



# Azionamenti e sistemi per l'automazione e la domotica

Luca Sani

Dipartimento di Sistemi Elettrici e Automazione

Università di Pisa

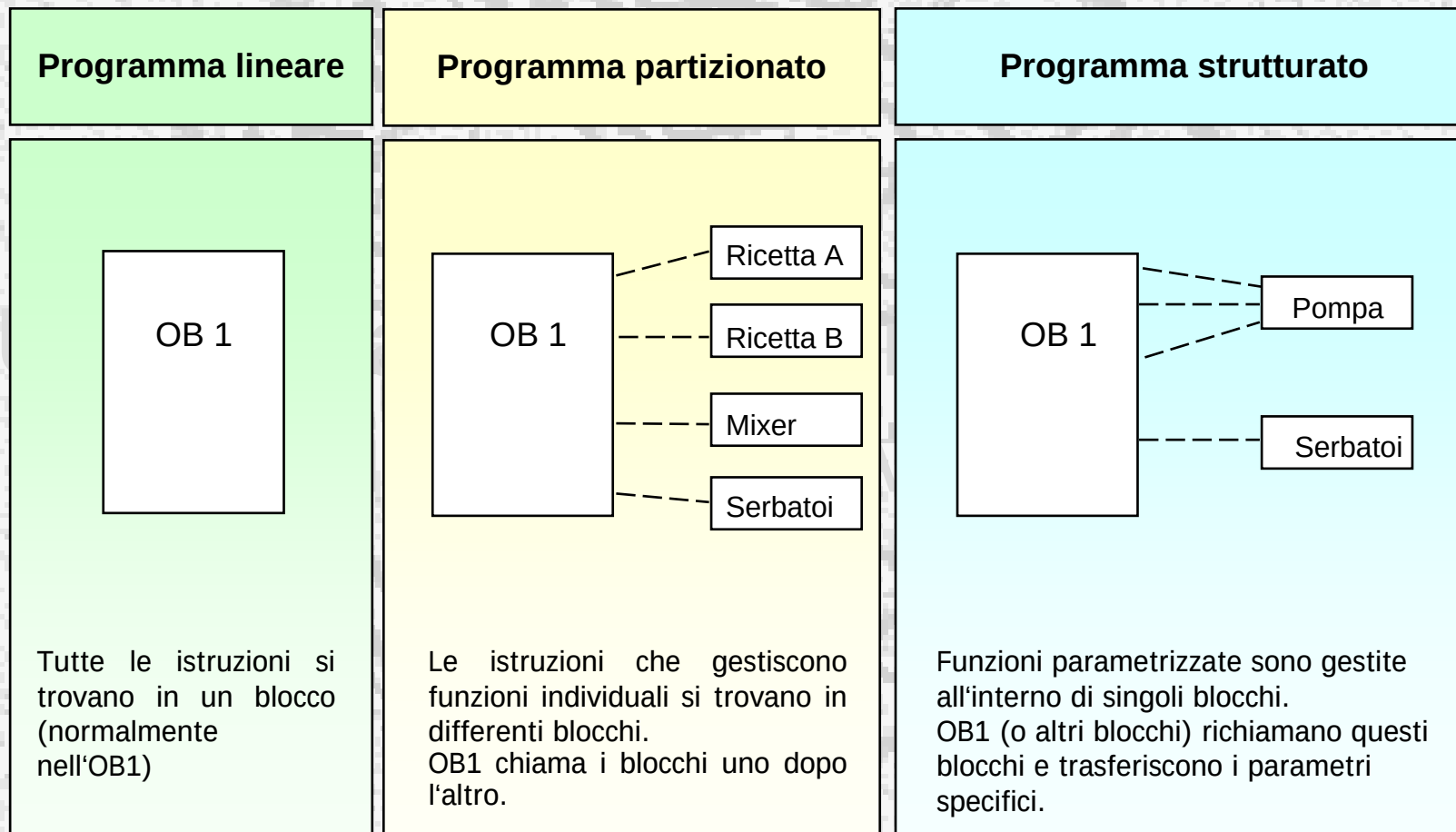
tel. 050 2217364

email [luca.sani@dsea.unipi.it](mailto:luca.sani@dsea.unipi.it)



# Programmazione strutturata con lo Step 7

# Struttura del programma



# Struttura del programma

**Programma lineare** L'intero programma utente si trova in un unico blocco. Questo modo di programmare è molto simile, dal punto di vista concettuale, alla logica cablata.

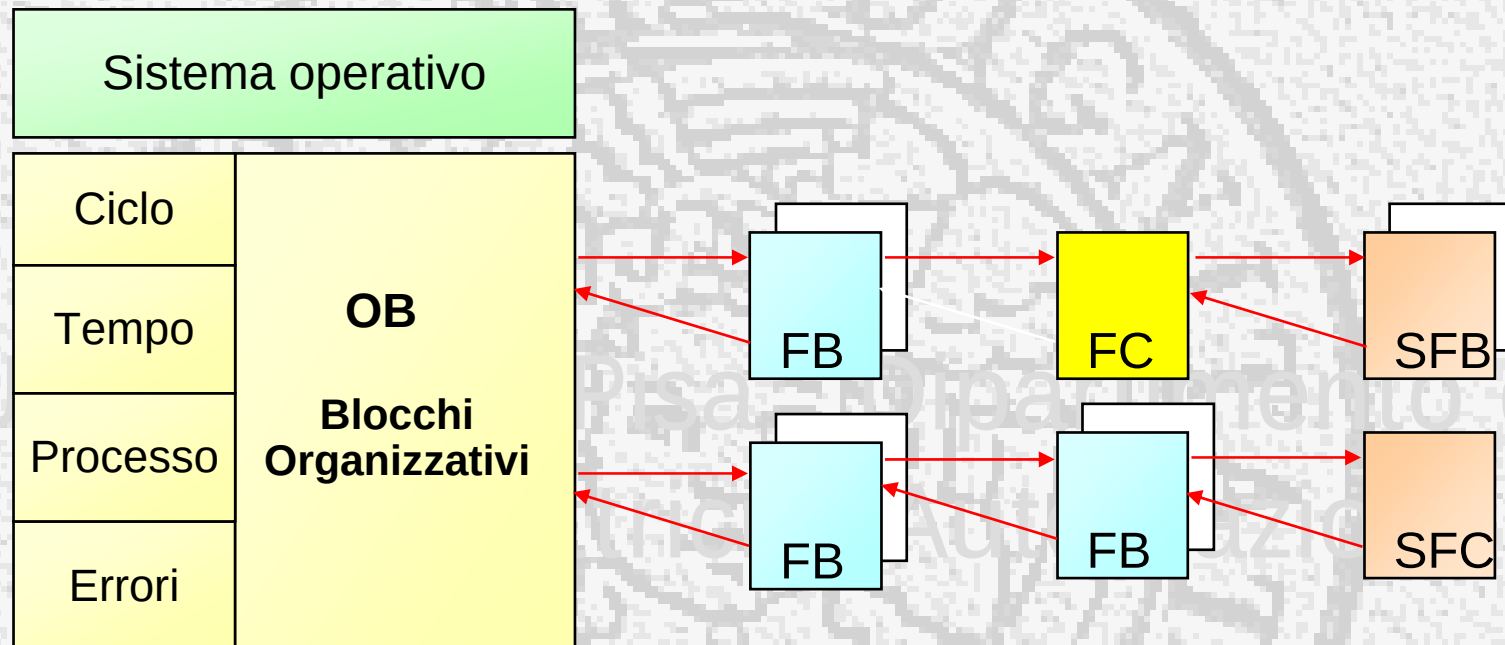
**Programma partizionato** Il programma utente si divide in blocchi, dove ogni blocco gestisce una parte dell'impianto. Il blocco organizzativo OB1 contiene le istruzioni che richiamano gli altri blocchi in una sequenza definita.

**Programma strutturato** Un programma strutturato contiene blocchi parametrizzati (funzioni). Questi blocchi sono progettati in modo da essere usati in molti campi differenti. Quando si assegna un operando reale ad un parametro formale, il blocco parametrizzato esegue il proprio programma utilizzando al posto dell'operando formale l'operando reale.

Esempio:

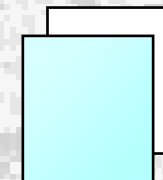
- Il "blocco pompa" contiene istruzioni per il controllo di una pompa generica.
- I blocchi di programma responsabili della gestione di specifiche pompe, richiamano il "blocco pompa" e forniscono i parametri specifici per la gestione della singola pompa.
- Quando il "blocco pompa" ha completato l'esecuzione delle sue istruzioni, il programma ritorna al blocco chiamante che continua con l'elaborazione delle istruzioni successive alla chiamata del "blocco pompa" .

# I blocchi di programma



OB = Blocchi Organizzativi  
FB = Blocchi Funzionali  
FC = Blocchi Funzioni  
SFB = Blocchi Funzionali di Sistema  
SFC = Blocchi Funzioni di Sistema

## Legenda :



FB con  
Blocco Dati  
d'istanza



FC

Creazione di un  
blocco di  
programma (FC)

## Blocco di programma (FC)

Un blocco FC può essere:

a) Non parametrico

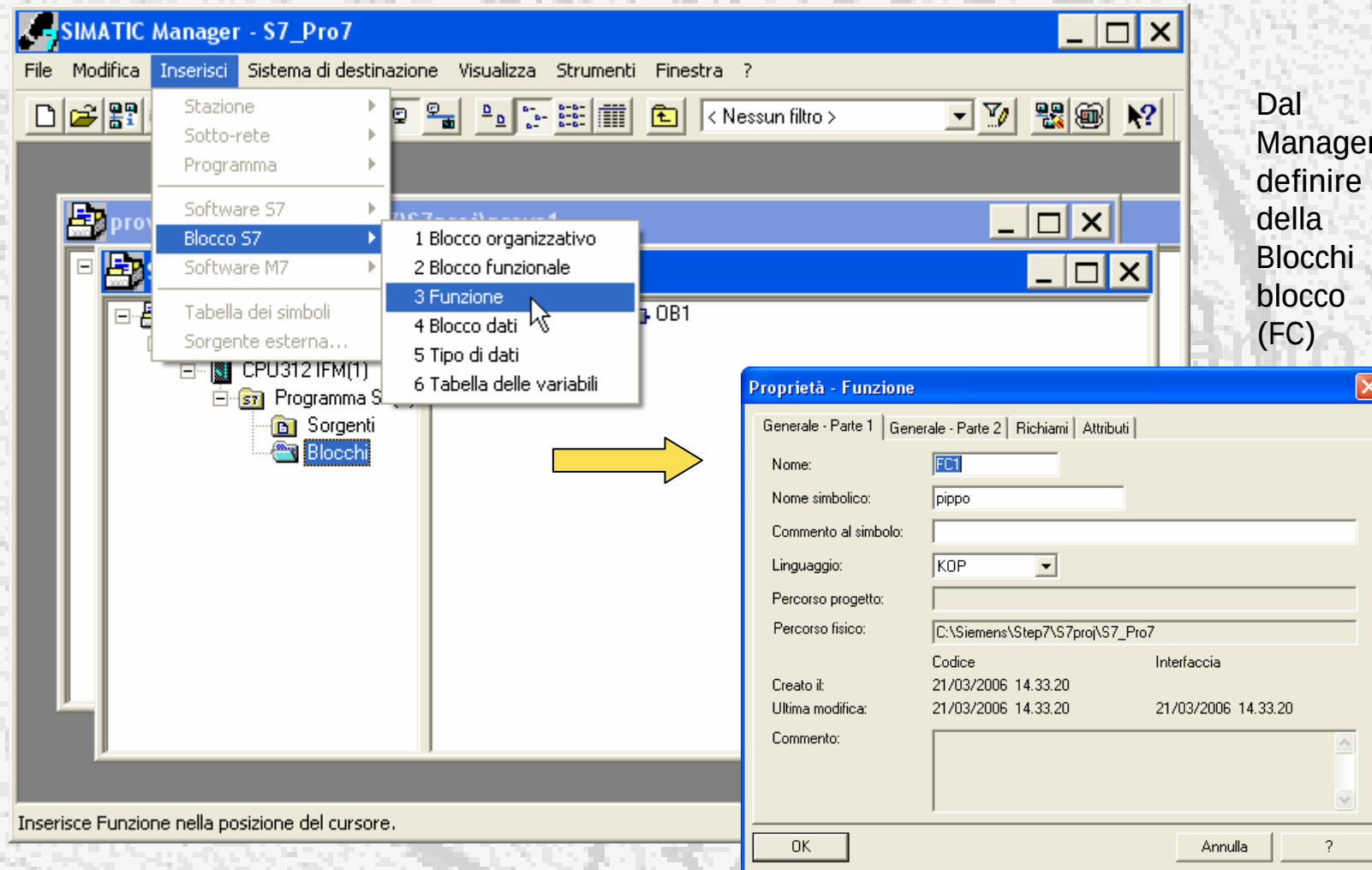
ogni blocco contiene una parte del programma. Le variabili sono quelle assolute. Si usano per semplificare la struttura del programma. Ogni blocco è richiamato una sola volta.

a) Parametrico

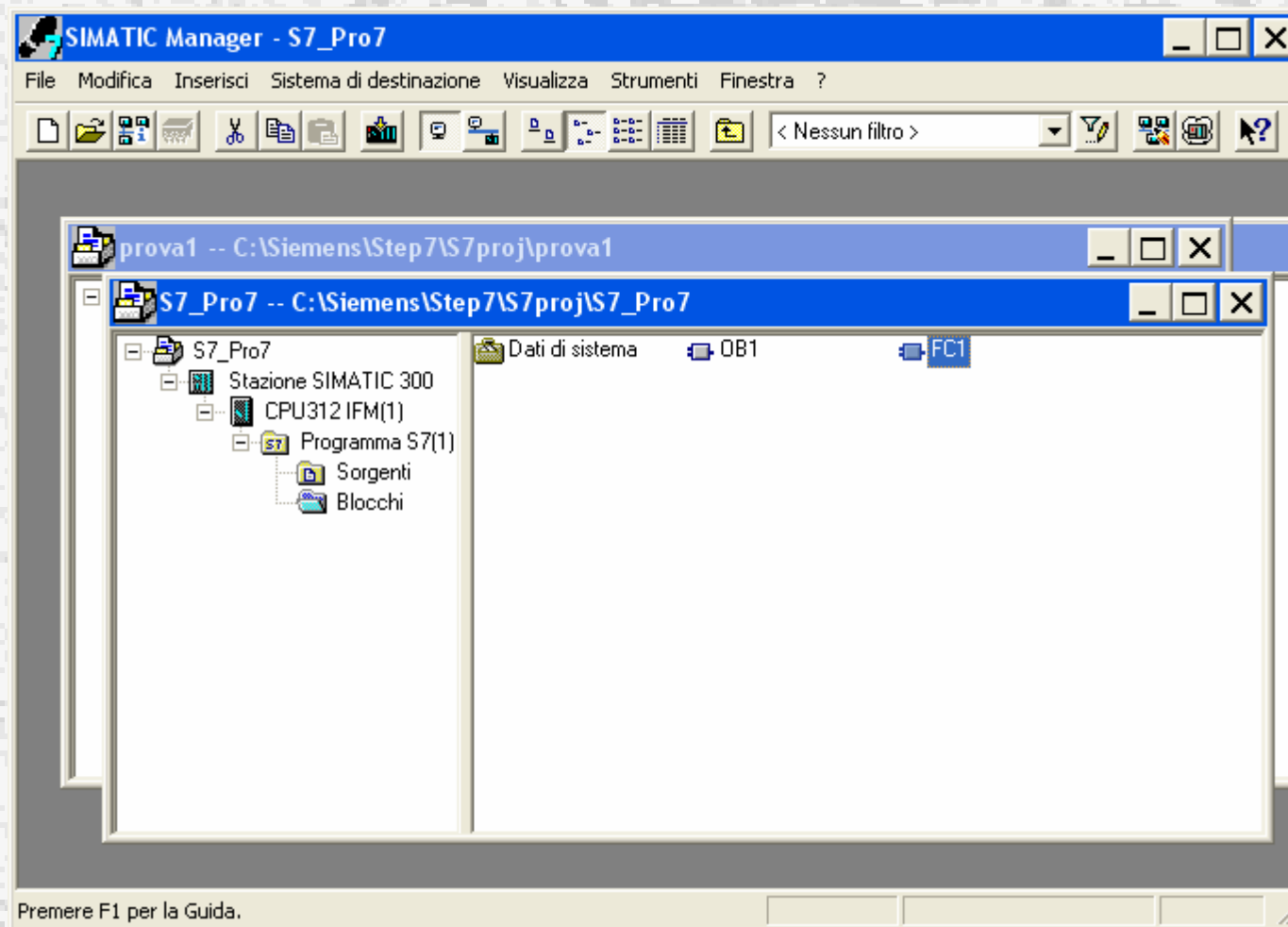
all'interno del blocco le variabili sono definite in modo formale. Si ha un passaggio per valori.

Un blocco può essere richiamato più volte.

# Creazione di un blocco di programma (FC)

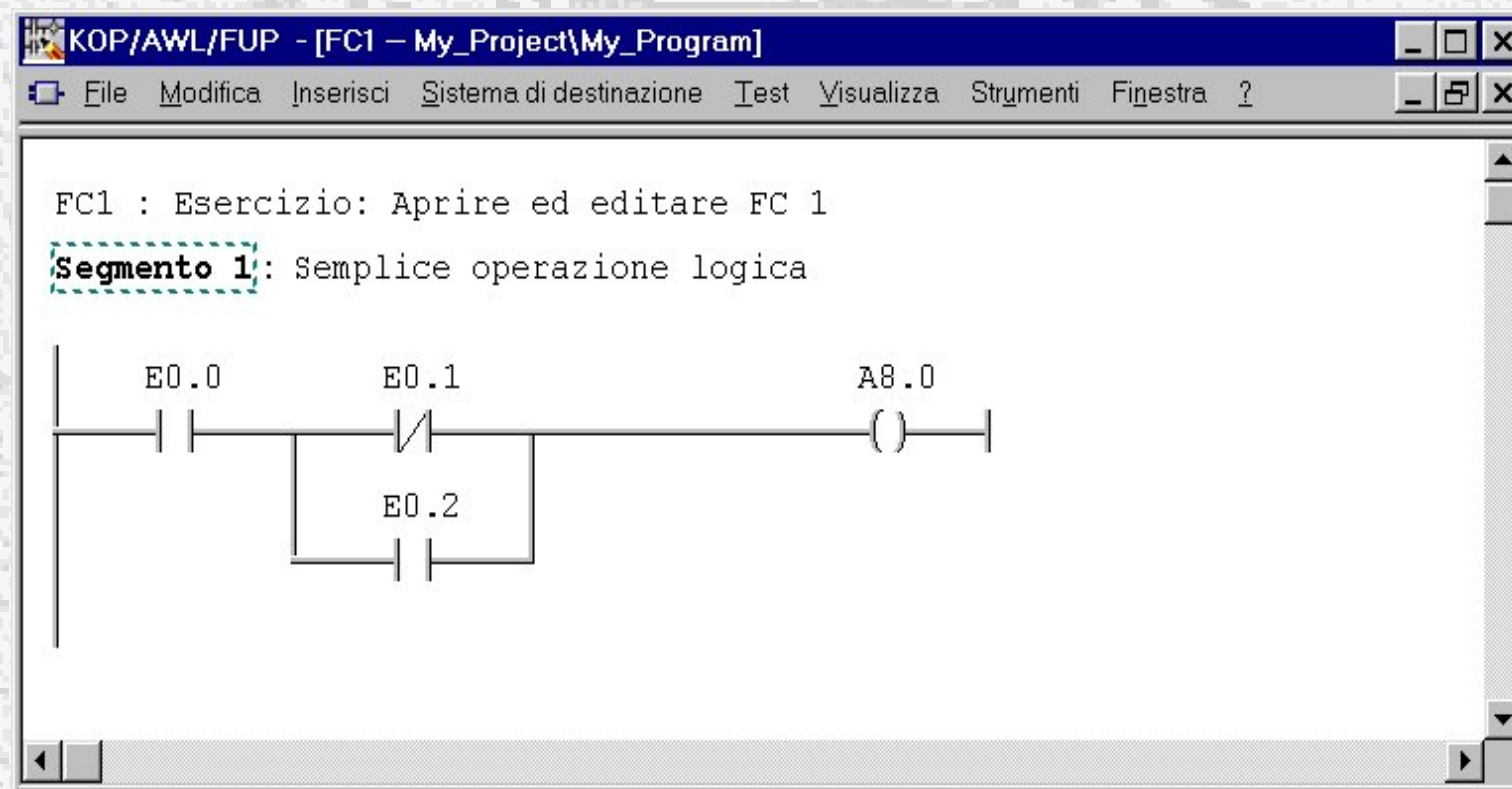


# Creazione di un blocco di programma (FC)



Nella cartella blocchi del progetto appare il simbolo di un blocco FC (per default viene assegnato un nome progressivo).

## Inserimento del codice nel blocco FC



Cliccando due volte sul simbolo del blocco FC si entra nell'editor del programma, all'interno del quale è possibile inserire il codice nel linguaggio scelto.

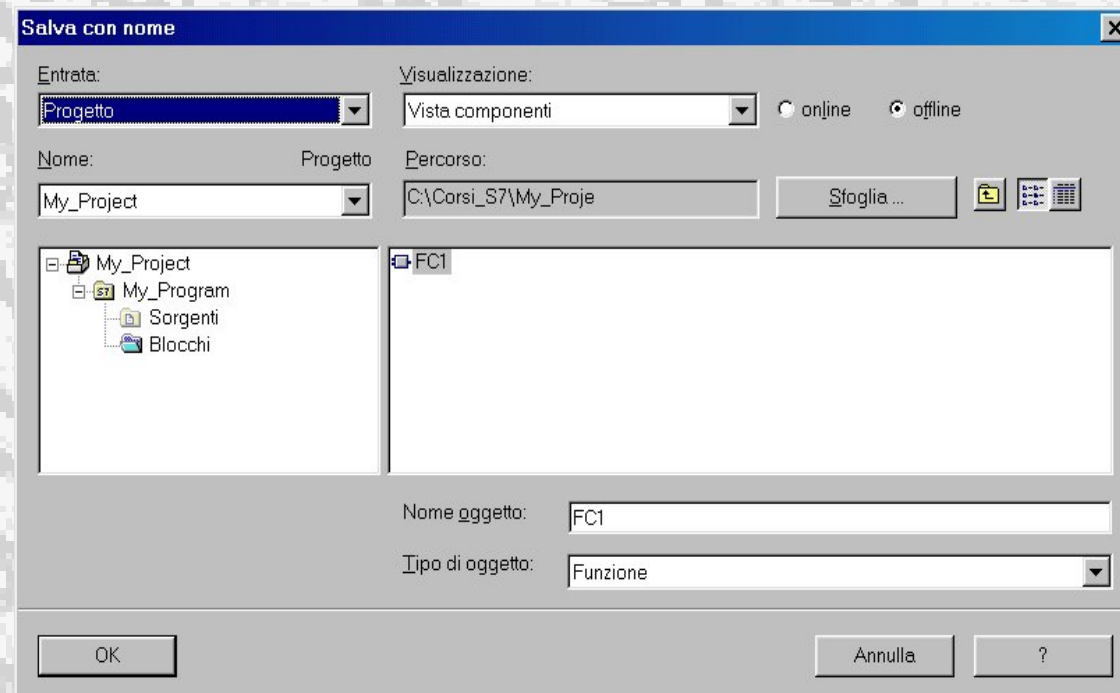
## Salvare FC 1

Per salvare un blocco o un file  
senza cambiare il nome ...  
Scegliere il comando  
del menu  
*File -> Salva*

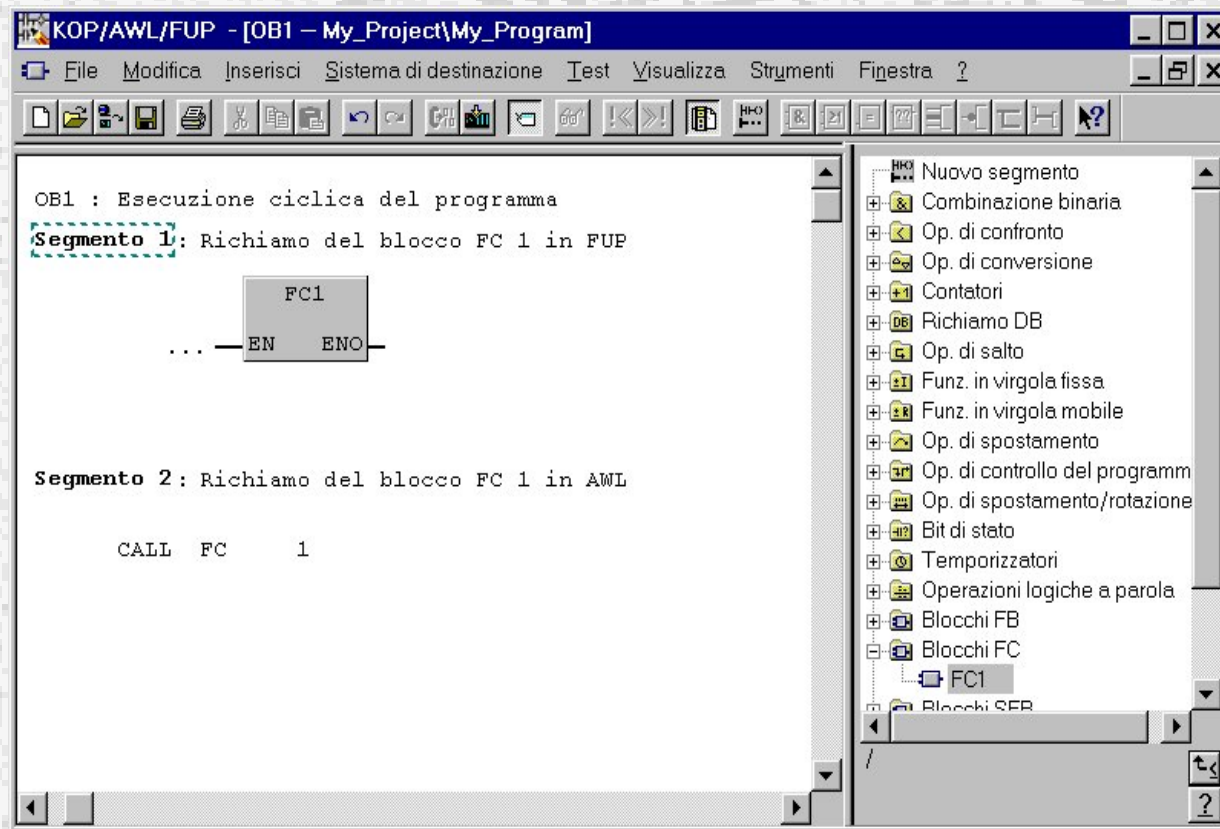


...o cliccare

Per salvare un blocco con  
un nome diverso o in una  
posizione differente:  
*File -> Salva come*



# Richiamo di un blocco in OB1



## Esecuzione Ciclica

Affinché un nuovo blocco venga eseguito dalla CPU occorre che esso venga richiamato nell'OB1. La maniera più semplice per inserire il richiamo ad un blocco è utilizzando la finestra elementi di programma.

Per default il blocco ha in ingresso la variabile EN e l'uscita è rappresentata dalla variabile ENO

# Uso dei Parametri EN/ENO nei richiami di blocco

**FC Standard** Esistono le seguenti regole per l'esecuzione degli FC standard:

- Se  $EN=0$ , il blocco non viene eseguito e  $ENO = 0$ .
- Se  $EN=1$ , il blocco viene eseguito, e se è eseguito senza errori  $ENO = 1$ .

In caso di errore durante l'esecuzione del blocco,  $ENO$  diventa  $=0$ .

**FC utente** Non è importante se un blocco utente sia scritto in KOP, FUP o AWL; quando viene richiamato in KOP o FUP, i parametri EN e ENO sono aggiunti automaticamente; in questo modo è possibile utilizzare il valore dell'RLC per abilitare la funzione.

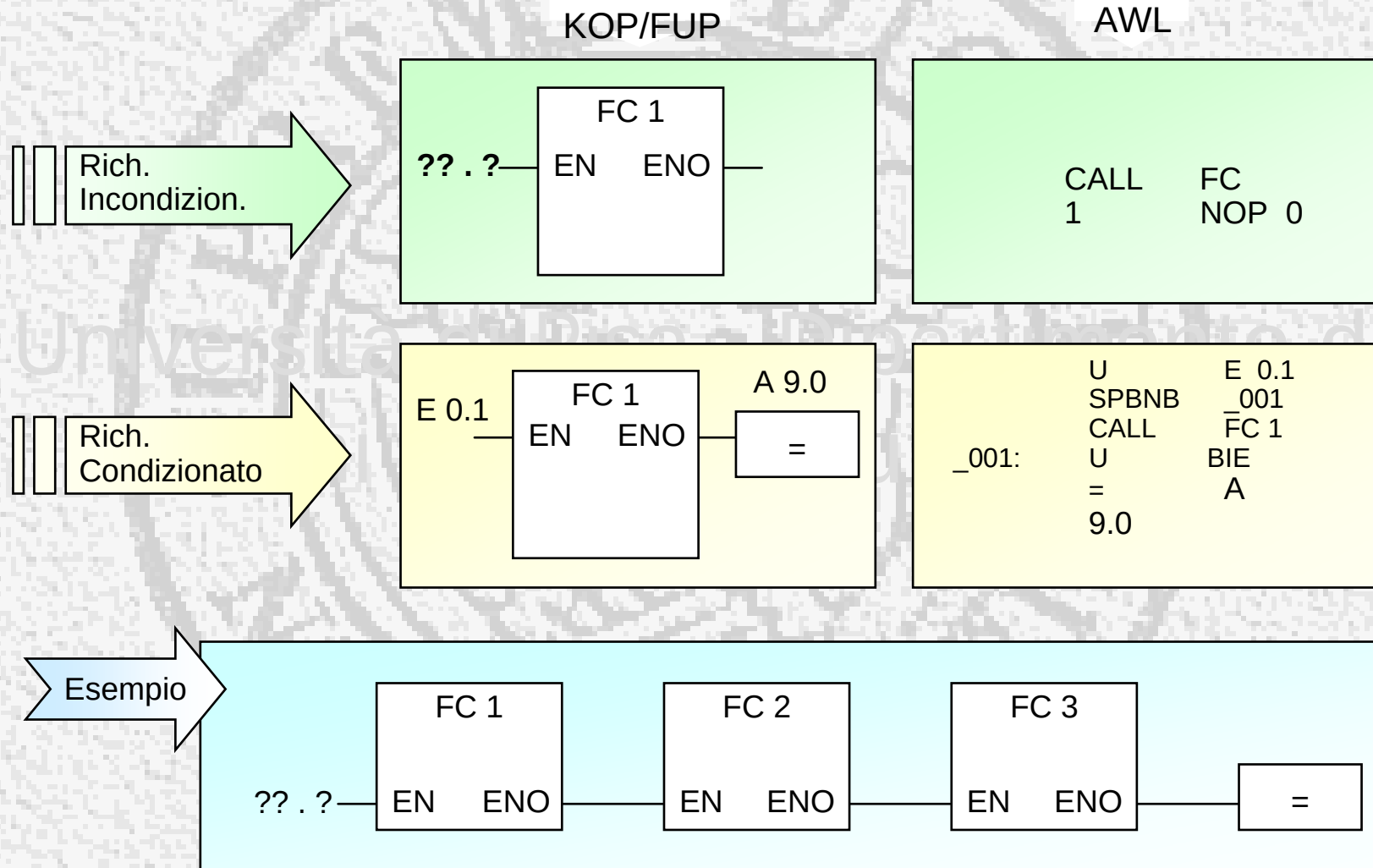
EN/ENO non esistono in AWL, ma si può in ogni caso emularli.

E' opportuno programmare -indipendentemente dal linguaggio di programmazione - una valutazione d'errore.

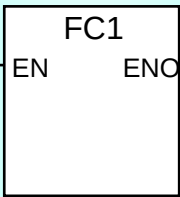
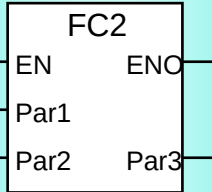
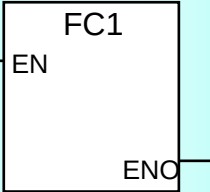
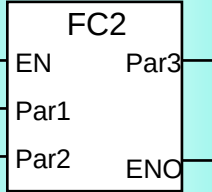
## Interconnessione

In KOP/FUP, diversi blocchi possono essere raggruppati insieme, uno dopo l'altro, e collegati logicamente tramite i parametri EN / ENO.

# Uso dei Parametri EN/ENO nei richiami di blocco



## Richiami di blocchi

| Lin-<br>guaggio | FC                                                                                                                                                                            |                                                                                                        |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------|
|                 | Senza parametri                                                                                                                                                               | Con parametri                                                                                          |
| AWL             | <ul style="list-style-type: none"> <li>• CALL FC1</li> <li>• UC FC1</li> <li>• CC FC1</li> </ul>                                                                              | <ul style="list-style-type: none"> <li>• CALL FC2<br/>Par1: ...<br/>Par2: ...<br/>Par3: ...</li> </ul> |
| KOP             | <p>FC1<br/>—— ( CALL )</p> <hr/>                                                             |                      |
| FUP             |  <hr/>  |                    |

**CALL** L'istruzione "CALL" viene usata per richiamare blocchi di programma indipendentemente dal valore dell'RLC o da altre condizioni.

Si può usare indifferentemente il nome assoluto o simbolico del blocco richiamato.

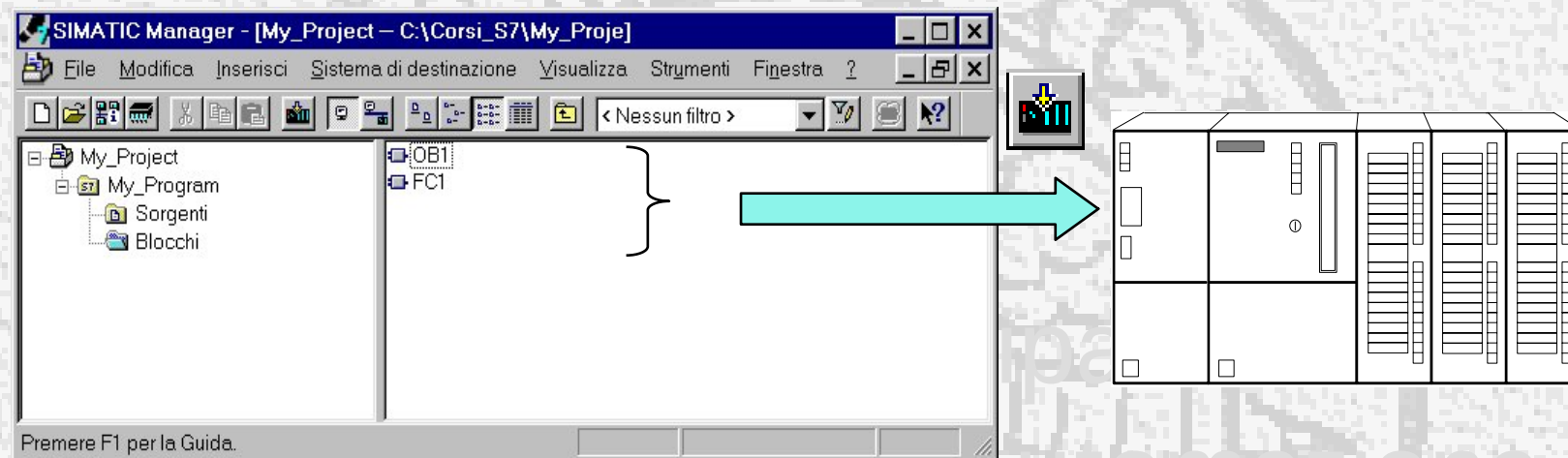
**UC** L'istruzione "UC" è un richiamo incondizionato di un blocco di tipo FC o FB, senza assegnazione parametrica. "UC" è per gli altri aspetti identico a "CALL".

**CC** L'istruzione "CC" richiama un blocco di tipo FC o FB senza parametri se l'RLC=1. "CC" è per gli altri aspetti identico a "CALL".

**Parametri** I parametri dichiarati nella tabella di dichiarazione sono conosciuti come "parametri formali". Gli indirizzi o i valori specificati durante il richiamo, vanno intesi riferiti ai "parametri attuali".

Le variabili statiche e temporanee non vanno specificate durante il richiamo.

## Caricamento dei blocchi nel simulatore (o nel PLC)



**Caricamento** I blocchi vengono caricati nel PLC (o simulatore di PLC) dal SIMATIC Manager :

- cliccando sull'icona oppure
- scegliendo il menu *Sistema di destinazione* -> *Carica*.

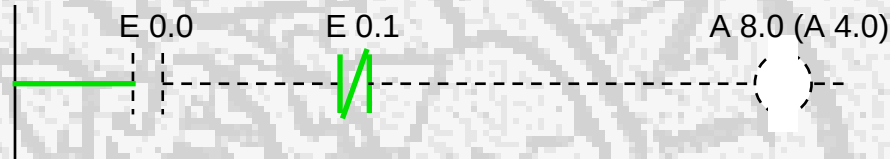
Prima di fare questo occorre selezionare i blocchi da caricare:

- **Tutti i blocchi:** Selezionare la cartella "blocchi" nel riquadro di sinistra della finestra del progetto (l)
- **Alcuni blocchi:** Premere il tasto CTRL e selezionare i blocchi
- **Un solo blocco:** Selezionare il blocco.

# Testare FC 1 (in KOP)

Per ogni blocco si può fare il debug con l'opzione online

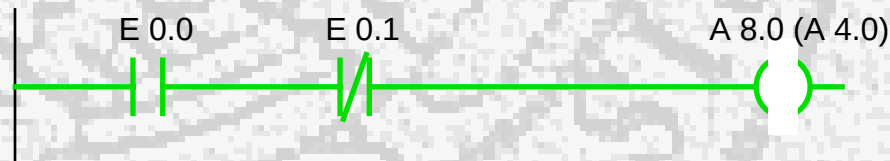
| Input | Stato del segnale | Risultato del controllo |
|-------|-------------------|-------------------------|
| E 0.0 | 0                 | 0                       |
| E 0.1 | 0                 | 1                       |



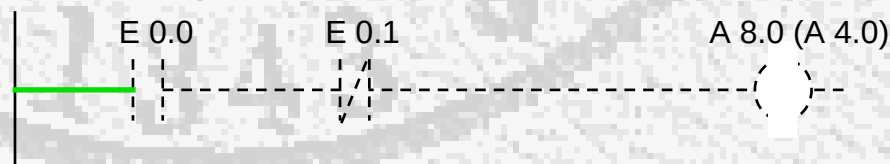
|       |   |   |
|-------|---|---|
| E 0.0 | 1 | 1 |
| E 0.1 | 1 | 0 |



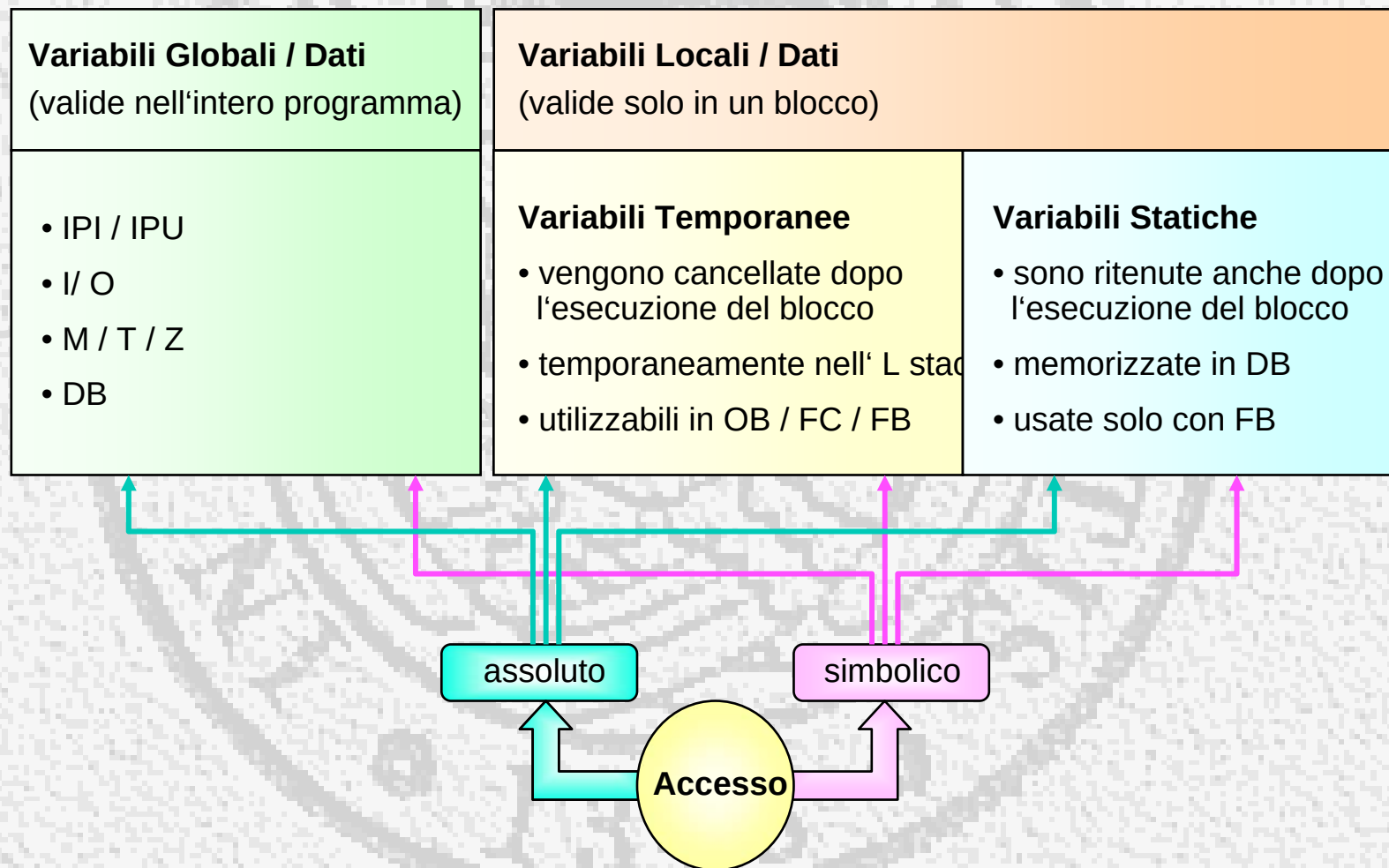
|       |   |   |
|-------|---|---|
| E 0.0 | 1 | 1 |
| E 0.1 | 0 | 1 |



|       |   |   |
|-------|---|---|
| E 0.0 | 0 | 0 |
| E 0.1 | 1 | 0 |



# Introduzione al concetto di variabili locali e statiche



# Introduzione

## Generale

Fino ad ora, gli ingressi e le uscite sono stati indirizzati attraverso il loro valore attuale.

E' però possibile assegnarli in modo parametrico al blocco che ne fa uso. Si può operare come abbiamo fatto fino ad ora, per esempio, per creare programmi che vengano usati una sola volta in modo specifico.

Per funzioni che ricorrano frequentemente in sistemi più complessi, è opportuno creare blocchi d'uso **universale**, tramite assegnamento parametrico (FC, FB). Questi possono usare parametri formali per gli ingressi e le uscite, ai quali verrà assegnato il valore attuale del parametro nel momento del richiamo del blocco stesso.

L'allineamento della funzionalità del blocco all' hardware avviene nella fase di assegnamento dei parametri, quando il blocco viene richiamato; la "funzionalità interna" del blocco non viene mai modificata.

# Introduzione

**Variabili Locali** Fino ad ora abbiamo fatto uso di variabili globali (bit di memoria o blocchi dati). In questo capitolo si troveranno indicazioni per l'uso delle variabili locali.

**Variabili temporanee** Le variabili temporanee sono variabili che vengono memorizzate solo mentre il blocco in cui se ne fa uso è in esecuzione. Possono essere usate in tutti i blocchi (OB, FC, FB).

**Variabili statiche** Se i dati devono essere memorizzati anche dopo che il blocco sia stato eseguito, allora devono essere salvati in variabili statiche. Le variabili statiche possono essere usate solo in blocchi funzionali (FB).

## Variabili Temporanee

**Generale** Le variabili temporanee possono essere usate in tutti i blocchi (OB, FC, FB). Vengono utilizzate per memorizzare temporaneamente, per il tempo in cui il blocco che le usa è in esecuzione. I dati verranno perduti all'uscita dal blocco. I dati vengono memorizzati nell' L stack (registro delle variabili locali). Questa è una zona di memoria a parte, all'interno della CPU.

**Dichiarazione** Si definiscono queste variabili nella tabella di dichiarazione di ogni singolo blocco. Nella linea "temp" viene introdotto il nome della variabile e il tipo dati relativo. Non è possibile per esse definire un valore iniziale. Dopo avere salvato il blocco, nella colonna "Indirizzo" compare la locazione all'interno dell' L-stack.

**N.B. #** I nomi di variabile che iniziano con il carattere speciale # sono variabili locali che hanno validità solo all'interno del blocco in cui sono dichiarate. L'Editor di programma introduce automaticamente il carattere speciale.

# Variabili Temporanee

The screenshot shows the SIMATIC Manager software interface. At the top, the title bar reads "KOP/AWL/FUP - [FC1 - My\_Project\SIMATIC 300(1)\CPU315-2 DP]". Below the title bar is a menu bar with options: File, Modifica, Inserisci, Sistema di destinazione, Test, Visualizza, Strumenti, Finestra, and ?. A toolbar with various icons is located below the menu bar.

Below the toolbar is a table with the following columns: Indirizzo, Dichiarazione, Nome, Tipo, Valore iniziale, and Commento.

| Indirizzo | Dichiarazione | Nome      | Tipo | Valore iniziale | Commento |
|-----------|---------------|-----------|------|-----------------|----------|
|           | in            |           |      |                 |          |
|           | out           |           |      |                 |          |
|           | in_out        |           |      |                 |          |
| 0.0       | temp          | risultato | INT  |                 |          |

Below the table, the main workspace shows the following text:

```
FC1 : Variabili temporanee  
Segmento 1: Calcolare e salvare il risultato intermedio
```

The workspace also displays a ladder logic diagram. It features a subtraction function block labeled "SUB\_I". The block has two input terminals: "IN1" and "IN2". The output terminal is labeled "OUT" and is connected to a variable named "#risultato". The inputs are connected to memory addresses: "MW102" is connected to "IN1" and "MW100" is connected to "IN2". The block also has "EN" (enable) and "ENO" (enable output) terminals.

At the bottom of the workspace, there is a status bar with the text "Premere F1 per richiedere la Guida." and several buttons: "offline", "Ass", "INS", and "Modifica".

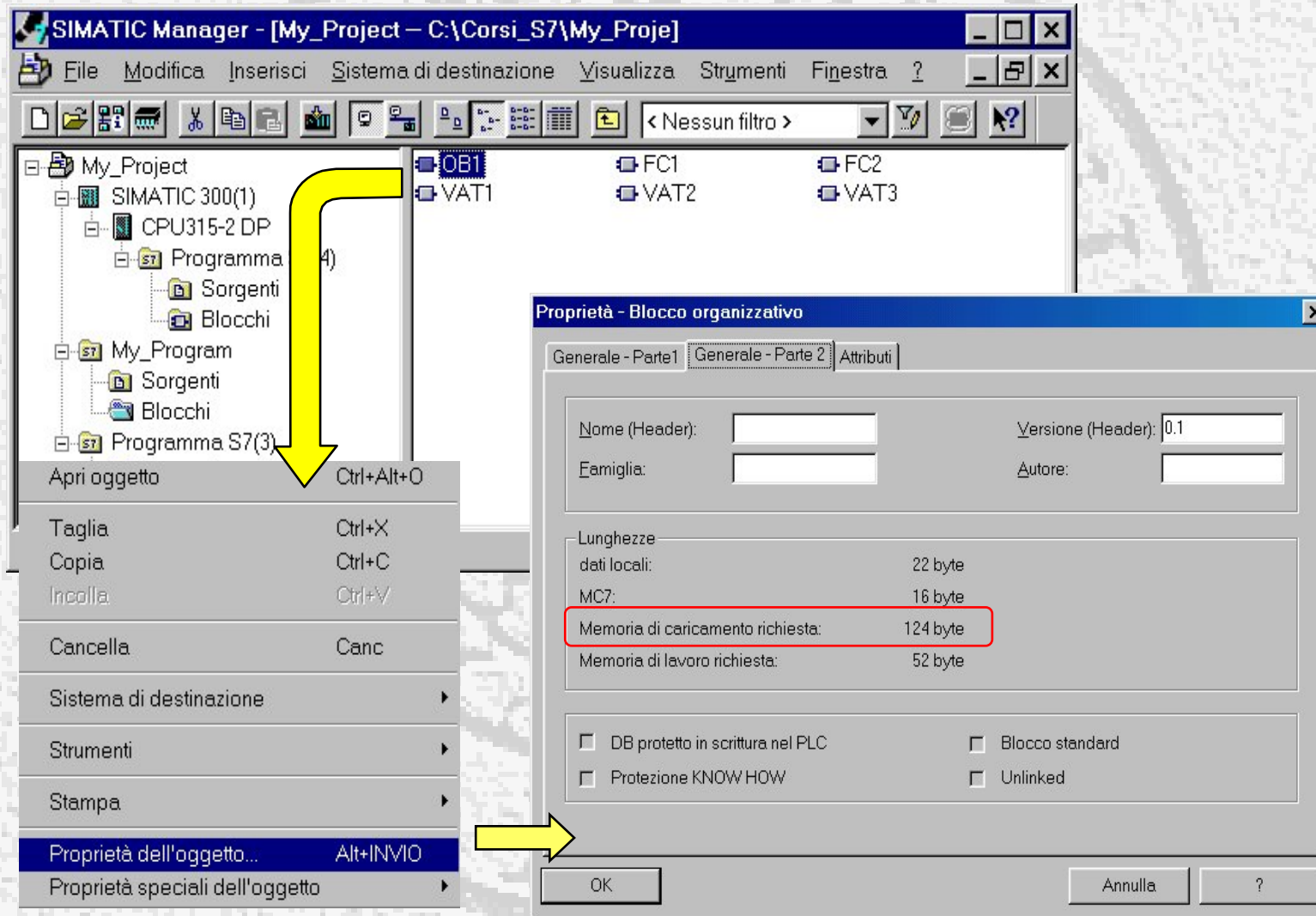
**Accesso** Nel segmento 1, si può vedere un esempio di accesso simbolico ad una variabile temporanea. Il risultato della sottrazione viene appoggiato nella variabile temporanea "risultato". Si può anche usare l'accesso assoluto (T LW0). Si dovrebbe, comunque, cercare di evitarlo, poiché rende di difficile lettura il programma.

# Dimensione del registro delle Variabili Locali

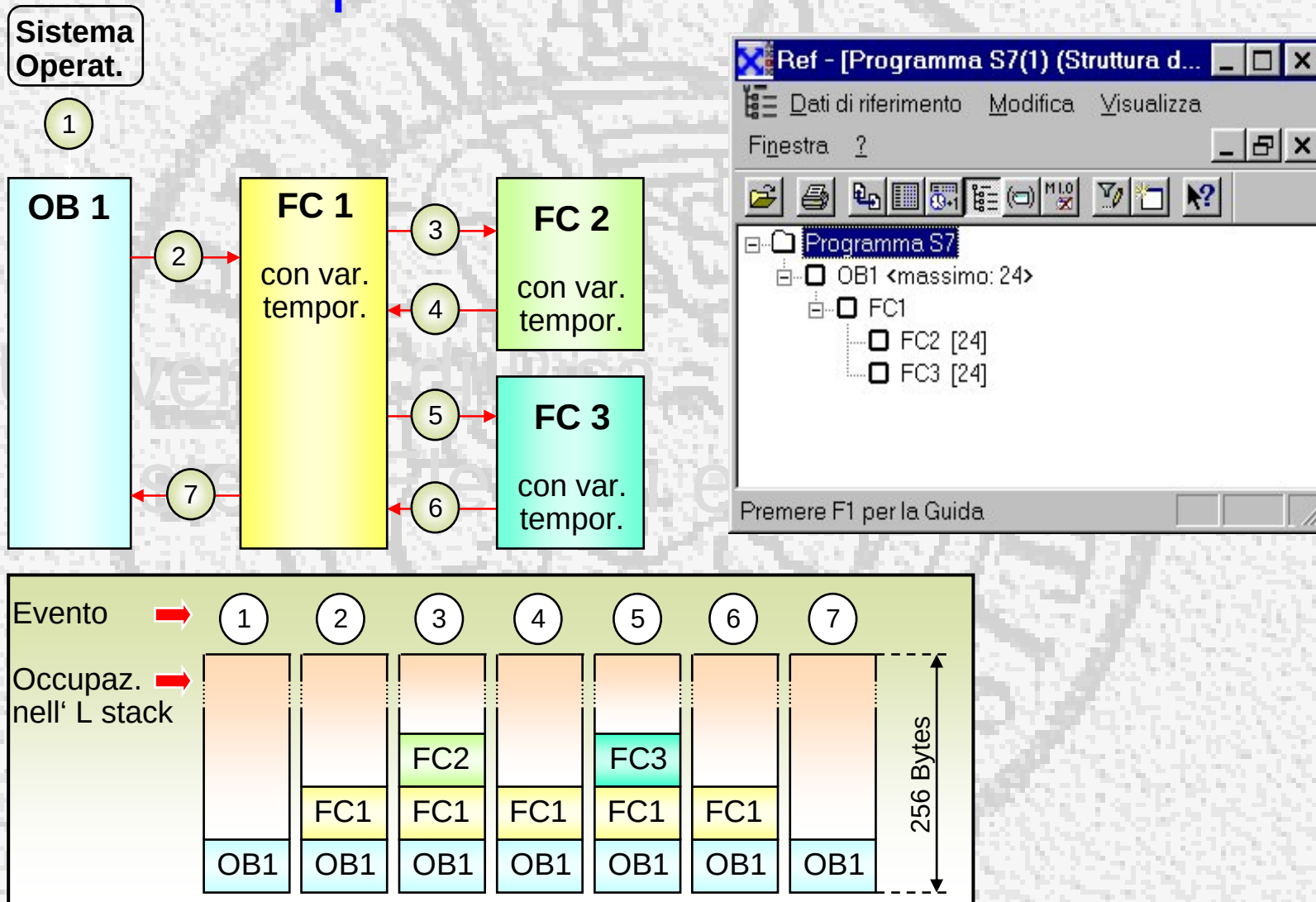
**Dimens. totale:**  
**1.5 Kbyte**  
**(CPU 313..316)**

| Elaborazione               |                         | Per S7-300:        |                    |
|----------------------------|-------------------------|--------------------|--------------------|
|                            |                         | Classe di priorità | Dimensione L stack |
| Avvio (eseguito una volta) |                         | 27                 | 256 bytes          |
| Elaborazione Ciclica       |                         | 1                  |                    |
| Elaborazione a tempo       | Allarme dall'orologio   | 2                  | 256 bytes          |
|                            | Allarme di ritardo      | 3                  | 256 bytes          |
|                            | Interrupt periodici     | 12                 | 256 bytes          |
| Elaborazione su Evento     | Interrupt Hardware      | 16                 | 256 bytes          |
|                            | Errore durante Avvio    | 28                 | 256 bytes          |
|                            | Errore durante il ciclo | 26                 |                    |

# Occupazione di un Blocco nell' L Stack



# Occupazione totale nell' L Stack



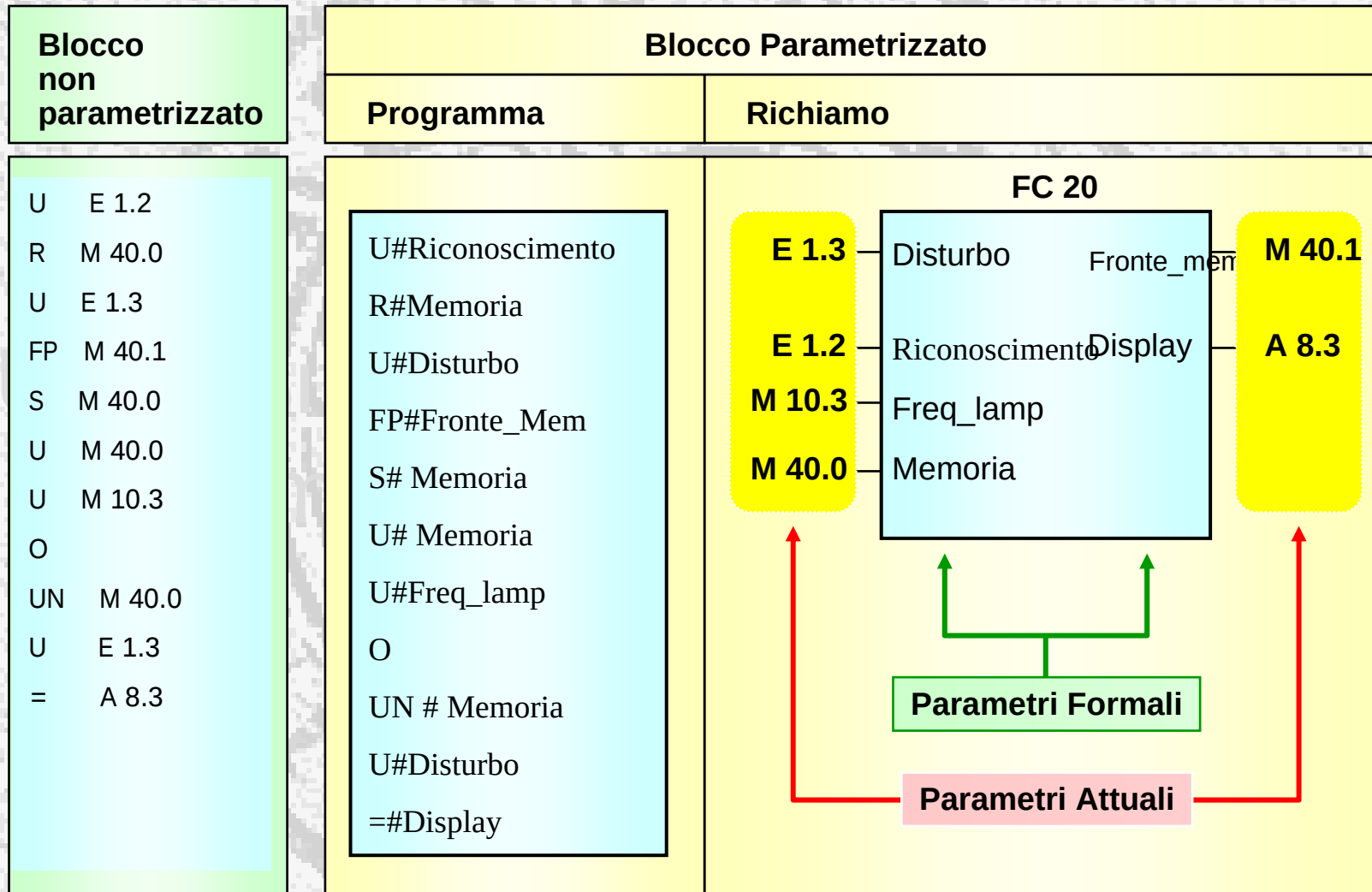
## Blocchi Parametrizzati

Si può usare l'assegnazione dei parametri ai blocchi quando si abbiano sezioni di programma il cui uso ricorra frequentemente. Questo comporta i seguenti vantaggi:

- il programma deve essere scritto una sola volta
- il blocco viene memorizzato una sola volta, ma è possibile richiamarlo tutte le volte volute
- il blocco può essere programmato facendo uso di parametri formali (di tipo IN, OUT, oppure IN/OUT) e solo quando il blocco viene richiamato viene ad essi assegnato l'indirizzo "reale" (parametri attuali).

**FC / FB** Può essere eseguito un assegnamento parametrico sia con blocchi di tipo FC che di tipo FB.

# Blocchi Parametrizzati



**Esempio** Quando il blocco viene eseguito, l'istruzione "U # Disturbo" verifica quale parametro attuale sia da assegnare al parametro formale "Disturbo". Se, quando il blocco è richiamato, E 1.4 viene interpretato come parametro attuale, allora l'istruzione eseguita sarà "U E 1.4".

# Dichiarazione dei Parametri Formali

**Parametri Formali** Prima di scrivere il programma per il blocco parametrizzato, bisogna definire i parametri formali nella tabella di dichiarazione.

**Tipi di Parametri** Nella tabella in figura, si possono vedere tre possibili tipi di parametri, ed il loro uso. Assicurarsi che quando si debba avere accesso sia in lettura che in scrittura ad un parametro formale, questo venga dichiarato del tipo IN/OUT

| Tipo di parametro     | Dichiarazione | Uso               | Visualizzazione grafica  |
|-----------------------|---------------|-------------------|--------------------------|
| Parametro di ingresso | <b>in</b>     | Sola lettura      | Alla sinistra del blocco |
| Parametro di uscita   | <b>out</b>    | Sola scrittura    | Alla destra del blocco   |
| Parametro in/out      | <b>In_out</b> | Lettura/scrittura | Alla sinistra del blocco |

**N.B.** C'è solo una riga per ogni tipo di parametri nella tabella di dichiarazione. Per introdurre ulteriori parametri si usi il tasto "Invio" una volta che la prima riga sia completata. In questo modo viene aperta una nuova riga per un parametro dello stesso tipo di quello appena dichiarato.

**Attenzione!** Se si vuole inserire o cancellare una riga nella tabella di dichiarazione in un secondo tempo - dopo che il blocco sia stato richiamato, è necessario riscrivere il richiamo del blocco stesso!

## Esempio di dichiarazione dei Parametri Formali

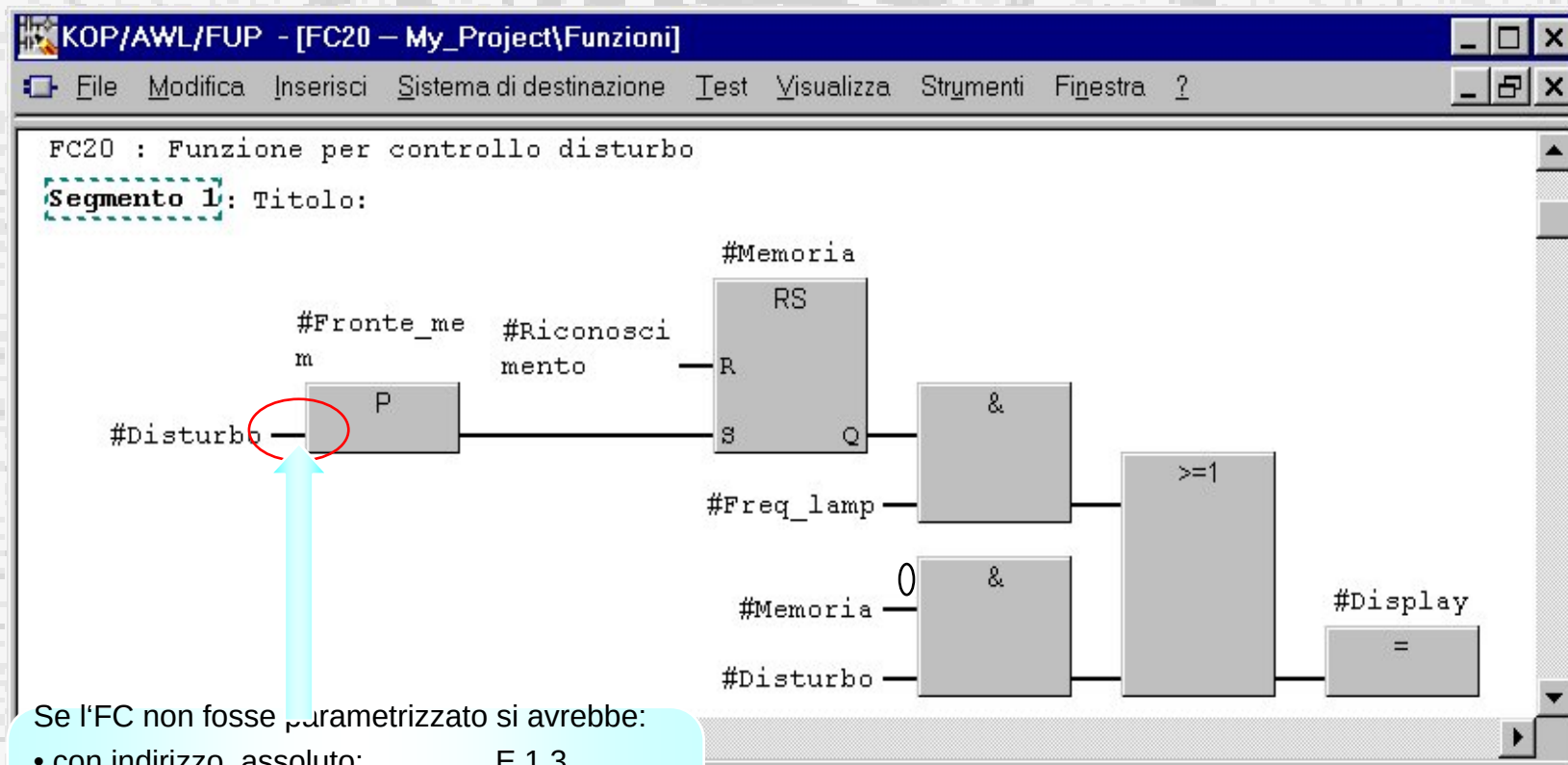
**Esempio: FC20** Nella parte bassa della figura, si vede la tabella di dichiarazione per il visualizzatore di disturbi Poiché la memoria di segnalazione deve essere utilizzata con accesso sia in lettura che in scrittura, la si dichiara del tipo IN/OUT.

| Tipo di parametro     | Dichiarazione | Uso               | Visualizzazione grafica  |
|-----------------------|---------------|-------------------|--------------------------|
| Parametro di ingresso | <b>in</b>     | Sola lettura      | Alla sinistra del blocco |
| Parametro di uscita   | <b>out</b>    | Sola scrittura    | Alla destra del blocco   |
| Parametro in/out      | <b>In_out</b> | Lettura/scrittura | Alla sinistra del blocco |

Tabella di dichiarazione dell' FC 20

| Indirizzo | Dichiarazione | Nome           | Tipo | Valore iniziale | Commento |
|-----------|---------------|----------------|------|-----------------|----------|
| 0.0       | in            | Disturbo       | BOOL |                 |          |
| 0.1       | in            | Riconoscimento | BOOL |                 |          |
| 0.2       | in            | Freq_lamp      | BOOL |                 |          |
| 2.0       | out           | Fronte_mem     | BOOL |                 |          |
| 2.1       | out           | Display        | BOOL |                 |          |
| 4.0       | in_out        | Memoria        | BOOL |                 |          |

# Editazione di un Blocco Parametrizzato



Se l'FC non fosse parametrizzato si avrebbe:

- con indirizzo assoluto: E 1.3
- con indirizzo simbolico: "Sensore\_Sx"

Non è importante che il nome di un parametro formale sia scritto in lettere maiuscole o minuscole. Il carattere "#" davanti al nome viene inserito automaticamente dal PG. Questo serve ad indicare che si sta utilizzando una variabile locale definita nella tabella di dichiarazione del blocco stesso.

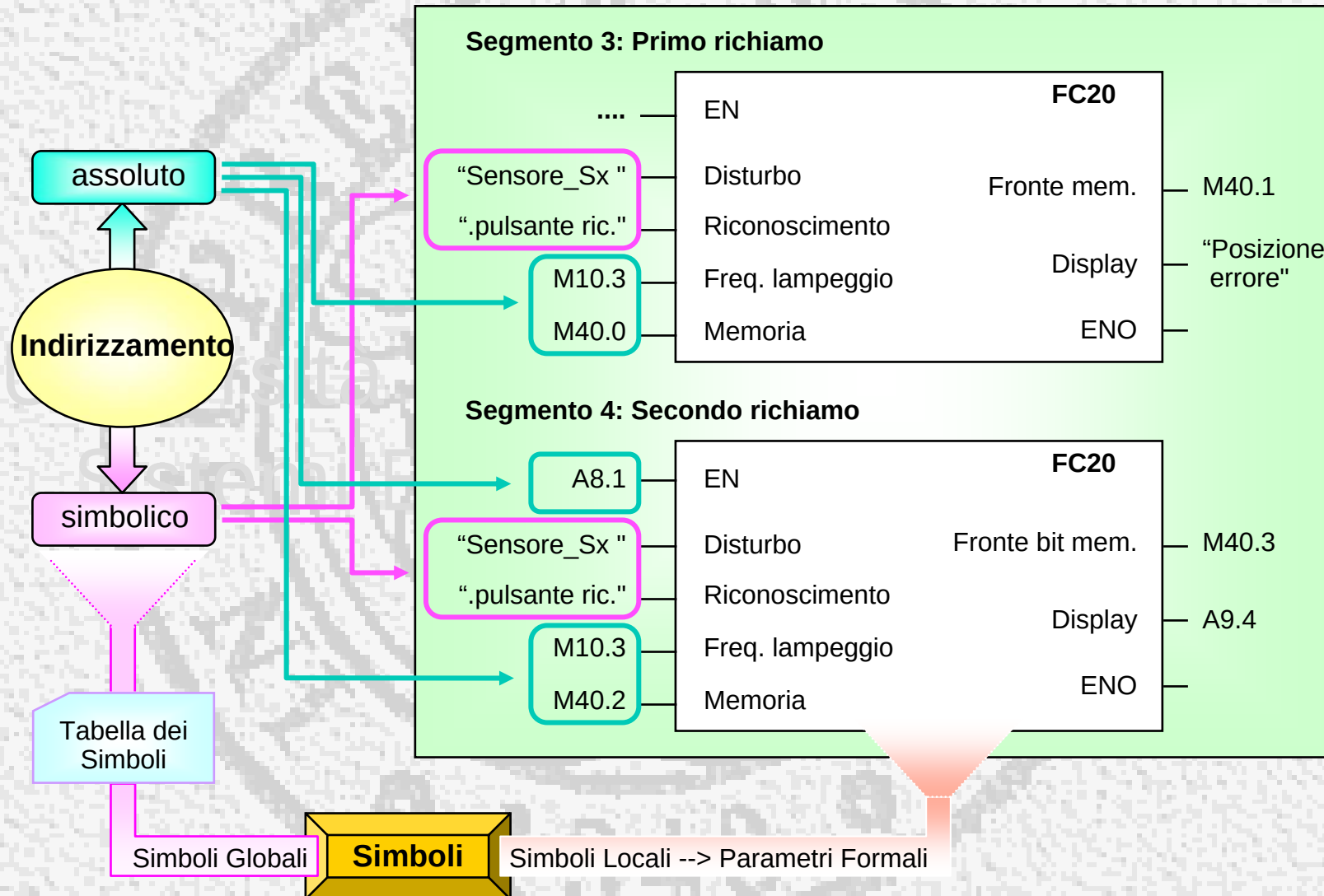
## Richiamo di un Blocco Parametrizzato

**Richiamo** In KOP/FUP, si può selezionare il blocco da richiamare nel catalogo "Elementi di Programma". Verranno visualizzati dei punti di domanda (???) in corrispondenza dei parametri formali di tipo IN, OUT e IN/OUT del blocco. Qui vanno inseriti i parametri attuali voluti.

**N.B.**

Quando si richiama un FC parametrizzato, bisogna fare una assegnazione per tutti i parametri dichiarati (con l'eccezione di EN e ENO).

# Richiamo di un Blocco Parametrizzato



## Esercizio: Creazione di un Blocco FC Parametrizzato

### 1. Tabella di Dichiarazione dell' FC 20

| Indirizzo | Dichiarazione | Nome           | Tipo | Valore iniziale | Commento |
|-----------|---------------|----------------|------|-----------------|----------|
| 0.0       | in            | Disturbo       | BOOL |                 |          |
| 0.1       | in            | Riconoscimento | BOOL |                 |          |
| 0.2       | in            | Freq_lamp      | BOOL |                 |          |
| 2.0       | out           | Fronte_mem     | BOOL |                 |          |
| 2.1       | out           | Display        | BOOL |                 |          |
| 4.0       | in_out        | Memoria        | BOOL |                 |          |

**Blocco  
non parametrizzato**

```

U      E 1.2
R      M 40.0
U      E 1.3
FP     M 40.1
S      M 40.0
U      M 40.0
U      M 10.3
O
UN     M 40.0
U      E 1.3
=      A 9.3(A5.3)
  
```

**Blocco FC 20  
Parametrizzato**

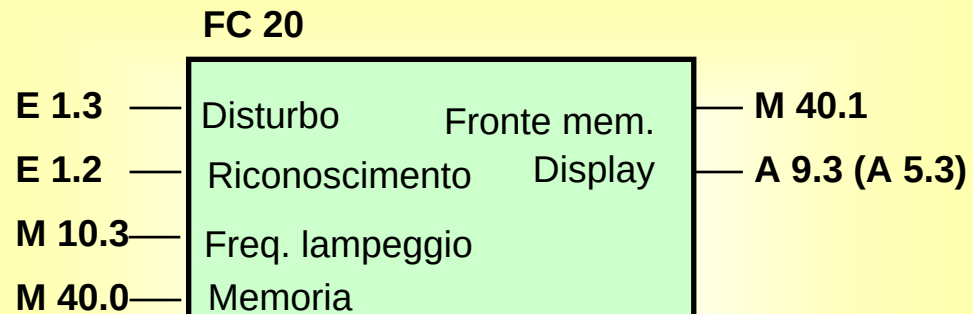
```

U#Riconoscimento
R#Memoria
U#Disturbo ...
:
::
:
:
:
:
:
:
:
  
```

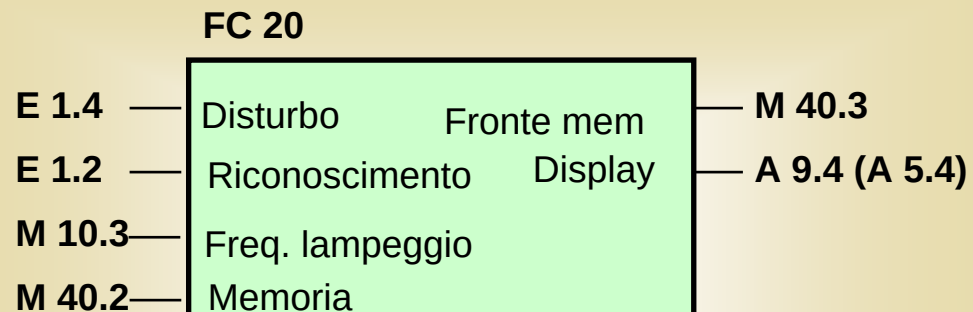
2.

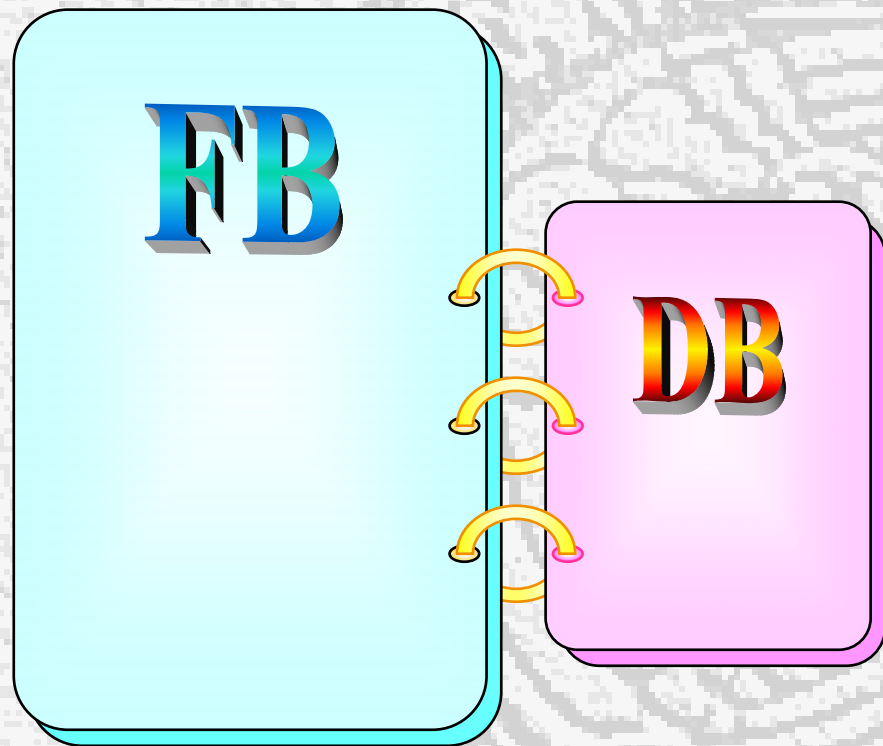
## Esercizio: richiamo di un blocco parametrizzato

Parametrizzazione FC20  
nel primo richiamo



Parametrizzazione FC20  
nel secondo richiamo





Creazione di un  
Blocco di  
Funzionale (FB)

## Blocchi Funzionali (FB)

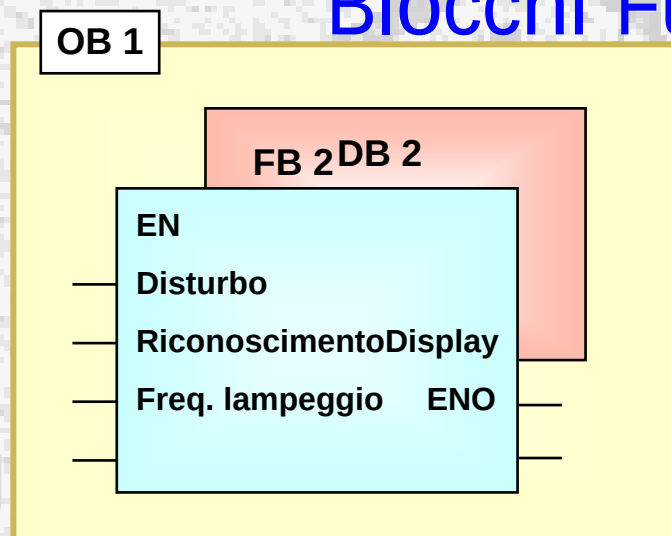


Tabella di dichiarazione del blocco funzionale

| Indirizzo | Dichiarazione | Nome           | Tipo | Valore iniziale | Commento |
|-----------|---------------|----------------|------|-----------------|----------|
| 0.0       | in            | Disturbo       | BOOL | FALSE           |          |
| 0.1       | in            | Riconoscimento | BOOL | FALSE           |          |
| 0.2       | in            | Freq_lamp      | BOOL | FALSE           |          |
| 2.0       | out           | Display        | BOOL | FALSE           |          |
|           | in_out        |                |      |                 |          |
| 4.0       | stat          | fronte_mem     | BOOL | FALSE           |          |
| 4.1       | stat          | memoria        | BOOL | FALSE           |          |
|           | temp          |                |      |                 |          |

## Blocchi Funzionali (FB)

### Caratteristiche speciali degli FB

A differenza dei blocchi funzione (FC), i blocchi funzionali (FB) possiedono una memoria (richiamata). Questo significa che viene assegnato un DB locale al blocco funzionale, il cosiddetto **blocco dati di istanza**.

Quando si richiama un FB, bisogna anche specificare il numero del DB di istanza, il quale viene aperto automaticamente.

Un DB di istanza viene usato per salvare le **variabili statiche**. Queste variabili locali possono essere usate solo nell' FB nella tabella di dichiarazione del quale siano state dichiarate. All'uscita dal blocco, perciò, esse sono mantenute.

### Parametri

Quando viene richiamato il blocco funzionale, il valore dei parametri attuali viene salvato nel DB di istanza.

Se non vengono assegnati parametri attuali ai parametri formali, allora viene usato dal programma in esecuzione l'ultimo valore memorizzato nel DB di istanza per quei parametri.

Si possono specificare differenti parametri attuali con ogni richiamo di FB. All'uscita dall'FB, vengono ritenuti i dati nel DB di istanza.

## Blocchi Funzionali (FB)

### Vantaggi degli FB

- Quando si scrive un programma per un FC, bisogna cercare bit di memoria inutilizzati nell'area indirizzi, oppure aree di dati, e bisogna ritenerli. Le variabili statiche di un FB, d'altra parte, vengono ritenute direttamente dal software STEP 7.
- Quando si usano variabili statiche, si evita il rischio di assegnare due volte bit di memoria dell'area indirizzi, oppure di aree dati.
- Invece dei parametri "Memoria" e "Fronte\_mem" nell' FC20, si possono usare le variabili statiche "Memoria " e "Fronte\_mem" nell'FB. Questo rende il richiamo del blocco più semplice, poiché due parametri formali vengono eliminati.

## Esempio di Blocco Funzionale per la visualizzazione del disturbo

| Indirizzo | Dichiarazione | Nome           | Tipo | Valore iniziale | Commento |
|-----------|---------------|----------------|------|-----------------|----------|
| 0.0       | in            | Disturbo       | BOOL | FALSE           |          |
| 0.1       | in            | Riconoscimento | BOOL | FALSE           |          |
| 0.2       | in            | Freq_lamp      | BOOL | FALSE           |          |
| 2.0       | out           | Display        | BOOL | FALSE           |          |
|           | in_out        |                |      |                 |          |
| 4.0       | stat          | fronte_mem     | BOOL | FALSE           |          |
| 4.1       | stat          | memoria        | BOOL | FALSE           |          |
|           | temn          |                |      |                 |          |

Tabella di dichiarazione del blocco funzionale



KOP/AWL/FUP - [DB2 - My\_Project\Funzioni]

| Indirizzo | Dichiarazione | Nome           | Tipo | Valore iniziale | Commento |
|-----------|---------------|----------------|------|-----------------|----------|
| 0.0       | in            | Disturbo       | BOOL | FALSE           |          |
| 0.1       | in            | Riconoscimento | BOOL | FALSE           |          |
| 0.2       | in            | Freq_lamp      | BOOL | FALSE           |          |
| 2.0       | out           | Display        | BOOL | FALSE           |          |
| 4.0       | stat          | fronte_mem     | BOOL | FALSE           |          |
| 4.1       | stat          | memoria        | BOOL | FALSE           |          |

Blocco dati di istanza

# Creazione di un blocco di funzionale (FB)

Dal Simatic Manager si può definire all'interno della cartella Blocchi un nuovo blocco funzionale (FB)

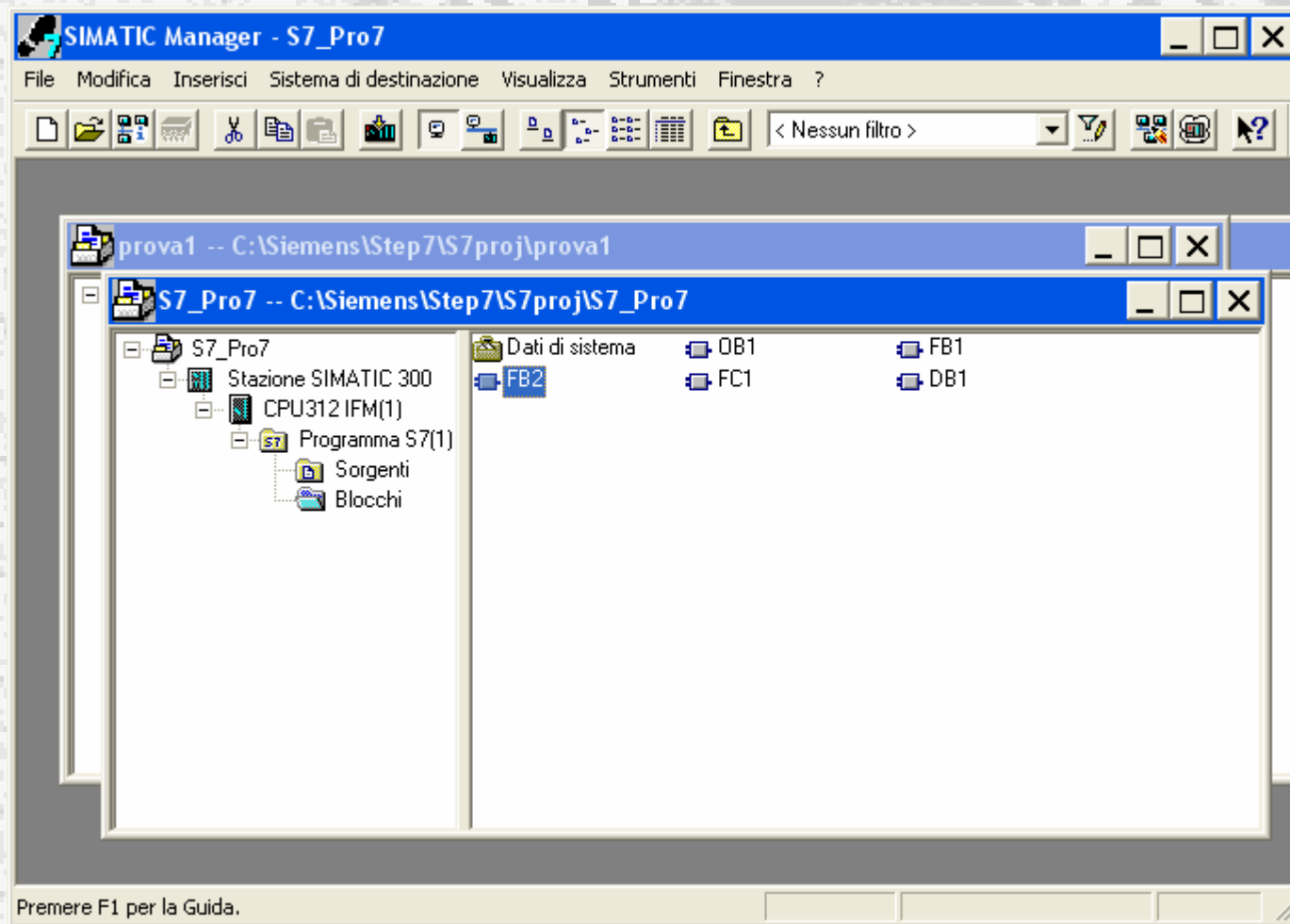
Proprietà - Blocco funzionale

| Generale - Parte 1   |                                 | Generale - Parte 2                                   | Richiami | Attributi           |
|----------------------|---------------------------------|------------------------------------------------------|----------|---------------------|
| Nome:                | FB2                             | <input checked="" type="checkbox"/> Con multiistanza |          |                     |
| Nome simbolico:      |                                 |                                                      |          |                     |
| Commento al simbolo: |                                 |                                                      |          |                     |
| Linguaggio:          | KOP                             |                                                      |          |                     |
| Percorso progetto:   |                                 |                                                      |          |                     |
| Percorso fisico:     | C:\Siemens\Step7\S7proj\S7_Pro7 |                                                      |          |                     |
| Creato il:           | 21/03/2006 17.05.42             |                                                      |          |                     |
| Ultima modifica:     | 21/03/2006 17.05.42             |                                                      |          | 21/03/2006 17.05.42 |
| Commento:            |                                 |                                                      |          |                     |

OK Annulla ?

Inserisce Blocco funzionale nella posizione del cursore.

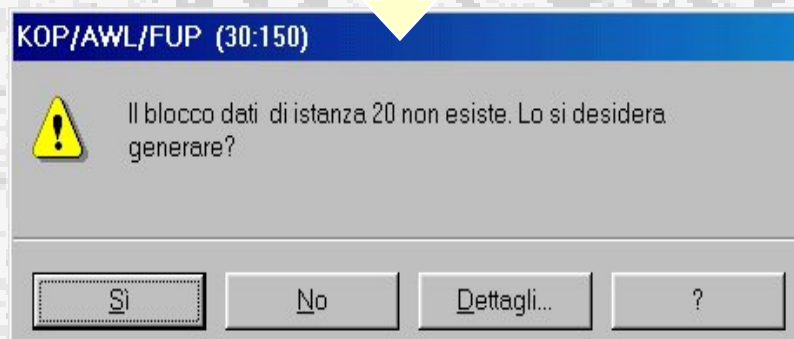
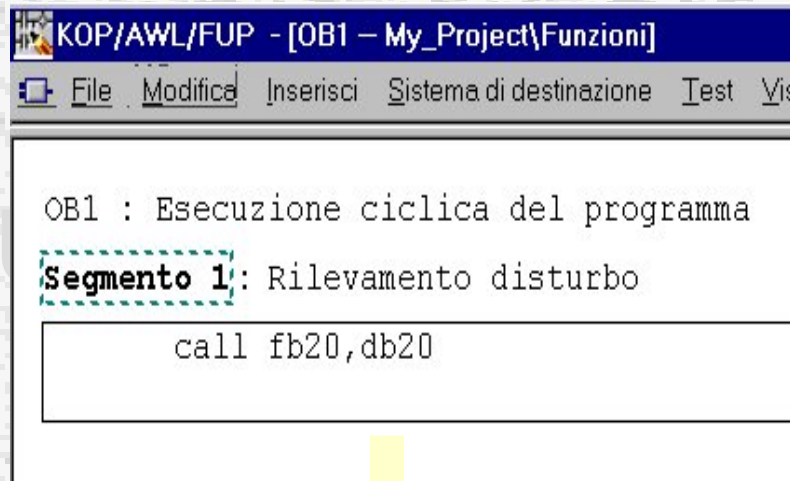
# Creazione di un blocco di programma (FB)



Nella cartella blocchi del progetto appare il simbolo di un blocco FB (per default viene assegnato un nome con un numero progressivo).

# Generazione di un blocco dati di istanza

## 1. Generazione DB durante richiamo di FB



## 2. Creazione nuovo DB di istanza



# Generazione di un blocco dati di istanza

## Generazione di un DB di istanza

Ci sono due modi per generare un DB di istanza:

- Quando si richiama un FB, si può specificare con quali DB di istanza si desidera lavorare. Appare allora il seguente messaggio:

“Il DB x non esiste. Si vuole generarlo ora?”.

- Quando si crea un nuovo DB, si selezioni l'opzione “Blocco dati associato a un blocco funzionale”.

### **N.B.**

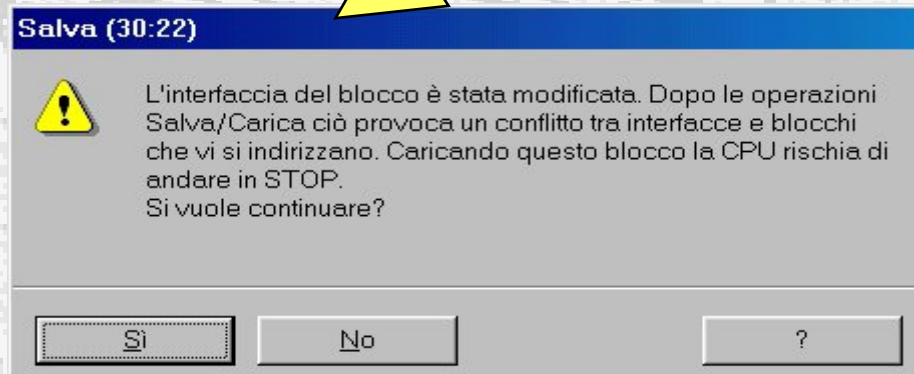
Un DB di istanza può essere associato ad un solo FB. Invece un FB può essere associato a più DB di istanza, diversi per ogni suo richiamo.

Se si modifica un FB (aggiunta di variabile statiche o di di parametri), bisogna rigenerare il DB di istanza.

# Inserzione/Cancellazione parametri in un secondo tempo



| Indirizzo | Dichiarazione | Nome           | Tipo | Valore inizi. | Commento |
|-----------|---------------|----------------|------|---------------|----------|
| 0.0       | in            | Disturbo       | BOOL | FALSE         |          |
| 0.1       | in            | Riconoscimento | BOOL | FALSE         |          |
| 0.2       | in            | Freq_lamp      | BOOL | FALSE         |          |
| 0.3       | in            | Controllo_led  | BOOL | FALSE         |          |
| 2.0       | out           | Display        | BOOL | FALSE         |          |
|           | in_out        |                |      |               |          |
| 4.0       | stat          | fronte_mem     | BOOL | FALSE         |          |
| 4.1       | stat          | memoria        | BOOL | FALSE         |          |
|           | temp          |                |      |               |          |

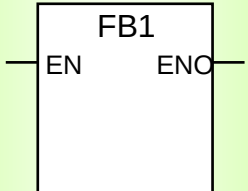
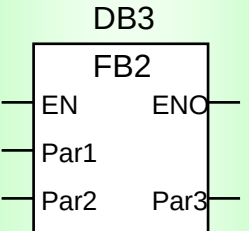
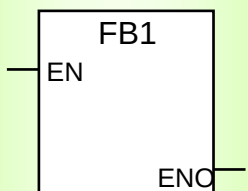
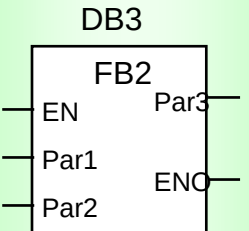


**Problema** Quando si aggiungano ulteriori parametri ad un blocco che sia già richiamato all'interno del programma, bisogna riscrivere il richiamo del blocco stesso.

In caso contrario la CPU potrebbe commutare in STOP oppure il blocco funzionale potrebbe non mantenere la sua funzionalità a causa dei parametri non ancora assegnati durante il richiamo.

Nell'esempio, è stato inserito il nuovo parametro di ingresso "Controllo\_led".

## Richiami di un blocco FB

| Lin-<br>guaggio | FB                                                                                  |                                                                                                             |
|-----------------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------|
|                 | Senza param. e DB istanz.                                                           | Con param. E DB istanz.                                                                                     |
| <b>AWL</b>      | <ul style="list-style-type: none"> <li>• UC FB1</li> <li>• CC FB1</li> </ul>        | <ul style="list-style-type: none"> <li>• CALL FB2, DB3<br/>Par1: ...<br/>Par2: ...<br/>Par3: ...</li> </ul> |
| <b>KOP</b>      |    |                           |
| <b>FUP</b>      |  |                         |

**CALL** L'istruzione "CALL" viene usata per richiamare blocchi di programma indipendentemente dal valore dell'RLC o da altre condizioni.

Si può usare indifferentemente il nome assoluto o simbolico del blocco richiamato.

Occorre indicare il blocco dati associato.

**UC** L'istruzione "UC" è un richiamo incondizionato di un blocco di tipo FC o FB, senza assegnazione parametrica. "UC" è per gli altri aspetti identico a "CALL".

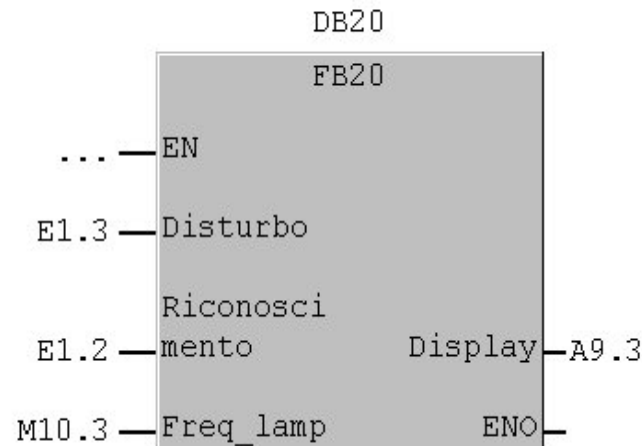
**CC** L'istruzione "CC" richiama un blocco di tipo FC o FB senza parametri se l'RLC=1. "CC" è per gli altri aspetti identico a "CALL".

**Parametri** I parametri dichiarati nella tabella di dichiarazione sono conosciuti come "parametri formali". Gli indirizzi o i valori specificati durante il richiamo, vanno intesi riferiti ai "parametri attuali".

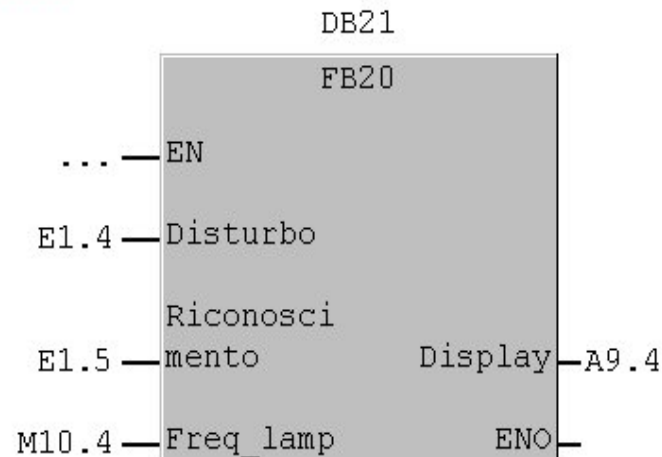
Le variabili statiche e temporanee non vanno specificate durante il richiamo.

# Esercizio: Richiamo e Test di un Blocco

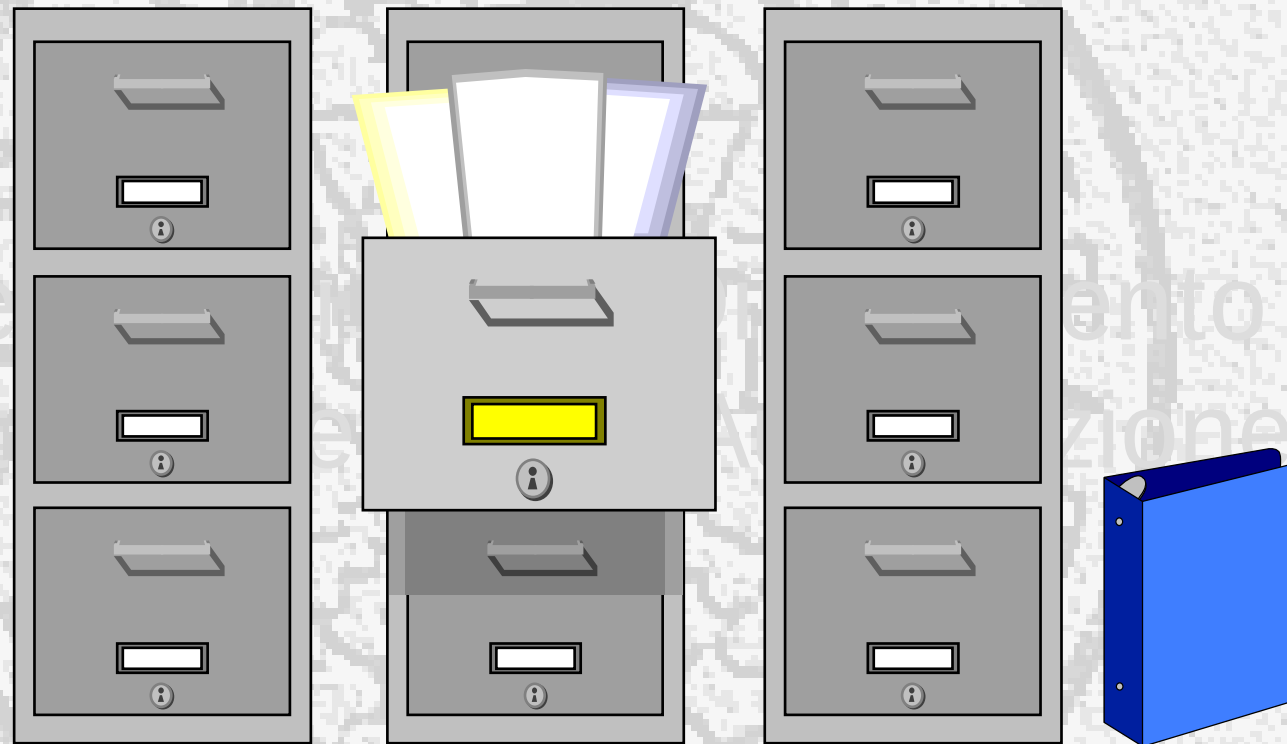
**Segmento 1:** 1° richiamo di FB20



**Segmento 2:** 2° richiamo di FB20



## Gestione dei dati nei Blocchi Dati



# Gestione dei dati nei Blocchi Dati

## Generale

Oltre che di blocchi di programma, un programma consiste anche di Dati contenenti informazioni riguardanti lo stato di processo, dei segnali, ecc., le quali verranno elaborate conformemente alle istruzioni del programma stesso.

I dati sono contenuti in variabili del programma, identificate univocamente da:

- Indirizzo in un'area di memoria (p.e. P, IPI, IPU, memorie, L stack, DB)
- Tipi Dati (tipi dati semplici e complessi, tipi Parametri)

A seconda della possibilità di accesso, si possono distinguere anche tra:

- Variabili Globali, le quali vengono dichiarate nella tabella globale dei simboli o in un blocco dati globale
- Variabili Locali, le quali vengono dichiarate nella parte di dichiarazione di OB, FB ed FC.

Le variabili possono avere una locazione permanente nell'immagine di processo, nell'area di memoria o in un blocco dati, oppure possono essere create dinamicamente nell' L stack quando un blocco viene eseguito.

# Gestione dei dati nei Blocchi Dati

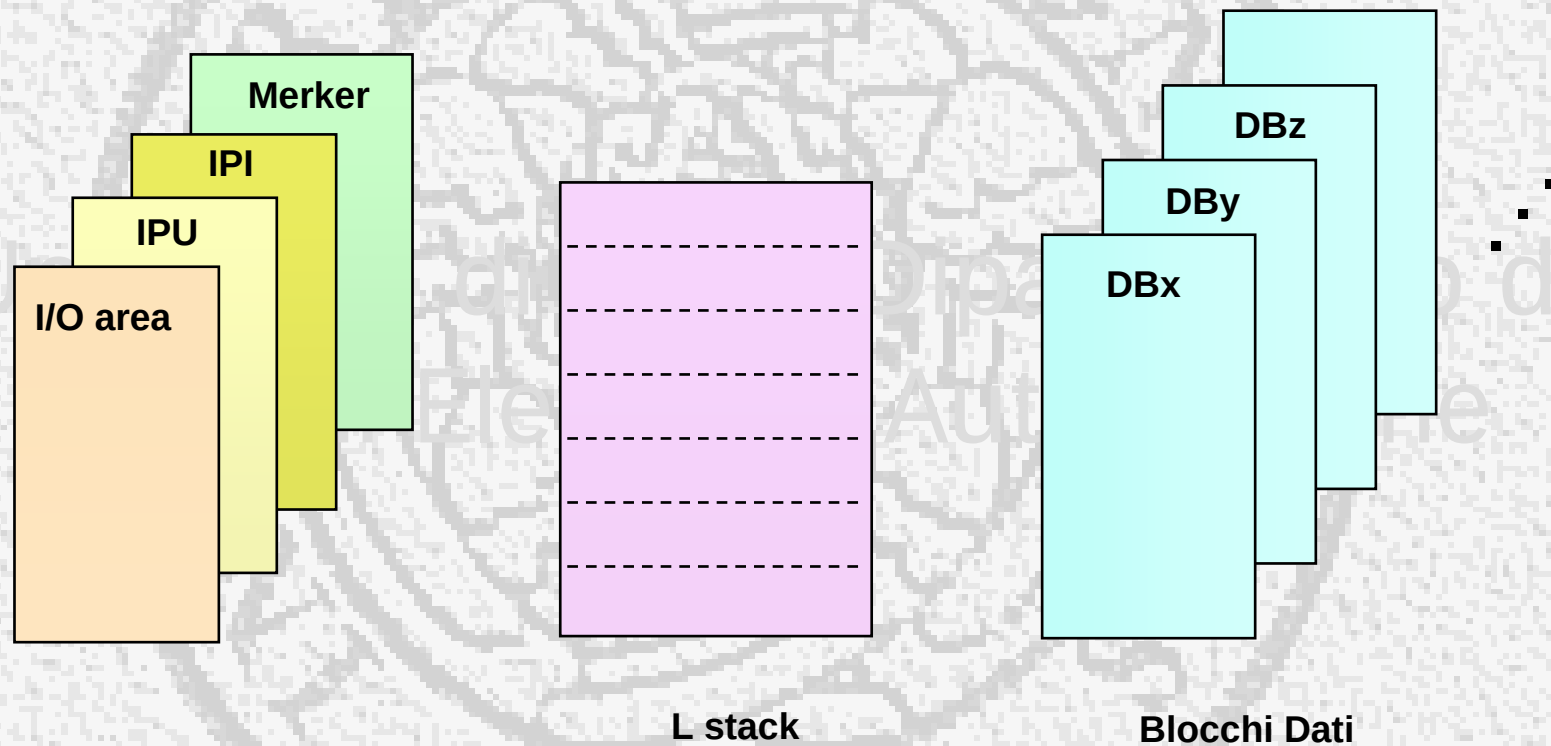
## Registro delle variabili Locali

Il registro delle variabili Locali (Local stack) è un'area di memoria per gestire:

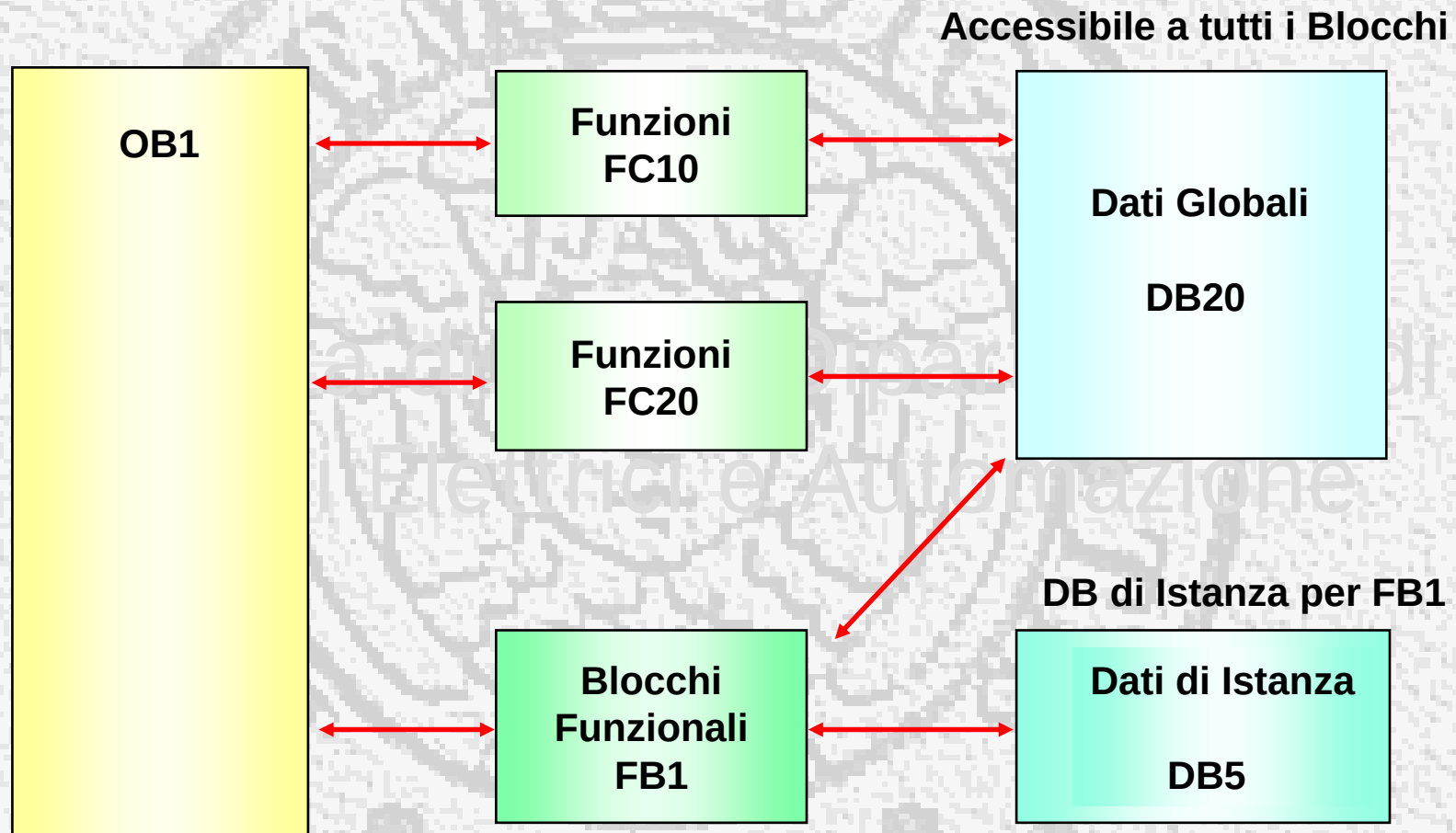
- variabili temporanee di un blocco, incluse le informazioni di start degli OB
- Parametri attuali da assegnare durante il richiamo di FC o FB
- Risultati logici intermedi in programmi scritti in KOP

**Blocchi Dati** I Blocchi Dati vengono utilizzati dai Blocchi logici di un programma per la gestione delle informazioni. A differenza dei dati temporanei, le informazioni in un Blocco Dati non vengono sovrascritte quando l'esecuzione di un blocco logico è terminata o quando il DB stesso viene chiuso.

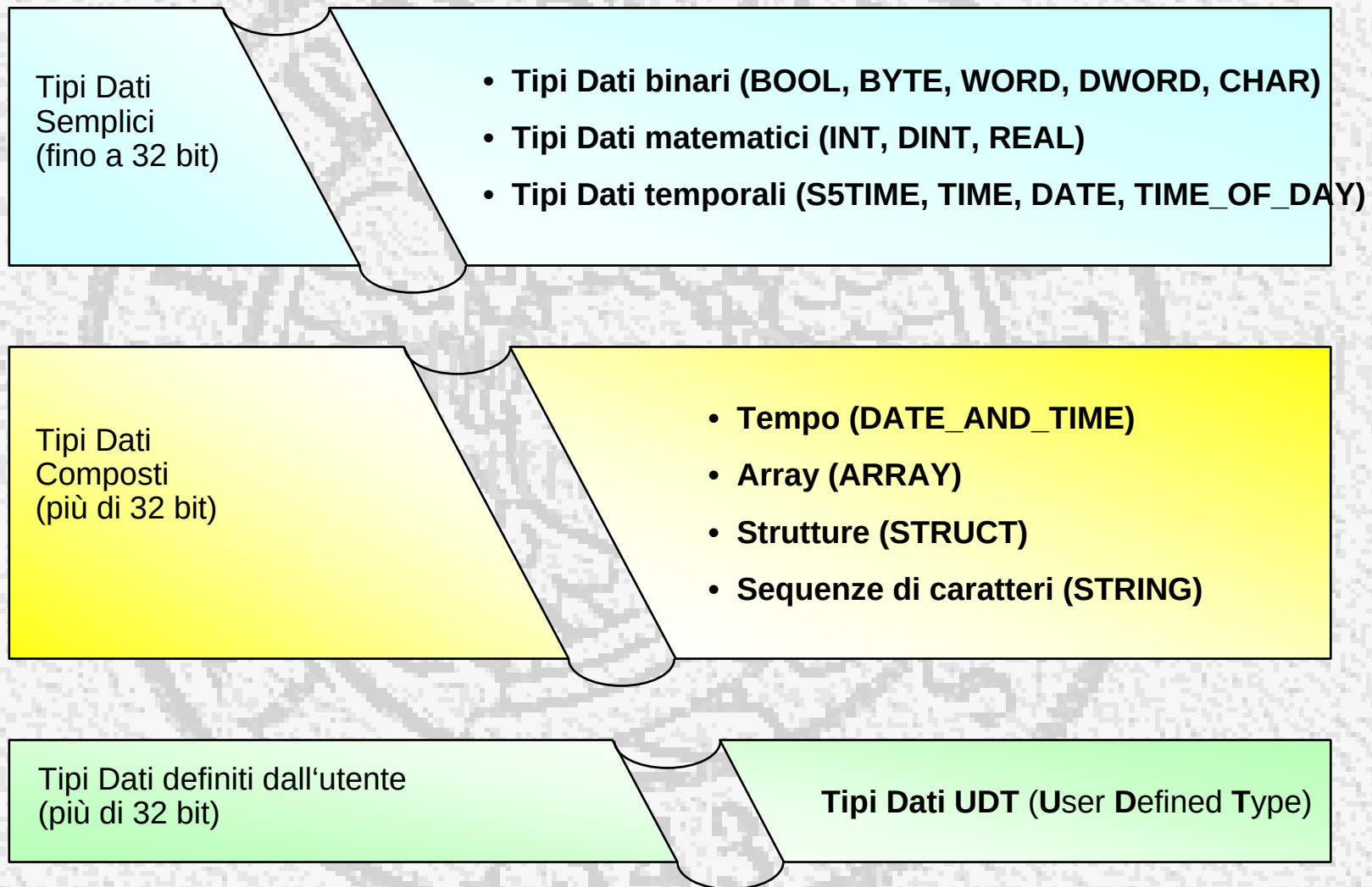
# Aree di memoria per la gestione dei dati



## Blocchi Dati (DB)



## Tipi dei Dati in STEP 7



# Tipi Dati Semplici in STEP 7

## Generale

I Tipi Dati determinano le proprietà dei dati, per esempio permettendo la rappresentazione del contenuto di uno o più indirizzi associati in modo di consentire diverse visualizzazioni e diversi intervalli ammissibili valori.

I tipi dati inoltre determinano il tipo di operazioni con cui è possibile elaborarli.

## Tipi Dati Semplici

Sono predefiniti in accordo con IEC 1131-3. I Tipi Dati determinano la quantità di memoria richiesta. Per esempio, Il Tipo Dati Word occupa 16 bit nella memoria di lavoro. I Tipi Dati semplici non sono mai più lunghi di 32 bit, e possono essere caricati negli accumulatori di un processore S7 ed elaborati con le istruzioni di base di STEP 7.

## Tipi Dati Composti

I tipi dati composti possono essere usati solo tramite variabili dichiarate in Blocchi dati globali. I tipi dati composti non possono essere completamente caricati negli accumulatori con istruzioni di caricamento. Vengono utilizzati dei blocchi standard di libreria ("IEC") per la loro elaborazione.

## Tipi Dati Semplici in STEP 7

### Tipi Dati definiti dall'utente

Un Tipo Dati definito dall'utente può essere usato in blocchi dati o come tipo di dati nella tabella di dichiarazione delle variabili.

Si creano UDT utilizzando l'Editor dei Blocchi Dati.

La struttura di un UDT può contenere gruppi di tipi dati semplici e/o composti.

## Tipi Dati Semplici in STEP 7

| Identificativo                        | Lungh. (in bit)         | Esempio di costante di questo tipo                            |
|---------------------------------------|-------------------------|---------------------------------------------------------------|
| BOOL<br>BYTE<br>WORD<br>DWORD<br>CHAR | 1<br>8<br>16<br>32<br>8 | TRUE o FALSE<br>B#16#A9<br>W#16#12AF<br>DW#16#ADAC1EF5<br>'w' |
| S5TIME                                | 16                      | S5T#5s_200ms                                                  |
| INT<br>DINT<br>REAL                   | 16<br>32<br>32          | 123<br>65539<br>1.2 o 34.5E-12                                |
| TIME<br>DATE<br>TIME_OF_DAY           | 32<br>16<br>32          | T#2D_1H_3M_45S_12MS<br>D#1993-01-20<br>TOD#12:23:45.12        |

## Tipi Dati Complessi

| Identificativo                                                                                     | Lungh. (in bit)              | Esempio                                                                      |                                         |
|----------------------------------------------------------------------------------------------------|------------------------------|------------------------------------------------------------------------------|-----------------------------------------|
| DATE_AND_TIME                                                                                      | 64                           | DT#97-09-24-12:14:55.0                                                       |                                         |
| STRING<br>(stringa di caratteri con max. 254 caratteri)                                            | 8 * (numero di caratteri +2) | ´ Questa è una string<br>´ SIEMENS´                                          |                                         |
| ARRAY<br>(Gruppo di elementi dello stesso Tipi Dati)                                               | defin. dall'utente           | Valori di Misure:    ARRAY[1..20]                                            |                                         |
| STRUCT<br>(Gruppo di elementi di differenti Tipi Dati)                                             | defin. dall'utente           | Motore: STRUCT<br>Velocità : INT<br>Corrente: REAL<br>END_STRUCT             |                                         |
| UDT<br>(User Defined Data Type =<br>"Maschera" consistente di Tipi Dati Semplici oppure complessi) | defin. dall'utente           | UDT come Blocco<br>STRUCT<br>Velocità : INT<br>Corrente : REAL<br>END_STRUCT | UDT come array<br>Drive:<br>ARRAY[1..4] |

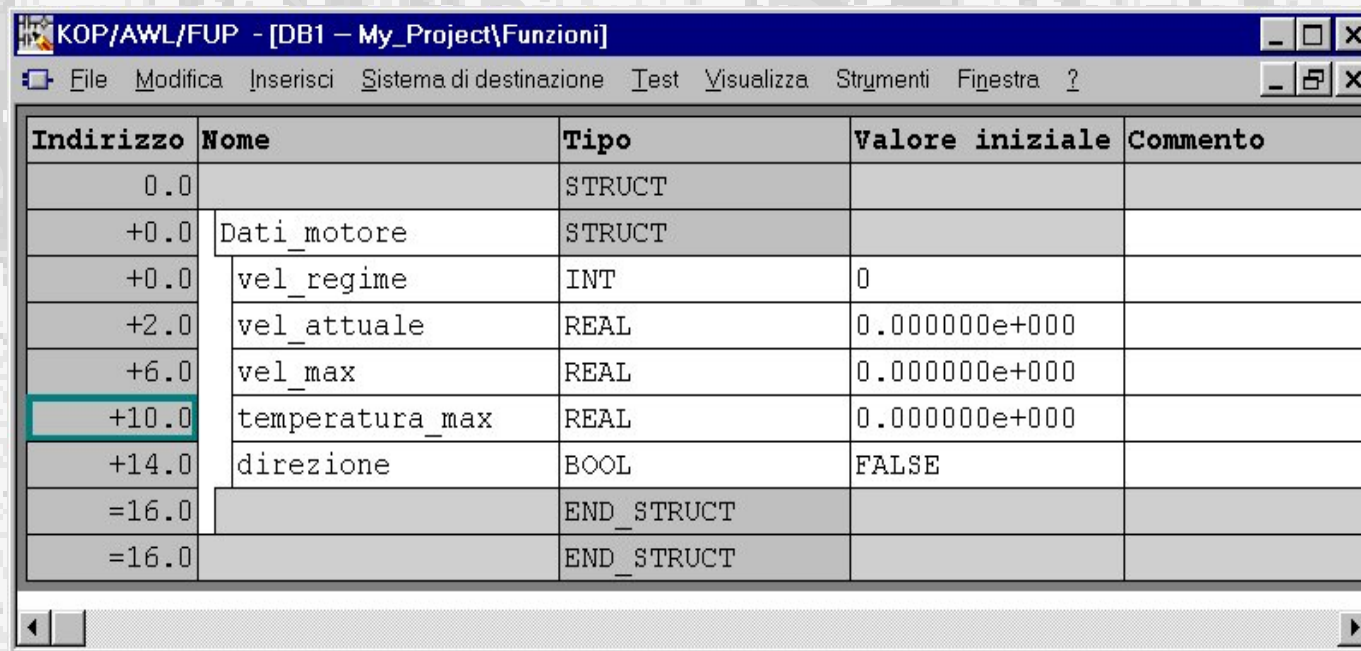
## Esempio di Struttura

### Dati\_motore

|                               |
|-------------------------------|
| Vel_regime, Tipo Dati Integer |
| Vel_attuale, Tipo Dati Real   |
| Vel_max, Tipo Dati Real       |
| Direzione, Tipo Dati Bool     |

Struttura denominata "Dati\_Motore"  
(diversi elementi  
con differenti Tipi Dati)

### Visualizzazione nell' Editor di Programma (Blocco Dati DB 1):



| Indirizzo | Nome            | Tipo       | Valore iniziale | Commento |
|-----------|-----------------|------------|-----------------|----------|
| 0.0       |                 | STRUCT     |                 |          |
| +0.0      | Dati_motore     | STRUCT     |                 |          |
| +0.0      | vel_regime      | INT        | 0               |          |
| +2.0      | vel_attuale     | REAL       | 0.000000e+000   |          |
| +6.0      | vel_max         | REAL       | 0.000000e+000   |          |
| +10.0     | temperatura_max | REAL       | 0.000000e+000   |          |
| +14.0     | direzione       | BOOL       | FALSE           |          |
| =16.0     |                 | END_STRUCT |                 |          |
| =16.0     |                 | END_STRUCT |                 |          |

## Esempio di Struttura

### Strutture

L'immagine mostra un esempio di struttura denominata "Dati\_Motore". Gli elementi della struttura sono dati di tipi differenti. Il singolo elemento di una struttura può essere di tipo semplice o composto.

L'accesso al singolo elemento di una struttura contiene al suo interno il nome della struttura stessa, così che il programma risulta di più facile lettura.

Perché sia possibile accedere simbolicamente agli elementi di una struttura, è necessario che al blocco dati sia stato assegnato a sua volta un nome simbolico globale, per esempio "Drive\_1".

Esempio di accesso individuale all'elemento di una struttura:

L "Drive\_1".Dati\_motore.vel\_regime o

L "Drive\_1".Dati\_motore.vel\_attuale

"Drive\_1" è il nome simbolico del blocco dati che contiene la struttura. Il nome della struttura viene indicato (separato da un punto) dopo il nome simbolico. Segue (separato da un punto) il nome di un elemento della struttura stessa.

### Definizione di una struttura in un DB

La parola chiave per definire una struttura è "STRUCT".

La fine di una struttura è indicata da "END\_STRUCT". La struttura è individuata da un nome (nell'esempio: "Dati\_motore").

## Esempio di Array

**Array** Un array consiste di diversi elementi dello stesso Tipo Dati. Nell'immagine sopra si può vedere l'array "Punto\_di\_misura" composto di 10 elementi del Tipo Dati REAL. In seguito, vari valori di misura potranno essere gestiti in questo array.

**Creazione di un Array** La parola chiave per un array è "ARRAY[n..m]". Il primo (n) e l'ultimo elemento (m) vengono specificati tra parentesi quadre. Nell'esempio, [1..10] significa 10 elementi, dei quali il primo è indirizzato con l'indice [1] e l'ultimo con l'indice [10]. Invece di [1..10] si sarebbe potuto, per esempio, definire [0..9]. Questo influisce solo sull'accesso all'elemento dell'array.

**Nota** Per creare un blocco dati vuoto, basta definire un array con il Tipo Dati voluto.

**Visualizzazione Dati** Per vedere quali valori siano memorizzati nei singoli elementi di un array, si selezioni l'opzione di menu *Visualizzazione -> Dati*. In "visualizzazione Dati", si può trovare il valore memorizzato attualmente nella colonna "Valore Attuale".

## Esempio di Array

### Punto\_di\_misura

- |                                     |
|-------------------------------------|
| 1. Punto_di_misura, Tipo Dati Real  |
| 2. Punto_di_misura, Tipo Dati Real  |
| 3. Punto_di_misura, Tipo Dati Real  |
| •                                   |
| •                                   |
| •                                   |
| 10. Punto_di_misura, Tipo Dati Real |

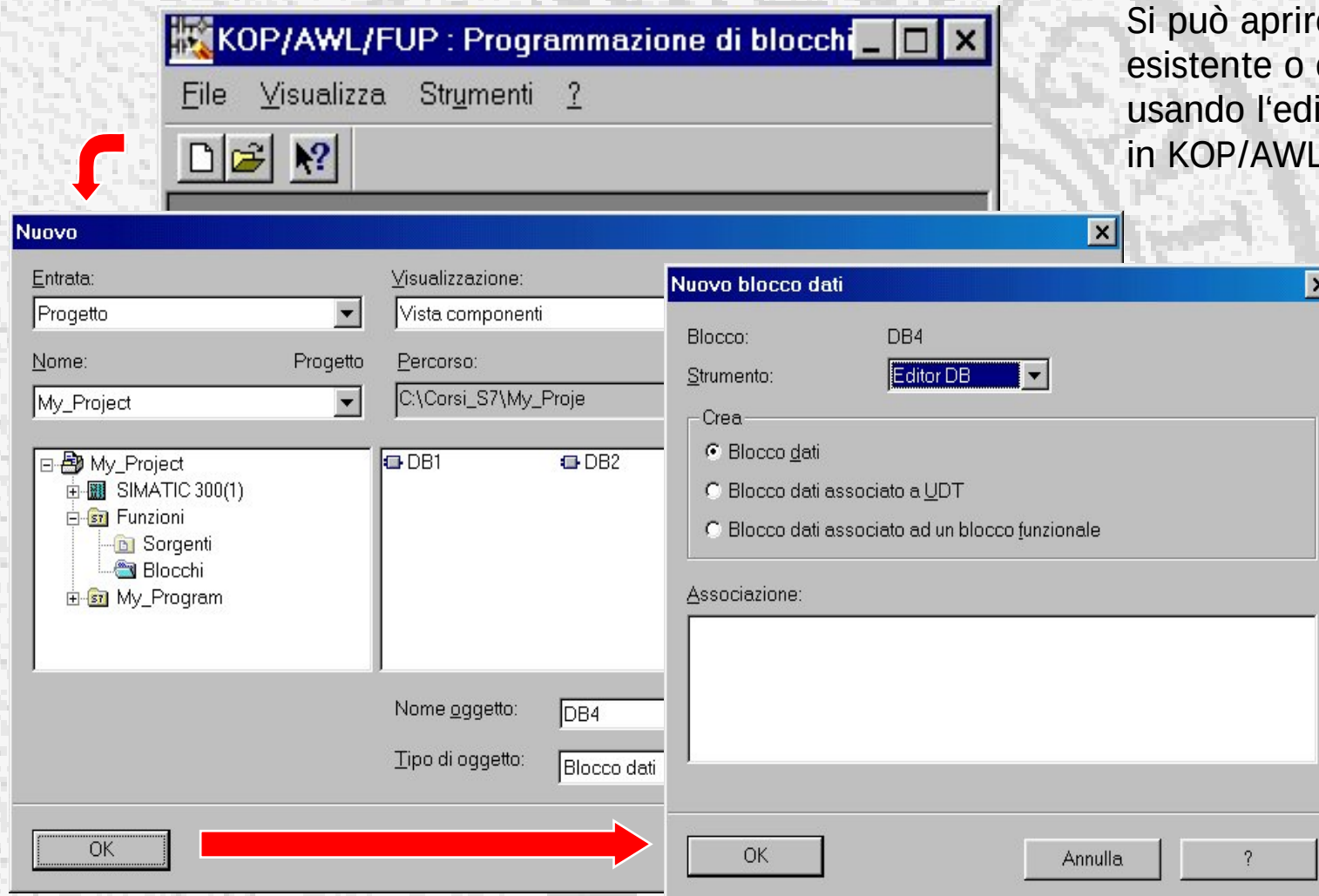
Array denominato "Punto\_di\_misura"  
(diversi elementi  
dello stesso Tipo Dati)

Visualizzazione nell'Editor di Programma (Blocco Dati DB 2):

| Indirizzo | Nome            | Tipo         | Valore iniziale | Commento |
|-----------|-----------------|--------------|-----------------|----------|
| 0.0       |                 | STRUCT       |                 |          |
| +0.0      | punto_di_misura | ARRAY[1..10] |                 |          |
| *4.0      |                 | REAL         |                 |          |
| =40.0     |                 | END_STRUCT   |                 |          |

# Creazione di un Nuovo Blocco Dati

Si può aprire un Blocco Dati esistente o crearne uno nuovo usando l'editor di programma in KOP/AWL/FUP.



# Editazione, Salvataggio, Trasferimento e Visualizzazione di un Blocco Dati

KOP/AWL/FUP - [DB4 - My\_Project\Funzioni]

File Modifica Inserisci Sistema di destinazione Test Visualizza Strumenti Finestra ?

| Indirizzo | Nome              | Tipo         | Valore iniziale | Commento |
|-----------|-------------------|--------------|-----------------|----------|
| 0.0       |                   | STRUCT       |                 |          |
| +0.0      | Numero_pezzi      | INT          | 0               |          |
| +2.0      | Tempo_riempimento | S5TIME       | S5T#3S          |          |
| +4.0      | Liv_min_riemp     | REAL         | 0.000000e+000   |          |
| +8.0      | Punto_di_misura   | ARRAY[1..10] |                 |          |
| +4.0      |                   | REAL         |                 |          |
| =48.0     |                   | END_STRUCT   |                 |          |

# Editazione, Salvataggio, Trasferimento e Visualizzazione di un Blocco Dati

**Introduzione Dati** Si introducono individualmente gli elementi nella tabella. Per farlo, si seleziona la prima linea vuota, e nella colonna "Nome" si inserisca la descrizione dell'elemento. Si può saltare alle altre colonne - Tipo Dati, Valore iniziale e Commento - usando il tasto TAB.

**Colonne** Il significato delle colonne è il seguente:

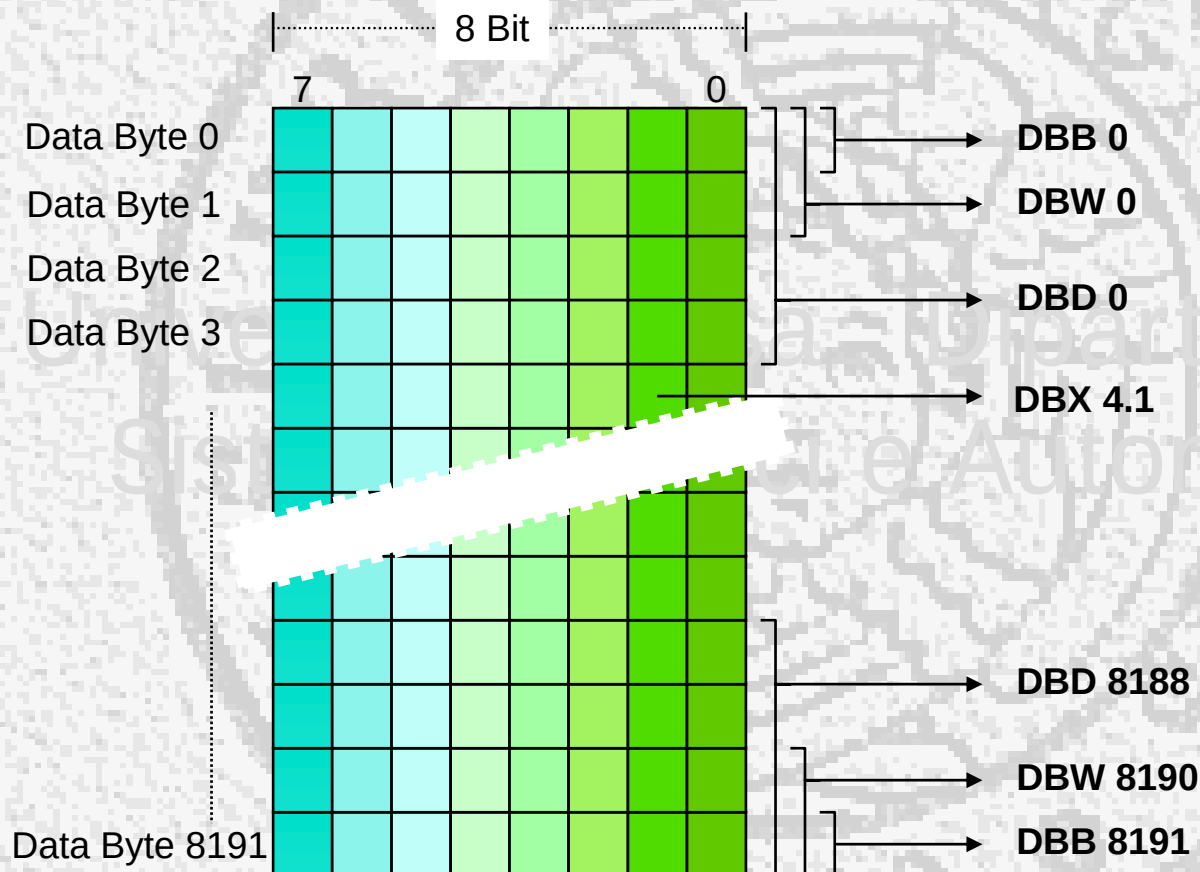
- Indirizzo - E' introdotto dall'Editor di programma quando si salva.  
E' l'indirizzo del primo byte occupato dalla variabile nel blocco dati.
- Nome - Nome simbolico di un elemento.
- Tipo - Tipo Dati (si seleziona con il tasto destro del mouse).
- Valore Iniziale - E' usato per inizializzare un elemento. Se non si digita nulla, verrà inserito zero come valore iniziale.
- Commento - Documentazione degli elementi di una struttura.

**Salvataggio** Si salva il blocco dati su disco fisso nel programma corrente usando l'icona "Dischetto".

**Trasferimento** Come per gli altri blocchi logici, bisogna trasferire il blocco dati nella CPU.

**Visualizzazione** Per visualizzare il valore corrente nel blocco dati, bisogna selezionare la visualizzazione "Dati", e fare clic sull'icona "occhiali" nella barra degli strumenti (visualizzazione permanente dei valori attuali di un DB nella CPU).

# Indirizzamento degli elementi di un Blocco Dati



La massima lunghezza di un blocco dati è di 8KByte per la serie S7-300 e di 64KByte per la serie S7-400.

## Accesso agli elementi di un Blocco Dati

### Apertura di un DB

L'istruzione "**AUF DB...**" apre un blocco dati globale. Se ci fosse un DB globale già aperto, questo verrebbe automaticamente chiuso.

Se il DB fosse definito con un nome simbolico (p.e.: "Valori"), potrebbe essere aperto anche con l'istruzione **AUF "Valori"**.

### Accesso a un DB

Le istruzioni per leggere (Load) o scrivere (Transfer) dati in un DB sono mostrate nell'immagine riportata nel lucido precedente.

Se il DB fosse già aperto, le semplici istruzioni di caricamento e di trasferimento sarebbero sufficienti.

Le istruzioni articolate, del tipo L DB19.DBW2, includono al loro interno l'istruzione di apertura del blocco dati specificato.

# Accesso agli elementi di un Blocco Dati

## Accesso simbolico

Un accesso simbolico è possibile solo se i seguenti requisiti sono soddisfatti:

1. Il DB possiede un nome simbolico nella tabella dei simboli.
2. Lo specifico elemento del blocco dati viene richiamato col suo nome

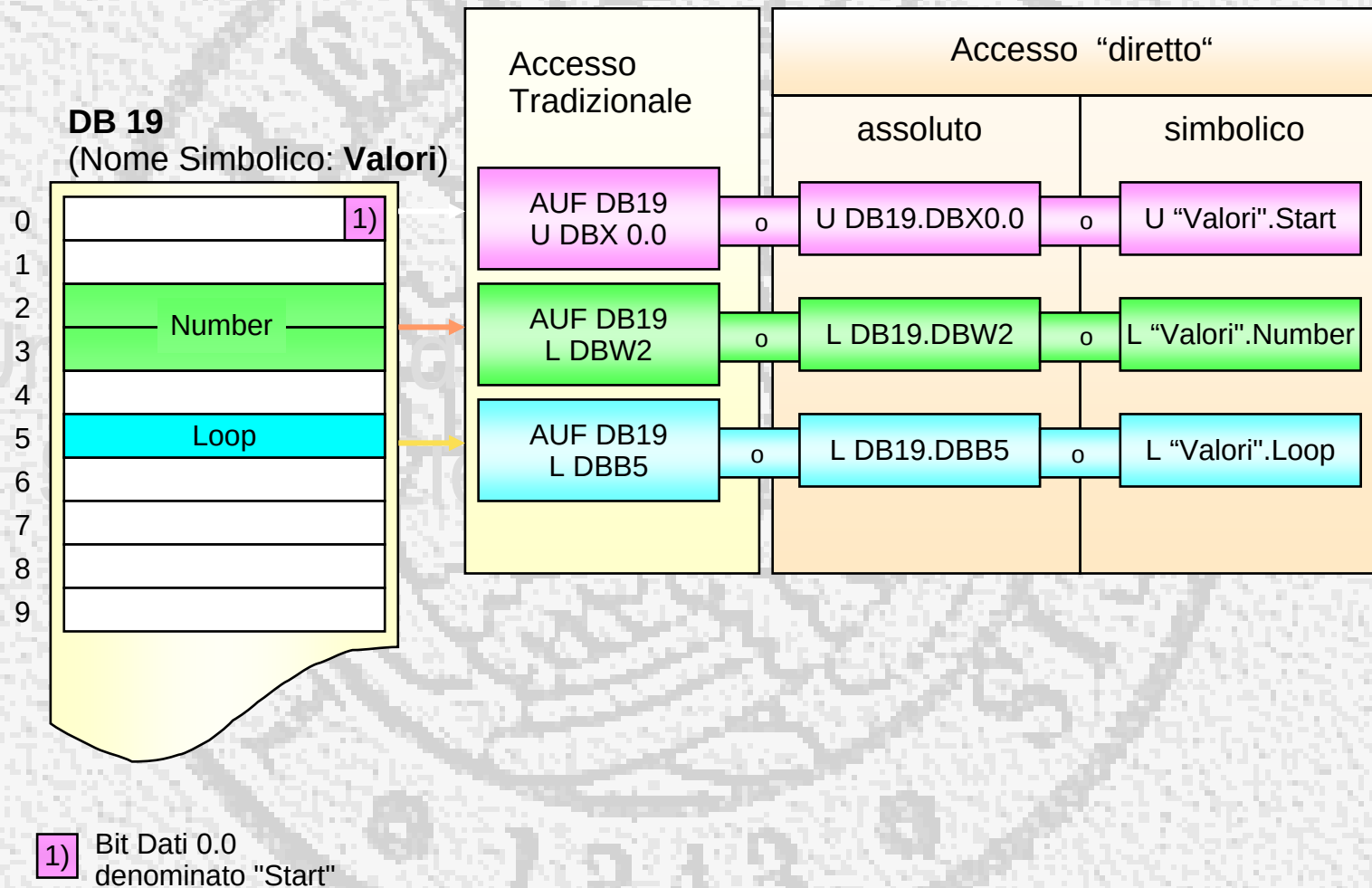
Simbolico nell'editor KOP/AWL/FUP.

Esempio: L'istruzione **L "Valori".Number** apre il DB con il nome 'Valori' e carica l'elemento con nome 'Number' nell'accumulatore.

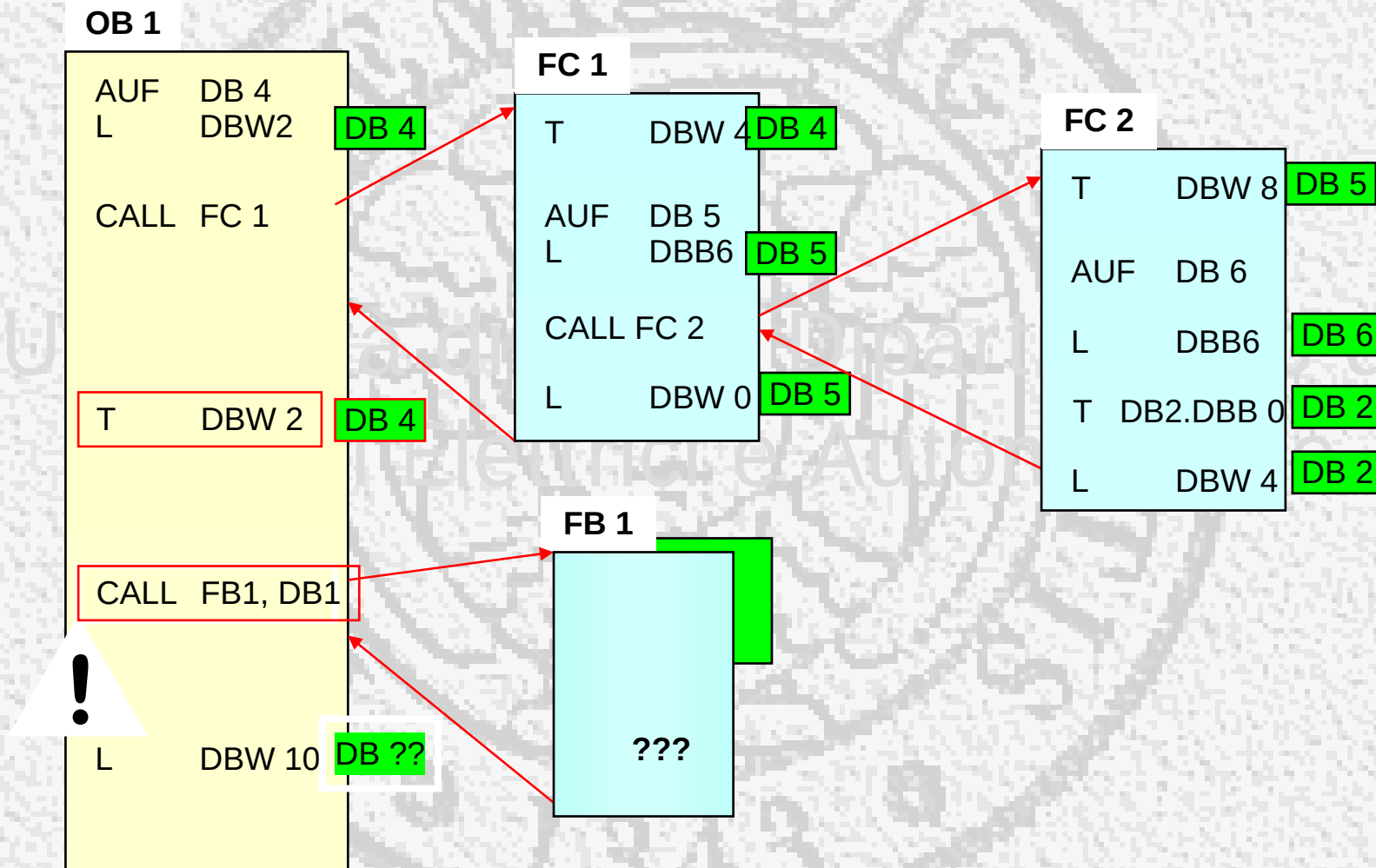
**N.B.** Di norma, si dovrebbe usare l'accesso simbolico ai DB. Questo comporta i seguenti vantaggi:

- il programma è più semplice da leggere,
- è garantito l'accesso al DB corretto,
- è più facile fare correzioni alla struttura dati in un secondo tempo.

# Accesso agli elementi di un Blocco Dati



## Validità di un'apertura di DB



## Validità di un'apertura di DB

**Introduzione** Un blocco dati che sia stato aperto, rimane tale fino a quando non venga aperto un nuovo DB oppure fino a che non si faccia uso di una istruzione composta (p.e. L DB4.DBW6) per accedere ad un altro DB.

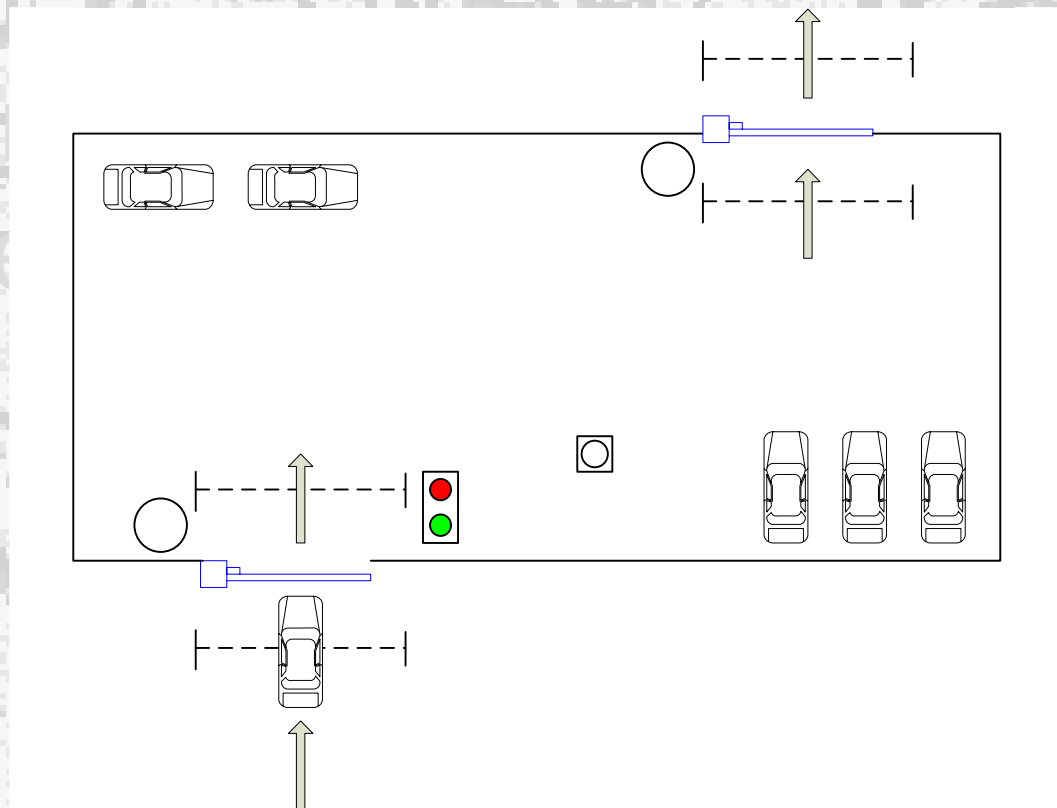
**Richiami di FC** Se un blocco organizzativo o un blocco FC sono terminati a causa del richiamo di un altro blocco FC, l'apertura del blocco dati corrente rimane valida. Al ritorno nel blocchi richiamante, il blocco dati valido al momento al momento dell'uscita viene riaperto.

**Richiami di FB** Il richiamo di un FB è diverso. Un blocco dati di istanza è sempre assegnato ad ogni FB (per ulteriori informazioni si veda il capitolo "Funzioni e Blocchi Funzionali "). Con il richiamo di un FB, viene automaticamente aperto anche il suo DB di istanza. Al ritorno nel blocco richiamante, l'apertura del blocco dati globale aperto precedentemente non è più valida. Questo significa che dopo il richiamo di un FB è necessaria la riapertura del DB globale richiesto.

## Esercizio Parcheggio automatico

### Descrizione

In un garage, l'entrata e l'uscita sono regolati da due sbarre. Ambedue le sbarre vengono chiuse e aperte tramite un motore (rispettivamente M1 e M2). Prima e immediatamente dopo ogni sbarra è installato un fotosensore (LS1.....LS4).



## Esercizio Parcheggio automatico

### Entrata

Se viene attivato il fotosensore LS1, la sbarra dell'entrata viene aperta con il motore M1. Se il mezzo ha superato il fotosensore LS2, il motore M1 richiude la sbarra dell'entrata. Contemporaneamente il contatore viene incrementato di una unità. Se nel garage sono presenti 100 mezzi la sbarra dell'entrata viene bloccata e il semaforo all'entrata diventa 'rosso'. Per il corretto comando del motore M1, la posizione della sbarra viene rivelata da due finecorsa S2 (sbarra aperta) e S3 (sbarra chiusa).

## Esercizio Parcheggio automatico

### Uscita

La sbarra dell'uscita viene comandata, analogamente a quella dell'entrata, tramite i fotosensori LS3 e LS4. quando il mezzo supera il fotosensore LS4, il contatore viene decrementato di una unità; viene così gestita in modo automatico l'occupazione del garage. Per il corretto comando del motore M2, la posizione della sbarra viene rivelata tramite i finecorsa S4 (sbarra aperta) e S5 (sbarra chiusa).

Con il pulsante S1 viene impostata la condizione iniziale (garage vuoto: contatore = 0).