	

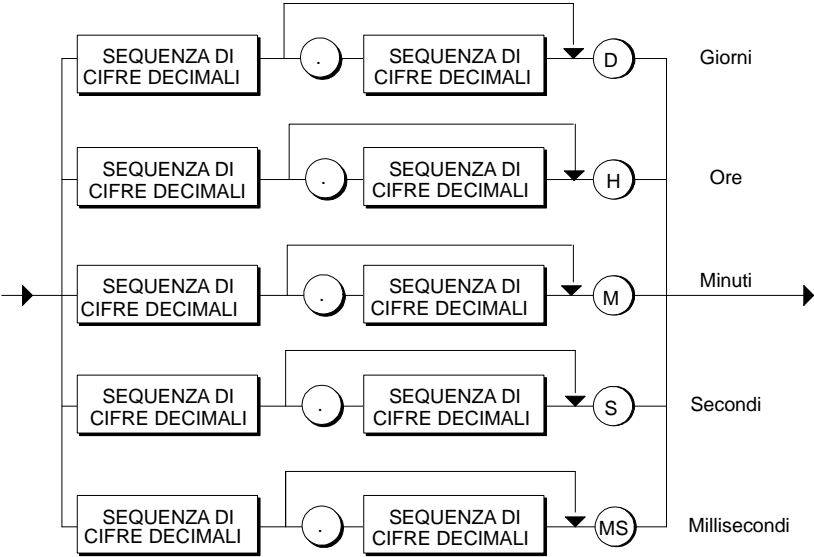
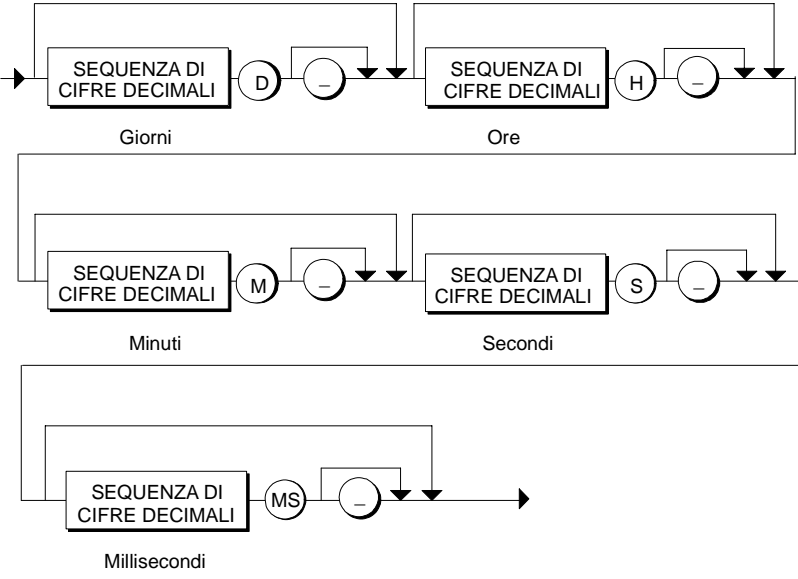
15.2.3 Costanti

Regola	Diagramma sintattico
Costante a bit	<p>COSTANTE DI BIT</p> <p>(1) solo nel tipo di dati BYTE</p>
Costante di numero intero	<p>COSTANTE DI NUMERO INTERO</p> <p>(1) solo nel tipo di dati INT</p>
Costante di numero reale	<p>COSTANTE DI NUMERO REALE</p>
Sequenza di cifre decimali	<p>Sequenza di cifre decimali</p> <p>Cifra decimale: 0-9</p> <p>Trattino</p>

Regola	Diagramma sintattico
Sequenza di cifre binarie	 <p>Cifra binaria: 0 o 1</p> <p>Trattino</p>
Sequenza di cifre ottali	<p>Sequenza di cifre ottali</p>  <p>Trattino</p>
Sequenza di cifre esadecimali	<p>CIFRA ESADECIMALE</p>  <p>Cifra esadecimale: 0 ... 9 A ... F</p> <p>Trattino</p>
Esponente	<p>Esponente</p>  <p>SEQUENZA DI CIFRE DECIMALI</p>
Costante Char	<p>COSTANTE CHAR</p>  <p>SEQUENZA DECIMALI</p>

Regola	Diagramma sintattico
Costante String	<p>COSTANTE STRING</p>
Caratteri	<p>Carattere</p> <p>Rappresentazione sostitutiva in codice esadecimale</p> <p>* P=avanzamento pagina L=avancamento riga R=ritorno carella T=tabulatore N=nuova riga</p> <p>** \$00 non consentito</p>
Interruzione di stringa	<p>Interruzione di stringa</p> <p>Spazio (blank) A capo automatico (linefeed) Ritorno del carrello (carriage return) Rimpagina (formfeed, page) oppure Tabulatore orizzontale (tabulator)</p>
Data	<p>DATA</p>

Regola	Diagramma sintattico
Intervallo	<p>DURATA</p>  <p>- Ogni unità di tempo (p. es. ore, minuti) può essere indicata una sola volta. - La sequenza - giorni, ore, minuti, secondi, millisecondi - deve essere rispettata</p>
Ora del giorno	<p>ORA DEL GIORNO</p> 
Data e ora	<p>DATA E ORA</p> 
Indicazione data	<p>Indicazione della data</p>  <p>Anno Mese Giorno</p>
Indicazione ora del giorno	<p>Indicazione ora del giorno</p>  <p>Indicazione ore Indicazione minuti</p> <p>Indicazione secondi Indicazione millisecondi</p>

Regola	Diagramma sintattico
<p>Rappresentazione decimale</p>	<p>Rappresentazione decimale</p>  <p>L'accesso alla rappresentazione decimale è possibile solo con unità di tempo non ancora definite.</p>
<p>Rappresentazione a livelli</p>	<p>Rappresentazione a livelli</p> 

15.2.4 Indirizzamento assoluto

Regola	Diagramma sintattico
ACCESSO SEMPLICE ALLA MEMORIA	
ACCESSO INDICIZZATO ALLA MEMORIA	
Identificazione di operando per la memoria	<p>Identificazione di operando</p>
Accesso assoluto al DB	
Accesso indicizzato al DB	
Accesso strutturato al DB	

Regola	Diagramma sintattico
IDENTIFICAZIONE operando DB	
Prefisso di memoria	
Prefisso di lunghezza per memoria e DB	
Indirizzo per memoria e DB	Indirizzo
Accesso a istanza locale	

15.2.5 Commenti


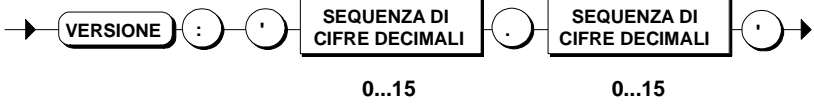
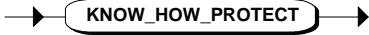
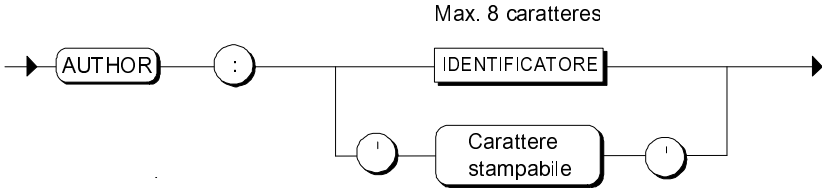
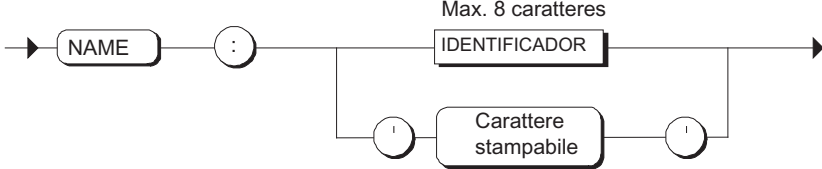
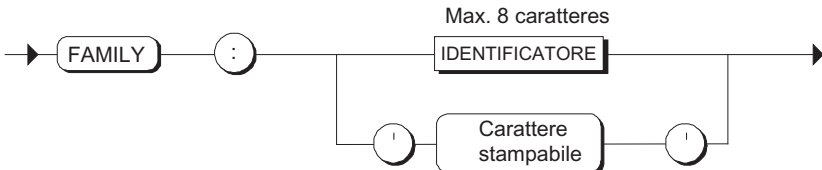
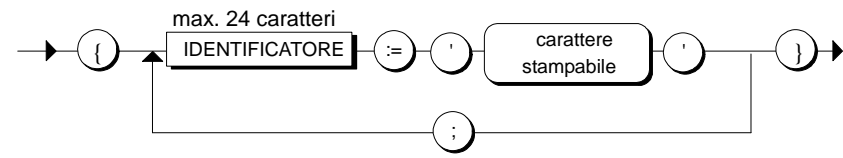
Seguono i punti più importanti da osservare quando si inseriscono dei commenti.

- L'annidamento dei commenti è ammesso se è attiva l'opzione "Autorizza annidamento commenti".
- L'inserimento di commenti è possibile in qualsiasi posizione nelle regole sintattiche, ma non nelle regole lessicali.

Regola	Diagramma sintattico
Commento	
Riga di commento	<p>Commento a una riga</p> <p>Ritorno del carrello (carriage return)</p>
Blocco di commenti	<p>Blocco di commento</p>

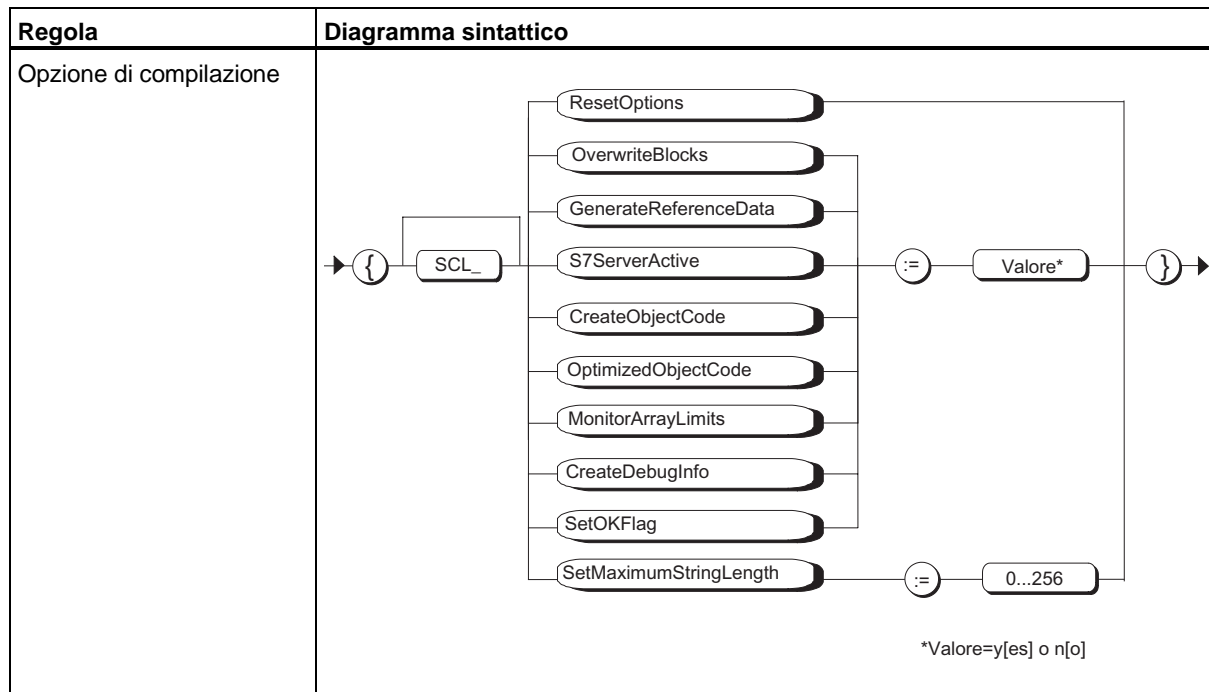
15.2.6 Attributi di blocco

Gli attributi di blocco possono essere situati con la seguente sintassi dopo l'IDENTIFICAZIONE DEL BLOCCO e prima della dichiarazione del primo blocco di variabili o parametri.

Regola	Diagramma sintattico
Titolo	 <pre> graph LR TITLE[TITLE] --> EQ[=] EQ --> QUOTE[''] QUOTE --> CS[Carattere stampabile] CS --> QUOTE2[''] QUOTE2 --> END[] </pre>
Versione	 <pre> graph LR VERSIONE[VERSIONE] --> COLON1[:] COLON1 --> QUOTE1[''] QUOTE1 --> SD1[SEQUENZA DI CIFRE DECIMALI] SD1 --> DOT1[.] DOT1 --> SD2[SEQUENZA DI CIFRE DECIMALI] SD2 --> QUOTE2[''] QUOTE2 --> END[] SD1 --- R1[0...15] SD2 --- R2[0...15] </pre>
Protezione del blocco	 <pre> graph LR KNOW_HOW_PROTECT[KNOW_HOW_PROTECT] --> END[] </pre>
Autore	 <pre> graph LR AUTHOR[AUTHOR] --> COLON[:] COLON --> IDENTIFICATORE[IDENTIFICATORE] IDENTIFICATORE --> END[] COLON --> QUOTE[''] QUOTE --> CS[Carattere stampabile] CS --> QUOTE2[''] QUOTE2 --> END IDENTIFICATORE --- R1[Max. 8 caratteres] </pre>
Nome	 <pre> graph LR NAME[NAME] --> COLON[:] COLON --> IDENTIFICADOR[IDENTIFICADOR] IDENTIFICADOR --> END[] COLON --> QUOTE[''] QUOTE --> CS[Carattere stampabile] CS --> QUOTE2[''] QUOTE2 --> END IDENTIFICADOR --- R1[Max. 8 caratteres] </pre>
Famiglia del blocco	 <pre> graph LR FAMILY[FAMILY] --> COLON[:] COLON --> IDENTIFICATORE[IDENTIFICATORE] IDENTIFICATORE --> END[] COLON --> QUOTE[''] QUOTE --> CS[Carattere stampabile] CS --> QUOTE2[''] QUOTE2 --> END IDENTIFICATORE --- R1[Max. 8 caratteres] </pre>
Attributi di sistema per blocchi	<p>Attributi di sistema per blocchi</p>  <pre> graph LR LBRACE[{ }] --> IDENTIFICATORE[IDENTIFICATORE] IDENTIFICATORE --> EQ[:=] EQ --> QUOTE[''] QUOTE --> CS[Carattere stampabile] CS --> QUOTE2[''] QUOTE2 --> RBRACE[}] IDENTIFICATORE --> SEMI[;] SEMI --> RBRACE IDENTIFICATORE --- R1[max. 24 caratteri] </pre>

15.2.7 Opzioni di compilazione

Le opzioni di compilazione si trovano nella sorgente al di fuori dei limiti del blocco in una riga propria. Non viene fatta distinzione tra maiuscolo e minuscolo.



15.3 Regole sintattiche

15.3.1 Regole sintattiche

Sulla base delle regole lessicali, le regole sintattiche descrivono la struttura di S7-SCL. Nell'ambito di tali regole l'utente può creare il proprio programma S7-SCL in formato libero:

Ogni regola possiede un nome che viene preposto. Se la regola viene utilizzata in una regola sovraordinata, questo nome appare in un rettangolo.

Se il nome nel rettangolo è scritto in caratteri maiuscoli si tratta di un token che viene descritto nelle regole lessicali.

Se le regole sono racchiuse in una cornice arrotondata o in un cerchio, le informazioni sul loro contenuto sono riportate nel capitolo "Descrizione formale del linguaggio".

Libertà di formato

Libertà di formato significa:


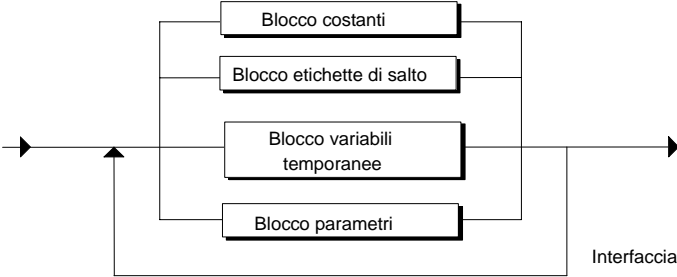
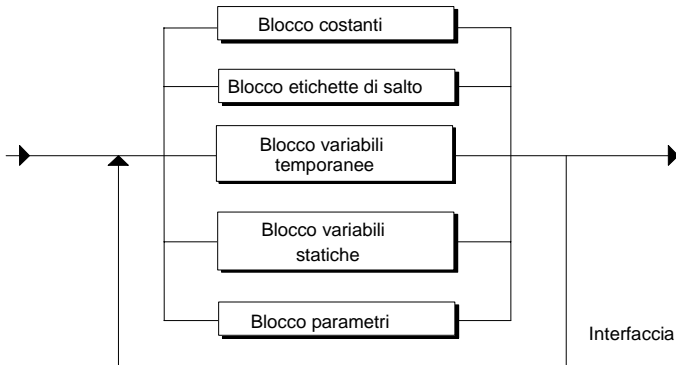

- Si possono inserire caratteri di formattazione in qualsiasi posizione
- Si possono inserire commenti di riga nei blocchi di commento

15.3.2 Suddivisione delle sorgenti S7-SCL

Regola	Diagramma sintattico
Programma S7-SCL	
Unità di programma S7-SCL	
Blocco organizzativo	<p>Blocco organizzativo</p>
Funzione	<p>Funzione</p> <p>Si osservi che nelle funzioni senza VOID nella parte istruzioni, il valore di ritorno deve essere assegnato al nome della funzione!</p>

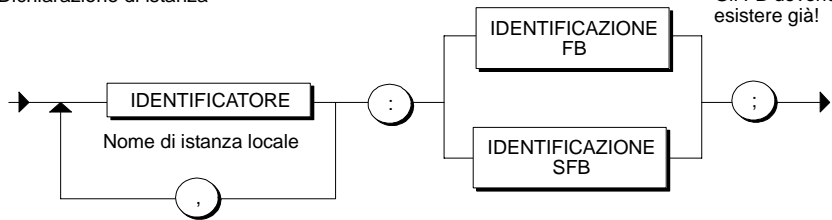
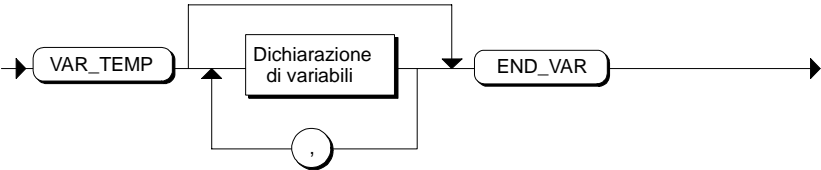
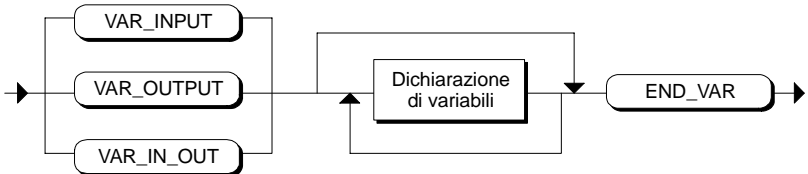
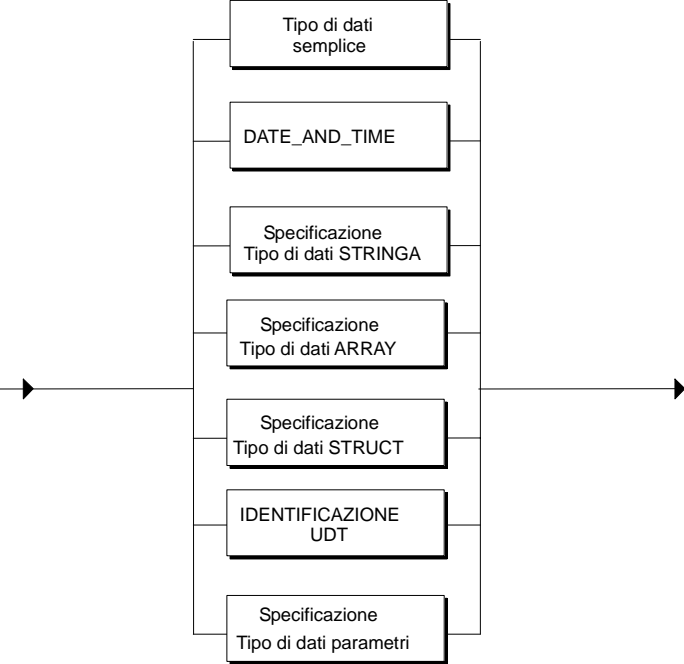
Regola	Diagramma sintattico
Blocco funzionale	<p>Blocco funzionale</p> <pre> graph LR In(()) --> P1(()) P1 --> PROGRAM P1 --> FB[FUNCTION_BLOCK] PROGRAM --> ID[IDENTIFICAZIONE FB] FB --> ID ID --> PD[Parte dichiarazioni FB] P1 --> B[BEGIN] B --> PI[Parte istruzioni] PD --> EP[END_PROGRAM] PI --> EP PI --> EFB[END_FUNCTION_BLOCK] EP --> Out(()) EFB --> Out </pre>
Blocco dati	<p>Blocco dati</p> <pre> graph LR In(()) --> P1(()) P1 --> DB[DATA_BLOCK] DB --> ID[IDENTIFICAZIONE DB] ID --> PD[Parte dichiarazioni DB] P1 --> B[BEGIN] B --> PA[Parte assegnazioni DB] PD --> EDB[END_DATA_BLOCK] PA --> EDB EDB --> Out(()) </pre>
Tipo di dati definito dall'utente	<p>Tipo di dati definito dall'utente</p> <pre> graph LR In(()) --> T[TYPE] T --> ID[IDENTIFICAZIONE UDT] ID --> S[Specificazione del tipo dati strutturati] S --> ET[END_TYPE] ET --> Out(()) </pre>

15.3.3 Struttura della parte convenzioni

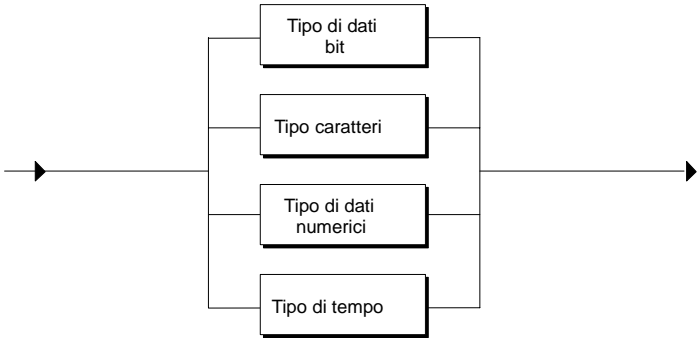
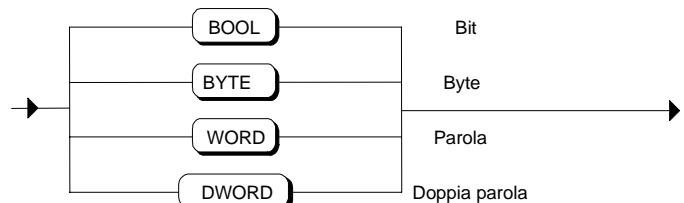
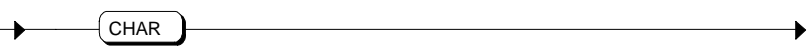
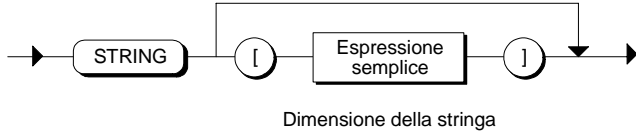
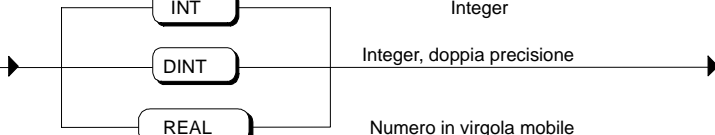
Regola	Diagramma sintattico
Parte convenzioni dell'OB	
Parte convenzioni della FC	
Parte convenzioni dell'FB	
Parte convenzioni del DB	

Regola	Diagramma sintattico
Parte assegnazioni DB	<p>Parte assegnazioni DB</p> <p>* modalità di scrittura AWL</p>
Blocco costanti	<p>Blocco costanti</p>
Blocco etichette di salto	<p>Blocco etichette di salto</p>
Blocco di variabili statiche	<p>Blocco variabili statico</p> <p>* solo negli FB</p>

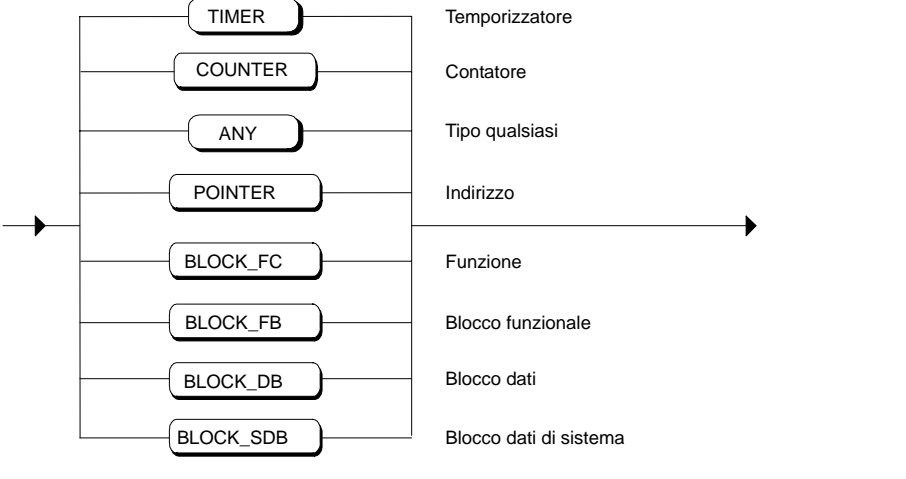
Regola	Diagramma sintattico
Dichiarazione di variabili e parametri	<p>Nome di variabile, nome di parametro o nome di componente</p> <p>1) Attributi di sistema per parametri</p> <p>max. 24 caratteri</p> <p>carattere stampabile</p> <p>Nome di componenti all'interno di strutture</p> <p>Non durante l'inizializzazione</p>
Inizializzazione tipo di dati	<p>COSTANTE</p> <p>ELENCO INIZIAL. DI CAMPO</p>
Lista inizializzazione di campo	<p>SEQUENZA DI NUMERI DECIMALI</p> <p>FATTORE DI REPETIZIONE</p> <p>ELENCO DI REPETIZIONE COSTANTI</p> <p>COSTANTE</p> <p>LISTA INIZIALIZZAZIONE DI CAMPO</p> <p>COSTANTE</p> <p>ELENCO DI REPET. DI COSTANTI</p>

Regola	Diagramma sintattico
Dichiarazione di istanze (solo nella componente VAR di un FB)	<p>Dichiarazione di istanza</p>  <p>Gli FB devono esistere già!</p>
Blocco di variabili temporanee	<p>Blocco di variabili temporanee</p>  <p>Non è possibile alcuna inizializzazione</p>
Blocco parametri	<p>Blocco di parametri</p>  <p>Inizializzazione possibile solo per VAR_INPUT e VAR_OUTPUT</p>
Specificazione tipo di dati	

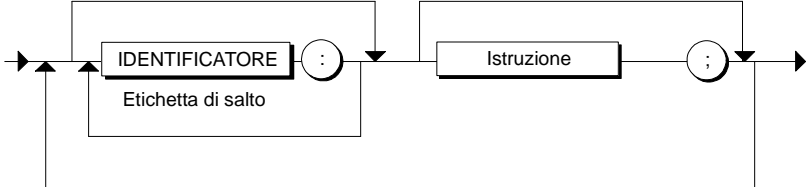

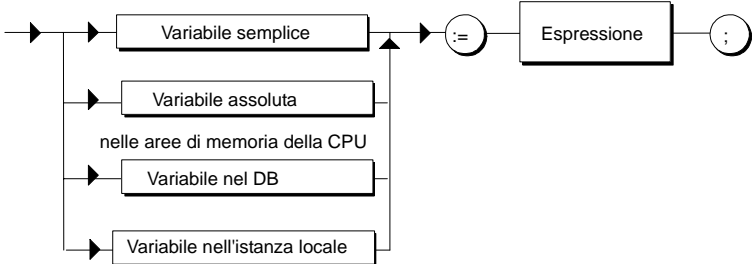
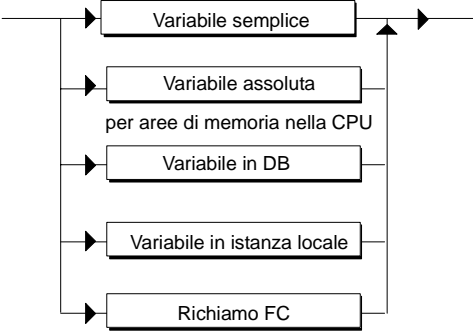
15.3.4 Tipi di dati in S7-SCL

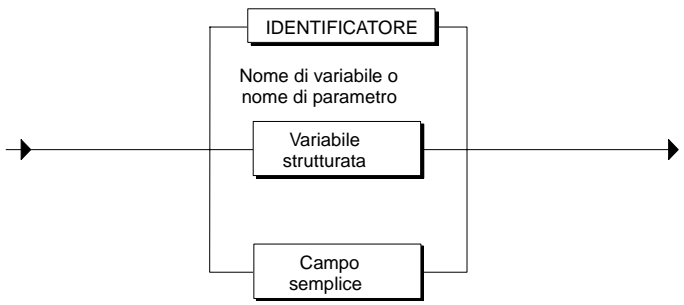
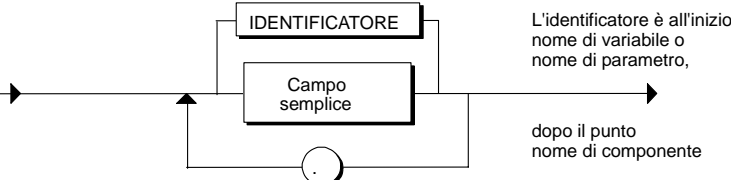
Regola	Diagramma sintattico
Tipo di dati semplice	
Tipo di dati bit	
Tipo del carattere	
Specificazione tipo di dati STRING	<p data-bbox="501 1276 820 1308">Specificazione del tipo dati STRING</p> 
Tipo di dati numerici	

Regola	Diagramma sintattico
Tipo di tempo	
DATE_AND_TIME	<p>DATE_AND_TIME</p>
Specificazione tipo di dati ARRAY	<p>Specificazione del tipo di dati ARRAY</p>
Specificazione tipo di dati STRUCT Non dimenticare di mettere un punto e virgola dopo la parola chiave END_STRUCT!	<p>Specificazione del tipo di dati STRUCT</p>
Dichiarazione componenti	

Regola	Diagramma sintattico																		
Specificazione tipo di dati parametri	 <p>Diagramma sintattico per la specificazione del tipo di dati dei parametri. Il diagramma mostra una lista di termini tecnici in rettangoli sovrapposti, con una freccia che li collega a una lista di descrizioni in italiano. Una freccia orizzontale esce dal gruppo di rettangoli e punta verso la destra.</p> <table border="1"> <thead> <tr> <th>Termine Tecnico</th> <th>Descrizione in Italiano</th> </tr> </thead> <tbody> <tr> <td>TIMER</td> <td>Temporizzatore</td> </tr> <tr> <td>COUNTER</td> <td>Contatore</td> </tr> <tr> <td>ANY</td> <td>Tipo qualsiasi</td> </tr> <tr> <td>POINTER</td> <td>Indirizzo</td> </tr> <tr> <td>BLOCK_FC</td> <td>Funzione</td> </tr> <tr> <td>BLOCK_FB</td> <td>Blocco funzionale</td> </tr> <tr> <td>BLOCK_DB</td> <td>Blocco dati</td> </tr> <tr> <td>BLOCK_SDB</td> <td>Blocco dati di sistema</td> </tr> </tbody> </table>	Termine Tecnico	Descrizione in Italiano	TIMER	Temporizzatore	COUNTER	Contatore	ANY	Tipo qualsiasi	POINTER	Indirizzo	BLOCK_FC	Funzione	BLOCK_FB	Blocco funzionale	BLOCK_DB	Blocco dati	BLOCK_SDB	Blocco dati di sistema
Termine Tecnico	Descrizione in Italiano																		
TIMER	Temporizzatore																		
COUNTER	Contatore																		
ANY	Tipo qualsiasi																		
POINTER	Indirizzo																		
BLOCK_FC	Funzione																		
BLOCK_FB	Blocco funzionale																		
BLOCK_DB	Blocco dati																		
BLOCK_SDB	Blocco dati di sistema																		

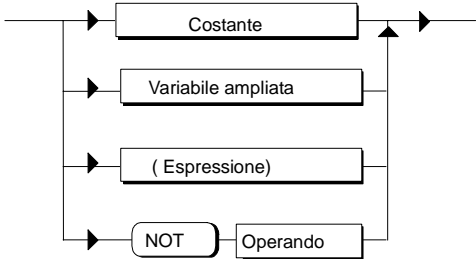
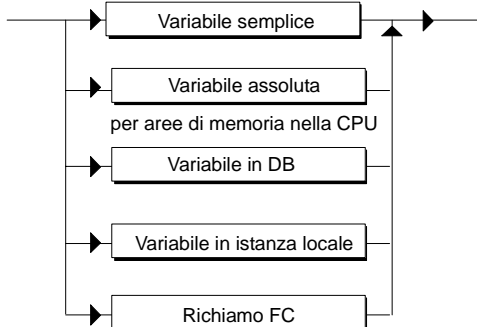
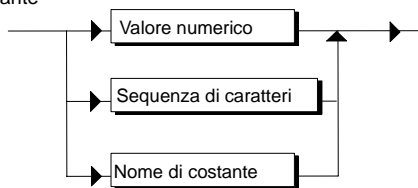
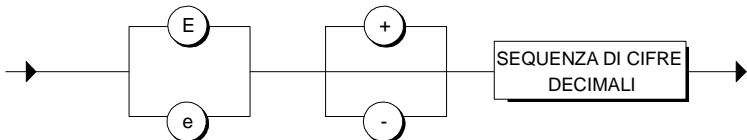
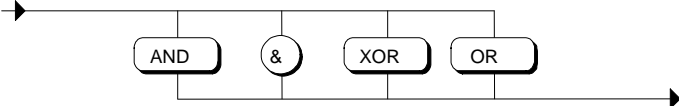
15.3.5 Parte istruzioni

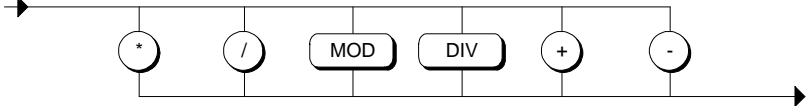
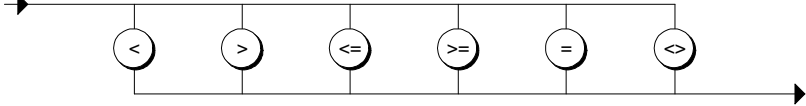
Regola	Diagramma sintattico
Parte istruzioni	<p>Parte istruzioni</p> 
Istruzione	<p>Istruzione</p> 
Assegnazione di valore	<p>Assegnazione di valore</p> 
Variabile ampliata	<p>Variabile ampliata</p> 

Regola	Diagramma sintattico
Variabile semplice	
Variabile strutturata	

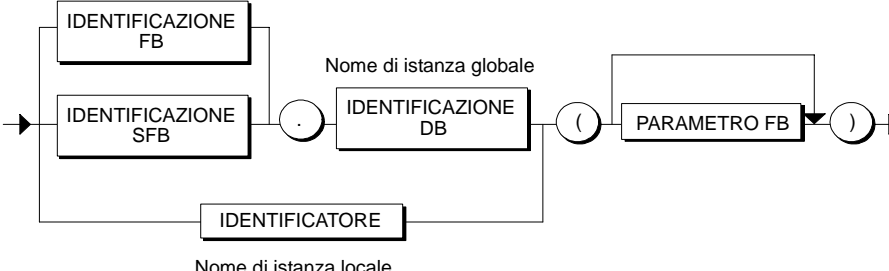
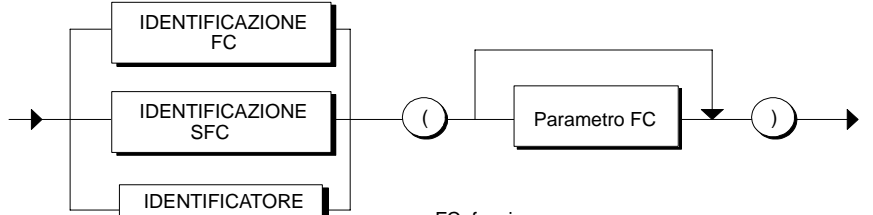
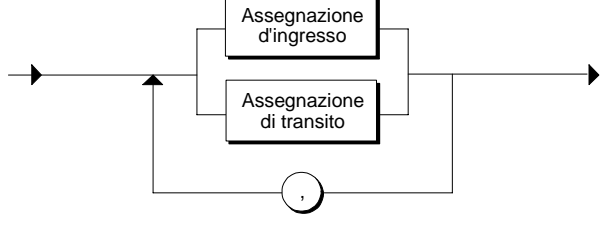
15.3.6 Assegnazione di valori

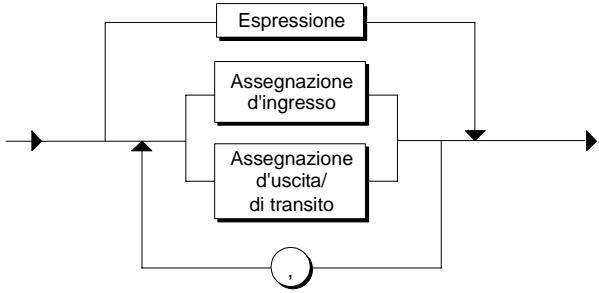
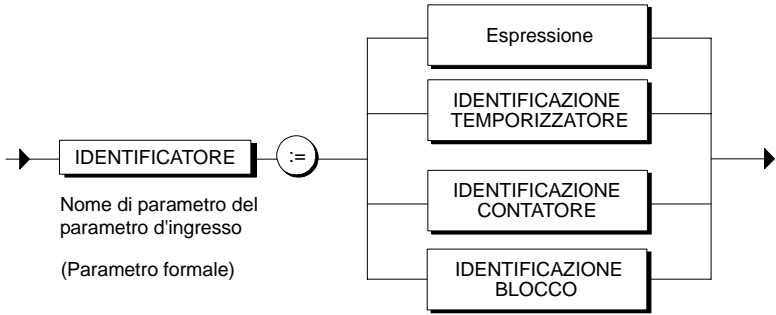
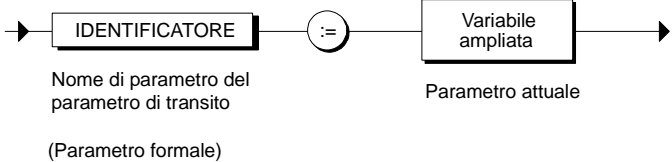
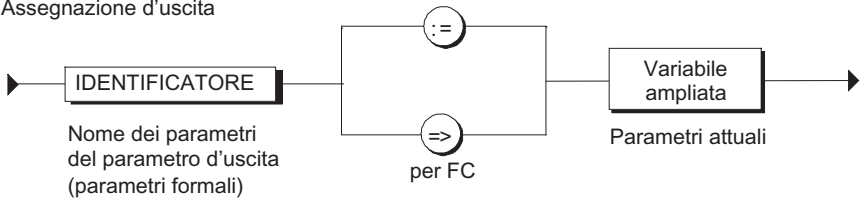
Regola	Diagramma sintattico
Espressione	<p>Espressione</p> <pre> graph LR E1[Espressione] --> O[Operando] E1 --> E2[Espressione] E2 --> OL[Operazioni logiche di base] E2 --> OC[Operazioni di confronto] E2 --> OA[Operazioni aritmetiche di base] OL --> E3[Espressione] OC --> E3 OA --> E3 E1 --> E4[Espressione] E4 --> P[Potenza **] P --> E5[Esponente] E5 --> E6[Espressione] E1 --> U1[+] U1 --> PU[Più unario] U1 --> MU[Meno unario] U1 --> N[NOT] N --> NE[Negazione] PU --> E7[Espressione] MU --> E7 NE --> E7 E1 --> P1[(] P1 --> E8[Espressione] E8 --> P2[)] </pre>
Espressione semplice	<pre> graph LR ES1[Espressione semplice] --> A1[+] ES1 --> A2[-] A1 --> MS[Moltiplicazione semplice] A2 --> MS MS --> Out1[] </pre>
Moltiplicazione semplice	<pre> graph LR MS1[Moltiplicazione semplice] --> M1[*] MS1 --> D1[/] MS1 --> DIV1[DIV] MS1 --> MOD1[MOD] MS1 --> N1[-] N1 --> C1[Costante] N1 --> ES2[Espressione semplice] C1 --> Out2[] ES2 --> Out2 </pre>

Regola	Diagramma sintattico
Operando	
Variabile ampliata	<p>Variabile ampliata</p> 
Costante	<p>Costante</p> 
Esponente	<p>Esponente</p> 
Operazione logica di base	

Regola	Diagramma sintattico
Operazione aritmetica di base	<p>Operazione aritmetica di base</p> 
Operazione di confronto	<p>Operazione di confronto</p> 

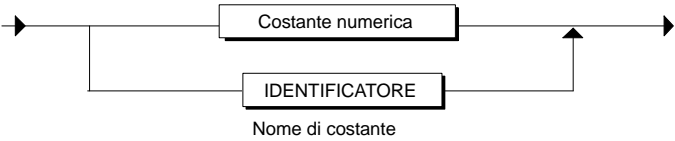
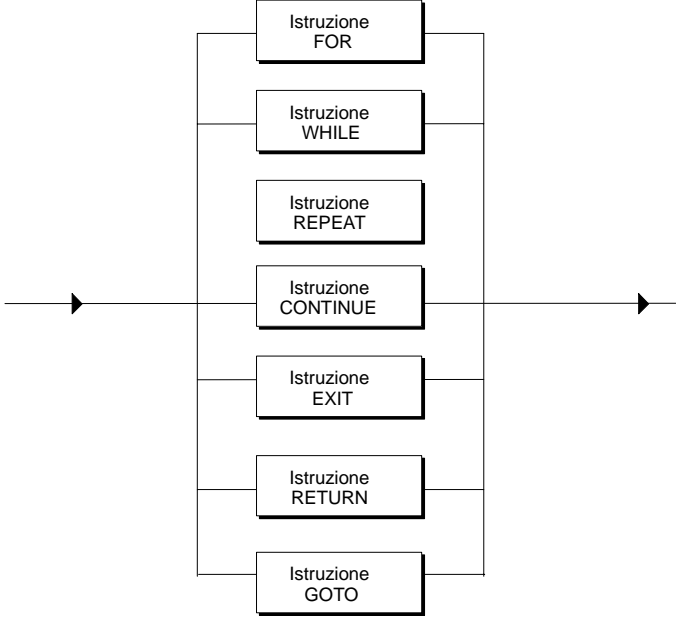
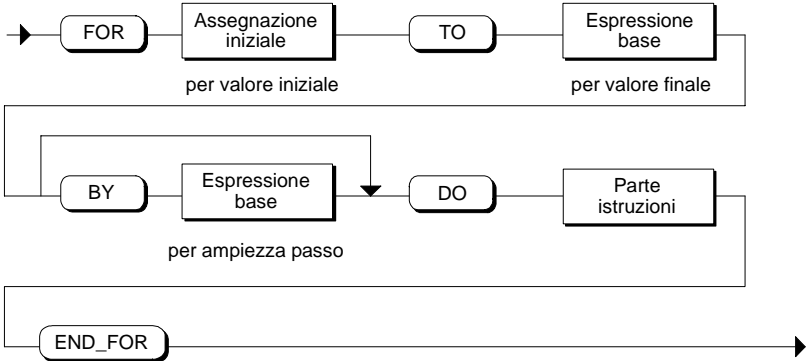
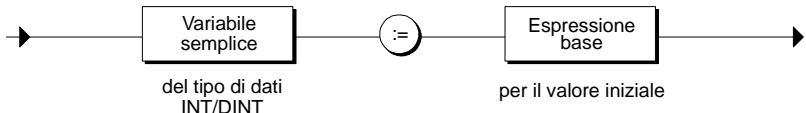
15.3.7 Richiamo di funzioni e blocchi funzionali

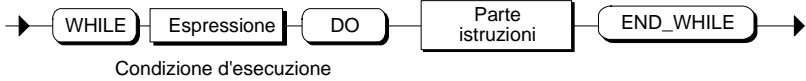
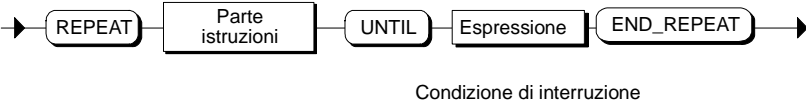



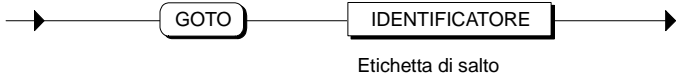
Regola	Diagramma sintattico
<p>Richiamo di FB</p>	<p>Richiami di FB</p> <p>FB: Blocco funzionale SFB: Blocco funzionale di sistema</p> 
<p>Richiamo della funzione</p>	<p>Richiamo di funzione</p>  <p>FC: funzione SFC: funzione di sistema funzione standard realizzata nel compilatore</p>
<p>Parametri FB</p>	<p>Parametri FB</p> 

Regola	Diagramma sintattico
Parametri FC	<p>Parametri FC</p> 
Assegnazione d'ingresso	<p>Assegnazione d'ingresso</p> <p>Parametro attuale</p> 
Assegnazione di ingresso/uscita	<p>Assegnazione di transito</p> 
Assegnazione di uscita	<p>Assegnazione d'uscita</p> 

15.3.8 Istruzioni di controllo

Regola	Diagramma sintattico
<p>Istruzione IF</p> <p>Non dimenticare di mettere un punto e virgola dopo la parola chiave END_IF!</p>	<p>Istruzione IF</p> <pre> graph LR Start(()) --> IF[IF] IF --> E[Espressione Condizione] E --> T1[THEN] T1 --> P1[Parte istruzioni] P1 --> ELSIF[ELSIF] ELSIF --> E2[Espressione Condizione] E2 --> T2[THEN] T2 --> P2[Parte istruzioni] P2 --> ELSE[ELSE] ELSE --> P3[Parte istruzioni] P1 --> END_IF[END_IF] P2 --> END_IF P3 --> END_IF END_IF --> End(()) </pre>
<p>Istruzione CASE</p> <p>Non dimenticare di mettere un punto e virgola dopo la parola chiave END_CASE!</p>	<p>Istruzione CASE</p> <p>Espressione di selezione (Integer)</p> <pre> graph LR Start(()) --> CASE[CASE] CASE --> E[Espressione Valore] E --> OF[OF] OF --> LV[Lista dei valori] LV --> C1[:] C1 --> P1[Parte istruzioni] P1 --> ELSE[ELSE] ELSE --> C2[:] C2 --> P2[Parte istruzioni] P1 --> END_CASE[END_CASE] P2 --> END_CASE END_CASE --> End(()) </pre>
<p>Lista dei valori</p>	<p>Lista dei valori</p> <p>Numero intero</p> <pre> graph LR Start(()) --> V1[Valore] V1 --> Dots[...] Dots --> V2[Valore] V2 --> Comma[,] Comma --> V3[Valore] V3 --> End(()) </pre>

Regola	Diagramma sintattico
Lista dei valori	
Istruzioni di ripetizione e istruzioni di salto	
<p>Istruzione FOR</p> <p>Non dimenticare di mettere un punto e virgola dopo la parola chiave END_FOR!</p>	<p>Istruzione FOR</p> 
Assegnazione iniziale	<p>Assegnazione iniziale</p> 

Regola	Diagramma sintattico
Istruzione WHILE Non dimenticare di mettere un punto e virgola dopo la parola chiave END_WHILE!	Istruzione WHILE 
Istruzione REPEAT Non dimenticare di mettere un punto e virgola dopo la parola chiave END_REPEAT!	Istruzione REPEAT 
Istruzione CONTINUE	Istruzione CONTINUE 
Istruzione RETURN	Istruzione RETURN 
Istruzione EXIT	Istruzione EXIT 
Salto di programma	Salto di programma 

16 Suggerimenti e strategie

Codice con tempo di esecuzione ottimizzato nell'accesso alle strutture dei blocchi dati

Per poter accedere più volte alla struttura di un blocco dati è consigliabile procedere nel seguente modo:

1. creare una variabile locale con il tipo della struttura.
2. Assegnare una volta alla variabile la struttura del blocco dati.
3. A questo punto la variabile può essere utilizzata più volte nel codice senza dover accedere nuovamente al DB.

Esempio:

```
DB100.campo[i].value :=  
DB100.campo[i].valore1 * DB100.campo[i].valore2 /  
DB100.campo[i].valore3 ;
```

Se si programma come indicato di seguito, questo esempio richiede poco spazio di memoria e un tempo di esecuzione breve:

```
VAR_TEMP  
  tmp : STRUCT  
      value : REAL;  
      valore1 : REAL;  
      valore2 : REAL;  
      valore3 : REAL;  
  END_STRUCT;  
END_VAR  
tmp := DB100.campo[i];  
DB100.campo[i].value := tmp.valore1 * tmp.valore2 / tmp.valore3;
```

Nota

Con VAR_TEMP si crea una variabile nello stack della CPU. Poiché nelle CPU più piccole può verificarsi un overflow dello stack è consigliare non utilizzare spesso le variabili temporanee.

Problemi di allocazione degli stack L nelle CPU di piccole dimensioni

I problemi di allocazione degli stack L sono dovuti alla dimensione ridotta degli stack nelle CPU meno potenti. Nella maggior parte dei casi il problema può essere facilmente risolto procedendo come indicato di seguito:

- utilizzare poco le variabili temporanee (componente VAR_TEMP o VAR).
- Non dichiarare variabili con tipo di dati superiore e ridurre al minimo il numero di variabili con tipo di dati elementare.
- Utilizzare variabili statiche:
 - quando si programma un FB si può utilizzare la componente VAR invece di VAR_TEMP
 - se si programma un OB o una FC si deve utilizzare un blocco dati globale ai merker.
- Evitare le espressioni molto estese. Poiché durante l'elaborazione di tali espressioni il compilatore memorizza i risultati temporanei nello stack, la dimensione dello stack potrebbe risultare insufficiente in funzione del tipo e del numero di risultati.
Rimedio:
suddividere l'espressione in espressioni più piccole e assegnare i risultati temporanei a variabili esplicite.

Emissione di numeri reali durante il controllo

Se vengono emessi numeri reali non rappresentabili, la funzione di test "Controlla" può fornire i seguenti pattern:

Valore	Rappresentazione
+ infinity	1.#INRandom-digits
- infinity	-1.#INRandom-digits
Indefinite	digit.#INDrandom-digits
NaN digit.	#NANrandom-digits

Visualizzazione dei programmi in AWL

È possibile aprire un blocco S7-SCL negli editor AWL/KOP/FUP e visualizzare i comandi MC7 generati. Tuttavia, se non si apportano modifiche in AWL, si dovrà considerare quanto segue:

- non sempre i comandi MC7 visualizzati rappresentano un blocco AWL valido
- una compilazione corretta con il compilatore AWL richiede generalmente delle modifiche per le quali è necessario conoscere bene sia AWL che S7-SCL
- il blocco compilato con AWL avrà il codice di linguaggio AWL e non S7-SCL
- la sorgente S7-SCL e il codice MC7 non saranno più coerenti.

Elaborazione della data e dell'ora della sorgente, dell'interfaccia e del codice

Data e ora della sorgente

Una sorgente riporta sempre la data e l'ora dell'ultima modifica.

Data e ora del codice di blocco

I blocchi (FB, FC e OB) assumono sempre la data e l'ora di compilazione.

Data e ora dell'interfaccia di blocco

Si modifica la data e l'ora di un'interfaccia quando la sua struttura viene modificata, ovvero:

- la data e l'ora di un'interfaccia restano invariate se vengono apportate modifiche alla parte istruzioni, agli attributi, al commento, alla componente VAR_TEMP (nelle FC anche a VAR) o nei nomi dei parametri e delle variabili. Questo vale anche per le interfacce subordinate.
- La data e l'ora di un'interfaccia vengono aggiornate se cambia il tipo di dati o l'eventuale inizializzazione di un parametro o una variabile oppure se vengono eliminati o aggiunti parametri o variabili e se, nel caso delle multiistanze, cambia il nome dell'FB. Questo vale anche per le interfacce subordinate.

Divisione di due valori interi con risultato in formato REAL

Programmare in S7-SCL la seguente istruzione:

```
Fraction:=dividendo/divisore;
```

dove Fraction è un valore reale mentre dividendo e divisore sono valori interi.

Si noti che in queste operazioni il compilatore S7-SCL esegue implicitamente una conversione del tipo di dati per cui compila l'istruzione ora descritta nel seguente modo:

```
Fraction:=INT_TO_REAL(dividendo/divisore);
```

Ne risulta che la divisione fornisce sempre un valore arrotondato, ad es. $1/3 = 0$ o $3/2 = 1$.

Valore di ritorno delle funzioni standard e di sistema di STEP 7

Molte funzioni standard e di sistema di STEP 7 hanno un valore di funzione di tipo INT contenente un codice d'errore. Nel manuale di riferimento delle funzioni i codici d'errore sono indicati come costanti WORD nel formato "W#16#8093".

In S7-SCL viene sempre verificato se i tipi sono uguali per cui non è possibile avere contemporaneamente INT e WORD. La seguente interrogazione non darebbe quindi il risultato desiderato.

```
IF SFCxx(..) = 16#8093 THEN ...
```

È tuttavia possibile indicare al compilatore S7-SCL di gestire una costante WORD come fosse di tipo INT procedendo come indicato di seguito.

- Mediante tipizzazione delle costanti. In questo caso l'interrogazione sopra descritta sarà:

```
IF SFCxx(..) = INT#16#8093 THEN ...
```
- Mediante la funzione di conversione WORD_TO_INT(). L'interrogazione sopra descritta deve essere quindi formulata nel seguente modo:

```
IF SFCxx(..) = WORD_TO_INT(16#8093) THEN ...
```

Ricablaggio dei blocchi

Non è possibile ricablare i richiami nei blocchi S7-SCL utilizzando la funzione **Strumenti > Ricablaggio** di SIMATIC Manager. È necessario elaborare manualmente i richiami dei blocchi interessati nella sorgente S7-SCL.

Suggerimenti:

- Nella tabella dei simboli definire nomi simbolici per i blocchi e richiamare i blocchi in modo simbolico.
- Nella tabella dei simboli definire nomi simbolici per gli indirizzi assoluti (E, M, A ecc.) e utilizzarli nel programma.

A questo punto, se si desidera ricablare un blocco non si deve far altro che modificare l'assegnazione nella tabella dei simboli senza dover apportare modifiche alla sorgente S7-SCL.

Assegnazione di strutture con lunghezza di byte dispari

La lunghezza di una struttura viene sempre completata fino ai limiti della parola. Per porre una struttura su un numero di byte dispari S7-SCL mette a disposizione il costrutto AT:

Esempio:

```
VAR
theStruct : STRUCT
    twoBytes : ARRAY [0..1] OF BYTE;
    oneInt : INT
    oneByte : BYTE;
END_STRUCT;
fiveBytes AT theStruct : ARRAY[0..4] OF BYTE;
END_VAR
```

Laddove si necessita di 5 BYTE, occorre utilizzare l'identificatore fiveBytes. È quindi possibile accedere a theStruct in maniera strutturata tramite l'identificatore.

Valori limite per istruzioni FOR

Per programmare istruzioni FOR "sicure" che non rimangono in funzione senza mai arrestarsi, occorre prestare attenzione alle regole e ai valori limite seguenti:

Regole

FOR ii := inizio TO fine BY fase DO

Se...	...allora	Osservazione
inizio < fine	fine < (PMAX - fase)	La variabile di esecuzione ii opera nella direzione positiva.
inizio > fine AND fase < 0	fine > (NMAX - fase)	La variabile di esecuzione ii opera nella direzione negativa.

Valori limite

Per entrambi i tipi di dati possibili valgono valori limite diversi:

Tipo di dati	PMAX	NMAX
ii di tipo INT	32_767	-32_768
ii di tipo DINT	2_147_483_647	-2_147_483_648

Glossario

Accesso

Per accedere ad una variabile dichiarata utilizzando un tipo di dati diverso è possibile definire degli accessi alla variabile o ad aree al suo interno. Un accesso può essere utilizzato nel blocco come qualsiasi altra variabile ed eredita le proprietà della variabile a cui punta, solo il tipo di dati è nuovo.

ALT

Lo stato di funzionamento ALT viene raggiunto dallo stato di funzionamento RUN mediante richiesta del dispositivo di programmazione. In questo stato di funzionamento sono possibili speciali funzioni di test.

Area di memoria

In SIMATIC S7 un'unità centrale possiede tre aree di memoria: la memoria di caricamento, la memoria di lavoro e l'area di sistema.

Assegnazione

Meccanismo di attribuzione di un valore ad una variabile.

Attributo

Un attributo è una proprietà che può essere associata p. es. ad un'identificazione di blocco o ad un nome di variabile. In SCL esistono attributi per le seguenti indicazioni: titolo blocco, versione, protezione blocco, autore, nome di blocco, famiglia di blocco.

BCD

Cifra decimale codificata in binario. In STEP 7, all'interno della CPU l'indicazione di temporizzatori e contatori avviene solo in formato BCD.

Blocchi organizzativi (OB)

I blocchi organizzativi costituiscono l'interfaccia fra il sistema operativo della CPU e il programma utente. Nei blocchi organizzativi viene definita la sequenza di elaborazione del programma utente.

Blocco dati (DB)

I blocchi dati sono blocchi contenenti i dati e i parametri elaborati dal programma. Al contrario degli altri blocchi i DB non contengono istruzioni.

Blocco dati di istanza (DB di istanza)

Un blocco dati di istanza memorizza i parametri formali e i dati statici locali dei blocchi funzionali. Un blocco dati di istanza può essere assegnato ad un richiamo di FB o ad una gerarchia di richiamo di blocchi funzionali.

Blocco dati di sistema (SDB)

I blocchi dati di sistema sono aree di dati della CPU S7 contenenti impostazioni di sistema e parametri di unità. I blocchi dati di sistema vengono generati e modificati con il software di base STEP 7.

Blocco di codice

In SIMATIC S7 un blocco di codice è un blocco che contiene una parte del programma utente STEP 7. Un blocco dati contiene invece esclusivamente dei dati. Esistono i seguenti blocchi di codice: blocchi organizzativi (OB), blocchi funzionali (FB), funzioni (FC), blocchi funzionali di sistema (SFB) e funzioni di sistema (SFC).

Blocco funzionale (FB)

Come stabilito dalla norma IEC 1131-3 un blocco funzionale (FB) è un blocco di codice con dati statici. Poiché un FB dispone di una memoria (blocco dati di istanza), si può accedere ai suoi parametri (p. es. uscite) in qualsiasi momento e in qualsiasi punto del programma utente.

Blocco funzionale di sistema (SFB)

Un blocco funzionale di sistema (SFB) è un blocco funzionale integrato nel sistema operativo della CPU che in caso di necessità può essere richiamato nel programma utente STEP 7.

Blocco

Per la loro funzione, struttura e scopi d'impiego, i blocchi sono parti ben definite di un programma utente. In STEP 7 esistono blocchi di codice (FB, FC, OB, SFC, SFB), blocchi dati (DB, SDB) e tipi di dati definiti dall'utente (UDT).

Campo

Un campo (ARRAY) è un tipo di dati composto, contenente elementi di dati dello stesso tipo. Questi elementi di dati possono a loro volta essere del tipo semplice o composto.

Caricamento nel sistema di destinazione

Trasferimento di oggetti caricabili (ad es. i blocchi di codice) dal dispositivo di programmazione nella memoria di caricamento di una CPU.

Classe di blocchi

I blocchi vengono suddivisi in due classi in base al loro contenuto: blocchi di codice e blocchi dati.

Commento di blocco

Informazioni supplementari su un blocco (p. es. spiegazioni relative al processo automatizzato) che non possono essere caricate nella memoria di lavoro dei sistemi di automazione SIMATIC S7.

Commento

Costruzione del linguaggio che consente di inserire in un programma un testo esplicativo che non influisce in alcun modo sull'esecuzione del programma.

Compilatore SCL

Il compilatore SCL è un compiler batch, con il quale il programma editato in precedenza (sorgente SCL) viene compilato in codice macchina MC7. I blocchi così generati vengono depositati nel programma S7 nel contenitore "Blocchi".

Compilazione orientata alla sorgente

In questo caso la compilazione viene innanzitutto effettuata in un programma utente eseguibile dopo che sono state specificate tutte le istruzioni. Durante la compilazione viene verificata la presenza di eventuali errori di immissione.

Compilazione

Generazione di un programma utente eseguibile partendo da una sorgente.

Componente di dichiarazione

La dichiarazione delle variabili di un blocco si suddivide in diverse componenti per la dichiarazione dei parametri del blocco. La componente IN contiene ad es. la dichiarazione dei parametri di ingresso, la componente OUT contiene la dichiarazione dei parametri di uscita.

Contatori

I contatori sono componenti della memoria di sistema della CPU. Il contenuto di questi contatori viene aggiornato in modo asincrono con il programma utente del sistema operativo. Con istruzioni STEP 7 viene definita la funzione dettagliata della cella di conteggio (p. es. contatore in avanti) e viene avviata la sua elaborazione (Start).

Controllo

Quando si effettua il controllo di un programma se ne verifica l'esecuzione nella CPU. Ad esempio i nomi e i valori attuali delle variabili e dei parametri vengono visualizzati e aggiornati ciclicamente in ordine cronologico.

Conversione dei tipi di dati

La conversione del tipo di dati è necessaria per poter combinare due operandi con tipo di dati diverso.

Costante (simbolica)

Le costanti con nomi simbolici sono segnaposti per valori costanti di blocchi di codice e vengono utilizzate per facilitare la lettura di un programma.

Costante

Le costanti sono segnaposti per valori costanti di blocchi di codice. Le costanti vengono utilizzate per facilitare la lettura di un programma. Esempio: Invece di inserire direttamente un valore (ad es. 10) viene specificato il segnaposto "max_esecuzioneiloop". Al richiamo viene sostituito con il valore della costante (ad es. 10).

Dati globali

I dati globali sono dati ai quali si può far riferimento da qualsiasi blocco di codice (FC, FB, OB). Si tratta in particolare di merker M, ingressi E, uscite A, temporizzatori, contatori ed elementi di blocchi dati DB. Ai dati globali si può accedere sia in modo assoluto che simbolico.

Dati locali

I dati locali sono i dati assegnati ad un blocco di codice che vengono dichiarati nella relativa parte dichiarazioni o dichiarazione delle variabili. Essi comprendono (a seconda del tipo di blocco): parametri formali, dati statici, dati temporanei.

Dati statici

I dati statici sono dati locali di un blocco funzionale che vengono memorizzati nel blocco dati di istanza e rimangono quindi invariati fino alla successiva elaborazione del blocco funzionale.

Dati temporanei

I dati temporanei sono dati locali di un blocco che vengono memorizzati nello stack L durante l'elaborazione del blocco e che al termine dell'elaborazione non sono più disponibili.

Dati utili

I dati utili vengono scambiati fra un'unità centrale e un'unità di segnale, fra un'unità funzionale e unità di comunicazione tramite l'immagine di processo o accessi diretti. I dati utili possono essere: segnali di ingresso/uscita digitali e analogici, informazioni di comando e di stato di unità funzionali.

Debugger SCL

Il Debugger SCL è un debugger per linguaggi avanzati con il quale si possono localizzare errori logici di programmazione nei programmi utente creati in SCL.

Dichiarazione di variabili

La dichiarazione di variabili comprende l'indicazione di un nome simbolico, di un tipo di dati, eventualmente di un valore di preassegnazione, indirizzo e commento.

Dichiarazione

Meccanismo per la definizione di un elemento del linguaggio. Una dichiarazione comprende il collegamento di un identificatore con l'elemento del linguaggio e l'assegnazione di attributi e tipi di dati.

Editor SCL

L'Editor SCL è un editor adattato a SCL con il quale si può creare una sorgente SCL.

Enable (EN)

In STEP 7 il blocco funzionale e le funzioni hanno il parametro di ingresso definito implicitamente "Enable" (EN) che può essere impostato al richiamo del blocco. Se EN è uguale a TRUE, il blocco richiamato viene eseguito, in caso contrario non viene eseguito.

Enable out (ENO)

In STEP 7 ogni blocco possiede un'uscita "Enable Output" (ENO). Alla fine dell'esecuzione di un blocco, il valore attuale del flag OK viene scritto in ENO. Subito dopo il richiamo di un blocco, in base al valore di ENO, si può verificare se tutte le operazioni del blocco sono state eseguite correttamente o se si sono verificati degli errori.

Espressione

In SCL un'espressione serve per l'elaborazione dei dati. Si distingue fra espressioni aritmetiche, logiche e di confronto.

Flag OK

Il flag OK viene utilizzato per rilevare l'esecuzione corretta o errata di una sequenza di comandi di blocco. Si tratta di una variabile globale del tipo BOOL.

Funzione (FC)

Come stabilito dalla norma IEC 1131-3 una funzione (FC) è un blocco di codice senza dati statici. Una funzione offre la possibilità di trasferire parametri nel programma utente. In tal modo, le funzioni sono particolarmente adatte per parametrizzare funzioni complesse di frequente ripetizione, come p. es. calcoli.

Funzione di sistema (SFC)

Una funzione di sistema (SFC) è una funzione integrata nel sistema operativo della CPU, che in caso di necessità può essere richiamata nel programma utente STEP 7.

Gerarchia di richiamo

Prima che possano essere elaborati, tutti i blocchi devono dapprima essere richiamati. La sequenza e l'annidamento di questi richiami viene denominata gerarchia di richiamo.

Guida online

STEP 7 consente all'utente di visualizzare una guida al contesto mentre utilizza il software di programmazione.

Identificatore

Combinazione di lettere, numeri e caratteri di sottolineatura che identifica un elemento del linguaggio.

Identificazione di operando

Un'identificazione di operando è una parte dell'operando di un'operazione contenente informazioni, come p. es. l'area di memoria in cui l'operazione trova un valore (oggetto dati) con cui esegue una combinazione oppure trova la grandezza di un valore (oggetto dati) con il quale esegue una combinazione. Nell'istruzione "Valore := EB10" "EB" è l'identificazione di operando ("E" indica l'area ingresso della memoria, "B" indica un byte in quest'area).

Immagine di processo degli ingressi (IPI)

Prima dell'elaborazione del programma utente, il sistema operativo legge l'immagine di processo degli ingressi dalle unità d'ingresso.

Immagine di processo delle uscite (IPU)

Alla fine del programma utente, l'immagine di processo delle uscite viene trasferita dal sistema operativo alle unità di uscita.

Immagine di processo

Nella CPU, gli stati dei segnali delle unità di ingresso e uscita digitali vengono trasferiti in un'immagine di processo. Si distingue fra l'immagine di processo degli ingressi (IPI) e quelle delle uscite (IPU).

Indirizzamento assoluto

Nell'indirizzamento assoluto viene indicato l'indirizzo dell'operando da elaborare. Esempio: l'indirizzo A 4.0 indica il bit 0 nel byte 4 dell'immagine di processo delle uscite.

Indirizzamento simbolico

Nell'indirizzamento simbolico l'operando da elaborare viene indicato in modo simbolico. L'assegnazione di simboli e indirizzi viene effettuata nella tabella dei simboli o in un file simbolico.

Indirizzamento

Assegnazione di un indirizzo nel programma utente. Gli indirizzi possono essere assegnati a determinati operandi o campi di operandi (esempi: ingresso E 12.1, parola di merker MW25).

Integer (INT)

Integer (INT) è uno dei tipi di dati semplici. La rappresentazione avviene con un numero intero a 16 bit.

Interfaccia di richiamo

L'interfaccia di richiamo viene definita tramite i parametri d'ingresso, d'uscita e di ingresso/uscita (parametri formali) di un blocco nel programma utente. Al richiamo del blocco, questi parametri vengono sostituiti dai parametri attuali.

Istanza

Con il termine "Istanza" si indica il richiamo di un blocco funzionale. Ad esso viene assegnato un blocco dati di istanza oppure un'istanza locale. Se un blocco funzionale nel programma utente STEP 7 viene richiamato n volte con diversi parametri e nomi di blocchi dati di istanza, esistono n istanze.

Istruzione CASE

Istruzione di diramazione utilizzata per selezionare una parte di programma da 1 a n in funzione del valore di un'espressione di selezione.

Istruzione CONTINUE

In SCL l'istruzione CONTINUE serve per interrompere l'esecuzione momentanea di un loop di un'istruzione di ripetizione (FOR, WHILE o REPEAT).

Istruzione EXIT

Costruzione del linguaggio interna al programma che consente di interrompere un loop in un punto qualsiasi indipendentemente dalle condizioni.

Istruzione FOR

Costruzione del linguaggio interna al programma. L'istruzione FOR esegue una sequenza di istruzioni in un loop nel quale vengono assegnati ad una variabile di esecuzione dei valori in ordine progressivo.

Istruzione GOTO

Costruzione del linguaggio interna al programma. Un'istruzione GOTO causa il salto immediato ad una determinata etichetta di salto e quindi ad un'istruzione all'interno dello stesso blocco.

Istruzione REPEAT

Costruzione del linguaggio interna al programma che consente di ripetere una sequenza di istruzioni fino alla condizione di interruzione.

Istruzione RETURN

Costruzione del linguaggio interna al programma che consente di uscire dal blocco attuale.

Istruzione

Un'istruzione è la più piccola unità indipendente di un programma utente creato in un linguaggio testuale. Essa rappresenta una norma di lavoro per il processore.

Letterale

Modalità di scrittura formale che definisce il valore e il tipo di una costante.

Memoria di sistema (area di sistema)

La memoria di sistema è integrata nella CPU S7 ed ha una struttura di memoria RAM. Nella memoria di sistema vengono depositate aree di operandi (p. es. temporizzatori, contatori, merker) e aree di dati usati dal sistema operativo per operazioni interne (p. es. buffer di comunicazione).

Merker (M)

Area della memoria di sistema di una CPU S7 SIMATIC. A quest'area si può accedere in lettura e scrittura (a bit, a byte, a parola e a doppia parola). L'area merker può essere utilizzata dall'utente per memorizzare risultati intermedi.

Mnemonico

Il mnemonico è una rappresentazione abbreviata degli operandi e delle operazioni di programmazione in un programma. STEP 7 supporta la rappresentazione inglese (nella quale ad es. "I" sta per ingresso) e tedesca (nella quale ad es. ingresso è indicato da "E").

Multiistanza

Quando si utilizza la multiistanza, il blocco dati di istanza contiene i dati di molti blocchi funzionali di una gerarchia di richiamo.

Non-terminale

Un non-terminale è un elemento composto di una descrizione sintattica che viene descritto da un'altra regola lessicale o sintattica.

Numero in virgola mobile

Un numero in virgola mobile è un numero positivo o negativo che rappresenta un valore decimale, ad es. 0,339 o -11,1.

Offline

Offline indica lo stato di funzionamento in cui il dispositivo di programmazione non ha alcun collegamento con il sistema di automazione (fisico, logico).

Online

Online indica lo stato di funzionamento in cui il dispositivo di programmazione è collegato con il sistema di automazione (fisico, logico).

Operando

Un operando è una parte di un'istruzione e indica con che cosa deve operare il processore. Esso può essere indirizzato sia in modo assoluto che in modo simbolico.

Operazione

Un'operazione fa parte di un'istruzione e indica che cosa deve fare il processore.

Parametri di ingresso/uscita

I parametri di ingresso/uscita sono presenti nelle funzioni e nei blocchi funzionali. Tramite parametri di ingresso/uscita i dati vengono trasferiti al blocco richiamato, elaborati e il blocco richiamato deposita nuovamente i risultati nella stessa variabile.

Parametri d'ingresso

I parametri di ingresso sono presenti solo nelle funzioni e nei blocchi funzionali. Con l'ausilio dei parametri d'ingresso, i dati vengono trasferiti al blocco richiamante per un'ulteriore elaborazione.

Parametri d'uscita (parametri A)

Tramite i parametri d'uscita di un blocco nel programma utente STEP 7 i risultati vengono trasferiti al blocco richiamante.

Parametri formali

Un parametro formale è un segnaposto per il parametro "effettivo" (parametro attuale) nei blocchi di codice parametrizzabili. Negli FB e nelle FC, i parametri formali vengono dichiarati dall'utente, negli SFB e nelle SFC essi sono già presenti. Durante il richiamo del blocco, al parametro formale viene assegnato un parametro attuale, dopodiché il blocco richiamato lavora con questo valore aggiornato. I parametri formali fanno parte dei dati locali del blocco e vengono suddivisi in parametri d'ingresso, d'uscita e di ingresso/uscita.

Parametro attuale

In un richiamo di un blocco funzionale (FB) o di una funzione (FC), i parametri attuali sostituiscono i parametri formali.

Esempio: il parametro formale "Start" viene sostituito con il parametro attuale "E 3.6"

Parola chiave

Unità lessicale che caratterizza un elemento del linguaggio, ad es. "IF".

In SCL le parole chiave vengono utilizzate per contrassegnare l'inizio di un blocco, per marcare sezioni nella parte di dichiarazione risp. nella parte convenzioni e per contrassegnare delle istruzioni. Esse vengono inoltre utilizzate per commenti e attributi.

Parola di stato

La parola di stato fa parte integrante dei registri dell'unità centrale. La parola di stato contiene informazioni di stato e informazioni di errore che possono verificarsi nel corso dell'elaborazione di comandi STEP 7. I bit di stato possono essere letti e scritti dall'utente; i bit di errore possono solo essere letti.

Parte convenzioni

Nella parte convenzioni vengono dichiarati i dati locali di un blocco di codice nel caso in cui il programma venga creato con un editor di testi.

Passo singolo

Il passo singolo è un passo di test nell'ambito della funzione passo singolo del Debugger SCL. Nella funzione passo singolo, si può eseguire il programma istruzione per istruzione e visualizzare i risultati nella finestra dei risultati.

Progetto

Un contenitore per tutti gli oggetti di una soluzione di automazione indipendentemente dal numero di stazioni, unità e dal loro impiego.

Programma utente S7

Cartella per i blocchi che vengono caricati in un'unità programmabile S7 (ad es. CPU, FM) e vi possono essere eseguiti per comandare un impianto o un processo.

Programma utente

Il programma utente contiene tutte le istruzioni e le dichiarazioni per l'elaborazione dei segnali, tramite i quali viene controllato l'impianto o un processo. Esso viene assegnato ad un'unità modulare programmabile (p. es. CPU, FM) e può essere strutturato in unità più piccole (blocchi).

Programmazione simbolica

Il linguaggio di programmazione SCL consente l'impiego di sequenze simboliche di caratteri al posto di operandi: p. es. l'operando A1.1 può essere sostituito da "Valvola_17". La tabella dei simboli assicura il collegamento fra l'operando e la sequenza simbolica di caratteri assegnati.

Programmazione strutturata

Per risolvere problemi di automazione piuttosto complessi, il programma utente viene suddiviso in piccole parti di programma completamente autonome (blocchi). La suddivisione del programma utente avviene in modo funzionale oppure conformemente alla struttura tecnologica dell'impianto.

Protezione dei blocchi

Come protezione di blocco viene definita la possibilità di proteggere blocchi singoli dalla decompilazione, se la compilazione della sorgente del blocco è stata eseguita con la parola chiave "KNOW_HOW_PROTECTED".

Punto d'arresto

Con questa funzione, in determinati punti del programma si può portare l'unità centrale (CPU) nello stato di funzionamento ALT. Quando viene raggiunto un punto d'arresto, si possono eseguire le funzioni di test come p. es. elaborazione passo passo dei comandi oppure comando/controllo di variabili.

Regola sintattica

Il livello di regole superiore relativo alla descrizione formale del linguaggio SCL si compone di regole sintattiche. Per il loro impiego esiste libertà di formato, ovvero p. es. si possono inserire spazi e caratteri di controllo.

Regole lessicali

Il livello di regole inferiore relativo alla descrizione formale del linguaggio SCL si compone di regole lessicali. Per il loro uso non esiste libertà di formato, ovvero l'integrazione con spazi e caratteri di controllo non è consentita.

Richiamo di blocco

Avvio di un blocco nel programma utente STEP 7: in linea di massima, i blocchi organizzativi vengono richiamati dal sistema operativo, tutti gli altri blocchi vengono richiamati dal programma utente STEP 7.

RUN

Nello stato di funzionamento RUN viene elaborato il programma utente e l'immagine di processo viene aggiornata ciclicamente. Tutte le uscite digitali sono abilitate.

RUN-P

Lo stato di funzionamento RUN-P corrisponde allo stato di funzionamento RUN, con la differenza che nello stato di funzionamento RUN-P sono consentite tutte le funzioni del dispositivo di programmazione senza alcuna restrizione.

SCL

Linguaggio di programmazione avanzato, simile a PASCAL secondo la norma DIN EN-61131-3 (int. IEC 1131-3) per la programmazione di compiti complessi in un PLC, p. es. algoritmi, compiti di elaborazione dati. Abbreviazione di "Structured Control Language".

Segnalibro

I segnalibro sono etichette di testo temporanee che contrassegnano una posizione specifica all'interno di una sorgente. Semplificano la navigazione nella sorgente.

Semantica

Relazione tra gli elementi simbolici di un linguaggio di programmazione e il loro significato, lettura e utilizzo.

Simbolo

Un simbolo è un nome definito dall'utente conformemente a determinate regole sintattiche. Dopo la definizione indicante il suo tipo (p. es. variabile, tipo di dati, blocco), questo nome può essere utilizzato in fase di programmazione e nelle operazioni di comando e osservazione. Esempio: operando: E5.0, tipo di dati: Bool, simbolo: Tasto_Arresto di emergenza.

Sorgente SCL

La sorgente SCL è il file in cui viene creato il programma in SCL. La sorgente viene quindi compilata con il compilatore SCL.

Sorgente

Parte di un programma creata con un editor grafico o testuale che viene generata mediante compilazione di un programma utente eseguibile.

Struttura (STRUCT)

Tipo di dati composto costituito da elementi di dati di tipo diverso. I tipi di dati delle strutture possono essere elementari o superiori.

Tabella dei simboli

Tabella per l'assegnazione di simboli (=nome) a indirizzi di dati globali e blocchi. Esempi: Arresto di emergenza (simbolo) - E 1.7 (indirizzo) o Regolatore (simbolo) - SFB 24 (blocco).

Tabella delle variabili

Nella tabella delle variabili vengono riunite le variabili che devono essere osservate e comandate, comprese le corrispondenti indicazioni sui formati.

Tempo di ciclo

Il tempo di ciclo è il tempo di cui ha bisogno la CPU per l'elaborazione del programma utente.

Tempo di controllo del ciclo

Se il tempo di elaborazione del programma utente supera il tempo di controllo del ciclo impostato in precedenza, il sistema operativo genera un messaggio di errore e la CPU si porta in STOP.

Temporizzatori

I temporizzatori sono componenti della memoria di sistema della CPU. Il contenuto di questi temporizzatori viene aggiornato in modo asincrono rispetto al programma utente dal sistema operativo. Con istruzioni STEP 7 viene definita la funzione dettagliata della cella di tempo (p. es. ritardo all'inserzione) e viene avviata la sua elaborazione (Start).

Terminale

Un terminale è un elemento base di una regola lessicale o sintattica, il quale non viene dichiarato con un'ulteriore regola, ma solo verbalmente. Un terminale può essere p. es. una parola chiave o solo un singolo carattere.

Tipi di dati composti

Tipi di dati complessi costituiti da elementi con tipo di dati elementare. Si distingue fra strutture e campi. Anche i tipi di dati STRING e DATE_AND_TIME sono tipi di dati composti.

Tipi di dati definiti dall'utente

I tipi di dati definiti dall'utente (UDT) vengono creati dall'utente con la dichiarazione dei tipi di dati. Possiedono un nome proprio e possono essere utilizzati più volte. In tal modo, un tipo di dati definito dall'utente può essere utilizzato per generare diversi blocchi dati con identica struttura (p. es. regolatore).

Tipi di dati semplici

I tipi di dati semplici sono tipi di dati predefiniti in conformità alla norma IEC 1131-3. Esempi: il tipo di dati "BOOL" definisce una variabile binaria ("bit"); il tipo di dati "INT" definisce una variabile in virgola fissa a 16 bit.

Tipo di blocco

L'architettura dei blocchi di STEP 7 conosce i seguenti tipi di blocchi: blocchi organizzativi, blocchi, funzionali, funzioni, blocchi dati e blocchi funzionali di sistema, funzioni di sistema, blocchi dati di sistema e tipi di dati definiti dall'utente.

Tipo di dati

I tipi di dati definiscono:

- il tipo e il significato degli elementi di dati
- i campi di memoria e di valori ammessi per gli elementi di dati
- il numero di operazioni eseguibili con un operando di un tipo di dati
- la modalità di scrittura degli elementi di dati.

Tipo di parametro

Un tipo di parametro è un tipo di dati speciale per temporizzatori, contatori e blocchi. Esso può essere utilizzato nei parametri d'ingresso di blocchi funzionali e funzioni, mentre i parametri di ingresso uscita possono essere usati solo da blocchi funzionali, per trasferire temporizzatori, contatori e blocchi al blocco richiamato.

UDT

Vedere: Tipi di dati definiti dall'utente

Valore di ritorno (RET_VAL)

Contrariamente ai blocchi funzionali, le funzioni forniscono quale risultato un valore di ritorno.

Valore iniziale

Valore assegnato ad una variabile all'avvio del sistema.

Variabile

Una variabile definisce dei dati con contenuto variabile, i quali possono essere utilizzati nel programma utente STEP 7. Una variabile si compone di un operando (p. es. M 3.1) e di un tipo di dati (p. es. Bool) e può essere contrassegnata con un simbolo (p. es. NASTRO_ON). La variabile viene dichiarata nella parte convenzioni.

Indice analitico

(
(S_ODT) 13-15

*
* 11-8, 11-9
** 11-8

/
/ 11-8

+
+ 11-8, 11-9

<
< 11-12
<= 11-12
<> 11-11, 11-12

=
= 11-11

>
> 11-12
>= 11-11

A
ABS 14-9
Accessi a campi di variabili 8-5
Accesso assoluto ai blocchi dati 10-8
Accesso assoluto alle aree di memoria della CPU 10-5
Accesso indicizzato ai blocchi dati 10-10
Accesso indicizzato alle aree di memoria della CPU 10-4
Accesso simbolico alle aree di memoria della CPU 10-3
Accesso strutturato ai blocchi dati 10-11
ACOS 14-10
Adattamento della superficie operativa 4-3
Allineamento automatico delle righe 4-14
Allineamento delle righe 4-14
Ambiente di richiamo 4-32
Ambiente di sviluppo 1-4
AND 11-2, 11-10

Annullamento dell'ultima azione di editazione 4-10
ANY 7-18, 7-19
Apertura di blocchi 4-6
Apertura di una sorgente S7-SCL 4-5
Area di lavoro 4-2
Area di memorizzazione delle variabili 8-5
Aree di memoria della CPU 5-9, 10-1 - 10-5
ARRAY 7-9, 7-10, 8-3, 8-4, 12-6
 assegnazione di valori con variabili di tipo ARRAY 12-5
ASIN 14-10
Assegnazione di ingresso/uscita 12-32, 12-40
 assegnazione di ingresso/uscita (FB/SFB) 12-32
 assegnazione di ingresso/uscita (FC) 12-40
Assegnazione di parametri nelle funzioni di conteggio 13-3
Assegnazione di parametri nelle funzioni temporali 13-10
Assegnazione di strutture con lunghezza di byte dispari 16-4
Assegnazione di valori 6-11, 12-2 - 12-10
 assegnazione di valore con variabili di tipo STRUCT e UDT 12-3
 assegnazione di valori con variabili assolute per aree di memoria 12-9
 assegnazione di valori con variabili di tipo DATE_AND_TIME 12-8
 assegnazione di valori con variabili di tipo di dati semplici 12-2
 assegnazione di valori con variabili di tipo STRING 12-7
 assegnazione di valori con variabili globali 12-10
 regole sintattiche 15-42
Assegnazione parametri 12-26
AT 8-5
ATAN 14-10
Attributi 6-5, 6-6, 6-7, 6-9
Attributi di blocco 6-5, 6-7
 attributi di sistema per i blocchi 6-7
 definizione 6-5
 regole lessicali 15-28
Attributi di sistema 6-7, 6-9
 per blocchi 6-7
Attributi di sistema per i parametri 6-9
AUTHORS.EXE 2-5

Automation License Manager 2-1
autorizzazione 2-5
autorizzazione all'utilizzo 2-5
autorizzazione provvisoria 2-5

B

Barra degli strumenti 4-2
Barra dei menu 4-2
Barra del titolo 4-2
Barra di stato 4-2
BIT 7-3
Blocchi 3-4, 3-5, 4-6, 6-1
Blocchi dati 6-17, 10-7 - 10-11
Blocchi di codice 4-8, 6-1
Blocchi di programma 3-4
Blocchi funzionali (FB) 6-12, 12-26
Blocchi nei diagrammi sintattici 5-1
Blocchi organizzativi 6-16
Blocco di commenti 5-15
Blocco funzionale (FB) 12-28, 12-30
BLOCK_DB_TO_WORD 14-4
Buffer di diagnostica 4-41
BYTE 7-3
BYTE_TO_BOOL 14-3
BYTE_TO_CHAR 14-3

C

Campi 7-9
Campo (ARRAY)
 assegnazione di valori con variabili
 di tipo ARRAY 12-5
 inizializzazione 8-3
Campo di applicazione 1-1
Cancellazione di oggetti di testo 4-12
Cancellazione totale della memoria CPU 4-26
Caratteri non stampabili 9-10, 9-11
Caratteri stampabili 9-9, 9-11
Caratteristiche dell'orologio 4-42
Caricamento 4-26, 4-27
Certificate of License 2-1, 2-2
CHAR 7-3
CHAR_TO_BYTE 14-3
CHAR_TO_INT 14-3
chiave di licenza 2-1, 2-2, 2-4
Chiusura di una sorgente S7-SCL 4-5
Cifre 5-11, 5-12
Colore e tipo di carattere del testo sorgente
 4-15, 4-25
Commenti
 regole lessicali 15-27
Commento
 blocco di commenti 5-15
 commento di una riga 5-16
 inserimento di modelli di commento 4-17
Commento di una riga 5-16
Compilatore 4-20
 impostazione del compilatore 4-20

Compilazione 4-19 - 4-22, 6-22, 6-23
compiler 1-4
Comunicazione con la CPU 4-42
CONCAT 14-13
Concessione della licenza d'utilizzo mediante
 Automation License Manager 2-1
Condizione di interruzione 12-21, 12-23
Condizioni 12-13
conforme alla norma 1-1
Contatori 13-1, 13-3, 13-4, 13-5, 13-6
 assegnazione di parametri nelle funzioni
 di conteggio 13-3
 Conteggio all'indietro (S_CD) 13-5
 Conteggio in avanti (S_CU) 13-5
 Conteggio in avanti e all'indietro (S_CUD)
 13-6
 esempio di funzioni di conteggio 13-6
 Introduzione ed analisi del valore del
 contatore 13-4
 richiamo di funzioni di conteggio 13-1
Conteggio all'indietro (S_CD) 13-5
Conteggio in avanti (S_CU) 13-5
Conteggio in avanti e all'indietro
 (S_CUD) 13-6
Controlla 4-29, 4-31, 4-32
Controllo continuo 4-29
Copia di oggetti di testo 4-11
COS 14-10
Costante Char 9-9
Costante di data 9-13
Costante di intervallo 9-13
Costante di numero intero 9-7
Costante di numero reale 9-8
Costante ora del giorno 9-16
Costanti 9-2 - 9-17
Costanti a bit 9-6
Costanti predefinite e flag
 Descrizione formale del linguaggio 15-17
Costanti simboliche 9-2
COUNTER 7-15, 13-1
Creazione di un file di controllo della
 compilazione 4-22
Creazione e visualizzazione dei dati
 di riferimento 4-37

D

DATE 7-4
DATE_AND_TIME 7-5
DATE_TO_DINT 14-3
Dati di riferimento 4-37
Dati globali 10-1
 dati globali 10-2
Dati locali 5-17, 8-1 - 8-10
Dati utente 10-1
 globali 10-1
debugger 1-4
Definizione delle proprietà di oggetti 4-6

Definizione di un ambiente di richiamo del blocco 4-32
 Definizione di un ambiente di richiamo per punti d'arresto 4-35
 DELETE 14-15
 Descrizione del linguaggio 5-1, 15-1
 Descrizione formale del linguaggio 15-1
 DI_STRNG 14-19
 Diagramma di flusso di ANALISI 3-13
 Diagramma di flusso di RILEVAZIONE 3-17
 Diagrammi sintattici 5-1, 15-1
 Dichiarazione 6-8
 Dichiarazione di istanze 8-6
 Dichiarazione di variabili statiche 8-2
 DINT 7-3
 DINT_TO_DATE 14-3
 DINT_TO_DWORD 14-3
 DINT_TO_INT 14-3, 14-4
 DINT_TO_TIME 14-3
 DINT_TO_TOD 14-3
 Diramazione del programma 12-12
 dischetto di autorizzazione 2-5
 Dischetto di autorizzazione 2-3
 Disuguaglianza 11-2
 DIV 11-8
 Divisione 11-2
 Divisione per numero intero 11-2
 Doppia parola 7-3
 Download 4-26
 DWORD 7-3
 DWORD_TO_BOOL 14-4
 DWORD_TO_BYTE 14-4
 DWORD_TO_DINT 14-4
 DWORD_TO_REAL 1) 14-3
 DWORD_TO_WORD 14-4

E

editor 1-4
 Elaborazione di loop 12-12
 Elaborazione di un sottoprogramma 6-11
 Elaborazione di una sorgente S7-SCL 4-10 - 4-18
 Eliminazione di errori dopo la compilazione 4-22
 EN 12-42
 ENO 12-43
 EQ_STRNG 14-17
 Errori
 Eliminazione dopo la compilazione 4-22
 Esempi 7-19, 12-33 - 12-41, 13-6, 13-18, 14-7 - 14-12
 Esempio 3-1
 Esempio introduttivo "Rilevazione di un valore di misura" 3-1
 Espressione booleana 11-11
 Espressione semplice 11-7
 Espressioni 11-2 - 11-12
 Espressioni aritmetiche 11-9

Espressioni di confronto 11-12
 Etichetta di testo 4-16
 Etichette (etichette di salto) 9-17
 Etichette di salto 9-17
 EXP 14-9
 EXPD 14-9

F

FC 6-14, 12-26, 12-36
 File di compilazione 6-22
 File di controllo della compilazione 4-22
 FIND 14-16
 Fine di un blocco 6-3
 Flag (flag OK) 8-7
 Flag OK 8-1
 FOR-Anweisung 16-5
 Formato di pagina 4-23
 Formattazione del testo sorgente in base alla sintassi 4-15
 Funzione (FC) 12-36
 Funzione modulo 11-2
 Funzione standard su stringhe di bit 14-11
 Funzioni
 blocchi funzionali di sistema e biblioteca standard 14-27
 Funzioni (FC) 6-14, 12-26
 Funzioni di arrotondamento e taglio 14-6
 Funzioni di conversione 14-3
 classe B 14-3
 Funzioni di conversione del tipo di dati 14-3, 14-6
 Funzioni di conversione di tipi di dati 14-3
 Funzioni di selezione dei valori 14-23
 Funzioni di test di S7-SCL 4-28, 4-30
 Funzioni di test di STEP 7 4-37, 4-38
 Funzioni di test S7-SCL 4-29, 4-32
 Funzioni logaritmiche 14-9
 Funzioni matematiche standard 14-9, 14-10
 Funzioni numeriche standard 14-9, 14-10
 Funzioni standard 14-3 - 14-11
 Funzioni trigonometriche 14-10

G

GE_STRNG 14-17
 Generazione di sorgenti S7-SCL con un editor standard 4-6
 Generazione di una nuova sorgente S7-SCL 4-4
 Grenzwerte für FOR-Anweisungen 16-5
 GT_STRNG 14-18

I

I_STRNG 14-19
 Identificatori 5-6, 15-14, 15-15
 definizione 5-6
 descrizione formale del linguaggio 15-13, 15-14
 esempi 5-6
 regole 5-6
 Regole lessicali 15-18
 Identificatori di blocchi 5-7
 Identificatori di operandi 5-9
 Identificatori standard 5-7
 Immagine di processo degli ingressi e delle uscite 10-2
 Impostazione 4-20
 Impostazione del formato di pagina 4-23
 Impostazione della data 4-40
 Impostazione dell'ora 4-40
 Impostazioni 4-3
 Indirizzamento assoluto
 Regole lessicali 15-25
 Indirizzi 10-2
 Ingressi e uscite di periferia 10-2
 Inizializzazione 8-3
 Inizio di un blocco 6-3
 Inserimento di modelli di commento 4-17
 Inserimento di modelli per i i blocchi 4-17
 Inserimento di modelli per i parametri 4-18
 Inserimento di richiami di blocco 4-17
 Inserimento di strutture di controllo 4-18
 INSERT 14-15
 Installazione 2-5
 Installazione dell'Automation License Manager 2-3
 Installazione dell'autorizzazione 2-3
 Istanza globale 12-28
 Istanza locale 12-28
 INT 7-3
 INT_TO_CHAR 14-4
 INT_TO_WORD 14-4
 Istanza globale 12-33
 Istanza locale 12-35
 Istruzione CASE 12-12, 12-15
 Istruzione CONTINUE 12-12, 12-22
 Istruzione di selezione 12-12
 Istruzione EXIT 12-12, 12-23
 Istruzione FOR 12-12, 12-17
 Istruzione GOTO 12-24
 Istruzione IF 12-12, 12-14
 Istruzione REPEAT 12-12, 12-21
 Istruzione RETURN 12-12, 12-25
 Istruzione WHILE 12-12, 12-20
 Istruzioni 12-1 - 12-25
 Istruzioni di controllo 4-18, 6-11, 12-14
 inserimento di istruzioni di controllo 4-18
 istruzione CASE 12-15
 istruzione CONTINUE 12-22

istruzione EXIT 12-23
 istruzione FOR 12-17
 istruzione GOTO 12-24
 istruzione IF 12-14
 istruzione REPEAT 12-21
 istruzione WHILE 12-20
 istruzioni 6-11
 regole sintattiche 15-47
 Istruzioni di salto 12-12

L

LE_STRNG 14-17
 LEFT 14-14
 LEN 14-13
 Letterale 9-13, 9-16
 Letterali 9-6 - 9-17
 vedere Costanti 15-22
 Lettura dei dati della CPU 4-41
 Lettura dei valori d'uscita
 assegnazione d'uscita nel richiamo di FC 12-40
 Lettura dei valori iniziali 12-33
 assegnazione d'uscita nel richiamo degli FB 12-33
 Lettura del buffer di diagnostica della CPU 4-41
 Libertà di formato 5-2, 5-3
 License Manager 2-1, 2-2
 Licenza 2-1, 2-2
 linguaggio di programmazione avanzato 1-1, 1-3
 LN 14-9
 LOG 14-9
 LT_STRNG 14-18

M

Memoria utente 4-41
 Meno unario 11-2
 Merker 10-2
 Messaggi di avviso 4-22
 MID 14-14
 MOD 11-8, 11-9
 modalità operativa di S7-SCL 1-4
 Modelli 4-17
 di commento 4-17
 di strutture di controllo 4-18
 Modelli per i i blocchi 4-17
 Modelli per i parametri 4-18
 Modelli per parametri 4-18
 Modifica di una sorgente S7-SCL 4-17, 4-18
 Moltiplicazione 11-2
 Multiistanze 8-6

N

NE_STRNG 14-17
 Negazione 11-2
 Nome del blocco 6-3, 6-4
 Nomi 5-6
 definizione 5-6
 descrizione formale del linguaggio 15-13, 15-14
 esempi 5-6
 regole 5-6
 Non-terminali (nei diagrammi sintattici) 15-12
 norma DIN EN-61131-3 1-1
 NOT 11-10
 Numeri delle righe 4-25
 Numeri di riga 4-3
 Nuove funzioni della versione V5.3 SP1 1-6

O

OB 6-16
 Operandi 11-3, 11-4
 Operazioni 15-7
 elenco alfabetico 15-5
 Operazioni su una sorgente S7-SCL 4-5, 4-6, 4-23, 4-24
 Opzioni di compilazione 6-22, 15-29
 Or 11-2
 OR 11-10
 Or esclusivo 11-2
 Ordine dei blocchi 4-8

P

Parametri 6-9, 7-15, 7-16, 8-1- 8-12, 12-30 - 12-40
 Parametri attuali 8-1
 assegnazione d'ingresso 12-39
 Definizione 8-1
 Parametri di blocchi 8-11
 Parametri di blocco 5-17
 Parametri di FC 12-40
 Parametri di ingresso 8-1, 12-31, 12-39, 12-42
 assegnazione d'ingresso (FB) 12-31
 assegnazione d'ingresso (FC) 12-39
 definizione 8-1
 parametro di ingresso EN 12-42
 Parametri di ingresso/uscita 8-1
 Parametri di uscita 8-1
 Parametri FB 12-30 - 12-33
 Parametri FC 12-38, 12-39
 Parametri formali 8-1
 Parametro d'uscita ENO 12-43
 Parentesi 11-2
 Parole chiave 5-5, 15-8
 Parole riservate 5-5

Parte convenzioni 6-8, 8-3, 8-8 - 8-11
 definizione 6-8
 inizializzazione 8-3
 parametri di blocchi 8-12
 regole sintattiche 15-33
 sommario dei blocchi convenzioni 8-8
 struttura 6-8
 variabili statiche 8-9
 Variabili temporanee 8-10

Parte istruzioni
 regole sintattiche 15-40
 struttura 6-10
 Più unario 11-2
 POINTER 7-16, 7-17
 Posizionamento dei segnalibro nel testo sorgente 4-16
 Posizionamento del cursore su una determinata riga 4-13
 Potenza 11-2
 Progettazione di programmi SCL 3-1
 Progettazione di un programma 3-1
 Programma di autorizzazione 2-3
 Programma utente 3-4, 6-1
 programmazione
 strutturata 1-2
 Programmazione simbolica 4-9
 Programmazione strutturata 3-4, 3-6
 Protezione dei blocchi 4-7
 Puntatore nullo 7-18
 Punti d'arresto 4-36
 Punti di arresto 4-33, 4-34

R

R_STRNG 14-20
 REAL 7-3
 REAL_TO_DINT 14-4
 REAL_TO_DWORD 2) 14-3
 REAL_TO_INT 14-4
 Regole
 per l'utilizzo delle chiavi di licenza 2-4
 Regole lessicali 15-18
 Regole per le sorgenti S7-SCL 4-8
 Regole per l'utilizzo delle chiavi di licenza 2-4
 Regole sintattiche 15-30
 REPLACE 14-16
 Requisiti del installazione 2-5
 Ricerca di oggetti di testo 4-10
 Richiami di blocco 4-17
 Richiamo di blocchi funzionali (FB o SFB) 12-28
 assegnazione di ingresso/uscita 12-32
 assegnazione di parametri FB 12-30
 assegnazione d'ingresso 12-31
 lettura dei valori iniziali 12-33
 procedura 12-28
 richiamo come istanza globale 12-28
 richiamo come istanza locale 12-28
 sintassi 12-28

Richiamo di blocco 4-17
 Richiamo di funzioni (FC) 12-36
 assegnazione di parametri 12-38
 assegnazione di uscita/ingresso 12-40
 assegnazione d'ingresso 12-39
 parametro di ingresso EN 12-42
 parametro d'uscita ENO 12-43
 procedura 12-36
 sintassi 12-36
 Valore di ritorno 12-37
 Richiamo di funzioni di conteggio 13-1
 Richiamo di funzioni di temporizzazione 13-8
 Richiamo di una sorgente S7-SCL 6-11
 RIGHT 14-14
 Ripristino di un'azione di editazione 4-10
 ROL 14-11
 ROR 14-11
 ROUND 14-6

S

S_CD 13-5
 S_CU 13-5
 S_CUD 13-6
 S_ODTS 13-16
 S_OFFDT 13-17
 S_PEXT 13-14
 S_PULSE 13-13
 S5-Time 13-11
 S5TIME 7-4
 S7-SCL 1-2, 1-4, 4-2
 modalità operativa 1-4
 vantaggi 1-2
 Salto di programma 12-12
 Salto pagina 4-25
 Salvataggio di una sorgente S7-SCL 4-23
 Segnalibro 4-16
 Selezione del temporizzatore giusto 13-19
 Selezione di oggetti di testo 4-11
 SFC/SFB 14-27
 SHL 14-11
 SHR 14-11
 SIN 14-10
 Singoli passi 4-30
 Somma 11-2
 Sorgente 4-4 - 4-24,
 6-10, 6-20
 Sostituzione di oggetti di testo 4-10
 Sottrazione 11-2
 SQR 14-9
 SQRT 14-9
 Stack 4-43
 Stampa di una sorgente S7-SCL 4-24
 Stato di funzionamento 4-40
 Stile e colore del carattere 4-15, 4-25
 STRING 7-7, 7-8, 9-9, 9-11, 9-12,
 14-13 - 14-20
 STRING_TO_CHAR 14-4
 Stringhe di caratteri 5-13

STRNG_DI 14-19
 STRNG_I 14-19
 STRNG_R 14-20
 STRUCT 7-11, 7-12
 Struttura del blocco 6-3
 Struttura della parte convenzioni 6-8
 Struttura di sorgenti S7-SCL 6-12
 Struttura di un blocco dati (DB) 6-17
 Struttura di un blocco funzionale (FB) 6-12
 Struttura di un blocco organizzativo (OB) 6-16
 Struttura di una funzione(FC) 6-14
 Struttura di una sorgente S7-SCL 6-8 - 6-17
 Strutture 7-11
 Strutture delle regole 5-1
 Superficie operativa 4-2
 Superficie operativa di S7-SCL 4-2

T

Taglio di oggetti di testo 4-12
 TAN 14-10
 Tempo di ciclo 4-42
 Temporizzatori 13-10 - 13-18
 assegnazione di parametri nelle funzioni
 temporali 13-10
 Attivare il tempo come ritardo all'inserzione
 (S_ODT) 13-15
 Avvia temporizzatore come impulso
 (S_PULSE) 13-13
 Avvia temporizzatore come impulso
 prolungato (S_PEXT) 13-14
 Avvia temporizzatore come ritardo alla
 disinserzione (S_OFFDT) 13-17
 Avvia temporizzatore come ritardo
 all'inserzione con memoria
 (S_ODTS) 13-16
 esempi 13-18
 introduzione ed analisi del valore temporale
 13-11
 Temporizzatori (funzioni temporali) 13-8
 Temporizzazione 13-8
 richiamo di funzioni di temporizzazione 13-8
 Terminali delle regole lessicali (diagrammi
 sintattici) 15-4
 Terminologia di base SCL 5-4 - 5-13
 Test con punti d'arresto 4-30
 Test con singoli passi 4-30
 TIME 7-4
 TIME_OF_DAY 7-4
 TIME_TO_DINT 14-4
 TIMER (tipo di dati) 7-15
 Tipi di dati 7-1 - 7-13
 composti 7-1
 definiti dall'utente (UDT) 6-20
 definitl dall'utente (UDT) 7-13
 descrizione 7-1
 semplici 7-1
 Tipi di dati a bit 7-3
 Tipi di dati a caratteri 7-3

Tipi di dati composti 7-2, 7-5, 7-7
Tipi di dati definiti dall'utente (UDT) 6-20, 12-3
Tipi di dati definiti dall'utente (UDT) 7-13
Tipi di dati numerici 7-3
Tipi di dati per i parametri 7-15
Tipi di dati per parametri 7-15, 7-16
Tipi di dati semplici 7-1, 7-3, 7-4
Tipi di licenza 2-1
 Enterprise License 2-1
 Floating License 2-2
 Rental License 2-2
 Single License 2-2
 Trial License 2-2
 Upgrade License 2-2
Tipo di dati ANY 7-18
Tipo di dati ARRAY 7-9
Tipo di dati BLOCK 7-16
Tipo di dati COUNTER 7-15
Tipo di dati DATE_AND_TIME 7-5
Tipo di dati POINTER 7-16
Tipo di dati STRING 7-7
Tipo di dati STRUCT 7-11
Tipo di dati TIMER 7-15
Tipo di dati UDT 7-13
TOD_TO_DINT 14-4
TRUNC 14-6

U

UDT 7-13, 7-14
 definizione 6-20, 6-21
 elementi 6-20
 richiamo 6-20
Uguaglianza 11-2

V

Vai a 4-13
Valore di ritorno (FC) 12-37
Valori iniziali 8-3
vantaggi di S7-SCL 1-2
VAR 8-8
VAR_IN_OUT 8-8
VAR_INPUT 8-8

VAR_OUTPUT 8-8
VAR_TEMP 8-8
Variabile ampliata 11-4
Variabili
 controllo e comando delle variabili 4-38
 dei blocchi convenzioni 8-8
 Dichiarazione di istanze 8-6
 inizializzazione 8-3, 8-4
 sintassi generale di una dichiarazione di
 variabili o parametri 8-2
 statiche 8-1
 temporanee 8-1
 variabili locali e parametri di blocco 8-1
 variabili statiche 5-17
 variabili temporanee 5-17
Variabili statiche 5-17, 8-1 - 8-9
Variabili temporanee 5-17, 8-1, 8-10
Verifica coerenza blocchi 4-38
Visualizzazione degli stack della CPU 4-43
Visualizzazione dei blocchi della CPU 4-42
Visualizzazione del tempo di ciclo
 della CPU 4-42
Visualizzazione delle caratteristiche
 dell'orologio della CPU 4-42
Visualizzazione delle informazioni per
 comunicare con la CPU 4-42
Visualizzazione e impostazione di data e ora
 della CPU 4-40
Visualizzazione e modifica dello stato di
 funzionamento della CPU 4-40
Visualizzazione/compressione della memoria
 utente della CPU 4-41

W

WORD 7-3
WORD_TO_BLOCK_DB 14-4
WORD_TO_BOOL 14-4
WORD_TO_BYTE 14-4
WORD_TO_INT 14-4

X

XOR 11-10

