1

Introduzione a LabVIEW

M. Bertocco
Università di Padova
Facoltà di Ingegneria

Introduzione

In questa sezione si descrivono le caratteristiche fondamentali dell'ambiente di sviluppo LabVIEW. Dapprima si daranno alcune notizie generali sul suo ambito di utilizzo e sulla sua interfaccia utente, successivamente si descriveranno gli elementi fondamentali per la programmazione e le principali strutture dati. Infine si illustreranno più in dettaglio le funzioni fondamentali di libreria messe a disposizione.

1 Cos'è Labview ?

LabVIEW (Laboratory Virtual Instrument Engineering Workbench) è un ambiente di sviluppo per applicazioni principalmente orientate :

- all'acquisizione di dati e alla gestione di strumentazione elettronica
- all'analisi ed elaborazione dei segnali

LabVIEW fornisce un ambiente di programmazione di tipo grafico ad oggetti denominato "G language", il quale consente di realizzare programmi in forma di diagrammi a blocchi.

LabVIEW conserva comunque molte similitudini con gli ambienti di programmazione tradizionali: presenta tutti i tipi di dati e gli operatori predefiniti di uso comune, permette di generare nuovi tipi di dati combinando tra loro i tipi di dati elementari e di controllare l'esecuzione dei programmi ricorrendo a strutture di controllo di flusso come ad esempio cicli e costrutti per l'esecuzione condizionale di codice.

Mette inoltre a disposizione del programmatore una serie di librerie di funzioni che possono essere richiamate ed utilizzate all'interno dei programmi: le librerie comprendono funzioni di uso comune (funzioni aritmetiche e statistiche, funzioni la manipolazione di stringhe, …) ed inoltre funzioni specializzate per l'acquisizione e l'elaborazione dei segnali, il controllo di strumentazione numerica via interfaccia IEEE-488 o VXI., la trasmissione di dati mediante l'uso di porte seriali oppure mediante il protocollo di comunicazione TCP/IP. E' possibile inoltre definire nuove funzioni ed arricchire le librerie in dotazione a LabVIEW.

Infine il programma consente di effettuare il debug delle applicazioni create in linguaggio G attraverso opportune modalità di esecuzione dei programmi, come ad esempio il modo "highlight execution" o "single step" e per mezzo di oggetti che consentono in run-time la modifica di variabili di programma.

LabVIEW presenta alcuni vantaggi rispetto ad un linguaggio di programmazione tradizionale:

- E` di facile apprendimento, in quanto presenta una modalità di programmazione a blocchi, di tipo visuale ed intuitivo.
- Permette di dare al codice una struttura modulare che consente di suddividere programmi complessi in sottoprogrammi più semplici che possono essere riutilizzati.

- Consente di raccogliere i VI in librerie, ovvero in un insieme di sub-VI uilizzabili da altri VI e velocemente inseribili nel codice sorgente dal programmatore .
- Fornisce un considerevole insieme di librerie per lo sviluppo di applicativi, tra le quali si trovano funzioni di tipo matematico e statistico, controllo di dispositivi per mezzo di alcuni tipi di interfaccia, comunicazione tra calcolatori, etc...

1.1 Utilizzo principale di LabVIEW

LabVIEW è stato pensato principalmente per il controllo di schede di espansione connesse direttamente al bus di un calcolatore o di strumentazione connessa al calcolatore stesso attraverso opportune interfacce come il bus IEEE 488, RS-232, strumenti VXI o ancora attraverso Internet mediante il protocollo TCP/IP.

L'ambiente di sviluppo consente di costruire programmi i quali prendono il nome di **strumenti virtuali** (Virtual Instrument, **VI**).

Un Virtual Instrument permette l'interazione tra calcolatore e strumentazione fornendo contemporaneamente all'utente un opportuno pannello frontale grafico per il dialogo con il VI stesso. In questo modo l'utente interagisce con un nuovo dispositivo (Instrument), costituito da calcolatore, interfacce, strumenti e programma il quale presenta una realtà (Virtual) diversa dai singoli oggetti fisici che compongono il sistema stesso. Tale fatto spiega il nome di Virtual Instrument dato ad un programma LabVIEW.

Ad esempio, si supponga di dover rilevare il modulo della risposta in frequenza di un circuito elettronico lineare avendo a disposizione un oscilloscopio ed un generatore di forme d'onda: una possibile soluzione consiste nel fornire all'ingresso dell'amplificatore una sinusoide con frequenza opportuna e di rilevare il rapporto tra l'ampiezza della sinusoide di uscita e l'ampiezza della sinusoide di ingresso allo stesso, al quale corrisponde il valore del modulo della risposta in frequenza per il valore scelto della frequenza della sinusoide di test.

Evidentemente tale procedura può risultare onerosa in termini di tempo e richiedere una certa familiarità dell'operatore con la strumentazione.

Mediante l'uso di LabVIEW è possibile costruire uno "strumento virtuale" che effettui automaticamente la misura . Tale strumento si presenta all'operatore come se fosse un nuovo strumento "misuratore della risposta in frequenza", il quale in realtà non esiste ma è dato dall'unione di un oscilloscopio, un generatore di funzioni, un calcolatore provvisto di interfaccia ed un programma. Un operatore interagisce cioè con uno strumento virtuale.

Una possibile soluzione è rappresentata nelle seguenti figure 1 e 2.

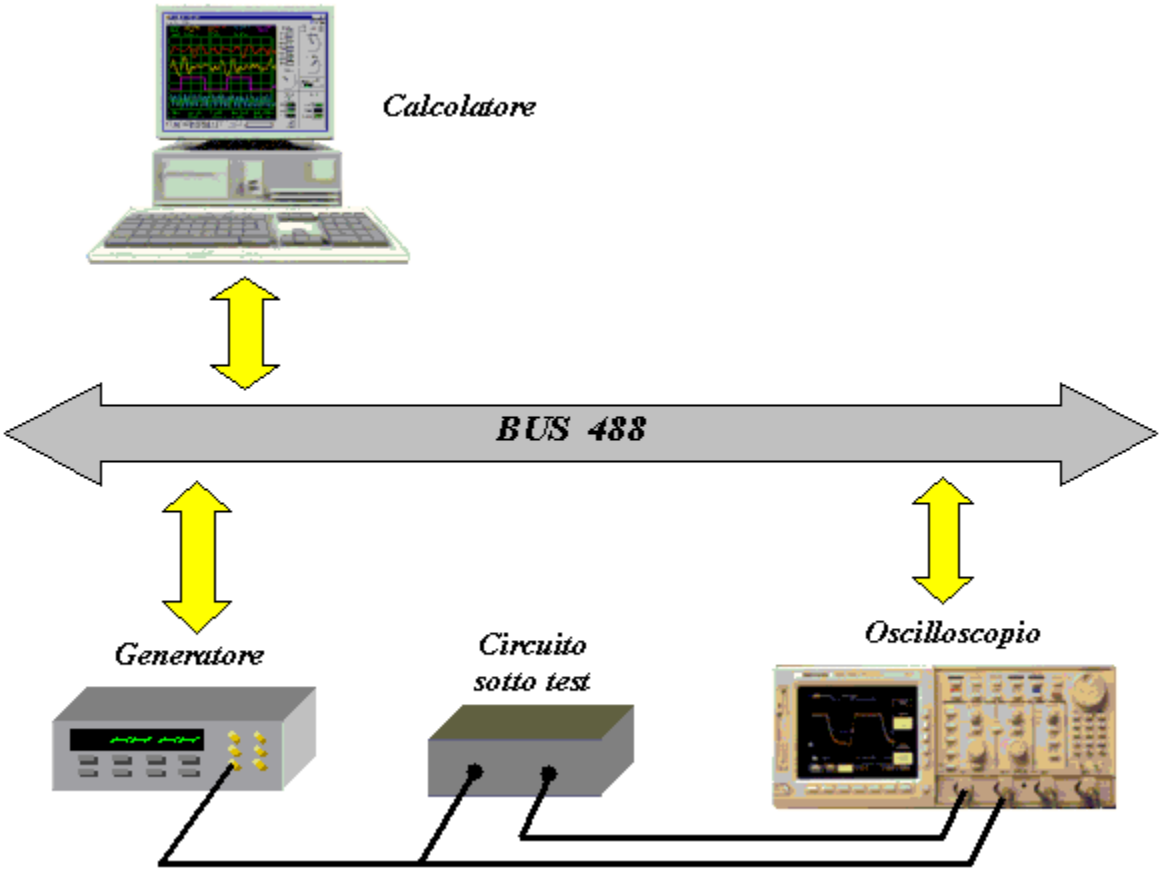


Fig. 1. Allestimento di una stazione di test per la valutazione del modulo della risposta in frequenza di un circuito lineare.

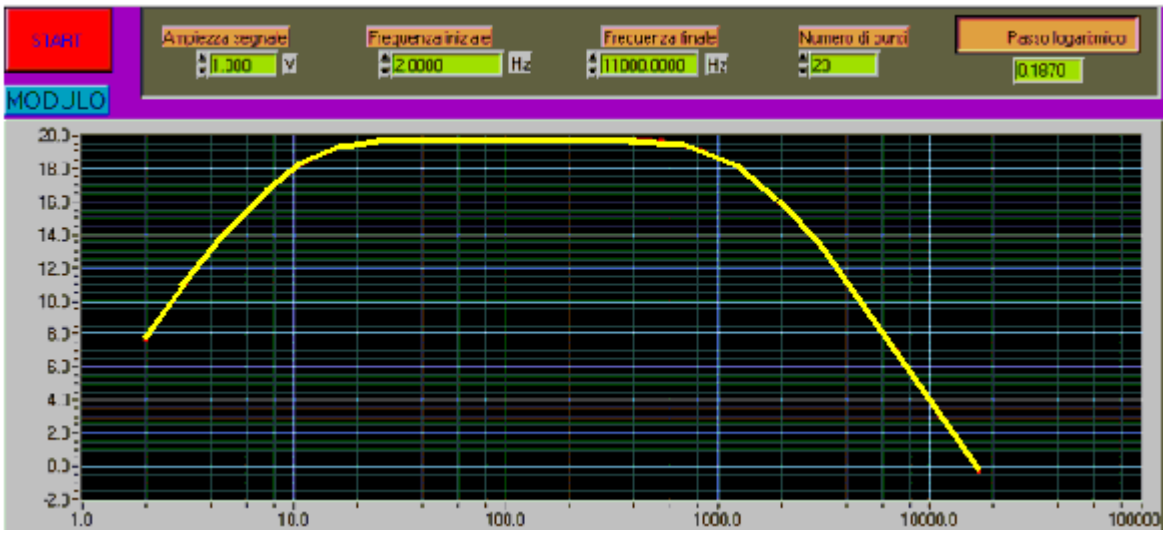


Fig. 2. Pannello frontale del programma per la misura della risposta in frequenza.

Interagendo con il pannello frontale l'utente può impostare i parametri della misura, quali la frequenza iniziale e quella finale, il numero di punti e l'ampiezza del segnale di uscita dal generatore.

Una volta mandato in esecuzione il programma LabVIEW invia agli strumenti i comandi che permettono di effettuare le misure ed acquisisce i dati di misura corrispondenti: al termine i risultati vengono visualizzati su di un grafico mostrato in Fig. 2 (modulo della risposta in frequenza).

Risultano a questo punto evidenti i vantaggi che offre un Virtual Instrument rispetto ad un banco di misura reale:

- Interazione indiretta con più strumenti attraverso mouse, tastiera, touch screen, …
- Pannello frontale dipendente dal programma.
- Possibilità di elaborazione dei dati o di coordinare le azioni di più strumenti.

Passiamo ora a descrivere più in dettaglio che cosa sia un VI.

2 Virtual Instrument (VI)

Si ricorda che i programmi che si possono realizzare utilizzando il linguaggio grafico LabVIEW sono chiamati **Virtual Instrument (VI)**, dove il termine "strumenti" è dovuto al fatto che durante l'esecuzione i programmi sviluppati presentano agli utilizzatori una interfaccia analoga a quella di uno strumento di misura, mentre il termine "virtuali" è dovuto al fatto che l'interazione avviene con un programma in esecuzione e non con un dispositivo fisico dedicato.

L'utilizzatore può modificare il valore di alcune grandezze agendo su opportune manopole o interruttori visualizzati dal programma e può osservare il risultato delle elaborazioni condotte internamente al VI su display grafici molto simili a quelli che si trovano sulla strumentazione numerica.

Un VI è composto da tre parti fondamentali:

- Pannello frontale (Front Panel)
- Diagramma a blocchi funzionale (Block diagram)
- Icona/connettore (Icon/connector)

Il **Front Panel** (pannello frontale) è la finestra che rappresenta l'interfaccia tra il programma e l'utilizzatore. Nel pannello frontale trovano posto tutti i **controllori** e gli **indicatori** dello strumento virtuale: per *controllore* si intende una variabile di ingresso che può essere modificata agendo sul pannello frontale, per *indicatore* si intende una variabile di uscita il cui valore può essere modificato dal programma e non dall'utente.

Il **Block Diagram** (diagramma a blocchi funzionale) contiene il codice nella forma di diagramma a blocchi ed è costituito da :

- *Nodi*: sono degli elementi di elaborazione

- *Collegamenti*: uniscono i nodi e permettono lo scambio di informazioni. Le informazioni passano da un nodo all'altro del pannello frontale per mezzo dei connettori che uniscono i nodi stessi.

La coppia **Icon / connector** (icona/connettore) è il terzo elemento fondamentale di un programma LabVIEW.

L'icona è un simbolo grafico di piccole dimensioni che rappresenta simbolicamente il VI stesso e che permette di trasformare il programma in un oggetto.

Il connettore stabilisce la corrispondenza tra aree dell'icona e controllori / indicatori del pannello frontale.

A titolo di esempio si consideri un semplice VI che consenta di sommare 2 numeri, A e B, e scriva il risultato in un terzo numero, C.

I componenti fondamentali di tale VI sono rappresentati in Fig. 3.

.

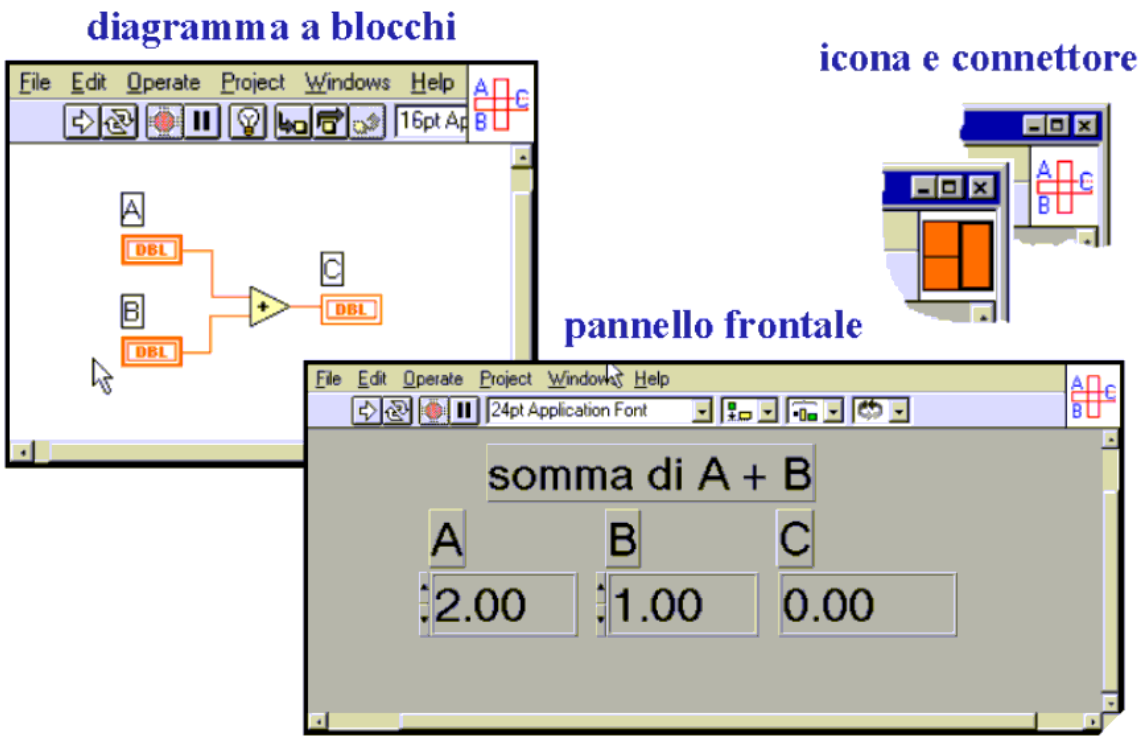


Fig. 3. Componenti di un Virtual Instrument (VI).

Con riferimento alla Fig. 3 sono due controllori le variabili A e B mentre il risultato della somma che viene visualizzato in C è un indicatore.

Si noti come opportuni simboli grafici presenti nel diagramma a blocchi fissino la corrispondenza tra elementi di ingresso/uscita del pannello frontale e nodi del diagramma a blocchi ad essi associati.

Si noti come tale corrispondenza viene fissata automaticamente da LabVIEW al momento della creazione del VI, come verrà meglio illustrato in seguito.

Passiamo ora ad analizzare più in dettaglio i 3 elementi che compongono un VI.

2.1 Front Panel

Il **Front Panel** (pannello frontale) rappresenta l'interfaccia utente di un VI: il nome deriva dal fatto che può essere strutturato in modo tale da ricordare il pannello frontale di uno strumento dotato di display, indicatori, manopole, tasti, eccetera.

È possibile interagire con un Front Panel mediante tastiera o mouse, introducendo valori numerici o stringhe di caratteri, modificare lo stato di elementi grafici, come ad esempio manopole, bottoni e così via.

Ad esempio la Fig. 4 riporta un semplice pannello frontale di un VI che consente di generare alcuni numeri casuali compresi tra due estremi. Il VI riceve in ingresso il numero di valori casuali che devono essere generati e i corrispondenti valori estremi. Il VI inoltre fornisce un grafico nel quale sono rappresentati i valori generati in funzione di un indice (0,1,2,...) che rappresenta il numero d'ordine di estrazione assieme alla media e deviazione standard campionarie associata ai valori generati.

Il pannello frontale che LabVIEW presenta all'utente è il seguente di Fig. 4.

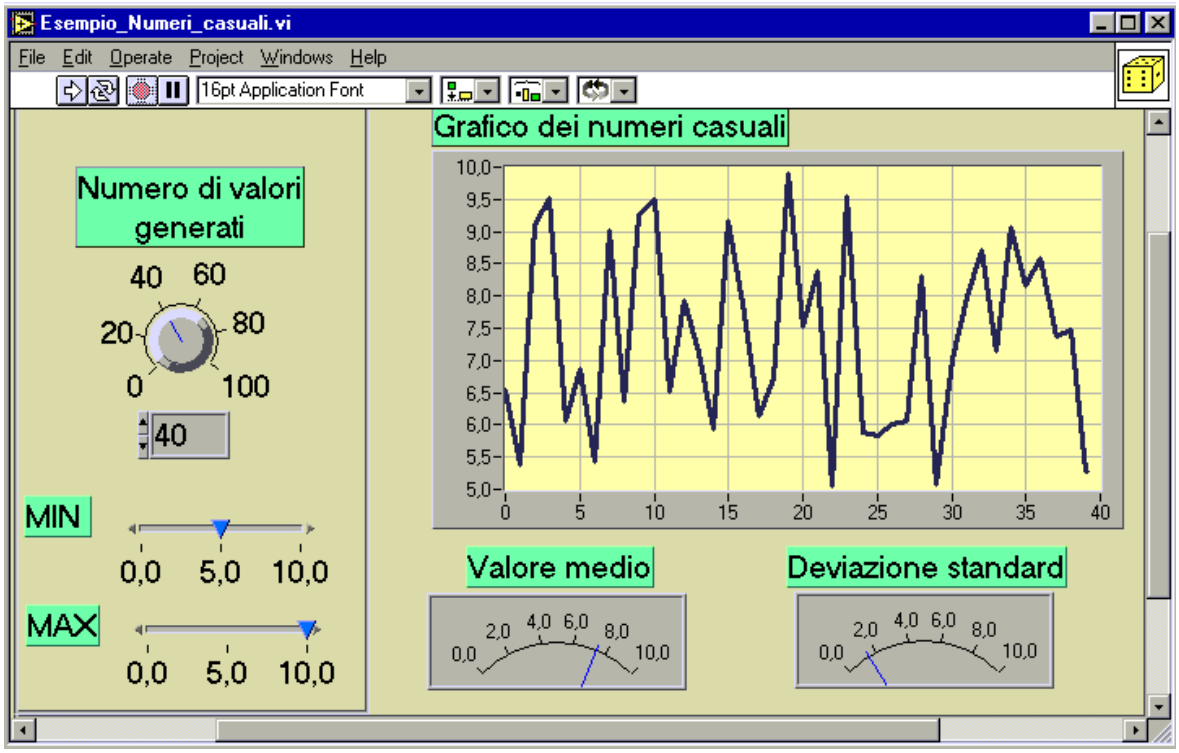


Fig. 4. Pannello frontale di un VI che genera una sequenza di numeri casuali.

Con riferimento alla Fig. 4 gli elementi denominati "Numero di valori generati" , "MIN", "MAX" sono dei controllori (controls), mentre gli elementi denominati "Grafico dei numeri casuali", "Valore medio", "Deviazione standard" sono degli indicatori (indicators).

Modificando i valori dei controllori, il VI genera una quantità di numeri casuali pari al valore impostato in "Numero di valori generati" e di ampiezza compresa tra "MIN" e "MAX". Una volta mandato in esecuzione, il VI genera la sequenza di numeri casuali la quale viene visualizzata sul grafico, mentre gli indicatori "Valore medio" e "Deviazione standard" vengono conseguentemente aggiornati.

2.2 Block Diagram

Il **Block Diagram** (diagramma a blocchi) contiene il codice sorgente, in linguaggio G, di un VI.

Pur presentandosi in forma grafica diversa, il diagramma a blocchi presenta possibilità di programmazione analoghe a quelle offerte da un comune linguaggio di programmazione del tipo text-based.

Si ricorda che il block diagram contiene due tipi di elementi fondamentali:

- **Nodi**: sono degli elementi di elaborazione elementare
- **Collegamenti**: uniscono i nodi e permettono lo scambio di informazioni.

Nel caso dell'esempio di Fig. 4 un possibile block diagram è rappresentato in Fig. 5.

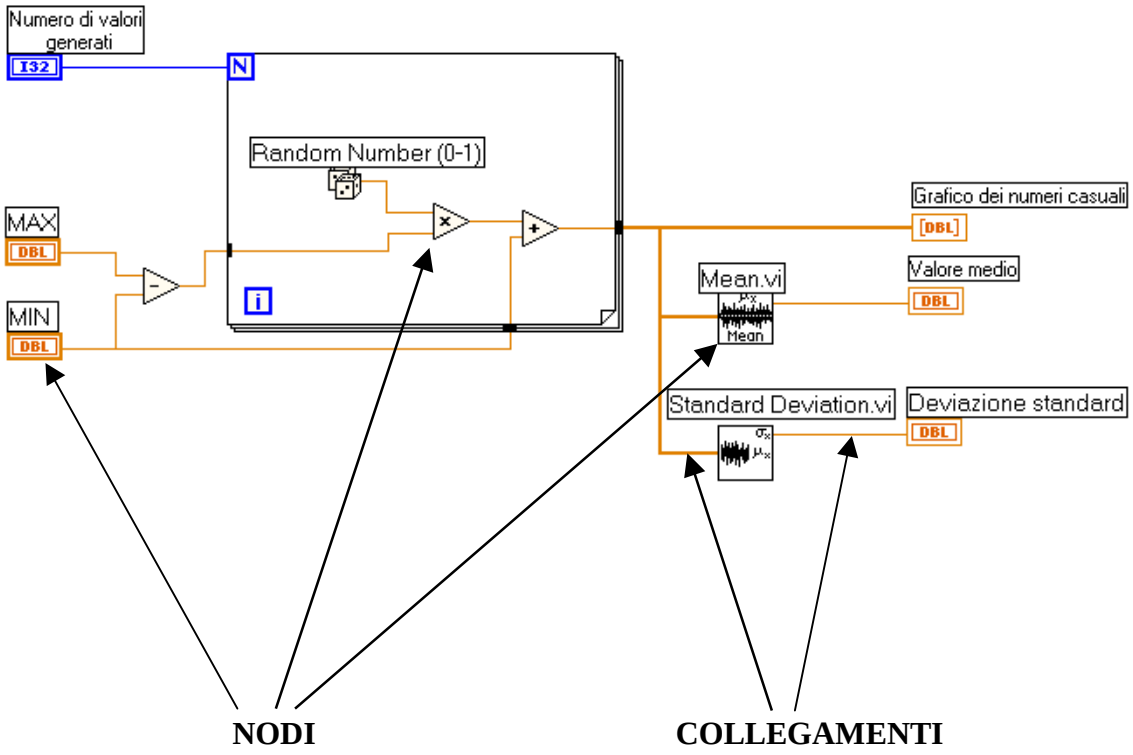


Fig. 5. Block Diagram di un VI che genera una sequenza di numeri casuali.

Nel Block Diagram si considerano nodi gli elementi terminali ai quali si collegano i fili: collegando i nodi del block diagram tra di loro è possibile far eseguire al VI la funzione desiderata.

Ad esempio il nodo "x" vede come ingresso i 2 dadi (generatore di numeri casuali) ed il fattore di scala dato dalla differenza tra "MAX" e "MIN": l'uscita del nodo, ad ogni iterazione del ciclo for, scarica sul grafico il numero casuale appena generato moltiplicato per il fattore di scala.

Gli elementi fondamentali di elaborazione dei dati e le strutture base (case, for, while, sequence), sono limitate e raccolte in un menù chiamato "**controls palette**" e descritto in un paragrafo successivo.

2.3 Icon & connector

La coppia **Icon & connector** (icona e connettore) è il terzo elemento fondamentale di un VI e consente di trasformare un programma in un oggetto, come verrà meglio chiarito in un paragrafo successivo.

L'icona è un simbolo grafico di piccole dimensioni che rappresenta sinteticamente il VI stesso: essa è normalmente visibile in alto a destra della finestra front panel e della finestra block diagram.

Quando il VI corrente viene impiegato all'interno di un altro programma, come sub-VI, l'icona permette di identificare il sub-VI stesso nel block diagram del programma principale che lo contiene.

Il sub-VI può essere collegato agli altri nodi del block diagram. In tal caso il connettore fissa la corrispondenza tra un'area dell'icona sulla quale viene eseguita la connessione ed uno degli elementi di ingresso/uscita del pannello frontale associato al sub-VI.

Le modalità secondo le quali può essere realizzata tale corrispondenza verranno illustrate in un paragrafo successivo. Per il momento è sufficiente tenere presente il significato associato al connettore.

Nell'esempio di Fig. 4 si può già osservare l'icona associata al VI: compare nel riquadro in alto a destra della finestra di interazione con il pannello frontale.

La corrispondenza tra VI e icona/connettore è meglio evidenziata in Fig. 6

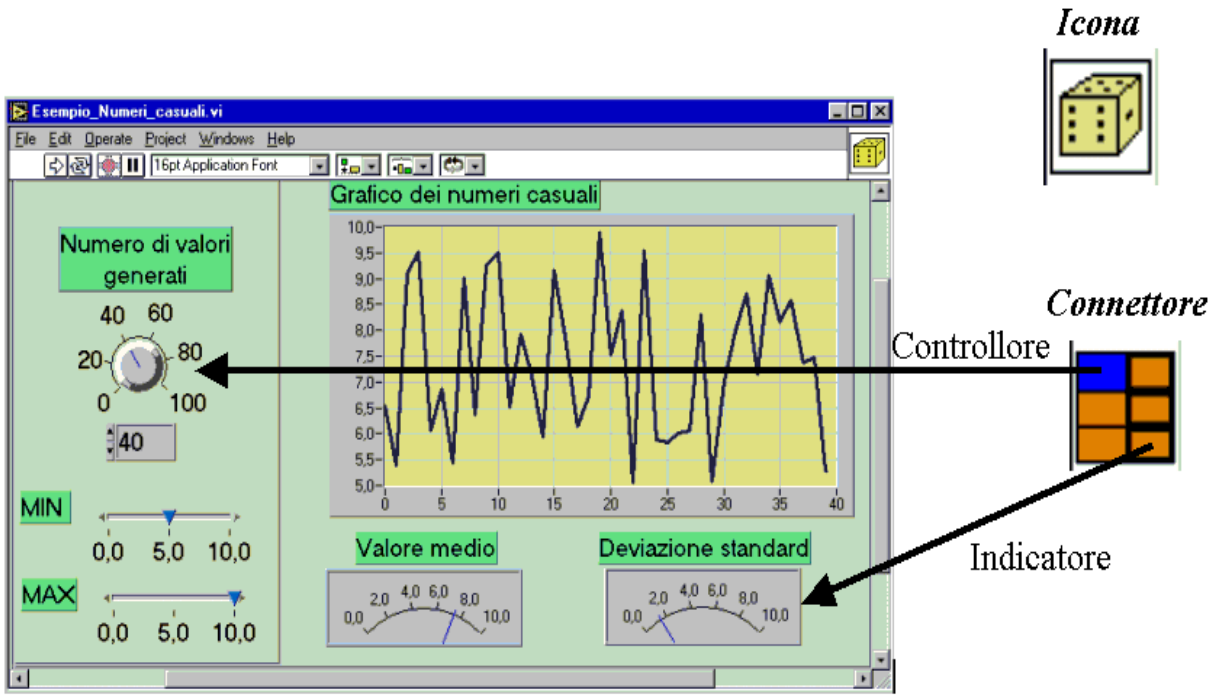


Fig. 6. Corrispondenza tra block diagram, icona e connettore.

Attraverso opportune operazioni (illustrate nel paragrafo intitolato "costruzione di un VI") è possibile impostare la finestra contenente il front panel in modo che questa mostri il connettore al posto dell'icona.

Nel caso dell'esempio illustrato, per facilitare la comprensione in Fig. 6 il connettore è stato evidenziato a parte e le frecce mostrano la corrispondenza impostata tra le aree del connettore e gli elementi del pannello frontale.

La definizione della coppia icona/connettore ha anche effetto nella rappresentazione del VI come sub-VI. Ad esempio il VI di Fig. 4 potrebbe, se inserito in un block diagram di un altro VI, assumere l'aspetto di Fig. 7.

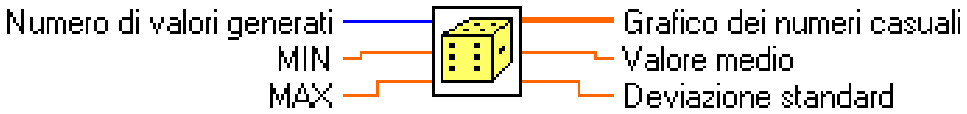


Fig. 7. Icona associata al VI che genera dei numeri casuali.

Si noti nella Fig. 7 la presenza dei terminali associati ai controllori ed agli indicatori presenti nel pannello frontale del VI:

- A sinistra sono presenti i terminali di ingresso del VI
- A destra sono presenti i terminali di uscita del VI

I nomi associati ai terminali sono gli stessi degli elementi presenti nel Front Panel.

3 Fondamenti sull'interazione con l'ambiente LabVIEW

Come già accennato LabVIEW è un ambiente interattivo che consente sia di eseguire VI precostruiti, sia di realizzarne di nuovi.

Appena mandato in esecuzione LabVIEW presenta la finestra di dialogo di Fig. 8 mediante la quale è possibile leggere un VI già presente su disco o prepararne uno di nuovo.

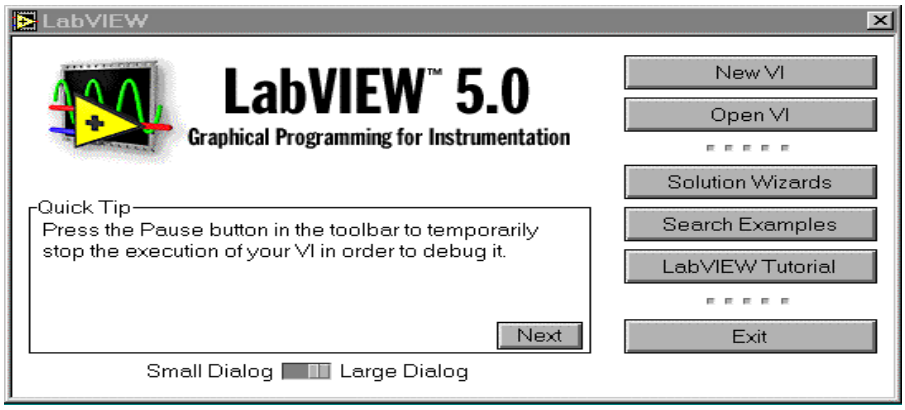


Fig. 8. Finestra di dialogo all'avvio di LabVIEW.

Premendo il tasto **OPEN VI,** Labview apre un file già presente su disco del quale deve essere specificato il nome in una apposita finestra che compare una volta premuto il tasto in questione.

Premendo il tasto **NEW VI,** LabVIEW apre un nuovo VI, denominato per default UNTITLED 1.VI, e fa comparire sul video il front panel ed block diagram ad esso associati.

Una volta eseguita questa scelta, l'interazione con l'ambiente prosegue agendo con alcune finestre a video. Le finestre di uso più frequente sono il pannello frontale ed il block diagram, illustrate in Fig. 9.

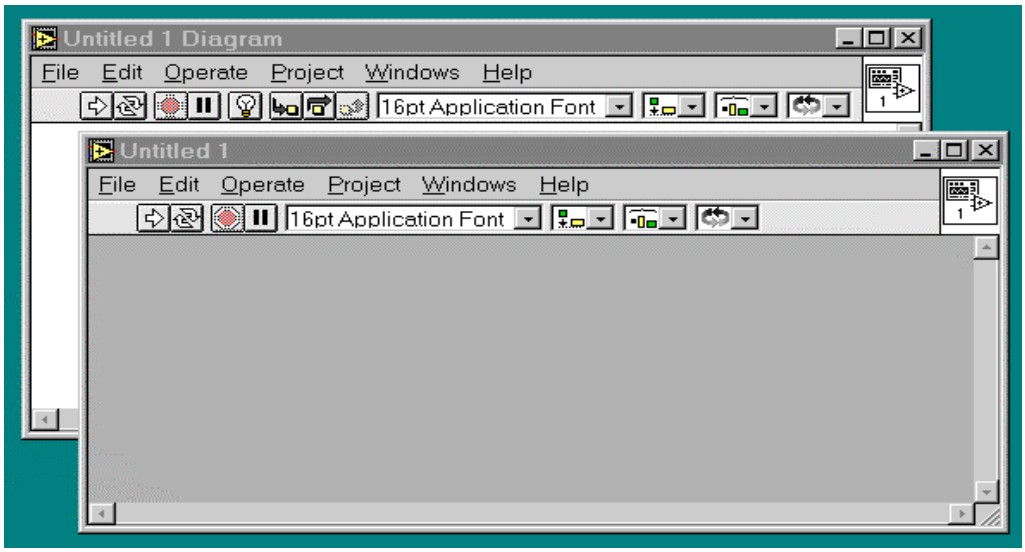


Fig. 9. Front Panel & Block Diagram all'apertura di un nuovo VI.

Tali finestre presentano una barra di menù a tendina: le principali opzioni a disposizione sono elencate qui sotto.

- *File*: consente di eseguire operazioni di I/O su file associati al VI corrente.
- *Edit*: consente di selezionare, cancellare, copiare elementi presenti sulla finestra attiva
- *Operate*: consente di mandare in esecuzione un VI, attribuire alle variabili i valori di default, …
- *Project*: permette di costruire applicativi eseguibili (se è presente il modulo *application builder*), di visualizzare la gerarchia dei VI, …
- *Windows*: consente di gestire la visualizzazione delle finestre presenti sullo schermo (front panel, block diagram, toools palette, controls palette, …).
- *Help*: consente di accedere ad informazioni di aiuto.

Si sottolinea inoltre come per alcune operazioni sarà necessario attivare dei menù del tipo "pop-up". Si tratta di menù a tendina i quali sono attivati interagendo con il mouse in aree a video associate a elementi grafici del pannello frontale e del diagramma a blocchi.

Nel caso di piattaforme Windows e Unix è sufficiente selezionare con il tasto destro del mouse l'elemento del quale si vuole far comparire il menù "pop-up". Nel caso di altre piattaforme Mac (oltre che Windows e Unix) si deve invece preventivamente selezionare lo strumento object pop-up dalla finestra "tools palette" e poi selezionare con il mouse l'oggetto del quale si desidera far comparire il menù "pop-up".

A titolo di esempio si veda la Fig. 10, la quale illustra il menù pop-up associato ad una manopola del pannello frontale.

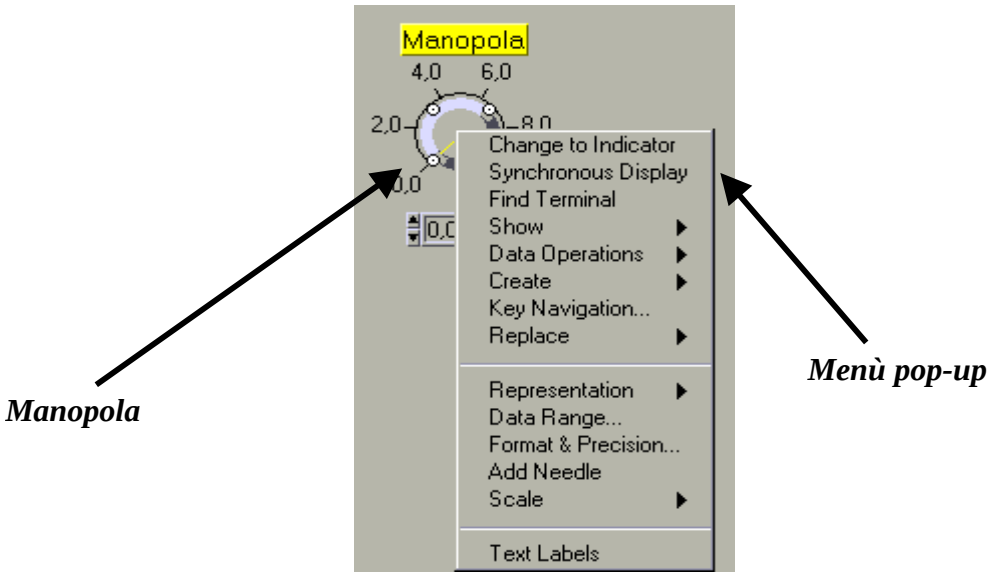


Fig. 10 Esempio di pop-up menù.

Ulteriori finestre consentono di interagire con l'ambiente: esse sono la ***tools palette,*** la ***controls palette***, la ***functions palette***.

La ***tools palette*** , illustrata in Fig. 11, può essere mostrata sia quando è attiva la finestra front panel, sia quando è attiva la finestra block diagram.

Fig. 11. Tools Palette.

La *tools palette* mette a disposizione la seguente serie di strumenti:

 **Operating tool**: consente di modificare il valore di un controllore o di selezionarne il testo interno.

 **Positioning tool**: consente di selezionare, posizionare, ridimensionare oggetti.

 **Labelling tool**: consente di editare testo e di creare etichette.

 **Wiring toll** : collega i terminali dei nodi presenti nel block diagram.

 **Object pop-up Menu tool**: apre il menù "pop-up" di un oggetto.

 **Scrool tool**: consente di effettuare lo scrool di una finestra selezionata senza utilizzare la barra di scrool.

 **Breakpoint tool** : inserisce un breakpoint nel punto selezionato.

 **Probe tool** : inserisce una probe nel punto selezionato.

 **Color copy tool**: consente di copiare il colore di un oggetto presente sul front panel per poi renderlo riutilizzabile nella Color tool.

 **Color tool**: consente di impostare i colori del front panel.

La ***Controls Palette***, rappresentata in Fig. 12, mette a disposizione una serie di controllori ed indicatori simili a quelli presenti in uno strumento reale, in modo che sia possibile realizzare un programma che possa essere facilmente utilizzato da un utente.

Fig. 12. Controls Palette.

Gli elementi fondamentali della controls palette sono elencati qui di seguito. Si noti che a ciascuno di essi può essere associato uno o più sotto-menu grafici simili alla controls palette stessa.

Numeric controls: controllori ed indicatori associati a variabili di tipo numerico.

Boolean Controls: controllori ed indicatori associati a variabili di tipo booleano

String&table controls: controllori ed indicatori associati a variabili di tipo stringa.

List & rings: controllori ed indicatori associati a variabili di tipo "lista di opzioni"

Array&Cluster controls: controllori ed indicatori di tipo array e di tipo cluster.

Graph controls: indicatori e controllori per la rappresentazione di dati in forma grafica.

Path&refnum controls: consentono la gestione di percorsi su disco e di riferimenti a file.

Decorations: consentono di modificare l'aspetto di un pannello frontale inserendo ulteriori elementi grafici.

 **User controls**: definizione di controllori preparati dall'utente.

 **Active X**: abilita il supporto per oggetti del tipo Active X.

 **Select a control**: consente di scegliere un controllore ottimizzato precedentemente costruito.

La ***Functions Palette***, rappresentata in Fig. 13, può essere visualizzata solo nel block diagram: essa mette a disposizione dell'utente una serie di strutture e funzioni predefinite e necessarie per la realizzazione del codice sorgente di un programma LabVIEW.

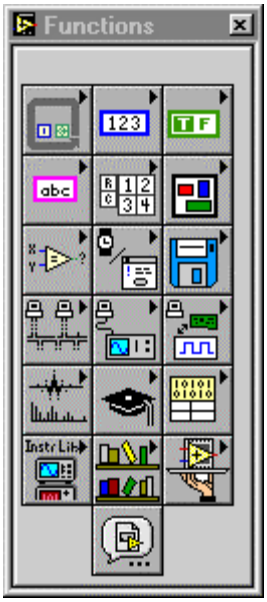


Fig. 13. La finestra Functions Palette.

Si elencano ora i sottomenù della functions palette dandone una breve descrizione: una descrizione più dettagliata verrà fornita successivamente.

 **Structures**: insieme di strutture messe a disposizione dal linguaggio G che sono: ciclo while, ciclo for, case, sequence. Contiene inoltre i nodi di tipo "global e local variable" ed il "formula node"

 **Numeric functions**: insieme di funzioni numeriche, che svolgono operazioni numeriche conversioni, calcoli trigonometrici. Questa palette contiene anche costanti numeriche.

Boolean functions: raccolta di funzioni booleane e di operatori logici

String functions: insieme di funzioni di manipolazione di stringhe e di conversione di numeri in stringhe e viceversa.

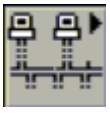
Array functions: raccolta di funzioni operanti sugli array.

Cluster functions: insieme di funzioni che permettono di costruire un cluster, modificarne elementi e disassemblarlo.

Comparison functions: raccolta di funzioni di confronto di dati (minore di, maggiore di, uguale a …)

Time&dialog functions: insieme di funzioni che gestiscono processi di temporizzazione e di comunicazione diretta con l'utente, grazie a finestre di dialogo aperte run-time.

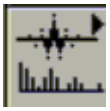
File i/o functions: raccolta di funzioni di comunicazione con unità disco o file.



Communication: raccolta di VI che consentono di gestire la comunicazione in rete e tra applicazioni diverse (ad esempio fornisce funzioni che utilizzano il protocollo TCP/IP).

Instrument I/O: insieme di VI che consentono di comunicare con gli strumenti utilizzando il bus 488, VISA e seriale.

DAQ: raccolta di VI di acquisizione e generazione di dati analogici e digitali real-time.



Analysis: insieme di VI che effettuano misure, generano e filtrano segnali, …

Tutorial: fornisce una serie di esempi di chiarimento.

Advanced functions: insieme di funzioni altamente specializzate

Instrument driver library: raccolta di VI che funzionano come driver di strumenti su bus 488, VISA e seriale.

User Library: permette di accedere velocemente, e quindi di posizionare sul block diagram, dei VI memorizzati in librerie o in file distinti.

Applications control: include VI per il controllo dell'esecuzione di un programma LabVIEW ;include funzioni di help, funzioni di menù, funzioni di stampa di VI, …

Select a VI: consente di selezionare un VI usando una finestra di dialogo opportuna e di inserirla nel block diagram corrente.

3.1 Costruzione di un VI

I 3 passi essenziali da seguire per costruire un VI sono:

- Costruzione del Front Panel.
- Costruzione del Block Diagram.
- Definizione della coppia Icon/connector.

3.1.1 Costruzione del pannello frontale

Poichè il pannello frontale rappresenta l'interfaccia utente, deve quindi essere realizzato in modo tale da essere facilmente ed intuitivamente utilizzabile.

LabVIEW mette a disposizione parecchi tipi di controllori ed indicatori, tutti raccolti nella controls palette descritta nel paragrafo precedente.

La costruzione del pannello frontale avviene mediante operazioni di "**drag & drop**" con il mouse degli elementi messi a disposizione dalla contols palette. Tale operazione consiste nella selezione di un elemento, controllore o indicatore, dalla *tools palette* (**drag**).

Successivamente si posiziona il puntatore del mouse nell'area del pannello frontale nella quale si desidera sia posizionato quell'elemento, e si agisce con il tasto sinistro del mouse per inserirlo (**drop**) nel pannello frontale.

Ad esempio, si supponga di voler inserire nel front panel una manopola: la sequenza di operazioni da seguire consiste nel selezionare dal menù "numeric" dalla *controls palette* l'elemento manopola e poi posizionarla nel front panel ove si desidera. Tale operazione è schematizzata in Fig. 14.

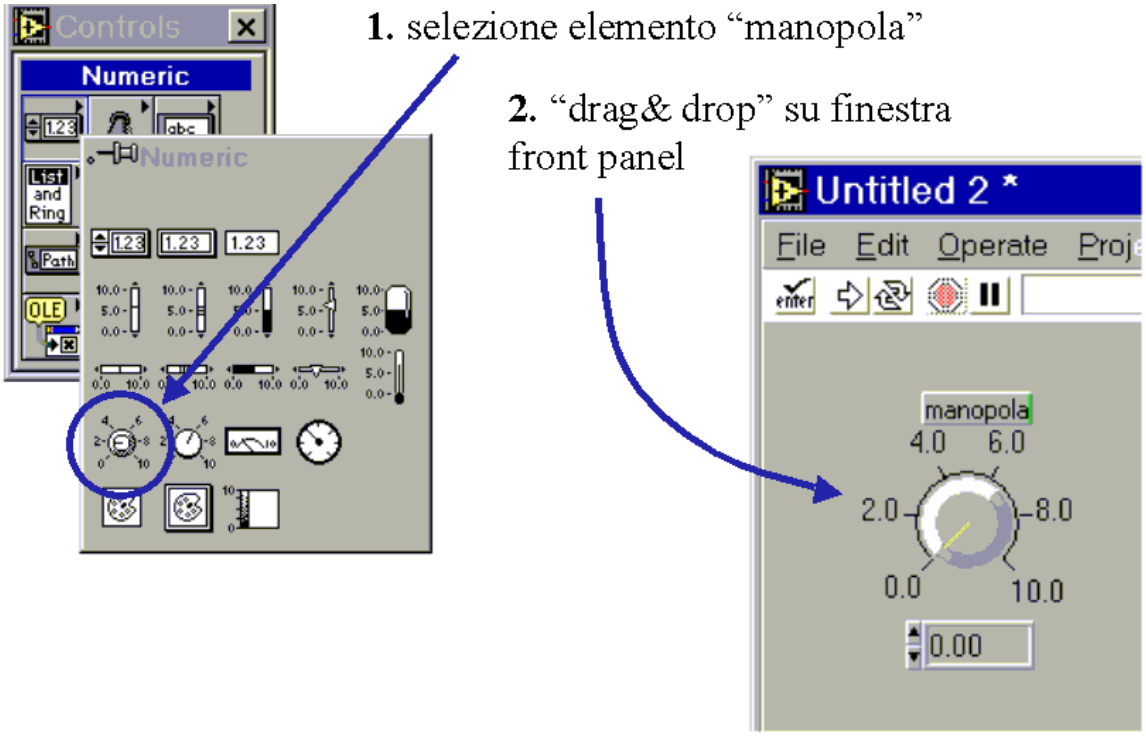


Fig. 14. Posizionamento di una manopola nel front panel.

Una volta posizionati i controllori e gli indicatori desiderati nel front panel, è possibile effettuare delle modifiche dell'aspetto, oppure cambiare valori numerici ad essi associati: queste operazioni sono rese possibili dagli strumenti della tools palette.

3.1.2 Costruzione della coppia Icon/Connector

Il passo successivo da seguire per la realizzazione del VI è la costruzione della coppia icona/connettore.

Una volta costruito un front panel, LabVIEW associa automaticamente al VI un'icona ed un connettore di default: mediante un menù pop-up associato all'area dell'icona, è possibile modificarli entrambi, come illustrato in Fig. 15.

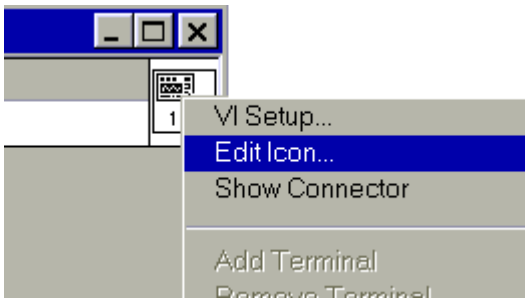


Fig. 15. Menù pop-up icona.

Selezionando Edit Icon compare la finestra di dialogo per l'editazione dell'icona la quale è rappresentata in Fig. 16.

A questo punto è possibile disegnare l'icona utilizzando gli strumenti messi a disposizione. Si noti che è anche possibile convertire un'immagine bitmap, di dimensioni qualsiasi, in icona semplicemente mediante un'operazione di "drag&drop" dal file che contiene la bitmap all'area dell'icona della finestra LabVIEW front panel o block diagram.

Con riferimento al VI che effettua la somma di 2 numeri, il processo di creazione dell'icona è rappresentato schematicamente in Fig. 16.

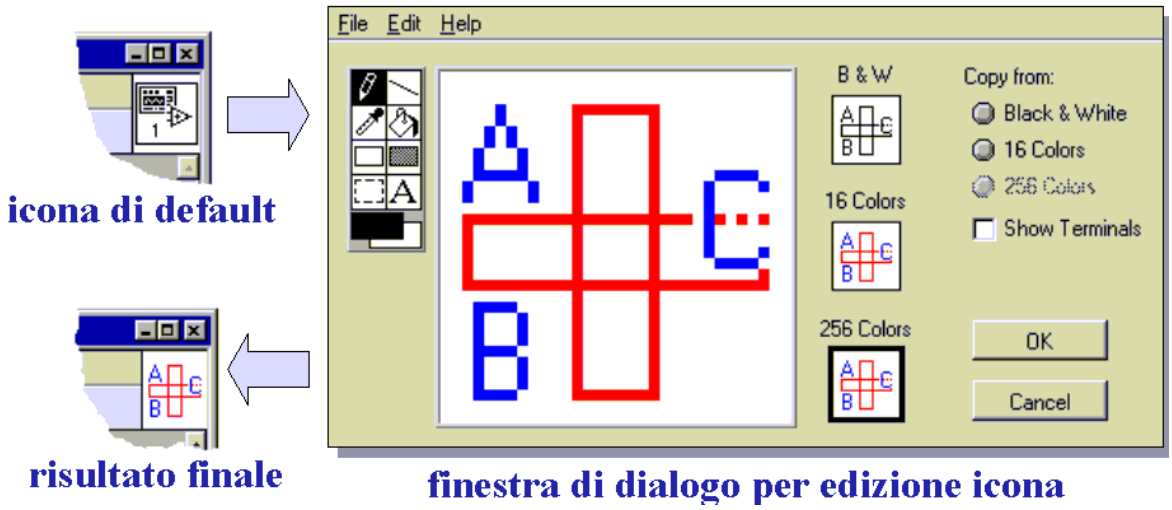


Fig. 16. Edizione di un'icona.

Una volta costruita l'icona, si può passare alla definizione del connettore.

Per fare in modo che venga visualizzato il connettore, basta eseguire un'operazione di pop-up sull'area del front panel associata all'icona e selezionare "Show Connector" dal menù corrispondente, come illustrato in Fig. 15.

Una volta eseguita tale operazione, compare il connettore di default associato al VI: mediante lo strumento "Wiring" è possibile collegare gli elementi del pannello frontale con aree distinte del connettore stesso selezionando con il mouse alternativamente un'area del connettore ed un elemento di ingresso/uscita del pannello frontale (Fig. 17).

Si noti che di solito si preferisce posizionare nel connettore a sinistra i controllori e a destra gli indicatori.

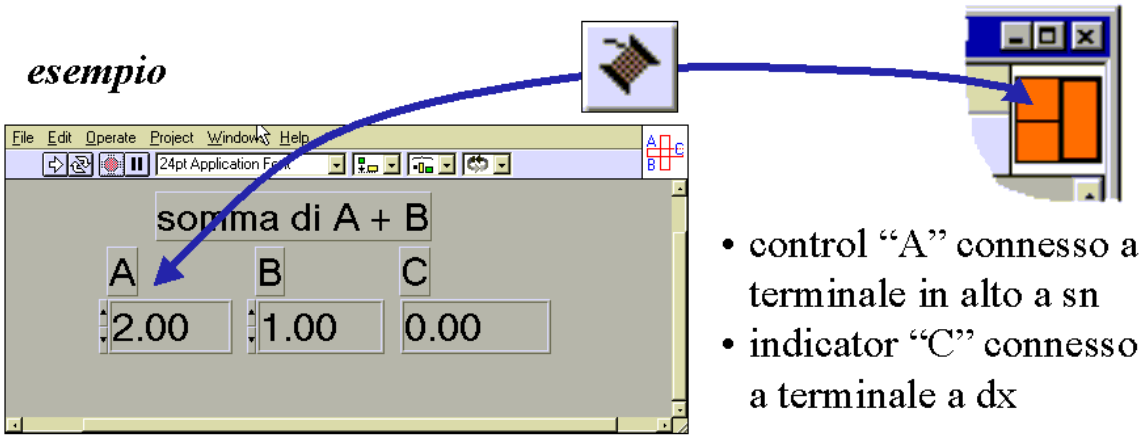


Fig. 17. Definizione di un connettore.

E` possibile modificare il pattern, ovvero la struttura grafica del connettore o il numero dei suoi elementi di ingresso/uscita, di un connettore selezionando il sottomenu "patterns" dal pop-up menu del connettore stesso, come illustrato in Fig. 18.

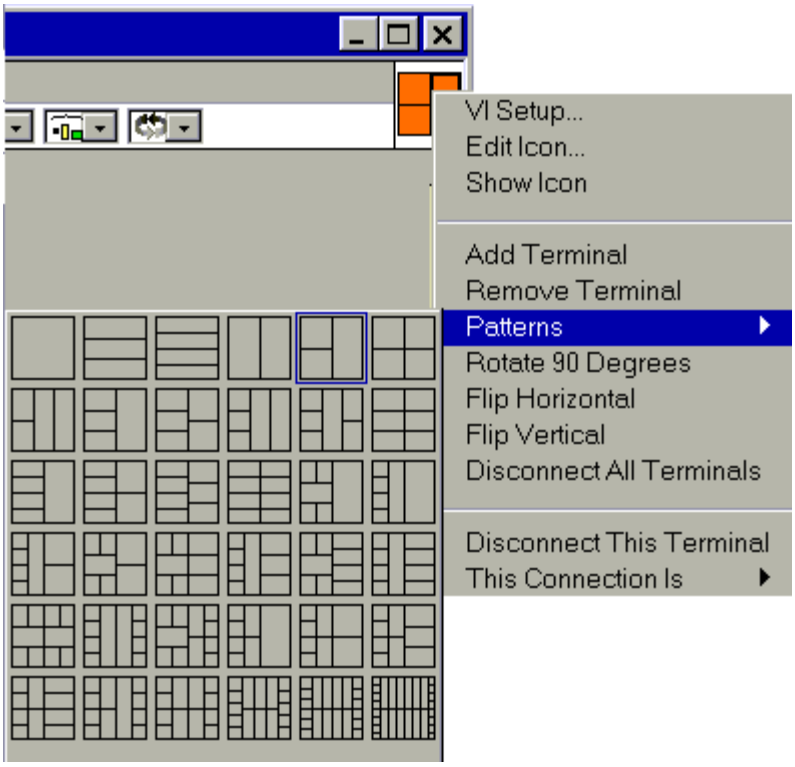


Fig. 18. Menu per il cambiamento del pattern di un connettore.

Le altre voci presenti nel menù di Fig. 18 consentono di modificare altri parametri del connettore: le più utilizzate sono:

- **Add terminal/remove terminal**: consentono di aggiungere o rimuovere un terminale del connettore.

- **Rotate 90 Degrees**: ruota il pattern del connettore di 90 gradi in senso orario.
- **Disconnect all terminals/Disconnect this terminal**: sconnette tutti i terminali collegati agli elementi sul pannello frontale, oppure sconnette il terminale selezionato.

3.1.3 Costruzione del Block Diagram

Una volta costruito il pannello frontale di un VI, nel Block Diagram corrispondente compaiono un'icona ed una label per ciascuno dei controllori ed indicatori presenti nel pannello frontale.

La costruzione del codice avviene mediante connessione dei controllori e degli indicatori a nodi di elaborazione o funzioni, rappresentate da icone o da opportuni elementi grafici e che permettono di effettuare le elaborazioni desiderate.

Le strutture e funzioni a disposizione sono raccolte nella **Function Palette** e possono essere posizionate nel Block Diagram mediante una operazione di "drag & drop" analoga a quella vista nel caso del pannello frontale.

Una volta posizionate possono essere collegate a controllori, a indicatori, a terminali di altre funzioni mediante una operazione di wiring utilizzando il "Wiring Tool" (icona che rappresenta un rocchetto di filo) della tools palette, come illustrato in Fig. 19.

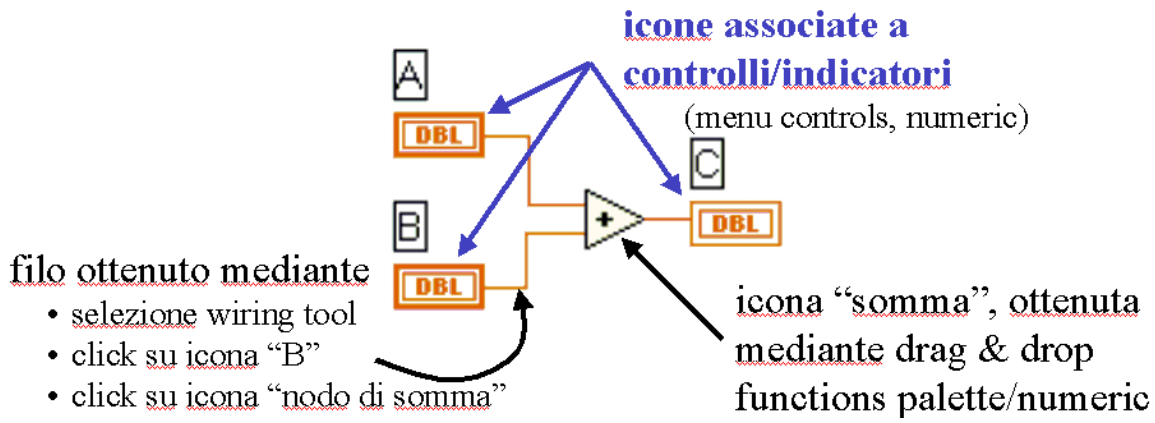


Fig. 19. Costruzione di un block diagram.

L'operazione di wiring si effettua seguendo i seguenti passi:

- Selezionare il wiring tool dalla tools palette  .
- Selezionare con il tasto del mouse l'area di interesse della prima icona.
- Spostare il rocchetto agendo con il mouse.
- Selezionare con il tasto del mouse l'area di interesse della seconda icona.
- Nel caso di collegamenti complessi, effettuare eventuali operazioni di selezione con il bottone del mouse in punti nei quali si desidera vincolare il passaggio del filo.

Si noti che un collegamento può essere spostato o rimosso, a tal fine è necessario preventivamente selezionare lo strumento *positioning tool* (tools palette) e poi si può agire con il mouse nel filo in oggetto (click&drag, click+tasto delete).

Quando si effettua un collegamento sbagliato, il filo appare tratteggiato: può essere successivamente rimosso selezionando l'opzione "**Remove bad wires**" dal menù **Edit,** oppure premendo (Ctll+B).

La costruzione di un programma LabVIEW avviene necessariamente mediante l'utilizzo delle funzioni di base raccolte nella ***functions palette***, descritta nel paragrafo 3: il meccanismo per inserire una nuova funzione o VI all'interno del VI corrente è lo stesso descritto per il pannello frontale, ovvero selezione dell'elemento dalla functions palette e posizionamento nel block diagram mediante "drag&drop". Alcune di queste funzioni sono utilizzate in pressoché qualunque programma LabVIEW: si tratta delle strutture base di controllo di flusso, raccolte nel sotto-menu **structures** della functions palette.

Le structures a disposizione sono **Sequence**, **Case**, **For Loop**, **While loop**, **Formula Node** : esse saranno descritte in dettaglio in un paragrafo successivo.

Per ora si noti che le strutture controllano il flusso di esecuzione della porzione di codice contenuta al loro interno: si consideri come esempio un ciclo while, di cui è riportato un esempio in Fig. 20.

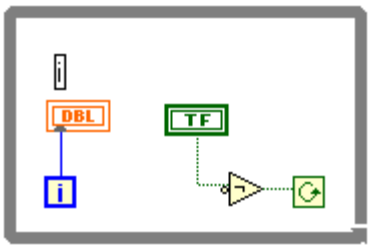


Fig. 20. Esempio d'uso di un ciclo while.

Il ciclo termina quando la variabile booleana di controllo, passata all'icona rappresentata da una freccia chiusa in un cerchio con direzione antioraria, diventa false: fino a che essa rimane true, sul pannello frontale viene visualizzato il valore dell'indice i (che si incrementa di uno ad ogni iterazione).

La porzione di codice eseguita (visualizzazione dell'indice i) è quella **interna** al bordo esterno del ciclo while: eventuale codice inserito all'esterno di tale limite **non** fà parte del ciclo.

Mediante la "positioning tool"  della tools palette è possibile allargare o rimpicciolire le dimensioni del bordo esterno che delimita le strutture: ciò consente di inserire al loro interno porzioni di codice più o meno grandi.

3.2 Sub-VI

Un **sub-VI** è un VI utilizzato all'interno di un altro VI come sottoprogramma.

Qualunque VI, una volta creato, può essere utilizzato come sub-VI all'interno di un altro VI di più alto livello. Di conseguenza un sub-VI è analogo ad una subroutine in un linguaggio di programmazione di tipo text-based.

Come avviene in altri linguaggi con le subroutine, non c'è alcun limite al numero di sub-VI inseribili all'interno di un programma in linguaggio G. È inoltre possibile chiamare un sub-VI dall'interno di un altro sub-VI.

Se inserito in un VI, un sub-VI presenta l'icona ad esso associata nel block-diagram che lo contiene.

Quando impiegato come sub-VI, un VI normalmente non mostra a video il proprio pannello frontale: riceve invece dati di ingresso per mezzo dei collegamenti tra il sub-VI stesso ed il resto del programma che lo contiene. Si noti che la corrispondenza tra connessioni nel programma LabVIEW, aree dell'icona associata al sub-VI, elementi di I/O del sub-VI stesso, è fissata nel momento della definizione del connettore: la Fig. 21 illustra tale corrispondenza.

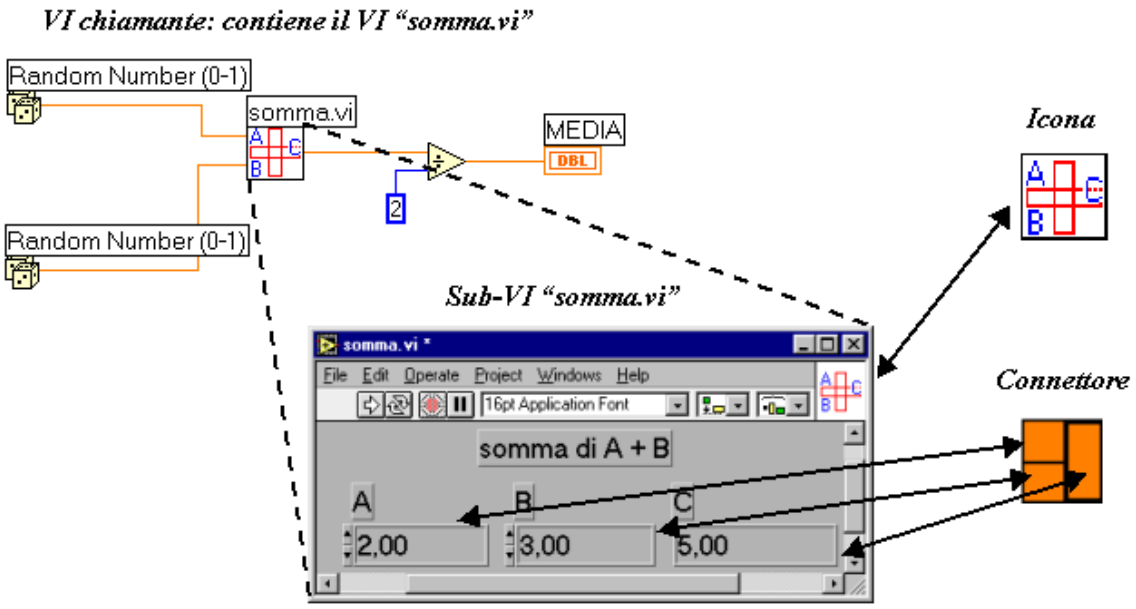


Fig. 21. Relazione tra VI, sub_VI, icona e connettore di un sub-VI.

Nella Fig. 21 è riportata una porzione di codice di un VI chiamante: al suo interno è stato inserito come sub-VI il VI chiamato "somma.vi".

Si vede che nel block diagram del programma chiamante, il VI "somma.vi" che viene utilizzato come sub-VI, compare nella forma dell'icona ad esso associata.

I collegamenti a "somma.vi" possono essere effettuati solo nelle aree definite nel connettore del VI stesso.

Quando il programma chiamante viene eseguito, il pannello frontale del VI "somma.vi" non viene mostrato, ma il dato che fluisce dal connettore in alto a sinistra viene assegnato al controllore "A" di "somma.vi", poichè "A" corrisponde all'area in alto a sinistra del connettore di "somma.vi". Analogamente per gli altri controllori ed indicatori.

3.2.1 Come creare un sub-VI da un VI

Come già detto, l'icona è un simbolo grafico che rappresenta un VI, mentre il connettore rappresenta l'interfaccia del VI con l'ambiente esterno (insieme di controllori ed indicatori rappresentati da terminali).

Se si vuole richiamare un VI dal block diagram di un altro VI, prima si deve creare l'icona ed il connettore associati al VI di livello più basso, seguendo la procedura descritta nel paragrafo 3.3.2.

Si può quindi concludere che la procedura per la creazione di un sub-VI è la stessa seguita per la creazione di un VI: ciò che distingue l'uno dall'altro é che un VI diviene un sub-VI se viene utilizzato all'interno di un altro block diagram.

Una volta costruito il VI che si vuole utilizzare come sub-VI, esso può essere salvato in 2 modi diversi:

- Come un normale file all'interno di una directory (SAVE AS *nome del file*).
- All'interno di una ***libreria***: per libreria si intende un file con estensione .LLB ove sono contenuti i VI che il programmatore ha inteso raggruppare perché di tipologia simile.

Una possibile libreria potrebbe contenere i VI che controllano la sezione di trigger di un oscilloscopio: per crearla si deve selezionare l'opzione *Save as* dal menù *File* e selezionare l'opzione *New VI library* dalla dialog box. A questo punto si deve inserire il nome che si vuole assegnare alla libreria con estensione .LLB (nel nostro caso *Trigger.llb*): successivamente compare una nuova finestra ove digitare il nome dei VI che verranno inseriti all'interno della libreria stessa (ad esempio *triglevel*, *triglope*, ...).

Il meccanismo di salvataggio dei VI in librerie consente di raggruppare più file per tipologia comune, di facilitare il trasferimento dei VI, di comprimere la dimensione dei file.

3.2.2 Come utilizzare un sub-VI all'interno di un altro VI

Un VI può essere utilizzato all'interno del block diagram di un altro VI richiamandolo premendo il tasto *Select a VI* della function palette.

Una volta selezionato il nome del VI, mediante la strumento *positioning*  della tools palette si posiziona nel block diagram l'icona che lo rappresenta. (con il consueto meccanismo di "drag&drop").

A questo punto il VI inserito diventa un sub-VI e può essere connesso agli altri elementi posizionati all'interno del block diagram mediante operazioni di *wiring*. Un semplice esempio può chiarire meglio questa procedura.

Si voglia costruire un VI che calcola le coordinate del baricentro di un triangolo una volta assegnate le coordinate dei vertici.

Si può pensare di risolvere il problema costruendo un VI che calcoli la proiezione delle coordinate del baricentro rispetto a ciascun asse coordinato, poi impiegare tale VI per la soluzione del problema posto: le Fig. 22 e Fig. 23 illustrano come procedere.

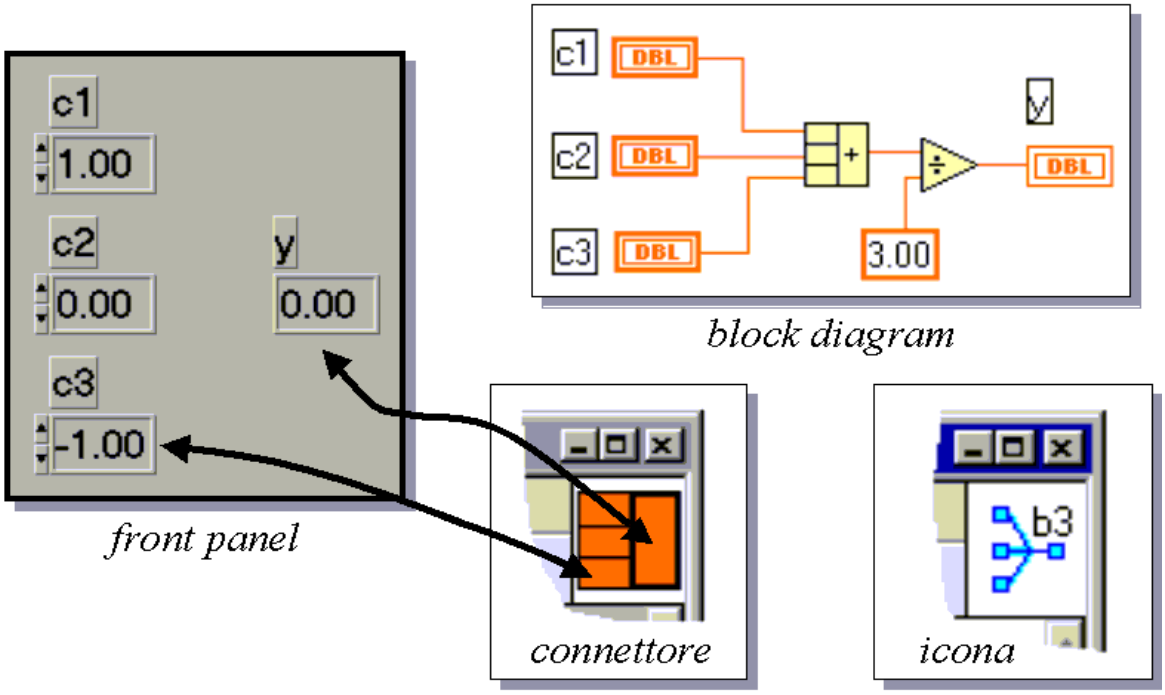


Fig. 22. VI che calcola il baricentro lungo un asse coordinato.

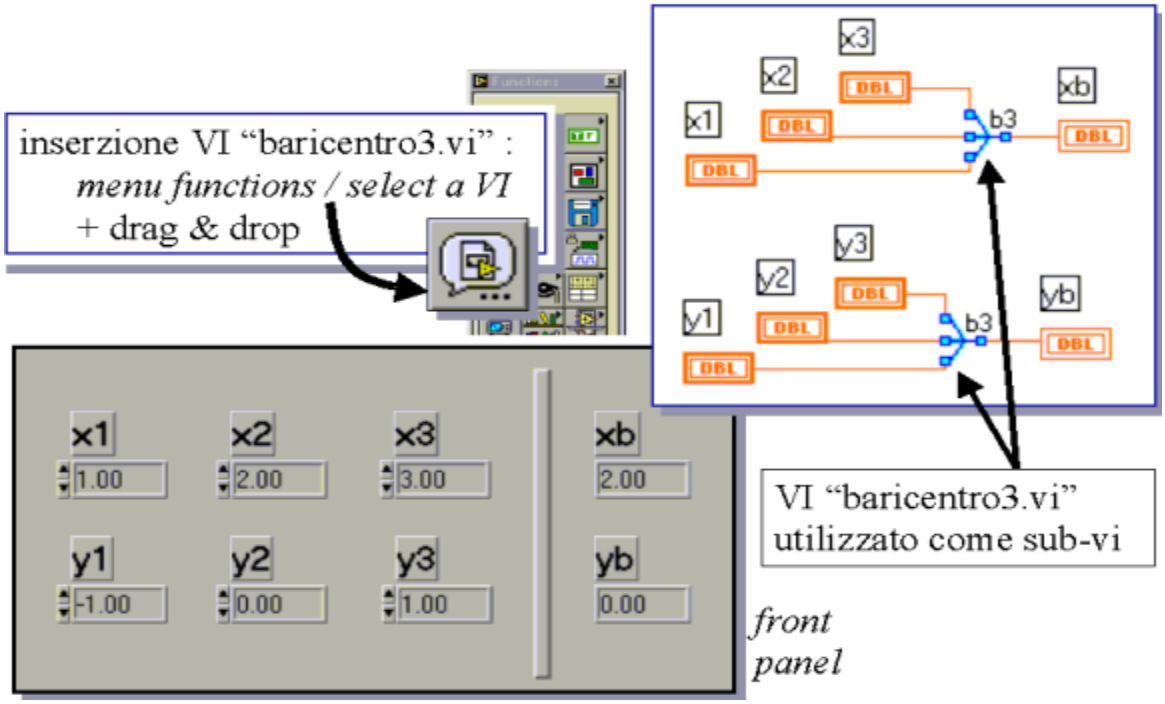


Fig. 23. Utilizzo del VI "baricentro3.VI" come sub-VI all'interno del VI che calcola le coordinate del baricentro di un triangolo.

Nella Fig. 22 è mostrato il VI che fornisce il calcolo della proiezione delle coordinate del baricentro lungo un asse coordinato, assegnate le corrispondenti coordinate dei vertici lungo quell'asse. Tale VI viene salvato nel disco con il nome "baricentro3.vi".

In Fig. 23 invece si mostra il pannello frontale e il codice del VI che consente il calcolo delle coordinate del baricentro di un triangolo.

Si noti come il VI "baricentro3.vi" sia incluso 2 volte nel diagramma a blocchi del VI chiamante.

3.3 Modifica delle proprietà di un controllore o di un indicatore

Gli indicatori ed i controllori posti sul pannello frontale assumono una configurazione di default: essi infatti presentano delle proprietà che possono essere ridefinite accedendo ad un menù pop-up selezionabile con il tasto sinistro del mouse.

Il pop-up menù presenta alcune voci comuni per tutti i tipi di controllori o indicatori selezionati, tra le quali si ricordano le seguenti:

- **Change to indicator/control**: consente di commutare un controllore in un indicatore e viceversa.
- **Find terminal**: trova nel block diagram il terminale associato al controllore o all'indicatore selezionato con il mouse.
- **Show**: apre un ulteriore menù pop-up che consente di scegliere se visualizzare sul front panel altri elementi grafici associati al controllore/indicatore (ad esempio una label).
- **Data operations**: permette di impostare delle proprietà associate al valore assunto dai controllori: in particolare l'opzione "make current value default" imposta il valore attualmente assunto come valore di default, in particolare se il VI viene letto da file quell'oggetto viene inizializzato con il valore di default impostato.
- **Create**: consente di creare un "*attribute node*" oppure una "*local variable*".
- **Replace**: consente di sostituire l'oggetto selezionato con un altro oggetto a scelta dalla controls palette.

3.3.1 Modifica delle proprietà di un elemento di tipo numerico

Per quanto riguarda gli elementi di tipo numerico, le voci più interessanti del menu pop-up illustrato in Fig. 24 sono: *representation, data range, format&precision.*

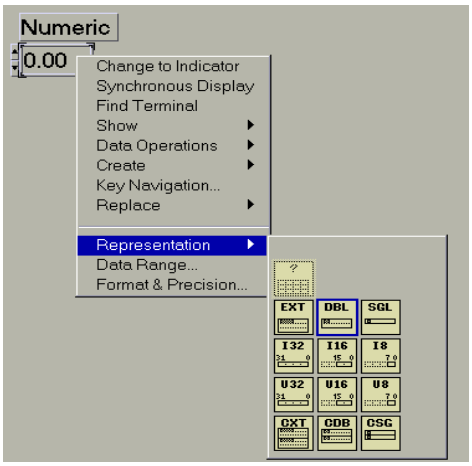


Fig. 24. Menu pop-up per elementi di tipo numerico.

- **Representation**: consente di modificare all'interno di una lista il tipo di rappresentazione interna dell'elemento selezionato.
  Per numeri floating point:
  SGL: 32 bit singola precisione (4 byte)
  DBL: 64 bit doppia precisione (8 byte)
  EXT: precisione estesa (10 byte)
  Per numeri interi:
  I8: byte con segno (8 bit)
  U8: byte senza segno (8 bit)
  I16: word con segno (16 bit)
  U16: word senza segno (16 bit)
  I32: long word con segno (32 bit)
  U32: long word senza segno (32 bit)

- **Data range**: consente di impostare i valori limite ed altri parametri che sono legati alla variabile in esame.

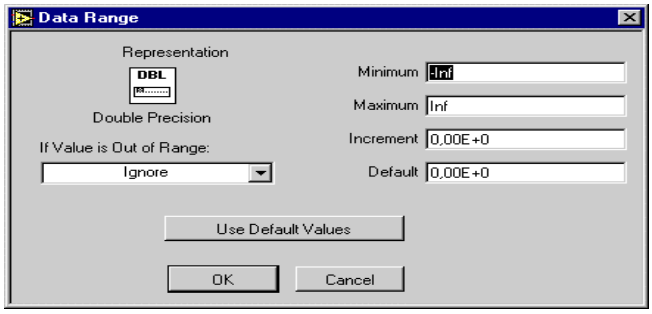


Fig. 25. Numeric Data Range.

- **Format&precision**: consente di selezionare il tipo di visualizzazione dei dati (scelta del tipo di notazione e del numero di cifre decimali).

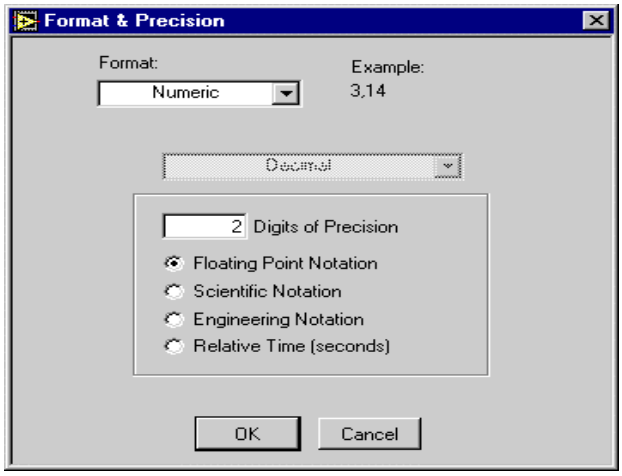


Fig. 26. Numeric Format&Precision.

3.3.2 Modifica delle proprietà di un elemento di tipo booleano

Per quanto riguarda gli elementi di tipo booleano, le voci più interessanti del pop-up menu illustrato in Fig. 27 sono: *data range* e *mechanical action.*

Fig. 27. Menu pop up per elementi di tipo boolean.

- **Data range**: questo menu permette di attivare due modalità che consentono di rilevare eventuali errori nei valori assunti da variabili booleane.

  Suspend if false: sospende l'esecuzione del VI se la variabile selezionata assume il valore false.
  Suspend if true: sospende l'esecuzione del VI se la variabile selezionata assume il valore true.

- **Mechanical action**: permette di selezionare la modalità con cui cambia lo stato logico della variabile. Le impostazioni possibili sono:

  Switch when Pressed: lo stato logico del controllo cambia non appena lo si seleziona con il mouse.

  Switch when Released: il cambiamento di stato della variabile di controllo avviene solo nel momento in cui il pulsante del mouse viene rilasciato.

  Switch until Released: il controllo si comporta come un pulsante, l'uscita cambia di stato solo nell'intervallo di tempo in cui il tasto del mouse viene tenuto premuto.

  Latch when Pressed: lo stato della variabile cambia quando il tasto viene premuto, ma viene mantenuto solo per il tempo necessario affinché il programma esegua una lettura del controllo, al termine della quale ritorna allo stato iniziale.

  Latch when Released: è del tutto simile alla precedente tranne per il fatto che il cambiamento di stato della variabile ha inizio solo quando il tasto del mouse viene rilasciato; il valore dello stato viene conservato per il tempo necessario a LabVIEW per una lettura del controllo.

  Latch until Released: la variabile di controllo cambia stato non appena il pulsante viene premuto. Il nuovo valore dello stato viene conservato durante tutto il periodo di tempo in cui il pulsante è tenuto premuto e successivamente finché il programma LabVIEW non ha effettuato una ulteriore lettura del valore.

3.3.3 Modifica delle proprietà di un elemento di tipo stringa

Per quanto riguarda gli elementi di tipo stringa, le voci più interessanti del pop-up menu illustrato in Fig. 28 sono: *Normal display, '\'codes display, Password display, Hex display.*

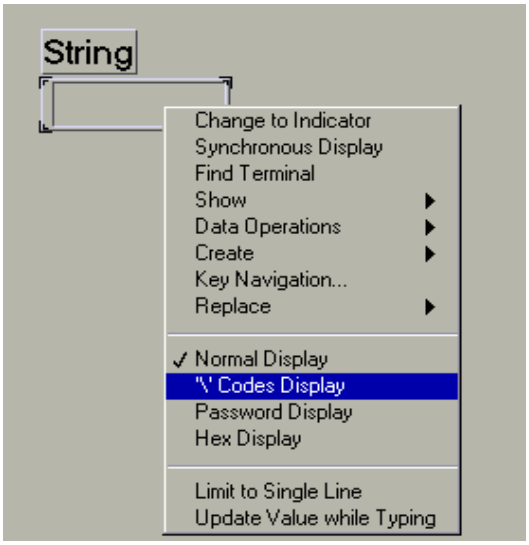


Fig. 28. Menu pop up per elementi di tipo stringa.

- **Normal display**: permette di visualizzare tutti i caratteri esattamente come vengono digitati.
- **`\`Codes display**: interpreta i caratteri immediatamente seguiti da una barra (\ ) come codice per i caratteri non visualizzabili (ad esempio **\n** corrisponde ad un ritorno a capo, **\s** ad uno spazio).
- **Password display**: i controllori di tipo stringa visualizzano tante * per ogni carattere inserito: dal block diagram viene comunque letta la stringa digitata.
- **Hex display**: visualizza la stringa sotto forma di caratteri esadecimali.

3.4 Modifica dell'aspetto di un VI

L'aspetto di un pannello frontale può essere modificato dal programmatore al fine di ottenere una interfaccia quanto più possibile intuitivamente utilizzabile.

LabVIEW consente di modificare l'aspetto del front panel mediante operazioni di spostamento di oggetti, di ridimensionamento, di aggiunta di elementi grafici di ornamento, di cambiamento di colore.

3.4.1 Spostamento di oggetti

È possibile spostare gli oggetti posizionati nel front panel selezionandoli utilizzando lo strumento  della tools palette: basta posizionare la freccetta sopra l'oggetto, selezionarlo cliccando il tasto sinistro del mouse, posizionarlo tenendo premuto il tasto sinistro e trascinarlo.

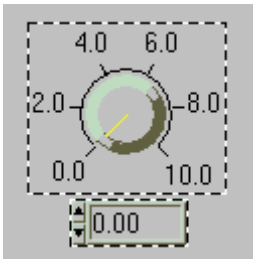


Fig. 29. Posizionamento di un oggetto nel front panel.

3.4.2 Ridimensionamento di oggetti

Una volta selezionato un oggetto, è possibile ridimensionarlo trascinando il bordo evidenziato utilizzando lo strumento  della tools palette.

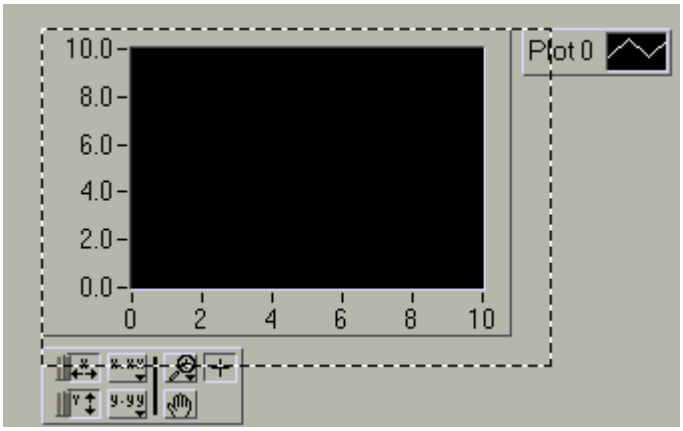
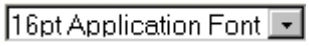


Fig. 30. Ridimensionamento di un grafico.

Per ridimensionare una label (o comunque un controllore o indicatore che contenga del testo, ad esempio una scala di valori graduata) la si deve selezionare con il tasto destro del mouse e cambiare il campo **size** dal menu pop-up che compare selezionando application font 

dalla front panel toolbar.

3.4.3 Aggiunta di elementi grafici di ornamento

Selezionando lo strumento **decorations**  dalla controls palette comparirà sul front panel una finestra nella quale è raccolta una serie di elementi di grafici di "abbellimento" che consentono di modificare l'aspetto del VI.

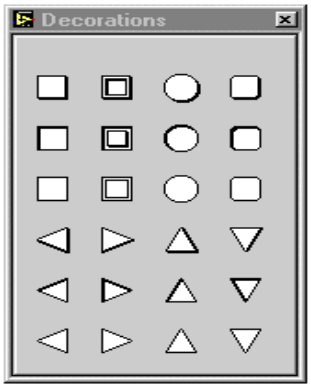


Fig. 31. Finestra "decorations".

È possibile posizionare gli elementi selezionati mediante la consueta operazione di "drag&drop" e ridimensionarli con la procedura descritta in precedenza.

3.4.4 Modifica del colore di oggetti

È possibile modificare il colore degli oggetti posizionati nel front panel utilizzando gli strumenti della tools palette.

Ad esempio si può copiare il colore di un oggetto posizionando su di esso lo strumento **color copy**  : il colore copiato compare nella finestra **color**  della tools palette. Posizionando successivamente il pennello sopra l'oggetto e premendo il tasto sinistro del mouse, esso assumerà il colore voluto.

È possibile scegliere direttamente i colori facendo comparire il menu pop-up dei color dalla tools palette, presentato in. Fig. 32.

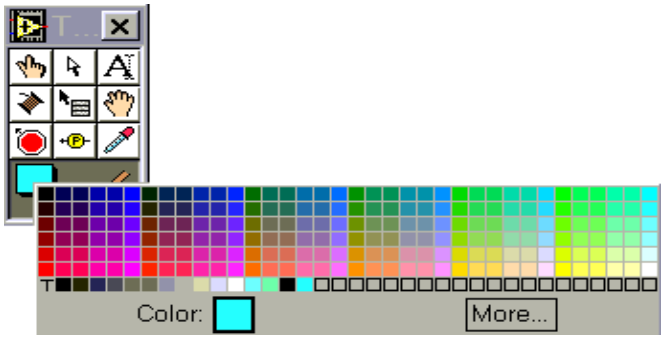


Fig. 32. Pop-up menu per la modifica del colore di un oggetto.

3.5 Strumenti per l'esecuzione di un VI

Una volta costruito il Virtual Instrument, è possibile mandarlo in esecuzione azionando il tasto della **Front Panel Toolbar**, presentata in Fig. 33.

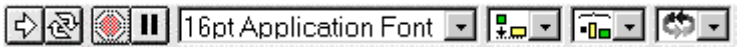


Fig. 33. Front Panel Toolbar.

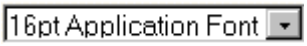
I singoli elementi della Front Panel Toolbar sono:

Run button: manda il VI in esecuzione singola: G compila automaticamente il VI, se necessario.

Run continously button: attiva la modalità di esecuzione continua del VI: il Vi viene eseguito ripetutamente finché l'utilizzatore preme il tasto **execution aborted**, oppure il tasto di pausa (**pause button**).

Abort button: premendo il tasto di "abort" LabVIEW fa terminare l'esecuzione del VI di livello più alto.

Pause button: premendo questo tasto, l'esecuzione viene posta in uno stato di pausa: per riprendere l'esecuzione normale è sufficiente premerlo nuovamente.



Application Font: consente di modificare le caratteristiche di un testo preventivamente selezionato: ad esempio consente di modificare il tipo di carattere, le dimensioni, il colore.

Align elements: consente di allineare degli elementi grafici selezionati, di porli in backgrund o in foreground, …

3.5.1 Esecuzione di un VI

Una volta costruito il Virtual Instrument, è possibile eseguirlo premendo il tasto della **front panel toolbar**.

Nel caso in cui il tasto di run singolo assuma l'aspetto , significa che nel programma sono presenti degli errori di scrittura del codice per cui non può essere mandato in esecuzione. Per trovare tali errori è possibile tentare ugualmente di eseguire il

VI. In tal caso LabVIEW mostra invece una finestra di dialogo con la lista degli errori di scrittura del codice.

In fase di esecuzione la porzione di front panel toolbar che consente il controllo dell'esecuzione di un VI, assume l'aspetto di Fig. 34

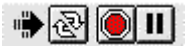


Fig. 34. Front Panel Toolbar di un VI in esecuzione singola.

Agendo sui tasti di stop e di pausa, è possibile fermare definitivamente oppure temporaneamente l'esecuzione del VI.

Una volta terminata la fase di run, il tasto di run singolo ritorna ad assumere l'aspetto originario.

Nel caso si voglia attivare la modalità di run continuo, l'utente deve attivare il pulsante  : a questo punto la front panel toolbar assume l'aspetto di Fig. 35.

In tal caso l'ambiente appena terminata l'esecuzione del VI lo ri-esegue in ciclo lasciando inalterati i valori dei controllori rispetto al momento di fine esecuzione del programma all'istante precedente.

Fig. 35. Front Panel Toolbar di un VI in esecuzione continua.

3.5.2 Debug di un VI

Per debug di un VI si intende l'attivazione di una modalità di esecuzione che consenta di verificare il buon funzionamento di un VI.

Si può effettuare il debug di un programma se non sono stati rilevati errori nella costruzione del block diagram, ovvero quando il run button appare come in Fig. 36.

Fig. 36. Run-button di un VI corretto.

Il debug può essere attivato dal block diagram agendo sui pulsanti della **Block Diagram Toolbar**, illustrata in Fig. 37.

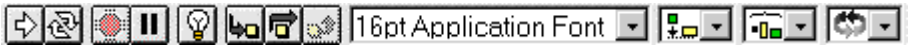


Fig. 37. Block diagram toolbar.

Rispetto alla front panel toolbar, vi sono 4 pulsanti che consentono di attivare diverse modalità di debug:

Highlight execution button: una volta attivato l'ambiente mostra una animazione nella quale i dati in transito sui connettori sono rappresentati da delle palline in movimento: consente quindi una facile comprensione della sequenza di operazioni svolte dal VI.

Step into button: esegue le singole operazioni del VI in esecuzione in modalità passo passo.

Step over button: consente di eseguire intere strutture in modalità passo passo (ad esempio esegue un intero loop).

Step out button: esegue l'intero VI fino a che la struttura corrente o il sub-VI corrente non termina la propria esecuzione.

LabVIEW mette a disposizione altri due strumenti per effettuare il debug di un VI:

Probe: è una "sonda" fornita dalla tools palette: una volta selezionata e posizionata sul punto desiderato del block diagram, compare una finestra che mostra il contenuto della variabile che scorre lungo quel filo.

Breakpoint: una volta selezionato lo strumento breakpoint dalla tools palette, se si seleziona (click con il tasto sinistro del mouse) un filo, compare una icona che rappresenta un segnale di stop. Quando un dato transita in quella posizione viene sospesa l'esecuzione del VI.

Vediamo un esempio di come si possa verificare il corretto funzionamento di un programma; si consideri un VI che esegua la seguente operazione scritta in pseudocodice:

```
For i = 0 to 19
   Vect [i] = 10 * i ;

Stampa Vect
```

Si supponga che il VI sia già stato costruito e compilato correttamente: se si vuole attivare la modalità di esecuzione single step di tipo highlated, si deve seguire la seguente sequenza di operazioni:

- premere il tasto "highlighted execution" dalla block diagram toolbar : una volta attivato diventerà
- premere il tasto "pause"
- lanciare il comando di run singolo : una volta attivato diventerà

- attivare uno dei pulsanti di esecuzione single step

Vediamo 2 figure, Fig. 38 e Fig. 39, che rappresentano 2 fasi distinte della esecuzione single step e con una sonda (probe) inserita all'uscita del nodo che effettua la moltiplicazione:

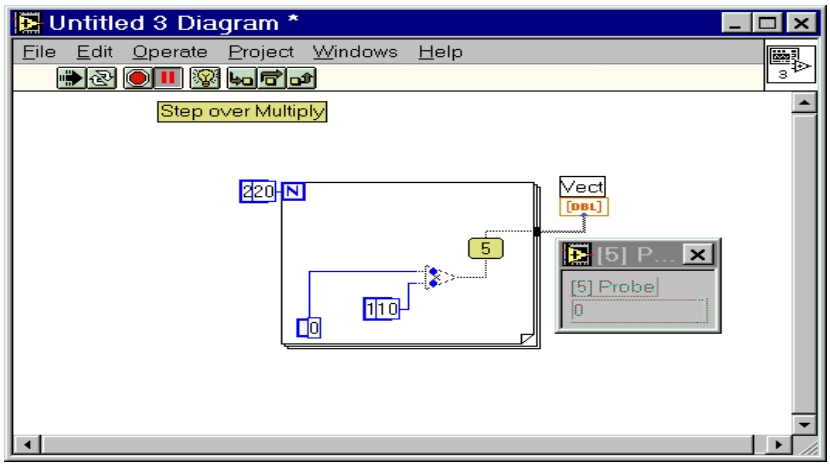


Fig. 38. Esecuzione highlighted, single step di un VI.

Una volta fatto andare in esecuzione, premendo ripetutamente il tasto "step into" viene visualizzato sul block diagram il flusso dei dati mano a mano che il programma procede nel suo funzionamento.

Giunto alla settima iterazione (ricordiamo che l'indice **i** parte da zero !), l'uscita dal nodo di moltiplicazione assume il valore (i-1)*10 = 6*10 = 60: si noti in Fig. 39 che la probe inserita dopo tale nodo assume proprio il valore 60.

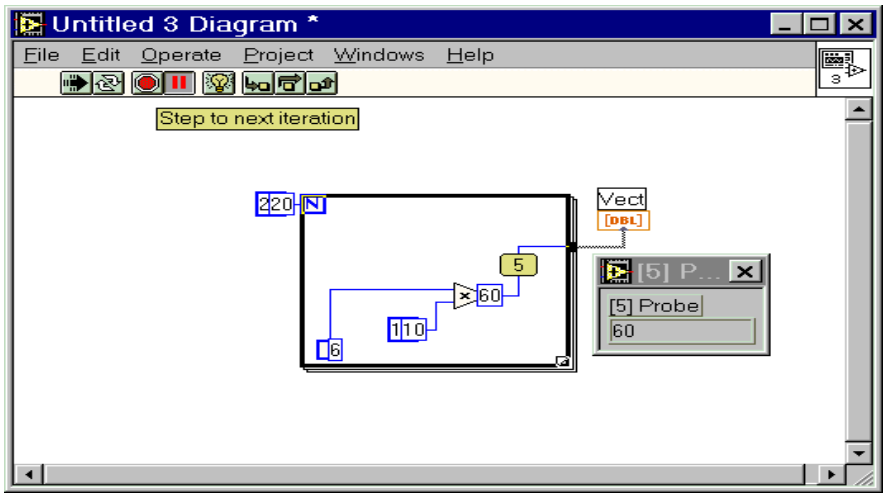


Fig. 39. Esecuzione highlighted, single step di un VI alla settima iterazione del ciclo For.

3.6 Help contestuale in linea

LabVIEW permette di associare a parecchi oggetti quali controllori ed indicatori o sub-VI posizionati nel pannello frontale una finestra di Help che descrive le caratteristiche dell'elemento in questione. Tale finestra viene attivata una volta per tutte selezionando la voce di menù *help/show help*, mentre viene aggiornata ogni volta che il mouse viene posizionato su un oggetto di LabVIEW (nodo, controllore, indicatore).

L'edizione dell'help di un controllore o un indicatore di un VI può avvenire selezionando l'opzione *Data operations/Description* dal menù pop-up di quell'oggetto mentre da un *Windows/show VI info* si può compilare l'help in linea per il VI corrente.

Le figure seguenti mostrano un esempio di quanto appena illustrato: la Fig. 40 illustra la modalità di accesso all'acceso all'help in linea, mentre la Fig. 41 illustra la finestra di help associata al nodo selezionato.

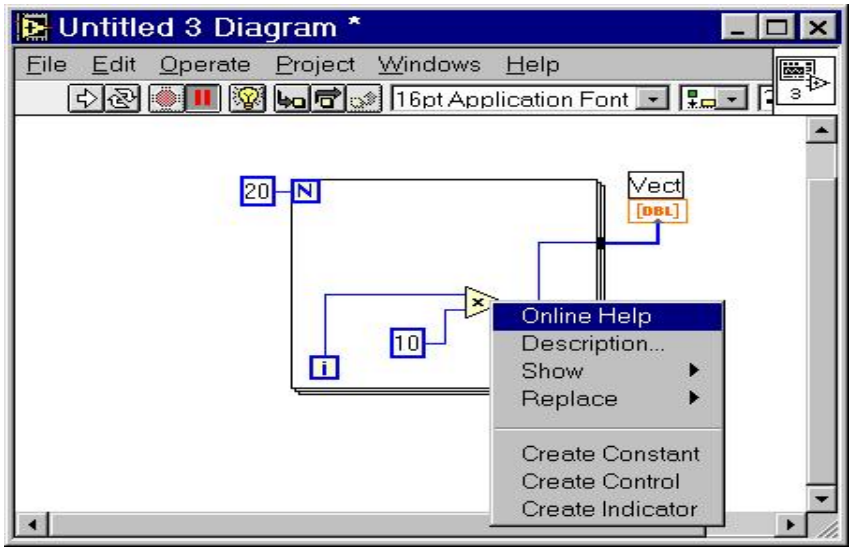


Fig. 40. Modalità d'accesso all'Help in linea.

Una volta selezionata la voce "online help" comparirà sul video una finestra di help contenente informazioni opportune associate al nodo selezionato. Nel caso di Fig. 40 si è selezionato il nodo di moltiplicazione; la corrispondente finestra di help è mostrata in Fig. 41.

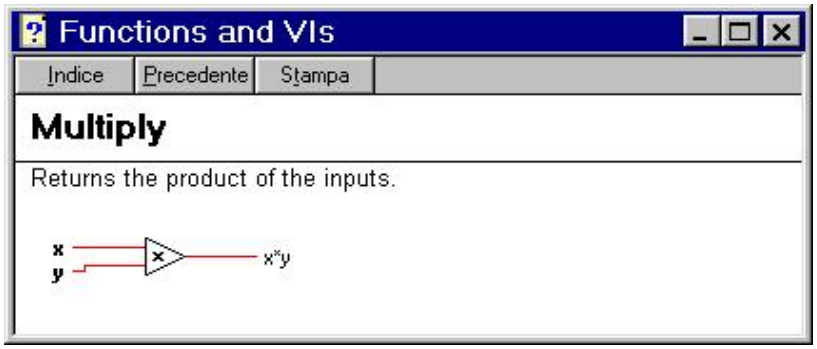


Fig. 41. Help associato al nodo di moltiplicazione mostrato nella fig.40.

Sono inoltre possibili ulteriori interazioni con la finestra dell'help on line. Ad esempio, posizionando la freccia del mouse sopra i controllori ed indicatori della finestra di help di

Fig. 41, comparirà la loro descrizione, ovvero il tipo di dato ed il range di valori ammessi.

4 Nozioni fondamentali sul G-language

Nei paragrafi precedenti si è visto come si presenta LabVIEW ad un utente, le modalità di interazione con esso e gli strumenti fondamentali per costruire un programma.

Nella presente sezione si analizzano regole e costrutti linguistici per la definizione del codice associato ad un VI e si richiamerà brevemente il significato semantico associato a tali costrutti.

4.1 Ordine di esecuzione di un block diagram

Si ricorda che l'ambiente LabVIEW tenta per quanto possibile di eseguire in parallelo tutti i nodi di un VI. Tuttavia un nodo può iniziare la propria attività solo se tutti i suoi elementi di ingresso (controllori) contengono dati.

Le comunicazioni tra i nodi di un VI fissano un ordine nell'esecuzione dei nodi che lo compongono nel senso che se due nodi sono interconnessi, quello che dei due fornisce dati di ingresso per l'altro viene eseguito per primo. La Fig. 42 illustra quanto detto. Nella figura è rappresentato il block diagram di un VI che somma due numeri A e B e ne memorizza il risultato in C: la figura illustra lo stesso block diagram in istanti successivi evidenziando il passaggio dei dati lungo i fili.

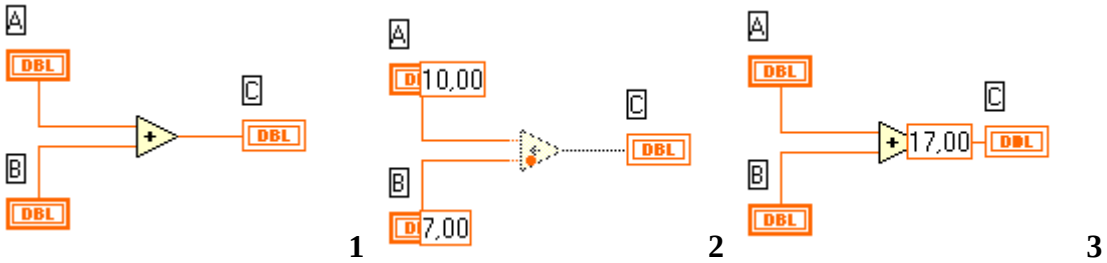


Fig. 42. Passaggio dei dati all'interno dei fili di un VI.

In Fig. 42 (1) i nodi corrispondenti ai controllori A e B vengono eseguiti in parallelo, provocando la lettura da pannello frontale di due valori numerici.

L'operazione di somma viene effettuata solo in Fig. 42 (2) quando entrambi i valori di ingresso (assegnati ai controllori x ed y) sono disponibili ai terminali del nodo di somma. Una volta effettuata l'operazione, il risultato viene scritto nella variabile di uscita.

Nel caso in cui siano presenti più nodi in parallelo il criterio di esecuzione è sempre lo stesso, nel senso che un nodo elabora i dati di ingresso solo quando tutti i valori sono disponibili ai suoi terminali.

Si consideri come esempio un VI che risolva l'espressione aritmetica $D = (A+B)*C$: il block diagram del VI che esegue tale operazione nell'ordine stabilito dalle parentesi è illustrato in Fig. 43, ove sono evidenziati i 4 passi dell'esecuzione.

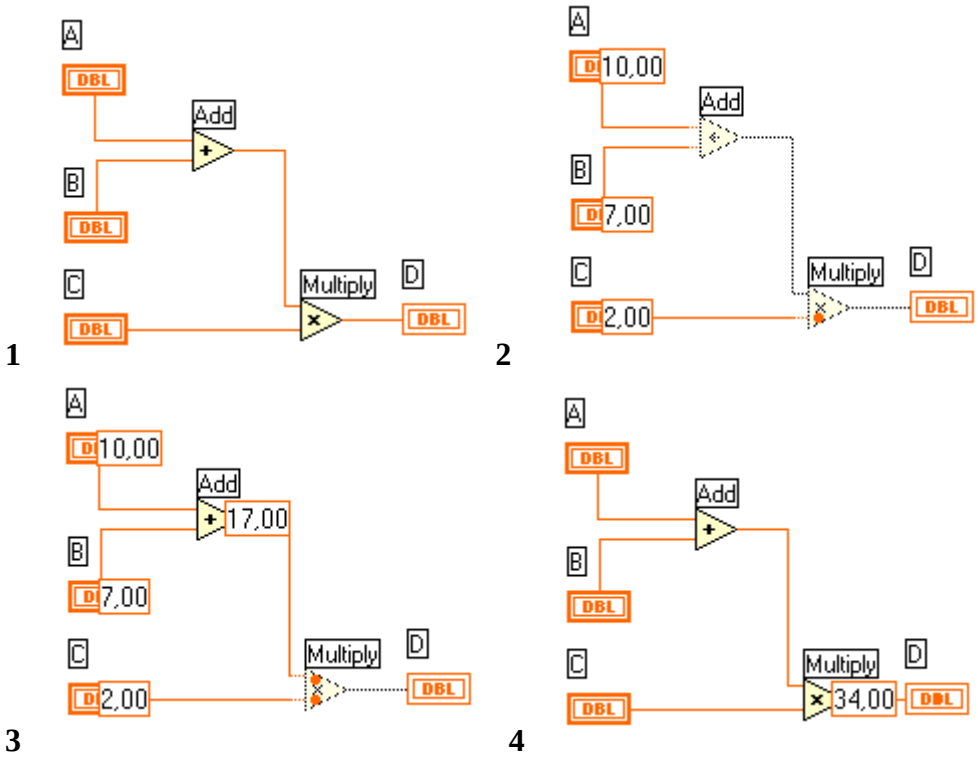


Fig. 43. Sequenza del passaggio dei dati in un VI che svolge l'operazione *D = (A+B)*C.*

Si noti come prima venga svolta l'operazione di somma e , solo quando tale risultato è disponibile, venga eseguita la moltiplicazione per la variabile di ingresso C: nell'ultimo passo il risultato viene scritto nell'indicatore di uscita D.

4.2 Principali tipi di dati

I tipi di dati messi a disposizione dall'ambiente LabVIEW sono raccolti nella *Controls Palette* e quindi resi disponibili nel pannello frontale e vengono descritti con maggior dettaglio nella presente sezione.

4.2.1 Dati di tipo numerico

I dati di tipo numerico sono raccolti nel menu pop-up ***Numeric*** della *functions palette*, illustrato in Fig. 44.

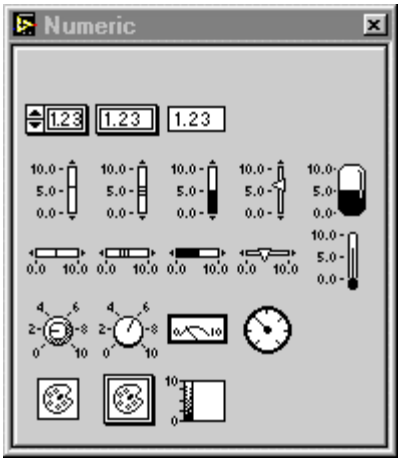
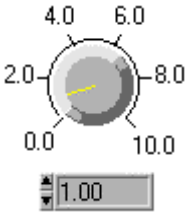


Fig. 44. Menu pop-up *Numeric*.

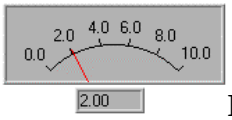
Tutti gli elementi raccolti nel menu *Numeric* servono per inserire (se controllori) o visualizzare (se indicatori) dei valori numerici.

Tra i controllori ed indicatori più frequentemente utilizzati si ricordano i seguenti:

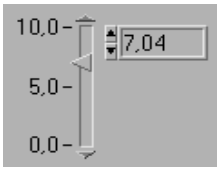
Digital control/indicator: sono i più semplici controllori/indicatori che consentono l'inserimento o la visualizzazione di quantità numeriche. Nel caso di un controllore, il valore visualizzato nel display può essere incrementato o decrementato agendo con la *operating tool* sulle due freccette poste all'estrema sinistra dell'elemento stesso. Naturalmente, è possibile modificare il valore numerico di ingresso scrivendo un nuovo valore o editando quello corrente nella casella di immissione prevista dal controllore.



Numeric Knob: la knob (manopola) viene normalmente usata come controllore e consente di modificare il valore della variabile numerica ad essa associata ruotandola avendo precedentemente selezionato il positioning tool utilizzando il mouse come se fosse una manopola usualmente presente in uno strumento reale. Per ottenere un controllo più fine si può interagire con il *digital control* associato alla manopola stessa il quale è una casella di interazione in tutto simile ad un numeric control.



Meter: è normalmente usato come indicatore ed è graficamente molto simile ai meter presenti negli strumenti analogici. Tra le possibili impostazioni dell'indicatore si ricordano la possibilità modificare il fondo scala e visualizzare sullo stesso display più valori di uscita.



Vertical slider: controllore (o indicatore) il cui valore cambia in funzione della posizione del cursore il quale può essere posizionato agendo con il mouse. Analogamente ad altri controllori è anche possibile impostare uno slider in modo che esso presenti un ulteriore digital control per un'interazione più fine.

4.2.2 Dati di tipo booleano

Un controllore o indicatore booleano può assumere solo due stati logici, indicati in LabVIEW con i simboli TRUE e FALSE. Gli elementi di tipo booleano sono raccolti nel menu pop-up ***Boolean*** della controls palette, riportato in Fig. 45.

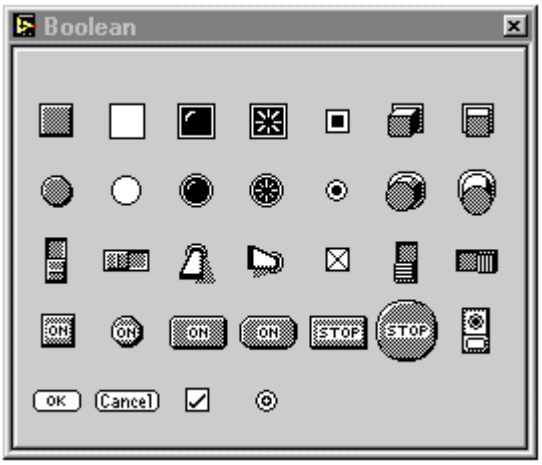


Fig. 45 Menu pop-up *Controls-Boolean.*

I controllori booleani simulano l'azione di pulsanti o interruttori; alcuni di essi sono riportati in Fig. 46.

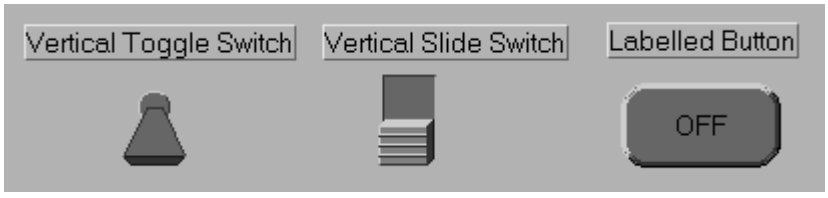


Fig. 46. Esempi di controllori booleani.

Gli indicatori booleani simulano led e indicatori luminosi, ed alcuni sono riportati in Fig. 47.

Fig. 47. Esempi di indicatori booleani.

Un'opzione particolarmente interessante e già descritta in precedenza, consente di modificare l'azione meccanica (*mechanical action*) di un controllore booleano per far sì ad esempio che il suo stato logico modifichi solo quando lo si fa commutare per mezzo della *operating tool*, oppure per far si che cambi di stato e ritorni successivamente allo stato originario, etc…

4.2.3 Dati di tipo stringa

I controllori e gli indicatori di tipo stringa consentono di inserire/visualizzare dati in formato ASCII. Gli elementi di tipo stringa sono raccolti nel menu pop-up ***String&Table*** della *controls palette*, riportato in Fig. 48.

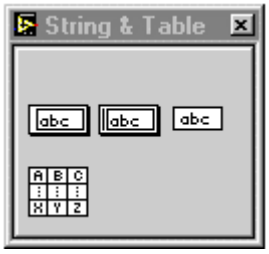


Fig. 48. Menu pop-up *String&Table.*

I due elementi più utilizzati sono Sting Control e String Indicator, illustrati in Fig. 49.

Fig. 49. Esempio di controllori ed indicatori del tipo String control & indicator.

È possibile inserire del testo all'interno di un display di tipo stringa utilizzando la *operating tool* o la *labelling tool* della tools palette. Durante l'esecuzione di un VI, per default il nuovo testo inserito viene passato al block diagram solo quando si preme il tasto *enter* oppure quando si seleziona con il puntatore del mouse un'area esterna al bordo della stringa.

4.2.4 Dati di tipo List&Ring

I controllori e gli indicatori di tipo *List&Ring* consentono di associare ad un numero intero non negativo (0, 1, 2) una stringa. Se usati come controllori forniscono un menu di scelta a tendina nel quale sono mostrate le stringhe impostate precedentemente e restituiscono in uscita un numero corrispondente alla stringa selezionata. Se usati come indicatori mostrano l'opzione (sotto forma di stringa) corrispondente al numero passato al loro ingresso. Si noti che è compito del programmatore gestire correttamente la corrispondenza tra numero e stringa.

Il menu pop-up per la selezione di un controllore/indicatore del tipo *List&Ring* è riportato in Fig. 50.

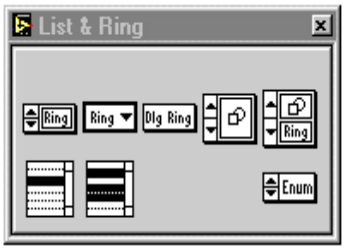


Fig. 50. Menù pop-up per la selezione di un controllore o indicatore del tipo List&Ring.

Gli elementi del tipo *List&Ring* sono particolarmente utili per la gestione di liste del tipo a menu oppure per forzare l'utente a compiere una scelta di uno tra alcuni elementi.

Una volta scelto un controllore o indicatore dal menù di Fig. 50 e dopo averlo posizionato nel front panel, LabVIEW lo inizializza con una lista vuota. Successivamente è possibile definire gli elementi della lista se si è impostata la modalità editing-mode e utilizzando la *labelling tool*. È anche possibile modificare la lista in run-time utilizzando un *attribute node.*

In editing mode si scrivono le stringhe direttamente all'interno del ring: per aggiungerne di nuove si seleziona l'opzione *Add Item After-Add Item Before* del pop-up menu associato all'elemento evidenziato. Un semplice esempio di ring è riportato in Fig. 51.

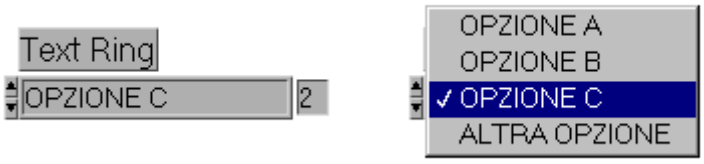


Fig. 51. Esempio di *ring.*

4.2.5 Dati di tipo Array&Cluster

Gli elementi di tipo array e cluster consentono di gestire collezioni di dati di dimensione variabile: essi sono raccolti nel menù pop-up *Array&Cluster* rappresentato in Fig. 52.

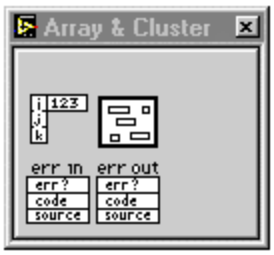


Fig. 52. Controllori ed indicatori del tipo *array&cluster*.

Un ***array*** è una collezione di dimensione variabile di elementi dello stesso tipo. Gli elementi di un array sono individuati da un indice compreso tra 0 ed *n-1* e si può accedere ad essi specificando l'indice relativo all'elemento che si vuole leggere da un array o sovrascrivere.

La costruzione di un array avviene posizionando nel front panel un array vuoto ed inserendo al suo interno un elemento prelevato dalla controls palette, utilizzando la consueta procedura di "drag&drop"

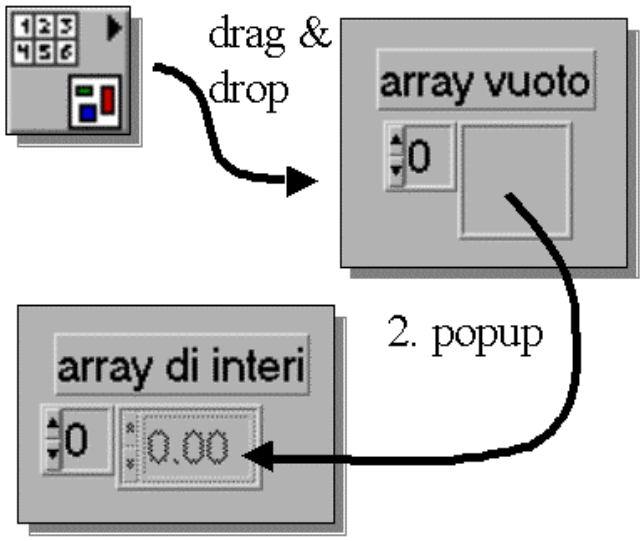


Fig. 53. Esempio di costruzione di un *array*.

In Fig. 53 si illustra graficamente la procedura per la costruzione di un array di controllori numerici: lo scorrimento degli elementi dell'array avviene selezionando le due freccette poste all'estrema sinistra dell'array stesso mentre il valore numerico dell'i-esimo elemento dell'array può essere modificato agendo con la *labelling tool* o con la *operating tool* nell'area interna dell'array.

È possibile ridimensionare la finestra associata all'array in modo da mostrare gli elementi desiderati, utilizzando la *positioning tool* e trascinando il bordo esterno che delimita l'array.

Una volta costruito, un array può essere collegato agli altri nodi presenti nel block diagram di un VI in maniera analoga a quanto possibile per gli altri tipi di variabili. Si noti che normalmente un array viene manipolato come se si trattasse di una struttura dati atomica. Ad esempio, lungo un connettore connesso ad un controllore del tipo array fluisce l'intero array, ovvero tutti i suoi elementi sono contemporaneamente trasferiti al nodo di destinazione.

Parecchi VI di libreria di LabVIEW inoltre supportano il polimorfismo includendo gli array tra i possibili elementi di I/O sovraccaricati. Ciò significa che è ad esempio possibile sommare tra loro gli elementi corrispondenti di due array aventi la stessa dimensione semplicemente collegando i due array all'ingresso di un VI di somma, o ancora è possibile sommare un dato scalare a tutti gli elementi di un array semplicemente collegando all'ingresso dell'usuale VI di somma il dato scalare e l'array.

Per accedere a un singolo elemento di un array è invece necessario ricorrere ad opportuni VI di libreria raccolti nel menu *Array* della functions palette. In tale menu si trovano ulteriori funzioni di utilità generale per la manipolazione di array.

L'utilizzo degli array nel block diagram è molteplice: essi possono ad esempio essere inizializzati mediante l'uso di cicli for, come illustrato in Fig. 54.

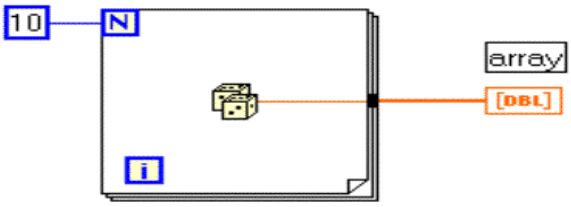


Fig. 54. Esempio di inizializzazione di un *array*.

Nell'esempio di Fig. 54 si costruisce un array formato da 10 elementi numerici casuali compresi tra 0 ed 1, forniti dalla funzione che genera numeri casuali e rappresentata dall'icona costituita da due dadi. È importante ricordare che per ottenere il vettore si deve selezionare l'opzione *enable indexing* dal menu pop-up che compare sul tunnel (filo in uscita dal ciclo for): si noti che il filo che collega tra loro dei vettori è più spesso rispetto ad un filo che unisce quantità scalari.

Utilizzando dei cicli innestati è possibile costruire e gestire delle matrici: in particolare si possono utilizzare due cicli, uno per indicizzare le righe ed uno per indicizzare le colonne. In Fig. 55 è illustrato un programma LabVIEW che costruisce una matrice 3x4, nella quale il valore associato all'elemento di posizione (i,j) è dato da $j + i*10$; si noti come il filo collegato alla matrice sia "doppio" rispetto agli altri.

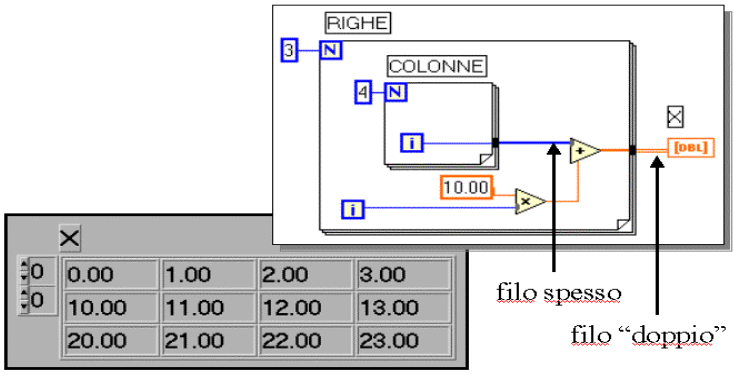


Fig. 55. Inizializzazione di una matrice 3x4.

I ***cluster*** sono dei tipi di dati che consentono di raggruppare più dati anche di tipo diverso: il loro utilizzo permette di costruire dei VI più ordinati in quanto evitano di trasportare lungo il block diagram quantità elevate di fili.

La procedura per la costruzione di un cluster è analoga a quella illustrata per la costruzione di un vettore: innanzitutto si deve effettuare una operazione di "drag&drop" di un cluster dal menu pop-up *controls/array&cluster* . Successivamente si posizionano all'interno della finestra del cluster gli elementi che lo dovranno comporre, come illustrato graficamente in Fig. 56.

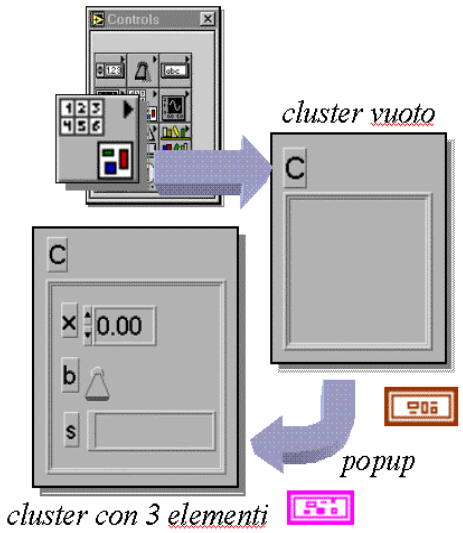


Fig. 56. Procedura per la costruzione di un cluster.

Nell'esempio di Fig. 56 viene creato un cluster composto da tre elementi rispettivamente di tipo numerico, booleano e stringa.

L'ordine di inserimento degli elementi nel cluster stabilisce anche l'ordine logico interno del cluster: nel caso dell'esempio di Fig. 56 il primo elemento del cluster è la variabile numerica, il secondo la variabile booleana, il terzo la variabile stringa.

LabVIEW fornisce due funzioni particolarmente utili nella gestione dei cluster: esse sono *bundle* e *unbundle*.

La funzione *bundle* riunisce in un cluster più collegamenti del block diagram.

La funzione *unbundle* decompone un cluster nei singoli elementi che lo definiscono. Tale funzione può essere utilizzata nel caso in cui sia necessario operare con un solo elemento appartenente al cluster, come illustrato in Fig. 57.

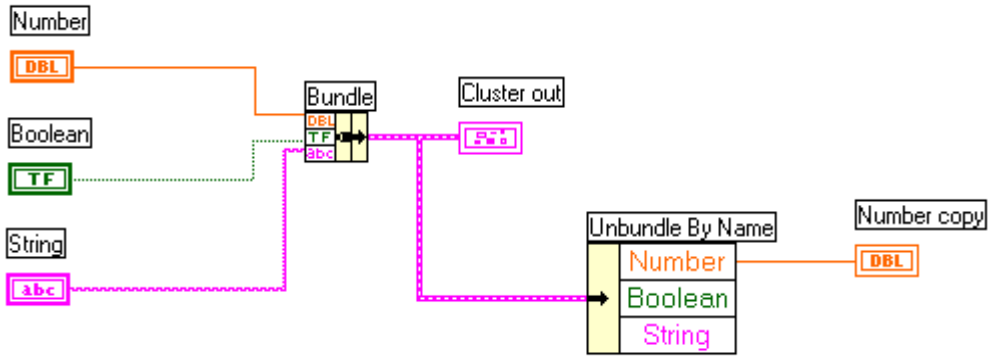


Fig. 57. Esempio di *bundle/unbundle* di un cluster.

4.2.6 Attribute Nodes

Un *Attribute Node* consente di modificare in run-time gli attributi di un controllore o indicatore . Ad esempio è possibile in run-time modificare il colore di un oggetto, renderlo invisibile da pannello frontale, farlo lampeggiare, etc… Se associato a un oggetto di tipo grafico, allora è possibile modificare molti parametri quali ad esempio i limiti delle scale della ascisse e delle ordinate.

La creazione di un *Attribute Node* avviene selezionando l'opzione *Create/Attribute Node* dal menu che compare effettuando un'operazione di pop-up su un nodo da pannello frontale o direttamente da block diagram, come illustrato graficamente in Fig. 58.

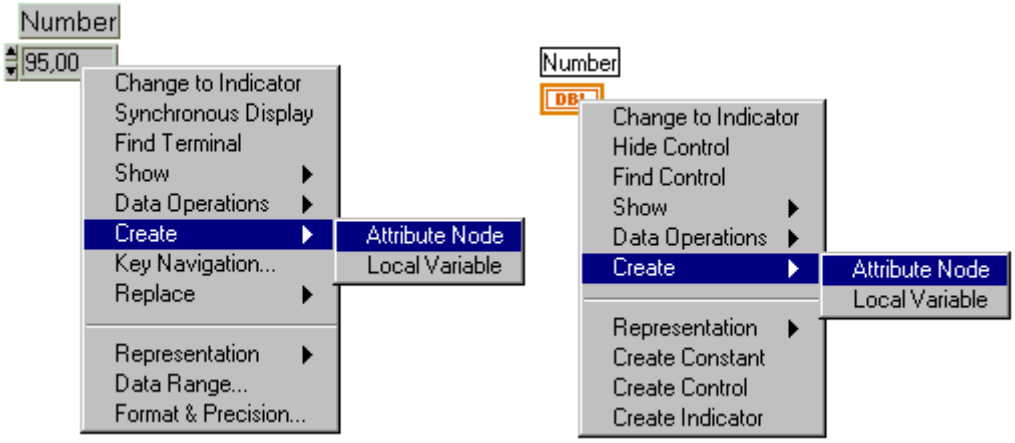


Fig. 58. Creazione di un *Attribute Node* da front panel e da block diagram.

Una volta selezionata l'opzione *Create/Attribute Node*, nel block diagram compare un nuovo nodo vicino a quello precedentemente selezionato, come mostrato in Fig. 59.

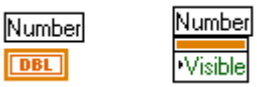


Fig. 59. *Attribute Node* associato al controllo *Number*.

Se la variabile da cui è stato creato l'*Attribute Node* ha un nome (ovvero una *label*) allora lo stesso nome sarà assegnato per default all'Attribute Node.

Aprendo il menu pop-up associato all'Attribute Node e selezionando la voce *Select Item*, compare una lista di tutti gli attributi che si possono scrivere o leggere dalla variabile, come illustrato in Fig. 60.

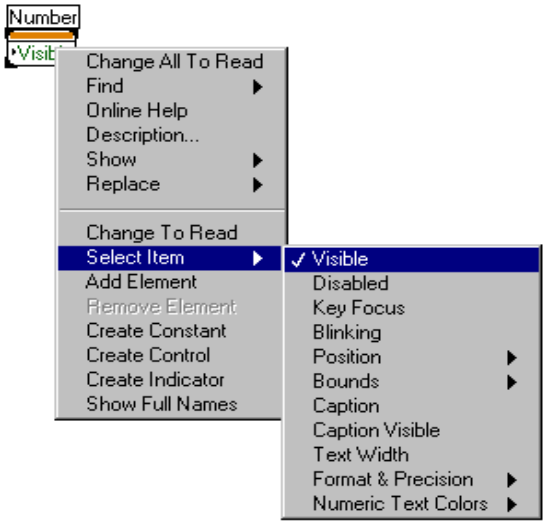


Fig. 60. Attribute Node/*Select Item.*

É possibile scegliere se leggere l'attributo dalla variabile o scriverlo all'interno della stessa selezionando l'opzione *Change To Read* oppure *Change To Write* dal pop-up menù dell'attribute node. Un *Attribute Node* di scrittura è evidenziato da una la freccia interna all'icona posizionata a sinistra, mentre lo stato di lettura è evidenziato da una freccia posta a destra, come illustrato in Fig. 61.

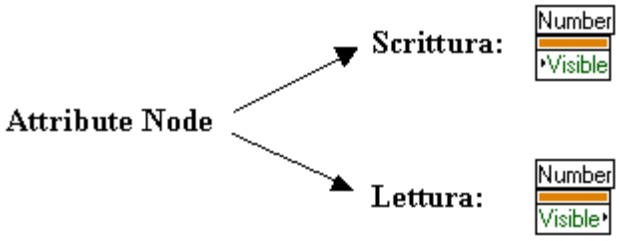


Fig. 61. *Attribute node* di scrittura e lettura

É possibile leggere o scrivere più A*ttribute Node* tutti associati a uno stesso nodo ridimensionando l'icona corrispondente all'attribute node stesso trascinandone il bordo esterno con il puntatore del mouse. Una volta ridimensionato si devono selezionare gli attributi che si vogliono associare al nodo in esame aprendo il menu pop-up associato ad ogni voce dell' *Attribute Node*, come in Fig. 62.

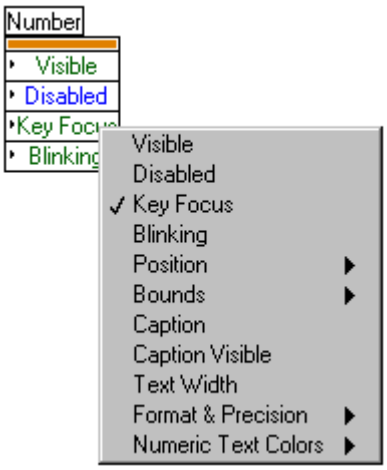
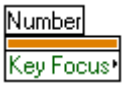


Fig. 62. Selezione delle voci associate ad un *Attribute Node.*

Gli *Attribute Node* principali a disposizione sono:

Visible attribute: rende invisibile da pannello frontale il nodo a cui è riferito (se false) oppure lo rende visibile (se true).

Disabled attribute: se è collegato ad una costante di valore 0, abilita l'utente ad effettuare modifiche in run-time sull'elemento a cui l'*Attribute Node* è associato; se la constante vale 1, eventuali modifiche da pannello frontale non hanno effetto, se la costante vale 2 l'oggetto a cui l'*Attribute Node* fa riferimento è disabilitato ed appare grigio.



Key Focus attribute: l'attribute node *Key Focus*, se true, consente di scrivere direttamente da tastiera un valore all'interno della variabile a cui esso è associato.

Blinking attribute: consente di far lampeggiare nel pannello frontale l'elemento a cui l'Attribute Node *Blinking* è riferito.

Position attribute: consente di leggere o impostare la posizione di un elemento presente nel pannello frontale

Particolare interesse assumono gli *Attribute Node* quando questi si riferiscono a grafici e diagrammi in quanto permettono di effettuare modifiche in run-time di alcuni parametri di presentazione dei dati, quali ad esempio i limiti dell'asse delle ascisse, delle ordinate, della modalità di aggiornamento del grafico, etc... Un esempio a proposito è riportato in

Fig. 63, ove è raffigurato un VI che genera una sequenza di numeri casuali compresi tra 0 ed 1 e li visualizza su un diagramma: i due controllori *Yminimum* e *Ymaximun* consentono di modificare in run-time il limite inferiore e superiore dell'asse delle ordinate del diagramma.

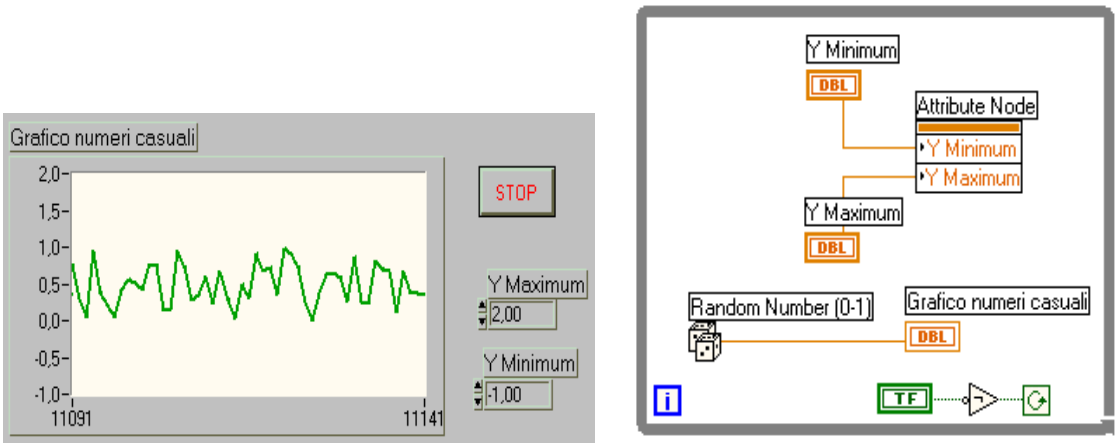


Fig. 63. Esempio di utilizzo di un *Attribute node* per l'impostazione dei limiti inferiore e superiore di un diagramma.

4.3 Strutture di controllo del flusso di esecuzione

Le *Structures* (strutture) sono costrutti grafici che consentono di controllare il flusso di esecuzione di un VI. Ogni struttura è delimitata da un bordo esterno che può essere ridimensionato mediante utilizzo della *positioning tool* . La *structure* esegue in una modalità che dipende dal tipo di struttura la porzione di block diagram inclusa entro il bordo che la delimita, come illustrato ad esempio per la struttura *Sequence* di Fig. 64.

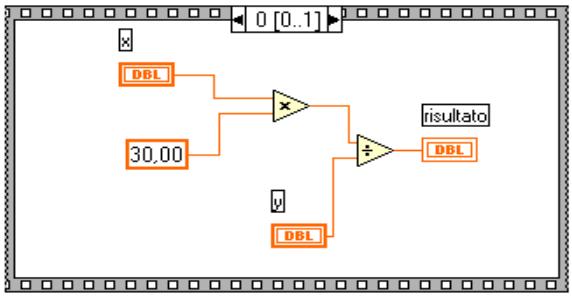


Fig. 64. Esempio di struttura *Sequence*.

Le strutture principali fornite dal G-language sono quattro:

- **For loop**: ripete l'esecuzione della porzione di codice in esso contenuta un determinato numero di volte

- **While loop**: ripete l'esecuzione della porzione di codice in esso contenuta fintanto che la variabile che lo controlla è true.
- **Case structure**: contiene molteplici sotto-diagrammi, di cui solo uno viene eseguito a seconda del valore passato al terminale selettore.
- **Sequence structure**: esegue il codice nell'ordine numerico dei suoi sotto-diagrammi.

Le strutture sono dei nodi e quindi hanno dei terminali che le collegano ad altri nodi; ad esempio i terminali che trasmettono i dati dentro e fuori le strutture sono chiamati *tunnel*.

4.3.1 Strutture For Loop e While Loop

Le strutture *For Loop* e *While Loop* sono utilizzate per il controllo di operazioni ripetitive.

La struttura *For Loop* esegue il diagramma contenuto nel rettangolo che la delimita per *i = 0,1,…,N-1:* essa è riportata in Fig. 65.

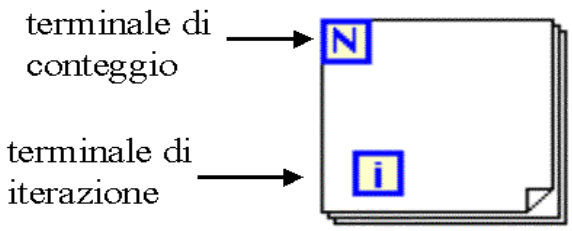


Fig. 65. Struttura *For Loop*.

La struttura *While Loop* esegue il diagramma contenuto nel rettangolo che la delimita finché al terminale di condizione non viene assegnato un valore booleano 'false', a ogni ciclo incrementa di 1 una variabile di conteggio inizialmente inizializzata a 0: la struttura *while loop* è illustrata in Fig. 66.

Fig. 66. La struttura *While Loop*.

Si consideri ora un semplice esempio di un VI che incrementa un contatore e ne mostra il valore finché non viene premuto un pulsante di stop: una possibile soluzione che fa uso di una struttura *while loop* è riportata in Fig. 67.

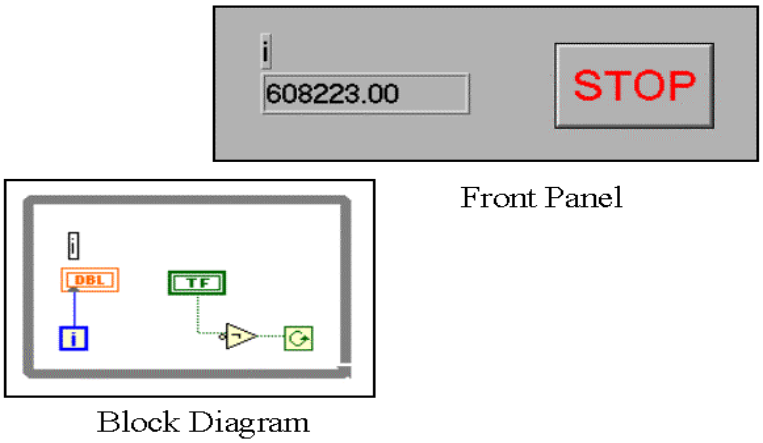


Fig. 67. Esempio d'uso di un ciclo *while.*

Con riferimento alla Fig. 67, la variabile contatore *i* viene incrementata fino all'istante in cui si preme il pulsante *stop.* Si noti che il valore associato al pulsante di stop viene controllato ad ogni iterazione del ciclo while: se tale pulsante fosse stato posizionato esternamente al bordo che delimita il ciclo *while*, il ciclo sarebbe proseguito all'infinito oppure sarebbe stato eseguito una sola volta, in base al valore assunto all'inizio del ciclo stesso.

LabVIEW mette a disposizione alcuni elementi che consentono di rendere più flessibile l'uso delle strutture *For Loop* e *While Loop* : esse sono gli *shift register,* le *local variable,* l'*auto indexing.*

Shift register

I registri a scorrimento (*shift register*) sono delle variabili locali che trasferiscono il valore assunto da una variabile a fine iterazione alla iterazione successiva. Uno *shift register* può essere creato selezionando la voce *Add Shift Register* dal menù pop-up che compare sul bordo esterno di una struttura *for loop* o *while loop*.

Uno shift register ha una coppia di terminali opposti posizionati sul bordo verticale della struttura. Il terminale di destra memorizza il valore assunto al termine di una iterazione dalla variabile ad esso collegata, mentre il terminale di sinistra rende disponibile tale valore all'iterazione successiva.

Entrambi i terminali di uno *shift register* devono essere collegati allo stesso tipo di dato; è possibile inoltre inizializzare uno *shift register* collegando al suo terminale di sinistra una costante.

Un esempio d'uso di uno *shift register* in un ciclo *for* è illustrato in Fig. 68.

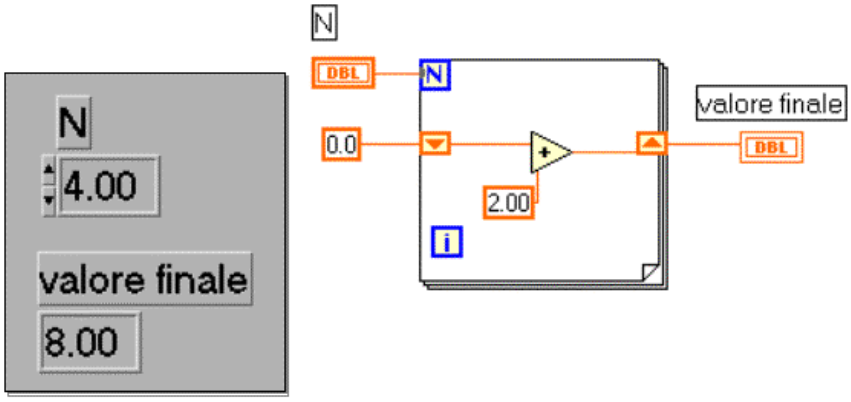


Fig. 68. Esempio d'uso di uno *shift register*.

Il VI di Fig. 68 permette di incrementare del valore 2 la variabile collegata all'ingresso dello *shift register*, cioè 2 per un numero di volte pari alla variabile N di controllo del ciclo *for*. Il VI di Fig. 68 esegue dunque una porzione di codice che in uno pseudo linguaggio di alto livello, simile al linguaggio C, potrebbe essere scritto come segue:

```
shift_reg = 0
for i = 0 to N-1
        shift_reg = shift_reg + 2
```

Local Variable

Una variabile locale (*local variable*) consente di compiere operazioni di assegnazione (lettura o scrittura) in un qualunque punto di un block diagram del valore associato ad un controllore o indicatore del front panel.

Scrivere un dato all'interno di una *local variable* equivale ad inserire il valore direttamente da pannello frontale; inoltre è possibile associare un numero qualsiasi di *local variable* ad un elemento del front panel.

Il modo più semplice per creare una *local variable* consiste nel posizionare il puntatore del mouse sopra l'elemento a cui si vuole associare la variabile locale, aprire il relativo menù pop-up e selezionare la voce *create local variable*. Una *local variable* apparirà automaticamente nel block diagram del VI. Il nome associato alla *local variable* sarà quello assunto dal controllore o indicatore a cui essa fà riferimento. Un esempio di tale procedimento è riportato graficamente in Fig. 69.

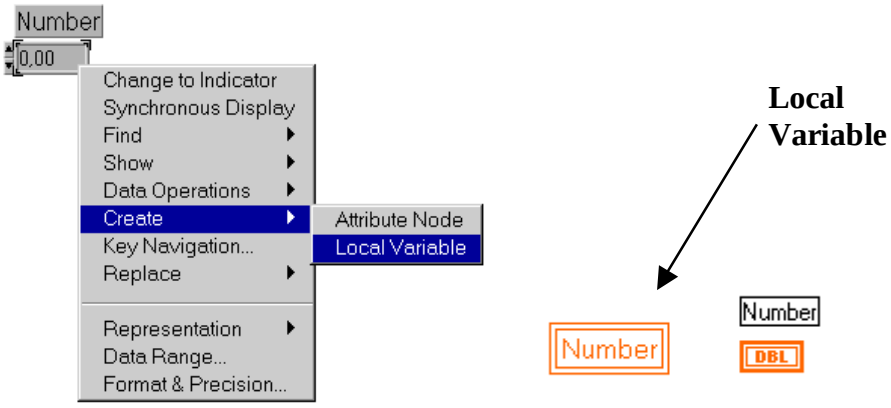


Fig. 69. Creazione di una *local variable.*

La *local variable* associata al controllore *number* è rappresentata da un rettangolo avente bordo doppio.

Le *local variable* sono utilizzate spesso all'interno di cicli per ottenere determinati comportamenti del VI: ad esempio si supponga di dover realizzare un VI che incrementa un contatore ogni 250 ms con la possibilità di fermare sia l'iterazione sia il mediante il pulsante *stop VI*.

Un possibile pannello frontale è illustrato in Fig. 70.

Fig. 70. VI di esempio dell'uso di una *local variable.*

In Fig. 71 si riportano i block diagram di due possibili soluzioni.

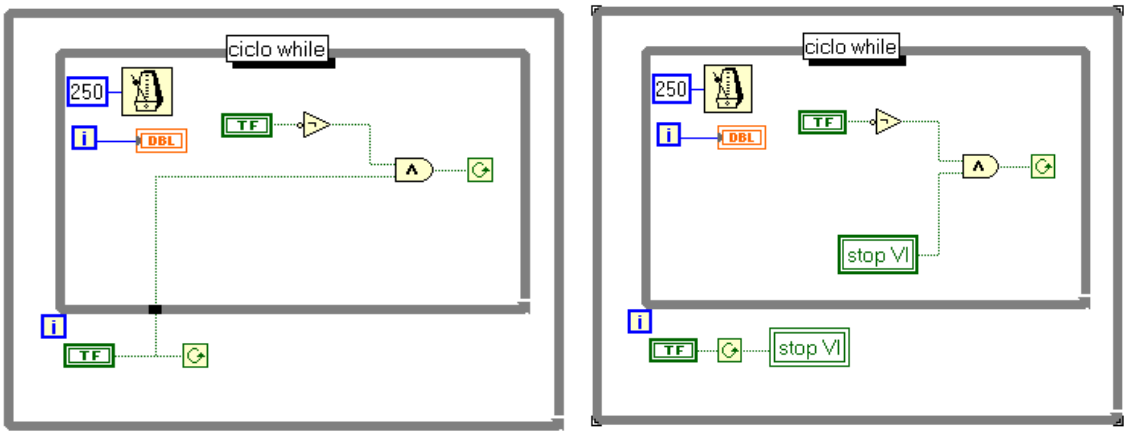


Fig. 71. Due possibili block diagram del VI di fig 70.

Con la prima soluzione proposta è possibile ottenere dal VI il comportamento voluto solo se il pulsante e l'interruttore vengono disattivati contemporaneamente. Infatti il valore di *stop VI* viene letto una sola volta all'inizio del ciclo più interno e non ad ogni iterazione. Tale soluzione ovviamente non è soddisfacente.

La seconda soluzione è quella corretta in quanto la *local variable "stop VI"* fa si che il valore assunto da *stop VI* sia disponibile anche nel ciclo più interno.

Auto indexing

Le strutture *for loop* e *while loop* consentono di indicizzare ed accumulare automaticamente dei valori all'interno di array: questa proprietà prende il nome di *auto-indexing*. Quando si collega un array dall'esterno verso l'interno di un ciclo e si abilita sul tunnel l'opzione di auto-indexing (mediante un'operazione di pop-up), i componenti del vettore sono assegnati alla porzione di codice interna al ciclo uno alla volta ad iniziare dal primo. Durante l'esecuzione del ciclo il vettore viene scomposto nei singoli elementi che lo compongono. In Fig. 72 si riporta un esempio di un VI che visualizza ogni 500 ms nell'indicatore numerico *number* il valore *vector[i]*.

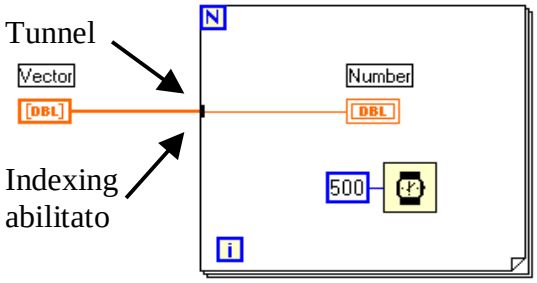


Fig. 72. Esempio d'uso dell'opzione *indexing* in ingresso ad un ciclo.

Nel caso in cui si colleghi un valore scalare dall'interno di un ciclo verso un vettore esterno, attivando l'opzione di *auto-indexing* (*enable indexing* dal menù pop-up associato), i vari valori scalari vengono accumulati nel bordo esterno della struttura per poi essere memorizzati nel vettore di uscita una volta terminata l'esecuzione del ciclo stesso.

4.3.2 Strutture Case e Sequence

La struttura *Case* permette l'elaborazione condizionata di sub-diagrammi a seconda del valore assunto da una variabile di controllo.

La struttura *Sequence* permette la sequenzializzazione dei sub-diagrammi contenuti nella struttura stessa.

Sia la struttura *Case* che la struttura *Sequence* possono avere un numero molteplice di sub-diagrammi, configurati in modo tale che uno solo di essi sia visibile alla volta.

Sul bordo superiore di queste strutture è visibile un identificatore e due bottoni di incremento/decremento. Nel caso della struttura *Case* l'identificatore indica il sub-diagramma che viene eseguito se la variabile di controllo assume il valore dell'identificatore visibile: l'identificatore può essere un valore booleano (Fig. 73) oppure numerico (Fig. 74), a seconda del tipo della variabile di controllo.

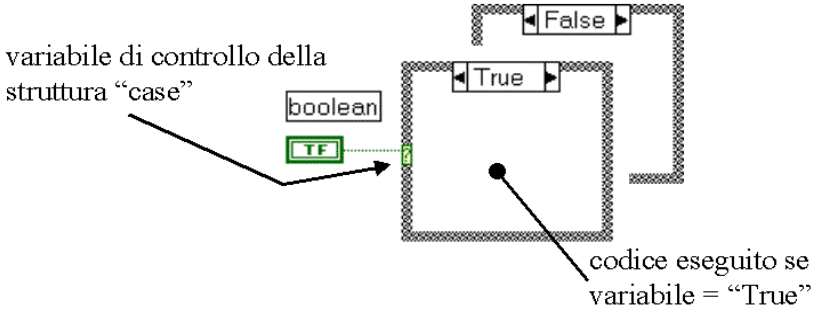


Fig. 73. Struttura *Case* controllata da una variabile booleana.

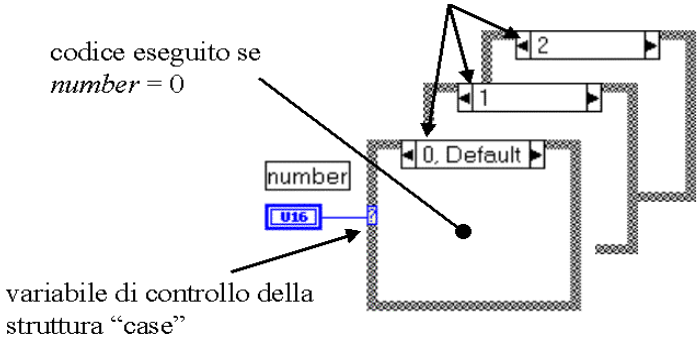


Fig. 74. Struttura *Case* controllata da una variabile numerica.

Nel caso della struttura *Sequence* l'identificatore rappresenta il numero d'ordine dei vari sub-diagrammi, chiamati *frames* , che compongono la struttura stessa: i *frames* appaiono sovrapposti l'uno all'altro ed è possibile cambiare il frame visualizzato agendo con la operating tool sulle freccette poste sul bordo superiore della struttura.

Una volta posizionata nel block diagram, una struttura *Sequence* presenta un solo frame di default: per aggiungerne altri si deve selezionare l'opzione "Add frame after/before" dal menù pop-up associato alla struttura stessa, come illustrato in Fig. 75.

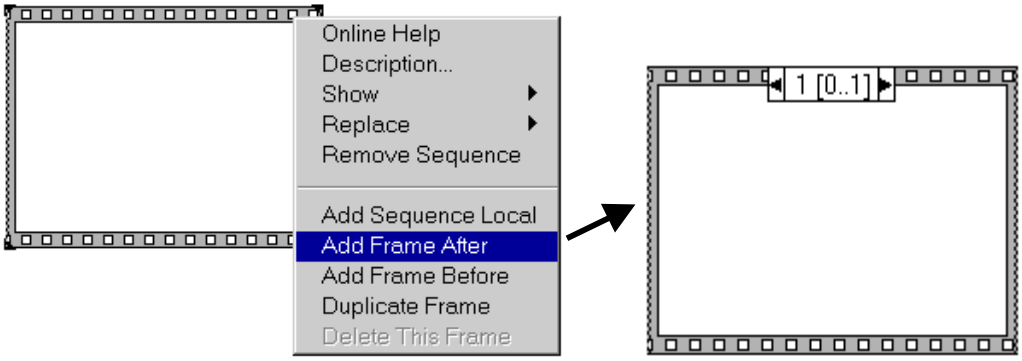


Fig. 75. Inserzione di un *frame* in una struttura *Sequence.*

Quando si utilizza una struttura *Sequence* può risultare utile rendere disponibile su più *frames* il valore assunto da una variabile appartenente ad un determinato frame: ciò è possibile selezionando l'opzione *Add sequence local* dal menu pop-up associato alla *Sequence*. Selezionando tale opzione, comparirà sul bordo esterno del frame corrente un simbolo grafico quadrato vuoto: una volta collegato un filo tra tale simbolo ed una variabile appartenente al frame corrente, all'interno del quadrato comparirà una freccia diretta verso l'esterno del frame. Tale freccia indica che per il frame corrente il valore collegato alla *Sequence local* è di uscita; in tutti gli altri frame la freccia sarà rivolta verso l'interno, ad indicare che il valore associato alla *Sequence local* è di ingresso per tali frame, come illustrato in Fig. 76.

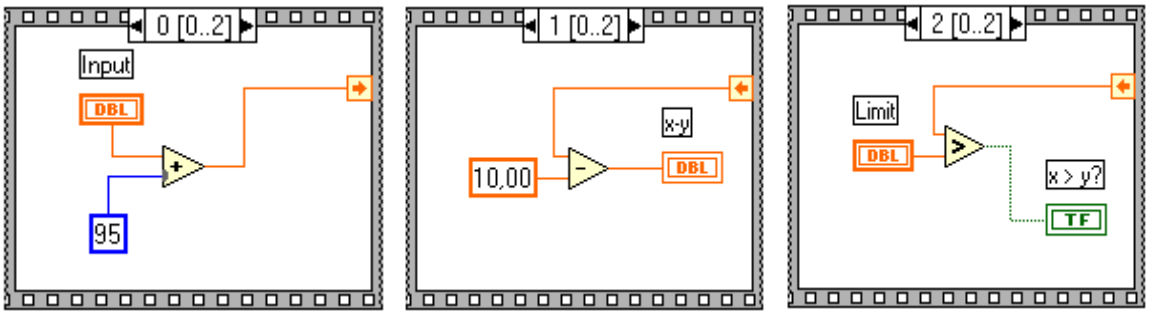


Fig. 76. Esempio di *Sequence local.*

Come si può notare dalla Fig. 76, l'uscita dal nodo di somma viene passata ai frame successivi per effettuare altre elaborazioni: si noti come il verso della freccia interna al quadratino che raffigura il *Sequence local* sia uscente per il frame 0 ed entrante per i frame 1 e 2.

Si illustrano ora due esempi di utilizzo delle strutture *Case* e *Sequence*: nel primo esempio si vuole costruire un VI che calcoli la radice quadrata di un numero e che fornisca nel front panel una segnalazione di errore nel caso in cui il numero di ingresso sia negativo. La soluzione proposta fa uso della struttura *case* e di alcune funzioni raccolte nella *functions palette:* il front panel è illustrato in Fig. 77.

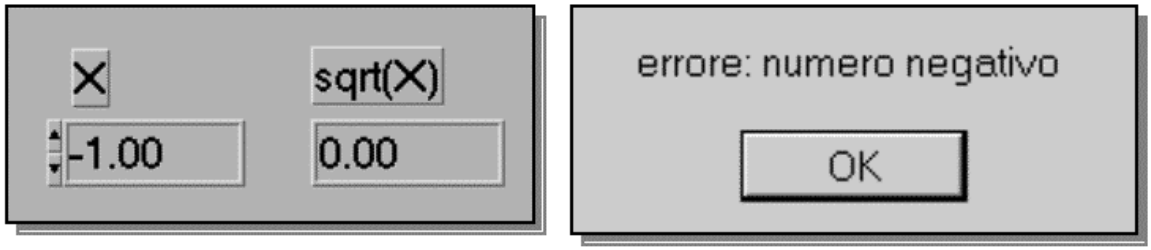


Fig. 77. Calcolo della radice quadrata di X - Front panel.

La seconda finestra di Fig. 77 è una segnalazione di errore che compare nel front panel solo nel caso in cui X sia un numero negativo; il block diagram del VI in esame è riportato in Fig. 78. Per facilitare la comprensione, il block diagram viene rappresentato due volte in figura, al variare della porzione di diagramma contenuta nella struttura *case*.

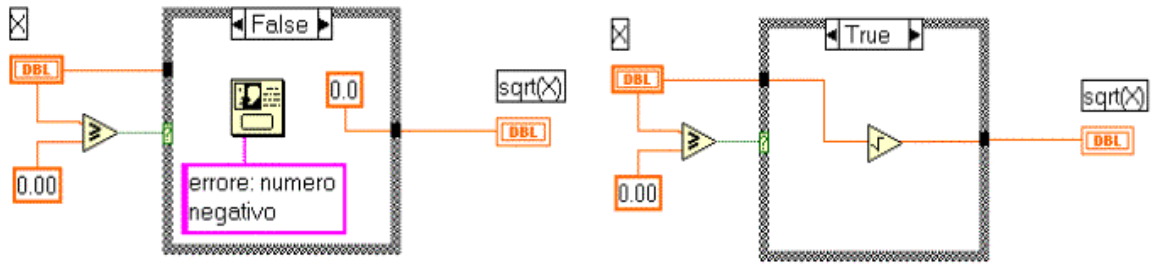


Fig. 78. Calcolo della radice quadrata di X - Block diagram.

Come si può notare dalla Fig. 78, nel caso in cui X sia negativo viene attivata una funzione di libreria che attiva una finestra "modale" nella quale compare il messaggio di errore, mentre se il numero è positivo ne viene calcolata la radice quadrata.

Il secondo esempio utilizza la struttura *Sequence*. Si desidera realizzare un VI che generi dei numeri casuali finché il numero casuale corrente non è uguale ad un valore preimpostato da pannello frontale. Il VI deve inoltre:

- terminare l'elaborazione se sono stati generati *N* numeri casuali (condizione di "timeout")
- calcolare il tempo totale di elaborazione.

Un possibile pannello frontale è illustrato in Fig. 79.

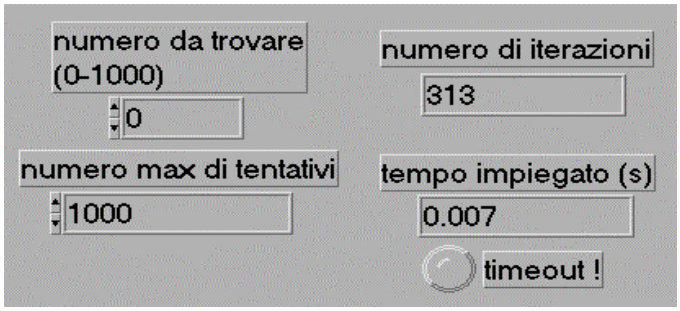


Fig. 79. Ricerca di un numero casuale: front panel.

La soluzione proposta utilizza una struttura *Sequence* formata da 3 *frame*, riportati in Fig. 80.

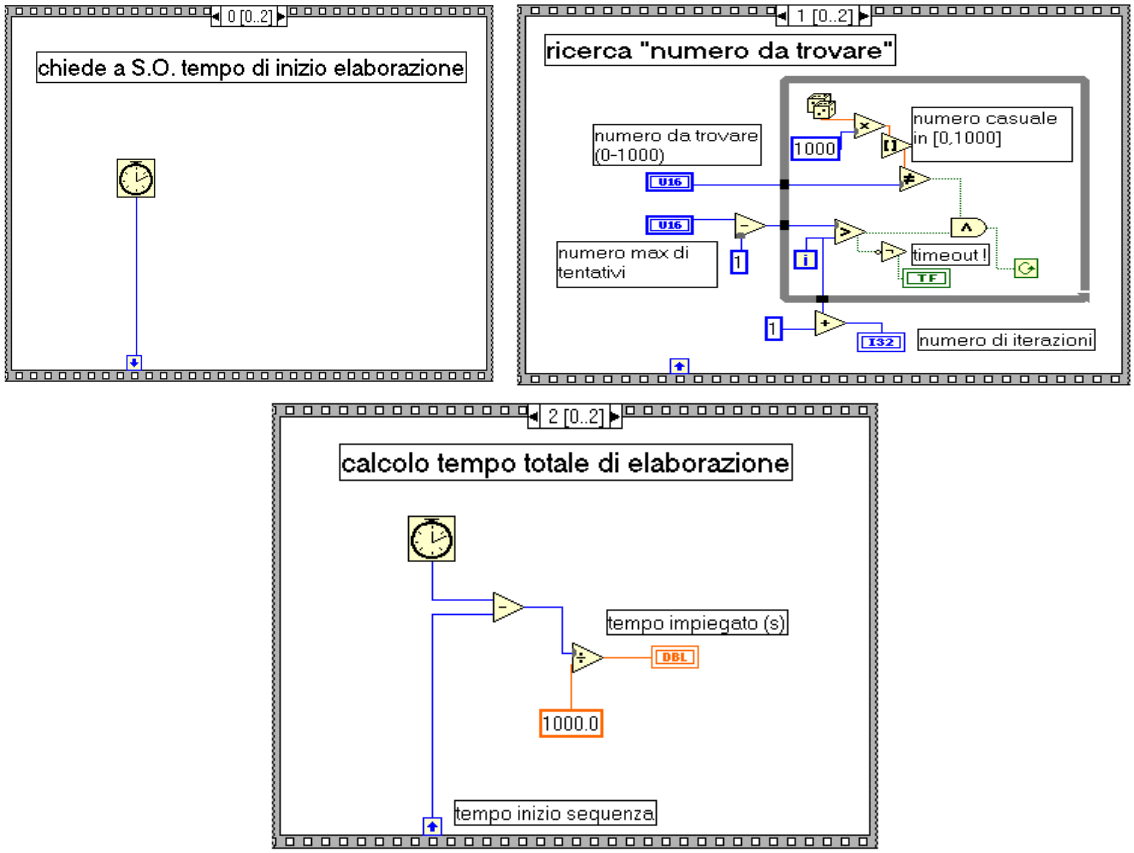


Fig. 80. Ricerca di un numero casuale: block diagram.

Con riferimento alla Fig. 80 nel primo *frame* della struttura *Sequence* viene richiesta al sistema operativo l'ora corrente, nel secondo si utilizza un ciclo *while* per generare i numeri casuali, confrontare i numeri generati con quello fornito da pannello frontale, controllare il numero di iterazioni.

Nel terzo ed ultimo *frame* si preleva dalla *Sequence local* il valore del tempo di inizio elaborazione e si calcola il tempo totale di elaborazione, espresso in s.

4.3.3 La struttura Formula Node

Una struttura *Formula Node* permette di scrivere al suo interno una serie di formule matematiche ed espressioni aritmetiche, separate dal carattere ";". La sintassi usata per la definizione delle espressioni è la stessa di molti linguaggi di programmazione text-based; inoltre possono essere inseriti dei commenti, racchiusi da una coppia di delimitatori '/*' …. '*/' come nel seguente esempio /* Stringa di commento */.

Le variabili di I/O di una struttura *Formula Node* sono definite mediante operazioni di pop-up sul bordo della struttura stessa. In particolare, la voce di menu *Add input*, permette di aggiungere una variabile di ingresso, mentre *Add output* consente di aggiungere un variabile di uscita.

È inoltre consentito convertire una variabile di ingresso in una variabile di uscita (o viceversa) aprendo il pop-up menu associato alla variabile desiderata e selezionando l'opzione *Change to Output/Input* (le variabili di output sono riconoscibili da quelle di input perché hanno un bordo più spesso).

I terminali creati sul bordo della struttura possono essere collegati ad altri oggetti con le consuete tecniche di "wiring".

Si consideri come esempio un VI che calcola il baricentro di tre punti lungo una retta parallela ad un asse coordinato, note le coordinate dei punti rispetto a tale asse. Una possibile soluzione è illustrata in Fig. 81.

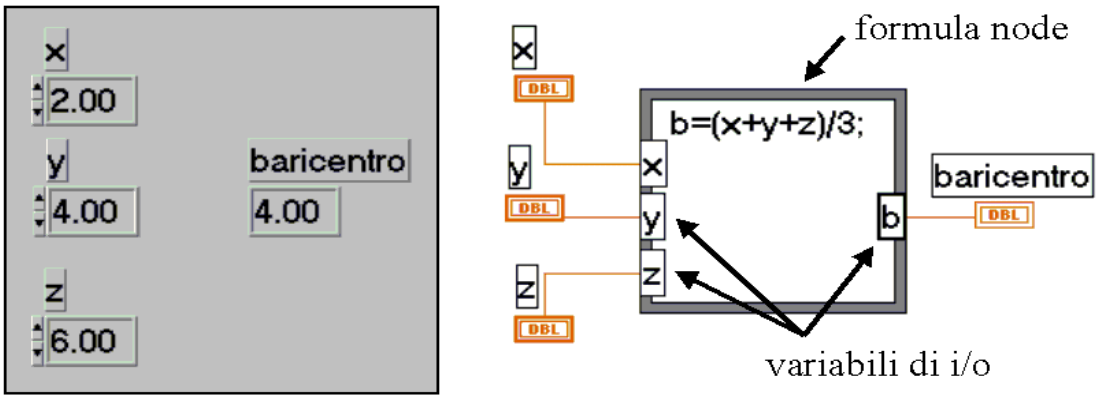


Fig. 81. Calcolo di una coordinata del baricentro di 3 punti mediante struttura *Formula Node.*

Le variabili di ingresso sono *x,y* e *z* mentre il risultato è reso disponibile nell'uscita *b*.

4.4 Modalità di rappresentazione grafica dei dati

I dati di uscita da un VI possono essere rappresentati sotto forma di singoli valori numerici, di vettori, oppure sotto forma di grafici: quest'ultima modalità di rappresentazione consente di fornire VI aventi un aspetto molto simile a strumenti del tipo oscilloscopi, analizzatori di spettro, etc…

Il G-language raccoglie gli elementi di rappresentazione grafica nel menu pop-up **Graph** della *Controls palette*, riportato in Fig. 82.

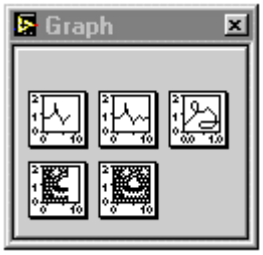


Fig. 82. Controllori e indicatori di tipo *Graph.*

In totale sono disponibili 3 tipi di grafici (*Graph*) e due tipi di diagrammi (*Chart*).

I diagrammi (*Waveform Chart*) visualizzano i dati "punto a punto", nel senso che ogni volta che giunge un nuovo campione al *Waveform Chart* esso viene visualizzato sul display, mantenendo un certo numero di valori precedenti in un buffer.

I grafici (*Waveform Graph*) appaiono identici ai diagrammi, ma vengono costruiti in un unico momento a partire da un vettore.

Utilizzando la funzione *build array* è possibile costruire grafici e diagrammi che rappresentino contemporaneamente più serie di dati.

Nel seguito si riportano le principali modalità di interazione con oggetti del tipo *graph* e *chart*.

4.4.1 Waveform Chart

Come già accennato, un *Waveform Chart*, a differenza di un *Waveform Graph*, riceve in ingresso singoli punti o blocchi parziali di dati e *non* interi vettori. In questo modo è possibile vedere il valore attuale di una variabile e confrontarlo con quelli precedenti (il massimo numero di dati visualizzabili è predisposto dall'utente).

È possibile passare ad un *Chart* sia valori scalari che vettoriali; inoltre è possibile realizzare grafici multitraccia.

Ad esempio, si voglia costruire un VI che generi una sequenza di numeri casuali e la visualizzi su un *Chart* fino a quando non si prema un pulsante di stop: una possibile soluzione è riportata in Fig. 83.

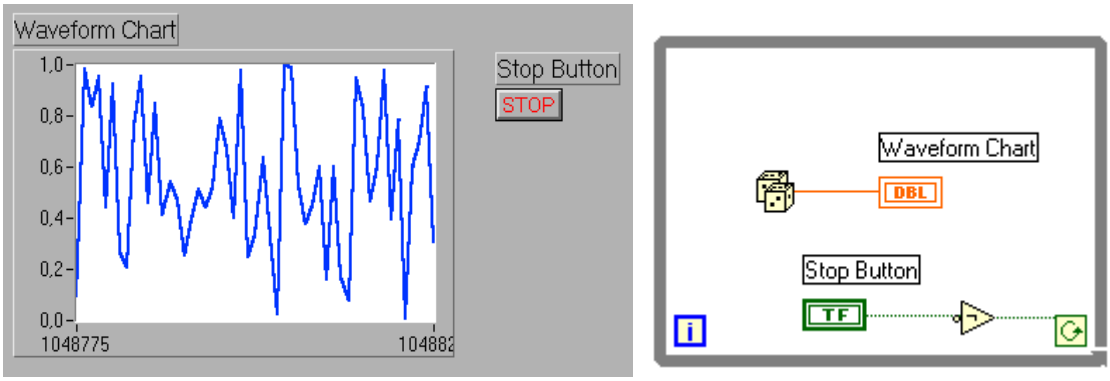


Fig. 83. Grafico di numeri casuali: Front panel e Block diagram.

La funzione che genera i numeri casuali è attiva finché non viene premuto il tasto *Stop Button*. I singoli valori generati vengono riportati nel *chart* uno ad uno.

È possibile creare dei *Waveform Chart* multitraccia: a proposito si consideri l'esempio di Fig. 84.

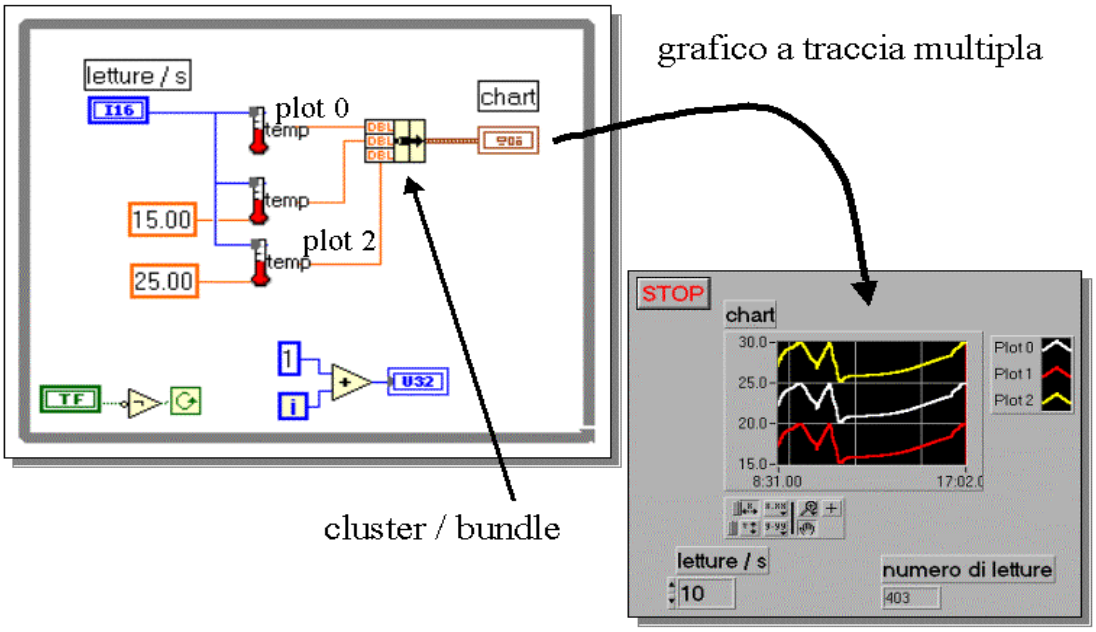


Fig. 84. Esempio di *Waveform Chart* multitraccia.

In Fig. 84 sono illustrati il pannello frontale ed il diagramma a blocchi di un VI che genera contemporaneamente 3 valori diversi di temperatura e li visualizza su un diagramma: la visualizzazione multitraccia è ottenuta grazie alla funzione *bundle* che crea un cluster formato da 3 valori il quale, una volta collegato al *waveform chart*, viene rappresentato graficamente sotto forma di 3 punti. Il *Waveform Chart* aggiorna di volta in volta i valori visualizzati aggiungendo i 3 nuovi punti di ingresso, costruendo progressivamente tre curve come ad esempio in Fig. 84.

4.4.2 Waveform Graph

Un *Waveform Graph* consente di tracciare un grafico collegando al suo ingresso un array: LabVIEW riporta in ordinata l'ampiezza dei singoli dati del vettore in funzione dell'indice.

L'aspetto assunto da un *Waveform Graph* nel pannello frontale è identico a quello assunto da un *Waveform Chart*, dal quale differisce in quanto costruito in un unico momento a partire da un vettore.

Si consideri come esempio un VI che riporta su un *Waveform Graph* dei valori di temperatura generati da un simulatore: una possibile soluzione è riportata in Fig. 85.

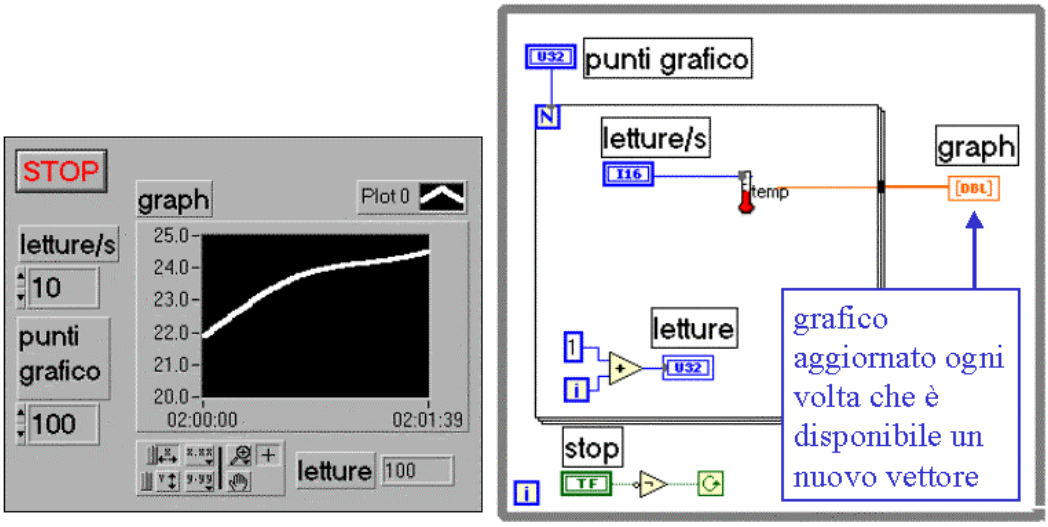


Fig. 85. *Waveform Graph* che visualizza dei valori di temperatura.

Da pannello frontale è possibile impostare il numero di letture della temperatura effettuate al secondo ed il valore "punti grafico": una volta che il VI ha generato il numero di campioni di temperatura specificato da "punti grafico", il *Waveform Graph* viene aggiornato. Tutta la procedura descritta si ripete sino a quando non si preme il bottone *stop*.

Anche con i *Waveform Graph* è possibile visualizzare più serie di dati utilizzando la funzione "*build array*", fornita dalla *functions palette*.

Un esempio che rappresenta un'estensione di quello presentato in Fig. 85 è illustrato in Fig. 86: in esso vengono rappresentate su un grafico le letture provenienti da due termometri simulati costruendo con la funzione build array un vettore bidimensionale dati i due vettori monodimensionali delle temperature di uscita.

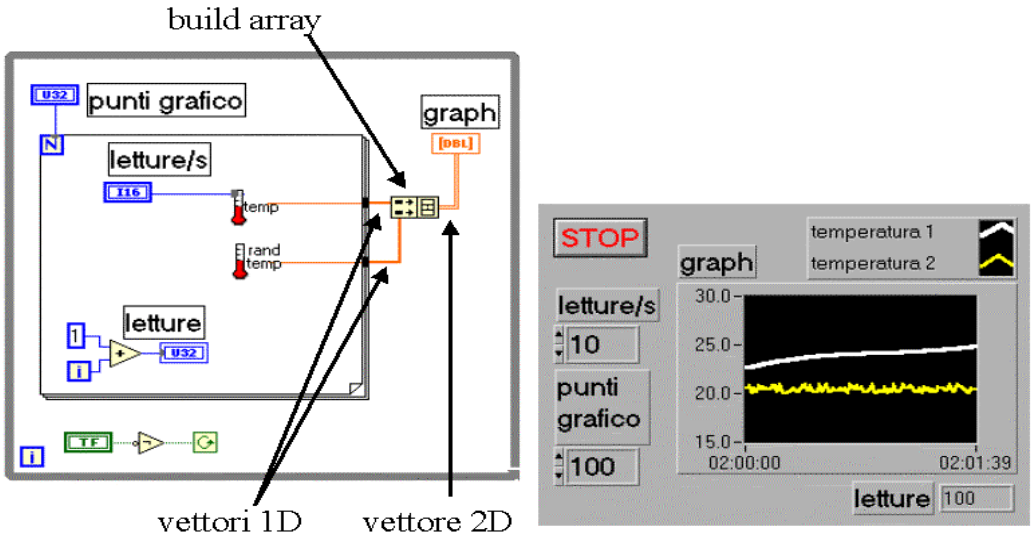


Fig. 86. *Waveform Graph* che visualizza su uno stesso grafico valori di temperatura provenienti da due sorgenti.

4.4.3 XY Graph

I grafici di tipo XY accettano in ingresso cluster formati da array numerici. Si può pensare di associare al primo array i valori dell'asse delle ascisse ed al secondo array i valori dell'asse delle ordinate.

Un esempio significativo di impiego di un *XY Graph* è fornito da un VI che riporta su un *XY Graph* una porzione di circonferenza ottenuta mediante calcolo delle coordinate x-y dei punti che la compongono. I due vettori ottenuti vengono raggruppati in un cluster mediante la funzione "build cluster" prima di essere passati al grafico, come illustrato in Fig. 87.

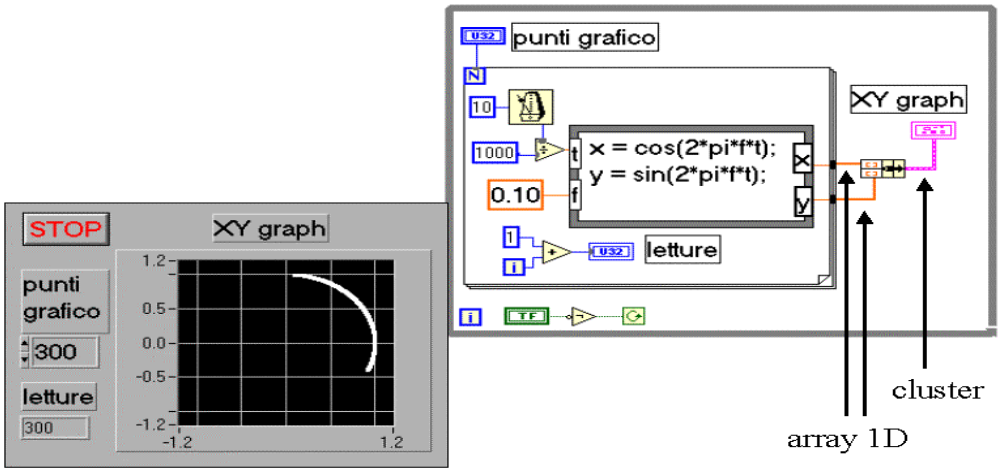


Fig. 87. Esempio di utilizzo di un *XY Graph* monotraccia.

Nel caso in cui si vogliano visualizzare sullo stesso grafico di tipo XY più tracce, è necessario creare un array di cluster utilizzando prima la funzione *bundle* e poi la funzione *build array*, come illustrato nell'esempio di Fig.88.

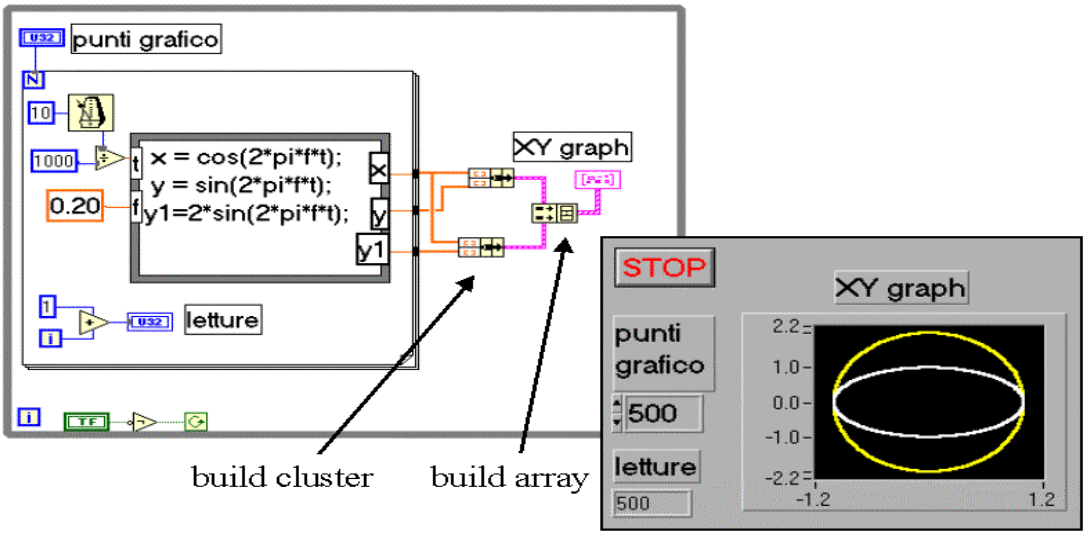


Fig. 88. Esempio di utilizzo di un indicatore *XY Graph* multitraccia.

5 Principali funzioni di libreria

LabVIEW mette a disposizione dell'utente parecchie funzioni di base che consentono di effettuare elaborazioni su dati numerici, stringhe, vettori e di gestire strumenti collegati ad un bus 488, di gestire una comunicazione seriale o attraverso Internet.

Tutte le funzioni sono raccolte nella **Functions Palette** disponibile nel block diagram: le principali sono elencate e brevemente illustrate qui di seguito.

5.1 Numeric functions

Il pop-up menu che raccoglie le funzioni che operano su dati di tipo numerico è illustrato in Fig.89.

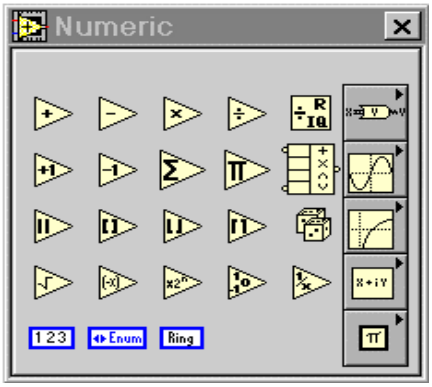
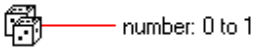


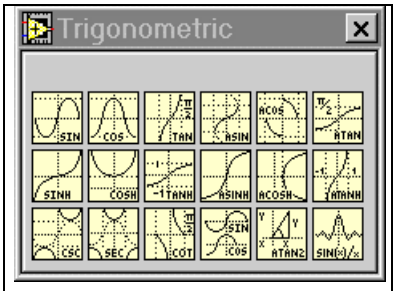
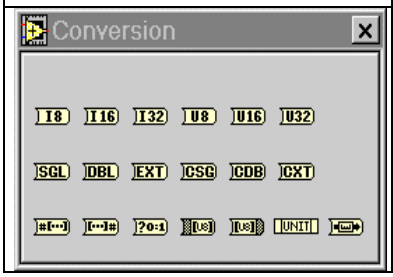
Fig.89. *Numeric functions.*

.

In esso sono raccolte funzioni che consentono di effettuare operazioni elementari sui dati di ingresso (somma, sottrazione, divisione, …), di effettuare operazioni logaritmiche, trigonometriche e conversioni numeriche. Sono inoltre disponibili le principali costanti matematiche fondamentali.

Le funzioni principali e più utilizzate sono raccolte nella seguente tabella.

	**Add:** calcola la somma dei due valori di ingresso x ed y (tutte le altre funzioni elementari come sottrazione, moltiplicazione e divisione sono identificate dalla stessa icona di *add* ma con il simbolo grafico che le rappresenta).
	**Reciprocal:** calcola il reciproco di x.
	**Square root**: calcola la radice quadrata di x.
	**Random number (0-1)**: genera un numero casuale compreso tra zero ed uno.
	**Absolute value**: calcola il valore assoluto della variabile di ingresso.

	**Trigonometric**: raccolta di funzioni trigonometriche che consentono di calcolare seno, coseno, tangente, secante, cosecante, seno e coseno iperbolico, etc…
	**Conversion functions:** raccolta di funzioni che consentono di effettuare conversioni di tipo sui dati d'ingresso: ad esempio conversioni da intero a 16 bit a intero a 8 bit, etc… Si noti che parecchie tra le precedenti funzioni supportano la conversione matematica tra dati di ingresso. Alcune consentono inoltre di manipolare vettori di numeri.

5.2 String functions

Il pop-up menù che raccoglie le funzioni che operano su dati di tipo stringa è illustrato in Fig.90.

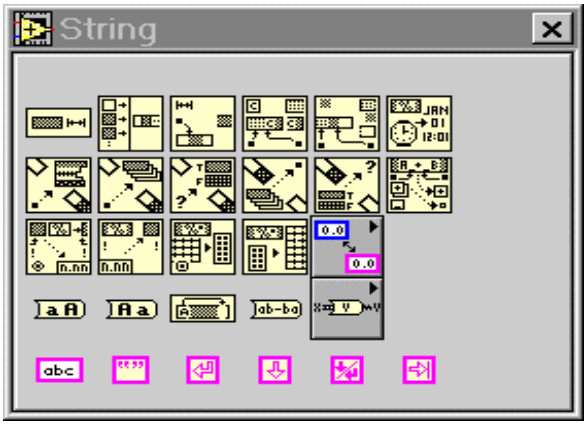
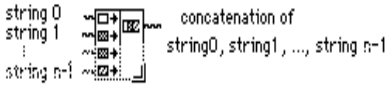
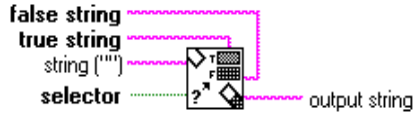
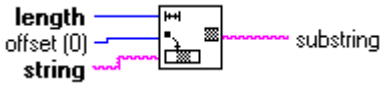


Fig. 90. Funzioni di tipo stringa.

Le funzioni raggruppate nel menu di Fig.90 consentono di effettuare vari tipi di elaborazioni sulle stringhe, come ad esempio conversioni da stringa a valore numerico e viceversa, concatenazione di più stringhe in una unica, scansioni su una sequenza di caratteri, etc…

Tra queste le funzioni più utilizzate sono indicate nella seguente tabella.

	**String lenght**: fornisce in uscita la lunghezza della stringa di ingresso pensata come un vettore di byte.
	**Concatenate Strings**: concatena le stringhe di ingresso ed un array mono-dimensionale di stringhe in un'unica stringa di uscita.
	**Select&Append**: concatena la stringa collegata al terminale *string* con la stringa collegata al terminale *false* o con quella collegata al terminal *true* in base al valore assunto dalla variabile booleana *selector.*
	**String Subset:** restituisce la sotto-stringa ottenuta dalla stringa di ingresso a partire da *offset* e di lunghezza pari al valore specificato dal parametro *lenght.*
	**Index&Append**: seleziona la index-esima stringa del vettore *string array* e poi la concatena a *string.*
	**Index&Strip**: confronta ogni stringa presente in *string array* con *string* finché trova una corrispondenza.

Esiste poi un ulteriore sottomenu appartenente al menu pop-up illustrato in Fig.90, il quale raccoglie molte funzioni di conversione di valori numerici in stringa e viceversa, ed è denominato *Additional String to Number Functions.*

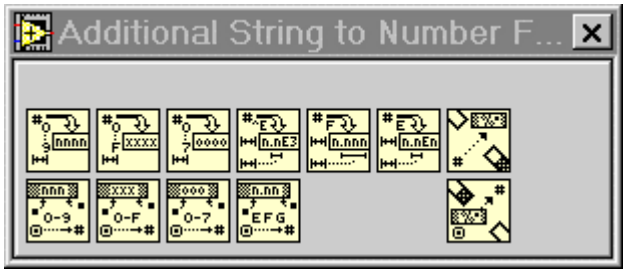
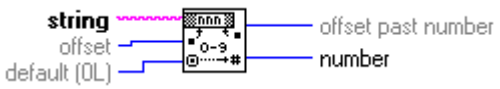
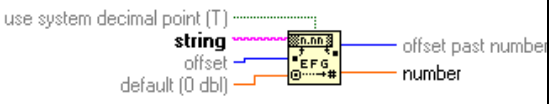
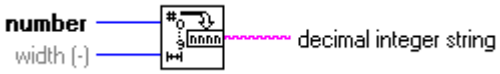
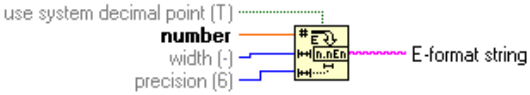


Fig.91. *Additional String to Number Functions.*

Tra le funzioni di tale sottomenu se ne ricordano alcune di particolarmente utili:

	**From decimal**: converte i caratteri numerici della variabile *string*, iniziando da *offset*, in un numero intero decimale.

	**From exponential/fract/eng:** interpreta i caratteri da 0 a 9, "+", "-" , "e", "E" ed il punto decimale inseriti nella stringa di ingresso a partire da *offset* e li converte in un numero.
	**To decimal:** converte il valore numerico *number* in una stringa di cifre decimali lunga *width* caratteri.
	**To exponential**: converte il valore numerico *number* in una stringa lunga *width* caratteri ed espressa in formato ingegneristico.
	**String to byte array:** converte la stringa di ingresso in un array di byte.

5.3 Boolean functions

Il pop-up menu che raccoglie le funzioni che operano su dati di tipo booleano è illustrato in Fig.92.

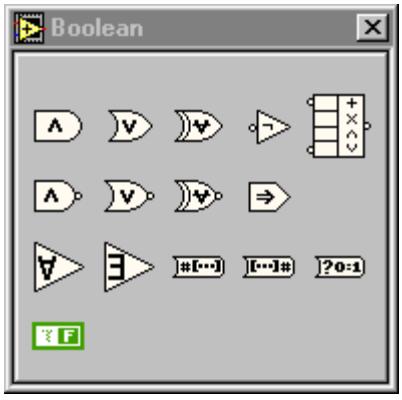
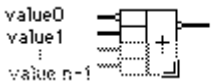


Fig. 92. Funzioni di tipo booleano.

Le funzioni raggruppate nel menù di Fig.92 consentono di effettuare vari tipi di elaborazioni su variabili di tipo booleano, come ad esempio operazioni logiche di AND, OR, NOT, XOR che consentono di gestire ad esempio situazioni in cui si debba scegliere tra più opzioni in base al valore assunto da più variabili di controllo.

Le funzioni più utilizzate sono:

	**AND:** effettua l'AND logico tra le variabili d'ingresso.
	**OR**: effettua l'OR logico tra le variabili d'ingresso.
	**NOT**: effettua la negazione della variabile d'ingresso.
	**XOR**: effettua l'XOR logico tra le variabili d'ingresso.
	**NAND**: effettua il NAND logico tra le variabili d'ingresso.
	**Compound Aritmetic:** effettua la somma, il prodotto, l'AND, l'OR di due o più valori numerici, cluster, valori booleani, in base al simbolo selezionato nel rettangolo a destra dell'icona che rappresenta tale funzione.

5.4 Array functions

Il pop-up menu che raccoglie le funzioni che permettono di manipolare array di dati è illustrato in Fig.93.

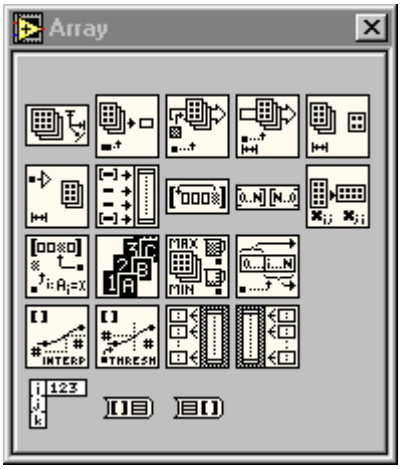
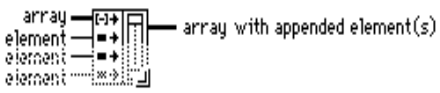
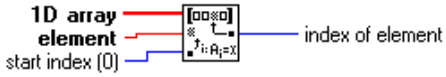
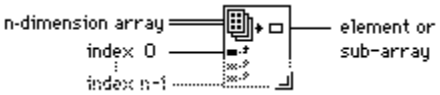
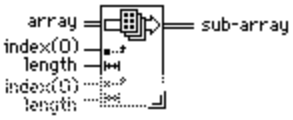


Fig. 93. Funzioni per la manipolazione di array.

Le funzioni raggruppate nel menu di Fig. 93 consentono di effettuare vari tipi di elaborazioni su vettori, come ad esempio la ricerca di un elemento dato un indice, la ricerca dell'elemento massimo in un dato vettore, il calcolo della dimensione di un vettore.

Le funzioni più utilizzate sono le seguenti

	**Build array**: concatena qualsiasi numero di array o singoli elementi collegati all'ingresso del VI nell'ordine prestabilito dai collegamenti (dall'alto verso il basso) e fornisce in uscita un singolo array.
	**Array size**: restituisce la dimensione dell'array di ingresso.
	**Search 1D array**: ricerca *element* nell'array iniziando dalla posizione *start index* e restituisce l'indice associato all'elemento trovato.
	**Index array:** fornisce in uscita l'elemento dell'*array* collocato nella posizione specificata da *index*. Se l'array è multidimensionale, si deve aggiungere un terminale che specifichi un ulteriore indice per ogni dimensione aggiuntiva del vettore.
	**Array subset**: restituisce la porzione di *array* ad iniziare da *index* e contenente *lenght* elementi
	**Array to cluster**: converte un array monodimensionale in un cluster di elementi dello stesso tipo dell'array.
	**Cluster to array**: converte un cluster formato da elementi dello stesso tipo in un array.

5.5 Cluster functions

Il pop-up menu che raccoglie le funzioni impiegate per la manipolazione di cluster è illustrato in Fig.94.

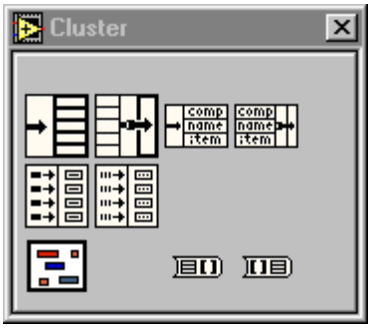
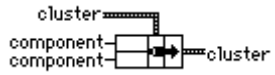
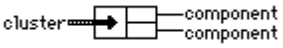
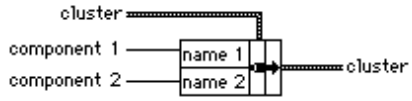
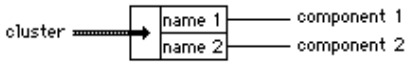


Fig. 94. Funzioni per la manipolazione di *cluster*.

Tali funzioni consentono fondamentalmente di costruire un cluster una volta dati i singoli elementi oppure di scomporre un cluster nei suoi costituenti.

Tra le varie disponibili, le funzioni più utilizzate sono le seguenti:

	**Bundle**: riunisce tutti i singoli componenti di ingresso in un unico cluster.
	**Unbundle**: scompone un cluster nei suoi elementi costituenti.
	**Bundle by name**: sostituisce alcuni componenti in un cluster preesistente. Una volta collegato il cluster di ingresso al VI, si deve effettuare un'operazione di pop-up sui terminali *name1*, *name2*,… per selezionare le componenti del cluster da sostitire
	**Unbundle by name**: restituisce gli elementi del cluster dei quali si specifica il nome: questa selezione avviene mediante un'operazione di pop-up sui terminali di uscita name1, name2, … e selezionando i nomi dalla lista degli elementi che compongono il cluster di ingresso.

5.6 Comparison functions

Le funzioni raggruppate nel seguente menu di Fig. 95, consentono di confrontare quantità di ingresso di tipo numerico, stringa, cluster.

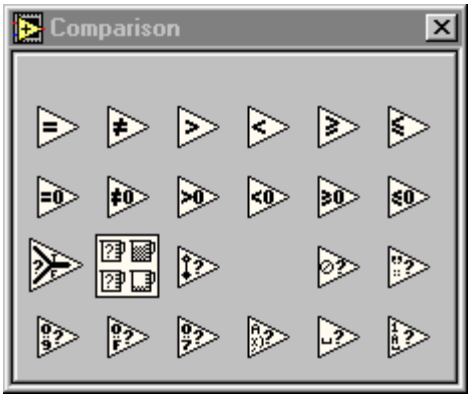


Fig. 95. Funzioni *Comparison.*

Si ricordano tra quelle disponibili in tale menu le seguenti:

	**Equal?** : restituisce il valore booleano *true* se x = y, *false* altrimenti.
	**Not equal?** : restituisce il valore booleano *true* se x ≠ y, *false* altrimenti.
	**Greater?** : restituisce il valore booleano *true* se x > y, *false* altrimenti.
	**Less or equal?** : restituisce il valore booleano *true* se x ≤ y, *false* altrimenti.
	**Select**: restituisce il valore connesso all'ingresso *t* o *f* in funzione dello stato logico assunto dal selettore booleano *s*. Se *s* è *true*, la funzione restituisce il valore collegato *t* , se *s* è *false*, restituisce il valore collegato ad *f*.

5.7 Time&Dialog functions

Le funzioni raggruppate nel menu *Time&Dialog*, rappresentato nella seguente Fig. 96, consentono di misurare intervalli temporali, sospendere operazioni per un determinato periodo di tempo, ecc.

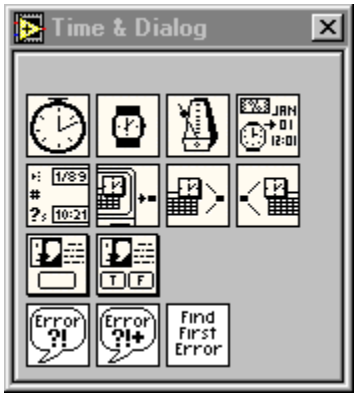
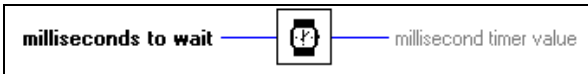
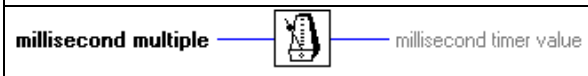
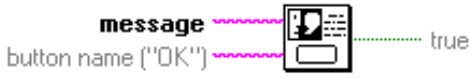


Fig. 96. Menu pop-up delle funzioni *Time&Dialog.*

Tra questa si ricordano quelle brevemente descritte nel seguito.

	**Wait(ms):** sospende l'esecuzione per una durata pari al valore specificato all'ingresso, in ms. Restituisce il valore dell'orologio di sistema, espresso in ms.
	**Wait until next ms multiple**: sospende l'esecuzione finché il valore dell'orologio di sistema diventa un multiplo del valore d'ingresso *milliseconds multiple*.
	**Tick count:** restituisce il valore dell'orologio di sistema, espresso in ms.
	**One button dialog box**: visualizza nel pannello frontale una finestra di dialogo che contiene un messaggio ed un bottone. Il controllore *button name* specifica il nome del bottone.

5.8 Instument I/O VIs

Il pop-up menu che raccoglie i VI *Instrument I/O* è illustrato in Fig. 97.

Fig.97. Funzioni *Instrument I/O.*

I VI raggruppati nel menu *Instrument I/O* di Fig. 97 gestiscono la comunicazione tra personal computer e strumentazione attraverso vari tipi di interfaccia (IEEE 488, EIA RS 232, VISA, VXI). Le funzioni implementate sono molteplici e consentono di gestire vari aspetti della comunicazione. In genere le più utilizzate nei laboratori di misure elettroniche sono quelle che gestiscono strumentazione collegata ad un bus IEEE-488. Quest'ultime funzioni sono raccolte in due pop-up menu: GPIB (Fig. 98) e GPIB 488.2 (Fig. 99).

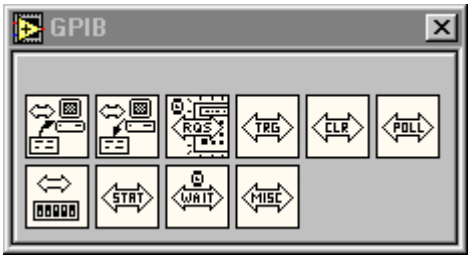


Fig. 98. *Instrument I/O GPIB.*

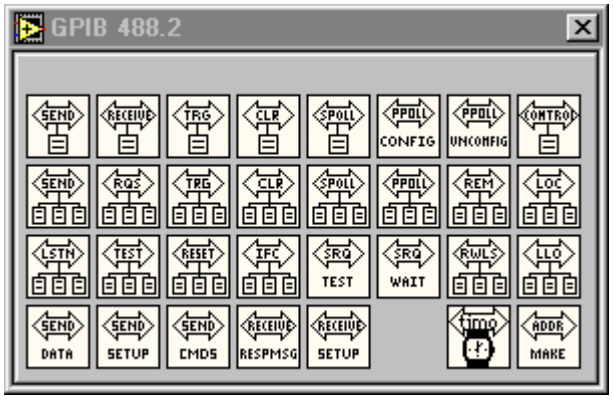


Fig. 99. *Instrument I/O 488.2.*

In genere i VI più utilizzati sono quelli della libreria GPIB 488.2 che permettono di pilotare strumenti collegati ad un bus in modo conforme a quanto dettato dallo standard

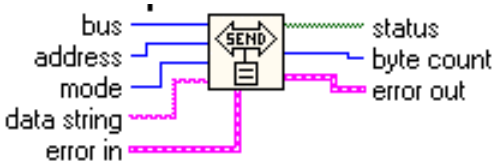
IEEE-488.2. I VI raccolti nella libreria possono essere suddivisi in quattro categorie funzionali: *single device, multiple device, bus management* e *low level*.

I VI richiedono in ingresso l'indirizzo del dispositivo su cui eseguire l'operazione richiesta; l'indirizzo può essere formato da indirizzo primario e secondario inseriti nei byte meno significativo e più significativo. LabVIEW permette di utilizzare il VI *MakeAddr* per combinare gli indirizzi primario e secondario da fornire ai VI. Tutti i *Single Device VI* presentano un ingresso bus che rappresenta il riferimento alla scheda 488 installata sul computer. Tale valore è fissato normalmente a 0 ma può essere modificato utilizzando il software di configurazione venduto con la scheda stessa.

I VI più utilizzati per effettuare una comunicazione con strumenti collegati ad un bus 488 sono riportati qui di seguito con maggior dettaglio.

Single Device VI – SEND

Il VI Send permette di spedire byte ad un dispositivo connesso al bus 488 identificato da un indirizzo.

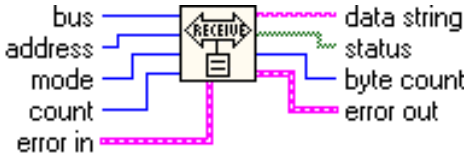


I terminali presenti nel VI sono:

bus	Intero senza segno a 16 bit che rappresenta l'indirizzo del dispositivo di interfaccia con il bus.
Address	Intero senza segno a 16 bit che rappresenta l'indirizzo del dispositivo sul bus 488.
mode	Intero senza segno a 8 bit che specifica il modo di terminazione nella trasmissione dei dati. Può assumere i valori: 0: nessun terminatore finale 1: imposta a 1 la linea EOI e spedisce un linefeed NL finale. 2: imposta a 1 la linea EOI quando è stato spedito l'ultimo byte.
Data string	Stringa contenente i dati da spedire.
Status	Vettore booleano che dà indicazioni sullo stato del bus dopo l'esecuzione di una operazione.
Byte count	Intero a 32 bit che fornisce il numero di byte trasmessi.
Error in	Cluster di ingresso che segnala eventuali precedenti errori nella comunicazione attraverso il bus 488.
Error out	Cluster di uscita che segnala eventuali errori avvenuti durante l'esecuzione del VI.

Single Device VI – RECEIVE

Il VI Receive permette di leggere i byte trasmessi da un dispositivo connesso al bus 488.



Il VI termina la lettura dei dati quando si verifica una delle situazioni seguenti:

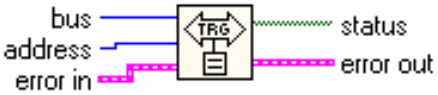
- i byte richiesti sono stati letti
- si è verificato un errore
- è passato un intervallo di tempo prefissato (timeout)
- la linea EOI segnala fine trasmissione.
- è stato trovato il carattere di terminazione EOS specificato nella chiamata.

I terminali presenti sono:

Bus	Intero senza segno a 16 bit che rappresenta l'indirizzo del dispositivo di interfaccia con il bus.
Address	Intero senza segno a 16 bit che rappresenta l'indirizzo del dispositivo sul bus 488.
Mode	Intero senza segno a 16 bit che specifica quale carattere EOS segnala la fine della ricezione dei dati. Se mode è uguale a 256 o non è collegato la trasmissione termina quando viene settata la linea EOI.
Count	Intero a 32 bit che specifica il numero dei byte da ricevere.
Data string	Stringa contenente i dati ricevuti
Status	Vettore di variabili booleane che fornisce indicazioni sullo stato del bus dopo l'esecuzione della operazione.
Byte count	Intero a 32 bit che fornisce il numero di byte ricevuti.
Error in	Cluster di ingresso che segnala eventuali precedenti errori nella comunicazione attraverso il bus 488.
Error out	Cluster di uscita che segnala eventuali errori avvenuti durante l'esecuzione del VI.

Single Device VI – TRIGGER

Il VI Trigger permette di spedire un messaggio *trigger* ad un dispositivo per provocare l'esecuzione di qualche operazione.

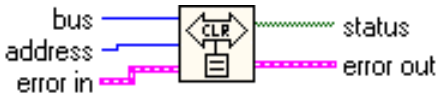


I terminali presenti nel VI sono:

Address	Intero senza segno a 16 bit che rappresenta l'indirizzo del dispositivo di interfaccia con il bus. L'indirizzo comprende indirizzo primario e secondario eventualmente combinati utilizzando l'apposito vi. Se l'indirizzo non è collegato, il VI trasmette il messaggio GET Group Execute Trigger a tutti i dispositivi indirizzati come listener presenti sul bus
Bus	Intero senza segno a 16 bit che rappresenta il riferimento al bus 488.
Status	Vettore di grandezze booleane che dà indicazioni sullo stato del bus dopo l'esecuzione di un'operazione.
Error in	Cluster di ingresso che segnala eventuali precedenti errori nella comunicazione attraverso il bus 488.
Error out	Cluster di uscita che segnala eventuali errori avvenuti durante l'esecuzione del VI.

Single Device VI – DEVCLEAR

Il VI DevClear permette di spedire un messaggio SDC Selected Device Clear ad un dispositivo per provocare l'esecuzione di qualche operazione.

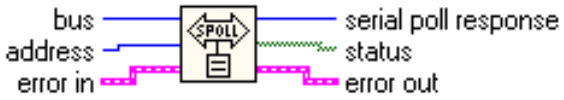


I terminali presenti nel VI sono:

bus	Intero senza segno a 16 bit che rappresenta l'indirizzo del dispositivo di interfaccia con il bus.
address	Intero senza segno a 16 bit che rappresenta l'indirizzo del dispositivo sul bus 488. Se il dispositivo non è collegato, il VI trasmette il messaggio UDC Universal Device Clear a tutti i dispositivi presenti sul bus
status	Vettore di grandezze booleane che dà indicazioni sullo stato del bus dopo l'esecuzione di una operazione.
Error in	Cluster di ingresso che segnala eventuali precedenti errori nella comunicazione attraverso il bus 488.
Error out	Cluster di uscita che segnala eventuali errori avvenuti durante l'esecuzione del VI.

Single Device VI – READSTATUS

Il VI ReadStatus viene utilizzato quando il controller rileva una richiesta di servizio sulle linee del bus . Il controller interroga i dispositivi eseguendo un polling seriale per sapere chi ha richiesto servizio.

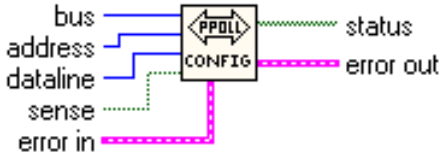


I terminali presenti nel VI sono:

Bus	Intero senza segno a 16 bit che rappresenta l'indirizzo del dispositivo di interfaccia con il bus.
Address	Intero senza segno a 16 bit che rappresenta l'indirizzo del dispositivo sul bus 488.
Serial poll Response	Intero a 16 bit che contiene il byte di status del dispositivo interrogato dal controller.
Status	Vettore di grandezze booleane che dà indicazioni sullo stato del bus dopo l'esecuzione di una operazione.
Error in	Cluster di ingresso che segnala eventuali precedenti errori nella comunicazione attraverso il bus 488.
Error out	Cluster di uscita che segnala eventuali errori avvenuti durante l'esecuzione del VI.

Single Device VI – PPOLLCONFIG

Il VI ReadStatus viene utilizzato quando il System controller rileva una richiesta di servizio sulle linee del bus e desidera sapere quali dispositivi hanno richiesto tale servizio. Il controller assegna ai dispositivi una delle linee dati del bus ed esegue un poll parallelo dei dispositivi. I dispositivi che hanno richiesto servizio settano la linea a loro assegnata permettendo il riconoscimento al controller. Il VI PPollConfig permette di assegnare le linee dati del bus ai dispositivi.



Bus	Intero senza segno a 16 bit che rappresenta l'indirizzo del dispositivo di interfaccia con il bus.
Address	Intero senza segno a 16 bit che rappresenta l' indirizzo del dispositivo sul bus 488.
Dataline	Intero a 16 bit che contiene il numero della linea 1-8;
sense	Variabile booleana che viene confrontata con il bit di stato dello strumento. Se c'è corrispondenza la linea dataline è settata.
Status	Vettore di grandezze booleane che dà indicazioni sullo stato del bus dopo l'esecuzione di un'operazione.
Error in	Cluster di ingresso che segnala eventuali precedenti errori nella comunicazione attraverso il bus 488.
Error out	Cluster d'uscita che segnala eventuali errori avvenuti durante l'esecuzione del VI.