ISaGRAF

Versione 3.4

MANUALE UTENTE

CJ INTERNATIONAL

Stampato in Francia da CJ International.
3 Rue Hector Berlioz
F-38600 FONTAINE
Phone: 33 (0)4 76 26 87 30
Fax: 33 (0)4 76 26 87 39

Versione italiana a cura di FAR SYSTEMS Spa

Sommario

A. Manuale utente

A.1 Per iniziare

Questo capitolo descrive l'installazione dell'ambiente di lavoro ISaGRAF. Comprende anche un breve esempio di applicazione ISaGRAF, che permette all'utente di valutare le principali caratteristiche e di iniziare immediatamente ad usare ISaGRAF.

A.1.1 Installazione di ISaGRAF

Questo capitolo descrive l'installazione dell'ambiente di lavoro ISaGRAF e la configurazione del computer necessaria per lo sviluppo di applicazioni.

Requisiti hardware e software

E' possibile installare ISaGRAF su qualunque computer in grado di far girare Windows Versione 3.1. Per lo sviluppo di applicazioni, si raccomanda, tuttavia, la seguente configurazione hardware:

- Un personal computer dotato di microprocessore 80486 o superiore (raccomandato un processore pentium)
- 8 megabytes di memoria convenzionale ed estesa (raccomandati 16 Mb)
- Un disk drive da 3.5-pollici (1.44 megabyte)
- Un hard disk con almeno 20 megabytes spazio disponibile
- Una scheda grafica VGA o SVGA con un monitor compatibile
- Un mouse (necessario per gli strumenti di sviluppo in modalità grafica)
- Una porta parallela LPT1 (necessaria per la chiave di protezione)

Uno dei seguenti sistemi operativi deve essere già presente, prima di poter procedere con l'installazione dell'ambiente di lavoro ISaGRAF:

- Windows Versione 3.1 funzionante in modalità 386 avanzata
- Windows 95
- Windows NT Versione 3.51 o 4.00

Il programma di installazione

L'ambiente di lavoro ISaGRAF viene installato da INSTALL, il programma di installazione ISaGRAF. Tale programma provvede a trasferire il software ISaGRAF dai dischetti al disco rigido. INSTALL provvede inoltre a creare il gruppo «ISaGRAF» nella finestra Gestione Risorse (Program Manager) ed il file di inizializzazione «ISA.ini» nella sottocartella **EXE** creata in fase di installazione.

INSTALL è un programma per Windows che deve quindi essere avviato dalla Gestione Risorse di Windows o dal comando Esegui del menù Avvio di Windows 95. Per installare ISaGRAF, si proceda come mostrato di seguito:

- Inserire il Disco 1 nel drive appropriato

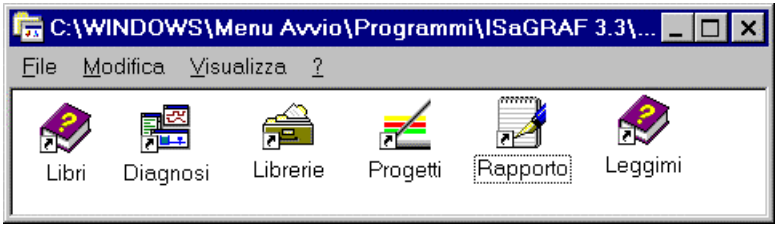
• Dalla Gestione Risorse (Program Manager), richiamare dal menù «File» la voce «Esegui»; immettere la linea di comando: «A:\INSTALL.exe». Alternativamente è possibile inserire il comando:
«WIN A:\INSTALL.exe» direttamente da DOS.
• Seguire le istruzioni *on-line* per completare l'installazione. Si consiglia di installare l'ambiente di lavoro ISaGRAF in una nuova cartella per evitare confusione con i file di eventuali altre versioni di ISaGRAF.

INSTALL chiederà quali dei seguenti componenti si vogliano installare:

• Programmi eseguibili ISaGRAF
• Informazioni on line e file guida
• Librerie standard ISaGRAF
• Applicazioni ISaGRAF d'esempio

Nel caso si installi ISaGRAF per la prima volta, si raccomanda vivamente di includere tutti i componenti. È comunque possibile aggiungere ulteriori componenti successivamente eseguendo nuovamente l'installazione di ISaGRAF.

Il nome predefinito per la cartella principale di ISaGRAF è «\ISAWIN». In questo modo è possibile installare facilmente ISaGRAF per Windows sullo stesso disco di una versione ISaGRAF per MS-DOS. Si faccia riferimento alla sezione «Cartelle ISaGRAF» del capitolo «Tecniche avanzate» per ulteriori informazioni sulla struttura dei file su disco. Al termine della copia dei file di ISaGRAF, verrà aggiunto alla Gestione Risorse (Program Manager) il seguente gruppo di icone file.



Queste sono le icone principali di ISaGRAF :

Projects:..... Gestione progetti
Libraries:..... Gestione librerie
Archive:..... Strumenti per archiviazione/ripristino
Book:..... Informazioni *on-line* su ISaGRAF
Diagnosis:..... Strumenti diagnostici per l'utente finale
Read Me:..... Informazioni relative alla nuova versione di ISaGRAF
Report:..... Modulo standardizzato per la segnalazione di errori

Qualora sorgano problemi, si compili il modulo di segnalazione errori. Lo si modifichi inserendo le informazioni richieste e lo si salvi richiamando dal menù File la voce *Save as* assegnandogli un nuovo nome. Si invii, poi, questo file alla CJ International tramite Fax o e-mail.

Aggiornamento file di sistema

Una volta completata l'installazione, si rende necessario, prima di riavviare il computer, aggiornare il file CONFIG.SYS. Non è necessario aggiungere alla variabile PATH il percorso della cartella ISaGRAF.
ISaGRAF non fa uso di alcuna variabile dell'ambiente MS-DOS. Si possono comunque aggiungere le seguenti righe al file CONFIG.SYS:

```
files=20
buffers=20
```

L'ambiente di lavoro ISaGRAF fa uso di una porta seriale per comunicare con il PLC ISaGRAF destinazione. La porta seriale predefinita per ISaGRAF è COM1. Se anche il mouse usa una porta seriale, è preferibile assegnare COM2 al mouse, in questo modo l'impostazione COM1 risulterà valida per ogni nuova applicazione ISaGRAF.

Dopo aver modificato il file CONFIG.SYS, è necessario riavviare il computer per rendere operative le nuove impostazioni.

⇒ **Importante per gli utenti di Windows NT:**
Per usare l'ambiente di lavoro sui sistemi Windows NT 3.51 o 4.00, è necessario inserire le seguenti righe nella sezione [WS001] del file ISA.ini contenuto nella cartella \ISAWIN\EXE:

```
[WS001]
NT=1
Isa=C:\ISAWIN
IsaExe=C:\ISAWIN\EXE
IsaApl=C:\ISAWIN\APL1
IsaTmp=C:\ISAWIN\TMP
```

Queste impostazioni sono assolutamente necessarie per le comunicazioni RS.

Chiave di protezione

Una chiave hardware impedisce le copie illegali del software ISaGRAF. Gran parte delle funzioni dell'ambiente di lavoro ISaGRAF sono comunque disponibili anche se la chiave non è inserita. Inoltre la chiave di protezione permette di definire le opzioni dell'ambiente di lavoro ISaGRAF e controlla la dimensione massima delle applicazioni sviluppabili. Qualora la chiave non sia inserita o lo sia in modo non corretto, alcune funzioni dell'ambiente di lavoro ISaGRAF non funzioneranno. Questa situazione deve essere considerata NORMALE. Per assicurarsi che la chiave sia correttamente inserita, richiamare la voce «**Informazioni su ...**» del menù «**Aiuto**» attivabile da qualsiasi finestra ISaGRAF. Verranno mostrate le opzioni rese disponibili dall'ambiente di lavoro ISaGRAF.

La chiave può essere connessa a qualunque porta parallela presente sul computer. Qualora la macchina metta a disposizione più porte parallele, è preferibile connettere chiave e stampante su porte differenti. In alcune configurazioni PC/stampante, la chiave potrebbe non essere riconosciuta nel caso fosse connessa ad una stampante in stato «off-line». In queste situazioni si proceda scollegando la stampante o ponendola in stato «on-line» e si avvii nuovamente l'ambiente di lavoro ISaGRAF.

Si noti che la chiave non è necessaria per l'ambiente di lavoro **ISaGRAF-32**.

⇒ **Importante per gli utenti di Windows NT:**
Su sistemi Windows NT , per rendere visibile la chiave di protezione deve essere installato il Driver Sentinel/Rainbow™, fornito su dischetto a parte.

A.1.2 Usare le informazioni *on-line*

Assieme all'ambiente di lavoro ISaGRAF vengono installate le informazioni *On-line* relative ai seguenti argomenti:
- Guida di riferimento ai linguaggi ISaGRAF
- Il manuale utente in forma completa (per ciascun strumento ISaGRAF)
- Note tecniche per gli elementi delle librerie

In ogni finestra ISaGRAF è presente il menù «**Aiuto**» che elenca le informazioni *on-line* relative alle caratteristiche principali (ad esempio i linguaggi) ed al programma che si sta utilizzando
In ISaGRAF è inoltre disponibile la tipica icona Windows di Guida in linea, per completare le risorse di aiuto in ISaGRAF

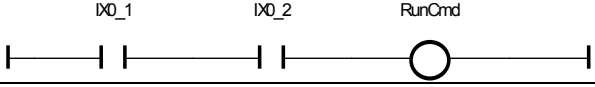
A.1.3 Esempio di un'applicazione.

Questo capitolo spiega, a passo a passo, tutte le operazioni basilari necessarie per creare, progettare, produrre e provare una piccola, ma completa applicazione multi-linguaggio.

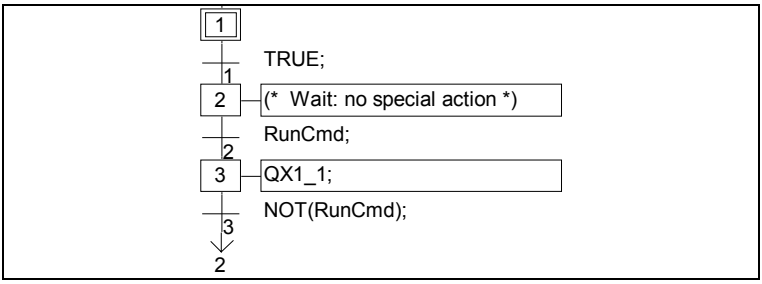
Di seguito sono riportate le specifiche complete di questa applicazione, che usa assieme le rappresentazioni in linguaggio LD e SFC:

Variabili Booleane:	
IX0_1, IX0_2:	variabili in ingresso per il processo command
RunCmd:	comando interno «run/stop»
QX1_1:	variabile di output: stato del processo

Programma Command:	Sezione Iniziale – linguaggio LD Interpretazione del comando interno «run/stop»



Programma RunStop:	Sezione sequenziale – Linguaggio SFC Controllo del processo



Avvio dell'ambiente di lavoro ISaGRAF

Un doppio click del mouse sull'icona **«Projects»** del gruppo «ISaGRAF», avvia l'esecuzione dell'ambiente di lavoro ISaGRAF. Viene aperta la finestra **Gestione progetti**.

Creare il progetto

Per creare il progetto (che chiameremo «RunStop»), richiamare dal menù **«File»** la voce **«Nuovo»** oppure utilizzare il pulsante **«Crea nuovo progetto»** dalla barra strumenti della finestra Gestione progetti.

Inserire il nome del progetto: «**RunStop**»
Selezionare la configurazione di I/O: «**Sim_Boo**»
Premere il pulsante «**OK**».
Il progetto è stato così creato.

Aprire il progetto

Per lavorare sui programmi di un progetto si usa la finestra **Programmi** di ISaGRAF. Per attivarla usare il comando **«File / Apri»** della finestra Gestione progetti o fare doppio click con il mouse sul nome del progetto o usare il pulsante **«Apri»** dalla barra strumenti.

Creare i programmi

La finestra Programmi è ora aperta e vuota (non contiene alcun programma). Per creare il primo programma, richiamare dal menù **«File»** la voce **«Nuovo»** o utilizzare il pulsante **«Crea nuovo programma»** dalla barra strumenti.

Inserire il nome del programma: «**Command**».
Selezionare il tipo di linguaggio: «**Quick LD**».
Selezionare la sezione: «**Iniziale**».
Premere il pulsante «**OK**» per creare il programma.

La stessa sequenza di operazioni deve essere ripetuta per il secondo programma richiamare dal menù **«File»** la voce **«Nuovo»** o utilizzare il pulsante **«Crea nuovo programma»** dalla barra strumenti:

Inserire il nome del programma: «**RunStop**».
Selezionare il tipo di linguaggio: «**SFC**».
Selezionare la sezione: «**Sequenziale**».
Premere il pulsante «**OK**» per creare il programma.

I programmi sono stati creati e sono visibili nella finestra Programmi.

Dichiarazione delle variabili

Per procedere con la stesura dei programmi, è necessario prima dichiarare le variabili che si intendono usare nella programmazione. Per far questo richiamare dal menù **«File»** la voce **«Dizionario»** oppure utilizzare il pulsante **«Dizionario»** dalla barra strumenti. Le variabili di I/O vengono automaticamente dichiarate alla creazione del progetto.

La finestra del dizionario, adesso, è aperta. Dal menù **«File»**, dal sottomenù **«Altro»**, richiamare la voce **«Variabili globali / Booleani»** per attivare il dizionario Booleani globali. Lo stesso risultato si ottiene con il pulsante **Oggetti globali** e selezionando la scheda **Booleani**.

Per creare nuove variabili booleane richiamare dal menù **«Modifica»** la voce **«Nuovo»** o, in alternativa, il pulsante **«Inserisci oggetti»** sulla barra strumenti. Inserire nella finestra di dialogo la descrizione della variabile interna:

Nome:	**RunCmd**
Commento:	**comando Run/Stop: interno**
Attributi:	Selezionare l'attributo di variabile «**Interna**»

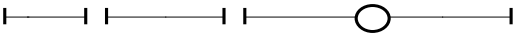
Premere il pulsante «**Memorizza**»: viene creata la variabile.
Premere il pulsante «**Annulla**» per uscire dalla finestra di dialogo.
Si esce, infine, dalla finestra del dizionario e si salvano le modifiche apportate: Menù «**File**» voce «**Esci**». Indicare **«Sì»** per salvare le modifiche correnti.

Modificare il programma in Quick LD

Per iniziare a modificare il programma LD «Command», fare doppio click sul nome del programma nella finestra Programmi o usare il pulsante **«Apri»** dalla barra strumenti.

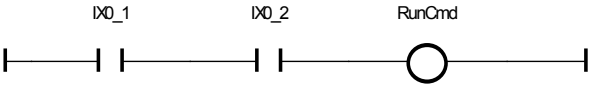
Ora è aperta la finestra di modifica per ISaGRAF Quick LD. Per aumentare lo spazio di lavoro, dimensionare la finestra a tutto schermo.

F2 F3 Premere i tasti F2 e F3:



INVIO Per associare le variabili ai simboli LD: muovere il cursore usando i tasti frecce. Posizionare il cursore su ciascun simbolo e premere il tasto INVIO. La finestra di dialogo **Selezione variabile** viene aperta. Selezionare **Validità** globale e tipo di variabile **Booleano**.
Per il primo contatto, inserire nella casella di selezione variabile: IX0_1 e premere INVIO.
Per il secondo contatto, inserire nella casella di selezione variabile: IX0_2 e premere INVIO.
Per la bobina, inserire nella casella di selezione variabile: RunCmd e premere INVIO.

Il programma adesso è completato. Ecco il risultato:



Uscire dall'editor e salvare le modifiche apportate: Menù «**File**», voce «**Esci**». Selezionare «**SI**» per salvare le modifiche correnti.

Modificare il programma SFC

Per iniziare la modifica del programma SFC «RunStop», fare doppio click sul nome del programma nella finestra Programmi o usare il pulsante **«Apri»** dalla barra strumenti.

Ora è aperta la finestra dell'editor SFC. Per aumentare lo spazio di lavoro, dimensionare la finestra a tutto schermo.

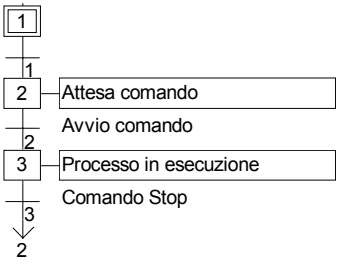
Il passo iniziale è già pronto e selezionato. Premere il tasto freccia **«↓«** della tastiera per selezionare una cella vuota successiva al passo iniziale (0,1)

F4 F3 Premere F4 quindi F3 per inserire prima una transizione poi un passo.

F4 F3 Premere F4 quindi F3 per inserire un'altra transizione e un altro passo.

F5 Premere F5 per inserire un salto ad un passo e selezionare GS2 come destinazione del salto.

Il diagramma è ora completo. Premere il pulsante **«Zoom»** nella barra strumenti per incrementare le dimensioni delle celle e ottenere spazio per visualizzare i riquadri per le istruzioni di livello 2. Il diagramma è qui rappresentato:



Per inserire la programmazione della transizione numero 2, selezionarla utilizzando i tasti freccia della tastiera e premere il tasto INVIO. A fianco del diagramma viene aperta la finestra di programmazione di Livello 2. Immettere la programmazione di Livello 2 per la transizione:

RunCmd;

^TAB Premere i tasti «CTRL + TAB» per riposizionarsi (dalla finestra di programmazione di Livello 2) sul diagramma SFC, quindi spostare la selezione sul passo numero 3, e premere il tasto INVIO per editare il testo di programmazione di Livello 2 del passo 3:

QX1_1;

Allo stesso modo si immette il testo di programmazione di Livello 2 delle transizione 3:

Not (RunCmd);

^F4 Premere i tasti «CTRL + F4» per chiudere la finestra di Livello 2.
Il programma SFC è ora completo. Uscire dall'editor richiamando dal menù «**File**» la voce «**Esci**», e salvare le modifiche facendo click su «**SI**».

Compilare il codice dell'applicazione

Per compilare il codice dell'applicazione richiamare dal menù «**Compila**» la voce «**Compila applicazione**» o premere il pulsante «**Compila applicazione**» sulla barra strumenti.

Al completamento della generazione del codice, appare una finestra di dialogo, nella quale viene chiesto se si voglia uscire (pulsante **«Exit»**) o continuare (pulsante **«Continue»**) con la compilazione del codice: Premere il pulsante «**Exit**». Con la compilazione di un progetto viene creato il codice dell'applicazione che andrà successivamente caricata sul PLC target.

Simulazione

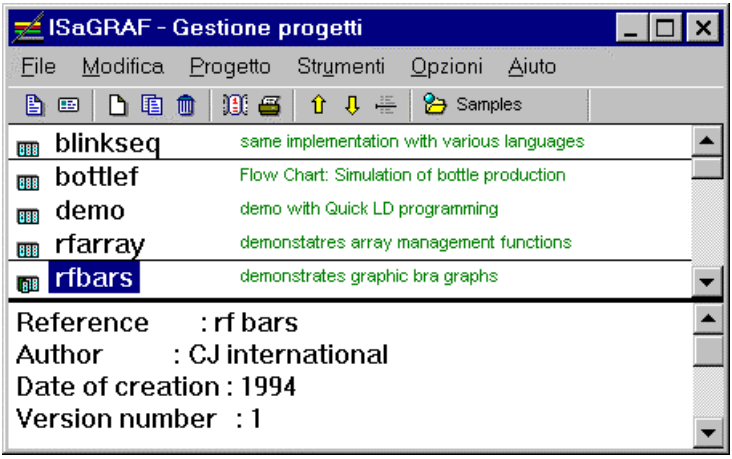
Per avviare il simulatore di kernel ISaGRAF, richiamare dal menù «**Debug**» la voce «**Simulazione**» dalla finestra Programmi o premere il pulsante «**Simulazione**» sulla barra strumenti.

Quando appare la finestra del simulatore (di ingressi e uscite), l'applicazione può essere sottoposta alle prove. In questo esempio, devono essere premuti entrambi gli ingressi 1 e 2 (pulsanti verdi) per avviare il processo (le uscite sono rappresentate dalle luci rosse).

Per chiudere la sessione di simulazione, selezionare la finestra Debugger e richiamare dal Menù «**File**» la voce «**Esci**».

A.2 Gestione dei progetti

Per lanciare la gestione progetti di ISaGRAF, fare doppio click con il mouse sull'icona **«Projects»** del gruppo ISaGRAF. Si apre la finestra **Gestione progetti**.



Un progetto viene eseguito in modo ciclico sul PLC destinazione. La parte superiore della finestra elenca i progetti esistenti. La descrizione del progetto selezionato è visibile, invece, nella porzione inferiore della finestra.

Dimensionare le finestre

Basta un click sul separatore (divisore) posto tra l'elenco dei progetti e la descrizione per dimensionare la finestra corrispondente. La finestra con la descrizione non può essere nascosta completamente. Conterrà sempre almeno una riga di testo.

Inserire dei separatori

È possibile inserire una linea di separazione prima del nome di ogni progetto. Questo permette di raggruppare nella lista visualizzata alcuni progetti collegati ad una stessa applicazione. Richiamare il comando «**Modifica / Metti/Togli separatore**» per inserire o eliminare un separatore posto prima del progetto selezionato.

Spostare i progetti all'interno dell'elenco

Per spostare un progetto nell'elenco, per prima cosa selezionarlo, evidenziandolo. Poi fare click sul nome e, tenendo premuto il mouse, trascinarlo in una nuova posizione nella lista. Durante lo spostamento del progetto, una piccola freccia sul margine sinistro indica dove verrà posizionato. È anche possibile richiamare il comando «**Sposta in su nella lista**» o «**Sposta in giù nella lista**» dal menù

«**Modifica**» per spostare il progetto evidenziato una riga per volta. Da notare che un separatore posto prima del progetto, viene spostato assieme al progetto.

A.2.1 Creare ed usare i progetti

Le voci del menù della Gestione progetti permettono di creare nuovi progetti, modificarli e gestire progetti esistenti.

Creare un nuovo progetto

Per creare un progetto richiamare la voce **«Nuovo»** dal menu **«File»** della finestra di Gestione progetti, oppure utilizzare il pulsante **«Crea nuovo progetto»** dalla barra strumenti. Ad ogni nuovo progetto bisogna prima assegnare un nome. A questo punto viene creato un progetto vuoto, privo di oggetti. È possibile collegare una configurazione di I/O al nuovo progetto. Tale configurazione di I/O deve essere definita nella libreria. Quando viene scelta una configurazione, ISaGRAF provvede automaticamente ad impostare le connessioni di I/O ed a dichiarare le corrispondenti variabili di I/O nel dizionario del nuovo progetto. Per assegnare o cambiare il nome ad un progetto, bisogna attenersi alle seguenti regole:

- il nome non può eccedere gli **8** caratteri
- il primo carattere deve essere una **lettera**
- i caratteri successivi possono essere **lettere**, **numeri** o il carattere «_»
- non c'è distinzione tra maiuscole e minuscole nel nome del progetto

Quando il progetto è stato creato, richiamare il comando «**Modifica / Imposta commento**» per inserire il testo che verrà visualizzato nella lista assieme al nome del progetto.

Modificare il testo descrittivo del progetto

Per modificare il testo descrittivo del progetto richiamare il comando «**Progetto / Descrizione progetto**». Questo documento serve a distinguere il progetto dagli altri nell'elenco dei progetti. Il testo descrittivo può anche essere usato per conservare le annotazioni relative al progetto.

Modificare il progetto

Il comando «**File / Apri**» apre la finestra Programmi del progetto evidenziato. Tale finestra permette di accedere a tutti i contenuti (programmi, parametri dell'applicazione…) del progetto. Alternativamente è possibile fare doppio click con il mouse sul nome del progetto nella lista.

L'elenco cronologico delle modifiche

Il sistema ISaGRAF gestisce un file contenente l'elenco cronologico di tutte le modifiche apportate ai componenti di un progetto. Ogni modifica nell'elenco è composta da un titolo, una data e da un orario. L'elenco cronologico mantiene le ultime **500** modifiche. Esiste un elenco cronologico per ciascun progetto. L'elenco cronologico del progetto è di complemento ai file «Diario» collegati ai programmi del progetto. Il comando «**Progetto / Cronologia**» permette all'utente di visualizzare e successivamente stampare l'elenco cronologico delle modifiche relativo al progetto evidenziato. L'utente può evidenziare uno o più riferimenti nella lista principale, e premere i seguenti pulsanti:

OK.....chiude la finestra
Stampa.....invia i contenuti della lista alla stampante
[Cancella] Selezione.....elimina dalla lista i riferimenti evidenziati
[Cancella] Tutto.....svuota completamente la lista
Trova.....trova un testo nella lista

Il testo da cercare viene inserito nella casella **Cerca**, sopra il pulsante «**Trova**». Questa funzione non distingue tra maiuscolo e minuscolo. Quando la ricerca arriva in fondo alla lista, riprende dall'inizio proseguendo fino alla posizione inizialmente selezionata.

Stampare un documento completo

La voce «**Progetto / Stampa**» permette all'utente di costruire e stampare un documento completo relativo al progetto evidenziato. Tale documento può contenere ogni componente (programma, variabile, parametri…) del progetto evidenziato. Per costruire un documento con contenuti specifici (non completo come il documento predefinito), l'utente deve solamente definirne la lista delle voci.

Password di protezione

La voce «**Progetto / Imposta password**» permette di definire una password di protezione per strumenti e dati del progetto evidenziato. Per ulteriori dettagli su livelli di password e protezione dei dati, fare riferimento alla sezione «**Password di protezione**» alla fine della prima parte di questo manuale. Le password sono relative solamente al progetto evidenziato. Non riguardano altri progetti o librerie ISaGRAF.

A.2.2 Lavorare con più gruppi di progetti

Ad un progetto ISaGRAF corrisponde sul disco una cartella che contiene tutti i file del progetto. Un gruppo di progetti corrisponde ad un elenco di cartelle di progetti raggruppate assieme sotto la stessa cartella radice. Un gruppo di progetti viene identificato da un nome. Per default, ISaGRAF crea due gruppi progetto:

«**Default**» in «\ISAWIN\APL» l'area di lavoro dell'utente.
«**Samples**» in «\ISAWIN\SMP» applicazioni di esempio distribuite assieme a ISaGRAF.

Il nome del gruppo progetti selezionato correntemente è scritto sulla barra strumenti, vicino al pulsante **Selezione gruppo progetti** usato per selezionare un gruppo di progetti:

Il comando «**File / Selezione gruppo progetti**» permette di selezionare un gruppo esistente oppure di crearne uno nuovo. Viene aperta la seguente finestra di dialogo:

Selezionare un gruppo nella lista e premere «**Seleziona**» per attivare tale gruppo nella finestra Gestione progetti. Per selezionare un gruppo si può anche fare un doppio click sul suo nome. Utilizzare il pulsante «**Nuovo gruppo**» per creare un gruppo nuovo. Questo comando può essere utilizzato sia per assegnare un nome di gruppo ad una cartella esistente, sia per creare un nuovo gruppo con una nuova cartella.

Nessun gruppo può essere selezionato o creato quando sono aperte altre finestre ISaGRAF (Programmi, editor...).

A.2.3 Opzioni

Le voci del menù «**Opzioni**» consentono di visualizzare o nascondere la barra strumenti, indicare il font di caratteri per le parti di testo ed abilitare la modalità «chiusura automatica» della finestra Gestione progetti. Il font di caratteri viene usato per mostrare il testo descrittore del progetto ed in tutti gli editor di testo ISaGRAF.

L'opzione «**Mantieni aperta Gestione progetti**» una volta disabilitata fa si che la finestra Gestione progetti venga automaticamente chiusa all'apertura di un progetto.

A.2.4 Strumenti

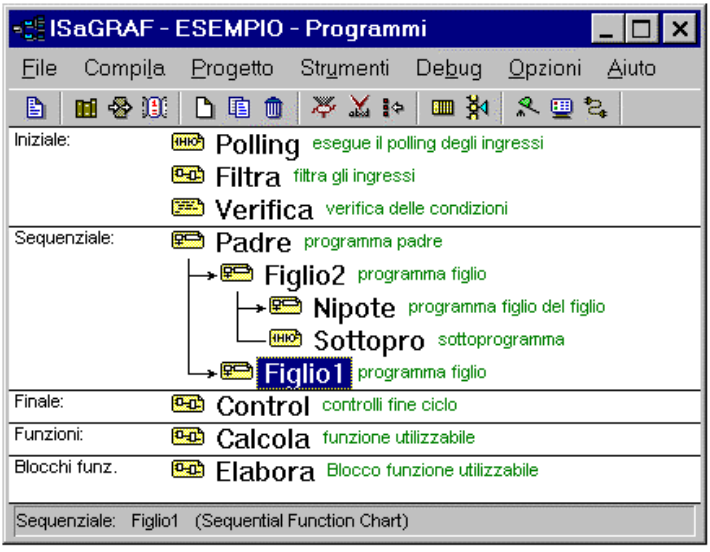
I comandi del menù «**Strumenti**» sono usati per lanciare altre applicazioni ISaGRAF. Il comando «**Strumenti / Archivio / Progetti**» lancia la gestione archivio di ISaGRAF per salvare o ripristinare progetti. Il comando «**Strumenti / Archivio / Dati comuni**» viene utilizzato per salvare o recuperare file usati da tutti i progetti (come i nomi definiti con campo di validità **comune** a tutti i progetti).

Il comando «**Strumenti / Librerie**» lancia la finestra **Librerie** di ISaGRAF.

Il comando « **Strumenti / Importa programma IL**» può essere utilizzata per importare un progetto descritto come un singolo programma IL in un file testo, in accordo con il formato di scambio file «PLC Open».

A.3 Gestione dei programmi

La finestra **Programmi**, mostra i programmi (chiamati anche moduli o unità di programmazione) dell'applicazione e mette a disposizione, nel proprio menù, le procedure disponibili per creare l'architettura del progetto, avviare gli editor, il compilatore ed il debugger. Questa finestra costituisce il centro dell'ambiente di lavoro, nella fase di sviluppo di un'applicazione. Si accede alla finestra Programmi aprendo un progetto della finestra Gestione progetti con il comando «**File / Apri**».



A.3.1 Le componenti di un progetto

Le componenti di un progetto vengono chiamate programmi. Un programma è un'entità logica che descrive una parte del controllo di esecuzione.
Le variabili di un progetto vengono definite nel dizionario del progetto. Le variabili di un progetto con campo di validità **globale** (come ad esempio le variabili di I/O) possono essere usate da qualsiasi programma appartenente al progetto. Le variabili con campo di validità **locale** possono essere usate da un unico programma del progetto.
I programmi sono organizzati in una **struttura gerarchica ad albero**, suddivisa in differenti **sezioni logiche** (Iniziale, Sequenziale, Finale). La finestra Programmi mostra tali sezioni ed i collegamenti gerarchici.

- ***Programmi di primo livello***

I programmi di **primo livello** in ciascuna sezione appaiono sul lato sinistro della struttura ad albero gerarchico. I programmi di primo livello delle prime tre sezioni (Iniziale, Sequenziale e Finale) sono sempre attivi, ad ogni ciclo di esecuzione del progetto vengono eseguiti nel seguente ordine:

- (Lettura degli input)
- Esecuzione dei programmi di primo livello della sezione **INIZIALE**
- Esecuzione dei programmi di primo livello della sezione **SEQUENZIALE**
- Esecuzione dei programmi di primo livello della sezione **FINALE**
- (Aggiornamento degli output)

I programmi delle sezioni «**Iniziale**» o «**Finale**» descrivono operazioni cicliche che non dipendono dal tempo. I programmi della sezione «**Sequenziale**» descrivono operazioni di tipo sequenziale, in cui la variabile tempo viene esplicitamente impiegata per distinguere le operazioni eseguite. I programmi principali della sezione «**Iniziale**» vengono sistematicamente eseguiti all'inizio di ogni ciclo di esecuzione. I programmi principali della sezione «**Finale**» vengono sistematicamente eseguiti al termine di ogni ciclo di esecuzione. I programmi principali della sezione «**Sequenziale**» vengono eseguiti in base alle regole di programmazione **SFC** o **FC** e devono essere scritti in linguaggio **SFC** o **FC**. I programmi delle sezioni cicliche non possono essere espressi in linguaggio **SFC** o **FC**. Ogni programma di ciascuna sezione può contenere, a sua volta, uno o più **sottoprogrammi** (questi ultimi sono dedicati esclusivamente al loro programma padre e non possono essere scritti in linguaggio **SFC** o **FC**).
I programmi principali **SFC** ed **FC** della sezione Sequenziale possono contenere (oltre che dei sottoprogrammi) dei **programmi figli** che, come vedremo, sono anch'essi dedicati al loro programma padre, ma sono scritti rispettivamente in linguaggio **SFC** e **FC**.

Funzioni e blocchi funzione

Una **funzione** rappresenta un algoritmo che restituisce un unico valore in uscita a partire da più valori in ingresso. Una funzione utilizza solamente variabili temporanee, che vengono cancellate tra una chiamata e l'altra.
A differenza di una funzione, un **blocco funzione** associa un algoritmo che opera su variabili in ingresso con dati di tipo statico non visibili, i quali vengono copiati (istanziati) dal sistema per ogni differente uso di un blocco funzione.
Questo implica che una funzione non dovrebbe mai chiamare un blocco funzione.
I programmi della sezione «**Funzioni**» (a differenza dei sottoprogrammi di un programma padre) possono essere chiamati da ogni programma di ogni sezione del progetto. Un programma della sezione «**Funzioni**» non può venire scritto in linguaggio **SFC** o **FC**.
I programmi della sezione «**Blocchi funzione**» possono essere chiamati da ogni programma di ogni sezione del progetto, ma non possono essere programmati in linguaggio **SFC** o **SFC**.

Sottoprogrammi

Un sottoprogramma è una funzione dedicata ad un programma padre (scritto in qualsiasi linguaggio: SFC, FC od altro). Un sottoprogramma può essere eseguito (chiamato) solamente dal proprio programma padre. Ogni programma di ciascuna

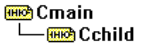
sezione può avere uno o più sottoprogrammi. Un sottoprogramma può essere scritto in ogni linguaggio eccetto il linguaggio **SFC** e **FC**.

Programmi figli SFC e FC

Un **programma figlio SFC** è un programma parallelo che può essere avviato e terminato dal proprio programma genitore. Sia il programma genitore che il programma figlio devono essere scritti utilizzando il linguaggio **SFC**. Un programma genitore che avvia un programma **SFC** figlio, inserisce un **token SFC** in ciascun passo iniziale del programma figlio. Un programma genitore che pone termine ad un programma **SFC** figlio, rimuove tutti i token esistenti nei passi del programma figlio.
I programmi figli **FC**, a differenza dei programmi figli **SFC**, non rappresentano operazioni parallele. Nessun programma **FC** della sezione Sequenziale può controllare dei programmi figli **FC**. Un programma **FC** padre viene bloccato (stato di attesa) durante l'esecuzione di un suo programma figlio **FC**. Non è possibile eseguire operazioni simultanee in un programma **FC** padre ed in un suo programma figlio.
(Il programma figlio **FC** si comporta come un sottoprogramma).

Collegamenti tra programmi e sottoprogrammi:

I sottoprogrammi e i programmi figli sono collegati nella struttura gerarchica ad albero ai rispettivi programmi genitori da una linea. Un collegamento tra un programma **SFC** genitore ed un programma **SFC** figlio termina con una freccia. Un collegamento di questo tipo serve a rappresentare le operazioni in **parallelo**.



Esempio del collegamento tra un programma genitore ed un sottoprogramma.

Esempio del collegamento tra un programma genitore SFC ed un sottoprogramma (obbligatoriamente non SFC, nel caso specifico in linguaggio FBD).

Esempio del collegamento tra un programma SFC genitore ed un programma figlio SFC.

Esempio del collegamento tra un programma genitore SFC, un programma SFC figlio ed un sottoprogramma (obbligatoriamente non SFC, nel caso specifico FBD).

Linguaggi di programmazione

Ogni programma può essere scritto esclusivamente in un solo **linguaggio**. Tale linguaggio, scelto alla creazione del programma non può essere cambiato in un secondo tempo. Tuttavia, i diagrammi **FBD** possono contenere parti in **LD**, ed i diagrammi in **LD** possono contenere chiamate a blocchi funzione. I linguaggi disponibili con supporto grafico sono: **SFC** (Sequential Function Chart), **FC** (Flow Chart), **FBD** (Functional Block Diagram) ed **LD** (Ladder Diagram). I linguaggi con supporto testuale sono: **ST** (Structured Text) e **IL** (Instruction List). I linguaggi **SFC** e **FC** sono riservati per i programmi principali e figli della sezione **Sequenziale**.

Nella finestra Programmi, un'icona posta a fianco del programma indica il tipo di linguaggio usato. Di seguito sono elencate le icone usate per rappresentate i linguaggi:

 SFC ..... Sequential Function Chart
 FC ..... Flow Chart
 FBD ..... Functional Block Diagram
 LD..... Ladder Diagram (immesso con l'editor Quick LD)
 ST..... Structured Text
 IL ..... Instruction List

A.3.2 Lavorare con i programmi

Il menù «**File**» contiene tutti i comandi atti a creare, aggiornare o modificare i programmi. Permette, inoltre, di avviare gli editor per la modifica dei programmi dell'applicazione.

Creare un nuovo programma

Il comando «**Nuovo**» del menù «**File**» permette la creazione, in ognuna delle sezioni, di un programma di primo livello, di un programma figlio o di un sottoprogramma.
La prima informazione da inserire è costituita dal nome del nuovo programma, secondo le seguenti regole:

- lunghezza massima **8** caratteri
- il primo carattere deve essere una **lettera**
- i successivi caratteri devono essere **lettere**, **numeri** o il carattere «_»
- non c'è distinzione tra maiuscole e minuscole

Bisogna poi indicare il linguaggio con il quale scrivere il nuovo programma:

SFC..... Sequential Function Chart
FC ..... Flow Chart
FBD..... Functional Block Diagram (può contenere parti in LD)
LD ..... Ladder Diagram da inserire mediante l'editor Quick LD
ST ..... Structured Text
IL..... Instruction List

Infine bisogna specificare la sezione dove risiede programma :

Iniziale..... primo livello della sezione «Iniziale»
Sequenziale ..... primo livello della sezione «Sequenziale»
Finale ..... primo livello della sezione «Finale»
Funzione ..... nella sezione «Funzioni»
Blocco funzione.. nella sezione «Blocchi funzione»
Figlio di..... SFC figlio o sottoprogramma di un programma esistente

Indicando una delle prime cinque scelte, il programma viene collocato al primo livello di una sezione **Iniziale**, **Finale**, **Sequenziale**, **Funzioni** o **Blocchi funzione**.

La rimanente scelta **Figlio di** indica che il nuovo è un programma **SFC** figlio, un sottoprogramma **FC** o un sottoprogramma. In questo caso bisogna specificare il nome del programma genitore, le possibili scelte sono riportate nell'elenco fianco di «Figlio di:».
È bene ricordare che un programma di primo livello sequenziale deve essere scritto utilizzando il linguaggio **SFC** o **FC** e che non è possibile utilizzare il linguaggio **SFC** o **FC** per i programmi ciclici ed i relativi sottoprogrammi.

Inserire commenti nel programma

ISaGRAF permette di affiancare una descrizione a ciascun programma del progetto. Tale commento viene mostrato con un carattere più piccolo accanto al nome del programma. Richiamare la voce «**File / Commento al programma**» per inserire o modificare il commento del programma evidenziato.

Modificare un programma

Questa funzione permette di modificare un programma. L'editor usato per modificare un programma dipende dal tipo di linguaggio scelto. La modifica dei programmi avviene in finestre indipendenti, in questo modo è possibile modificare più di un programma in finestre affiancate. Premere il tasto **INVIO** per modificare il programma evidenziato. Alternativamente è possibile accedere al programma facendo doppio click sul nome nella lista.

Modificare il file «diario»

Esiste un **file diario** assegnato a ciascun programma. Si tratta di un file di testo che conserva tutte le note relative alle modifiche effettuate sul programma. Il file diario può essere liberamente modificato o stampato a piacere. All'uscita dell'editor usato per modificare il codice sorgente di un programma, appare automaticamente una finestra che permette di inserire le nuove annotazioni con data e ora corrette

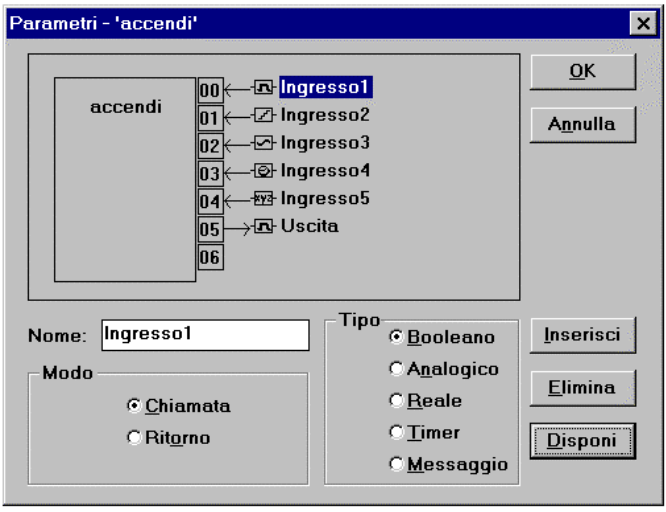
Il dizionario delle variabili

Il comando «**File / Dizionario**» avvia l'editor del dizionario dove sono dichiarate le variabili del progetto. Le variabili possono essere **globali** (riconosciute da ogni programma del progetto) o **locali** al programma evidenziato. Nell'editor del dizionario si possono anche dichiarare dei **Nomi definiti** che sono degli pseudonimi che permettono la sostituzione di un nome o di un'espressione nel codice sorgente di un programma.

Parametri di una funzione, sottoprogramma o blocco funzione

Il comando «**File / Parametri**» consente di definire i parametri di ingresso ed uscita del sottoprogramma, funzione o blocco funzione evidenziato. Questo comando non sortisce alcun effetto qualora nella finestra Programmi sia evidenziato un programma principale della sezione «**Iniziale**» o «**Finale**» o un programma SFC.
Una funzione o un sottoprogramma ha sempre un unico parametro di ritorno, che deve avere lo stesso nome del sottoprogramma o della funzione, per uniformità con le convenzioni del linguaggio ST (il parametro di ritorno deve essere sempre presente).
Un blocco funzione può avere più di un parametro di ritorno.

Complessivamente si possono avere al massimo **32** parametri (di chiamata o di ritorno). La seguente finestra di dialogo permette di assegnare i parametri del sottoprogramma:



La lista in alto a sinistra elenca i parametri nell'ordine della modalità di chiamata: prima i parametri in **ingresso** ed alla fine il parametro di **ritorno**. La porzione inferiore della finestra mostra la descrizione dettagliata del parametro evidenziato nell'elenco. Qualunque tipo di dato ISaGRAF può essere usato come parametro. Il parametro di **ritorno** deve essere posizionato, nell'elenco, dopo i parametri di **chiamata**. Il nome dei parametri deve sottostare alle seguenti regole:

- la lunghezza del nome non deve eccedere la lunghezza di 16 caratteri
- il primo carattere deve essere una lettera
- i caratteri successivi devono essere lettere, numeri o il carattere «_».
- non c'è distinzione tra maiuscolo e minuscolo

Il comando «**Inserisci**» permette di inserire un nuovo parametro prima del parametro evidenziato. Il comando «**Elimina**» elimina il parametro evidenziato. Il comando «**Disponi**» riordina automaticamente i parametri, ponendo alla fine dell'elenco i parametri di ritorno.

Spostare un programma nella struttura gerarchica ad albero

Il comando «**Rinomina/Sposta**» del menù «**File**» permette di cambiare il nome ad un programma o di spostarlo in un'altra sezione e nella struttura ad albero. Non è, però, possibile modificare il tipo di linguaggio di un programma esistente. Richiamando questa voce, appare la stessa finestra della creazione di nuovi programmi, con tutti i campi aggiornati alle caratteristiche del programma evidenziato. Il nome del programma può essere modificato e può essere

evidenziata una sezione o un programma genitore all'interno della struttura ad albero, in cui spostare il programma stesso.
Richiamando la voce «**Disponi programmi**» del menù «**File**» è possibile assegnare un esplicito ordinamento ad un elenco di programmi con lo stesso livello o genitore. Se il programma evidenziato si trova al primo livello, vengono sistemati i programmi di primo livello della attuale sezione. Se il programma evidenziato si trova ad un livello inferiore, vengono sistemati solamente i figli SFC ed i sottoprogrammi aventi lo stesso genitore del programma evidenziato. Quando la finestra di dialogo **Disposizione programmi** è aperta, evidenziare il programma voluto e premere il pulsante «**Su**» o «**Giù**» per spostarlo nell'elenco.

Copiare i programmi

Per fare la copia di un programma, evidenziare il programma sorgente nella finestra Programmi e richiamare il comando «**File / Copia**». Attivando questa procedura, appare la stessa finestra usata all'atto della creazione del programma, con tutti i campi aggiornati alle caratteristiche del programma evidenziato. Inserire il nome del programma destinazione e la sua posizione nelle sezioni e nella struttura ad albero. Se il programma destinazione non esiste, viene creato nella posizione specificata. Se il programma destinazione esiste già, viene sovrascritto. Tutte le dichiarazioni locali e i nomi definiti vengono copiate assieme al programma. Il tipo di linguaggio del programma destinazione deve essere il medesimo del programma sorgente. Premere il pulsante «**OK**» per copiare il programma.
Sempre nella finestra Programmi, il comando «**Copia in altro progetto**» del menù «**File**» permette di copiare il programma evidenziato in un progetto diverso, mantenendo lo stesso nome. Eventuali programmi SFC figli e sottoprogrammi del programma evidenziato verranno copiati nell'altro progetto. I nomi del programma evidenziato e dei suoi eventuali figli non devono essere in uso nel progetto destinazione. Questa procedura non permette la sovrascrittura dei programmi. Tutte le dichiarazioni locali e i nomi definiti vengono copiati assieme al programma.

Eliminare i programmi

Per eliminare un programma, per prima cosa evidenziarlo nell'elenco dei programmi e poi richiamare la voce «**File / Elimina**». Un programma che possiede figli o sottoprogrammi non può essere eliminato. In questo caso bisogna prima eliminare i programmi figli o i sottoprogrammi. Tutte le dichiarazioni locali e i nomi definiti vengono eliminati assieme al programma.

Importare funzioni o blocchi funzione dalla libreria

Il comando «**Strumenti / Importa da libreria**» permette di copiare funzioni o blocchi funzione scritti in linguaggio IEC dalla libreria nella sezione «**Funzioni**» o «**Blocchi funzione**» del progetto aperto. Le variabili locali e i nomi definiti della funzione importata, vengono copiati assieme ad essa. Quando una funzione è stata importata in modo corretto dalla libreria, può essere spostata in un'altra sezione o in un'altra posizione all'interno della struttura ad albero richiamando la voce «**File / Rinomina/Sposta**» al fine di evitare sovrapposizione di nomi. Non bisogna dimenticare, nel caso di una funzione, di modificare anche il nome del parametro di ritorno.

Esportare funzioni o blocchi funzione nella libreria

Il comando «**File / Esporta in libreria**» permette di copiare un programma della sezione «**Funzioni**» o «**Blocchi funzione**» (del progetto aperto) nella Libreria. Le variabili e i nomi definiti locali contenuti nella funzione o nel blocco da esportare, vengono copiati con essa. La funzione esportata dovrà essere nuovamente compilata (verificata) nel gestore di libreria ISaGRAF, in modo da poter essere usata nella libreria. Funzioni e blocchi funzione della libreria non possono far uso di variabili globali.

Nota: Se una funzione o blocco funzione viene aggiunto alla Libreria, può essere richiamato da ogni progetto compreso il progetto a cui apparteneva e da cui va eliminato.
Se una funzione o blocco funzione viene aggiunto alla Libreria, va poi compilato indipendentemente dal progetto a cui apparteneva (utilizzando il comando **«File / Verifica (compila)»** dalla finestra **Librerie**) con le opportune opzioni del compilatore che lo rendano compatibile con i progetti che lo richiamano.
La funzione o il blocco funzione esportato conserva solo le variabili locali e non quelle globali del progetto a cui apparteneva. Ne consegue che eventuali variabili con attributo **ingresso** o **uscita** non vengono esportate, poiché tali variabili non possono avere campo di validità **locale**, ma solamente **globale**.

A.3.3 Usare gli strumenti di generazione del codice

I comandi del menù «**Compila**» della finestra Progetti permettono di accedere al generatore di codice e di impostare opzioni e dati aggiuntivi usati nella produzione del codice dell'applicazione. Consultare il capitolo «Usare il generatore di codice» in questo manuale per ulteriori informazioni su questi strumenti.

Compilare il codice dell'applicazione

Il comando «**Compila / Compila applicazione**» avvia la generazione del codice del progetto. Per poter usare questo comando, le opzioni del compilatore devono essere impostate correttamente. Prima di generare il codice destinazione, ogni programma che non sia stato controllato in precedenza, viene **verificato** per evidenziare eventuali errori di sintassi. ISaGRAF mette a disposizione un compilatore incrementale che non necessita di ricompilare i programmi già compilati.

Verificare il programma selezionato

Il comando «**Compila / Verifica**» permette di verificare la sintassi del programma evidenziato nell'elenco. Quando un programma è stato verificato e non è stato evidenziato alcun errore, non viene ulteriormente controllato in fase di generazione del codice, a meno che non siano stati modificati contenuti o nomi definiti da cui dipende.

Simulazione di una modifica

Il comando «**Compila / Tocca**» simula la modifica di ciascun programma, così da forzare la ricompilazione successivamente in fase di generazione del codice.

Opzioni per l'esecuzione dell'applicazione

Il comando «**Compila / Opzioni esecuzione applicazione**» apre una finestra di dialogo in cui è possibile inserire i principali parametri di l'esecuzione dell'applicazione. Sono previsti: il tempo ciclo, la gestione degli errori durante l'esecuzione, la modalità di avvio e l'implementazione hardware delle variabili. Consultare il capitolo «Usare il generatore di codice» in questo manuale per ulteriori informazioni su questo comando.

Opzioni del compilatore

Il comando «**Compila / Opzioni compilatore**» serve ad impostare le opzioni usate dal Generatore di codice ISaGRAF per produrre ed ottimizzare il codice destinazione. Consultare il capitolo «Usare il Generatore di codice» in questo manuale per ulteriori informazioni su questo comando.

Elenco cronologico delle modifiche

Il comando «**Progetto / Cronologia modifiche**» apre una finestra di dialogo che mostra l'elenco cronologico delle modifiche apportate al progetto. Consultare il capitolo «Gestione dei progetti» in questo manuale per ulteriori informazioni su questo comando.

Definizione di risorse

Si chiamano «**risorse**» i dati definiti dall'utente (ad esempio un file) che sono stati aggiunti al codice destinazione per poter essere utilizzati con esso. Consultare la sezione «Usare il Generatore di codice» in questo stesso manuale per ulteriori informazioni sul formato del file di definizione delle risorse.

A.3.4 Altri strumenti ISaGRAF

Il menù «**Progetto**» raggruppa i comandi per avviare gli strumenti ISaGRAF per il progetto selezionato. Far riferimento ai relativi capitoli in questo manuale per ulteriori informazioni su questi strumenti.

Connettere le variabili I/O

Il comando «**Progetto / Connessione di IO**» avvia l'editor delle connessioni ISaGRAF per le variabili di I/O. Questo strumento viene utilizzato per stabilire le relazioni tra le variabili di I/O dichiarate nel dizionario del progetto ed i corrispondenti sistemi di I/O hardware.

Usare l'editor dei Riferimenti incrociati

Il comando «**Progetto / Riferimenti incrociati**» permette di calcolare, vedere o stampare i riferimenti incrociati del progetto. Mediante i riferimenti incrociati è possibile controllare la presenza di ogni variabile nel codice sorgente dei programmi nell'intero progetto. Questa funzione risulta molto utile per trovare i punti di accesso ad una variabile o ad una risorsa globale o per elencare tutti i punti in cui una variabile globale compare nel codice sorgente.

Il testo descrittivo del progetto

Il comando «**Progetto / Descrizione progetto**» permette di modificare il testo descrittivo del progetto. Questo documento serve a distinguere il progetto dagli altri nella lista dei progetti. Il testo descrittivo può anche essere usato per conservare le annotazioni relative al progetto. Viene visualizzato nelle finestra Gestione progetti.

Stampare un documento completo

La voce «**Progetto / Stampa documentazione**» permette all'utente di compilare e stampare un documento completo relativo al progetto evidenziato. Tale documento può contenere ogni componente (programma, variabile, parametri…) del progetto evidenziato. Per costruire un documento con contenuti specifici (non completo come il documento predefinito), l'utente deve solamente definirne la lista delle voci.

A.3.5 Aggiungere comandi al menù Strumenti

ISaGRAF prevede anche la possibilità di inserire altre voci al menù «**Strumenti**». Possono essere aggiunti fino a 10 comandi.
Le voci definite dall'utente sono contenute nel file di testo «**\ISAWIN\COM\ISA.MNU**». In tale file si possono aggiungere commenti su qualsiasi riga, preceduti dal carattere «**;**». Ogni comando viene descritto in due righe di testo con la seguente sintassi:

```
M=stringa_menù
C=linea_di_comando
```

La `stringa_menù` corrisponde al testo che apparirà nel menù «**Strumenti**». La `linea_di_comando` è un qualsiasi comando eseguibile da MS-DOS o da Windows e può essere completato con dei parametri. È possibile usare i caratteri «**%A**» per indicare il nome del progetto aperto ed i caratteri «**%P**» per indicare il nome del programma evidenziato. Nell'esempio seguente viene avviato il «Blocco note» di Windows per modificare il programma evidenziato (utile con i programmi di tipo ST e IL):

```
M=Edita con Notepad
C=Notepad.exe \isawin\apl\%A\%P.lsf
```

A.3.6 Simulazione e Debug dell'applicazione

I comandi del menù «**Debug**» avviano il debugger grafico ISaGRAF, sia in simulazione che con connessioni reali.

Simulazione

Il comando «**Debug / Simulazione**» avvia il debugger in modo simulazione. Appare la finestra del simulatore. Questo comando risulta molto utile per provare l'applicazione qualora il dispositivo destinazione non sia disponibile. L'avvio del simulatore chiude la finestra Programmi, che viene poi riaperta in modo debug dopo l'apertura di entrambe le finestre Debugger e del simulatore. Il simulatore non può essere avviato senza che sia stato generato il codice destinazione. Il simulatore non viene avviato se ci sono finestre figlie aperte (editor, generazione di codice,

connessioni I/O…). Ognuna di queste deve essere chiusa prima di avviare questo comando. Il comando «**File / Simulazione**» appare anche nei menù degli editor ISaGRAF dei programmi.

Debug

Il comando « **Debug / Debug**» apre la finestra principale Debugger, e chiude la finestra Programmi. La finestra Programmi viene, quindi, riaperta in modo debug non appena attivata la comunicazione tra il debugger e l'applicazione target. Il debugger non può essere avviato se il codice target non è stato generato. Il debugger non viene avviato se ci sono finestre figlie aperte (editor, generazione di codice, connessioni I/O…). Ognuna di queste deve essere chiusa prima di avviare questo comando. Questa voce appare anche nei menù degli editor ISaGRAF.

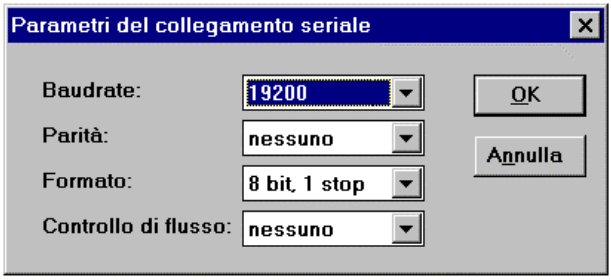
Impostare i collegamenti

Il comando « **Debug / Parametri collegamento**» apre la seguente finestra di dialogo. che permette di impostare i parametri del collegamento per le comunicazioni tra il debugger sul PC host ed il sistema target ISaGRAF.

Il parametro «**Numero del Target Slave**» identifica il sistema o il processo target ISaGRAF. Può assumere un valore compreso tra **1** e **255**. Consultare il manuale del fornitore per ottenere il numero slave del sistema destinazione in uso.
Il parametro «**Porta di comunicazione**» identifica la comunicazione hardware tra l'ambiente di lavoro ISaGRAF ed il dispositivo destinazione. Può essere il nome di una porta seriale, «**Ethernet**» o un collegamento TCP-IP riservato che faccia uso di «Winsock» Versione 1.1.
Il parametro «**Time out**» rappresenta il tempo lasciato al sistema destinazione per rispondere ad un'interrogazione del debugger. Questo tempo viene indicato in **secondi**. Il campo «**Tentativi**» indica il numero di tentativi eseguiti dal debugger prima di restituire un errore di comunicazione.

Impostare un collegamento seriale

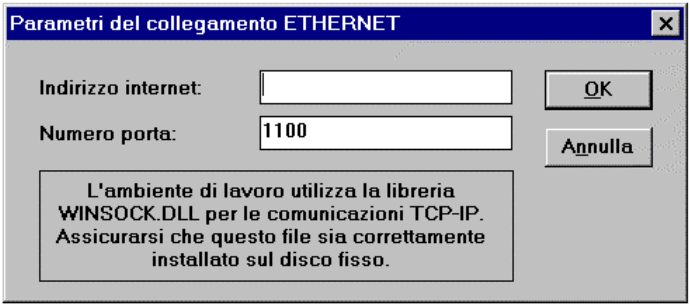
Scegliendo una porta seriale (COM1..4) il pulsante «**Imposta**» permette di accedere ai parametri di comunicazione seriale. Appare la seguente finestra di dialogo:



Possono essere indicati **Baud rate**, **Parità** e **Formato**. Scegliendo «**hardware**» per il parametro «**Controllo di flusso**», l'ambiente ISaGRAF provvederà a tenere sotto controllo le linee CTS e DSR per permettere gli scambi in modalità handshaking hardware.

Impostare il collegamento Ethernet

Scegliendo il tipo di comunicazione «Ethernet», il pulsante «**Imposta**» permette di impostare l'**Indirizzo Internet** ed il **Numero di porta** Internet per le comunicazioni TCP-IP. Appare la seguente finestra di dialogo:



In questi campi i parametri vengono definiti seguendo i formati standard dell'interfaccia Socket. Per le comunicazioni TCP-IP, l'ambiente di lavoro fa uso fa uso della libreria dinamica WINSOCK.DLL Versione 1.1, che deve essere correttamente installata sul disco. Il valore predefinito del numero di porta in uso è «**1100**», se non diversamente specificato dal dispositivo ISaGRAF destinazione.

A.4 Usare l'editor SFC

In questo capitolo vengono descritte le caratteristiche ed i comandi specifici dell'editor SFC. Gli altri comandi, comuni anche agli altri editor, vengono descritti nel capitolo «Ulteriori informazioni sugli editor dei programmi», in questo stesso documento.

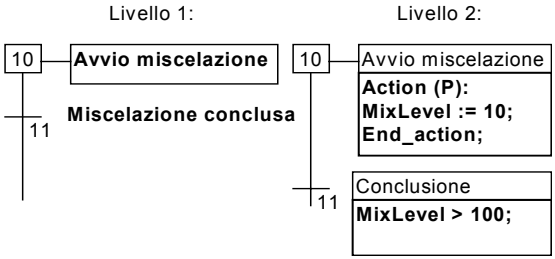
Si adopera il linguaggio SFC per descrivere le operazioni di un processo sequenziale. Una semplice rappresentazione grafica identifica i passi e le condizioni che indicano la sequenza delle operazioni attivate da un token. L'editor SFC grafico ISaGRAF permette di inserire il codice in linguaggio SFC, che costituisce la parte centrale dello standard IEC 1131-3. Gli altri linguaggi di solito permettono di descrivere le azioni per i passi e le condizioni logiche per le transizioni. L'editor SFC grafico ISaGRAF permette di inserire programmi SFC completi. La possibilità di operare contemporaneamente in modo grafico e testuale, permette di lavorare sul diagramma SFC con le corrispondenti azioni e condizioni.

A.4.1 Argomenti principali del linguaggio SFC

Il linguaggio SFC viene usato per rappresentare processi di tipo sequenziale. Il ciclo del processo viene suddiviso in un numero di **passi** ben definiti (contenenti a loro volta delle azioni) in successione, separati da **transizioni**. Ulteriori dettagli sul linguaggio SFC sono reperibili nel Manuale di riferimento Linguaggi ISaGRAF. I componenti SFC sono uniti tra loro mediante **linee orientate**. L'orientamento predefinito di una linea è dall'**alto** verso il **basso**. Questi sono i componenti grafici di base per creare un diagramma SFC:

 ..... **Passo iniziale**
 ..... **Passo**
 ..... **Transizione**
 ..... **Salto ad un passo**
 ..... **Macro passo**
 ..... **Passo iniziale di una macro**
 ..... **Passo finale di una macro**

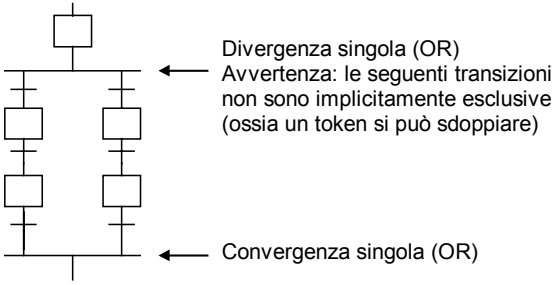
La programmazioni SFC si compone normalmente di due differenti livelli. Il **Livello 1** mostra il diagramma grafico, la numerazione dei passi e delle transizioni, i commenti ai passi ed alle transizioni. Il **Livello 2** si riferisce alla programmazione in linguaggio **ST** o **IL** delle azioni all'interno dei passi o alle condizioni relative alle transizioni. Azioni o condizioni possono far riferimento a **sottoprogrammi** scritti in altri linguaggi (**FBD**, **LD**, **ST** o **IL**). Subito sotto, viene riportato un esempio di programmazione di livello 1 e di livello 2:



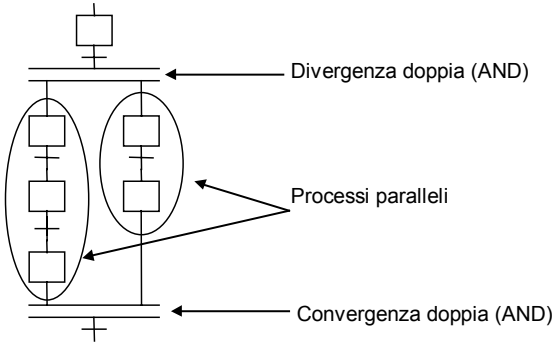
La programmazione di livello2 per un passo viene inserita con un editor di testo. Possono essere inclusi blocchi di azioni di tipo diverso (booleane, impulsive, non memorizzate, chiamate a sottoprogrammi SFC) programmate in ST o IL.
Il livello 2 di programmazione per una transizione può essere inserito sia con il linguaggio testuale IL o ST, sia con l'editor Quick LD.

Divergenze e convergenze

Divergenze e convergenze permettono di rappresentare i **collegamenti multipli** tra passi e transizioni. Divergenze o convergenze semplici rappresentano differenti possibilità inclusive fra varie parti del processo.

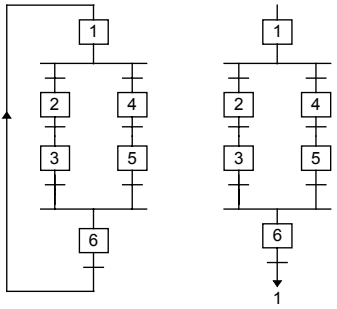


Una doppia divergenza rappresenta dei processi in **parallelo**.



Salto ad un passo

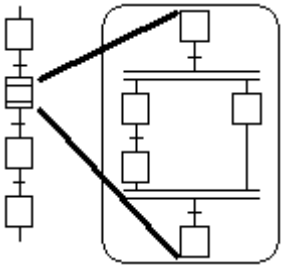
L'editor SFC permette di disegnare collegamenti solamente dall'**alto** verso il **basso**. Un **salto** verso un passo serve a rappresentare un collegamento con una parte più in alto nel diagramma. I seguenti diagrammi sono equivalenti:



Il salto verso una transizione è proibito e deve essere rappresentato esplicitamente come una doppia (AND) convergenza.

Macro passi

Un macro passo è la rappresentazione **unica** di un gruppo **isolabile** di passi e transizioni. L'inizio di un macro passo è segnato da un **passo iniziale** e termina con un **passo finale**.



La rappresentazione in dettaglio di un macro passo deve essere inserita nel medesimo programma SFC. Il simbolo del macro passo deve avere lo stesso **numero di riferimento** del passo iniziale della macro.

A.4.2 Diagramma SFC

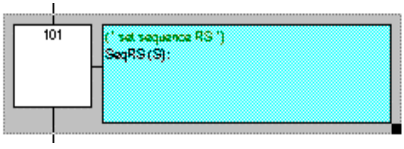
Per disegnare un diagramma SFC è sufficiente inserire i componenti significativi del diagramma. L'editor SFC provvede automaticamente a tracciare le singole linee (orizzontali o verticali) di unione tra due elementi. Per inserire un simbolo SFC nel diagramma, si deve selezionare la cella nella posizione desiderata del diagramma, quindi fare click con il mouse su uno dei componenti della barra strumenti.

In alternativa è possibile utilizzare la tastiera, premendo:

F2: ..... Inserimento di un passo iniziale
F3: ..... Inserimento di un passo finale
F4: ..... Inserimento di una transizione
F5: ..... Inserimento di un salto ad un passo
F6: F7: ..... Inserimento di una divergenza o convergenza OR / Aggiunta di rami
↑F6: ↑F7: ..... Inserimento di una divergenza o convergenza AND / Aggiunta di rami
F8: ..... Inserimento di un macro passo
F9: ↑F9: ..... Inserimento di un passo iniziale o finale del corpo di un macro passo

(Il simbolo freccia che precede un tasto funzione "↑" indica una combinazione con il tasto SHIFT)

La **griglia di modifica** mostra una **matrice di celle**. L'editor SFC permette di visualizzare o nascondere la griglia nella fase di inserimento del diagramma. La griglia risulta molto utile nella disposizione iniziale del diagramma o per selezionare porzioni del diagramma. Nella finestra dell'editor SFC, selezionando «**Opzioni / Disposizione / Visualizza griglia**» si possono visualizzare o nascondere i punti della griglia.
L'editor SFC mostra sempre la posizione corrente nella matrice. La cella attiva viene evidenziata in colore grigio. Il piccolo quadrato nella sua parte destra in basso può essere utilizzato per ridimensionare a piacimento le celle dell'intero diagramma. In questo modo è possibile cambiare il rapporto X/Y tra le dimensioni delle celle.



Creare una divergenza o una convergenza

Divergenze e convergenze vengono sempre disegnate **da sinistra verso destra**. Per disegnare una convergenza o una divergenza, bisogna posizionarne l'angolo sinistro sul diagramma. Il tipo di disegno (semplice o doppio) si ottiene selezionando uno di questi pulsanti sulla barra degli strumenti.

F6: F7: ..... Inserimento di una divergenza o convergenza OR / Aggiunta di rami
↑F6: ↑F7: ..... Inserimento di una divergenza o convergenza AND / Aggiunta di rami

Aggiungere diramazioni alle divergenze

Per posizionare **l'inizio** e la **fine** di una **diramazione ausiliaria** sulla linea di una divergenza o di una convergenza, si usano questi pulsanti della barra degli strumenti.

F6: F7: ..... Inserimento di una divergenza o convergenza OR / Aggiunta di rami

↑F6: ↑F7: ..... Inserimento di una divergenza o convergenza AND / Aggiunta di rami

L'inserimento di nuove diramazioni è possibile solo con la presenza dell'angolo sinistro di una divergenza o di una convergenza. L'angolo destro ha lo stesso stile (semplice o doppio) dell'angolo principale sinistro. Gli angoli destri non possono essere inseriti senza che sia prima stato aggiunto l'angolo principale sinistro.

Inserire un macro passo

Questo pulsante viene utilizzato per inserire un macro passo nel diagramma principale. Il corpo del macro passo deve essere disegnato in un qualunque altro punto dello stesso programma SFC

Corpo di un macro passo

I macro passi devono essere descritti nello stesso programma SFC del diagramma principale.
Un macro passo deve iniziare con un apposito **passo iniziale** e terminare con un apposito **passo finale**. Il sottodiagramma che descrive l'implementazione del macro passo **non deve avere riferimenti esterni**. Il passo di inizio del macro passo deve avere lo **stesso numero di riferimento** del simbolo di macro passo del ramo principale del diagramma.

A.4.3 Lavorare su un diagramma SFC esistente

Si possono utilizzare sia il mouse che la tastiera per selezionare un'area rettangolare sul diagramma. L'area selezionata viene evidenziata in colore grigio. Si possono quindi usare i seguenti comandi del menù «**Modifica**».

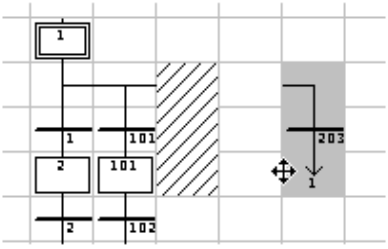
I comandi Taglia/Copia/Cancella/Incolla

Il menù «**Modifica**» mette a disposizione le seguenti voci:
Taglia ..... Sposta il rettangolo evidenziato dallo schermo agli appunti SFC
Copia..... Copia il rettangolo evidenziato dallo schermo agli appunti SFC
Cancella ..... Cancella il rettangolo evidenziato
Incolla..... Inserisce nella posizione corrente il contenuto degli appunti SFC

La voce «**Modifica / Incolla**» copia il contenuto del blocco appunti SFC sulla schermo. I comandi copia/incolla funzionano sia sul diagramma SFC, sia con la programmazione di livello 2 di passi e transizioni. È anche possibile copiare un diagramma da un programma ed incollarlo in un diverso programma SFC. Gli elementi vengono inseriti prima della posizione correntemente selezionata.

Spostare elementi

Quando degli elementi SFC sono selezionati nel diagramma SFC, si può spostarli in un altra posizione del diagramma trascinando la selezione con il mouse. Mentre si trascina la selezione, la posizione iniziale degli elementi selezionati viene tratteggiata.



L'area di destinazione per gli elementi spostati deve essere vuota. Non sono possibili inserzioni mentre si spostano simboli SFC.

Rinumerare passi e transizioni.

Ogni passo o transizione viene identificato sul diagramma SFC da un numero logico. La voce «**Modifica / Rinumera**» mette automaticamente in sequenza i numeri di riferimento di tutti i passi e transizioni del programma SFC in lavorazione. Quando il comando cambia il numero di un passo, tutti i salti a tale passo vengono automaticamente aggiornati con il nuovo numero di riferimento (parimenti accade con i macro passi ed i passi iniziali).

Accesso diretto ad un passo o ad una transizione

La voce «**Modifica / Vai a**» permette di accedere ad un passo o ad una transizione esistenti. La finestra scorre automaticamente il diagramma SFC fino a rendere visibile il passo o la transizione.

Trovare e sostituire testi

I comandi «**Modifica /Trova /Sostituisci**» permettono di trovare e rimpiazzare stringhe di testo in tutto il programma (in tutti i passi e le transizioni). La finestra di dialogo Trova/Sostituisci viene utilizzata per immettere il testo da trovare ed apre direttamente la sezione del programma di livello 2 dove il testo è presente.

A.4.4 Inserire la programmazione di livello2

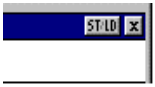
Per inserire le istruzioni di Livello 2, l'utente deve fare doppio click sul simbolo del passo o della transizione. La finestra di programmazione del Livello 2 viene mostrata alla destra della finestra principale SFC. La linea di separazione tra il diagramma SFC e il programma livello 2 può essere spostata liberamente.

Si possono aprire una o due aree di livello 2 contemporaneamente. I seguenti comandi sono disponibili dalla tastiera, dal mouse o dal menù «Modifica»:

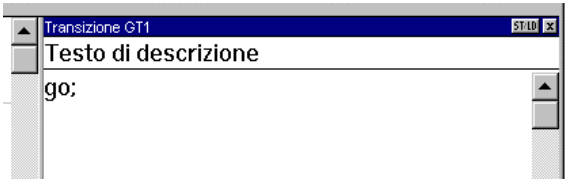
	Tastiera	*Mouse*	*Menù «Modifica»*
Apre nell'ultima finestra	INVIO	Doppio Click	Modifica livello 2
Apre finestra separata	CTRL+INVIO	CTRL + Doppio Click	Modifica livello 2 in una finestra separata

Quando due finestre di livello 2 sono visibili, la separazione tra esse può essere liberamente spostata. Il pulsante alla destra della barra del titolo chiude la finestra di livello 2.

Il linguaggio predefinito del Livello 2 è **ST** (Structured Text). Per quanto riguarda, invece, le transizioni, la programmazione di livello 2 può anche essere inserita con l'editor **Quick LD**. Usare il pulsante «**ST/LD**» nella barra del titolo della finestra di livello 2 per selezionare il linguaggio attivo. Questo comando è valido solo se la finestra di programmazione di livello 2 è vuota.



Una linea di testo appare nella parte superiore della finestra di livello 2, viene usata per immettere un breve testo di descrizione. Questo testo verrà visualizzato come commento IEC nel disegno dei simboli SFC. E' molto utile ed è utilizzato da altri comandi, come ad esempio «Vai a» ed anche nella stampa (passi e transizioni) di documenti SFC.



Il comando «**Opzioni / Rinfresca diagramma**» può essere usato ogni volta che una finestra di livello 2 è aperta per rinfrescare il diagramma SFC con le modifiche apportate ai programmi di livello 2.

Inserire il nome di una variabile

In fase di programmazione testuale, premere questo pulsante per selezionare una variabile dichiarata nel dizionario del progetto ed inserire il nome nella posizione corrente del cursore. Programmando in Quick LD, premere questo pulsante per selezionare la variabile da collegare al contatto o al blocco di parametri I/O evidenziati.

Inserire un blocco di Azioni Impulsive in un passo

Durante la programmazione di livello 2, premere questo pulsante per inserire lo schema di un blocco **Azione impulsiva** (Pulse action) nella posizione corrente del cursore. Quello che segue è il formato di un blocco azione Impulsiva.

```
Action (P) :
   istruzione ST;
   ...
End_Action;
```

Le istruzioni di tipo azione Impulsiva, vengono eseguite una sola volta all'attivazione del passo. Per ulteriori dettagli sulla programmazione SFC, si veda la Guida di riferimento ai linguaggi ISaGRAF.

N

Inserire in un passo un blocco Azione Non-memorizzata

Durante la programmazione di livello 2, premere questo pulsante per inserire lo schema di un blocco **Azione non-memorizzata** (Non stored action) nella posizione corrente del cursore. Quello che segue è il formato di un blocco azione non-memorizzata.

```
Action (N) :
   istruzione ST;
   ...
End_Action;
```

Le istruzioni di tipo azione non-memorizzata vengono eseguite ad ogni ciclo PLC quando il passo è attivo. Per ulteriori dettagli sulla programmazione SFC, si veda la Guida di riferimento ai linguaggi ISaGRAF.

P0 P1

I nuovi qualificatori di Azione P0 e P1

ISaGRAF supporta i nuovi qualificatori di **Azione P0** e **P1**. Durante la programmazione del livello 2 di un passo, premere questi pulsanti per inserire lo schema di un blocco azione P0 o P1 alla posizione corrente del cursore. Quello che segue è il formato di tali blocchi:

```
Action (P0) :              Action (P1) :
   istruzione ST;                 istruzione ST;
   ...                            ...
End_Action;                End_Action;
```

Le azioni P1 sono istruzioni che vengono eseguite solamente una volta all'attivazione del passo (come per le Azioni impulsive). Le azioni P0 sono istruzioni sono istruzioni che vengono eseguite solo una volta quando il passo cessa di essere attivo. Per ulteriori dettagli sulla programmazione SFC, si veda la Guida di riferimento ai linguaggi ISaGRAF.

Azioni booleane

Altre locuzioni semantiche sono disponibili per agire direttamente su una variabile booleana che controlla l'attività del passo. Tali azioni consistono nel collegare **il segnale di attività di un passo** ad una variabile booleana interna o di uscita

<variabile_booleana> (N);	assegna il segnale di attività del passo alla variabile
<variabile_booleana >;	stesso effetto (il parametro N è facoltativo)
/ <variabile_booleana >;	assegna la negazione del segnale di attività del passo alla variabile

Esistono altre possibilità di «accendere» o «spegnere» una variabile booleana all'attivazione del passo. Ecco la sintassi delle azioni che accendono o spengono le variabili booleane:

<variabile_booleana> (S);	«Set», imposta la variabile al valore «TRUE» (VERO) quando il segnale di attività del passo diventa VERO
<variabile_booleana > (R);	«Reset», assegna alla variabile il valore «FALSE» (FALSO) quando il segnale di attività del passo diventa VERO

Azioni SFC

Altre locuzioni semantiche permettono di controllare l'esecuzione di un programma figlio SFC. Un'**Azione SFC** è una sequenza figlia SFC, avviata o terminata a seconda del segnale di attività del passo. Un'azione SFC può avere i qualificatori **N** (Non-memorizzata), **S** («Set») o **R** («Reset»). Ecco la sintassi delle azioni SFC di base:

<programma_figlio> (N);	avvia la sequenza figlia quando viene eseguito il passo e termina la sequenza figlia quando il passo cessa di essere attivo.
<programma_figlio>;	stesso effetto della sintassi precedente (il parametro N è facoltativo)
<programma_figlio> (S);	avvia la sequenza figlia quando il passo viene eseguito – nulla avviene quando il passo cessa di essere in esecuzione
<programma_figlio> (R);	termina la sequenza figlia quando il passo viene eseguito – nulla avviene quando il passo cessa di essere in esecuzione

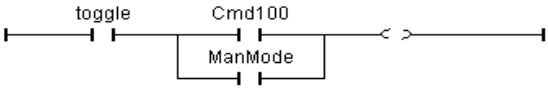
La sequenza SFC specificata deve essere un **programma SFC figlio** del programma correntemente in fase di modifica, creato con il gestore di programmai ISaGRAF.

Transizioni scritte in ST

Una transizione di livello 2 è costituita da un'espressione booleana. Per programmarla in linguaggio ST, è sufficiente inserire la condizione booleana rispettando la sintassi ST. L'inserimento di un punto e virgola alla fine dell'espressione è facoltativo.

Transizioni scritte in Quick Ladder

Per programmare condizioni o transizioni di livello 2, è disponibile l'editor Quick LD. In questo caso il diagramma è costituito da un solo ramo, con una sola bobina che rappresenta la transizione. Il nome della transizione non viene ripetuto con il simbolo di bobina. Di seguito l'esempio di una condizione di transizione programmata in Quick LD.



In fase di programmazione Quick LD, usare i tasti freccia per spostare la selezione sulla griglia logica di programmazione ed inserire i simboli con i seguenti tasti scorciatoia:
F2:..... inserisce un contatto dopo il simbolo evidenziato / inizia il ramo
F3:..... inserisce un contatto prima del simbolo evidenziato
F4:..... inserisce un contatto parallelamente al simbolo evidenziato
F6:..... inserisce un blocco dopo il simbolo evidenziato
F7:..... inserisce un blocco prima del simbolo evidenziato
F8:..... inserisce un blocco parallelamente al simbolo evidenziato
È anche possibile fare click con il mouse sulla barra dei tasti funzione posta nella parte inferiore della finestra di livello 2, invece di premere i tasti funzione.

Premere INVIO quando è evidenziato un contatto o un blocco di parametri I/O per selezionare una variabile o inserire un valore costante. Premere INVIO quando è evidenziato un blocco funzione per indicare il tipo di blocco funzione. Si ottiene lo stesso effetto anche con un doppio click sul simbolo.

Premere la «barra spazio» quando è evidenziato un contatto per cambiare il tipo di contatto (diretto, inverso o con riconoscimento di impulso). Far riferimento al capitolo «Usare l'editor Quick LD» in questo stesso documento per maggiori dettagli sulle caratteristiche Quick LD.

A.4.5 Utilizzo della Collezione SFC

L'editor SFC di ISaGRAF permette di gestire una collezione SFC: tale collezione è costituita da un insieme di strutture SFC che possono essere inserite in ogni diagramma SFC. Gli elementi della collezione SFC possono contenere la programmazione di livello 2 in passi e transizioni. Per gestire la collezione SFC usare i seguenti comandi del menù «**Strumenti**»:

Copia nella collezione SFC copia gli elementi selezionati nella collezione SFC.
Incolla dalla collezione SFC incolla un elemento dalla collezione SFC nella posizione corrente.

Quando si copiano degli elementi nella collezione SFC (i.e. creando un nuovo elemento della collezione SFC), si può scegliere se inglobare o no la programmazione di livello 2 dei simboli SFC selezionati.

A.5 Usare l'editor FC

L'ditor grafico ISaGRAF di diagrammi di flusso permette all'utente di immettere dei programmi FC (Flow Chart) completi, con Azioni e Test (decisioni) programmati in linguaggio ST, IL o Quick LD. Il Flow Chart è un diagramma di decisione che può anche essere utilizzato per descrivere operazioni sequenziali, poiché permette di utilizzare caratteristiche avanzate, come ad esempio di non bloccare i salti all'indietro.

A.5.1 Basi del linguaggio FC

Flow Chart (**FC**) è un linguaggio grafico utilizzato per descrivere **operazioni sequenziali**. Un diagramma Flow Chart è composto di **Azioni** e **Test**. Tra Azioni e Test sono posti dei **collegamenti orientati** che rappresentano il flusso dati. Segue la lista dei componenti grafici del linguaggio Flow Chart:

Inizio di un diagramma FC: Un simbolo «**Start**» deve apparire all'inizio di un programma Flow Chart. Tale simbolo è unico e non può essere omesso. Esso rappresenta lo stato iniziale quando il diagramma viene attivato.

Fine di un diagramma FC: Un simbolo «**Stop**» deve apparire alla fine di un programma Flow Chart. Tale simbolo è unico e non può essere omesso. E' possibile che nessuna connessione sia tracciata verso il simbolo «Stop» (ad esempio in un diagramma che itera sempre), ma il simbolo «Stop» viene in ogni caso disegnato alla fine del diagramma. Esso rappresenta lo stato finale del diagramma, quando la sua esecuzione è stata terminata.

Collegamento FC: Un **collegamento** è una linea che rappresenta un flusso (Flow) tra due punti del diagramma. Un collegamento è sempre terminato da una freccia. Due collegamenti non possono essere connessi allo stesso punto sorgente della connessione.

Azione FC: Un simbolo **Azione** rappresenta azioni da eseguire. Un azione è identificata da un numero e da un nome. Due differenti oggetti dello stesso diagramma non possono avere lo stesso nome o lo stesso numero logico. Il linguaggio di programmazione per un'azione può essere ST, LD o IL. Un'azione è sempre connessa con dei collegamenti, uno che arriva su di essa, uno che parte da essa.

Test FC: Un simbolo **Test** (decisione) rappresenta una condizione booleana. Un test è identificato da un nome e da un numero logico. In accordo con la valutazione dell'espressione ST, LD o IL associata, il flusso viene diretto verso il percorso «SI» o «NO». Quando si programma in linguaggio ST, l'espressione può essere opzionalmente seguita da un simbolo di punto-e-virgola. Quando si programma in linguaggio LD, l'unica bobina presente rappresenta il risultato del test.

Sottoprogramma FC: Il sistema permette la descrizione di strutture verticali nei programmi FC. I programmi FC sono organizzati con una **gerarchia ad albero**. Ogni programma FC può chiamare altri programmi FC. Tali programmi sono chiamati **programmi figli** del programma che li ha chiamati. I programmi FC che chiamano sottoprogrammi FC sono chiamati **programmi padre** I programmi FC sono collegati assieme in gerarchia ad albero, usando una relazione «padre-figlio». Un simbolo di **sottoprogramma** in un Flow Chart rappresenta una chiamata ad un sottoprogramma del Flow Chart. L'esecuzione del programma FC chiamante viene sospesa fino a quando l'esecuzione del sottoprogramma è completata.

Azione FC specifica di I/O: Un simbolo **Azione specifica di I/O** rappresenta azioni che devono essere eseguite. Come per altre azioni, un azione specifica di I/O è definita da un nome e da un numero logico. La stessa semantica viene utilizzata per azioni standard e azioni specifiche di I/O. Lo scopo delle azioni specifiche di I/O è quello di rendere il diagramma più leggibile e di focalizzare alle parti del diagramma non-portabili. L'utilizzo delle azioni specifiche di I/O è opzionale. I blocchi specifici di I/O hanno esattamente lo stesso comportamento delle azioni standard.

Connettori FC: I simboli **Connettore** sono usati per rappresentare collegamenti tra due punti del diagramma senza doverli disegnare. Un connettore è rappresentato con un cerchio ed è connesso alla sorgente del flusso. Il disegno del connettore viene completato, sul lato appropriato (dipendente dalla direzione del flusso di dati), con l'identificazione del punto destinazione (generalmente il nome del simbolo destinazione). Un connettore riporta il flusso sempre ad un simbolo definito sul diagramma Flow Chart. Il simbolo destinazione è identificato con il suo numero logico.

Commenti FC: Un blocco **Commento** contiene del testo che non influenza la semantica del diagramma. Un commento può essere inserito ovunque in un qualsiasi spazio inutilizzato della finestra documento Flow Chart, e viene utilizzato per documentare il programma.

A.5.2 Immettere un Flow Chart

Per immettere un diagramma bisogna posizionare gli elementi (Azioni, Test, Connettori...) nell'area grafica, e tracciare dei collegamenti di flusso tra di essi.

Inserire oggetti

Per inserire un oggetto nel diagramma, selezionare il corrispondente pulsante sulla barra strumenti, quindi fare click nell'area grafica, nella posizione dove si desidera inserirlo. Si può posizionare l'elemento su un'area vuota oppure inserirlo all'interno di un collegamento di flusso facendo click sul collegamento stesso. L'inserimento all'interno di un collegamento è permesso solo per i collegamenti verticali diretti dall'alto verso il basso.

Si possono inserire i seguenti elementi base:

 Azione programmata in linguaggio ST, IL o Quick LD

 Azione specifica di IO (per evidenziare una particolare azione non portabile)

 Test (decisione) programmata in linguaggio ST, IL o Quick LD

 Connettore

 Chiamata a sottoprogramma FC

 Commento (testo descrittivo)

L'editor ISaGRAF di diagrammi Flow Chart propone inoltre una lista di strutture Flow Chart classiche. Tali strutture possono essere inserite solo su un collegamento già esistente. Non possono essere messe in un area vuota:

 If / Then / Else (selezione binaria)

 Repeat Until (attesa di una condizione)

 While (itera finché una condizione è vera)

Selezionare oggetti

La maggior parte dei comandi di editazione necessitano di selezionare degli oggetti grafici. L'editor ISaGRAF di diagrammi grafici FC permette la selezione di uno o più oggetti esistenti nell'area del diagramma. Per selezionare oggetti, bisogna utilizzare il pulsante «**Seleziona**» (identificato da una freccia) sulla barra degli strumenti. Per selezionare un oggetto, l'utente deve solo fare click sul simbolo corrispondente.

Per selezionare una lista di oggetti, trascinare il mouse sul diagramma in modo da tracciare un area rettangolare. Tutti gli oggetti grafici all'interno del rettangolo di selezione sono marcati come «**selezionati**».

Un oggetto selezionato viene disegnato in colore blu scuro, con dei piccoli quadratini neri attorno al suo simbolo grafico. E' anche possibile aggiungere o rimuovere un oggetto da una selezione multipla, mantenendo premuto il tasto SHIFT o CTRL e facendo click sul suo simbolo.

Facendo una nuova selezione, la precedente selezione di oggetti viene rimossa. Per rimuovere una selezione esistente, basta fare click con il mouse su un area vuota, fuori dal rettangolo che racchiude gli oggetti selezionati.

Per selezionare un singolo oggetto, è possibile usare i tasti freccia per muovere la selezione da un oggetto ad un altro nel diagramma. Anche i simboli Collegamento possono essere selezionati.

Inserire commenti

Degli oggetti Commento possono essere inseriti in qualunque parte vuota del diagramma. I commenti non hanno influenza sull'esecuzione del programma. Consentono una maggiore leggibilità del diagramma. Per inserire un blocco Commento, selezionare il corrispondente pulsante nella barra strumenti, e fare click nel punto del diagramma dove il commento deve essere posizionato. Per immettere o cambiare il testo di un commento fare doppio click su di esso. Quando si immette

del testo in un commento, non è necessario utilizzare i caratteri «**(***» e «***)**». Un blocco Commento può essere ridimensionato trascinando gli angoli del suo bordo, dopo averlo selezionato.

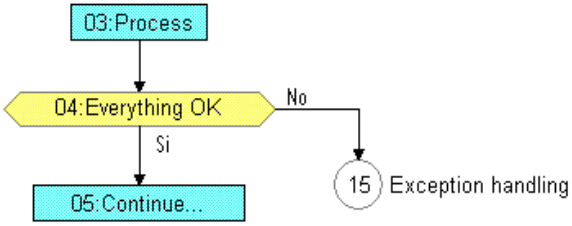
Disegnare linee di flusso

Per disegnare un flusso di collegamento tra elementi esistenti, selezionare questo pulsante sulla barra strumenti. Un collegamento deve sempre essere disegnato nella direzione del flusso. Per prima cosa si seleziona un punto di uscita non connesso di un elemento FC, quindi si trascina il mouse fino al punto destinazione del collegamento. Il punto destinazione può essere la parte superiore (punto di input) di un elemento FC non connesso, oppure un qualsiasi punto su un collegamento già esistente. I punti di convergenza tra collegamenti sono mostrati nel diagramma Flow Chart con dei piccoli cerchietti di colore grigio. Anche i punti di convergenza possono essere selezionati e mossi per ridisporre il diagramma.

Usare connettori

L'editor ISaGRAF dei diagrammi Flow Chart permette di utilizzare dei connettori grafici, in sostituzione di un flusso di collegamento visibile. Gli oggetti Connettore possono essere molto utili per evitare collegamenti molto lunghi e per migliorare la leggibilità del diagramma. Un Connettore non può essere utilizzato per stabilire un collegamento con un altro programma FC.

Un Connettore viene immesso nel diagramma come gli altri oggetti FC. Viene rappresentato con un cerchio contenente il riferimento numerico all'elemento destinazione. Il testo della descrizione breve dell'elemento destinazione viene mostrato vicino al simbolo del Connettore.



Muovere oggetti

Per muovere degli oggetti nel diagramma, bisogna selezionarli, e quindi trascinare il mouse all'interno del diagramma. Si può spostare sia un elemento singolo che una selezione multipla. Gli elementi non possono essere sovrapposti. Non è possibile utilizzare lo spostamento di oggetti per connetterli ad un Collegamento esistente.

Quando un singolo elemento (Azione, Test...) viene spostato, l'editor ISaGRAF sposta automaticamente tutti gli oggetti collegati e posti sotto ad esso. Questa caratteristica non interviene nel caso di una selezione multipla.

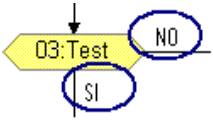
Ridimensionare oggetti

Ogni elemento grafico di un diagramma FC, eccetto i simboli «Start», «Stop» e i Connettori, può essere liberamente ridimensionato. Per ridimensionare un elemento, bisogna innanzitutto selezionarlo. Quindi basta trascinare con il mouse i quadratini neri disegnati sul suo bordo per cambiarne la dimensione.

Quando un elemento è connesso ad un flusso di collegamento, il ridimensionamento orizzontale agisce su entrambi i bordi destro e sinistro, di modo che l'elemento sia sempre correttamente centrato sul collegamento.

Invertire le uscite di un Test

Si possono invertire le uscite SI / NO di un oggetto Test. Per far ciò, basta fare doppio click sulla scritta «SI» oppure sulla scritta «NO» mostrate vicino al simbolo di Test.



A.5.3 Lavorare su un diagramma esistente

I comandi del menù «**Modifica**» sono utilizzati per cambiare o per completare un diagramma esistente. La maggior parte di questi comandi agisce sugli elementi correntemente selezionati nel diagramma.

Correggere un diagramma

Il tasto CANC può essere utilizzato per rimuovere gli elementi selezionati. I collegamenti pendenti sono cancellati con gli elementi selezionati. Usare il comando «**Modifica / Annulla**» per ripristinare gli elementi dopo un comando CANC. Il comando CANC può anche essere applicato ad un gruppo di elementi selezionati nel diagramma. I comandi «**Taglia**», «**Copia**», «**Incolla**» del menù «**Modifica**» sono utilizzati per muovere o copiare gli elementi selezionati.

Trova e sostituisci

I comandi «**Modifica / Trova Sostituisci**» possono essere utilizzati per trovare o per sostituire stringhe di testo nell'intero programma (tutte le Azioni e i Test programmati in linguaggio ST, IL o Quick LD). La finestra di dialogo Trova/Sostituisci è utilizzata per immettere il testo che deve essere cercato e per aprire direttamente la sezione di programmazione dove il testo compare.

Accesso diretto ad un elemento

Il comando «**Modifica / Vai a**» permette all'utente di accedere ad un elemento grafico esistente nel diagramma. La sezione di diagramma visualizzata viene automaticamente riposizionata di modo che l'elemento sia visibile. L'elemento, una volta raggiunto, viene selezionato.

Rinumerazione degli elementi

Il comando «**Modifica / Rinumera**» viene utilizzato per rinumerare gli elementi del diagramma. Ogni elemento FC messo nel diagramma è identificato con un unico numero di riferimento. I numeri di riferimento sono allocati dall'editor ogni volta che dei nuovi elementi vengono inseriti. Il comando «**Modifica / Rinumera**» permette di riaggiustare la numerazione degli elementi in accordo con la loro posizione nel diagramma. La progressione numerica avviene dall'alto verso il basso e da sinistra verso destra.

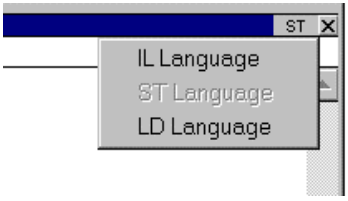
A.5.4 Programmazione di livello 2

Per immettere la programmazione di livello 2, l'utente deve fare doppio click sul simbolo dell'Azione o del Test. La finestra di programmazione di livello 2 viene aperta alla destra del diagramma FC. La linea di separazione tra il diagramma FC e la finestra di programmazione di livello 2 può essere liberamente spostata. Si possono aprire una o contemporaneamente due aree di livello 2. I seguenti comandi sono disponibili dalla tastiera, dal mouse o dal menù «Modifica»:

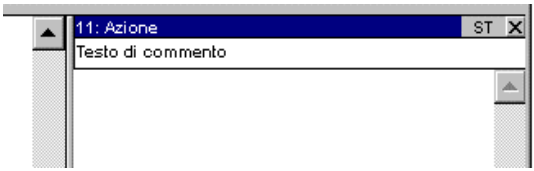
	Tastiera	*Mouse*	*Menù «Modifica»*
Apre nell'ultima finestra	INVIO	Doppio Click	Modifica livello 2
Apre finestra separata	CTRL+INVIO	CTRL + Doppio Click	Modifica livello 2 in una finestra separata

Quando due finestre di livello 2 sono visibili, la separazione tra esse può essere liberamente spostata. Il pulsante alla destra della barra del titolo chiude la finestra di livello 2.

Il linguaggio predefinito del Livello 2 è **ST** (Structured Text). Il linguaggio di programmazione può anche essere **IL** o **Quick LD**. Il nome del linguaggio selezionato viene mostrato in un piccolo riquadro nella barra del titolo della finestra di livello 2. Facendo click su tale riquadro si può cambiare il linguaggio di programmazione, oppure si può usare il comando «**Opzioni / Imposta linguaggio di livello 2**» del menù principale. Questo comando è valido solo se la finestra di programmazione di livello 2 è vuota.



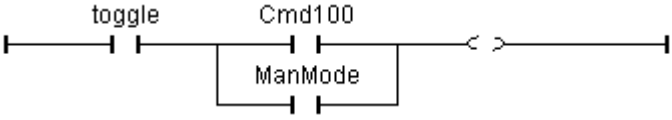
Una linea di testo appare nella parte superiore della finestra di livello 2, viene usata per immettere un breve testo di descrizione. Questo testo verrà visualizzato come commento IEC nel disegno dei simboli FC. E' molto utile ed è utilizzato da altri comandi, come ad esempio «Vai a» ed anche nella stampa di (Azioni e Test) documenti FC.



Il comando «**Opzioni / Rinfresca diagramma**» può essere usato ogni volta che una finestra di livello 2 è aperta per rinfrescare il diagramma FC con le modifiche apportate ai programmi di livello 2.

A.5.5 Programmazione di livello 2 con Quick LD

L'editor Quick LD è disponibile per la programmazione di livello 2. Nel caso di un Test, il diagramma LD è composto solo da un ramo, con una sola bobina che rappresenta il risultato del test. Il nome del Test non viene ripetuto sul simbolo della bobina. Segue un esempio di Test programmato in linguaggio Quick LD.



Quando si programma in linguaggio Quick LD, si possono utilizzare le frecce sulla tastiera per muovere la selezione sulla griglia di programmazione, quindi si possono usare i seguenti tasti funzione per l'inserimento dei simboli:

F2:..... aggiunge un contatto dopo il simbolo evidenziato / inizia il ramo
F3:..... aggiunge un contatto prima del simbolo evidenziato
F4:..... aggiunge un contatto in parallelo con il simbolo evidenziato
F5:..... aggiunge una bobina in parallelo con quella selezionata(non per i Test, in tal caso la bobina è unica)
F6:..... aggiunge un blocco dopo il simbolo evidenziato
F7:..... aggiunge un blocco prima del simbolo evidenziato
F8:..... aggiunge un blocco in parallelo con il simbolo evidenziato
F9:..... aggiunge un salto in parallelo con la selezione (non per i Test)

Un salto porta al nome di un ramo LD. Il nome del ramo destinazione può essere immesso premendo INVIO quando è selezionata la testa del ramo di partenza. L'editor ISaGRAF memorizza le etichette dei rami che sono stati precedentemente immessi, sia che siano state specificate per il nome di un ramo o per un'operazione di salto. La finestra di dialogo **Salto / Etichetta** permette di immettere una nuova etichetta oppure di selezionarne una esistente. Se si immette un nuovo nome, questo verrà automaticamente aggiunto alla lista. Il pulsante «**Rimuovi**» viene usato per rimuovere il nome selezionato dalla lista, ma non rimuove l'etichetta dal ramo selezionato nel diagramma. Per far questo bisogna premere **OK** quando la casella di edit è vuota.

Per immettere i simboli LD, si possono usare i pulsanti nella barra strumenti LD invece di premere i tasti funzione.

Premendo INVIO quando è selezionato un contatto o un parametro I/O di un blocco funzione, si può scegliere la variabile da associare oppure si può immettere un valore costante.
Premendo INVIO quando è selezionato un blocco, si può scegliere il tipo di blocco funzione da utilizzare.
Facendo doppio click su un oggetto si ottiene lo stesso effetto.

Premendo Control + BARRA SPAZIO quando un contatto è selezionato, si può cambiare il tipo di contatto o bobina (diretto, inverso). Fare riferimento al capitolo «Usare l'editor Quick LD» per ulteriori dettagli sulle capacità dell'editor Quick LD.

A.5.6 Opzioni di visualizzazione

Il comando «**Opzioni / Disposizione**» apre una finestra di dialogo dove sono raggruppati tutti i parametri e le opzioni riguardanti l'ambiente di lavoro dell'editor e il tracciamento del diagramma. Usare le caselle di selezione del gruppo **Ambiente di lavoro** per visualizzare o nascondere la barra strumenti e la barra di stato. Le opzioni del gruppo **Documento** permettono di mostrare o nascondere i punti della griglia e di visualizzare il diagramma a colori o in bianco e nero.

Utilizzare il pulsante **«Zoom»** della barra strumenti per cambiare il rapporto di zoom corrente. Questo comando è disponibile anche quando si lavora su un programma Quick LD associato ad un'Azione o ad un Test.

Utilizzare il pulsante **«Visualizza / Nascondi griglia»** della barra strumenti per mostrare o nascondere i punti della griglia. Questo comando è disponibile anche quando si lavora su un programma Quick LD associato ad un'Azione o ad un Test.

Utilizzare il comando «**Opzioni / Carattere**» per selezionare il nome del font di caratteri da utilizzare in tutti i documenti ISaGRAF. Quando questo comando viene richiamato da un programma ST o IL, si può specificare la dimensione del carattere. Quando si seleziona il font di caratteri per una finestra grafica (FC o Quick LD), la dimensione e lo stile non sono rilevanti e non è necessario specificarli, infatti gli editor grafici di ISaGRAF adattano sempre la dimensione del carattere in accordo con il rapporto di zoom corrente.

A.6 Usare l'editor Quick LD

Il linguaggio LD permette la rappresentazione grafica di espressioni logiche/booleane. Gli operatori booleani vengono rappresentati esplicitamente nella topologia del diagramma. Le variabili booleane di ingresso sono collegate ai simboli grafici dei contatti. Le variabili booleane di uscita sono collegate ai simboli grafici delle bobine. L'editor Quick LD ISaGRAF facilita l'inserimento del diagramma LD e permette l'uso sia della tastiera che del mouse. L'editor Quick LD ISaGRAF provvede a collegare e sistemare automaticamente gli elementi sui rami. Non c'è bisogno di disegnare le connessioni manualmente. L'editor Quick LD sistema anche i rami nel diagramma, così da ottimizzare lo spazio usato dal diagramma stesso.

A.6.1 Elementi di base del linguaggio LD

Un programma LD viene espresso come una lista di **rami** (da cui segue il nome Ladder Diagram, cioè Diagramma a scala) in cui sono sistemati contatti e bobine. Di seguito sono descritti i componenti di base di un diagramma LD.

Inizio di un ramo (barra di alimentazione sinistra)

Ogni ramo inizia con una barra di alimentazione sinistra, che rappresenta lo stato iniziale «TRUE». L'editor Quick LD ISaGRAF provvede automaticamente a creare la barra di alimentazione sinistra nel momento in cui viene posizionato il primo contatto del ramo. Ad ogni ramo può essere assegnato un nome logico, che può essere usato come etichetta nelle istruzioni di salto.

Contatti

Un contatto modifica il flusso dati booleano in relazione allo stato di una variabile booleana. Il nome della variabile è visibile sopra al simbolo del contatto. L'editor Quick LD ISaGRAF incorpora i seguenti tipi di contatti:

 ..... contatto diretto

 ..... contatto inverso

 ..... contatto con riconoscimento di fronte positivo (in salita)

 ..... contatto con riconoscimento di fronte negativo (in caduta)

Bobine

Una bobina rappresenta un'azione. Lo stato del ramo (lo stato del collegamento alla sinistra della bobina) viene usato per assegnare un valore ad una variabile booleana. Il nome della variabile è visibile sopra al simbolo della bobina. L'editor Quick LD ISaGRAF incorpora i seguenti tipi di bobina:

 ..... bobina diretta

 ..... bobina inversa (negata)

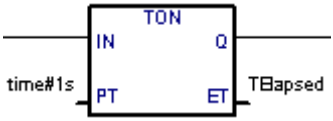
 ..... bobina di tipo «Set»

 ..... bobina di tipo «Reset»

-(P)- ..... bobina con riconoscimento di fronte positivo (in salita)
-(N)- ..... bobina con riconoscimento di fronte negativo (in caduta)

Blocchi funzione

In un diagramma LD, un blocco può rappresentare una funzione, un blocco funzione, un sottoprogramma o un operatore. I primi parametri di entrata e di uscita sono sempre collegati al ramo. I rimanenti parametri di input ed output sono scritti in modo testuale all'esterno della casella del blocco.



Fine di un ramo (barra di alimentazione destra)

Un ramo termina con una barra di alimentazione destra. L'editor Quick LD provvede automaticamente ad inserire la barra di alimentazione destra quando viene posizionata una bobina.

Simbolo di salto

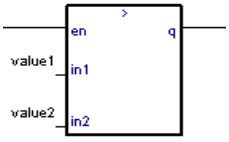
Un simbolo di salto fa sempre riferimento ad una etichetta che si trova in un diverso punto dello stesso diagramma LD. L'etichetta è riportata a fianco del simbolo di salto. Quando la connessione sinistra ha valore «TRUE», l'esecuzione del diagramma salta direttamente alla etichetta destinazione. Si ricorda la pericolosità dei salti all'indietro, che potrebbero, in alcuni casi, causare il blocco del ciclo PLC.

Simbolo di Return

Il simbolo di return viene posizionato alla fine del ramo. Indica di fermare l'esecuzione del programma quando lo stato del ramo vale «TRUE».

Input «EN»

Con alcuni operatori, funzioni o blocchi funzione, il primo input non è di tipo booleano. Poiché il primo input deve essere sempre connesso al ramo, nella prima posizione viene automaticamente inserito un altro input chiamato «**EN**». Il blocco viene eseguito solo se l'input **EN** vale «TRUE». Di seguito viene proposto l'esempio di un operatore di confronto affiancato dall'equivalente codice espresso in linguaggio ST:

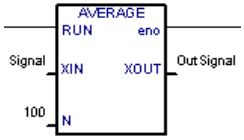



```
IF stato_ramo THEN
   q := (valore1 > valore2);
ELSE
   q := FALSE;
END_IF;
(*prosegue il ramo con lo stato q *)
```

Output «ENO»

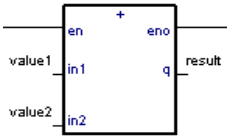
Con alcuni operatori, funzioni o blocchi funzione, il primo output non è di tipo booleano. Poiché il primo output deve essere sempre connesso al ramo, nella

prima posizione viene automaticamente inserito un altro output chiamato «**ENO**». L'output **ENO** assume sempre lo stesso stato del primo input del blocco. Di seguito viene proposto l'esempio con un blocco funzione di tipo AVERAGE (MEDIA) affiancato all'equivalente codice espresso in linguaggio ST:




```
AVERAGE(stato_ramo, Segnale, 100);
SegnaleOut := AVERAGE.XOUT;
eno := stato_ramo;
(*prosegue il ramo con lo stato eno*)
```

In alcuni casi è richiesto l'uso di entrambi **EN** ed **ENO**. Di seguito viene proposto un esempio con un operatore aritmetico affiancato all'equivalente codice espresso in linguaggio ST:




```
IF stato_ramo THEN
   resultato := (valore1 + valore2);
END_IF;
eno := stato_ramo;
(*prosegue il ramo con lo stato eno *)
```

Limiti dell'editor Quick LD

L'editor Quick LD ISaGRAF non permette di continuare un ramo (inserire contatti o bobine) alla destra di una bobina. Qualora si debbano avere più uscite sullo stesso ramo, è necessario disegnare in parallelo le corrispondenti bobine.

A.6.2 Inserire un diagramma LD

Tutti i comandi dell'editor Quick LD Si possono ottenere sia con l'uso della tastiera che del mouse.

La griglia di modifica

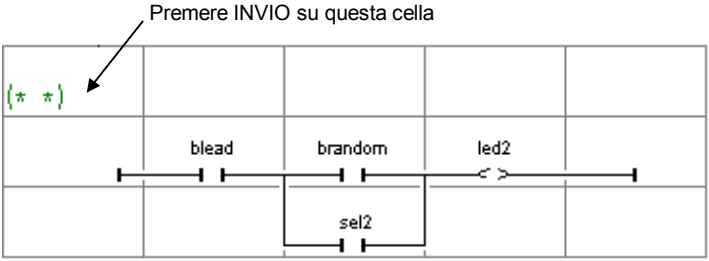
Il diagramma LD viene inserito in una matrice logica. Ogni cella della matrice può contenere un solo simbolo LD. Per spostare la selezione corrente, si usano i tasti cursore della tastiera o si fa un click sulla cella voluta. La cella evidenziata è segnata con colorazione inversa. Per alcune operazioni del tipo taglia/copia/incolla, è possibile evidenziare più celle. Per far questo con il mouse, è sufficiente trascinare il cursore del mouse sul diagramma. Con la tastiera bisogna usare i tasti cursore tenendo premuto il tasto SHIFT.

Iniziare un nuovo ramo

Per aggiungere un nuovo ramo ad un diagramma, spostare la selezione dopo l'ultimo ramo esistente ed inserire un contatto (con il tasto F2 o premendo il corrispondente pulsante della barra strumenti tasti funzione). Viene creato un nuovo ramo con un contatto ed una bobina.

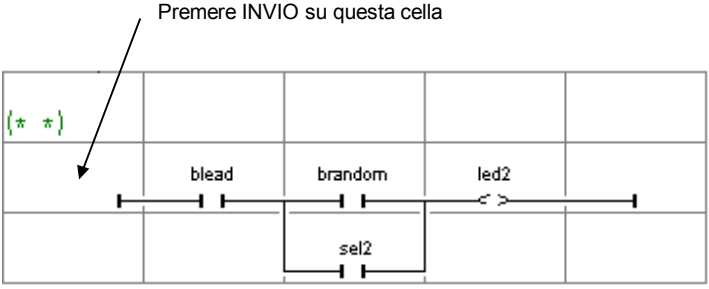
Inserire un commento al ramo

Ogni ramo può essere documentato con al massimo due righe di testo. Per inserire un testo di commento al ramo, posizionare la cella di selezione sopra il ramo e premere il tasto INVIO, o fare doppio click con il mouse sulla cella voluta:

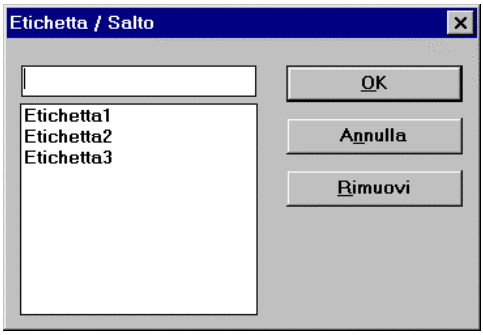


Inserire l'etichetta del ramo

Ogni ramo può essere identificato da un nome, che può essere usato come etichetta destinazione per le operazioni di salto. Per inserire o cambiare l'etichetta di un ramo, muovere la selezione sull'inizio di un ramo e premere il tasto INVIO, o fare doppio click con il mouse sulla cella voluta.



L'editor Quick LD ISaGRAF memorizza le etichette dei rami già introdotte, se sono state specificate come nome di un ramo o per operazioni di salto. La seguente finestra di dialogo permette di inserire una nuova etichetta o di selezionarne una esistente.



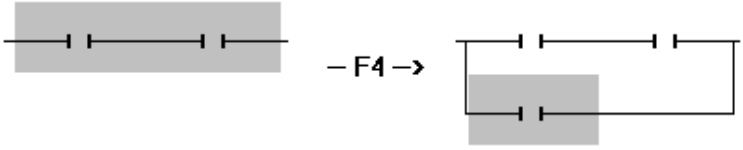
Se viene specificato un nuovo nome, questo viene automaticamente inserito nella lista. Il pulsante «**Rimuovi**» si usa per eliminare dalla lista il nome correntemente evidenziato, ma non elimina dal diagramma l'etichetta del ramo evidenziato. Per fare questo, premere **OK** quando l'elenco e la riga di inserzione sono vuote.

Inserire simboli su un ramo

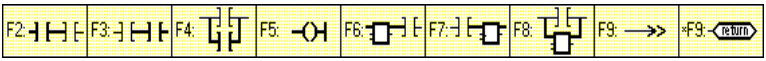
L'inserimento di simboli (contatti, bobine, blocchi …) su un ramo esistente avviene sempre sulla selezione corrente. L'inserimento avviene evidenziando la posizione di una cella valida all'interno del ramo e premendo uno dei seguenti tasti funzione:
F2..... un contatto prima del simbolo evidenziato (a sinistra)
F3..... un contatto dopo il simbolo evidenziato (a destra)
F4..... un contatto in parallelo con il simbolo evidenziato
F6..... un blocco prima del simbolo evidenziato (a sinistra)
F7..... un blocco dopo il simbolo evidenziato (a destra)
F8..... un blocco in parallelo con il simbolo evidenziato

I comandi che seguono hanno validità quando è evidenziato l'output del ramo (bobina):
F5..... aggiunge una bobina in parallelo con la selezione
F9..... aggiunge un salto in parallelo con la selezione
SHIFT + F9 ..... aggiunge un simbolo Return in parallelo con la selezione

Per quanto riguarda gli inserimenti in parallelo (F4/F8), se più contatti di un ramo sono evidenziati contemporaneamente, il simbolo viene inserito in parallelo con il gruppo degli elementi evidenziati. Di seguito viene proposto un esempio:



Per l'inserimento dei simboli nel diagramma è anche possibile utilizzare la barra strumenti. Facendo click sul tipo di simbolo voluto sulla barra, questo viene inserito nella posizione corrente:



Inserire simboli

Evidenziando un contatto od una bobina e premendo INVIO, è possibile assegnargli una variabile. Usando il mouse bisogna fare doppio click sul contatto o sulla bobina. Appare un riquadro che mostra un elenco di variabili. Far riferimento al capitolo «Ulteriori informazioni sugli editor di programmi» in questo stesso documento per l'uso di questa finestra di dialogo.
Selezionando una funzione, blocco funzione o un operatore e premendo INVIO è possibile scegliere il tipo di blocco funzione da utilizzare.
Per associare una variabile ad un parametro di ingresso o di uscita di un blocco, bisogna selezionare l'uscita o l'ingresso, all'esterno del rettangolo del blocco, quindi premere INVIO.
Alternativamente è possibile fare un doppio click con il mouse sull'elemento selezionato.

Normalmente per l'inserimento di testo vengono utilizzate delle finestre di dialogo che includono liste delle variabili o dei blocchi. Se è selezionata la voce "**Immissione manuale da tastiera**" nel menù "**Opzioni**", i simboli di variabili e i nomi dei blocchi sono immessi direttamente in una riga di testo. Immettere il nuovo testo e premere il tasto "**Invio**" per validarlo, oppure premere il tasto "**Esc**" per uscire dalla riga di testo senza effettuare modifiche. La riga di testo della modalità "Input manuale da tastiera" non può essere chiusa con il mouse.

Cambiare il tipo di contatti e bobine

La voce «**Modifica / Cambia tipo di bobina/contatto**» permette di cambiare il tipo del contatto o bobina evidenziata. Un contatto può essere diretto, inverso, con riconoscimento di fronte positivo o negativo. Una bobina può essere diretta, inversa, con riconoscimento di fronte positivo o negativo. Lo stesso effetto si ottiene premendo la barra spazio.

Inserire un ramo in un diagramma

La voce «**Modifica / Inserisci ramo**» permette di inserire un nuovo ramo nel diagramma, prima di quello evidenziato. Il ramo viene inizializzato con un contatto ed una bobina.

A.6.3 Operare su un diagramma esistente

Le voci del menù «**Modifica**» permettono di modificare o completare un diagramma esistente. Questi comandi hanno effetto sul diagramma correntemente evidenziato.

Correggere un diagramma

Il tasto CANC permette di eliminare gli elementi evidenziati. Non è possibile eliminare un simbolo di bobina, salto o return, qualora sia l'unico output del ramo. Per ripristinare gli elementi dopo un comando CANC, usare la voce «**Modifica / Annulla**». Il comando CANC ha effetto anche su gruppi di elementi evidenziati sul diagramma. Il comando CANC può essere impiegato anche per cancellare, quando

evidenziato, il testo di commento al ramo. Se è evidenziato l'inizio del ramo, allora il comando CANC cancella l'intero ramo.

Copiare i simboli

Le voci «**Taglia**», «**Copia**», «**Incolla**» del menù «**Modifica**» permettono di spostare o copiare gli elementi evidenziati. La voce «**Modifica / Incolla speciale**» di inserire gli elementi degli appunti:

- prima dell'elemento evidenziato (a sinistra)
- dopo l'elemento evidenziato (a destra)
- in parallelo con l'elemento evidenziato

Gestire l'intero ramo

Nel caso a selezione si trovi all'inizio del ramo (barra di alimentazione sinistra), i comandi di editing (cancella, copia, taglia) hanno effetto sull'intero ramo. Questo permette di gestire con semplicità i rami sul diagramma muovendo la selezione sulla prima colonna. È anche possibile estendere la selezione verticalmente in modo da includere più rami. In questo caso i comandi di modifica possono essere applicati ad una lista di rami completi.

Trovare e sostituire

Le voci «**Modifica / Trova**» e «**Modifica / Sostituisci**» permettono di trovare e sostituire le parti testuali del diagramma. La ricerca ha successo solo su nomi completi e funziona su contatti, bobine, nomi di blocchi, blocchi di parametri e etichette di esecuzione. Non può essere usata per trovare una stringa all'interno del commento di un ramo. Il comando Sostituisci non può essere usato per cambiare la tipologia di un blocco. La ricerca può avvenire sia verso l'alto che verso il basso a partire dalla posizione della selezione corrente. L'algoritmo di ricerca «cicla» quando raggiunge i limiti del diagramma. Sono disponibili i seguenti «tasti scorciatoia» per la ricerca veloce dei nomi delle variabili:

ALT + F2 trova l'elemento successivo avente lo stesso nome dell'elemento correntemente selezionato. Questa possibilità è applicabile anche a blocchi funzione e etichette di rami.

ALT + F5 trova la successiva bobina con lo stesso nome della bobina correntemente selezionata. Principalmente usato in modo debug, per trovare rapidamente i rami in relazione con una variabile sospetta.

A.6.4 Opzioni di visualizzazione

Le voci del menù «**Opzioni**» permettono di personalizzare il disegno sullo schermo dei diagrammi LD e di nascondere o rendere visibili alcuni tipi di informazioni.

Commenti ai rami

La voce «**Opzioni / Commenti ai rami**» permette di nascondere o rendere visibili i commenti ai rami per l'intero diagramma. Torna utile qualora si desideri una visione compressa di un diagramma molto grande, poiché, sulla griglia di modifica, ogni commento occupa una riga. Questa opzione non ha effetto sui contenuti dei commenti esistenti e si può attivare e disattivare in qualsiasi momento.

Nomi ed alias

Ogni variabile, associata ad un contatto, ad una bobina o ad un blocco di parametri I/O, viene identificata dal proprio nome simbolico. L'editor Quick LD ISaGRAF introduce anche il concetto di **alias** per ciascuna variabile. L'alias della variabile è costituito dal testo di commento alla variabile stessa, troncato prima del primo carattere '**:**' e limitato a 16 caratteri. Di seguito sono proposti alcuni esempi:

```
commento alla variabile:          alias:
testo breve                       testo breve
testo lungo senza separatori      testo lungo senz
testo breve: descrizione lunga    testo breve
```

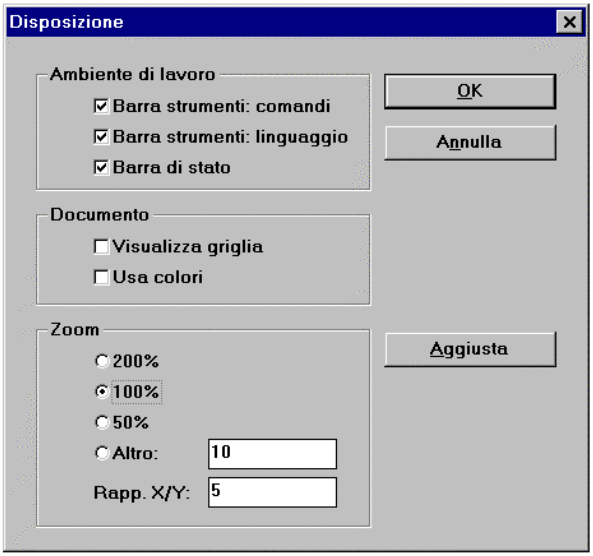
Gli alias non hanno alcun effetto sull'esecuzione del diagramma LD e devono essere considerati, dal punto di vista sintattico, alla stregua di commenti. L'alias di una variabile viene automaticamente estratto dal commento della variabile nel momento in cui viene selezionato il nome dall'elenco delle variabili. Usare la voce «**Opzioni / Contatti e bobine**» per scegliere il modo di visualizzazione per l'identificazione delle variabili. Sono disponibili i seguenti modi:

- solo i nomi delle variabili
- solo gli alias delle variabili
- nomi ed alias contemporaneamente

L'editor Quick LD non aggiorna automaticamente i documenti LD quando gli alias delle variabili sono cambiati nel dizionario. Per aggiornare tutti gli alias nel diagramma editato, usare il comando «**Opzioni / Contatti e bobine / Aggiorna alias**». Inoltre è possibile aggiornare automaticamente tutti gli alias utilizzati ogni volta che un programma Quick LD viene aperto, selezionando «**Aggiorna sempre all'apertura**» dal menù «**Opzioni / Contatti e bobine**».
Attenzione: Impostando questa opzione viene incrementato significativamente il tempo speso per aprire un programma.

Opzioni di disegno

La voce «**Opzioni / Disposizione**» apre una finestra di dialogo in cui sono raggruppati tutti i parametri e le opzioni relative all'area di lavoro dell'editor ed il disegno del diagramma grafico LD:



I pulsanti opzione del gruppo **«Ambiente di lavoro»** di lavoro permettono di rendere visibile o nascondere la barra strumenti dell'editor, la barra di stato e la barra dei tasti funzione. Le opzioni del gruppo **«Documento»** permettono di visualizzare o nascondere i punti della griglia di modifica e di abilitare/disabilitare l'uso dei colori nel disegno.

Le opzioni del gruppo **«Zoom»** permettono di scegliere il fattore di ingrandimento. È anche possibile usare il pulsante **«Zoom»** della barra strumenti dell'editor per indicare uno dei fattori di ingrandimento predefiniti.

È anche possibile personalizzare il rapporto X/Y dell'aspetto delle celle. Questa ultima opzione può essere usata per ridurre l'ampiezza predefinita delle celle, usando nomi brevi per le variabili. Per modificare il rapporto X/Y senza attivare la finestra di dialogo Disposizione, si usa il pulsante **«Larghezza cella»** della barra strumenti dell'editor.

Il comando "**Opzioni / Carattere**" permette di selezionare il tipo di caratteri (font) che deve essere usato in tutti i documenti grafici di ISaGRAF. Lo stile e la dimensione dei caratteri non sono rilevanti e non serve specificarli, gli editor grafici di ISaGRAF calcolano sempre la dimensione del carattere in accordo con il rapporto di zoom selezionato.

A.7 Usare l'editor FBD/LD

In questo capitolo vengono descritte le caratteristiche ed i comandi specifici dell'editor FBD/LD. Gli altri comandi, comuni anche agli altri editor, vengono descritti nel capitolo «Ulteriori informazioni sugli editor dei programmi», in questo stesso documento

L'editor grafico FBD/LD ISaGRAF consente di inserire programmi FBD completi, compresi di eventuali parti in LD. La possibilità di operare contemporaneamente in modo grafico e testuale, permette di lavorare sia sul diagramma che sui corrispondenti ingressi ed uscite.
Poiché questo editor è principalmente dedicato al linguaggio FBD, è preferibile usare l'editor Quick LD ISaGRAF per i diagrammi esclusivamente LD.

A.7.1 Elementi di base dei linguaggi FBD/LD

Il linguaggio **FBD** è la rappresentazione grafica di molti differenti tipi di formule. Gli **Operatori** vengono rappresentati come caselle funzione rettangolari. Le funzioni di input sono connesse al lato sinistro di tali caselle, mentre le funzioni di output al lato destro. Gli input e gli output del diagramma sono connessi alle caselle funzione mediante **collegamenti logici**. L'uscita di una casella funzione può essere connessa all'ingresso di un'altra casella.

Il linguaggio **LD** permette di rappresentare graficamente espressioni booleane. Gli operatori booleani **AND**, **OR**, **NOT** vengono esplicitamente rappresentati dalla topologia del diagramma. Le variabili di ingresso booleane sono collegate a **contatti** resi graficamente. Le variabili booleane di uscita sono collegate a **bobine**. Contatti e bobine sono collegati tra loro e alle barre di alimentazione sinistra e destra mediante **linee orizzontali**. Ogni segmento di linea possiede uno stato booleano «FALSE» (falso) o «TRUE» (vero). Lo stato booleano è il medesimo per tutti i segmenti connessi tra loro direttamente. Qualsiasi linea orizzontale collegata alla **barra verticale di alimentazione sinistra** è in stato «TRUE».

I diagrammi LD e FBD vengono sempre valutati da sinistra verso destra e dall'alto verso il basso. Consultare la «Guida di riferimento ai linguaggi» ISaGRAF per maggiori dettagli sui linguaggi LD e FBD. Questi sono i componenti grafici di base dei linguaggi LD e FBD supportati dall'edito FBD/LD:

Barra di alimentazione sinistra

Ogni ramo deve essere connesso a sinistra con la **barra di alimentazione sinistra**, che rappresenta lo stato iniziale «VERO». L'editor FBD ISaGRAF permette di connettere qualunque simbolo booleano alla barra di alimentazione sinistra.

Barra di alimentazione destra

Le bobine possono essere connesse a destra con la **barra di alimentazione destra**. Questa caratteristica è facoltativa, usando l'editor FBD/LD ISaGRAF. Una

bobina, anche se non connessa a destra, comprende comunque una barra di alimentazione destra.

Connessione verticale LD «OR»

La connessione verticale accetta collegamenti multipli sulla sinistra e sulla destra. Ogni collegamento a destra equivale alla combinazione OR dei collegamenti a sinistra.

Contatti

Un contatto modifica il flusso dati booleano in relazione allo stato di una variabile booleana. Il nome della variabile è visibile sopra al simbolo del contatto. L'editor FBD/LD ISaGRAF incorpora i seguenti tipi di contatti:

- ..... contatto diretto
- ..... contatto inverso
- ..... contatto con riconoscimento di fronte positivo (in salita)
- ..... contatto con riconoscimento di fronte negativo (in discesa)

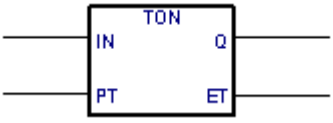
Bobine

Una bobina rappresenta un'azione. Deve essere connessa a sinistra ad un simbolo booleano come ad esempio un contatto. Il nome della variabile è visibile sopra al simbolo della bobina. L'editor FBD/LD ISaGRAF incorpora i seguenti tipi di bobina:

- ..... bobina diretta
- ..... bobina inversa (negata)
- ..... bobina di tipo «Set»
- ..... bobina di tipo «Reset»

Blocchi funzione

Un blocco può rappresentare una funzione, un blocco funzione, un sottoprogramma o un operatore. Ingressi ed uscite devono essere collegati a variabili, contatti o bobine, o ad altri blocchi di ingresso o di uscita. I nomi dei parametri formali sono visibili all'interno della casella del blocco.



Etichette

Le etichette possono essere posizionate dovunque sul diagramma. Le etichette rappresentano la destinazione per le istruzioni di salto, per cambiare l'ordine di esecuzione del diagramma. Le etichette non hanno connessioni con altri elementi. Al fine di migliorare la leggibilità del diagramma, si raccomanda di disporre le etichette alla sinistra del diagramma.

Salti

Un simbolo di salto fa sempre riferimento ad una etichetta che si trova in un diverso punto del diagramma. Quando la connessione sinistra ha valore «TRUE», l'esecuzione del diagramma salta direttamente alla etichetta destinazione. Si ricorda la pericolosità dei salti all'indietro, che potrebbero, in alcuni casi, causare il blocco del ciclo PLC.

Simbolo di Return

Il simbolo di return è connesso ad un «punto» booleano. Indica di fermare l'esecuzione del programma quando lo stato del ramo vale «TRUE».

Variabili

Le variabili sul diagramma vengono rappresentate all'interno di piccole caselle, collegate a sinistra e a destra con gli altri elementi del diagramma.

Variabili a sola lettura (senza connessione a sinistra)

Alcune variabili vengono disegnate in caselle prive della connessione a sinistra. Questo tipo di disegno va privilegiato per le variabili in ingresso e le espressioni costanti, poiché provvede al controllo dei collegamenti non validi.

Connessioni

Le linee di connessione sono tracciate tra gli elementi del diagramma. I collegamenti sono sempre tracciati da un punto di uscita verso un punto ingresso (in direzione del flusso di dati).

Connessioni con negazione booleana

Alcuni collegamenti booleani sono rappresenti con un piccolo cerchio all'estremità destra. Esso rappresenta una negazione booleana dell'informazione trasmessa dal collegamento.

Angoli definibili

Si possono definire dei punti sui collegamenti. Essi consentono all'utente di controllare manualmente il tracciamento di una connessione. Se gli angoli non vengono indicati, l'editor ISaGRAF FBD/LD usa un algoritmo di tracciamento predefinito.

A.7.2 Inserimento di un diagramma FBD

Per inserire un diagramma, si devono disporre gli elementi (blocchi, variabili, contatti, bobine,..) nell'area del grafico e tracciare i collegamenti tra loro.

Inserire oggetti

Per inserire un oggetto nel diagramma, selezionare il pulsante corrispondente della barra strumenti e fare click sull'area grafica, dove lo si vuole collocare.

Selezione oggetti

Per la maggior parte dei comandi di editing è necessario selezionare gli oggetti grafici. L'editor grafico LD/FBD ISaGRAF permette la selezione di uno o più oggetti

nell'area del diagramma. Per selezionare gli oggetti, è necessario scegliere l'opzione **«Seleziona»** (pulsante con la freccia) nella barra strumenti dell'editor. Per evidenziare un oggetto basta semplicemente fare click sul suo simbolo. Per selezionare una lista di oggetti portare il mouse nel diagramma e selezionare un'area rettangolare. Tutti gli oggetti grafici che hanno intersezione con il rettangolo selezionato vengono segnati come «selezione». Un oggetto selezionato è evidenziato con piccoli angoli neri attorno al suo simbolo grafico. Nel fare una nuova selezione vengono annullate tutte le precedenti selezioni di oggetti. Per togliere l'attuale selezione fare semplicemente click su un'area vuota fuori dal rettangolo entro il quale sono collocati gli oggetti selezionali.

Aggiungere commenti

Possono essere inseriti commenti ovunque nel diagramma. Essi non hanno alcuna influenza sull'esecuzione dello stesso. I commenti permettono la migliore lettura del programma. Per inserire un blocco di commenti premere questo pulsante sulla barra strumenti e trascinare il mouse per selezionare l'area rettangolare in cui collocare il commento. Inserire poi il testo del commento nell'apposita finestra di dialogo. Quando si inserisce il testo di un blocco commento non sono necessari particolari caratteri di inizio o fine quali «(*» e «*)». Un blocco commento può essere ridimensionato spostando gli angoli del bordo quando è evidenziato.

Spostamento di oggetti

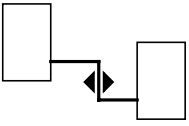
Per spostare oggetti all'interno del diagramma, selezionarli e trascinare il mouse per muovere l'area selezionata nel diagramma. Per spostare oggetti connessi, basta semplicemente spostare il loro simbolo all'interno del diagramma. L'editor ISaGRAF LD/FBD provvederà a tracciare automaticamente le connessioni tra gli oggetti spostati, basandosi sulla loro nuova collocazione.

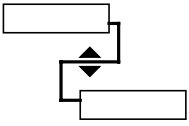
Tracciare connessioni

Per tracciare una connessione tra elementi esistenti premere uno di questi tasti sulla barra strumenti. Se si traccia un collegamento tra un punto connessione ed uno spazio vuoto nel diagramma, verrà automaticamente terminato con un «angolo definibile» in modo che sia possibile continuare a tracciare un altro segmento.

Modificare il tracciamento dei collegamenti

Il comando «**Strumenti / Sposta linea**» serve a modificare il tracciamento automatico di un collegamento precedentemente evidenziato sul diagramma. Questo comando non ha alcun effetto se la connessione è diretta ad un angolo definibile. Nel caso la connessione sia composta da tre segmenti, il comando modifica la posizione del secondo segmento. Sotto sono riportati degli esempi:





Modifica del tipo di collegamento

E' possibile cambiare facilmente il tipo di collegamento (con o senza la negazione booleana) facendo doppio click con il mouse sulla sua estremità destra.

Tracciare rami LD

Per tracciare un nuovo ramo LD, innanzitutto inserire la barra di alimentazione sinistra. Poi inserire una bobina: verrà automaticamente collegata all'alimentazione. Altri contatti e connessioni verticali OR possono essere inseriti direttamente sulla linea del ramo senza tracciare nuove linee di collegamento.
Quando si inserisce un nuovo contatto LD o una nuova bobina in uno spazio vuoto dell'area di lavoro, un nuovo ramo orizzontale viene tracciato automaticamente a partire dal nuovo elemento inserito fino alle esistenti barre di alimentazione sulla destra e sulla sinistra. La linea non viene tracciata automaticamente se il nuovo elemento non è collocato tra le barre di alimentazione. Il nuovo elemento inserito può poi essere liberamente spostato sul ramo tracciato. Le linee orizzontali tracciate automaticamente dall'editor all'inserimento di un nuovo simbolo di contatto LD o bobina, possono essere evidenziate e cancellate. Si possono inserire nuovi contatti LD o bobine sulla linea orizzontale di un ramo esistente. L'editor taglia automaticamente i rami e li connette ai punti di connessione destro e sinistro del nuovo contatto o bobina inseriti.

Connessioni multiple

Si possono creare connessioni multiple a destra di ogni punto di uscita. In questo modo l'informazione viene trasmessa a diversi altri punti del diagramma. Lo stesso stato, viene propagato ad ogni estremità a destra. Non esiste limite al numero di linee che possono essere tracciate alla destra di un punto di connessione.
Due linee di connessione non possono avere le estremità destre connesse allo stesso punto di **input**, ad eccezione dei seguenti simboli LD:

 ..... barra di alimentazione destra

 ..... connessioni multiple a destra dell'operatore (OR)

Questi simboli LD possono avere un numero illimitato di input.

A.7.3 Lavorare su un diagramma esistente

I comandi del menù «**Modifica»** permettono di cambiare o completare un diagramma esistente. La maggior parte di questi comandi agisce sugli elementi evidenziati nel diagramma.

Apportare correzioni ad un diagramma

Il tasto CANC serve ad eliminare gli elementi evidenziati. Eventuali collegamenti interrotti vengono cancellati con l'eliminazione degli elementi. Per ripristinare gli elementi cancellati con il comando CANC, si utilizza il comando «**Modifica / Annulla».** Il comando CANC si può applicare anche a gruppi di elementi evidenziati nel diagramma. Le voci «**Taglia**», «**Copia**», «**Incolla**» del menù «**Modifica**» servono a spostare o copiare gli elementi evidenziati.

Trova e sostituisci

Le voci «**Modifica / Trova**» e «**Modifica / Sostituisci**» vengono usate per trovare e sostituire testi all'interno del diagramma. La ricerca avviene solo su nomi completi. La ricerca viene fatta su contatti, bobine, nomi di blocchi, variabili e etichette. Non è, invece, possibile trovare una stringa all'interno del testo di un commento. Il comando Sostituisci non può essere usato per cambiare il nome di un blocco.

Partendo dalla posizione corrente, la ricerca può essere fatta sia verso il basso sia verso l'alto. Raggiunti i limiti del diagramma la ricerca «cicla».

Visualizzazione dell'ordine di esecuzione

Se il diagramma FBD comprende cicli all'indietro l'ordine di esecuzione non può seguire il metodo unico da destra a sinistra / dall'alto al basso. Per evitare confusione si usa il comando «**Strumenti / Visualizza ordine di esecuzione**» o si premono i tasti **CTRL + F1** per visualizzare l'ordine di esecuzione che verrà usato per la compilazione. Etichette numerate da 1 ad N vengono visualizzate vicino ai simboli che determinano un'azione (bobine, variabili di tipo «Set» e blocchi funzione).

Inserire simboli e testo

Per associare un simbolo o un testo ad un elemento, fare un doppio click con il mouse sull'elemento voluto. Questo vale per variabili, contatti e bobine, testi di commento e etichette. Nel caso di contatti o bobine, la procedura permette anche di cambiarne il tipo (diretto, inverso…).

Per immettere il testo normalmente sono previste delle finestre di dialogo che includono la lista delle variabili o la lista dei blocchi. Se nel menù «**Opzioni**» è selezionato «**Immissione manuale da tastiera**», i simboli di variabili e i nomi di blocchi sono introdotti direttamente in una apposita **riga di testo**. All'interno di tale riga si scrive il nuovo testo e lo si valida premendo il tasto INVIO, oppure si annulla l'operazione premendo il tasto ESC. La riga di testo usata nell'ingresso manuale da tastiera non può essere chiusa con il mouse.

Se nel menù «**Opzioni»** è indicato il modo «**Immissione automatica**», viene automaticamente richiesto il nome della variabile inserendo un nuovo contatto o una nuova bobina. Il nome viene sempre richiesto automaticamente nel caso di inserimento di una variabile o di una etichetta.

Selezionare il tipo di blocco funzione

Per modificare il tipo di blocco, fare doppio click con il mouse sul simbolo di blocco. La tipologia del blocco viene scelta da una lista di operatori, funzioni e blocchi funzione disponibili. Questo comando consente inoltre di cambiare il numero di punti di input nel caso di operatori commutativi (es. AND, OR, ADD, MUL,…).

Ottenere spazio libero

Quando viene premuto il tasto destro del mouse nell'area di disegno FBD, viene mostrato un menù, che contiene i seguenti comandi utilizzabili per inserire o rimuovere spazio libero nella posizione del cursore del mouse:

Inserisci righe: Questo comando inserisce dello spazio libero orizzontale, composto da 4 righe in accordo con il passo della griglia, iniziando dalla posizione del cursore del mouse all'atto di chiamata del menù.

Cancella righe: Questo comando rimuove lo spazio orizzontale non utilizzato (righe) iniziando dalla posizione del cursore del mouse all'atto di chiamata del menù. Questo comando non può essere utilizzato per rimuovere elementi FBD.

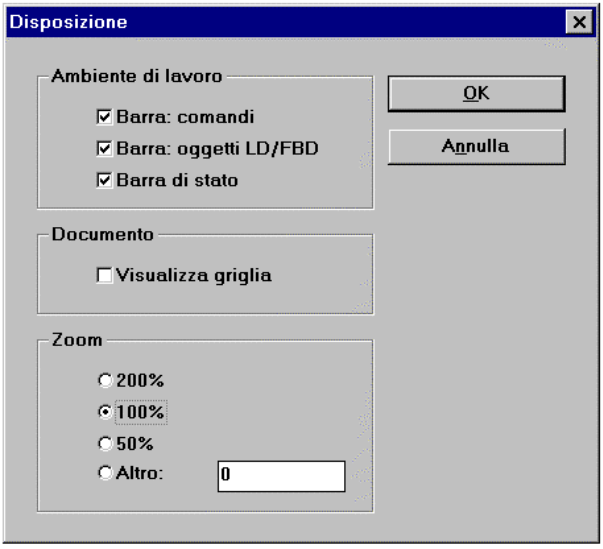
Quando il menù viene aperto, una linea grigia nell'area di disegno FBD indica dove lo spazio vuoto verrà inserito o rimosso.

A.7.4 Opzioni di visualizzazione

Le voci del menù «**Opzioni**» permettono di personalizzare il disegno sullo schermo del diagramma FBD.

Personalizzazione della disposizione (layout)

Il comando «**Opzioni / Disposizione**» apre una finestra di dialogo nella quale sono raggruppati tutti i parametri e le opzioni riguardanti l'area di lavoro dell'editor ed il disegno del diagramma grafico.



Si utilizzano le opzioni del gruppo «**Ambiente di lavoro**» per visualizzare o nascondere la barra strumenti e la barra di stato dell'editor. Le opzioni del gruppo «**Documento**» permettono di mostrare o nascondere i punti della griglia di modifica e di attivare/disattivare l'uso dei colori per il disegno.

Le opzioni del gruppo «**Zoom**» permettono di scegliere il fattore di ingrandimento. È anche possibile usare il pulsante **«Zoom»** della barra strumenti dell'editor per indicare uno dei fattori di ingrandimento predefiniti.

Il comando «**Opzioni / Carattere**» permette di selezionare il tipo di font di caratteri da utilizzare in tutti i documenti grafici di ISaGRAF. Quando si seleziona il font, lo stile del font e la dimensione non sono rilevanti e non necessitano di essere specificati. L'editor grafico di ISaGRAF calcola sempre la dimensione del font in accordo con il rapporto di zoom selezionato.

A.7.5 Stili e monitoraggio delle modifiche

L'editor ISaGRAF LD/FBD permette di assegnare uno stile grafico ad ogni componente di un diagramma LD/FBD. Per stile si intende principalmente una colorazione speciale del diagramma. Ma ISaGRAF utilizza gli stili anche per permettere di monitorare le modifiche nei diagrammi, allo scopo di controllare il numero di versione.

Notare che gli stili non sono visibili durante la simulazione o il debug On-Line, poiché in tale modalità i colori (rosso e blu) sono utilizzati per evidenziare gli stati VERO / FALSO delle variabili spiate.

Stili predefiniti

I seguenti stili sono pre-definiti:

Stile Normale: predefinito per il disegno (nero). Per il monitoraggio delle modifiche, lo stile Normale indica che gli elementi aventi questo stile sono parte del diagramma originale. Durante la fase di esecuzione viene fatta una scansione degli elementi con stile Normale.

Stile Modificato: gli elementi marcati con stile Modificato sono colorati in rosa. Per il monitoraggio delle modifiche, lo stile Modificato viene utilizzato per evidenziare elementi che sono stati aggiunti o cambiati dopo la versione originale del diagramma. Durante la fase di esecuzione viene fatta una scansione degli elementi con stile Modificato.

Stile Cancellato: Gli elementi marcati con stile Cancellato sono colorati in grigio, con linea tratteggiata. Tali elementi non vengono presi in considerazione durante l'esecuzione del diagramma. Questo stile viene usato per tenere traccia degli elementi cancellati dalla versione originale quando è richiesto il controllo della versione.

Stile Personalizzato: oltre agli stili predefiniti, l'editor ISaGRAF LD/FBD permette di selezionare il colore da applicare ad ogni parte del diagramma. Tali elementi vengono considerati come «personalizzati». L'utilizzo dello stile Personalizzato non ha effetto sull'esecuzione del diagramma.

Utilizzare i comandi del sottomenù «**Stile**» nel menù «**Modifica**» per applicare uno stile agli elementi selezionati.

Monitoraggio delle modifiche

L'utilizzo degli stili, e la disponibilità dello stile Cancellato permette il monitoraggio automatico delle modifiche in un diagramma esistente. Selezionando l'opzione «**Marca modifiche**» nel menù «**Modifica / Stile**» per abilitare o disabilitare il monitoraggio delle modifiche.

Quando l'opzione «**Marca modifiche**» è selezionata, tutti gli elementi cambiati o addizionati al diagramma sono automaticamente impostati con lo stile Modificato.

Quando un elemento viene cancellato, usando i comandi **«Cancella»** o **«Taglia»**, non viene rimosso visualmente dal diagramma, ma semplicemente viene marcato con lo stile Cancellato. Questo permette all'utente di tenere traccia automaticamente di tutte le modifiche immesse nel diagramma.

Usare il comando «**Modifica / Stile/ Rimuovi tutti gli elementi cancellati**» per rimuovere tutti gli elementi marcati con stile Cancellato dal diagramma LD/FBD. Questo comando non agisce solo sulla selezione corrente, ma viene applicato sempre all'intero diagramma.

Per «recuperare» un elemento marcato con stile Cancellato, si seleziona l'elemento desiderato e vi si applica lo stile Normale, Modificato o uno stile Personalizzato. Tale operazione può portare a delle connessioni non valide (più di un collegamento connesso allo stesso punto di ingresso) che verranno rilevate durante la prossima verifica del programma.

A.8 Usare l'editor testuale

Questo capitolo descrive le caratteristiche ed i comandi specifici dell'editor testuale ISaGRAF, in particolare per l'inserimento del codice sorgente di programmi ST ed IL.
Gli altri comandi, comuni anche agli altri editor, vengono descritti nel capitolo «Ulteriori informazioni sugli editor dei programmi», in questo stesso documento.

A.8.1 Comandi di editing

I comandi del menù «**Modifica**» permettono di lavorare sul testo in modifica. La maggior parte di questi comandi, agisce sui caratteri evidenziati o che si trovano alla posizione corrente del cursore.

Taglia e incolla

Il tasto CANC permette di cancellare il testo evidenziato. È possibile ripristinare gli elementi cancellati in seguito all'uso del comando CANC, usando la voce «**Modifica / Annulla**». Le voci «**Taglia**», «**Copia**», «**Incolla**» del menù «**Modifica**» permettono di spostare o copiare blocchi di testo all'interno del programma o di inserire parti di testo copiate negli appunti da altre applicazioni.

Trova e sostituisci

Le voci di menù «**Modifica / Trova**» e «**Modifica / Sostituisci**» servono a trovare e sostituire blocchi di testo all'interno del programma. È possibile effettuare la ricerca con qualsiasi stringa, sia verso l'alto che verso il basso, a partire dalla posizione corrente del cursore. Non esiste il pericolo della ripetizione della ricerca al raggiungimento dei limiti del programma.

Raggiungi riga

La voce «**Modifica / Vai alla linea**» serve a muovere il cursore ad un numero di riga specificato. Si rivela molto utile per accedere ad una riga di un programma ST o IL, al cui numero il compilatore ISaGRAF abbia segnalato un errore.

Inserire un simbolo dal dizionario

Si usa la voce «**Modifica / Inserisci variabile**» per inserire, alla posizione del cursore, il simbolo di una variabile o di un oggetto dichiarato nel dizionario del progetto. Il simbolo voluto si seleziona tramite la casella di selezione variabili, descritta nel capitolo «Ulteriori informazioni sugli editor di programmi» in questo stesso documento.

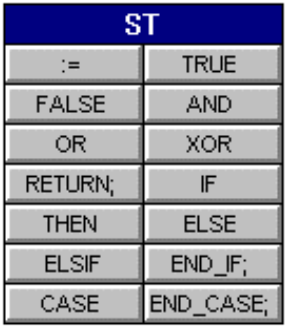
Inserire un file

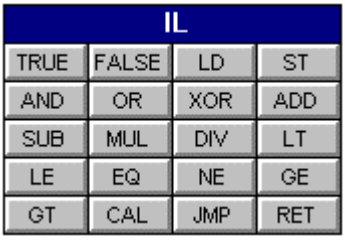
La voce «**Modifica / Inserisci file**» permette di inserire l'intero contenuto di un file alla posizione corrente del cursore. Questo comando è in grado di gestire solamente file tipo testo esclusivamente in formato ASCII.

A.8.2 Opzioni

Le voci del menù «**Opzioni**» servono a rendere visibili o a nascondere le barre strumenti degli editor ed a selezionare il font di caratteri. Il font di caratteri scelto verrà usato per qualsiasi operazione sui testi all'interno dell'ambiente di lavoro ISaGRAF.

Durante l'inserimento del codice sorgente di un programma ST o IL, la voce «**Opzioni / Visualizza parole chiave**» permette di rendere visibile o nascondere un riquadro strumenti che raggruppa le parole chiave più comuni del linguaggio ST o IL.





Fare click su un pulsante del riquadro strumenti per inserire la corrispondente parola chiave o operatore nella posizione corrente del cursore.

A.9 Ulteriori informazioni sugli editor dei programmi

Questo capitolo contiene informazioni utili relative alle caratteristiche comuni a tutti gli editor di programmi ISaGRAF. Questo si riferisce principalmente ai collegamenti con gli altri strumenti ISaGRAF ed alle finestre di dialogo ISaGRAF comuni.

A.9.1 Richiamare gli altri strumenti ISaGRAF

Verificare (compilare) un programma

La voce «**File / Verifica**» della finestra dell'editor avvia il generatore di codice per verificare la sintassi di programmazione del programma in modifica. Nel caso di linguaggio SFC, vengono controllati sia il livello 1 che il livello 2. Al completamento della verifica sintattica, per continuare a lavorare sul programma, è necessario chiudere la finestra Generatore di codice che è stata automaticamente aperta. Nel caso ci sia un solo programma nell'applicazione (quello in fase di modifica) e non vengono evidenziati errori di sintassi, verrà generato il codice dell'applicazione. La voce «**Opzioni / Opzioni compilatore**» permette di impostare i parametri di compilazione ed ottimizzazione. Consultare il capitolo «Usare il generatore di codice» in questo stesso documento, per ulteriori informazioni riguardo la compilazione e la generazione di codice.

Effettuare la simulazione o il debug dell'applicazione

Le voci «**File / Simulazione**» e «**File / Debug**» avviano il debugger grafico ISaGRAF, rispettivamente in modo simulato e in modo realmente connesso, e provvedono a riaprire il programma SFC su cui si sta lavorando in modo debug. Non è possibile inserire modifiche in un programma aperto in modalità debug.

Modificare il dizionario delle variabili

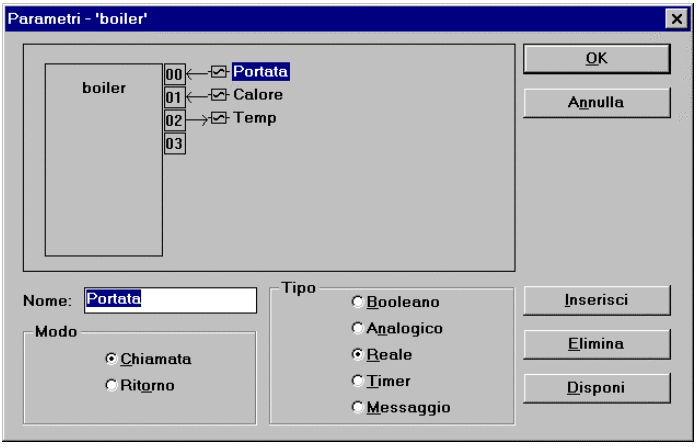
La voce «**File / Dizionario**» serve a modificare il dizionario delle variabili per l'applicazione ed il programma correnti. Contiene anche i punti d'ingresso per la modifica dei nomi definiti. Tutte le dichiarazioni del dizionario con campo di validità **locale** sono relative al programma aperto con l'editor.

A.9.2 Parametri del programma

Se il programma in modifica è una funzione, un blocco funzione o un sottoprogramma, la voce «**File / Parametri**» permette di definire i parametri di chiamata e ritorno. Questo comando non ha effetto se il programma in oggetto è di tipo SFC o di primo livello della sezione **Iniziale** o **Finale**.

Sotto programmi, funzioni o blocchi funzione possono avere al massimo 32 parametri (di ingresso o uscita). Una funzione o un sottoprogramma hanno sempre uno (ed un solo) parametro di ritorno, che deve avere lo stesso nome della

funzione, per renderlo conforme al linguaggio ST. Un blocco funzione può avere più di un parametro di ritorno.
La seguente finestra di dialogo serve a descrivere i parametri di un sottoprogramma:



La lista nella parte superiore destra della finestra mostra i parametri nell'ordine del modello di chiamata, prima i parametri di chiamata e per ultimi i parametri di ritorno. La parte inferiore della finestra mostra una descrizione dettagliata del parametro evidenziato nella lista. Qualsiasi tipo di dato ISaGRAF può essere usato come parametro. I parametri di ritorno devono essere collocati nella lista dopo quelli di chiamata. Nell'attribuire un nome ai parametri bisogna rispettare le seguenti regole:

- La lunghezza del nome non può eccedere i 16 caratteri
- Il primo carattere deve essere una lettera
- I caratteri successivi devono essere lettere, cifre o il carattere «_».
- Non viene fatta distinzione tra maiuscole e minuscole

Il comando «**Inserisci**» serve ad inserire un nuovo parametro prima del parametro evidenziato. Il comando «**Elimina**» si usa per cancellare il parametro evidenziato. Il comando «**Disponi**» riordina automaticamente i parametri, in modo che i parametri di ritorno vengano collocati in fondo alla lista.

A.9.3 Altri comandi del menù «File»

I seguenti comandi sono disponibili nel menù «**File**» di tutti gli editor di programmi:

Aprire un programma diverso

Il comando «**File / Apri**» permette di chiudere il programma correntemente usato per iniziare la modifica con lo stesso linguaggio di un nuovo programma dello stesso progetto. Questa funzione non può esser usata per aprire un programma scritto con linguaggio diverso. Il nuovo programma selezionato sostituisce il corrente nella finestra di modifica.

Stampa del programma

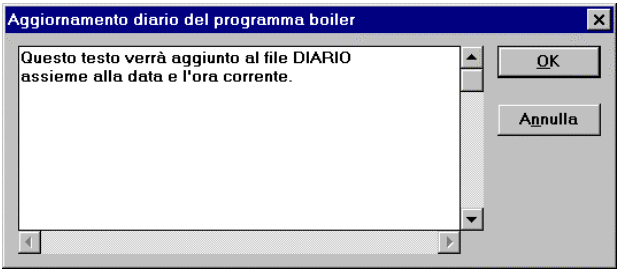
Il comando «**File / Stampa**» apre la finestra Generatore documentazione, che permette di organizzare e stampare un documento che contiene informazioni sull'intero progetto a cui appartiene il programma aperto.

Per alcuni editor grafici (SFC, FBD, Quick LD) è possibile usare il comando «**Modifica / Copia diagramma negli appunti (metafile)**» per copiare negli appunti il tracciato del diagramma in formato metafile, in modo da poterlo inserire in altre applicazioni quali word processor. Per quanto riguarda i programmi SFC, nel metafile copiato appaiono solo le informazioni di primo livello (diagramma, numerazione commenti di primo livello).

A.9.4 Aggiornamento del diario del programma

Si può accedere manualmente al file diario associato al programma in modifica usando il comando «**File / Diario**». Il file diario viene aggiornato automaticamente ad ogni compilazione con l'aggiunta dei messaggi di verifica della sintassi. Gli output della compilazione vengono completati dalla data ed ora di compilazione.

Se è selezionata la modalità «**Aggiorna diario**» nel menù «**Opzioni**» degli editor di programmi, al momento del salvataggio su disco del programma, appare la seguente finestra di dialogo.



Premendo il tasto OK il testo nota inserito verrà salvato alla fine del file diario con l'indicazione della data ed ora correnti. Ciò è particolarmente utile per la manutenzione dell'intero programma in quanto fornisce informazioni importanti circa il ciclo di vita del programma stesso.

A.9.5 Selezionare una variabile dal dizionario

Nell'utilizzo di un programma testuale (ST o IL) la funzione «**Modifica / Inserisci variabile**» permette di selezionare una variabile dichiarata per inserirla alla posizione corrente del cursore. Per lavorare con programmi LD o FBD è richiesta la selezione delle variabili per descrivere bobine, contatti, parametri di blocchi I/O, variabili FBD. In entrambi i casi per selezionare una variabile dichiarata si apre la seguente finestra di dialogo:

La casella di selezione «**Validità**» è utilizzata per scegliere il **Campo di validità:** globale (l'intero progetto) o locale (in tal caso si deve scegliere il nome del programma). La casella di selezione di destra consente di scegliere il **tipo** di variabile. Piccole icone poste accanto al tipo di dato nella casella di selezione sono pulsanti scorciatoia per selezionare più rapidamente i tipi di dato usati più frequentemente:

 ..... Booleano
 ..... Intero / Reale
 ..... Timer
 ..... Messaggio

Per selezionare una variabile basta fare click sul suo nome nella lista. Nome e relativo commento vengono visualizzati in cima alla lista. Premere poi «**OK»** per confermare la selezione. E' inoltre possibile inserire direttamente una variabile nella casella di modifica senza utilizzare la lista (in tal caso se la variabile è nuova non viene aggiunta al dizionario).

A.9.6 Comandi del menù «Strumenti»

I seguenti comandi sono disponibili nel menù **Strumenti**. Sono utilizzati per mostrare informazioni in una *finestra di testo* nella parte inferiore della finestra principale LD:

Visualizza risultati compilazione Mostra nella finestra di testo i messaggi di errore dell'ultima compilazione (solo per programmi ST/IL)

Trova nel testo (diagramma)	Trova tutte le occorrenze di un dato testo nell'intero testo (diagramma) del programma aperto. Produce una lista delle occorrenze nell'apposita finestra di testo aperta in basso.
Nascondi risultati	Chiude la finestra di testo.

Quando i messaggi di errore o le occorrenze sono mostrate nella finestra di testo, facendo doppio click su una linea viene evidenziata nel testo (diagramma) del programma la corrispondente localizzazione.

A.10 Usare l'editor del dizionario

Il dizionario di ISaGRAF è uno strumento di editing per dichiarare le variabili interne, le variabili di I/O, le istanze di blocchi funzione, e i nomi definiti dell'applicazione. Il dizionario raggruppa insieme come stringhe costanti: le variabili dichiarate, le istanze di blocchi funzione dell'applicazione e i nomi definiti.

Variabili, blocchi funzione e nomi definiti devono essere dichiarati nel dizionario prima di poter essere utilizzati nei programmi sorgente. Le variabili e i nomi definiti possono essere usati in un qualunque linguaggio di programmazione: SFC, FC FBD, LD, ST e IL. I blocchi funzione usati nel linguaggio FBD non devono essere dichiarati, perché gli editor ISaGRAF FBD e Quick LD dichiarano automaticamente le istanze dei blocchi utilizzati.

Variabili

Le variabili vengono ordinate secondo il loro **tipo** ed il loro campo di **validità** (ambiente di programmazione entro cui sono utilizzabili). Solo variabili dello stesso tipo e con lo stesso campo di validità possono essere inserite nella stessa scheda della finestra dizionario. Questi sono i campi di validità delle variabili:

GLOBALE.... può essere utilizzata da ogni programma del progetto corrente

LOCALE ..... può essere usata solo da uno specifico programma

Questi sono i tipi di base delle variabili:

BOOLEANO. valori binari «TRUE» / «FALSE» (vero/falso)

ANALOGICO valori reali o interi

TIMER..... valori temporali

MESSAGGIO stringhe di caratteri

Una variabile è identificata da un nome, un commento, un attributo, un indirizzo di rete e da altri campi specifici.
Questi sono i possibili **attributi** per una variabile:

INTERNA ..... variabile in memoria
INGRESSO ..... variabile legata ad un dispositivo di input
USCITA..... variabile legata ad un dispositivo di output
COSTANTE..... variabile interna di sola lettura (con valore iniziale)

Nota: Una variabile di tipo **timer** ha sempre come attributo quello di variabile **interna**.
Le variabili con attributo di **ingresso** o di **uscita** hanno sempre un campo di validità **Globale**.

Nomi definiti

Un **nome definito** è uno pseudonimo che può essere usato in ognuno dei linguaggi di programmazione per sostituire una stringa di testo. Il testo sostituito può essere il nome di una variabile, un'espressione costante o un'espressione complessa. I nomi

definiti sono ordinati in base al loro campo di validità. Solo nomi definiti dello stesso tipo e con lo stesso campo di validità possono appartenere alla stessa scheda della finestra dizionario. Questi sono i possibili campi di validità per un nome definito:

COMUNE ..... può essere utilizzato da ogni programma di qualsiasi progetto

GLOBALE.... può essere usato da qualunque programma del presente progetto

LOCALE ..... può essere usato solo da uno specifico programma

Un nome definito è identificato con una stringa di testo (nome assegnato), un ben definito blocco di testo ST equivalente e da un commento libero.
Si noti che i nomi definiti, a differenza delle variabili e delle istanze di blocchi funzione, possono avere campo di validità **Comune** (ed in tal caso possono essere utilizzati ovunque).

Istanze di blocchi funzione

Un blocco funzione contiene dati interni «non visibili», e quindi ogni copia (istanza) di un blocco funzione deve essere sempre identificata univocamente.
La modalità di utilizzo di un blocco funzione, dipende dal tipo di linguaggio di programmazione che lo richiama.
Nei linguaggi ST e IL le istanze di blocchi funzione (**istanze BF**) devono essere dichiarate nel dizionario. Nel seguente esempio viene mostrato come utilizzare il blocco funzione «R_TRIG», definito nella Libreria, per il riconoscimento del fronte di salita di variabili diverse. Tale blocco funzione è così definito:

Nome del blocco:	**R_TRIG**
***Parametri*:**	**Input=CLK**
	Output=Q

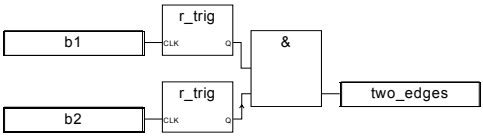
Per dichiarare le istanze si usa la scheda **Istanze BF** della finestra dizionario:

***Nome istanza*:**	**TRIG_B1**	**Nome blocco: R_TRIG**
***Nome istanza*:**	**TRIG_B2**	**Nome blocco: R_TRIG**

Le istanze dichiarate possono essere utilizzate nei programmi ST nel seguente modo:

```
TRIG_B1 (b1);
edge_b1 := TRIG_B1.Q;     (* b1 variabile con riconoscimento di fronte *)
TRIG_B2 (b2);
edge_b2 := TRIG_B2.Q;     (* b2 variabile con riconoscimento di fronte *)
```

Le istanze di blocchi funzione dichiarate possono avere validità **GLOBALE** (note a qualunque programma del progetto) o **LOCALE** (note ad uno specifico programma)
Nei linguaggi di programmazione FBD ed LD i blocchi funzione non devono essere dichiarati, poiché l'editor FBD di ISaGRAF dichiara automaticamente le istanze dei blocchi utilizzati.



(* i blocchi funzione hanno sempre il nome del blocco definito nella libreria. Gli editor FBD e Quick LD di ISaGRAF dichiarano automaticamente un'istanza ogni volta che un blocco viene disegnato nel diagramma*)

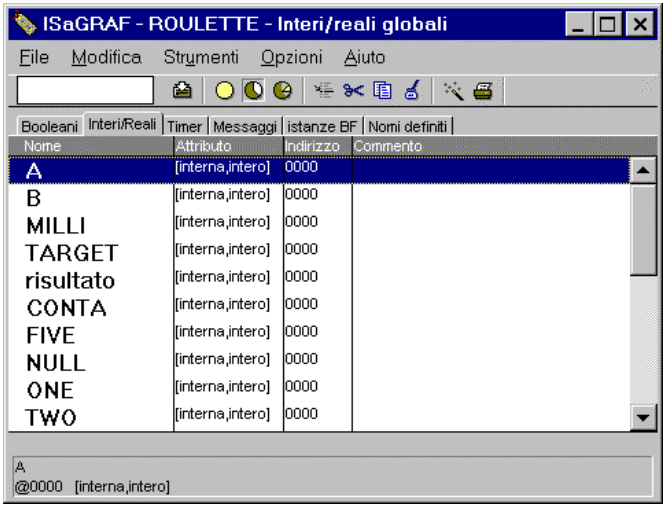
Le istanze di blocchi funzione dichiarate automaticamente dagli editor FBD e Quick LD hanno sempre campo di validità **LOCALE** nel programma in uso.

Indirizzi di rete

Gli indirizzi di rete sono **facoltativi.** Una variabile con un indirizzo di rete non nullo può essere **spiata** da un sistema esterno (ad esempio un sistema di visualizzazione di processi) in fase di esecuzione. In generale l'indirizzo di rete fornisce un meccanismo di identificazione per ogni sistema di comunicazione che in fase di esecuzione non possa lavorare con nomi simbolici. Un indirizzo di rete può essere inserito per ogni variabile, all'atto della sua descrizione completa, quando la variabile viene creata o modificata.

A.10.1 La finestra principale del dizionario

La finestra del dizionario mostra una lista di variabili di uno stesso tipo e con lo stesso campo di validità. Il **tipo** e la **validità** delle variabili che si stanno modificando, vengono sempre visualizzati nella barra strumenti. Di seguito viene mostrata finestra di editing del dizionario:



La finestra mostra solo i principali campi di descrizione della variabile: nome, attributo, indirizzo di rete, e testo di commento. La descrizione completa della variabile evidenziata viene sempre mostrata in basso, nella barra di stato.

Per selezionare il campo di validità di una variabile da modificare si usano i seguenti pulsanti nella barra degli strumenti

COMUNE.....può essere usato da qualsiasi programma di qualunque progetto

GLOBALE.....può essere usato da qualunque programma del progetto corrente

LOCALE .....può essere utilizzato da un unico programma

Per visualizzare le variabili di un dato tipo, selezionare la scheda corrispondente:

Per individuare una variabile in base alle prime lettere del suo nome è sufficiente scrivere nel campo di testo sulla destra della barra strumenti. In questo modo la ricerca è effettuata sull'intera lista, dall'inizio. Nella finestra Dizionario è disponibile anche il comando «**Modifica / Trova**» per effettuare la ricerca di stringhe di testo inserite in nomi di variabili o in commenti ed evidenziare nella lista la variabile trovata. La ricerca tratta allo stesso modo le lettere maiuscole e minuscole.

A.10.2 Gestione delle variabili

I comandi disponibili nel menù «**File**» della finestra del dizionario agiscono su tutta la classe di variabili, istanze di blocchi funzione e nomi definiti selezionati. Per selezionare il tipo e il campo di validità degli oggetti da modificare, si utilizza il comando «**File / Altro»**.
Il comando «**Strumenti / Connessioni di I/O**» consente all'utente di avviare l'editor delle connessioni di I/O ISaGRAF relativamente al progetto aperto. Se la variabile di I/O selezionata nella finestra del dizionario è già connessa, la corrispondente scheda di I/O verrà visualizzata all'apertura della finestra **Connessioni di I/O**.
Il comando «**Strumenti / Tabelle di conversione**» avvia l'editor delle tabelle di conversione, che possono essere associate a variabili di I/O intere o reali del dizionario.
Il comando «**Strumenti / Riferimenti incrociati**» avvia l'editor dei riferimenti incrociati per il progetto aperto.

Stampa di variabili

Per stampare la lista corrente delle variabili, o dei nomi definiti su una stampante standard Windows™, si usa il comando «**File / Stampa**», con cui viene avviato il **Generatore documentazione**. La stampa comprenderà la descrizione completa di ogni singola variabile o nome definito del tipo correntemente selezionato.

Creazione di nuove variabili

Il comando «**Modifica / Nuovo**» consente di creare nuove variabili, istanze di blocchi funzione o nomi definiti. Il tipo e il campo di validità sono gli stessi di quelli correntemente selezionati. Le nuove variabili vengono inserite subito prima di quella indicata dalla barra di selezione. Una volta dato il comando, si aprirà una casella di input per inserire la descrizione della variabile. Completata la descrizione si preme il tasto «**Memorizza**» per inserire la variabile nella lista. A questo punto la casella di input si aprirà nuovamente permettendo di inserire altre variabili con lo stesso comando. Premendo il tasto «**Annulla**» si interrompe la procedura di inserimento variabili.

Modifica di variabili esistenti

Una variabile indicata dalla barra di selezione può essere modificata mediante il comando «**Modifica**» del menù «**Modifica**». Una volta dato il comando si aprirà una casella di input per modificare la descrizione della variabile. Completata la descrizione, premendo il tasto «**Memorizza**», si renderà operativa la modifica. Premendo i tasti «**Prossima**» e «**Precedente**» la modifica viene estesa alle variabili adiacenti (successiva o precedente). Premendo il tasto «**Annulla**» viene chiusa la finestra di dialogo senza salvare alcuna modifica.

Taglia ed incolla

La finestra dizionario di ISaGRAF consente di selezionare contemporaneamente più variabili della lista (selezione multipla). Sono poi disponibili comandi per lavorare sulla lista di variabili evidenziate. Di seguito sono riportati i comandi disponibili del menù «**Modifica**»:

COPIA..... Copia il gruppo di variabili selezionato negli appunti

TAGLIA..... Copia il gruppo di variabili selezionato e lo toglie dalla lista sulla quale si sta lavorando

INCOLLA..... Inserisce il contenuto degli appunti prima della variabile selezionata

Le funzioni Copia/Taglia/Incolla possono essere usate tra liste di variabili. Non possono essere usate tra liste di oggetti di tipo diverso.

Ordinamento delle variabili

Il comando «**Strumenti / Ordina**» ordina le variabili o i nomi definiti della lista corrente. L'ordine è dato dagli attributi delle variabili:

- prima le variabili con attributo interna
- poi le variabili con attributo ingresso
- infine le variabili con attributo uscita

Le variabili con le stesse caratteristiche sono inserite in ordine alfabetico. I nomi definiti sono sempre ordinati alfabeticamente.

Assegnazione degli indirizzi di rete

Gli indirizzi di rete sono **facoltativi**. Una variabile con un indirizzo di rete non nullo può essere **spiata** da un sistema esterno (ad esempio un sistema di visualizzazione di processi) in fase di esecuzione. Un indirizzo di rete può essere inserito per ogni variabile, durante la sua descrizione completa, quando la variabile viene creata o modificata.

Si possono attribuire indirizzi di rete ad un intero gruppo di variabili con il comando «**Strumenti / Rinumera indirizzi**». Il comando agisce sull'intero gruppo di variabili selezionate nella lista. Inserendo un valore esadecimale per l'**indirizzo base** (indirizzo della prima variabile del gruppo), alle variabili del gruppo verranno assegnati **indirizzi consecutivi**. Inserendo un indirizzo base nullo tutti gli indirizzi di rete delle variabili selezionate verranno impostati a zero.

Importare stringhe booleane «vero/falso»

Lavorando su un nome definito, il comando «**Strumenti / Importa definizioni vero/falso**» consente di definire automaticamente come parole chiave del linguaggio, stringhe associate a variabili booleane per rappresentare gli stati «TRUE» (vero) e «FALSE» (falso). Queste stringhe sono definite normalmente per la rappresentazione in debug. Devono essere specificate come nome definito per essere usate nei programmi. Questo comando effettua la ricerca di stringhe booleane vero/falso nelle dichiarazioni che hanno lo stesso campo di validità del nome definito selezionato.

A.10.3 Descrizione di oggetti

Ad ogni variabile, istanza di blocchi funzione o nome definito bisogna fornire la descrizione completa. I campi di descrizione variano per ogni tipo di oggetto. I seguenti sono campi comuni per tutti i tipi di variabile:

Nome.....Nome della variabile: il primo carattere deve essere una lettera, i seguenti possono essere lettere, cifre o il carattere «_».
Indirizzo rete.....Indirizzo di rete esadecimale (facoltativo). Quando questo campo è non nullo, la variabile può essere spiata da sistemi esterni in fase di esecuzione.
Commento.....Commento libero per descrivere la variabile.
Ritentiva.....Questa opzione indica una variabile ritentiva.

Questi sono altri campi descrittivi di variabili **booleane.**

Attributi.....Indica una variabile interna, costante, di ingresso o di uscita
Valori - «Falso».....Stringa usata per valori falsi in fase di debug.
Valori - «Vero».....Stringa usata per valori veri in fase di debug.
Imposta a vero all'inizio Se è selezionata questa opzione il valore iniziale è VERO, altrimenti è FALSO.

Questi sono altri campi descrittivi di variabili analogiche **intere o reali**

Attributi.....Definisce una variabile interna, costante, di ingresso o di uscita
Formato.....Specifica una variabile intera o reale (floating). Si può selezionare il formato di visualizzazione da usare durante il debug.
Unità.....Stringa utilizzata per identificare l'unità fisica in fase debug.

Conversione .....Nome della tabella o funzione di conversione connessa alla variabile (solo per variabili input o output)
Valore iniziale .....Valore iniziale della variabile (deve avere lo stesso formato della variabile). Se non specificato il valore iniziale è zero.

Di seguito sono riportati altri campi per la definizione di variabili di tipo **timer**.

Attributi .....Specifica una variabile interna o costante.
Valore iniziale .....Valore (tempo) iniziale della variabile. Se non specificato sarà posto pari a: time#0s.

Campi di descrizione di una variabile **messaggio**.

Attributi .....Specifica una variabile interna, costante, di ingresso o di uscita.
Lunghezza massima .....definisce il numero massimo di caratteri che possono essere inseriti nel messaggio.
Valore iniziale .....Valore iniziale della variabile (la cui lunghezza non può eccedere la capacità del messaggio). Se non specificato il valore iniziale sarà la stringa nulla (vuota).

Campi descrittivi di un **Nome definito**.

Nome .....Nome usato nel file sorgente ST: il primo carattere deve essere una lettera, i successivi possono essere lettere, cifre o il carattere «_».
Definizione .....Stringa conforme alla sintassi ST che sostituisce il Nome definito durante la compilazione. Ad esempio: Nome = PI - Equivale = 3.14159
Commento .....Commento libero per l'equivalenza definita.

Campi descrittivi di una **istanza di blocco funzione**

Nome .....Nome dell'istanza utilizzato nei file sorgente ST: il primo carattere deve essere una lettera, i successivi possono essere lettere, cifre o il carattere «_».
Tipo .....Nome del corrispondente blocco funzione all'interno della libreria.
Commento .....Commento libero dell'istanza di blocco funzione.

A.10.4 Dichiarazione rapida di un blocco di variabili

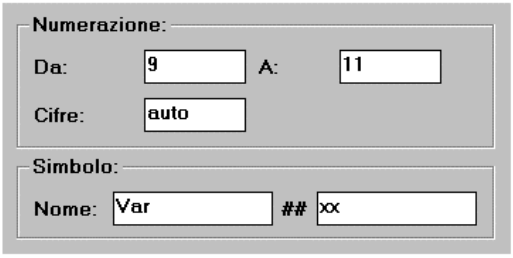
Il comando «**Strumenti / Dichiarazione rapida**» permette di dichiarare un gruppo di variabili simultaneamente. Le variabili create con una dichiarazione rapida sono numerate utilizzando una numerazione convenzionale, per la quale si devono definire:

- l'indice (numero) della prima e dell'ultima variabile,
- il testo da aggiungere prima e dopo il numero nel simbolo della variabile,
- Il numero di cifre utilizzate per esprimere il numero nel simbolo della variabile.

Inoltre, si possono specificare gli attributi base delle variabili create (interna, ingresso o uscita ...), più alcune proprietà che dipendono dal tipo di variabile (l'attributo di variabile ritentiva, il formato intero o reale, la massima lunghezza della stinga messaggio).

Bisogna sempre definire un testo da inserire prima del numero della variabile, poiché un simbolo di variabile non può cominciare con una cifra. Quando il numero di cifre è impostato su «Auto», ISaGRAF formatta il campo numerico di ogni singola variabile creata con il numero minimo di cifre necessarie. Quando il numero di cifre viene specificato, ISaGRAF formatta tutti i campi numerici alla lunghezza specificata aggiungendo dei caratteri '0'. Impostare un numero fisso di cifre per il campo numerico delle variabili può essere molto utile per prevenire un errato ordinamento alfanumerico. Seguono alcuni esempi.

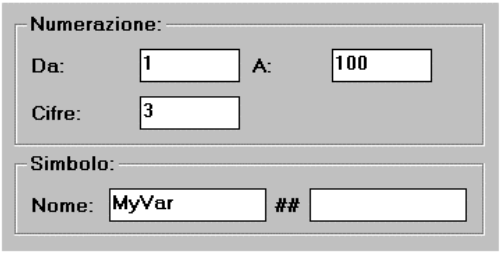
Esempio 1: Queste impostazioni in una dichiarazione rapida:



creano le seguenti 3 variabili:

Var9xx Var10xx Var11xx

Esempio 2: Queste impostazioni in una dichiarazione rapida:



creano 100 variabili con i nomi da MyVar001 a MyVar100

A.10.5 Mappa indirizzi Modbus per SCADA

Gli indirizzi di rete di ISaGRAF sono spesso utilizzati per stabilire un collegamento, tra un sistema ISaGRAF e uno SCADA, basato su comunicazione Modbus. In questo caso, lo SCADA è un Modbus master e il target ISaGRAF agisce come un Modbus slave. Gli indirizzi di rete sono usati per creare una mappa Modbus virtuale per tutte le variabili ISaGRAF che devono essere controllate dallo SCADA. Il comando «**Strumenti / Mappa indirizzi SCADA per Modbus»** permette di creare velocemente una mappa virtuale Modbus con le variabili dell'applicazione.

La finestra Mappa indirizzi Modbus per SCADA mostra due liste. La lista superiore è un segmento (4096 locazioni) della mappa Modbus, che mostra le variabili mappate (quelle a cui è associato un indirizzo di rete). La lista inferiore mostra le variabili non mappate (senza indirizzo di rete). L'indirizzo «0» non può essere usato per mappare una variabile.

Per muovere variabili da una lista all'altra e costruire così la mappa, usare i comandi «**Mappa**» e «**Rimuovi**» dal menù «**Modifica**». Oppure facendo doppio click sul simbolo di una variabile in una lista per spostarla nell'altra lista. In ogni momento, si può usare il menù a tendina «Segmento» per vedere un altro segmento della mappa.

I comandi del menù «**Opzioni**» permettono di visualizzare gli indirizzi in formato decimale o esadecimale.

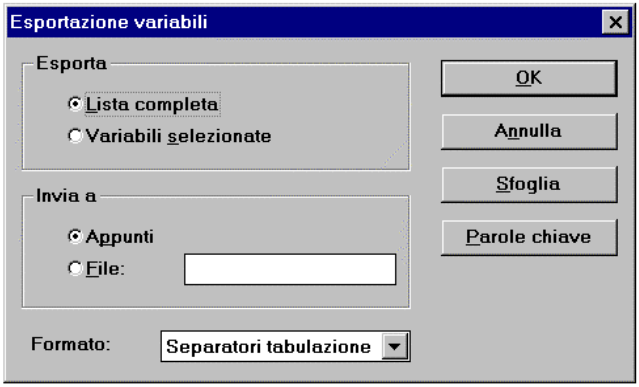
Il comando «Modifica / Trova**» permette di cercare una variabile dichiarata, sia che sia stata mappata o no.**

A.10.6 Scambio di informazioni con altre applicazioni

Il dizionario ISaGRAF offre funzioni importazione/esportazione che permettono di scambiare informazioni con altri applicativi quali word processor, fogli elettronici, data base … . Questi comandi si trovano nel menù «**Strumenti**». Il comando «**Esporta testo**» fornisce una descrizione formato ASCII dei campi che descrivono un insieme di oggetti selezionati ed archivia questo testo negli appunti di Windows oppure in un file. Queste informazioni sono poi di solito usate da altre applicazioni. Il comando «**Importa testo**» importa i campi descrittivi di variabili dichiarate, descritti in formato ASCII ed archiviati negli appunti di Windows oppure in un file e aggiorna la lista in uso con i campi importati. Si tratta generalmente di informazioni prodotte da altri applicativi.

Esportazione di dati

La seguente finestra di dialogo appare selezionando il comando «**Esporta testo**». Essa consente di controllare la procedura di esportazione dati.



Indicando la scelta «**Lista completa**» si esporta l'intera lista editata. In questo caso la selezione corrente viene ignorata. Scegliendo invece «**Variabili selezionate**» verranno esportate solo le variabili evidenziate.

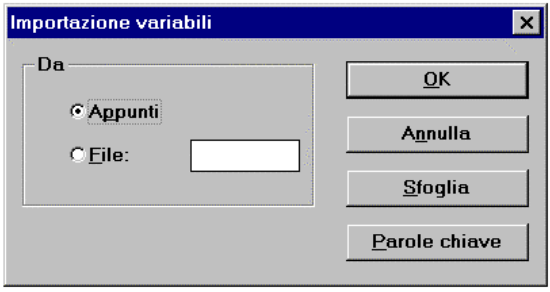
Scegliendo l'opzione «**Appunti**» le informazioni da esportare, tradotte in formato ASCII, verranno salvate negli appunti di Windows. Il testo è quindi disponibile per l'operazione «incolla» in altri applicativi. Se si sceglie invece l'opzione «**File**» il testo verrà archiviato in un file ASCII. Bisogna inserire il percorso completo del file. Il comando «**Sfoglia**» permette di cercare tra le cartelle esistenti.

Si sceglie poi il formato del file da esportare. I formati disponibili verranno descritti in seguito. Premendo poi «**OK**» si da il via all'operazione di export. Premendo «**Annulla**» si chiude la finestra di dialogo annullando la procedura di export.

Tutti i campi degli oggetti selezionati vengono salvati nel testo esportato seguendo l'ordine standard di dichiarazione. La prima riga di testo esportato contiene il nome dei campi. Ogni oggetto viene descritto in una riga di testo. Il separatore di fine riga è la sequenza standard MS-DOS «**0d-0a**». Premendo il tasto «**Parole chiavi**» si può modificare il nome usato per identificare i campi nella prima riga del testo esportato. Questo comando verrà descritto in seguito.

Importazione di dati

Il comando «**Importa testo**» apre la seguente finestra di dialogo. Essa consente di controllare la procedura di import.



Scegliendo l'opzione «**Appunti**» l'informazione importata viene presa dagli appunti di Windows in formato ASCII. Scegliendo invece l'opzione «**File**» il testo viene preso da un file ASCII. In questo caso è necessario inserire il percorso completo del file. Per cercare un percorso esistente selezionare il tasto «**Sfoglia**».

La funzione di importazione riconosce automaticamente il formato (separatori) usato nel testo importato. I formati disponibili verranno descritti in seguito. Con il tasto «**OK**» viene avviata l'importazione. Premendo «**Annulla**» si chiude la finestra di dialogo annullando la procedura di importazione. «. Premendo il tasto «**Parole chiavi**» si può modificare il nome usato per identificare i campi nella prima riga del testo esportato. Questo comando verrà descritto in seguito.

La prima riga di testo deve contenere il nome dei campi secondo l'ordine usato nelle righe successive. Ogni oggetto deve essere descritto su una riga di testo. Il separatore di fine riga è la sequenza standard MS-DOS «**0d-0a**». I campi possono apparire in qualsiasi ordine. Eventuali campi mancanti verranno sostituiti da valori di default per gli oggetti importati. Se nella lista editata esiste già uno degli oggetti importati, verrà chiesta conferma prima della sostituzione. La descrizione dell'oggetto viene poi aggiornata con i campi importati. Eventuali campi mancanti non verranno aggiornati.

Formati testo disponibili

In seguito viene riportata la lista dei formati disponibili per i comandi di esportazione. I comandi importazione riconoscono automaticamente questi formati.

□ Tabulatori

Descrizione: ***I campi sono separati dal carattere tabulazione***

Esempio:

Nome	Attributo	Commento
livello	interno	Livello interno acqua calcolato
allarm1	uscita	allarme uscita principale

□ virgole

Descrizione: ***I campi vengono separati da virgole.***

Esempio: Nome,Attributo,Commento

livello,interno,Livello interno acqua calcolato
allarm1,uscita,allarme uscita principale

□ separatori punto e virgola

Descrizione: ***I campi vengono separati con il punto e virgola***

Esempio: Nome;Attributo;Commento
livello;interno;Livello interno acqua calcolato
allarm1;uscita;allarme uscita principale

□ virgole ed apici

Descrizione: ***I campi sono separati da virgole.***
Ogni campo è scritto tra apici.

Esempio: «Nome», «Attributo», «Commento»
«livello», «interno», «Livello interno acqua calcolato»
«allarm1», «uscita», «allarme uscita principale »

Parole chiave

I nomi utilizzati per definire i campi nella prima riga importata o esportata possono essere modificati attraverso il pulsante «**Parole chiavi**». Questo comando apre la seguente finestra di dialogo

La finestra mostra la lista dei campi e le parole chiave ad essi associate. Per modificare una parola chiave selezionare un campo dell'elenco e premere il tasto «**Modifica**». Selezionando «**Predefinito**» si ripristina la lista originale delle parole chiave. Per definire una parola chiave bisogna rispettare le seguenti regole:

- il nome non può eccedere i **16** caratteri
- il primo carattere deve essere una **lettera**
- i caratteri successivi possono essere **lettere**, **cifre** o il carattere «_»
- non è possibile utilizzare lo stesso nome per differenti parole chiave

Di seguito le parole chiave standard trovate in ISaGRAF:

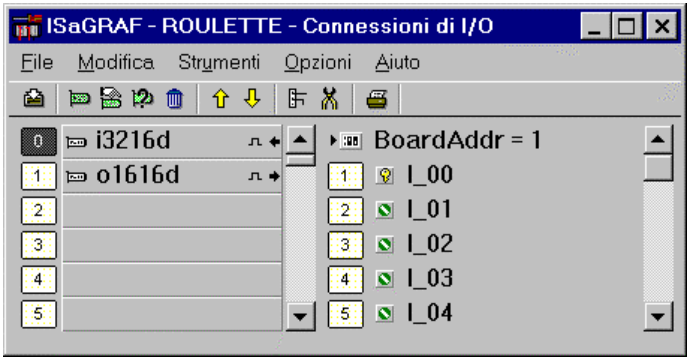
Nome oggetto.....**Name**
Commento testo.....**Comment**

Indirizzo di rete.....**Address**
Attributi (interno, input, output) .....**Attribute**
Stringa booleana 'False' .....**False**
Stringa booleana 'True'.....**True**
Formato analogico (reale o intero).....**Format**
Stringa unità analogica.....**Unit**
Nome conversione analogica .....**Conversion**
Lunghezza massima messaggio .....**MaxLength**
Tipo libreria blocchi funzione .....**Library**
Equivalente Nome definito.....**Equivalence**
Attributo interno.....**Internal**
Attributo Input.....**Input**
Attributo Output .....**Output**
Attributo Costante .....**Constant**
Formato reale analogico.....**Reale**
Formato intero analogico.....**Integer**

A.11 Usare l'editor delle connessioni di I/O

Scopo delle operazioni di connessione I/O è stabilire un collegamento tra le variabili I/O dell'applicazione ed i canali fisici delle schede collocate sulla macchina target. Per effettuare il collegamento è necessario identificare ed impostare tutte le schede della macchina target ed assegnare le variabili di I/O ai corrispondenti canali di I/O.

Questa è la finestra **Connessioni di I/O**



L'elenco di sinistra mostra lo schema a cestello della macchina destinazione, con gli **alloggiamenti (slot) per le schede** . Un alloggiamento può essere libero oppure utilizzato da una scheda di I/O o da un'apparecchiatura complessa. Ogni alloggiamento è identificato da un **numero d'ordine**. Nello schema possono essere contenute fino a **255** schede. L'elenco di destra mostra i parametri della scheda selezionata e le variabili connesse ai canali. Una scheda può avere fino a **128** canali I/O.

Icone

Le icone visualizzate a fianco dei simboli di scheda indicano il tipo e gli attributi delle variabili che possono essere connesse ai canali della scheda. Il sistema ISaGRAF non consente di connettere alla stessa scheda variabili di tipo diverso. Di seguito viene riportato il significato delle icone utilizzate:

- ..... tipo booleano
- ..... tipo intero/reale (possono essere connessi entrambi i tipi di variabile)
- ..... tipo messaggio
- ..... ingresso – nessun canale connesso
- ..... uscita - nessun canale connesso
- ..... ingresso – almeno un canale connesso
- ..... uscita – almeno un canale connesso

Queste sono le icone utilizzate per mostrare il tipo di dispositivo I/O installato in un alloggiamento (slot):
 ..... apparecchiatura I/O complessa
 ..... scheda reale I/O
 ..... scheda virtuale I/O

Icone utilizzate per rappresentare un parametro o un canale:
 ..... parametri della scheda
 ..... canale libero
 ..... canale connesso

Posizionare le schede nella lista

Si possono utilizzare questi pulsanti della barra strumenti, oppure i comandi del menù «**Modifica / Sposta scheda in su/giù**», per spostare di una posizione verso l'alto o verso il basso una scheda I/O nella lista principale. Il comando «**Modifica / Inserisci slot**» inserisce nella posizione corrente un alloggiamento (slot) vuoto.

A.11.1 Definizione delle schede I/O

I comandi per definire la scheda selezionata (impostarne i parametri) e per connettere le variabili di I/O ai canali sono contenute nel menù «**Modifica**».

Selezione del tipo di scheda I/O

E' necessario inserire l'identificazione della scheda prima di potervi connettere le variabili I/O. L'ambiente di sviluppo ISaGRAF mette a disposizione una libreria di schede predefinite. Questa libreria può essere stata costruita dai fornitori dei dispositivi di I/O. Per impostare l'identificazione di una scheda si usa il comando «**Modifica / Imposta scheda/dispositivo**». Questo comando si può usare per selezionare dalla libreria ISaGRAF sia una singola scheda, sia un'apparecchiatura complessa di I/O. Con un doppio click su un alloggiamento è possibile impostare la scheda o l'apparecchiatura corrispondente.

Tutti i canali di una singola scheda hanno lo stesso tipo (booleano, intero/reale o messaggio) e lo stesso verso (input o output). Non viene fatta distinzione tra variabili reali ed intere per la connessione di I/O. Un'apparecchiatura complessa di I/O rappresenta un dispositivo di I/O con canali di diverso tipo o verso. Essa è rappresentata come una lista di singole schede di I/O ed usa un solo alloggiamento della lista.

Rimozione di una scheda

Per rimuovere la scheda o l'apparecchiatura I/O evidenziata si utilizza il comando «**Modifica / Libera slot**». Se le variabili sono già connesse ai corrispondenti canali verranno automaticamente disconnesse.

Schede reali e schede virtuali

Il comando «**Modifica / Scheda reale/virtuale**» serve ad indicare il tipo di scheda o apparecchiatura complessa di I/O evidenziata. Le seguenti icone nella lista permettono di riconoscere il tipo di scheda:

 ..... scheda I/O reale
 ..... scheda I/O virtuale

In **modalità reale** le variabili di I/O sono direttamente collegate ai corrispondenti dispositivi di I/O. Le operazioni di ingresso o uscita del progetto sono legate direttamente alle corrispondenti condizioni di ingresso o uscita dei dispositivi I/O. In **modalità virtuale** le variabili di I/O vengono processate esattamente come variabili interne. Esse possono essere lette o aggiornate dal debugger, permettendo di simulare il processo di I/O senza che in realtà venga effettuata alcuna connessione.

Note tecniche

Il comando «**Strumenti / Note tecniche**» visualizza la guida utente on-line relativa alla scheda o all'apparecchiatura complessa selezionata. Le note tecniche relative alla scheda vengono stilate dal fornitore hardware della scheda I/O. In esse sono contenute tutte le informazioni riguardanti la gestione della scheda I/O ed il significato dei parametri.

Rimozione di variabili connesse

Con il comando «**Strumenti / Disconnetti canali scheda**» vengono disconnesse tutte le variabili I/O già collegate alla scheda selezionata.

Commenti

Nell'elenco delle schede viene visualizzato, assieme al nome, anche il testo di commento ad una variabile di I/O dichiarata. Dal momento che ISaGRAF consente anche l'utilizzo di variabili rappresentate direttamente (notazione %) i commenti possono essere associati a canali liberi. Per inserire un commento al canale libero attualmente evidenziato nella lista delle schede, usare il comando «**Strumenti / Commenti canale**». Questo comando non può essere usato per modificare il commento di una variabile di I/O dichiarata nel dizionario del progetto.

A.11.2 Impostazione dei parametri della scheda

Per impostare il valore dei parametri di una scheda fare doppio click sul nome nella lista di destra, oppure, evidenziarla e scegliere il comando «**Imposta canale/parametri**» nel menù «**Modifica**». I parametri vengono elencati all'inizio della lista. La seguente è l'icona che viene usata per rappresentare i parametri nella lista.

 ..... parametri della scheda

Il significato ed il formato di input dei parametri vengono definiti dal fornitore della corrispondente scheda o apparecchiatura di I/O. Utilizzare il comando «**Strumenti / Note tecniche**» oppure consultare il manuale hardware per maggiori informazioni sui parametri della scheda.

Nota: Quando è selezionata una scheda nella lista di sinistra il comando «**Imposta canale/parametri**» apre la finestra di impostazione dei parametri.
Quando è selezionato un canale nella lista di destra il comando «**Imposta canale/parametri**» apre la finestra di impostazione del canale.

A.11.3 Connessione dei canali I/O

Una volta definita la scheda si devono connettere le variabili di I/O ai suoi canali.

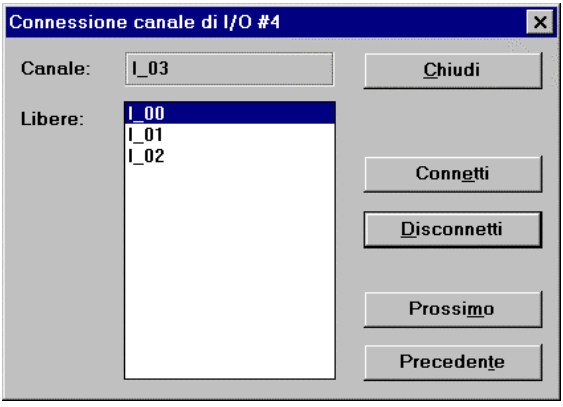
Connessione delle variabili

Per impostare la connessione di un canale fare doppio click sulla sua locazione nella lista di destra. È anche possibile evidenziare il canale e scegliere il comando «**Modifica / Imposta canale/parametri**». le seguenti icone nella lista rappresentano i canali:

..... canale libero

..... canale connesso

Si apre la seguente finestra di dialogo:



La lista contiene tutte le variabili compatibili con il tipo ed il verso della scheda evidenziata. L'elenco comprende solo variabili non ancora connesse. Premendo il pulsante «**Connetti**» la variabile evidenziata nell'elenco verrà connessa al canale scelto. Il tasto «**Disconnetti**» rimuove (disconnette) la variabile dal canale selezionato. Per passare ad un altro canale si usano i pulsanti «**Prossimo**» e «**Precedente**». La posizione del canale selezionato viene sempre mostrata nel titolo della finestra di dialogo.

Rimozione di variabili connesse

Con il comando «**Strumenti / Disconnetti canali scheda**» vengono disconnesse tutte le variabili I/O già collegate alla scheda selezionata.

Commenti

Nell'elenco delle schede viene visualizzato, assieme al nome, anche il testo di commento ad una variabile di I/O dichiarata. Dal momento che ISaGRAF consente anche l'utilizzo di variabili rappresentate direttamente (notazione %) i commenti possono essere associati a canali liberi. Per inserire un commento al canale libero attualmente evidenziato nella lista delle schede, usare il comando «**Strumenti /**

Commenti canale». Questo comando non può essere usato per modificare il commento di una variabile I/O dichiarata nel dizionario del progetto.

A.11.4 Variabili rappresentate direttamente

I canali liberi sono quelli non collegati ad una variabile di I/O dichiarata. ISaGRAF permette l'utilizzo nel codice sorgente dei programmi di **variabili rappresentate direttamente** ad indicare canali liberi. L'identificatore di una variabile rappresentata direttamente inizia sempre con il carattere **«%».**

Di seguito vengono riportati i nomi convenzionali di una variabile rappresentata direttamente per un canale di una singola scheda. «**s**» rappresenta il numero di slot (alloggiamento) della scheda e «**c**» il numero del canale.

%IX*s.c*..... canale libero di una scheda di input di tipo booleano
%ID*s.c*..... canale libero di una scheda di input di tipo intero
%IS*s.c*..... canale libero di una scheda di input di tipo messaggio
%QX*s.c*..... canale libero di una scheda di output di tipo booleano
%QD*s.c*..... canale libero di una scheda di output di tipo intero
%QS*s.c*..... canale libero di una scheda di output di tipo messaggio

In seguito vengono elencate le convenzioni per attribuire un nome a variabili rappresentate direttamente per il canale di un'apparecchiatura complessa. «**s**» è il numero di slot (alloggiamento) dell'apparecchiatura, «**b**» l'indice della singola scheda all'interno dell'apparecchiatura e «**c**» il numero del canale.

%IX*s.b.c*..... canale libero di una scheda di input di tipo booleano
%ID*s.b.c*..... canale libero di una scheda di input di tipo intero
%IS*s.b.c*..... canale libero di una scheda di input di tipo messaggio
%QX*s.b.c*..... canale libero di una scheda di output di tipo booleano
%QD*s.b.c*..... canale libero di una scheda di output di tipo intero
%QS*s.b.c*..... canale libero di una scheda di output di tipo messaggio

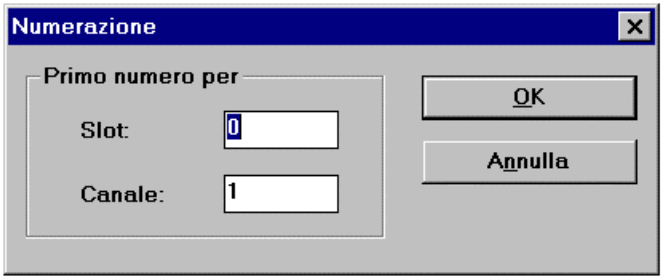
Ecco alcuni esempi:

`%QX1.6` sesto canale della scheda #1 (output booleano)
`%ID2.1.7` settimo canale della scheda #1 dell'apparecchiatura #2 (input intero)

Una variabile direttamente rappresentata non può essere di tipo **reale**.

A.11.5 Numerazione

Per impostare le convenzioni di numerazione usare il comando «**Opzioni / Numerazione**». Nella seguente finestra di dialogo è possibile specificare il numero usato per il primo alloggiamento (slot) ed il numero utilizzato per il primo canale di ogni scheda:



Per default, la numerazione degli slot inizia con l'indice **0**, la numerazione dei canali inizia con l'indice **1**.

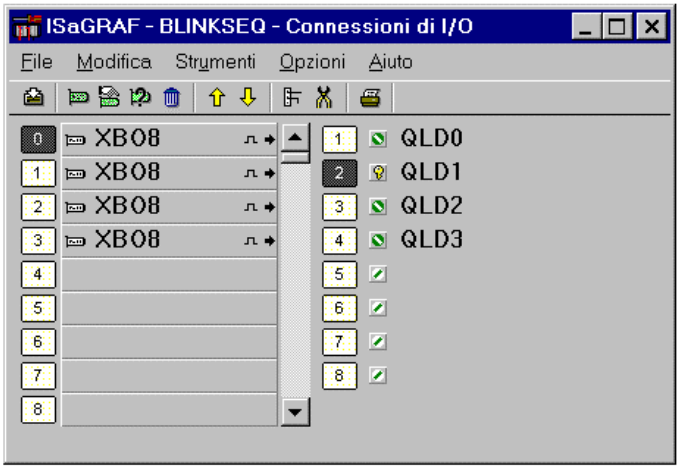
Attenzione: bisogna essere molto cauti nel cambiare le convenzioni di numerazione, poiché l'effetto si ripercuote sui simboli utilizzati per variabili rappresentate direttamente e possono sorgere errori di compilazione se le variabili I/O rappresentate direttamente sono utilizzate in programmi esistenti.

A.11.6 Impostazione di protezioni individuali

L'ambiente di lavoro ISaGRAF fornisce un completo sistema di protezione dati basato su password gerarchizzate. Le connessioni di I/O possono essere protette globalmente da una password, inoltre ISaGRAF permette di impostare *protezioni individuali* per ogni calale di I/O. Si assume che:

- le password sono precedentemente definite nel sistema di password (usare il comando «**Progetto / Imposta password**» dalla finestra Gestione progetti) di modo che siano disponibili livelli di protezione per ogni *protezione individuale*
- si utilizzano livelli di protezione con priorità maggiore per le *protezioni individuali* rispetto all protezione globale di I/O.

Quando un canale I/O ha protezione individuale, una piccola icona viene mostrata vicino al suo nome nella finestra delle connessioni I/O.



Utilizzare i comandi «**Imposta protezione canale**» e «**Rimuovi protezione canale**» del menù «**Modifica**» per impostare o rimuovere la protezione individuale per il canale selezionato. Entrambi i comandi chiedono di immettere una password valida di modo da poter associare un livello di protezione al canale. Quindi , ogni volta che si desidera cambiare la connessione ad un canale avente protezione individuale bisogna immettere una password con livello di priorità sufficiente.

Attenzione: se un canale è protetto con un certo livello, e la password associata viene rimossa dal sistema di protezione, e non sono definite password di livello superiore, la connessione al canale non potrà più essere cambiata, a meno che non sia definita una nuova password con livello sufficiente.

A.12 Creazione di tabelle di conversione

L'ambiente di lavoro ISaGRAF consente di creare tabelle di conversione. Queste sono costituite da un insieme di punti utilizzati per definire una conversione analogica. Una tavola di conversione può essere associata ad una variabile analogica di ingresso o di uscita. La tavola crea una relazione proporzionale tra valori elettrici (letti su sensori di ingresso o inviati a dispositivi uscita) e valori fisici (usati nella programmazione dell'applicazione). Per visualizzare le tabelle di conversione utilizzare il comando «**Strumenti / Tabelle di conversione»** dalla finestra del dizionario.

Una volta definita, una tabella di conversione può essere usata per filtrare i valori di qualsiasi variabile analogica di ingresso o di uscita del progetto selezionato. Per associare una tabella di conversione ad una variabile, deve essere aperto l'editor di dichiarazione delle variabili. Deve essere evidenziata una variabile analogica di ingresso o uscita e deve essere aperta la finestra di modifica dei parametri. Una variabile non può essere associata ad una tabella di conversione che non sia già stata definita.

A.12.1 Comandi del menù

La finestra **Tabelle di conversione** mostra la lista delle tabelle di conversione già definite e contiene una lista di comandi per controllare l'intero formato delle tabelle e per la loro definizione come insieme di punti.

Creazione di una nuova tabella

Con il pulsante «**Nuovo**» è possibile creare una nuova tabella di conversione. Per ogni progetto possono essere create fino a **127** tabelle di conversione. Nel codice eseguibile dell'applicazione vengono inserite solo le tabelle utilizzate (quelle associate a variabili analogiche). Per assegnare un nome ad una tabella è necessario seguire le seguenti regole:

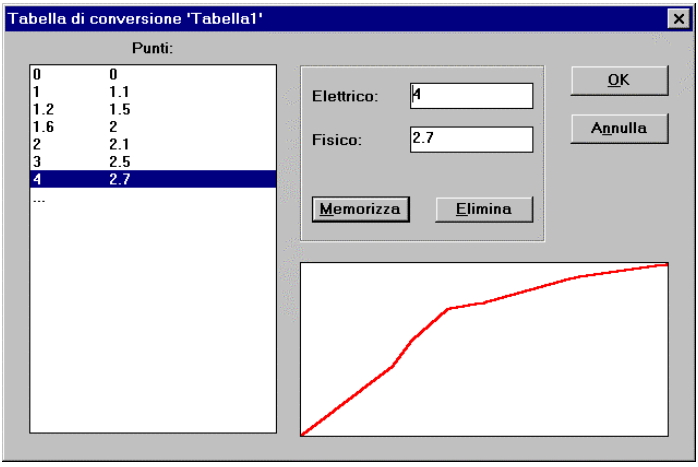
- il nome non può eccedere i **16** caratteri
- il primo carattere deve essere una **lettera**
- i caratteri successivi possono essere **lettere**, **cifre** o il carattere «_»
- il nome della tabella non fa differenza tra lettere maiuscole e minuscole.

Modifica del contenuto di una tabella

Per inserire i punti di una tabella selezionata dalla lista si usa il pulsante «**Modifica**». Per accedere alla finestra di dialogo **Modifica** è anche possibile fare doppio click sulla tabella o premere il tasto **INVIO**. La finestra di dialogo **Modifica** viene richiamata automaticamente alla creazione di una nuova tabella. Per ogni tabella è necessario inserire almeno due punti.

A.12.2 Inserimento di punti in una tabella

La finestra di dialogo **Modifica** permette di inserire i punti di una tabella di conversione. La finestra mostra a sinistra l'elenco dei punti già definiti. La finestra in basso a destra mostra l'andamento grafico della tavola:



I punti vengono inseriti usando i comandi della finestra. Per la definizione dei punti è necessario rispettare le regole numeriche descritte alla fine di questo capitolo. La casella di sinistra contiene sempre l'elenco dei punti già definiti per la tabella selezionata. La colonna di sinistra mostra i valori elettrici (esterni) dei punti. La colonna di destra mostra i valori fisici (interni). Per modificare il valore di un punto o per eliminarlo dalla lista è necessario selezionarlo. L'ultima riga dell'elenco («**... ...**») viene usata per definire un nuovo punto. La casella in basso a destra visualizza la rappresentazione grafica della tabella corrente. Non sono visibili né assi, né coordinate dato che si tratta di una rappresentazione proporzionale della curva. E' utile per controllare velocemente la corretta definizione della curva.

Definizione di un nuovo punto

Per definire un nuovo punto, evidenziare l'ultima riga («**... ...**») dell'elenco dei punti. La stessa procedura è prevista per iniziare la definizione di una nuova tabella. Inserire prima il valore elettrico (esterno) e quindi il valore fisico (interno) di ogni punto. I valori vengono memorizzati come numeri in virgola mobile a precisione singola. E' importante ricordare che è necessario inserire almeno **due punti** per definire una curva. Una volta inseriti i due valori, il comando «**Memorizza**» li aggiunge alla tabella. Per ogni tabella di conversione possono essere definiti al massimo **32** punti.

Modifica di un punto

Per modificare i valori di un punto esistente, innanzitutto evidenziarlo nella lista. Possono essere inseriti i nuovi valori: elettrico (esterno) e fisico (interno) del punto. I

valori vengono archiviati come numeri in virgola mobile a precisione singola. Una volta inseriti entrambi i valori si aggiorna la tavola con il pulsante «**Memorizza**».

Cancellazione di un punto

Un punto esistente viene eliminato evidenziandolo sull'elenco e premendo il tasto «**Elimina**». E' importante ricordare che il numero di punti definiti nella tabella non deve mai essere inferiore a **due**.

A.12.3 Regole e limiti

Nel definire una tabella di conversione bisogna rispettare le seguenti regole. La tabella può essere usata per convertire variabili analogiche sia di input che di output:

- Due punti non possono essere definiti con lo stesso valore elettrico
- La curva deve essere o crescente o decrescente
- Due punti non possono essere definiti con lo stesso valore fisico.

Nel definire una tavola di conversione per un progetto è necessario rispettare anche i seguenti limiti:

- Per ogni singolo progetto non si possono definire più di **127** tabelle di conversione
- Per ogni singola tabella non possono essere definiti più di **32** punti.

A.13 Usare il generatore di codice

La finestra **Generatore di codice** viene aperta automaticamente con i comandi «**Verifica**» e «**Compila applicazione**» delle finestre di lavoro di ISaGRAF.
La finestra di generazione del codice non si chiude in modo automatico al termine dell'operazione, questo per permettere di avere ancora accesso a tutti i comandi ed alle opzioni previste per il compilatore e accessibili dai menù.

A.13.1 Comandi principali

Il menù **«File»** contiene i comandi per la verifica della sintassi del programma e la generazione del codice.

Generazione del codice dell'applicazione

Il comando «**File / Compila**» genera l'intero codice del progetto. Prima della generazione, il programma controlla la sintassi delle dichiarazioni e dei programmi. Ogni errore che non può essere rilevato durante la compilazione del singolo programma viene scoperto durante la generazione del codice. Questo vale per tavole di conversione, le connessioni di variabili I/O e i collegamenti con le librerie. Il Generazione di codice blocca la compilazione del programma quando vengono rilevati degli errori. Per poter procedere con la generazione del codice è necessario correggere il programma errato. I programmi che sono già stati controllati (ed in cui non si sono trovati errori) e che non sono stati modificati dopo la l'ultima operazione «**Verifica**» non vengono ricompilati. Vengono sempre eseguite la verifica delle dichiarazioni delle variabili ed il controllo di coerenza dell'applicazione. Durante il controllo del programma è possibile interrompere la procedura di compilazione premendo il tasto **ESC.**

Nota: Se la dichiarazione di una variabile con campo di validità LOCALE di un programma è stata modificata, solo tale programma verrà verificato. Se invece è stata modificata una variabile GLOBALE, verranno verificati tutti i programmi.

Controllo della sintassi del programma

Il comando «**File / Verifica programma**» consente di verificare un singolo programma. Il programma selezionato viene compilato anche se non è avvenuta alcuna modifica dopo l'ultima verifica. Il comando «**File / Verifica dizionario**» consente di verificare le dichiarazioni di tutte le variabili del progetto.

Il comando «**File / Verifica tutti i programmi**» controlla la sintassi di tutti i programmi del progetto anche se alcuni di essi non sono stati modificati. Questa procedura **non** si interrompe quando viene incontrato un errore in un programma. Produce un elenco completo di tutti gli errori presenti nei programmi del progetto. E' possibile interrompere la verifica premendo **ESC**.

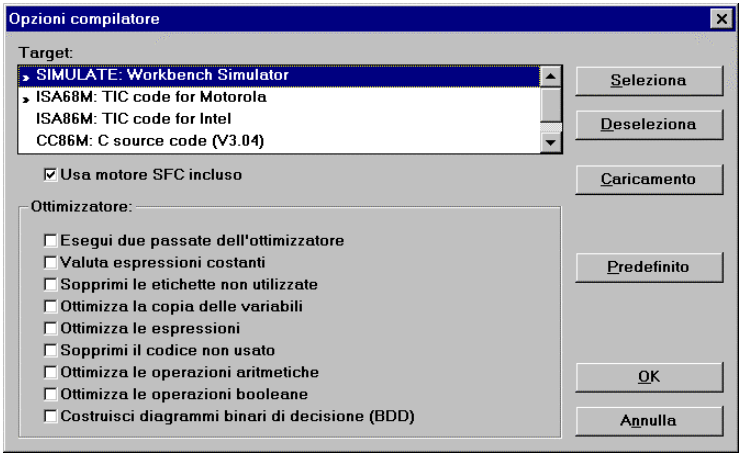
Ricompilazione completa

Il comando «**File / Tocca**» simula la modifica di tutti i programmi del progetto, in modo da forzarne la verifica alla successiva operazione «**Compila**». Il comando

«**File / Apri**» apre l'ultimo programma su cui è stata effettuata la verifica. Questo comando si rivela molto utile per accedere direttamente ad un programma in cui siano stati rilevati degli errori di sintassi. Un doppio click sul messaggio di errore apre il programma e posiziona il cursore alla riga dell'errore.

A.13.2 Opzioni del compilatore

Il comando «**Opzioni / Opzioni compilatore**» permette di impostare i parametri principali usati dal Generatore di codice ISaGRAF per produrre ed ottimizzare il codice target. La finestra Opzioni compilatore permette di selezionare il tipo di codice da generare, a seconda dei target ISaGRAF, e di impostare i parametri di ottimizzazione per il tempo di compilazione ed i requisiti di esecuzione dell'applicazione.



Selezione del target

In alto viene visualizzato l'elenco dei tipi di codice target che possono essere prodotti. Il segno «**>>**» indica i target selezionati. Il generatore di codice ISaGRAF è in grado di produrre fino a **3** diversi codici nella stessa sessione di compilazione. I pulsanti «**Seleziona**» e «**Deseleziona**» permettono di selezionare i codici target in relazione all'hardware target. Quelli che seguono sono i target standard ISaGRAF:

SIMULATE:..... Questo tipo di codice si riferisce al Simulatore ISaGRAF dell'ambiente di debug. Non è possibile usare il simulatore (comando **«Debug / Simulazione»** dalla finestra Programmi) se il codice dell'applicazione non è stato prodotto selezionando anche il target SIMULATE.

ISA86M: ..... Questo codice **TIC** (Codice Target Indipendente) è per i kernel ISaGRAF installati su processori Intel. Il tipo di processore influisce sull'ordine di disposizione dei byte nel codice generato.

ISA68M:..... Questo codice **TIC** (Codice Target Indipendente) è per i kernel ISaGRAF installati su processori Motorola. Il tipo di processore influisce sull'ordine di disposizione dei byte nel codice generato.

CC86M:..... Con questo target il compilatore produce codice sorgente «C» che può essere compilato e collegato alle librerie del kernel del target ISaGRAF da incorporare nell'eseguibile

Consultare il manuale dell'hardware per conoscere il tipo di kernel del target ISaGRAF installato sul PLC. Altri tipi di target (codice macchina, codice sorgente «C»…) potranno essere implementati nelle versioni successive dell'ambiente ISaGRAF.

Elaborazione SFC

Selezionando l'opzione «**Usa motore SFC incluso**» nella finestra Opzioni compilatore si abilita l'uso del motore SFC di ISaGRAF. Questa modalità è da preferirsi, in quanto offre le migliori prestazioni in esecuzione. Tuttavia, il motore potrebbe risultare mancante in alcune particolari implementazioni del target ISaGRAF o, più comunemente, su target personalizzati basati su elaborazioni successive del codice ISaGRAF. In questo caso è necessario disabilitare questa opzione per permettere al compilatore ISaGRAF di tradurre i diagrammi SFC in istruzioni di basso livello. Consultare la documentazione hardware per ulteriori informazioni sull'uso di questa opzione.

Opzioni dell'ottimizzatore

Quelli che seguono sono i parametri, impostabili dalla finestra Opzioni compilatore, usati dal Generatore di codice ISaGRAF per ottimizzare il codice target. Il pulsante «**Predefinito**» disabilita tutte le ottimizzazioni per ridurre il tempo di compilazione.

L'opzione «**Esegui due passate dell'ottimizzatore**» fa si che venga eseguito per due volte l'ottimizzatore di codice ISaGRAF. Le ottimizzazioni apportate nel secondo passaggio sono, generalmente, meno significative di quelle del primo passaggio.

Con l'opzione «**Valuta espressioni costanti**» le espressioni costanti vengono valutate dal compilatore. Ad esempio, l'espressione numerica «**2 + 3**» viene sostituita nel codice target da «**5**». Quando questa opzione non è indicata, le espressioni costanti vengono calcolate in fase di esecuzione.

L'opzione «**Sopprimi le etichette non utilizzate**» fa si che l'Ottimizzatore semplifichi la struttura di salti e etichette del programma eliminando le etichette destinazione non usate o i salti nulli.

L'opzione «**Ottimizza la copia delle variabili**» ottimizza l'uso delle variabili temporanee (usate per memorizzare risultati intermedi). Questa opzione viene comunemente usata assieme all'opzione «**Ottimizza le espressioni**». In questo modo l'ottimizzatore è in grado di usare i risultati di espressioni e sottoespressioni usate più volte nel programma.

- Con l'opzione «**Sopprimi il codice non usato**», l'ottimizzatore elimina le parti di codice non significative. Ad esempio, delle seguenti istruzioni: «**var := 1; var := X;**», verrà generato solo il codice corrispondente a: «**var := X;**».

- Con l'opzione «**Ottimizza le operazioni aritmetiche**», l'ottimizzatore semplifica le operazioni aritmetiche nel caso di particolari operandi. Ad esempio, l'espressione «**A + 0**» viene sostituita da «**A**». Con l'opzione «**Ottimizza le operazioni booleane**», l'ottimizzatore semplifica le operazioni booleane nel caso di particolari operandi. Ad esempio, l'espressione booleana «**A & A**» viene sostituita da «**A**».

- Con l'opzione «**Costruisci diagrammi binari di decisione (BDD)**», l'Ottimizzatore sostituisce le formule booleane (che combinano gli operatori **AND**, **OR** , **XOR** e **NOT**) con una lista ridotta di salti condizionati. La traduzione avviene solamente se il tempo di esecuzione previsto per la sequenza di salti risulta inferiore a quello previsto per l'espressione originale.

  La seguente tabella riassume i tempi di compilazione rispetto ad ogni parametro di ottimizzazione:

	guadagno (prestazioni)	tempo di compilazione
Esegui 2 passate	■■■■	(*)
Espressioni costanti	■■■■■■■■■■	■■■■■
Etichette non utilizzate	■■■■■	■■■■■■■■■■
Copia delle variabili	■■■■■	■■■■■■■■■■
Espressioni	■■■■■	■■■■■■■■■■
Codice non usato	■■■■■	■■■■■■■■■■
Operazioni aritmetiche	■■■■■■■■■■	■■■■■
Operazioni booleane	■■■■■■■■■■	■■■■■
Diagrammi binari decisionali	■■■■■■■■■■■■■■■■	■■■■■■■■■■■■■■■■

  (*) il tempo di compilazione può anche raddoppiare.

A.13.3 Produrre codice sorgente «C»

L'ambiente ISaGRAF permette la produzione di codice sorgente in linguaggio «C». In questo caso, tutti i componenti dell'applicazione, inclusi diagrammi SFC, definizioni di data base e sequenze di codice, vengono generati in formato codice sorgente «C». Ci sono due possibilità per la generazione di codice C:

CC86M (C source code - V3.04) produce sorgente codice "C" non strutturato. Questa opzione deve essere selezionata se il software per il target è basato su una versione ISaGRAF precedente alla 3.23.

SCC (structured C source code) produce sorgente codice "C" strutturato. Questa opzione è da preferire se il software per il target è basato su una versione ISaGRAF 3.23 o successiva.

I seguenti due file vengono creati nelle cartella del progetto:

APPLI.C ..... codice sorgente dell'applicazione
APPLI.H ..... definizioni del linguaggio "C"

Nel caso di generazione di codice sorgente "C" strutturato, per ogni programma dell'applicazione vengono creati un file sorgente "**.C**" e un file di definizioni "**.H**", oltre ai file comuni "**APPLI.C**" e "**APPLI.H**".

Questi file devono essere compilati e collegati alle librerie target ISaGRAF per poter produrre il codice eseguibile finale. Consultare la Guida Utente al sistema di sviluppo I/O ISaGRAF per ulteriori informazioni sulle tecniche di implementazione consigliate.

Note: Alcune possibilità offerte in fase di debug, come lo scaricamento dell'applicazione, le modifiche on line e i breakpoint, non sono disponibili quando l'applicazione ISaGRAF è stata compilata in «C».

A.13.4 Visualizzazione delle informazioni

Il menù «**Modifica**» della finestra Generatore di codice comprende i comandi per la visualizzazione, dei diversi file di testo prodotti in fase di generazione del codice o di controllo sintattico. Tutte le informazioni vengono memorizzate su disco così da poter essere esaminate con i comandi del menù

- ***Comandi di editing***

Il comando «**Modifica / Cancella**» serve a ripulire l'area di testo della finestra. La finestra viene ripulita automaticamente prima di ogni operazione di generazione di codice o controllo sintattico. Il comando « **Modifica / Copia**» permette di copiare il testo visualizzato negli appunti di Windows, così da poter essere usato da altre applicazioni come gli editor di testo ISaGRAF.

- ***Visualizzare i messaggi di output del compilatore***

Il comando «**Modifica / Messaggi esecuzione**» visualizza, nell'area di testo della finestra, tutti i messaggi prodotti dall'ultima operazione «**Compila**» o «**Verifica**». Riguarda tutti i messaggi di errore.

Altre voci del menù «**Modifica**» permettono di vedere i file di testo ausiliari prodotti durante la verifica sintattica e la generazione di codice. Questi file non vengono generalmente usati nel caso di un progetto comune ISaGRAF.

A.13.5 Definizione delle risorse

La voce «**Risorse**» del menù «**Opzioni**» permette di definire le risorse. Per risorsa si intende qualsiasi dato prodotto dall'utente (configurazioni di rete, impostazioni hardware…) in qualsiasi formato (file, elenco di valori) che debba essere unito al codice generato per poter essere trasferito al PLC target. Questi dati non vengono elaborati direttamente dal kernel ISaGRAF e sono comunemente rivolti ad altro software installato sul PLC target. Consultare il manuale hardware per ulteriori informazioni sulle risorse disponibili.

Il file di definizione delle risorse

Le risorse vengono definite in un «**File di definizione delle risorse**» memorizzato con gli altri file del progetto ISaGRAF. SI tratta di un file testo ASCII, che viene elaborato dal Compilatore di risorse ISaGRAF. Il compilatore viene avviato automaticamente nel momento in cui viene prodotto il codice dell'applicazione. In questa sezione viene esaminata la sintassi di questo file. Il file di definizione delle risorse fa uso delle regole lessicali del linguaggio ST. I commenti iniziano con i caratteri «**(***» e terminano con «***)**» e possono essere inseriti dovunque nel testo. Le stringhe sono delimitate da singoli apostrofi. Consultare la seconda parte di questo manuale per ulteriori spiegazioni sul formato lessicale in uso per l'inserimento di valori numerici.

Riferimenti al linguaggio

Quello che segue è un elenco di parole chiave ed istruzioni usate in un file di definizione delle risorse.

ULONGDATA

Significato: Indica una risorsa costituita da un elenco di valori interi. I valori vengono memorizzati all'interno del codice del target come interi a 32 bit senza segno. I valori vengono memorizzati nell'ordine specificato dal file di definizione delle risorse. I valori devono essere separati da virgole. Il nome della risorsa non può eccedere i 15 caratteri.

Sintassi:
```
ULONGDATA '<nome_risorsa>'
BEGIN
  ...selezione_target...
  ...elenco dei valori...
END
```

Esempio:
```
ULongData 'MYDATA'
Begin
  ...
  0, -1, 100_000,        (* decimale*)
  16#A0B1, 2#1011_0101   (* esadecimale, binario *)
End
```

VARLIST

Significato: Indica una risorsa costituita da un elenco di indirizzi di variabili. Le variabili nel file di definizione delle risorse vengono identificate dal nome. Gli indirizzi di variabili vengono memorizzati nel codice del target come interi a 16 bit senza segno. Gli indirizzi vengono memorizzati nell'ordine specificato dal file di definizione delle risorse. I valori devono essere separati da virgole. Il nome della risorsa non può eccedere i 15 caratteri.

Sintassi:
```
VARLIST '<nome_risorsa>'
BEGIN
  ...selezione_target...
```

```
  ...elenco di nomi di variabili...
END
```

Esempio:
```
VarList 'LIST'
Begin
  ...
  Var100, MyParameter, Command, Alarm
End
```

BINARYFILE

Significato: Indica una risorsa costituita da un File Binario. I dati sorgente sono memorizzati in un file MS-DOS. La definizione delle risorse del target viene completata dal percorso del target. I caratteri *end of line* non vengono convertiti dal Compilatore di Risorse ISaGRAF. Il nome della risorsa non deve eccedere i **15** caratteri.

Sintassi:
```
BINARYFILE '<nome_risorsa>'
BEGIN
  ...selezione target...
  FROM '<percorso_sorgente>'
  TO '<percorso_destinazione>'
END
```

Esempio:
```
BinaryFile 'MYFILE'
Begin
  ...
  From 'c:\user\config.bin'
  To '/dd/user/appl/config.dat'
End
```

TEXTFILE

Significato: Indica una risorsa costituita da un File Testo. I dati sorgente sono memorizzati in un file ASCII. La definizione delle risorse del target viene completata dal percorso del target. I caratteri *end of line* vengono convertiti dal Compilatore di Risorse ISaGRAF secondo le convenzioni del sistema host. Il nome della risorsa non deve eccedere i **15** caratteri.

Sintassi:
```
TEXTFILE '<nome_risorsa>'
BEGIN
  ...selezione target...
  FROM '<percorso_sorgente>'
  TO '<percorso_destinazione>'
END
```

Esempio:
```
TextFile 'MYFILE'
Begin
  ...
  From 'c:\user\config.bin'
  To '/dd/user/appl/config.dat'
End
```

TARGET

Significato: Indica il nome del codice target in cui deve essere inclusa la risorsa. Consultare la precedente sezione (Opzioni del compilatore) per ulteriori informazioni sui target possibili. L'istruzione «**Target**» può essere presente più volte all'interno dello stesso blocco di risorsa, per permettere più target. Non è possibile usare questa direttiva se è stato specificata la direttiva «**AnyTarget**» .

Sintassi: **TARGET '**<target_name>**'**

Esempio:
```
BinaryFile 'MYFILE'
Begin
  Target 'ISA86M'
  Target 'ISA68M'
  ...
End
```

ANYTARGET

Significato: Indica che la risorsa deve essere aggiunta al codice di tutti i target prodotti dal Generatore di Codice. Il Generatore di Codice ISaGRAF può produrre molteplici codici target con un unico comando «**Compila**». Non è possibile usare questa direttiva se sono state inserite una o più direttive «**Target**».

Sintassi: **ANYTARGET**

Esempio:
```
ULongData 'MYDATA'
Begin
  AnyTarget
  ...
End
```

FROM

Significato: Indica il percorso sorgente (sul PC in cui è installato l'ambiente ISaGRAF) di una risorsa **BinaryFile** o **TextFile**. I caratteri usati per separare i componenti del percorso (drive, cartella, prefisso, estensione) devono essere conformi al sistema MS-DOS.

Sintassi: **FROM '**<percorso sorgente>**'**

Esempio:
```
BinaryFile 'MYFILE'
Begin
  ...
  From 'c:\user\config.dat'
  To '/dd/user/appl/config.dat'
End
```

TO

Significato: Indica il percorso destinazione (sul sistema target) di una risorsa **BinaryFile** o **TextFile**. I caratteri usati per separare i componenti del percorso (drive, cartella, prefisso, estensione) devono essere conformi al sistema MS-DOS.

Sintassi: **TO '**<percorso destinazione>**'**

Esempio:
```
TextFile 'MYFILE'
Begin
  ...
  From 'c:\user\config.dat'
  To '/dd/user/appl/config.dat'
End
```

Esempio

Quello che segue è un esempio completo di un file di definizione delle risorse:

```
(* file di definizione delle risorse *)

ULongData 'DATA1'             (* elenco di valori *)
Begin
   Target 'ISA68M'            (* solo per questo target *)
   1, 0, 16#1A2B3C4D, +1, -1  (* valori numerici *)
End

VarList 'VLIST1'              (* elenco di variabili *)
Begin
   Target 'ISA68M'            (*solo per questo target *)
   Valve1, StateX, Command, Alrm1 (* nomi di variabili *)
End

BinaryFile 'FILE1'            (* risorsa file binario *)
Begin
   AnyTarget                  (* rivolto a tutti i target *)
   From 'c:\user\updatef.bin' (* file sorgente sul PC *)
   To 'updatef.cfg'           (* file target sul PLC *)
End

TextFile 'FILE2'              (* risorsa file di testo *)
Begin
   Target 'ISA68M'
   From 'c:\nw\nwbd.txt'      (* file sorgente sul PC *)
   To '/nw/dat/nwbd'          (* file target sul PLC *)
End
```

Compilazione delle risorse

Se sono state inserite delle risorse nel file di definizione, in fase di compilazione viene avviato anche il compilatore delle risorse. Nel caso vengano rilevati degli errori nel file di definizione delle risorse, viene visualizzata una finestra nella quale è riportata la lista degli errori. Premendo «**Exit**» si abbandona la finestra ed in questo caso, le risorse non verranno aggiunte al codice ISaGRAF.

Implementazione

Il numero di risorse, la dimensione delle righe dati ed i file non sono limitati da ISaGRAF. Le risorse vengono aggiunte alla fine del codice generato con una cartella di risorse. Quello che segue è il formato (in notazione «C») della cartella delle risorse.

```
__RESOURCE:
{
  long nbres;                 /* numero di risorse definite */
  {
    char name[16];            /* nome risorsa */
    long type;                /* tipo di dati delle risorsa */
    long size;                /* dimensione del blocco dati */
    void *data;
    char *path_offset;        /* puntatore ad una stringa */
  } /*nb di records */
}
```

Quelli che seguono sono i possibili valori del campo «type»:

- 1 = file binario
- 2 = file testo
- 3 = dati ulong (il campo path_offset in questo caso non viene usato)
- 4 = elenco di variabili (il campo path_offset in questo caso non viene usato)

Per i file di testo, i caratteri di fine riga vengono tradotti dal compilatore di risorse, secondo le convenzioni del sistema target. Tutti i puntatori hanno offset a 32 bit a partire dall'indirizzo della corrispondente struttura. Tutti i nomi e percorsi delle risorse sono stringhe terminate da NULL. Percorsi e dati seguono la cartella delle risorse.

A.14 Riferimenti incrociati

L'ambiente ISaGRAF incorpora un editor di riferimenti incrociati che permette una visione totale delle variabili dichiarate nei programmi del progetto e dove vengono usate. La funzione dei riferimenti incrociati è di elencare tutte le variabili dichiarate nel progetto e di localizzare nel sorgente di ogni programma le parti di codice che ne fanno uso. I riferimenti incrociati sono molto utili per avere la visione globale del ciclo di utilizzo di una variabile. Aiutano ad individuare gli effetti collaterali e a ridurre il tempo di apprendimento per le operazioni di mantenimento del progetto. I riferimenti incrociati possono anche essere usati per ottenere una visione globale dell'intero dizionario di un progetto, per trovare le variabili inutilizzate e valutare la complessità del progetto.

Il comando **«Progetti / Riferimenti incrociati»** della finestra Programmi avvia l'editor dei Riferimenti incrociati. La lista di sinistra visualizza gli oggetti dichiarati nel progetto (programmi, variabili e nomi definiti) e gli elementi della libreria (funzioni e blocchi funzione) che hanno riferimenti nel progetto. La lista di destra mostra le ricorrenze dell'oggetto correntemente evidenziato nei programmi del progetto in questione.
La descrizione di un occorrenza include il nome del programma, il numero del passo o della transizione SFC, il numero del blocco o del Test FC, il numero della linea per un linguaggio testuale, le coordinate per un diagramma LD o FBD. Per i diagrammi Quick LD la descrizione viene completata con il numero del ramo, se la variabile viene utilizzata come uscita (associata a una bobina) il numero il numero del ramo e seguito da un carattere asterisco («*»).
Impostando **«Visualizza variabili non utilizzate»** nel menù **«Opzioni»** della finestra Riferimenti incrociati, vengono visualizzate nella lista principale anche le variabili che non sono utilizzate nei programmi del progetto.

Selezione del tipo di oggetto

Dal momento che in un progetto può essere dichiarato un gran numero di oggetti, la casella di selezione nella barra strumenti dell'editor permette di selezionare il tipo di oggetti da visualizzare nell'elenco. In questo modo è possibile avere accesso solamente alle informazioni selezionate. Ogni volta che i riferimenti incrociati vengono ri-calcolati, la selezione viene impostata a «**Tutto**» per raffigurare l'elenco completo.

Ricalcolare i riferimenti incrociati

Il comando «**File / Ri-calcola**» permette di aggiornare in qualsiasi momento i riferimenti incrociati in base alle modifiche apportate nelle finestre di modifica ISaGRAF.

Esportare i riferimenti incrociati

Il comando «**Strumenti / Esporta (file testo)**» permette di stilare l'elenco completo dei riferimenti incrociati in un file testo ASCII. Questo file può, poi, essere aperto con altre applicazioni come il blocco note Windows o dei word processor.

Errori nel Dizionario

Il comando «**Modifica / Errori nel dizionario**» visualizza in una finestra di dialogo l'elenco degli errori rilevati al caricamento del dizionario del progetto.

Statistiche

Il comando «**Strumenti / Statistiche**» visualizza in una finestra di dialogo il numero di oggetti e variabili dichiarate nel progetto, in base al tipo di variabile ed agli attributi. In particolare questo comando permette di conoscere il numero di variabili di I/O dichiarate nel progetto, informazione utile per consentire la compilazione nel caso si stia usando una versione limitata dell'ambiente di lavoro ISaGRAF.

Ricerca nell'elenco degli oggetti

Il comando «**Modifica / Cerca**» permette di individuare direttamente un oggetto dell'elenco. Non è possibile trovare un oggetto che non è presente nell'elenco (nel caso di una visualizzazione filtrata). Si raccomanda, prima di effettuare la ricerca, di attivare «**Tutto**» nella barra strumenti.

Aprire un programma

L'elenco sulla destra contiene le ricorrenze dell'oggetto selezionato nei file sorgenti e nelle connessioni di I/O del progetto aperto. Il comando «**Modifica / Apri programma**» consente di aprire direttamente il programma in cui compare la ricorrenza selezionata dell'oggetto. E' anche possibile aprire il corrispondente programma con un doppio click sulla ricorrenza (nell'elenco di sinistra delle ricorrenze).

A.15 Utilizzo del debugger grafico

ISaGRAF incorpora un completo debugger grafico e simbolico. Il comando «**Debug**» della finestra Programmi avvia il debugger per controllare l'applicazione scaricata sul PLC destinazione. In questo modo il debugger comunica con il sistema destinazione tramite collegamento hardware. Il comando «**Simulazione**» della finestra Programmi avvia contemporaneamente il debugger ed un completo simulatore del target. Questo permette di controllare le applicazioni anche quando il sistema di I/O del target non sia ancora completo. La finestra Debugger contiene i comandi per controllare l'intera applicazione.

Quando viene avviato, il debugger, se l'applicazione del PLC destinazione è la stessa di quella dell'ambiente di lavoro, provvede ad aprire automaticamente la finestra Programmi in modalità debug. I comandi di questa finestra permettono di aprire le altre finestre ISaGRAF (editor grafici e testuali, il dizionario, elenchi di variabili, connessioni di I/O…). Tutte la finestre aperte durante una sessione di debug operano in **modalità debug**, il che significa che non è possibile apportare modifiche. I componenti del programma visualizzato (passi, transizioni, variabili…) vengono mostrati con lo stato o valore corrente che hanno in esecuzione. Un doppio click su un oggetto ne modifica lo stato o il valore nell'applicazione destinazione.

Eseguendo il debugger in **modalità simulazione**, le comunicazioni con il sistema destinazione ISaGRAF vengono interrotte. Il debugger comunica solamente con la finestra del simulatore. Poiché il sistema destinazione non è raggiungibile in questa modalità, i comandi «**Scarica applicazione**», «**Avvia applicazione**» e «**Ferma applicazione**» o non sono disponibili nel menù del debugger.

A.15.1 La finestra del debugger

La finestra Debugger contiene informazioni relative allo stato dell'applicazione. Collegata alle altre finestre ISaGRAF diventa un completo sistema interattivo di debug. Gli errori rilevati in esecuzione vengono mostrati nell'area in basso della finestra Debugger. I comandi del menù «**Opzioni**» permettono di nascondere, visualizzare o cancellare l'elenco degli errori.

Il pannello di controllo (l'area posta sotto al menù del debugger) mostra lo stato globale dell'applicazione destinazione ed informazioni relative al tempo ciclo di esecuzione. Quello seguente è l'elenco dei possibili stati che può assumere il target:

Collegamento…:.....Il debugger stabilisce una comunicazione con il sistema target.

Disconnesso:.....Il debugger non riesce a comunicare con il sistema target. Controllare la validità dei cavi di connessione e dei parametri di comunicazione.

Nessuna applicazione:..La connessione funziona, ma non sono attualmente presenti applicazioni ISaGRAF sul sistema target. Scaricare un'applicazione.

'*nome*' Attivo: .....La connessione funziona ed esiste un'applicazione attiva sul sistema target. Il debugger sta stabilendo la comunicazione con l'applicazione, se è la stessa dell'ambiente di lavoro...
IN ESECUZIONE: .....L'applicazione target è in modalità «Tempo Reale».
FERMO: .....L'applicazione target è in modalità «Ciclo a ciclo».
BreakPoint: .....L'applicazione target è in modalità «Ciclo a ciclo» poiché è stato raggiunto un breakpoint.
Errore fatale: .....L'applicazione target è terminata a causa di un grave errore.

Le possibili informazioni sul tempo ciclo di esecuzione sono le seguenti:

Permesso: .....timing programmato.
Corrente: .....timing esatto dell'ultima esecuzione completa del ciclo.
Massimo:.....massimo timing rilevato dall'avvio dell'applicazione.
Overflow:.....numero di cicli di esecuzione con un timing superiore al timing permesso.

Tutti i valori temporali sono in millisecondi. I valori temporali non vengono visualizzati quando il debugger viene usato in modalità simulata.

A.15.2 Controllo dell'applicazione

Il menù «**File**» ed il menù «**Controlli**» contengono i comandi per l'installazione ed il controllo sul sistema ISaGRAF target dell'applicazione ISaGRAF correntemente in modifica.

Nota: Alcuni di questi comandi non sono disponibili in fase di simulazione, poiché l'applicazione processata dal simulatore viene gestita automaticamente dall'ambiente ISaGRAF.

Fermare l'applicazione target

Il comando «**File / Ferma applicazione**» ferma l'esecuzione dell'applicazione correntemente attiva nel sistema target ISaGRAF.

Attivare l'applicazione target

Il comando «**File / Avvia applicazione**» avvia l'applicazione che si trova nel sistema target. Quando un'applicazione viene scaricata, viene avviata automaticamente, senza bisogno di usare il comando «**Avvia**», che viene, invece, tipicamente usato dopo il comando «**Ferma**».

Nota: l'applicazione target deve essere fermata (resa inattiva) prima di poter scaricare una nuova applicazione.

Trasferire l'applicazione

Il comando «**File / Scarica applicazione**» permette di trasferire il codice dell'applicazione al sistema target. Selezionare il tipo di codice da scaricare in base al processore del sistema target ed alle opzioni dell'applicazione.

Numero di versione

Il comando «**File / Numero di versione**» permette di vedere l'identificazione completa sia dell'ambiente di lavoro che delle applicazioni target. L'applicazione di lavoro è quella correntemente aperta nell'ambiente di lavoro ISaGRAF. L'applicazione target è quella che viene eseguita sul PLC target ISaGRAF. Vengono visualizzate le seguenti voci:

VERSIONE: .....Rappresenta il numero di versione del codice dell'applicazione. Questo numero viene calcolato dal generatore di codice.
DATA: .....Questa voce mostra data ed ora della creazione del codice.
CRC: .....Questo è un checksum (numero di verifica) calcolato sulla tavola dei simboli. Questo numero viene calcolato dal generatore di codice e dipende dai contenuti del dizionario delle variabili.

Nota: Il comando «**Numero di versione**» è disponibile anche durante la simulazione. In modalità debug reale, questo comando non può essere richiamato senza che il target PLC sia connesso.

Modifica on line

Il comando «**File / Aggiornamento applicazione**» permette di modificare «on line» l'applicazione target in esecuzione. Questo comando viene trattato in dettaglio nelle sezioni successive di questo capitolo. Non è disponibile quando il debugger viene usato in modo simulazione.

Modalità tempo reale

Il comando «**Controlli / Tempo reale**» non è disponibile senza un'applicazione attiva. Imposta l'esecuzione dell'applicazione target nella normale modalità «tempo reale»: i cicli di esecuzione vengono scanditi dal tempo ciclo programmato.

Modalità ciclo a ciclo

Il comando «**Controlli / Ciclo per ciclo**» non è disponibile senza un'applicazione attiva. Imposta l'esecuzione dell'applicazione target nella normale modalità «ciclo per ciclo»: in questa modalità i cicli vengono eseguiti uno ad uno, in base ai comandi «**Esegui un ciclo**» dati dal menù del debugger.

Esegui un ciclo

Quando l target si trova in modalità ciclo a ciclo, il comando «**Controlli / Esegui un ciclo**» avvia l'esecuzione di un ciclo.

Tempo di ciclo

Il comando «**Controlli / Cambia il tempo di ciclo**» permette di modificare il tempo ciclo programmato. Questo tempo viene visualizzato come «**Permesso**» nella barra di controllo della finestra del debugger. Il tempo ciclo può essere modificato solo in modalità «**Ciclo per ciclo**». Il tempo ciclo viene inserito in milli-secondi come numero intero.

Rimozione di tutti i breakpoint

Il comando «**Controlli / Elimina tutti i breakpoint**» rimuove tutti i breakpoint correntemente installati (raggiunti o ancora attivi) in tutta l'applicazione. I breakpoint esistenti non vengono automaticamente rimossi alla chiusura della finestra del debugger.

Sbloccare le variabili I/O

Il comando «**Controlli / Sblocca tutte le variabili di I/O**» sblocca tutte le variabili di I/O correntemente bloccate nell'applicazione. Quando una variabile di I/O è bloccata, nessun cambiamento di stato in ingresso o in uscita viene effettuato sul corrispondente dispositivo di I/O. Le variabili assegnate agli I/O vengono ancora scritte dall'applicazione o dal debugger. Le variabili ancora bloccate non vengono automaticamente sbloccate alla chiusura della finestra del debugger.

A.15.3 Opzioni

Il menù «**Opzioni**» contiene le opzioni per controllare le informazioni visualizzate nella finestra del debugger.

- Impostazione dei parametri di comunicazione
- Visualizzazione del tempo ciclo
- Visualizzazione degli errori in esecuzione
- Azzeramento degli errori
- Visualizza o nasconde la barra strumenti del debugger

Parametri di comunicazione

I parametri di temporizzazione di comunicazione possono essere modificati quando il debugger è attivo. Da qui possono essere impostati solo il Time out di comunicazione e il Tempo di rinfresco. Gli altri parametri di comunicazione (baudrate, frequenza, parità…) devono essere impostati dal menù «**Debug**» della finestra Programmi.

Per **Time out di comunicazione** si intende il tempo lasciato al sistema destinazione per rispondere ad una richiesta dell'ambiente di lavoro. Il **Tempo di rinfresco** è il periodo di tempo utilizzato per l'INVIO delle richieste «**di lettura**» da parte del debugger per l'aggiornamento dei dati nelle finestre aperte.

Tutti i valori temporali vengono visualizzati ed inseriti come numeri interi in **millisecondi**. Non è possibile impostare i parametri temporali di comunicazione con il debugger in modalità simulazione.

Opzioni di visualizzazione

L'opzione «**Visualizza tempi di ciclo**» nasconde o mostra i valori del **tempo ciclo** nella barra di controllo del debugger. Quando questa opzione è abilitata, tutti i componenti del tempo ciclo (permesso, corrente, massimo, overflow) vengono visualizzati ed aggiornati. Disabilitando questa opzione si riduce il carico di comunicazione del debugger.

Quando l'opzione «**Visualizza errori**» è abilitata, gli errori di esecuzione rilevati vengono elencati nell'area inferiore della finestra del debugger. Quando questa opzione e disabilitata, la lista degli errori viene chiusa. Disabilitando questa opzione,

si riduce il carico di visualizzazione e comunicazione del debugger. Il comando «**Opzioni / Cancella errori**» azzera l'elenco degli errori di esecuzione correntemente visualizzato nella finestra del debugger.

Il comando «**Opzioni / Minimizza finestra**» riduce la dimensione della finestra del debugger che viene visualizzata come un piccolo pannello sempre in primo piano contenente lo stato dell'applicazione e dei pulsanti grafici per i comandi più frequentemente usati.

A.15.4 Comandi «scrittura»

Il debugger simbolico ISaGRAF offre molti comandi per cambiare il **valore** o lo **stato** dei componenti dell'applicazione. La selezione del componente da modificare avviene con un doppio click sul nome o sul disegno in una finestra di editing, quando la finestra del debugger è aperta.

Variabili

Lo stato di una variabile viene modificato con un doppio click sul nome, in una delle seguenti finestre:

- Dizionario
- Elenchi di variabili o diagrammi temporali
- Programmi LD o FBD
- Connessioni I/O

I seguenti comandi sono contenuti nella finestra di dialogo di debug:

- Assegnazione di un nuovo valore alla variabile
- **Blocca** una variabile di I/O (solo per le variabili I/O)
- **Sblocca** una variabile di I/O (solamente per le variabili I/O bloccate)
- **Avvia** o **Ferma** una variabile timer (modalità aggiornamento automatico)

I valori simbolici usati per rappresentare i valori booleani «FALSE» e «TRUE» sono le stringhe definite per la specifica variabile nel dizionario (riquadro **Valori** della finestra **Variabile booleana**). Il valore analogico specificato per un comando «**Scrivi**» deve essere inserito in formato intero o reale, in base alla definizione della variabile data nel dizionario. La stringa specificata per un comando «**Scrivi**» per un messaggio non può essere più lunga della capacità del messaggio associato alla variabile nel dizionario.

Oggetti SFC

I comandi del menù «**File**» della finestra Programmi aperta in modalità di debug, permettono di controllare i programmi SFC in fase di debug dell'applicazione. Il programma SFC deve essere evidenziato dall'elenco dei programmi. Sono disponibili i seguenti programmi:

Avvia programma SFC: ..... Abilita il programma evidenziato, inserendo un token SFC in ciascuno dei passi iniziali.

Termina programma SFC:..... Termina il programma evidenziato, rimuovendo tutti i token esistenti.

Sospendi programma SFC: ... Rimuove tutti i token esistenti del programma evidenziato e ne memorizza la posizione.

Riavvia programma SFC: ..... Riavvia un programma sospeso posizionando nuovamente i token che erano stati rimossi dal comando «Sospendi».

Per quanto riguarda i programmi figli, questi comandi corrispondono alle funzioni **GSTART**», «**GKILL**», «**GFREEZE**» e «**GRST**» del linguaggio di programmazione.

In fase di debug, è possibile eseguire un'operazione di controllo di un **passo SFC** con un doppio click sulla sua rappresentazione grafica nella finestra di editing SFC.

Il debugger ISaGRAF permette inoltre di:
- Inserire un breakpoint all'**attivazione** del passo
- Inserire un breakpoint alla **disattivazione** del passo
- **Azzerare** i breakpoint inseriti nel passo

Selezionare il passo desiderato nel diagramma SFC editato in modalità debug e quindi utilizzare il comando **«Modifica / Breakpoint»**.

Nota: I breakpoint di attivazione e disattivazione non possono essere contemporaneamente presenti in uno stesso passo.

In fase di debug, è possibile eseguire un'operazione di controllo di una **transizione SFC** con un doppio click sulla sua rappresentazione grafica nella finestra di editing SFC.

Il debugger ISaGRAF permette inoltre di:
- Inserire un **breakpoint** all'attivazione della transizione
- Eliminare un breakpoint precedentemente aggiunto alla transizione
- **Attivare** manualmente la transizione (sposta o aggiunge i token)

Selezionare la transizione desiderata nel diagramma SFC editato in modalità debug e quindi utilizzare il comando **«Modifica / Breakpoint»**.

Nel caso di una transizione sono possibili due tipi di attivazione: Incondizionata e Condizionata. Con l'**Attivazione condizionata**: viene creato un token nei passi successivi alla transizione. Vengono rimossi i token dei passi precedenti. Con l'**Attivazione incondizionata**: viene creato un token nei passi successivi la transizione. I token presenti nei passi precedenti non vengono rimossi.

A.15.5 Modifica on line

Con la modifica «on line», è possibile apportare modifiche all'applicazione durante l'esecuzione del processo. Questo tipo di intervento si rende necessario nel caso di processi chimici in cui una qualsiasi interruzione potrebbe mettere a repentaglio la produzione o la sicurezza. Questa funzione va usata con **molta cautela**. ISaGRAF potrebbe non essere in grado di rilevare tutti i possibili conflitti generati dalle operazioni definite dall'utente in seguito a modifiche on line.

Sequenze di codice

Poiché ISaGRAF offre molte possibilità di accedere dal debugger a variabili, programmi o schede di I/O, le funzioni di modifica «on line» esaminate qui, si riferiscono solamente alle modifiche di sequenze di codice. Una sequenza di codice è costituita da un insieme completo di istruzioni ST, IL, LD o FBD eseguite in una riga. In un programma della sezione Iniziale o della sezione Finale, una sequenza di codice è costituita dall'intero elenco di istruzioni del programma. In un programma SFC (sezione sequenziale) una sequenza di codice corrisponde alla programmazione di secondo livello di un passo o di una transizione. La modifica «on line» consiste nel sostituire una o più sequenze di codice senza che venga fermata l'esecuzione del ciclo PLC. Poiché il controllo dei token SFC è assai critico, **non è possibile modificare una struttura SFC con l'aggiunta, la rinumerazione o la rimozione di passi, transizioni o programmi SFC**.

Variabili

Il database delle variabili è un componente molto importante dell'applicazione, altre applicazioni possono accedervi in qualsiasi momento (su PLC multi-processo). Dal debugger è anche possibile apportare modifiche ai valori delle variabili. Per questa ragione**, ISaGRAF non permette, on line, di aggiungere, rinominare o rimuovere una variabile**. È, tuttavia, possibile modificare il modo in cui una variabile viene usata dall'applicazione. È anche possibile, nella prima versione dell'applicazione, riservare delle variabili interne o di I/O «non utilizzate», in modo che modifiche successive possano farne uso.

Esistono diverse tipologie di variabili nel data base del target ISaGRAF. Limitazioni agiscono su ciascuna di esse:

- *Varibili dichiarate*

Sono quelle dichiarate usando il dizionario di ISaGRAF. Non possono essere né cambiate né rinominate «on line». E' raccomandabile che nell'applicazione siano dichiarate e inizializzate alcune variabili extra, anche se non attualmente utilizzate. In seguito a future modifiche, tali variabili extra permetteranno all'applicazione di continuare a lavorare senza cambiamento del numero di checksum.

- *Istanze di blocchi funzione*

Ogni istanza di blocco funzione scritto in linguaggio «C» o in IEC corrisponde a dati memorizzati nel data base di esecuzione del target ISaGRAF. Quando vengono aggiunte o sottratte istanze di blocchi funzione, non è più possibile il cambiamento «on line». E' consigliabile lavorare in linguaggio ST con istanze BF dichiarate nel dizionario, piuttosto di aggiungere blocchi (che corrisponderanno a nuove istanze automaticamente dichiarate) in linguaggio Quick LD o con diagrammi FBD. Inoltre, ogni modifica nella definizione di blocchi funzione disponibili nella libreria di ISaGRAF renderà impossibile il cambiamento «on line».

- *Passi*

Ad ogni passo SFC corrisponde una parte del data base dove sono memorizzati gli attributi dinamici dei passi SFC (attività, tempo, e flag). Aggiungere o rimuovere

passi SFC cambia il data base dell'applicazione e non sono permessi cambiamenti «on line».

- *Variabili nascoste allocate dai compilatori*

Il compilatore ISaGRAF genera variabili temporanee «nascoste» per risolvere espressioni complesse. In qualche caso, il cambiamento di un espressione può comportare la modifica dell'insieme di variabili temporanee nascoste, ciò renderà impossibile il cambiamento «on line». Per evitare tale situazione, si possono aggiungere le seguenti righe nel file ISA.INI, di modo da forzare l'allocazione di un numero minimo di variabili temporanee nascoste in ogni programma, anche se non utilizzate durante la compilazione della prima versione. Di seguito vengono dati dei valori come esempio:

```
[DEBUG]
MNTVboo=8  ; per booleani
MNTVana=4  ; per interi e reali
MNTVtmr=4  ; per timer
MNTVmsg=2 ; per messaggi
```

Quando queste istruzioni vengono scritte in ISA.INI, il compilatore produce un messaggio di avviso nel caso una nuova compilazione dell'applicazione porti ad un numero superiore di variabili temporanee allocate.

Input e output

Poiché il sistema I/O di ISaGRAF è molto aperto, eventuali modifiche richieste devono essere implementate dall'OEM, usando le caratteristiche specifiche del corrispondente hardware. Il sistema ISaGRAF non permette, «on line», **di aggiungere, collegare o rimuovere variabili I/O o di modificare la descrizione di una scheda I/O**. Operazioni come la modifica dei parametri delle schede ed il blocco di canali I/O, sono disponibili con le caratteristiche standard OEM e la funzione «**OPERATE**».

Operazioni in esecuzione

La modifica di un'applicazione in esecuzione avviene con le seguenti operazioni:

- modificare il codice sorgente dell'applicazione nell'ambiente di lavoro
- generare il nuovo codice dell'applicazione
- trasferire il codice della nuova applicazione usando il comando «**Aggiornamento applicazione**» invece di «**Scarica applicazione**»
- passare dalla precedente alla nuova applicazione durante i cicli di esecuzione PLC usando il comando «**Attiva aggiornamento**».

Questa procedura garantisce che sul PLC sia sempre in esecuzione un'applicazione affidabile e permette di controllare il tempo delle operazioni campione in modo molto efficiente e sicuro. Permette anche di modificare spesso il progetto. Per quanto riguarda il processo stesso, la modifica «on line» corrisponde, essenzialmente, alla serie di comandi «**Ferma**, **Scarica** e **Avvia** «. La sola differenza è che lo stato delle variabili non viene perduto ed il tempo dello scambio è molto breve (generalmente la durata di 1 o 2 cicli). Durante lo scambio, le variabili non subiscono modifiche: **tutte le variabili interne di input o output mantengono**

lo stesso valore prima e dopo la modifica dell'applicazione. Nel momento dello scambio, non vengono eseguite operazioni ed i **token SFC non vengono mossi**.

Requisiti di memoria

Per supportare la modifica «on line», il PLC destinazione deve avere abbastanza spazio libero in memoria per contenere la versione modificata del codice dell'applicazione. Durante lo scambio, entrambe le versioni del codice dell'applicazione devono essere contenute nella memoria del PLC.

Limitazioni

Come precedentemente descritto, sono possibili modifiche solamente a sequenze codice. Non possono essere modificate definizioni di variabili, parametri dell'applicazione e connessioni di I/O. Durante il trasferimento di una versione modificata dell'applicazione, ISaGRAF esegue un confronto tra l'applicazione modificata e quella in esecuzione, al fine di rilevare i cambiamenti non sicuri. Nel caso lo scambio si riveli pericoloso o impossibile, viene generato un errore di trasferimento. Una delle operazioni svolte da ISaGRAF consiste nel confrontare il checksum della tavola dei simboli, per rilevare la modifica del nome di qualsiasi variabile, programma o elemento SFC. Le azioni non-memorizzate (N) di un passo attivo al momento dello scambio vengono perdute. Non vengono eseguite le azioni di attivazione del nuovo passo. Le azioni eseguite alla disattivazione del passo sono quelle portate con il nuovo codice dell'applicazione. Se durante lo scambio una transizione è valida, viene aggiornata la condizione ad essa associata. Non viene eseguito il salvataggio di sicurezza sul PLC della nuova applicazione scaricata. Il salvataggio è costituito dalla versione precedentemente scaricata con i comandi standard di trasferimento.

Operazioni

L'aggiornamento del codice di un'applicazione in esecuzione avviene con le seguenti operazioni:

- Prima di apportare qualsiasi modifica ad un'applicazione in esecuzione, si raccomanda vivamente di fare una copia del progetto corrente con un nome diverso. Le modifiche possono essere effettuate sulle copie.
- Prima di modificare qualsiasi programma, assicurarsi che l'opzione «**Aggiorna diario**» del programma di editing sia abilitata, per facilitare la manutenzione del programma in futuro.
- Una volta modificata una o più sequenze (senza che siano state modificate le strutture SFC e la gerarchia dei programmi), è necessario generare nell'ambiente di lavoro il codice della nuova applicazione prima del trasferimento.
- Usando il debugger, con il vecchio progetto, connettersi al PLC destinazione ed eseguire tutte le operazioni che possano rendere più veloce o sicuro l'aggiornamento dell'applicazione.
- Usando il debugger, con il nuovo progetto, connettersi al PLC destinazione. Se è stato cambiato il nome dell'applicazione, non è possibile accedere al data base del target. È necessario usare il comando «**File / Aggiornamento applicazione**».
- Il trasferimento dell'applicazione modificata avviene selezionando l'opzione «**Aggiornamento Successivo**» . Questo può rallentare leggermente il PLC nella fase di trasferimento.

• Al completamento del trasferimento, è possibile eseguire il comando «**File / Attiva aggiornamento**» per avviare lo scambio nel momento più opportuno. Lo scambio avrà la durata di 1 o 2 cicli.

Una volta eseguito correttamente lo scambio, vengono mostrati i programmi dell'applicazione modificata in esecuzione. In caso contrario, rimane l'applicazione in esecuzione esistente.

A.15.6 Scambi DDE

Il debugger ISaGRAF comprende un server DDE (Dynamic Data Exchange). È possibile stabilire un ciclo di notifica tra il debugger ed altre applicazioni, per la visualizzazione dinamica del valore corrente delle variabili in applicazioni non ISaGRAF.

Il server DDE del debugger ISaGRAF supporta solamente le transazioni «advise» e «poke». Le transazioni «request» possono essere usate solo per le variabili che vengono già spiate in un ciclo di notifica. Non sono disponibili altri servizi DDE come «execute». Una volta stabilito un ciclo di notifica su una variabile, il valore viene aggiornato nell'applicazione client ogni volta che cambia. È possibile esaminare qualsiasi tipo di variabile. L'identificazione del collegamento dinamico usa i seguenti nomi:

Service name: .... «ISaGRAF»
Topic name: ..... Nome del progetto ISaGRAF
Item name: ..... Nome della variabile

Nel caso la variabile sia locale ad un programma, il suo nome deve essere seguito dal nome del programma cui appartiene scritto tra parentesi con la seguente sintassi:

nome_variabile(nome_programma)

Il server DDE del debugger ISaGRAF agisce solamente sull'applicazione ISaGRAF correntemente in fase di debug. Il server ISaGRAF è in grado di esaminare fino a **256** variabili. Il server DDE funziona con il debugger sia connesso che in simulazione. Il tempo di aggiornamento è quello impostato tra il debugger ed il sistema destinazione o il simulatore ISaGRAF.

A.16 Lista di variabili spiate

Il comando «**Lista variabili spiate**» del menù «**Strumenti**» della finestra Debugger, permette di creare liste di variabili, i cui valori vengono aggiornati durante la sessione di debug. Le liste vengono generate durante il debug dell'applicazione. Le liste possono essere salvate su disco e aperte in altre sessioni di debug. Una lista può contenere fino a **32** variabili. In una stessa lista possono coesistere variabili di tipo diverso, con campo di validità sia globale che locale. La lista di variabili si riferisce ad un determinato progetto.
Le liste di variabili risultano particolarmente utili per la verifica funzionale dell'applicazione. Permettono di vedere i cambiamenti che avvengono in una ben determinata porzione del processo in esame, in maniera indipendente dal corrispondente codice sorgente dei programmi dell'applicazione. Le liste di variabili si rivelano anche utili per il debug di programmi testuali come ST e IL. Si può facilmente raggruppare in una lista un insieme di variabili usate in un programma per controllare o esaminare l'esecuzione delle istruzioni.
Per ogni variabile della lista, ISaGRAF mostra il suo nome, il suo valore corrente ed il testo di commento. Le colonne possono essere ridimensionate trascinando le linee di separazione con il mouse nella barra titolo della lista.

Salvataggio delle liste su disco

I comandi del menù «**File**» della finestra Lista di variabili permettono la creazione, l'apertura ed il salvataggio delle liste di variabili. ISaGRAF non limita il numero di liste per progetto. Per assegnare il nome alle liste di variabili da salvare su disco, bisogna rispettare le regole qui esposte:

- il nome non può eccedere gli **8** caratteri
- il primo carattere deve essere una **lettera**
- i caratteri successivi possono essere **lettere**, **cifre** o il carattere «_»
- il nome non tiene conto di maiuscole e minuscole

L'editor delle liste non è in grado di visualizzare più di una lista di variabili nella stessa finestra. Avviare più volte l'editor delle liste per vedere contemporaneamente diverse liste.

Inserimento di variabili nella lista

Il comando «**Modifica / Inserisci**» della finestra Lista di variabili permette di inserire variabili nella lista. Il nome della variabile viene selezionato dalla lista degli oggetti definiti nel dizionario del progetto. Questo evita di dover inserire manualmente gli identificatori di variabili. La nuova variabile viene inserita prima della variabile correntemente evidenziata nella lista. La lista non può contenere più di **32** variabili. La stessa variabile non può comparire più di una volta nella stessa lista.

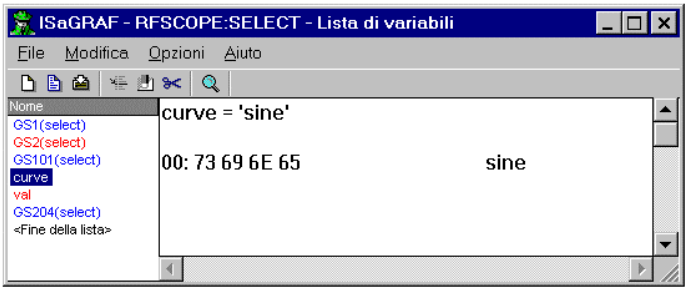
Cambiare la variabile evidenziata

Il comando «**Modifica / Cambia**» della finestra Lista di variabili sostituisce la variabile evidenziata con un'altra variabile. Il comando «**Taglia**» rimuove dalla lista la variabile evidenziata.

Visualizzazione dei dettagli

In ogni momento è possibile visualizzare o nascondere, in una finestra affiancata, i dettagli della variabile selezionata nella lista, premere il pulsante "**Zoom**" della barra strumenti oppure usare il comando "**Opzioni / Visualizza dettagli**".

Quando è selezionato il modo «Visualizza dettagli», solo il valore di una variabile viene visualizzato. Il valore viene visualizzato sia in formato numerico/simbolico, nella parte superiore della finestra, sia in formato binario con ogni byte rappresentato da due cifre esadecimali.



Il modo «Visualizza dettagli» è molto utile per spiare e interpretare messaggi stringa che contengono caratteri non stampabili.

A.17 Debug di programmi ST e IL

Durante una sessione di simulazione o di debug di programmi ST e IL, il testo del programma non può essere modificato.

ST Per i programmi ST, nella parte inferiore della finestra dell'editor, viene inserita una lista delle variabili spiate. Si può ridimensionare tale lista trascinando con il mouse la linea di separazione.

IL Per i programmi IL, le istruzioni sono riordinate e visualizzate in una lista. Il valore corrente di una variabile viene mostrato a fianco dell'istruzione in cui compare. Si può fare doppio click su un istruzione per cambiare il valore della variabile corrispondente.

Per ogni variabile della lista, ISaGRAF mostra il suo nome, il suo valore corrente ed il testo di commento. Le colonne possono essere ridimensionate trascinando le linee di separazione con il mouse nella barra titolo della lista.

Salvataggio delle liste su disco

Il comando «**File / Salva lista**» dell'editor ST in modalità di debug, salva la lista delle variabili sul disco fisso con lo stesso nome del programma editato. Tale lista verrà automaticamente ricaricata ogni volta che un programma ST o IL viene aperto in modalità di debug. La lista può essere liberamente editata e modificata usando il comando «**Strumenti / Lista variabili spiate**» dalla finestra del debugger.

Inserimento di variabili nella lista

Il comando «**Modifica / Inserisci variabile**» dell'editor ST in modalità di debug permette di inserire variabili nella lista. Il nome della variabile viene selezionato dalla lista degli oggetti definiti nel dizionario di progetto. Questo evita di dover inserire manualmente gli identificatori di variabili. La nuova variabile viene inserita prima della variabile correntemente evidenziata nella lista. La lista non può contenere più di **32** variabili. La stessa variabile non può comparire più di una volta nella stessa lista.

Quando il nome di una variabile è evidenziato in un testo ST, premendo il pulsante **«Spia selezione»** nella barra strumenti o eseguendo il comando «**Modifica / Spia selezione**» si aggiunge la variabile alla lista delle variabili spiate.

Cambiare la variabile evidenziata

Quando una variabile è selezionata nella lista delle variabili spiate dell'editor ST in modalità debug, il comando «**Modifica / Cambia variabile**» sostituisce la variabile evidenziata nella lista con un'altra variabile. Il comando «**Taglia variabile**» rimuove dalla lista la variabile evidenziata.

A.18 Rappresentazioni grafiche di variabili

ISaGRAF contiene un editor grafico che consente la creazione di rappresentazioni grafiche animate in fase di debug. Gli elementi grafici devono essere collegati alle variabili del progetto aperto. Gli elementi grafici sono animati «on line» durante il debug.
E' possibile forzare il valore di una variabile, con un doppio click sul corrispondente elemento grafico o sul corrispondente elemento nella lista (gli elementi grafici possono essere visualizzati anche in formato di lista), oppure premendo INVIO quando l'elemento è selezionato.

E' anche possibile bloccare (negare ogni modifica da parte di ISaGRAF) la finestra Rappresentazioni grafiche utilizzando il comando «**File / Blocca**». Quando un documento viene bloccato (non viene animato), si può ancora forzare una variabile facendo doppio click sul simbolo corrispondente.

A.18.1 Disegnare le rappresentazioni grafiche

Una rappresentazione grafica è costituita da una figura di sfondo (facoltativa) e da un insieme di oggetti grafici che verranno animati durante il debug. Per definire una rappresentazione grafica è necessario eseguire le seguenti operazioni: Selezionare una figura di sfondo, inserire gli oggetti grafici, collegare gli oggetti alle variabili del progetto.

Figure di sfondo

Le figure di sfondo sono file «bitmap» (.BMP) o «metafile» (.WMF). Il numero di figure incluse nel grafico non è limitato. Le figure possono essere spostate o ridimensionate. Le figure non compaiono nella lista degli elementi grafici animati. Le figure devono essere generate con altri strumenti grafici non inclusi in ISaGRAF. Il comando «**Opzioni / Colore sfondo**» della finestra Rappresentazioni grafiche è utilizzato per selezionare un colore uniforme per gli spazi vuoti tra gli elementi grafici.

Note: I file Bitmap consumano molta memoria. E' raccomandabile dimensionare correttamente la figura, e limitare lo spazio inutilizzato nel rettangolo della figura bitmap.

Testo singolo

L'elemento grafico **«Testo singolo»** è un campo testo inscritto in un rettangolo. Il testo mostrato è il valore della variabile associata. In tal modo questo elemento può essere collegato a variabili di tipo messaggio.

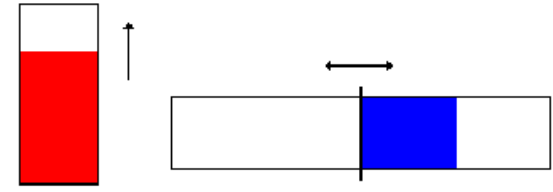
Il rettangolo dove il testo è visualizzato può essere riempito con un colore oppure può essere lasciato trasparente. Quando tale oggetto grafico viene ridimensionato, la dimensione del front di caratteri utilizzato viene aggiustata in modo da riempire l'altezza del rettangolo.

Barre grafiche unipolari e bipolari

L'elemento grafico **«Barra»** è un rettangolo con una porzione colorata che rappresenta il valore numerico della variabile associata. La parte restante del rettangolo può essere riempita con un colore. Una barra grafica può essere sia verticale che orizzontale.

Le barre grafiche unipolari possono crescere in ogni direzione: verso l'alto, verso il basso, verso sinistra o verso destra.

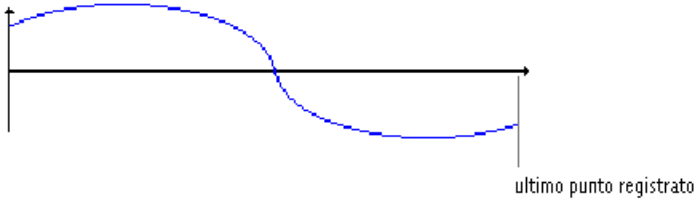
Le barre grafiche bipolari possono crescere sia in direzione positiva, che negativa, in accordo con il valore della variabile associata. Nel caso di una barra bipolare, il massimo valore permesso è lo stesso sia per la scala positiva che per quella negativa.



Curve

L'elemento grafico **«Curva»** mostra l'andamento nel tempo della variabile associata. Sebbene non sia uno strumento di misura preciso, può fornire utili informazioni, durante la fase di debug, in merito al sincronismo tra variabili.

Una curva memorizza gli ultimi 200 valori di una variabile. Il numero dei campioni non viene modificato quando si cambiano le dimensioni di tale oggetto grafico.

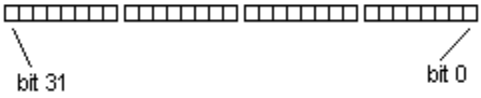


Icone booleane

L'elemento grafico **«Icona booleana»** viene utilizzata per visualizzare uno stato binario. Per lo stato logico «FALSE» (falso) è definito un file icona (.ICO), come pure per lo stato logico «TRUE» (vero). L'ambiente ISaGRAF non include nessun editor di icone, i file icona devono essere generati con altri strumenti grafici.

Campi di bit

L'elemento grafico **«Campo bit»** mostra 32 bit di un valore intero organizzati in un pannello grafico. Il bit meno significativo viene sempre visualizzato sulla destra. Non è raccomandabile utilizzare il Campo bit per altri tipi di dati come ad esempio i reali analogici, poiché l'informazione visualizzata può portare a confusione.



Selezionare, spostare o ridimensionare gli elementi

Un elemento grafico viene aggiunto alla rappresentazione utilizzando le voci del menù **«Inserisci»**, viene avviata la finestra **Stile dell'elemento** che permette di impostarne le caratteristiche.
La maggior parte dei comandi di editing richiede che gli oggetti grafici siano evidenziati. E' possibile selezionare uno o più oggetti grafici. Per selezionare un elemento, l'utente deve semplicemente fare click sul simbolo grafico corrispondente. Per selezionare una lista di oggetti, bisogna trascinare il mouse nell'area grafica selezionando una porzione rettangolare. Tutti gli oggetti grafici che intersecano il rettangolo di selezione sono marcati come «**selezionati**». Un oggetto «**selezionato**» viene rappresentato con dei piccoli quadratini neri attorno al suo simbolo grafico.

Facendo una nuova selezione, ogni oggetto precedentemente selezionato viene deselezionato. Per rimuovere la selezione esistente, basta semplicemente fare click con il mouse in un area vuota, fuori dal rettangolo di selezione.

Per spostare oggetti grafici, bisogna prima selezionarli. Quindi si posiziona il cursore del mouse sul bordo dell'oggetto selezionato e lo si trascina in un'altra posizione.

Per ridimensionare un oggetto, bisogna prima selezionarlo. Quindi si posiziona il cursore del mouse su uno dei piccoli rettangoli mostrati nel bordo di selezione, e lo si trascina nella direzione desiderata. Anche le figure possono essere ridimensionate. In questo caso, il corrispondente file bitmap o metafile viene deformato in modo da riempire il nuovo rettangolo specificato dall'utente.

Raggruppare / Separare gli elementi

Si possono raggruppare assieme gli oggetti grafici, di modo che siano manipolabili come un singolo oggetto. Per costruire un gruppo, bisogna selezionare gli oggetti grafici e quindi eseguire il comando «**Modifica / Raggruppa**». Il comando «**Modifica / Separa**» è utilizzato per separare un gruppo di oggetti in singoli oggetti.

Un gruppo può contenere una figura. Un gruppo può contenere un altro gruppo.

Quando gli oggetti vengono raggruppati, il loro stile non può essere più cambiato. Gli oggetti di un gruppo sono singolarmente visibili, ma non permettono di modificare le variabili associate (con un doppio click).

Nella visualizzazione degli oggetti grafici in formato lista, un gruppo occupa una sola linea.

A.18.2 La lista degli elementi grafici

In ogni momento, l'utente può decidere di visualizzare gli elementi come una «Disposizione grafica» o come una «Lista di oggetti grafici», è possibile commutare tra questi due modi di visualizzazione premendo il pulsante della barra strumenti a fianco riportato. Oppure si può utilizzare il comando «**Opzioni / Lista / Disposizione**» dalla finestra Rappresentazioni grafiche.

Nel modo di visualizzazione come lista, gli oggetti grafici vengono mostrati come una classica lista di finestre. L'altezza riservata a ciascun elemento è calcolata in accordo con il suo stile grafico. Le figure (bitmap e metafile) non sono visibili dalla visualizzazione in formato lista. In tale modalità non è permessa la selezione multipla di oggetti, viene invece permessa la selezione di un singolo oggetto, da utilizzare per impostare lo stile dell'elemento e per cambiare il valore di una variabile.

Si possono riordinare gli elementi della lista utilizzando i comandi «**Modifica / Sposta in su nella lista**» e «**Modifica / Sposta in giù nella lista**». L'oggetto da spostare deve essere selezionato nella lista.

A.18.3 Definire il tipo e lo stile degli elementi

Lo stile grafico e le impostazioni di un oggetto grafico esistente possono essere modificate, facendo un doppio click sul simbolo corrispondente nell'area grafica, oppure mediante il comando «**Modifica / Imposta elemento**» quando l'elemento è selezionato. La finestra di dialogo **Stile dell'elemento** viene aperta anche quando un nuovo oggetto viene aggiunto al documento. L'utente può impostare le seguenti caratteristiche:

- ***Stile grafico e impostazioni:***

  La tipologia grafica (testo singolo, barra grafica, curva…) di un elemento può essere cambiata dinamicamente. Se in un oggetto vengono utilizzati i colori di primo piano e di sfondo, questi possono essere personalizzati utilizzando le corrispondenti caselle. Quando viene selezionata la tipologia Icona booleana, deve essere specificato il percorso dei file .ICO corrispondenti agli stati logici «TRUE» e

«FALSE». Si utilizzano i pulsanti **«…»** a destra delle caselle **Se vero**, **Se falso** per scegliere i file icona presenti su disco.

- ***Fondoscala:***

  Questo è il massimo valore che può essere visualizzato nelle barre grafiche e nelle curve. Per le barre grafiche bipolari e per le curve bipolari, viene utilizzato lo stesso valore assoluto sia per l'asse positivo che per quello negativo.

- ***Nome della variabile:***

  Quando il campo «**Nome**» è attivo, premendo il pulsante **«…»** a fianco della casella l'utente può selezionare il nome della variabile tra quelle precedentemente selezionate nel dizionario.

- ***Legenda:***

  Un testo Legenda può essere mostrato vicino all'elemento grafico. Si può personalizzare la posizione di tale legenda (sopra, sotto, sinistra, destra) ed il suo contenuto. Come legenda si può scegliere: il nome di una variabile, il valore di una variabile, entrambi nome e valore. La personalizzazione non ha effetto nella visualizzazione degli oggetti grafici in formato lista.

- ***Variabile di comando:***

  Se viene selezionata l'opzione **"Variabile di comando»**, l'utente può modificare il valore della variabile associata all'elemento grafico durante il debug, facendo doppio click sul simbolo dell'oggetto grafico.

A.18.4 Comandi del menù «File»

I comandi del menù «**File**» permettono la gestione del file della rappresentazione grafica.

Il comando «**Nuovo**» del menù «**File**» serve per iniziare una nuova rappresentazione grafica. ISaGRAF non limita in alcun modo il numero di rappresentazioni grafiche definibili per un progetto. Una eventuale rappresentazione grafica aperta precedentemente viene chiusa prima dell'editazione di una nuova. La finestra di editing della rappresentazione grafica ISaGRAF non permette la modifica contemporanea di più grafici. Tuttavia, è possibile aprire più finestre di editing ISaGRAF simultaneamente, ciascuna con una rappresentazione grafica diversa.

Il comando «**Apri**» del menù «**File**» chiude la rappresentazione grafica corrente ed avvia la modifica di un altra del progetto corrente. La nuova rappresentazione grafica selezionata sostituisce la corrente nella finestra di modifica. Nella finestra di selezione della nuova rappresentazione grafica, è possibile usare il pulsante «**Elimina**» per cancellarne una esistente, al fine di tenere in ordine la cartella del progetto. Quando viene cancellato una rappresentazione grafica, i file icona e bitmap ad essa riferiti, non vengono cancellati.

Il comando «**Salva**» del menù «**File**» memorizza la rappresentazione grafica corrente su disco. Se si tratta di un documento nuovo, è necessario assegnargli un nome prima di salvarlo. Le regole per attribuire il nome ad una rappresentazione grafica sono le seguenti:

- La lunghezza del file non deve eccedere gli **8** caratteri
- Il primo carattere deve essere una **lettera**
- I seguenti devono essere **lettere**, **numeri** o il carattere «_»
- Il nome è non distingue tra lettere maiuscole e lettere minuscole

Il comando «**Salva come**» del menù «**File**» permette di salvare con un nome diverso la rappresentazione grafica corrente.

A.18.5 Note per gli utenti ISaGRAF V3.2

Dalla finestra Rappresentazioni grafiche si possono leggere grafici e liste di diagrammi temporali costruite con gli strumenti ISaGRAF V3.0 o V3.2. Tali file appaiono nella finestra di dialogo del comando «**Apri**», con la descrizione della loro origine. Tali file possono essere letti e liberamente modificati.

Quando si apre un grafico ISaGRAF V3.2, il documento viene automaticamente marcato come «Bloccato». Se si desidera modificarlo è necessario rimuovere l'opzione **«Blocca»** dal menù «**File**».

Quando si apre un grafico o una lista di diagrammi temporali ISaGRAF 3.2, la finestra Rappresentazioni grafiche propone sempre di salvarli in formato nativo. Quando si chiudono documenti di questo tipo viene automaticamente aperta la finestra **Salva come**.

A.19 Caricamento di applicazioni dal target

ISaGRAF supporta il caricamento nell'ambiente di lavoro di un'applicazione memorizzata sul target. La procedura di caricamento comunica con il target per caricare il codice sorgente zippato (EZS, Embedded Zipped Source code) e quindi rigenerare il progetto originario nell'ambiente di lavoro.
Il progetto funzionate sul sistema target connesso può essere caricato nell'ambiente di lavoro ISaGRAF, se la versione del target è V3.22 o successiva, e se il codice sorgente zippato è stato incluso nell'applicazione.
L'aggiunta del codice sorgente per il suo successivo caricamento è una caratteristica opzionale.

A.19.1 Caricare un progetto dal target

La finestra di dialogo **Caricamento progetto dal Target** viene chiamata con il comando «**Carica progetto**» del menù «**File**» della finestra Gestione progetti. Il caricamento non si riferisce ad un progetto già esistente nell'ambiente di lavoro. Il progetto correntemente selezionato nella lista di Gestione progetti non ha alcuna relazione con il meccanismo di caricamento. Per caricare l'applicazione funzionante sul target si deve:

1- assicurarsi che il target sia correttamente connesso
2- impostare i parametri di comunicazione in accordo con il collegamento
3- premere il pulsante «**Esegui**»

Il caricamento del codice sorgente zippato (EZS, Embedded Zipped Source) e la sua decompressione può richiedere alcuni secondi. Dei messaggi nella finestra di dialogo informano quando il caricamento è stato completato, oppure se sono avvenuti degli errori.
Il nome utilizzato per creare il progetto ISaGRAF è quello letto sul target attraverso la comunicazione. Se questo nome è già utilizzato da un progetto esistente nell'ambiente di lavoro, verrà richiesto all'utente di sovrascrivere il progetto oppure di fornire un nome non utilizzato. Una volta che il caricamento è completo, non è possibile cancellare la registrazione dei codici sorgente caricati in un progetto (nel caso di sovrascrittura di un progetto). Il progetto caricato è ora pronto e può essere aperto.

Errori possibili

I seguenti errori possono essere riscontrati nel caricamento di un progetto. Si viene informati degli errori dalla finestra Caricamento progetto dal Target.

- non può essere stabilita la comunicazione con il target.
- il target connesso è un sistema ISaGRAF con versione precedente alla 3.22
- non è presente l'applicazione funzionante nel target.
- non è presente EZS (codice sorgente zippato) incluso nel target.

A.19.2 Impostazioni di comunicazione

Premendo il pulsante «**Imposta**» nella finestra Caricamento progetto dal target, l'utente può definire i parametri del collegamento utilizzato durante il caricamento, per la comunicazione tra l'ambiente di lavoro ISaGRAF e il sistema target. Prima di far partire il caricamento, bisogna assicurarsi che i parametri configurati corrispondano a quelli del target connesso.

A.19.3 Preparare il progetto per il caricamento sul target

Il generatore di codice di ISaGRAF deve essere informato che il codice sorgente zippato deve essere aggiunto al codice dell'applicazione da scaricare sul target, se si desidera permettere successivamente il caricamento dal target nell'ambiente di lavoro ISaGRAF.
Per eseguire questa operazione, aprire la finestra **Opzioni compilatore** con il comando «**Compila / Opzioni compilatore**» dalla finestra Programmi. Premere il pulsante «**Caricamento**», compare la nuova finestra di dialogo **Preparazione per il caricamento dal target.**
Qui è possibile specificare, come opzione, di includere il codice sorgente zippato nel prossimo scaricamento del codice compilato sul target (scaricamento che verrà eseguito sempre con comando «**Scarica applicazione**» del debugger).
In tal caso, solo i file sorgente strettamente necessari verranno inclusi. Per includere altri file opzionali, bisogna selezionare le apposite caselle di opzione nel riquadro **Includi anche** della finestra **Preparazione per il caricamento dal target**.

Nota importante: Le librerie non vengono scaricate assieme al codice sorgente. Ciò include funzioni e blocchi funzione, schede e dispositivi di I/O.

- ***File opzionali***

Oltre ai file sorgente strettamente necessari, possono anche essere inclusi i seguenti file. Nel caso questi vengano selezionati è necessaria la disponibilità di memoria extra sul target.

Descrizione del progetto
Se non è incluso, la descrizione del progetto dopo il caricamento indicherà solo la data di caricamento.
Password di protezione
La funzione di caricamento non è protetta da password. Se si desidera proteggere il progetto caricato, bisogna includere il sistema di password di protezione nel codice sorgente.
Commenti per i canali di I/O non connessi
ISaGRAF da la possibilità di immettere un testo di descrizione per i canali di I/O non connessi. Non bisogna selezionare questa opzione se si lavora solo con canali di I/O connessi.
Elenco cronologico delle modifiche
Questo è l'elenco cronologico delle modifiche fatte al progetto.
File diario
Il file diario di ogni programma contiene: le note scritte dall'utente, più la storia dei messaggi di uscita prodotti con la compilazione del programma. Includere i file diario può comportare un grande consumo di memoria nel target.

Lista delle variabili e dei diagrammi temporali
Questi sono i file creati durante il debug, contengono i nomi delle variabili delle liste spiate e dei diagrammi temporali.
Grafici, icone e bitmap
Questa opzione include i grafici ISaGRAF, più tutte le icone e i file bitmap, se sono localizzati nella cartella del progetto. Attenzione: includere tali file può comportare un grande consumo di memoria nel target.

A.19.4 Come i sorgenti zippati sono memorizzati nel target

Il sorgente zippato incluso (EZS, Embedded zipped source) è memorizzato nel codice generato assieme alle risorse. Il file delle risorse generato, è chiamato «**EZS**». Se viene selezionata l'opzione per includere il codice sorgente EZS, non bisogna usare questo nome per un'altra risorsa. L'inclusione del codice sorgente EZS non implica nessuna limitazione nella definizione delle risorse. Il file di definizione delle risorse scritto dall'utente non è affetto dall'operazione di inclusione del codice sorgente.

Si faccia riferimento alla documentazione ISaGRAF sul Generatore di codice per ulteriori dettagli ed informazioni sulle risorse.

A.19.5 Requisiti di memoria sul target

Il codice sorgente zippato (EZS, Embedded zipped source) richiede della memoria extra per essere memorizzato con il codice dell'applicazione nel target. Una stima generale prevede per il codice EZS (senza selezionare opzioni extra) uno spazio pari ad una volta e mezzo la dimensione del codice eseguibile. Questo significa che includere il codice EZS moltiplicherà per 2.5 la dimensione del codice scaricato sul target.

Limitazioni speciali possono comparire su alcuni sistemi target basati su memoria segmentata. Poiché i file EZS sono memorizzati come risorse nel codice generato, devono essere memorizzati nello stesso segmento di dati del codice applicazione.

A.19.6 Informazioni sul progetto caricato dal target

Il progetto caricato dal target nell'ambiente di lavoro ISaGRAF contiene tutti i file e i dati richiesti per la ricompilazione. Inoltre, a seconda delle opzioni selezionate durante la compilazione precedente, può contenere file ausiliari come la descrizione del progetto e i file diario del progetto.

Bisogna compilare (comando «**Compila applicazione**») il progetto caricato dal target prima di monitorarlo o di effettuare il debug.
Attenzione: poiché ISaGRAF utilizza la data di compilazione per comparare le applicazioni, all'apertura del debugger si verrà informati che l'ambiente di lavoro e l'applicazione target hanno codice di versione differente.

Nota importante: Le librerie non vengono caricate dal target con il codice sorgente. Bisogna assicurarsi che le appropriate librerie di funzioni e blocchi funzione siano istallate nell'ambiente di lavoro ISaGRAF prima di ricompilare l'applicazione caricata dal target.

A.19.7 Compatibilità delle versioni

Il caricamento da target è supportato dai target ISaGRAF e dagli ambienti di lavoro con versione 3.22 o successiva. Per supportare il caricamento sono state fatte delle estensioni al protocollo di comunicazione.

Non esistono restrizioni nell'inclusione di codice sorgente zippato (EZS, Embedded zipped source) nei target basati su sistemi ISaGRAF con versione da 3.03 a 3.21, poiché i file EZS vengono memorizzati nel codice dell'applicazione come risorse standard. Ma le informazioni aggiunte, in questo caso, non potranno poi essere caricate dal target nell'ambiente di lavoro, poiché tali target non supportano i servizi di comunicazione richiesti.

A.20 Utilizzo dello strumento Diagnosis

«**Diagnosis**» è un sottoinsieme degli strumenti utilizzabili in ambiente di Debug. Permette all'utente finale di lavorare su un insieme predefinito di variabili, in modo da esaminare e controllare il processo. Il Debugger ISaGRAF è uno strumento molto potente che include funzioni di alto livello. Lo strumento Diagnosis fornisce un metodo immediato per controllare l'applicazione sul target in fase di esecuzione o per eseguire la manutenzione. Lo strumento **Diagnosis** viene lanciato direttamente dal gruppo ISaGRAF in ambiente Windows, facendo un doppio click sulla seguente icona:

La lista dei progetti esistenti viene mostrata in una finestra di dialogo. Permette all'utente di lanciare una versione limitata del Debugger ISaGRAF per una applicazione ISaGRAF esistente e scaricata sul target. Premendo il pulsante «**OK**» viene lanciata la versione limitata del Debugger per il progetto selezionato. Premendo il pulsante «**Cancel**» viene chiusa la finestra di dialogo. Il pulsante «**Setup**» è utilizzato per impostare i parametri di comunicazione per il collegamento tra l'ambiente di lavoro ISaGRAF e il PLC target. Per ulteriori informazioni su questo comando fare riferimento al capitolo «Gestione dei programmi» di questo manuale.

Nota: Lo strumento **Diagnosis** (versione limitata del Debugger) non può essere utilizzato per scaricare, per aggiornare o per fermare un'applicazione funzionante sul PLC target.
Se il progetto selezionato nella finestra di dialogo non è lo stesso di quello istallato sul PLC target, non può essere eseguita nessuna operazione.

Quando viene eseguita la versione limitata del Debugger ISaGRAF, correttamente connessa con l'applicazione del target, sono disponibili i seguenti comandi:

- Spy lists of variables (spiare liste di variabili)
- Spy graphic documents with SpotLight (spiare dei documenti grafici mediante l'utilità ISaGRAF «Rappresentazioni grafiche»)

A.21 Usare il simulatore ISaGRAF

Il simulatore di kernel ISaGRAF viene avviato dal debugger con il comando **«Simulazione»** del menù **«Debug»** della finestra Programmi. Il simulatore di kernel è costituito da un completo sistema target ISaGRAF in grado di supportare le peculiarità standard ISaGRAF e tutte le funzioni e blocchi funzione «C» della libreria standard fornite dalla CJ International. Le schede di I/O vengono simulate graficamente in una finestra. è possibile simulare qualsiasi tipo di scheda di I/O, anche le schede definite come **Virtuali** durante la connessione di I/O.

A.21.1 Collegamenti con il debugger

Il simulatore di kernel supporta la completa comunicazione con il debugger ISaGRAF, in modo da poter usare anche in simulazione tutte le possibilità offerte dal debugger. Il simulatore di kernel funziona sempre con il progetto ISaGRAF corrente. Durante la simulazione i comandi del debugger «**Ferma applicazione**», «**Avvia applicazione**», «**Scarica applicazione**» e «**Aggiornamento applicazione**» non sono disponibili. Non è possibile usare il simulatore se non è stato selezionato il target «**SIMULATE**» nelle opzioni del compilatore durante la generazione del codice target. La chiusura della finestra del simulatore implica che anche la finestra Debugger (assieme a qualsiasi altra finestra ISaGRAF aperta durante la sessione di debug) venga chiusa.

A.21.2 Simulazione I/O

Le schede di I/O appaiono nella finestra del simulatore indicate con nome e numero di alloggiamento. Può essere gestito qualsiasi tipo di I/O (booleano, analogico o di messaggio). I canali delle schede di ingresso vengono visualizzati con tasti e campi speciali. I canali delle schede di uscita sono visualizzati con LED grafici di stato e campi dati.

Ingressi booleani: Un ingresso di tipo booleano viene rappresentato da un pulsante quadrato verde. Il numero di canale viene visualizzato con il pulsante di I/O. Il valore di ingresso è «TRUE» quando il pulsante è premuto. Premendo il pulsante si modifica il corrispondente valore della variabile di I/O. Il tasto di destra del mouse permette di impostare l'ingresso solamente quando è premuto il pulsante.

Uscite booleane: Un uscita di tipo booleano viene rappresentata da un piccolo cerchio. Il numero di canale viene visualizzato con il pulsante di I/O. Il valore di uscita è «TRUE» quando il simbolo grafico è evidenziato.

Ingressi analogici: Un canale di ingresso analogico è un semplice campo numerico nel quale è possibile inserire il valore dell'ingresso corrispondente. Con un click sulla casella appare il cursore ed è possibile inserire un nuovo valore per il canale. Non è necessario usare il tasto **INVIO** dopo l'inserimento. Gli ingressi

analogici possono essere inseriti sia in base decimale che esadecimale. Per aumentare o ridurre il valore corrente si possono usare le frecce su/giù.

Uscite analogiche: Un uscita analogica viene rappresentata da un campo numerico di uscita. Il valore di uscita può essere visualizzato sia in formato decimale che esadecimale. Non è possibile agire in alcun modo su un campo di uscita.

Ingressi di tipo messaggio: Un canale di ingresso di tipo messaggio è rappresentato da un semplice campo testo, in cui è possibile inserire il valore dell'ingresso corrispondente. Un click sulla casella fa apparire il cursore ed è possibile inserire un nuovo valore per il canale. Non è necessario usare il tasto **INVIO** dopo l'inserimento.

Uscite di tipo messaggio: Un canale di uscita di tipo messaggio è rappresentato da un campo di testo. Non è possibile agire in alcun modo su un campo di uscita.

A.21.3 Componenti della libreria

Il simulatore ISaGRAF supporta in modo completo conversioni, funzioni e blocchi funzione standard fornite dalla CJ International. Di seguito è riportata la lista degli oggetti.

- ***Funzioni di conversione:***

  bcd, scale

- ***Funzioni:***

  abs, acos, ArCreate, ArRead, ArWrite, ascii, asin, atan, char, cos, delete, expt, find, insert, left, limit, log, max, mid, min, mlen, mod, mux4, mux8, odd, rand, replace, right, rol, ror, sel, shl, shr, sin, sqrt, tan, trunc

- ***Blocchi funzione:***

  average, blink, cmp, ctd, ctu, ctud, derivate, f_trig, hyster, integral, lim_alrm, r_trig, rs, sema, sr, stackint, tof, ton, tp

Le conversioni, le funzioni ed i blocchi funzione «C» definiti dall'utente non sono normalmente integrati nel simulatore ISaGRAF. Normalmente questo tipo di oggetti viene progettato per usare le risorse software ed hardware del sistema destinazione. Risorse che non sono, normalmente, disponibili in ambiente Windows. Il simulatore ISaGRAF, nel caso di conversioni, funzioni o blocchi funzione definiti dall'utente, adotta il seguente comportamento standard:

- Una nuova conversione elaborata dal simulatore, viene sostituita dalla conversione «nulla». Questo significa che il valore fisico delle variabili analogiche è sempre uguale al valore elettrico (come inserito o visualizzato nel pannello del Simulatore).
- Una nuova funzione o blocco funzione «C» elaborata dal simulatore non esegue alcuna operazione. Il valore risultante non viene assegnato.

A.21.4 Opzioni

I comandi del menù «**Opzioni**» nel pannello del simulatore permettono di controllare la visualizzazione degli I/O. E' possibile impostare o rimuovere queste opzioni in qualsiasi momento della fase di debug

- L'opzione «**Display a colori**» visualizza i canali di I/O con bitmap a colori. Su schermi LCD che non permettono di distinguere i colori, è bene disabilitare questa opzione per visualizzare i simboli grafici di ingresso e di uscita dei canali di I/O in bianco e nero.
- L'opzione «**Nomi di variabili**» visualizza una barretta accanto ad ogni canale di I/O con il nome della variabile di I/O connessa. Disabilitando questa opzione si riduce la dimensione del pannello del simulatore
- L'opzione «**Valori esadecimali**» permette di vedere od inserire i canali analogici di input o output in formato esadecimale.
- Con l'opzione «**Sempre in primo piano**» la finestra del simulatore è sempre visibile, anche se viene selezionata una diversa finestra.

A.21.5 Salvataggio e ripristino degli stati di ingresso

Utilizzando il simulatore ISaGRAF, i canali di ingresso sono forzati attraverso operazioni manuali, agendo sui pulsanti di commutazione e sui controlli del pannello di simulazione. In ogni momento si possono usare i seguenti comandi del menù «**Strumenti**» per salvare e ripristinare lo stato di tutti i canali di ingresso:

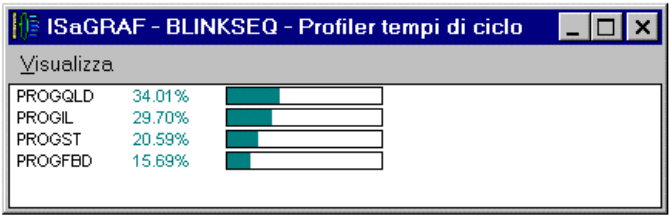
Carica stato ingressi: imposta i valori dei canali di ingresso con i valori memorizzati in un file che è stato creato su disco usando il comando **«Salva stato ingressi».**
Salva stato ingressi: salva lo stato dei canali di ingresso in un file di modo che sia possibile ripristinarli successivamente utilizzando il comando **«Carica stato ingressi»**. Il file è memorizzato nella cartella del progetto e viene salvato assieme agli altri file dal processo di archiviazione di ISaGRAF.

Nota: Solo i canali di ingresso con un nome (quelli che hanno una variabile connessa) sono salvati sul disco.

A.21.6 Profiler dei tempi di ciclo

Il profiler dei tempi di ciclo ISaGRAF è un potente strumento di diagnostica che mostra come sono distribuiti i tempi di ciclo tra i vari programmi, funzioni e blocchi funzione di un applicazione. Questo strumento è molto utile per avere una veloce diagnosi delle prestazioni di un'applicazione, e permette al programmatore di individuare le parti di codice che possono aver bisogno di un ottimizzazione.

Il profiler dei tempi di ciclo viene eseguito con il comando «**Strumenti / Profiler tempi di ciclo**» della finestra del simulatore ISaGRAF. Per ogni programma, funzione o blocco funzione, viene mostrata la percentuale di tempo di ciclo spesa per eseguirlo:



Quando viene impostata l'opzione «**Visualizza / Media**» dalla finestra Profiler tempi di ciclo, le informazioni visualizzate rappresentano una media delle percentuali calcolate dopo che l'applicazione è stata avviata, oppure dall'ultima volta che è stato dato il comando «**Visualizza / Reset**».

Se l'opzione «**Visualizza / Media**» non è impostata, le informazioni visualizzate rappresentano misurazioni eseguite durante l'esecuzione dell'ultimo ciclo. Si può utilizzare questa opzione anche quando l'applicazione viene simulata in modo **Ciclo per ciclo** per avere un insieme di misurazioni che dipendono dal contesto dell'applicazione.

Il comando «**Visualizza / Copia**» copia i nomi del programma e le percentuali negli appunti di Windows in formato ASCII. I dati possono quindi essere incollati in documenti testo o fogli di lavoro.

Note importanti:

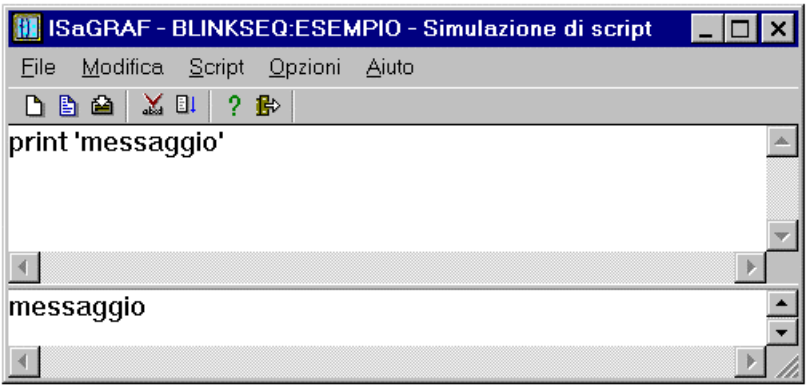
Queste misurazioni non sono esatte. Il calcolo della percentuale è basato sul conteggio dei TIC delle istruzioni, tenendo in considerazione i tempi di esecuzione delle varie istruzioni. Il calcolo non include il tempo speso dalle funzioni «C» e dai blocchi funzione.

Il valore mostrato per una funzione o un blocco funzione è la somma di tutti i «tempi di chiamata» prodotti dai programmi dell'applicazione nello stesso ciclo.

Il calcolo dei tempi è basato sul codice TIC e non fornisce informazioni affidabili se l'attuale codice dell'applicazione è generato nel linguaggio «C» e compilato usando un compilatore «C».

A.21.7 Script di simulazione

Il simulatore ISaGRAF include uno strumento per costruire ed eseguire degli **Script di simulazione**. Uno Script è descritto con un semplice linguaggio simile al linguaggio ST, ed è utilizzato per automatizzare i test con il simulatore ISaGRAF.
L'editor degli script di simulazione viene eseguito con il comando «**Strumenti / Script di simulazione**» dalla finestra del simulatore. Di seguito è riportata la finestra dell'editor di script:



La finestra superiore è un editor testuale dove vengono immesse le istruzioni dello script. Viene utilizzata come le altre finestre di editor testuali in ISaGRAF ed include caratteristiche di alto livello, come ad esempio la selezione del simbolo di una variabile con il mouse. Si possono utilizzare i comandi del menù opzioni «**Opzioni**» per impostare l'intervallo di tabulazione e selezionare il font di caratteri.

La finestra in basso mostra tutti i messaggi di uscita mentre lo script è in funzione. La linea di separazione tra le finestre può essere liberamente spostata per ridimensionare le finestre. La finestra di uscita può essere nascosta durante l'editazione dello script, ma viene automaticamente aperta ogni volta che uno script viene eseguito.

Modifica degli script

Usare i comandi del menù **«File»** per gestire i file script:

Nuovo	Crea un nuovo script senza titolo
Apri	Carica uno script esistente da file
Salva	Salva il testo dello script e il contenuto della finestra di uscita sul disco, nella cartella del progetto
Salva con nome	Salva lo script con un altro nome

Per ogni script vengono creati due file nella cartella del progetto ISaGRAF:

<*nomescript*>.SCC testo dello script (istruzioni)
<*nomescript*>.SCO contenuti della finestra di uscita

dove <*nomescript*> è il nome dello script. Entrambi i file sono file di testo standard e possono essere aperti utilizzando un qualsiasi editor di testi.

Mentre si edita uno script, si può utilizzare il comando «**Modifica / Inserisci simbolo**» per selezionare il nome di una variabile dichiarata da inserire nella posizione corrente.

Esecuzione degli script

Gli script devono essere controllati e compilati prima di essere eseguiti. Se necessario, il controllo sintattico viene eseguito automaticamente con il comando **«Esegui script»**. Dal menù «**Script**» sono disponibili i seguenti comandi:

Verifica.....verifica la sintassi e compila lo script

Esegui script.....inizia l'esecuzione dello script corrente

Nel caso di un nuovo script senza titolo, questo deve essere salvato (deve essere immesso un nome) prima di verificarne la sintassi. Nel caso lo script abbia già un nome, viene automaticamente salvato su disco prima di verificarne la sintassi.

Quando uno script viene eseguito, il suo contenuto non può essere cambiato. Quando è raggiunta la fine dello script, viene mostrato un messaggio di avviso. E' anche possibile abbandonare l'esecuzione di uno script usando il seguente comando del menù «**Script**»:

Termina Script.....termina l'esecuzione dello script

L'esecuzione dello script avviene durante i cicli target. Nel caso di un loop infinito programmato nel ciclo, il simulatore ISaGRAF assicura che questo loop sia sempre interrotto in modo che i cicli ISaGRAF sono sempre eseguiti e le altre applicazioni ISaGRAF non sono bloccate. L'interprete ISaGRAF degli script decide di interrompere l'esecuzione di uno script, se la stessa etichetta viene incontrata più di una volta nello stesso ciclo del target. L'esecuzione dello script può essere normalmente interrotta dalle istruzioni del linguaggio script «Cycle» o «Wait».

Linguaggio per la descrizione degli script

Il linguaggio di descrizione degli script è un linguaggio testuale molto semplice simile al linguaggio ST, ma nel quale ogni istruzione viene immessa su una linea di testo separata, e non necessita di essere terminata da un punto-e-virgola. Usare il seguente pulsante della barra strumenti della finestra **Simulazione di script** per visualizzare la lista delle istruzioni disponibili e per inserire una delle parole chiave nella posizione corrente:

inserisce istruzione (Parola chiave e commento di aiuto)

Ci sono vari tipi di istruzioni. La prima è l'assegnazione (forzatura) di una variabile:
:=..... simbolo di assegnazione

Altre istruzioni permettono di inviare messaggi alla finestra di uscita (quella in basso):

Print	riproduce in uscita una stringa testuale o il valore di una variabile
PrintTime	riproduce in uscita il tempo corrente

Altre istruzioni sono utilizzate per sincronizzare lo script con il ciclo ISaGRAF:

Cycle	interrompe l'esecuzione dello script per un ciclo
Wait	aspetta per un tempo prestabilito

Altre istruzioni sono utilizzate per controllare il flusso di esecuzione dello script:

Etichette	possono essere poste ovunque nello script
Goto	salto incondizionato ad un'etichetta
If goto	salto condizionato ad un'etichetta
End	termina lo script

Il linguaggio script non fa distinzione tra lettere maiuscole e minuscole. Dei commenti possono essere posti alla fine di ogni linea di testo. I commenti possono essere scritti in accordo con le convenzioni ST (tra i caratteri «**(***» e «***)**»), o preceduti da un carattere «**;**».

Simbolo di assegnazione «:=».

Significato: Forza il valore di una variabile ISaGRAF, che può essere una variabile interna, un canale di ingresso o un canale di uscita.

Sintassi: <nomevar> **:=** <espressione_costante>
<nomevar > **=** < espressione_costante>

Argomenti: <nomevar> è un simbolo valido di una variabile dichiarata nell'applicazione, o una variabile di I/O direttamente rappresentata utilizzando le convenzioni di scrittura «%».
<espressione_costante> è un'espressione costante valida, che corrisponde al tipo di variabile specificato. Per il tipo booleano, si possono usare «0» e «1» invece di «FALSE» (falso) e «TRUE» (vero). Per i tipi timer, il prefisso «T#» o «TIME#» può essere omesso.

Note: Una variabile di ingresso forzata con uno script non necessita di essere bloccata. Quando una variabile di ingresso è forzata da uno script, viene aggiornata la rappresentazione dell'ingresso corrispondente.
Attenzione: non forzare le variabili analogiche di ingresso o di uscita collegate ad una tabella di conversione, poiché l'esecuzione di script non supporta le funzioni o le tabelle di conversione.

Esempio:
```
MyBooVar := 1       (* lo stesso di VERO *)
MyIntVar := 1234
MyRealVar := 1.2345
MyMsgVar := 'Ciao'
MyTmrVar := t#12s
```

Istruzione di stampa

Significato: Scrive una stringa o il valore di una variabile nella finestra di uscita. Il testo è riportato su una nuova riga alla fine del testo precedentemente scritto nella finestra di uscita.

Sintassi: **Print '***<testo>***'**

Print <*nomevar*>

Argomenti: <testo> è una qualsiasi stringa di testo racchiusa tra singoli apici
<nomevar> è un simbolo valido di una variabile dichiarata nell'applicazione, o una variabile di I/O direttamente rappresentata utilizzando le convenzioni di scrittura «%».

Note: I valori delle variabili riportati in uscita sono formattati in accordo con le convenzioni IEC.

Esempio: Print 'Ciao'
Print MyBooVar

Uscita: Ciao
MyBoovar = TRUE

Istruzione di stampa del tempo

Significato: Scrive il tempo corrente nella finestra di uscita. Il testo è riportato su una nuova riga alla fine del testo precedentemente scritto nella finestra di uscita.

Sintassi: **PrintTime**

Note: La stampa del tempo è formattata in accordo con la corrente impostazione del sistema Windows

Esempio: Print 'Sono le ore:'
PrintTime

Uscita: Sono le ore:
15:45:22

Istruzione di sospensione per un ciclo

Significato: Sospende l'esecuzione dello script fino a quando il prossimo ciclo ISaGRAF viene eseguito.

Sintassi: **Cycle**

Note: Le istruzioni script sono eseguite all'inizio di un ciclo ISaGRAF. Se il simulatore è in modalità «Ciclo per ciclo», l'istruzione «Cycle» viene immediatamente seguita da un ciclo. Le istruzioni successive dello script verranno eseguite al prossimo comando «Esegui un ciclo» dalla finestra Debugger.

Esempio: *(* dato un programma ISaGRAF che copia A in B *)*
A := 0
Cycle
Print B

```
A := 1
Print B (* non viene eseguito il ciclo / B non è impostato a 1 *)
Cycle
Print B
```

Uscita:
```
B = 0
B = 0
B = 1
```

Istruzione di attesa

Significato: Sospende l'esecuzione di uno script fino e quando non è trascorso un certo ritardo

Sintassi: **Wait** <ritardo>

Argomenti: <ritardo> è un ritardo in accordo con le convenzioni IEC per le espressioni costanti temporali. Il prefisso «T#» o «TIME#» può essere omesso. Il valore del ritardo deve essere compreso tra 10 millisecondi e 1 ora.

Note: L'accuratezza dell'istruzione «Wait» non è precisa e dipende dal sistema Windows. Inoltre, tale ritardo deve essere considerato con un'accuratezza di più o meno un ciclo ISaGRAF.
Quando viene raggiunta un'istruzione «Wait», prima di proseguire nell'esecuzione dello script, vengono eseguiti dei cicli ISaGRAF fino a quando è trascorso il tempo di ritardo.

Esempio:
```
PrintTime
Wait 2s
PrintTime
```

Uscita:
```
15:45:27
15:45:29
```

Etichette

Significato: Le etichette possono essere posizionate in qualunque parte dello script. Sono utilizzate come una destinazione per le istruzioni «Goto» e permettono il controllo del flusso di esecuzione dello script.

Sintassi: <nomeetichetta>**:**

Argomenti: <nomeetichetta> nome unico in accordo con le convenzioni ISaGRAF per la denominazione delle variabili: limitazione a 16 caratteri, inizio con una lettera, seguita da lettere, cifre numeriche o caratteri «_» . Quando viene definito, il nome di etichetta deve essere seguito dal carattere «**:**».

Note: Nessuna istruzione deve essere posta sulla linea dove è definita una etichetta.
Il nome di un etichetta non deve essere lo stesso di un simbolo di variabile ISaGRAF già dichiarata.

Esempio:
```
(* esempio di uno script con un loop infinito*)
loop:
PrintTime
Wait 1s
Goto loop
```

Istruzione di salto

Significato: Salto incondizionato ad un'etichetta

Sintassi: **Goto** <nomeetichetta>

Argomenti: <nomeetichetta> è il nome di un'etichetta definita nello script.

Note: Sono permessi salti all'indietro. Nel caso di un loop infinito, l'esecuzione dello script viene automaticamente interrotta ad ogni loop per preservare l'esecuzione dei cicli ISaGRAF.

Esempio:
```
Print 'Prima del salto'
Goto MyLabel
Print 'Con il salto' (* istruzione mai eseguita *)
MyLabel:
Print 'Dopo il salto'
```

Uscita:
```
Prima del salto
Dopo il salto
```

Istruzione di salto condizionale

Significato: Salto condizionato ad un'etichetta. La condizione è il risultato della comparazione tra due variabili ISaGRAF, oppure tra una variabile e un'espressione costante.

Sintassi: **If** <var1> ***test*** <var2> **Goto** < nomeetichetta>
If <var1> ***test*** <espressione_costante> **Goto** <nomeetichetta>

I ***test*** di comparazione disponibili sono::

- **=** vero se entrambi i membri hanno lo stesso valore
- **<>** vero se i membri hanno valori differenti
- **<** vero se il primo membro è minore del secondo
- **<=** vero se il primo membro è minore o uguale del secondo
- **>** vero se il primo membro è maggiore del secondo
- **>=** vero se il primo membro è maggiore o uguale del secondo

Argomenti: <var1> <var2> sono simboli validi di variabili dichiarate nell'applicazione, variabili I/O direttamente rappresentate utilizzando le convenzioni di scrittura «%».
<espressione_costante> è un'espressione costante valida, che corrisponde al tipo di variabile specificato. Per il tipo booleano, si possono usare «0» e «1» invece di «FALSE» (falso) e «TRUE» (vero). Per i timer, il prefisso «T#» o «TIME#» può essere omesso.
<nomeetichetta> è il nome di un'etichetta definita nello script.

Note: Sono permessi salti all'indietro. Nel caso di un loop infinito, l'esecuzione dello script viene automaticamente interrotta ad ogni loop per preservare l'esecuzione dei cicli ISaGRAF.

Esempio:
```
(* Questo script itera finché MyVar vale TRUE *)
Loop:
If MyVar = TRUE Goto TheEnd
Print MyVar
Goto Loop
TheEnd:
```

Istruzione di fine script

Significato: Termina lo script

Sintassi: **End**

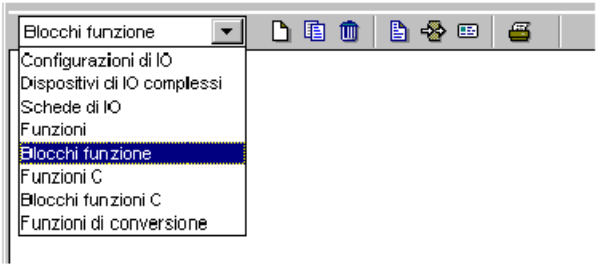
Note: Non è obbligatorio porre un istruzione «End» nell'ultima riga dello script

Esempio:
```
(*Questo script itera finché MyVar vale TRUE *)
Loop:
If MyVar = FALSE Goto Continue
End
Continue:
Print MyVar
Goto Loop
```

A.22 Usare il Gestore di libreria

Le librerie di ISaGRAF forniscono un'interfaccia standard tra l'ambiente di sviluppo dell'automazione e le capacità software o hardware del sistema target ISaGRAF. Esiste una libreria per ogni tipo di interfaccia. La gestione libreria dell'ambiente di lavoro ISaGRAF è rivolta sia ai fornitori di hardware che agli sviluppatori di software, in quanto permette di definire l'interfaccia di programmazione ISaGRAF degli oggetti creati.

La finestra **Librerie** dell'ambiente di lavoro ISaGRAF (comando «**Strumenti / Librerie**» dalla finestra Gestione progetti) mostra gli elementi appartenenti alla libreria di ISaGRAF correntemente selezionata. Per selezionare una delle librerie ISaGRAF si utilizza il comando «**File / Altre librerie**» o si usa la casella ad elenco posta a sinistra della barra strumenti.
Nella parte sinistra della finestra principale viene mostrata la lista di elementi appartenenti alla libreria selezionata. Nella parte destra vi sono le note tecniche (manuale utente) relative all'elemento evidenziato nella lista di sinistra. Il menù contiene i comandi per creare, definire o modificare elementi nella libreria attiva.



A.22.1 Gestione degli elementi della libreria

Per creare nuovi elementi o lavorare su quelli già esistenti di una libreria aperta si usano i comandi del menù «**File**».

Creazione di un nuovo elemento

Il comando «**Nuovo**» del menù «**File**» crea un nuovo elemento nella libreria aperta. Il nome del nuovo elemento viene inserito seguendo le seguenti regole:

- la lunghezza massima del nome è **8** caratteri
- il primo carattere deve essere una **lettera**
- i caratteri seguenti possono essere **lettere**, **cifre** o il carattere «_»
- nell'attribuzione del nome non viene fatta distinzione tra lettere maiuscole e minuscole.

Ad ogni elemento della libreria è associato un testo di commento che viene inserito alla creazione dell'elemento. Quando si crea un nuovo elemento sono necessarie le seguenti informazioni:

- la definizione completa per una configurazione di I/O,
- i parametri per una scheda di I/O,
- l'interfaccia utente per funzioni o blocchi funzione.

Quando si crea una funzione conversione «C», una funzione «C» o un blocco funzione «C» viene automaticamente generato uno schema completo del codice sorgente.

Elaborazione di elementi della libreria

E' possibile modificare il nome o il commento dell'elemento evidenziato nella lista mediante il comando «**File / Rinomina**». Per copiare l'elemento evidenziato nella libreria attiva su un altro elemento della stessa libreria si usa il comando «**File / Copia**». Se esiste già l'elemento di destinazione, tutto il suo contenuto viene sostituito dal nuovo. Se invece non esiste ancora, viene automaticamente creato. Per eliminare dalla libreria un elemento evidenziato si usa il comando «**File / Elimina**». Con i comandi «**Rinomina**», «**Copia**» ed «**Elimina**» è possibile agire sulle seguenti componenti degli elementi:

- note tecniche
- definizioni complete per una configurazione di I/O
- parametri per una scheda di I/O o un'apparecchiatura complessa di I/O
- definizione dell'interfaccia per una funzione o blocco funzione
- codice sorgente di una funzione o blocco funzione scritto in linguaggio IEC
- codice sorgente di una conversione, una funzione o un blocco funzione in linguaggio «C».

⚠ i comandi «**Rinomina**» o «**Copia**» non aggiornano automaticamente il nome nel codice sorgente nel caso di una conversione, una funzione o un blocco funzione «C».

⚠ se l'elemento è una funzione scritta in linguaggio IEC, il nome del parametro di ritorno non viene modificato dai comandi «**Rinomina**» o «**Copia**».

Impostazione di una parola chiave di protezione

Per definire una password di protezione per un elemento selezionato nella libreria aperta si utilizza il comando «**File / Imposta password**». Per ulteriori informazioni circa livelli delle parole chiave e protezione dei dati, consultare la sezione «Parole chiave di protezione» alla fine della prima parte di questo manuale. Le parole chiave si riferiscono solo agli elementi selezionati. Non hanno alcuna influenza sugli altri elementi delle librerie ISaGRAF.

Compilazione di funzioni e blocchi funzione

Selezionando la libreria di funzioni o blocchi funzione scritta in linguaggio IEC, è possibile, con il comando «**Verifica (compila)**» del menù «**File**», controllare la sintassi dell'elemento selezionato e crearne il codice oggetto. E' necessario che le funzioni e i blocchi scritti con i linguaggi IEC vengano compilate senza errori prima di poter essere usate nei progetti ISaGRAF. Questo comando non ha effetto se è selezionata una libreria di tipo diverso.

Note tecniche

Il comando «**Modifica / Note tecniche**» permette di inserire delle note tecniche per l'elemento selezionato nella libreria attiva. Le note tecniche sono inserite con l'editor di testo ISaGRAF e rappresentano la Guida utente dell'elemento. Verranno consultate nella fase di integrazione in un progetto ISaGRAF. Le note tecniche relative all'uso di un elemento dovrebbero contenere la descrizione delle funzioni principali e la spiegazione dettagliata della programmazione, dei parametri, del contesto e dei limiti.
Il comando «**Strumenti / Formato standard note**» permette di definire un formato testo standard per tutti gli elementi della libreria attiva. Quando si inserisce la nota tecnica di un nuovo elemento questo formato verrà proposto come schema principale. Questo consente all'utente di ottimizzare la modifica delle note tecniche.

Parametri

I parametri di un elemento costituiscono l'interfaccia tra le operazioni svolte dall'elemento e l'uso dell'elemento stesso in un'applicazione ISaGRAF. I parametri hanno un differente significato per ogni tipo di elemento della libreria.
I parametri di una configurazione di I/O definiscono l'insieme completo delle schede di I/O della configurazione e delle variabili per i canali di I/O. I parametri di una scheda o di un'apparecchiatura complessa di I/O definiscono le configurazioni fisica e logica della scheda. I parametri di una funzione o blocco funzione rappresentano l'interfaccia dell'elemento in accordo con le convenzioni di chiamata del linguaggio ST. Non sono previsti parametri per la funzione di conversione in quanto essa fa uso di una interfaccia standard predefinita.

Codice sorgente «C»

L'ambiente ISaGRAF permette al programmatore di gestire il codice sorgente di una conversione, di una funzione o di un blocco funzione della libreria. Il codice sorgente di una funzione o blocco scritto in linguaggio IEC è costituito da un testo o da un diagramma descritti con il linguaggio associato alla funzione. Il codice sorgente dei componenti «C» (funzioni «C», blocchi funzione «C» e funzioni di conversione) è suddiviso in due file separati: un **header sorgente «C»** che contiene l'esatta definizione dell'interfaccia secondo la definizione dei parametri dell'elemento e un file **codice sorgente «C»** che contiene l'implementazione delle operazioni dell'elemento.
L'ambiente ISaGRAF genera il file codice sorgente quando viene creato un nuovo elemento della libreria. Provvede anche a creare ed aggiornare l'header sorgente in base alla definizione dei parametri. Il programmatore può usare l'editor di testi ISaGRAF per completare il codice sorgente.

Archiviazione degli elementi della libreria

Il comando «**Strumenti / Archivio**» apre la finestra **Archivio** di ISaGRAF per salvare o ripristinare elementi della libreria. Prima di eseguire il comando «**Archivio**» è necessario selezionare una libreria. La finestra Archivio mostra la lista degli elementi di una sola libreria per volta.

A.22.2 Configurazioni di I/O

La libreria ISaGRAF **Configurazioni di I/O** fornisce un modo semplice per inizializzare nuovi progetti ISaGRAF con configurazioni di I/O predefinite. Una configurazione di I/O comprende:

- un insieme di schede di I/O
- valori predefiniti per i parametri delle schede di I/O
- nomi predefiniti per i canali di I/O.

Quando viene creato un nuovo progetto ISaGRAF usando un elemento della libreria Configurazioni di I/O, viene automaticamente impostata la corrispondente connessione di I/O, inoltre le variabili di I/O associate ai nomi dei canali vengono automaticamente dichiarate nel dizionario del progetto.

La definizione di una configurazione di I/O viene fatta attraverso la finestra Configurazione di I/O di ISaGRAF (la stessa usata all'interno del progetto). Per ulteriori informazioni circa l'uso di questo strumento consultare la sezione «Usare l'editor delle connessioni di I/O» del manuale. Quando si inserisce una nuova scheda di I/O nella configurazione, tutti i canali della nuova scheda vengono dichiarati con nomi standard predefiniti. Il nome standard predefinito di un canale di I/O ha il seguente formato:

<direzione><tipo><numero_slot>_<numero_canale>

Il primo carattere indica la direzione del canale di I/O:
«**I**»..... canale di ingresso
«**O**»..... canale di uscita

Il secondo carattere indica il tipo di canale di I/O:
«**X**» ..... booleano
«**D**» ..... analogico
«**M**»..... messaggio

Ecco alcuni esempi di nomi standard di canali di I/O:
IX0_7..... canale di ingresso di tipo booleano – scheda #0 - canale #7
QD2_4..... canale di uscita di tipo intero - scheda #2 - canale #4

Per modificare il nome predefinito associato ad un canale di I/O si usa il comando **«Modifica / Imposta canale / parametri»** della finestra Connessioni di I/O.

A.22.3 Apparecchiature complesse di I/O

Tutti i canali di una singola scheda sono dello stesso tipo (booleano, analogico, messaggio) ed hanno lo stesso verso (ingresso o uscita). Un'apparecchiatura complessa di I/O rappresenta un dispositivo di I/O con canali di diverso tipo e con versi differenti. Viene rappresentata come una lista di singole schede di I/O ed usa un solo alloggiamento nello schema a cestello della finestra Configurazione di I/O.

Per definire un'apparecchiatura complessa di I/O è necessario definire l'elenco delle singole schede che la costituiscono nonché i parametri dettagliati di ogni singola scheda. L'elenco delle singole schede viene inserito mediante una apposita finestra di dialogo, che viene aperta automaticamente all'atto della creazione di un nuovo elemento della libreria **Dispositivi di I/O complessi**.
Nella finestra Librerie selezionare la voce Dispositivi di I/O complessi. Per creare un nuovo elemento utilizzare il comando «**File / Nuovo**»; per modificare un elemento esistente fare doppio click su di esso, oppure selezionarlo nella lista di sinistra e premere il pulsante «**Visualizza interfaccia / parametri**» sulla barra strumenti (oppure usare il comando «**Modifica / Parametri**»). Viene aperta la finestra **Dispositivo di I/O.**

Premendo il pulsante «**Aggiungi**» è possibile aggiungere una scheda alla fine dell'elenco corrente. Con il pulsante «**Inserisci**» si può inserire una nuova scheda singola prima di quella correntemente evidenziata. Con il pulsante «**Elimina**» si rimuove la singola scheda evidenziata nell'elenco. Per modificare il nome ed i parametri della scheda evidenziata, si usano, rispettivamente, i pulsanti «**Rinomina**» e «**Parametri**». Per una spiegazione più dettagliata dei parametri di una scheda si veda la sezione successiva. Un'apparecchiatura complessa di I/O può comprendere al massimo **16** schede I/O. Il nome delle singole schede (comprese nell'apparecchiatura) non può eccedere gli **8** caratteri.

A.22.4 Schede di I/O

La libreria **Schede di I/O** ISaGRAF definisce un'interfaccia standard tra le variabili dell'applicazione e l'hardware di destinazione. Durante la stesura dell'applicazione tutte le variabili di I/O vengono connesse ai canali delle schede di destinazione I/O. Una scheda di I/O di ISaGRAF viene definita da un nome e da un **«codice chiave OEM»** che identifica il fornitore. Altri parametri della scheda di I/O definiscono la tipologia della scheda I/O (numero di canali, verso dei canali e tipo) e la sua configurazione hardware o software.

Parametri per una scheda I/O

Nella finestra Librerie selezionare la voce **Schede di I/O.** Per creare un nuovo elemento utilizzare il comando «**File / Nuovo**»; per modificare un elemento esistente fare doppio click su di esso, oppure selezionarlo nella lista di sinistra e premere il pulsante «**Visualizza interfaccia / parametri**» sulla barra strumenti (oppure usare il comando «**Modifica / Parametri**»). Viene aperta la finestra **Parametri della scheda di I/O.**
Esistono due diversi tipi di parametri per una scheda di I/O: **parametri comuni** definiti per ogni scheda di libreria ISaGRAF e **parametri OEM** che sono specifici dell'implementazione della scheda e sono dati dal fornitore hardware.

Parametri comuni

I parametri comuni vengono inseriti nella parte superiore della finestra Parametri della scheda di I/O. Essi (insieme al nome della scheda di I/O) identificano l'interfaccia standard delle schede I/O ISaGRAF.

Il «**codice chiave OEM**» è un semplice numero che identifica il **fornitore hardware**. Tutte le schede che hanno lo stesso fornitore devono avere lo stesso codice chiave OEM. Il codice consiste in una **word a 16 bit senza segno** in formato esadecimale. Il codice chiave OEM per **CJ International** è «**1**».
I parametri principali definiscono la topologia della scheda di I/O. Il **Numero di canali** indica il numero di canali liberi sulla scheda. Il **Tipo** della scheda indica il tipo di variabile che può essere connessa ai canali della scheda. La **Direzione** indica se le variabili connesse alla scheda sono variabili di **ingresso** o **uscita**.

Note: Variabili di I/O di diverso tipo o verso non possono essere assegnate alla stessa scheda di I/O ISaGRAF. In questo caso è necessario usare un'apparecchiatura di I/O complessa.

Parametri OEM

I parametri OEM vengono inseriti nella parte inferiore della finestra Parametri della scheda di I/O. Questi parametri sono definiti dal fornitore hardware e sono specifici per ogni scheda. Ci possono essere al massimo **16** parametri OEM per scheda. Una scheda può non avere parametri OEM. Il gestore di libreria ISaGRAF permette al fornitore hardware di definire l'identificazione ed il formato di ogni parametro, nonché il modo in cui il programmatore di automazione può usarlo.

La lista di sinistra contiene l'elenco dei parametri OEM. Ogni parametro viene identificato da un **nome** e da un **numero** logico compreso tra **0** e **15**. L'area di destra contiene la descrizione dettagliata del parametro correntemente selezionato nella lista di sinistra.
Il pulsante «**Pulisci**» cancella la descrizione del parametro e lo toglie dall'elenco.
Attenzione: questo comando è **irreversibile**.

Il nome di un parametro viene usato per identificare il corrispondente campo di input durante la connessione della scheda di I/O, qualora il campo debba essere definito dall'operatore di automazione. Il nome di un parametro deve essere conforme alle seguenti regole:
- La lunghezza massima del nome è **16** caratteri
- il primo carattere deve essere una **lettera**
- i caratteri successivi devono essere **lettere**, **cifre** o il carattere «_».

Il gruppo **Formato** di un parametro definisce il suo formato interno ed il suo formato di ingresso durante la connessione di I/O all'applicazione. Di seguito sono riportati i formati interni disponibili:
word ..... word a 16 bit senza segno
long ..... word a 32 bit senza segno
word hexa ..... word a 16 bit senza segno
long hexa ..... word a 32 bit senza segno
boolean ..... word a 16 bit senza segno (viene usato solo il bit più basso)
character ..... word a 16 bit senza segno (viene usato solo il byte più basso)
string ..... array di 16 byte contenente una stringa terminata da null
float ..... valore in virgola mobile a 32 bit a singola precisione

In seguito si riportano i formati di ingresso disponibili:
word ..... word decimale senza segno

long ..... long word decimale
word hexa ..... word esadecimale senza segno
long hexa ..... long word esadecimale senza segno
boolean ..... «TRUE» o «FALSE»
character ..... carattere singolo
string ..... stringa ascii (massimo 15 caratteri)
float ..... valore in virgola mobile a singola precisione

Il gruppo **Accesso** viene usato per definire in che modo l'utente finale possa accedere al parametro. Se è selezionata l'opzione «**Definito dall'utente**», i parametri vengono mostrati come un campo input durante la connessione di I/O della scheda. Il valore predefinito del parametro OEM viene usato come valore predefinito per la modifica dei parametri. Se è selezionata l'opzione «**Nascosto**», il parametro è una costante e non appare nella casella di connessione della scheda di I/O. Il valore predefinito del parametro OEM costituisce il valore del parametro costante. L'opzione «**Sola lettura**» indica che il parametro è visibile per l'utente ma non può essere modificato. Il suo valore predefinito viene usato come valore costante.

A.22.5 Funzioni e blocchi scritti con linguaggi IEC

ISaGRAF gestisce una libreria di funzioni e blocchi funzione scritti con linguaggi IEC. I linguaggi utilizzabili per la stesura di funzioni o blocchi sono **FBD** (Function Block Diagram), **LD** (Ladder Diagram), **ST** (Structured Text) o **IL** (Instruction list). Si noti che i linguaggi LD e FBD possono essere combinati nelle stesso diagramma. Il linguaggio **SFC** (Sequential Function Chart) non può essere usato per descrivere una funzione o un blocco della libreria. Il linguaggio associato ad un elemento della libreria viene scelto nel momento in cui si crea la funzione e non può più essere modificato.

Compilazione

Funzioni e blocchi definiti nella libreria devono essere compilati (verificati) prima di poter essere usati in un progetto ISaGRAF. Non è necessario modificare nient'altro nella libreria per quanto riguarda funzioni e blocchi. Gli elementi della libreria verranno visualizzati automaticamente nel menù di selezione a casella usando l'editor grafico LD/FBD nello sviluppo di un progetto.

⚠ Una funzione definita nella libreria può chiamare altre funzioni della stessa. Tuttavia il sistema ISaGRAF **non supporta la ricorsività** nelle chiamate a funzione. Un blocco funzione scritto in linguaggio IEC **non** può chiamare altri blocchi funzione (scritti con i linguaggi IEC, «C»).

Inserimento del codice sorgente

Il codice sorgente di una funzione o blocco funzione di libreria viene scritto usando i programmi standard ISaGRAF: editor grafico per i programmi LD o FBD, editor di testi per i programmi ST o IL. Per ulteriori informazioni su questi programmi si vedano i corrispondenti capitoli del presente manuale. Il generatore di codice ISaGRAF per compilare il codice sorgente di una funzione o blocco di libreria può essere chiamato direttamente dalla finestra dell'editor grafico o di testuale.

Dizionario delle variabili locali

Una funzione o blocco funzione di libreria può avere variabili e nomi definiti con campo di validità locale. Per accedere alla dichiarazione delle variabili si utilizza la finestra del dizionario, richiamabile dalla finestra dell'editor nel momento in cui si modifica il codice sorgente della funzione, con il comando «**File / Dizionario**».

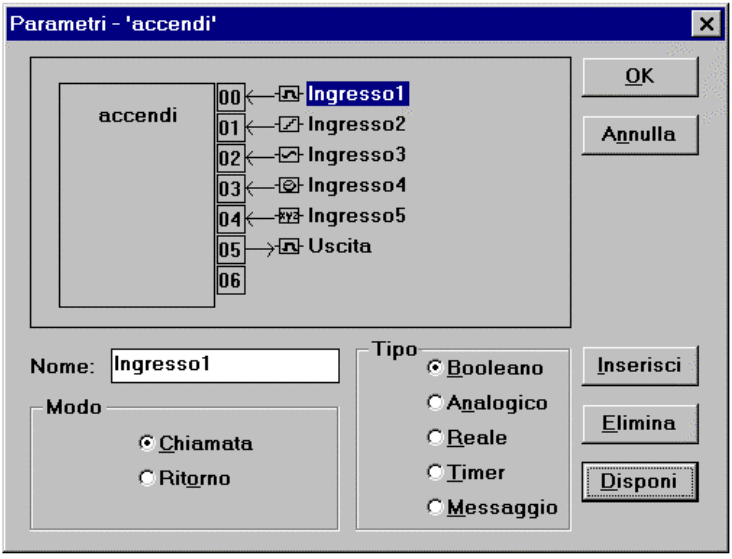
⚠ Una funzione o blocco funzione di libreria non può accedere ad una variabile globale o ad una istanza di blocco funzione. Le variabili locali di una funzione devono essere inizializzate all'interno della funzione.

Le variabili locali di un blocco funzione scritte in linguaggio IEC vengono copiate (istanziate) ogni volta che il blocco viene usato in un progetto. Le variabili locali di un'istanza mantengono il loro valore tra una chiamata e l'altra.

Definizione dell'interfaccia

Nella finestra Librerie selezionare una delle voci **Funzioni, Blocchi funzione, Funzioni C, Blocchi funzione C, Funzioni di conversione.** Per creare un nuovo elemento utilizzare il comando «**File / Nuovo**»; per modificare un elemento esistente fare doppio click su di esso, viene aperto automaticamente l'editor ISaGRAF del linguaggio di programmazione utilizzato per l'elemento in questione.
Selezionando un elemento nella lista di sinistra e premendo il pulsante «**Visualizza interfaccia / parametri**» sulla barra strumenti (oppure con il comando «**Modifica / Parametri**»). Viene aperta la finestra **Parametri** relativa all'oggetto selezionato.

Funzioni o blocchi funzione possono avere complessivamente **32** parametri (di ingresso o uscita). Una funzione ha sempre uno (ed un solo) parametro di ritorno che deve avere lo stesso nome della funzione, per conformità con le norme del linguaggio ST. Un blocco funzione può invece avere più di un parametro di uscita.
Per descrivere i parametri di una funzione o blocco si usa la seguente finestra di dialogo:



L'elenco nella parte superiore sinistra della finestra mostra i parametri ordinati secondo il modello di chiamata: prima i parametri di chiamata, alla fine i parametri di ritorno. La parte più bassa della finestra mostra la descrizione dettagliata del parametro evidenziato nell'elenco. Un qualsiasi tipo di dato ISaGRAF può essere assegnato ad un parametro. Nell'elenco, i parametri di ritorno devono essere posti dopo quelli di chiamata. Per attribuire un nome ai parametri è necessario attenersi alle seguenti regole:

- la lunghezza del nome non può eccedere i 16 caratteri
- il primo carattere deve essere una lettera
- i caratteri successivi possono essere lettere, cifre o il carattere «_»
- nell'attribuzione del nome non viene fatta distinzione tra lettere maiuscole e minuscole.

Si utilizza il comando «**Inserisci**» per inserire un nuovo parametro prima di quello evidenziato. Si usa il comando «**Elimina**» per cancellare il parametro evidenziato. Con il comando «**Disponi**» vengono automaticamente riordinati i parametri, con i parametri di ritorno posti alla fine dell'elenco.

A.22.6 Funzioni e blocchi funzione «C»

Le funzioni e i blocchi funzione «C» sono **funzioni di computazione** chiamate dall'applicazione di automazione, con le convenzioni delle chiamate a funzione del linguaggio ST.

Le funzioni sono processi **sincroni**. L'applicazione target ISaGRAF viene sospesa durante l'esecuzione della funzione. I blocchi funzione associano operazioni e dati non visibili di tipo statico. Ad esempio, un blocco funzione «contatore» rappresenta

l'operazione di conteggio così come il risultato del conteggio. Funzioni e blocchi funzione possono essere utilizzati per completare la capacità standard del linguaggio di automazione o per accedere alle risorse del sistema.

La casella di definizione dei parametri viene utilizzata per la definizione del nome e della tipologia di ogni parametro di chiamata o di ritorno della funzione o del blocco funzione. Per la definizione dei parametri della funzione o del blocco funzione evidenziati si utilizzano i comandi del menù «**Modifica**». Una funzione può avere fino a **31** parametri di chiamata e sempre **un solo** parametro di ritorno. Un blocco funzione può avere fino a **32** parametri con qualsiasi combinazione di parametri di chiamata e di ritorno. Si riporta di seguito la corrispondenza tra le tipologie ISaGRAF e «C»:

BOOLEANO	unsigned long	Word senza segno a 32 bit: 1=true / 0=false
ANALOGICO	Long	Word intera con segno a 32 bit
REALE	Float	Valore in virgola mobile a singola precisione
TIMER	unsigned long	Word intera senza segno a 32 bit (l'unità vale 1 ms)
MESSAGGIO	char *	Stringa di caratteri.

Quando un valore di tipo messaggio viene passato in una funzione o blocco funzione «C», esso non può contenere caratteri nulli. La stringa trasferita al codice «C» viene terminata da null.

Per ulteriori informazioni sulla gestione del codice sorgente «C» di una funzione o blocco funzione e sulle modalità di integrazione di un nuovo elemento nel sistema target ISaGRAF, si veda la Guida Utente al Target ISaGRAF.

A.22.7 Funzioni di conversione

Una funzione di conversione è una funzione «C» chiamata dal gestore di I/O ISaGRAF ogni volta che le variabili analogiche ad essa associate vengono lette dagli ingressi del progetto o vengono inviate alle uscite del progetto.

La funzione crea una relazione tra il **valore elettrico** della variabile (letto dai sensori di ingresso o inviato al dispositivo uscita) ed il corrispondente **valore fisico** (usato nelle espressioni dell'applicazione). La funzione è quindi divisa in due parti: conversione di ingresso e conversione di uscita. Il gestore di libreria ISaGRAF consente il controllo del codice sorgente «C» di una funzione di conversione.

Una conversione può essere usata per una variabile analogica di tipo **intero** o **reale**. Questo implica che l'interfaccia della funzione di conversione è sempre definita con valori in virgola mobile. L'interfaccia è la stessa per ogni funzione di conversione. La definizione «C» di questa interfaccia è contenuta nel file di definizione «**TACN0DEF.H**».

Per ulteriori informazioni sulla gestione del codice sorgente «C» di una funzione di conversione e su come integrare un nuovo elemento nel sistema target ISaGRAF si faccia riferimento alla Guida Utente al Target ISaGRAF.

A.23 Usare il programma di utilità Archivio

Il programma di utilità Archivio di ISaGRAF consente di salvare i progetti e le librerie ISaGRAF su dischetti o cartelle di backup. La gestione archivio di ISaGRAF è una finestra di dialogo che può essere chiamata dalla finestra Gestione progetti di ISaGRAF o dalla finestra Librerie.

⚠ Per creare e mantenere archivi affidabili, si suggerisce di utilizzare le seguenti linee guida:

- Scrivere il nome e la descrizione dell'oggetto salvato sull'etichetta del dischetto
- Non salvare progetti e librerie sullo stesso dischetto
- Non salvare progetti differenti sullo stesso dischetto

A.23.1 Chiamare la gestione Archivio

La finestra **Archivio** può essere chiamata con il menù «**Strumenti / Archivio**» della finestra Gestione progetti, per salvare o recuperare un progetto, oppure dei dati comuni:
La finestra Archivio può essere chiamata anche con il comando «**Strumenti / Archivio**» dalla finestra Librerie di ISaGRAF, per salvare o ripristinare elementi della libreria correntemente selezionata nella finestra stessa:

Progetti

Un progetto viene sempre salvato interamente. Tutti i componenti del progetto (file sorgenti del programma, moduli oggetto e codice eseguibile dell'applicazione) vengono salvati assieme nello stesso file archivio. E' possibile ridurre la dimensione dell'archivio del progetto selezionando l'opzione «**Compressione**».

Elementi di libreria

Gli elementi contenuti nelle librerie ISaGRAF possono essere salvati individualmente. Tutti i componenti di un elemento della libreria (note tecniche, definizione, interfaccia, codice sorgente..) vengono salvati assieme nello stesso file archivio.

Dati comuni

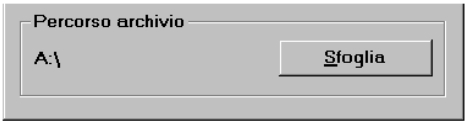
Il comando «**Strumenti / Archivio / Dati comuni**» della finestra Gestione progetti permette all'utente di archiviare e ripristinare i dati con campo di validità **comune** (visibili da ogni programma di ogni progetto) esistenti nell'ambiente di lavoro ISaGRAF. Questo comando non agisce sulle librerie di ISaGRAF. Sotto è riportata la lista dei file che possono essere copiati con questo comando:

common.eqv	Nomi definiti con campo di validità Comune
oem.bat	File di comandi MS-DOS definiti dall'utente

Questi file sono salvati uno per uno sul disco archivio, nella loro forma originale. I file di archivio corrispondenti non sono mai compressi.

A.23.2 Opzioni

Il percorso usato per gli archivi ISaGRAF è mostrato nella parte bassa della finestra di dialogo. Premere il pulsante **«Sfoglia»** per sfogliare i dischi e selezionare un altro disco e un altra cartella di archivio.



Quando l'opzione «**Compressione**» è selezionata, tutti i file di archivio creati durante una procedura di archiviazione sono compressi. Questa opzione è molto utile per ridurre la dimensione di un grande file progetto, e poterlo salvare su un solo dischetto. La compressione di archivio non è generalmente necessaria per i componenti di libreria. Il gestore di archivio di ISaGRAF riconosce automaticamente lo stato di un file archivio (compresso o no) quando recupera tale archivio. Questo implica che l'opzione «**Compressione**» non ha effetto sulla procedura di ripristino.

A.23.3 Archivia e Ripristina

La lista «**Ambiente di lavoro**» (sulla sinistra) mostra gli oggetti esistenti nell'ambiente di lavoro di ISaGRAF istallato sul disco rigido. La lista «**Archivio**» (sulla destra) mostra gli oggetti salvati sul disco di archivio e la cartella del caso specifico.

- ***Archivia***

Per salvare un oggetto in un archivio, è necessario evidenziarlo nell'elenco di sinistra (oggetti di ambiente ISaGRAF) e premere poi il pulsante «**Archivia**». E' possibile evidenziare contemporaneamente più oggetti. Il pulsante «**Archivia**» viene disattivato quando si seleziona un elemento dell'elenco di destra (modalità ripristino).

- ***Ripristina***

Per trasferire un oggetto dall'archivio all'ambiente di lavoro ISaGRAF, è necessario evidenziare l'oggetto nell'elenco di destra (oggetti archiviati) e premere il pulsante «**Ripristina**». E' possibile evidenziare più oggetti dell'elenco. Il pulsante «**Ripristina**» viene disattivato quando si seleziona un elemento dell'elenco di sinistra (modalità salvataggio).

A.23.4 File archivio

Il gestore archivi ISaGRAF crea un unico file di archivio per ogni oggetto salvato. Il file di archivio ha lo stesso nome dell'oggetto salvato. L'estensione del file ne indica il tipo. Queste sono le estensioni usate:

.pia.....progetto
.bia.....scheda di I/O
.iia.....funzioni in linguaggio IEC
.aia.....blocchi funzione in linguaggio IEC
.uia.....funzioni «C»
.fia.....blocchi funzione «C»
.cia.....funzioni di conversione «C»
.ria .....configurazione di I/O
.xia.....apparecchiatura di I/O

A.24 Stampa di un documento completo

Il Generatore di documentazione ISaGRAF consente di stilare e stampare la documentazione completa del progetto evidenziato. La finestra **Generatore documentazione** viene visualizzata con il comando «**Stampa**» del menù «**Progetto**» dalla finestra **Gestione progetti**. Diversamente dai comandi «Stampa» delle altre finestre ISaGRAF, il generatore di documentazione può essere usato per stampare più di un componente del progetto nello stesso documento, con impaginazione e numerazione di pagina.

Con i comandi del menù «**Modifica**» della finestra Generatore documentazione si determinano i componenti del progetto da inserire nella documentazione. In questo modo si costruisce una lista che elenca gli elementi che si desidera includere nel documento. Nella documentazione del progetto possono essere inserite tutte le informazioni riguardanti il progetto (programmi, variabili, opzioni, connessioni I/O…). In questo documento non possono apparire informazioni riguardanti altri progetti di librerie ISaGRAF.

Il comando «**File / Stampa**» genera il documento e lo invia alla stampante, secondo la lista che è stata costruita. L'operazione richiede pochi minuti per generare e stampare il documento. Si raccomanda vivamente di attendere la conclusione dell'operazione di stampa. La generazione del documento può richiedere molto spazio sull'hard disk. Qualora lo spazio non sia sufficiente, appare un messaggio di errore. In questo caso è necessario o liberare spazio su disco eliminando file o riducendo la dimensione della stampa. Una volta dato il comando «**Stampa**» appare una finestra di dialogo che permette di inserire una descrizione dell'attuale stampa. Queste note vengono archiviate in un file cronologico e vengono stampate sulla prima pagina di ogni documento futuro (compreso l'attuale).

A.24.1 Personalizzazione delle tavole dei contenuti

Il menù «**Modifica**» contiene i comandi necessari a definire la lista di contenuti del documento. Una serie di comandi consente di usare una lista predefinita (con tutti i componenti del progetto), di costruire una lista particolare (solo con alcuni componenti) o di spostare i componenti e modificare il contenuto della lista.

Lista predefinita

Il comando «**Lista predefinita**» del menù «**Modifica**» definisce una lista standard per il documento comprendente tutti i componenti del progetto. La lista standard consiste in:

- Descrizione del progetto
- Albero gerarchico (collegamenti tra programmi)
- Codice sorgente per ogni programma
- File Diario per ogni programma
- Definizioni comuni
- Definizioni globali
- Definizioni locali per ogni programma

- Variabili globali
- Variabili locali per ogni programma
- Opzioni dell'applicazione
- Connessioni I/O
- Elenco delle variabili
- Tavole di conversione
- Riassunto dei riferimenti incrociati
- Dettaglio dei riferimenti incrociati
- Sommario delle dichiarazioni
- Mappa degli indirizzi di rete
- Elenco cronologico delle modifiche.

Taglia e incolla

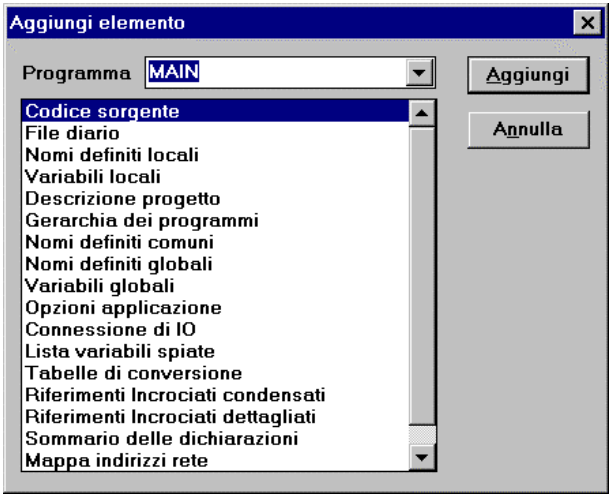
Per spostare il contenuto degli elenchi o per personalizzare l'ordine della lista si usano i comandi «**Modifica / Taglia**» e «**Modifica / Incolla**». E' possibile selezionare, tagliare ed incollare gruppi di più voci.

Svuotare la lista

Il comando «**Modifica / Cancella tutto**». permette di svuotare la lista, in questo modo è possibile ricostruirla completamente inserendo le singole voci.

Inserimento delle voci nella lista

Il comando «**Modifica / Inserisci**» apre la seguente finestra di dialogo. Servirà ad inserire le voci (componenti del progetto) nella lista.

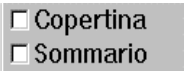


Per inserire una voce relativa ad un programma si seleziona il nome del programma nella casella elenco «**Programma**» e si seleziona la voce desiderata nella lista sottostante. Per inserire la voce selezionata nella lista premere il pulsante

«**Aggiungi**». La stessa voce relativa ad un programma può comparire una sola volta nella lista.

A.24.2 Opzioni

Per definire e personalizzare il formato del documento generato si usano i comandi del menù «**Opzioni**».
Altre opzioni sono disponibili direttamente dalla finestra Generatore di documentazione:



Quando si seleziona l'opzione «**Copertina**», viene stampata una pagina di intestazione all'inizio del documento, contenente il titolo del progetto e le stampe eseguite. Quando tale opzione non è selezionata, la prima pagina inizia con il primo elemento da stampare.

Quando si seleziona l'opzione «**Sommario»** viene stampato, un sommario alla fine del documento generato.

Quando il Generatore di documentazione viene lanciato con un comando **Stampa** da uno degli editor ISaGRAF (es: nell'editor del dizionario con il comando **File/Stampa**), entrambe le opzioni sono inizialmente non selezionate.

Diagrammi SFC

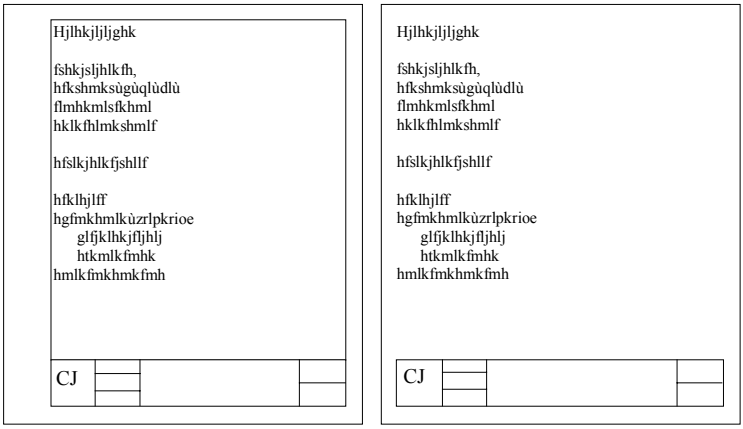
L'opzione «**Separa livelli SFC**» fa si che il sistema stampi, per ogni programma, prima il livello 1 del linguaggio SFC (diagrammi e commenti) e poi il livello 2 di programmazione. Quando non è selezionata questa opzione i livelli 1 e 2 appaiono assieme nella stessa stampa.

Formato pagina

Il comando «**Formato pagina**» del menù «**Opzioni**» permette di definire i principali parametri utilizzati dal generatore di documentazione per l'impostazione della pagina. Possono essere specificati i seguenti parametri:

- **Margine sinistro**: (1 o 2 centimetri, oppure margine assente)
- **Bordo pagina**: Quando è selezionata questa opzione, attorno ad ogni pagina stampata viene tracciato un bordo.

Questi sono alcuni esempi di documenti formattati:



Con margine sinistro e bordo principale niente margine, niente bordo principale

Schema del titolo della pagina

Il comando «**Titolo pagina**» del menù «**Opzioni**» viene utilizzato per definire il contenuto della casella di titolo stampata nella parte inferiore di ogni pagina. L'impostazione standard di questa casella è la seguente:

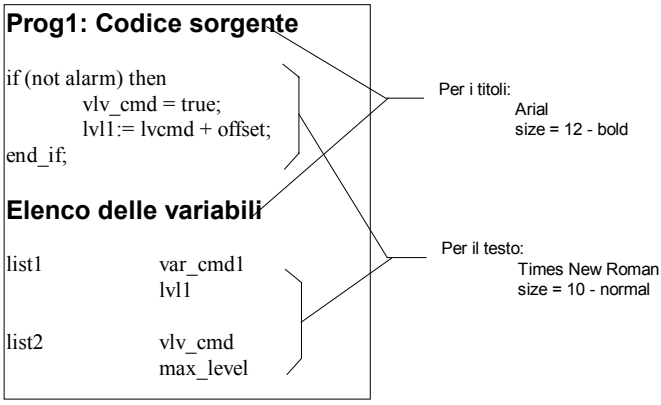
C/A	Text1	ISaGRAF - Progetto 'Nome'	data
	Text2		
	Text3	**Titolo definito dall'utente**	pag

La prima riga del titolo principale (col nome del progetto ISaGRAF), la data corrente ed il numero di pagina vengono generati automaticamente non possono essere modificati.
Le tre righe di testo sulla parte sinistra della casella (text1, text2, text3) e la seconda riga del titolo principale vengono definite dall'utente. L'utente può modificare anche il logo stampato nella casella di sinistra. Per usare un logo diverso è necessario specificare il percorso del file immagine tipo bitmap (.**BMP**). L'immagine scelta potrà avere qualsiasi dimensione e verrà adattata all'esatta dimensione della pagina di stampa. Facendo click nella casella del logo della finestra di dialogo verrà visualizzata la nuova immagine specificata. Il file immagine deve trovarsi sul disco (nella cartella e con il nome specificati) quando viene dato il comando di stampa «**Stampa**».

Scelta dei font di caratteri

I comandi «**Carattere testo**» e «**Carattere titolo**» del menù «**Opzioni**» permettono di definire i font di caratteri utilizzati per la stampa del testo e dei titoli di ogni parte del documento. Per i testi ed i titoli è possibile scegliere la dimensione ed il tipo di carattere. La selezione di un font è fatta con la finestra di dialogo standard di Windows. Ogni testo (programmi testuali, nomi all'interno dei diagrammi,..) viene

stampata con la dimensione, lo stile ed il font di carattere selezionato. I titoli vengono stampati con il font scelto per i titoli. Questo è un esempio di personalizzazione della stampa:



Se non vengono definiti i font di caratteri, per ogni testo verrà usato il font standard della stampante con i seguenti stili:

- Stile «Normale» per i testi ed i nomi all'interno dei diagrammi
- Stile «Grassetto» per i titoli

A.25 Parole chiave di protezione

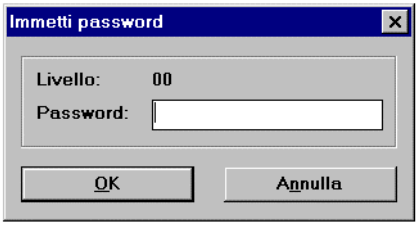
L'ambiente di lavoro ISaGRAF include un sistema di protezione dati completo, che permette all'utente di proteggere oggetti ed elementi della libreria mediante password. Un elemento della libreria può essere una configurazione I/O, una scheda o un dispositivo complesso di I/O, una funzione o un blocco funzione scritti nel linguaggio IEC, una funzione «C», un blocco funzione «C» o una funzione di conversione «C». Il data base di password di protezione è dedicato ed un progetto o a un elemento di libreria, e non può essere condiviso tra ciascuno di questi.

Livelli di protezione

All'interno di un progetto o un elemento della libreria, l'utente può definire fino a **16** livelli di accesso, corrispondenti a password diverse. I livelli di accesso sono ordinati in un sistema gerarchico. Sono numerati da **0** a **15**. Il più alto livello di accesso è numerato come **0**. Quando un utente conosce una password, può accedere a tutti gli elementi protetti dal corrispondente livello di accesso, oltre a tutti quelli protetti con livello inferiore. Ogni comando o dato elementare di un progetto o di un elemento di libreria può essere protetto separatamente con un livello di accesso. Ad esempio, il comando «Compila applicazione» nei vari menù ISaGRAF può essere protetto separatamente. Come dato elementare si intende un programma, una lista di opzioni, le note tecniche di un elemento di libreria, etc...

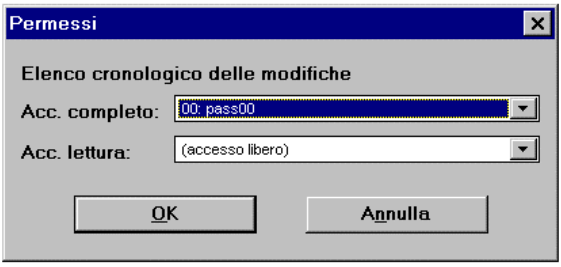
Definizione di parole chiave di protezione

Il comando «**Imposta password**» dei menù ISaGRAF definisce le parole chiave ed i livelli di accesso per un progetto o un elemento di libreria. Questo comando può essere chiamato dalla finestra Gestione progetti (per un progetto) o dalla finestra Librerie (per un elemento di libreria). La prima volta che viene dato questo comando non viene richiesta alcuna parola chiave. Se sono già inserite parole chiave, prima di accedere a questo comando, l'utente dovrà inserire la parola chiave del livello più alto che gli sia nota. Le parole chiave e gli elementi protetti di livello superiore non potranno essere modificati. Il comando «**Imposta password**» consente di definire le parole chiave corrispondenti a diversi livelli di accesso e di proteggere i comandi e i dati elementari dei livelli definiti. Le password (corrispondenti a livelli di protezione) sono immesse facendo doppio click su una linea della lista nella parte superiore. La seguente finestra viene visualizzata per immettere una password.



La lista nella parte bassa mostra i differenti elementi (dati o funzioni) che possono essere protetti, e il livello di protezione corrente associato ad ogni permesso per accesso in **«Lettura»** o accesso **«Completo»**. Assegnando un livello di protezione per l'accesso in «Lettura» si previene che degli utenti senza permesso sufficiente possano aprire o stampare un documento.

Facendo doppio click su una linea nella lista in basso si impostano i permessi per l'elemento o il dato selezionato. La seguente finestra viene visualizzata:



Entrambi i permessi possono essere impostati o come «**accesso libero**», oppure ad un livello di protezione definito da una password. Un permesso di accesso completo non può essere associato ad un livello con minore priorità di uno selezionato per accesso in lettura.

Notare che per quei documenti, naturalmente visibili quando si utilizza l'ambiente di lavoro ISaGRAF, come la descrizione del progetto, l'accesso in lettura non può essere protetto con una password.

Accesso a dati protetti

Quando viene avviato l'ambiente di lavoro non viene richiesta alcuna parola chiave o nome utente. Le parole chiave vanno inserite nella finestra di dialogo ogni volta che si voglia accedere a dati o funzioni protette.

Sarà possibile continuare il lavoro inserendo la parola chiave richiesta o una parola chiave corrispondente ad un livello di protezione superiore. Ogni volta che viene introdotta una parola chiave essa verrà memorizzata e non sarà quindi necessario reinserirla successivamente. Le parole chiave archiviate verranno richiamate ogni volta che un'applicazione ISaGRAF viene avviata da un altro strumento ISaGRAF (ad esempio quando il Project Manager avvia il Program Manager). Le parole chiave vengono mantenute in memoria fintanto che ci sono finestre ISaGRAF aperte. Le parole chiave inserite durante la modifica di un progetto o usando la gestione Librerie o la gestione Archivio non possono essere condivise. Se l'utente inserisce una parola chiave non corretta non avrà accesso alla funzione richiesta.

Collegamenti con la gestione degli archivi

Quando si salva un oggetto (progetto o elemento di libreria) sul disco archivio viene invocata la procedura di protezione dati «**Backup on archive**». Essa fa riferimento

al sistema di protezione dati associato all'oggetto dell'ambiente di lavoro (hard disk). Non viene fatto alcun controllo sul sistema di protezione dati dell'oggetto nel disco archivio se l'oggetto è già esistente. Il comando «**Archivia**» della Gestione Archivio di ISaGRAF salva le informazioni di protezione dati con l'oggetto nel disco archivio.

Quando viene recuperato un oggetto già esistente nell'ambiente di lavoro (disco rigido), viene invocata la procedura di protezione dati «**Overwrite with archive**». Essa fa riferimento al sistema di protezione dati associato all'oggetto nell'ambiente di lavoro (disco rigido). Non viene controllato il sistema di protezione dati dell'oggetto sul disco archivio. Se viene confermato questo comando l'informazione di protezione dati richiamata sostituirà quella esistente sul disco rigido.

Impostazione di protezioni individuali per variabili e canali di I/O

L'ambiente di lavoro ISaGRAF fornisce una sistema di protezione dati basato su una gerarchia di password. Dichiarazioni di variabili e connessioni di I/O possono essere protette globalmente da una password. Inoltre, ISaGRAF permette di impostare una protezione individuale per ogni variabile o canale di I/O. Viene assunto che:

- le password sono precedentemente definite nel sistema (usando il comando «**Progetto / Imposta password**» della finestra Gestione progetti) in modo tale che i livelli di protezione sono disponibili per le protezioni individuali.

- si utilizzano livelli di protezione con priorità più alta per le protezioni individuali rispetto a variabili globali o protezioni di I/O.

Quando una variabile o un canale di I/O ha protezione individuale, una piccola icona viene mostrata vicino al suo nome nel dizionario o nella finestra di connessione di I/O.

Si utilizzino i comandi «**Imposta protezione**» e «**Rimuovi protezione**» del menù «**Modifica**» nella finestra dizionario o della finestra Connessioni di I/O per impostare o rimuovere una protezione individuale per una variabile o un canale selezionato. Entrambi i comandi chiedono di immettere una password valida in modo tale che un livello di protezione possa essere associato alla variabile o al canale. Quindi, ogni volta che si desidera cambiare una variabile o una connessione ad un canale avente protezione individuale si deve immettere una password con sufficiente livello di priorità.

Attenzione: se una variabile o un canale è protetto con un livello, e la corrispondente password viene rimossa dal sistema di protezione, e se nessuna password con livello più alto è definita, la variabile o il canale non può essere più cambiato a meno che una nuova password con sufficiente livello sia definita.

A.26 Tecniche di programmazione avanzata

Questo capitolo contiene ulteriori informazioni sull'ambiente di lavoro ISaGRAF ed il sistema target. L'utente deve aver familiarizzato con strumenti e metodi di ISaGRAF, prima di proseguire nella lettura di questa sezione.

A.26.1 Informazioni aggiuntive sugli strumenti ISaGRAF

Quando si utilizzano gli strumenti editor di ISaGRAF, l'utente può premere il **tasto destro del mouse** per aprire un menù a tendina, che contiene i comandi principali dell'editor. Il menù viene aperto nella posizione corrente del cursore. Questo è molto utile per ridurre le operazioni col mouse nell'esecuzione dei comandi taglia e incolla.

Gli strumenti di ISaGRAF supportano l'**esecuzione multipla**. Sebbene lo stesso oggetto non possa essere aperto due volte per editare lo stesso documento, è possibile aprire finestre differenti con lo stesso strumento ed editare oggetti diversi come operazioni parallele.

Altri comandi sono disponibili per trovare informazioni sui pulsanti grafici nelle barre strumenti. Facendo doppio click su un area vuota di una barra strumenti viene mostrato il contenuto della barra strumenti in un menù a tendina. Fermandosi con il cursore del mouse su un pulsante grafico viene mostrato il corrispondente testo associato al comando.

A.26.2 I/O bloccati ed I/O virtuali

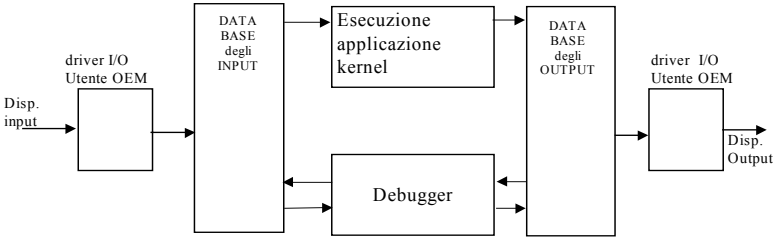
Definendo una scheda di I/O come **Virtuale** viene disconnessa l'elaborazione dei canali I/O fisici. Quando una scheda viene definita come virtuale, le operazioni del kernel ISaGRAF non vengono cambiate. L'unica differenza è che i sensori di ingresso non sono letti e i dispositivi di uscita non vengono aggiornati. In tal modo, è possibile utilizzare il debugger ISaGRAF per modificare i valori degli ingressi. L'attributo Virtuale è applicato ad un'intera scheda. Viene programmato durante la definizione della scheda I/O, **prima** della generazione del codice dell'applicazione. L'attributo Virtuale è una proprietà **statica**, ed è memorizzato quando l'applicazione viene fermata e fatta ripartire.

Un altra possibilità è il **blocco** della variabile di I/O. Il blocco consiste nello sconnettere un dispositivo fisico e la corrispondente variabile I/O di ISaGRAF. Il blocco e lo sblocco di una variabile viene effettuato attraverso il debugger. IL blocco di variabile è un'operazione **dinamica**, e non viene memorizzato quando l'applicazione riparte. L'operazione di **blocco** si applica solo ad **una** variabile (un canale I/O) alla volta. Di seguito sono riassunte le principali proprietà dei controlli I/O:

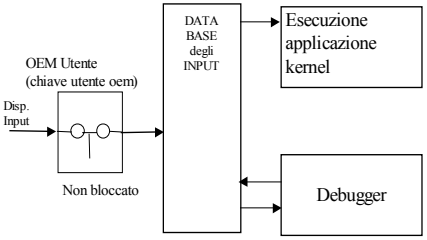
	Attributo virtuale	**Comando di blocco**
Strumento selezionato	connessione scheda di	debugger

	I/O	
Definizione	statico	dinamico
Modo di selezione	scheda	variabile
Applicazione	validazione e test	manutenzione

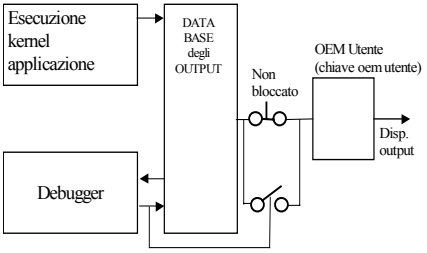
Il diagramma seguente spiega il flusso di dati I/O tra le applicazioni ISaGRAF:



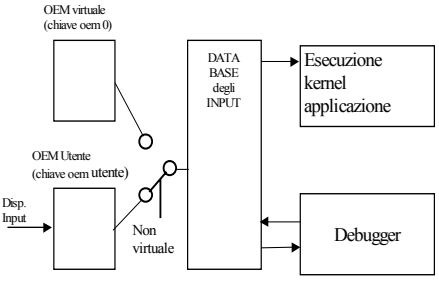
Quando una variabile di ingresso è bloccata, i vari accessi al data base non provocano cambiamenti, il dispositivo di ingresso è disconnesso. I valori degli ingressi possono essere impostati con il debugger ed elaborati dal kernel ISaGRAF:



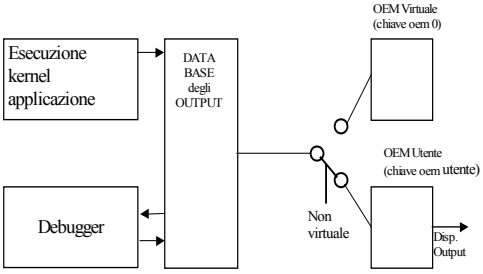
Quando si blocca una variabile di uscita vengono disconnessi il driver di uscita ed il kernel di esecuzione. In questo caso è ancora possibile accedere al dispositivo di uscita, tramite il driver di uscita, con il debugger ISaGRAF:



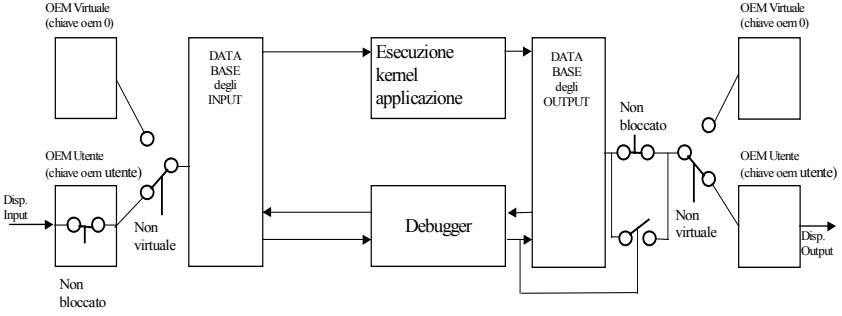
Quando si imposta l'attributo Virtuale per un ingresso, il data base degli ingressi e il dispositivo di ingresso associato sono disconnessi. Un driver I/O virtuale sostituisce quello reale.



Impostando l'attributo Virtuale valgono le stesse regole per una scheda di ingresso o di uscita. Per le schede di uscita, il kernel ISaGRAF aggiorna il data base delle uscite, ma questo data base e i dispositivi di uscita associati sono disconnessi. Un driver di I/O virtuale sostituisce quello reale.



Riassumendo tutte le possibilità:



A.26.3 Validazione del collegamento PC-PLC

La maggior parte dei problemi legati ad una non adeguata comunicazione tra l'ambiente di lavoro ISaGRAF ed il target PLC vengono segnalati nella finestra di debugger dal messaggio di stato «**Disconnesso**». Prima di eseguire qualsiasi test

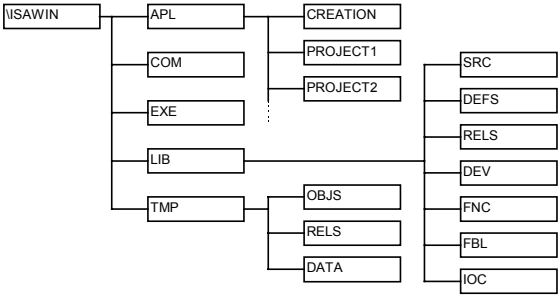
diagnostico è necessario validare la comunicazione senza che vi sia alcuna applicazione attiva nel PLC target. In questo modo il collegamento seriale può essere validato indipendentemente dagli effetti dell'esecuzione.

Il linguaggio «C», usato per la stesura di funzioni di conversione e funzioni «C», consente di accedere direttamente al sistema target. Un errore di programmazione in questa componente software può generare errori di sistema o un funzionamento non corretto del sistema ISaGRAF. Questi problemi si verificano con i driver di I/O sviluppati con l'opzione ISaGRAF **IO development tool.** Ad esempio un errore di sistema può essere causato dalla connessione di una scheda di I/O ad un indirizzo bus non valido. La tavola che segue contiene una sintesi degli errori diagnostici:

stato	Contesto	diagnosi
«non connesso» (prima del caricamento)		- il target non è in esecuzione - cavo mancante o non adatto - parametri di collegamento errati - target ISaGRAF installato non correttamente
«non connesso» (dopo il caricamento)	Avvio in modalità ciclo a ciclo	- configurazione I/O non valida - crollo del sistema
	Avvio in modalità tempo reale	- configurazione I/O non valida - crollo del sistema (dovuto alla programmazione «C»)
«Nessuna applicazione»		- applicazione non caricata - applicazione non avviata (dovuto a programmazione «C») - errata corrispondenza Intel/Motorola - Versione del target non corretta

A.26.4 Cartelle ISaGRAF

L'ambiente ISaGRAF opera su una struttura dedicata di cartelle. La cartella radice di questa architettura viene indicata dall'utente durante l'installazione di ISaGRAF. Il nome predefinito della cartella radice è **ISAWIN.** Viene di seguito riportata l'architettura del disco fisso predefinita, creata dal programma di installazione:



Si riportano le sottocartelle ISaGRAF standard

CARTELLA	CONTENUT0
APL	Cartella radice per i progetti ISaGRAF, ogni progetto corrisponde ad una sottocartelle che contiene tutti i dati del progetto
COM	Dati comuni, possono essere usati da ogni progetto
EXE	Programmi ISaGRAF e file della guida
LIB	Librerie ISaGRAF: - elenchi di elementi - parametri o interfaccia per ogni elemento - note tecniche
LIB\IOC	Codice sorgente per le configurazioni di I/O
LIB\FNC	Codice sorgente di funzioni scritte in linguaggi IEC
LIB\FBL	Codice sorgente di blocchi funzione scritti in linguaggi IEC
LIB\SRC	Codice sorgente per conversioni e funzioni «C»
LIB\DEFS	Header sorgente per conversioni e funzioni «C»
LIB\RELS	Modulo oggetto di conversioni e funzioni «C»
LIB\DEV	File comandi per lo sviluppo di librerie «C» makefile, link list, ecc...
TMP	File temporanei: le sottocartelle di tipo TMP vengono riservate ad ISaGRAF Code Generator e non possono essere cancellate.

Le sottocartelle possono essere spostate ad altra locazione su disco. Se si usa una architettura non-standard, è necessario dichiarare nella sezione **WS001** del file di inizializzazione **ISA.ini** contenuto nella sottocartella **EXE** di ISaGRAF, il percorso delle sottocartelle. Ecco le voci della sezione **WS001**:

Isa	cartella radice per architettura ISaGRAF
IsaExe	cartella radice per programmi e file della guida ISaGRAF
IsaApl	cartella radice per progetti ISaGRAF
IsaTmp	cartella per file temporanei
IsaSrc	cartella per codice sorgente di libreria
IsaDefs	cartella per collettore sorgente di libreria

Si noti che se la voce IsaTmp viene assegnata ad una diversa cartella, è necessario creare nella nuova cartella le sottocartelle OBJS, RELS e DATA.

Il seguente esempio utilizza le voci della sezione **WS001** per ridefinire l'architettura standard di disco ISaGRAF:

;file c:\ISAWIN\EXE\ISA.ini

[WS001]
Isa=c:\isawin
IsaExe=c:\isawin\exe
IsaApl=c:\isawin\apl
IsaTmp=c:\isawin\tmp
IsaSrc=c:\isawin\lib\src

IsaDefs=c:\isawin\lib\defs

Volendo aggiungere funzioni «C» o blocchi funzione al target ISaGRAF, si utilizza la cartella **\ISAWIN\LIB\DEV** per archiviare i file sviluppati: file di comandi, makefiles, mappe, ecc… La cartella **\ISAWIN\LIB\RELS** serve ad archiviare i moduli oggetto generati dalla compilazione «C» e le librerie ISaGRAF richieste per le operazioni di LINK.

A.26.5 Simboli dell'applicazione

Ogni oggetto di una applicazione ISaGRAF viene identificato da un nome (inserito durante la dichiarazione delle variabili) e da un **indirizzo virtuale** interno, calcolato dal generatore di codice. L'indirizzo virtuale di una variabile non è l'indirizzo di rete inserito alla dichiarazione della variabile. Gli indirizzi virtuali vengono usati per comunicazioni e per speciali applicazioni «C» che utilizzano l'opzione OEM. Il generatore di codice ISaGRAF crea un file ASCII con le corrispondenze logiche tra nomi ed indirizzi virtuali per tutti gli oggetti (variabili, programmi, passi,..) del progetto. Questo file può essere facilmente consultato per informazioni circa il data base statico ISaGRAF da qualsiasi applicazione. Il file viene chiamato «**APPLI.TST**» ed è posizionato nella cartella del progetto ISaGRAF: «**\ISAWIN\APL\nomepro**» (nomepro è il nome del progetto). Questa sezione descrive in dettaglio il formato del file «**APPLI.TST**». Le principali notazioni utilizzate per la seguente descrizione, vengono mostrate subito sotto:

VA indirizzo virtuale
ATTR attributo di una variabile
USP funzione «C»

Si riportano in seguito i possibili valori per gli attributi di una variabile che vanno assegnati ai campi «**attributi**»:

+X variabile interna
+C variabile interna di sola lettura
+I variabile di ingresso
+O variabile di uscita

Tutti i numeri, ad eccezione degli indirizzi virtuali, vengono espressi come decimali interi. Gli indirizzi virtuali (**VA**) sono espressi come numeri esadecimali di 4 cifre e sono preceduti dal carattere «**!**». Ad esempio:

123 questo è un numero decimale
!A003 questo è un indirizzo virtuale esadecimale

Viene mostrata in seguito la struttura principale del file «**APPLI.TST**». Il file è strutturato come lista di **blocchi**. Un blocco corrisponde ad un elenco di **record**. Ogni record viene descritto da una riga di testo ed inizia con un'intestazione posta su una riga di testo.

Start block

description blocks
end block

Quella che segue è la sintassi generale di un blocco:

@ <nome_blocco> <argomenti>
#record...
#record...
...

Quella che segue è la struttura del primo blocco, che contiene le principali informazioni riguardanti l'applicazione:

@ISA_SYMBOLS,<appli_crc>
#NAME,<appli_name>,<version>
#DATE,<creation_date>
#SIZE,G=<nbprg>,S=<nbstep>,T=<nbtra>,L=0,P=<nbpro>,V=<nbvar>
#COMMENT,cj international

appli_crc	riassunto dei simboli dell'applicazione
appli_name	nome dell'applicazione
version	numero della versione dell'ambiente ISaGRAF
creation_date	data della generazione dell'applicazione
nbprg	numero di programmi
nbstep	numero di passi SFC
nbtra	numero delle transizioni SFC
nbpro	numero delle funzioni «C» usate
nbvar	numero totale delle variabili

Quella che segue è la struttura dell'ultimo blocco, che segnala la fine del file:

@END_SYMBOLS

Ecco la struttura del blocco utilizzato per descrivere i programmi dell'applicazione:

@PROGRAMS,<nbprg>
#<va>,<name>
#...

nbprg	numero di programmi definiti in questo blocco
va	indirizzo virtuale del programma
name	nome del programma

La struttura del blocco usato per descrivere i passi SFC dell'applicazione è mostrata sotto. Notare che c'è un passo virtuale definito per ogni programma non SFC:

@STEPS,<nbsteps>
#<va>,<name>,<father>
#...

nbsteps numero di passi definiti in questo blocco
va indirizzo virtuale del passo
name nome del passo
father indirizzo virtuale del padre

Quella che segue è la struttura del blocco usato per descrivere le transizioni SFC dell'applicazione:

@TRANSITIONS,<nbtrans>
#<va>,<name>,<father>
#...

nbtrans numero di transizioni definite in questo blocco
va indirizzo virtuale della transizione
name nome della transazione
father indirizzo virtuale del padre

Quella che segue è la struttura del blocco usato per descrivere le variabili booleane dell'applicazione:

@BOOLEANS,<nb_boo>
#<va>,<name>,<attr>,<program>,<eq_false>,<eq_true>
#...

nb_boo numero di variabili di questo blocco
va indirizzo virtuale della variabile
name nome della variabile
attr attributo della variabile
program indirizzo virtuale del programma genitore
o «**!0000**» per una variabile globale
eq_false stringa utilizzata per valori «FALSE» (falso)
eq_true stringa usata per valori «TRUE» (vero)

Quella che segue è la struttura del blocco che descrive le variabili analogiche dell'applicazione:

@ANALOGS,<nb_ana>
#<va>,<name>,<attr>,<program>,<format>,<unit>
#...

nb_ana numero di variabili del blocco
va indirizzo virtuale della variabile
name nome della variabile
attr attributo della variabile
program indirizzo virtuale del programma genitore
o «**!0000**» per una variabile globale
format = «**I**» per una variabile intera
= **«F**» per una variabile real
unit stringa unità

Quella che segue è la struttura del blocco che descrive le variabili timer dell'applicazione:

@TIMERS,<nb_tmr>
#<va>,<name>,<attr>,<program>
#...

nb_tmr numero di variabili di questo blocco
va indirizzo virtuale della variabile
name nome della variabile
attr attributo della variabile (sempre +X: interna)
program indirizzo virtuale del programma genitore
o «**!0000**» per una variabile globale

Quella che segue è la struttura del blocco che descrive le variabili di tipo messaggio dell'applicazione:

@MESSAGES,<nb_msg>
#<va>,<name>,<attr>,<program>,<max_len>
#...

nb_msg numero di variabili in questo blocco
va indirizzo virtuale della variabile
name nome della variabile
attr attributo della variabile
program indirizzo virtuale del programma genitore
o «**!0000**» per una variabile globale
max_len lunghezza massima (capacità dichiarata)

Quella che segue è la struttura utilizzata per descrivere le funzioni «C» utilizzate nell'applicazione:

@USP,<nb_usp>
#<va>,<name>
#...

nb_usp numero di funzioni «C» in questo blocco
va indirizzo virtuale della funzione «C»
name nome della funzione «C»

Quella che segue è la struttura del blocco che descrive le istanze di blocchi funzione «C» utilizzate nell'applicazione:

@FBINSTANCES,<nb_fb>
#<va>,<inst_name>,<fb_name>
#...

nb_fb numero di istanze di un blocco funzione «C» in questo blocco
va indirizzo virtuale dell'istanza di blocco funzione «C»
inst_name nome dell'istanza di blocco funzione «C»
fb_name nome del blocco funzione «C» di riferimento

A.26.6 Limiti dell'ambiente ISaGRAF «LARGE» (WDL)

Esistono alcune limitazioni per gli oggetti usati in ambiente ISaGRAF. Naturalmente altri limiti pratici sono dovuti alla configurazione del computer utilizzato (memoria disponibile e spazio su disco),ed alle capacità del sistema target ISaGRAF (memoria disponibile, risorse software ed hardware disponibili…). Sono riportati di seguito i limiti massimi che non possono essere oltrepassati

❑ Per un progetto:

Oggetto	Massimo	Note
Programmi	255	Compresi programmi principali, sottoprogrammi e programmi figli
Livelli nella gerarchia	20	
Il numero di progetti installati nell'ambiente di lavoro viene limitato esclusivamente dallo spazio su hard disk.		

❑ Per i nomi:

Nomi per:	Massimo	Note
Progetto	8 caratteri	
Programmi	8 caratteri	
Variabile	16 caratteri	+ 60 caratteri di commento
Etichetta di nomi definiti	16 caratteri	
Equivalenze definite	255 caratteri	+ 60 caratteri di commento
Tavole di conversione	16 caratteri	
Elenco variabili	16 caratteri	
funzioni / blocchi funzione (libreria)	8 caratteri	Si applica a funzioni «C», blocchi funzione «C» o funzioni scritte in linguaggio IEC
Parametri di funzione (libreria)	16 caratteri	Si applica a funzioni «C», blocchi funzione «C» o funzioni scritte in linguaggio IEC
Scheda di I/O	8 caratteri	
Configurazione di I/O	8 caratteri	
Parametri scheda OEM	16 caratteri	
Funzioni di conversione	8 caratteri	

❑ Editing (per un programma):

Oggetto	Massimo	Note
Righe SFC	200	
Colonne SFC	10	
Passi SFC	4095	Per l'intero progetto, raggruppamenti di passi, passi iniziali, passi di inizio e fine
Transizioni SFC	4095	Per l'intera applicazione

Simboli SC	200	Un simbolo può essere un passo, una transizione, un macro passo, un passo di inizio e fine, un passo iniziale, un angolo di divergenza o convergenza, o un salto ad un passo.
Modifica LD/FBD	200 colonne 500 righe	Questa è la dimensione dell'area di modifica espressa in unità celle
Modifica Quick LD	nessun limite	I limiti sono imposti dalle capacità del PC
Etichette IL	251	Nello stesso programma IL
Modifica testo	64KBytes	Il testo totale presente in un programma non può eccedere i 64 Kbyte

❑ Per il dizionario (per un progetto):

Oggetto	Massimo	Note
Variabili booleane	4095	
Variabili analogiche	4095	Comprendenti variabili intere e reali
Variabili timer	4095	
Variabili di messaggio	4095	
Il limite dato relativo al numero massimo di variabili booleane, analogiche o di tipo messaggio, comprende variabili interne, di ingresso e di uscita.		
Nomi definiti	4095	Nello stessa lista (stessa validità)
Nomi definiti	255	Utilizzate nello stesso programma
Tavole di conversione	127	Usate nell'applicazione
Punti in una tavola	32	Definiti nella stessa tavola di conversione

❑ Connessioni di I/O:

Oggetto	Massimo	Note
Schede di IO	256	Definite nella stessa applicazione (schede o apparecchiature complesse)
Canali di IO	128	Sulla stessa scheda

❑ Per le librerie:

Oggetto	Massimo	Note
Funzioni (linguaggi IEC)	255	Installati assieme nella libreria
Blocchi funzione (linguaggi IEC)	255	Installati assieme nella libreria
Funzioni «C»	255	Installate assieme nella libreria
Blocchi funzione «C»	255	Installati assieme nella libreria
Istanze di blocchi funzione	4095	Per lo stesso tipo di blocco funzione nella stessa applicazione
Parametri di ingresso delle funzioni	31	Si applica a funzioni «C» e a funzioni scritte in linguaggi IEC

Parametri di blocchi funzione	32	Liberamente distribuiti tra parametri di ingresso e di uscita. E' richiesto minimo un parametro di uscita.
Funzioni di conversione	128	Installate insieme nella libreria
Configurazioni di IO	255	Installate insieme nella libreria
Schede di IO	255	Installate insieme nella libreria
Apparecchiature complesse di IO.	255	Installate insieme nella libreria
Parametri di schede OEM	16	

B. Guida di riferimento ai linguaggi

B.1 Architettura del progetto

Un progetto ISaGRAF è composto da più unità di programmazione chiamate **programmi.** I programmi di un progetto sono legati assieme con un'architettura di tipo ad albero. I programmi possono essere scritti con i linguaggi **SFC, FC (Flow Chart)**, **FBD**, **LD, ST** o **IL** sia in modo grafico che testuale.

B.1.1 Programmi

Un programma è costituito da un'unità logica di programmazione, che descrive le operazioni tra le **variabili** di un processo. Possono essere rappresentate sia operazioni **sequenziali** che **cicliche**. I programmi di tipo ciclico sono eseguiti ad ogni ciclo del sistema. L'esecuzione di programmi di tipo sequenziale si attengono alle regole del linguaggio **SFC** o del linguaggio **FC.**

I programmi sono legati assieme in una struttura del tipo ad albero gerarchico. I programmi principali sono posti in cima all'albero gerarchico e sono attivati dal sistema. I sottoprogrammi e i programmi figli (i livelli inferiori della struttura ad albero) vengono attivati dai rispettivi programmi padre. Un programma può essere scritto usando i seguenti linguaggi:

Sequential Function Chart (SFC) per operazioni di livello alto
Flow Chart (FC) per operazioni di livello alto
Function Block Diagram (FBD) per operazioni cicliche complesse
Ladder Diagram (LD) solamente per operazioni di tipo booleano
Structured Text (ST) per qualsiasi operazione di tipo ciclico
Instruction List (IL) per operazioni di livello basso

In uno stesso programma non possono coesistere linguaggi diversi, ad eccezione dei linguaggi LD e FBD che possono essere combinati assieme su uno stesso diagramma.

B.1.2 Operazioni di tipo ciclico e sequenziale

La gerarchia dei programmi consta di quattro principali **sezioni** di gruppi:

Sezione Iniziale	programmi che vengono eseguiti all'inizio di ciascun ciclo
Sezione Sequenziale	programmi che rispettano le regole della dinamica SFC o FC
Sezione Finale	programmi eseguiti al termine di ogni ciclo
Sezione Funzioni	insieme di sottoprogrammi non dedicati

I programmi della sezione **Iniziale** o **Finale** servono a rappresentare operazioni di tipo ciclico e non dipendono dal tempo. I programmi della sezione **Sequenziale** rappresentano operazioni di tipo sequenziale in cui la variabile tempo sincronizza esplicitamente le operazioni di base.
I programmi principali della sezione **Iniziale** vengono eseguiti sistematicamente all'inizio di ogni ciclo di esecuzione. I programmi principali della sezione **Finale** vengono eseguiti sistematicamente al termine di ciascun ciclo di esecuzione. I programmi principali della sezione **Sequenziale** vengono eseguiti secondo le regole della dinamica **SFC** o **FC**.

I programmi delle sezioni di tipo ciclico (**Iniziale** ed **Finale**) non possono venire scritti in linguaggio **SFC** o **FC**.

I programmi principali della sezione **Sequenziale** sono obbligatoriamente scritti in linguaggio **SFC** o **FC**, così come i loro programmi figli sono obbligatoriamente scritti rispettivamente in linguaggio **SFC** e **FC** (a differenza dei sottoprogrammi che non possono essere scritti in linguaggio **SFC** o **FC**).

I programmi della sezione **Funzioni** sono sottoprogrammi che possono essere chiamati da un qualsiasi altro programma (sia un padre, che un figlio) del progetto. Un programma della sezione **Funzioni** può chiamare un altro programma della medesima sezione **Funzioni**.
I sottoprogrammi (sezione **Funzioni**) non possono venire scritti in linguaggio **SFC** o **FC**.

Qualsiasi programma di qualsiasi sezione può avere uno o più sottoprogrammi, che obbligatoriamente non devono essere scritti in linguaggio **SFC** o **FC.**

I programmi della sezione **Iniziale** vengono tipicamente usati per descrivere operazioni preliminari su dispositivi di ingresso per costruire variabili filtrate ad alto livello. Queste variabili vengono usate frequentemente dai programmi della sezione **Sequenziale**. I programmi della sezione **Finale**, normalmente, eseguono controlli di sicurezza sulle variabili elaborate dalla sezione **Sequenziale**, prima dell'invio dei loro valori ai dispositivi di output.

B.1.3 Programmi figli SFC e FC

Ogni programma **SFC** della sezione **Sequenziale** può controllare altri programmi **SFC**. Tali programmi di livello inferiore vengono chiamati **programmi figli SFC.** Un **programma figlio SFC** è un programma **parallelo** (entrambi padre e figlio vengono eseguiti in parallelo), che può essere avviato, terminato, sospeso o riavviato dal proprio programma padre. Sia il programma padre che il figlio devono essere entrambi scritti in linguaggio **SFC**. Un programma SFC figlio può contenere variabili e nomi definiti con campo di validità locale.

Un programma genitore che **avvia** un programma **SFC** figlio, inserisce un **token** SFC in ciascun passo iniziale del programma figlio. Un programma genitore che **termina** un programma **SFC** figlio, rimuove tutti i token esistenti nei **passi** del programma figlio.
L'avvio di un programma figlio avviene con l'istruzione **GSTART**, la terminazione di un programma figlio avviene con l'istruzione **GKILL.**

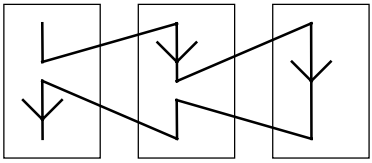
Quando un programma genitore **sospende** un programma figlio **SFC**, rimuove da esso tutti i token esistenti, memorizzandone la posizione. Quando un programma genitore **riavvia** un programma figlio **SFC** sospeso, ripristina tutti i token rimossi al momento della sospensione del programma figlio.
La sospensione di un programma figlio avviene con l'istruzione **GFREEZE**, il riavvio di un programma figlio avviene con l'istruzione **GRST**.

Per quanto riguarda la gestione dei sottoprogrammi il linguaggio **FC** (Flow Chart) si differenzia dal linguaggio **SFC** (Sequential Function Chart).
Ogni programma **FC** della sezione **Sequenziale** può controllare un altro sottoprogramma **FC**. Durante l'esecuzione di un sottoprogramma **FC,** il programma **FC** padre viene bloccato (in

stato di attesa). Non è possibile eseguire operazioni simultanee in un programma **FC** padre ed in uno dei suoi sottoprogrammi **FC**.

B.1.4 Funzioni e sottoprogrammi

L'esecuzione di un sottoprogramma o di una funzione viene guidata dal relativo programma padre.
A differenza del caso di un programma figlio attivato da un programma padre, nel caso di chiamata a un sottoprogramma o ad una funzione, l'esecuzione del programma padre viene sospesa fino al termine del sottoprogramma o della funzione:



Ogni programma di ogni sezione può avere uno o più sottoprogrammi. Un sottoprogramma è una funzione dedicata ad un solo programma padre. Un sottoprogramma può avere variabili e definizioni locali. Un sottoprogramma può essere scritto con qualunque linguaggio, ad eccezione del linguaggio **SFC** o **FC** (tali linguaggi possono essere usati per descrivere un programma figlio che, a differenza di un sottoprogramma, può risiedere solo nella sezione Sequenziale).
I programmi della sezione **Funzioni** sono sottoprogrammi che possono essere richiamati da qualunque altro programma del progetto. A differenza degli altri sottoprogrammi, non sono dedicati ad un unico programma padre. Un programma della sezione **Funzioni** può chiamare un altro programma della stessa sezione (ma non se stesso). Una funzione può risiedere nella libreria.

Avvertenza: Il sistema ISaGRAF non supporta la **ricorsività** delle chiamate a funzioni. Si verificherà un errore in esecuzione nel caso in cui un programma della sezione **Funzioni** venga chiamato da se stesso o da uno dei sottoprogrammi da esso chiamati.
Avvertenza: Una funzione o un sottoprogramma (funzione dedicata ad un padre) non «memorizza» i valori delle proprie variabili locali. Una funzione o un sottoprogramma non viene istanziato e non può, quindi, richiamare blocchi funzione.

L'interfaccia di una funzione o di un sottoprogramma deve essere definita esplicitamente usando un unico nome e un unico tipo di dato per ogni parametro di ingresso o per il parametro di ritorno.
Una finzione può avere un unico parametro di ritorno e complessivamente 32 parametri di chiamata o ritorno. Per compatibilità con il linguaggio **ST**, il parametro di ritorno, deve avere lo stesso nome del sottoprogramma.

La tavola seguente mostra come impostare il valore del parametro di ritorno all'interno di una funzione o di un sottoprogramma, con i differenti linguaggi:

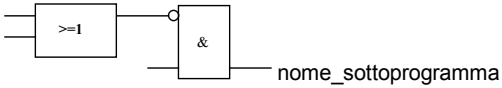
ST: Al parametro di ritorno viene assegnato il proprio nome (lo stesso nome del sotto programma):

```
nome_sottoprogramma := <espressione>;
```

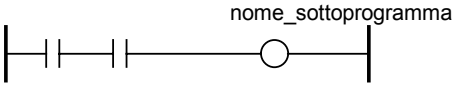
IL: Il parametro di ritorno è il valore del risultato corrente (registro IL), al termine della sequenza:

```
LD   10
ADD  20  (* valore del parametro di ritorno = 30 *)
```

FBD: Il parametro di ritorno viene impostato usando il proprio nome:

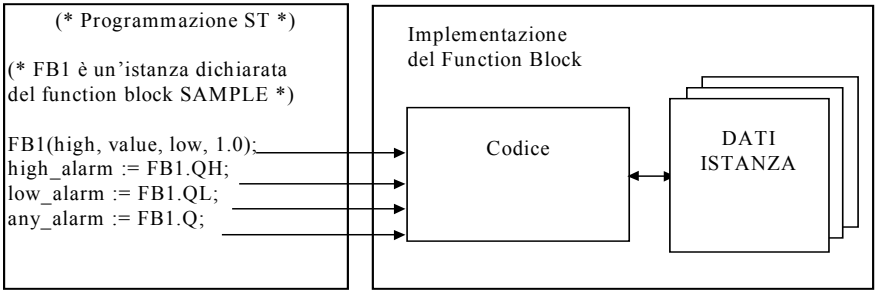


LD: Si utilizza il simbolo di bobina con il nome del parametro di ritorno:



B.1.5 Blocchi funzione

Un blocco funzione può essere scritto in linguaggio: LD, FBD, ST o IL. Un blocco funzione può avere più parametri di ritorno e complessivamente 32 parametri di chiamata e ritorno.
Un blocco funzione viene istanziato. Ciò significa che viene eseguita una copia delle variabili locali di un blocco funzione per ogni istanza: viene richiamato lo stesso codice, ma i dati usati sono quelli allocati per l'istanza. I valori delle variabili dell'istanza vengono memorizzati tra un ciclo e l'altro.



Avvertenza:
- Un blocco funzione scritto in uno dei linguaggi IEC **non** può chiamare altri blocchi funzione: il meccanismo che regola le istanze gestisce solamente le variabili locali del blocco stesso.

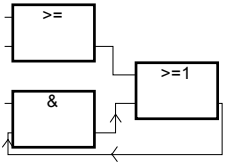
Di seguito viene riportata una lista di blocchi funzione standard che non possono essere richiamate da un blocco funzione IEC:
SR, RS, R_Trig, F_Trig, SEMA, CTU, CTD, CTUD, TON, TOF, TP, CMP, StackInt, AVERAGE, HYSTER, LIM_ALRM, INTEGRAL, DERIVATE, BLINK, SIG_GEN

- Per la stessa ragione, non è possibile usare contatti o bobine di tipo Positivo o Negativo, o bobine di tipo set o reset.

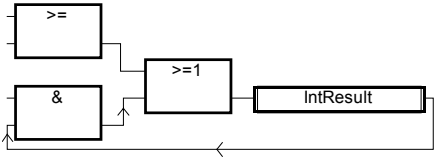
- Le funzioni TSTART e TSTOP, che provvedono ad avviare e fermare i timer, non possono essere usate in un blocco funzione di un target versione 3.0x. Funzionano a partire dalla versione 3.20.

- Volendo usare un loop in un blocco funzione, è necessario dichiarare la variabile locale prima. Si veda il seguente esempio:

Questo non funziona:



Questo va bene:



B.1.6 Descrizione del linguaggio

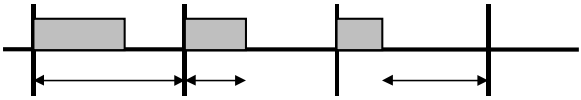
Un programma può essere scritto usando uno qualsiasi dei seguenti linguaggi grafici o testuali:

- **Sequential Function Chart** (SFC) per operazioni di alto livello
- **Flow Chart** (FC) per operazioni di alto livello
- **Function Block Diagram** (FBD) per operazioni cicliche complesse
- **Ladder Diagram** (LD) solamente per operazioni di tipo booleano
- **Structured Text** (ST) per qualsiasi operazione di tipo ciclico
- **Instruction List** (IL) per operazioni di livello basso

In uno stesso programma non possono coesistere linguaggi diversi. Il linguaggio viene scelto alla creazione del programma e non può più essere cambiato. Fanno eccezione i linguaggi LD e FBD, che possono essere combinati assieme in uno stesso programma.

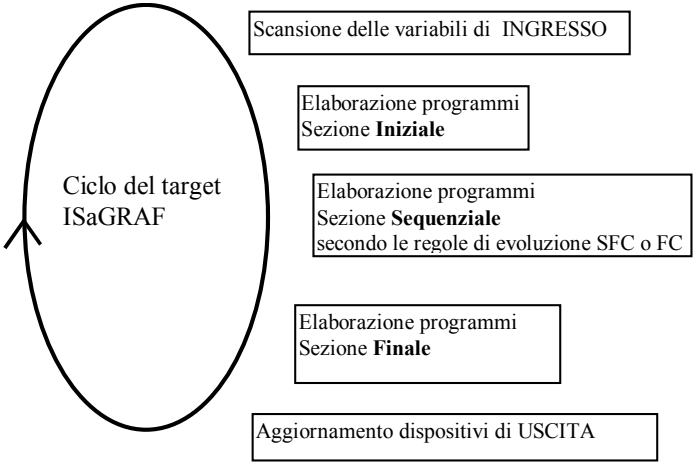
B.1.7 Regole di esecuzione

Il sistema ISaGRAF è **sincrono**. Tutte le operazioni eseguite sul PLC target sono comandate da un impulso di clock. La durata base del clock, viene chiamata **Tempo di ciclo**:



Tempo di ciclo Programmato | Tempo di ciclo Usato | Tempo di ciclo Libero

Le operazioni di base eseguite durante un singolo ciclo del taget:



Il sistema provvede a:

- garantire che una variabile di ingresso mantenga lo stesso valore all'interno di un ciclo,
- garantire che un dispositivo di uscita non venga aggiornato più di una volta per ciclo,
- operare in modo sicuro sulla stessa variabile globale usata in programmi diversi,
- stimare e controllare il tempo di ciclo utilizzato da un'applicazione completa.

B.2 Oggetti comuni

Vengono qui illustrate le caratteristiche principali e gli **oggetti** comuni del data base di programmazione ISaGRAF. Tali oggetti possono essere usati in uno qualsiasi dei linguaggi **SFC**, **FC**, **FBD**, **LD**, **ST** o **IL**.

B.2.1 Tipi base

Qualsiasi costante, espressione o variabile usata in un programma (qualunque sia il linguaggio) deve essere caratterizzata da un tipo. È importante rispettare la coerenza tra tipi sia nella costruzione di diagrammi grafici, che nella programmazione con istruzioni testuali. Di seguito sono elencati i tipi base disponibili per gli oggetti di programmazione:

- **BOOLEANO**: valore logico («TRUE» «FALSE»)
- **ANALOGICO**: valore continuo intero o reale (floating)
- **TIMER**: valore temporale
- **MESSAGGIO**: stringa di caratteri

Nota: I timers contengono valori inferiori ad un giorno e non possono essere usati per memorizzare date.

B.2.2 Espressioni costanti

Le espressioni costanti si riferiscono ad un solo tipo. La stessa notazione non può essere usata per rappresentare espressioni costanti di tipo differente.

B.2.2.1 Espressioni costanti di tipo booleano

Esistono solo due espressioni costanti booleane:

- **TRUE** equivalente al valore intero 1
- **FALSE** equivalente al valore intero 0

Nelle parole chiave «TRUE» e «FALSE» non viene fatta distinzione tra maiuscole e minuscole.

B.2.2.2 Espressioni costanti di tipo analogico intero

Le espressioni costanti intere rappresentano valori interi lunghi con segno (32 bit):
da **-2147483647** a **+2147483647**
Le costanti intere di tipo analogico possono essere espresse con uno dei seguenti **tipi base**.
Le costanti intere devono iniziare con un **prefisso** che identifica il tipo base usato:

Base	Prefisso	Esempio
DECIMALE	**(nessuno)**	**-908**
ESADECIMALE	**"16#"**	**16#1A2B3C4D**

OTTALE	**"8#"**	**8#1756402**
BINARIO	**"2#"**	**2#1101_0001_0101_1101**

Il carattere underscore «_» permette di separare gruppi di cifre. Non ha alcun significato particolare, ma migliora la leggibilità delle espressioni costanti.

B.2.2.3 Espressioni costanti di tipo analogico Reale

Le espressioni costanti di tipo reale analogico possono essere scritte usando la rappresentazione **decimale** o **scientifica**. Il **punto decimale** ('**.**') separa la parte intera dalla parte decimale. Nella rappresentazione scientifica la **mantissa** e **l'esponente** sono separate dalle lettere '**E**' o '**F**'. L'esponente di un'espressione in notazione scientifica deve essere un valore di tipo intero con segno compreso tra **-37** e **+37**. Di seguito sono riportati alcuni esempi di espressioni costanti di tipo analogico reale:

3.14159	**-1.0E+12**
+1.0	**1.0F-15**
-789.56	**+1.0E-37**

L'espressione "**123**" non è di tipo reale. La corretta rappresentazione di tipo reale è "**123.0**".

B.2.2.4 Espressioni costanti di tipo Timer

Le espressioni costanti di tipo Timer rappresentano valori temporali compresi tra **0 secondi** e **23h59m59s999ms**. La più piccola entità rappresentabile è il millisecondo. Alcune unità temporali standard usate in espressioni costanti sono:

- **Hour** Lettera "**h**" posta dopo il numero di ore
- **Minute** Lettera "**m**" posta dopo il numero di minuti
- **Second** Lettera "**s**" posta dopo il numero di secondi
- **Millisecond** Lettere "**ms**" poste dopo il numero di millisecondi

Le espressioni costanti di tipo timer hanno il prefisso "**T#**" o "**TIME#**". Non viene fatta distinzione tra maiuscole e minuscole nei prefissi e nelle lettere. Seguono alcuni esempi di espressioni costanti di tipo timer:

T#1H450MS	1 ora, 450 millisecondi
time#1H3M	1 ora, 3 minuti

L'espressione "0" non rappresenta un valore temporale, ma una costante di tipo analogico.

B.2.2.5 Espressioni costanti di tipo stringa messaggio

Le espressioni costanti di tipo stringa o messaggio rappresentano stringhe di caratteri. I caratteri devono essere preceduti da un **apice** e seguiti da un **apostrofo**. Ad esempio:

'ECCO UN MESSAGGIO'

Avvertenza: Non è possibile usare il carattere apostrofo «**'**» all'interno di una espressione costante di tipo stringa. Un'espressione costante di tipo stringa deve essere contenuta in una sola riga del codice sorgente del programma. La lunghezza complessiva non deve eccedere i 255 caratteri, spazi compresi.

L'espressione costante stringa vuota viene rappresentata da due apostrofi che non contengono il carattere spazio o tabulazione.:

'' (* questa è una stringa vuota *)

Il carattere speciale dollaro ('**$**'), seguito da altri caratteri speciali, si usa all'interno di una stringa per rappresentare i caratteri non stampabili:

Sequenze	Significato	ASCII (esa)	Esempio
$$	carattere '$'	16#24	'Mi è costato $$5'
$'	apostrofo	16#27	'Inserire $'Y$' per YES'
$L	line feed	16#0a	'riga $L seguente'
$R	carriage return (ritorno carrello)	16#0d	' ao $R Ci'
$N	nuova riga	16#0d0a	'Questa è una riga $N'
$P	nuova pagina	16#0c	'ultima riga $P prima riga
$T	tabulazione	16#09	'nome$Tdim$Tdata'
$hh (*)	qualsiasi carattere	16#hh	'ABCD = $41$42$43$44'

(*) "**hh**" rappresenta il valore esadecimale del codice ASCII del carattere da esprimere.

B.2.3 Variabili

Le variabili possono avere campo di validità **LOCALE** o **GLOBALE**. Le variabili locali possono essere usate da un solo programma. Le variabili globali vengono viste da ogni programma del progetto. I nomi delle variabili devono rispettare le seguenti regole:

- Il nome non può eccedere i **16** caratteri
- il primo carattere deve essere una **lettera**
- i caratteri seguenti possono essere **lettere**, **cifre** o il carattere «_»

B.2.3.1 Parole chiave riservate

Viene riportato sotto l'elenco delle parole chiave . Tali identificatori non possono essere usati come nome di programma, variabile, funzione «C» o blocco funzione:

A ANA, ABS, ACOS, ADD, ANA, AND, AND_MASK, ANDN, ARRAY, ASIN, AT, ATAN,

B BCD_TO_BOOL, BCD_TO_INT, BCD_TO_REAL, BCD_TO_STRING, BCD_TO_TIME, BOO, BOOL, BOOL_TO_BCD, BOOL_TO_INT, BOOL_TO_REAL, BOOL_TO_STRING, BOOL_TO_TIME, BY, BYTE,

C	CAL, CALC, CALCN, CALN, CALNC, CASE, CONCAT, CONSTANT, COS,
D	DATE, DATE_AND_TIME, DELETE, DINT, DIV, DO, DT, DWORD,
E	ELSE, ELSIF, EN, END_CASE, END_FOR, END_FUNCTION, END_IF, END_PROGRAM, END_REPEAT, END_RESSOURCE, END_STRUCT, END_TYPE, END_VAR, END_WHILE, ENO, EQ, EXIT, EXP, EXPT,
F	FALSE, FEDGE, FIND, FOR, FUNCTION,
G	GE, GFREEZE, GKILL, GRST, GSTART, GSTATUS, GT,
I	IF, INSERT, INT, INT_TO_BCD, INT_TO_BOOL, INT_TO_REAL, INT_TO_STRING, INT_TO_TIME,
J	JMP, JMPC, JMPCN, JMPN, JMPNC,
L	LD, LDN, LE, LEFT, LEN, LIMIT, LINT, LN, LOG, LREAL, LT, LWORD,
M	MAX, MID, MIN, MOD, MOVE, MSG, MUL, MUX,
N	NE, NOT,
O	OF, ON, OPERATE, OR, OR_MASK, ORN,
P	PROGRAM
R	R, REDGE, READ_ONLY, READ_WRITE, REAL, REAL_TO_BCD, REAL_TO_BOOL, REAL_TO_INT, REAL_TO_STRING, REAL_TO_TIME, REDGE, REPEAT, REPLACE, RESSOURCE, RET, RETAIN, RETC, RETCN, RETN, RETNC, RETURN, RIGHT, ROL, ROR,
S	S, SEL, SHL, SHR, SIN, SINT, SQRT, ST, STN, STRING, STRING_TO_BCD, STRING_TO_BOOL, STRING_TO_INT, STRING_TO_REAL, STRING_TO_TIME, STRUCT, SUB, SYS_ERR_READ, SYS_ERR_TEST, SYS_INITALL, SYS_INITANA, SYS_INITBOO, SYS_INITTMR, SYS_RESTALL, SYS_RESTANA, SYS_RESTBOO, SYS_RESTTMR, SYS_SAVALL, SYS_SAVANA, SYS_SAVBOO, SYS_SAVTMR, SYS_TALLOWED, SYS_TCURRENT, SYS_TMAXIMUM, SYS_TOVERFLOW, SYS_TRESET, SYS_TWRITE, SYSTEM,
T	TAN, TASK, THEN, TIME, TIME_OF_DAY, TIME_TO_BCD, TIME_TO_BOOL, TIME_TO_INT, TIME_TO_REAL, TIME_TO_STRING, TMR, TO, TOD, TRUE, TSTART, TSTOP, TYPE,
U	UDINT, UINT, ULINT, UNTIL, USINT,
V	VAR, VAR_ACCESS, VAR_EXTERNAL, VAR_GLOBAL, VAR_IN_OUT, VAR_INPUT, ,VAR_OUTPUT,
W	WHILE, WITH, WORD,
X	XOR, XOR_MASK, XORN

Tutte le parole chiave che iniziano con il carattere underscore «_» sono ad uso interno e non devono essere usate in istruzioni testuali.

B.2.3.2 Variabili rappresentate direttamente.

ISaGRAF consente l'uso nel codice del programma di **variabili rappresentate direttamente** per identificare un canale libero di una scheda di I/O. Per canali liberi si intendono quelli che non sono associati ad una variabile di I/O. L'identificatore di una variabile rappresentata direttamente inizia sempre con il carattere "**%**"

Seguono le convenzioni per assegnare il nome ad una variabile rappresentata direttamente per un canale di una singola scheda. "**s**" è il numero dell'alloggiamento (slot) della scheda. "**c**" il numero del canale.

%IX*s.c* canale libero di una scheda di ingresso booleana

`%ID`***`s.c`*** canale libero di una scheda di ingresso intera
`%IS`***`s.c`*** canale libero di una scheda di ingresso di tipo messaggio
`%QX`***`s.c`*** canale libero di una scheda di uscita booleana
`%QD`***`s.c`*** canale libero di una scheda di uscita intera
`%QS`***`s.c`*** canale libero di una scheda di uscita di tipo messaggio

Seguono le convenzioni per assegnare il nome ad una variabile rappresentata direttamente per un canale di una apparecchiatura complessa. "**s**" è il numero dell'alloggiamento (slot) della scheda. "**b**" è l'indice della singola scheda all'interno dell'apparecchiatura. "**c**" il numero del canale.

`%IX`***`s.b.c`*** canale libero di una scheda di ingresso booleana
`%ID`***`s.b.c`*** canale libero di una scheda di ingresso intera
`%IS`***`s.b.c`*** canale libero di una scheda di ingresso di tipo messaggio
`%QX`***`s.b.c`*** canale libero di una scheda di uscita booleana
`%QD`***`s.b.c`*** canale libero di una scheda di uscita intera
`%QS`***`s.b.c`*** canale libero di una scheda di uscita di tipo messaggio

Seguono degli esempi:

`%QX1.6` 6° canale della scheda #1 (uscita booleano)
`%ID2.1.7` 7° canale della scheda #1 nell'apparecchiatura #2 (ingresso intero)

Una variabile rappresentata direttamente non può essere di tipo **reale**.

B.2.3.3 Variabili di tipo booleano

Booleano significa **logico**. Variabili di questo tipo possono assumere valori booleani: **TRUE** o **FALSE**. Le variabili booleane vengono tipicamente usate in espressioni booleane. Le variabili booleane possono avere uno dei seguenti **attributi**:

Interna: variabile gestita dal programma
Costante: variabile di sola lettura con un valore iniziale
Ingresso: variabile riferita ad un dispositivo di ingresso (aggiornata dal sistema)
Uscita: variabile riferita ad un dispositivo di uscita

Avvertenza: Dichiarando una variabile booleana, si possono definire delle stringhe sostitutive dei valori «TRUE» e «FALSE» in fase di debug. Tali stringhe sostitutive possono essere usate dai programmi solo se dichiarate come **nomi definiti** nel linguaggio.

B.2.3.4 Variabili di tipo analogico

Analogico significa **continuo**. Tali variabili sono di tipo intero con segno o reale (floating). I formati disponibili per una variabile di tipo analogico:

Intero intero con segno a 32 bit da **-2147483647** a **+2147483647**
Reale valore IEEE standard 32 bit in virgola mobile (precisione singola)
1 bit di segno + 23 bit mantissa + 8 bit di esponente

Il valore dell'esponente reale analogico deve essere compreso tra **-37** e **+37**. Le variabili di tipo analogico possono avere uno dei seguenti **attributi**:

Interna: variabile gestita dal programma
Costante: variabile di sola lettura con un valore iniziale
Ingresso: variabile riferita ad un dispositivo di ingresso (aggiornata dal sistema)
Uscita: variabile riferita ad un dispositivo di uscita

Nota: Quando una variabile di tipo reale è connessa ad un dispositivo di I/O, il driver I/O agisce sul relativo valore intero.

Avvertenza: Variabili analogiche di tipo intero o reale non possono coesistere nella stessa espressione analogica con espressioni costanti.

B.2.3.5 Variabili di tipo Timer

Timer equivale ad **orologio** o **contatore**. Tali variabili contengono valori temporali e vengono tipicamente usate in espressioni di tipo timer. Un valore temporale non può eccedere **23h59m59s99** e non può essere negativo. Le variabili di tipo timer vengono memorizzate in word a 32 bit. La rappresentazione interna consiste in un numero positivo di millisecondi.
Le variabili di tipo timer possono avere i seguenti **attributi**:

Interna: variabile gestita dal programma, aggiornata dal sistema ISaGRAF
Costante: variabile di sola lettura con un valore iniziale

Avvertenza: Le variabili di tipo timer non hanno gli attributi Ingresso o Uscita.

Le variabili di tipo timer vengono aggiornate automaticamente dal sistema ISaGRAF. Quando un timer è **attivo**, il suo valore viene automaticamente incrementato seguendo l'orologio in tempo reale del sistema target. Per controllare un timer si usano le seguenti istruzioni **ST**:

TSTART avvia l'aggiornamento automatico di un timer
TSTOP ferma l'aggiornamento automatico di un timer

B.2.3.6 Variabili di tipo messaggio stringa

Le variabili di tipo messaggio o stringa contengono stringhe di caratteri. La lunghezza della stringa può cambiare in fase di esecuzione. La lunghezza di una variabile messaggio non può eccedere la capacità (la lunghezza massima) specificata al momento della dichiarazione della variabile. La capacità di un messaggio è limitata a 255 caratteri. Le variabili di tipo messaggio possono avere uno dei seguenti **attributi**:

Interna: variabile gestita dal programma
Costante: variabile di sola lettura con un valore iniziale
Ingresso: variabile riferita ad un dispositivo di input (aggiornata dal sistema)
Uscita: variabile riferita ad un dispositivo di output

Le variabili di tipo stringa possono contenere qualsiasi carattere della tabella standard ASCII. (codici ASCII compresi tra **0** e **255**). Una stringa di caratteri può contenere Il carattere null.

Alcune funzioni «C» della libreria standard ISaGRAF, tuttavia, non tratteranno correttamente i messaggi contenenti caratteri null (**0**).

B.2.4 Commenti

Nei linguaggi testuali come **ST** o **IL** si possono liberamente inserire dei commenti. Un commento deve iniziare con i caratteri speciali "**(***" e terminare con i caratteri "***)**". I commenti possono essere inseriti dovunque in un programma **ST**, e possono essere scritti su più di una riga.

Alcuni esempi di commenti:

```
counter := ivalue;   (* assegna il contatore principale *)
(* questo commento occupa
due righe *)
c := counter (* i commenti possono essere inseriti ovunque *) + base_value + 1;
```

Non sono possibili commenti annidati. In pratica i caratteri "**(***" non possono comparire all'interno di un commento.

Avvertenza: Il linguaggio **IL** accetta commenti solamente se inseriti al termine di una riga di istruzione..

B.2.5 Nomi definiti

Il sistema ISaGRAF permette di ridefinire espressioni costanti, espressioni booleane «TRUE» e «FALSE», parole chiave o espressioni **ST** articolate. Per far questo, si deve assegnare il nome di un **identificatore** all'espressione. Ad esempio:

YES	equivale a	**TRUE**
PI	equivale a	**3.14159**
OK	equivale a	**(auto_mode AND NOT (alarm))**

Una volta definita questa equivalenza, il relativo **identificatore** può essere usato in qualsiasi punto del programma **ST** per sostituire l'espressione abbinata. Segue un esempio di programmazione in linguaggio **ST** con l'uso delle definizioni:

```
If OK Then
  angolo := PI / 2.0;
  fatto := YES;
End_if;
```

I nomi definiti possono essere **LOCALI** ad un programma, **GLOBALI** o **COMUNI**. I nomi definiti Locali possono essere usati da un solo programma. I nomi definiti Globali possono essere usate da qualsiasi programma di un progetto. I nomi definiti Comuni possono essere usati da qualsiasi programma di qualsiasi progetto.
Da notare che le definizioni COMUNI possono essere archiviate separatamente usando la gestione di archivio (comando «**Strumenti / Archivio / Dati comuni**» dalla finestra Gestione progetti).

Avvertenza: Se lo stesso identificatore viene definito due volte con differenti equivalenze **ST**, viene usata l'ultima espressione definita. Ad esempio:

Definendo:	**OPEN**	equivalente a	**FALSE**
	OPEN	equivalente a	**TRUE**
si avrà:	**OPEN**	equivalente a	**TRUE**

L'assegnamento del nome ai nomi definiti deve rispettare le seguenti direttive:

- il nome non può eccedere i **16** caratteri
- il primo carattere deve essere una **lettera**
- i successivi caratteri possono essere **lettere**, **cifre** o il carattere «_»

Avvertenza: Un Nome definito non può far uso a sua volta di un altro Nome definito, ad esempio, non è possibile:

PI	equivale a	**3.14159**
~~**PI2**~~	~~equivale a~~	**PI*2**

bisogna scrivere l'equivalenza completa usando costanti, variabili ed operatori:

PI2	equivale a	**6.28318**

B.3 Linguaggio SFC

Il linguaggio **grafico** Sequential Function Chart (SFC) permette di rappresentare **operazioni sequenziali**. Un processo viene rappresentato da un insieme di ben definiti **passi**, collegati tra loro da **transizioni**. Ad ogni transizione è abbinata una **condizione booleana**. All'interno dei passi vengono inserite delle **Azioni** utilizzando gli altri linguaggi di programmazione (**ST**, **IL**, **LD** e **FDB**).

B.3.1 Formato di un diagramma SFC

Un programma SFC è un insieme grafico di **passi** e **transizioni**, connesse tra loro da **collegamenti orientati**. Divergenze e convergenze vengono rappresentate da collegamenti multipli. Alcune porzioni del diagramma possono essere disegnate separatamente e rappresentate nel diagramma principale con simboli chiamati **macro passi**. Le **direttive base** SFC sono:

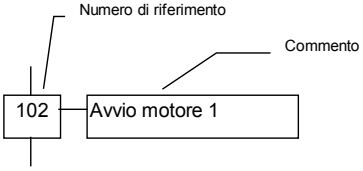
- Un passo non può essere seguito da un altro passo
- Una transizione non può essere seguita da un'altra transizione

B.3.2 Componenti di base SFC

I componenti di base (simboli grafici) del linguaggio SFC sono passi, passi iniziali, transizioni, collegamenti orientati e salti ad un passo.

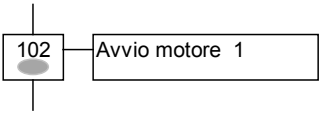
B.3.2.1 Passi e passi iniziali

Un passo è rappresentato da un singolo **riquadro**. Per ogni passo esiste un numero di **riferimento** rappresentato all'interno di un simbolo quadrato. La descrizione del passo appare invece in una casella collegata al simbolo di passo. Questa descrizione consiste in un **commento libero** (che non fa parte del linguaggio di programmazione). L'informazione di cui sopra, viene chiamata **Livello 1** del passo:

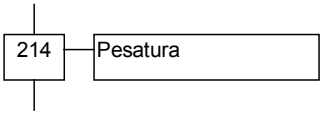


In fase di esecuzione, un **token (segnale)** indica che il passo è **attivo** :

Passo attivo:

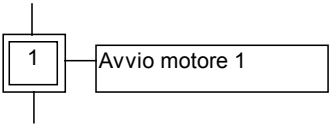


Passo non attivo:



La **situazione iniziale** di un programma SFC viene introdotta con un **passo iniziale**. Un passo iniziale è rappresentato da un simbolo grafico dotato di **doppio bordo**. All'avvio del programma, in ciascun passo iniziale, viene automaticamente inserito un token.

Passo iniziale:



Un programma SFC deve contenere **almeno un** passo iniziale.

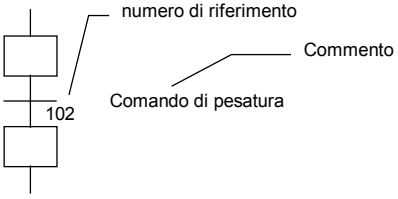
Queste sono le proprietà di un passo. Questi campi si possono usare anche con gli altri linguaggi:

GSnnn.x..... stato di attività del passo (valore booleano)
GSnnn.t..... durata dell'attività del passo (valore temporale)

(con **nnn** che rappresenta il numero di riferimento del passo)

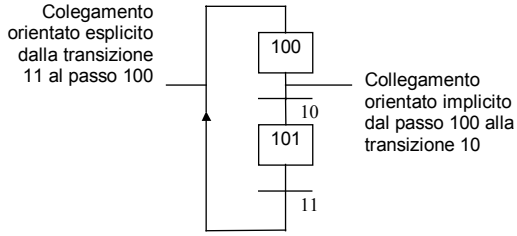
B.3.2.2 Transizioni

Le transizioni sono rappresentate da piccole barre orizzontali che attraversano i collegamenti. Ad ogni transizione è associato un numero di **riferimento** , visibile in prossimità del simbolo di transizione. La descrizione della transizione è posta alla destra del simbolo di transizione. Questa descrizione consiste in un **commento libero** (che non fa parte del linguaggio di programmazione). L'informazione di cui sopra viene chiamata **Livello 1** della transizione:



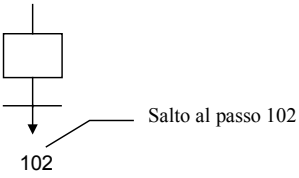
B.3.2.3 Collegamenti orientati

Passi e transizioni sono collegati da singole linee . Si tratta di collegamenti orientati. Se l'orientamento non viene esplicitamente specificato, il collegamento viene orientato dall'alto verso il basso.

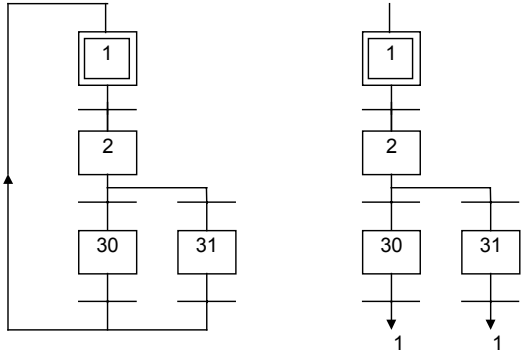


B.3.2.4 Salto ad un passo

I simboli di salto possono essere usati per indicare un collegamento da una transizione ad un passo, senza dover tracciare la linea di connessione. Al simbolo di salto deve far riferimento il numero del passo destinazione.



Un simbolo di salto non può essere usato per rappresentare il collegamento da un passo ad una transizione.
Esempio di salti – i seguenti diagrammi sono equivalenti:

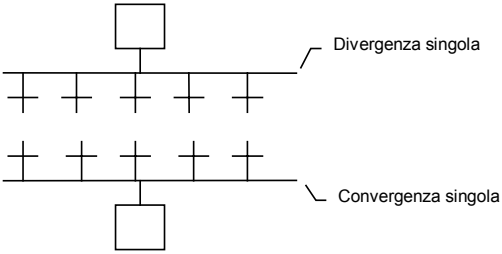


B.3.3 Divergenze e convergenze

Le divergenze vengono rappresentate con **collegamenti multipli** da un simbolo SFC (passo o transizione) verso molti altri simboli SFC. Le convergenze vengono rappresentate con collegamenti multipli da più simboli SFC verso un altro singolo simbolo.

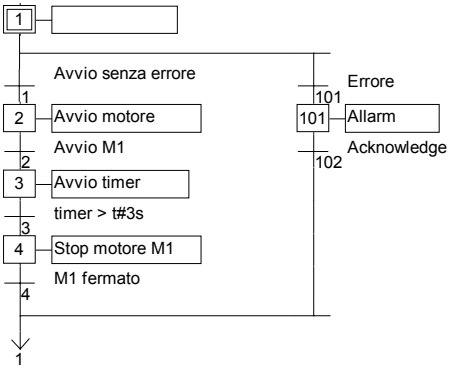
B.3.3.1 Divergenze singole

Una singola divergenza è un collegamento multiplo tra un solo passo e molte transizioni. Permette al segnale attivo di passare in una di un certo numero di diramazioni. Una convergenza singola è un collegamento multiplo da molte transizioni ad uno stesso passo. Una convergenza singola viene generalmente usata per raggruppare le diramazioni SFC che si dipartono da una divergenza singola. Divergenze e convergenze singole vengono rappresentate da linee orizzontali semplici.



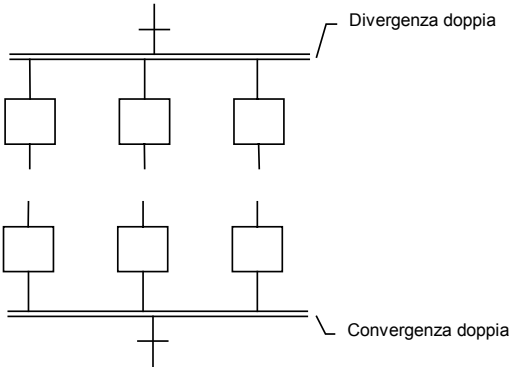
Avvertenza: le condizioni abbinate a differenti transizioni all'inizio di una divergenza singola **non sono implicitamente esclusive**. La condizione di esclusività deve essere esplicitamente indicata nelle condizioni delle transizioni per assicurare che, in fase di esecuzione, un solo segnale avanzi in una diramazione della divergenza. Sotto viene riportato un esempio di una divergenza e convergenza singola:

(* programma SFC con divergenza e convergenza singola *)



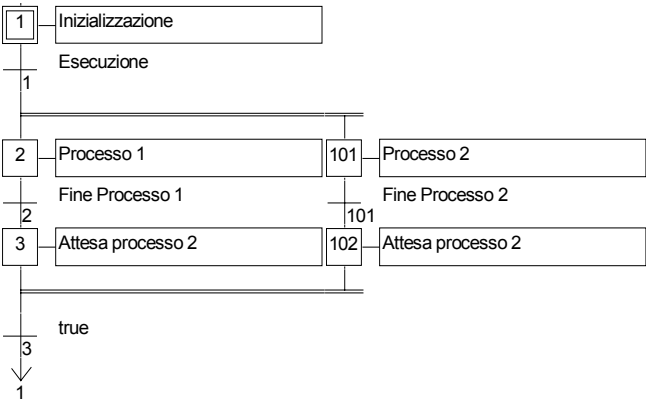
B.3.3.2 Divergenze doppie

Una divergenza doppia è costituita da un collegamento multiplo da una transizione a molti passi. Rappresenta le operazioni in parallelo del processo. Una convergenza doppia è costituita da un collegamento multiplo da molteplici passi ad una stessa transizione. La convergenza doppia si usa, normalmente, per raggruppare le diramazioni SFC che si dipartono da una divergenza doppia. Divergenze e convergenze doppie vengono rappresentate da linee orizzontali doppie.



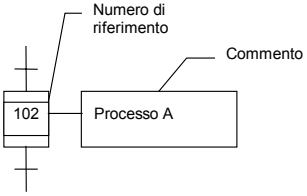
Esempio di una divergenza e convergenza doppie:

(* programma SFC con divergenza e convergenza doppie *)



B.3.4 Macro passi

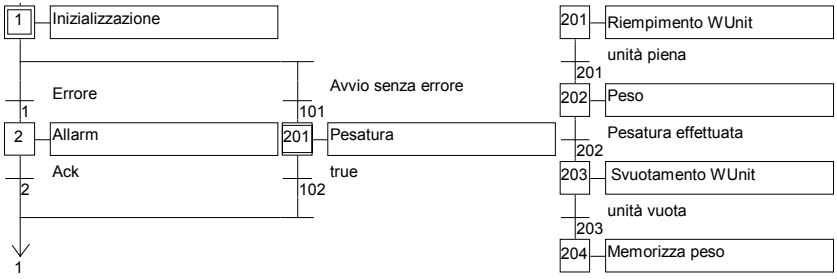
Un macro passo è la rappresentazione unica di un gruppo unico di passi e transizioni. Il corpo di un macro passo viene descritto in un punto diverso dello stesso programma SFC. Nel diagramma SFC principale appare come un singolo simbolo. Questo è la rappresentazione di un macro passo:



Il numero di riferimento visibile nel simbolo del macro passo è il numero di riferimento del primo passo nel corpo del macro passo. Il corpo di un macro passo deve aver inizio con un **passo iniziale di macro** e deve terminare con un **passo finale di macro**. Il diagramma non deve avere riferimenti esterni. Un passo iniziale di macro non ha collegamenti superiori (nessuna transizione precedente). Un passo finale di macro non ha collegamenti inferiori (nessuna transizione successiva). Il simbolo di un macro passo può essere inserito nel corpo di un altro macro passo.

Avvertenza: Poiché un macro passo costituisce un **unico** insieme di passi e transizioni, lo stesso macro passo non può essere usato più di una volta in un programma SFC.

Esempio di macro passo:
(* Programma SFC con un macro passo *)
(* Diagramma principale *) **(* Corpo di un macro passo *)**



B.3.5 Azioni all'interno di un passo

La programmazione di **livello 2** di un passo SFC è l'elenco dettagliato delle **azioni** eseguite **quando il passo è attivo**, e viene inserito in modalità **SFC testuale**, con la possibilità di usare anche altri linguaggi come ad esempio Structured Text (**ST**). I tipi base delle azioni sono:

- Azioni di tipo booleano

- Azioni di tipo impulsivo programmate in ST
- Azioni di tipo non-memorizzato programmate in ST
- Azioni SFC

Uno stesso passo può contenere più azioni (di tipo uguale o differente). Le situazioni particolari che permettono l'uso di altri linguaggi sono:

- Chiamate a sottoprogrammi
- Convenzioni del linguaggio Instruction List (IL)

B.3.5.1 Azioni booleane

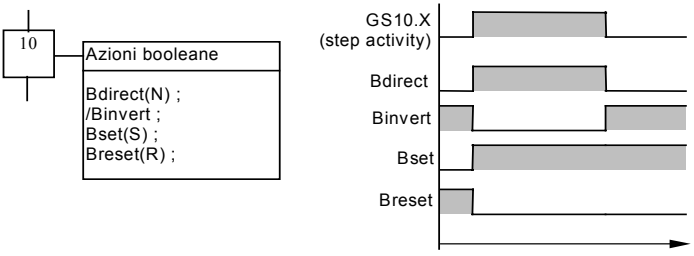
Le azioni booleane assegnano ad una variabile booleana lo stato di attività del passo. La variabile booleana può avere attributo di variabile Interna o di Uscita. Il valore della variabile viene aggiornato ogni volta che viene avviata o fermata l'attività del passo. Questa è la sintassi delle azioni booleane di base:

<variabile_booleana> (N) ;	assegna alla variabile lo stato di attività del passo
<variabile_booleana> ;	stesso effetto (l'attributo N è facoltativo)
/ <variabile_booleana> ;	assegna alla variabile l'inverso dello stato di attività del passo

Esistono altre possibilità di modificare il valore della variabile booleana all'attivazione del passo. Questa è la sintassi delle azioni booleane di tipo set e reset:

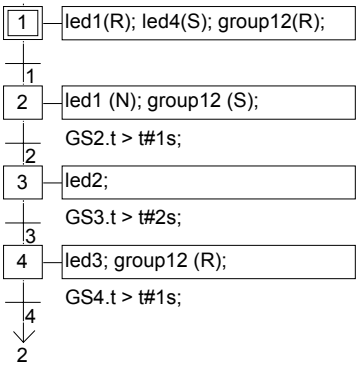
<variabile_booleana> (S) ;	imposta la variabile booleana a TRUE quando lo stato di attività del passo diventa TRUE
<variabile_booleana> (R) ;	imposta la variabile a FALSE quando lo stato di attività del passo diventa TRUE

La variabile booleana deve avere attributo di variabile di **USCITA** o **INTERNA**. Nell'esempio, il risultato della programmazione SFC:



Esempio di azioni booleane:

(* Programma SFC che usa azioni BOOLEANE *)

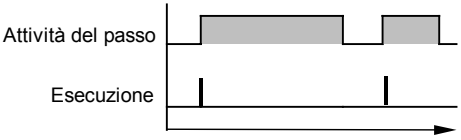


B.3.5.2 Azioni di tipo impulsivo

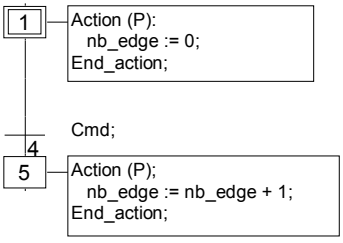
Un'azione di tipo impulsivo è costituita da un elenco di istruzioni ST o IL, che vengono eseguite solamente **una volta** al momento dell'**attivazione** del passo. Le istruzioni rispettano la seguente sintassi SFC:

```
ACTION (P) :
        (* istruzioni ST *)
END_ACTION ;
```

Di seguito vengono illustrati i risultati di un'azione di tipo impulsivo



Esempio di un'azione di tipo impulsivo:



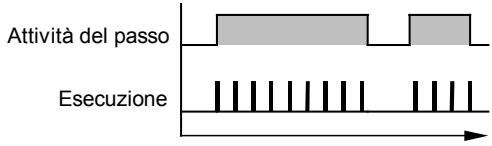
B.3.5.3 Azioni di tipo non-memorizzato

Un'azione di tipo non-memorizzato (normale) è costituita da un elenco di istruzioni ST o IL, che vengono eseguite **ad ogni ciclo** durante l'intero periodo di **attivazione** del passo. Le istruzioni rispettano la seguente sintassi SFC:

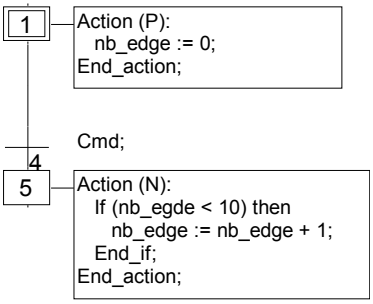
```
ACTION (N) :
        (* istruzioni ST*)
END_ACTION ;
```

Nota: non confondere ACTION(N) con un azione booleana Normale.

Di seguito vengono illustrati i risultati di un'azione di tipo non-memorizzata:



Esempio di un'azione di tipo non-memorizzato:



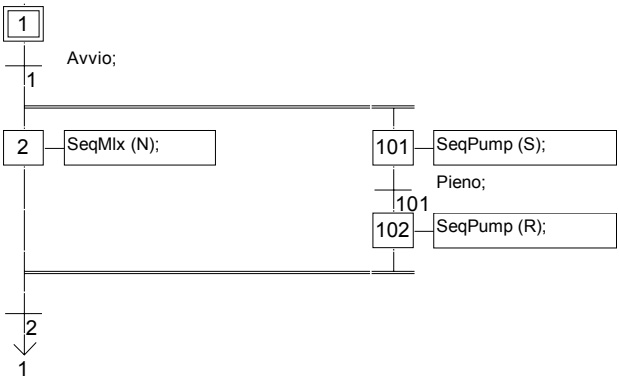
B.3.5.4 Azioni SFC

Un'azione SFC è costituita da un **programma figlio SFC**, avviato o terminato in seguito al mutare dello stato di attività del passo. Un'azione SFC può venire qualificata come **N** (Non-memorizzata), **S** (Set), o **R** (Reset). Questa è la sintassi delle azioni SFC di base:

<prog_figlio> (N);	avvia la sequenza figlia al momento dell'attivazione del passo, e termina la sequenza figlia quando il passo diventa inattivo
<prog_figlio> ;	stesso effetto (l'attributo N è facoltativo)
<prog_figlio> (S);	avvia la sequenza figlia al momento dell'attivazione del passo. – nessuna azione viene intrapresa quando il passo torna inattivo
<prog_figlio> (R);	termina la sequenza figlia quando il passo diventa inattivo

La sequenza SFC specificata come azione deve essere un **programma figlio SFC** del programma attualmente in fase di modifica. L'uso dei qualificatori **S** (Set) o **R** (Reset) in un'azione SFC, produce lo stesso effetto delle istruzioni di avvio e terminazione del programma figlio SFC programmate con il linguaggio ST in un'azione di tipo Impulsivo.
Segue sotto un esempio di azione SFC. Il programma principale SFC viene chiamato **padre**. Ha due figli SFC, chiamati **SeqMIx** (che controlla mescolatore) e **SeqPump** (che controlla una pompa). La programmazione SFC per il programma SFC padre è la seguente:

(* Programma SFC che usa un'azione SFC *)



B.3.5.5 Richiamare funzioni e blocchi funzione da un'azione

Sottoprogrammi, funzioni o blocchi funzione (scritti in linguaggio ST, IL, LD o FBD) o funzioni «C» e blocchi funzione "C", possono essere chiamati direttamente dall'interno di un'azione SFC, attenendosi alla seguente sintassi:

Per sottoprogrammi, funzioni e funzioni "C":

```
ACTION (P) :
        result := sub_program ( ) ;
END_ACTION;
```

o

```
ACTION (N) :
        result := sub_program ( ) ;
END_ACTION;
```

Per blocchi funzione in "C" o in ST, IL, LD, FBD:

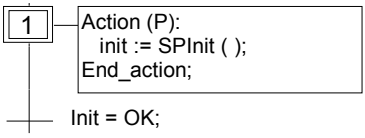
```
ACTION (P) :
        Fbinst(in1, in2);
        result1 := Fbinst.out1;
        result2 := Fbinst.out2;
END_ACTION;
```

o

```
ACTION (N) :
        Fbinst(in1, in2);
        result1 := Fbinst.out1;
        result2 := Fbinst.out2;
END_ACTION;
```

La sintassi dettagliata è disponibile nella sezione dedicata al linguaggio ST.
Esempio di una chiamata ad un sottoprogramma dall'interno di un'azione:

(* Programma SFC con sottoprogramma chiamato dall'interno di un'azione *)



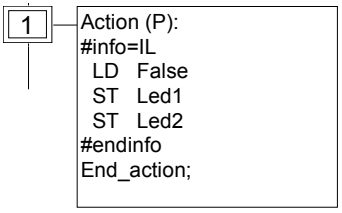
B.3.5.6 Convenzione IL

La programmazione di tipo Instruction List (IL) può essere introdotta direttamente in un'azione SFC rispettando la seguente sintassi:

```
ACTION (P) :            (* o N *)
#info=IL
    <istruzione>
    <istruzione>
    ....
#endinfo
END_ACTION;
```

Le parole chiave speciali "#info=IL" e "#endinfo" devono essere introdotte esattamente nel modo esposto, viene fatta **distinzione** tra lettere maiuscole e minuscole. I caratteri spazio o tabulazione non possono essere inseriti all'interno, prima o dopo le parole chiave. Segue un esempio di programma IL all'interno di in un'azione:

(* Programma SFC con una sequenza IL in un blocco di tipo Action *)



B.3.6 Condizioni abbinate alle transizioni

Ad ogni transizione è abbinata un'**espressione booleana** che regola l'attivazione della transizione. La condizione viene generalmente espressa con il linguaggio ST o con il linguaggio LD (editor Quick LD) e corrisponde alla programmazione di **Livello 2** della transizione. È, comunque, possibile usare altre strutture:

- Convenzioni del linguaggio ST
- Convenzioni del linguaggio LD
- Convenzioni del linguaggio IL
- Funzione chiamata da una transizione

Avvertenza: Qualora non ci siano espressioni abbinate alla transizione, la condizione predefinita è **TRUE**.

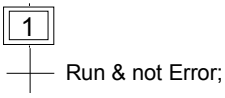
B.3.6.1 Convenzione ST

La **condizione** abbinata ad una transizione può essere scritta usando il linguaggio **Structured Text** (ST) L'espressione completa deve essere di tipo booleano e deve terminare con **punto e virgola**, rispettando la seguente sintassi:

< espressione_booleana > ;

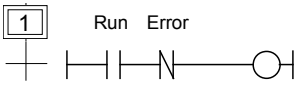
L'espressione può essere costituita da un'espressione costante TRUE o FALSE, un singolo ingresso, una variabile interna booleana, o una combinazione di variabili che diano come risultato un valore booleano. Di seguito, viene proposto un esempio di programmazione ST per le transizioni:

(* Programma SFC con programmazione ST per le transizioni *)



B.3.6.2 Convenzione LD

Il linguaggio **Ladder Diagram** (LD) permette di descrivere la **condizione** abbinata ad una transizione. Il diagramma è composto di un solo ramo con una sola bobina. La bobina rappresenta il valore della transizione. Di seguito, viene proposto un esempio di programmazione LD per le transizioni:



B.3.6.3 Convenzione IL

Con la programmazione Instruction List (IL) , è possibile descrivere direttamente una transizione SFC rispettando la seguente sintassi:

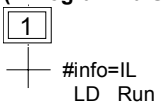
#info=IL
<istruzione>
<istruzione>
....
#endinfo

Il valore contenuto nel **risultato corrente** (registro IL) al termine della sequenza IL, viene abbinato alla condizione della transizione:

current result = 0	→	la condizione vale **FALSE**
current result <> 0	→	la condizione vale **TRUE**

Le parole chiave speciali "#info=IL" e "#endinfo" devono essere inserite esattamente nel modo descritto, viene fatta **distinzione** tra lettere maiuscole e minuscole. I caratteri spazio o tabulazione non possono essere inseriti all'interno, prima o dopo le parole chiave. Segue sotto un esempio di programma IL in una transizione:

(* Programma SFC con un programma IL per transizioni *)



B.3.6.4 Richiamare funzioni da una transizione

La condizione abbinata ad una transizione può essere valutata richiamando qualsiasi sottoprogramma, funzione (scritta in linguaggio FBD, LD, ST o IL) o funzione "C", rispettando la seguente sintassi:

< sub_program > ( ) ;

Il valore restituito dal sottoprogramma o dalla funzione deve essere di tipo booleano e determina il risultato della condizione:

return value = FALSE	→	la condizione vale **FALSE**
return value = TRUE	→	la condizione vale **TRUE**

Esempio di un sottoprogramma chiamato in una transizione:

(* Programma SFC con una chiamata a sottoprogramma per transizioni *)

B.3.7 Regole dinamiche SFC

Le **cinque** regole dinamiche del linguaggio SFC sono:

Situazione iniziale

La situazione iniziale è caratterizzata dai **passi iniziali** che sono, per definizione, in stato attivo all'inizio dell'operazione. In ogni programma SFC deve essere presente **almeno un** passo iniziale.

Attivazione di una transizione

Una transizione può essere sia **abilitata** che **disabilitata**. Viene detta abilitata quando tutti i passi precedenti connessi al corrispondente simbolo di transizione sono **attivi**, altrimenti è disabilitata. Una transizione non può essere **attivata** senza che:

- sia abilitata e
- la condizione associata alla transizione valga TRUE.

Modifica dello stato di passi attivi

L'attivazione di una transizione, porta simultaneamente allo stato attivo i passi immediatamente successivi ed allo stato inattivo i passi immediatamente precedenti.

Attivazione simultanea di transizioni

La sincronizzazione delle transizioni di due rami paralleli può essere fatta in due modi diversi:

1. Dando una regola di sincronizzazione esplicita nelle condizioni associate alle transizioni (usando l'istruzione step.X).
2. Disegnando una doppia divergenza che indica una sincronizzazione ed utilizza una sola transizione.

Attivazione e disattivazione simultanea di un passo

Se, durante un'operazione, un passo viene contemporaneamente attivato e disattivato, la priorità viene data all'attivazione.

B.3.8 Gerarchia dei programmi SFC

Il sistema ISaGRAF permette di descrivere la struttura verticale dei programmi SFC. I programmi SFC sono organizzati ad **albero gerarchico**. Ogni programma SFC può controllare (avviare, terminare …) altri programmi SFC. Questi programmi vengono chiamati **programmi figli** del programma SFC che li controlla. I programmi SFC sono collegati tra loro tramite un **albero gerarchico** rappresentato dalla relazione **padre-figlio**:

Le regole basilari che la struttura gerarchica implica sono:

- I programmi SFC che non hanno padre, vengono chiamati **programmi principali SFC**
- I programmi principali SFC vengono attivati dal sistema all'avvio dell'applicazione
- Un programma può avere molteplici programmi figli
- Un programma figlio non può avere più di un padre
- Un programma figlio può essere controllato solamente dal proprio padre
- Un programma non può controllare il figlio di uno dei suoi propri figli

Le azioni basilari che permettono ad un programma padre SFC di controllare i propri programmi figli sono:

- Avviare (**GSTART**) **Avvia** il programma figlio: attiva ognuno dei suoi passi iniziali. I figli del programma figlio **non** vengono avviati automaticamente.
- Terminare (**GKILL**) **Termina** il programma figlio disattivando ognuno dei suoi passi attivi. I figli del programma figlio vengono a loro volta terminati.
- Sospendere (**GFREEZE**) **Sospende** l'esecuzione del programma (disattiva le azioni di ognuno dei passi attivi e sospende il calcolo delle transizioni), e memorizza lo stato dei passi del programma di modo che il programma possa essere fatto ripartire. I figli del programma figlio vengono a loro volta sospesi.
- Riavviare (**GRST**) **Riavvia** un programma figlio SFC sospeso, riattivando tutti i passi sospesi. I figli del programma figlio vengono a loro volta riavviati.
- Stato (**GSTATUS**) Permette di conoscere l'attuale stato (attivo, inattivo o sospeso) del programma figlio.

B.4 Linguaggio FC

Flow Chart (**FC**) è un linguaggio grafico usato per descrivere **operazioni sequenziali**. Un diagramma Flow Chart è composto di **azioni** e **test**. Tra le azioni ed i test sono interposti dei **collegamenti orientati** che rappresentano il flusso dei dati. Le azioni ed i test possono essere descritti con i linguaggi ST, LD o IL. Funzioni e blocchi funzione di ogni linguaggio (eccetto SFC e FC) possono essere chiamati da azioni e test. Un programma Flow Chart può chiamare un altro programma Flow Chart. Il programma FC richiamato è un **sottoprogramma FC** del programma FC chiamante.

B.4.1 Componenti FC

Segue una lista dei componenti grafici del linguaggio Flow Chart:

Inizio di un diagramma FC

Un simbolo "**Start**" deve apparire all'inizio di un programma Flow Chart. Il simbolo è unico e non può essere omesso. Esso rappresenta lo stato iniziale quando il diagramma viene attivato . Segue il disegno del simbolo "Start":

Il simbolo "**Start**" ha sempre una connessione (nella parte inferiore) verso un altro oggetto del diagramma. Un diagramma Flow Chart non è valido se non esiste una connessione da "Start" ad un altro oggetto.

Fine di un diagramma FC

Un simbolo "**Stop**" deve apparire alla fine di un programma Flow Chart. Il simbolo è unico e non può essere omesso. Esso rappresenta lo stato finale del diagramma, quando la sua esecuzione è stata terminata. Segue il disegno del simbolo "Stop":

Il simbolo "Stop" generalmente ha una connessione (nella parte superiore) proveniente da altri oggetti del diagramma. E' possibile che nessuna connessione sia tracciata verso il simbolo "Stop" (ad esempio in un diagramma che itera sempre), ma il simbolo "Stop" viene in ogni caso disegnato alla fine del diagramma.

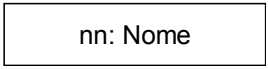
Flusso dei collegamenti FC

Un **collegamento** è una linea che rappresenta un flusso tra due punti del diagramma. Un collegamento è sempre terminato da una freccia. Segue il disegno del simbolo **collegamento**:

Due collegamenti non possono essere connessi allo stesso punto sorgente della connessione.

Azioni FC

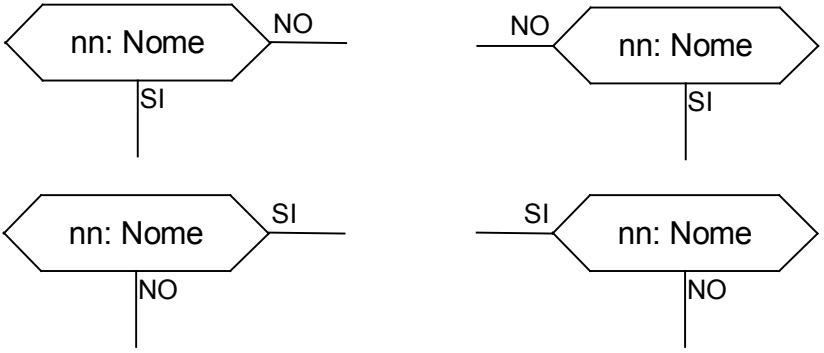
Un simbolo di **azione** rappresenta azioni da eseguire. Un'azione è identificata da un numero e da un nome. Segue il disegno del simbolo "Azione":



Due differenti oggetti dello stesso diagramma non possono avere lo stesso nome o lo stesso numero logico. Il linguaggio di programmazione per un'azione può essere ST, LD o IL. Un'azione è sempre connessa con dei collegamenti, uno che arriva su di essa, uno che parte da essa.

Test FC

Un **test** rappresenta una **condizione** booleana. Un test è identificato da un nome e da un numero logico. In accordo con la valutazione dell'espressione ST, LD o IL associata, il flusso viene diretto verso il percorso "SI" o "NO". Seguono i disegni possibili per un simbolo di test:



Due differenti oggetti test dello stesso diagramma non possono avere lo stesso nome o lo stesso numero logico. Il programma di un oggetto test può essere
- un espressione in linguaggio ST,
- un singolo ramo in linguaggio LD, con nessun simbolo collegato all'unica bobina,
- una sequenza di istruzioni in linguaggio IL. Il registro IL (o il risultato corrente) è utilizzato per valutare il test.

Quando si programma in linguaggio ST, l'espressione può essere opzionalmente seguita da un simbolo di punto-e-virgola. Quando si programma in linguaggio LD, l'unica bobina presente rappresenta il risultato del test. Un test uguale a:

- 0 o FALSO dirige il flusso verso il NO
- 1 o VERO dirige il flusso verso il SI

Un oggetto test è sempre connesso con un collegamento di arrivo, ed entrambe le due connessioni di partenza devono essere definite.

Sottoprogramma FC

Il sistema permette la descrizione di strutture verticali nei programmi FC. I programmi FC sono organizzati con una **gerarchia ad albero**. Ogni programma FC può chiamare altri programmi FC. Tali programmi sono chiamati **sottoprogrammi FC** del programma che li ha chiamati. I programmi FC che chiamano sottoprogrammi FC sono chiamati **programmi padre FC**. I programmi FC sono collegati assieme in gerarchia ad albero, usando una relazione "padre-figlio":

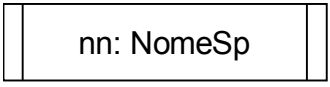
Nota: Non si confondano tra di loro i seguenti 3 oggetti:

- Sottoprogramma
- Programma figlio SFC
- Sottoprogramma FC

Nota: Non si confondano tra di loro i seguenti 2 oggetti:

- Programma padre SFC
- Programma padre FC

Un simbolo di **sottoprogramma FC** in un Flow Chart rappresenta una chiamata ad un sottoprogramma del Flow Chart. L'esecuzione del programma padre FC viene sospesa fino a quando l'esecuzione del sottoprogramma FC è completata. Un sottoprogramma FC è identificato da un nome e da un numero logico, come per gli altri programmi, funzioni o blocchi funzione. Segue il disegno del simbolo di "sottoprogramma FC":



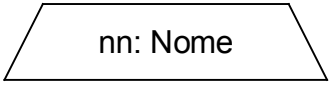
Due oggetti differenti dello stesso diagramma non possono avere lo stesso numero logico. Le regole base della struttura gerarchica FC sono le seguenti:
- I programmi FC che non hanno padre sono chiamati programmi FC principali.
- I programmi FC principali sono attivati dal sistema quando l'applicazione inizia
- Un programma può avere più sottoprogrammi FC
- Un sottoprogramma FC non può avere più di un programma padre FC
- Un sottoprogramma FC può essere chiamato solo dal suo programma padre FC
- Un programma non può chiamare i sottoprogrammi FC di un suo sottoprogramma FC

Lo stesso sottoprogramma FC può comparire più volte nel diagramma padre FC. Una chiamata del sottoprogramma FC rappresenta una esecuzione completa del sotto

diagramma. L'esecuzione del diagramma padre FC viene sospesa durante l'esecuzione del sottodiagramma FC. I blocchi di chiamata a sottoprogrammi FC devono seguire le stesse regole di connessione definite per le azioni.

Azione FC specifica per I/O

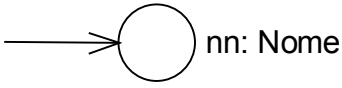
Un simbolo di **azione specifica I/O** rappresenta azioni che devono essere eseguite. Come per altre azioni, un azione specifica I/O è definita da un nome e da un numero logico. La stessa semantica viene utilizzata per azioni standard e azioni specifiche I/O. Lo scopo delle azioni specifiche I/O è quello di rendere il diagramma più leggibile e di focalizzare alle parti del diagramma non-portabili. L'utilizzo delle azioni specifiche I/O è opzionale. Segue il disegno del simbolo "azione specifica I/O":



I blocchi specifici I/O hanno esattamente lo stesso comportamento delle azioni standard. Questo si riferisce alle loro proprietà, alla programmazione ST, LD o IL, e alle regole di connessione.

Connettori FC

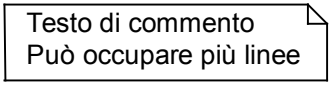
I **Connettori** sono usati per rappresentare collegamenti tra due punti del diagramma senza doverli disegnare. Un connettore è rappresentato con un cerchio ed è connesso alla sorgente del flusso. Il disegno del connettore viene completato, sul lato appropriato (dipendente dalla direzione del flusso di dati), con l'identificazione del punto destinazione (generalmente il nome del simbolo destinazione). Segue il disegno del simbolo connettore:



Un connettore riporta il flusso sempre ad un simbolo definito sul diagramma Flow Chart. Il simbolo destinazione è identificato con il suo numero logico.

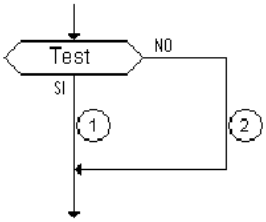
Commenti FC

Un blocco **commento** contiene del testo che non influenza la semantica del diagramma. Un commento può essere inserito ovunque in un qualsiasi spazio inutilizzato della finestra documento Flow Chart, e viene utilizzato per documentare il programma. Segue il disegno del simbolo "commento":



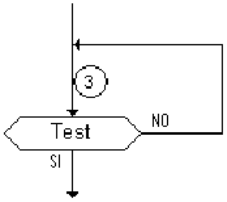
B.4.2 Esempi di strutture complesse FC

Questa sezione mostra esempi di **strutture complesse** che possono essere definite in un diagramma Flow Chart. Tali strutture sono combinazioni di oggetti base collegati insieme.



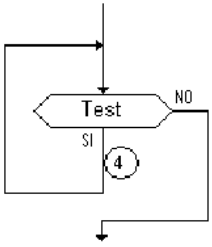
IF / THEN / ELSE

(1) posizione di inserimento di un azione "THEN"
(2) posizione di inserimento di un azione "ELSE"



REPEAT / UNTIL

(3) posizione di inserimento di un azione ripetitiva



WHILE / DO

(3) posizione di inserimento di un azione ripetitiva

B.4.3 Comportamento dinamico FC

L'**esecuzione** di un diagramma può essere spiegata nel modo seguente:

- Il simbolo Start utilizza un ciclo del target
- Il simbolo Stop utilizza un ciclo del target e termina l'esecuzione del diagramma. Dopo che questo simbolo è stato raggiunto, non viene più eseguita nessuna azione del diagramma.
- L'esecuzione di un azione utilizza un ciclo del target.

Nota: Contrariamente ad un diagramma SFC, un azione non è uno stato stabile. Mentre il simbolo di azione è attivo, non avviene nessuna ripetizione di istruzioni.

B.4.4 Controlli FC

A parte la programmazione ST, LD o IL allegata, altre **regole sintattiche** si applicano al Flow Chart. Sotto è riportata la lista delle regole principali:

- Tutti i punti di "connessione" di tutti i simboli devono essere collegati. (la connessione al simbolo "Stop" può essere omessa)

- Tutti i simboli devono essere collegati assieme (non devono apparire parti isolate)
- Tutte le connessioni devono avere una destinazione valida

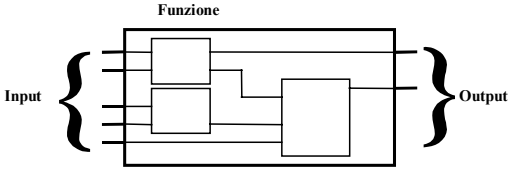
Altri errori di sintassi di minore importanza sono qui riportati:
- Le azioni vuote (senza programma) sono considerate come passi durante lo scheduling
- I test vuoti (senza programma) sono considerati come "sempre veri"

B.5 Linguaggio FBD

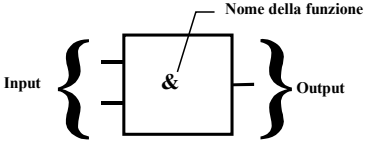
Il linguaggio **Functional Block Diagram** (FBD) è di tipo grafico. Consente al programmatore la generazione di procedure complesse attingendo alle **funzioni** della libreria ISaGRAF ed assemblandole assieme nell'area del diagramma grafico.

B.5.1 Formato del diagramma FBD

Un diagramma FBD descrive una funzione tra **variabili di input** e **variabili di output**. Una funzione viene descritta come insieme di **blocchi funzione elementari**. Le variabili di input ed output sono collegate ai blocchi tramite **linee di connessione**. L'output di un blocco funzione può essere connesso all'input di un altro blocco.



Un'intera funzione gestita con un programma FBD è costituita da blocchi funzione standard **elementari** presi dalla libreria ISaGRAF. Ogni blocco funzione possiede un numero fisso di **punti di connessione di input** ed un numero fisso di **punti di connessione di output**. Un blocco funzione è rappresentato da un singolo **rettangolo**. Gli input collegati sul lato **sinistro**. Gli output collegati sul lato **destro**. Un blocco funzione elementare esegue una singola **funzione** tra i suoi input ed output. Il nome della funzione che deve essere eseguita dal blocco è scritto all'interno del simbolo rettangolare. Ciascun input o output di un blocco ha un ben definito **tipo**.



Le variabili di input di un programma FBD devono essere connesse ai punti di connessione di input di un blocco funzione. Il tipo di ciascuna variabile deve essere lo stesso dell'input associato. Un input di un diagramma FBD può essere costituito da un'espressione **costante**, una qualsiasi variabile **interna** o di **ingresso**, o una variabile di **uscita**.

Le variabili di output di un programma FBD devono essere connesse ai punti di connessione di output dei blocchi funzione. Il tipo di ciascuna variabile deve essere lo stesso dell'output associato. L'output di un programma FBD può essere costituito da una qualunque variabile interna, di uscita, o dal nome del programma (solo per i **sottoprogrammi**). Quando l'output è costituito dal nome del sottoprogramma corrente, rappresenta il valore di ritorno del sottoprogramma (restituito al programma chiamante).

Le variabili di input ed output, gli input e gli output dei blocchi funzione, sono connessi tra loro mediante linee di connessione. Si possono usare delle linee singole per **collegare** due punti logici del diagramma:

- Una variabile di ingresso con l'input di un blocco funzione
- L'output di un blocco funzione e l'input di un altro blocco
- L'output di un blocco funzione ed una variabile di uscita

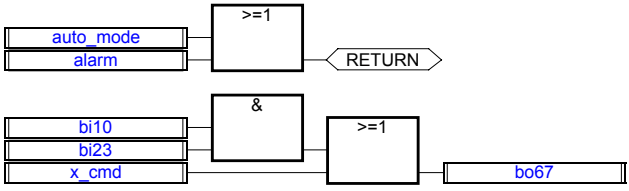
La connessione è **orientata**, ovvero, la linea trasporta i dati dall'estremità sinistra verso l'estremità destra. Le estremità sinistra e destra della connessione devono essere dello **stesso tipo**.

Connessioni multiple a destra permettono di trasmettere un'informazione dall'estremità sinistra a ciascuna delle estremità a destra. Tutte le estremità della connessione devono essere dello stesso tipo.

B.5.1.1 Istruzione RETURN

L'output di un diagramma può essere costituito dalla parola chiave "**<RETURN>**". Tale parola chiave deve essere connessa ad un punto di connessione di output, di tipo booleano, di un blocco funzione. L'istruzione RETURN rappresenta la **fine condizionata** del programma: se l'output della casella collegata all'istruzione ha il valore booleano **TRUE**, la fine (la parte restante) del diagramma non viene eseguita.

(* Esempio di un programma FBD che fa uso dell'istruzione RETURN *)




```
(* Equivalente ST: *)
If auto_mode OR alarm Then
        Return;
End_if;
bo67 := (bi10 AND bi23) OR x_cmd;
```

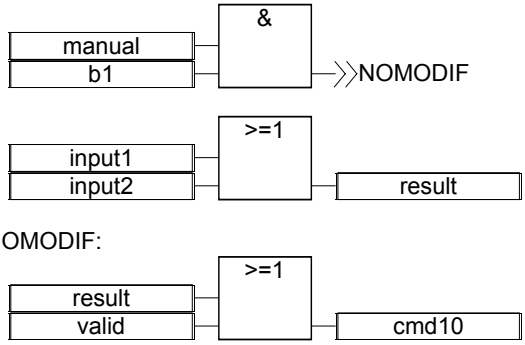
B.5.1.2 Salti ed etichette

Etichette e salti permettono di controllare l'esecuzione del diagramma. Non si possono collegare oggetti alla destra di un salto o di una etichetta. Si usa la seguente notazione:

>>LAB ..... salto ad una etichetta (il nome dell'etichetta è "LAB")
LAB: ..... definizione di una etichetta (il nome dell'etichetta è "LAB")

Quando la linea di connessione alla **sinistra** del salto ha stato booleano **TRUE**, l'esecuzione del programma salta direttamente al simbolo successivo l'etichetta corrispondente.

(* Esempio di un programma FBD che fa uso di etichette e salti *)



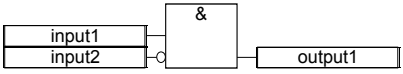
(* Equivalente IL: *)

```
                ld      manual
                and     b1
                jmpc    NOMODIF
                ld      input1
                or      input2
                st      result
NOMODIF:        ld      result
                or      valid
                st      cmd10
```

B.5.1.3 Negazione (Inversione) booleana

Una singola linea di connessione con l'estremità destra connessa all'input di un blocco funzione, può terminare con una **negazione booleana** . La negazione viene rappresentata da un piccolo cerchietto. Per usare una negazione booleana, le estremità sinistra e destra della connessione devono essere di tipo **BOOLEANO**.

(*Esempio di un programma FBD che fa uso di una negazione booleana *)



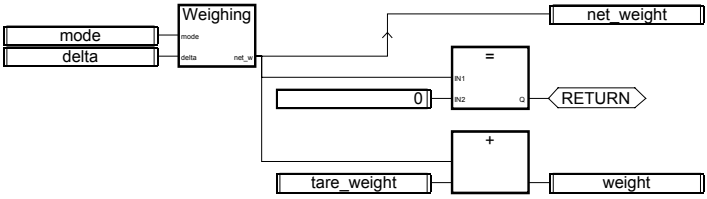
(* Equivalente ST: *)

```
output1 := input1 AND NOT (input2);
```

B.5.1.4 Chiamate a funzioni o blocchi funzione dal diagramma FBD

Il linguaggio FBD permette di richiamare sottoprogrammi, funzioni o blocchi funzione. Un sottoprogramma, una funzione o un blocco funzione, vengono rappresentati da caselle funzione. Il nome scritto nella casella è il nome del sottoprogramma, della funzione o del blocco funzione.
Nel caso di un sottoprogramma o di una funzione, il valore di ritorno è il solo output della casella funzione. Un blocco funzione può avere più output.

(* Esempio di un programma FBD che fa uso di un sottoprogramma *)

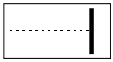


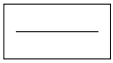
(* Equivalente ST *)
net_weight := Weighing (mode, delta); (* chiamata a sottoprogramma *)
If (net_weight = 0) Then Return; End_if;
weight := net_weight + tare_weight;

B.6 Linguaggio LD

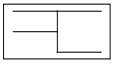
Il linguaggio Ladder Diagram (LD) è la rappresentazione grafica di formule booleane che combinano **contatti** (argomenti di input) con **bobine** (risultati di output). Il linguaggio LD permette di descrivere interrogazioni e modifiche di dati di tipo **booleano** inserendo **simboli grafici** sul diagramma del programma. I simboli grafici LD sono organizzati sul diagramma esattamente come per un diagramma di contatti elettrici. I diagrammi LD sono connessi sul lato sinistro e sul lato destro a **barre di alimentazione** verticali. Questi sono i componenti grafici di base di un diagramma LD:

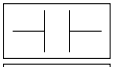

---------- Barra verticale sinistra di alimentazione

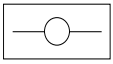

---------- Barra verticale destra di alimentazione


---------- Linea di connessione orizzontale


---------- Linea di connessione verticale

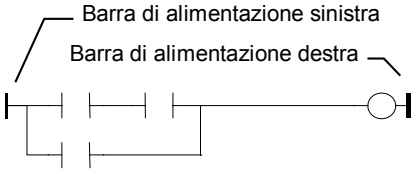

---------- Linee di connessione multiple (tutte connesse tra loro)


---------- Contatto associato ad una variabile

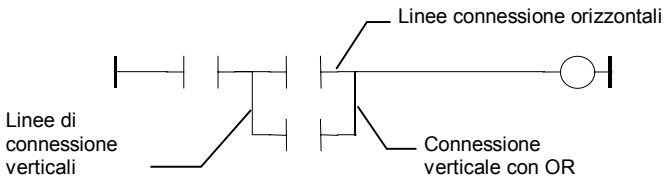

---------- Bobina associata ad una variabile di uscita o interna

B.6.1 Barre di alimentazione e linee di connessione

Un diagramma LD è delimitato sul lato sinistro e destro da linee verticali chiamate, rispettivamente, **barra di alimentazione sinistra** e **barra di alimentazione destra**.



I simboli grafici di un diagramma LD sono connessi alle barre di alimentazione o agli altri simboli mediante **linee di connessione** . Le linee di connessione possono essere orizzontali o verticali.



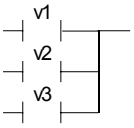
Ogni segmento può assumere uno stato booleano **FALSE** o **TRUE**. Lo stato booleano è il medesimo per tutti i segmenti collegati tra loro direttamente. Ogni linea orizzontale connessa ad una **barra verticale sinistra di alimentazione** ha stato **TRUE**.

B.6.2 Connessioni multiple

Lo stato booleano assunto da una singola connessione orizzontale, è lo stesso alle estremità sinistra e destra della linea stessa. Si possono creare **connessioni multiple** combinando assieme linee verticali ed orizzontali. Lo stato booleano delle estremità di una connessione multipla viene determinato dalle seguenti regole

Una **connessione multipla a sinistra** è costituita da **molteplici** linee orizzontali collegate al lato **sinistro** di una linea verticale e da **una** linea connessa al lato **destro**. Lo stato booleano dell'estremità destra è costituito dall'**OR LOGICO** di tutte le estremità a sinistra.

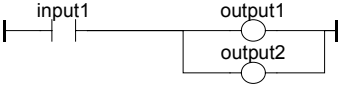
(* Esempio di connessione multipla a SINISTRA *)



(* Lo stato dell'estremità destra vale (v1 OR v2 OR v3) *)

Una **connessione multipla a destra** è costituita da una linea orizzontale collegata al lato **sinistro** di una linea verticale, al cui lato **destro**, sono collegate **molteplici** linee. Lo stato booleano dell'estremità sinistra viene propagato a ciascuna estremità destra.

(* Esempio di connessione multipla a DESTRA *)

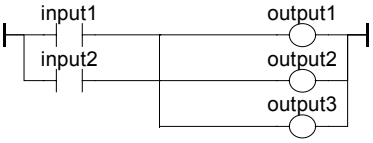



```
(* Equivalente ST: *)
output1 := input1;
output2 := input1;
```

Una **connessione multipla a sinistra e destra** è costituita da **molteplici** linee orizzontali collegate a **sinistra** di una linea verticale, al cui lato destro sono collegate **molteplici** linee.

Lo stato booleano di ciascuna delle estremità a destra è costituito dall'**OR LOGICO** di tutte le estremità a sinistra.

(* Esempio di connessione multipla a SINISTRA e DESTRA *)



(* Equivalente ST: *)
output1 := input1 OR input2;
output2 := input1 OR input2;
output3 := input1 OR input2;

B.6.3 Contatti e bobine LD

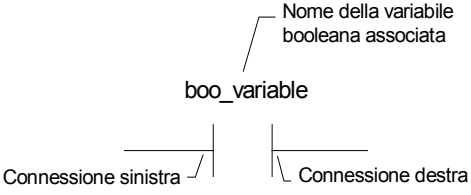
Svariati simboli sono disponibili per rappresentare i contatti in input :

- Contatto diretto
- Contatto inverso
- Contatto con riconoscimento di fronte

Svariati simboli sono disponibili per rappresentare le bobine in output :

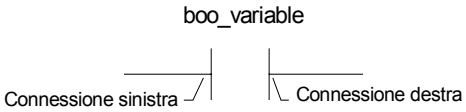
- Bobina diretta
- Bobina inversa
- Bobina di tipo SET
- Bobina di tipo RESET
- Bobina con riconoscimento di fronte

Il nome della variabile è scritto sopra ogni simbolo grafico:



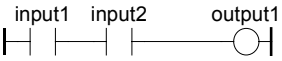
B.6.3.1 Contatto diretto

Un contatto diretto corrisponde ad un'**operazione booleana** tra lo stato di una **linea di connessione** ed una **variabile** booleana.



Lo stato della linea di connessione destra del contatto è l'**AND LOGICO** tra lo stato della linea di connessione sinistra ed il valore della variabile associata al contatto.

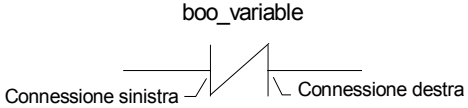
(* Esempio che fa uso di contatti DIRETTI *)



(* Equivalente ST: *)
output1 := input1 AND input2;

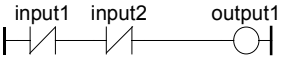
B.6.3.2 Contatto inverso

Un contatto inverso corrisponde ad un'**operazione booleana** tra lo stato di una **linea di connessione** e la **negazione booleana** di una **variabile** booleana.



Lo stato della linea di connessione destra del contatto è l'**AND LOGICO** tra lo stato della linea di connessione sinistra e la **negazione booleana** del valore della variabile associata al contatto.

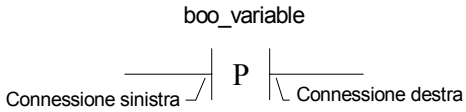
(* Esempio che fa uso di contatti INVERSI *)



(*Equivalente ST: *)
output1 := NOT (input1) AND NOT (input2);

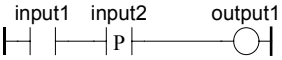
B.6.3.3 Contatto con riconoscimento di fronte di salita

Questo contatto (positivo) corrisponde ad un'**operazione booleana** tra lo stato di una **linea di connessione** ed il fronte di salita di una **variabile** booleana.



Lo stato della linea di connessione destra del contatto viene impostato a TRUE quando lo stato della linea di connessione a sinistra vale TRUE e lo stato della variabile associata **sale** da FALSE a TRUE. In tutti gli altri casi viene impostato a FALSE.

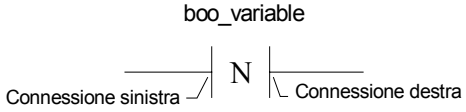
(* Esempio che fa uso di contatti con FRONTE DI SALITA *)



(* Equivalente ST: *)
output1 := input1 AND (input2 AND NOT (input2prev));
(* input2prev è il valore di input2 nel ciclo precedente *)

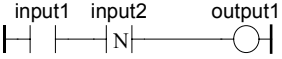
B.6.3.4 Contatto con riconoscimento di fronte di discesa

Questo contatto (negativo) corrisponde ad un'**operazione booleana** tra lo stato di una **linea di connessione** ed il fronte di discesa di una **variabile** booleana.



Lo stato della linea di connessione destra del contatto viene impostato a TRUE quando lo stato della linea di connessione a sinistra vale TRUE e lo stato della variabile associata **cade** da TRUE a. FALSE In tutti gli altri casi viene impostato a FALSE.

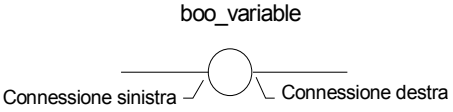
(*Esempio che fa uso di contatti con FRONTE DI DISCESA*)



(*Equivalente ST: *)
output1 := input1 AND (NOT (input2) AND input2prev);
(*input2prev è il valore di input2 nel ciclo precedente*)

B.6.3.5 Bobina diretta

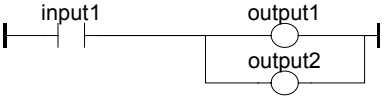
Le bobine dirette corrispondono all'**output booleano** di una **linea di connessione**.



Alla variabile associata viene assegnato lo **stato booleano della connessione sinistra**. Lo stato della connessione sinistra viene propagato alla connessione destra. La connessione destra può essere collegata alla barra verticale destra di alimentazione.

La variabile booleana associata deve avere attributo di variabile di **USCITA** o **INTERNA**. Il nome associato può essere il nome del programma (solamente per **sottoprogrammi**). In questo modo viene assegnato il valore di ritorno del sottoprogramma.

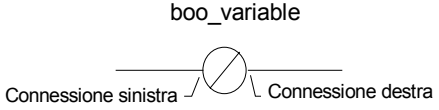
(* Esempio che fa uso di bobine DIRETTE *)




```
(* Equivalente ST: *)
output1 := input1;
output2 := input1;
```

B.6.3.6 Bobina inversa

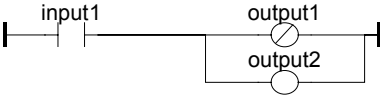
Le bobine inverse corrispondono ad un **output booleano** dato dalla **negazione** booleana dello stato di una linea di connessione.



Alla variabile associata viene assegnata la **negazione** dello **stato booleano della connessione sinistra**. Lo stato della **connessione sinistra** viene propagato alla connessione destra. La connessione destra può essere collegata alla barra verticale destra di alimentazione.

La variabile booleana associata deve avere attributo di variabile di **USCITA** o **INTERNA**. Il nome associato può essere il nome del programma (solamente per **sottoprogrammi**). In questo modo viene assegnato il valore di ritorno del sottoprogramma.

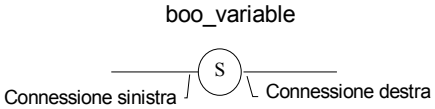
(*Esempio che fa uso di bobine INVERSE*)



```
(* Equivalente ST: *)
output1 := NOT (input1);
output2 := input1;
```

B.6.3.7 Bobina di tipo SET

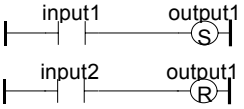
Le bobine di tipo "SET" corrispondono ad un **output booleano** dato dallo stato di una **linea di connessione**.



La variabile associata viene impostata a TRUE quando lo **stato booleano della connessione di sinistra** diventa TRUE. La variabile di output mantiene tale valore fino a quando viene attuata un'inversione da parte di una bobina di tipo "RESET". Lo stato della connessione sinistra viene propagato alla connessione destra. La connessione destra può essere collegata alla barra verticale destra di alimentazione.

La variabile booleana associata deve avere attributo di variabile di **USCITA** o **INTERNA**.

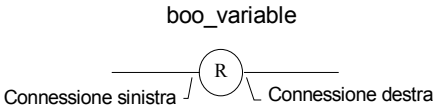
(* Esempio che fa uso di bobine di tipo "SET" e "RESET" *)




```
(* Equivalente ST: *)
IF input1 THEN
  output1 := TRUE;
END_IF;
IF input2 THEN
  output1 := FALSE;
END_IF;
```

B.6.3.8 Bobina di tipo RESET

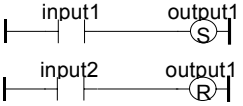
Le bobine di tipo "Reset" corrispondono ad un **output booleano** dato dallo stato di una **linea di connessione**.



La variabile associata viene impostata a FALSE quando lo **stato booleano della connessione di sinistra** diventa TRUE. La variabile di output mantiene tale valore fino a quando viene attuata un'inversione da parte di una bobina di tipo "SET". Lo stato della connessione sinistra viene propagato alla connessione destra. La connessione destra può essere collegata alla barra verticale destra di alimentazione.

La variabile booleana associata deve avere attributo di variabile di **USCITA** o **INTERNA**.

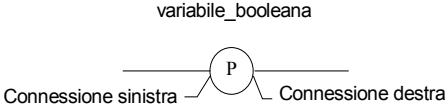
(*Esempio che fa uso di bobine di tipo "SET" e "RESET" *)




```
(*Equivalente ST: *)
IF input1 THEN
  output1 := TRUE;
END_IF;
IF input2 THEN
  output1 := FALSE;
END_IF;
```

B.6.3.9 Bobina con riconoscimento di fronte di salita

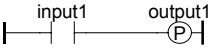
Le bobine "positive" corrispondono ad un **output booleano** dato dallo stato di una **linea di connessione**. Questo tipo di bobine è disponibile solo usando l'editor Quick Ladder.



La variabile associata viene impostata a TRUE quando lo **stato booleano della connessione di sinistra** sale da FALSE a TRUE. In tutti gli altri casi, il valore della variabile di output, viene impostato a FALSE. Lo stato della connessione sinistra viene propagato alla connessione destra. La connessione destra può essere collegata alla barra verticale destra di alimentazione.

La variabile booleana associata deve avere attributo di variabile di **USCITA** o **INTERNA**.

(* Esempio che fa uso di una bobina "positiva" *)

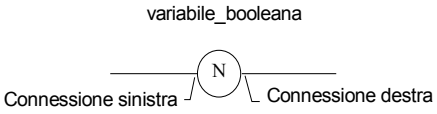



```
(* Equivalente ST: *)
IF (input1 and NOT(input1prev)) THEN
  output1 := TRUE;
ELSE
```

```
  output1 := FALSE;
END_IF;
(*input1prev è il valore di input1 nel ciclo precedente*)
```

B.6.3.10 Bobina con riconoscimento di fronte di discesa

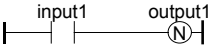
Le bobine "negative" corrispondono ad un **output booleano** dato dallo stato di una **linea di connessione**. Questo tipo di bobine è disponibile solo usando l'editor Quick Ladder.



La variabile associata viene impostata a TRUE quando lo **stato booleano della connessione di sinistra** cade da TRUE a FALSE. In tutti gli altri casi, il valore della variabile di output, viene impostato a FALSE. Lo stato della connessione sinistra viene propagato alla connessione destra. La connessione destra può essere collegata alla barra verticale destra di alimentazione.

La variabile booleana associata deve avere attributo di variabile di **USCITA** o **INTERNA**.

(* Esempio che fa uso di una bobina "negativa" *)



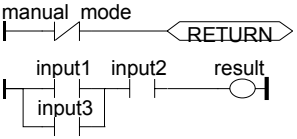

```
(*Equivalente ST: *)
IF (NOT(input1) and input1prev) THEN
  output1 := TRUE;
ELSE
  output1 := FALSE;
END_IF;
(*input1prev è il valore di input1 nel ciclo precedente*)
```

B.6.4 Istruzione RETURN

L'etichetta **RETURN** può essere usata come output per rappresentare la fine condizionata del programma. Nessun'altra connessione è possibile alla destra del simbolo di RETURN.

Quando la linea di **connessione sinistra** ha stato booleano TRUE, il programma termina senza che vengano eseguite le formule inserite nelle successive linee del diagramma.
Nota: Nel caso il programma LD sia un sottoprogramma, il suo nome deve essere associato ad una bobina di output per assegnare il valore di ritorno (restituito al programma chiamante).

(* Esempio che fa uso del simbolo RETURN *)



(* Equivalente ST: *)
If Not (manual_mode) Then RETURN; End_if;
result := (input1 OR input3) AND input2;

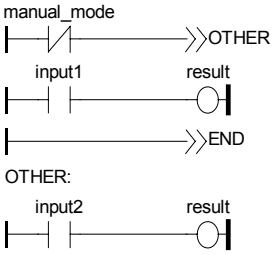
B.6.5 Salti ed etichette

Simboli etichetta, SALTI condizionati o incondizionati, si usano per controllare l'esecuzione del diagramma. Non è possibile alcuna connessione alla destra del simbolo di etichetta o di salto. Si usano le seguenti notazioni:

>>LAB ..... salto ad una etichetta chiamata "LAB"
LAB: ..... definizione di una etichetta chiamata "LAB"

Quando la **connessione sinistra** del simbolo di salto vale TRUE, l'esecuzione del programma viene indirizzata dopo il simbolo dell'etichetta.

(* Esempio che fa uso dei simboli di SALTO e ETICHETTA *)



(* Equivalente IL: *)

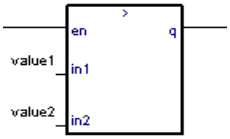
```
                ldn         manual_mode
                jmpc        other
                ld          input1
                st          result
                jmp         END
OTHER:          ld          input2
                st          result
END:            (* fine del programma *)
```

B.6.6 Blocchi in LD

L'editor **Quick LD** permette di collegare caselle funzione a linee booleane. Per funzione si intende un operatore, un blocco funzione o una funzione. Poiché non tutti i blocchi hanno sempre un input booleano e/o un output booleano, per inserire blocchi in un **diagramma LD**, è necessario introdurre i nuovi parametri EN, ENO all'interfaccia del blocco. I parametri EN, ENO non vengono aggiunti nell'editor di programmi FBD/LD, poiché in tale ambiente di programmazione è possibile collegare variabili del tipo necessario.

• L'input "EN"

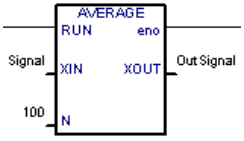
Alcuni operatori, funzioni o blocchi funzione, non hanno il primo input di tipo booleano. Poiché il primo input deve essere sempre collegato al ramo, un altro input, chiamato "**EN**" viene automaticamente inserito in prima posizione. Il blocco viene eseguito solamente se l'input **EN** vale TRUE. Segue un esempio che riporta un operatore di confronto ed il codice equivalente espresso in codice ST:




```
IF rung_state THEN
   q := (value1 > value 2);
ELSE
   q := FALSE;
END_IF;
(* il ramo prosegue in stato q *)
```

• L'output "ENO"

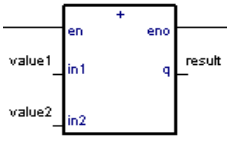
Alcuni operatori, funzioni o blocchi funzione, non hanno il primo output di tipo booleano. Poiché il primo output deve essere sempre collegato al ramo, un altro output, chiamato "**ENO**" viene automaticamente inserito in prima posizione. L'output **ENO**, assume sempre lo stesso stato del primo input del blocco. Segue un esempio che riporta un blocco funzione AVERAGE (MEDIA) ed il codice equivalente espresso in codice ST:




```
AVERAGE(rung_state, Signal, 100);
OutSignal := AVERAGE.XOUT;
eno := rung_state;
(*il ramo prosegue in stato q *)
```

• I parametri "EN" ed "ENO"

In alcuni casi sono necessari entrambi **EN** ed **ENO**. Segue un esempio che riporta operatore aritmetico ed il codice equivalente espresso in codice:




```
IF rung_state THEN
   result := (value1 + value2);
END_IF;
eno := rung_state;
(*il ramo prosegue in stato q *)
```

B.7 Linguaggio ST

ST (Structured Text) è un linguaggio strutturato di alto livello pensato per l'automazione di processi. Questo linguaggio viene principalmente usato per implementare procedure complesse che non possono essere facilmente espresse con i linguaggi di tipo grafico. ST è il linguaggio predefinito per le azioni all'interno di passi e condizioni abbinate alle transizioni del linguaggio **SFC**.

B.7.1 Sintassi ST

Un programma ST consiste in un elenco di **istruzioni** ST. Ogni istruzione termina con il separatore "punto e virgola" carattere **";"**. I nomi usati nel codice sorgente (variabili, identificatori, costanti, parole chiave del linguaggio …) sono separati da **separatori inattivi** (carattere spazio, fine riga, tabulazione) o da **separatori attivi**, con significato ben definito (ad esempio il separatore **">"** indica " più grande di".
Nel testo possono essere liberamente inseriti commenti. Un commento inizia con la coppia di caratteri **"(*"** e termina con **"*)"**.
Vengono qui riportati i tipi base di istruzione ST:

- istruzione di **assegnamento** (variabile := espressione;)
- chiamata a **sottoprogramma** o **funzione**
- chiamata a **blocchi funzione**
- istruzioni di **selezione** (IF, THEN, ELSE, CASE...)
- istruzioni di **iterazione** (FOR, WHILE, REPEAT...)
- istruzioni di **controllo** (RETURN, EXIT...)
- istruzioni speciali per il collegamento con altri linguaggi, come ad esempio **SFC**

I separatori inattivi possono essere inseriti liberamente tra separatori attivi, espressioni costanti ed identificatori. I separatori inattivi ST sono: il carattere **Spazio** (blank), **Tabulazione** ed il carattere **Fine riga** (End of line). Diversamente dai linguaggi con formattazione a riga, come ad esempio IL, la fine riga può essere inserita in qualunque punto del programma. È bene, tuttavia, rispettare alcune regole per migliorare la leggibilità di un programma ST:

- Non scrivere più di una istruzione per riga
- Inserire dei rientri (indentazione) con le tabulazioni nel caso di sequenze di istruzioni complesse
- Inserire commenti per migliorare la leggibilità di righe e paragrafi

Leggibilità dei sorgenti - esempi:

Scarsa leggibilità	Ottima leggibilità

imax := max_ite; cond := X12;
if not(cond (* allarme *)
then return; end_if;
for i (* indice *) := 1 to max_ite
do if i <> 2 then Spcall();
end_if; end_for;
(* nessun effetto se allarme *)
	(* imax : numero di iterazioni *)
	(* i: indice ciclo FOR *)
	(* cond: validità del processo *)
	imax := max_ite;
	cond := X12;
	if not (cond) then
	  return;
	end_if;
	
	(* ciclo del processo *)
	for i := 1 to max_ite do
	  if i <> 2 then
	  Spcall ();
	  end_if;
	end_for;

B.7.2 Espressioni e parentesi

Le espressioni ST combinano assieme **operatori** ST ed operandi variabili o costanti. Nel caso di espressione singola (un operatore ST assieme ad operandi), gli operandi devono essere dello stesso **tipo**. L'espressione singola risulta essere dello stesso tipo dei suoi operandi e può, a sua volta, essere usata in espressioni più complesse. Ad esempio:

(boo_var1 AND boo_var2)	è di tipo BOOLEANO
not (boo_var1)	è di tipo BOOLEANO
(sin (3.14) + 0.72)	è di tipo REALE ANALOG
(t#1s23 + 1.78)	non è un'espressione valida

Le **parentesi** servono ad isolare parti di un'espressione e a definire esplicitamente la priorità delle operazioni. In un'espressione complessa priva di parentesi, la sequenza delle operazioni è implicitamente data dalla **priorità** predefinita tra operatori ST. Ad esempio:

2 + 3 * 6	equivale a 2+18=20	poiché l'operatore moltiplicazione ha priorità più alta
(2+3) * 6	equivale a 5*6=30	la priorità è data dalle parentesi

Avvertenza: In un'espressione si possono annidare fino ad un massimo di **8** livelli di parentesi.

B.7.3 Chiamate a funzioni o blocchi funzione

Funzioni ST standard possono essere usate negli oggetti seguenti:

- Sottoprogrammi
- Funzioni e blocchi funzione di libreria scritte con linguaggi IEC
- Funzioni e blocchi funzione "C"

- Funzioni di conversione di tipo

B.7.3.1 Chiamate a sottoprogrammi o funzioni

Nome: nome del sottoprogramma chiamato
o della funzione di libreria scritta in linguaggio IEC o "C"

Descrizione: chiama un sottoprogramma o una funzione ST, IL, LD o FBD o una funzione “C” ed ottiene il valore di ritorno

Sintassi: **<variabile> := <sottoprog> (<par1>,** ... **<parN>);**

Operandi: Il tipo del valore di ritorno e dei parametri di chiamata deve essere lo stesso di quello definito per il sottoprogramma.

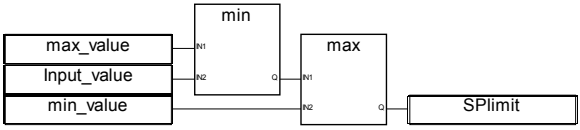
Valore di ritorno: valore restituito dal sottoprogramma

Qualsiasi espressione può contenere chiamate a sottoprogrammi, anche transizioni SFC.

Esempio1: Chiamata a sottoprogramma

```
(* Programma principale ST *)
(* ottiene un valore analogico e lo converte in valore temporale *)
ana_timeprog := SPlimit ( tprog_cmd );
appl_timer := tmr (ana_timeprog * 100);
```

(* Il programma FBD chiamato è 'SPlimit' *)



Esempio2: Chiamata a funzione

```
(* le funzioni usate in espressioni complesse: min, max, right, mlen e left sono funzioni
standard "C" *)
limited_value := min (16, max (0, input_value) );
rol_msg := right (message, mlen (message) - 1) + left (message, 1);
```

B.7.3.2 Chiamata a blocchi funzione

Nome: nome dell’istanza di blocco funzione

Descrizione: chiama un blocco funzione dalla libreria ISaGRAF o dalla libreria utente ed accede ai parametri di ritorno

Sintassi: **(* chiamata a blocco funzione *)**

<blockname> (<p1>, <p2> ...);
(ottiene i parametri di ritorno *)
<result> := <blockname>. <ret_param1>;
...
<result> := <blockname>. <ret_paramN>;

Operandi: i parametri di chiamata sono costituiti da espressioni il cui tipo coincide con quello dei parametri dello specifico blocco funzione

Valore di ritorno: i parametri di ritorno si leggono come specificato nella Sintassi.

Consultare la libreria ISaGRAF per la descrizione ed il tipo di ciascun parametro di blocco funzione. L'istanza (il nome della copia) di blocco funzione deve essere dichiarata nel dizionario

Esempio :

```
(* Programma ST che chiama un blocco funzione *)

(* dichiarazione dell'istanza del blocco nel dizionario: *)
(* trigb1 : blocco R_TRIG – riconoscimento di fronte di salita *)

(* attivazione del blocco funzione da linguaggio ST *)
trigb1 (b1);
(* accesso ai parametri di ritorno *)
If (trigb1.Q) Then nb_edge := nb_edge + 1; End_if;
```

B.7.4 Operatori booleani ST specifici

I seguenti operatori booleani sono specifici del linguaggio ST:

- REDGE riconoscimento di fronte di salita
- FEDGE riconoscimento di fronte di discesa

Altri operatori booleani standard:
- NOT negazione booleana
- AND (&) AND logico
- OR OR logico
- XOR OR logico esclusivo

possono essere usati. Per la descrizione, si veda la sezione «Operatori, blocchi funzione e funzioni standard».

B.7.4.1 L'operatore "REDGE"

Nome: **REDGE**

Descrizione: valuta il fronte di salita di un'espressione booleana completa

Sintassi: **<edge> := REDGE (<espressione_booleana>,<variabile_memo>);**

Operandi: il primo operando è costituito da una qualsiasi variabile booleana o espressione complessa
il secondo operando è una variabile booleana interna usata per mantenere il precedente stato dell'espressione

Valore di ritorno: TRUE quando l'espressione passa da FALSE a TRUE
FALSE negli altri casi

Non è possibile, con l'operatore REDGE, interrogare il fronte di salita di un'espressione più di **una volta** nello stesso ciclo di esecuzione. Questo operatore può essere usato per descrivere la condizione associata ad una transizione SFC.
Avvertenza: La variabile booleana <variabile_memo>, usata per mantenere il precedente stato dell'espressione, non può essere usata per fronti di espressioni differenti.

Nel caso l'espressione sia una variabile booleana chiamata "**xxx**", è consigliabile dichiarare una variabile interna unica chiamata "**EDGE_xxx**" da usarsi nelle espressioni REDGE per la variabile "xxx". In questo modo si evita che la variabile memo venga sovrascritta da altre valutazioni REDGE.

Esempio:

```
(* Programma ST che fa uso dell'operatore REDGE *)

(* questo programma conta i fronti di salita di un input booleano *)
(* Bi120 è una variabile booleana di input *)
(* Edge_Bi120 è la memoria dello stato della variabile Bi120 *)

If REDGE (Bi120, Edge_Bi120) Then
        Counter := Counter + 1;
End_if;
```

Nota: questo operatore non è conforme alle norme IEC1131-3. È possibile usare il blocco standard R_TRIG. Viene mantenuto per ragioni di compatibilità.

B.7.4.2 L'operatore "FEDGE"

Nome: **FEDGE**

Descrizione: valuta il fronte di discesa di un'espressione booleana

Sintassi: **<edge> := FEDGE (<espressione_booleana>, <variabile_memo>);**

Operandi: il primo operando è costituito da una qualsiasi variabile booleana o espressione complessa
il secondo operando è una variabile booleana interna usata per mantenere il precedente stato dell'espressione

Valore di ritorno: TRUE quando l'espressione passa da TRUE a FALSE
FALSE negli altri casi

Non è possibile, con l'operatore FEDGE, interrogare il fronte di salita di un'espressione più di **una volta** nello stesso ciclo di esecuzione. Questo operatore può essere usato per descrivere la condizione abbinata ad una transizione SFC.
Avvertenza: La variabile booleana "variabile_memo", usata per mantenere il precedente stato dell'espressione, non può essere usata per fronti di espressioni differenti.

Nel caso l'espressione sia una variabile booleana chiamata "**xxx**", è consigliabile dichiarare una variabile interna unica chiamata "**EDGE_xxx**" da usarsi nelle espressioni FEDGE per la variabile "xxx". In questo modo si evita che la variabile memo venga sovrascritta da altre valutazioni FEDGE.

Esempio:

```
(*Programma ST che fa uso dell'operatore FEDGE*)

(*questo programma conta i fronti di caduta di un input booleano *)
(* Bi120 è una variabile booleana di input *)
(* Edge_Bi120 è la memoria dello stato della variabile Bi120*)

If FEDGE (Bi120, Edge_Bi120) Then
        Counter := Counter + 1;
End_if;
```

Nota: questo operatore non è conforme alle norme IEC1131-3. È possibile usare il blocco standard F_TRIG. Viene mantenuto per ragioni di compatibilità.

B.7.5 Istruzioni base ST

Le istruzioni base del linguaggio ST sono:

- Assegnazione
- Istruzione RETURN
- Struttura IF-THEN-ELSIF-ELSE
- Istruzione CASE
- Istruzione iterativa WHILE
- Istruzione iterativa REPEAT
- Istruzione iterativa FOR
- Istruzione EXIT

B.7.5.1 Assegnazione

Nome: **:=**

Descrizione: assegna un'espressione ad una variabile

Sintassi: **<variable> := <any_expression> ;**

Operandi: la variabile deve avere attributo di variabile di **USCITA** o **INTERNA** variabile ed espressione devono essere dello stesso tipo

L'espressione può essere una chiamata a sottoprogramma o una funzione della libreria ISaGRAF

Esempio:

```
(* Programma ST con assegnazioni *)

(* variabile <<= variabile *)
bo23 := bo10;

(* variabile <<= espressione *)
bo56 := bx34 OR alrm100 & (level >= over_value);
result := (100 * input_value) / scale;

(* assegnazione con valore di ritorno da sottoprogramma *)
rc := PSelect ( );

(* asseganzione con chiamata a funzione *)
limited_value := min (16, max (0, input_value) );
```

B.7.5.2 Istruzione RETURN

Nome: **RETURN**

Descrizione: termina l'esecuzione del programma corrente

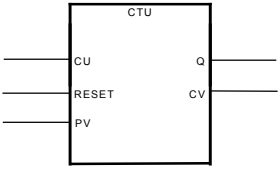
Sintassi: **RETURN ;**

Operandi: (nessuno)

All'interno di un **blocco SFC di tipo Action**, l'istruzione RETURN indica la fine dell'esecuzione solamente di tale blocco.

Esempio:

```
(* Specifica FBD del programma: contatore programmabile *)
```




```
(* Implementazione ST del programma, con l'uso dell'istruzione RETURN *)

If not (CU) then
```

```
        Q := false;
        CV := 0;
        RETURN;   (* termina il programma *)
end_if;

if R then
        CV := 0;
else
        if (CV < PV) then
                CV := CV + 1;
        end_if;
end_if;
Q := (CV >= PV);
```

B.7.5.3 Istruzione IF-THEN-ELSIF-ELSE

Nome: **IF ... THEN ... ELSIF ... THEN ... ELSE ... END_IF**

Descrizione: esegue uno di due elenchi di istruzioni ST la scelta avviene in base al valore di un'espressione booleana

Sintassi:

```
IF <espressione_booleana> THEN
  <istruzione> ;
  <istruzione> ;
  ...
ELSIF <espressione_booleana> THEN
  <istruzione> ;
  <istruzione> ;
  ...
ELSE
  <istruzione> ;
  <istruzione> ;
  ...
END_IF;
```

Le istruzioni ELSE ed ELSIF sono facoltative. Se non viene specificata l'istruzione ELSE, non viene eseguita alcuna istruzione quando la condizione vale FALSE.

Esempio:

```
(* Programma ST che fa uso dell'istruzione IF *)

IF manual AND not (alarm) THEN
        level := manual_level;
        bx126 := bi12 OR bi45;
ELSIF over_mode THEN
        level := max_level;
ELSE
        level := (lv16 * 100) / scale;
END_IF;
```

```
(* La struttura IF usata senza ELSE *)
If overflow THEN
        alarm_level := true;
END_IF;
```

B.7.5.4 Istruzione CASE

Nome: **CASE ... OF ... ELSE ... END_CASE**

Descrizione: esegue uno di più elenchi di istruzioni ST
la scelta avviene in base ad un'espressione intera

Sintassi:

CASE <espressione_intera> OF
<valore> : <istruzioni> ;
<valore> , <valore> : <istruzioni> ;
...
ELSE
<istruzioni> ;
END_CASE;

I valori valutati dall'istruzione CASE, devono essere espressioni costanti intere. Valori diversi, separati da virgole, portano allo stesso elenco di istruzioni. L'istruzione ELSE è facoltativa.

Esempio:

```
(* Programma ST che fa uso dell'istruzione CASE*)

CASE codice_errore OF
        255:    err_msg := 'Divisione per zero;
                fatal_error := TRUE;
        1:      err_msg := 'Overflow';
        2, 3:   err_msg := 'Segno errato;
ELSE
        err_msg := 'Errore sconosciuto;
END_CASE;
```

B.7.5.5 Istruzione WHILE

Nome: **WHILE ... DO ... END_WHILE**

Descrizione: struttura iterativa per un gruppo di istruzioni ST
la condizione che permette di continuare, viene valutata PRIMA di ogni iterazione

Sintassi:

WHILE <espressione_booleana> DO
<istruzione> ;
<istruzione> ;
...
END_WHILE ;

Avvertenza: Poiché il sistema ISaGRAF è **sincrono**, le variabili di ingresso, non vengono aggiornate durante le iterazioni WHILE. Il cambiamento di stato di una variabile di ingresso non può essere usato come condizione per un'istruzione WHILE.

Esempio:

```
(* Programma ST che fa uso dell'istruzione WHILE *)

(* questo programma utilizza funzioni specifiche «C» per leggere caratteri *)
(* da una porta seriale *)

string := ''; (*stringa vuota *)
nbchar := 0;

WHILE ((nbchar < 16) & ComIsReady ( )) DO
        string := string + ComGetChar ( );
        nbchar := nbchar + 1;
END_WHILE;
```

B.7.5.6 istruzione REPEAT

Nome: **REPEAT ... UNTIL ... END_REPEAT**

Descrizione: struttura iterativa per un gruppo di istruzioni ST
la condizione che permette di continuare, viene valutata DOPO ogni iterazione

Sintassi:

```
REPEAT
  <istruzione> ;
  <istruzione> ;
  ...
UNTIL <espressione_booleana>
END_REPEAT ;
```

Avvertenza: Poiché il sistema ISaGRAF è **sincrono**, le variabili di ingresso, **non** vengono aggiornate durante le iterazioni REPEAT. Il cambiamento di stato di una variabile di ingresso **non** può essere usato come condizione per un'istruzione REPEAT.

Esempio:

```
(* Programma ST che fa uso dell'istruzione REPEAT *)

(* questo programma utilizza funzioni specifiche «C» per leggere caratteri *)
(* da una porta seriale *)

string := ''; (*stringa vuota *)
nbchar := 0;
IF ComIsReady ( ) THEN
        REPEAT
                string := string + ComGetChar ( );
```

```
            nbchar := nbchar + 1;
        UNTIL ( (nbchar >= 16) OR NOT (ComIsReady ( )) )
        END_REPEAT;
END_IF;
```

B.7.5.7 Istruzione FOR

Nome: **FOR ... TO ... BY ... DO ... END_FOR**

Descrizione: esegue un numero limitato di iterazioni, usando come indice una variabile intera di tipo analogico

Sintassi:

```
FOR <indice> := <mini> TO <maxi> BY <passo> DO
  <istruzione> ;
  <istruzione> ;
END_FOR;
```

Operandi:

	index:	variabile analogica interna incrementata ad ogni ciclo
	mini:	valore iniziale (prima del primo ciclo) dell'indice
	maxi:	valore massimo che può assumere l'indice
	step:	incremento dell'indice ad ogni ciclo

La direttiva [BY passo] è facoltativa. Se non diversamente specificato, l'incremento è di passo 1.

Avvertenza: Poiché il sistema ISaGRAF è **sincrono**, le variabili di ingresso, **non** vengono aggiornate durante le iterazioni FOR.

Questo è l'equivalente ciclo "while" per l'istruzione FOR:

```
index := mini;
while (index <= maxi) do
  <istruzione> ;
  <istruzione> ;
  index := index + step;
end_while;
```

Esempio:

```
(* Programma ST che fa uso dell'istruzione FOR *)
(* questo programma estrae i caratteri corrispondenti a cifre da una stringa *)

length := mlen (message);
target := ''; (*stringa vuota *)
FOR index := 1 TO length BY 1 DO
        code := ascii (message, index);
        IF (code >= 48) & (code <= 57) THEN
                target := target + char (code);
        END_IF;
END_FOR;
```

B.7.5.8 Istruzione EXIT

Nome: **EXIT**

Descrizione: uscita da un'istruzione iterativa FOR, WHILE o REPEAT

Sintassi: **EXIT;**

Operandi: (nessuno)

EXIT viene comunemente usata in un'istruzione IF all'interno di un blocco FOR, WHILE o REPEAT.

Esempio:

```
(* Programma ST che fa uso dell'istruzione EXIT *)
(* questo programma esegue la ricerca di un carattere in una stringa *)

length := mlen (message);
found := NO;
FOR index := 1 TO length BY 1 DO
        code := ascii (message, index);
        IF (code  = searched_char) THEN
                found := YES;
                EXIT;
        END_IF;
END_FOR;
```

B.7.6 Estensioni ST

Le seguenti funzioni estendono il linguaggio ST:

- TSTART - TSTOP: controllo timer

Le seguenti istruzioni sono disponibili per controllare l'esecuzione di programmi **figli** SFC. Possono essere usate nei passi SFC, all'interno di Blocchi azione ACTION(): ... END_ACTION;.

- GSTART — avvia un programma SFC
- GKILL — termina un programma SFC
- GFREEZE — sospende un programma SFC
- GRST — riavvia un programma SFC sospeso
- GSTATUS — ottiene lo stato corrente di un programma SFC

Avvertenza: Queste funzioni **non** fanno parte della normativa IEC 1131-3.
Equivalgono, in un passo SFC, a GSTART e GKILL:

```
child_name(S); (* equivale a GSTART(child_name); *)
child_name(R); (* equivale a GKILL(child_name); *)
```

I seguenti campi permettono di conoscere lo stato di un passo SFC:

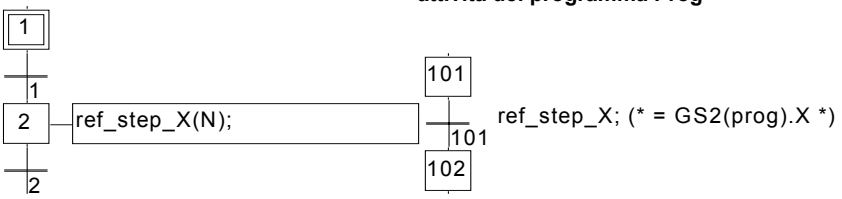
GSnnn.x valore booleano che rappresenta lo stato di l'attività del passo
GSnnn.t tempo trascorso dall'ultima attivazione del passo
("**nnn**" rappresenta il numero di riferimento del passo SFC)

È possibile, anche, controllare lo stato di l'attività di un passo dichiarato in un diverso programma SFC, usando la sintassi riportata di seguito:

GSnnn(progname).x

Avvertenza: il riferimento ad un passo di un diverso programma, con questa sintassi, **non** fa parte delle norme IEC 1131-3. Un modo semplice per ottenere lo stesso risultato, ottemperando alle regole IEC, consiste nel dichiarare nel dizionario una variabile booleana globale che rappresenta l'attività del passo da controllare (ad esempio ref_step_X). Inserire, poi, nel passo, la variabile qualificata da N (ref_step_X(N)). Usare poi la variabile, nel programma da cui si vuole controllare l'attività del passo.

Programma Prog **l'altro programma che necessita dello stato di attività del programma Prog**



B.7.6.1 Istruzione TSTART

Nome: **TSTART**

Descrizione: avvia il conteggio di una variabile di tipo timer
il valore del timer non viene modificato dal comando TSTART, ovvero, il conteggio viene avviato a partire dall'attuale valore del timer.

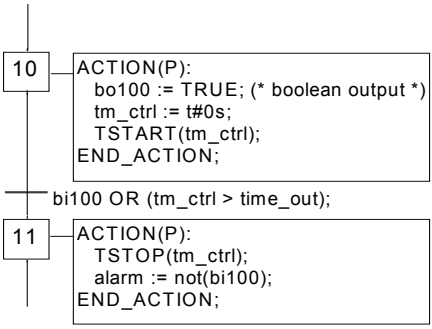
Sintassi: **TSTART (<variabile_timer>);**

Operandi: qualsiasi timer non attivo

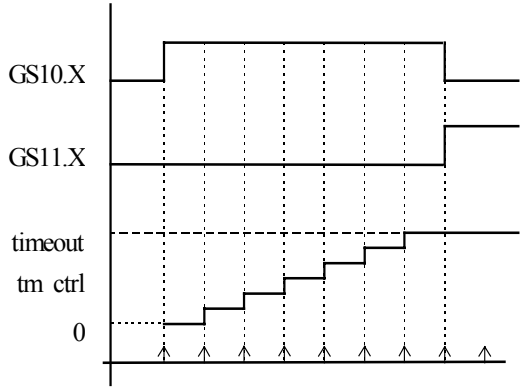
Valore di ritorno: (nessuno)

Esempio:

(* Programma SFC che fa uso delle istruzioni TSTART e TSTOP *)



Il diagramma Tempo se bi100 vale sempre FALSE:



Il timer mantiene lo stesso valore per la durata di un ciclo.

B.7.6.2 Istruzione TSTOP

Nome: **TSTOP**

Descrizione: ferma l'aggiornamento di una variabile di tipo timer
Il valore del timer non viene modificato dal comando TSTOP

Sintassi: **TSTOP (<variabile_timer>);**

Operandi: qualsiasi variabile timer attiva

Valore di ritorno: (nessuno)

Esempio: Si veda TSTART (la funzione descritta precedentemente)

B.7.6.3 Istruzione GSTART

Nome: **GSTART**

Descrizione: Avvia un programma figlio SFC inserendovi un token in ciascuno dei passi iniziali

Sintassi: **GSTART (<child_program>);**

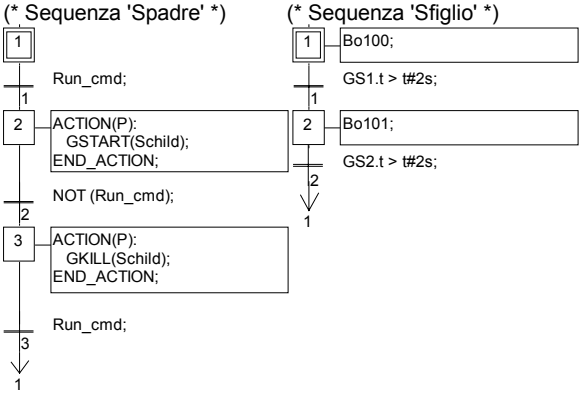
Operandi: il programma SFC specificato **deve** essere figlio di quello in cui è scritta l'istruzione

Valore di ritorno: (nessuno)

I figli del programma figlio, **non** vengono avviati automaticamente dall'istruzione GSTART.
Nota: Poiché GSTART non fa parte delle norme IEC 1131-3, è preferibile avviare un programma figlio SFC, usando il qualificatore S, con la seguente sintassi:

```
Child_name(S);
```

Esempio: Uso di GSTART e GKILL



B.7.6.4 Istruzione GKILL

Nome: **GKILL**

Descrizione: termina un programma figlio SFC eliminandovi i token presenti nei passi

Sintassi: **GKILL (<programma_figlio>);**

Operandi: il programma SFC specificato **deve** essere figlio di quello in cui è scritta l'istruzione

Valore di ritorno: (nessuno)

I figli del programma figlio vengono, assieme ad esso, automaticamente terminati.
Nota: Poichè GKILL non fa parte delle norme IEC 1131-3, è preferibile terminare un programma figlio SFC, usando il qualificatore R, con la seguente sintassi:

```
Child_name(R);
```

Esempio: Si veda GSTART (la funzione descritta precedentemente)

B.7.6.5 Istruzione GFREEZE

Nome: **GFREEZE**

Descrizione: Sospende l'esecuzione di un programma figlio SFC.
Il programma sospeso può essere riavviato con l'istruzione GRST.

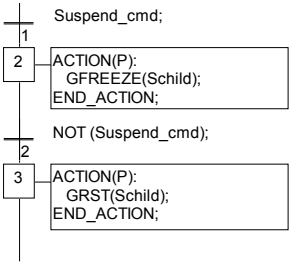
Sintassi: **GFREEZE (<programma_figlio>);**

Operandi: il programma SFC specificato **deve** essere figlio di quello in cui è scritta l'istruzione

Valore di ritorno: (nessuno)

I figli del programma figlio vengono automaticamente sospesi assieme al programma specificato.
Nota: GFREEZE non è compreso nelle norme IEC 1131-3.

Esempio:



B.7.6.6 Istruzione GRST

Nome: **GRST**

Descrizione: riavvia un programma figlio SFC, precedentemente sospeso dall'istruzione GFREEZE.

Sintassi: **GRST (<programma_figlio>);**

Operandi: il programma SFC specificato **deve** essere figlio di quello in cui è scritta l'istruzione

Valore di ritorno: (nessuno)

L'istruzione GRST provvede automaticamente, a riavviare i figli del programma figlio.
Nota: GRST non è compreso nelle norme IEC 1131-3.

Esempio: Si veda GFREEZE (la funzione descritta precedentemente)

B.7.6.7 Istruzione GSTATUS

Nome: **GSTATUS**

Descrizione: restituisce l'attuale stato di un programma SFC

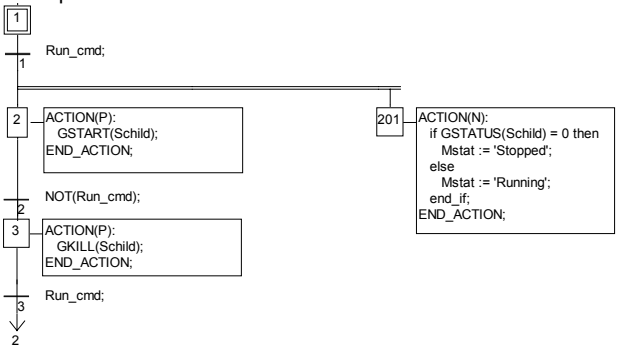
Sintassi: **<ana_var> := GSTATUS (<programma_figlio>);**

Operandi: il programma SFC specificato **deve** essere figlio di quello in cui è scritta l'istruzione

Valore di ritorno: 0 = programma non attivo (terminato)
1 = programma attivo (avviato)
2 = programma sospeso

Nota: GSTATUS non è compreso nelle norme IEC 1131-3.

Esempio:



B.8 Linguaggio IL

Instruction List, o **IL** è un linguaggio di basso livello, molto efficace per piccole applicazioni o per ottimizzare parti di un'applicazione. Le istruzioni sono sempre relative al **risultato corrente (o registro IL**). L'operatore indica l'operazione che deve essere svolta tra il valore corrente e l'operando. Il risultato dell'operazione viene, nuovamente, memorizzato nel risultato corrente.

B.8.1 Sintassi IL

Un programma IL consiste in un elenco di **istruzioni**. Ogni istruzione deve iniziare su una nuova riga e deve contenere un **operatore** completato con **modificatori** (facoltativi) e con uno o più **operandi** (se richiesti dall'operazione specifica) separati da caratteri virgola **','**.
L'istruzione può essere preceduta da una **etichetta** seguita dal carattere due punti **':'**.
Un eventuale commento, abbinato all'istruzione, deve essere posto come ultima parte della riga. I commenti iniziano sempre con la coppia di caratteri **'(*'** e terminano con **'*)'**. Possono esserci righe vuote tra un'istruzione e l'altra. I commenti possono essere messi anche su righe vuote.
Di seguito viene riportato un esempio di righe di istruzione:

```
Etichetta     Operatore  Operando  Commenti

Start:        LD         IX1       (* pressione del pulsante *)
              ANDN       MX5       (* il comando è permesso *)
              ST         QX2       (* avvio motore *)
```

B.8.1.1 Etichette

Un **etichetta** seguita dai due punti **':'** può precedere l'istruzione. Una etichetta può trovar posto anche su una riga vuota. Le etichette vengono usate come operandi per alcune operazioni come i salti. Il nome delle etichette deve rispettare le seguenti regole:

- il nome non deve eccedere i **16** caratteri
- il primo carattere deve essere una **lettera**
- i successivi caratteri devono essere **lettere**, **cifre** o il carattere **'_'**

Lo stesso nome **non** può essere usato per più di una etichetta nello stesso programma IL. Una etichetta può avere lo stesso nome di una variabile.

B.8.1.2 Modificatori di operatori

Di seguito vengono riportati i modificatori di operatori disponibili. Il carattere modificatore deve completare il nome dell'operatore, senza che vi sia alcun carattere vuoto tra essi:

N negazione booleana dell'operando
(operazione rinviata
C operazione condizionata

Il modificatore **'N'** indica una negazione booleana dell'operando. Ad esempio, l'istruzione **ORN IX12** viene interpretata come: **result := result OR NOT (IX12)**.

Il modificatore parentesi **'('** indica che la valutazione dell'istruzione deve essere rinviata fino a quando non venga incontrato il modificatore parentesi chiusa **')'**.

Il modificatore **'C'** indica che l'istruzione abbinata deve essere eseguita solamente se il risultato corrente ha valore booleano TRUE (diverso da 0 nel caso di operatori non booleani). Il modificatore **'C'** può essere usato in combinazione con il modificatore **'N'** per indicare che l'istruzione deve essere eseguita solamente nel caso il risultato corrente abbia valore booleano FALSE (o 0 per valori non booleani).

B.8.1.3 Operazioni rinviate

Poiché esiste un solo registro IL (il risultato corrente), alcune operazioni devono essere rinviate , così da permettere di cambiare l'ordine di esecuzione delle istruzioni.
Le parentesi permettono di indicare le operazioni rinviate:

'('	è un modificatore	indica che l'operazione deve essere rinviata
')'	è un operatore	esegue l'operazione rinviata

Il modificatore parentesi aperta **'('** indica che la valutazione dell'istruzione deve essere rinviata fino a quando non venga incontrato l'operatore parentesi chiusa **')'**. Ad esempio, la seguente sequenza:

```
AND(    IX12
OR      IX35
)
```

viene interpretata come:

result := result AND (IX12 OR IX35)

B.8.2 Operatori IL

La seguente tavola riassume gli operatori standard del linguaggio IL:

Operatore	Modificatori	Operandi	Descrizione
LD	N	Variabile, costante	Carica (load) operando
ST	N	Variabile	memorizza (store) il risultato corrente
S		variabile BOOLEANA	imposta a TRUE
R		variabile BOOLEANA	imposta a FALSE
AND	N (	BOO	AND booleano
&	N (	BOO	AND booleano
OR	N (	BOO	OR booleano
XOR	N (	BOO	OR esclusivo

ADD	(	variabile, costante	Addizione
SUB	(	variabile, costante	Sottrazione
MUL	(	variabile, costante	Moltiplicazione
DIV	(	variabile, costante	Divisione
GT	(	variabile, costante	Controllo: >
GE	(	variabile, costante	Controllo: >=
EQ	(	variabile, costante	Controllo: =
LE	(	variabile, costante	Controllo <=
LT	(	variabile, costante	Controllo <
NE	(	variabile, costante	Controllo <>
CAL	C N	Istanza di blocco funzione	Chiamata a blocco funzione
JMP	C N	Nome dell'istanza	Salto a etichetta
RET	C N	Etichetta	Ritorna da sottoprogramma
)			Esegue operazioni rinviate

Nella seguente sezione, sono descritti solo gli operatori specifici del linguaggio. Per gli altri operatori standard, si rimanda alla sezione «Operatori, blocchi funzione e funzioni standard» di questo manuale.

B.8.2.1 Operatore LD

Operazione: carica un valore nel risultato corrente

Modificatori permessi: N

Operando: espressione costante
variabile interna, di input o output

Esempio:

```
                        (* ESEMPI DI OPERAZIONI LD *)
LDex:       LD          false       (* risultato := costante booleana FALSE *)
            LD          true        (* risultato := costante booleana TRUE *)
            LD          123         (* risultato := costante intera *)
            LD          123.1       (* risultato := costante reale *)
            LD          t#3ms       (* risultato := costante temporale *)
            LD          boo_var1    (* risultato := variabile booleana *)
            LD          ana_var1    (* risultato := variabile analogica *)
            LD          tmr_var1    (* risultato := variabile temporale *)
            LDN         boo_var2    (* risultato := NOT ( variabile booleana ) *)
```

B.8.2.2 Operatore ST

Operazione: memorizza in una variabile il risultato corrente
questo operatore non modifica il risultato corrente

Modificatori permessi: N

Operando: una variabile interna o di output

Esempio:

```
                        (* ESEMPI DI OPERAZIONI ST *)
STboo:      LD          false
            ST          boo_var1    (* boo_var1 := FALSE *)
            STN         boo_var2    (* boo_var2 := TRUE *)
STana:      LD          123
            ST          ana_var1    (* ana_var1 := 123 *)
STtmr:      LD          t#12s
            ST          tmr_var1    (* tmr_var1 := t#12s *)
```

B.8.2.3 Operatore S

Operazione: assegna il valore booleano TRUE ad una variabile booleana se il risultato corrente ha valore booleano TRUE. Non viene eseguita alcuna operazione se il risultato corrente vale FALSE. Questa operazione non modifica il risultato corrente.

Modificatori permessi: (nessuno)

Operando: variabile booleana di output o interna

Esempio:

```
                        (* ESEMPI DI OPERAZIONI S *)
SETex:      LD          true        (* risultato corrente:= TRUE *)
            S           boo_var1    (* boo_var1 := TRUE *)
                                    (* il risultato corrente non viene modificato *)
            LD          false       (* risultato corrente:= FALSE *)
            S           boo_var1    (* niente di fatto - boo_var1 invariata *)
```

B.8.2.4 Operatore R

Operazione: assegna il valore booleano FALSE ad una variabile booleana, se il risultato corrente ha il valore booleano TRUE. Non viene eseguita alcuna operazione se il risultato corrente vale FALSE. Questa operazione non modifica il risultato corrente.

Modificatori permessi: (nessuno)

Operando: variabile booleana di output o interna

Esempio:

```
                        (* ESEMPI DI OPERAZIONI R *)
RESETex:    LD          true        (* risultato corrente:= TRUE *)
            R           boo_var1    (* boo_var1 := FALSE *)
                                    (*il risultato corrente non viene modificato *)
            ST          boo_var2    (* boo_var2 := TRUE *)
            LD          false       (*risultato corrente := FALSE *)
```

```
        R       boo_var1    (*niente di fatto - boo_var1 invariata *)
```

B.8.2.5 Operatore JMP

Operazione: salta all'etichetta specificata

Modificatori permessi: C N

Operando: una etichetta definita nello stesso programma IL

Esempio:

```
(* il seguente esempio controlla il valore di un selettore analogico (0 o 1 o 2) *)
(* per impostare uno di tre output booleani. Il controllo "è uguale a 0" viene eseguito*)
(* tramite l'operatore JMPC                                                          *)

JMPex:  LD      selector    (* selector vale 0 o 1 o 2 *)
        BOO                 (* conversione a booleano *)
        JMPC    test1       (* se selector = 0 allora *)
        LD      true
        ST      bo0         (* bo0 := true *)
        JMP     JMPend      (* fine del programma *)
test1:  LD      selector
        SUB     1           (* decrementa selector: adesso vale 0 o 1 *)
        BOO                 (*conversione a booleano *)
        JMPC    test2       (* se selector = 0 allora *)
        LD      true
        ST      bo1         (* bo1 := true *)
        JMP     JMPend      (* fine del programma *)
test2:  LD      true        (* ultimo caso *)
        ST      bo2         (* bo2 := true *)
JMPend:                     (* fine del programma IL *)
```

B.8.2.6 operatore RET

Operazione: termina 'esecuzione della lista corrente di istruzioni. Qualora la sequenza IL sia un sottoprogramma, il risultato corrente viene restituito al programma chiamante

Modificatori permessi: C N

Operando: (nessuno)

Esempio:

```
(* il seguente esempio controlla il valore di un selettore analogico (0 o 1 o 2) *)
(* per impostare uno di tre output booleani. Il controllo "è uguale a 0" viene eseguito*)
(* tramite l'operatore JMPC                                                          *)

JMPex:  LD      selector    (* selector vale 0 o 1 o 2 *)
```

```
                BOO             (*conversione a booleano *)
                JMPC    test1   (* if selector = 0 then *)
                LD      true
                ST      bo0     (* bo0 := true *)
                RET             (* fine - ritorna 0 *)
                                (* decrementa selector *)
test1:          LD      selector
                SUB     1       (* selector vale ora 0 o 1 *)
                BOO             (*conversione a booleano *)
                JMPC    test2   (* se selector = 0 allora *)
                LD      true
                ST      bo1     (* bo1 := true *)
                LD      1       (*carica il valore reale del selettore *)
                RET             (* fine - ritorna 1 *)
                                (* utlimo caso *)
test2:          RETNC           (* ritorna se selector *)
                                (* ha valore non valido *)
                LD      true
                ST      bo2     (* bo2 := true *)
                LD      2       (* carica il valore reale del selettore *)
                                (*fine- ritorna 2 *)
```

B.8.2.7 Operatore ")"

Operazione: esegue un'operazione rinviata. L'operazione rinviata è stata notificata con l'operatore '('

Modificatori permessi: (nessuno)

Operando: (nessuno)

Esempio:

```
(* Il seguente programma alterna operazioni rinviate: *)
(* res := a1 + (a2 * (a3 - a4) * a5) + a6; *)

Delayed:        LD      a1      (* risultato := a1; *)
                ADD(    a2      (* rinviata ADD - result := a2; *)
                MUL(    a3      (*rinviata MUL - result := a3; *)
                SUB     a4      (* risultato := a3 - a4; *)
                )               (* esegue MUL rinviata - risultato := a2 * (a3-a4); *)
                MUL     a5      (* risultato := a2 * (a3 - a4) * a5; *)
                )               (* esegue ADD rinviata*)
                                (* risultato := a1 + (a2 * (a3 - a4) * a5); *)
                ADD     a6      (* risultato := a1 + (a2 * (a3 - a4) * a5) + a6; *)
                ST      res     (* assegna alla variabile res il risultato corrente *)
```

B.8.2.8 Chiamare sottoprogrammi o funzioni

Un sottoprogramma o una funzione (scritta in un qualunque linguaggio IL, ST, LD, FBD o "C") può essere richiamata dal linguaggio IL usando il nome come operatore.

Operazione: esegue un sottoprogramma o una funzione – il valore restituito dal sottoprogramma o funzione è assegnato al risultato corrente IL

Modificatori permessi: (nessuno)

Operando: Il **primo** parametro di chiamata deve essere assegnato al risultato corrente prima della chiamata. I **seguenti** vengono espressi nel campo dell'operando, separati da virgole.

Esempio:

```
(* Programma chiamante: converte un valore analogico in valore temporale *)

Main:   LD      bi0
        SUBPRO  bi1,bi2   (* chiama sottoprog. per avere valore analogico *)
        ST      result    (* risultato := valore restituito dal sottoprog. *)
        GT      vmax      (* controllo overflow valore*)
        RETC              (* ritorno se overflow *)
        LD      result
        MUL     1000      (* converte secondi in millisecondi *)
        TMR               (* converte a timer *)
        ST      tmval     (* assegna valore convertito ad un timer *)

(* 'SUBPRO': valuta il valore analogico *)
(* passato come valore binario su tre input booleani: in0, in1, in2 sono i tre parametri di input
del sottoprogramma *)
        LD      in2
        ANA               (* risultato = ana (in2); *)
        MUL     2         (* risultato := 2*ana (in2); *)
        ST      temporary (* temporary := risultato *)
        LD      in1
        ANA
        ADD     temporary (*risultato := 2*ana (in2) + ana (in1); *)
        MUL     2         (*risultato := 4*ana (in2) + 2*ana (in1); *)
        ST      temporary (* temporary := risultato *)
        LD      in0
        ANA
        ADD     temporary (*risultato := 4*ana (in2) + 2*ana (in1)+ana (in0); *)
        ST      SUBPRO    (* ritorna al programma chiamante il risultato corrente
*)
```

B.8.2.9 Chiamare blocchi funzione: operatore CAL

Operazione: chiama un blocco funzione

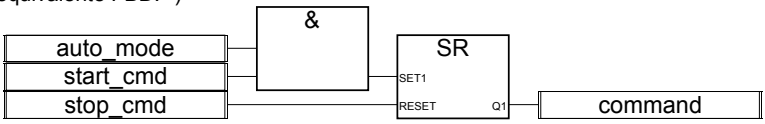
Modificatori permessi: C N

Operando: Nome dell'istanza del blocco funzione.
i parametri di input dei blocchi devono essere assegnati prima della chiamata, usando una sequenza di operazioni LD/ST.
I parametri di output sono accessibili se usati.

Esempio1:

```
(* Chiamata al blocco funzione SR : SR1 è un'istanza di SR *)
LD      auto_mode
AND     start_cmd
ST      SR1.set1
LD      stop_cmd
ST      SR1.reset
CAL     SR1
LD      SR1.Q1
ST      command
```

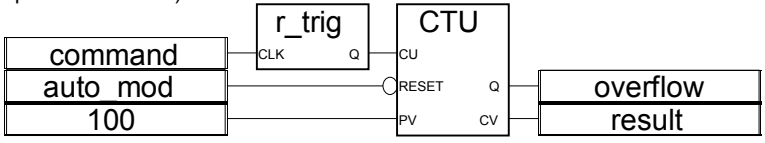
(* equivalente FBD: *)



Esempio 2

```
(*Si suppone R_TRIG1 istanza del blocco R_TRIG e CTU1 istanza del blocco CTU *)
LD      command
ST      R_TRIG1.clk
CAL     R_TRIG1
LD      R_TRIG1.Q
ST      CTU1.cu
LDN     auto_mode
ST      CTU1.reset
LD      100
ST      CTU1.pv
CAL     CTU1
LD      CTU1.Q
ST      overflow
LD      CTU1.cv
ST      result
```

(* equivalente FBD: *)



B.9 Operatori, blocchi funzione e funzioni standard

B.9.1 Operatori standard

I seguenti sono operatori standard dei linguaggi IEC.

- Manipolazione dei dati.....Assegnazione, Negazione analogica
- Operazioni booleane .....AND booleano
  OR booleano
  OR esclusivo booleano
- Operazioni aritmetiche .....Addizione
  Sottrazione
  Moltiplicazione
  Divisione
- Operazioni logiche.....Mascheramento analogico AND bit a bit
  Mascheramento analogico OR bit a bit
  Mascheramento analogico OR esclusivo bit a bit
  Negazione bit a bit
- Test di comparazione.....Minore di
  Minore o uguale a
  Maggiore di
  Maggiore o uguale a
  Uguale a
  Non uguale a
- Conversione dati .....Conversione a Booleano
  Conversione ad Intero Analogico
  Conversione a Reale Analogico
  Conversione a Timer
  Conversione a Messaggio
- Altri.....Concatenazione di messaggi
  Accesso al sistema
  Operazioni su canali I/O

1 gain

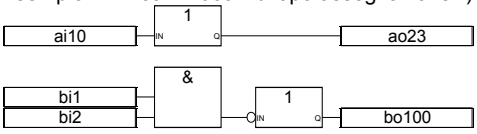
Argomenti:

IN	qualsiasi tipo
Q	qualsiasi tipo

Descrizione:
assegnazione di una variabile ad un'altra

Questo blocco risulta molto utile per collegare direttamente l'input di un diagramma all'output di un altro diagramma. Può anche essere usato (con una linea di negazione booleana) per invertire lo stato di una linea connessa all'output di un diagramma.

(*Esempio FBD con Blocchi di tipo assegnazione *)

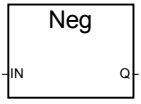


(* equivalente ST: *)
ao23 := ai10;
bo100 := NOT (bi1 AND bi2);

(*equivalente IL: *)

LD	ai10
ST	ao23
LD	bi1
AND	bi2
STN	bo100

NEG



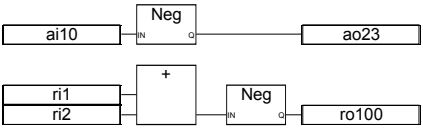
Argomenti:

IN	INT-REALE	input ed output devono avere lo stesso formato
Q	INT-REALE	

Descrizione:

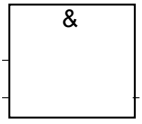
Assegnazione della negazione di una variabile.

(*Esempio FBD con Blocchi di tipo negazione *)




```
(* equivalente ST: *)
ao23 := - (ai10);
ro100 := - (ri1 + ri2);

(* equivalente IL: *)
LD          ai10
MUL         -1
ST          ao23
LD          ri1
ADD         ri2
MUL         -1.0
ST          ro100
```

& AND



Nota: Per questo operatore, il numero degli input può essere esteso a più di due.

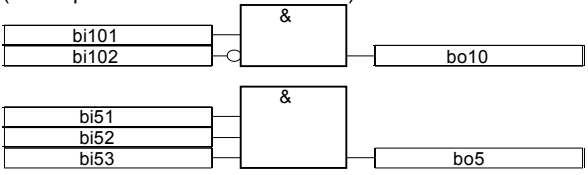
Argomenti:

input	BOOLEANO	
output	BOOLEANO	AND booleano dei termini in input

Descrizione:

AND booleano tra due o più termini.

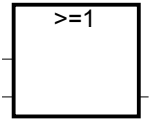
(*Esempio FBD con Blocchi "AND" *)




```
(* equivalente ST: *)
bo10 := bi101 AND NOT (bi102);
bo5 := (bi51 AND bi52) AND bi53;
```

```
(*equivalente IL *)
            LD          bi101       (* risultato corrente := bi101 *)
            ANDN        bi102       (* risultato corrente := bi101 AND not (bi102) *)
            ST          bo10        (* bo10 := risultato corrente *)
            LD          bi51        (* risultato corrente := bi51;
            &           bi52        (* risultato corrente := bi51 AND bi52 *)
            &           bi53        (* risultato corrente := (bi51 AND bi52) AND bi53
*)
            ST          bo5         (* bo5 := risultato corrente *)
```

>=1 OR



Nota: Per questo operatore, il numero degli input può essere esteso a più di due.

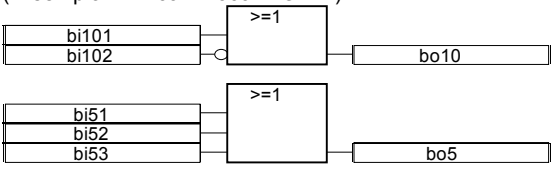
Argomenti:

input	BOOLEANO
output	BOOLEANO OR booleano dei termini di input

Descrizione:

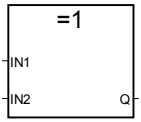
OR booleano di due o più termini.

(*Esempio FBD con Blocchi "OR" *)




```
(* Equivalente ST: *)
bo10 := bi101 OR NOT (bi102);
bo5 := (bi51 OR bi52) OR bi53;

(*Equivalente IL: *)
LD          bi101
ORN         bi102
ST          bo10
LD          bi51
OR          bi52
OR          bi53
ST          bo5
```

=1 XOR

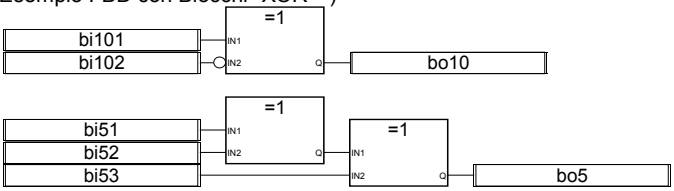


Argomenti:

IN1	BOOLEANO	
IN2	BOOLEANO	
Q	BOOLEANO	OR esclusivo booleano dei due termini di input

Descrizione:
OR esclusivo booleano tra due termini.

(* Esempio FBD con Blocchi "XOR" *)

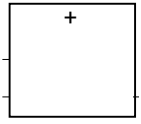


(* Equivalente ST: *)
bo10 := bi101 XOR NOT (bi102);
bo5 := (bi51 XOR bi52) XOR bi53;

(* Equivalente IL: *)

LD	bi101
XORN	bi102
ST	bo10
LD	bi51
XOR	bi52
XOR	bi53
ST	bo5

+



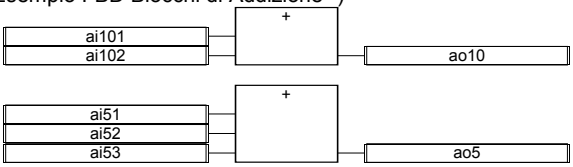
Nota: Per questo operatore, il numero degli input può essere esteso a più di due.

Argomenti:

input	INT-REALE	può essere di tipo INTERO o REALE (tutti gli input devono avere lo stesso formato)
output	INT-REALE	addizione con segno dei termini di input

Descrizione:
Addizione di due o più variabili analogiche.

(*Esempio FBD Blocchi di Addizione *)



(*equivalente ST: *)
ao10 := ai101 + ai102;
ao5 := (ai51 + ai52) + ai53;

(*equivalente IL: *)

LD	ai101
ADD	ai102
ST	ao10
LD	ai51
ADD	ai52
ADD	ai53
ST	ao5

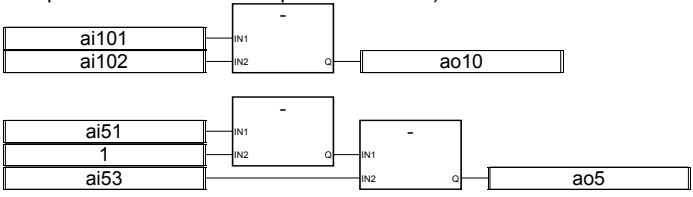
-

Argomenti:

IN1	INT-REALE	può essere INTERO o REALE
IN2	INT-REALE	(IN1 e IN2 devono avere lo stesso formato)
Q	INT-REALE	sottrazione (primo - secondo)

Descrizione:
Sottrazione tra due variabili analogiche (primo - secondo).

(*Esempio FBD con Blocchi di tipo Sottrazione *)



(* Equivalente ST: *)
ao10 := ai101 - ai102;

```
ao5 := (ai51 - 1) - ai53;

(* Equivalente IL: *)
LD          ai101
SUB         ai102
ST          ao10
LD          ai51
SUB         1
SUB         ai53
ST          ao5
```

*

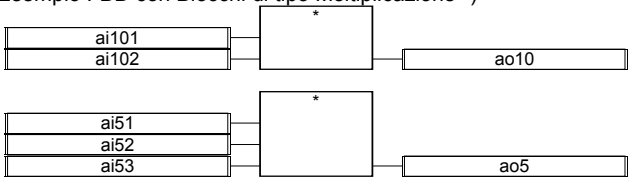
Nota: Per questo operatore, il numero degli input può essere esteso a più di due.

Argomenti:

input	INT-REALE	può essere di tipo INTERO o REALE (tutti gli input devono avere lo stesso formato)
output	INT-REALE	moltiplicazione con segno dei termini di input

Descrizione:
Moltiplicazione di due o più variabili analogiche.

(* Esempio FBD con Blocchi di tipo Moltiplicazione *)




```
(* Equivalente ST *)
ao10 := ai101 * ai102;
ao5 := (ai51 * ai52) * ai53;

(* Equivalente IL: *)
LD          ai101
MUL         ai102
ST          ao10
LD          ai51
MUL         ai52
MUL         ai53
ST          ao5
```

/

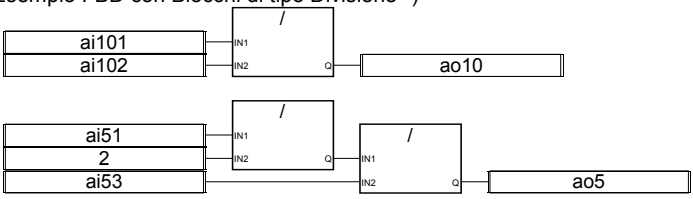
Argomenti:

IN1	INT-REALE	può essere di tipo INTERO o REALE (operando)
IN2	INT-REALE	valore analogico diverso da 0 (divisore) (IN1 e IN2 devono avere lo stesso formato)
Q	INT-REALE	divisione con segno intera o reale tra IN1 e IN2

Descrizione:

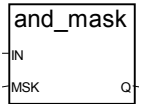
Divisione di due variabili analogiche (la prima diviso la seconda).

(*Esempio FBD con Blocchi di tipo Divisione *)




```
(* Equivalente ST: *)
ao10 := ai101 / ai102;
ao5 := (ai51 / 2) / ai53;

(* Equivalente IL: *)
LD      ai101
DIV     ai102
ST      ao10
LD      ai51
DIV     2
DIV     ai53
ST      ao5
```

AND_MASK



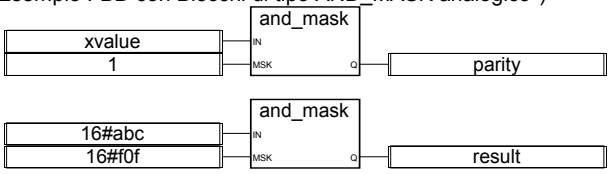
Argomenti:

IN	INT	deve avere formato intero
MSK	INT	deve avere formato intero

Q	INT	AND logico bit a bit tra IN e MSK

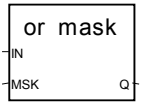
Descrizione:
Mascheramento intero analogico AND bit a bit.

(* Esempio FBD con Blocchi di tipo AND_MASK analogico*)




```
(* Equivalente ST: *)
parity := AND_MASK (xvalue, 1); (* 1 se xvalue è dispari *)
result := AND_MASK (16#abc, 16#f0f); (* uguale a 16#a0c *)

(* Equivalente IL: *)
LD          xvalue
AND_MASK    1
ST          parity
LD          16#abc
AND_MASK    16#f0f
ST          result
```

OR_MASK

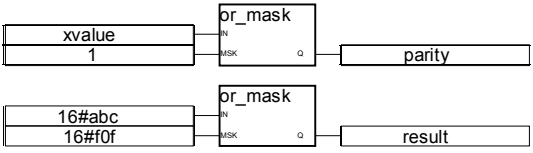


Argomenti:

IN	INT	deve avere formato intero
MSK	INT	deve avere formato intero
Q	INT	OR logico bit a bit tra IN e MSK

Descrizione:
Mascheramento intero analogico OR bit a bit.

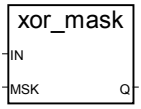
(* Esempio FBD con Blocchi di tipo OR_MASK analogico*)




```
(* Equivalente ST: *)
is_odd := OR_MASK (xvalue, 1); (* rende sempre dispari il valore *)
```

```
result := OR_MASK (16#abc, 16#f0f); (* uguale a 16#fbf *)

(* Equivalente IL: *)
LD          xvalue
OR_MASK     1
ST          is_odd
LD          16#abc
OR_MASK     16#f0f
ST          result
```

XOR_MASK

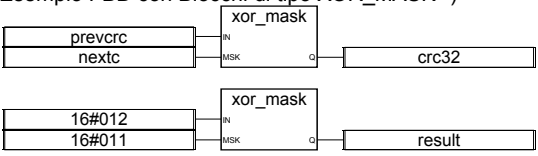


Argomenti:

IN	INT	deve avere formato intero
MSK	INT	deve avere formato intero
Q	INT	OR esclusivo logico bit a bit tra IN e MSK

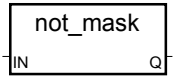
Descrizione:
Mascheramento intero analogico OR esclusivo bit a bit

(* Esempio FBD con Blocchi di tipo XOR_MASK *)




```
(* Equivalente ST: *)
crc32 := XOR_MASK (prevcrc, nextc);
result := XOR_MASK (16#012, 16#011); (* uguale a 16#003 *)

(* Equivalente IL: *)
LD          prevcrc
XOR_MASK    nextc
ST          crc32
LD          16#012
XOR_MASK    16#011
ST          result
```

NOT_MASK

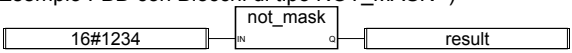


Argomenti :

IN	INT	deve avere formato intero
Q	INT	negazione bit a bit a 32 bit di IN

Descrizione:
Mascheramento negato intero analogico bit a bit.

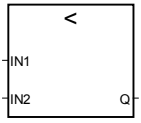
(* Esempio FBD con Blocchi di tipo NOT_MASK *)



(*Equivalente ST: *)
result := NOT_MASK (16#1234);
(* result is 16#FFFF_EDCB *)

(* Equivalente IL: *)
LD 16#1234
NOT_MASK
ST result

<

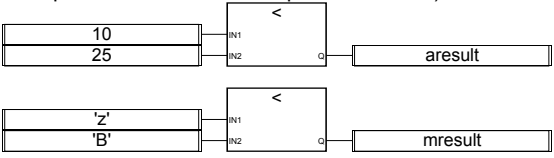


Argomenti:

IN1	INT-REALE-TMR-MSG	
IN2	INT-REALE-TMR-MSG	entrambi gli input devono essere dello stesso tipo
Q	BOOLEANO	TRUE se IN1 < IN2

Descrizione:
Controlla se un valore sia MINORE DI un altro (per tipo analogico, timer o messaggio)

(* Esempio FBD con Blocchi di tipo "Minore di" *)

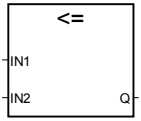


(* Equivalente ST: *)
aresult := (10 < 25); (* aresult vale TRUE *)
mresult := ('z' < 'B'); (* mresult vale FALSE *)

(* Equivalente IL: *)

```
LD      10
LT      25
ST      aresult
LD      'z'
LT      'B'
ST      mresult
```

<=



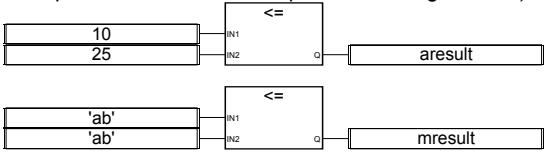
Argomenti:

IN1	INT-REALE-MSG
IN2	INT-REALE-MSGentrambi gli input devono essere dello stesso tipo
Q	BOOLEANO TRUE se IN1 <= IN2

Descrizione:

Controlla se un valore sia MINORE DI o UGUALE A un altro (per tipo analogico o messaggio)

(* Esempio FBD con Blocchi di tipo "Minore o uguale a" *)




```
(* Equivalente ST: *)
aresult := (10 <= 25); (* aresult vale TRUE *)
mresult := ('ab' <= 'ab'); (* mresult vale TRUE *)
```

```
(* Equivalente IL: *)
LD      10
LE      25
ST      aresult
LD      'ab'
LE      'ab'
ST      mresult
```

>

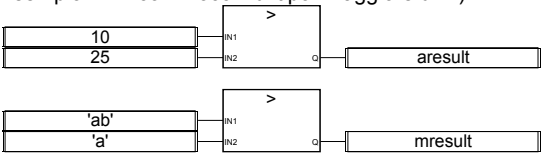
Argomenti:

IN1	INT-REALE-TMR-MSG	
IN2	INT-REALE-TMR-MSG	entrambi gli input devono essere dello stesso tipo
Q	BOOLEANO	TRUE se IN1 > IN2

Descrizione:

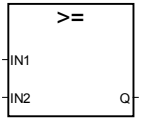
Controlla se un valore sia MAGGIORE DI un altro (per tipo analogico, timer o messaggio)

(* Esempio FBD con Blocchi di tipo "Maggiore di" *)




```
(* Equivalente ST: *)
aresult := (10 > 25); (* aresult vale FALSE *)
mresult := ('ab' > 'a'); (* mresult vale TRUE *)

(* Equivalente IL: *)
LD      10
GT      25
ST      aresult
LD      'ab'
GT      'a'
ST      mresult
```

>=



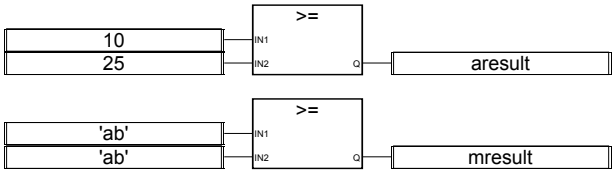
Argomenti:

IN1	INT-REALE-MSG	
IN2	INT-REALE-MSG	entrambi gli input devono essere dello stesso tipo
Q	BOOLEANO	TRUE se IN1 >= IN2

Descrizione:

Controlla se un valore sia MAGGIORE DI o UGUALE A un altro (per tipo analogico o messaggio)

(* Esempio FBD con Blocchi di tipo "Maggiore o uguale a " *)




```
(* Equivalente ST: *)
aresult := (10 >= 25); (* aresult vale FALSE *)
mresult := ('ab' >= 'ab'); (* mresult vale TRUE *)

(* Equivalente IL: *)
LD          10
GE          25
ST          aresult
LD          'ab'
GE          'ab'
ST          mresult
```

=

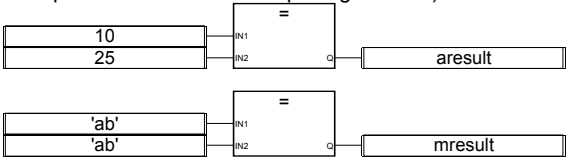
Argomenti:

IN1	INT-REALE-MSG	
IN2	INT-REALE-MSG	entrambi gli input devono essere dello stesso tipo
Q	BOOLEANO TRUE se IN1 = IN2	

Descrizione:

Controlla se un valore sia UGUALE A un altro (per tipo analogico o messaggio)

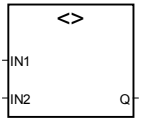
(* Esempio FBD con Blocchi di tipo "Uguale a" *)




```
(* Equivalente ST: *)
aresult := (10 = 25); (* aresult vale FALSE *)
mresult := ('ab' = 'ab'); (* mresult vale TRUE *)

(* Equivalente IL: *)
LD          10
```

```
EQ          25
ST          aresult
LD          'ab'
EQ          'ab'
ST          mresult
```

<>

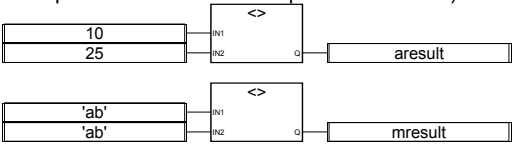


Argomenti:

IN1	INT-REALE-MSG	
IN2	INT-REALE-MSG	entrambi gli input devono essere dello stesso tipo
Q	BOOLEANO TRUE se primo <> secondo	

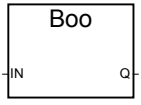
Descrizione:
Controlla se un valore sia DIVERSO DA un altro (per tipo analogico o messaggio)

(* Esempio FBD con Blocchi di tipo "Diverso da" *)




```
(* Equivalente ST: *)
aresult := (10 <> 25); (* aresult vale TRUE *)
mresult := ('ab' <> 'ab'); (* mresult vale FALSE *)

(* Equivalente IL: *)
LD          10
NE          25
ST          aresult
LD          'ab'
NE          'ab'
ST          mresult
```

BOO

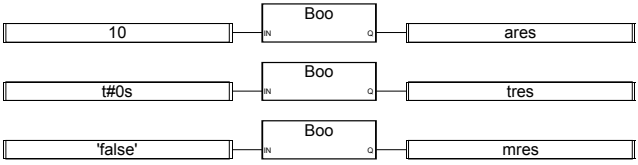


Argomenti:

IN	ANY	qualsiasi valore non booleano
Q	BOO	TRUE per valori numerici diversi da zero FALSE per valori numerici uguali a zero TRUE per il messaggio 'TRUE' FALSE per il messaggio 'FALSE'

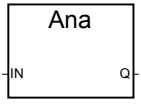
Descrizione:
Converte una variabile da un qualsiasi tipo a tipo booleano

(* Esempio FBD con Blocchi di tipo "Conversione a booleano " *)




```
(* Equivalente ST: *)
ares := BOO (10);                (* ares vale TRUE *)
tres := BOO (t#0s);              (* tres vale FALSE *)
mres := BOO ('false');           (* mres vale FALSE *)

(* Equivalente IL: *)
LD          10
BOO
ST          ares
LD          t#0s
BOO
ST          tres
LD          'false'
BOO
ST          mres
```

ANA

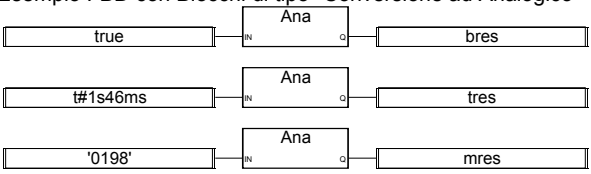


Argomenti:

IN	ANY	qualsiasi valore analogico non intero
Q	INT	0 se IN vale FALSE / 1 se IN vale TRUE numero di millisecondi nel caso di un timer parte intera nel caso di reale analogico numero decimale rappresentato da una stringa

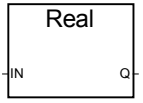
Descrizione:
Converte una variabile da un qualsiasi tipo a tipo intero

(* Esempio FBD con Blocchi di tipo "Conversione ad Analogico" *)




```
(* Equivalente ST: *)
bres := ANA (true);                 (* bres vale 1 *)
tres := ANA (t#1s46ms);             (* tres vale 1046 *)
mres := ANA ('0198');               (* mres vale 198 *)

(* Equivalente IL: *)
LD          true
ANA
ST          bres
LD          t#1s46ms
ANA
ST          tres
LD          '0198'
ANA
ST          mres
```

REAL



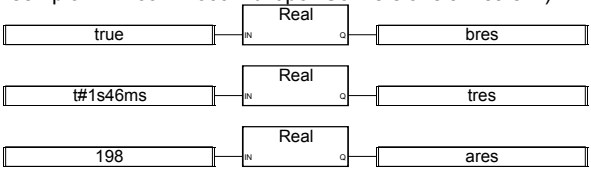
Argomenti:

IN	BOO-INT-TMR	qualsiasi valore di tipo diverso da reale analogico o messaggio
Q	REAL	0.0 se IN vale FALSE / 1.0 se IN vale TRUE numero di millisecondi nel caso di un timer il numero equivalente nel caso di intero analogico

Descrizione:

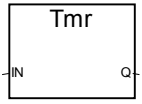
Converte una variabile da un qualsiasi tipo a tipo reale

(* Esempio FBD con Blocchi di tipo "Conversione a Reale" *)



```
(* Equivalente ST: *)
bres := REAL (true);                (* bres vale 1.0 *)
tres := REAL (t#1s46ms);            (* tres vale 1046.0 *)
ares := REAL (198);                 (* ares vale 198.0 *)

(* Equivalente IL: *)
LD          true
REAL
ST          bres
LD          t#1s46ms
REAL
ST          tres
LD          198
REAL
ST          ares
```

TMR



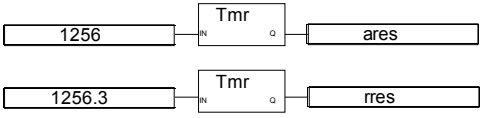
Argomenti:

IN	INT-REALE	qualsiasi valore non temporale IN (o la parte intera di IN nel caso sia di tipo reale) rappresenta il numero di millisecondi
Q	TIMER	valore temporale rappresentato da IN

Descrizione:

Converte una variabile di tipo analogico a tipo timer

(* Esempio FBD con Blocchi di tipo "Conversione a Timer" *)




```
 (* Equivalente ST: *)
ares := TMR (1256);                 (* ares := t#1s256ms *)
rres := TMR (1256.3);               (*rres := t#1s256ms *)

(* Equivalente IL: *)
LD          1256
TMR
ST          ares
LD          1256.3
TMR
ST          rres
```

MSG

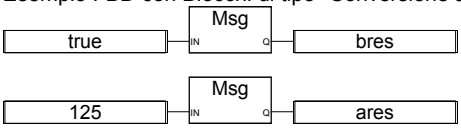
Argomenti:

IN	BOO-	
	INT-REA	qualsiasi valore che non sia un messaggio
Q	MSG	'false' o 'true' se IN è di tipo booleano
		rappresentazione decimale se IN è di tipo analogico

Descrizione:

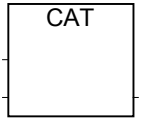
Converte una variabile in messaggio

(* Esempio FBD con Blocchi di tipo "Conversione a Messaggio" *)




```
(* Equivalente ST: *)
bres := MSG (true); (* bres vale 'TRUE' *)
ares := MSG (125); (* ares vale '125' *)

(* Equivalente IL: *)
LD          true
MSG
ST          bres
LD          125
MSG
ST          ares
```

CAT



Nota: Per questo operatore, il numero degli input può essere esteso a più di due.

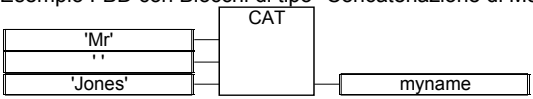
Argomenti:

input	MSG	(la lunghezza complessiva dei messaggi non deve eccedere la capacità del messaggio in output)
output	MSG	concatenazione dei messaggi in input

Descrizione:

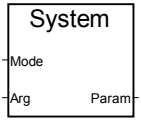
Concatena più messaggi in uno

(* Esempio FBD con Blocchi di tipo "Concatenazione di Messaggi" *)




```
(* Equivalente ST: si usa l'operatore + *)
myname := ('Mr' + ' ') + 'Jones';
(*equivale a: myname := 'Mr Jones' *)

(* Equivalente IL: *)
LD          'Mr'
ADD         ' '
ADD         'Jones'
ST          myname
```

SYSTEM



Argomenti:

Mode	INT	rappresenta il parametro di sistema ed il modo di accesso
Arg	INT-TMR	nuovo valore per un accesso "in scrittura"
Param	INT	valore del parametro cui si accede

Descrizione:

Accesso ai parametri di sistema

La seguente lista elenca i comandi disponibili (parole chiave predefinite) per la funzione SYSTEM:

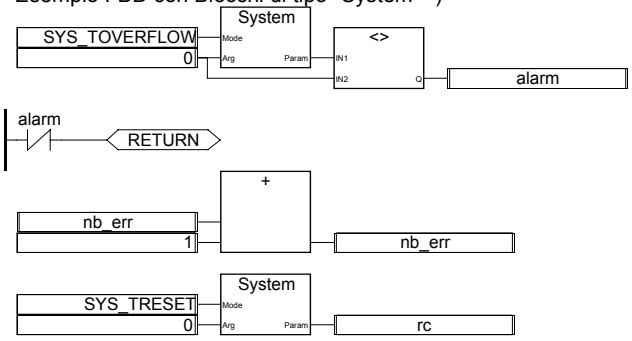
Comando	Descrizione
SYS_TALLOWED	lettura del tempo ciclo consentito
SYS_TCURRENT	lettura del tempo ciclo corrente
SYS_TMAXIMUM	lettura del tempo ciclo massimo
SYS_TOVERFLOW	lettura overflow tempo ciclo
SYS_TRESET	azzeramento contatori temporali
SYS_TWRITE	modifica tempo ciclo
SYS_ERR_TEST	controllo errori in esecuzione
SYS_ERR_READ	lettura errore iniziale in esecuzione

Argomenti richiesti per le funzioni predefinite della funzione SYSTEM:

comando	argomento	Valore di ritorno

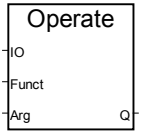
SYS_TALLOWED	0	tempo ciclo consentito
SYS_TCURRENT	0	tempo ciclo corrente
SYS_TMAXIMUM	0	timing massimo rilevato
SYS_TOVERFLOW	0	numero di overflow timing
SYS_TRESET	0	0
SYS_TWRITE	nuovo tempo ciclo consentito	tempo accettato
SYS_ERR_TEST	0	0 nel caso non sia stato rilevato alcun errore
SYS_ERR_READ	0	codice di errore iniziale

(* Esempio FBD con Blocchi di tipo "System" *)




```
(* Equivalente ST: *)
alarm := (SYSTEM (SYS_TOVERFLOW, 0) <> 0);
If (alarm) Then
        nb_err := nb_err + 1;
        rc := SYSTEM (SYS_TRESET, 0);
End_If;
```

OPERATE



Argomenti:

IO	ANY	variabile di input o output
Funct	INT	azione da intraprendere
Arg	INT	argomento per l'azione di I/O
Q	INT	controllo di ritorno

Descrizione:

Accesso ad un canale di I/O

Il significato degli argomenti di OPERATE dipende dall'implementazione dell'interfaccia I/O. Per maggiori ragguagli sulle possibilità di OPERATE, si veda il manuale relativo alla parte hardware o le note tecniche della scheda I/O.

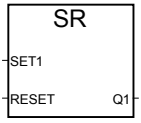
B.9.2 Blocchi funzione standard

Questi sono i blocchi funzione standard supportati dal sistema ISaGRAF. Sono predefiniti e non devono essere dichiarati nella libreria.

● Booleani	SR	Bistabile con set dominante
	RS	Bistabile con reset dominante
	R_Trig	Riconoscimento di fronte di salita
	F_Trig	Riconoscimento di fronte di discesa
	SEMA	Semaforo
● Contatore	CTU	Contatore ad incremento
	CTD	Contatore a decremento
	CTUD	Contatore ad incremento - decremento
● Temporizzatori	TON	Temporizzatore con ritardo di attivazione
	TOF	Temporizzatore con ritardo di disattivazione
	TP	Temporizzatore di impulsi
● Interi analogici	CMP	Blocco funzione di comparazione completa
	StackInt	Stack di interi analogici
● Analogici di tipo reale	AVERAGE	Media mobile su N campioni
	HYSTER	Isteresi booleana su differenza di dati di tipo reale
	LIM_ALRM	Allarme di limite superiore/inferiore con isteresi
	INTEGRAL	Integrazione rispetto al tempo
	DERIVATE	Derivazione rispetto al tempo
● Generatore di segnali	BLINK	Segnale booleano intermittente
	SIG_GEN	Generatore di segnale

Nota: Nuovi blocchi funzione creati in "C", possono essere richiamati dal linguaggio FBD.

SR



Argomenti:

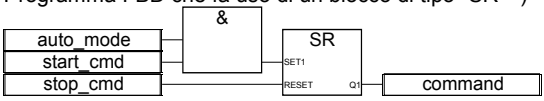
SET1	BOO	se TRUE, imposta Q1 a TRUE (dominante)
RESET	BOO	se TRUE, imposta Q1 a FALSE
Q1	BOO	memoria di stato booleano

Descrizione:

Bistabile con set dominante: Si veda la seguente "tavola di verità":

Set1	Reset	Q1	risultato Q1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

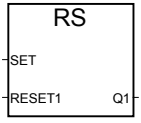
(* Programma FBD che fa uso di un blocco di tipo "SR" *)




```
(* Equivalente ST: Si assuma SR1 istanza del blocco SR *)
SR1((auto_mode & start_cmd), stop_cmd);
command := SR1.Q1;
```

```
(* Equivalente IL: *)
LD          auto_mode
AND         start_cmd
ST          SR1.set1
LD          stop_cmd
ST          SR1.reset
CAL         SR1
LD          SR1.Q1
ST          command
```

RS



Argomenti:

SET	BOO	se TRUE, imposta Q1 a TRUE
RESET1	BOO	se TRUE, imposta Q1 a FALSE (dominante)
Q1	BOO	memoria di stato booleano

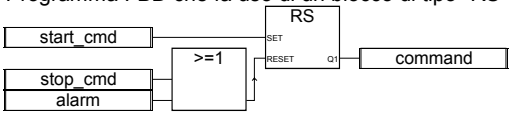
Descrizione:

Bistabile con reset dominante: Si veda la seguente “tavola di verità”:

Set	Reset1	Q1	result Q1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1

1	0	1	1
1	1	0	0
1	1	1	0

(* Programma FBD che fa uso di un blocco di tipo "RS" block *)

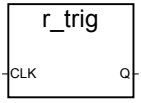


(* Equivalente ST: Si assuma RS1 istanza del blocco RS*)
RS1(start_cmd, (stop_cmd OR alarm));
command := RS1.Q1;

(* Equivalente IL: *)

LD	start_cmd
ST	RS1.set
LD	stop_cmd
OR	alarm
ST	RS1.reset1
CAL	RS1
LD	RS1.Q1
ST	command

R_TRIG

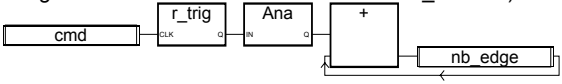


Argomenti:

CLK	BOO	qualsiasi variabile booleana
Q	BOO	TRUE quando CLK sale da FALSE a TRUE FALSE in tutti gli altri casi

Descrizione:
Rileva un fronte di salita di una variabile booleana

(* Programma FBD che fa uso di un blocco "R_TRIG" *)



(* Equivalente ST: Si assuma R_TRIG1 istanza del blocco R_TRIG *)
R_TRIG1(cmd);
nb_edge := ANA(R_TRIG1.Q) + nb_edge;

(* Equivalente IL: *)

LD	cmd
ST	R_TRIG1.clk

```
CAL     R_TRIG1
LD      R_TRIG1.Q
ANA
ADD     nb_edge
ST      nb_edge
```

F_TRIG

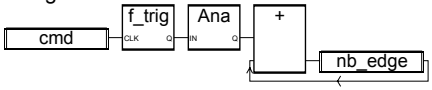
Argomenti:

CLK	BOO	qualsiasi variabile booleana
Q	BOO	TRUE quando CLK cambia da TRUE a FALSE FALSE in tutti gli altri casi

Descrizione:

Rileva un fronte di discesa di una variabile booleana

(* Programma FBD che fa uso di un blocco di tipo "F_TRIG" *)




```
(* Equivalente ST: Si assuma F_TRIG1 istanza del blocco F_TRIG *)
F_TRIG1(cmd);
nb_edge := ANA(F_TRIG1.Q) + nb_edge;

(* Equivalente IL: *)
LD      cmd
ST      F_TRIG1.clk
CAL     F_TRIG1
LD      F_TRIG1.Q
ANA
ADD     nb_edge
ST      nb_edge
```

SEMA

Argomenti:

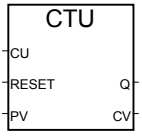
CLAIM	BOOLEANO	comando di "controllo ed impostazione"
RELEASE	BOOLEANO	rilascia il semaforo

BUSY	BOOLEANO	stato del semaforo

Descrizione:

```
(* "x" è una variabile booleana inizializzata a FALSE *)
busy := x;
If claim Then
        x := True;
Else
        If release Then
                busy := False;
                x := False;
        End_if;
End_if;
```

CTU



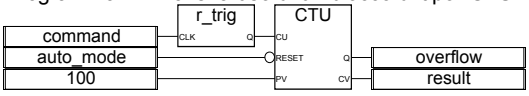
Argomenti:

CU	BOO	input del contatore (il conteggio avviene quando CU vale TRUE)
RESET	BOO	comando di reset (dominante)
PV	INT	valore massimo programmato
Q	BOO	overflow: TRUE quando CV = PV
CV	INT	valore del contatore

Avvertenza: Il blocco CTU non rileva fronti di salita o di caduta dell'input del contatore (CU). Per ottenere un contatore ad impulso, si deve associare un blocco di tipo "R_TRIG" o "F_TRIG".

Descrizione:
Conteggio (intero) da 0 fino ad un dato valore ad incrementi di uno

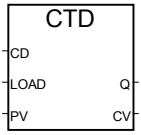
(* Programma FBD che fa uso di un blocco di tipo "CTU" block *)




```
(* Equivalente ST: Si assuma R_TRIG1 istanza del blocco R_TRIG e CTU1 istanza del CTU
*)
CTU1(R_TRIG1(command),NOT(auto_mode),100);
overflow := CTU1.Q;
result := CTU1.CV;

(* Equivalente IL: *)
LD              command
```

```
ST      R_TRIG1.clk
CAL     R_TRIG1
LD      R_TRIG1.Q
ST      CTU1.cu
LDN     auto_mode
ST      CTU1.reset
LD      100
ST      CTU1.pv
CAL     CTU1
LD      CTU1.Q
ST      overflow
LD      CTU1.cv
ST      result
```

CTD



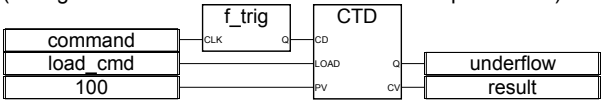
Argomenti:

CD	BOO	input del contatore (il conteggio a ritroso avviene quando CD vale TRUE)
LOAD	BOO	comando di caricamento (dominante) (CV = PV quando LOAD vale TRUE)
PV	INT	valore iniziale programmato
Q	BOO	underflow: TRUE quando CV = 0
CV	INT	valore del contatore

Avvertenza: Il blocco CTD non rileva fronti di salita o di caduta dell'input del contatore (CD). Per ottenere un contatore ad impulso, si deve associare un blocco di tipo "R_TRIG" o "F_TRIG".

Descrizione:
Conteggio (intero) da un dato valore fino a zero a decrementi di 1

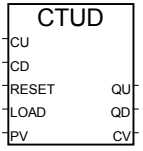
(* Programma FBD che fa uso di un blocco di tipo "CTD" *)




```
 (* Equivalente ST: Si assuma F_TRIG1 istanza del blocco F_TRIG e CTD1 istanza del
blocco CTD *)
CTD1(F_TRIG1(command),load_cmd,100);
underflow := CTD1.Q;
result := CTD1.CV;

(* Equivalente IL: *)
```

```
LD      command
ST      F_TRIG1.clk
CAL     F_TRIG1
LD      F_TRIG1.Q
ST      CTD1.cd
LD      load_cmd
ST      CTD1.load
LD      100
ST      CTD1.pv
CAL     CTD1
LD      CTD1.Q
ST      underflow
LD      CTD1.cv
ST      result
```

CTUD



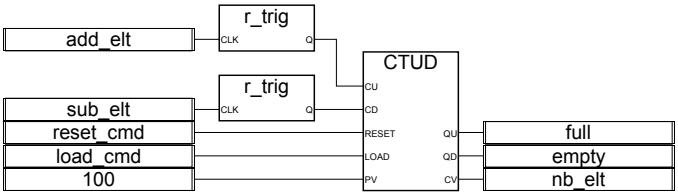
Argomenti:

CU	BOO	contatore a salire (quando CU vale TRUE)
CD	BOO	contatore a scendere (quando CD vale TRUE)
RESET	BOO	comando di reset (dominante) (CV = 0 quando RESET vale TRUE)
LOAD	BOO	comando di caricamento (CV = PV quando LOAD vale TRUE)
PV	INT	valore massimo programmato
QU	BOO	overflow: TRUE quando CV = PV
QD	BOO	underflow: TRUE quando CV = 0
CV	INT	valore del contatore

Avvertenza: Il blocco CTUD non rileva fronti di salita o di caduta degli input del contatore (CU e CD). Per ottenere un contatore ad impulso, si deve associare un blocco di tipo "R_TRIG" o "F_TRIG".

Descrizione:
Conteggio (intero) da 0 fino ad un dato valore ad incrementi di 1
o da un dato valore fino a 0 a decrementi di 1

```
(* Programma FBD che fa uso di un blocco di tipo "CTUD" *)
```

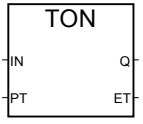


(* Equivalente ST: Si assumano R_TRIG1 e R_TRIG2 due istanze del blocco R_TRIG e CTUD1 istanza del blocco CTUD *)
CTUD1(R_TRIG1(add_elt), R_TRIG2(sub_elt), reset_cmd, load_cmd,100);
full := CTUD1.QU;
empty := CTUD1.QD;
nb_elt := CTUD1.CV;

(* Equivalente IL: *)

```
LD          add_elt
ST          R_TRIG1.clk
CAL         R_TRIG1
LD          R_TRIG1.Q
ST          CTUD1.cu
LD          sub_elt
ST          R_TRIG2.clk
CAL         R_TRIG2
LD          R_TRIG2.Q
ST          CTUD1.cd
LD          reset_cmd
ST          CTUD1.reset
LD          load_cmd
ST          CTUD1.load
LD          100
ST          CTUD1.pv
CAL         CTUD1
LD          CTUD1.QU
ST          full
LD          CTUD1.QD
ST          empty
LD          CTUD1.CV
ST          nb_elt
```

TON



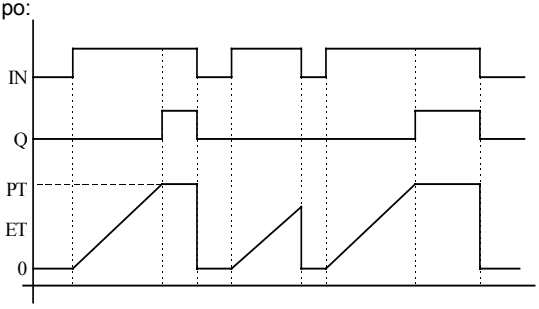
Argomenti: {XE "TON"} {XE "Temporizzatore On-delay"}

IN	BOO	Se fronte di salita, inizia ad incrementare un timer interno Se fronte di discesa, ferma ed azzera il timer interno

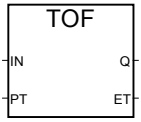
PT	TMR	massimo tempo programmato
Q	BOO	Se TRUE, il tempo programmato è trascorso
ET	TMR	tempo attualmente trascorso

Descrizione:
Incrementa un timer interno fino ad un dato valore.

Diagramma del tempo:



TOF

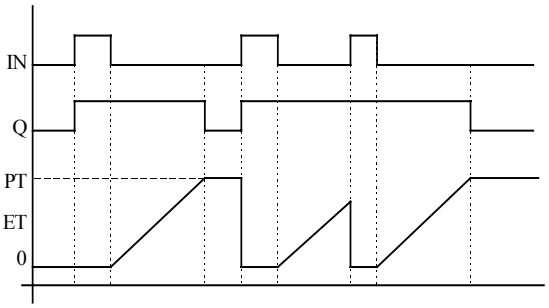


Argomenti:

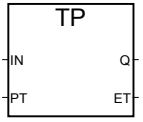
IN	BOO	Se fronte di discesa, inizia ad incrementare un timer interno Se fronte di salita ferma ed azzera il timer interno
PT	TMR	massimo tempo programmato
Q	BOO	Se TRUE, il tempo programmato non è trascorso
ET	TMR	tempo attualmente trascorso

Descrizione:
Incrementa un timer interno fino ad un dato valore.

Diagramma del tempo:



TP



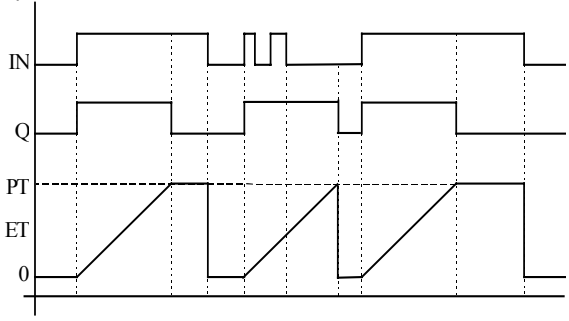
Argomenti:

IN	BOO	Se fronte di salita, inizia l'incremento di un timer interno (se non viene già attualmente incrementato) Se FALSE e solamente se il tempo è trascorso, azzera il timer interno I cambiamenti di IN durante il conteggio, non producono alcun effetto.
PT	TMR	massimo tempo programmato
Q	BOO	Se TRUE: il timer è attivo (sta contando)
ET	TMR	tempo attualmente trascorso

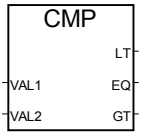
Descrizione:

Incrementa un timer interno fino ad un dato valore.

Diagramma del tempo:



CMP



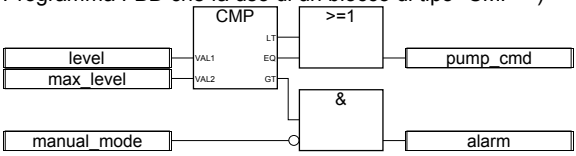
Argomenti:

VAL1	INT	qualsiasi valore analogico intero con segno
VAL2	INT	qualsiasi valore analogico intero con segno
LT	BOO	TRUE se val1 è Minore di val2
EQ	BOO	TRUE se val1 è Uguale a val2
GT	BOO	TRUE se val1 è Maggiore di val2

Descrizione:

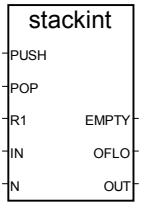
Confronta due valori: riporta se sono uguali, o se il primo è minore o maggiore del secondo.

(* Programma FBD che fa uso di un blocco di tipo "CMP" *)




```
(* Equivalente ST: Si assuma CMP1 istanza del blocco CMP *)
CMP1(level, max_level);
pump_cmd := CMP1.LT OR CMP1.EQ;
alarm := CMP1.GT AND NOT(manual_mode);

(* Equivalente IL: *)
LD      level
ST      CMP1.val1
LD      max_level
ST      CMP1.val2
CAL     CMP1
LD      CMP1.LT
OR      CMP1.EQ
ST      pump_cmd
LD      CMP1.GT
ANDN    manual_mode
ST      alarm
```

STACKINT



Argomenti:

PUSH	BOO	comando push (solamente per fronti in salita) aggiunge il valore IN sullo stack
POP	BOO	comando pop (solamente per fronti in salita) elimina dallo stack l'ultimo valore inserito (in cima allo stack)
R1	BOO	svuota lo stack
IN	INT	valore inserito
N	INT	dimensione dello stack definita dall'applicazione
EMPTY	BOO	TRUE se lo stack è vuoto
OFLO	BOO	overflow: TRUE se lo stack è pieno
OUT	INT	valore in cima allo stack

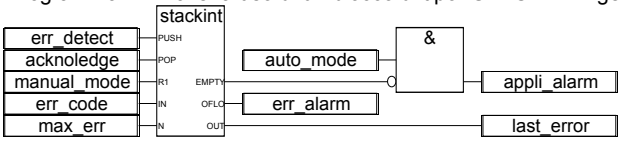
Descrizione:

Gestione di uno stack di valori interi.

Il blocco funzione "STACKINT" comprende la rilevazione di un fronte di salita per entrambi i comandi PUSH e POP. La dimensione massima dello stack è **128**. La dimensione **N** dello stack definita dall'applicazione non può essere inferiore ad 1 o superiore a 128.

Si noti che il valore OFLO è valido solamente dopo un reset (R1 è stato impostato a TRUE almeno una volta e poi di nuovo a FALSE).

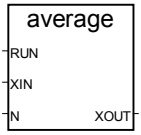
(* Programma FBD che fa uso di un blocco di tipo "STACKINT": gestione dell'errore *)




```
(* Equivalente ST: Si assuma STACKINT1 istanza del blocco STACKINT *)
STACKINT1(err_detect, acknoledge, manual_mode, err_code, max_err);
appli_alarm := auto_mode AND NOT(STACKINT1.EMPTY);
err_alarm := STACKINT1.OFLO;
last_error := STACKINT1.OUT;

(* Equivalente IL: *)
LD      err_detect
ST      STACKINT1.push
LD      acknoledge
ST      STACKINT1.pop
LD      manual_mode
```

```
ST      STACKINT1.r1
LD      err_code
ST      STACKINT1.IN
LD      max_err
ST      STACKINT1.N
CAL     STACKINT1
LD      auto_mode
ANDN    STACKINT1.empty
ST      appli_alarm
LD      STACKINT1.OFLO
ST      err_alarm
LD      STACKINT1.OUT
ST      last_error
```

AVERAGE



Argomenti:

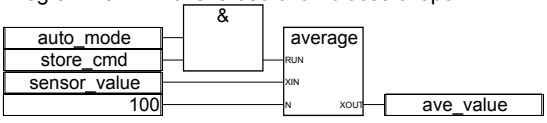
RUN	BOO	TRUE=esecuzione / FALSE=reset
XIN	REALE	qualsiasi variabile reale analogica
N	INT	Numero di campioni definito dall'applicazione
XOUT	REALE	Media mobile del valore XIN

Descrizione:

Memorizza un valore ad ogni ciclo e calcola il valore tra tutti i valori precedentemente memorizzati. Vengono memorizzati solamente gli ultimi N valori.

Il numero **N** di campioni non può eccedere **128**.
Quando il comando "**RUN**" vale **FALSE** (modo reset), il valore in output è uguale al valore in input.
Quando viene raggiunto il massimo N di valori memorizzati, il primo di questi viene sovrascritto dall'ultimo entrato.

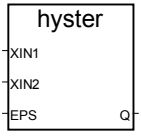
(* Programma FBD che fa uso di un blocco di tipo "AVERAGE": *)




```
(* Equivalente ST: AVERAGE1 istanza del blocco AVERAGE*)
AVERAGE1((auto_mode & store_cmd), sensor_value, 100);
ave_value := AVERAGE1.XOUT;

(* Equivalente IL: *)
LD      auto_mode
AND     store_cmd
```

```
ST      AVERAGE1.run
LD      sensor_value
ST      AVERAGE1.xin
LD      100
ST      AVERAGE1.N
CAL     AVERAGE1
LD      AVERAGE1.XOUT
ST      ave_value
```

HYSTER



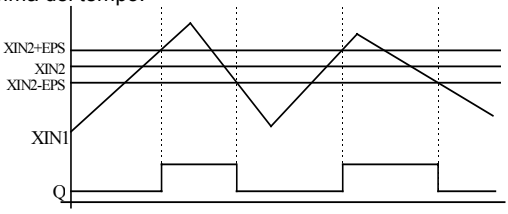
Argomenti:

XIN1	REALE	qualsiasi valore reale analogico
XIN2	REALE	per controllare se XIN1 abbia oltrepassato XIN2+EPS
EPS	REALE	valore di isteresi (deve essere maggiore di zero)
Q	BOO	TRUE se XIN1 ha oltrepassato XIN2+EPS e non è ancora inferiore a XIN2-EPS

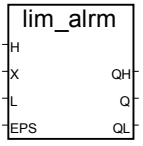
Descrizione:

Isteresi con limite superiore su un valore reale.

Esempio del diagramma del tempo:



LIM_ALRM



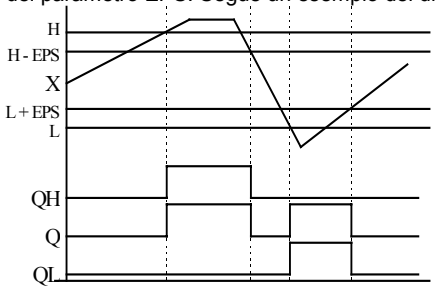
Argomenti:

H	REALE	limite alto
X	REALE	input: qualsiasi valore reale analogico
L	REALE	limite basso

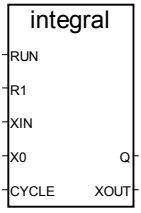
EPS	REALE	valore di isteresi (deve essere maggiore di zero)
QH	BOO	allarme "alto": TRUE se X sopra il limite alto H
Q	BOO	allarme output: TRUE se X fuori dai limiti
QL	BOO	allarme “basso”: TRUE se X sotto il limite basso L

Descrizione:
Isteresi con limite superiore/inferiore su un valore reale.

Isteresi applicata su limiti inferiore/superiore. Il valore delta usato sia per il limite superiore che inferiore vale la metà del parametro EPS. Segue un esempio del diagramma del tempo:



INTEGRAL



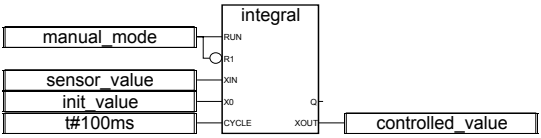
Argomenti:

RUN	BOO	modo: TRUE=integrate / FALSE=hold
R1	BOO	reset dominante
XIN	REALE	ingresso: qualsiasi valore analogico reale
X0	REALE	valore iniziale
CYCLE	TMR	periodo di campionamento
Q	BOO	Not R1
XOUT	REALE	uscita dell’integrale

Descrizione:
Integrazione di un valore reale.

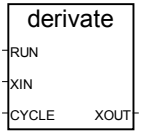
Se il valore del parametro "**CYCLE**" è inferiore al tempo ciclo dell’applicazione ISaGRAF, il periodo di campionamento equivale al tempo ciclo dell’applicazione.

(* Programma FBD che fa uso di un blocco di tipo "INTEGRAL" *)




```
(* Equivalente ST: INTEGRAL1 istanza del blocco INTEGRAL *)
INTEGRAL1(manual_mode, NOT(manual_mode), sensor_value, init_value, t#100ms);
controlled_value := INTEGRAL1.XOUT;

(* Equivalente IL: *)
LD          manual_mode
ST          INTEGRAL1.run
STN         INTEGRAL1.R1
LD          sensor_value
ST          INTEGRAL1.XIN
LD          init_value
ST          INTEGRAL1.X0
LD          t#100ms
ST          INTEGRAL1.CYCLE
CAL         INTEGRAL1
LD          INTEGRAL1.XOUT
ST          controlled_value
```

DERIVATE



Argomenti:

RUN	BOO	modo: TRUE=normale / FALSE=reset
XIN	REALE	ingresso: qualsiasi valore analogico reale
CYCLE	TMR	periodo di campionamento
XOUT	REALE	uscita della derivata

Descrizione:

Derivazione di un valore reale.

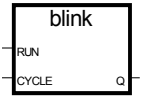
Se il valore del parametro "**CYCLE**" è inferiore al tempo ciclo dell'applicazione ISaGRAF, il periodo di campionamento equivale al tempo ciclo dell'applicazione.

(* Programma FBD che fa uso di un blocco di tipo "DERIVATE" *)

(* Equivalente ST: DERIVATE1 istanza del blocco DERIVATE *)

```
DERIVATE1(manual_mode, sensor_value, t#100ms);
derivated_value := DERIVATE1.XOUT;

(* Equivalente IL: *)
LD      manual_mode
ST      DERIVATE1.run
LD      sensor_value
ST      DERIVATE1.XIN
LD      t#100ms
ST      DERIVATE1.CYCLE
CAL     DERIVATE1
LD      DERIVATE1.XOUT
ST      derivated_value
```

BLINK



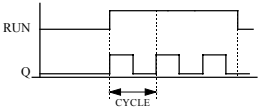
Argomenti:

RUN	BOO	modo: TRUE=intermittenza / FALSE=imposta output a false
CYCLE	TMR	periodo intermittenza
Q	BOO	output segnale intermittente

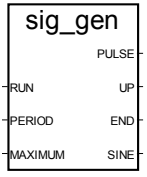
Descrizione:

Genera un segnale intermittente.

Diagramma del tempo:



SIG_GEN



Argomenti:

RUN	BOO	modo: TRUE=attivo / FALSE=reset a false
PERIOD	TMR	durata di un campione
MAXIMUM	INT	valore massimo del contatore

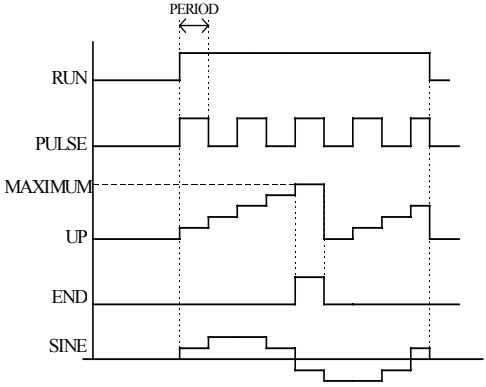
PULSE	BOO	inversione dopo ogni campione
UP	INT	contatore ad incremento, incrementato dopo ogni campione
END	BOO	TRUE al termine del conteggio incrementale
SINE	REALE	segnale sinusoidale (periodo = durata conteggio)

Descrizione:

Genera l'intermittenza di vari segnali su un valore booleano, un contatore intero ad incremento, un'onda sinusoidale reale.

Quando il contatore raggiunge il valore massimo, riparte da 0 (zero). Così END assume il valore TRUE solamente per la durata di 1 PERIOD.

Timing diagram:



B.9.3 Funzioni standard

Queste sono le funzioni standard incorporate dal sistema ISaGRAF. Queste funzioni sono predefinite e non devono essere dichiarate nella libreria.

● Matematiche	ABS	Valore assoluto
	EXPT, POW	Calcolo di esponente ed elevazione a potenza.
	LOG	Logaritmo
	SQRT	Radice quadrata
	TRUNC	Troncamento della parte decimale
● Trigonometriche	ACOS, ASIN,	Arco coseno, Arco seno,
	ATAN	Arco tangente
	COS, SIN,	Coseno, Seno,
	TAN	Tangente
● Controllo registri	ROL, ROR	Rotazione a sinistra, Rotazione a destra
	SHL, SHR	Shift a sinistra, Shift a destra
● Manipolazione dati	MIN, MAX,	Minimo, Massimo,
	LIMIT	Limite

	MOD	Modulo
	MUX4, MUX8	Multiplexer (4 o 8 entrate),
	SEL	Selettore binario
	ODD	Odd parity (parità dispari)
	RAND	Valore random (casuale)
● Conversione dati	ASCII	Carattere → Codice ASCII
	CHAR	Codice ASCII → Carattere
● Gestione stringhe	MLEN	Lunghezza stringa
	DELETE	Cancellazione sottostringa
	INSERT	Inserimento stringa
	FIND,	Ricerca sottostringa
	REPLACE	Sostituzione sottostringa
	LEFT, MID	Estrazione della parte sinistra, centrale
	RIGHT	o destra di una stringa
	DAY_TIME	Tempo del giorno
● Operazioni su Array.....	ARCREATE	Creazione di array (matrici) di valori interi
	ARREAD	Lettura /
	ARWRITE	Scrittura elemento di array
● Gestione file di tipo binario	F_ROPEN	Apertura file binario in modo Lettura
	F_WOPEN	Apertura file binario in modo Scrittura
	F_CLOSE	Chiusura file binario
	F_EOF	Controllo raggiungimento fine file binario
	FA_READ	Lettura di un valore analogico da file binario
	FA_WRITE	Scrittura di un valore analogico in un file binario
	FM_READ	Lettura di un messaggio stringa da file binario
	FM_WRITE	Scrittura di un messaggio stringa in file binario

ABS

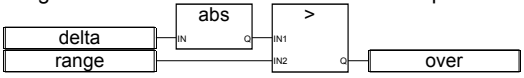
Argomenti:

IN	REALE	qualsiasi valore analogico reale con segno
Q	REALE	valore assoluto (sempre positivo)

Descrizione:

Restituisce il valore assoluto (positivo) di un valore reale.

(* Programma FBD che fa uso di un blocco di tipo "ABS" *)



(* Equivalente ST: *)
over := (ABS (delta) > range);

```
(* Equivalente IL: *)
LD          delta
ABS
GT          range
ST          over
```

EXPT

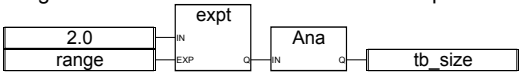
Argomenti:

IN	REALE	qualsiasi valore analogico reale con segno
EXP	INT	esponente intero analogico
Q	REALE	(IN ^{EXP})

Descrizione:

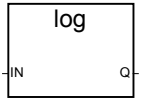
Restituisce il risultato reale dell'operazione: ($\texttt{Base}^{\texttt{Esponente}}$) con '`Base`' corrispondente al primo argomento ed '`Esponente`' al secondo.

(* Programma FBD che fa uso di un blocco di tipo "EXPT" *)




```
(* Equivalente ST: *)
tb_size := ANA (EXPT (2.0, range) );

(* Equivalente IL: *)
LD          2.0
EXPT        range
ANA
ST          tb_size
```

LOG



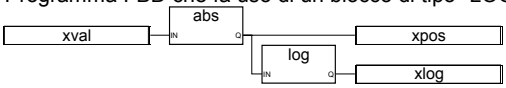
Argomenti:

IN	REALE	deve essere maggiore di zero
Q	REALE	logaritmo (in base 10) del valore di input

Descrizione:

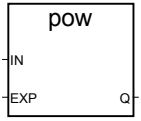
Calcola il logaritmo (in base 10) di un valore reale.

(* Programma FBD che fa uso di un blocco di tipo "LOG" *)




```
(* Equivalente ST: *)
xpos := ABS (xval);
xlog := LOG (xpos);

(* Equivalente IL: *)
LD              xval
ABS
ST              xpos
LOG
ST              xlog
```

POW



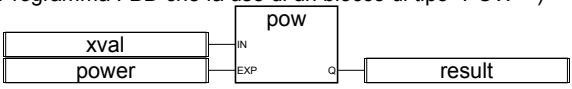
Argomenti:

IN	REALE	numero reale analogico da elevare
EXP	REALE	potenza (esponente)
Q	REALE	(IN ^{EXP}) 1.0 se IN è non 0.0 ed EXP è 0.0 0.0 se IN è 0.0 ed EXP è negativo 0.0 se entrambi IN e EXP sono 0.0 0.0 se IN è negativo e Y non corrisponde ad un intero

Descrizione:

Restituisce il risultato reale dell'operazione: (`Base` $^{\texttt{Esponente}}$) con '`Base`' corrispondente al primo argomento ed '`Esponente`' al secondo.

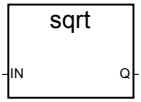
(* Programma FBD che fa uso di un blocco di tipo "POW" *)




```
(* Equivalente ST: *)
result := POW (xval, power);

(* Equivalente IL: *)
LD              xval
```

```
POW     power
ST      result
```

SQRT



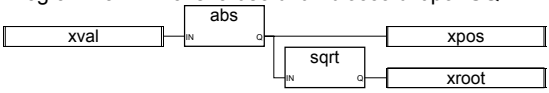
Argomenti:

IN	REALE	deve essere maggiore od uguale a zero
Q	REALE	radice quadrata del valore in input

Descrizione:

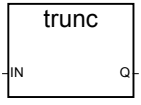
Calcola la radice quadrata di un valore reale.

(* Programma FBD che fa uso di un blocco di tipo "SQRT" *)




```
(* Equivalente ST: *)
xpos := ABS (xval);
xroot := SQRT (xpos);

(* Equivalente IL: *)
LD      xval
ABS
ST      xpos
SQRT
ST      xrout
```

TRUNC



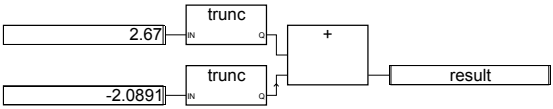
Argomenti:

IN	REALE	qualsiasi valore analogico reale
Q	REALE	se IN>0, il più grande intero minore o uguale dell'input se IN<0, il più piccolo intero maggiore o uguale dell'input

Descrizione:

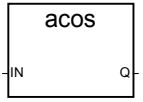
Tronca un valore reale restituendo la sola parte intera

(* Programma FBD che fa uso di un blocco di tipo "TRUNC" *)




```
(* Equivalente ST: *)
result := TRUNC (+2.67) + TRUNC (-2.0891);
(* significa: result := 2.0 + (-2.0) := 0.0; *)

(* Equivalente IL: *)
LD          2.67
TRUNC
ST          temporary   (* risultato temporaneo del primo TRUNC *)
LD          -2.0891
TRUNC
ADD         temporary
ST          result
```

ACOS



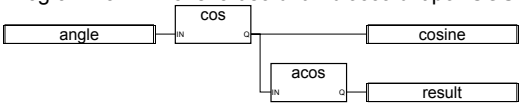
Argomenti:

IN	REALE	deve essere compreso tra [-1.0 .. +1.0]
Q	REALE	arco coseno del valore in input (compreso tra [0.0 .. PI]) = 0.0 per input non valido

Descrizione:

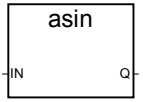
Calcola l'arco coseno di un valore reale.

(* Programma FBD che fa uso di un blocco di tipo "COS" e "ACOS" *)




```
(* Equivalente ST: *)
cosine := COS (angle);
result := ACOS (cosine); (* result equivale all'angolo *)

(* Equivalente IL: *)
LD          angle
COS
ST          cosine
ACOS
ST          result
```

ASIN

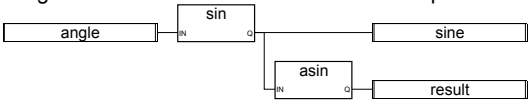


Argomenti: **{**XE "ASIN"**} {**XE "Arc sine"**}**

IN	REALE	deve essere compreso tra [-1.0 .. +1.0]
Q	REALE	arco seno del valore in input (compreso tra [-PI/2 .. +PI/2]) = 0.0 per input non valido

Descrizione:
Calcola l'arco seno di un valore reale.

(* Programma FBD che fa uso di un blocco di tipo "SIN" e "ASIN" *)

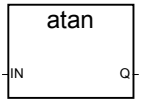


(* Equivalente ST: *)
sine := SIN (angle);
result := ASIN (sine); (* result equivale all'angolo *)

(* Equivalente IL: *)

LD	angle
SIN	
ST	sine
ASIN	
ST	result

ATAN

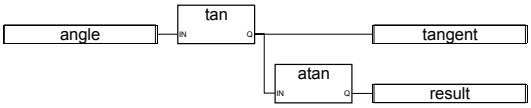


Argomenti:

IN	REALE	qualsiasi valore analogico reale
Q	REALE	arco tangente del valore in input (compreso tra [-PI/2 .. +PI/2]) = 0.0 per input non valido

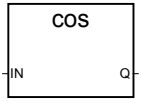
Descrizione:
Calcola l'arco tangente di un valore reale.

(* Programma FBD che fa uso di un blocco di tipo "TAN" e "ATAN" *)




```
(* Equivalente ST: *)
tangent := TAN (angle);
result := ATAN (tangent); (* result equivale all'angolo *)

(* Equivalente IL: *)
LD          angle
TAN
ST          tangent
ATAN
ST          result
```

COS

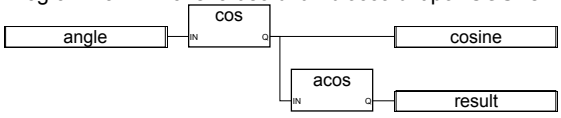


Argomenti:

IN	REALE	qualsiasi valore analogico reale
Q	REALE	coseno del valore in input (compreso tra [-1.0 .. +1.0])

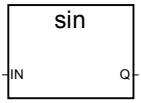
Descrizione:
Calcola il Coseno di un valore reale.

(* Programma FBD che fa uso di un blocco di tipo "COS" e "ACOS" *)




```
(* Equivalente ST: *)
cosine := COS (angle);
result := ACOS (cosine); (* result equivale all'angolo *)

(* Equivalente IL: *)
LD          angle
COS
ST          cosine
ACOS
ST          result
```

SIN

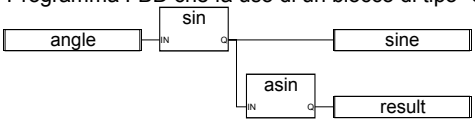


Argomenti:

IN	REALE	qualsiasi valore analogico reale
Q	REALE	seno del valore in input (compreso tra [-1.0 .. +1.0])

Descrizione:
Calcola il Seno di un valore reale.

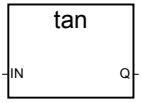
(* Programma FBD che fa uso di un blocco di tipo "SIN" e "ASIN" *)



(* Equivalente ST: *)
sine := SIN (angle);
result := ASIN (sine); (* result equivale all'angolo *)

(* Equivalente IL: *)
LD angle
SIN
ST sine
ASIN
ST result

TAN

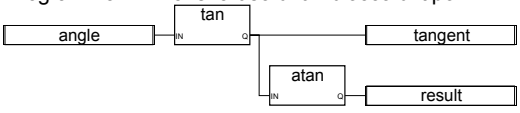


Argomenti:

IN	REALE	non può essere uguale a PI/2 modulo PI
Q	REALE	tangente del valore in input = 1E+38 per input non valido

Descrizione:
Calcola la Tangente di un valore reale.

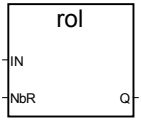
(* Programma FBD che fa uso di un blocco di tipo "TAN" e "ATAN" *)



(* Equivalente ST: *)

```
tangent := TAN (angle);
result := ATAN (tangent); (* result equivale all'angolo*)

(* Equivalente IL: *)
LD          angle
TAN
ST          tangent
ATAN
ST          result
```

ROL

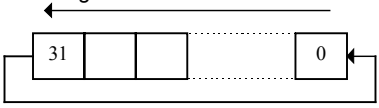


Argomenti:

IN	INT	qualsiasi valore analogico intero
NbR	INT	numero di rotazioni di 1 bit (compreso tra [1..31])
Q	INT	valore dopo la rotazione a sinistra nessun effetto se NbR <= 0

Descrizione:

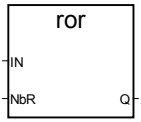
Esegue la rotazione a sinistra dei bit di un intero. La rotazione viene eseguita su 32 bit:



(* Programma FBD che fa uso di un blocco di tipo "ROL" *)


```
(* Equivalente ST: *)
result := ROL (register, 1);
(* register = 2#0100_1101_0011_0101*)
(* result   = 2#1001_1010_0110_1010*)

(* Equivalente IL: *)
LD          register
ROL         1
ST          result
```

ROR

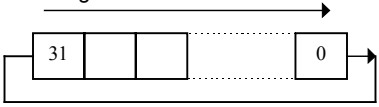


Argomenti:

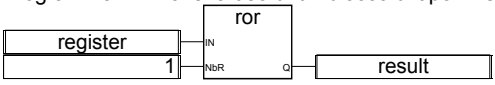
IN	INT	qualsiasi valore analogico intero
NbR	INT	numero di rotazioni di 1 bit (compreso tra [1..31])
Q	INT	valore dopo la rotazione a destra nessun effetto se NbR <= 0

Descrizione:

Esegue la rotazione a destra dei bit di un intero. La rotazione viene eseguita su 32 bit:

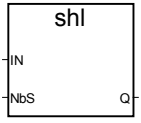


(* Programma FBD che fa uso di un blocco di tipo "ROR" *)




```
(* Equivalente ST: *)
result := ROR (register, 1);
(* register = 2#0100_1101_0011_0101 *)
(* result   = 2#1010_0110_1001_1010 *)

(* Equivalente IL: *)
LD          register
ROR         1
ST          result
```

SHL

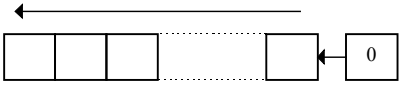


Argomenti:

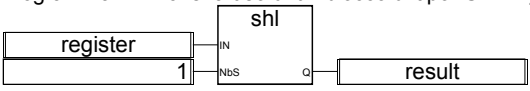
IN	INT	qualsiasi valore analogico intero
NbS	INT	numero di spostamenti di 1 bit (compreso tra [1..31])
Q	INT	valore dopo lo shift a sinistra nessun effetto se NbS <= 0 il bit più basso viene sostituito da 0

Descrizione:

Esegue lo shift (spostamento) a sinistra dei bit di un intero. Lo shift viene eseguito su 32 bit:

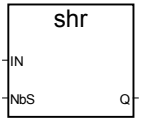


(* Programma FBD che fa uso di un blocco di tipo "SHL" *)




```
(* Equivalente ST: *)
result := SHL (register,1);
(* register = 2#0100_1101_0011_0101 *)
(* result   = 2#1001_1010_0110_1010 *)

(* Equivalente IL: *)
LD          register
SHL         1
ST          result
```

SHR

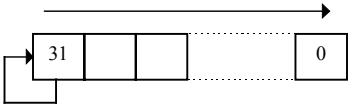


Argomenti:

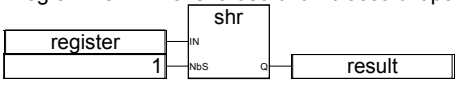
IN	INT	qualsiasi valore analogico intero
NbS	INT	numero di spostamenti di 1 bit (compreso tra [1..31])
Q	INT	valore dopo lo shift a destra nessun effetto se NbS <= 0 il bit più alto viene copiato ad ogni shift

Descrizione:

Esegue lo shift (spostamento) a destra dei bit di un intero. Lo shift viene eseguito su 32 bit:



(* Programma FBD che fa uso di un blocco di tipo "SHR" *)



(* Equivalente ST: *)

```
result := SHR (register,1);
(* register = 2#1100_1101_0011_0101 *)
(* result   = 2#1110_0110_1001_1010 *)

(* Equivalente IL: *)
LD          register
SHR         1
ST          result
```

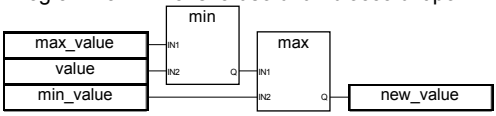
MIN

Argomenti:

IN1	INT	qualsiasi valore analogico intero con segno
IN2	INT	(non può essere REALE)
Q	INT	il minimo di entrambi i valori di input

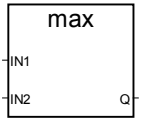
Descrizione:
Restituisce il minimo di due valori interi.

(* Programma FBD che fa uso di un blocco di tipo "MIN" e "MAX" *)




```
(* Equivalente ST: *)
new_value := MAX (MIN (max_value, value), min_value);
(* arrotonda il valore all'insieme [min_value..max_value] *)

(* Equivalente IL: *)
LD          max_value
MIN         value
MAX         min_value
ST          new_value
```

MAX

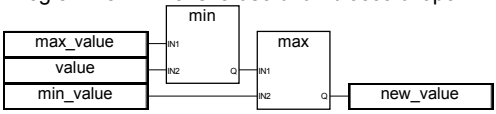


Argomenti:

IN1	INT	qualsiasi valore analogico intero con segno
IN2	INT	(non può essere REALE)
Q	INT	il massimo di entrambi i valori di input

Descrizione:
Restituisce il massimo di due valori interi.

(* Programma FBD che fa uso di un blocco di tipo "MIN" e "MAX" *)

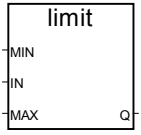


(* Equivalente ST: *)
new_value := MAX (MIN (max_value, value), min_value);
(* arrotonda il valore all'insieme [min_value..max_value] *)

(* Equivalente IL: *)

```
LD      max_value
MIN     value
MAX     min_value
ST      new_value
```

LIMIT

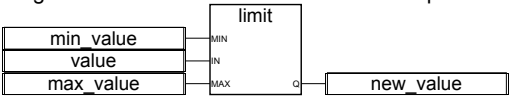


Argomenti:

MIN	INT	valore minimo consentito
IN	INT	qualsiasi valore analogico intero con segno
MAX	INT	valore massimo consentito
Q	INT	il valore di input arrotondato all'intervallo consentito

Descrizione:
Limita un valore intero all'interno di un dato intervallo. Mantiene lo stesso valore se è compreso tra massimo e minimo, assume il valore massimo se è superiore, assume il minimo se inferiore.

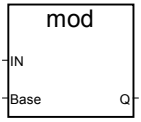
(* Programma FBD che fa uso di un blocco di tipo "LIMIT" *)



(* Equivalente ST: *)
new_value := LIMIT (min_value, value, max_value);
(* arrotonda il valore all'insieme [min_value..max_value] *)

```
(* Equivalente IL: *)
LD          min_value
LIMIT       value, max_value
ST          new_value
```

MOD

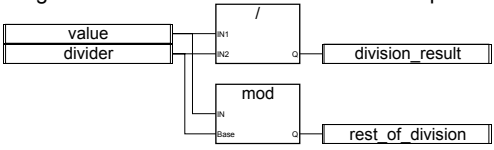


Argomenti:

IN	INT	qualsiasi valore analogico INTERO
Base	INT	deve essere maggiore di zero
Q	INT	calcolo del modulo (input MOD base) restituisce -1 se Base <= 0

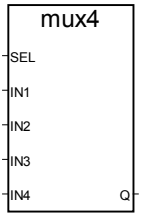
Descrizione:
Calcola il modulo di un valore intero.

(* Programma FBD che fa uso di un blocco di tipo "MOD" *)




```
(* Equivalente ST: *)
division_result := (value / divider); (* divisione intera *)
rest_of_division := MOD (value, divider); (* resto della divisione *)

(* Equivalente IL: *)
LD          value
DIV         divider
ST          division_result
LD          value
MOD         divider
ST          rest_of_division
```

MUX4



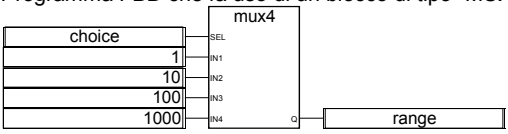
Argomenti:

SEL	INT	valore intero del selettore (deve essere compreso tra [0..3])
IN1..IN4	INT	qualsiasi valore analogico intero
Q	INT	= valore1 se SEL = 0 = valore2 se SEL = 1 = valore3 se SEL = 2 = valore4 se SEL = 3 = 0 per tutti gli altri valori del selettore

Descrizione:

Multiplexer a 4 entrate: seleziona un valore tra 4 valori interi.

(* Programma FBD che fa uso di un blocco di tipo "MUX4" *)

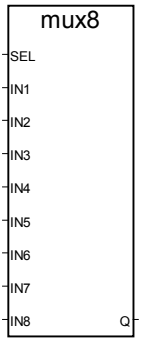


(* Equivalente ST: *)
range := MUX4 (choice, 1, 10, 100, 1000);
(* seleziona a partire da 4 intervalli predefiniti, ad esempio se la scelta è 1, l'intervallo sarà 10 *)

(* Equivalente IL: *)

```
LD      choice
MUX4    1,10,100,1000
ST      range
```

MUX8



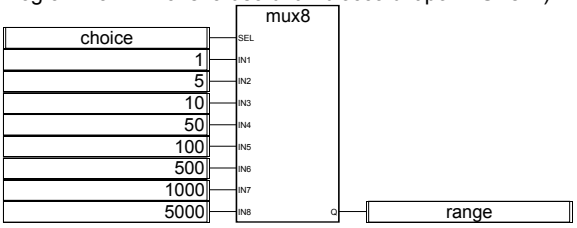
Argomenti:

SEL	INT	valore intero del selettore (deve essere compreso tra [0..7])
IN1..IN8	INT	qualsiasi valore analogico intero
Q	INT	= valore1 se SEL = 0 = valore2 se SEL = 1 ... = valore8 se SEL = 7 = 0 per tutti gli altri valori del selettore

Descrizione:

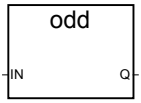
Multiplexer a 8 entrate: seleziona un valore tra 8 valori interi.

(* Programma FBD che fa uso di un blocco di tipo "MUX8" *)




```
(* Equivalente ST: *)
range := MUX8 (choice, 1, 5, 10, 50, 100, 500, 1000, 5000);
(* seleziona a partire da 8 intervalli predefiniti, ad esempio se la scelta è 3, l'intervallo sarà 50
*)

(* Equivalente IL: *)
LD          choice
MUX8        1,5,10,50,100,500,1000,5000
ST          range
```

ODD



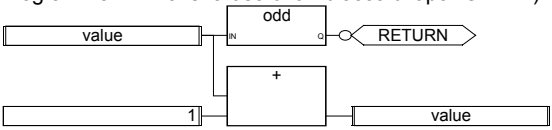
Argomenti:

IN	INT	qualsiasi valore analogico intero con segno
Q	BOO	TRUE se il valore di input è dispari FALSE se il valore di input è pari

Descrizione:

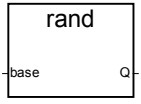
Valuta la parità di un intero: il risultato è dispari o pari.

(* Programma FBD che fa uso di un blocco di tipo "ODD" *)




```
(* Equivalente ST: *)
If Not (ODD (value)) Then Return; End_if;
value := value + 1;
(* forza il valore ad essere sempre pari *)

(* Equivalente IL: *)
LD          value
ODD
RETNC
LD          value
ADD         1
ST          value
```

RAND



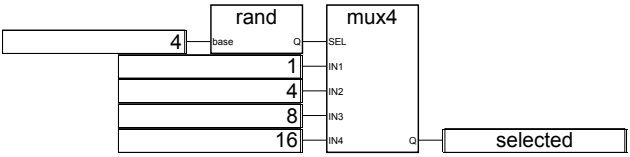
Argomenti:

base	INT	definisce un insieme di numeri permessi
Q	INT	valore casuale (random) compreso nell'insieme [0..base-1]

Descrizione:

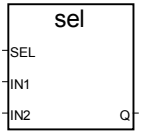
Restituisce un valore intero casuale (random) compreso nell'intervallo dato.

(* Programma FBD che fa uso di un blocco di tipo "RAND" *)




```
(* Equivalente ST: *)
selected := MUX4 ( RAND (4), 1, 4, 8, 16 );
(*
selezione casuale di 1 di 4 valori predefiniti
il valore uscente dalla chiamata a RAND è compreso nell'insieme [0..3],
così 'selected' restituito da MUX4, avrà valore 'casuale'
1 se 0 viene restituito da RAND,
o 4 se 1 viene restituito da RAND,
o 8 se 2 viene restituito da RAND,
o 16 se 3 viene restituito da RAND,
*)

(* Equivalente IL: *)
LD          4
RAND
MUX4        1,4,8,16
ST          selected
```

SEL



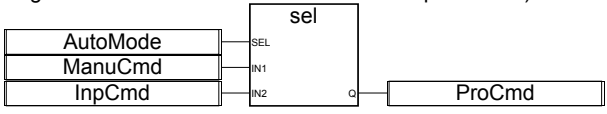
Argomenti:

SEL	BOO	indica il valore scelto
IN1, IN2	INT	qualsiasi valore intero analogico
Q	INT	= valore1 se SEL vale FALSE = valore2 se SEL vale TRUE

Descrizione:

Selettore binario: seleziona un valore tra due valori interi.

(* Programma FBD che fa uso di un blocco di tipo "SEL" *)

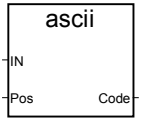



```
(* Equivalente ST: *)
ProCmd := SEL (AutoMode, ManuCmd, InpCmd);
```

(* processa il comando di selezione *)

(* Equivalente IL: *)
LD AutoMode
SEL ManuCmd,InpCmd
ST ProCmd

ASCII



Argomenti:

IN	MSG	qualsiasi stinga non nulla
Pos	INT	posizione del carattere voluto compresa tra [1.. len] (dove len corrisponde alla lunghezza del messaggio IN)
Code	INT	codice del carattere voluto (compreso tra [0 .. 255]) restituisce 0 se Pos è esterna alla stringa

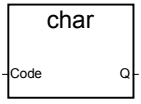
Descrizione:
Restituisce il codice ASCII di un carattere di una stringa messaggio.

(* Programma FBD che fa uso di un blocco di tipo "ASCII" *)

(* Equivalente ST: *)
FirstChr := ASCII (message, 1);
(* FirstChr è il codice Ascii del primo carattere della stringa *)

(* Equivalente IL: *)
LD message
ASCII 1
ST FirstChr

CHAR



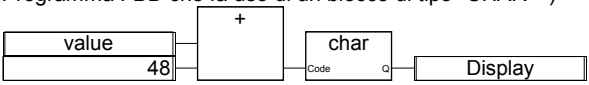
Argomenti:

Code	INT	codice compreso nell'insieme [0 .. 255]
Q	MSG	stringa di un carattere

il carattere ha il codice ASCII equivalente al codice di input
(il codice ASCII viene usato modulo 256)

Descrizione:
Restituisce un stringa messaggio composta da un carattere a partire da un dato codice ASCII.

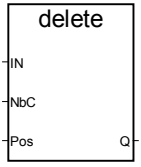
(* Programma FBD che fa uso di un blocco di tipo "CHAR" *)



(* Equivalente ST: *)
Display := CHAR (value + 48);
(* il valore appartiene all'insieme [0..9] *)
(* 48 è il codice ascii per '0' *)
(* il risultato è costituito da una stringa di un carattere da '0' a '9' *)

(* Equivalente IL: *)
LD value
ADD 48
CHAR
ST Display

DELETE

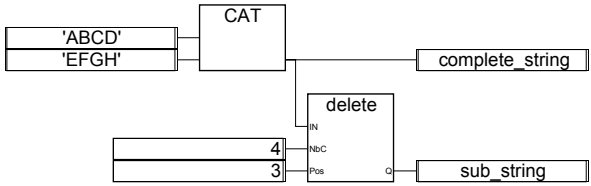


Argomenti:

IN	MSG	qualsiasi messaggio non vuoto
NbC	INT	numero di caratteri da cancellare
Pos	INT	posizione del primo carattere da cancellare (il primo carattere della stringa ha posizione 1)
Q	MSG	stringa modificata stringa vuota se Pos < 1 stringa iniziale se Pos > lunghezza della stringaIN stringa iniziale se NbC <= 0

Descrizione:
Cancella una parte di una stringa messaggio.

(* Programma FBD che fa uso di un blocco di tipo "DELETE" *)




```
(* Equivalente ST: *)
complete_string := 'ABCD' + 'EFGH'; (* complete_string vale 'ABCDEFGH' *)
sub_string := DELETE (complete_string, 4, 3); (* sub_string vale 'ABGH' *)

(* Equivalente IL: *)
LD          'ABCD'
ADD         'EFGH'
ST          complete_string
DELETE      4,3
ST          sub_string
```

FIND

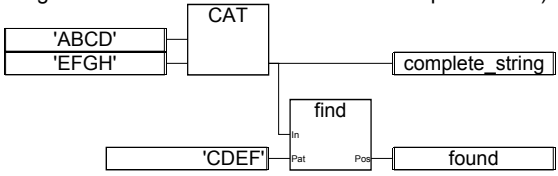
Argomenti:

In	MSG	qualsiasi stringa messaggio
Pat	MSG	qualsiasi stringa non vuota (Pattern)
Pos	INT	= 0 se sottostringa Pat non trovata = posizione del primo carattere della prima occorrenza della sottostringa Pat (la prima posizione equivale ad 1) questa funzione è **case sensitive** (distingue tra maiuscole e minuscole)

Descrizione:

Cerca una sottostringa in una stringa messaggio. Restituisce la posizione della sottostringa nella stringa.

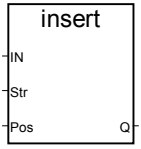
(* Programma FBD che fa uso di un blocco di tipo "FIND" *)



(* Equivalente ST: *)

```
complete_string := 'ABCD' + 'EFGH'; (* complete_string vale 'ABCDEFGH' *)
found := FIND (complete_string, 'CDEF'); (* la variabile found vale  3 *)

(* Equivalente IL: *)
LD          'ABCD'
ADD         'EFGH'
ST          complete_string
FIND        'CDEF'
ST          found
```

INSERT



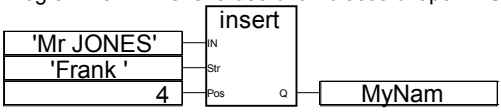
Argomenti:

IN	MSG	stringa iniziale
Str	MSG	stringa da inserire
Pos	INT	posizione per l'inserimento l'inserimento avviene prima della posizione (la prima posizione valida è 1)
Q	MSG	stringa modificata stringa vuota se Pos <= 0 concatenazione delle due stringhe se Pos è maggiore della lunghezza della stringa IN

Descrizione:

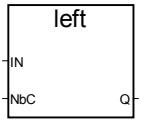
Inserimento in una data posizione, di una sottostringa in una stringa messaggio.

(* Programma FBD che fa uso di un blocco di tipo "INSERT" *)




```
(* Equivalente ST: *)
MyName := INSERT ('Mr JONES', 'Frank ', 4);
(* la variabile MyName contiene 'Mr Frank JONES' *)

(* Equivalente IL: *)
LD          'Mr JONES'
INSERT      'Frank ',4
ST          MyName
```

LEFT



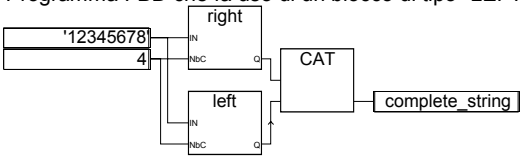
Argomenti:

IN	MSG	Qualsiasi stringa non vuota
NbC	INT	Numero di caratteri da estrarre non può essere superiore alla lunghezza della stringa IN
Q	MSG	la parte a sinistra della stringa IN (con lunghezza = NbC) stringa vuota se NbC <= 0 la stringa IN completa nel caso NbC >= lunghezza stringa IN

Descrizione:

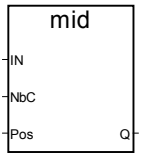
Estrazione della parte sinistra di una stringa messaggio, dato il numero di caratteri da estrarre.

(* Programma FBD che fa uso di un blocco di tipo "LEFT" e "RIGHT" *)




```
(* Equivalente ST: *)
complete_string := RIGHT ('12345678', 4) + LEFT ('12345678', 4);
(* complete_string vale '56781234'
il valore risultante dalla chiamata a RIGHT è '5678'
il valore risultante dalla chiamata a LEFT è '1234'
*)

(* Equivalente IL: viene chiamato per primo LEFT *)
LD          '12345678'
LEFT        4
ST          sub_string  (* risultato intermedio *)
LD          '12345678'
RIGHT       4
ADD         sub_string
ST          complete_string
```

MID



Argomenti:

IN	MSG	qualsiasi stringa non vuota
NbC	INT	Numero di caratteri da estrarre non può essere superiore alla lunghezza della stringa IN
Pos	INT	posizione della sottostringa il primo carattere della sottostringa è quello cui si riferisce Pos (la prima posizione valida è 1)
Q	MSG	una parte contenuta nella stringa (con lunghezza = NbC) stringa vuota se parametri non corretti

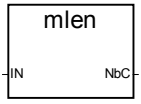
Descrizione:

Estrazione di una parte di una stringa messaggio, dati il numero di caratteri da estrarre e la posizione del primo carattere.

(* Programma FBD che fa uso di un blocco di tipo "MID" *)


```
(* Equivalente ST: *)
sub_string := MID ('abcdefgh', 2, 4);
(* sub_string vale 'de' *)

(* Equivalente IL: *)
LD      'abcdefgh'
MID     2,4
ST      sub_string
```

MLEN



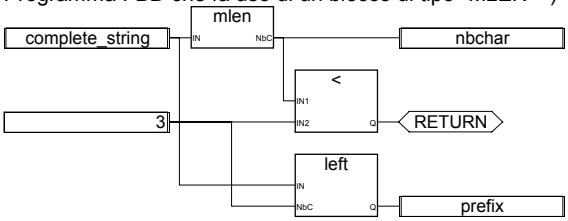
Argomenti:

IN	MSG	qualsiasi stringa messaggio
NbC	INT	numero di caratteri della stringa IN

Descrizione:

Calcola la lunghezza di una stringa messaggio.

(* Programma FBD che fa uso di un blocco di tipo "MLEN" *)

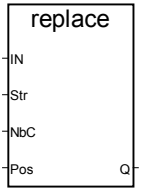


(* Equivalente ST: *)
nbchar := MLEN (complete_string);
If (nbchar < 3) Then Return; End_if;
prefix := LEFT (complete_string, 3);
(* questo programma estrae i 3 caratteri alla sinistra della stringa ed assegna risultato alla variabile messaggio prefix
non viene compiuta alcuna operazione se la lunghezza della stringa è inferiore ai 3 caratteri *)

(* Equivalente IL: *)

```
LD          complete_string
MLEN
ST          nbchar
LT          3
RETC
LD          complete_string
LEFT        3
ST          prefix
```

REPLACE



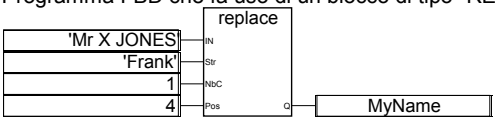
Argomenti:

IN	MSG	qualsiasi stringa
Str	MSG	stringa da inserire (per sostituire NbC caratteri)
NbC	INT	numero di caratteri da cancellare
Pos	INT	posizione del primo carattere modificato (la prima posizione valida è 1)
Q	MSG	stringa modificata: - NbC caratteri vengono cancellati dalla posizione Pos - la sottostringa Str viene inserita in questa posizione restituisce una stringa vuota se Pos <= 0 restituisce la concatenazione delle stringhe (IN+Str) se Pos è maggiore della lunghezza della stringa IN

restituisce la stringa iniziale IN se NbC <= 0

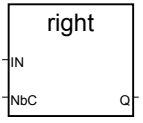
Descrizione:
Sostituisce una parte di un messaggio con un nuovo insieme di caratteri.

(* Programma FBD che fa uso di un blocco di tipo "REPLACE" *)




```
(* Equivalente ST: *)
MyName := REPLACE ('Mr X JONES, 'Frank', 1, 4);
(* MyName vale 'Mr Frank JONES' *)

(* Equivalente IL: *)
LD          'Mr X JONES'
REPLACE     'Frank',1,4
ST          MyName
```

RIGHT

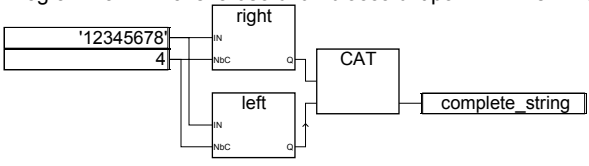


Argomenti:

IN	MSG	Qualsiasi stringa non vuota
NbC	INT	non può essere superiore alla lunghezza della stringa IN
Q	MSG	la parte a destra della stringa IN (con lunghezza = NbC)) stringa vuota se NbC <= 0 la stringa IN completa nel caso NbC >= lunghezza stringa IN

Descrizione:
Estrazione della parte destra di una stringa messaggio, dato il numero di caratteri da estrarre.

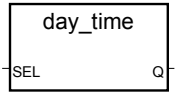
(* Programma FBD che fa uso di un blocco di tipo "LEFT" e "RIGHT" *)



(* Equivalente ST: *)

```
complete_string := RIGHT ('12345678', 4) + LEFT ('12345678', 4);
(* complete_string vale '56781234'
the value issued from RIGHT call is '5678'
the value issued from LEFT call is '1234'
*)

(* IL Equivalence: First done is call to LEFT *)
LD          '12345678'
LEFT        4
ST          sub_string  (* risultato intermedio *)
LD          '12345678'
RIGHT       4
ADD         sub_string
ST          complete_string
```

DAY_TIME



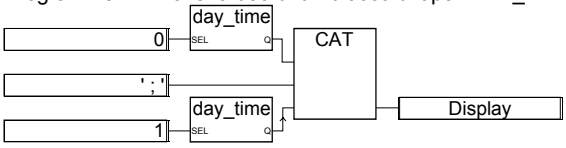
Argomenti:

SEL	INT	selezione output 0= data corrente 1= orario corrente 2= giorno della settimana
Q	MSG	orario/data espresso da una stringa di caratteri 'YYYY/MM/DD' se SEL = 0 'HH:MM:SS' se SEL = 1 nome del giorno se SEL = 2 (es: 'Lunedì')

Descrizione:

Restituisce la data o l'orario del giorno in formato stringa messaggio.

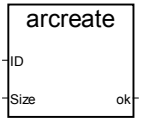
(* Programma FBD che fa uso di un blocco di tipo "DAY_TIME" *)




```
(* Equivalente ST: *)
Display := Day_Time (0) + ' ; ' + Day_Time (1);
(* Il formato di visualizzazione del messaggio è: 'YYYY/MM/DD ; HH:MM:SS' *)

(* IL Equivalence: La prima chiamata è day_time(1) *)
LD          1
DAY_TIME
ST          hour_str     (* risultato intermedio *)
```

```
LD          0
DAY_TIME
ADD         ' ; '
ADD         hour_str
ST          Display
```

ARCREATE



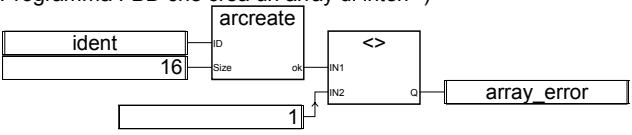
Argomenti:

ID	INT	Identificativo dell'array (deve essere compreso tra [0..15])
Size	INT	numero di elementi dell'array
ok	INT	resoconto dell'operazione: **1** = se l'array è stato creato **2** = identificativo dell'array errato o array già creato **3** = dimensione (size) dell'array non corretta **4** = memoria non sufficiente

Descrizione:
Creazione di un array di interi.

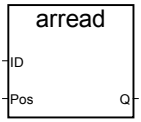
Avvertenza: In un'applicazione, sono disponibili al più **16** array. Gli array contengono valori **interi analogici**. Poiché l'allocazione della memoria avviene in modo dinamico, questa funzione può causare un errore di sistema qualora la dimensione dell'array sia prossima alla dimensione della memoria disponibile.

(* Programma FBD che crea un array di interi *)




```
(* Equivalente ST: *)
array_error := (ARCREATE (ident, 16) <> 1));

(* Equivalente IL: *)
LD          ident
ARCREATE    16
NE          1
ST          array_error
```

ARREAD

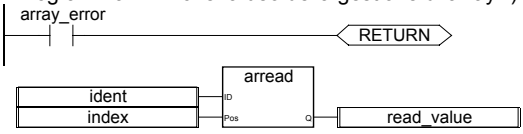


Argomenti:

ID	INT	Identificativo dell'array (deve essere compreso tra [0..15])
Pos	INT	posizione dell'elemento nell'array deve essere compreso tra [0 .. dimensione-1]
value	INT	valore dell'elemento letto 0 se gli argomenti non sono corretti

Descrizione:
Lettura di un elemento di un array di interi.

(* Programma FBD che fa uso della gestione di array *)




```
(* Equivalente ST: *)
If (array_error) Then Return; End_if;
read_value := ARREAD (ident, index);
(* array_error deriva da una chiamata a ARCREATE *)

(* Equivalente IL: *)
LD          array_error
RETC
LD          ident
ARREAD      index
ST          read_value
```

ARWRITE

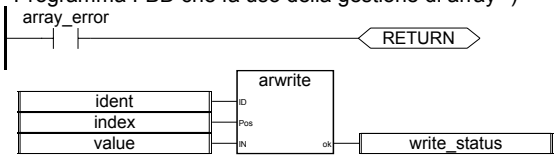
Argomenti:

ID	INT	Identificativo dell'array (deve essere compreso tra [0..15])
Pos	INT	posizione dell'elemento nell'array deve essere compreso tra [0 .. dimensione-1]
IN	INT	nuovo valore per l'elemento
ok	INT	resoconto dell'operazione **1** = scrittura avvenuta

2 = identificativo dell'array errato
3 = indice errato

Descrizione:
memorizza (scrive) un valore in un array di interi.

(* Programma FBD che fa uso della gestione di array *)




```
(* Equivalente ST: *)
If (array_error) Then Return; End_if;
write_status := ARWRITE (Ident, Index, value);
(*array_error deriva da una chiamata a ARCREATE *)

(* Equivalente IL: *)
LD          array_error
RETC
LD          ident
ARWRITE     index,value
ST          write_status
```

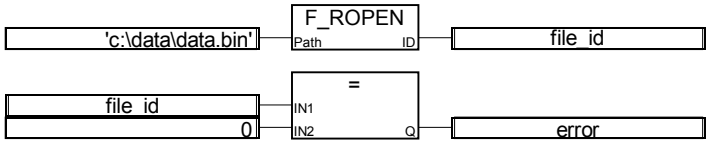
F_ROPEN

Argomenti:

Path	MSG	nome file Può includere il percorso (path) di accesso al file con i simboli \ o / per specificare una directory. Per facilitare la portabilità dell'applicazione, / o \ sono equivalenti.
ID	INT	numero file 0 se si è verificato un errore: il file non esiste.

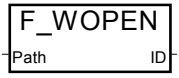
Descrizione:
Apre in file binario in modo lettura, per l'uso con FX_READ e F_CLOSE.
Questa funzione non è inclusa nel simulatore ISaGRAF.

(* Programma FBD che fa uso della gestione file *)




```
(* Equivalente ST: *)
file_id := F_ROPEN('c:\data \data.bin');
error := (file_id=0);

(* Equivalente IL: *)
LD          'c:\data\data.bin'
F_ROPEN
ST          file_id
EQ          0
ST          error
```

F_WOPEN



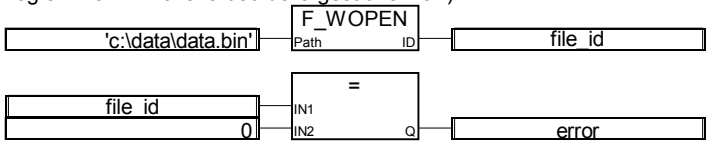
Argomenti:

Path	MSG	nome file Può includere il path (percorso) di accesso al file con i simboli \ o / per specificare una directory. Per facilitare la portabilità dell'applicazione, / o \ sono equivalenti.
ID	INT	numero file 0 se si è verificato un errore. Un eventuale file già esistente viene sovrascritto.

Descrizione:

Apre un file **binario** in modo scrittura, per l'uso con FX_WRITE e F_CLOSE.
Questa funzione non è inclusa nel simulatore ISaGRAF.

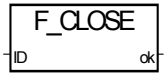
```
(* Programma FBD che fa uso della gestione file *)
```




```
(* Equivalente ST: *)
file_id := F_WOPEN('c:\data\data.bin');
error := (file_id=0);

(* Equivalente IL: *)
LD          'c:\data\data.bin'
F_WOPEN
```

```
ST          file_id
EQ          0
ST          error
```

F_CLOSE



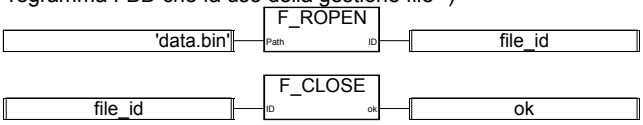
Argomenti:

ID	INT	numero file: restituito da F_ROPEN o F_WOPEN.
ok	BOO	stato TRUE se la chiusura del file è OK FALSE se si è verificato un errore

Descrizione:

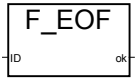
Chiude un file **binario** aperto con le funzioni F_ROPEN o F_WOPEN.
Questa funzione non è inclusa nel simulatore ISaGRAF.

(* Programma FBD che fa uso della gestione file *)




```
(* Equivalente ST: *)
file_id := F_ROPEN('data.bin');
ok := F_CLOSE(file_id);

(* Equivalente IL: *)
LD          'data.bin'
F_ROPEN
ST          file_id
F_CLOSE                 (* file_id è già nel risultato corrente IL *)
ST          ok
```

F_EOF



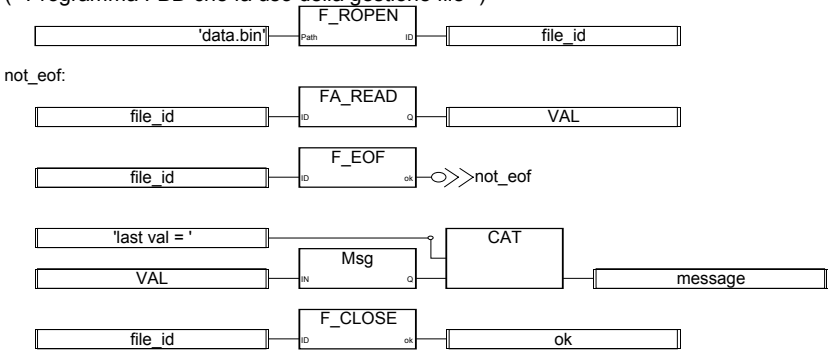
Argomenti:

ID	INT	numero file restituito da F_ROPEN o F_WOPEN.
ok	BOO	indicatore di fine file (end of file)

TRUE se è stata raggiunta la fine del file nella precedente operazione di lettura o scrittura.
Con FM_READ, l'ultimo messaggio letto da un file potrebbe non essere corretto, qualora l'ultimo carattere non sia il terminatore di stringa.

Descrizione:
Controlla se sia stata raggiunta la fine del file.
Questa funzione non è inclusa nel simulatore ISaGRAF.

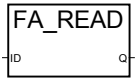
(* Programma FBD che fa uso della gestione file *)




```
(* Equivalente ST: *)
file_id := F_ROPEN('data.bin');
WHILE not(F_EOF(file_id))
        VAL := FA_READ(file_id);
END_WHILE;
MESSAGE := 'last val = ' + msg(VAL);
ok := F_CLOSE(file_id);

(* Equivalente IL: *)
                LD      'data.bin'
                F_ROPEN
                ST      file_id
                LD      file_id
                F_EOF
                JMPC    END_OF_FILE
NOT_EOF:        LD      file_id
                FA_READ
                ST      VAL
                LD      file_id
                F_EOF
                JMPNC   NOT_EOF  (* se non eof, continua la lettura *)
END_OF_FILE:    LD      VAL
                MSG
                ST      val_msg     (* conversione di VAL in messaggio *)
                LD      'last val = '
```

```
ADD       val_msg
ST        MESSAGE
LD        file_id
F_CLOSE
ST        ok
```

FA_READ



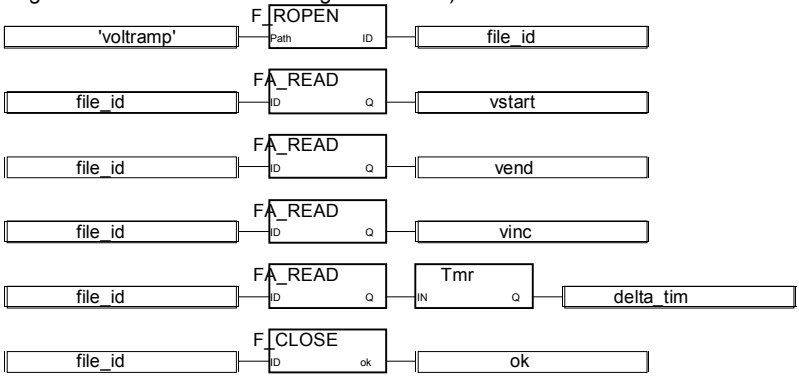
Argomenti:

ID	INT	numero file: restituito da F_ROPEN.
Q	INT	valore intero analogico letto dal file

Descrizione:

Lettura di una variabile ANALOGICA da un file binario. Da usarsi assieme in concomitanza con F_ROPEN e F_CLOSE.
Questa procedura effettua un accesso sequenziale al file, dalla precedente posizione.
La prima chiamata successiva a F_ROPEN legge i primi 4 byte del file,
ogni chiamata sposta il puntatore di lettura.
Usare F_EOF per controllare se sia stata raggiunta la fine del file.
Questa funzione non è inclusa nel simulatore ISaGRAF.

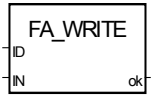
(* Programma FBD che fa uso della gestione file *)




```
(* Equivalente ST: *)
file_id := F_ROPEN('voltramp.bin');
vstart := FA_READ(file_id);
vend := FA_READ(file_id);
vinc := FA_READ(file_id);
delta_tim := tmr(FA_READ(file_id));
ok := F_CLOSE(file_id);
```

```
(* Equivalente IL: *)
        LD      'voltramp.bin'
        F_ROPEN
        ST      file_id
        FA_READ             (* lettura vstart *)
        ST      vstart
        LD      file_id
        FA_READ             (*lettura vend *)
        ST      vend
        LD      file_id
        FA_READ             (*lettura vinc *)
        ST      vinc
        LD      file_id
        FA_READ             (*lettura delta_tim *)
        TMR                 (* conversione a timer *)
        ST      delta_tim
        LD      file_id
        F_CLOSE
        ST      ok
```

FA_WRITE



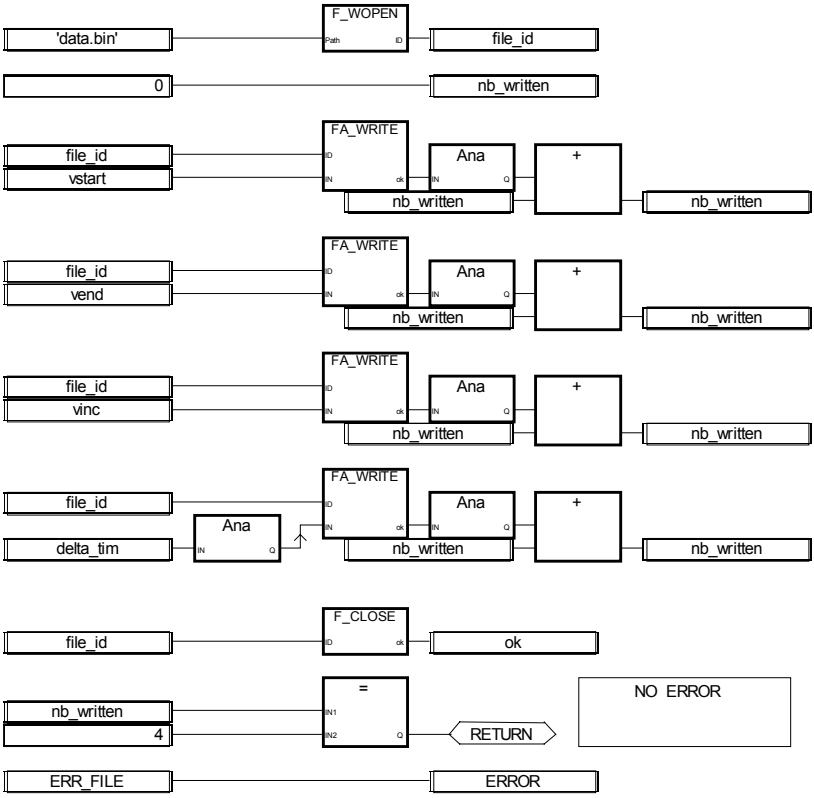
Argomenti:

ID	INT	numero file: restituito da F_WOPEN.
IN	INT	valore intero analogico che deve essere scritto nel file.
OK	BOO	stato: TRUE se ok

Descrizione:

Scrittura di una variabile ANALOGICA in un file binario.
Questa procedura effettua un accesso sequenziale al file, dalla precedente posizione.
La prima chiamata successiva a F_WOPEN scrive i primi 4 byte del file,
ogni chiamata sposta il puntatore di lettura.
Questa funzione non è inclusa nel simulatore ISaGRAF.

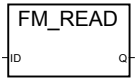
(* Programma FBD *)




```
(* Equivalente ST: *)
file_id := F_WOPEN('voltramp.bin');
nb_written  := 0;
nb_written := nb_written + ana(FA_WRITE(file_id,vstart));
nb_written := nb_written + ana(FA_WRITE(file_id,vend));
nb_written := nb_written + ana(FA_WRITE(file_id,vinc));
nb_written := nb_written + ana(FA_WRITE(file_id,ana(delta_tim)));
ok := F_CLOSE(file_id);
IF ( nb_written <> 4) THEN
        ERROR := ERR_FILE;
END_IF;

(* Equivalente IL: *)
                LD          'voltramp.bin'
                F_ROPEN
                ST          file_id
                LD          0
```

```
ST          nb_written
LD          file_id       (* scrittura vstart *)
FA_WRITE    vstart
ANA
ADD         nb_written
ST          nb_written
LD          file_id       (*scrittura vend *)
FA_WRITE    vend
ANA
ADD         nb_written
ST          nb_written
LD          file_id       (*scrittura vinc *)
FA_WRITE    vinc
ANA
ADD         nb_written
ST          nb_written
LD          delta_tim     (*scrittura delta_tim *)
ANA                       (* conversione ad intero *)
ST          ana_delta_tim
LD          file_id
FA_WRITE    ana_delta_tim
ANA
ADD         nb_written
ST          nb_written
F_CLOSE
ST          ok
LD          nb_written
EQ          4
RETC                      (* ritorno se uguale a 4 *)
LD          ERR_FILE      (* altrimenti errore *)
ST          ERROR
```

FM_READ



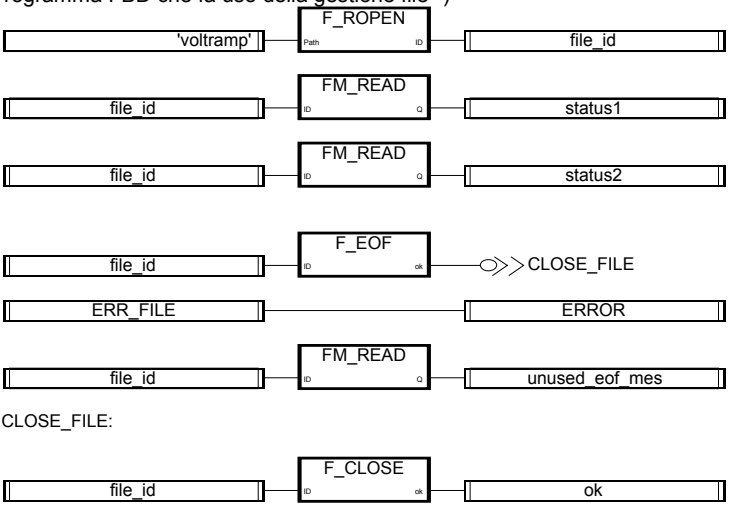
Argomenti:

ID	INT	numero file: restituito da F_ROPEN.
Q	MSG	messaggio letto dal file

Descrizione:

Lettura di variabili MESSAGGIO da file binario.
Da usarsi in concomitanza con F_ROPEN e F_CLOSE.
Questa procedura effettua un accesso sequenziale al file, dalla precedente posizione.
La prima chiamata successiva a F_ROPEN legge la prima stringa del file,
ogni chiamata sposta il puntatore di lettura.
Una stringa viene conclusa da null (0), end of line ('\n') o return ('\r');
Usare F_EOF per controllare se sia stata raggiunta la fine del file.
Questa funzione non è inclusa nel simulatore ISaGRAF.

(* Programma FBD che fa uso della gestione file *)

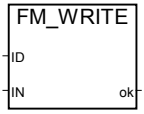



```
(* Equivalente ST: *)
file_id := F_ROPEN('voltramp.bin');
status1 := FM_READ(file_id);
status2 := FM_READ(file_id);
IF (F_EOF(file_id)) THEN
        ERROR := ERR_FILE;
        unused_eof_mes := FM_READ(file_id);
END_IF;
ok := F_CLOSE(file_id);
```

```
(* Equivalente IL: *)
                LD      'voltramp.bin'
                F_ROPEN
                ST      file_id
                FM_READ             (* lettura status1 *)
                ST      status1
                LD      file_id
                FM_READ             (* lettura status2 *)
                ST      status2
                LD      file_id
                F_EOF
                JMPNC   CLOSE_FILE (* se non end of file salta sotto *)
                LD      ERR_FILE
                ST      ERROR
                LD      file_id
                FM_READ             (* lettura unused_eof_mes *)
                ST      unused_eof_mes
CLOSE_FILE      LD      file_id
```

```
F_CLOSE
ST          ok
```

FM_WRITE



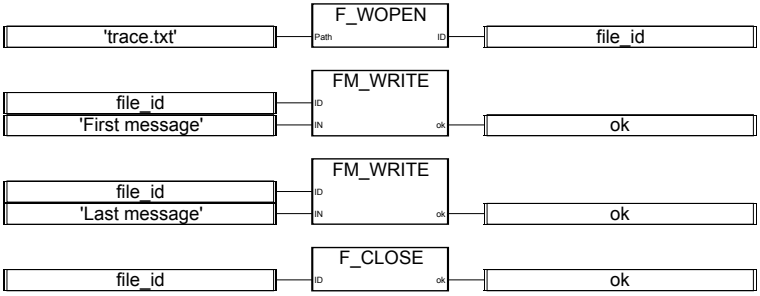
Argomenti:

ID	INT	numero file: restituito da F_WOPEN.
IN	MSG	messaggio che deve essere scritto nel file
ok	BOO	stato TRUE se operazione riuscita

Descrizione:

Scrittura di variabili MESSAGGIO su file binario.
Da usarsi in concomitanza con F_WOPEN e F_CLOSE.
Un messaggio scritto su file ha formato stringa terminata da null.
Questa procedura effettua un accesso sequenziale al file, dalla precedente posizione.
La prima chiamata successiva a F_WOPEN scrive la prima stringa sul file,
ogni chiamata sposta il puntatore di scrittura.
Questa funzione non è inclusa nel simulatore ISaGRAF.

(* Programma FBD che fa uso della gestione file *)




```
(* Equivalente ST: *)
file_id := F_WOPEN('trace.txt');
ok := FM_WRITE(file_id,'First message');
ok := FM_WRITE(file_id,'Last message');
ok := F_CLOSE(file_id);

(* Equivalente IL: *)
            LD          'trace.txt'
            F_WOPEN
            ST          file_id
```

```
FM_WRITE'First message'      (* scrittura primo msg *)
ST          ok
LD          file_id
FM_WRITE'Last message'       (* scrittura secondo msg *)
ST          ok
LD          file_id
F_CLOSE
ST          ok
```

C. Guida utente al target

C.1 Introduzione

Per target ISaGRAF si intende un software in grado di eseguire in tempo reale un'applicazione ISaGRAF su un computer industriale, sistema o scheda, secondo il seguente ben noto schema:

Il ciclo del target consiste nella scansione degli ingressi fisici del processo da pilotare, l'elaborazione dei dati dell'applicazione secondo i programmi dell'ambiente di lavoro ISaGRAF [1], e quindi nell'aggiornamento delle uscite fisiche.

- La prima parte di questa sezione spiega come iniziare ad utilizzare il target di uno specifico sistema. Rispettivamente i target DOS, OS-9, VxWorks ed NT. Per ciascun sistema, verrà inizialmente spiegato come avviare il target ISaGRAF. Successivamente verranno esposte le informazioni relative a caratteristiche specifiche come: avvio del target all'accensione, gestione degli errori, norme generali, …

- La seconda parte tratta il metodo di implementazione da parte dell'utente di funzioni C, blocchi funzione e funzioni di conversione, che permettono di completare il target ISaGRAF.

- La terza parte fornisce informazioni su Modbus e l'implementazione ISaGRAF. Descrive il formato degli schemi dei differenti codici funzione.

- La quarta parte fornisce alcuni strumenti per gestire le interruzioni di energia elettrica ed il riavvio del target.

[1] Questo manuale prevede la conoscenza dell'ambiente di sviluppo ISaGRAF

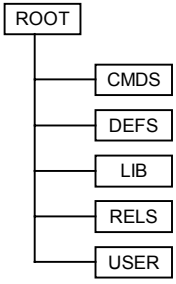
C.2 Installazione

L'installazione richiede circa 1 Mbyte di spazio libero su disco.

Il file batch install.bat fornito con il disco, provvede ad installare su PC tutti i file necessari per una piattaforma specifica.

Esempio: a:\install a: c:*path*
installerà i file da disco a: al disco c: directory *path*.

Lo schema delle directory è il seguente :



la directory ROOT contiene alcuni programmi ed i file readme (file leggimi)
la directory CMDS contiene i file eseguibili
la directory DEFS contiene i file header
la directory LIB contiene le librerie
la directory RELS contiene i file rilocabili (file oggetto)
la directory USER contiene le procedure utente 'C' per funzioni, blocchi funzione e funzioni di conversione C (sorgente ed header)

Basta poi iniziare con la piattaforma installata.

C.3 Target ISaGRAF per DOS

C.3.1 Eseguire ISaGRAF: ISA.EXE

Nell'implementazione MS-DOS, il target viene eseguito come singolo programma: ISA.EXE
Per iniziare è possibile avviare il comando di guida isa -? dalla directory CMDS.

Con questo sistema, le operazioni possono essere critiche. Si raccomanda di non sovraccaricare la parte relativa alle comunicazioni per ottenere buone prestazioni.
Il programma target non è in grado di prevenire l'esecuzione di routine gestite da interrupt.

Collegamento e configurazione comunicazioni: opzione –t

Il target ISaGRAF fa uso di un collegamento seriale per le comunicazioni in fase di debug. È possibile definire il nome della porta con l'opzione –t. L'interfaccia di comunicazione è progettata per essere compatibile con qualsiasi macchina e, quindi, si possono usare le porte COM1, COM2 o COM3, a seconda della versione del BIOS.

Non è previsto valore predefinito: Senza questa opzione, non è possibile alcuna comunicazione con il target. In questo caso, compare l'errore numero 7.

Il target ISaGRAF per DOS non permette comunicazioni con collegamento Ethernet. Implementazioni speciali possono essere richieste ai fornitori.

I parametri di comunicazione devono essere impostati prima dell'avvio di ISaGRAF, in modo che l'utente sia libero di assegnare i parametri di cui necessita. Usando l'ambiente di debug, è necessario assicurarsi (vedere sul Manuale utente: Gestione dei programmi) della corrispondenza tra i parametri di comunicazione dell'ambiente di lavoro e del target.

Esempio:
MODE COM1:9600,N,8,1
I parametri di comunicazione vengono impostati ai seguenti valori:
- baud rate vale 9600
- nessun controllo di parità
- 8 bit di dati
- 1 bit di stop

Si noti che con alcune versioni del BIOS, non è possibile un'impostazione predefinita a 19200 baud. CJ fornisce il programma di utilità ISAMOD.EXE per impostare i parametri dell'ambiente di lavoro:
ISAMOD COM1
equivale a MODE COM1:19200,N,8,1

Numero slave: opzione -s

Questa opzione specifica il numero slave per il target. Sono possibili valori compresi tra 1 e 255, ad eccezione del numero 13 ($0D). Il numero slave viene usato dal protocollo di

comunicazione. Serve principalmente per distinguere i dispositivi slave nel caso ci siano più target collegati tra loro. Quando si usa il debugger dell'ambiente di lavoro, assicurarsi che il parametro slave dell'ambiente (vedere sul Manuale utente: Gestione dei programmi) corrisponda a quello del target.

Valore predefinito: Il valore predefinito del numero slave è 1 (lo stesso dell'ambiente di lavoro)

Esempi:

isamod COM1 Configura COM1 a 19200 baud, nessuna parità, 8 bit di dati, 1 bit di stop.

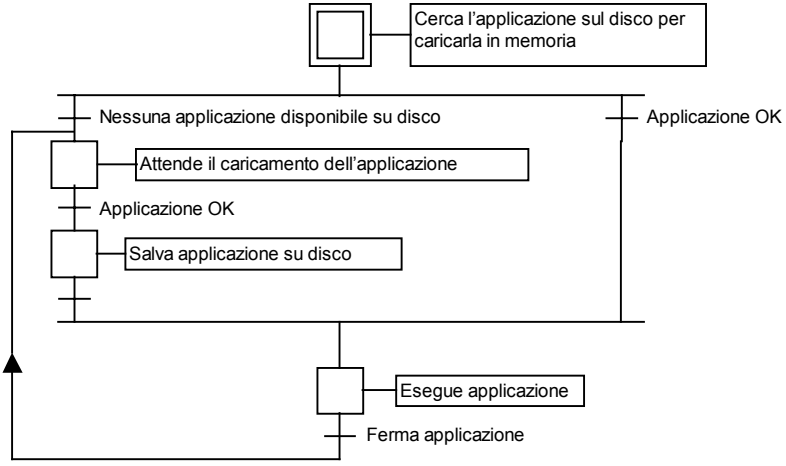
isa -t=COM1 Avvia il target ISaGRAF con il numero slave predefinito (1) e con la porta di comunicazione COM1.

isa -s=3 -t=COM1 Avvia il target ISaGRAF con il numero slave 3 e con COM1 come porta di comunicazione.

C.3.2 Caratteristiche specifiche

Avvio ISaGRAF

All'avvio del target, viene eseguito il seguente algoritmo.



- **Definizioni**

Il codice dell'applicazione consiste nel data base binario generato e scaricato dall'ambiente di lavoro, successivamente eseguito dal target. Può essere completato dalla tavola dei simboli.
La tavola dei simboli dell'applicazione è un data base ASCII generato e scaricato dall'ambiente di lavoro. Questa tavola rappresenta la corrispondenza tra gli oggetti simbolici e gli oggetti interni del target. Non è normalmente richiesta dal target, ad eccezione di quando

sia necessario gestire simboli specifici definiti dall'utente. Ulteriori informazioni sulla tavola dei simboli si possono trovare sul Manuale utente alla sezione : Tecniche di programmazione avanzata.

- **Salvataggio dell'applicazione**

Quando una nuova applicazione viene scaricata sul target dal debugger dell'ambiente di lavoro, il codice dell'applicazione viene salvato nella directory corrente del target, con il nome:

ISAx1 file di salvataggio del codice dell'applicazione ISaGRAF (dove x rappresenta il numero slave)

La tavola dei simboli, anche se scaricata precedentemente, viene salvata nella directory corrente del target con il nome:

ISAx6 file di salvataggio dei simboli dell'applicazione ISaGRAF (dove x rappresenta il numero slave)

All'avvio, il target ISaGRAF, cerca nella directory corrente i file relativi a codice e simboli dell'applicazione e li carica in memoria.
Se non ci sono simboli disponibili, il target avvia l'esecuzione del codice dell'applicazione senza aver caricato i simboli.
Se il codice dell'applicazione non è disponibile, il target rimane in attesa del caricamento dell'applicazione.

Per eseguire il target all'accensione con un'applicazione specifica, senza usare il collegamento con il debugger, questi file possono essere copiati direttamente sul disco della directory corrente del target, dallo stesso disco se l'ambiente di lavoro si trova sullo stesso PC o usando un dischetto floppy in caso contrario. Se la macchina target non possiede un disco, è possibile usare un disco virtuale.

Se l'ambiente di lavoro ISaGRAF è installato nella directory standard \ISAWIN:
il file del codice dell'applicazione relativa al progetto MYPROJ è:
\ISAWIN\APL\MYPROJ\appli.x8m
il file dei simboli dell'applicazione relativa al progetto MYPROJ è:
\ISAWIN\APL\MYPROJ\appli.tst

Esempio:
Dalla directory in cui è installato isa.exe, con il seguente comando:
copy \ISAWIN\APL\MYPROJ\appli.x8m isa11
Il programma isa.exe troverà ed eseguirà l'applicazione 'myproj'.

Tutti questi comandi possono essere raggruppati in un file batch ed avviati dal menu Strumenti dell'ambiente di lavoro (vedere il Manuale utente: Gestione dei programmi).

Gestione degli errori e messaggi di output

Il software target ISaGRAF incorpora una gestione di riconoscimento dell'errore. In appendice è riportata la lista degli errori con la rispettiva descrizione.

La gestione dell'errore avviene nel seguente modo:

- Un errore è composto da un numero di errore e da un numero di argomento che vengono inviati alla routine di gestione degli errori ISaGRAF.

- Se è attivo il flag di gestione errori nelle opzioni per l'esecuzione dell'applicazione dell'ambiente di lavoro, l'errore viene processato. In caso contrario, l'informazione viene tralasciata senza l'intervento della routine di gestione degli errori.

Quando l'errore viene processato:
- Il numero di errore (decimale) e l'argomento (esadecimale) vengono visualizzati sul dispositivo di output predefinito stdout.
- Il numero di errore e il numero di argomento, vengono inseriti in un buffer circolare FIFO, così da poter essere recuperati in un secondo tempo. La dimensione del buffer di gestione degli errori è definibile nelle opzioni per l'esecuzione dell'applicazione dell'ambiente di lavoro. A buffer pieno, ogni nuovo errore sovrascrive il primo contenuto nel buffer stesso in ordine cronologico.
- I codici di errore possono essere recuperati sia dal debugger che dall'applicazione durante l'esecuzione mediante le chiamate SYSTEM (si veda il Manuale utente).

Quando il debugger rileva un errore, viene visualizzato, nella finestra di comunicazione, un messaggio che descrive l'errore. A seconda dello stato dell'applicazione (in esecuzione o meno), il debugger può mostrare il nome dell'oggetto (variabile o programma) che ha causato l'errore, o il numero di argomento (decimale) racchiuso tra parentesi quadre [x], che ha diverso significato a seconda dell'errore.

Un messaggio di avvio ed i codici di errori vengono mostrati sul dispositivo di output predefinito stdout, rispettivamente all'avvio del target e quando si verifica un errore. Se non si vuole che il messaggio appaia sul canale di output predefinito, si può usare un comando di redirezione come:

isa -t=COM1 -s=1 **>NUL**

Clock di sistema

Poiché un target ISaGRAF è progettato per essere eseguito su qualsiasi sistema, il tempo di riferimento usato sia per la sincronizzazione del ciclo, che per l'aggiornamento delle variabili timer, corrisponde al tick standard di circa 55 millisecondi.

Quindi non è possibile un'accuratezza migliore di 55ms per le variabili di tipo timer. Per questa ragione, impostare una durata di ciclo minore od uguale a 55ms e diversa da zero, causa un errore di overflow di durata del ciclo (errore 62) e cicli non temporizzati regolarmente.

Il vantaggio del non modificare il tick di sistema, consiste nel fatto che nessuna delle applicazioni residenti, o delle funzioni C o blocchi funzione integrate nell'applicazione, possono essere disturbate dall'esecuzione ISaGRAF.

Chiedere al fornitore un'implementazione speciale, nel caso l'applicazione necessiti di maggior accuratezza.

Tasto uscita

In fase di prova delle applicazioni non in condizioni "industriali" su un PC desktop, si può voler fermare l'esecuzione del programma ISaGRAF: questo è possibile tramite una combinazione di tasti, piuttosto articolata, in modo da evitare che venga attivata in modo accidentale. La sequenza di tasti è:

shift + ctrl + alt

Naturalmente, le applicazioni industriali non devono poter essere fermate con la pressione di un tasto: è bene disabilitare queste combinazioni.

Un effetto collaterale pericoloso di queste uscite immediate, è costituito dal fatto che l'interfacciamento con le schede IO non viene chiuso. Il modo corretto per fermare il target ISaGRAF è:
- fermare l'applicazione dal debugger (chiudendo così le schede IO)
- fermare il target ISaGRAF dalla tastiera

Dimensione dell'applicazione

Il target ISaGRAF per MS-DOS è progettato per la modalità reale Intel, per questo la dimensione massima di una struttura dati è di 64K. Il codice dell'applicazione scaricato dall'ambiente di lavoro non deve quindi eccedere questo limite. In casi molto rari può succedere che la struttura interna allocata da ISaGRAF ecceda questo limite e provochi il crollo dell'applicazione dopo lo scaricamento. Inoltre, l'intera memoria disponibile è limitata ai 640K della memoria convenzionale.

Chiedere al fornitore un'implementazione speciale, nel caso l'applicazione necessiti di maggiori capacità di memoria.

C.4 Target ISaGRAF per OS9

Per prima cosa è necessario trasferire i file (almeno gli eseguibili dalla directory CMDS) al target OS-9 usando un qualsiasi programma di comunicazione. Per iniziare si può, semplicemente, avviare i comandi di guida dalla directory CMDS del sistema OS-9:

```
isa -?
isaker -?
isatst -?
Isanet -?
```

C.4.1 Eseguire ISaGRAF come processo singolo: isa

Il target ISaGRAF può essere avviato come processo singolo. In questo modo, però, le operazioni possono risultare critiche. Si raccomanda di non sovraccaricare la parte relativa alle comunicazioni per ottenere buone prestazioni. Sul sistema multitasking OS-9, possono essere avviati più processi ISaGRAF singoli sulla stessa CPU, sempre che il loro numero slave e porta di comunicazione siano differenti.
L'implementazione a singolo processo è stata progettata per piattaforme scarsamente dotate dal punto di vista hardware, come ad esempio schede a basso costo o PC MS-DOS, o nel caso della realizzazione di un primo prototipo per il passaggio ad una nuova piattaforma. È quindi preferibile l'uso dell'implementazione del target ISaGRAF multiprocesso.
Il target ISaGRAF a singolo processo non è in grado di prevenire l'esecuzione di processi in parallelo o di routine gestite da interrupt.

Collegamento e configurazione comunicazioni: opzione –t

Il target ISaGRAF a processo singolo, fa uso di un collegamento seriale per le comunicazioni con il debugger. È possibile definire il nome della porta con l'opzione –t.

Non è previsto valore predefinito: Senza questa opzione, non è possibile alcuna comunicazione con il target. In questo caso, compare l'errore numero 7.

Il target ISaGRAF a processo singolo non permette comunicazioni con collegamento Ethernet.

Il dispositivo di collegamento seriale viene aperto in modo trasferimento dati binario (senza controllo caratteri, senza XON/XOFF). I parametri di comunicazione vengono impostati prima dell'avvio di ISaGRAF, permettendo di assegnare liberamente i parametri necessari. Usando l'ambiente di debug, è necessario assicurarsi (vedere il Manuale utente: Gestione dei programmi) della corrispondenza tra i parametri di comunicazione dell'ambiente debug e del target.

Esempio:
xmode /t0 baud=19200
Imposta il baud rate di comunicazione a 19200 baud sul dispositivo /t0

Numero slave: opzione -s

Questa opzione specifica il numero slave per il target. Sono possibili valori compresi tra 1 e 255, ad eccezione del numero 13 ($0D). Il numero slave viene usato dal protocollo di comunicazione. Serve principalmente per distinguere i dispositivi slave nel caso ci siano più target in esecuzione. Quando si usa il debugger dell'ambiente di lavoro, assicurarsi che il parametro slave dell'ambiente debug (vedere il Manuale utente: Gestione dei programmi) corrisponda a quello di un target esistente.

Valore predefinito: Il valore predefinito del numero slave è 1 (lo stesso dell'ambiente di lavoro)

Esempi:

isa -t=/t0 Avvia un target ISaGRAF a processo singolo con il numero slave predefinito (1) e la porta di comunicazione /t0.

isa -s=3 -t=/t1 Avvia un target ISaGRAF a processo singolo con il numero slave 3 e la porta di comunicazione /t1.

isa -t=/t0 &
isa -s=3 -t=/t1 Avvia due target ISaGRAF a processo singolo. Uno con il numero slave predefinito (1) e la porta di comunicazione /t0. L'altro con il numero slave 3 e la porta di comunicazione /t1.

C.4.2 Eseguire processi multipli ISaGRAF: isaker, isatst, isanet

Per migliorare il tempo di risposta del kernel del target ISaGRAF e delle comunicazioni, il target viene suddiviso in due processi distinguendo la parte relativa alle comunicazioni (il processo di comunicazione isatst o isanet) dall'esecuzione dell'applicazione (processo relativo al kernel isaker).
Questa architettura risulta molto più flessibile. Consente di avviare più processi di comunicazione collegati allo stesso processo kernel, oppure fino a 4 kernel collegati allo stesso processo di comunicazione. In questo modo vengono semplificate alcune integrazioni come il collegamento di un processo di visualizzazione ed il debugger dell'ambiente di lavoro alla stessa applicazione o un collegamento singolo tramite la stessa porta fisica a diverse applicazioni (fino a 4).

I processi relativi al kernel ed alle comunicazioni possono essere fatti partire indipendentemente. L'unico requisito è costituito dal fatto che il processo/i kernel deve essere avviato per primo per permettergli di inizializzare l'ambiente di sistema e per permettere al processo/i di comunicazione di potersi collegare ad esso.

Il target ISaGRAF multiprocesso non è in grado di prevenire l'esecuzione di processi in parallelo o di routine gestite da interrupt.

C.4.2.1 Esecuzione del processo kernel: isaker

Numero slave: opzione -s

Questa opzione specifica il numero slave per il kernel del target. Sono possibili valori compresi tra 1 e 255, ad eccezione del numero 13 ($0D). Il numero slave viene usato dal

protocollo di comunicazione e dal processo(i) collegato al kernel. Serve principalmente per distinguere i dispositivi slave nel caso ci siano più target in esecuzione.

Valore predefinito: Il valore predefinito del numero slave è 1 (lo stesso dell'ambiente di lavoro)

C.4.2.2 Esecuzione del processo di comunicazione seriale: isatst

Collegamento e configurazione comunicazioni: opzione –t

Il processo isatst di comunicazione del target, fa uso di un collegamento seriale per le comunicazioni con il debugger. È possibile definire il nome del descrittore con l'opzione –t.

Non è previsto valore predefinito: Senza questa opzione, non è possibile alcuna comunicazione con il target. In questo caso, compare l'errore numero 7.

L'implementazione del processo isatst non prevede comunicazioni con collegamento Ethernet.

Il dispositivo di collegamento seriale viene aperto in modo trasferimento dati binario (senza controllo caratteri, senza XON/XOFF). I parametri di comunicazione vengono impostati prima dell'avvio di ISaGRAF, permettendo di assegnare liberamente i parametri necessari. Usando l'ambiente di debug, è necessario assicurarsi (vedere il Manuale utente: Gestione dei programmi) della corrispondenza tra i parametri di comunicazione dell'ambiente debug e del target.

Esempio:
xmode /t0 baud=19200
Imposta il baud rate di comunicazione a 19200 baud sul dispositivo /t0

Numero slave: opzione -s

Questa opzione specifica il numero(i) slave del kernel del target cui è connesso del processo di comunicazione. Sono possibili valori compresi tra 1 e 255, ad eccezione del numero 13 ($0D). Questa opzione può essere ripetuta fino a 4 volte per collegare fino a 4 diversi kernel slave. Il numero slave consente di distinguere i dispositivi slave nel caso ci siano più target in esecuzione. Quando si usa il debugger dell'ambiente di lavoro, assicurarsi che il parametro slave dell'ambiente debugger (vedere il Manuale utente: Gestione dei programmi) corrisponda a quello di un target esistente.

Valore predefinito: Il valore predefinito del numero slave è 1 (lo stesso dell'ambiente di lavoro)

Numero logico di un processo di comunicazione: opzione -c

Questa opzione specifica il numero logico di un processo di comunicazione. Serve a gestire più di un processo di comunicazione per volta. Può essere compreso tra 1 e 255 e deve essere diverso per ogni processo di comunicazione.

Valore predefinito: Viene usata la precedente opzione –s specificata. Il valore predefinito assicura la compatibilità con le versioni precedenti (3.0) di ISaGRAF.

C.4.2.3 Esecuzione del processo di comunicazione Ethernet: isanet

Collegamento e configurazione comunicazioni: opzione –t

Il processo di comunicazione del target fa uso di una connessione Ethernet per le comunicazioni con il debugger. Il numero di porta viene specificato con l'opzione -t.

Non è previsto valore predefinito: Senza questa opzione, non è possibile alcuna comunicazione con il target. In questo caso, compare l'errore numero 7.

Usando il debugger dell'ambiente di lavoro, assicurarsi che i parametri di comunicazione dell'ambiente di lavoro (si veda il Manuale utente: Gestione dei programmi) corrispondano a quelli del target.

Per ISaGRAF, il target OS-9 corrisponde al server ed il debugger al client che si connette al numero di porta specificato.
Prima di avviare la prima sessione di debug con Ethernet, assicurarsi che il dispositivo Ethernet di OS-9 sia configurato correttamente, inviando, ad esempio, un ping al sistema OS-9.

Numero slave: opzione -s

Questa opzione specifica il numero(i) slave del kernel target cui è connesso il processo di comunicazione. Questa opzione può essere ripetuta fino a 4 volte per collegare fino a 4 diversi kernel di tipo slave. Il numero slave consente di distinguere i dispositivi slave nel caso ci siano più target in esecuzione. Quando si usa il debugger dell'ambiente di lavoro, assicurarsi che il parametro slave dell'ambiente (vedere il Manuale utente: Gestione dei programmi) corrisponda a quello di un target esistente (processi kernel e di comunicazione).

Valore predefinito: Il valore predefinito del numero slave è 1 (lo stesso dell'ambiente di lavoro)

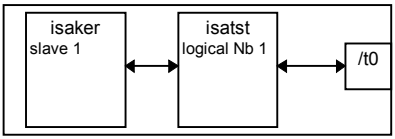
Numero logico di un processo di comunicazione: opzione -c

Questa opzione specifica il numero logico di un processo di comunicazione. Serve a gestire più di un processo di comunicazione per volta. Può essere compreso tra 1 e 255 e deve essere diverso per ogni processo di comunicazione.

Valore predefinito: Viene usata la precedente opzione –s specificata. Il valore predefinito assicura la compatibilità con le versioni precedenti (3.0) di ISaGRAF.

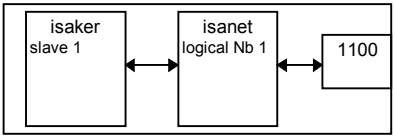
C.4.2.4 Esempi:

isaker &
isatst -t=/t0



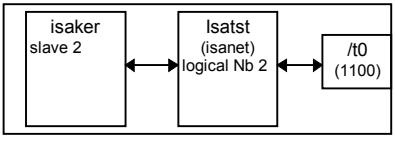
Avvia:
Un processo kernel ISaGRAF con il numero slave predefinito (1).
Un processo di comunicazione seriale ISaGRAF, sulla porta seriale /t0, connesso al numero slave predefinito (1) e con il numero logico predefinito (l'ultimo numero slave specificato = valore predefinito = 1).

isaker &
isanet -t=1100



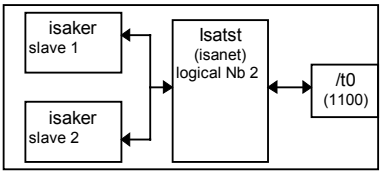
Avvia:
Un processo kernel ISaGRAF con numero slave predefinito (1).
Un processo di comunicazione Ethernet ISaGRAF, sulla porta 1100, connesso al numero slave predefinito (1), con il numero logico predefinito (ultimo numero slave specificato = predefinito = 1).

isaker -s=2 &
isatst -t=/t0 -s=2 (rispettivamente isanet -t=1100 -s=2)



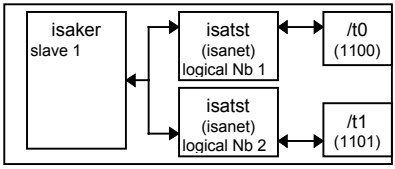
Avvia:
Un processo kernel ISaGRAF con numero slave 2
Un processo di comunicazione seriale (Ethernet) ISaGRAF, sulla porta seriale /t0 (Porta numero 1100), connesso al numero slave 2, con numero logico predefinito (ultimo numero slave specificato = 2).

Isaker -s=1 &
isaker -s=2 &
isatst -t=/t0 -s=1 -s=2 (rispettivamente isanet -t=1100 -s=1 -s=2)



Avvia:
Un processo kernel ISaGRAF con numero slave 1
Un processo kernel ISaGRAF con numero slave 2
Un processo di comunicazione seriale (Ethernet) ISaGRAF, sulla porta seriale /t0 (Porta numero 1100), connesso ai numeri slave 1 e 2, con numero logico predefinito (ultimo numero slave specificato = 2).

Isaker -s=1 &
isatst -t=/t0 -s=1 -c=1 & (rispettivamente isanet -t=1100 -s=1 -c=1 &)
isatst -t=/t1 -s=1 -c=2 (rispettivamente isanet -t=1101 -s=1 -c=2)



Avvia:

Un processo kernel ISaGRAF con numero slave 1
Un processo di comunicazione seriale (Ethernet) ISaGRAF, sulla porta seriale /t0 (Porta numero 1100), connesso al numero slave 1, con numero logico 1.
Un processo di comunicazione seriale (Ethernet) ISaGRAF, sulla porta seriale /t1 (Porta numero 1101), connesso al numero slave 1, con numero logico 2.

Nota:
I processi di comunicazione seriale ed Ethernet possono essere usati contemporaneamente.

C.4.3 Caratteristiche specifiche

Collegamento comunicazioni

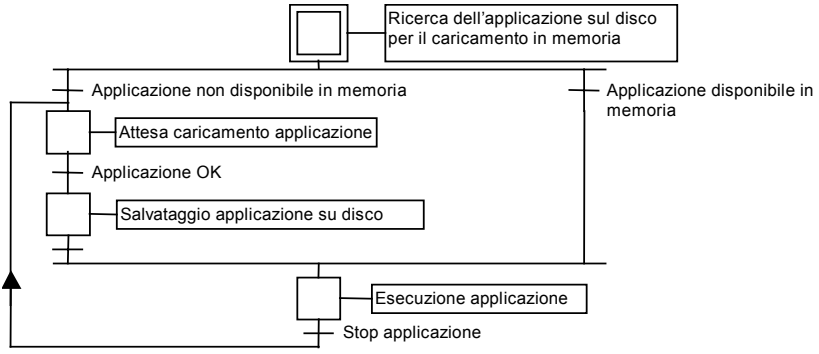
Poiché il Serial Character Manager di OS-9 è molto flessibile, possono essere usati quasi tutti i dispositivi fisici bi-direzionali supportati da Microware :

Esempio:
Il collegamento seriale può essere costituito da un percorso di rete diretto ad una porta fisica presente su una diversa CPU.
Allora l'opzione –t verrà, ad esempio, usata ne seguente modo: -t=/nr/MASTER/t0

In cui la comunicazione viene indirizzata verso una CPU chiamata MASTER su una rete ramnet. La porta fisica usta è /t0.

Avvio di ISaGRAF

All'avvio del target, viene eseguito il seguente algoritmo.



- **Definizioni**

Il codice dell'applicazione è costituito dal data base binario generato e trasferito dall'ambiente di lavoro e successivamente eseguito dal target. Può essere completato dalla tavola dei simboli.
La tavola dei simboli dell'applicazione è un data base ASCII generato e trasferito dall'ambiente di lavoro. Questa tavola rappresenta la corrispondenza tra gli oggetti simbolici e gli oggetti interni del target. Non è normalmente richiesta dal target, ad eccezione di quando sia necessario gestire simboli specifici definiti dall'utente. Ulteriori informazioni sulla tavola dei simboli si possono trovare sul Manuale utente alla sezione: Tecniche avanzate di programmazione.

- **Oggetti OS-9 ISaGRAF ed applicazioni multiple**

Ogni nome di oggetto pubblico in ISaGRAF inizia con '**ISAxn**', dove **x** rappresenta il numero slave del kernel ed **n** una cifra numerica con uno specifico significato, eccetto per **ISAy3** in cui **y** rappresenta il numero logico del processo di comunicazione nell'implementazione multiprocesso.
Su una stessa CPU possono essere eseguite diverse applicazioni (processi kernel e di comunicazione) simultaneamente, con numeri slave diversi e numeri logici del processo di comunicazione diversi. Tuttavia, nel caso vengano eseguite applicazioni diverse, è necessario tener conto dell'accesso condiviso ad alcuni oggetti condivisi delle applicazioni, come le schede I/O. Per esempio applicazioni diverse (kernel) possono usare schede fisiche distinte a meno che non vengano implementati server di I/O o semafori tramite il driver I/O.

Nomi di oggetti OS-9:
File su disco:

ISAx1	file di salvataggio del codice dell'applicazione ISaGRAF
ISAx6	file di salvataggio dei simboli dell'applicazione ISaGRAF

Moduli in memoria:

ISAx0	Dati del kernel di sistema ISaGRAF
ISAx1	Codice dell'applicazione ISaGRAF
ISAx2	Data base in tempo reale del kernel ISaGRAF
ISAy3	Buffer di scambio dati comunicazioni ISaGRAF
ISAx4	Modificazione on line 1 del codice dell'applicazione ISaGRAF
ISAx5	Modificazione on line 2 del codice dell'applicazione ISaGRAF
ISAx6	Simboli dell'applicazione ISaGRAF

È necessario prestare attenzione a non usare questi nomi.

- **Salvataggio dell'applicazione**

Quando una nuova applicazione viene scaricata dal debugger dell'ambiente di lavoro al target, il codice dell'applicazione viene salvato nella directory corrente del target, con il nome:

ISAx1 file di salvataggio del codice dell'applicazione ISaGRAF (dove x rappresenta il numero slave)

La tavola dei simboli, anche se scaricata precedentemente, viene salvata nella directory corrente del target con il nome:

ISAx6 file di salvataggio dei simboli dell'applicazione ISaGRAF (dove x rappresenta il numero slave)

All'avvio del target ISaGRAF, i file relativi a codice e simboli dell'applicazione vengono cercati nella directory corrente e caricati in memoria come moduli dati con gli stessi nomi.

Se la tavola dei simboli non è disponibile, il target avvia l'esecuzione del codice dell'applicazione senza aver caricato i simboli.
Se il codice dell'applicazione non è disponibile, il target rimane in attesa del caricamento dell'applicazione.

Per avviare il target all'accensione con un'applicazione specifica, senza usare il collegamento con il debugger:

- Un primo modo consiste nel copiare direttamente, con un qualsiasi programma di trasferimento, questi file sul disco della directory corrente dal PC host in cui è installato l'ambiente di lavoro. Queste operazioni vengono semplificate usando il menu Strumenti dell'ambiente di lavoro (si veda il Manuale utente: Gestione dei programmi).

- Una seconda maniera è quella di memorizzare il codice dell'applicazione (e, se necessario, la tavola dei simboli dell'applicazione), contenuto nei file del PC host in cui è installato l'ambiente di lavoro, su una memoria non volatile (come una PROM o EPROM), con gli appositi programmi.

  All'accensione del sistema, se richiesto (ad esempio per un accesso più veloce o per la gestione di breakpoint), è possibile caricare il codice dell'applicazione (e, se necessario la tavola dei simboli dell'applicazione) come moduli dati in memoria **ISAx1** (e, se necessario, **ISAx6**) dalla PROM alla RAM, con gli appositi programmi.

  AVVERTENZA:
  La gestione dei breakpoint del debugger ISaGRAF non funziona correttamente se il modulo del codice dell'applicazione non è accessibile in scrittura. Questo non è un problema, poiché l'applicazione è stata provata esaustivamente in precedenza .

Sul PC host, se l'ambiente di lavoro ISaGRAF è installato nella directory standard \ISAWIN:
il file del codice dell'applicazione del progetto MYPROJ è:
\ISAWIN\APL\MYPROJ\appli.x6m (corrispondente a isax1 sul target).
il file dei simboli dell'applicazione del progetto MYPROJ è:
\ISAWIN\APL\MYPROJ\appli.tst (corrispondente a isax6 sul target).

Gestione degli errori e messaggi di output

Il software target ISaGRAF incorpora una gestione di riconoscimento dell'errore. In appendice è riportata la lista degli errori con la rispettiva descrizione.

La gestione dell'errore avviene nel seguente modo:
- Un errore è composto da un numero di errore e da un numero di argomento che vengono inviati alla routine di gestione degli errori ISaGRAF.
- Se è attivo il flag di gestione errori nelle opzioni per l'esecuzione dell'applicazione dell'ambiente di lavoro, l'errore viene processato. In caso contrario, l'informazione viene tralasciata, senza l'intervento della routine di gestione degli errori.

Quando l'errore viene processato:
- Il numero di errore (decimale) e l'argomento (esadecimale) vengono visualizzati sul dispositivo di output predefinito stdout.
- Il numero di errore e di argomento, vengono inseriti in un buffer circolare FIFO, così da poter essere recuperati in un secondo tempo. La dimensione del buffer di gestione degli errori è definibile nelle opzioni per l'esecuzione dell'applicazione dell'ambiente di lavoro. A buffer pieno, ogni nuovo errore sovrascrive il primo contenuto nel buffer stesso in ordine cronologico.
- I codice di errore possono essere recuperati, sia dal debugger che dall'applicazione durante l'esecuzione, mediante le chiamate SYSTEM (si veda il Manuale utente).

Quando il debugger rileva un errore, viene visualizzato, nella finestra di comunicazione, un messaggio che descrive l'errore. A seconda dello stato dell'applicazione (in esecuzione o meno), il debugger può mostrare il nome dell'oggetto (variabile o programma) che ha causato l'errore, o il numero di argomento (decimale) racchiuso tra parentesi quadre [x], che ha diverso significato a seconda dell'errore.

Un messaggio di avvio ed i codici di errori vengono mostrati sul dispositivo di output predefinito stdout, rispettivamente all'avvio del target e quando si verifica un errore. Se non si vuole che il messaggio appaia sul canale di output predefinito, si può usare un comando di ridirezione come:

prog_name [options] **>>>/nil**

Durata del ciclo, funzionamento e priorità dei processi

- Alla fine di un ciclo ISaGRAF, poco prima dell'inizio del successivo, viene eseguito il seguente algoritmo:
  Se è stato specificato un tempo di ciclo (dall'ambiente di lavoro: si veda il Manuale utente: Gestione dei Programmi), allora la CPU viene rilasciata per il tempo rimanente (tempo di ciclo specificato – tempo corrente dell'applicazione). Se il tempo rimanente risulta negativo, viene generato un errore di overflow e viene rilasciata la CPU per il tempo di 1 tick per forzare la schedulazione.

Se non è stato specificato il tempo di ciclo, o se il tempo rimanente è minore o uguale ad un tick, o uguale a zero, allora la CPU viene rilasciata per il tempo di un tick per forzare la schedulazione.

L'accuratezza nella temporizzazione del target corrisponde ad un tick di sistema OS-9.

I cicli vengono comunemente controllati da un tempo di ciclo, così da rilasciare la CPU ad altri processi in esecuzione sul sistema OS-9.

- Il processo di comunicazione è in stato inattivo quando non arrivano dati dal collegamento di comunicazione. Quando necessario, questo processo ottiene informazioni dall'applicazione in esecuzione mediante un protocollo domanda/risposta con il processo kernel. Il processo di comunicazione invia una domanda al kernel. Al termine del ciclo (per mantenere sincrona l'immagine di memoria dell'applicazione), il kernel risponde al processo di comunicazione.

I processi ISaGRAF non modificano la priorità che è stata loro assegnata. È possibile regolare liberamente queste priorità, rispettando il funzionamento dei processi ISaGRAF precedentemente descritto e le esigenze dell'applicazione.
Ad esempio, ci si può assicurare che ISaGRAF non venga interrotto da un processo a bassa priorità modificando i parametri di gestione dei processi di OS-9 come **MIN_AGE** e **MAX_AGE**.

Modo terminal

Il protocollo di comunicazione seriale del target riconosce una sequenza di 3 caratteri INVIO (Codice ASCII $0D) e poi avvia una shell OS-9, se disponibile, sul dispositivo seriale.
Questo permette di avere il prompt di una shell OS-9 su qualsiasi terminale usando il collegamento seriale al target ISAGRAF.

Esempio:
Dal PC host:
- Chiudere il debugger. ISaGRAF
- Avviare una sessione del Terminale Windows (gruppo accessori) con i parametri di comunicazione correttamente impostati
- Premere 3 volte INVIO.

Ora siete collegati ad una shell OS-9
- Scrivere **logout** per uscire dal modo terminal.

AVVERTENZA:
Una sessione in modalità terminal, deve essere sempre conclusa in modo corretto usando logout e niente altro, altrimenti non sarà possibile una successiva connessione con l'ambiente di lavoro.

C.5 Target ISaGRAF per VxWorks

Sul sistema VxWorks, i pochi comandi necessari ad avviare il target ISaGRAF sono relativi alla configurazione dell'ambiente ed all'avvio delle sessioni del target stesso. I comandi possono essere contenuti in un file script. Vengono descritti nei seguenti capitoli.

C.5.1 Il gestore di risorse di sistema: isassr.o

Questo modulo è sempre necessario, in qualsiasi configurazione del target ISaGRAF. Deve essere il primo dei moduli del target ISaGRAF ad essere caricato. Il suo compito è di gestire le risorse di sistema con molteplici target in esecuzione.

C.5.2 Caratteristiche comuni a isa.o, isakerse.o e isakeret.o

L'avvio di ISaGRAF è subordinato ad uno di questi moduli .

isa.o: permette di avviare target ISaGRAF a processo singolo (con comunicazioni solamente seriali).

Isakerse.o: permette di avviare target ISaGRAF multiprocesso (con comunicazioni solamente seriali).

Isakeret.o: permette di avviare target ISaGRAF multiprocesso (con comunicazioni seriali e/o Ethernet)

Maggiori dettagli su questi moduli sono riportati nei prossimi capitoli

Configurazione delle comunicazioni seriali

Il target ISaGRAF, di base, usa un collegamento seriale per le comunicazioni con il debugger. All'apertura, il target ISaGRAF non effettua alcuna operazione di configurazione sul dispositivo di collegamento seriale specificato. In questo modo i parametri necessari possono essere assegnati liberamente. Tuttavia, è richiesto il trasferimento dati in modalità binaria (modo RAW). In questa ottica viene fornita la subroutine *ISAMOD ()*.

```
uchar ISAMOD
  (
  char *desc,        /* Nome del dispositivo seriale */
  uint32 baudrate    /* Baud rate                    */
  )
```

Descrizione:
Configura il dato dispositivo di collegamento seriale per il trasferimento dati in modalità binaria allo specificato baud rate

valore di ritorno:
0 in caso di successo, BAD_RET se si verifica un errore

Usando il debugger dell'ambiente di lavoro, assicurarsi che i parametri di comunicazione dell'ambiente di lavoro (si veda il Manuale utente: Gestione dei programmi) corrispondano a quelli del target.

Frequenza di clock del sistema

La variabile globale CLKRATE (uint32) deve essere inizializzata alla frequenza di clock del sistema VxWorks. Per questo scopo si può usare:

CLKRATE = sysClkRateGet ()

Il valore predefinito di CLKRATE è 60Hz.

C.5.3 Eseguire ISaGRAF come processo singolo: isa.o

Il target ISaGRAF può essere eseguito come processo singolo. In questo modo, però, le operazioni possono risultare critiche. Si raccomanda di non sovraccaricare la parte relativa alle comunicazioni per ottenere buone prestazioni. Sul sistema multitasking VxWorks, possono essere avviati più processi ISaGRAF singoli sulla stessa CPU, sempre che il loro numero slave e porta di comunicazione siano differenti.
L'implementazione a singolo processo è stata progettata per piattaforme scarsamente dotate dal punto di vista hardware, come ad esempio schede a basso costo o PC MS-DOS, o nel caso di realizzazione di un primo prototipo per il passaggio ad una nuova piattaforma. È quindi preferibile l'uso dell'implementazione del target ISaGRAF multiprocesso.
Il target ISaGRAF a singolo processo non è in grado di prevenire l'esecuzione di processi in parallelo o di routine gestite da interrupt.

Registrazione degli slave

Un target ISaGRAF è caratterizzato dal numero slave. Sono possibili valori compresi tra 1 e 255, ad eccezione del numero 13 ($0D). Il numero slave viene usato dal protocollo di comunicazione. È necessario per distinguere i dispositivi slave nel caso ci siano più target in esecuzione. Perciò, prima di avviare il processo(i) target ISaGRAF, è necessario registrarlo(i). Per questo scopo esiste la subroutine *isa_register_slave()*.

```
uchar isa_register_slave
  (
  uchar slave    /* numero slave */
  )
```

Descrizione:
Aggiunge la registrazione di un nuovo slave al sistema di gestione di target multipli

valore di ritorno:
0 in caso di successo, BAD_RET se si verifica un errore

Unità di memorizzazione dei file di salvataggio dell'applicazione

Alla variabile globale TSK_FUNIT (char *) può essere assegnata una stringa relativa al percorso dell'unità per i file di salvataggio dell'applicazione. Per il salvataggio dei file dell'applicazione, il target ISaGRAF fa semplicemente uso delle routine standard di gestione file: fopen, fread, fwrite, fclose.
Il valore predefinito è costituito da una stringa nulla ("") che indica che non ci sono unità di memorizzazione.

Esempio:
TSK_FUNIT = "*host name:*/C:/ISaGRAF/target/apl/"

Specifica la directory ISaGRAF\target\apl\, sulla root di C, sul PC *host_name* come unità di salvataggio per i file dell'applicazione. Attenzione a non dimenticare l'ultimo slash, senza il quale il salvataggio verrà effettuato sulla directory ISaGRAF\target\ con tutti i nomi dei file con il prefisso apl.

Se necessario, a questa variabile, possono essere assegnati, per ciascun target, differenti percorsi di unità prima dell'avvio di ciascun kernel.
Informazioni più dettagliate riguardo al salvataggio dei file dell'applicazione si possono trovare nel capitolo relativo al salvataggio dell'applicazione nella sezione dedicata alle caratteristiche specifiche.

Controllo a fine ciclo

Alla variabile TSK_NBTCKSCHED (uint 32) si può assegnare un valore che specifica un ritardo in tick, usato dal target ISaGRAF alla fine del ciclo.
Il valore predefinito è 0 (schedulazione dei processi alla stessa priorità)
Se necessario, a questa variabile possono essere assegnati valori diversi per ciascun target, prima dell'avvio.
Informazioni più dettagliate sono riportate nella sezione dedicata alle caratteristiche specifiche del capitolo Durata del ciclo, funzionamento e priorità dei processi.

Avvio del target ISaGRAF

Una volta impostata la configurazione dell'ambiente, l'ultimo passo consiste nel lanciare il target (o i target) ISaGRAF: isa_main.

```
uchar isa_main
  (
  uchar slave,      /* Numero slave */
  char *com         /* Nome del dispositivo seriale */
  )
```

Descrizione:
Avvia un processo target ISaGRAF.

valore di ritorno:
ritorna un valore diverso da zero se si verifica un errore.

Il numero slave è lo stesso descritto nel paragrafo relativo alla registrazione degli slave.
Possono essere avviati più target con numeri slave diversi e diverse porte di comunicazione.
Usando il debugger dell'ambiente di lavoro, assicurarsi che il parametro slave dell'ambiente di lavoro (si veda il Manuale utente: Gestione dei programmi) corrisponda a quello di un target esistente.

Esempio

Questo esempio mostra come avviare un target ISaGRAF come processo singolo con il numero slave 1 ed il dispositivo di collegamento seriale /tyCo/1.
La directory host corrente è quella in cui è installato il target.

carica il modulo isassr.o
ld < RELS/isassr.o

carica il modulo isa.o

ld < CMDS/isa.o

configurazione delle comunicazioni seriali
ISAMOD ("/tyCo/1", 19200)

frequenza di clock del sistema
CLKRATE = sysClkRateGet ()

registrazione slave
isa_register_slave (1)

Unità di memorizzazione file (potrebbe essere omessa in quanto impostazione predefinita)
TSK_FUNIT = ""

Controllo a fine ciclo (potrebbe essere omessa in quanto impostazione predefinita)
TSK_NBTCKSCHED = 0

Avvio del target ISaGRAF
sp (isa_main, 1, "/tyCo/1")

C.5.4 Eseguire processi multipli ISaGRAF: isakerse.o and isakeret.o

Per migliorare il tempo di risposta del kernel del target ISaGRAF e delle comunicazioni, il target viene suddiviso in due processi distinguendo la parte relativa alle comunicazioni (il processo di comunicazione) dall'esecuzione dell'applicazione (processo relativo al kernel).
Questa architettura risulta molto più flessibile. Consente di avviare più processi di comunicazione collegati allo stesso processo kernel, oppure fino a 4 kernel collegati allo stesso processo di comunicazione. In questo modo vengono semplificate alcune integrazioni come il collegamento di un processo di visualizzazione ed il debugger dell'ambiente di lavoro alla stessa applicazione o un collegamento singolo tramite la stessa porta fisica a diverse applicazioni (fino a 4).

I processi relativi al kernel ed alle comunicazioni possono essere fatti partire indipendentemente. L'unico requisito è costituito dal fatto che il processo/i kernel deve essere avviato per primo per permettergli di inizializzare l'ambiente di sistema e per permettere al processo/i di comunicazione di potersi collegare ad esso.

Il target ISaGRAF multiprocesso non è in grado di prevenire l'esecuzione di processi in parallelo o di routine gestite da interrupt.

Vengono proposti due moduli, a seconda delle capacità dell'hardaware di comunicazione:
- Kernel e collegamento seriale: isakerse.o
Questo modulo permette di avviare il processo(i) kernel ed il processo(i) di comunicazione seriale.
- Kernel e collegamento seriale e/o Ethernet: isakeret.o
Questo modulo permette di avviare il processo(i) kernel ed il processo(i) di comunicazione seriale e/o Ethernet

Il modo di avvio di ISaGRAF è il medesimo per i moduli isakerse.o e isakeret.o, ad eccezione del fatto che per isakeret.o è possibile specificare il nome di un dispositivo seriale o un numero di porta per il collegamento Ethernet come parametro relativo al nome del dispositivo

di comunicazione all'avvio del processo(i) di comunicazione ISaGRAF: tst_main_ex (vedere sotto).

Per IsaGRAF il target VxWorks rappresenta il server, mentre il debugger è il client collegato al numero di porta specificato.

Registrazione Kernel

Un kernel ISaGRAF è caratterizzato dal numero slave . Sono possibili valori compresi tra 1 e 255, ad eccezione del numero 13 ($0D). Il numero slave viene usato dal protocollo di comunicazione e dal processo(i) di comunicazione collegato al kernel. È necessario per distinguere i dispositivi slave nel caso ci siano più target in esecuzione. Perciò, prima di avviare il processo(i) target ISaGRAF, è necessario registrarlo(i). Per questo scopo esiste la subroutine *isa_register_slave().*

```
uchar isa_register_slave
  (
  uchar slave    /* numero slave */
  )
```

Descrizione:
Aggiunge la registrazione di un nuovo kernel slave al sistema di gestione di target multipli

valore di ritorno:
0 in caso di successo, BAD_RET se si verifica un errore

Registrazione processi di comunicazione

Un processo di comunicazione ISaGRAF è caratterizzato da un numero logico. Serve a gestire processi di comunicazione simultanei. Può essere compreso tra 1 e 255 e deve essere diverso per ogni processo di comunicazione. Perciò, prima di avviare il processo(i) di comunicazione ISaGRAF, è necessario registrarlo(i). Per questo scopo esiste la subroutine *isa_register_com ()*.

```
uchar isa_register_com
  (
  uchar com_id       /* identificatore del processo di comunicazione */
  )
```

Descrizione:
Aggiunge la registrazione di un nuovo processo di comunicazione al sistema di gestione di target multipli

valore di ritorno:
0 in caso di successo, BAD_RET se si verifica un errore

Unità di memorizzazione dei file di salvataggio dell'applicazione

Alla variabile globale TSK_FUNIT (char *) può essere assegnata una stringa relativa al percorso dell'unità per i file di salvataggio dell'applicazione. Per il salvataggio dei file dell'applicazione, il target ISaGRAF fa semplicemente uso delle routine standard di gestione file: fopen, fread, fwrite, fclose.

Il valore predefinito è costituito da una stringa nulla (“”) che indica che non ci sono unità di memorizzazione.

Esempio:
TSK_FUNIT = "*host name:*/C:/ISaGRAF/target/apl/"
Specifica la directory ISaGRAF\target\apl\, sulla root di C, sul PC *host_name* come unità di salvataggio per i file dell’applicazione. Attenzione a non dimenticare l’ultimo slash, senza il quale il salvataggio verrà effettuato sulla directory ISaGRAF\target\ con tutti i nomi dei file con il prefisso apl.

Se necessario, a questa variabile, possono essere assegnati, per ciascun target, differenti percorsi di unità prima dell’avvio di ciascun kernel.
Informazioni più dettagliate riguardo al salvataggio dei file dell’applicazione si possono trovare nel capitolo relativo al salvataggio dell’applicazione nella sezione dedicata alle caratteristiche specifiche.

Controllo a fine ciclo

Alla variabile TSK_NBTCKSCHED (uint 32) si può assegnare un valore che specifica un ritardo in tick, usato dal target ISaGRAF alla fine del ciclo.
Il valore predefinito è 0 (schedulazione dei processi alla medesima priorità).
Se necessario, a questa variabile possono essere assegnati valori diversi per ciascun kernel, prima dell’avvio.
Informazioni più dettagliate sono riportate nelle sezioni dedicate alle caratteristiche specifiche; capitolo Durata del ciclo, funzionamento e priorità dei processi.

ISaGRAF kernel spawning

Una volta completata la configurazione dell’ambiente, uno degli ultimi passi consiste nel lanciare i kernel ISaGRAF: isa_main.

```
uchar isa_main
  (
  uchar slave,      /* Numero slave */
  char *com         /* NON USATO la stringa nulla va bene */
  )
```

Descrizione:
Avvia un processo kernel ISaGRAF.

valore di ritorno:
ritorna un valore diverso da zero se si verifica un errore.

Il numero slave è lo stesso descritto nel paragrafo relativo alla registrazione degli slave.
Possono essere avviati più kernel con numeri slave diversi.

ISaGRAF communication task spawning

Una volta impostata la configurazione dell’ambiente, uno degli ultimi passi consiste nel lanciare il processo(i) di comunicazione ISaGRAF : tst_main_ex.

```
uchar tst_main_ex
  (
  char *com,        /* Nome del dispositivo di comunicazione */
```

```
  uchar *slave,      /* Locazione di un campo di 4 Byte che indica gli slave kernel
  cui collegarsi */
  uchar com_id       /* identificativo del processo di comunicazione */
  )
```

Descrizione:
Avvia un processo di comunicazione ISaGRAF

valore di ritorno:
ritorna un valore diverso da zero se si verifica un errore.

Il campo di 4 byte indica i kernel slave cui è collegato il processo di comunicazione. Se i kernel slave da indicare sono meno di quattro, il campo deve essere completato da zeri. Una volta avviato il processo, questo campo non è più di alcuna utilità.
Il nome del dispositivo di comunicazione corrisponde al nome del dispositivo seriale che deve essere usato per le comunicazioni.
Possono essere avviati più processi di comunicazione se i rispettivi identificativi di processo sono differenti.
Usando il debugger dell'ambiente di lavoro, assicurarsi che i parametri di comunicazione dell'ambiente di lavoro (si veda il Manuale utente: Gestione dei programmi) corrispondano ad un target esistente (processi kernel e di comunicazione).

Esempio:

Questo esempio mostra come avviare:
Un processo kernel ISaGRAF con numero slave 1.
Un processo di comunicazione ISaGRAF identificato dal numero 1, connesso al kernel slave 1 e con il dispositivo /tyCo/1 per il collegamento seriale.
Un processo di comunicazione ISaGRAF identificato dal numero 2, connesso al kernel slave 1 e con il numero di porta 1100 per il collegamento Ethernet.
La directory host corrente è quella in cui è installato il target.

carica il modulo isassr.o
ld < RELS/isassr.o

carica il modulo isakeret.o (si può caricare isakerse.o nel caso non sia necessario il collegamento Ethernet)
ld < CMDS/isakeret.o

configurazione delle comunicazioni seriali
ISAMOD ("/tyCo/1", 19200)

Frequenza di clock del sistema
CLKRATE = sysClkRateGet ()

registrazione slave
isa_register_slave (1)

registrazione comunicazioni
isa_register_com (1)
isa_register_com (2)

Unità di memorizzazione file (potrebbe essere omessa in quanto impostazione predefinita)

TSK_FUNIT = ""

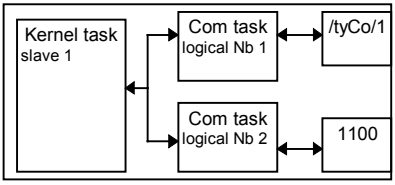
Controllo a fine ciclo (potrebbe essere omessa in quanto impostazione predefinita)
TSK_NBTCKSCHED = 0

Avvio del kernel ISaGRAF
sp (isa_main, 1, "")

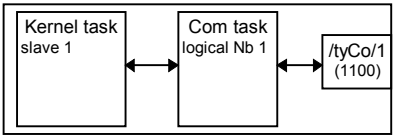
processo di comunicazione, collegamento agli slave
SlavesLink = 0x01000000

avvio dei processi di comunicazione ISaGRAF
sp (tst_main_ex, "/tyCo/1", &SlavesLink, 1)
sp (tst_main_ex, "1100", &SlavesLink, 2)

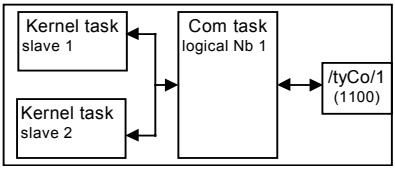
Questo avvio corrisponde alla seguente figura.



Sono anche possibili le seguenti configurazioni base.



La configurazione base è costituita da un processo kernel associato ad un processo di comunicazione con collegamento seriale (Ethernet).

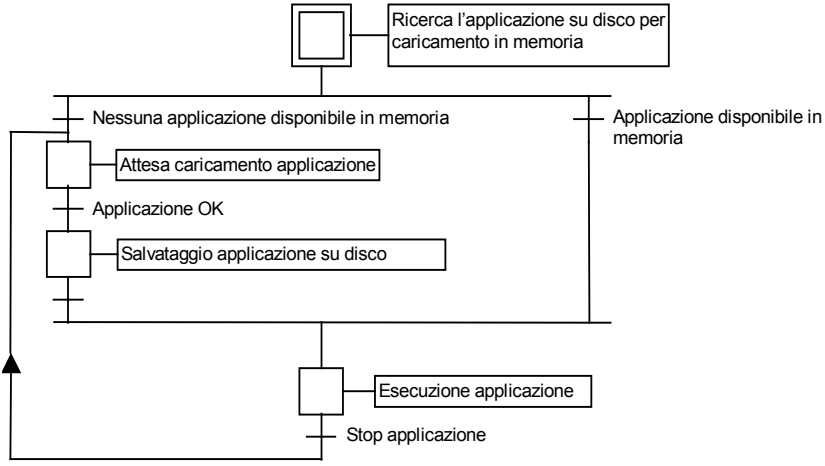


Un'altra configurazione è costituita da 2 kernel associati ad un processo di comunicazione con collegamento seriale (Ethernet). In questo caso, SlavesLink = 0x01020000.

C.5.5 Caratteristiche specifiche

Avvio ISaGRAF

All'avvio del target, viene eseguito il seguente algoritmo.



- **Definizioni**

Il codice dell'applicazione è costituito dal data base binario generato e trasferito dall'ambiente di lavoro, successivamente eseguito dal target. Può essere completato dalla tavola dei simboli.
La tavola dei simboli dell'applicazione è un data base ASCII generato e trasferito dall'ambiente di lavoro. Questa tavola rappresenta la corrispondenza tra gli oggetti simbolici e gli oggetti interni del target. Non è normalmente richiesta dal target, ad eccezione di quando sia necessario gestire simboli specifici definiti dall'utente. Ulteriori informazioni sulla tavola dei simboli si possono trovare sul Manuale utente alla sezione : Tecniche avanzate di programmazione.
Il percorso indicante l'unità disco del file viene specificato all'avvio del target ISaGRAF impostando la variabile globale TSK_FUNIT (valore predefinito = "" per indicare nessuna unità disco)

- **Applicazioni multiple ISaGRAF**

Su una stessa CPU possono essere eseguite diverse applicazioni (processi kernel e di comunicazione) simultaneamente, con numeri slave diversi e numeri logici del processo di comunicazione diversi. Tuttavia, nel caso vengano eseguite applicazioni diverse, è necessario tenere conto dell'accesso condiviso ad alcuni oggetti condivisi delle applicazioni, come le schede I/O. Per esempio applicazioni diverse (kernel) possono usare schede fisiche distinte a meno che non vengano implementati server di I/O o semafori tramite il driver I/O.

- **Salvataggio dell'applicazione**

Quando una nuova applicazione viene scaricata dal debugger dell'ambiente di lavoro al target, il codice dell'applicazione viene salvato (il target fa uso delle routine standard di gestione dei file come fopen,…) nella directory corrente del target, con il nome:

*path***ISAx1** file di salvataggio del codice dell'applicazione ISaGRAF (dove x rappresenta il numero slave)

La tavola dei simboli, anche se scaricata precedentemente, viene salvata nella directory corrente del target con il nome:

*path***ISAx6** file di salvataggio dei simboli dell'applicazione ISaGRAF (dove x rappresenta il numero slave)

Il percorso *path* viene specificato all'avvio del target ISaGRAF impostando la variabile globale TSK_FUNIT. Una stringa vuota ("") indica nessuna unità disco (valore predefinito).

All'avvio del target ISaGRAF, i file relativi a codice e simboli dell'applicazione vengono cercati nella directory corrente e caricati in memoria.

Se la tavola dei simboli non è disponibile in memoria, il target avvia l'esecuzione del codice dell'applicazione senza aver caricato i simboli.
Se il codice dell'applicazione non è disponibile in memoria, il target rimane in attesa del caricamento dell'applicazione.
Per avviare il target all'accensione con un'applicazione specifica, senza usare il collegamento con il debugger:

- Un primo modo consiste nel copiare direttamente, con un qualsiasi programma di trasferimento, questi file sull'unità di salvataggio dell'applicazione dal PC host in cui è installato l'ambiente di lavoro. Queste operazioni vengono semplificate usando il menu "Strumenti" dell'ambiente di lavoro (si veda il Manuale utente: Gestione dei programmi).

- Una seconda maniera è quella di memorizzare il codice dell'applicazione (e, se necessario, la tavola dei simboli dell'applicazione), contenuto nei file del PC host in cui è installato l'ambiente di lavoro, su una memoria non volatile (come una PROM o EPROM), con gli appositi programmi.

  Poi, all'accensione del sistema, se richiesto (ad esempio per un accesso più veloce o per la gestione di breakpoint), è possibile caricare il codice dell'applicazione (e, se necessario la tavola dei simboli dell'applicazione) dalla PROM alla RAM, con gli appositi programmi.

  Poi, all'avvio di ISaGRAF (poco prima del lancio dei processi) è necessario specificare l'indirizzo(i) di memoria del codice dell'applicazione (e, se necessario, la tavola dei simboli dell'applicazione). Per questo, è necessario inizializzare la variabile globale SSR nel seguente modo:

  SSR[x][1].space = *indirizzo del codice dell'applicazione*

  E, se necessario:

  SSR[x][6].space = *indirizzo della tavola dei simboli dell'applicazione*

  Per questo, è possibile scrivere una breve procedura. La variabile globale SSR è dichiarata come struttura str_ssr, definita nel file tasy0ssr.h.

  AVVERTENZA:

La gestione dei breakpoint del debugger ISaGRAF non funziona correttamente se il modulo del codice dell'applicazione non è accessibile in scrittura. Questo non è un problema, poiché l'applicazione è stata precedentemente provata esaustivamente.

Sul PC host, se l'ambiente di lavoro ISaGRAF è installato nella directory standard \ISAWIN:
il file del codice dell'applicazione del progetto MYPROJ è:
\ISAWIN\APL\MYPROJ\appli.x6m (corrispondente a isax1 sul target).
il file dei simboli dell'applicazione del progetto MYPROJ è:
\ISAWIN\APL\MYPROJ\appli.tst (corrispondente a isax6 sul target).

Gestione degli errori e messaggi di output

Il software target ISaGRAF incorpora una gestione di riconoscimento dell'errore. In appendice è riportata la lista degli errori con la rispettiva descrizione.

La gestione dell'errore avviene nel seguente modo:
- Un errore è composto da un numero di errore e da un numero di argomento che vengono inviati alla routine di gestione degli errori ISaGRAF.
- Se è attivo il flag di gestione errori nelle opzioni per l'esecuzione dell'applicazione dell'ambiente di lavoro, l'errore viene processato. In caso contrario, l'informazione viene tralasciata, senza l'intervento della routine di gestione degli errori.

Quando l'errore viene processato:
- Il numero di errore (decimale) e l'argomento (esadecimale) vengono visualizzati sul dispositivo di output predefinito stdout.
- Il numero di errore e di argomento, vengono inseriti in un buffer circolare FIFO, così da poter essere recuperati in un secondo tempo. La dimensione del buffer di gestione degli errori è definibile nelle opzioni per l'esecuzione dell'applicazione dell'ambiente di lavoro. A buffer pieno, ogni nuovo errore sovrascrive il primo contenuto nel buffer stesso in ordine cronologico.
- I codice di errore possono essere recuperati, sia dal debugger che dall'applicazione durante l'esecuzione, mediante le chiamate SYSTEM (si veda il Manuale utente).

Quando il debugger rileva un errore, viene visualizzato, nella finestra di comunicazione, un messaggio che descrive l'errore. A seconda dello stato dell'applicazione (in esecuzione o meno), il debugger può mostrare il nome dell'oggetto (variabile o programma) che ha causato l'errore, o il numero di argomento (decimale) racchiuso tra parentesi quadre [x], che ha diverso significato a seconda dell'errore.

Sul target, quando si verifica un errore, i codici di errore vengono mostrati sull'output predefinito stdout. In questo modo, la visualizzazione può essere rediretta con le routines VxWorks come *ioGlobalStdSet()*
o *ioTaskStdSet()*
Nell'ultimo caso, si noti che entrambi i processi, kernel e di comunicazione, possono generare errori.

Durata del ciclo, funzionamento e priorità dei processi

- Alla fine di un ciclo ISaGRAF, poco prima dell'inizio del successivo, viene eseguito il seguente algoritmo:
  Se è stato specificato un tempo di ciclo (dall'ambiente di lavoro: si veda il Manuale utente: Gestione dei Programmi), allora la CPU viene rilasciata per il tempo rimanente

(tempo di ciclo specificato – tempo corrente dell'applicazione). Se il tempo rimanente risulta negativo, viene generato un errore di overflow e viene rilasciata la CPU per il tempo di TSK_NBTCKSCHED (variabile impostata all'avvio di ISaGRAF) tick per forzare la schedulazione.

Se non è stato specificato il tempo di ciclo, o se il tempo rimanente è minore o uguale ad 1 tick o uguale a zero, allora la CPU viene rilasciata per il tempo di TSK_NBTCKSCHED tick per forzare la schedulazione.

L'accuratezza nella temporizzazione del target corrisponde ad un tick di sistema VxWorks.

I cicli vengono comunemente controllati da un tempo di ciclo, così da rilasciare la CPU ad altri processi in esecuzione sul sistema VxWorks.

- Il processo di comunicazione è in stato inattivo quando non arrivano dati dal collegamento di comunicazione. Quando necessario, questo processo ottiene informazioni dall'applicazione in esecuzione mediante un protocollo domanda/risposta con il processo kernel. Il processo di comunicazione invia una domanda al kernel. Al termine del ciclo (per mantenere sincrona l'mmagine di memoria dell'applicazione), il kernel risponde al processo di comunicazione.

I processi ISaGRAF non modificano la priorità che è stata loro assegnata. È possibile regolare liberamente queste priorità, rispettando il funzionamento dei processi ISaGRAF precedentemente descritto e le esigenze dell'applicazione.

C.6 Target ISaGRAF per NT

C.6.1 Avviare ISaGRAF

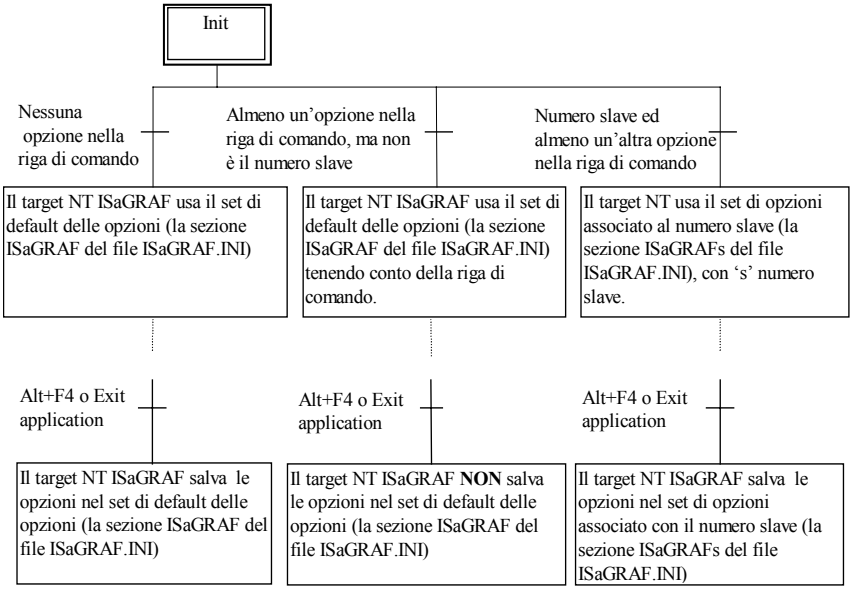
Nell'implementazione NT, il target viene eseguito come programma singolo: WISAKER.EXE, che può essere lanciato più volte. Questo consente l'esecuzione di molteplici target NT ISaGRAF, poiché ciascuna istanza ha un numero slave distinto.

Il programma target non è in grado di prevenire l'esecuzione di routine gestite da interrupt.

Il software WISAKER è progettato per essere eseguito su Windows NT 3.51 o superiore.

C.6.2 informazioni generali sulle opzioni

Le opzioni vengono salvate e recuperate secondo il seguente diagramma:



Nota: il file ISAGRAF.INI viene salvato nella directory di lavoro corrente.

- ***Numero slave: opzione -s***

Questa opzione specifica il numero slave per il target. Sono possibili valori compresi tra 1 e 255, ad eccezione del numero 13 ($0D). Il numero slave viene usato dal protocollo di comunicazione. Serve principalmente per distinguere i dispositivi slave nel caso ci siano più target connessi allo stesso ambiente di lavoro host o quando sono in esecuzione più target sullo stesso PC. Quando si usa il debugger dell'ambiente di lavoro, assicurarsi che le impostazioni slave dell'ambiente di lavoro (vedere il Manuale utente: Gestione dei programmi) corrispondano a quelle del target.

Valore predefinito: Il valore predefinito del numero slave è 1 o quello del file ISaGRAF.INI

Esempio:
WISAKER.EXE -s=2

Interfaccia utente: Questa finestra appare con il comando "Options/Slave" della finestra principale del target NT ISaGRAF.

Usando il mouse sulle frecce (Alto e Basso), è possibile modificare il valore dell'opzione. Per attivarla è necessario riavviare il target NT ISaGRAF.

Collegamento e configurazione comunicazioni: opzione –t

Il target ISaGRAF può usare, per comunicare con il debugger, un collegamento seriale o Ethernet.
Il nome della porta viene specificato con l'opzione –t. Poiché l'interfaccia delle comunicazioni è progettata per essere compatibile con qualsiasi macchina, possono essere usate le porte COM1, COM2, COM3, COM4 per le comunicazioni seriali e le porte con numeri a partire da 1100 per le comunicazioni Ethernet.

Valore predefinito: La porta di comunicazione predefinita è 1100 per Ethernet e COM1 per le comunicazioni seriali o quella definita nel file ISaGRAF.INI.

NOTA BENE: Il collegamento predefinito per le comunicazioni è Ethernet.

Esempi:
WISAKER -t=COM2
WISAKER -t=1101

Configurazione seriale:
È possibile definire alcune opzioni solo se è stata specificata l'opzione -t=COMx.

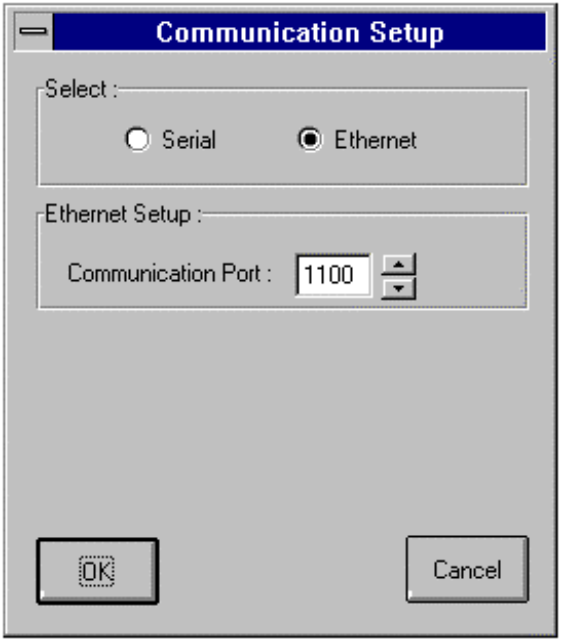
Queste sono le opzioni di configurazione per il collegamento seriale:

Opzione	Values	Significato
baud	600 1200 2400 4800 9600 **19200**	Baud rate
parity	**n** e o	No parity Even Odd
data	7 or **8**	Numero di bits
stop	**1** or 2	Lunghezza del bit di stop
flow	h **n**	Controllo hardware Nessun controllo

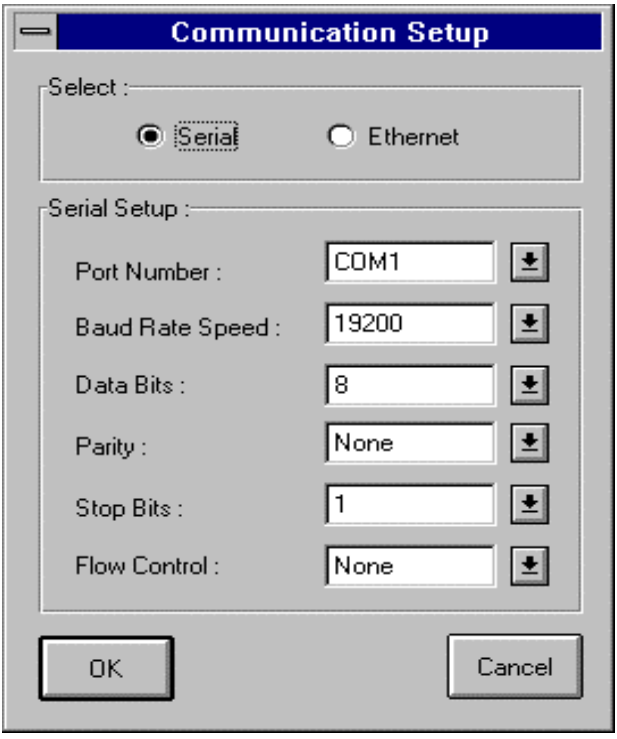
I valori di default sono 19200, no parity, 8 bit di dati, 1 di stop, nessun controllo di flusso

Esempio:
WISAKER -t=COM1 baud=1200 data=8 parity=n stop=2

Interfaccia utente: Questa finestra appare con il comando "Options/Communication" della finestra principale del target NT ISaGRAF.



È possibile scegliere tra comunicazione seriale e comunicazione Ethernet. Per la comunicazione Ethernet è possibile modificare il numero della porta. Il numero della porta deve essere lo stesso di quello specificato nel collegamento PC-PLC dell'ambiente di lavoro.



Selezionando la comunicazione seriale, appare la relativa configurazione. Tale configurazione deve essere la stessa di quella specificata nel collegamento PC-PLC dell'ambiente di lavoro.

Simulazione grafica di schede virtuali: opzione -x

Questa opzione abilita la simulazione delle schede dichiarate virtuali nell'editor delle connessioni I/O (Vedere il Manuale utente).
I valori possibili sono 0 o 1, 0 disabilita la simulazione ed 1 attiva la simulazione.

Valore predefinito: Il valore predefinito è 0 o quello nel file ISaGRAF.INI.

Esempio:
WISAKER -x=1 abilita la simulazione delle schede virtuali,

Interfaccia utente: La voce di menu appare segnata o meno secondo lo stato dell'opzione. Le schede simulate appaiono in un pannello grafico.

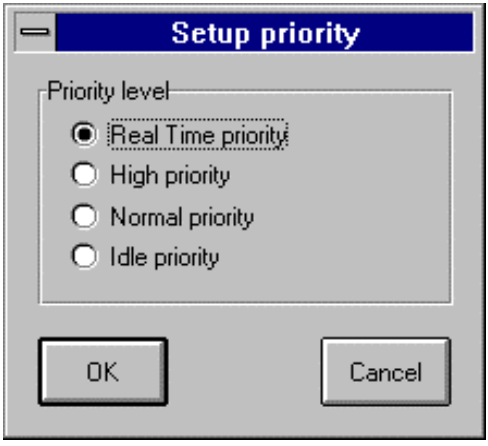
priorità del target NT ISaGRAF: opzione -p

Poiché il target viene eseguito sotto NT, è molto utile poter specificare un livello di priorità. Ad esempio, è possibile avere un'applicazione ISaGRAF impegnativa in esecuzione con un target a priorità alta ed uno o più target in esecuzione in parallelo con priorità bassa.

I valori possibili sono 0, 1, 2 o 3. 0 rappresenta la priorità più alta e 3 la più bassa.

Esempi:
WISAKER -p=0
WISAKER -p=1

Interfaccia utente: Questa finestra appare con il comando "Options/Priority" della finestra principale del target NT ISaGRAF.



La priorità più alta è quella in tempo reale e la più bassa quella in attesa.
0: Priorità in tempo reale
1: Priorità alta
2: Priorità normale
3: Priorità in attesa

Esempi:

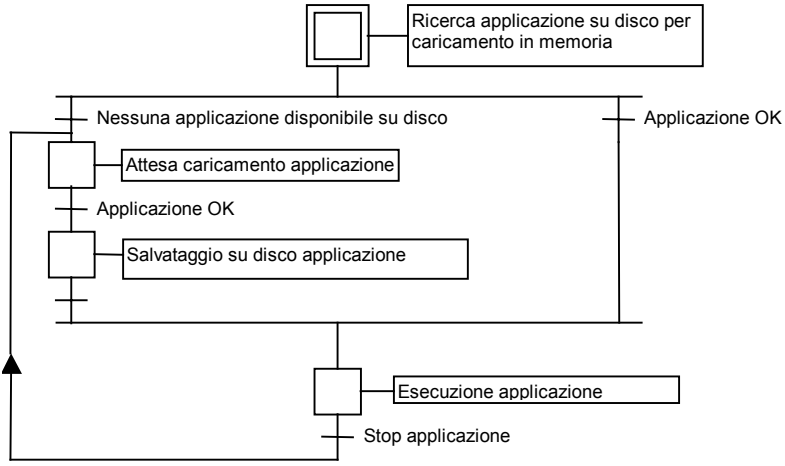
wisaker -t=COM1 Avvia il target ISaGRAF con il numero slave predefinito (1) e con la porta di comunicazione COM1.

wisaker -s=3 -t=COM1 Avvia il target ISaGRAF con il numero slave 3 e con la porta di comunicazione COM1.

C.6.3 Caratteristiche specifiche

Avvio ISaGRAF

All'avvio del target, viene eseguito il seguente algoritmo.



- **Definizioni**

Il codice dell'applicazione consiste nel data base binario generato e scaricato dall'ambiente di lavoro, successivamente eseguito dal target. Può essere completato dalla tavola dei simboli.
La tavola dei simboli dell'applicazione è un data base ASCII generato e scaricato dall'ambiente di lavoro. Questa tavola rappresenta la corrispondenza tra gli oggetti simbolici e gli oggetti interni del target. Non è normalmente richiesta dal target, ad eccezione di quando sia necessario gestire simboli specifici definiti dall'utente come ad esempio le funzionalità DDE o la simulazione I/O con i nomi dei simboli. Ulteriori informazioni sulla tavola dei simboli si possono trovare sul Manuale utente alla sezione : Tecniche di programmazione avanzata.

- **Applicazioni multiple ISaGRAF**

Su una stessa CPU possono essere eseguite diverse applicazioni simultaneamente, con numeri slave diversi e numeri logici del processo di comunicazione diversi. Tuttavia, nel caso vengano eseguite applicazioni diverse, è necessario tenere conto dell'accesso condiviso ad alcuni oggetti condivisi delle applicazioni, come le schede di I/O. Per esempio applicazioni diverse possono usare schede fisiche distinte a meno che non vengano implementati server di I/O o semafori tramite il driver I/O.

- **Salvataggio dell'applicazione**

Quando una nuova applicazione viene scaricata dal debugger dell'ambiente di lavoro al target, il codice dell'applicazione viene nella directory corrente del target, con il nome:

ISAx1 **f**ile di salvataggio del codice dell'applicazione ISaGRAF (dove x rappresenta il numero slave)

La tavola dei simboli, anche se scaricata precedentemente, viene salvata nella directory corrente del target con il nome:

ISAx6 file di salvataggio dei simboli dell'applicazione ISaGRAF (dove x rappresenta il numero slave)

All'avvio del target ISaGRAF, i file relativi al codice dell'applicazione ed ai simboli dell'applicazione vengono cercati nella directory corrente e caricati in memoria.
Se non è disponibile il file dei simboli, il target avvia l'esecuzione del codice dell'applicazione senza aver caricato i simboli.
Se il codice dell'applicazione non è disponibile, il target rimane in attesa del caricamento dell'applicazione.

Per eseguire il target all'accensione con un'applicazione specifica, senza usare il collegamento con il debugger, questi file possono essere copiati direttamente sul disco della directory corrente del target, dallo stesso disco se l'ambiente di lavoro si trova sullo stesso PC o usando un dischetto floppy in caso contrario.

Se l'ambiente di lavoro ISaGRAF è installato nella directory standard \ISAWIN:
il file del codice dell'applicazione relativa al progetto MYPROJ è:
\ISAWIN\APL\MYPROJ\appli.x8m
il file dei simboli dell'applicazione relativa al progetto MYPROJ è:
\ISAWIN\APL\MYPROJ\appli.tst

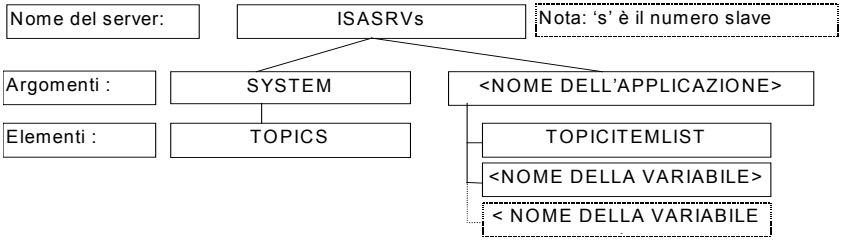
Esempio:
Dalla directory in cui è installato isa.exe, con il seguente comando:
copy \ISAWIN\APL\MYPROJ\appli.x8m isa11
Il programma WISAKER.EXE troverà ed eseguirà l'applicazione 'myproj'.

Tutti questi comandi possono essere raggruppati in un file batch ed avviati dal menu Strumenti dell'ambiente di lavoro (vedere il Manuale utente: Gestione dei programmi).

Specifiche DDE

Il target NT ISaGRAF è un server DDE (Dynamic Data Exchange). Qualsiasi software client può essere connesso con il target per scambiare variabili. Ad esempio MSEXCEL può animare dei grafici usando i valori provenienti dal target ISaGRAF via DDE.
La funzionalità DDE richiede la presenza sul target, della tavola dei simboli dell'applicazione:

Gli argomenti DDE sono definiti come segue:



« ISASRVs » è il nome del server DDE, 's' è il numero slave.
« SYSTEM » è l'argomento standard per accedere agli elementi « TOPICS »,
« TOPICS » lista gli argomenti correntemente definiti: il sistema ed il nome dell'applicazione che viene eseguita dal target NT ISaGRAF.
« APPLICATION NAME » è il nome dell'applicazione.

« TOPICITEMLIST » è la lista degli elementi disponibili relativi all'argomento corrente, si ottiene la lista delle variabili cui si può accedere vie DDE.
« VARIABLE NAME » nome di una variabile.

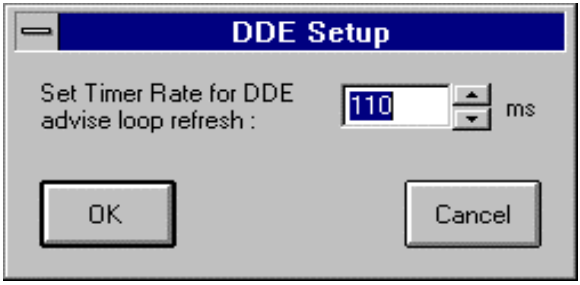
Frequenza del ciclo di notifica DDE per il target NT ISaGRAF: opzione -d
Il client DDE, generalmente, interroga le variabili ogni qual volta ne ha bisogno. Questo può occupare molto tempo nel caso ci siano molte variabili. Esiste un diverso modo, chiamato modalità notifica (ciclo di notifica), in cui il server stesso provvede ad inviare solamente le variabili che hanno subito modifiche. Così le comunicazioni risultano efficienti e ridotte al minimo. In questo modo il server periodicamente scorre le variabili da notificare per conoscere quali siano da inviare. Questo periodo viene chiamato frequenza del ciclo di notifica DDE.

Questa opzione permette di specificare la frequenza (in ms) del ciclo di notifica DDE.

Valore predefinito: Il valore predefinito è di 1000 ms o quello contenuto nel file ISaGRAF.INI

Esempio:
WISAKER -d=100

Interfaccia utente: Questa finestra appare con il comando "Options/DDE" della finestra principale del target NT ISaGRAF.



Gestione degli errori e messaggi di output

Il software target ISaGRAF incorpora una gestione di riconoscimento dell'errore. In appendice è riportata la lista degli errori con la rispettiva descrizione.

La gestione dell'errore avviene nel seguente modo:
- Un errore è composto da un numero di errore e di argomento che vengono inviati alla routine di gestione degli errori ISaGRAF.
- Se è attivo il flag di gestione errori nelle opzioni per l'esecuzione dell'applicazione dell'ambiente di lavoro, l'errore viene processato. In caso contrario, l'informazione viene tralasciata senza l'intervento della routine di gestione degli errori.

Quando l'errore viene processato:
- Il numero di errore (decimale) e l'argomento (esadecimale) vengono visualizzati in output (la finestra del programma WISAKER.EXE).

- Il numero di errore e di argomento, vengono inseriti in un buffer circolare FIFO, così da poter essere recuperati in un secondo tempo. La dimensione del buffer di gestione degli errori è definibile nelle opzioni per l'esecuzione dell'applicazione dell'ambiente di lavoro. A buffer pieno, ogni nuovo errore sovrascrive il primo contenuto nel buffer stesso in ordine cronologico.
- I codice di errore possono essere recuperati sia dal debugger che dall'applicazione durante l'esecuzione mediante le chiamate SYSTEM (si veda il Manuale utente).

Quando il debugger rileva un errore, viene visualizzato, nella finestra di comunicazione, un messaggio che descrive l'errore. A seconda dello stato dell'applicazione (in esecuzione o meno), il debugger può mostrare il nome dell'oggetto (variabile o programma) che ha causato l'errore, o il numero di argomento (decimale) racchiuso tra parentesi quadre [x], che ha diverso significato a seconda dell'errore.

All'avvio del target viene mostrato sull'output un messaggio iniziale composto dal numero slave, dalla configurazione di comunicazione e dal nome del server DDE.

Clock di sistema

Poiché un target ISaGRAF è progettato per essere eseguito su qualsiasi sistema, il tempo di riferimento usato sia per la sincronizzazione del ciclo, che per l'aggiornamento delle variabili timer, corrisponde al tick standard di circa 10 millisecondi.

Per questo non è possibile un'accuratezza migliore di 10ms per le variabili timer. Per questa ragione, impostare una durata di ciclo minore od uguale a 10ms e diversa da zero, causa un errore di overflow di durata del ciclo (errore 62). Si rimanda al seguente paragrafo *Durata del ciclo, funzionamento del target* per ulteriori informazioni.

Chiedere al fornitore un'implementazione speciale, nel caso l'applicazione necessiti di maggior accuratezza.

Durata del ciclo, funzionamento del target

Alla fine di un ciclo ISaGRAF, poco prima dell'inizio del successivo, viene eseguito il seguente algoritmo:

Se è stato specificato un tempo di ciclo (dall'ambiente di lavoro: si veda il Manuale utente: Gestione dei Programmi), allora la CPU viene rilasciata per il tempo rimanente (tempo di ciclo specificato – tempo corrente dell'applicazione). Se il tempo rimanente risulta negativo, viene generato un errore di overflow e viene rilasciata la CPU per il tempo di 1 tick per forzare la schedulazione.

Se non è stato specificato il tempo di ciclo, o se il tempo rimanente è minore o uguale ad un tick, o uguale a zero, allora la CPU viene rilasciata per il tempo di un tick per forzare la schedulazione.

L'accuratezza nella temporizzazione del target corrisponde ad un tick di sistema Windows NT.

I cicli vengono comunemente controllati da un tempo di ciclo, così da rilasciare la CPU ad altri processi in esecuzione sul sistema Windows NT.

Tasti di uscita

In fase di prova delle applicazioni non in condizioni “industriali” su un PC desktop, si può voler fermare l’esecuzione del programma ISaGRAF: questo è possibile tramite una combinazione di tasti, in modo da evitare che venga attivata in modo accidentale. La sequenza di tasti è:

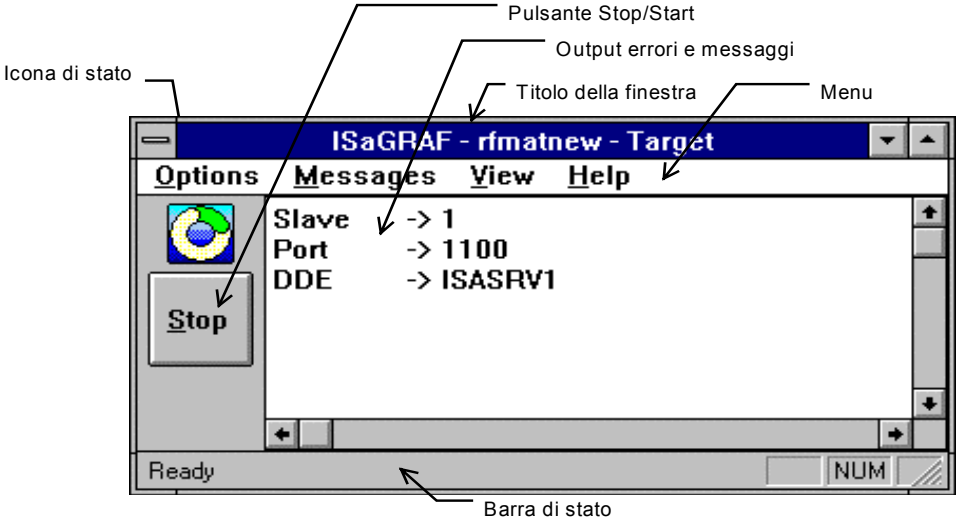
alt + F4

Un effetto collaterale pericoloso di queste uscite immediate, è costituito dal fatto che l’interfacciamento con le schede IO **non** viene chiuso. Il modo corretto per fermare il target ISaGRAF è il seguente:

- fermare l’applicazione dal debugger o premendo il pulsante Stop/Start (chiudendo così le schede IO)
- fermare il target ISaGRAF menu di sistema

C.6.4 Interfaccia utente

Questa è l’interfaccia utente del target NT ISaGRAF:



Gli elementi principali sono::

il titolo della finestra
la barra dei menu
l’icona che mostra lo stato in esecuzione
il pulsante Start/Stop
l’output per gli errori ed i messaggi
la barra di stato.

Il titolo della finestra contiene «ISaGRAF - nome_appli – target», dove nome_appli rappresenta il nome dell’applicazione in esecuzione. Contiene solo «ISaGRAF - - Target» nel caso no ci siano applicazioni in esecuzione.

Barra dei menu del target NT ISaGRAF:

La barra dei menu è composta da quattro menu:

Options
Messages
View
Help

- **menu "Options"**

(si veda anche la prima sezione su NT: Informazioni generali sulle opzioni)
Il menu "**Options**" permette di accedere alle opzioni di esecuzione. Vengono proposte le seguente opzioni:

Slave per modificare il numero slave . La modifica all'opzione avrà effetto solo al successivo riavvio del target.
Questa funzionalità non è disponibile se il target è stato avviato con almeno un'opzione nella riga di comando.

Communication per modificare la configurazione delle comunicazioni . La modifica all'opzione avrà effetto solo al successivo riavvio del target.
Questa funzionalità non è disponibile se il target è stato avviato con almeno un'opzione nella riga di comando diversa dall'opzione -s.

DDE per modificare la frequenza del ciclo di notifica DDE . La modifica all'opzione avrà effetto solo al successivo riavvio del target.
Questa funzionalità non è disponibile se il target è stato avviato con almeno un'opzione nella riga di comando diversa dall'opzione -s.

Simulate I/O segnato o meno secondo lo stato dell'opzione . La modifica all'opzione avrà effetto solo al successivo riavvio del target.

Priority per modificare la priorità. La modifica a questa opzione ha effetto immediato.

Default Options recupera le opzioni di esecuzione predefinite solamente per i seguenti elementi:
- Comunicazioni
- DDE
- coordinate della finestra sullo schermo

Le modifiche alle opzioni avranno effetto solo al successivo riavvio del target.
Questa funzionalità non è disponibile se il target è stato avviato con almeno un'opzione diversa, però, dall'opzione -s.

- **Menu "Messages"**

Il menu "**Messages**" è relativo alla gestione dell'output. Comprende i due seguenti comandi:

Acknowledge : interrompe il lampeggio rosso in caso di errori o messaggi.

Clear : cancella completamente l'output.

L'icona del target NT ISaGRAF:

L'icona esplicita lo stato del target:

- se l'applicazione è in esecuzione, allora l'icona ruota
- se nessuna applicazione è presente (o l'applicazione è ferma), l'icona è ferma
- se sono presenti errori o messaggi nella finestra di output. Il centro dell'icona diventa rosso lampeggiante. Per fermare il lampeggiare, è possibile selezionare la voce «Acknowledge» del menu «Messages» o la voce «Clear» dello stesso menu (si badi bene che con questa voce si cancella completamente la finestra di output). Ulteriori informazioni sugli errori sono disponibili nel paragrafo dedicato alla gestione degli errori ed ai messaggi di output.

I diversi stati sono riassunti nella seguente tabella:

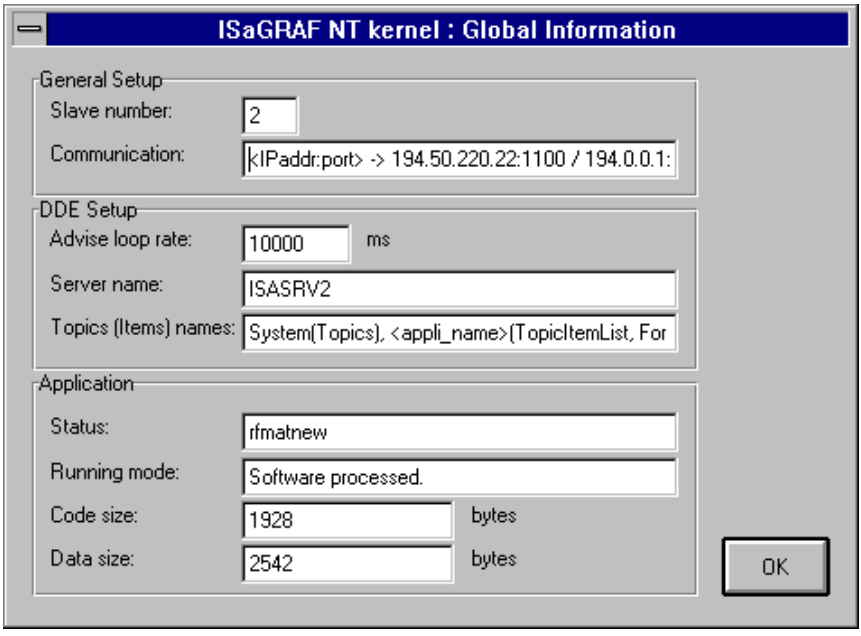
	nessun errore	errori o messaggi (il centro è rosso)
Applicazione in esecuzione		
nessuna applicazione		

Il pulsante Start/Stop del target NT ISaGRAF:

Il pulsante Start/Stop è assolutamente identico alla funzione start/stop del debugger. Il testo riportato dal pulsante esplicita lo stato di esecuzione dell'applicazione. Se l'applicazione è in esecuzione, il testo sarà «Stop», se l'applicazione è ferma (o non è presente), il testo sarà «Start» (si tenga presente che, nel caso non ci sia l'applicazione e venga dato l'avvio, il pulsante assumerà lo stato Stop e poi tornerà allo stato Start).

Informazioni generali su target NT ISaGRAF

Con il comando "View / Information" appare la seguente finestra di dialogo con informazioni generali sul target e sull'esecuzione dell'applicazione:



Sono riportati tre argomenti:

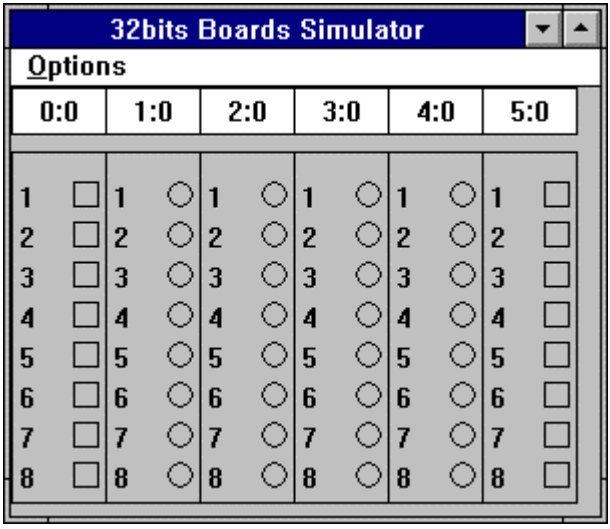
a) General setup:
   - Il numero slave
   - la configurazione di comunicazione (Nel caso il collegamento sia Ethernet, oltre al numero di porta, viene mostrato anche l'elenco degli indirizzi IP disponibili in quel momento sul sistema NT)

b) DDE setup
   - la frequenza del ciclo di notifica
   - il nome del server DDE
   - gli argomenti e gli elementi DDE. Questa informazione è di tipo generico che non rispecchia valori reali. Infatti i campi compresi tra < > vengono sostituiti da valori reali.

c) Application
   - Lo stato dell'applicazione, costituito dal nome dell'applicazione stessa nel caso ci sia un'applicazione in esecuzione, o dalla stringa 'No application' in caso contrario.
   - Il modo di esecuzione dell'applicazione, che indica se l'applicazione venga eseguita dal processore software. In questo caso verrà visualizzata la stringa: «Software processed». O se l'applicazione sia stata compilata da un compilatore C. In questo caso la stringa riportata sarà: «C compiled». nel caso non ci sia un'applicazione in esecuzione la stringa sarà: «No application».
   - La dimensione del codice in byte . Se il modo di esecuzione è «C compiled», questo campo vale 0.
   - La dimensione dei dati in byte, data dalla somma dei dati interni di esecuzione. e dal database delle variabili.

Simulazione di schede virtuali con il target NT ISaGRAF:

Selezionando l'opzione «Simulate I/O» , al successivo riavvio dell'applicativo, appare la seguente finestra:

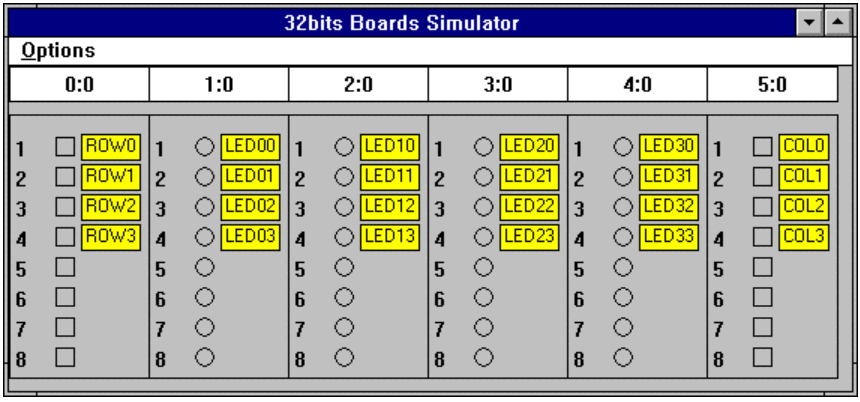


A seconda della configurazione delle connessioni I/O, ci saranno più o meno schede (e diverse) e più o meno variabili (e diverse). I numeri «s:b» in cima a ciascuna scheda rappresentano l'identificativo dello slot (s) e l'identificativo della scheda (b). La numerazione parte da 0 e non è possibile modificarla.
Il 'Simulatore di schede a 32 bit' funziona con lo stato Start/Stop dell'applicazione. Quindi, questa finestra appare se c'è un'applicazione con schede virtuali (o che usa un simulatore di schede) in esecuzione ed è attivo il flag «Simulate I/O». Al contrario, non appena viene premuto il pulsante Stop, viene chiusa. Questa finestra funziona assieme alle chiamate I/O.

Il menu "**Options**" elenca due voci:

> **Variable names** mostra i nomi delle variabili solamente se la tavola dei simboli è stata caricata prima del codice tic.
> **Hexadecimal values** mostra gli interi in formato esadecimale invece del formato decimale predefinito

I nomi delle variabili appaiono nel seguente modo:



C.7 Programmazione "C"

C.7.1 Panoramica

Questo manuale si rivolge agli utenti che siano già a conoscenza dei concetti e dei programmi dell'ambiente di lavoro ISaGRAF. Dopo aver sviluppato applicazioni esclusivamente orientate all'automazione usando le **funzioni di conversione**, le **funzioni "C"** ed i **blocchi funzione** contenuti nelle librerie standard fornite dalla CJ International, è possibile sviluppare funzioni di conversione, funzioni "C" e blocchi funzione "definiti dall'utente". In questo modo è possibile arricchire il PLC target ISaGRAF creando nuove librerie sfruttando al massimo la versatilità della stazione di lavoro e della piattaforma hardware.

Con un sistema di sviluppo "C" ed un po' di esperienza di programmazione "C", questo manuale Vi metterà in condizione di personalizzare il PLC target ISaGRAF per il miglior controllo possibile. Lo sviluppo di queste personalizzazioni migliora le prestazioni del PLC target tanto quanto la comodità e la qualità del sistema di sviluppo ISaGRAF migliorano la programmazione dell'automazione.

Le informazioni riportate in questo documento non si riferiscono al target di uno specifico sistema. Alcune funzionalità, tuttavia, (come le capacità multitasking) non sono applicabili ai sistemi monotasking.

Funzionalità standard dell'ambiente di lavoro ISaGRAF

L'ambiente di lavoro ISaGRAF offre molte funzioni che permettono di gestire librerie di componenti "C" per quanto riguarda lo sviluppo orientato all'automazione. Per quanto riguarda la programmazione dell'automazione, una conversione, una funzione o un blocco funzione "C" possono essere considerati come una "**scatola nera**" completamente definibili dalla propria interfaccia.

Il gestore di libreria ISaGRAF permette di aggiungere componenti alle librerie esistenti e di definirne l'interfaccia tra l'implementazione "C" e l'uso dei componenti nella programmazione **ST/FBD**. Il gestore di librerie ISaGRAF dispone della generazione automatica di schemi per i codici sorgenti di conversioni, funzioni e blocchi funzione ed incorpora strumenti per la modifica di questi file sorgenti. Fare riferimento al Manuale utente **ISaGRAF** per ulteriori informazioni sulle funzioni del Gestore di Libreria.

Sviluppo in linguaggio "C"

L'ambiente di lavoro ISaGRAF non prevede alcun compilatore o programma di compilazione incrociata "C". L'utente, ISaGRAF, per integrare i propri componenti nel kernel ISaGRAF, deve essere in possesso di un compilatore "C", orientato al sistema del target.

Per usare un cross compiler, l'ambiente di lavoro ISaGRAF, mette a disposizione dei punti d'ingresso per avviare , in una finestra DOS, dei file comandi MS-DOS (.bat) definibili dall'utente. Il cross compiler deve essere in grado di funzionare in una finestra di emulazione

DOS, altrimenti è necessario uscire da Windows per eseguire i compilatori ed i linker in ambiente DOS effettivo.

Note tecniche

Il gestore di Libreria permette di abbinare una descrizione ad ogni componente della libreria. Questa **nota tecnica** rappresenta la guida utente del componente "C" in cui il programmatore di automazione trova descritte le corrispondenti conversioni, funzioni o blocchi funzione nelle applicazioni ISaGRAF.

Conversioni, funzioni o blocchi funzione "C" devono essere definite nelle note tecniche con precisione, in modo da permettere al programmatore di automazione di usarle realmente come funzioni ISaGRAF già pronte.

Nel caso di una funzione "C", la nota tecnica deve contenere:

- ❑ in dettaglio, le funzioni elaborate dalla funzione stessa
- ❑ la descrizione completa dei parametri di chiamata
- ❑ il significato del valore di ritorno
- ❑ il tipo dei parametri di chiamata e del valore di ritorno
- ❑ il contesto dell'applicazione

Per un blocco funzione "C", la nota tecnica deve contenere:

- ❑ in dettaglio, le funzioni elaborate dalla funzione di attivazione del blocco
- ❑ la descrizione completa dei parametri di chiamata
- ❑ il significato del valore di ritorno
- ❑ il tipo dei parametri di chiamata e del valore di ritorno
- ❑ il contesto nel quale può essere applicato

Per una funzione di conversione, la nota tecnica deve contenere:

- ❑ l'esatto significato della conversione usata con una variabile di input
- ❑ l'esatto significato della conversione usata con una variabile di output
- ❑ i limiti dei valori che possono essere elaborati dalla conversione

Le note tecniche possono anche comprendere informazioni relative a:

- ❑ l'identificazione completa della conversione, funzione o blocco funzione
- ❑ qualsiasi informazione relativa alla sua manutenzione ed aggiornamenti
- ❑ il sistema target supportato
- ❑ speciali funzionalità multitasking
- ❑ servizi di sistema richiesti, la memoria, i driver …

C.7.2 Funzioni di conversione "C"

L'ambiente di lavoro ISaGRAF incorpora un programma di utilità di **conversione lineare** in grado di effettuare semplici conversioni di I/O analogici in fase di esecuzione sul target PLC ISaGRAF. Questo programma di utilità **non** necessita di sviluppo "C" in quanto è limitato a funzioni continue rigorosamente crescenti o decrescenti. Si veda il Manuale utente ISaGRAF per una descrizione completa di questi strumenti.

Le funzioni di conversione permettono di ottenere qualsiasi conversione complessa usando le funzioni del linguaggio “C”. Normalmente, una funzione di conversione viene definita **sia per gli input che per gli output**. Anche nel caso venga usato un solo verso, per prevenire il crollo del sistema a causa di chiamate errate, è necessario implementare e provare la conversione in entrambi i versi prima di integrarla nel kernel ISaGRAF.

Le funzioni di conversione vengono scritte in linguaggio “C”, compilate e collegate al kernel ISaGRAF. Prima di poter usare le nuove funzioni di conversione nei progetti ISaGRAF, è necessario installare il **kernel espanso** sul target PLC ISaGRAF. Le nuove funzioni di conversione **non** possono essere integrate nel Simulatore ISaGRAF. Le applicazioni ISaGRAF possono essere simulate solo **prima** dell’inserimento delle funzioni di conversione non standard.

Con l’ambiente di lavoro ISaGRAF, viene installato il codice sorgente “C” delle conversioni standard scritte dalla CJ International, che può essere una base di partenza per la creazione di nuove funzioni. Si raccomanda di **non modificare** le funzioni standard per permetterne l’uso da qualsiasi applicazione ISaGRAF. Le conversioni standard fornite con l’ambiente di lavoro ISaGRAF sono supportate dal simulatore ISaGRAF.

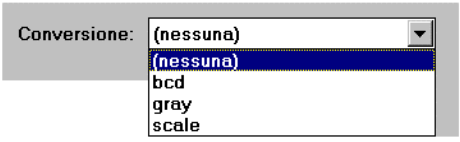
Avvertenza: Le funzioni di conversione sono operazioni **sincrone**, attivate all’esecuzione dal gestore di I/O ISaGRAF, durante le fasi di input o output del ciclo dell’applicazione. Il tempo usato per eseguire una funzione di conversione viene incluso nel calcolo del **tempo di ciclo**. È bene assicurarsi che una funzione di conversione **non** contenga “operazioni di attesa”, in modo da evitare che il ciclo di elaborazione ISaGRAF venga esteso senza motivo.

Aggiungere funzioni alla libreria ISaGRAF

Per aggiungere una nuova funzione di conversione alla libreria ISaGRAF, nell’ambiente di lavoro, si deve usare il Gestore di libreria ISaGRAF. Si usa il comando **«Nuovo»** del menu **«File»** con la libreria **Funzioni di conversione** evidenziata. Non è necessario indicare alcun parametro nell’ambiente di lavoro, in quanto le funzioni di conversione fanno uso di un’interfaccia standard predefinita. Una volta creata una nuova funzione di conversione, è necessario scrivere la relativa **nota tecnica**. Lo schema del codice sorgente “C” per la nuova funzione viene generato automaticamente dal Gestore di libreria ISaGRAF.

Usare una conversione in un progetto ISaGRAF

Le funzioni di conversione definite permettono di filtrare i valori di qualunque variabile analogica di input o di output del progetto selezionato. Per abbinare una funzione di conversione ad una variabile, avviare il dizionario delle variabili, selezionare una variabile analogica di input o di output e modificarne i parametri. Il campo “**Conversione**” della finestra di dialogo per la dichiarazione analogica, permette di impostare la funzione di conversione abbinata ad una variabile analogica di I/O:



Nell'elenco appaiono sia le funzioni che le tabelle. Questo implica che non è possibile usare lo stesso nome per una funzione e per una tabella. Una variabile non può essere abbinata ad una funzione di conversione ancora da definire o da incorporare nel kernel ISaGRAF.

Interfaccia "C" standard

L'interfaccia di una funzione di conversione ha sempre lo stesso formato. I parametri di chiamata e di ritorno vengono passati tramite una struttura. Tale struttura è definita nel file "**TACN0DEF.h**":

```
/*
  Nome: tacn0def.h
  File di definizione delle conversioni per il target
*/

#define DIR_INPUT 0             /* verso = conversione in input */
#define DIR_OUTPUT 1            /* verso = conversione in output */

typedef int32   T_ANA;          /* tipo intero ANA */
typedef float   T_REAL;         /* tipo reale ANA */

typedef struct {                /* struttura di conversione */
     uint16 number;             /* numero di conversione (riservato)  */
     uint16 direction;          /* verso della conversione */
     T_REAL *before;            /* valore prima della conversione */
     T_REAL *after;             /* valore dopo la conversione */
   } str_cnv;

#define ARG_BEFORE (*(arg->before))
#define ARG_AFTER  (*(arg->after))
#define DIRECTION  (arg->direction)

/* eof */
```

La struttura "**str_cnv**" descrive completamente l'interfaccia. L'unico parametro di una funzione di conversione "C" è un puntatore a tale struttura. Il campo "**number**" rappresenta il numero logico della funzione di conversione (la posizione nella libreria ISaGRAF) e non deve essere usato nella programmazione.

Il campo "**direction**" indica se la conversione debba essere applicata ad una variabile di input o ad una variabile di output. Assume il valore **DIR_INPUT** per una conversione in input o il valore **DIR_OUTPUT** per una conversione in output.

Il campo "**before**" rappresenta il valore prima della conversione. Questo campo ha significato diverso per una conversione in input o per una conversione in output. Rappresenta il valore

elettrico (letto dal dispositivo di input) per una conversione in input, quando il campo **direction** ha valore **DIR_INPUT**. Rappresenta il valore fisico (usato nelle formule programmate) per una conversione in output, quando il campo **direction** ha valore **DIR_OUTPUT**.

Il campo " **after** " rappresenta il valore dopo la conversione. Questo campo ha significato diverso per una conversione in input o in output. Rappresenta il valore fisico (usato nelle formule programmate) per una conversione in input, quando il campo **direction** ha valore **DIR_INPUT**. Rappresenta il valore elettrico (inviato al dispositivo di output) per una conversione in output, quando il campo **direction** ha valore **DIR_OUTPUT**.

Il programmatore può usare le definizioni "**ARG_BEFORE**" e "**ARG_AFTER**" per accedere direttamente ai campi **before** ed **after** della struttura passata alla funzione di conversione "C". I valori elaborati sono valori in **virgola mobile a singola precisione**. Il risultato viene convertito ad intero lungo quando la conversione viene effettuata su una variabile analogica intera. Questo significa che la stessa conversione può essere usata per variabili analogiche di I/O sia intere che reali.

Codice sorgente

Poiché la funzione di conversione può essere usata per variabili analogiche sia di input che di output, il codice sorgente "C" della funzione è suddiviso in due parti principali: la conversione in input e la conversione in output. Il campo **direction** della struttura dell'interfaccia indica la conversione che deve essere effettuata. Il Gestore di libreria ISaGRAF genera automaticamente lo schema completo della funzione, una volta creata la funzione di conversione. Questo include l'algoritmo decisionale principale "**IF**". Di seguito è riportato lo schema standard di una funzione di conversione:

```
/*
  funzione di conversione
  nome: sample
*/

#include <tasy0def.h>
#include <tacn0def.h>

void CNV_sample (str_cnv *arg)
{
        if (DIRECTION == DIR_INPUT) { /*INPUT CONV*/

        }
        else { /*OUTPUT CONV*/

        }
}
```

/* La seguente funzione mostra il collegamento con la gestione I/O ISaGRAF, che usa il nome della conversione. Questa funzione viene completamente generata dal Library Manager ISaGRAF. */

```
UFP cnvdef_sample (char *name)
{
        sys_strcpy (name, "SAMPLE"); /* assegna il nome della conversione */
        return (CNV_sample);        /* ritorna l'implementazione della funzione */
}
```

Il modo migliore per completare la parte specifica della funzione, consiste nello scrivere due funzioni locali separate, una per la conversione in input ed una per la conversione in output. Queste funzioni verranno chiamate dall'algoritmo principale, come mostrato nei commenti dell'esempio precedente, nella struttura principale **IF**.

Il file di include "**TASY0DEF.H**" del kernel ISaGRAF è necessario per le definizioni specifiche del sistema. Contiene anche la definizione del tipo **UFP**, che rappresenta un puntatore ad una funzione void e viene usato nella dichiarazione di funzione.

Collegamenti tra progetti ed implementazione "C"

Il collegamento logico tra l'implementazione di una funzione di conversione e l'uso della conversione in un progetto ISaGRAF avviene tramite il nome della conversione. Una funzione di "dichiarazione" viene aggiunta al codice sorgente "C" della funzione di conversione. Questa funzione viene chiamata una sola volta, all'avvio dell'applicazione ed indica al gestore di I/O ISaGRAF il nome della conversione cui corrisponde la funzione da implementare. Di seguito è riportato il formato standard di questa funzione di dichiarazione:

```
UFP cnvdef_xxx (char *name)
{
        strcpy (name, "XXX");       /* assegna il nome della conversione */
        return (CNV_xxx);           /* ritorna l'implementazione della funzione */
}
/* (xxx è il nome della conversione) */
```

Il nome della funzione, usato dall'istruzione **strcpy**, deve essere scritto in **maiuscolo**. Deve essere scritto in minuscolo nella funzione di implementazione della conversione e nella dichiarazione di funzione.

L'uso dei prefissi "**CNV_**" e "**cnvdef_**" consente di usare, per una conversione, una parola chiave riservata del linguaggio "C", oppure il nome di una funzione esistente nelle librerie "C" ISaGRAF.

Alla funzione di dichiarazione possono essere aggiunte altre istruzioni per specifiche operazioni di inizializzazione relative alla conversione. Il sistema ISaGRAF assicura che questa funzione venga chiamata **una sola volta** all'avvio dell'applicazione.

La funzione di dichiarazione viene chiamata per qualsiasi funzione di conversione integrata, anche se non viene usata dall'applicazione ISaGRAF. L'esecuzione del kernel ISaGRAF fallisce con un errore fatale se una conversione usata nell'applicazione non è integrata nel kernel.

Prima di collegare le nuove funzioni al kernel, è necessario scrivere un altro file sorgente "C", chiamato "**GRCN0LIB.C**", ed inserirlo (con le relative funzioni di conversione) nelle lista dei file per il linker. Il file "**GRCN0LIB.C**" contiene solamente un array di dichiarazioni di funzioni. Questo array viene letto durante le operazioni di inizializzazione per creare un collegamento dinamico con le funzioni di conversione scritte in "C". Questo è un esempio di tale file:

```
/* File "GRCN0LIB.c" – Esempio con conversioni della libreria standard */

#include <tasy0def.h>                /* richiesto per le definizioni di tipo */

extern UFP cnvdef_scale (char *name);      /* dichiarazione di funzione per la
conversione SCALE */
extern UFP cnvdef_bcd (char *name);        /* dichiarazione di funzione per la
conversione BCD */

UFP_LIST CNVDEF[ ] = {          /* array di dichiarazione di funzioni per */
      /* le funzioni di conversione integrate */
      cnvdef_scale,
      cnvdef_bcd,

NULL };

/* fine del file */
```

L'array **CNVDEF** deve essere concluso da un puntatore NULL. Non rispettando questa condizione, possono verificarsi conflitti. Se l'array **CNVDEF** non viene definito, collegando il nuovo kernel ISaGRAF, si verificheranno situazioni di riferimenti non risolti.

Con questo file, è possibile generare un nuovo kernel che include tutte le conversioni esistenti. Un kernel può anche essere realizzato specificatamente per un singolo progetto, inserendo nell'array **CNVDEF** unicamente le conversioni usate nel progetto. Il file "**GRCN0LIB.C**" viene generato automaticamente dal Generatore di codice ISaGRAF, nella fase di realizzazione del codice dell'applicazione. Il file viene collocato nella directory del progetto ISaGRAF ed incorpora solamente le conversioni usate nel progetto.

Limiti

La libreria ISaGRAF può contenere fino a **128** funzioni di conversione. Qualsiasi tipo di operazione può essere elaborata in una funzione di conversione. Si noti che le funzioni vengono chiamate dal ciclo ISaGRAF in modo **sincrono**, quindi l'esecuzione della funzione influisce direttamente sul tempo di ciclo.

C.7.3 Funzioni "C"

Le funzioni “C” permettono di ampliare le peculiarità standard dei linguaggi **ST** e **FBD**. Possono essere usate per realizzare calcoli specifici, chiamate di sistema, comunicazioni o per installare un insieme di servizi per permettere il dialogo tra un’applicazione ISaGRAF ed altri processi. Le funzioni vengono scritte in linguaggio “C”, compilate e collegate al kernel ISaGRAF. Il **kernel ampliato** deve essere installato su un PLC target ISaGRAF prima di poter usare le nuove funzioni in progetti ISaGRAF.

Le nuove funzioni **non** possono essere integrate nel Simulatore ISaGRAF. Le applicazioni ISaGRAF possono essere simulate solo **prima** dell’inserimento delle funzioni di conversione non standard.

Avvertenza: Le funzioni sono operazioni **sincrone**, attivate all’esecuzione dal kernel ISaGRAF, durante il ciclo dell’applicazione. Il tempo usato per eseguire una funzione viene incluso nel calcolo del **tempo di ciclo** dell’applicazione ISaGRAF. È bene assicurarsi che una funzione non contenga “operazioni di attesa”, in modo da evitare che il ciclo di elaborazione ISaGRAF venga esteso senza motivo.

Aggiungere funzioni alla libreria ISaGRAF

Per aggiungere una nuova funzione “C” alla libreria ISaGRAF, si deve usare il Gestore di libreria ISaGRAF. Si usa il comando **«Nuovo»** del menu **«File»** con la libreria **Funzioni C** evidenziata. Una volta creata una nuova funzione, è necessario scrivere la relativa **nota tecnica**. Lo schema del codice sorgente “C” per la nuova funzione viene generato automaticamente dal gestore di libreria ISaGRAF.

Il comando **«Parametri»** del menu **«Modifica»** consente di definire i parametri di chiamata e ritorno della nuova funzione.

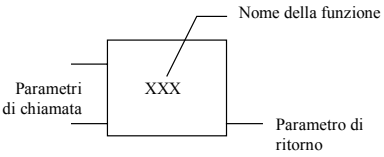
Usare una funzione "C" in un progetto ISaGRAF

Qualsiasi funzione integrata “C” può essere usata come funzione standard nei programmi di un progetto ISaGRAF. Le funzioni “C” possono essere chiamate dai linguaggi **ST** e **FBD** e da istruzioni particolari del linguaggio **SFC**.

Per chiamare una funzione “C” dal linguaggio **ST**, è necessario rispettare le convenzioni di chiamata a funzione del linguaggio. I parametri di chiamata della funzione sono scritti dopo il nome della funzione, circondati da parentesi e separati da virgole. L’espressione costituisce il valore ritornato dalla funzione. Una chiamata a funzione “C” può essere inserita in una qualunque istruzione di assegnamento o espressione composita. Questo che segue, è un esempio di una funzione “C” usata in un’istruzione di assegnamento:

```
result := ProcName (par1, par2, ... parN);
```

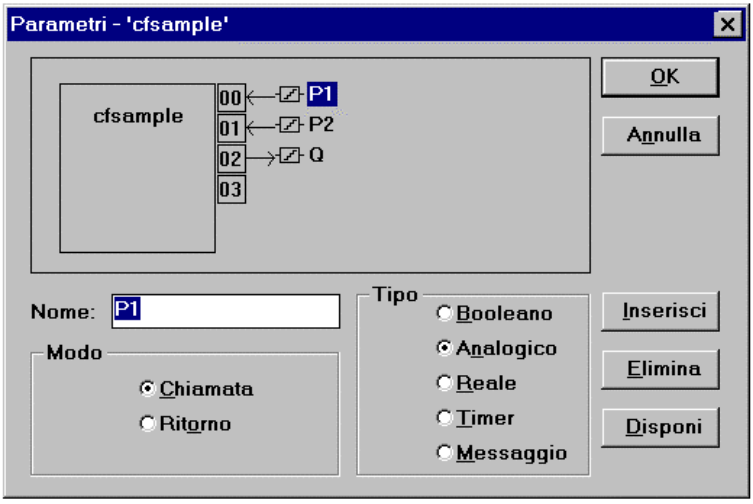
Un programma **FBD** può chiamare qualsiasi funzione “C”. Una funzione viene usata come una qualunque casella di funzione standard. Il parametro di ritorno è collegato al lato destro della casella. Quello che segue è l’aspetto standard di tale casella funzione:



Una funzione "C" può essere chiamata da qualsiasi blocco azione **SFC** o usata in qualsiasi condizione booleana associata ad una transizione.

Definizione dell'interfaccia "C" di una funzione

Il comando **«Parametri»** del menu **«Modifica»** permette di definire i parametri di chiamata e di ritorno di una nuova funzione. Una funzione può avere fino a **31** parametri di chiamata e sempre **un solo** parametro di ritorno. La seguente finestra di dialogo serve a descrivere i parametri di una funzione "C":



L'elenco nella parte superiore della finestra mostra i parametri della funzione "C", ordinati secondo il prototipo di chiamata di funzione: per primi i parametri di chiamata, infine il parametro di ritorno. La porzione inferiore della finestra mostra la descrizione dettagliata del parametro correntemente evidenziato nell'elenco:

- il nome del parametro
- il verso (chiamata o ritorno) del parametro
- il tipo del parametro

Un parametro può avere uno qualsiasi dei tipi di dati ISaGRAF: booleano, intero analogico, reale analogico, timer o messaggio. è necessario distinguere tra analogici interi e reali. Di seguito viene riportata la corrispondenza tra tipi ISaGRAF e tipi "C":

BOOLEANO	Unsigned long	word senza segno a 32 bit: 1=true / 0=false
ANALOGICO	Long	word intera con segno a 32 bit
REALE	Float	valore in virgola mobile a singola precisione
TIMER	Unsigned long	word intera senza segno a 32 bit (l'unità corrisponde ad 1 millisecondo)
MESSAGGIO	Char *	stringa di caratteri.

Un messaggio passato ad una funzione "C" non può contenere caratteri null. La stringa passata al codice "C" è conclusa da null. Non bisogna dimenticare che il parametro di ritorno deve essere l'ultimo della lista. L'assegnamento del nome deve sottostare alle seguenti regole:

- la lunghezza del nome non può eccedere i 16 caratteri
- il primo carattere deve essere una lettera
- i caratteri seguenti possono essere lettere, cifre o il carattere «_»
- nell'attribuzione del nome non viene fatta distinzione tra lettere maiuscole e minuscole.

Non è possibile assegnare lo stesso nome a più variabili. Un parametro di chiamata non può avere lo stesso nome del parametro di ritorno. Lo stesso nome **può** essere usato per parametri di diverse funzioni. Il nome predefinito per il parametro di ritorno è **"Q"**. Il nome può essere modificato liberamente. Il nome del parametro serve ad identificare il parametro stesso nel codice sorgente "C".

Il comando **«Inserisci»** permette di inserire un nuovo parametro prima del parametro evidenziato. Il comando **«Elimina»** permette di cancellare il parametro evidenziato. Il comando **«Disponi»** riordina automaticamente i parametri, sistemando il parametro di ritorno in fondo all'elenco. Premendo il pulsante **«OK»**, viene salvata la definizione dell'interfaccia della funzione e si chiude la finestra di dialogo. Premendo il pulsante **«Annulla»**, la finestra di dialogo si chiude senza che vengano apportate modifiche all'interfaccia della funzione.

Interfaccia delle funzioni "C"

L'interfaccia di una funzione dipende dalla definizione dei suoi parametri. I parametri di chiamata e di ritorno vengono passati tramite una struttura. Questa struttura è definita nel file "**GRUS0nnn.H**", con "**nnn**" che rappresenta il numero logico della funzione nella libreria ISaGRAF. Quello che segue è un esempio di interfaccia "C" per la funzione "**SIN**" (calcolo del seno):

```
/* File: GRUS0255.h - funzione "esempio" */

typedef  long          T_BOO;
typedef  long          T_ANA;
typedef  float         T_REAL;
typedef  long          T_TMR;
typedef  char          *T_MSG;
```

```
typedef struct {
        /* CHIAMATA  */          T_REAL   _param1;
        /* RITORNO */            T_REAL   _param2;
} str_arg;

#define PARAM1                  (arg->_param1)
#define PARAM2                  (arg->_param2)

/* fine del file */
```

La relazione tra i tipi ISaGRAF ed i tipi “C” viene mostrata subito sotto. La definizione dei tipi ISaGRAF come tipi “C”, è contenuta del file di definizione della funzione.

booleano	T_BOO	long (32 bit)
analogico intero	T_ANA	long
analogico reale	T_REAL	float (32 bit – singola precisione)
timer	T_TMR	long
messaggio	T_MSG	char * (32 bit – puntatore a carattere)

Ogni campo della struttura "**str_arg**" corrisponde ad un parametro della funzione. Il parametro di ritorno della funzione è l’ultimo della struttura. I parametri di chiamata hanno, nella struttura, lo stesso ordine stabilito nella definizione di funzione. È definito un identificativo in maiuscolo per aver accesso direttamente ad un parametro della struttura passata all’implementazione “C” della funzione. I nomi degli identificativi corrispondono a quelli inseriti per la definizione della funzione dal Gestore di libreria ISaGRAF.

Il file di definizione “C” viene aggiornato ogni volta che l’interfaccia della funzione viene modificata dal Gestore di libreria ISaGRAF. Questo assicura la perfetta corrispondenza tra l’implementazione della funzione ed il suo uso nei programmi delle applicazioni ISaGRAF.

Codice sorgente

Quello che segue è lo schema standard dell’implementazione di una funzione “C”:

```
/* Esempio di funzione utente – Il numero è "255" – Il nome è "SAMPLE" */

#include "tasy0def.h"           /* Definizioni comuni del kernel ISaGRAF */
#include "grus0255.h"           /* Definizione dell’interfaccia per la funzione
255 */

void USP_sample (str_arg *arg)
{
        /* corpo della funzione */
}
```

/* La seguente funzione viene usata per l'inizializzazione della funzione e per la dichiarazione della sua implementazione. Realizza il collegamento con il kernel ISaGRAF, usando il nome della funzione. Questa funzione viene completamente generata dal Library Manager ISaGRAF. */

```
UFP uspdef_sample (char *name)
{
        strcpy (name, "SAMPLE"); /* assegna il nome della funzione */
        return (USP_sample);     /* ritorna l'implementazione della funzione */
}

/* fine del file */
```

Il file della dichiarazione include "**TASY0DEF.H**" del kernel ISaGRAF è necessario per le definizioni specifiche del sistema. Contiene anche la definizione del tipo **UFP**, che rappresenta un puntatore ad una funzione void e viene usato nella dichiarazione di funzione.

Collegamenti tra i progetti e l'implementazione "C"

Il collegamento logico tra l'implementazione di una funzione "C" e l'uso in un progetto ISaGRAF avviene tramite il nome della funzione. Una funzione di "dichiarazione" viene aggiunta al codice sorgente "C" della funzione. Questa funzione viene chiamata una sola volta, all'avvio dell'applicazione ed indica al gestore I/O ISaGRAF il nome della funzione corrispondente alla funzione implementata. Di seguito è riportato il formato standard di questa funzione di dichiarazione:

```
UFP uspdef_xxx (char *name)
{
        strcpy (name, "XXX");      /* assegna il nome della conversione */
        return (USP_xxx);          /* ritorna l'implementazione della funzione */
}
/* (xxx è il nome della funzione) */
```

Il nome della funzione "C", usato dall'istruzione **strcpy**, deve essere scritto in **maiuscolo**. Deve essere scritto in minuscolo nella funzione di implementazione e nella dichiarazione di funzione. L'uso dei prefissi "**USP_**" e "**uspdef_**" nella funzione di implementazione e nella funzione di definizione, consente di assegnare ad una funzione il nome di una parola chiave riservata del linguaggio "C", oppure il nome di una funzione esistente nelle librerie "C" ISaGRAF.

Alla funzione di dichiarazione possono essere aggiunte altre istruzioni per specifiche operazioni di inizializzazione relative alla funzione. Il sistema ISaGRAF assicura che questa funzione venga chiamata **una sola volta** all'avvio dell'applicazione. La funzione di dichiarazione viene chiamata per qualsiasi funzione integrata, anche se non viene usata nei programmi dell'applicazione ISaGRAF. L'esecuzione del kernel ISaGRAF fallisce con un errore fatale se una funzione "C" usata nell'applicazione non è integrata nel kernel.

Prima di collegare le nuove funzioni al kernel, è necessario scrivere un altro file sorgente “C”, chiamato "**GRUS0LIB.C**", ed inserirlo (con le relative funzioni) nelle lista dei file per il linker. Il file "**GRUS0LIB.C**" contiene solamente un array di dichiarazioni di funzioni. Questo array viene letto durante le operazioni di inizializzazione per creare un collegamento dinamico con le funzioni scritte in “C”. Questo è un esempio di tale file:

```
/* File "GRUS0LIB.c" – Esempio che fa uso di funzioni trigonometriche */

#include <tasy0def.h>                /* richiesto per le definizioni di tipo */

extern UFP uspdef_fc1 (char *name);        /* dichiarazione delle funzioni */
extern UFP uspdef_fc2 (char *name);
extern UFP uspdef_fc3 (char *name);
extern UFP uspdef_fc4 (char *name);

UFP_LIST USPDEF[ ] = {            /* array di dichiarazione delle funzioni */
        /* for integrated functions */
        uspdef_fc1,
        uspdef_fc2,
        uspdef_fc3,
        uspdef_fc4,

NULL };

/* fine del file */
```

L’array **USPDEF** deve essere concluso da un puntatore NULL. Non rispettando questa condizione, possono verificarsi conflitti. Se l’array **USPDEF** non viene definito, collegando il nuovo kernel ISaGRAF, si verificheranno situazioni di riferimenti non risolti. Scrivendo questo file, è possibile generare un nuovo kernel che include tutte le funzioni esistenti. Un kernel può anche essere realizzato specificatamente per un singolo progetto, inserendo nell’array **USPDEF** unicamente le conversioni usate nel progetto. Il file " **GRUS0LIB.C** " viene generato automaticamente dal Generatore di codice ISaGRAF nella fase di realizzazione del codice dell’applicazione. Il file viene collocato nella directory del progetto ISaGRAF ed incorpora solamente le conversioni usate nel progetto.

Limiti

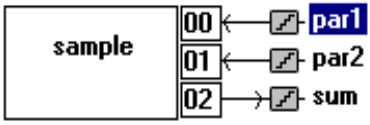
La libreria ISaGRAF può contenere fino a **128** funzioni “C”. Qualsiasi tipo di operazione può essere elaborata in una funzione. Si noti che le funzioni vengono chiamate dal ciclo ISaGRAF in modo **sincrono**, quindi l’esecuzione della funzione influisce direttamente sul tempo di ciclo.

Un esempio completo

Quella che segue è la programmazione completa della funzione "**sample**", che esegue solamente un’addizione. Subito sotto la nota tecnica della funzione:

nome:	SAMPLE
descrizione:	esegue solamente un'addizione intera analogica
data di creazione:	1 Luglio 1992
autore:	CJ International
chiamata:	par1, par2: operandi interi
ritorno:	somma intera
prototipo:	somma := sample (par1, par2);

Subito sotto l'interfaccia della funzione:



Subito sotto il codice sorgente "C" dell'header della funzione:

```
/* File: GRUS0255.h – definizioni di funzione C utente - Nome: sample */

/* definizione dei tipi di dati ISaGRAF standard */

typedef long  T_BOO;
typedef long  T_ANA;
typedef float T_REAL;
typedef long  T_TMR;
typedef char  *T_MSG;

/* definizione della struttura dei parametri di chiamata e di ritorno */

typedef struct {
        T_ANA _par1;            /* parametro di chiamata #1 */
        T_ANA _par2;            /* parametro di chiamata #2 */
        T_ANA _sum;             /* parametro di ritorno */
} str_arg;

/* identificatori usati per accedere ai parametri di chiamata e di ritorno */

#define PAR1                    (arg->_par1)
#define PAR2                    (arg->_par2)
#define SUM                     (arg->_sum)
```

```
/* fine del file */
```

Subito sotto il codice sorgente “C” della funzione. Solamente le righe in grassetto sono state inserite manualmente dal programmatore C.

```
/* File: GRUS0255.c – funzione utente C - Nome: SAMPLE */

#include "tasy0def.h"          /* Necessario per la definizione dei tipi */
#include "grus0255.h"          /* Header del sorgente C della funzione */

/* Servizio principale C: calcolo dell'addizione */

void USP_sample (str_arg *arg)
{
        SUM = PAR1 + PAR2;
}

/* servizio di dichiarazione richiesto per il collegamento dinamico con il kernel
ISaGRAF */

UFP uspdef_sample (char *name)
{
        strcpy (name, "SAMPLE");
        return (USP_sample);
}
/* fine del file */
```

C.7.4 BLOCCHI FUNZIONE "C"

I blocchi funzione "C" associano operazioni e dati **statici**. Completano l’insieme delle funzioni “C” permettendo l’elaborazione di oggetti statici. Vengono comunemente usati per estendere le peculiarità standard dei linguaggi **ST** e **FBD**. Diversamente dalle funzioni, che processano **valori**, i blocchi funzione possono processare dati statici. Questo significa che l’algoritmo di un blocco funzione è in grado di gestire variazioni di dati nel tempo.

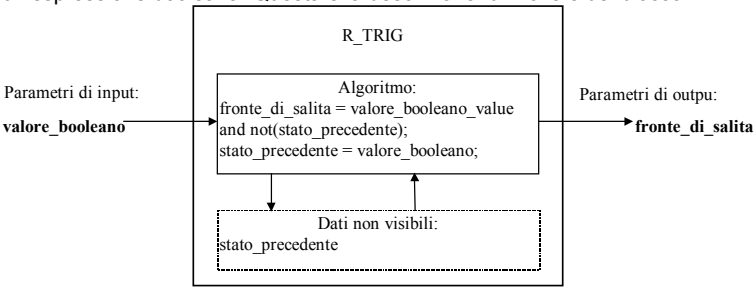
I blocchi funzione vengono scritti in linguaggio “C”, compilati e collegati al kernel ISaGRAF. Prima di poter usare i nuovi blocchi funzione nei progetti ISaGRAF, è necessario installare il kernel espanso sul target PLC ISaGRAF. I nuovi blocchi funzione non possono essere integrati nel Simulatore ISaGRAF. Le applicazioni ISaGRAF possono essere simulate solo **prima** dell’inserimento delle funzioni non standard.

Avvertenza: Le chiamate a blocchi funzione sono operazioni **sincrone**, attivate all’esecuzione dal gestore di I/O ISaGRAF, durante il ciclo dell’applicazione. Il tempo usato per eseguire un

blocco funzione viene incluso nel calcolo del **tempo di ciclo**. È bene assicurarsi che un blocco funzione non contenga "operazioni di attesa", in modo da evitare che il ciclo di elaborazione ISaGRAF non ecceda il tempo massimo consentito.

Dichiarare le istanze di blocchi funzione

Un blocco funzione è un oggetto che combina operazioni e dati statici. Viene riportato di seguito l'esempio di un blocco funzione "**R_TRIG**" che rileva il fronte di salita di un'espressione booleana. Questa è la descrizione funzionale del blocco:



La variabile statica non visibile **"stato_precedente"** è necessaria al calcolo del fronte. Questa variabile deve essere diversa per ogni uso del blocco funzione "**TRIG**" nell'applicazione. Le istanze dei blocchi funzione usati nel linguaggio ST devono essere dichiarate nel dizionario. Poiché un blocco funzione possiede internamente dei dati non visibili, **ogni copia** (istanza) di un blocco funzione deve essere identificata da un nome univoco. L'assegnamento del nome al tipo di blocco viene effettuato usando il gestore di libreria. L'assegnamento del nome alle istanze viene effettuata usando l'editor del dizionario.

I blocchi funzione usati nel linguaggio FBD non devono essere dichiarati, poiché l'editor FBD ISaGRAF provvede a dichiarare automaticamente le istanze dei blocchi usati. Le istanze di blocco funzione dichiarate automaticamente dall'editor FBD, hanno sempre campo di validità **LOCALE** al programma modificato.

Aggiungere un blocco funzione alla libreria ISaGRAF

Per aggiungere un nuovo blocco funzione alla libreria ISaGRAF, nell'ambiente di lavoro, si deve usare il Gestore di libreria ISaGRAF. Si usa il comando **«Nuovo»** del menu **«File»** con la libreria di blocchi funzione evidenziata. Una volta creato un nuovo blocco funzione, è necessario scrivere la relativa **nota tecnica**. Lo schema del codice sorgente "C" per il nuovo blocco funzione viene generato automaticamente dal gestore di libreria ISaGRAF.
Il comando **«Parametri»** del menu **«Modifica»** consente di definire i parametri di chiamata e ritorno del nuovo blocco funzione.

Usare un blocco funzione "C" in un progetto ISaGRAF

Qualsiasi blocco funzione "C" integrato può essere usato nei programmi di un progetto ISaGRAF. I blocchi funzione "C" possono essere chiamati dai linguaggi **ST** e **FBD**

Per chiamare un blocco funzione "C" dal linguaggio **ST**, è necessario rispettare le convenzioni di chiamata del linguaggio. I parametri di chiamata del blocco sono scritti dopo il nome della

funzione, circondati da parentesi e separati da virgole. I parametri di ritorno sono accessibili ad uno ad uno. Ogni parametro di ritorno è rappresentato da un nome composto dal nome dell'istanza del blocco e dal nome dei parametri. Gli elementi del nome sono separati tra loro da un punto. Ad esempio, il nome:

FBINSTNAME.parname

indica il parametro di ritorno "**parname**", dell'istanza di blocco funzione "**FBINSTNAME**".

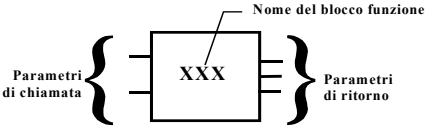
Le istanze dei blocchi funzione usati nel linguaggio ST devono essere dichiarate nel dizionario. Ogni copia (istanza) di un blocco funzione deve essere identificata da un nome univoco: segue sotto un esempio di una dichiarazione di istanza nel dizionario ISaGRAF:

istanza:	TRIG1	tipo:	R_TRIG
	TRIG2		R_TRIG

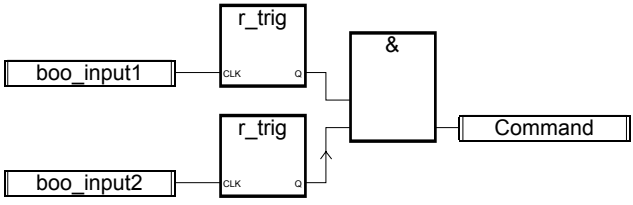
E subito sotto un esempio che fa uso in un programma ST di queste istanze dichiarate:

```
TRIG1 (boo_input1);
TRIG2 (boo_input2);
Command := (TRIG1.Q & TRIG2.Q);
```

Un programma **FBD** può chiamare qualsiasi blocco funzione "C". Un blocco funzione viene usato come una qualunque casella di funzione standard. Il parametri di chiamata sono collegati al lato sinistro della casella. I parametri di ritorno sono connessi al lato destro della casella. Quello che segue è il formato standard di un blocco funzione:



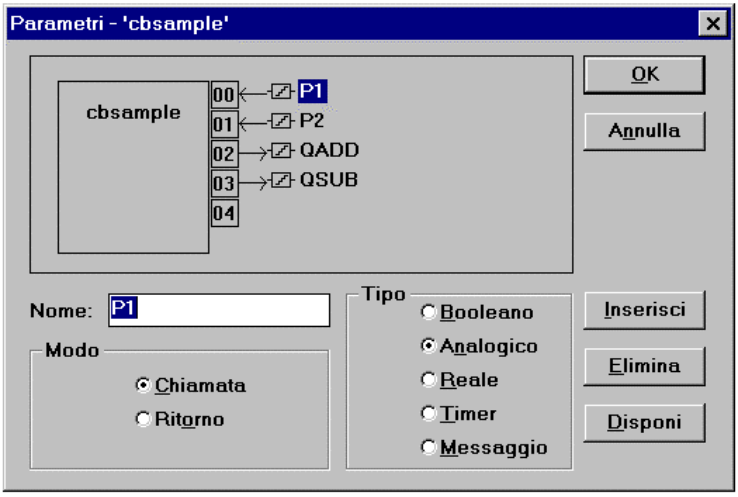
I blocchi funzione usati dal linguaggio FBD non devono essere dichiarati, poiché l'editor FBD provvede a dichiarare automaticamente le istanze dei blocchi usati. Le istanze dei blocchi funzione dichiarate automaticamente dall'editor FBD hanno sempre campo di validità **LOCALE** al programma modificato. Subito sotto viene riportato l'esempio precedente programmato in linguaggio FBD:



Definizione dell'interfaccia di un blocco funzione "C"

Il comando **«Parametri»** del menu **«Modifica»** permette di definire i parametri di un nuovo blocco funzione. Un blocco funzione può avere fino a **32** parametri complessivi, liberamente

definibili come parametri di chiamata o di ritorno. Diversamente da una funzione “C”, un blocco funzione può avere più parametri di ritorno. La seguente finestra di dialogo serve a descrivere i parametri di un blocco funzione “C”:



L’elenco nella parte superiore della finestra mostra i parametri del blocco funzione “C”, ordinati secondo il prototipo di chiamata di funzione: per primi i parametri di chiamata, infine i parametri di ritorno. La porzione inferiore della finestra mostra la descrizione dettagliata del parametro correntemente evidenziato nell’elenco:

- il nome del parametro
- il verso (chiamata o ritorno) del parametro
- il tipo del parametro

Un parametro può avere uno qualsiasi dei tipi di dati ISaGRAF: booleano, intero analogico, reale analogico, timer o messaggio. è necessario distinguere tra analogici interi e reali. Di seguito viene riportata la corrispondenza tra tipi ISaGRAF e tipi “C”:

BOOLEANO	unsigned long	word senza segno a 32 bit: 1=true / 0=false
ANALOGICO	long	word intera con segno a 32 bit
REALE	float	valore in virgola mobile a singola precisione
TIMER	unsigned long	word intera senza segno a 32 bit (l’unità corrisponde ad 1 millisecondo)
MESSAGGIO	char *	stringa di caratteri.

Un messaggio passato ad una funzione “C” non può contenere caratteri null. La stringa passata al codice “C” è conclusa da null. Non bisogna dimenticare che i parametri di ritorno

devono essere gli ultimi della lista. L'assegnamento del nome deve sottostare alle seguenti regole:

- la lunghezza del nome non può eccedere i 16 caratteri
- il primo carattere deve essere una lettera
- i caratteri seguenti possono essere lettere, cifre o il carattere «_»
- il nome non è sensibile a minuscole o maiuscole

Non è possibile assegnare lo stesso nome a più parametri di un blocco funzione. Un parametro di chiamata non può avere lo stesso nome di un parametro di ritorno. Lo stesso nome può essere usato per parametri di differenti blocchi funzione. Il nome del parametro serve ad identificare il parametro stesso nel codice sorgente "C".

Il comando **«Inserisci»** permette di inserire un nuovo parametro prima del parametro evidenziato. Il comando **«Elimina»** permette di cancellare il parametro evidenziato. Il comando **«Disponi»** riordina automaticamente i parametri, sistemando il parametro di ritorno in fondo all'elenco. Premendo il pulsante **«OK»**, viene salvata la definizione dell'interfaccia del blocchi funzione e si chiude la finestra di dialogo. Premendo il pulsante **«Annulla»**, la finestra di dialogo si chiude senza che vengano apportate modifiche all'interfaccia del blocchi funzione.

Interfaccia "C" per blocchi funzione

L'interfaccia di un blocco funzione dipende dalla definizione dei suoi parametri. I parametri di chiamata vengono passati tramite una struttura. Questa struttura è definita nel file "**GRFB0nnn.H**", con "**nnn**" che rappresenta il numero logico del blocco funzione nella libreria ISaGRAF. Quello che segue è un esempio di interfaccia "C" per il blocco funzione "**LIM_ALRM**" (allarme di limiti):

```
/* interfaccia di blocco funzione - nome: sample */

/* tipi IsaGRAF standard */

typedef                 long    T_BOO;
typedef                 long    T_ANA;
typedef                 float   T_REAL;
typedef                 long    T_TMR;
typedef                 char    *T_MSG;

/* struttura dei parametri di chiamata */

typedef struct {
        /* CALL  */             T_BOO _par1;
        /* CALL  */             T_BOO _par2;
} str_arg;

/* accesso ai campi della struttura str_arg */
```

```
#define                 PAR1    (arg->_par1)
#define                 PAR2    (arg->_par2)

/* numeri logici dei parametri di ritorno */

#define                 FBLPNO_Q1    0
#define                 FBLPNO_Q2    1

/* fine del file */
```

La relazione tra i tipi ISaGRAF ed i tipi “C” viene mostrata subito sotto. La definizione dei tipi ISaGRAF come tipi “C”, è contenuta del file di definizione della funzione.

booleano	T_BOO	lungo (32 bits)
analogico	T_ANA	lungo
reale	T_REAL	float (32 bits – singola precisione)
timer	T_TMR	lungo
messaggio	T_MSG	char * (32 bits – puntatore a carattere)

Ogni campo della struttura "**str_arg**" corrisponde ad un parametro di chiamata del blocco funzione. I parametri hanno, nella struttura, lo stesso ordine stabilito nella definizione di blocco funzione. È definito un identificativo in maiuscolo per aver accesso direttamente ad un parametro della struttura passata all’implementazione “C” del blocco funzione. I nomi degli identificativi corrispondono a quelli inseriti per la definizione del blocco funzione dal Gestore di libreria ISaGRAF.

L’ordine usato per la numerazione dei parametri di ritorno è quello stabilito nella definizione di blocco funzione. Il numero logico del primo parametro di ritorno è sempre **0**.

è preferibile usare gli identificativi definiti invece dei valori numerici per rappresentare i parametri di ritorno nella programmazione del sorgente “C”. In questo modo il file sorgente può essere facilmente ricompilato dopo una modifica alla definizione dell’interfaccia.

Il file di definizione “C” viene aggiornato ogni volta che l’interfaccia del blocco funzione viene modificata dal Gestore di libreria ISaGRAF. Questo assicura la perfetta corrispondenza tra l’implementazione del blocco funzione ed il suo uso nei programmi delle applicazioni ISaGRAF.

Codice sorgente

L’implementazione in linguaggio "C" di un blocco funzione è suddivisa in tre diversi punti d’ingresso:

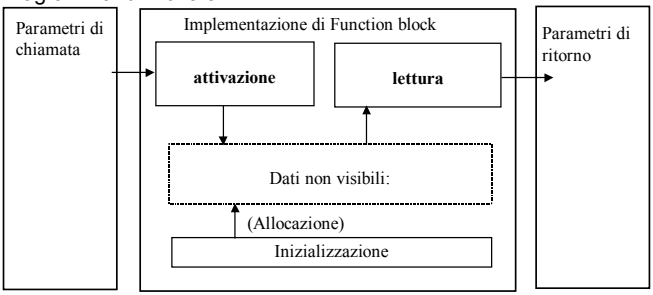
- ❑ servizio di inizializzazione
- ❑ servizio di attivazione – elaborazione dei parametri di chiamata
- ❑ servizio di lettura dei parametri di ritorno

Per ogni istanza di uno stesso blocco funzione viene usato lo stesso codice, che non viene quindi duplicato. Una struttura di **dati statici** è associata ad ogni istanza. L'accesso a tali dati, che contengono le "variabili non visibili" dell'istanza del blocco funzione, non può avvenire direttamente da programmazione ISaGRAF.

Il "servizio di attivazione" viene chiamato una volta per ogni istanza di ciascun blocco usato, per ciascun ciclo target. Esso provvede ad elaborare i parametri di chiamata ed aggiorna i dati ad essi associati. Rappresenta "l'algoritmo principale" del blocco funzione.

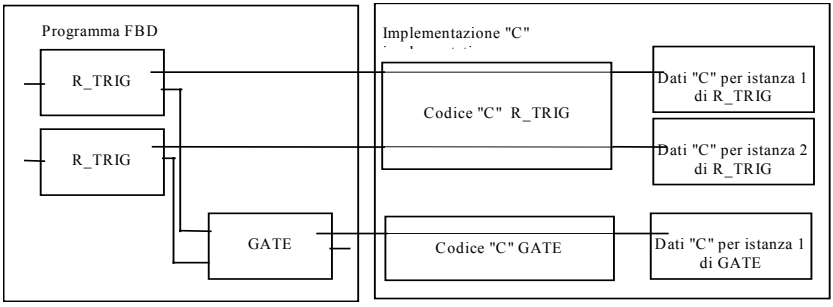
Il "servizio di lettura" viene chiamato dal kernel ISaGRAF per leggere il valore corrente di un parametro di ritorno di un'istanza. Questo servizio non necessita di particolari operazioni di calcolo. Provvede solo a trasferire i dati non visibili all'applicazione ISaGRAF.

Diagramma funzionale:



- **Dati statici dei blocchi funzione**

Un blocco funzione associa operazioni e dati statici. Una struttura dati è associata a ciascuna istanza di uno stesso blocco funzione. Ogni volta che un blocco funzione viene usato dalla programmazione ST o FBD, corrisponde ad una istanza ed a una struttura dati. Il seguente esempio mostra la corrispondenza tra strutture dati "C" e le istanze di blocco funzione usate in un programma FBD:



La memoria occorrente per la struttura dati di ogni istanza viene allocata dal sistema ISaGRAF all'avvio dell'applicazione. Un puntatore alla struttura dati dell'istanza associata, viene passato ai servizi "attivazione" e "lettura".

Lo schema del codice sorgente "C" per definizione di tipo della struttura dati viene generato automaticamente dal gestore di libreria ISaGRAF. Il tipo della struttura dati viene sempre chiamato "**str_data**". Il programmatore non dovrebbe cambiare questo nome, per mantenere la compatibilità con gli header dei servizi. I dati non visibili generalmente raggruppano le variabili interne con un'immagine dei parametri di ritorno. Il servizio "lettura" di un blocco funzione viene usato solamente per accedere ai programmi di ritorno e non dovrebbe essere usato per eseguire altre operazioni.

- **Il servizio di inizializzazione**

Il servizio di "inizializzazione" di un blocco funzione viene chiamato dal kernel ISaGRAF all'avvio dell'applicazione. Permette al programmatore "C" di richiedere al sistema l'allocazione della memoria per un'istanza. Subito sotto è riportato un esempio di programmazione standard per il servizio di inizializzazione:

```
uint16 FBINIT_xxx (uint16 hinstance)
/* "xxx" è il nome del f. block */
{
  return (sizeof (str_data));
}
```

L'argomento "**hinstance**" rappresenta il numero logico dell'istanza. È riservato per operazioni interne ISaGRAF e non deve essere usato nella programmazione del servizio. Il servizio di inizializzazione restituisce la quantità di memoria in byte necessaria per i dati di una istanza. La quantità di memoria richiesta (valore di ritorno) non può eccedere i **64** Kbyte. In questo servizio non dovrebbe essere eseguita nessun'altra operazione. Il codice sorgente "C" di questo servizio viene generato automaticamente del Gestore di libreria ISaGRAF alla creazione del blocco funzione.

- **Il servizio di attivazione**

Il servizio di "attivazione" viene chiamato ad ogni ciclo del target, per ogni istanza di blocco funzione usata dall'applicazione. Questo servizio elabora i parametri di chiamata ed esegue l'algoritmo principale del blocco funzione per aggiornare i dati statici non visibili ed i parametri relativi ai valori di ritorno. Subito sotto lo schema standard del servizio di attivazione:

```
void FBACT_xxx (
uint16 hinstance,       /* "xxx" è il nome del blocco funzione */
                        /* numero logico dell'istanza */
str_data *data,         /* data: puntatore alla struttura dati dell'istanza */
str_arg *arg            /* puntatore alla struttura dei parametri di chiamata */
)
{
}
```

L'argomento "**hinstance**" rappresenta il numero logico dell'istanza. È riservato per operazioni interne ISaGRAF e non deve essere usato nella programmazione del servizio. L'argomento "**data**" è un puntatore far alla struttura dati associata all'istanza. L'argomento "**arg**" è un puntatore far alla struttura contenente i valori dei parametri di chiamata. Per accedere ai

campi della struttura "**arg**", il programmatore dovrebbe usare gli identificatori definiti nell'header "C" del blocco funzione.

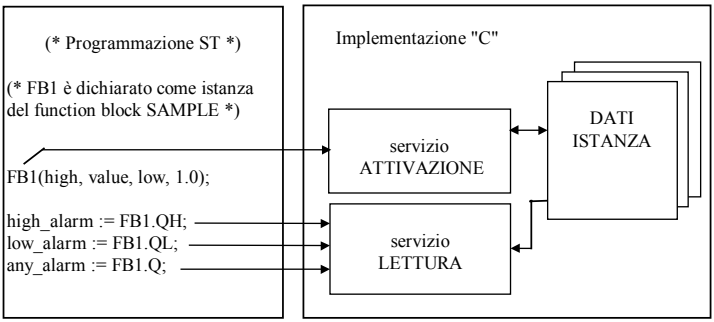
L'algoritmo di "attivazione" elabora i parametri di chiamata (memorizzati nella struttura "**arg**") ed aggiorna i campi della struttura "**data**". L'esempio seguente illustra il servizio di "attivazione" relativo al blocco funzione **TRIG** (riconoscimento di fronte di salita)

```
/* definizioni memorizzate nell'header "C" del blocco funzione */

typedef struct {                    /* parametri di chiamata */
        T_BOO  _clk;                /* trigger input */
} str_arg;

#define CLK      (arg->_clk)

/* struttura dati dell'istanza di blocco funzione */

typedef struct {
        T_BOO prev_state;           /* stato precedente dell'input del trigger */
        T_BOO edge_detect;          /* valore del fronte: immagine del parametro di
ritorno */
} str_data;

/* activation service */

void FBACT_trig (uint16 hinstance, str_data *data, str_arg *arg)
{
        data->edge_detect = (T_BOO)(CLK && !data->prev_state);
        data->prev_state = CLK;  /* parametro di chiamata */
}
```

Lo schema del codice sorgente di questo servizio viene generato automaticamente dal Gestore di libreria ISaGRAF alla creazione del blocco funzione.

- **Lettura dei parametri di ritorno**

Il servizio di "lettura" viene chiamato ogni volta che un parametro di ritorno di un blocco funzione viene referenziato in un programma ST o FBD. Serve per ricevere il valore di un parametro di ritorno. Il seguente esempio illustra le chiamate "lettura" che avvengono durante l'esecuzione di un programma ST:



Poiché il servizio di "lettura" può essere chiamato più di una volta nello stesso ciclo, per lo stesso parametro di ritorno per la stessa istanza di blocco funzione, non devono essere eseguite particolari operazioni di calcolo. Esso si occupa solamente di trasferire i dati non visibili all'applicazione ISaGRAF. Subito sotto viene riportato lo schema standard del servizio di lettura:

```
/* tipi di operazioni usate per copiare il valore del parametro di ritorno */

#define BOO_VALUE       ((T_BOO *)value)
#define ANA_VALUE       ((T_ANA *)value)
#define REAL_VALUE      ((T_REAL *)value)
#define TMR_VALUE       ((T_TMR *)value)
#define MSG_VALUE       ((T_MSG *)value)

/* servizio lettura dei parametri di ritorno: chiamato per ogni parametro di ritorno
*/

void FBREAD_xxx (            /* "xxx" è il nome del blocco funzione */
uint16 hinstance,            /* numero logico dell'istanza */
str_data *data,              /* puntatore alla struttura dati dell'istanza */
uint16 parno,                /* numero logico del parametro da leggere */
void *value)                 /* buffer in cui copiare il valore del parametro
*/
{
        switch (parno) {
            case FBLPNO_XX: /* ... */ break;
            case FBLPNO_YY: /* ... */ break;
             /* .... */
        }
}
```

L'argomento "**hinstance**" rappresenta il numero logico dell'istanza. È riservato per operazioni interne ISaGRAF e non deve essere usato nella programmazione del servizio. L'argomento "**data**" è un puntatore far alla struttura dati associata all'istanza.

L'argomento "**parno**" è il numero logico del parametro di ritorno il cui valore è richiesto. Usare gli identificatori definiti nell'header "C" del blocco funzione per indicare i parametri di ritorno. Tali identificatori iniziano con il prefisso "**FBLPNO_**". L'argomento "**value**" è un puntatore far al buffer in cui viene copiato il valore corrente del parametro di ritorno voluto. Il tipo di dato cui punta questo argomento dipende dal tipo ISaGRAF del parametro di ritorno. La seguente tabella mostra la relazione tra i tipi ISaGRAF ed i tipi di dati del buffer "C":

booleano	long	word senza segno a 32 bit (0=false / 1=true)
Analogico	long	word con segno a 32 bit
Reale	float	valore in virgola mobile a 32 bit in singola precisione
Timer	long	word senza segno a 32 bit (l'unità rappresenta 1ms)
messaggio	char *	array di caratteri

Le seguenti macro permettono di accedere al buffer di copia, secondo il tipo del parametro di ritorno voluto:

```
#define     BOO_VALUE       ((T_BOO *)value)
#define     ANA_VALUE       ((T_ANA *)value)
#define     REAL_VALUE      ((T_REAL *)value)
#define     TMR_VALUE       ((T_TMR *)value)
#define     MSG_VALUE       ((T_MSG *)value)
```

Queste sono le operazioni comunemente usate per copiare il valore del parametro nel buffer ISaGRAF:

```
/* per un parametro booleano: */
  *BOO_VALUE = valore_parametro;
/* per un parametro intero analogico: */
  *ANA_VALUE = valore_parametro;
/* per un parametro intero reale: */
  *REAL_VALUE = valore_parametro;
/* per un parametro time: */
  *TMR_VALUE = valore_parametro;
/* per un parametro stringa: */
  strcpy (*MSG_VALUE, valore_parametro);
```

Lo schema del codice sorgente di questo servizio viene generato automaticamente dal Gestore di libreria ISaGRAF alla creazione del blocco funzione.

- **Esempio di file sorgente "C"**

Viene riportato di seguito lo schema standard dell'implementazione "C" di un blocco funzione:

```
/* blocco funzione (xxx è il nome del blocco funzione) */
```

```
#include <tasy0def.h>
#include <grfb0nnn.h>   /* nnn è il numero del f.block nella libreria */

/* struttura di dati non visibili, per ogni istanza del blocco */
typedef struct {
        /* definizione dei campi */
} str_data;

/* servizio di inizializzazione: restituisce la dimensione dei dati non visibili */
word FBINIT_xxx (uint16 hinstance)
{
        return (sizeof (str_data));
}

/* servizio di attivazione: elabora i parametri di chiamata */
void FBACT_xxx (uint16 hinstance, str_data *data, str_arg *arg)
{
        /* ... */
}

/* operazioni usate per copiare il valore di un parametro di ritorno */
#define BOO_VALUE   ((T_BOO *)value)
#define ANA_VALUE   ((T_ANA *)value)
#define REAL_VALUE  ((T_REAL *)value)
#define TMR_VALUE   ((T_TMR *)value)
#define MSG_VALUE   ((T_MSG *)value)

/* servizio di lettura dei parametri di ritorno: chiamato per ogni parametro di
ritorno */
void FBREAD_xxx (uint16 hinstance, str_data *data, uint16 parno, void *value)
{
        switch(parno)
        {
          case FBLPNO_XX: *???_VALUE = ...; break;
          case FBLPNO_YY: *???_VALUE = ...; break;
        ....

}
```

/* La seguente funzione viene usata per l'inizializzazione del blocco funzione e per la dichiarazione della sua implementazione. Realizza il collegamento con il kernel ISaGRAF, usando il nome del blocco funzione. Questa funzione viene completamente generata dal Library Manager ISaGRAF. */

```
ABP fbldef_xxx (char *name, IBP *initproc, RBP *readproc)
{
        strcpy (name, "XXX");
        *initproc = (IBP)FBINIT_xxx;
        *readproc = (RBP)FBREAD_xxx;
        return ((ABP)FBACT_xxx);
}

/* fine del file */
```

Il file di include "**TASY0DEF.H**" del kernel ISaGRAF è necessario per le definizioni specifiche del sistema. Contiene anche la definizione dei tipi di dati che rappresentano i puntatori far ai servizi implementati.

Collegamenti tra progetti ed implementazione "C"

Il collegamento logico tra l'implementazione di un blocco funzione "C" e l'uso nei programmi di un progetto ISaGRAF avviene tramite il nome della funzione. Un servizio di "dichiarazione" (funzione) viene aggiunto al codice sorgente "C" del blocco funzione. Questo servizio viene chiamato una sola volta, all'avvio dell'applicazione ed indica al kernel ISaGRAF il nome del blocco funzione cui corrispondono i servizi implementati. Di seguito è riportato il formato standard di questa funzione che realizza il servizio di dichiarazione:

```
ABP fbldef_xxx (char *name, IBP *initproc, RBP *readproc)
{
        strcpy (name, "XXX");                   /* nome del f.block */
        *initproc = (IBP)FBINIT_xxx;                    /* servizio di inizializzazione
*/
        *readproc = (RBP)FBREAD_xxx;                    /* servizio di lettura */
        return ((ABP)FBACT_xxx);                        /* servizio di attivazione */
}
/* xxx è il nome del blocco funzione */
```

Il nome del blocco funzione, usato dall'istruzione **strcpy**, deve essere scritto in maiuscolo. Deve essere usato il minuscolo per i nomi dei servizi implementati e la dichiarazione del servizio.

L'uso dei prefissi "**FBACT_**", "**FBINIT_**", "**FBREAD_**" e "**fbldef_**" per i servizi implementati ed il servizio di definizione, permette di assegnare ad un blocco funzione il nome di una parola chiave riservata del linguaggio "C" oppure il nome di una funzione esistente nelle

libreria “C” ISaGRAF. Nessun altra istruzione deve essere aggiunta al servizio di dichiarazione.

Il servizio di dichiarazione viene chiamato per qualsiasi blocco funzione “C” integrato, anche se non viene usato nei programmi dall’applicazione ISaGRAF. Il kernel ISaGRAF rileverà un errore fatale nel caso un blocco funzione “C” usato nell’applicazione non sia integrato nel kernel.

Prima di collegare i nuovi blocchi funzione al kernel, è necessario scrivere un altro file sorgente “C”, chiamato " **GRFB0LIB.C**", ed inserirlo (con i relativi blocchi funzione) nella lista dei files per il linker. Il file "**GRCN0LIB.C**" contiene solamente un array di dichiarazioni di servizi. Questo array viene letto durante le operazioni di inizializzazione per creare un collegamento dinamico con i blocchi funzione scritti in “C”. Questo è un esempio di tale file:

```
/* File: grfb0lib.c - blocchi funzione implementati */

#include <tasy0def.h>

extern ABP fbldef_fb1(char *name, IBP *init, RBP *read);
extern ABP fbldef_fb2(char *name, IBP *init, RBP *read);

FBL_LIST FBLDEF[ ] = {
        fbldef_fb1,
        fbldef_fb2,

NULL };

/* fine del file */
```

L’array **FBLDEF** deve essere concluso da un puntatore NULL. Non rispettando questa condizione, possono verificarsi conflitti. Se l’array **FBLDEF** non viene definito, collegando il nuovo kernel ISaGRAF, si verificheranno situazioni di riferimenti non risolti.

Con questo file, è possibile generare un nuovo kernel che include tutti i blocchi funzione esistenti. Un kernel può anche essere realizzato specificatamente per un singolo progetto, inserendo nell’array **FBLDEF** unicamente i blocchi funzione usati nel progetto. Il file " **GRFB0LIB.C**" viene generato automaticamente dal Generatore di codice ISaGRAF, nella fase di realizzazione del codice dell’applicazione. Il file viene collocato nella directory del progetto ISaGRAF ed incorpora solamente le conversioni usate nel progetto.

Limiti

La libreria ISaGRAF può contenere fino a **255** blocchi funzione "C". Qualsiasi tipo di operazione può essere usata in una funzione. Qualsiasi tipo di blocco funzione può essere copiato (istanziato) fino a **255** volte nello stesso progetto.

è bene ricordare che i servizi di blocco funzione vengono chiamati in modo sincrono dal ciclo ISaGRAF, quindi l’esecuzione del blocco funzione influisce direttamente sul tempo di ciclo.

Un esempio completo

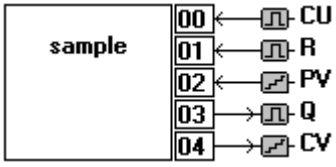
Quella che segue è la programmazione completa del blocco funzione "**sample**", contatore ad incremento.

Subito sotto la nota tecnica del blocco funzione:

nome:	SAMPLE
descrizione:	Contatore ad incremento
data creazione:	01 February 1994
autore:	CJ international
chiamata:	CU : input del contatore R : comando di reset PV : valore massimo programmato
ritorno:	Q : rilevazione max CV : risultato contatore

prototipo:	SAMPLE (count, reset_command, maximum_value);
	max_detect := SAMPLE.Q;
	count_result := SAMPLE.CV;

Subito sotto l'interfaccia del blocco funzione:



Subito sotto l'header sorgente "C" del blocco funzione:

```
/* interfaccia del blocco funzione - nome: SAMPLE */

/* definizione dei tipi di dati ISaGRAF standard */

typedef long  T_BOO;
typedef long  T_ANA;
typedef float T_REAL;
typedef long  T_TMR;
typedef char  *T_MSG;

/* definizione della struttura dei parametri di chiamata */
```

```
typedef struct {
  T_BOO  _cu;
  T_BOO  _r;
  T_ANA  _pv;
} str_arg;

/* identificatori usati per accedere ai parametri di chiamata */

#define CU                      (arg->_cu)
#define R                       (arg->_r)
#define PV                      (arg->_pv)

/* numerazione logica dei parametri di ritorno */

#define FBLPNO_Q                0
#define FBLPNO_CV               1

/* fine del file */
```

Subito sotto il codice sorgente "C" del blocco funzione. Solamente le righe in grassetto sono state inserite manualmente dal programmatore C.

.

```
/* blocco funzione - nome: SAMPLE */

#include <tasy0def.h>           /* Necessario per la definizione dei tipi */
#include <grfb0255.h>           /* Header del sorgente C della funzione */

/* definizione della struttura contenente i dati per un'istanza */

typedef struct {
        T_BOO overflow;         /* true:valore contatore >= valore program. */
        T_ANA value;            /* valore corrente del contatore */
} str_data;

/* servizio di inizializzazione: memoria dei dati dell'istanza */

word FBINIT_sample (uint16 hinstance)
{
        return (sizeof (str_data));
}
```

```
/* servizio di attivazione: algoritmo del contatore ad incremento */

void FBACT_sample(uint16 hinstance, str_data *data, str_arg *arg)
{
        if (R) data->value = 0;
        else if (CU && data->value < PV) (data->value)++;
        data->overflow = (data->value >= PV) ? (T_BOO)1 : (T_BOO)0;
}

/* operazioni per la copia dei parametri al buffer iSaGRAF */

#define BOO_VALUE   ((T_BOO *)value)
#define ANA_VALUE   ((T_ANA *)value)
#define REAL_VALUE  ((T_REAL *)value)
#define TMR_VALUE   ((T_TMR *)value)
#define MSG_VALUE   ((T_MSG *)value)

/* servizio di lettura: prende il valore di un parametro di ritorno */

void FBREAD_sample (uint16 hinstance, str_data *data, uint16 parno, void
*value)
{
        switch (parno) {
          case FBLPNO_Q   : *BOO_VALUE = data->overflow; break;
          case FBLPNO_CV : *ANA_VALUE = data->value; break;
        }
}

/* servizio di dichiarazione usato per il collegamento dinamico con il kernel
ISaGRAF */

ABP fbldef_sample (char *name, IBP *initproc, RBP *readproc)
{
        strcpy (name, "SAMPLE");
        *initproc = (IBP)FBINIT_sample;
        *readproc = (RBP)FBREAD_sample;
        return ((ABP)FBACT_sample);
}

/* fine del file */
```

C.7.5 Tecniche di compilazione e linking

L'ambiente di lavoro ISaGRAF non incorpora un compilatore o un linker "C". In questo capitolo, tuttavia, verranno esaminate le tecniche che facilitano l'uso dei file creati dal Gestore di libreria ISaGRAF con altri strumenti di programmazione quali compilatori e linker.

File sorgenti "C"

I file sorgenti "C" relativi a conversioni, funzioni e blocchi funzione vengono posti dal Gestore di libreria ISaGRAF nelle directory **ISAWIN\LIB\DEFS** e **ISAWIN\LIB\SRC.** Il nome di un file sorgente viene composto usando il corrispondente numero della libreria ISaGRAF della conversione, funzione o blocco funzione. Questi sono i nomi di file usati:

\isawin\lib\defs\TACN0DEF.H	file delle definizioni per conversioni
\isawin\lib\src\GRCN0nnn.H	sorgente di una funzione di conversione
\isawin\lib\defs\GRUS0nnn.H	file delle definizioni di una funzione
\isawin\lib\src\GRUS0nnn.C	sorgente di una funzione
\isawin\lib\defs\GRFB0nnn.H	definizioni per blocchi funzione
\isawin\lib\src\GRFB0nnn.C	sorgente per un blocco funzione

(**nnn** rappresenta il numero della conversione, funzione o blocco funzione)

Avvertenza: Il gestore di libreria ISaGRAF non aggiorna i nomi e i numeri logici contenuti nel testo e nelle righe di programma di elementi di libreria che vengano rinominati o copiati. L'aggiornamento deve essere eseguito manualmente nel file sorgente "C".

Il file **\ISAWIN\LIB\USPNUMS** contiene le relazioni tra nomi e numeri logici delle funzioni presenti nella libreria ISaGRAF. Esempio di questo file:

```
1     funct_A
10    funct_B
16    funct_C
```

Il file **\ISAWIN\LIB\FBLNUMS** contiene le relazioni tra nomi e numeri logici dei blocchi funzione presenti nella libreria ISaGRAF. Esempio di questo file:

```
0     fbl_A
1     fbl_B
2     fbl_C
```

Il file **\ISAWIN\LIB\CNVNUMS** contiene le relazioni tra nomi e numeri logici delle funzioni di conversione presenti nella libreria ISaGRAF. Questo è un esempio relativo alle conversioni della libreria standard:

```
0     SCALE
1     BCD
```

Questi file vengono automaticamente aggiornati dal Gestore di libreria ISaGRAF ogni volta che una conversione, una funzione o un blocco funzione viene creato, rinominato, copiato o cancellato. Questi file vengono generati automaticamente dal Generatore di codice ISaGRAF alla realizzazione dell'applicazione:

\isawin\apl\ppp\GRCN0LIB.C	Dichiarazione in forma di array di tutte le funzioni di conversione usate nel progetto.
\isawin\apl\ppp\GRUS0LIB.C	Dichiarazione in forma di array di tutte le funzioni usate nel progetto.
\isawin\apl\ppp\GRFB0LIB.C	Dichiarazione in forma di array di tutti i blocchi funzione usati nel progetto.

(**ppp** rappresenta il nome del progetto)

Questi file possono essere usati in fase di link per creare un nuovo kernel ISaGRAF specifico per il progetto, che contenga solamente le conversioni, le funzioni ed i blocchi funzione usati nel progetto.

Trasferimento di file sorgente ad un sistema nativo

I file sorgenti e di definizione "C" creati dal Gestore di libreria ISaGRAF possono essere scaricati al sistema target ISaGRAF, se esso supporta un programma di compilazione nativo. Per far questo, è possibile usare il programma standard **TERMINAL** fornito assieme a Windows.

Quando i file sorgente vengono gestiti sul sistema target, è necessario aggiornare i file delle definizioni con un nuovo trasferimento ogni volta che l'interfaccia di una funzione viene modificata con il Gestore di libreria ISaGRAF.

I comandi per il trasferimento possono essere raggruppati, ad esempio, in un file batch richiamabile dal menu strumenti dell'ambiente di lavoro (Si veda il Manuale utente: Gestione dei programmi)

Usare un cross compiler

I file sorgente possono anche essere gestiti da PC, se il target è un PC o se si dispone di un cross compiler, eseguibile su PC ed in grado di generare codice per il sistema target.

In questo caso, è possibile usare il Gestore di libreria ISaGRAF per completare e modificare i sorgenti di conversioni, funzioni e blocchi funzione.
I comandi per lanciare il compilatore ed il linker possono essere raggruppati, ad esempio, in un file batch richiamabile dal menu strumenti dell'ambiente di lavoro (Si veda il Manuale utente: Gestione dei programmi).

Quando conversioni, funzioni e blocchi funzione vengono compilati sul PC, è sufficiente trasferire il nuovo kernel ISaGRAF così generato (contenente i nuovi componenti) al sistema target prima di eseguire le applicazioni. Se il target è un PC, il nuovo kernel ISaGRAF può essere caricato sulla macchina target con un dischetto o tramite rete.

Linking con le librerie del kernel ISaGRAF

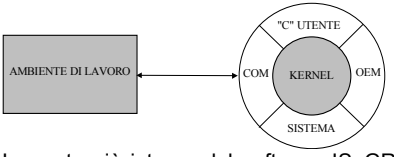
Avvertenza:
Quelle che seguono sono informazioni generali che potrebbero non corrispondere esattamente al sistema target.
Si rimanda ai file leggimi e .TXT a corredo del dischetto target.

Il dischetto target ISaGRAF contiene molti file di utilità per la compilazione ed il link di conversioni, funzioni e blocchi funzione alle librerie del kernel ISaGRAF.

Esistono due diverse implementazioni:
- ISaGRAF a processo singolo: tutte le funzioni vengono eseguite dallo stesso programma
- ISaGRAF multiprocesso: un processo (thread) a parte è riservato alle comunicazioni

In entrambi i casi, i componenti "C" vengono incorporati nella stessa libreria: il programmatore "C" non deve tener conto tra processo singolo e multiplo. Per la versione a processo singolo, le libreria utente "C" vengono collegate al processo singolo (chiamato generalmente **isa**), mentre per la versione multiprocesso, le librerie vengono collegate al processo kernel (chiamato generalmente **isaker**).

Sistema di sviluppo **Sistema target**

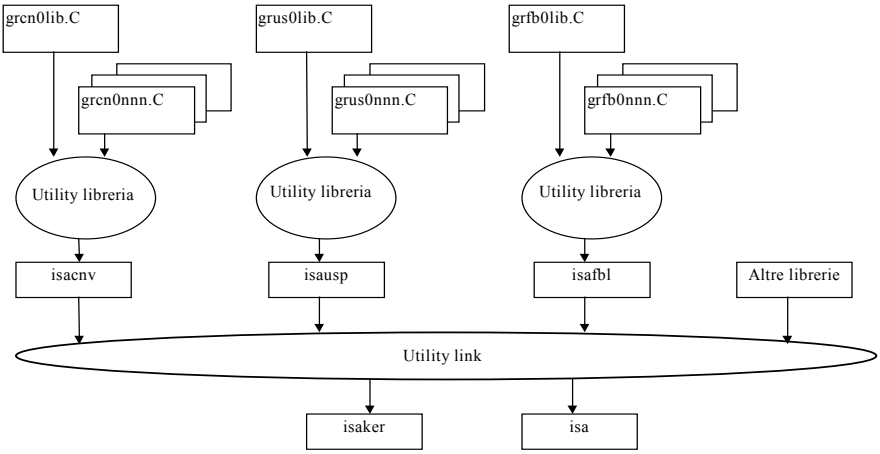


La parte più interna del software ISaGRAF è indipendente dall'hardware. Esegue linguaggi IEC e possiede un proprio data base di variabili.

Il primo passo per il link al kernel, consiste nel generare le librerie di tutte le conversioni, funzioni e blocchi funzione necessarie allo specifico progetto:

libreria	contenuto
ISAUSP	- GRUS0LIB file object (array di funzioni dichiarate) - file oggetto di ogni funzione integrata
ISAFBL	- GRFB0LIB file object (array di blocchi funzione dichiarati) - file oggetto di ogni blocco funzione integrato
ISACNV	- GRCN0LIB object file (array di conversioni dichiarate) - file oggetto di ogni conversione integrata

Successivamente, si devono collegare queste nuove librerie con i file oggetto e le librerie del kernel ISaGRAF. Nel seguente diagramma vengono delineate le diverse fasi dell'integrazione con parti sviluppate in "C" dall'utente:



Viene qui riportato l'esatto elenco dei moduli oggetto e delle librerie che devono essere unite nella fase di link:

Per produrre isaker:		
Modulo Oggetto:	**tast0mai**	
Modulo Oggetto:	**tats0com**	
Libreria Kernel:	**isaker**	
Libreria Kernel:	**isaoem**	
Libreria Utente:	**isausp**	Funzioni definite dall'utente
Libreria Utente:	**isafbl**	Blocchi funzione definiti dall'utente
Libreria Utente:	**isacnv**	Funzioni di conv. definite dall'utente
Libreria Kernel:	**isasys**	
Librerie di sistema:	(Fare riferimento al manuale del proprio compilatore "C")	

Per produrre isa:		
Modulo Oggetto:	**tast0mai**	
Modulo Oggetto:	**tast0com**	
Libreria Kernel:	**isaker**	
Libreria Kernel:	**isatst**	
Libreria Kernel:	**isaoem**	
Libreria Utente:	**isausp**	Funzioni definite dall'utente
Libreria Utente:	**isafbl**	Blocchi funzione definiti dall'utente
Libreria Utente:	**isacnv**	Funzioni di conv. definite dall'utente

Libreria Kernel:	**isasys**
Librerie di sistema:	(Fare riferimento al manuale del proprio compilatore “C”)

Potrebbe essere necessario seguire, per i moduli oggetto e le librerie, l’esatto ordinamento riportato nelle figure precedenti. I moduli oggetto e le librerie hanno le estensioni standard (**".lib"**, **".obj"**, **".l"**, **".r"**...) del sistema destinazione.

Opzioni richieste per compilazione e linking

In fase di compilazione e link verranno usate le opzioni adeguate; dipendenti dal tipo di operazioni elaborate da conversioni, funzioni e blocchi funzione. Alcune operazioni richiedono, in fase di link, ulteriori librerie di sistema (math, graphics...).

Tutti i file sorgente "C" del kernel ISaGRAF devono essere compilati con il modello di memoria **LARGE**. È necessario usare il medesimo modello per la compilazione sia di conversioni che di funzioni e blocchi funzione.

Per compilare elementi della libreria “C” , è necessario definire una costante particolare, che indica il tipo di sistema destinazione ed il processore, in modo che i sorgenti di conversioni, funzioni e blocchi funzione risultino indipendenti dal sistema. Seguono i nomi di questi valori costanti:

DOS.....sistemi basati su DOS (processore INTEL)
ISAWNT .....sistemi basati su Windows-NT (processore INTEL)
OS9 .....sistema OS9 (processore MOTOROLA)
VxWorks .....sistema VxWorks (processore MOTOROLA)

I file comandi di utilità (per la compilazione ed il link) forniti con il software target ISaGRAF, illustrano come definire l’adeguato valore costante nella riga di comando del compilatore.

Compilatori supportati

Per lo sviluppo di conversioni, funzioni, blocchi funzione ed il link al kernel ISaGRAF, vengono supportati i seguenti compilatori.

Compilatore Microsoft MSC 7.00	target basati su DOS
Compilatore Microsoft MSVC 4.00	target basati su Windows-NT
Compilatore Microware ULTRA-C	target OS-9
Tornado 1.0; GNU Toolkit 2.6	target VxWorks

Si prega di contattare la CJ International per l’uso di altri compilatori.

Sommario

Quello che segue è il sommario delle operazioni da eseguirsi per sviluppare una nuova conversione, funzione o blocco funzione.

⇨1. Usando il Gestore di Libreria ISaGRAF, creare un nuovo elemento e attribuirgli un nome ed un testo di commento. Lo schema del file sorgente “C” viene generato automaticamente.

⇨2. Usando il Gestore di Libreria ISaGRAF, definire l’interfaccia (i parametri di chiamata e di ritorno), nel caso l’elemento sia una funzione od un blocco funzione. Il file header sorgente “C” viene generato automaticamente.

⇨3. Usando il Gestore di Libreria ISaGRAF, inserire il testo dettagliato delle note tecniche (guida utente) dell’elemento.

⇨4. Usando il Gestore di Libreria ISaGRAF, completare il file sorgente “C”, inserendo la programmazione “C” dell’algoritmo della conversione, funzione o blocco funzione. Ora, il codice sorgente dell’elemento è completo. Può essere usato qualsiasi altro editor.

⇨5. Selezionare l’opzione "**Show logical number**" del Gestore di libreria per conoscere a quale numero logico sia abbinato il nuovo elemento. Questo numero viene usato nel nome dei corrispondenti file ".C" e".H".

⇨6. Copiare / trasferire i file.C e .H al target (nel caso di compilatore nativo) o all’ambiente corrispondente (nel caso di cross compiler) in cui sono state installate le librerie ed i task ISaGRAF.

⇨7. Avviare il compilatore “C” per i nuovi file sorgente e correggere gli eventuali errori di sintassi.

⇨8. Inserire il nome della dichiarazione del servizio del nuovo elemento nel file sorgente "**GR??0LIB.C**" in cui è definito l’array degli elementi inseriti.

⇨9. Avviare il compilatore "C" per compilare il file "**GR??0LIB.C**".

⇨10. Inserire il nome del modulo oggetto nell’elenco dei file oggetto usati per produrre la corrispondente libreria.

⇨11. Avviare il generatore di libreria “C”. Avviare il linker “C” per produrre il nuovo kernel.

⇨12. Installare sulla macchina destinazione il nuovo kernel.

⇨13. Scrivere un’applicazione ISaGRAF di esempio per provare l’attivazione e l’interfaccia del nuovo elemento.

C.8 Collegamento a Modbus

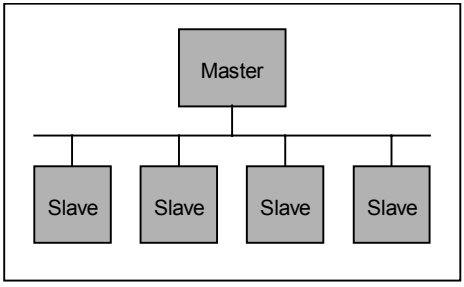
Una volta sviluppata e collaudata l'applicazione, è possibile collegarla ad un sistema di visualizzazione di processo.

ISaGRAF è un sistema aperto che offre ampie possibilità di collegamento in rete.
La rete industriale più semplice è il protocollo standard MODBUS/MODICON, che è disponibile su gran parte dei sistemi di visualizzazione di processo e necessita solamente di un collegamento seriale (RS232, RS485, Current Loop).

Il protocollo di comunicazione del debugger ISaGRAF è compatibile MODBUS e permette l'accesso in lettura/scrittura alle variabili da un master MODBUS.

C.8.1 Rete e protocollo MODBUS

Una rete Modbus è composta da una sola stazione master (normalmente un sistema di visualizzazione di processo) e da una o più stazioni slave (normalmente dei PLC)



Il master invia una richiesta per volta ad uno slave (usando il corrispondente numero slave) e rimane in attesa della risposta dallo slave prima di inviare la successiva richiesta. Gli altri slave non coinvolti non inviano risposta.

Ciascun pacchetto contiene un numero slave, un numero richiesta con i relativi dati ed un checksum a 16 bit (CRC).

Se non viene ottenuta risposta dopo un certo tempo limite (time out), la domanda può essere ripetuta un certo numero di volte prima che il master dichiari "non connesso" lo slave.
Il valore del tempo limite ed il numero di tentativi può essere regolato sulla stazione master per soddisfare i requisiti dello slave (secondo l'applicazione, ecc…).

Se si verifica un errore durante l'elaborazione della richiesta, lo slave può generare un messaggio di errore invece di inviare il pacchetto di risposta atteso.

Modbus è un protocollo Modicon, ma non è uno standard internazionale, ci sono molte diverse implementazioni di protocolli compatibili 'Modbus', con molte possibili differenze, come:

- Elenco dei codici funzione implementati
- Mappatura degli indirizzi
- protocollo RTU (codici binari) o ASCII

  ecc...

C.8.2 Implementazione ISaGRAF

Accesso alle variabili dell'applicazione

Il collegamento di comunicazione ISaGRAF riconosce cinque codici funzione Modbus:

1	lettura di N bit
3	lettura di N word
5	scrittura di 1 bit
6	scrittura di 1 word
16	scrittura di N word

È possibile accedere alle variabili di applicazioni ISaGRAF tramite il loro 'indirizzo di rete', naturalmente, se questo è stato impostato nel dizionario dell'ambiente di lavoro. Queste variabili possono essere:

- Variabili booleane o analogiche
- Variabili con attributo di ingresso, di uscita o interne
- Variabili con campo di validità locale o globale.

Per impostare un valore booleano, possono essere usate le funzioni 5, 6, o 16. Un valore TRUE in scrittura è rappresentato da un qualunque valore diverso da zero.

La lettura di un valore booleano può avvenire tramite le funzioni 1 o 3. Con la funzione 1, i valori vengono restituiti in un campo a bit, con la funzione 3, vengono restituiti in byte (il valore TRUE corrisponde a 0xFFFF).

Per impostare una variabile analogica, possono usare le funzioni 6 o 16. Il valore letto è un intero a 16 bit compreso nell'intervallo da –32768 a +32767 (le variabili target ISaGRAF sono a 32 bit).

La lettura di una variabile analogica avviene tramite la funzione 3. Il valore letto è un intero a 16 bit compreso nell'intervallo da –32768 a +32767. Dal punto di vista del target, le variabili analogiche sono a 32 bit, tuttavia, un valore, sul target, superiore al range di 16 bit (positivo o negativo) verrà letto con il massimo valore di range a 16 bit (positivo o negativo).

Non è possibile accedere alle variabili reali con una richiesta Modbus.

Avvertenza:
L'implementazione ISaGRAF non gestisce i codici di errore come 'indirizzo modbus sconosciuto'.

Notazioni:

slv	numero slave
nbw	numero di word
nbb	numero di byte
nbi	numero di bit
addH	indirizzo di rete (Byte Alto))
addL	indirizzo di rete (Byte Basso)
vH	valore (Byte Alto)
vL	valore (Byte Basso)
V	Valore in Byte
bfd	Campo a bit (nbb Byte)
crcH	checksum (Byte Alto)
crcL	checksum (Byte Basso)

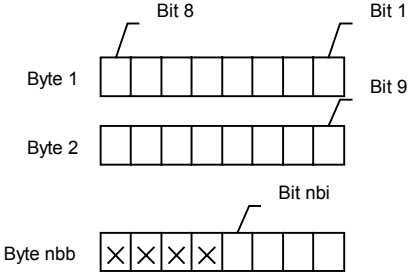
FUNZIONE 1: lettura di N bits

Lettura di nbi bit (Booleani) a partire dall'indirizzo di rete addH/addL

Richiesta	slv	01	addH	addL	00	nbi	crcH	crcL

Answer	slv	01	nbb	bfd	...		crcH	crcL
				Byte 1		Byte nbb		

bfd è un campo a bit di nbb Byte con il seguente formato:



Il Bit 1 corrisponde al valore della variabile contenuta all'indirizzo di rete addH/addL.
Il Bit nbi corrisponde al valore della variabile contenuta all'indirizzo di rete addH/addL + nbi -1
X denota un valore non definito.

FUNZIONE 3: lettura di N word

Lettura di nbw word a partire dall'indirizzo di rete addH/addL

Richiesta	slv	03	addH	addL	00	nbw	crcH	crcL

Risposta	slv	03	nbb	vH	vL	...	crcH	crcL

nbb corrisponde al numero vH, vL di byte

FUNZIONE 5: scrittura di 1 bit

Scrittura di un bit (Booleano) all'indirizzo di rete addH/addL

Richiesta	slv	05	addH	addL	vH	00	crcH	crcL

Risposta	slv	05	addH	addL	vH	00	crcH	crcL

FUNZIONE 6: scrittura di 1 word

Scrittura di una word all'indirizzo di rete addH/addL

Richiesta	slv	06	addH	addL	vH	vL	crcH	crcL

Risposta	slv	06	addH	addL	vH	vL	crcH	crcL

FUNZIONE 16: scrittura di N words

Scrittura di nbw word a partire dall'indirizzo di rete addH/addL (nbb = 2nbw)

Richiesta	slv	10	addH	addL	00	nbw	nbb	vH	vL	...	crcH	crcL

Risposta	slv	10	addH	addL	00	nbw	crcH	crcL

Esempi:

- FUNZIONE 1: lettura di 15 bit a partire dall'indirizzo di rete 0x1020, sullo slave 1

Richiesta	01	01	10	20	00	0F	79	04

Risposta	01	01	02	00	12	39	F1

Il valore letto è 0x0012, corrispondente al campo a bit 00000000 00010010.
In questo esempio, le variabili 0x1029 e 0x102C valgono TRUE, le rimanenti valgono FALSE.

- FUNZIONE 16: scrittura di 2 word all'indirizzo 0x2100 sullo slave 1, i valori scritti sono 0x1234 e 0x5678.

Richiesta	01	10	21	00	00	02	04	12	34	56	78	1C	CA

Risposta	01	10	21	00	00	02	4B	F4

Trasferimento file

A confronto dei moderni bus di campo, il protocollo Modbus offre dei servizi modesti se non è esteso da codici funzione specifici dei produttori.

Nel nostro caso, eseguendo ISaGRAF con una piattaforma di computer potente e flessibile, esistono due restrizioni imposte dal protocollo Modbus:
- è possibile accedere solamente alle variabili ISaGRAF
- risulta difficoltoso il trasferimento veloce di grosse quantità di dati

Due sono le ragioni per cui ISaGRAF offre un insieme di richieste di trasferimento file del tipo Modbus o un protocollo di "gestione remota file". Queste caratteristiche sono state implementate per permettere:
- lo scaricamento di file binari o ASCII
- il caricamento di file binari o ASCII
- lo scambio dinamico di dati tra file virtuali o fisici condivisi

Così, tramite il collegamento ISaGRAF, qualsiasi applicazione "indipendente da ISaGRAF" può facilmente comunicare con un target remoto.

Il protocollo si basa sui seguenti concetti:
- Il file dal punto di vista del target ISaGRAF viene chiamato **file remoto**
- Il file sul computer master viene chiamato **file locale**
- Ad ogni byte di un file, si ha accesso con un **indirizzo di base** a 32 bit unito ad un **indirizzo di byte** a 16 bit.

Esistono richieste per selezionare il nome del file remoto, selezionare l'indirizzo di base, leggere o scrivere dati sul file remoto usando indirizzi byte a 16 bit.

FUNZIONE 17: scrittura dati

nbb corrisponde al numero vH, vL di byte

Richiesta	slv	11	addH	addL	00	nbb	nbb	vH	vL	...	crcH	crcL

Risposta	slv	11	addH	addL	00	nbb	crcH	crcL

Il significato di questa richiesta varia a seconda del range dell'indirizzo addH/addL:

- **0xF000: Inizializzazione del nome del file remoto**
  nbb rappresenta il numero di caratteri del nome del file, specificato nei campi vH vL (in questo caso Alto e Basso non sono significativi) **incluso \0** come fine stringa.
  Se il file non esiste, viene creato con gli attributi di scrittura + lettura + esecuzione.

- **0xF002: Modifica dell'indirizzo base al valore specificato**
  nbb deve essere uguale a 4.Il primo byte vH/vL corrisponde alla word Alta del valore specificato. È possibile qualunque valore a 32 bit.
  Tutti le successive richieste useranno questo indirizzo base. Quando non viene usata questa richiesta, il valore predefinito dell'indirizzo base è zero.

- **0xF004: Cancella file**
  nbb deve essere uguale a zero.
  Il file verrà cancellato se esiste ed è possibile farlo.

- **Superiore a 0xF004: Riservato**

- **Inferiore a 0xF000: Scrittura byte**
  L'indirizzo in cui scrivere i byte viene specificato da addH/addL. Deve essere inferiore a F000. I byte specificati (gli nbb byte sono specificati nei campi vH vL in cui Alto e Basso potrebbero non essere significativi) vengono scritti nell'ordine dato (da sinistra verso destra) nel file remoto precedentemente indicato. L'indirizzo di scrittura di partenza è quello specificato aggiunto al precedentemente selezionato indirizzo base. Il file viene esteso nel caso gli indirizzi risultanti ne eccedano la dimensione. Non è possibile ridurre la dimensione del file.

FUNZIONE 18: lettura dati

Richiesta	slv	12	addH	addL	00	nbb	crcH	crcL

Risposta	slv	12	nbb	V	V	...	crcH	crcL

L'indirizzo da cui leggere i byte è specificato in addH/addL. Deve essere inferiore a F000. Viene letto lo specifico numero di byte, dal precedentemente indicato file remoto, a partire dall'indirizzo specificato (addH/addL con qualsiasi valore a 16 bit) aggiunto al precedentemente selezionato indirizzo base.
I valori vengono restituiti (V campi da sinistra verso destra) nell'ordine in cui vengono letti dal file.

Esempio:

Selezione del nome di file remoto: 'target.fil'.

Richiesta	01	11	F0	00	00	0B	0B	74	...	00	25	9F

Risposta	01	11	F0	00	00	0B	8F	0E

Selezione dell'indirizzo base: 0x10000.

Richiesta	01	11	F0	02	00	04	04	00	01	00	00	76	11

Risposta	01	11	F0	02	00	04	6E	CA

Scrittura di 4 byte: indirizzo assoluto 0x107D0, valori 01,02,03,04.

Richiesta	01	11	07	D0	00	04	04	01	02	03	04	28	6F

Risposta	01	11	07	D0	00	04	FC	87

Lettura di 4 byte: indirizzo assoluto 0x107D0.

Richiesta	01	12	07	D0	00	04	B8	87

Risposta	01	12	04	01	02	03	04	58	7D

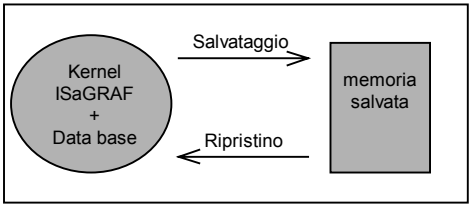
C.9 Gestione delle interruzioni di alimentazione

C.9.1 Concetti di base

La gestione delle interruzioni di alimentazione risulta molto critica in un'applicazione, per tre ragioni:
- dipende dalle specifiche del processo
- dipende dalle peculiarità hardware
- dipende dai metodi di programmazione

Per questo, la soluzione ISaGRAF al problema della gestione delle interruzioni di alimentazione, non rappresenta un metodo assoluto ed universale, ma una serie di regole, metodi e strumenti che devono essere combinati assieme in uno specifico modo per ogni diversa applicazione o, perlomeno, hardware

Tre sono i problemi da risolvere per consentire ad un sistema di controllo di processi di ripartire correttamente dopo un'interruzione di alimentazione:



- Effettuare un salvataggio dei dati
- Rilevare alla partenza che si è verificato un'interruzione di alimentazione
- Ripristinare i dati salvati

Non è possibile trovare una soluzione software standard al secondo problema, ma il fornitore del sistema può provvedere gli strumenti necessari per accedere allo stato dell'hardware dall'applicazione ISaGRAF o da un programma C.

Inoltre, è importante decidere quali dati debbano essere salvati e recuperati. Definiamo 2 tipi di dati:
- Variabili dell'applicazione:
  Variabili del processo come numero di elementi elaborati, data dell'interruzione di alimentazione, valori dei parametri dell'applicazione, ecc. …
  Variabili del programma come contatori, timer, valori temporanei e flag.

- Stato del programma:
  Riferimenti agli step attivi, stato di ogni programma C, ecc. ...

Questi due casi ed il tipo di risposta che può dare ISaGRAF vengono esaminati nei seguenti paragrafi.

C.9.2 Salvataggio delle variabili dell'applicazione

Variabili ritentive

L'editor delle variabili dell'ambiente di lavoro permette di assegnare ad ogni variabile interna (variabili non di I/O) l'attributo **Ritentiva**.

Al termine di ogni ciclo del target, i valori delle variabili ritentive vengono copiati nella locazione di memoria specificata. Questa locazione di memoria è generalmente una RAM alimentata da batteria tampone.

All'avvio, se almeno una variabile possiede l'attributo **Ritentiva**, ISaGRAF controlla le variabili ritentive:

- se la stessa applicazione è stata precedentemente eseguita, ISaGRAF recupera i valori memorizzati e li assegna per tutte le variabili con attributo **Ritentiva**.
- se la precedente applicazione eseguita era diversa dall'attuale o se l'applicazione non è mai stata eseguita precedentemente, ISaGRAF rileva che gli attributi **Ritentiva** non sono validi e imposta tutte le variabili con attributo **Ritentiva** a null.

L'area di memoria usata per memorizzare i diversi tipi di variabile viene specificata dall'ambiente di lavoro, dal menu **Compila** / **Opzioni esecuzione applicazione: Variabili ritentive.**
La stringa specificata deve avere il seguente formato:

boo_add , boo_size , ana_add , ana_size , tmr_add , tmr_size , msg_add , msg_size

dove:

boo_add: Indirizzi esadecimali usati per memorizzare variabili booleane. Devono essere sempre diversi da zero.

boo_size: Dimensione, in formato esadecimale, in byte, disponibile a questo indirizzo. È necessario un byte per ogni variabile booleana da memorizzare.

ana_add: Indirizzi esadecimali usati per memorizzare variabili analogiche. Devono essere sempre diversi da zero.

ana_size: Dimensione, in formato esadecimale, in byte, disponibile a questo indirizzo. È richiesto sempre un minimo di quattro byte da aggiungere a quattro byte per ogni variabile analogica da memorizzare.

tmr_add: Indirizzi esadecimali usati per memorizzare variabili timer. Devono essere sempre diversi da zero.

tmr_size: Dimensione, in formato esadecimale, in byte, disponibile a questo indirizzo. Sono necessari cinque byte per ogni variabile timer da memorizzare.

msg_add: Indirizzi esadecimali usati per memorizzare variabili messaggio. Devono essere sempre diversi da zero.

msg_size: Dimensione, in formato esadecimale, in byte, disponibile a questo indirizzo. Sono necessari 256 byte per ogni messaggio da memorizzare.

Requisiti:

- è necessario indicare tutti i campi per tutti i tipi anche se non si intende effettuare il salvataggio di tutti i tipi di variabili. In questo caso, specificare una dimensione zero (eccetto che per gli analogici, per i quali la dimensione deve essere almeno di quattro byte) e qualsiasi indirizzo diverso da zero, per i tipi di variabile non voluti.

Esempio:

Supponiamo di voler effettuare il salvataggio di:

20 Variabili booleane
0 Variabili analogiche
0 Variabili timer
3 Variabili messaggio

La memoria alimentata si trova all'indirizzo esadecimale 0xA2F200.

Supponiamo che:

I valori booleani vengano salvati a partire dall'indirizzo 0xA2F200 con l'esatta dimensione richiesta di 20 byte.

Lo spazio minimo di 4 byte richiesto dai valori analogici venga memorizzato all'indirizzo 0xA2F214.

Gli indirizzi fittizi dei timer saranno 0xA2F200, specificati con dimensione zero.

I messaggi verranno memorizzati all'indirizzo 0x A2F218 con l'esatta dimensione richiesta di 256*3 byte.

La stringa da inserire dall'ambiente di lavoro dovrà essere:

```
A2F200,14,A2F214,4,A2F200,0,A2F218,300
```

Chiamate alla funzione SYSTEM

Nel caso debba essere salvata la maggior parte delle variabili dell'applicazione, allora è comodo usare la funzione SYSTEM che permette di operare su un insieme completo di variabili (per ulteriori informazioni sulla funzione SYSTEM, si veda il Manuale utente). Si noti che, in questo caso, il salvataggio ed il ripristino sono gestiti dal programmatore a livello dell'applicazione

Per prima cosa è necessario definire la locazione della memoria di salvataggio per uno specifico tipo di variabile o per tutti i tipi di variabili:

<nuovo_indirizzo> := SYSTEM(SYS_INITxxx,<indirizzo>);

dove:

- <indirizzo> rappresenta l'indirizzo della locazione della memoria di salvataggio (16# valore in formato esadecimale). Deve essere un indirizzo pari, altrimenti l'operazione fallisce.
- SYS_INITxxx puo essere:
  - SYS_INITBOO indica la locazione di memoria di salvataggio per le variabili booleane.
  - SYS_INITANA indica la locazione di memoria di salvataggio per le variabili analogiche.
  - SYS_INITTMR indica la locazione di memoria di salvataggio per le variabili timer.
  - SYS_INITALL indica la locazione di memoria di salvataggio per le variabili booleane, analogiche e timer.
- <nuovo_indirizzo> riporta il seguente indirizzo libero, ovvero <indirizzo> + la dimensione (in byte) delle variabili salvate, in accordo con SYS_INIYxxx. In questo modo è possibile

controllare la dimensione richiesta della memoria di salvataggio. Se l'operazione fallisce, <nuovo_indirizzo> vale zero.

Successivamente, è possibile richiedere il salvataggio. Questa procedura può essere chiamata dall'applicazione in qualsiasi momento, il salvataggio viene effettuato al **termine** del ciclo corrente e solamente una volta. Se l'hardware è in grado di avvertire, tramite un input booleano o una funzione C, dell'avvenuta interruzione di alimentazione e permette l'esecuzione di almeno un ciclo prima del crollo, è possibile effettuare il salvataggio solo alla rilevazione dell'interruzione di alimentazione.
:

<errore> :=SYSTEM(SYS_SAVxxx,0);

dove:
- SYS_SAVxxx può essere:
  - SYS_SAVBOO per richiedere il salvataggio di tutte le variabili booleane.
  - SYS_SAVANA per richiedere il salvataggio di tutte le variabili analogiche.
  - SYS_SAVTMR per richiedere il salvataggio di tutte le variabili timer.
  - SYS_SAVALL per richiedere il salvataggio di tutte le variabili booleane, analogiche e timer.
- <errore> riporta uno stato di errore diverso da 0 nel caso l'operazione sia fallita (SYS_INITxxx non sia stato chiamato).

Infine, si deve poter ripristinare le variabili. Questa procedura può essere chiamata dall'applicazione in ogni momento, il ripristino avverrà una volta solamente e al **termine** del ciclo corrente. Per assicurarsi della correttezza dei dati salvati, è bene usare una variabile analogica con assegnato un valore costante, con funzione di contrassegno:

<errore> := SYSTEM(SYS_RESTxxx,0);

dove:
- SYS_RESTxxx può essere:
  - SYS_RESTBOO per ripristinare tutte le variabili booleane.
  - SYS_RESTANA per ripristinare tutte le variabili analogiche.
  - SYS_RESTTMR per ripristinare tutte le variabili timer.
  - SYS_RESTALL per ripristinare tutte le variabili booleane, analogiche e timer.
- <errore> riporta uno stato di errore diverso da 0 nel caso l'operazione sia fallita (SYS_INITxxx non sia stato chiamato).

Quello che segue è il sommario dei comandi della funzione SYSTEM per la gestione delle variabili di salvataggio:

comando		Significato
parola chiave pre-definita	**Valore**	
SYS_INITBOO	16#20	inizializza il salvataggio di valori booleani
SYS_SAVBOO	16#21	salvataggio valori booleani
SYS_RESTBOO	16#22	ripristino valori booleani
SYS_INITANA	16#24	inizializza il salvataggio di valori analogici
SYS_SAVANA	16#25	salvataggio valori analogici

SYS_RESTANA	16#26	rispristino valori analogici
SYS_INITTMR	16#28	inizializza il salvataggio di valori timer
SYS_SAVTMR	16#29	salvataggio valori timer
SYS_RESTTMR	16#2A	rispristino valori timer
SYS_INITALL	16#2C	inizializza il salvataggio di tutti i tipi
SYS_SAVALL	16#2D	salvataggio di tutti i tipi
SYS_RESTALL	16#2E	ripristino di tutti i tipi

comando (parola chiave predefinita)	**Argomento**	**Valore di ritorno**
SYS_INITxxx	indirizzo memoria	successivo indirizzo libero
SYS_SAVxxx	0	zero se OK
SYS_RESTxxx	0	zero se OK

Implementazione personalizzata

Infine, usando funzioni o blocchi funzione C, è possibile sviluppare procedure specifiche per avere accesso alla memoria alimentata da batteria tampone, per salvare e ripristinare le variabili dall'applicazione in qualunque momento.

Esempi:

1) Procedure dedicate ad un'applicazione:
backup, restore_temp, restore_date, restore_cpt procedure C definite dall'utente.

backup(temperature, date, cnt); salva 3 valori critici

temperature := **restore_temp()**; ripristina temperature
date := **restore_date()**; ripristina date
cnt := **restore_cnt()**; ripristina counter

2) Procedure generalizzate:
backup_init, backup, backup_link, restore procedure C definite dall'utente.

save_id := **backup_init**(address, size); alloca un array di memoria alimentata.
backup(save_id, cpt1, 3); salva cpt1 come terzo elemento.

rest_id := **backup_link**(address, size) collegamento alla memoria alimentata.
cpt1 := **restore**(rest_id, 3); ripristina il valore salvato di cpt1.

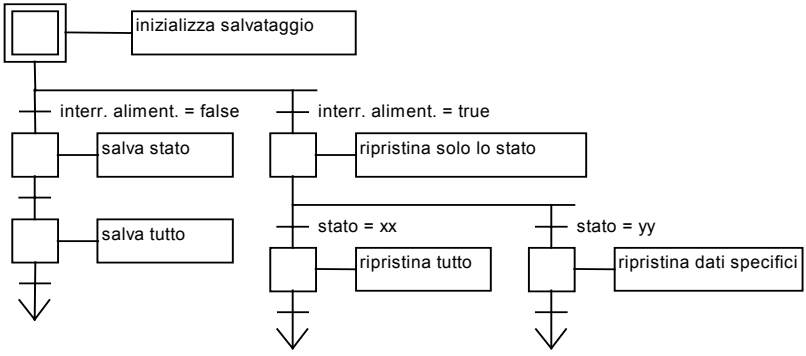
C.9.3 Salvataggio dello stato del programma

è possibile salvare qualsiasi stato di qualsiasi applicazione, ma potrebbe rivelarsi pericoloso ripristinare qualunque programma allo stato in cui si trovava al momento dell'ultimo salvataggio, per almeno 3 ragioni:

- Alcuni processi necessitano di operazioni specifiche prima di essere avviati
- Trattare tutti gli stati di un'applicazione completa è noioso
- Alcune risorse esterne come ad esempio programmi C, periferiche, ecc. non possono essere riavviati automaticamente

La miglior soluzione è probabilmente quella di effettuare il salvataggio di variabili analogiche o booleane che descrivano lo stato del processo, nel momento in cui il programmatore ritiene che nelle fasi di riavvio potranno essere usate queste informazioni. In questo modo è possibile, a partire da "un'immagine" del processo incompleta, ma razionale, avviare, terminare o sospendere programmi SFC ed inizializzare variabili in modo da impostare l'applicazione in uno stato opportuno. Non è, comunque, possibile per ISaGRAF, provvedere procedure di riavvio automatico.

Esempio:



C.10 Appendice: Elenco di errori e relativa descrizione

Elenco di errori:

Codice	Messaggio	Tipo
1	impossibile allocare memoria per il data base in esecuzione	sistema
2	data base applicazione non corretto (Motorola/Intel)	applicazione
3	impossibile allocare spazio per la mailbox	sistema
4	impossibile collegarsi al data base del kernel	sistema
5	time out inviando richiesta al kernel	sistema
6	time out attendendo risposta dal kernel	sistema
7	impossibile inizializzare la comunicazione	sistema
8	impossibile allocare memoria per le variabili ritentive	applicazione
9	applicazione fermata	applicazione
10	troppe azioni N o P simultanee	applicazione
11	troppe azioni di tipo set simultanee	applicazione
12	troppe azioni di tipo reset simultanee	applicazione
13	istruzione TIC non riconosciuta	applicazione
16	impossibile rispondere a richiesta di lettura dati	sistema
17	impossibile rispondere a richiesta di scrittura dati	sistema
18	impossibile rispondere a richiesta di una sessione di debugger	sistema
19	impossibile rispondere a richiesta modbus	sistema
20	impossibile rispondere alla richiesta del debugger	sistema
21	impossibile rispondere al debugger	sistema
23	codice di richiesta non riconosciuto	sistema
24	errore di comunicazione ethernet	sistema
25	errore di sincronizzazione di comunicazione	sistema
28	impossibile allocare memoria per l'applicazione	sistema
29	impossibile allocare memoria per l'aggiornamento dell'applicazione	sistema
30	codice chiave OEM non riconosciuto	applicazione
31	impossibile inizializzare scheda di ingresso di tipo booleano	applicazione
32	impossibile inizializzare scheda di ingresso di tipo analogico	applicazione
33	impossibile inizializzare scheda di ingresso di tipo messaggio	applicazione
34	impossibile inizializzare scheda di uscita di tipo booleano	applicazione
35	impossibile inizializzare scheda di uscita di tipo analogico	applicazione
36	impossibile inizializzare scheda di uscita di tipo messaggio	applicazione
37	impossibile ottenere ingresso da scheda di tipo booleano	applicazione
38	impossibile ottenere ingresso da scheda di tipo analogico	applicazione
39	impossibile ottenere ingresso da scheda di tipo messaggio	applicazione
40	impossibile attuare variabile di uscita di tipo booleano	applicazione
41	impossibile attuare variabile di uscita di tipo analogico	applicazione
42	impossibile attuare variabile di uscita di tipo messaggio	applicazione
43	impossibile operare su variabile di tipo booleano	applicazione
44	impossibile operare su variabile di tipo analogico	applicazione
45	impossibile operare su variabile di tipo messaggio	applicazione

46	impossibile effettuare apertura scheda	applicazione
47	impossibile effettuare chiusura scheda	applicazione
50	impossibile sovrascrivere variabile di uscita di tipo booleano	programma
51	impossibile sovrascrivere variabile di uscita di tipo analogico	programma
52	impossibile sovrascrivere variabile di uscita di tipo messaggio	programma
61	codice richiesta di sistema non riconosciuto	programma
62	overflow periodo di campionamento	programma
63	funzione utente non implementata	applicazione
64	intero diviso per zero	programma
65	funzione di conversione non implementata	applicazione
66	blocco funzione non implementato	applicazione
67	funzione standard non implementata	applicazione
68	reale diviso per zero	programma
69	parametri non validi per operate	applicazione
72	impossibile modificare i simboli dell'applicazione	applicazione
73	aggiornamento impossibile: insieme diverso di variabili di tipo booleano	applicazione
74	aggiornamento impossibile: insieme diverso di variabili di tipo analogico	applicazione
75	aggiornamento impossibile: insieme diverso di variabili di tipo timer	applicazione
76	aggiornamento impossibile: insieme diverso di variabili di tipo messaggio	applicazione
77	aggiornamento impossibile: nuova applicazione non trovata	applicazione
> 100	codice di errore specifico OEM, chiedere ulteriori dettagli al fornitore	

I 3 tipi di errore corrispondono alle diverse provenienze del problema:

– **Errori di sistema:**
Questi problemi sono dovuti probabilmente al software o hardware target, non ad impostazioni dell'applicazione o all'esecuzione del programma.
Provare con il reset a freddo (spegnimento) del target e provare a mandare in esecuzione altre applicazioni.
Questi errori devono essere riportati al supporto ISaGRAF.

– Errori dell'applicazione:
Questi problemi sono dovuti a parametri, dimensione o contenuto dell'applicazione
Questi errori dovrebbero scomparire caricando un'applicazione conosciuta e precedentemente collaudata. Se il problema permane, va incluso negli errori di sistema precedentemente descritti.

– Errori di programma:
Questi problemi sono dovuti a particolari sequenze del programma.
Questo tipo di errore dovrebbe scomparire avviando l'applicazione in modalità ciclo per ciclo o fermando il programma critico.

Descrizione degli errori:

1. impossibile allocare memoria per data base in esecuzione	***sistema***

Memoria non disponibile. Controllare l'hardware.

2. data base applicazione non corretto (Motorola/Intel)	***applicazione***

Il file applicazione, scaricato o salvato non è corretto. Questo errore avviene quando una procedura è stata generata per INTEL e scaricata per MOTOROLA (o viceversa) o se il file è alterato.

3. impossibile allocare spazio per la mailbox	***sistema***

Questo errore è prodotto dal processo di comunicazione se non riesce ad allocare spazio di tipo 3 per le comunicazioni tra processi.

4. impossibile collegarsi al data base del kernel	***sistema***

Il processo di comunicazione genera questo errore quando al numero slave specificato nella propria riga di comando non corrisponde alcun kernel in esecuzione.

5. time out inviando richiesta al kernel	***sistema***

Il processo di comunicazione non può inviare una richiesta al kernel. Probabilmente il kernel è impegnato o non è in esecuzione.

6. time out attendendo risposta dal kernel	***sistema***

Il processo di comunicazione non riceve risposta dal kernel. Probabilmente il kernel è impegnato o non è in esecuzione.

7. impossibile inizializzare la comunicazione	***sistema***

Questo avvertimento viene dato quando lo strato di comunicazione non riesce ad inizializzare la connessione fisica. Questo avvertimento viene anche dato se non è stato specificato un percorso di comunicazione. Questo non impedisce al target di venire eseguito correttamente, senza, però, la possibilità di comunicare

8. impossibile allocare memoria per le variabili ritentive	***applicazione***

ISaGRAF non è in grado di gestire le variabili ritentive. Due possono essere le cause di questo problema:

- la stringa passata come parametro al target host non è sintatticamente corretta
- la dimensione della memoria specificata per ogni blocco non è sufficiente

bisogna verificare la sintassi dell'attributo di variabile ritentiva e tentare di ridurre il numero di variabili ritentive.

9. applicazione fermata	***applicazione***

Questo avvertimento viene generato ogni volta che l'applicazione viene fermata dal debugger.

10. troppe azioni N o P simultanee	***applicazione***

Questo errore si verifica quando uno dei cicli target si trova a dover eseguire troppe azioni non-memorizzate, impulsive o blocchi ciclici. è possibile localizzare il problema in modalità CC.
Il massimo numero di azioni simultanee è 2 + 4 per programma SFC.

11. troppe azioni di tipo set simultanee	***applicazione***

Questo errore si verifica quando uno dei cicli target si trova a dover eseguire troppe azioni di tipo set (eseguite all'attivazione del passo). Si proceda come sopra.

12. troppe azioni di tipo reset simultanee	***applicazione***

Questo errore si verifica quando uno dei cicli target si trova a dover eseguire troppe azioni di tipo reset (eseguite alla disattivazione del passo). Si proceda come sopra.

13. istruzione TIC non riconosciuta	***applicazione***

Il kernel ha rilevato qualcosa di errato nel codice dell'applicazione (chiamato Codice Target Indipendente), in un programma. Due sono le possibili spiegazioni:

- un programma esterno sta probabilmente scrivendo nel codice dell'applicazione. Tentare di localizzare il crollo in modalità CC ed assicurarsi che nessuna interfaccia IO abbia parametri errati.
- il target che si sta usando ha un insieme ridotto di istruzioni e l'applicazione fa uso di istruzioni o tipi di variabile non autorizzati.

16. impossibile rispondere a richiesta di lettura dati	***sistema***

È stato rilevato un errore di comunicazione nella risposta allo specifico codice funzione di richiesta 18 (lettura file) del Modbus ISaGRAF. Controllare le connessioni e la configurazione di sistema sia dal lato target che master.

17. impossibile rispondere a richiesta di scrittura dati	***sistema***

È stato rilevato un errore di comunicazione nella risposta allo specifico codice funzione di richiesta 17 (scrittura file) del Modbus ISaGRAF. Controllare le connessioni e la configurazione di sistema sia dal lato target che master.

18. impossibile rispondere a richiesta di una sessione di debugger	***sistema***

È stato rilevato un errore di comunicazione nella risposta ad una richiesta del debugger. Controllare le connessioni e la configurazione di sistema sia dal lato target che master.

19. impossibile rispondere a richiesta modbus	***sistema***

È stato rilevato un errore di comunicazione nella risposta ad una richiesta Modbus. Controllare le connessioni e la configurazione di sistema sia dal lato target che master.

20. impossibile rispondere alla richiesta del debugger	***sistema***

È stato rilevato un errore di comunicazione nella risposta ad una richiesta del debugger. Controllare le connessioni e la configurazione di sistema sia dal lato target che master.

21. impossibile rispondere al debugger	***sistema***

È stato rilevato un errore di comunicazione nella risposta ad una richiesta del debugger. Controllare le connessioni e la configurazione di sistema sia dal lato target che master.

23. codice di richiesta non riconosciuto	***sistema***

Una richiesta del debugger è priva di senso.

24. errore di comunicazione ethernet	***sistema***

Si verifica ogni volta che viene chiusa la connessione con il debugger chiuso: il sistema funziona correttamente. Altrimenti indica il rilevamento di un errore di comunicazione ethernet. Controllare le connessioni e la configurazione di sistema sia dal lato target che master.
Viene dato un secondo campo che può essere:
1: errore in invio o ricezione
2: errore nella creazione del socket
3: errore in fase di binding o ricezione dal socket
4: errore in fase di accettazione di un nuovo client

25. errore di sincronizzazione di comunicazione	***sistema***

Errata sincronizzazione tra il processo di comunicazione del target ed il master.
Controllare le connessioni e la configurazione di sistema (parametri di comunicazione) sia dal lato target che master.

28. impossibile allocare memoria per l'applicazione	***sistema***

Memoria non disponibile. Controllare l'hardware in relazione con la dimensione dell'applicazione.

29. impossibile allocare memoria per l'aggiornamento dell'applicazione	***sistema***

Memoria non disponibile. Controllare l'hardware in relazione con la dimensione dell'applicazione.

30. codice chiave OEM non riconosciuto	***applicazione***

L'applicazione fa uso di una scheda il cui codice del costruttore (OEM) non viene riconosciuto dal target.
Controllare le connessioni di I/O nell'ambiente di lavoro ed usare l'attributo "VIRTUAL" per localizzare la scheda non corretta. La libreria dell'ambiente di lavoro potrebbe non corrispondere alla versione del target.

31. impossibile inizializzare scheda di ingresso di tipo booleano	***applicazione***

L'inizializzazione di una scheda di ingresso di tipo booleano è fallita. Controllare le connessioni di I/O nell'ambiente di lavoro ed i parametri delle schede di ingresso di tipo booleano.

32. impossibile inizializzare scheda di ingresso di tipo analogico	***applicazione***

L'inizializzazione di una scheda di ingresso di tipo analogico è fallita. Controllare le connessioni di I/O nell'ambiente di lavoro ed i parametri delle schede di ingresso di tipo analogico.

33. impossibile inizializzare scheda di ingresso di tipo messaggio	***applicazione***

L'inizializzazione di una scheda di ingresso di tipo messaggio è fallita. Controllare le connessioni di I/O nell'ambiente di lavoro ed i parametri delle schede di ingresso di tipo messaggio.

34. impossibile inizializzare scheda di uscita di tipo booleano	***applicazione***

L'inizializzazione di una scheda di uscita di tipo booleano è fallita. Controllare le connessioni I/O nell'ambiente di lavoro ed i parametri delle schede di uscita di tipo booleano.

35. impossibile inizializzare scheda di uscita di tipo analogico	***applicazione***

L'inizializzazione di una scheda di uscita di tipo analogico è fallita. Controllare le connessioni di I/O nell'ambiente di lavoro ed i parametri delle schede di uscita di tipo analogico.

36. impossibile inizializzare scheda di uscita di tipo messaggio	***applicazione***

L'inizializzazione di una scheda di uscita di tipo messaggio è fallita. Controllare le connessioni di I/O nell'ambiente di lavoro ed i parametri delle schede di uscita di tipo messaggio.

37. impossibile ottenere ingresso da scheda di tipo booleano	***applicazione***

È stato rilevato un errore nell'aggiornamento di una scheda di ingresso di tipo booleano. Controllare le connessioni di I/O nell'ambiente di lavoro ed i parametri della scheda.

38. impossibile ottenere ingresso da scheda di tipo analogico	***applicazione***

È stato rilevato un errore nell'aggiornamento di una scheda di ingresso di tipo analogico. Controllare le connessioni di I/O nell'ambiente di lavoro ed i parametri della scheda.

39. impossibile ottenere ingresso da scheda di tipo messaggio	***applicazione***

È stato rilevato un errore nell'aggiornamento di una scheda di ingresso di tipo messaggio. Controllare le connessioni di I/O nell'ambiente di lavoro ed i parametri della scheda.

40. impossibile attuare variabile di uscita di tipo booleano	***applicazione***

È stato rilevato un errore nell'aggiornamento di una variabile booleana di uscita. Controllare le connessioni di I/O nell'ambiente di lavoro ed i parametri della scheda.

41. impossibile attuare variabile di uscita di tipo analogico	***applicazione***

È stato rilevato un errore nell'aggiornamento di una variabile analogica di uscita. Controllare le connessioni di I/O nell'ambiente di lavoro ed i parametri della scheda.

42. impossibile attuare variabile di uscita di tipo messaggio	***applicazione***

È stato rilevato un errore nell'aggiornamento di una variabile di tipo messaggio di uscita. Controllare le connessioni di I/O nell'ambiente di lavoro ed i parametri della scheda.

43. impossibile operare su variabile di tipo booleano	***applicazione***

È stato rilevato un errore nell'esecuzione di una chiamata OPERATE per una variabile booleana. Verificare i parametri dell'istruzione OPERATE e le note utente della scheda.

44. impossibile operare su variabile di tipo analogico	***applicazione***

È stato rilevato un errore nell'esecuzione di una chiamata OPERATE per una variabile analogica. Verificare i parametri dell'istruzione OPERATE e le note utente della scheda.

45. impossibile operare su variabile di tipo messaggio	***applicazione***

È stato rilevato un errore nell'esecuzione di una chiamata OPERATE per una variabile di tipo messaggio. Verificare i parametri dell'istruzione OPERATE e le note utente della scheda.

46. impossibile effettuare apertura scheda	***applicazione***

L'applicazione usa un riferimento a scheda non riconosciuto dal target. Controllare le connessioni di I/O nell'ambiente di lavoro. La libreria dell'ambiente di lavoro potrebbe non corrispondere alla versione del target.

47. impossibile effettuare chiusura scheda	***applicazione***

L'applicazione usa un riferimento a scheda non riconosciuto dal target. Controllare le connessioni di I/O nell'ambiente di lavoro.

50. impossibile sovrascrivere variabile di uscita di tipo booleano	***programma***

Due sequenze SFC tentano di scrivere la stessa variabile di uscita di tipo booleano durante lo stesso ciclo del target. Questa situazione è da evitare per prevenire comportamenti rischiosi degli I/O. Nel caso di un simile conflitto, la priorità viene data al programma più in alto nella gerarchia. Se i due programmi SFC sono di pari livello, il risultato è imprevedibile.

51. impossibile sovrascrivere variabile di uscita di tipo analogico	***programma***

Due programmi SFC tentano di scrivere la stessa variabile di uscita di tipo analogico durante lo stesso ciclo del target. Vedere il commento sopra.

52. impossibile sovrascrivere variabile di uscita di tipo messaggio	***programma***

Due programmi SFC tentano di scrivere la stessa variabile di uscita di tipo messaggio durante lo stesso ciclo del target. Vedere il commento sopra.

61. codice richiesta di sistema non riconosciuto	***programma***

Un programma usa una chiamata SYSTEM con un codice non valido.

62. overflow periodo di campionamento	***programma***

La durata del periodo del ciclo del target è superiore a quella specificata nel menu dell'ambiente di lavoro.
Per un sistema multiprocesso, questo significa che non c'è abbastanza tempo CPU per l'esecuzione di un ciclo, anche se "la durata del ciclo corrente" è inferiore al periodo specificato.
Per un sistema a processo singolo, questo significa sempre che ci sono troppe operazioni in un ciclo del target.

Esistono molti modi possibili per ovviare a questo errore:
- ridurre il numero delle operazioni elaborate al momento della rilevazione dell'errore.
- ridurre il numero di token o transizioni valide o ottimizzare le elaborazioni complesse, ecc.
- ridurre il carico della CPU dato da altri processi per lasciare più tempo ad ISaGRAF.
- ridurre il traffico delle comunicazioni per lasciare più tempo ad ISaGRAF.
- usare la modifica dinamica della durata del ciclo per adattare la durata del ciclo alle diverse fasi del processo.
- impostare la durata del ciclo a zero per permettere al kernel ISaGRAF di girare quanto più velocemente possibile, senza il controllo di overflow.

63. funzione utente non implementata	***applicazione***

Un programma usa una funzione C sconosciuta al target. La libreria dell'ambiente di lavoro potrebbe non corrispondere alla versione del target.

64. intero diviso per zero	***programma***

Un programma tenta la divisione per zero di un intero analogico. L'applicazione deve prevenire questa situazione che potrebbe portare ad effetti imprevedibili.
Nel caso questo accada, ISaGRAF assegna come risultato il massimo valore analogico.
Nel caso l'operando sia negativo, il risultato viene invertito.

65. funzione di conversione non implementata	***applicazione***

Un programma usa una funzione di conversione C non riconosciuta dal target. La libreria dell'ambiente di lavoro potrebbe non corrispondere alla versione del target.
Nel caso si verifichi questa situazione, ISaGRAF non converte il valore.

66. blocco funzione non implementato	***applicazione***

Un programma usa un blocco funzione C non riconosciuto dal target. La libreria dell'ambiente di lavoro potrebbe non corrispondere alla versione del target.

67. funzione standard non implementata	***applicazione***

Un programma usa un blocco funzione C non riconosciuto dal target. sebbene si supponga essere disponibile sulla maggior parte dei target. Rivolgersi al fornitore.

68. reale diviso per zero	***programma***

Un programma tenta la divisione per zero di un reale analogico. L'applicazione deve prevenire questa situazione che potrebbe portare ad effetti imprevedibili.
Nel caso questo accada, ISaGRAF assegna come risultato il massimo valore reale analogico.
Nel caso l'operando sia negativo, il risultato viene invertito.

69. parametri non validi per operate	***applicazione***

l'applicazione usa una chiamata OPERATE con parametri errati. Questo caso viene normalmente rilevato dal compilatore. Potrebbe trattarsi di un parametro timer o di una variabili non di ingresso o di uscita.

72. impossibile modificare i simboli dell'applicazione	***applicazione***

Nel tentativo di aggiornare un'applicazione, l'applicazione modificata non può essere lanciata in quanto i simboli risultano differenti. Una o più variabili o istanze di blocco funzione possono essere stati aggiunti o modificati, rispetto all'applicazione corrente.

73. aggiornamento impossibile: insieme diverso di variabili di tipo booleano	***applicazione***

L'applicazione modificata non può essere lanciata in quanto una o più variabili di tipo booleano possono essere state aggiunte o rimosse, rispetto all'applicazione corrente.

74. aggiornamento impossibile: insieme diverso di variabili di tipo analogico	***applicazione***

L'applicazione modificata non può essere lanciata in quanto una o più variabili di tipo analogico possono essere state aggiunte o rimosse, rispetto all'applicazione corrente.

75. aggiornamento impossibile: insieme diverso di variabili di tipo timer	***applicazione***

L'applicazione modificata non può essere lanciata in quanto una o più variabili di tipo timer possono essere state aggiunte o rimosse, rispetto all'applicazione corrente.

76. aggiornamento impossibile: insieme diverso di variabili di tipo messaggio	***applicazione***

L'applicazione modificata non può essere lanciata in quanto una o più variabili di tipo messaggio possono essere state aggiunte o rimosse, rispetto all'applicazione corrente.

77. aggiornamento impossibile: nuova applicazione non trovata	***applicazione***

Non è presente in memoria l'applicazione modificata: può essersi verificato qualche errore in fase di scaricamento.

D. GLOSSARIO

Analogico Tipo di variabili. Si tratta di variabili intere o reali continue.

Attivazione di una transizione Operazione in fase di esecuzione: vengono rimossi tutti i token presenti nei passi precedenti. Viene creato un token per ciascuno dei passi seguenti.

Attività di un passo Attributo di un passo marcato da un token SFC. Le azioni associate ad un passo vengono eseguite in base all'attività dello stesso.

Attributo Classe di variabili. Gli attributi disponibili per le variabili sono quelli di variabile: interna, ingresso, uscita, costante.

Azione Elenco di istruzioni o assegnazioni eseguite quando è attivo un passo o un programma SFC.

Azione di tipo Impulsivo Azione contenuta in un passo SFC contenente un elenco di istruzioni eseguite solo una volta all'attivazione del corrispondente passo.

Azione Non-memorizzata Azione contenuta in un passo SFC contenente un elenco di istruzioni eseguite ad ogni ciclo del target quando il corrispondente passo è attivo.

Azioni booleane Azione contenuta in un passo SFC: in base al segnale di attività del passo viene impostata una variabile booleana.

Barra di alimentazione Barre verticali a sinistra e a destra poste alle estremità di un diagramma LD.

Barra Strumenti Piccola finestra posizionata all'interno della finestra principale di un programma di editing grafico, che incorpora i pulsanti per selezionare i componenti grafici ed altri comandi.

Bobina Componente grafica dei programmi LD, usata per rappresentare l'assegnamento di una variabile di output.

Booleano Tipo di variabile. Sono variabili che possono assumere solamente il valore TRUE (vero) o FALSE (falso).

Breakpoint Segno inserito in un passo o in transizione SFC dall'utente in fase di debug. Il sistema target si arresta quando un token SFC raggiunge un breakpoint.

Campo di visibilità o di validità. Insieme di programmi che può far uso di un oggetto. I campi di visibilità predefiniti ISaGRAF sono: Comune, Globale e Locale.

Canale di I/O Singola connessione in una scheda di I/O. Un canale di I/O può essere associato ad una variabile di I/O.

Cella Area fondamentale della matrice grafica dei linguaggi grafici come SFC, FBD or LD.

Ciclico Attributo di un programma che è sempre in esecuzione.

Ciclo target Insieme delle operazioni eseguite ogni volta che viene attivato il sistema target ISaGRAF. I cicli vengono controllati dal tempo ciclo programmabile.

Codice chiave OEM (scheda di I/O) Codice esadecimale a 16 bit per ogni scheda della libreria ISaGRAF. Il codice OEM identifica il fabbricante della scheda.

Codice sorgente C File di testo contenente il codice sorgente "C" di una funzione o di una funzione di conversione.

Commento Testo incluso nel programma che non ha alcun effetto durante l'esecuzione del programma.

Commento (SFC) Testo associato ad un passo o ad una transizione SFC. Non alcun effetto durante l'esecuzione del programma.

Comune Campo di visibilità (o di validità) riservato ai *nomi definiti* (non alle variabili). Un *nome definito* con campo di visibilità *Comune* può essere utilizzato in qualsiasi programma di qualsiasi progetto.

Condizione (per una transizione) Espressione booleana associata ad una transizione SFC. La transizione non può essere attivata quando la condizione assume valore false.

Connessione di I/O Definizione dei collegamenti tra le variabili dell'applicazione ed i canali delle schede presenti sul sistema destinazione.

Contatto Componente grafico di un programma LD. Rappresenta lo stato di una variabile di input del ramo LD.

Conversione Filtro associato ad una variabile analogica di ingresso o di uscita. La conversione (da un valore *fisico* a un valore *elettrico*) agisce ogni volta che la variabile interagisce con ingressi o uscite.

Cross reference Informazioni generate dall'ambiente di lavoro ISaGRAF relative alle variabili del dizionario ed ai punti in cui vengono usate nel progetto.

Diario File di testo contenente tutte le note relative alle modifiche apportate ad un programma. Ogni nota è completata dalla data.

Dizionario Insieme di variabili interne, di ingresso o uscita e nomi definiti usate in un programma di un progetto.

Errore in esecuzione Errore dell'applicazione rilevato in fase di esecuzione dal sistema destinazione ISaGRAF.

Espressione	Insieme di operatori ed identificatori che rappresentano un valore.
Espressione costante	Espressione testuale usata per descrivere un valore costante. Ogni espressione costante può essere solo di un *tipo* di variabile.
Etichetta (IL)	Identificatore all'inizio di una riga di istruzione IL, che identifica l'istruzione è può essere usato come operando per le istruzioni di salto.
FBD	Linguaggio grafico Functional Block Diagram.
Fronte	Variazione di una variabile booleana. Un fronte di salita indica un cambiamento dallo stato logico FALSE a TRUE. Un fronte di discesa indica una variazione da TRUE a FALSE.
Function block	Componente grafico del linguaggio FBD, che rappresenta una funzione elementare standard delle librerie ISaGRAF.
Functional Block Diagram	Linguaggio grafico: le formule sono costituite da blocchi elementari standard delle librerie ISaGRAF, collegati assieme nel diagramma.
Funzione C	Funzione scritta in linguaggio "C", chiamata dai programmi ISaGRAF (scritti in altri linguaggi), in modo sincrono. Le funzioni C sono fornite dalla CJ International o possono essere sviluppate dall'utente.
Funzione di conversione	Funzioni scritte in linguaggio "C" che descrivono una conversione. Queste conversioni possono essere associate a qualsiasi variabile analogica di ingresso o di uscita.
Gerarchia	Architettura di un progetto, suddivisa in più programmi. L'albero gerarchico rappresenta i collegamenti tra programmi padre, figli e sottoprogrammi.
Globale	Campo di visibilità (o di validità) disponibile per variabili o per nomi definiti. Gli oggetti con campo di visibilità globale possono essere utilizzati all'interno di un progetto in qualsiasi programma, ma non in altri progetti.
Header sorgente C	File di testo contenente definizioni e tipi "C" necessari alla programmazione di funzioni o funzioni di conversione C.
I/O bloccati	Variabile di ingresso o di uscita scollegata logicamente dal corrispondente dispositivo ingresso o di uscita con un comando «Blocca» innoltrato dall'utente tramite il debugger.
Identificatore	Termine univoco usato per rappresentare una variabile o un'espressione costante durante la programmazione.
IL	Linguaggio Instruction List.

Indirizzo di rete Indirizzo esadecimale facoltativo, liberamente definibile per ogni variabile. Questo indirizzo viene usato dal protocollo Modbus quando il sistema destinazione è connesso ad un sistema esterno.

Ingresso Attributo di una variabile. Tali variabili sono connesse ad un dispositivo di ingresso.

Instruction List Linguaggio testuale di basso livello, inserito in forma di liste sequenziali di operazioni elementari.

Interno Attributo di una variabile che non è collegata ad un dispositivo di ingresso o di uscita, viene utilizzata per elaborazioni interne.

Intero Tipo di variabili analogiche in formato intero a 32 bit.

Istruzione IL Operazione elementare di un programma IL, inserita su una riga di testo.

Istruzione ST Operazione base del linguaggio ST.

Ladder Diagram Linguaggio grafico che combina contatti e bobine per la rappresentazione di formule booleane.

LD Linguaggio Ladder Diagram.

Libreria Insieme di risorse hardware e software che possono essere inserite direttamente in qualsiasi applicazione.

Linguaggio C Linguaggio testuale di alto livello usato per redigere operazione computer coma funzioni C e funzioni di conversione.

Livello 1 SFC Descrizione principale di un programma SFC. La programmazione di Livello 1 comprende il diagramma (passi e transizioni) ed i commenti associati.

Livello 2 SFC Descrizione dettagliata di un programma SFC. La programmazione di Livello 2 rappresenta la descrizione delle azioni all'interno dei passi e le condizioni booleane associate alle transizioni.

Locale Campo di visibilità (o di validità) disponibile per variabili e *nomi definiti*. Gli oggetti con campo di visibilità *Locale* possono essere utilizzati solo in uno specifico programma di un progetto.

Macro passo Componente grafico SFC. Un macro passo è costituito da un gruppo unico di passi e transizioni rappresentato da un simbolo unico sul diagramma principale e descritto separatamente.

Matrice Suddivisione logica i celle rettangolari dell'area di editing del programma di un linguaggio grafico.

Messaggio Tipo di variabile. Queste variabili contengono stringhe di caratteri a lunghezza variabile.

Modalità Ciclo per ciclo	In esecuzione: in questa modalità, i cicli vengono eseguiti uno ad uno, in base alle direttive date al debugger dall'utente.
Modalità Tempo reale	Normale modalità di esecuzione in tempo reale: i cicli del target sono controllati dal tempo ciclo programmato.
Modbus	Protocollo di comunicazione Master-Slave. Un sistema destinazione ISaGRAF può essere uno Slave Modbus per il collegamento con un sistema esterno (come ad esempio un sistema supervisore) in un'architettura completa.
Modificatore (IL)	Carattere singolo posto alla fine di un'istruzione IL che modifica il significato dell'operazione..
Nome definito	Identificatore univoco usato per sostituire qualsiasi espressione in un programma.
Note tecniche	Guida utente per un elemento della libreria ISaGRAF (funzioni o function block C, funzioni di conversione o schede di I/O). La nota tecnica viene redatta dal progettista dell'elemento.
Numero di riferimento (SFC)	Numero decimale (da 1 a 65535) che identifica un passo o una transizione SFC in un diagramma SFC.
Operando (IL)	Variabile o espressione costante elaborata da un'istruzione elementare IL.
Operazione (IL)	Istruzione base del linguaggio IL. Un operazione è generalmente associata, in un'istruzione, ad un operando.
Operazione rinviata (IL)	operazione di un programma IL, eseguita quando viene incontrata l'istruzione "(" più avanti nel programma.
Parametro (funzione C)	Valore dato come input ad una funzione "C". Un parametro è caratterizzato da un Tipo.
Parametro (scheda di I/O)	Parametro definito dall'utente o costante di una scheda standard di I/O. Un parametro definito dall'utente viene inserito dal programmatore durante la fase di Connessione di I/O.
Parametro OEM (scheda I/O)	Parametro di I/O della scheda, definito dal progettista della scheda. Può essere una costante o un parametro variabile inserito dall'utente la fase di Connessione di I/O.
Parola chiave	Termine riservato del linguaggio.
Passo	Componente grafico di base del linguaggio SFC. un passo rappresenta una situazione stabile del processo e viene disegnato come un quadrato. Ogni passo è identificato con un numero logico. L'attività di un passo permette di controllare le corrispondenti azioni.

Passo finale di Macro	Ultimo passo di una macro SFC. Un passo finale di macro non ha transizioni successive.
Passo iniziale	Passo speciale di un programma SFC, attivato all'avvio del programma.
Passo Iniziale di Macro	Il primo passo nel corpo di una macro SFC. Nessuna transizione precede un passo iniziale di macro.
Progetto	Area di programmazione che raggruppa tutte le informazioni (programmi, variabili, codice destinazione…) di un'applicazione ISaGRAF.
Programma	Unità base di programmazione di un progetto. Un programma viene redatto con un unico linguaggio e viene disposto nell'architettura gerarchica del progetto.
Programma figlio SFC	Programma SFC controllato da un altro programma SFC chiamato padre.
Programma padre	Programma scritto con un qualsiasi linguaggio, che controlla (chiama) un altro programma non SFC, chiamato sottoprogramma.
Programma principale	Programma posto in cima all'albero gerarchico. I programmi principali vengono attivati automaticamente dal sistema.
Programma SFC padre	Programma SFC che controlla altri programmi SFC detti figli.
Reale	Classe di variabili di tipo analogico, in formato virgola mobile IEEE a 32 bit singola precisione.
Registro(IL)	Risultato corrente in una sequenza IL.
Risultato corrente (IL)	Risultato di un'istruzione in un programma IL. Il risultato corrente può essere modificato da un'istruzione o usato per l'assegnamento ad una variabile.
Salto a un passo	Componente grafico SFC che rappresenta un collegamento da una transizione ad uno passo. Il simbolo grafico di un salto, è costituito da una freccia, su cui è riportato il numero logico del passo destinazione.
Scheda di I/O	Risorsa hardware. una scheda di I/O è caratterizzata da un *tipo* (comune a tutte le variabili) e da un *verso* (di ingresso o di uscita). I parametri di una scheda di I/O sono riportati nella libreria ISaGRAF.
Scheda reale	Scheda di I/O fisicamente connessa ad un dispositivo di I/O della macchina target.

Scheda virtuale	Scheda di I/O che non è fisicamente connessa ad un dispositivo di I/O sulla macchina destinazione.
Separatore	Carattere speciale (o gruppo di caratteri) usato per separare gli identificatori in un linguaggio testuale. Un separatore può rappresentare un'operazione.
Sequential Function Chart	Linguaggio grafico: il processo viene rappresentato come un insieme di passi collegati tra loro da transizioni. Ai passi sono associate delle azioni. Le transizioni rappresentano condizioni booleane.
Sezione	Gruppo di programmi all'interno di un progetto eseguiti con le stesse regole dinamiche.
Sezione Finale	Gruppo di programmi ciclici eseguiti al termine di ogni ciclo del PLC target.
Sezione Iniziale	Gruppo di programmi ciclici eseguiti all'inizio di ogni ciclo del PLC target.
Sezione Sequenziale	Gruppo di programmi eseguiti durante ogni ciclo del PLC target. L'esecuzione di questi programmi rispetta le regole dinamiche del linguaggio SFC.
SFC	Linguaggio Sequential Function Chart.
Situazione iniziale	Impostazione dei passi iniziali di un programma SFC che rappresenta il contesto iniziale del programma all'avvio.
Sottoprogramma	Programma scritto in qualsiasi linguaggio, eccetto SFC ed FC, chiamato da un altro programma detto programma chiamante.
ST	Linguaggio Structured Text.
Stringa	Insieme di caratteri memorizzato in una variabile di tipo messaggio.
Structured Text	Linguaggio strutturato testuale di alto livello, che combina assegnazioni e strutture di alto livello come If/Then/Else e chiamate a funzioni.
Tabella di conversione	Insieme di punti che definiscono una conversione lineare (per segmenti). Queste conversioni possono essere associate a qualsiasi variabile analogica di ingresso o di uscita.
Target	Macchina destinazione ISaGRAF che supporta il software kernel ISaGRAF.
Tempo di ciclo	Durata di un ciclo di esecuzione sul PLC target.
Timer	Tipo di variabile. Queste variabili contengono valori temporali e vengono aggiornate automaticamente in esecuzione dal sistema ISaGRAF.

Tipo	Classe di variabili con lo stesso formato. I tipi base sono: booleano, analogico, timer e messaggio.
Token (SFC)	Segnale grafico usato per mostrare i passi attivi di un programma SFC.
Transizione	Componente grafico SFC di base. Una transizione rappresenta una condizione posta tra due diversi passi SFC. Ad ogni transizione è associato un numero di riferimento. Ogni transizione rappresenta una condizione booleana.
Uscita	Attributo di una variabile. Tali variabili sono collegate ad un dispositivo di uscita.
Validità di una transizione	Attributo di una transizione. Una transizione è valida quando sono attivi i passi che la precedono.
Valore di ritorno di un sottoprogramma	Valore restituito da un sottoprogramma al termine della sua esecuzione. Il valore di ritorno viene usato nelle operazioni del programma chiamante.
Variabile	Rappresentazione univoca di un dato elementare elaborato dai programmi del progetto.
Variabile I/O	Variabile connessa ad un dispositivo di ingresso o di uscita. Una variabile di I/O deve essere connessa al canale di una scheda di I/O.

E. Indice analitico

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z