

## L'ambiente di sviluppo STEP 7

STEP 7 è il pacchetto software per sviluppare *progetti* di automazione basati sui prodotti Siemens della serie SIMATIC [1].

I *progetti* sono costituiti dall'insieme dei dati e dei programmi di una soluzione di automazione. In particolare un progetto contiene:

- dati di configurazione relativi alla struttura hardware e dati di parametrizzazione per le unità;
- dati progettazione per le comunicazioni di rete;
- programmi per le unità programmabili.

Il sistema di sviluppo è composto da un software di base e da un numero di componenti aggiuntivi, eseguibili su piattaforme Windows.

Con il pacchetto base di STEP 7 è possibile:

- creare e gestire progetti;
- configurare e parametrizzare l'hardware e le comunicazioni dei sistemi di automazione;
- creare programmi per i PLC delle serie SIMATIC S7 (per sviluppare software per i PLC della serie M7 è necessario installare un componente aggiuntivo);
- caricare i programmi nei sistemi di destinazione;
- validare il software sviluppato.

Nelle pagine successive verrà presentata la versione 5 di questo prodotto e verrà fatta una breve descrizione dei linguaggi di programmazione messi a disposizione dell'utente (il linguaggio a contatti KOP, la lista di istruzioni AWL, l'SCL, ecc.). Per approfondimenti si rimanda alla documentazione della casa produttrice [2-8].

## 1 - Creazione di un nuovo progetto e configurazione hardware

Dopo aver installato STEP 7 sul proprio PC, nel menù Avvio di Windows comparirà la cartella SIMATIC. Al suo interno si trovano le licenze, i manuali e le varie applicazioni di STEP 7 installate.

Le applicazioni disponibili nel pacchetto base di STEP 7 sono:



Figura 1 - Step 7, software di base

SIMATIC Manager consente di creare e gestire i dati di un progetto d'automazione; le applicazioni necessarie per la modifica di un tipo di dati vengono avviate automaticamente da SIMATIC Manager.

Una volta mandato in esecuzione SIMATIC Manager è possibile creare un nuovo progetto selezionando **File→New...**<sup>1</sup>.

La finestra che compare è divisa in due: nella parte sinistra viene visualizzata la struttura del progetto in maniera analoga alla struttura di una directory di Windows, mentre nella parte destra viene visualizzato il contenuto dei singoli componenti.

Si noti nel nuovo progetto risulta già presente la rete MPI, per la quale, però, non è definito alcun nodo, essendo il progetto vuoto.

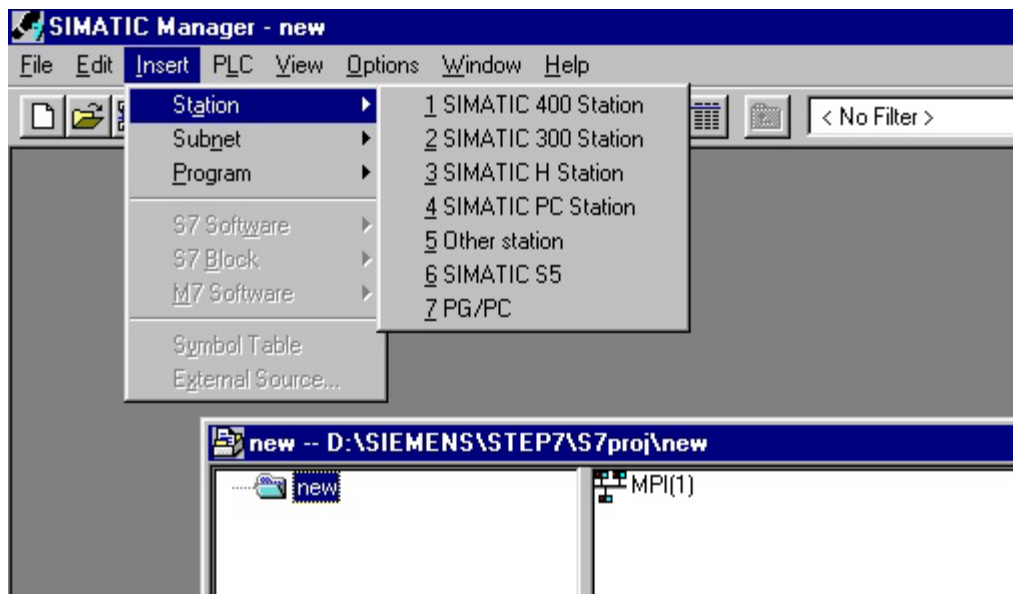


Figura 2 - new project

<sup>1</sup> nel seguito si userà la notazione **File→Save** per indicare la selezione dell'opzione Save nel menù File.

Questa presenza è giustificata dal fatto che la rete MPI (rete proprietaria Siemens) è l'unica rete alla quale è possibile collegare un PLC Siemens non ancora configurato.

Solo attraverso MPI, quindi, è possibile caricare la configurazione hardware in una CPU.

Facendo clic sulla cartella del progetto ed entrando nel menù **Insert** è possibile inserire nel progetto:

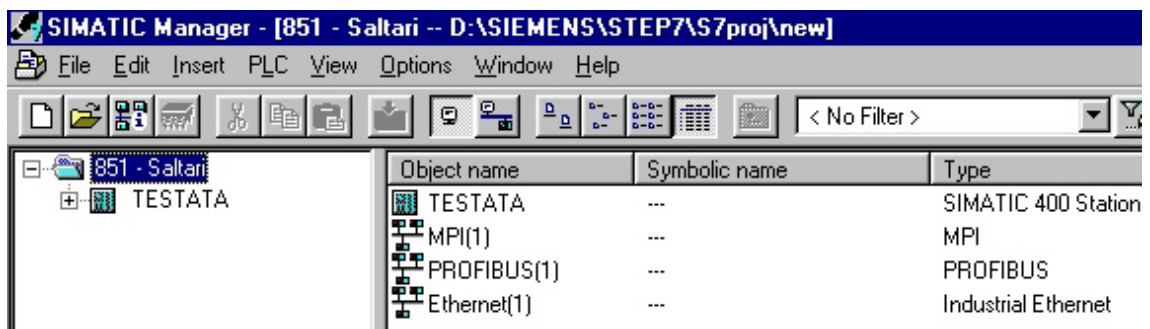
- una stazione SIMATIC;
- un'altra rete di comunicazione;
- un programma (S7 oppure M7).

Si noti che si può inserire un programma senza aver definito la stazione sulla quale andrà in esecuzione; è necessario, invece, definire la stazione di lavoro se si vuole caricare il software sviluppato sul PLC oppure se si vuole utilizzare il simulatore.

Se si desidera inserire un PLC della serie S7-400 selezioniamo **Insert**→**Station**→**SIMATIC 400 Station** e gli si assegni un nome (nell'esempio di fig. 3 è stata scelta il nome *Testata*; per rinominare gli oggetti basta selezionarli e premere F2).

A questo punto nella cartella del progetto comparirà il simbolo della stazione.

In maniera analoga è possibile inserire altre reti di comunicazione che fanno parte del sistema di controllo (per esempio reti Profibus e Industrial Ethernet).



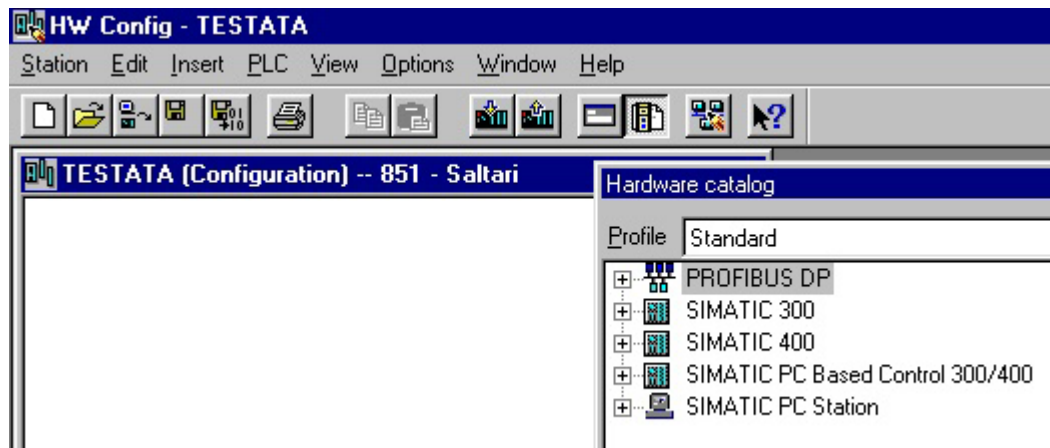
**Figura 3 - Stazione SIMATIC 400**

Selezionando la stazione, sulla destra compare l'icona *Hardware*.



**Figura 4 - Icona Hardware**

Facendo doppio clic su quest'icona viene richiamata l'applicazione *HW Config* con la quale è possibile configurare l'hardware della stazione:



**Figura 5 - HW Configuration**

Sulla destra è presente il *Catalogo hardware* dal quale si possono selezionare i componenti che compongono la stazione. Nel caso il catalogo non appaia è sufficiente selezionare **Insert→Hardware Components**.

Nella finestra a sinistra (*Configuration*) comparirà la struttura della stazione mano a mano che essa verrà definita.

Per portare un oggetto dalla finestra di destra a quella di sinistra è sufficiente utilizzare il *drag&drop*.

In STEP 7 le unità vengono collocate su *telai di montaggio (rack)*, proprio come in un impianto reale. L'unica differenza consiste nel fatto che i telai vengono rappresentati mediante *tabelle di configurazione*, caratterizzate da un numero di righe pari a quello delle unità inseribili nel telaio di montaggio reale.

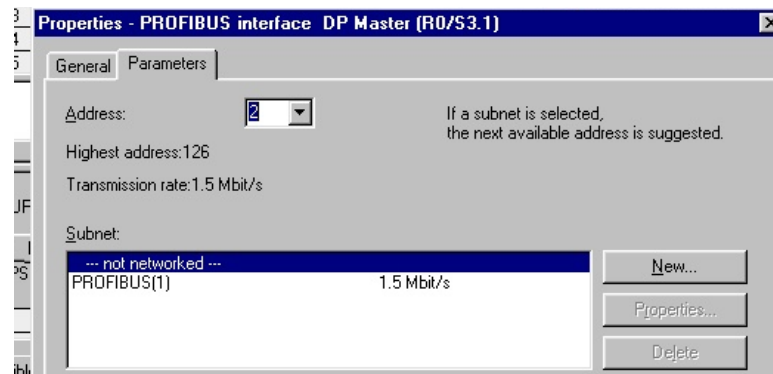
Il primo passo, quindi, è la scelta del *rack*. Per il nostro esempio inseriamo un UR1 (*Universal Rack*). Nella parte inferiore della finestra del catalogo, compare una breve descrizione del componente selezionato.

A questo punto, tenendo conto della configurazione della stazione di *Testata* riportata in fig. 6, si possono inserire le varie unità negli *slot* del telaio.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Alimentatore 10 A 120-230 V Ac		Unità Centrale CPU 414-2-DP		Interfaccia seriale CP 441-1	Interfaccia ethernet CP 443-1	Modulo 32 Ingressi Digitali 24 V Dc	Modulo 32 Ingressi Digitali 24 V Dc	Modulo 32 Ingressi Digitali 24 V Dc	Modulo 32 Ingressi Digitali 24 V Dc	Modulo 32 Ingressi Digitali 24 V Dc	Modulo 32 Ingressi Digitali 24 V Dc	Modulo 32 Ingressi Digitali 24 V Dc	Modulo 32 Uscite Digitali 24 V Dc	Modulo 32 Uscite Digitali 24 V Dc	Modulo 32 Uscite Digitali 24 V Dc	Modulo 32 Uscite Digitali 24 V Dc	Modulo 32 Uscite Digitali 24 V Dc

**Figura 6 - Configurazione della stazione di Testata**

Appena si inserisce il modulo processore (CPU 414-2 DP) viene chiesto di impostare l'indirizzo Profibus dell'unità; premendo il pulsante Properties si può scegliere la velocità di trasmissione della rete (vedi fig. 7).

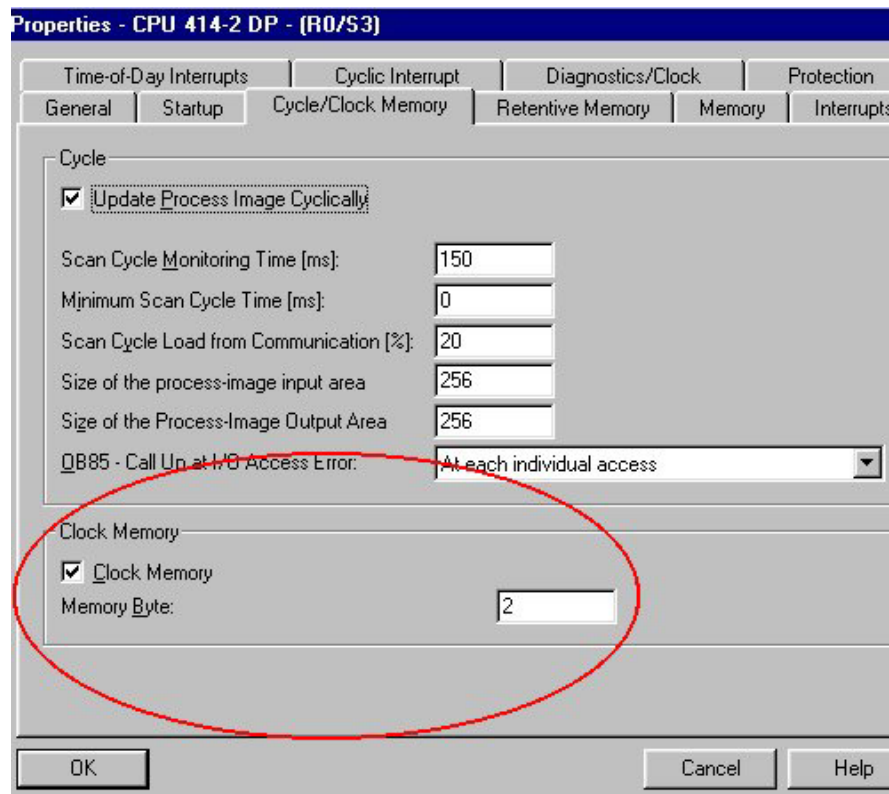


**Figura 7 - Finestra di dialogo per l'inserimento dell'indirizzo Profibus della CPU 414-2 DP**

Nella parte inferiore della finestra di configurazione è presente un riepilogo dei parametri per ogni unità inserita.

Selezionando ogni unità è possibile impostare dei parametri semplicemente premendo il tasto destro del mouse e selezionando **Object Properties...** Per esempio è possibile definire gli indirizzi per le unità di ingresso/uscita digitali.

Si può anche definire, nelle proprietà della CPU, un byte della memoria utente (memoria di merker) come *byte di clock* (vedi fig. 8).



**Figura 8 - Byte di clock**

Il programmatore, quindi, avrà a disposizione 8 clock a frequenze diverse (uno per ogni bit); questi segnali possono essere utili quando si vuol far lampeggiare una lampada oppure far suonare una sirena ad intermittenza.

Una volta inserito il modulo processore compare il simbolo della rete Profibus, quindi si possono inserire tutte le unità collegate al PLC *Testata* mediante questa rete.

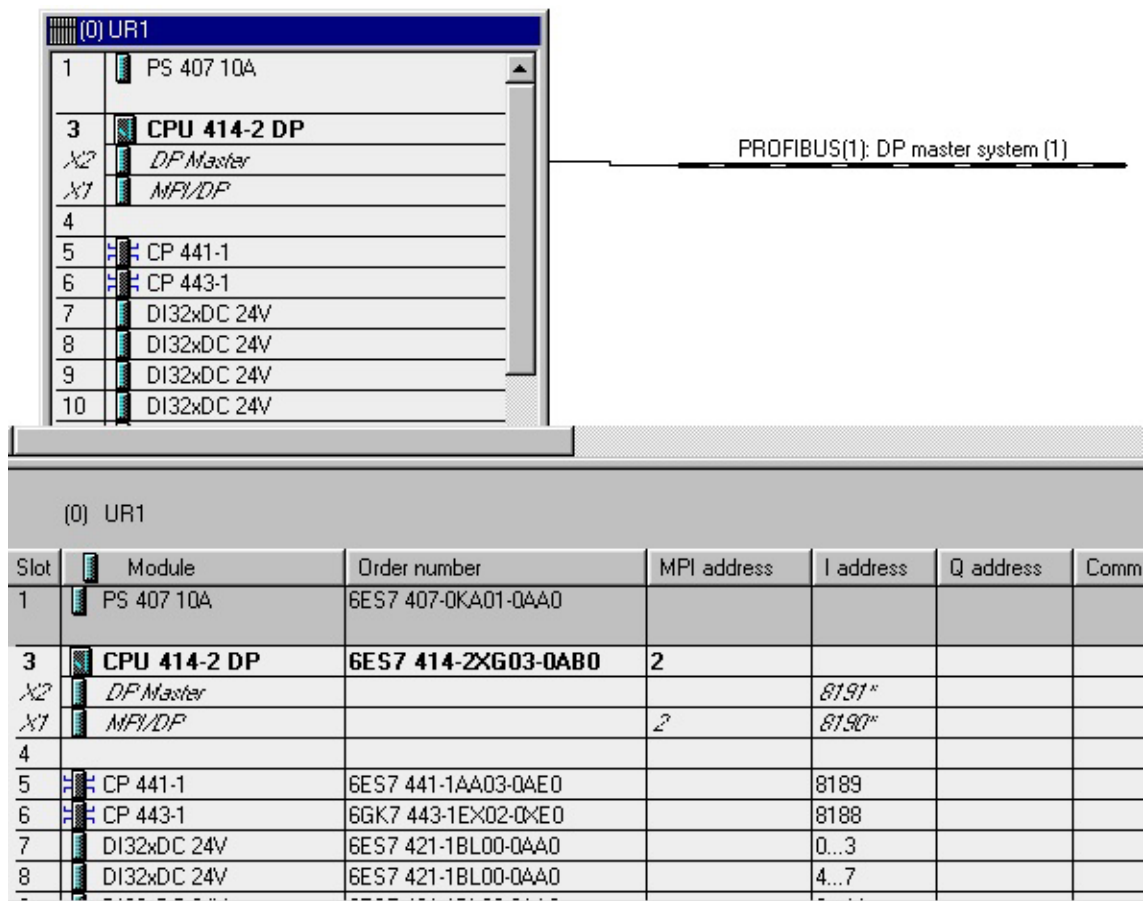
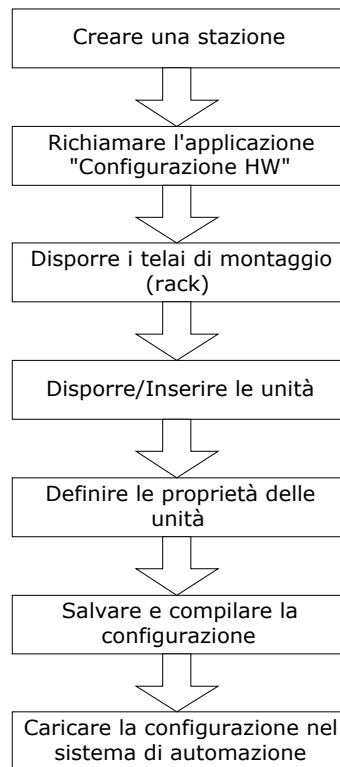


Figura 9 - PLC di Testata

Una volta completata la configurazione hardware, dovrà essere compilata per verificarne la correttezza.

Se la compilazione va a buon fine ed il PC è collegato con il PLC, si può caricare la configurazione fatta nella CPU.

Se si ha a disposizione la stazione già montata e collegata al PC si può caricare la configurazione direttamente selezionando l'opzione **PLC→Upload to PG**; a questo punto si potranno settare tutti i parametri (per esempio gli indirizzi Profibus) e poi caricare di nuovo il tutto nel PLC.



**Figura 10 - Configurazione Hardware**

In fig. 10 è riportata la sequenza di operazioni necessarie per configurare una stazione SIMATIC.

A questo punto in SIMATIC Manager, nella stazione Testata, saranno presenti i moduli processore definiti in precedenza e, al loro interno, ci saranno dei programmi vuoti.



**Figura 11 - Programma S7**

In ogni programma sono presenti due cartelle: una con i sorgenti del codice a l'altra contenente i blocchi eseguibili che dovranno essere caricati nel controllore.

Alcuni linguaggi (AWL, KOP e FUP) permettono di creare dei blocchi che possono essere caricati direttamente nel controllore senza dover essere compilati, mentre con altri linguaggi (SCL) si possono scrivere solo programmi sorgente.

## 2 - Struttura dei programmi S7

Nella CPU dei PLC vengono eseguiti due programmi distinti:

- il sistema operativo;
- il programma utente.

Il sistema operativo (s.o.) è presente in tutte le CPU e organizza tutte le funzioni e le procedure che non sono legate ad un compito di controllo specifico. È il s.o. che realizza la modalità di funzionamento chiamata *ciclo a copia massiva degli ingressi e delle uscite*, ovvero:

- aggiorna l'immagine di processo degli ingressi;
- esegue il programma utente, operando sui valori della memoria e conservando i risultati sempre in memoria;
- esegue i programmi di gestione del sistema;
- scrive l'immagine di processo delle uscite.

L'*immagine di processo* è una copia degli ingressi e delle uscite che viene conservata in un'apposita area di memoria del controllore. Il programmatore, quindi, non accede direttamente alle unità di I/O, bensì a quest'area di memoria dedicata.

I programmi di gestione si occupano di:

- rilevare gli allarmi e richiamare i segmenti di codice da eseguire in caso di allarme;
- gestire la memoria;
- riconoscere e gestire gli errori;
- gestire la comunicazione con i dispositivi di comunicazione e gli altri nodi della rete.

In realtà per i controllori della serie S7 l'ordine delle operazioni effettuate nel ciclo è differente, infatti per questi controllori il s.o.:

- avvia il *tempo di controllo ciclo*;
- scrive l'immagine di processo delle uscite nelle unità di uscita;
- legge lo stato degli ingressi nelle unità d'ingresso e ne aggiorna l'immagine di processo;
- esegue il programma utente;
- esegue i programmi di gestione del sistema.

Il *tempo di ciclo* è il tempo che richiede il s.o. per effettuare il ciclo a copia massiva. Questo tempo non deve essere superiore ad un valore massimo che è possibile impostare, e viene controllato dal s.o. ad ogni scan.

Se questo tempo massimo viene superato la CPU va in STOP, oppure viene richiamato un segmento di codice particolare nel quale si può stabilire la modalità di reazione della CPU all'errore di tempo.

Il programma utente deve essere creato e caricato nella CPU. Questo programma contiene tutte le funzioni necessarie per risolvere il problema di automazione considerato.



STEP 7 offre la possibilità di strutturare il programma utente, ovvero di suddividerlo in singole sezioni indipendenti, ottenendo i seguenti vantaggi:

- le singole parti del programma possono essere standardizzate;
- l'organizzazione del programma viene semplificata;
- le modifiche del programma si possono eseguire più facilmente;
- la semplificazione del testing e della validazione del programma, poiché possono essere eseguiti per sezioni;

Con STEP 7 il programmatore ha a disposizione dei blocchi elementari attraverso i quali può strutturare il software; questi blocchi possono essere divisi in due gruppi:

- *blocchi di codice*;
- *blocchi di dati*.

Esiste un numero massimo di blocchi inseribili per ogni tipo e questo limite varia a seconda del modulo processore utilizzato.

Ogni tipo di blocco può essere inserito come sorgente in un opportuno linguaggio di programmazione oppure come blocco direttamente eseguibile e caricabile nella memoria del controllore.

Per poter inserire un blocco eseguibile in un programma S7 bisogna aprire la cartella *Blocks* nella finestra destra di SIMATIC Manager, e selezionare l'opzione **Insert→S7 Block**. Se, invece, si vuole inserire un sorgente bisogna selezionare **Insert→S7 Software**, dopo aver cliccato sulla cartella *Sources*.

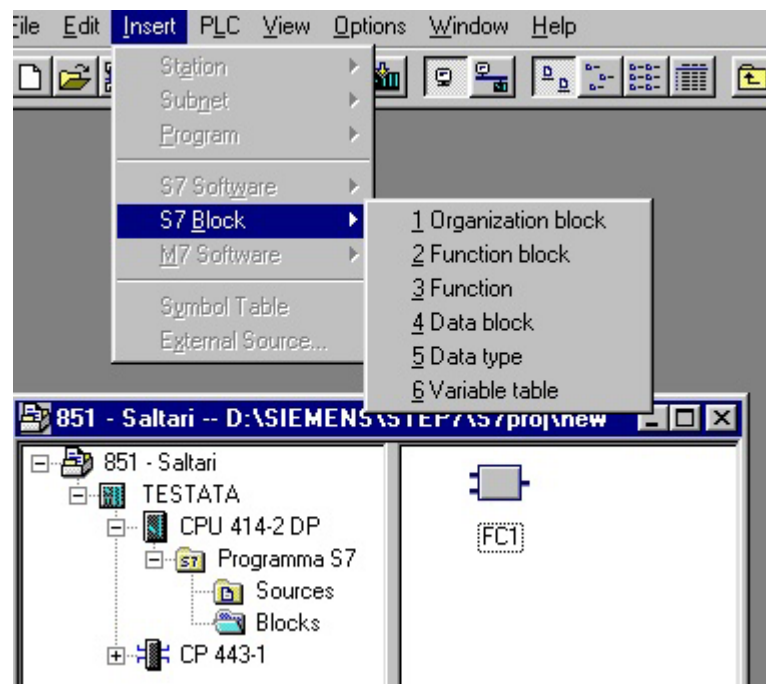


Figura 12 - Menù Insert→S7 Block

Quando si inserisce un blocco eseguibile, compare un'icona nella finestra destra di SIMATIC Manager; facendo doppio click sull'icona il blocco viene aperto e SIMATIC Manager richiama l'*editor AWL, KOP e FUP*; attraverso questa applicazione l'utente può scrivere il codice e creare strutture dati.



Blocco	Descrizione
Blocchi organizzativi (OB)	Gli OB determinano la struttura del programma utente
Funzioni (FC)	Le FC contengono routine di programma
Blocchi Funzionali (FB)	Gli FB sono blocchi funzionali con "memoria"
Blocchi Funzionali di Sistema (SFB) e Funzioni di Sistema (SFC)	Gli SFB e gli SFC sono integrati nella CPU e rendono accessibili alcune importanti funzioni di sistema

**Tabella 1 - Blocchi di codice**

Il blocco OB1 viene richiamato dal s.o. ad ogni ciclo (*scan*).

È in questo blocco che andranno messe le chiamate agli altri blocchi del programma utente; grazie all'OB1, quindi, si implementa la modalità "normale" di funzionamento del controllore programmabile.

L'OB1 è come il main nel linguaggio C e quindi sarà presente in ogni programma S7.

È possibile scrivere l'intero programma utente in questo blocco (*programmazione lineare*). Questa operazione, però, è consigliabile solo per programmi molto semplici. In generale è conveniente avere diversi segmenti di codice in blocchi diversi, ognuno dei quali esegue una funzione elementare (*programmazione strutturata*); nell'OB1 saranno presenti solo le chiamate a questi blocchi.

La priorità dell'OB1 vale 1 e non può essere cambiata dal programmatore. Esiste solo un blocco organizzativo con priorità più bassa ed è l'OB90 (la cui priorità è, per convenzione, 29).

Oltre all'OB1 ci sono altri blocchi organizzativi che vengono chiamati dal s.o. su eventi speciali come un guasto nel rack di montaggio (OB86), il riavviamento del controllore (OB101) oppure il superamento del tempo ciclo massimo (OB80).

L'OB100, invece, viene richiamato solo al primo scan.

Altri OB, come l'OB10, vengono richiamati periodicamente ad intervalli prefissati di tempo impostabili dall'utente (*OB a tempo*).

In fig. 14 è riportato un diagramma temporale che mostra le chiamate del s.o. ad alcuni OB in un tempo ciclo.

### **Funzioni (FC)**

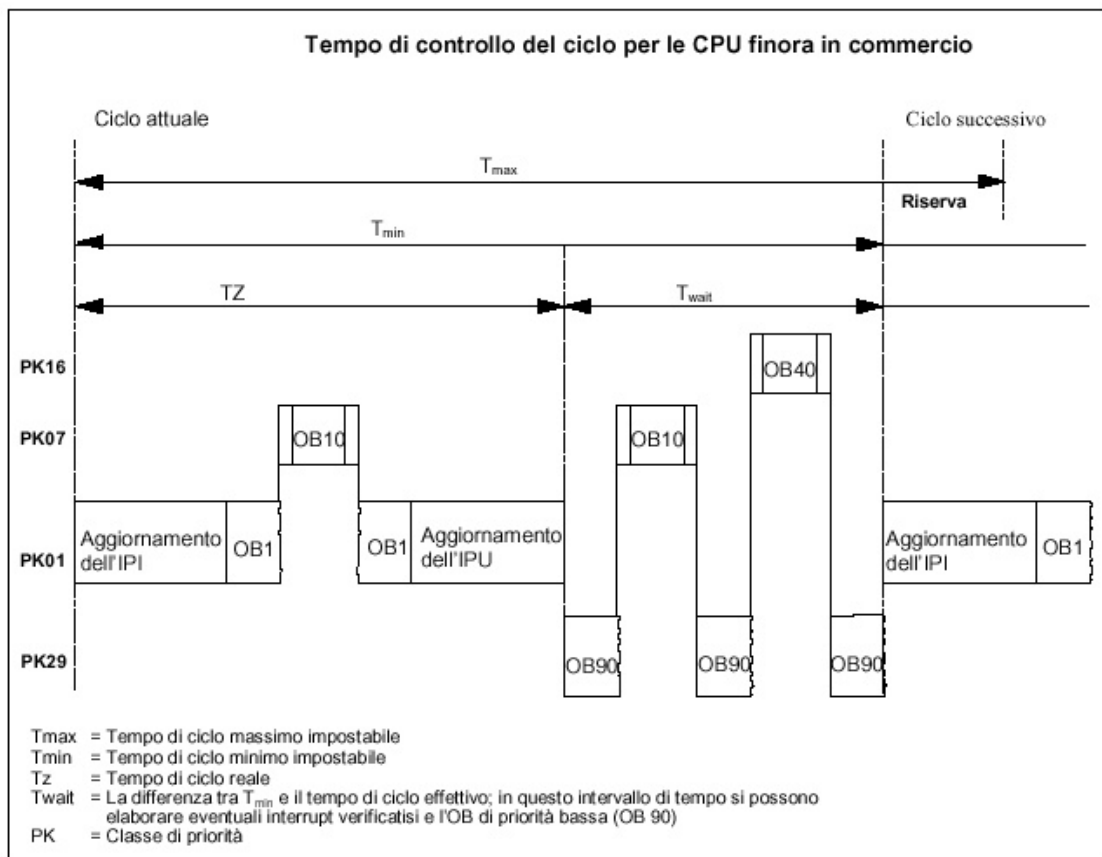
Una FC contiene un programma che viene eseguito ogni qualvolta essa viene richiamata da un altro blocco di codice.

Per ogni funzione è possibile dichiarare dei parametri di ingresso, di uscita, di ingresso/uscita, nonché delle variabili temporanee.

Quest'ultime vengono memorizzate nello stack dei dati locali e vanno perdute dopo l'elaborazione dell'FC.

Non è possibile assegnare alcun valore iniziale a nessun dei parametri di una funzione.

Per la memorizzazione dei dati, le funzioni possono utilizzare blocchi dati globali (*DB*).



**Figura 14 - Tempo ciclo**

Poiché le FC sono blocchi “*senza memoria*”, ogni qual volta se ne effettua la chiamata, bisogna specificare i valori attuali dei parametri d’ingresso, di ingresso/uscita e i parametri d’uscita.

### **Blocchi funzionali (FB)**

Anche il codice contenuto in un FB viene eseguito ogni qualvolta l’FB viene richiamato da un altro blocco di codice.

Un blocco funzionale è un blocco “*con memoria*”; esso dispone di un blocco dati correlato (*blocco dati di istanza*), nel quale vengono memorizzati i valori delle variabili di ingresso, d’uscita, d’ingresso/uscita e *statiche*.

La struttura del blocco dati di istanza corrisponde con le variabili statiche specificate nella finestra di dichiarazione dell’editor KOP, FUP e AWL.

Nelle FB è possibile utilizzare anche variabili temporanee che, però, vengono memorizzate nello stack dei dati locali.

Al termine dell’elaborazione dell’FB, i dati memorizzati nel blocco dati di istanza non vanno perduti, al contrario di quelli memorizzati nello stack dei dati locali.

Ad ogni variabile, tranne per quelle temporanee, è possibile assegnare un valore iniziale.

Ogni volta che un blocco funzionale viene richiamato, bisogna assegnargli il blocco dati di istanza.

Con il richiamo di diverse istanze di un FB si possono comandare più apparecchiature con un solo blocco funzionale. Per esempio, usando blocchi di dati di istanza differenti per vari motori, una sola FB può controllare più motori.

Quando si effettua la chiamata di un FB non è necessario assegnare dei valori a tutti i parametri del blocco; in questo caso verranno utilizzati i valori memorizzati nel blocco dati di istanza.

### **Blocchi funzionali di sistema e funzioni di sistema (SFB, SFC)**

Gli SFB e gli SFC sono blocchi integrati nelle CPU della serie S7.

Questi blocchi sono analoghi ai blocchi FB e FC e realizzano delle funzioni predefinite come la copia di un'area di memoria in un'altra, l'impostazione dell'orologio di sistema, l'invio di un pacchetto di dati ad un nodo di una rete progettata, ecc.

#### **4.2.2 – Blocchi di dati e la memoria utente**

Prima di parlare dei blocchi di codice diamo qualche cenno sulla memoria utente dei controllori S7.

Il programmatore può accedere a sette diverse aree di memoria utente:

- l'immagine di processo degli ingressi (*IPI*);
- l'immagine di processo delle uscite (*IPU*);
- l'area dei merker;
- i blocchi dati;
- lo stack dei dati locali;
- l'area dei timer;
- l'area dei contatori.

Le dimensioni di ognuna di queste aree dipendono dal particolare controllore. Tutte le aree sono strutturate a byte anche se si può accedere ai singoli bit di ogni byte.

Abbiamo già visto che nelle aree IPI e IPU il s.o. copia, ad ogni scan, lo stato dei moduli di I/O.

L'area dei merker è un'area tutta a disposizione del programmatore; non è possibile, da parte dell'utente, definire una particolare struttura per quest'area di memoria.

I blocchi dati vengono memorizzati in un'apposita area di memoria. In questo caso l'utente può definire all'interno di ogni blocco delle strutture dati come gli array e i record, che in S7 vengono chiamati *struct*.

Il programmatore, inoltre, può anche accedere allo stack dei dati locali, dove vengono memorizzate le variabili temporanee dei blocchi di codice.

Infine, nelle ultime due aree, il programmatore può accedere direttamente a dei timer e a dei contatori.

Per poter utilizzare nel codice un operando bisogna specificarne l'indirizzo oppure, come vedremo in seguito, il simbolo associato.

Un indirizzo è composto da un prefisso, che specifica l'area di memoria alla quale si vuole accedere, e da un numero che specifica l'indirizzo all'interno dell'area stessa.

Nel prefisso sono contenute informazioni anche sulla dimensione dell'informazione alla quale si vuole accedere, cioè se si vuole leggere un singolo bit, un byte, una word oppure una doppia word.

I prefissi per le varie aree sono riportati in Tabella 2.

Facciamo qualche esempio:

E0.4 → 5° bit del 1° byte dell'area di memoria degli ingressi

AW4 → word composta dal 5° e 6° byte dell'area di memoria delle uscite

DB100.DBX11.5 → 6° bit del 12° byte del blocco dati globali 100

LD5 → doppia word dai byte 6...9 dello stack dei dati locali

**Nota:** la generica word n è composta dai byte n (byte più significativo) ed n+1 (byte meno significativo). Analogamente la doppia word n è composta dalle word n (word più significativa) ed n+2 (word meno significativa).

Oltre ad accedere alle immagini di processo di ingresso e uscita, il programmatore S7 può anche accedere direttamente ai moduli di I/O, utilizzando i prefissi PE e PA. Questa modalità di accesso agli I/O viene spesso utilizzato all'interno di OB a tempo per controllare delle grandezze critiche. In questo caso, però, non si può accedere al singolo bit, ma solo ai byte, alle word oppure alle doppie word.

Torniamo, ora, ai blocchi di dati. A differenza dei blocchi di codice, questi non contengono istruzioni, ma servono alla memorizzazione dei dati utente.

Esistono due tipi di blocchi dati e sono elencati in Tabella 3.

Le *DB* contengono dati che possono essere utilizzati da tutti i blocchi di codice. In altre parole le *DB* contengono le variabili dichiarate dal programmatore.

La struttura di una *DB globale* viene determinata dall'utente e non dipende da nessun blocco di codice, mentre la struttura delle *DB d'istanza* rispecchia la dichiarazione delle variabili dell'FB a cui fa riferimento.

Quando si apre un blocco dati viene richiamato l'editor KOP, AWL e FUP e viene visualizzata la struttura del blocco (vedi fig. 15).

Nella prima colonna, per ogni variabile, è riportato l'indirizzo interno al blocco di dati. Nella altre colonne si devono inserire, rispettivamente, il nome del dato, il tipo, il valore iniziale e un commento.

Quella mostrata in figura è la *Declaration View* ed è la visualizzazione che permette di editare la struttura del blocco (per poter passare a questa modalità di visualizzazione basta selezionare **View→Declaration View**).

All'interno di una *DB* si possono definire dei dati strutturati: nell'esempio in figura esiste una struttura *FrontiDiSalita*, che non è altro che un record di bit, ognuno dei quali rappresenta il fronte di un segnale d'ingresso. Il codice che genera questi fronti a partire dagli ingressi si troverà in un blocco FC.

Area di memoria	Dimensione del dato	Prefisso	Area di memoria	Dimensione del dato	Prefisso
Immagine di processo degli ingressi	Bit	E	Blocchi dati di istanza	Blocco	DI
	Byte	EB		Bit	DIX
	Word	EW		Byte	DIB
	Double	ED		Word	DIW
Immagine di processo delle uscite	Bit	A	Stack dei dati locali	Double	DID
	Byte	AB		Bit	L
	Word	AW		Byte	LB
	Double	AD		Word	LW
Area merker	Bit	M		Double	LD
	Byte	MB		<b>Area di memoria</b>	<b>Prefisso</b>
	Word	MW		Area Timer	T
	Double	MD		Area Contatori	Z
Blocchi dati globali	Blocco	DB			
	Bit	DBX			
	Byte	DBB			
	Word	DBW			
	Double	DBD			

Tabella 2 - Prefissi delle aree di memoria

Blocco	Descrizione
Blocchi dati di istanza (DB di istanza)	Le DB di istanza vengono assegnate al blocco funzionale quando viene richiamato un FB oppure un SFB. Essi vengono generati automaticamente nella compilazione.
Blocchi Dati globali (DB)	Le DB sono aree dati per la memorizzazione dei dati utente.

Tabella 3 - Blocchi di dati

L'utente ha la possibilità di definire dei tipi di dati attraverso i blocchi *UDT*. Per inserire uno di questi blocchi bisogna selezionare l'opzione **Insert→S7 Block→Data Type**.

Una volta aperto, un blocco UDT è uguale ad una DB. Nella fig. 16 è un esempio di UDT.

LAD/STL/FBD - [DB50 -- 851-Saltari\TESTATA\CPU 414-2 DP(1)]

File Edit Insert PLC Debug View Options Window Help

+2.3	AreaD_InAnomalia	BOOL	FALSE	
+2.4	AreaE_InAnomalia	BOOL	FALSE	
+2.5	AreaF_InAnomalia	BOOL	FALSE	
+2.6	AreaG_InAnomalia	BOOL	FALSE	
+2.7	spare25	BOOL	FALSE	
+3.0	spare26	BOOL	FALSE	
+3.1	spare27	BOOL	FALSE	
+3.2	spare28	BOOL	FALSE	
+3.3	spare29	BOOL	FALSE	
+3.4	spare30	BOOL	FALSE	
+3.5	spare31	BOOL	FALSE	
+3.6	spare32	BOOL	FALSE	
+3.7	spare33	BOOL	FALSE	
+4.0	FrontiDiSalita	STRUCT		
+0.0	SB8	BOOL	FALSE	
+0.1	SB9	BOOL	FALSE	
+0.2	KAI18	BOOL	FALSE	
+0.3	KAI22	BOOL	FALSE	
+0.4	Seq6_11	BOOL	FALSE	
+0.5	Seq8_11	BOOL	FALSE	
+0.6	Seq10_11	BOOL	FALSE	

Figura 15 - DB, declaration view

LAD/STL/FBD - [UDT1 -- 851-Saltari\TESTATA\CPU 414-2 DP(1)]

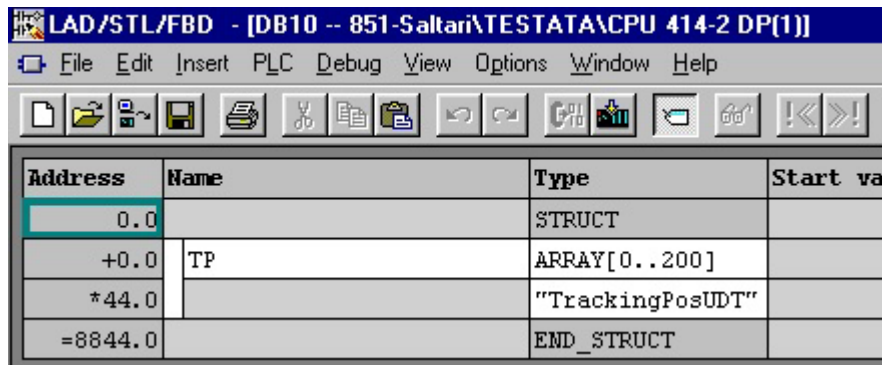
File Edit Insert PLC Debug View Options Window Help

Address	Name	Type	Start value	Comment
0.0		STRUCT		
+0.0	PresenzaLogica	BOOL	FALSE	0=Assenza logica 1=Presen
+0.1	MissioneAccettata	BOOL	FALSE	
+0.2	MissioneRifiutata	BOOL	FALSE	
+0.3	MissioneRichiesta	BOOL	FALSE	0= Missione non richiesta
+0.4	AnomaliaVuotoVuoto	BOOL	FALSE	
+0.5	AnomaliaPienoPieno	BOOL	FALSE	
+0.6	AnomaliaLoc2inacc	BOOL	FALSE	
+0.7	PresenzaFisica	BOOL	FALSE	0=Indefinito 1=UdC present
+1.0	AssenzaFisica	BOOL	TRUE	0=Indefinito 1=UdC assente
+1.1	ZonaInAutomatico	BOOL	FALSE	0=Indefinito 1=Zona in aut
+1.2	ZonaInManuale	BOOL	FALSE	0=Indefinito 1=Zona in mam

Figura 16 - Tracking Position UDT



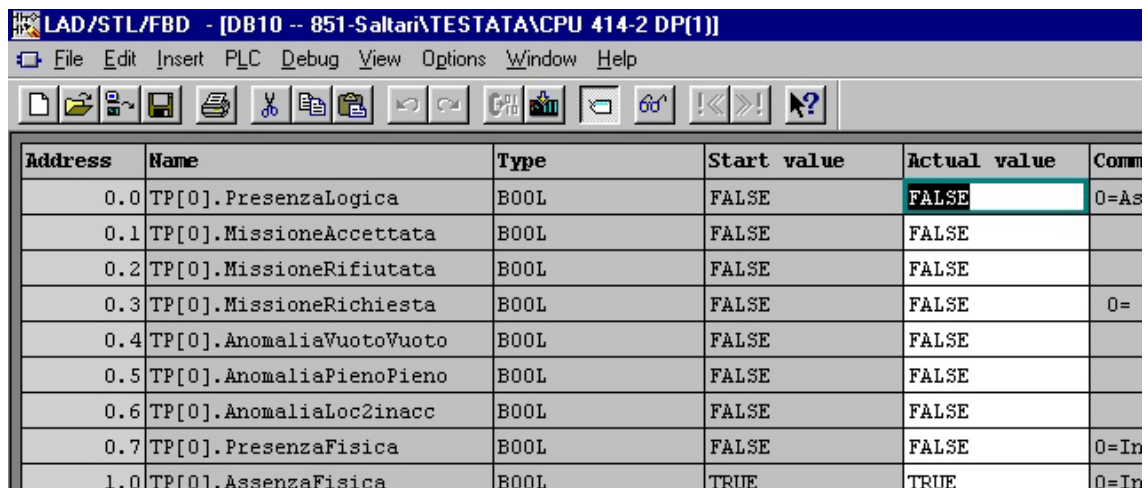
Una volta definito un tipo di dati con un UDT, questo può essere utilizzato all'interno di una DB, oppure all'interno della sezione di dichiarazione variabili di un blocco di codice. L'UDT1, per esempio, viene richiamato nella DB10 (vedi fig. 17).



Address	Name	Type	Start va
0.0		STRUCT	
+0.0	TP	ARRAY[0..200]	
*44.0		"TrackingPosUDT"	
=8844.0		END_STRUCT	


Figura 17 - Track DB


Dalla dichiarazione si capisce che la DB10 contiene un array di 200 UDT1 ("TrackingPosUDT" è il nome simbolico che è stato associato all'UDT1). Se si seleziona **View→Data View** non verrà più visualizzata la struttura della DB, bensì i singoli dati accessibili:



Address	Name	Type	Start value	Actual value	Comm
0.0	TP[0].PresenzaLogica	BOOL	FALSE	FALSE	0=As
0.1	TP[0].MissioneAccettata	BOOL	FALSE	FALSE	
0.2	TP[0].MissioneRifiutata	BOOL	FALSE	FALSE	
0.3	TP[0].MissioneRichiesta	BOOL	FALSE	FALSE	0=
0.4	TP[0].AnomaliaVuotoVuoto	BOOL	FALSE	FALSE	
0.5	TP[0].AnomaliaPienoPieno	BOOL	FALSE	FALSE	
0.6	TP[0].AnomaliaLoc2inacc	BOOL	FALSE	FALSE	
0.7	TP[0].PresenzaFisica	BOOL	FALSE	FALSE	0=In
1.0	TP[0].AssenzaFisica	BOOL	TRUE	TRUE	0=In

Figura 18 - Data View

Questa modalità di visualizzazione è l'unica che può essere utilizzata quando si è *on-line* con il controllore oppure con il simulatore; in questo caso nell'ultima colonna viene mostrato il valore istantaneo dei dati. Per poter andare *on-line* è sufficiente premere il pulsante  oppure selezionare l'opzione **Debug→Monitor**.

Con l'opzione **Edit→Initialize Data Block** ( pulsante  ), invece, è possibile forzare i valori iniziali nelle singole variabili che costituiscono la DB.

### 3 – Linguaggi di programmazione S7

Ognuno dei blocchi di codice che abbiamo introdotto nel paragrafo precedente può contenere istruzioni in diversi linguaggi di programmazione.

Il software di base di STEP 7 mette a disposizione tre linguaggi di programmazione:

- *KOP o LAD (schema a contatti)*; è un linguaggio di programmazione grafico che consente all'utente di seguire in modo semplice il flusso dei segnali tra le sbarre collettrici, i contatti, gli elementi complessi e le bobine.
- *FUP o FBD (schema funzionale)*; anche questo è un linguaggio grafico e rappresenta la logica mediante i blocchi dell'algebra booleana. Esso consente, inoltre, di rappresentare funzioni complesse (ad esempio funzioni matematiche) direttamente connesse a i blocchi booleani.
- *AWL o STL (lista di istruzioni)*; è un linguaggio di programmazione testuale vicino al linguaggio macchina. Nella maggior parte dei casi, quindi, c'è una corrispondenza uno ad uno tra le istruzioni AWL e le operazioni che la CPU del controllore esegue.

Questi tre linguaggi sono, rispettivamente, la versione Siemens del linguaggio a contatti, del diagramma a blocchi funzionali e della lista d'istruzioni specificati dallo standard IEC 61131.

AWL, KOP e FUP consentono di scrivere blocchi di codice direttamente caricabili nel controllore, senza dover effettuare alcuna compilazione.

In STEP 7 viene messo a disposizione anche l'SCL. Questo è un linguaggio testuale messo a disposizione come software opzionale; l'SCL ha espressioni linguistiche simili al Pascal e corrisponde al Testo Strutturato dello standard IEC 61131.

A differenza dei linguaggi precedenti, i sorgenti SCL vanno compilati prima di poter essere caricati nel controllore.

Solo per funzioni molto semplici, in cui si è voluta sfruttare l'immediatezza di un linguaggio grafico, sono stati utilizzati il KOP oppure il FUP.

Qualsiasi sia il linguaggio utilizzato all'interno di un blocco di codice, è sempre possibile richiamare, al suo interno, altri blocchi scritti in un linguaggio diverso.

Nelle pagine che seguono, presenteremo l'AWL, il KOP, il FUP e l'SCL; nella sezione dedicata all'AWL faremo alcuni cenni sull'architettura interna delle CPU S7.

#### 3.1 – AWL

Tra tutti i linguaggi di programmazione S7, l'AWL rappresenta quello più vicino al linguaggio macchina MC7 delle CPU S7.

Per questa ragione il codice sviluppato in AWL risulterà ottimale sia per quanto riguarda l'occupazione di memoria, sia per quanto riguarda il tempo di esecuzione.

Come tutti i linguaggi di tipo assemblativo, però, la leggibilità dei programmi risulta difficoltosa, sebbene l'editor AWL consenta di inserire vari tipi di commento nel codice.

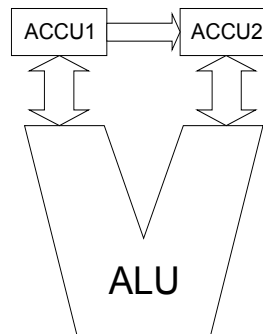
A questo riguardo, si ricordi che l'unica vera istruzione disponibile in AWL per il controllo del flusso di programma è il salto, ed è con il salto che il programmatore deve implementare le scelte ed i cicli.

## Cenni sull'architettura interna delle CPU della serie S7

### Accumulatori

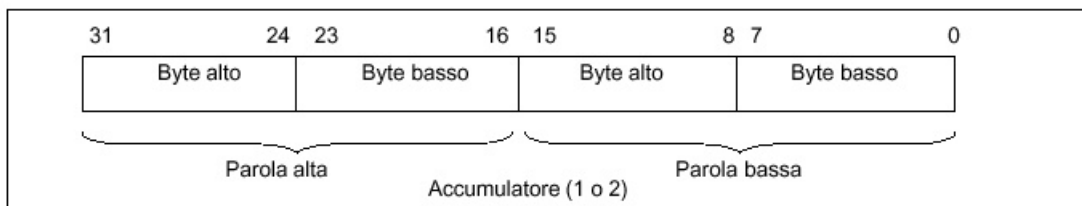
L'architettura delle CPU S7 è basata su una struttura a doppio accumulatore.

I due accumulatori ACCU1 ed ACCU2 sono i due registri operando dell'ALU e consentono di lavorare con byte, word e double word. In essi si possono caricare costanti oppure valori di memoria, eseguire delle operazioni aritmetico/logiche e trasferirne il contenuto in un'area di memoria.



**Figura 19 - ALU e Accumulatori**

In fig. 20 è riportata la struttura di un accumulatore.



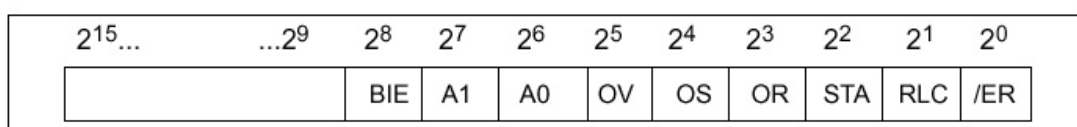
**Figura 20 - Struttura di un accumulatore**

Il meccanismo di gestione degli accumulatori è il seguente:

- un'operazione di caricamento, per esempio L (*load*), trasferisce il contenuto di ACCU1 in ACCU2 e carica l'operando desiderato in ACCU1;
- un'operazione di trasferimento, per esempio T (*transfer*), non modifica il contenuto degli accumulatori (ad eccezione delle istruzioni TAR1 e TAR2);
- l'operazione TAK scambia tra loro i contenuti degli accumulatori.

### Registro di stato

Il registro di stato è una word di cui vengono utilizzati solo i primi 9 bit; la struttura è mostrata in fig. 21.



**Figura 21 - Registro di stato delle CPU S7**

L'utente può accedere ad alcuni di questi bit come operandi per le operazioni logiche.

Prima di illustrare il significato dei singoli bit della word di stato, descriviamo come, le CPU S7, effettuano le operazioni logiche su bit.

Una catena di istruzioni logico/combinatorie in successione viene chiamata *stringa logica*; ogni istruzione di una stringa interroga lo stato dell'operando e lo combina con il contenuto del bit 1 della word di stato (*risultato logico combinatorio - RLC*). Il risultato dell'operazione viene memorizzato ancora nel bit RLC.

Quando una stringa logica termina, ad esempio con un'istruzione di assegnazione, il valore il bit RLC conterrà il risultato di tutte le istruzioni.

L'unica eccezione a questa modalità di funzionamento riguarda la prima istruzione di una stringa logica. In questo caso lo stato dell'operando viene trasferito direttamente nell'RLC.

Il bit 0 (*/ER*) della word di stato viene chiamato *bit di prima interrogazione* e indica l'inizio di una stringa logica; il suo funzionamento è in logica negata. È questo bit che permette di riconoscere la prima istruzione di una stringa logica.

Se */ER* vale 0 l'operando indirizzato in un'istruzione logico/combinatoria viene memorizzato nel bit RLC, dopodichè il valore di */ER* viene impostato ad 1.

Quando */ER* vale 1 un'operazione logico/combinatoria combina il proprio operando con l'RLC; il risultato viene memorizzato nell'RLC stesso.

*/ER* viene rimesso a 0 alla fine di una stringa logica, cioè dopo un'istruzione di assegnazione, oppure dopo un'istruzione di salto correlata al risultato logico della stringa, oppure con un'espressione di annidamento.

Come abbiamo già visto il bit 1, che viene chiamato *risultato logico combinatorio*, memorizza il risultato di un'operazione logica o di confronto.

In fig. 22 è mostrato il comportamento dei bit */ER* ed RLC quando viene valutata un'espressione logica.

Programma AWL	Stato di segnale di ingressi (E) o di uscite (A)	Risultato interrog.	Bit RLC	Bit /ER	Spiegazione
				0	Bit /ER = 0 indica che l'operazione successiva inizia una stringa logica
U E 1.0	1	1	1	1	Il risultato di prima interrogazione viene memorizzato nel bit RLC. Il bit /ER viene settato a 1.
UN E 1.1	0	1	1	1	Il risultato dell'interr. viene combinato con l'RLC preced., secondo la tabella della verità AND. Il bit /ER rimane "1".
= A 4.0	1			0	L'RLC viene assegnato alla bobina di uscita. Il bit /ER viene resettato a "0".

Figura 22 - /EO ed RLC

L'RLC può essere settato incodizionatamente ad 1 oppure a 0 adoperando, rispettivamente, l'istruzione *AWL SET* oppure *CLR*.

In base al valore dell'RLC si possono eseguire o meno operazioni di salto, utilizzando le istruzioni di salto condizionato *SPB* e *SPBN*.

Il bit 2 della word di stato, *STA - bit di stato*, memorizza il valore di un bit a cui viene fatto riferimento in una determinata istruzione.

Nel caso di istruzioni logico/combinatorie che accedono alla memoria in lettura, il bit STA è uguale al valore del bit a cui si accede.

Per le istruzioni logico/combinatorie che accedono alla memoria in scrittura, il valore di STA è uguale al valore scritto oppure, nel caso in cui la scrittura non abbia luogo, al valore dell'operando a cui l'istruzione si riferisce.

Il bit di stato non ha significato per le operazioni logico/combinatorie che non accedono in memoria.

Il bit 3 viene chiamato *bit OR*.

Questo bit ha un significato solo nelle stringhe logiche in cui siano presenti operazioni di AND prima di operazioni di OR. In questo caso il bit di OR indica se una funzione AND, eseguita prima di una OR, ha fornito valore 1, anticipando così il risultato dell'operazione di OR.

Il 5° bit della word di stato è il *bit di overflow (OV)* e indica un errore. OV viene messo a 1 dopo che si è verificato un errore in un'operazione matematica oppure in un'operazione di confronto in virgola mobile.

Il bit successivo è il *bit di overflow con memoria (OS)*; questo bit viene messo a 1 assieme a OV quando si verifica un errore.

Mentre OV viene resettato quando si effettua una nuova operazione, OS rimane ad 1 anche dopo il verificarsi dell'errore, quindi mantiene memoria del verificarsi di un errore in una delle operazioni precedenti.

Il comando *AWL SPS (salta se OS=1)* e i comandi di richiamo blocco e di fine blocco resettano a 0 il valore di OS.

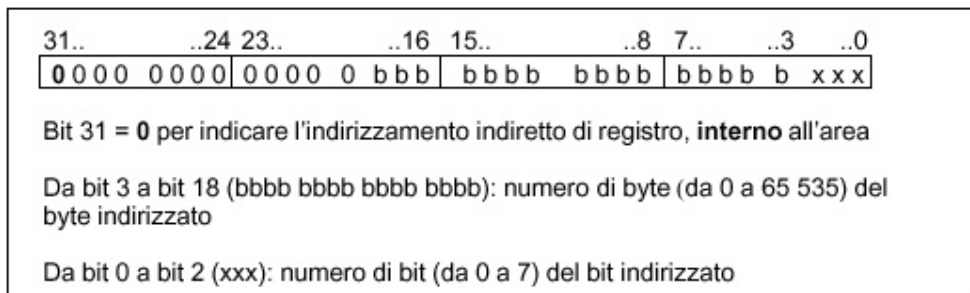
I bit 6 e 7 della word di stato vengono chiamati *bit di visualizzazione A0 e A1* e forniscono informazioni sull'esito di alcune operazioni. Per esempio, in base al valore di questi due bit si può conoscere se il risultato di un'operazione matematica è minore, maggiore oppure uguale a zero, oppure si può risalire al valore del bit traslato in un'operazione di shift.

L'ultimo bit della word di stato a cui è associato un significato è il *bit di risultato binario (BIE)*.

Il bit BIE rappresenta un merker interno alla macchina, in cui viene salvato l'RLC prima di un'operazione che ne modifichi il valore (per esempio un'espressione annidata). Dopo l'operazione, grazie a BIE, è possibile ripristinare il valore dell'RLC e continuare la stringa logica interrotta.

### Registri indirizzo

Nei controllori della serie S7 sono presenti due registri indirizzo AR1 e AR2, che vengono utilizzati delle operazioni di indirizzamento indiretto dei dati.



**Figura 23 - Formato puntatore**

In questi registri è contenuto un dato di tipo puntatore il cui formato è illustrato in fig. 23. Una costante di tipo puntatore si specifica con il prefisso P#.

Un esempio di utilizzo dei registri AR1 e AR2 per un puntamento indiretto è l'istruzione *L EBJ[AR1,P#100.0]*, che carica in ACCU1 il byte d'ingresso il cui indirizzo si ottiene sommando 100 byte al contenuto di AR1.

#### Registri blocco dati

Quando si utilizzano operandi contenuti in un blocco di dati, il numero della DB a cui si accede viene memorizzato nel registro interno *DB1*.

Quando si vuole accedere a più dati all'interno di una stessa DB, si può specificare solo l'indirizzo interno al blocco utilizzando il comando *AWL AUF* che carica in *DB1* il numero della DB da aprire.

Esiste anche un secondo registro *DB2* in cui viene memorizzato il numero della DB di istanza alla quale si sta accedendo.

#### **Istruzioni AWL e sorgenti**

L'AWL di STEP 7 mette a disposizione un ampio set di istruzioni; abbiamo:

- operazioni logico/combinatorie su bit, su word e doppie word;
- timer e contatori;
- operazioni di caricamento e trasferimento;
- operazioni matematiche su numeri interi ed in virgola mobile;
- operazioni di confronto;
- operazioni di conversione tra tipi diversi;
- operazioni di scorrimento;
- operazioni per la manipolazione delle DB;
- operazioni di salto;
- altre operazioni per il controllo del flusso di programma (per esempio chiamata condizionata di un FC/FB, MCR)

Tutte le istruzioni AWL possono essere suddivise in due gruppi: *istruzioni senza operando* ed *istruzioni a singolo operando*.

L'operando può essere una costante, oppure un indirizzo in cui l'istruzione trova un valore. L'operando, inoltre, può essere specificato sia attraverso l'indirizzo, sia attraverso il nome simbolico.

Per poter editare il codice AWL in un blocco di programma, bisogna aprire il blocco ed assicurarsi che il codice scritto venga visualizzato in AWL (selezionare **View→STL** nell'editor KOP/FUP/AWL).

STEP 7 consente di convertire il codice scritto da un linguaggio ad un altro, semplicemente selezionando il linguaggio di visualizzazione nel menù View.

Questa conversione è sempre possibile se si vuole passare da KOP ad AWL o FUP, oppure da FUP ad AWL o KOP.

La conversione da AWL a KOP o FUP, invece, non è sempre possibile: l'editor riesce nell'operazione solo per segmenti di codice semplici.

Con l'editor AWL, KOP e FUP, selezionando l'opzione **File→Generate Source...**, è possibile generare i sorgenti dei blocchi scritti in uno di questi tre linguaggi. Bisogna notare che qualsiasi sia il linguaggio di partenza, si ottiene sempre un sorgente AWL.

Un sorgente è un documento in formato testo che può essere salvato in un file esterno al progetto (**Edit→Export Source...**). È possibile, quindi, creare una libreria di funzioni da riutilizzare in progetti diversi.

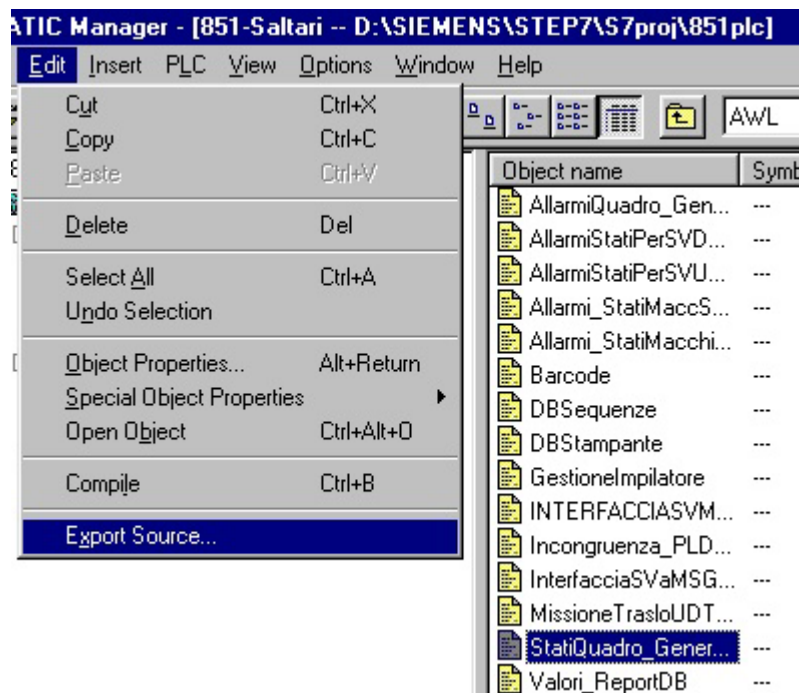


Figura 24 - Esportare un sorgente

Per poter inserire un sorgente esterno all'interno di un progetto è sufficiente aprire la cartella *Sources* del programma S7 e selezionare **Insert→External Source...**. Una volta importato il blocco sorgente questo va compilato per ottenere il blocco eseguibile, selezionando **Edit→Compile** in SIMATIC Manager oppure **File→Compile** nel menù dell'editor.

### 3.2 – KOP e FUP

Il linguaggio a contatti (KOP) e quello a blocchi funzionali (FUP) sono i due linguaggi grafici integrati nel software di base di STEP 7.

Nel KOP le funzioni booleane vengono rappresentate attraverso la logica a relè, mentre il FUP mette a disposizione direttamente dei box che effettuano le operazioni logiche.

Entrambi i linguaggi permettono di realizzare anche operazioni complesse come le operazioni matematiche, le chiamate di FC e l'utilizzo di temporizzatori e contatori.

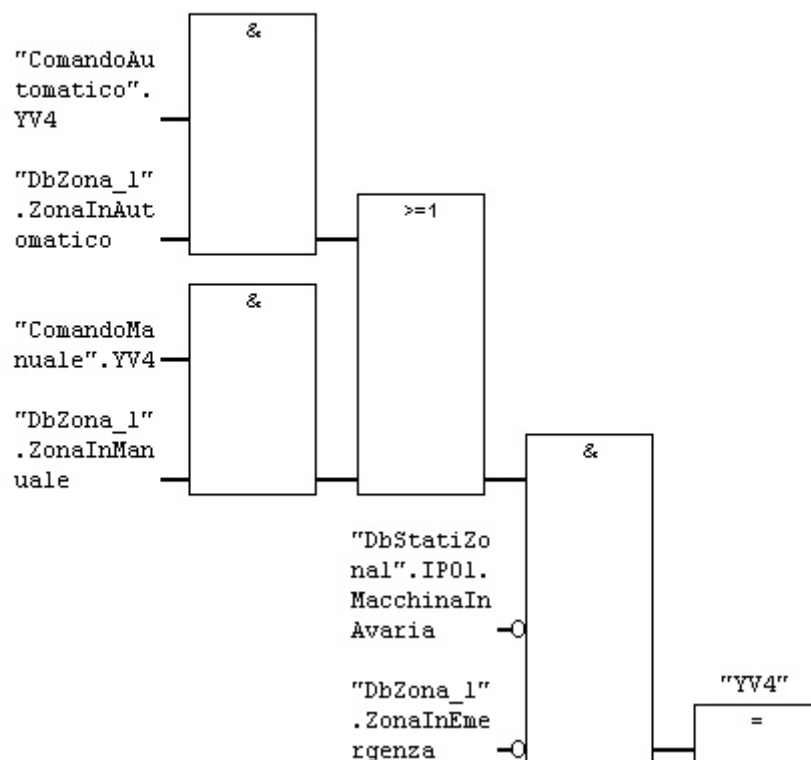
In questo modo l'utente KOP/FUP ha a disposizione tutti gli elementi necessari per creare un programma completo.

Solo per quanto riguarda il KOP, l'editor prevede una limitazione: ogni rung deve essere scritto in un segmento di codice (network) diverso.

Nelle figure 25 e 26 vengono mostrati due esempi di codice KOP e FUP.

**Network 64:** Uscita elettrovalvola YV4 - chiusura pinze IP01

Comment:



**Figura 25 – Segmento di codice FUP**

### 3.3 – SCL

L'SCL è un linguaggio testuale di alto livello per la programmazione dei sistemi di controllo S7.

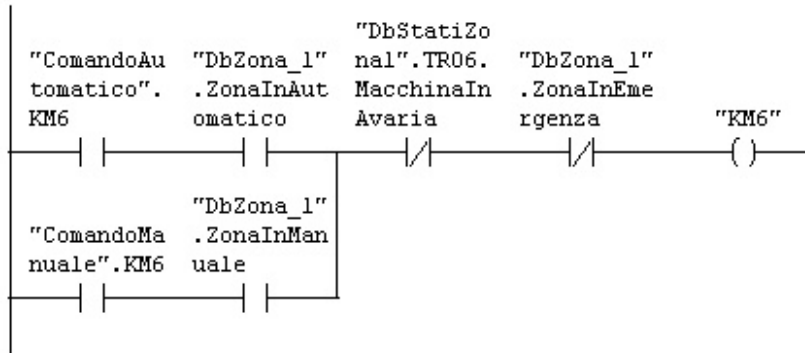
Le espressioni linguistiche e la sintassi dell'SCL sono simili a quelle del linguaggio Pascal.

Con l'SCL si possono creare OB, FC, FB, DB e UDT.



**Network 15 : Uscita motore rulliera TR06 spostamento a destra**

Comment:



**Figura 26 - Rung in linguaggio KOP**

Per poter inserire in un programma un sorgente SCL, bisogna aprire la cartella *Sources* in SIMATIC Manager e selezionare l'opzione **Insert→S7 Software→5 SCL Source**.

A questo punto comparirà, nella parte destra di SIMATIC Manager, l'icona del blocco inserito; aprendo questo blocco viene richiamato l'editor SCL.

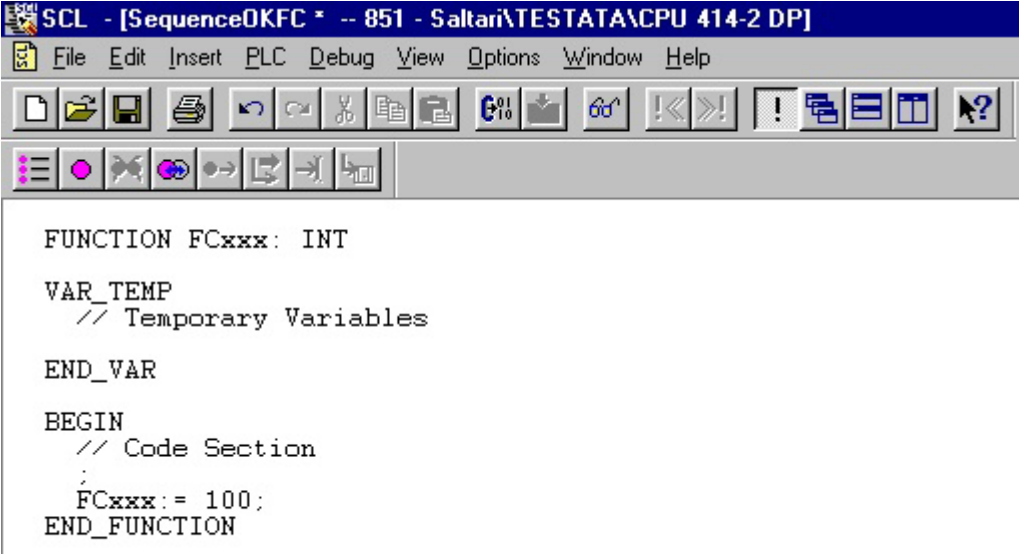
Questo editor permette di inserire i template dei vari blocchi di programma (OB, FC, DB, ecc.) mediante il menù **Insert→Block Template**. Per esempio, se si inserisce la maschera per un FC, l'editor provvede a scrivere automaticamente le intestazioni delle varie parti del blocco (dichiarazione della variabili, blocco istruzioni), così come mostrato in fig. 27.

Per default, l'editor inserisce una funzione che restituisce un intero e prevede solo il blocco per i parametri temporanei. Se si vogliono inserire anche i parametri di I/O o le costanti basta selezionare rispettivamente **Insert→Block Template→Parameter** oppure **Insert→Block Template→Constant**.

Una volta completato un blocco in SCL, questo va compilato per poter generare il blocco eseguibile.

Nella maggior parte dei casi, quando si compila un blocco SCL, il codice eseguibile ottenuto occuperà uno spazio maggiore di memoria rispetto ad un blocco AWL che realizza la stessa funzione.

A volte, per codici molto complessi, questa differenza può raggiungere anche un fattore dieci.



```

FUNCTION FCxxxx: INT
VAR_TEMP
  // Temporary Variables
END_VAR
BEGIN
  // Code Section
  ;
  FCxxxx:= 100;
END_FUNCTION

```

Figura 27 - Blocco FC in SCL

Va detto, però, che, per funzioni molto complesse, anche un programmatore esperto di AWL spesso giunge ad una soluzione poco leggibile e, quindi, di difficile interpretazione da parte di un altro programmatore.

Quando si compila un sorgente SCL bisogna assicurarsi di aver già compilato tutti i blocchi, sia codice che dati, richiamati al suo interno, altrimenti il compilatore darà un errore.

Il *Make File* può essere utilizzato per facilitare la compilazione di più blocchi SCL. Per inserire questo file nella cartella *Sources* basta selezionare **Insert**→**S7 Software**→**6\_SCL compile control file**.

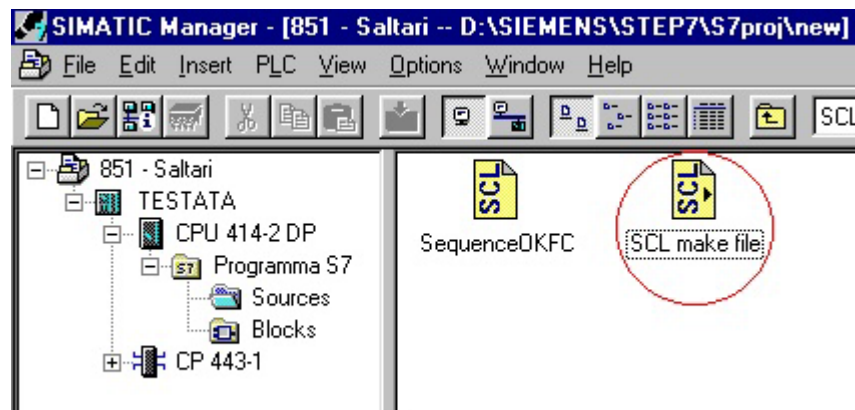


Figura 28 - Make File

Il Make File è un file testo in cui vengono scritti i nomi dei vari sorgenti SCL da compilare. La successione con la quale avviene la compilazione è quella in cui vengono scritti i nomi dei blocchi. Per evitare errori di compilazione, allora, è conveniente strutturare il Make File nel modo seguente:

1. compilazione degli UDT;
2. compilazione delle DB che utilizzano UDT;
3. compilazione delle DB globali;

4. compilazione di FC ed FB, facendo attenzione alla successione in con cui queste vengono compilate (se FC1 viene richiamato in FC2 va compilato prima FC1 e poi FC2) ;
5. compilazione delle DB di istanza;
6. compilazione degli OB.

Se si desidera, si possono inserire più blocchi di programma S7 nello stesso sorgente SCL. La sequenza con la quale i vari blocchi vanno scritti deve rispecchiare le stesse regole con le quali avviene la compilazione dei sorgenti.

Con le istruzioni messe a disposizione dall'SCL si possono valutare in maniera agevole espressioni aritmetiche e logiche anche molto complesse.

Le istruzioni di controllo sono quelle tipiche del Pascal, cioè la scelta (*IF*, *THEN*, *ELSE* e *CASE*), i cicli (*FOR*, *WHILE* e *REPEAT*) e il salto (*GOTO*).

L'utente ha a disposizione anche istruzioni particolari, tipiche dei linguaggi per i controllori, per la gestione dei temporizzatori e dei contatori.


In maniera analoga a quanto visto per i sorgenti AWL, è possibile esportare ed importare da file esterni anche i sorgenti SCL.

### 3.4 – Indirizzi e nomi simbolici

Con STEP 7 è possibile assegnare dei nomi simbolici ai vari blocchi di programma ed agli operandi in memoria; in questo modo si aumenta notevolmente la leggibilità del codice prodotto e viene facilitato il debug.

All'interno di un blocco S7 è possibile riferirsi ad un altro blocco o ad un operando sia con l'indirizzo assoluto, sia con il simbolo.

Con l'editor AWL, KOP e FUP, l'utente può scegliere se visualizzare gli indirizzi assoluti oppure i simboli, selezionando l'opzione **View→Display with→Symbolic Representation** o agendo sul pulsante .

In un sorgente SCL, invece, possono essere  visualizzati contemporaneamente sia gli indirizzi che i simboli, così come vengono digitati dal programmatore.

Per poter fare l'assegnazione indirizzo→simbolo, bisogna aprire l'oggetto *Symbols* nella cartella Programma S7:

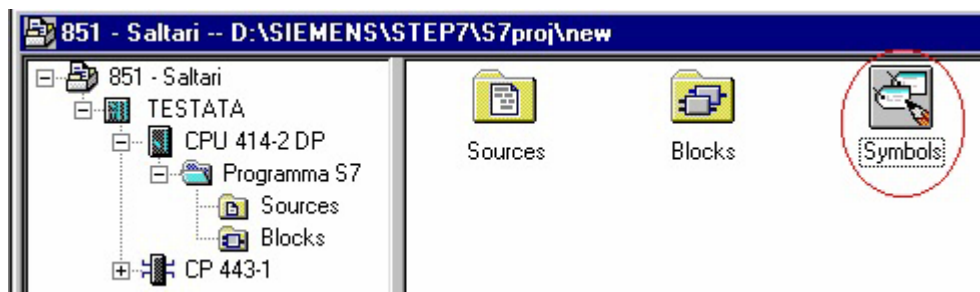
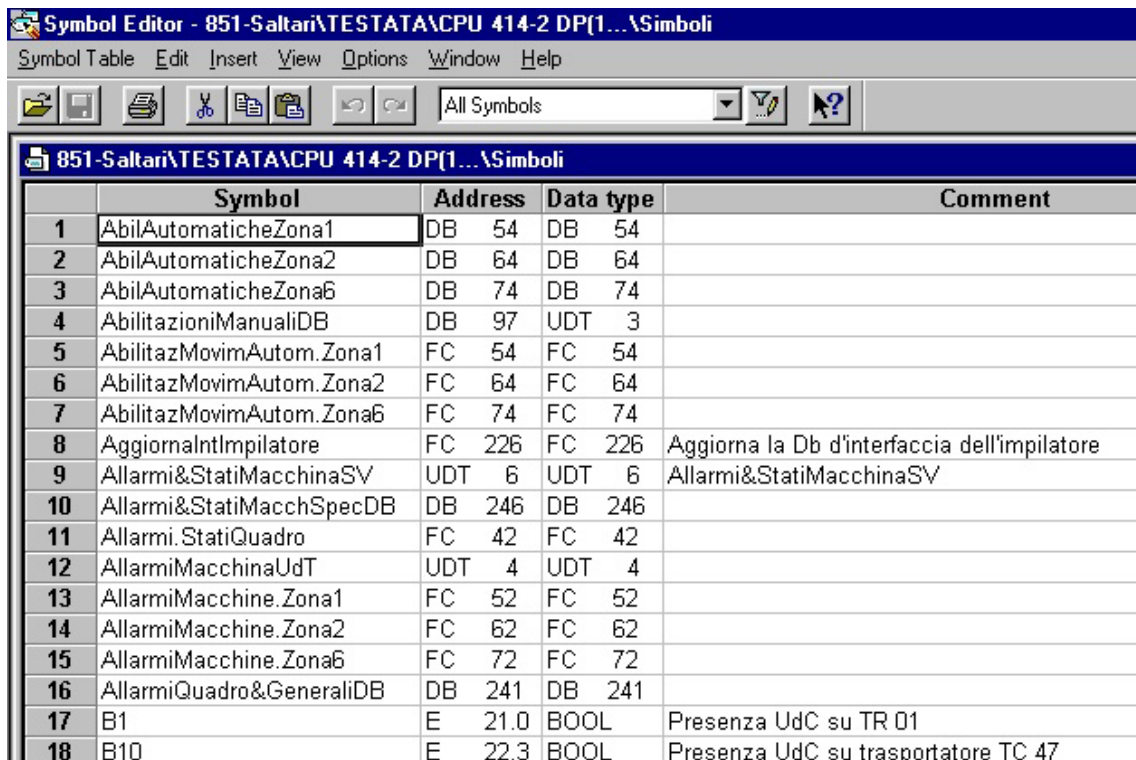


Figura 29 – Symbols

SIMATIC Manager richiama l'*editor dei simboli*, un'applicazione simile ad un foglio di calcolo: nella prima colonna bisogna inserire il simbolo, nella seconda l'indirizzo assoluto al quale il

simbolo si riferisce, nella terza il tipo dell'operando o del blocco e nella quarta colonna si può inserire un commento.



	Symbol	Address	Data type	Comment
1	AbilAutomaticheZona1	DB 54	DB 54	
2	AbilAutomaticheZona2	DB 64	DB 64	
3	AbilAutomaticheZona6	DB 74	DB 74	
4	AbilitazioniManualiDB	DB 97	UDT 3	
5	AbilitazMovimAutom.Zona1	FC 54	FC 54	
6	AbilitazMovimAutom.Zona2	FC 64	FC 64	
7	AbilitazMovimAutom.Zona6	FC 74	FC 74	
8	AggiornaIntImpilatore	FC 226	FC 226	Aggiorna la Db d'interfaccia dell'impilatore
9	Allarmi&StatiMacchinaSV	UDT 6	UDT 6	Allarmi&StatiMacchinaSV
10	Allarmi&StatiMacchSpecDB	DB 246	DB 246	
11	Allarmi.StatiQuadro	FC 42	FC 42	
12	AllarmiMacchinaUdT	UDT 4	UDT 4	
13	AllarmiMacchine.Zona1	FC 52	FC 52	
14	AllarmiMacchine.Zona2	FC 62	FC 62	
15	AllarmiMacchine.Zona6	FC 72	FC 72	
16	AllarmiQuadro&GeneraliDB	DB 241	DB 241	
17	B1	E 21.0	BOOL	Presenza UdC su TR 01
18	B10	E 22.3	BOOL	Presenza UdC su trasportatore TC 47

Figura 30 - Symbol Editor

STEP 7 permette anche di editare i simboli utilizzando Excel, salvando il file in formato *.dif*, per poter importare il file con l'editor dei simboli basta selezionare l'opzione **Symbol Table**→**Import...** In maniera analoga, con **Symbol Table**→**Export...**, è possibile salvare in un file esterno al progetto, i simboli utilizzati.

Anche con l'editor AWL, KOP e FUP si può assegnare un simbolo ad un indirizzo assoluto, selezionandolo e utilizzando l'opzione **Edit**→**Symbols...**

Vediamo, ora, cosa succede in un programma S7 quando si cambiano alcuni nomi simbolici.

Supponiamo di effettuare i seguenti cambiamenti:

<i>Prima</i>		<i>Dopo</i>	
Nome Simbolico	Indirizzo Assoluto	Nome Simbolico	Indirizzo Assoluto
B23	E0.0	B45	E0.0
B45	E0.1	B23	E0.1

Se in un segmento di codice AWL, per esempio, è presente l'istruzione:

U "B23"

quando si effettua il cambio del nome simbolico possono succedere due cose:

1. l'editor assegna una priorità maggiore all'indirizzo assoluto, quindi nel codice troveremo l'istruzione U "B45", perché si farà sempre riferimento all'ingresso E0.0.
2. L'editor assegna una priorità maggiore al nome simbolico, quindi nel codice l'istruzione rimarrà U "B23", ma in questo caso viene fatto riferimento all'ingresso con indirizzo E0.1.

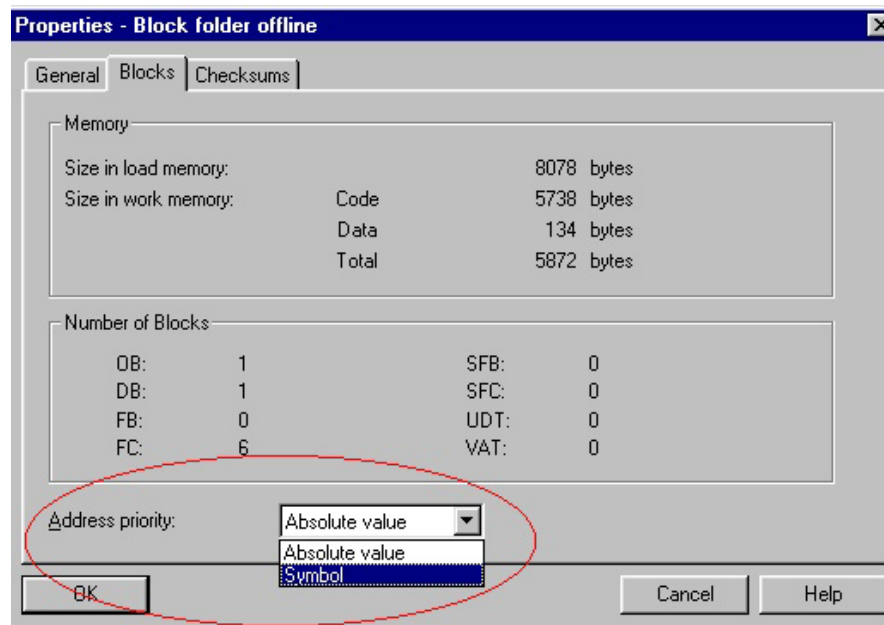


Figura 31 – Blocks→Object Properties

È possibile scegliere se dare una priorità maggiore ai nomi simbolici oppure agli indirizzi assoluti selezionando la cartella *Blocks* e l'opzione **Edit→Object Properties...** (vedi fig. 31).

### 3.5 – Il simulatore di controllori S7-PLCSIM e le tabelle di variabili (VAT)

Il simulatore di PLC permette di testare i programmi scritti anche quando non è disponibile un controllore reale da collegare al PC.

È possibile avviare S7-PLCSIM dalla cartella *Simatic→STEP 7* del menù avvio, oppure premendo il pulsante in SIMATIC  Manager.

Una volta avviata l'applicazione si ha a disposizione un vero e proprio PLC Siemens:

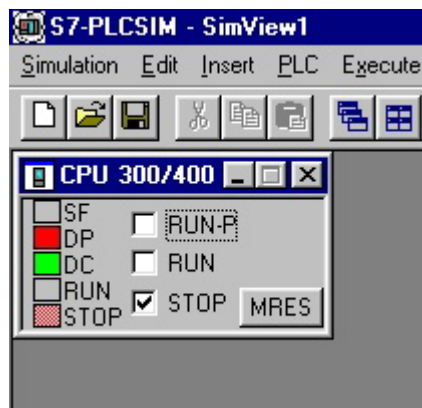



Figura 32 – S7-PLCSIM

Per poter testare il codice, quindi, questo deve essere prima caricato nel controllore selezionando la cartella *Blocks* e l'opzione **PLC→Download** oppure il tasto .

Una volta mandato in run il PLC è possibile visualizzare i blocchi di programma S7 in modalità *on-line*.

È in questa modalità che è possibile vedere i valori attuali dei parametri all'interno delle DB.

Per i blocchi di codice, la modalità *on-line* permette di visualizzare il valore contenuto nei registri interni del processore in corrispondenza di ogni istruzione eseguita:


RLO	STA	STANDARD	DB1
0	0	0	--
0	0	0	10
1	0	0	10
1	1	0	10
0	0	0	50
0	0	0	10
0	0	0	50
0	0	0	10

Figura 33 - FC on-line

RLO è il bit RLC del registro di stato, mentre STANDARD è ACCU1.

Premendo il tasto destro nella finestra di destra compare un menù a tendina che permette di aggiungere o rimuovere i registri di cui si vuole visualizzare il contenuto; è anche possibile scegliere la rappresentazione dei dati (binaria, esadecimale, decimale, ecc.).

Il simulatore stesso mette a disposizione delle finestre attraverso le quali è possibile visualizzare il valore dei parametri di una DB, dei temporizzatori, dei contatori, dei registri interni, ecc. Per inserire queste finestre bisogna utilizzare il menù **Insert** oppure la barra di pulsanti evidenziata in fig. 34.

Con il pulsante si  può far eseguire al PLC uno scan alla volta.

Un altro strumento utile per la prova ed il debug dei programmi sono le *tabelle di variabili* (VAT). Anche queste tabelle servono per visualizzare gli operandi in memoria.

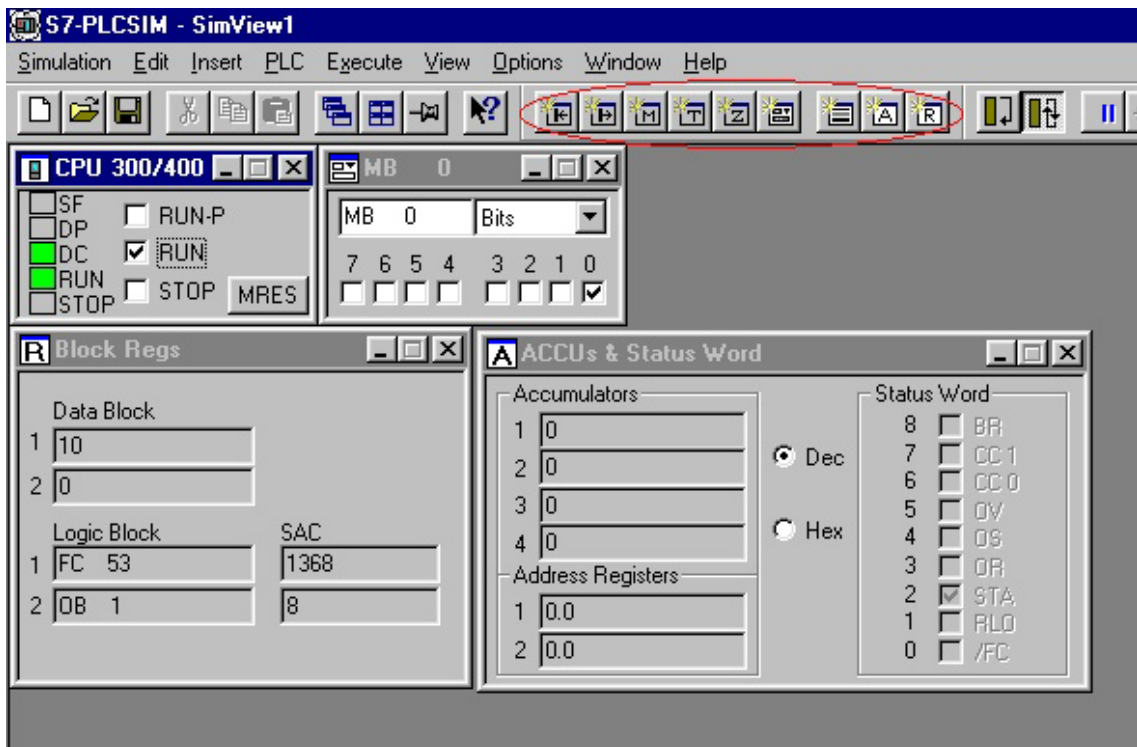


Figura 34 - S7-PLCSIM, finestre di visualizzazione dei dati

Una volta inserita una VAT nella cartella *Blocks*, selezionando **Insert→S7 Block→6 Variable table** da SIMATIC Manager, si potrà aprire facendo doppio click sull'icona.

A questo punto verrà richiamata l'applicazione *Monitoring and modifying variables*, che visualizza la VAT come un foglio di calcolo nel quale è possibile inserire l'indirizzo della variabile da visualizzare. Nelle altre colonne della tabella vengono riportati il simbolo della variabile, il formato di rappresentazione, il valore attuale ed un eventuale valore da forzare (vedi fig. 36).

Per poter forzare dei valori, bisogna collegare la VAT al PLC o al simulatore che esegue il programma selezionando l'opzione **PLC→Connect to**, dopodichè bisogna passare on-line

(Variable→Monitor), e forzare il nuovo valore premendo il pulsante oppure selezionando Variable→Modify.

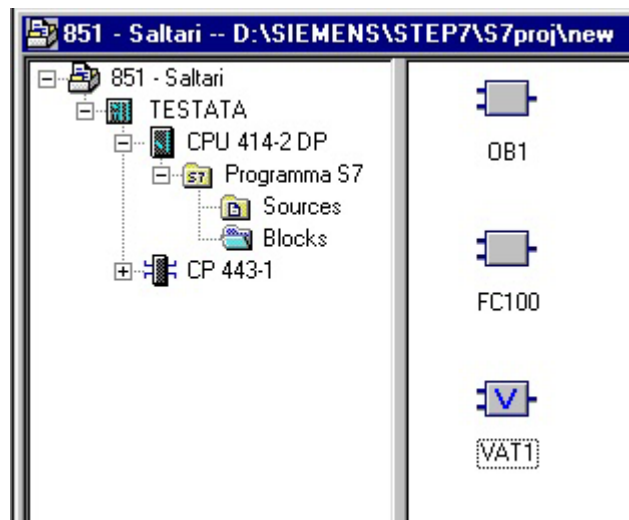


Figura 35 - Tabella di variabili

Mentre con le finestre di visualizzazione del simulatore è possibile modificare i valori delle variabili anche se il PLC è in RUN, per forzare un valore con una VAT bisogna assicurarsi che il PLC sia nella modalità RUN-P.

Ricordiamo, infine, che non è possibile forzare i valori degli ingressi del controllore attraverso le VAT. Esistono tre modi per modificare gli ingressi di un controllore: agire fisicamente sui moduli I/O della stazione, utilizzare il simulatore, oppure prevedere una funzione ad hoc che vada a scrivere sull'immagine di processo degli ingressi.

Address	Symbol	Monitor Format	Monitor Value	Modify Value
DB50.DBW 0	---	DEC	8	38
M 0.0	---	BIN	2#0	

Figura 36 - Monitoring and modifying variables



## Riferimenti

- [1] [http://www.ad.siemens.de/meta/index\\_76.htm](http://www.ad.siemens.de/meta/index_76.htm), sito ufficiale dei prodotti per l'automazione Siemens.
- [2] SIMATIC S7 – Primi passi ed esercitazioni con STEP 7 V5.0, 6ES7810-4CA04-8EA0, 1998 Siemens AG.
- [3] SIMATIC S7 – Programmazione con STEP 7 V5.0, 6ES7810-4CA04-8EA0, 1998 Siemens AG.
- [4] SIMATIC S7 – Lista istruzioni (AWL) per S7-300/400, 6ES7810-4CA04-8ER0, 1998 Siemens AG.
- [5] SIMATIC S7 – Schema a contatti (KOP) per S7-300/400, 6ES7810-4CA04-8ER0, 1998 Siemens AG.
- [6] SIMATIC S7 – Schema logico (FUP) per S7-300/400, 6ES7810-4CA04-8ER0, Siemens 1998 AG.
- [7] SIMATIC – SCL per S7-300/400 - Programmazione di blocchi, 6ES7811-1CA02-8EA0, 1998 Siemens.
- [8] SIMATIC – S7-PLCSIM per testare i programmi S7, C79000-G7072-C201, 1998 Siemens.