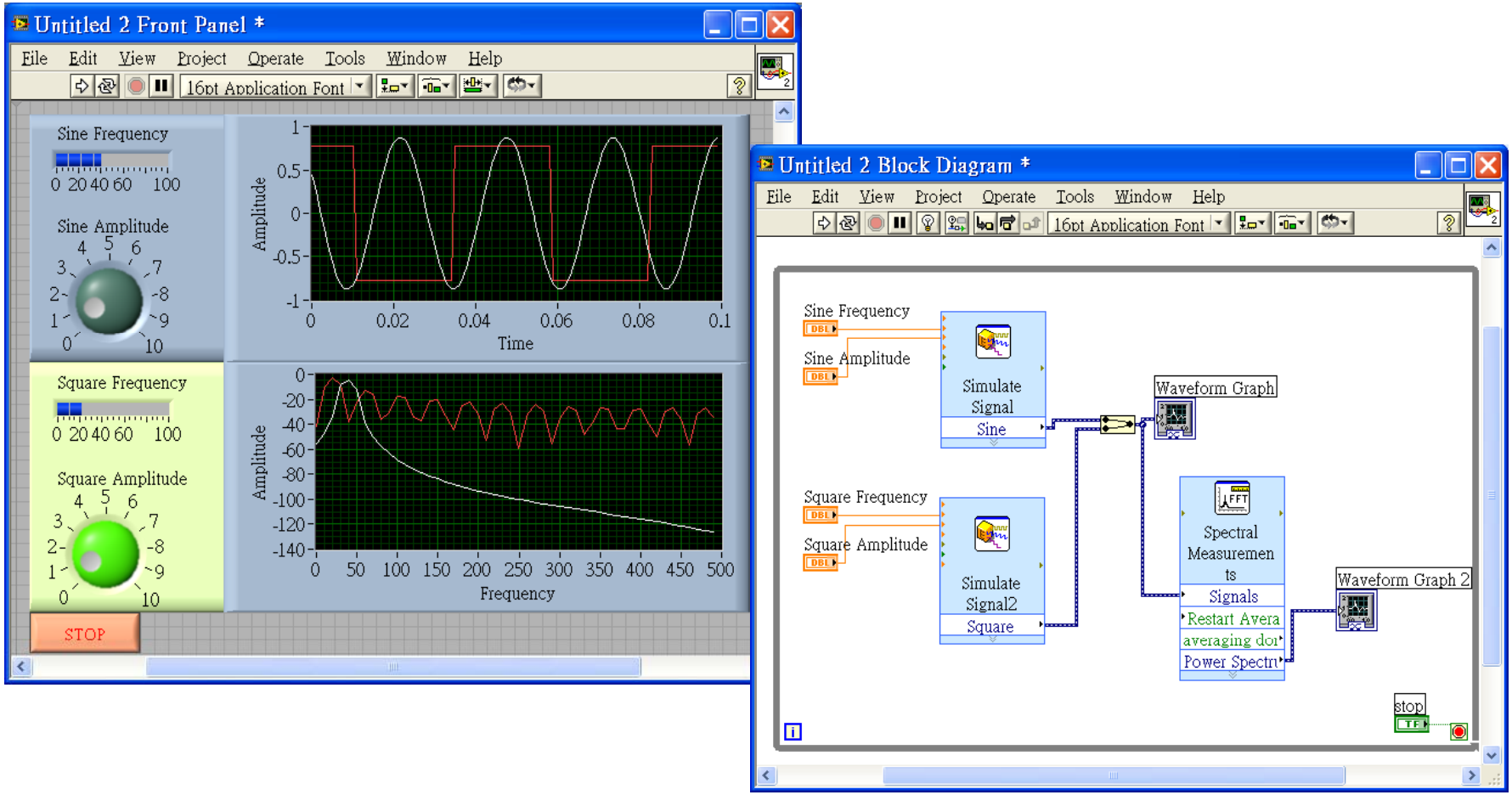
認識 LabVIEW

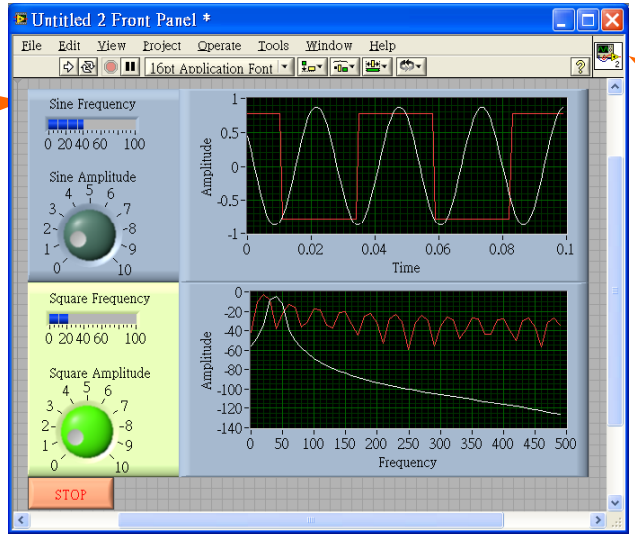
short for **Lab**oratory **V**irtual **I**nstrumentation **E**ngineering **W**orkbench

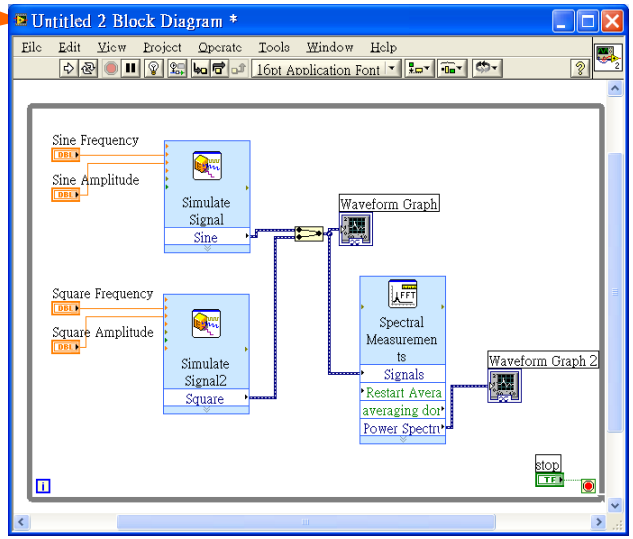
實驗室虛擬儀器專案平台



VI由哪些部分組成？

- 儀器面板（Front Panel）：
  - 是一個VI的操作介面環境，提供安置人機介面的各項物件
- 圖示程式碼（Block Diagram）：
  - 是一個圖形程式設計環境，由許多小圖像（SubVI）與結點所組成

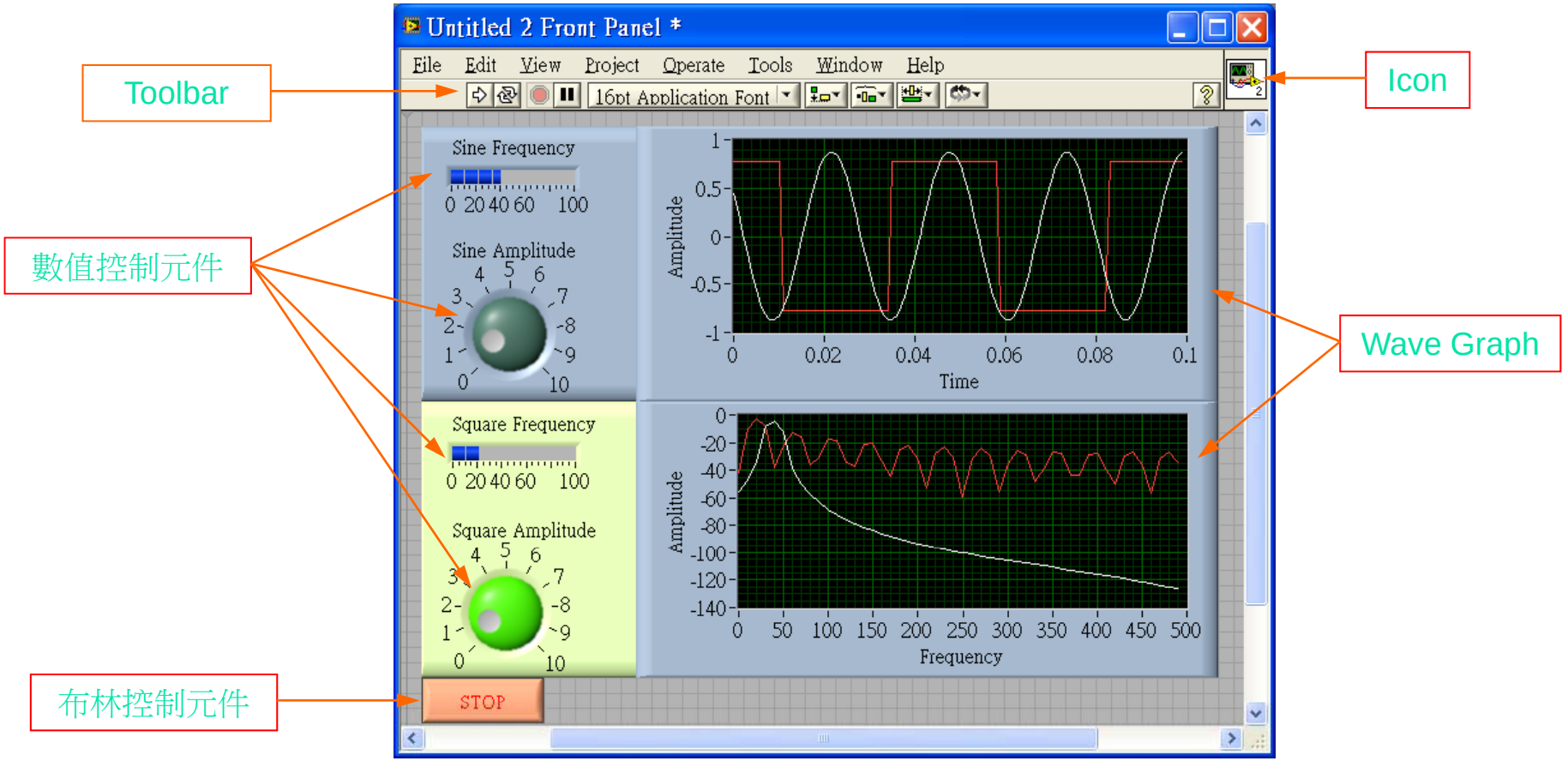




- 圖像（Icon）與聯結器（Connector）：
  - 圖像是一個VI或SubVI的代表，而聯結器則是定義每一個圖像的輸出與輸入

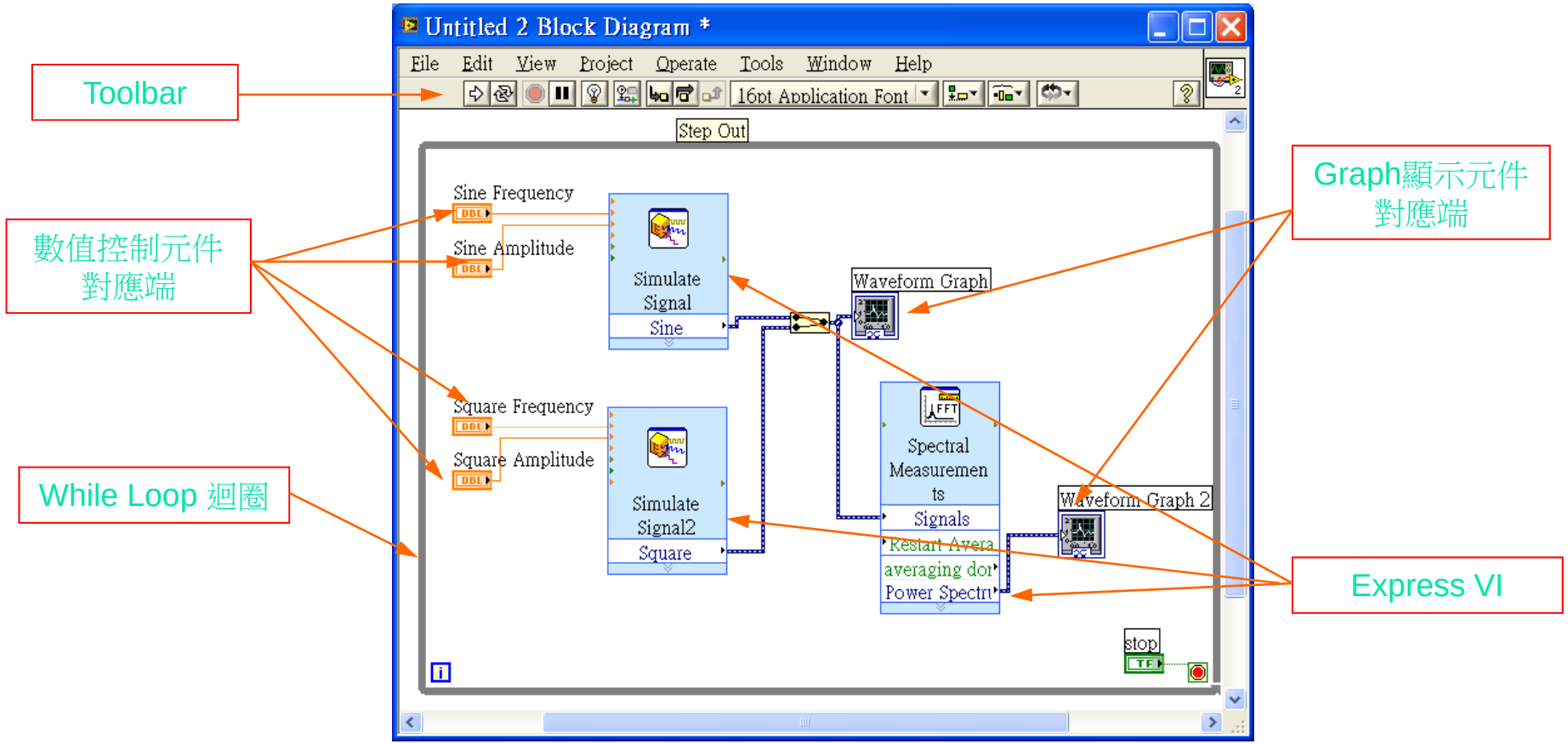
前置面板（Front Panel）

前置面板（Front Panel）



圖示程式（Block Diagram）

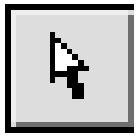
圖示區（Block Diagram）



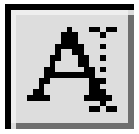
編輯工具面版（Tools Palette）

edit and modify both front panel and block diagram

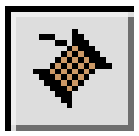
操作工具（Operating tool）：用來操作前置面板上的各項人機介面物件



定位工具（Positioning tool）：用來選擇、移動物件以及改變物件的大小

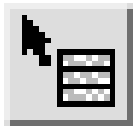


文字標示工具（Labeling tool）：用來編輯註解或輸入文字



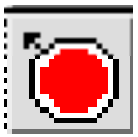
連線工具（Wiring tool）：用來連接圖示區內的各個結點

編輯工具面版（Tools Palette）



隨機選單工具（Object pop-up menu tool）：可隨時呼叫物件的隨機式選單

捲軸工具（Scroll tool）：可不用捲軸棒而能移動整個視窗

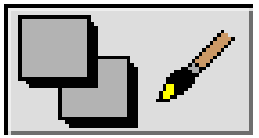


中斷點工具（Breakpoint tool）：用來設定中斷點，讓該VI執行到該點時中斷

探查工具（Probe tool）：用來設定探查點，以顯示該連線上通過的數據

編輯工具面版（Tools Palette）

顏色複製工具（Color copy tool）：方便複製顏色，再由著色工具進行著色

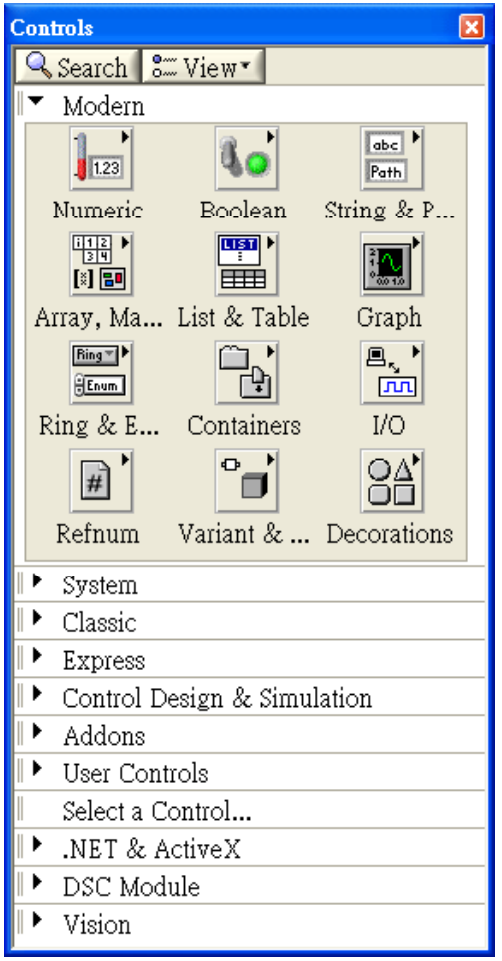


著色工具（Color tool）：用來設定物件的顏色，可設定前景與背景的顏色

儀器面板–控制物件面盤

控制物件面盤（**Controls Palette**）包含了常用的控制/顯示物件，簡單介紹如下：

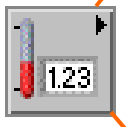
- Numeric
- Boolean
- String & Path
- Array, Matrix & Cluster
- List & Table
- Graph
- Ring & Enum
- Containers
- I/O
- Refnum
- Variant & Class
- Decorations

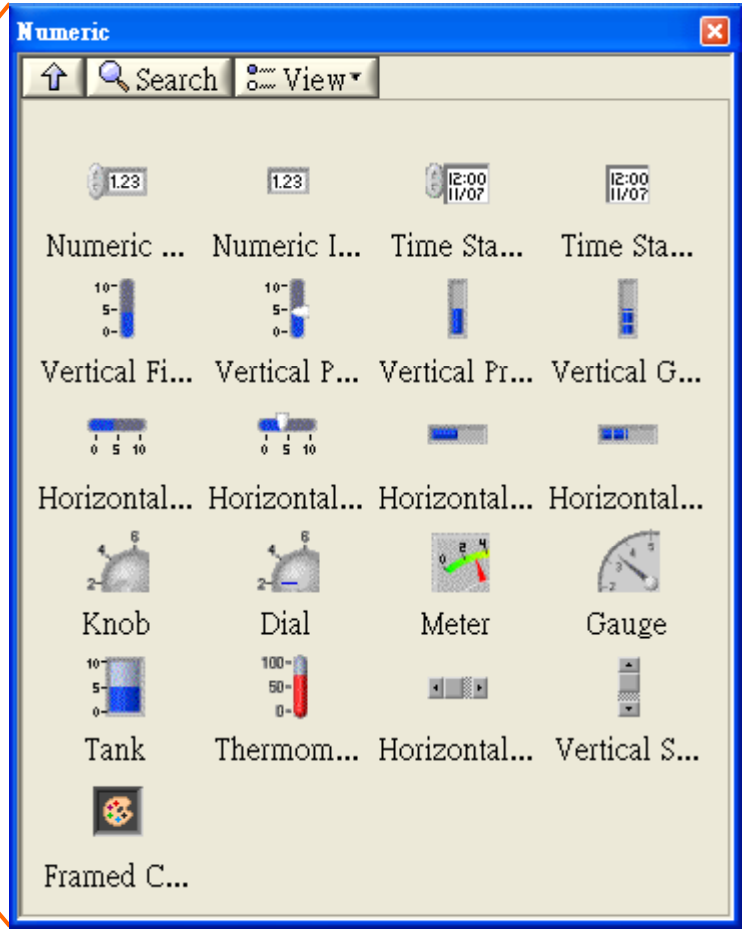


控制物件調色板（Controls Palette）

- 數值物件庫（ Numeric ）

內建多種數值控制或顯示物件，例如：滑動開關、旋鈕、溫度計等

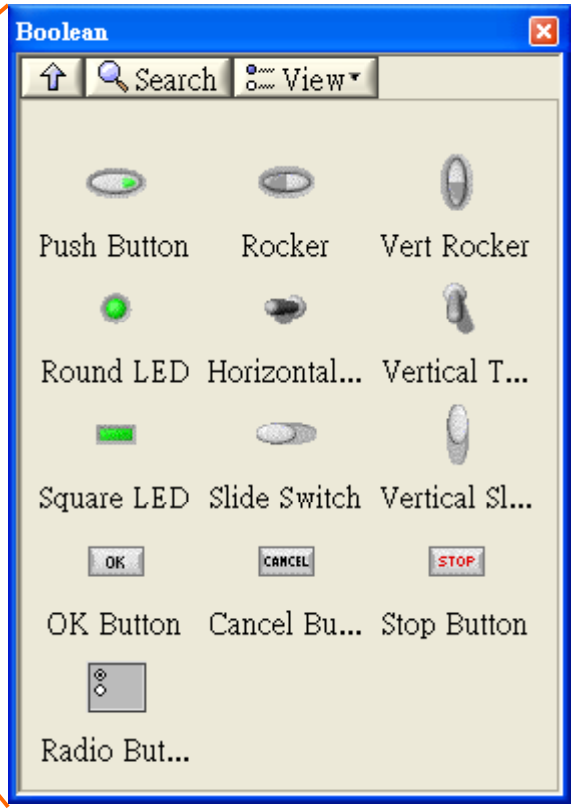




控制物件面版（Controls Palette）

- 布林物件庫（Boolean）：

內建多種布林控制或顯示物件，並且提供六種機械動作選項

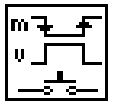


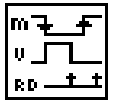
控制物件面版（Controls Palette）

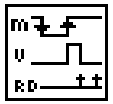
- 布林物件的機械動作

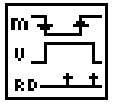

Switch When Pressed — 壓一次改變一次狀態


Switch When Released — 壓住後放掉就改變狀態


Switch Until Released — 壓住時才改變狀態（單擊開關）

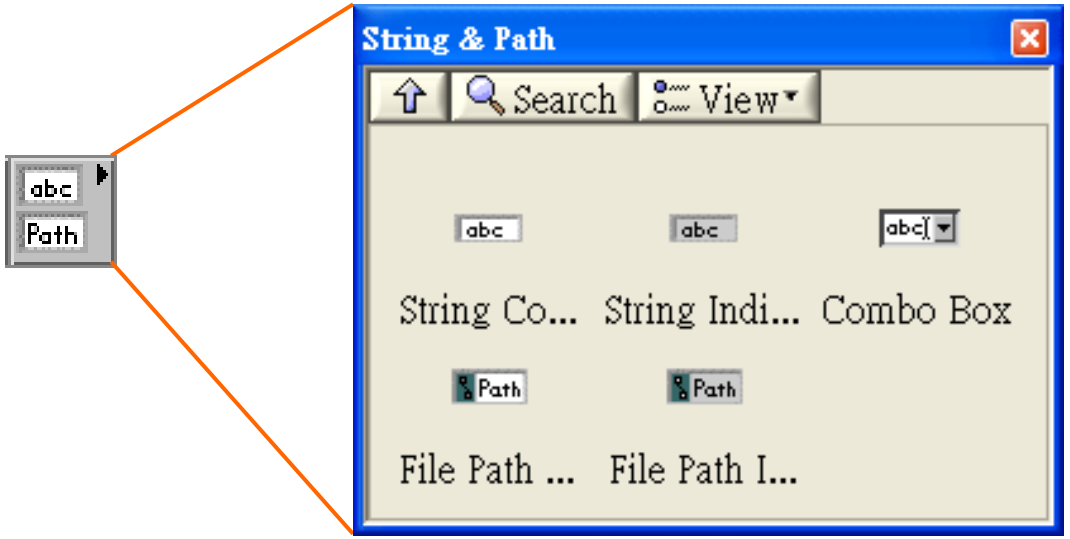

Latch When Pressed — 壓住後立即改變狀態，不久後又回到原狀態


Latch When Released —壓住後放掉才改變狀態，但馬上又回到原狀態


Latch Until Released — 壓住後立即改變狀態，直到放掉後過一下才回到原狀態

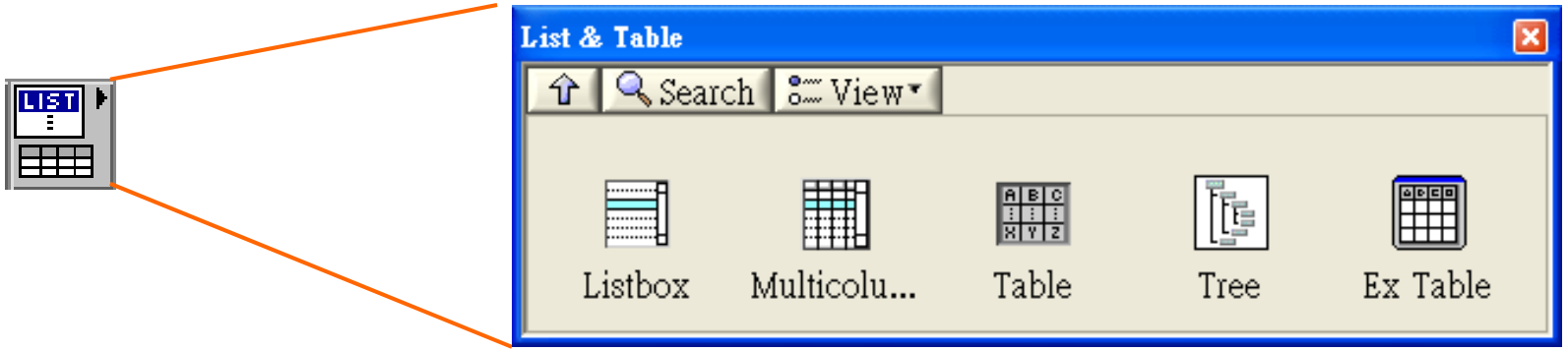
控制物件面版（Controls Palette）

- 字串與路徑物件庫（String & Path）



控制物件面版（Controls Palette）

- 目錄與表格物件庫（List & Table）

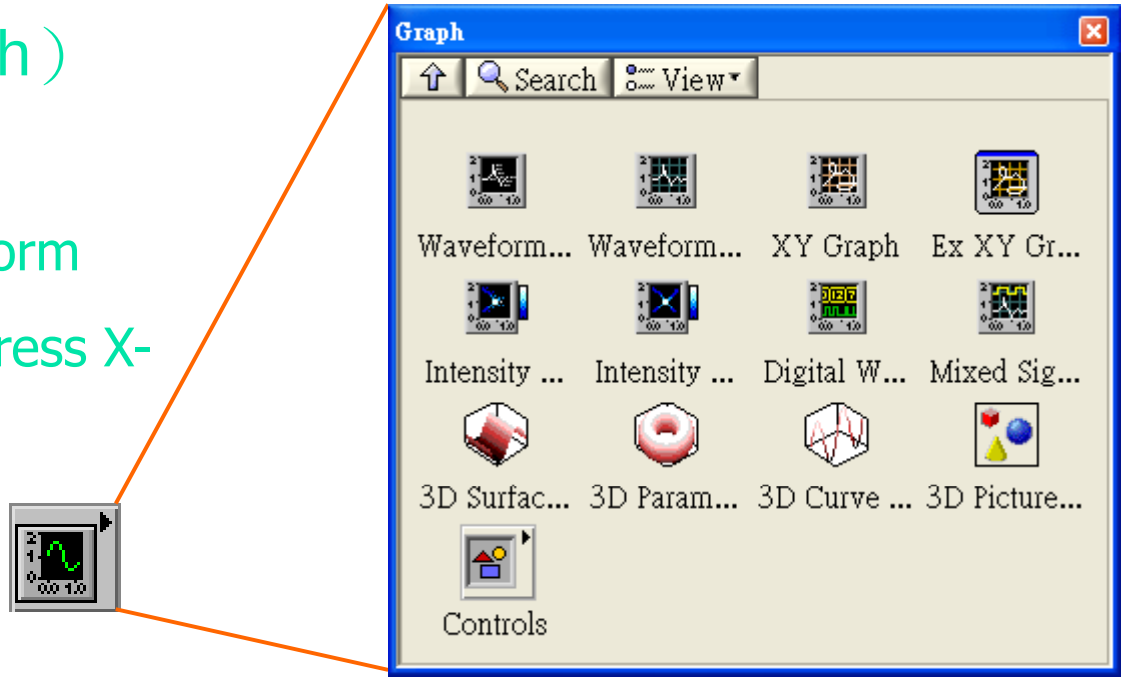


控制物件面版（Controls Palette）

- 繪圖物件庫（Graph）

內建多種繪圖工具如：

Waveform Chart、Waveform Graph、X-Y Graph、 Express X-Y Graph 、 3D Graph...等



儀器面板工具列 Front Panel Tool Bar

執行鍵：按下此鍵可執行VI

連續執行鍵：按下此鍵VI會不斷執行，值到按下停止鍵

停止鍵：按下此鍵可以放棄執行VI

暫停鍵：按下此鍵可以暫停VI的執行，再按一下便恢復執行

排列環：設定排列選擇等

分配環：設定間隙分配等

尺寸環：設定最大與最小尺寸

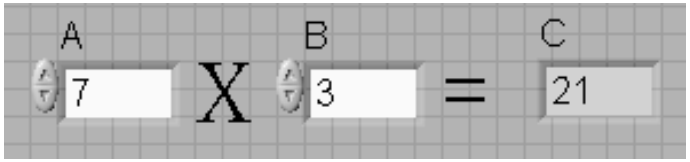
順序環：設定重疊時，出現的順序

字體框：設定字體的選項， 包含字型、大小、顏色等

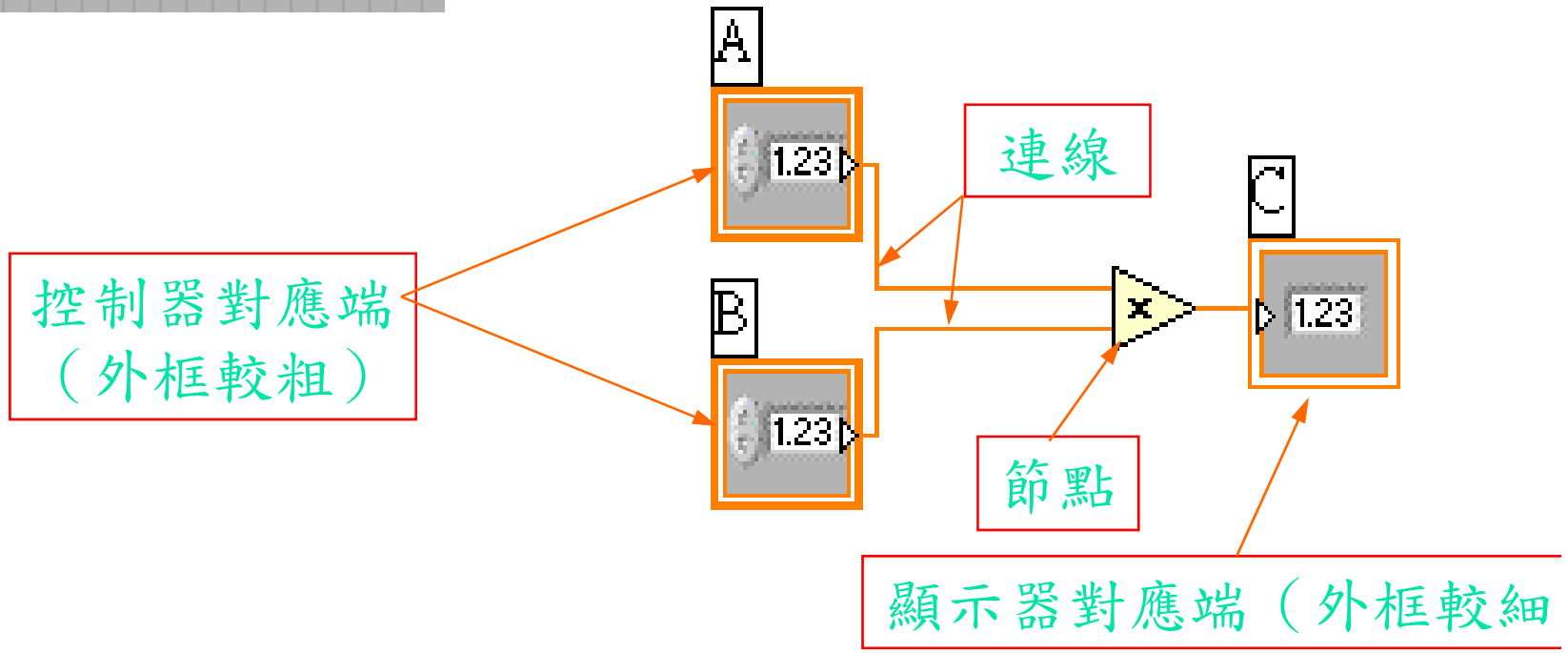
圖示程式碼 3元件

- 圖示程式分為三種基本元件：控制與顯示對應端、結點、連線

前置面板



圖示區



圖示程式碼:控制或指示對應端

- ***對應端（Terminal）***：

  當你在前置面版區加入一個控制或指示物件時，在圖示區就會出現一個相對應的對應端。並且以顏色與形狀來區分。例如外框粗的代表它是一個控制物件而細的代表它是一個指示物件。

圖示程式碼: 數值控制與指示對應端

控制	指示	資料型態說明	顏色
EXT	EXT	高精密度浮點數	橙色
DBL	DBL	雙精密度浮點數	橙色
SGL	SGL	單精密度浮點數	橙色
CXT	CXT	複數高精密度浮點數	橙色
CDB	CDB	複數雙精密度浮點數	橙色
CSG	CSG	複數單精密度浮點數	橙色
I32	I32	32-bit 整數	藍色
I16	I16	16-bit 整數	藍色
I8	I8	8-bit 整數	藍色
U32	U32	無正負號 32-bit 整數	藍色
U16	U16	無正負號16-bit 整數	藍色
U8	U8	無正負號8-bit 整數	藍色

圖示程式碼：控制與指示對應端

		數列集(Cluster)	棕或粉紅色
		數列(Array)	多變的顏色
		布林(Boolean)	綠色
		字串列(String)	粉紅色
		路徑(Path)	青綠色

圖示程式碼：對應端 連線

- ***連線（Wire）：***

  藉由連線，資料可由輸出端點傳輸到輸入端點，但是不同型態的資料是不可互相連線，輸入端點與輸入端點或輸出端點與輸出端點也是不可互相連線的。

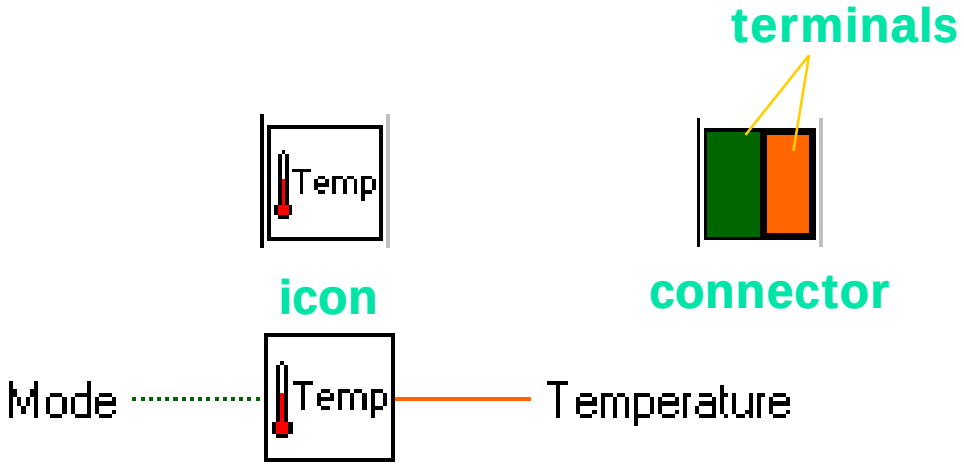
階層（Hierarchy）和圖像的聯結

- ***階層（Hierarchy）：***

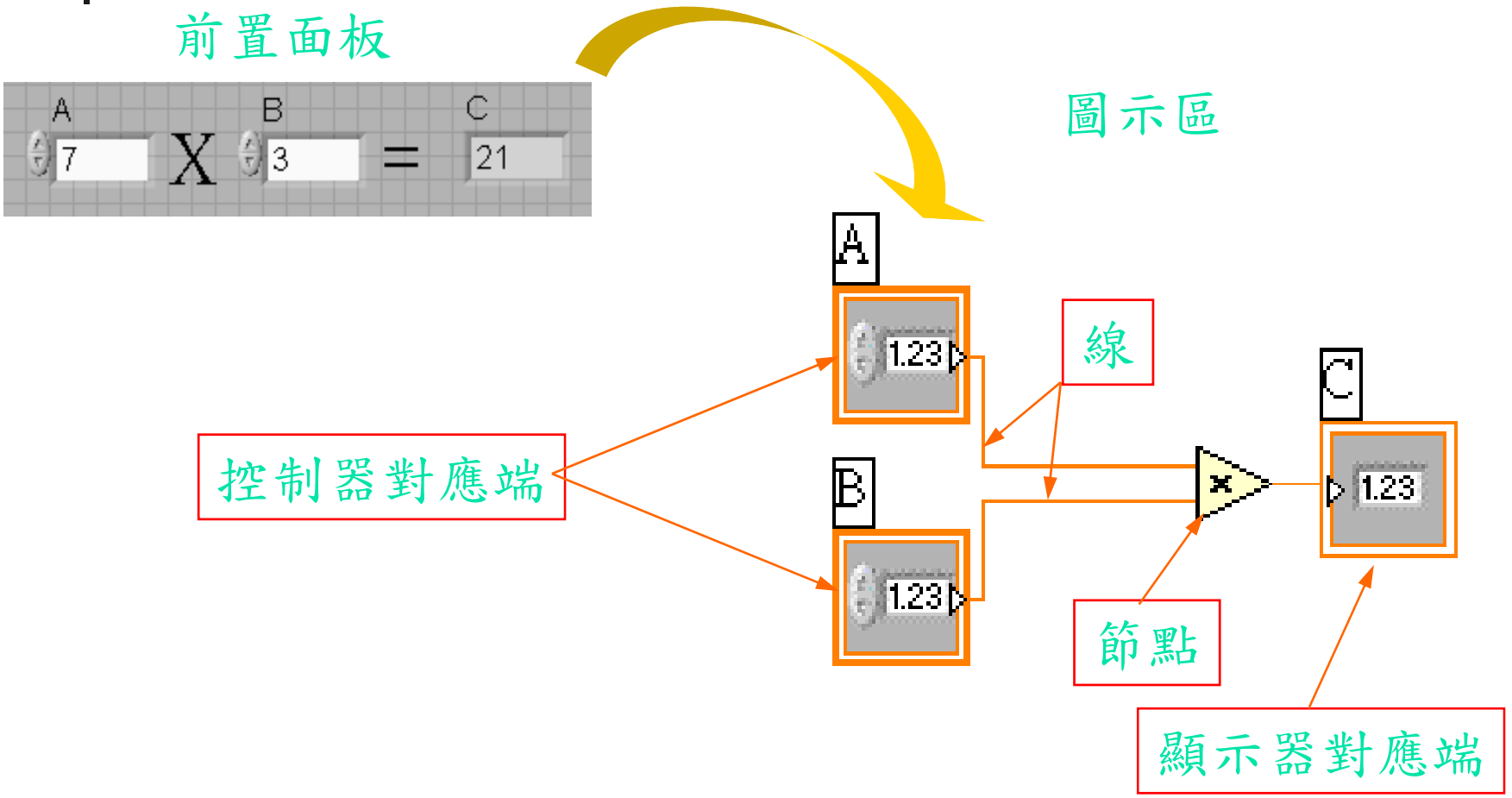
  是LabVIEW的特點之一，也就是說每一個VI之中會有數個SubVI，SubVI之中又會有數個SubVI，如此一來可以讓程式看起來更加簡潔。

- ***圖像（icon）的聯結：***

  每一個VI都可以設計一個圖像來代表它，這個圖像就在前置面版的右上方，可利用選取顯示聯接器（Show Connector）來進行物件與聯接器之間的關係的建立。



建立圖示程式碼（Diagram Panel）



圖示程式碼-函數面盤

函數面盤（**Functions Palette**）
包含了常用的Function Sub-Palette，簡單介紹如下：

- Structures
- Array
- Cluster & Variant
- Numeric
- File I/O
- Boolean
- String
- Comparison
- Timing
- Dialog & User Interface
- Waveform
- Application Control
- Synchronization
- Graphics & Sound
- Report Generation
- ...etc.

函數面版（Functions Palette）

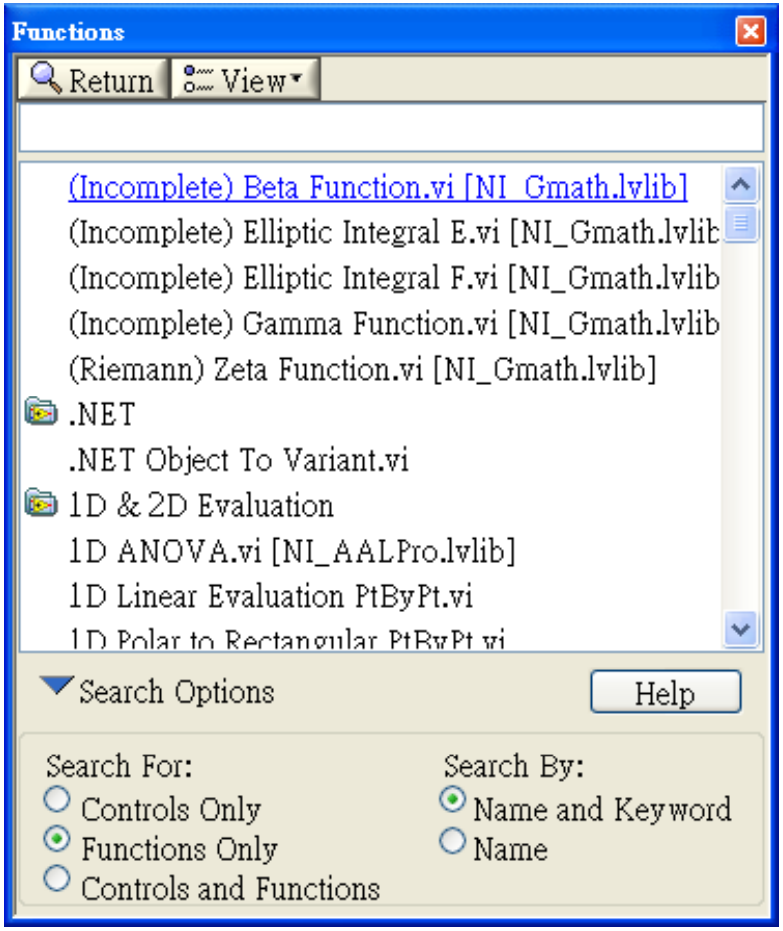
尋找

瀏覽選項（View）：

- 可以在這裡選擇要呈現的函數面版
- 可以在這裡增減或規劃常用的函數庫

搜尋Controls、VIs或Functions

- 按下<搜尋>按鍵可以如右圖輸入文字執行搜尋Controls、VIs或Functions
- 在圖示區可將選擇的項目直接拖曳到圖示區
- 也可以double-click選擇的項目直接打開這個VI



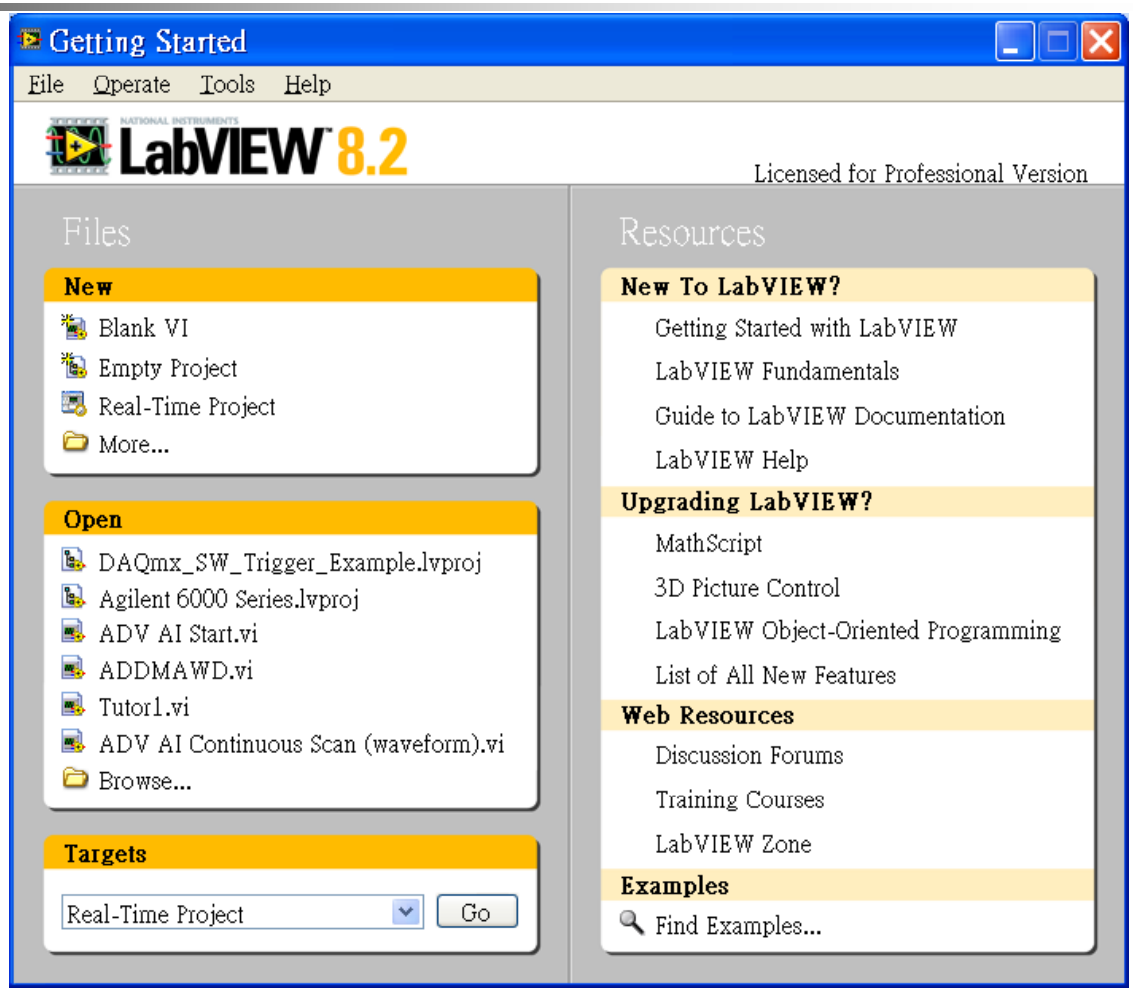
規劃自己的控制面盤或函數面盤

- Programs\National Instruments\LabVIEW 8.2

- 將vi.lib放在LabVIEW 8.2的目錄裡面
- 將你自建的函數或控制物件放在user.lib或instr.lib內讓它們出現在控制面盤或函數面盤

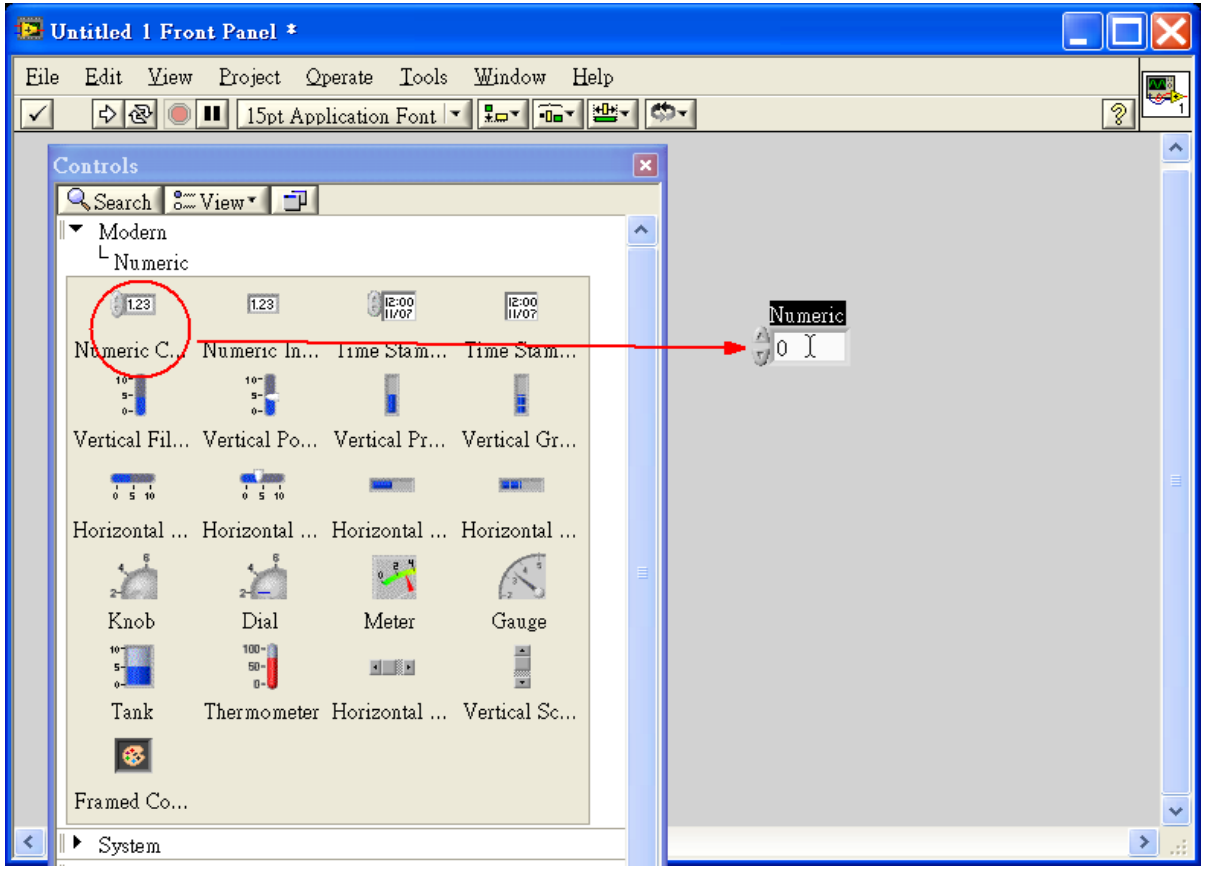
建立一個虛擬儀表

這是LabVIEW 8.2的起始畫面，我們可以經由選取“File”裡面的“New VI”或畫面中的“Blank VI”來開始一個虛擬儀表的建立

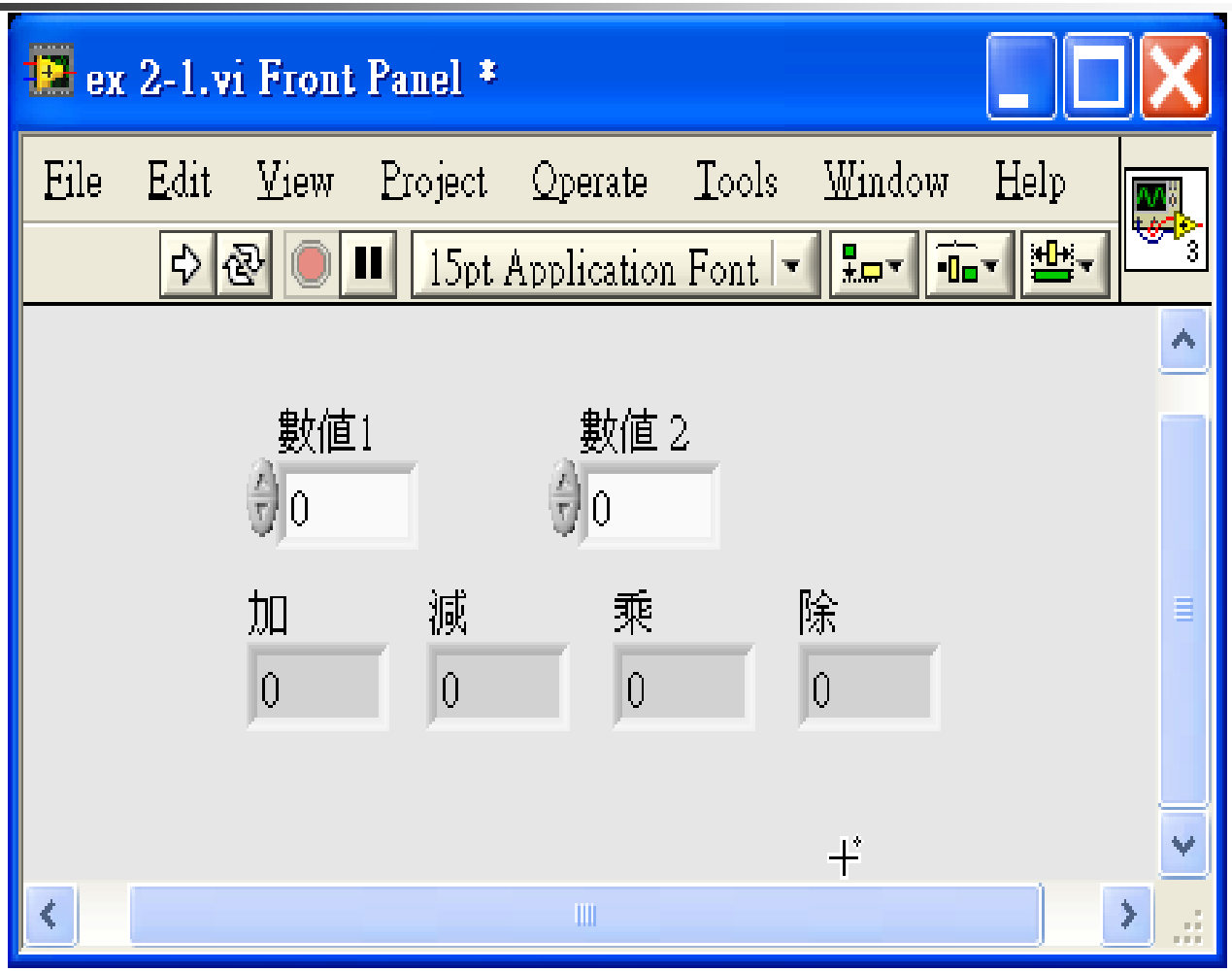


輸入及輸出元件放置

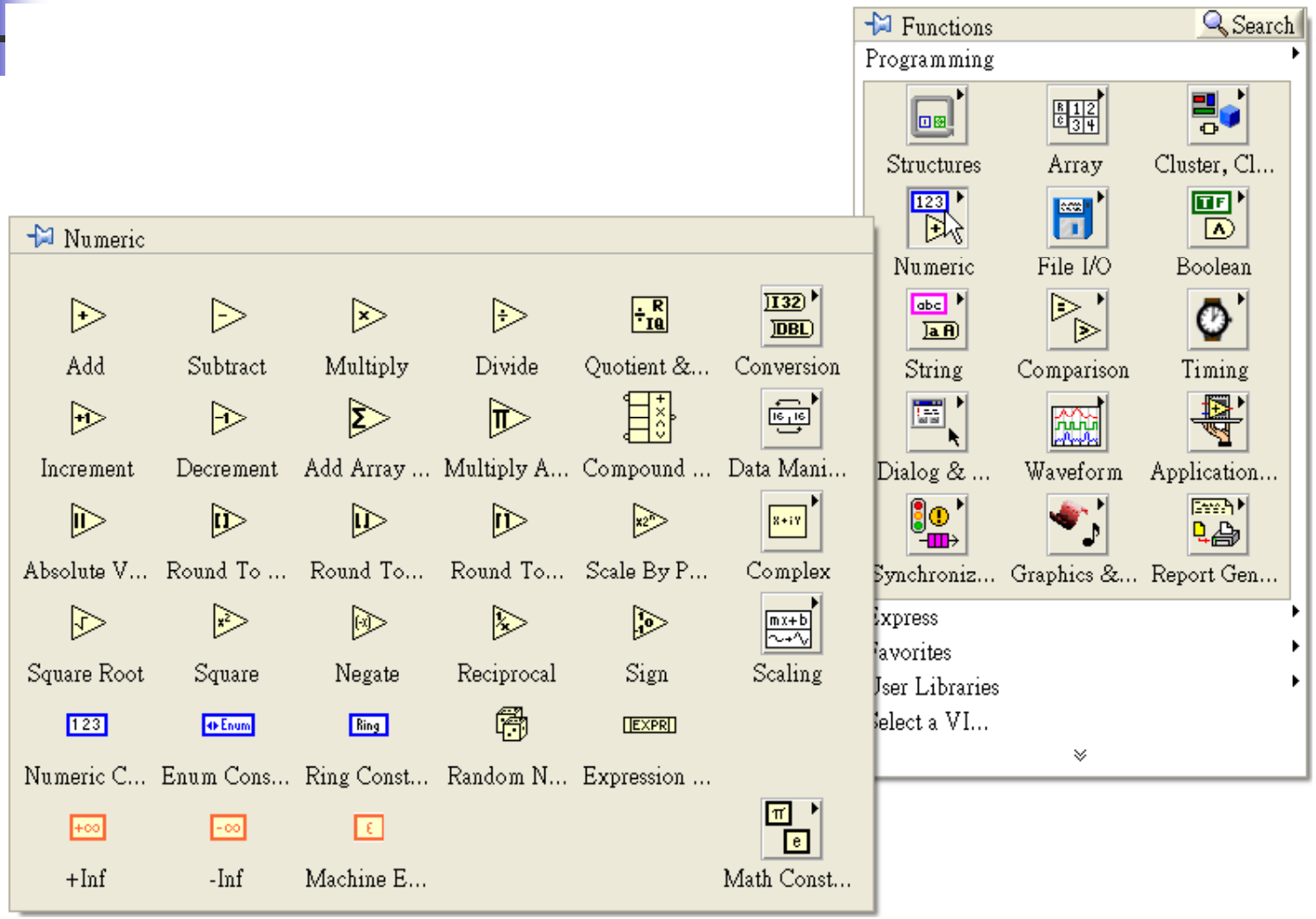
- 前置面板是由控制元件（輸入）與顯示元件（輸出）所組成



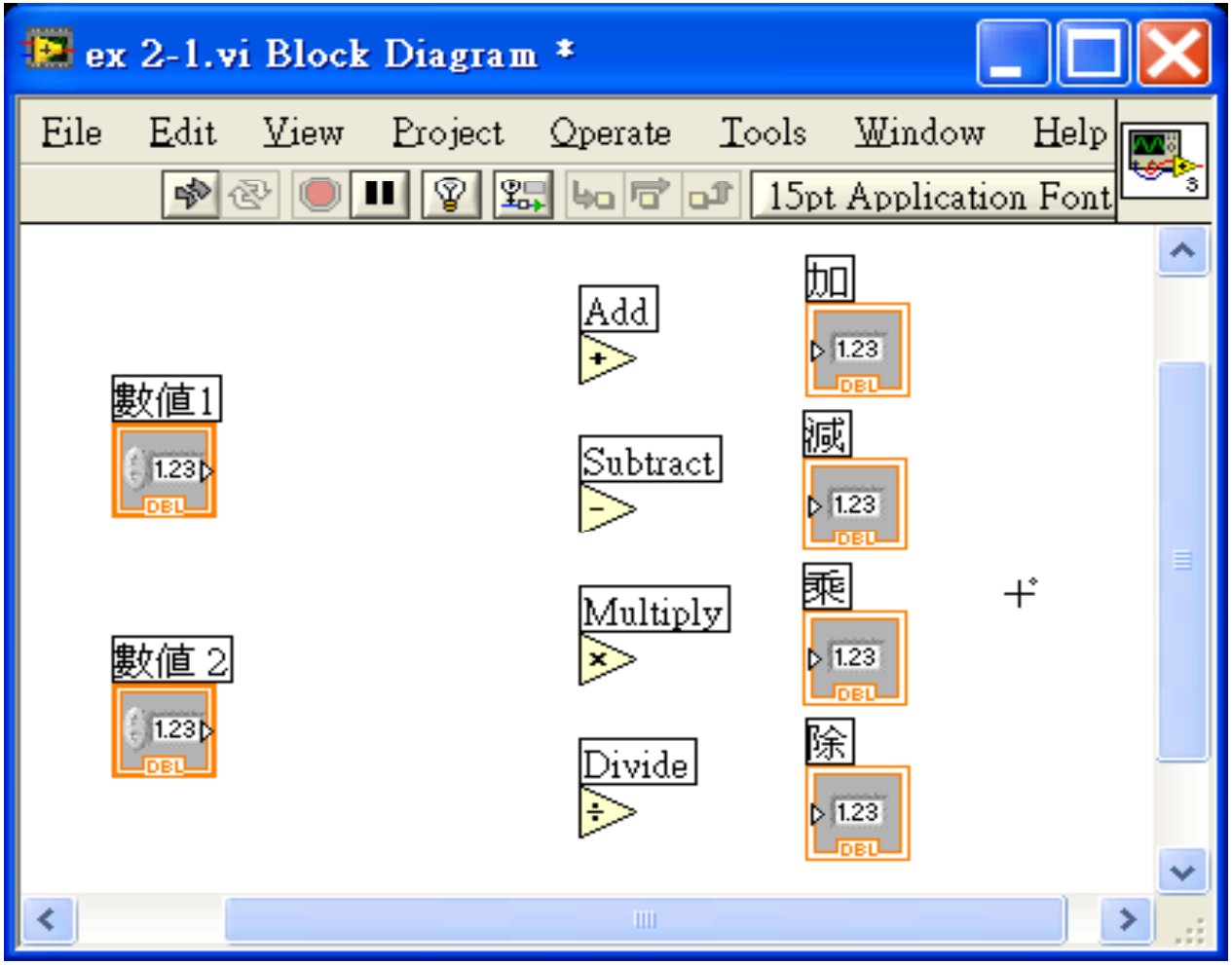
更改元件標籤


ex 2-1.vi Front Panel *
File Edit View Project Operate Tools Window Help
15pt Application Font
數值1
0
數值 2
0
加
0
減
0
乘
0
除
0

將函數放置於程式方塊圖區

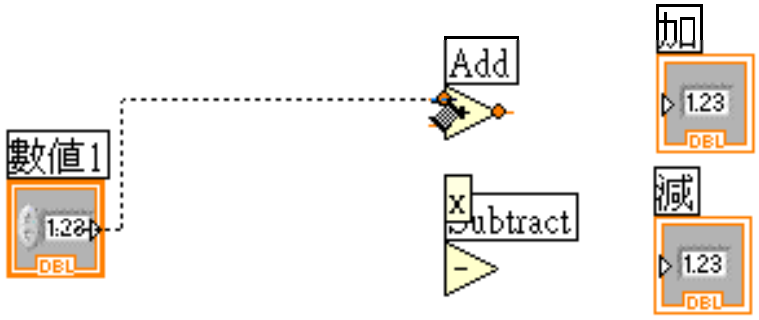


程式方塊圖區

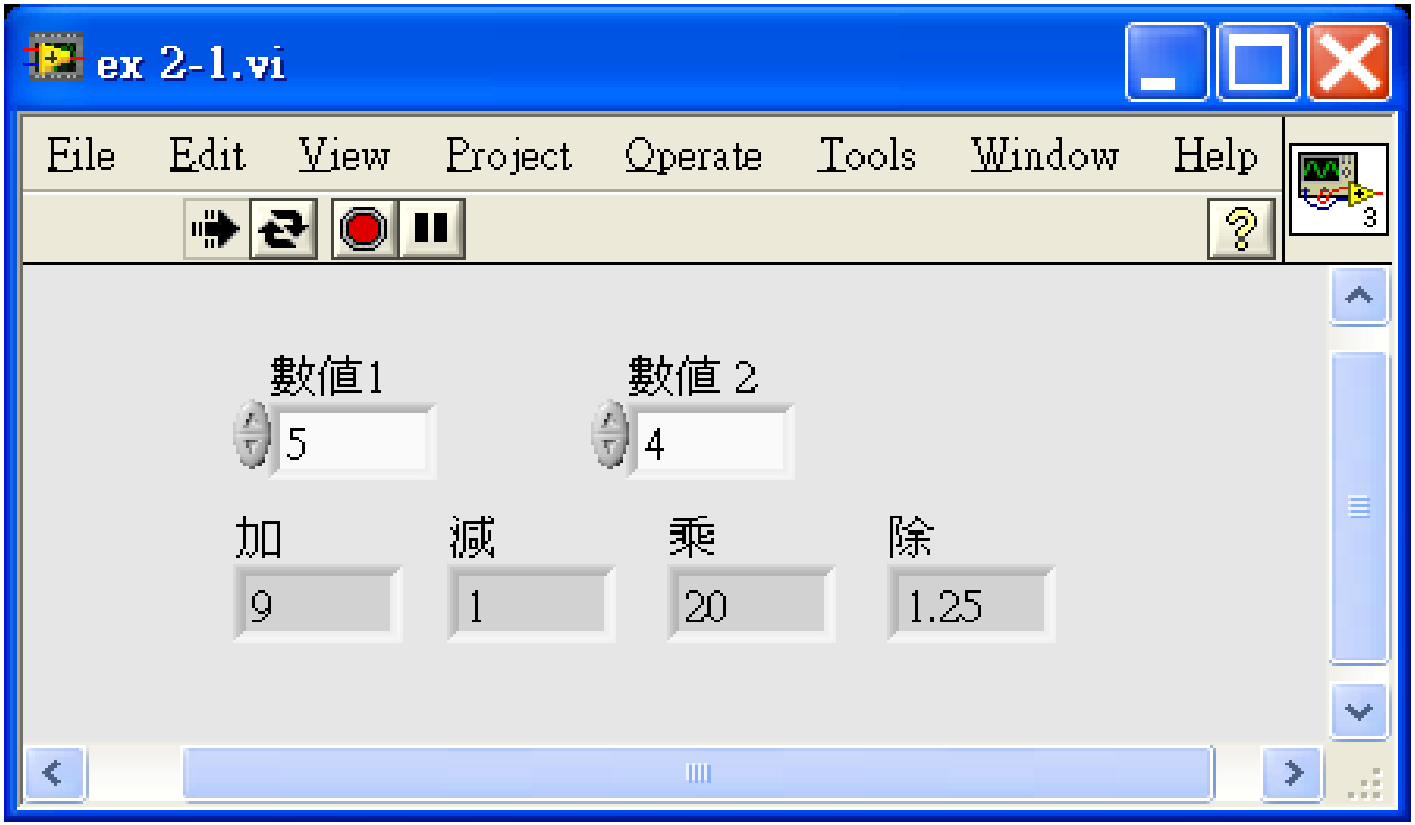

ex 2-1.vi Block Diagram *
File Edit View Project Operate Tools Window Help
15pt Application Font
數值 1
1.23
DBL
數值 2
1.23
DBL
Add
+
加
1.23
DBL
Subtract
-
減
1.23
DBL
Multiply
×
乘
1.23
DBL
Divide
÷
除
1.23
DBL

連線

- 連線，是為兩個端點連接一條資料路徑，且其資料為單一流向，資料的來源可為一端點，但目的可以為多個端點，因LabVIEW是依據資料流的模式來執行VIs，因此，所有的元件都要經過連線有資料流流過之後方可執行，其執行方向則是依據流程控制的模式進行。

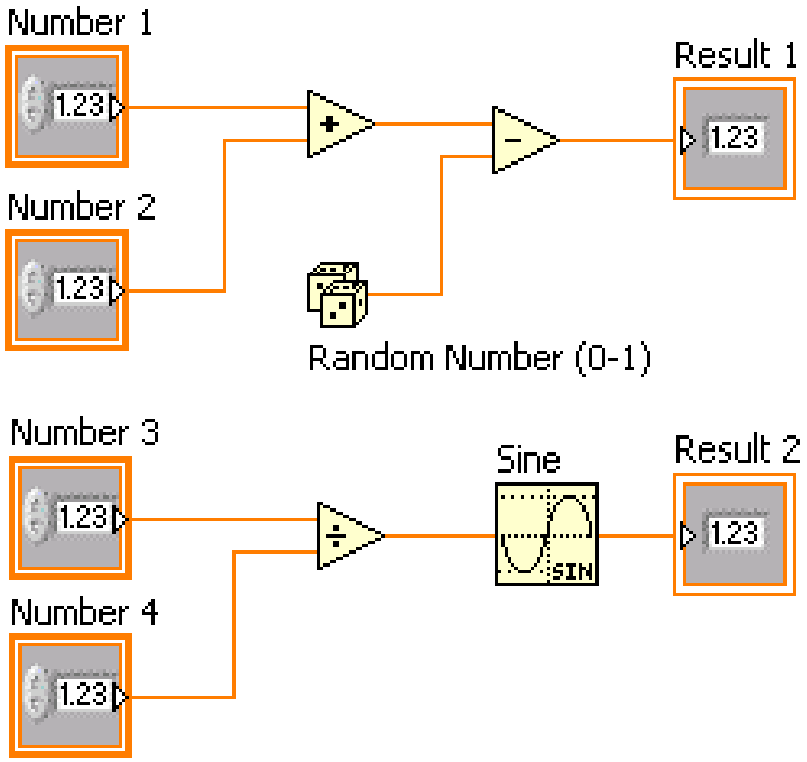


VI程式的執行


ex 2-1.vi
File Edit View Project Operate Tools Window Help
數值1
5
數值 2
4
加
9
減
1
乘
20
除
1.25

LabVIEW資料流程式的觀念

- 程式的執行是根據資料的流動順序而非由左至右
- 節點或SubVI的執行與否是根據所有的輸入端的資料是否到齊
- 節點會在執行完畢後提供資料給所有輸出端



LabVIEW的偵錯工具

- 圖示區工具棒上的偵錯按鍵的應用
  - 介紹圖示區工具棒區中的相關除錯工具

執行提示鍵 — 可以清楚看到程式執行的順序與資料（Data）的狀態

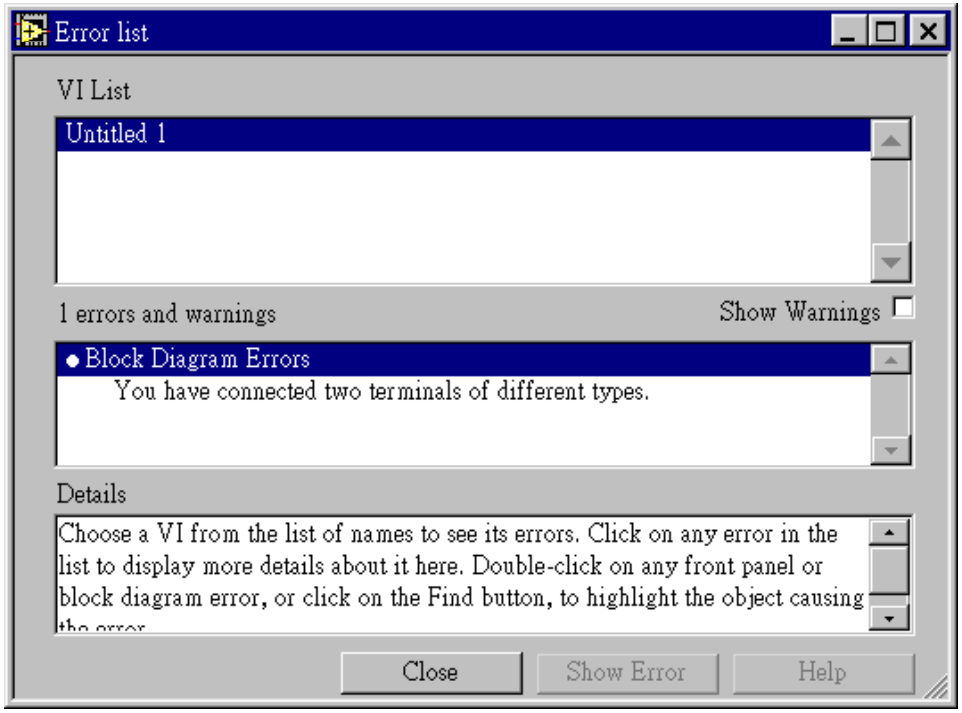
單步進入鍵 — 每按一次只執行一個步驟，當碰到SubVI或迴圈時則進入該SubVI或迴圈

單步越過鍵 — 和上一個鍵功能類似，唯一不同的是當碰到SubVI或迴圈時，不會進入該SubVI或迴圈而直接執行下一步驟

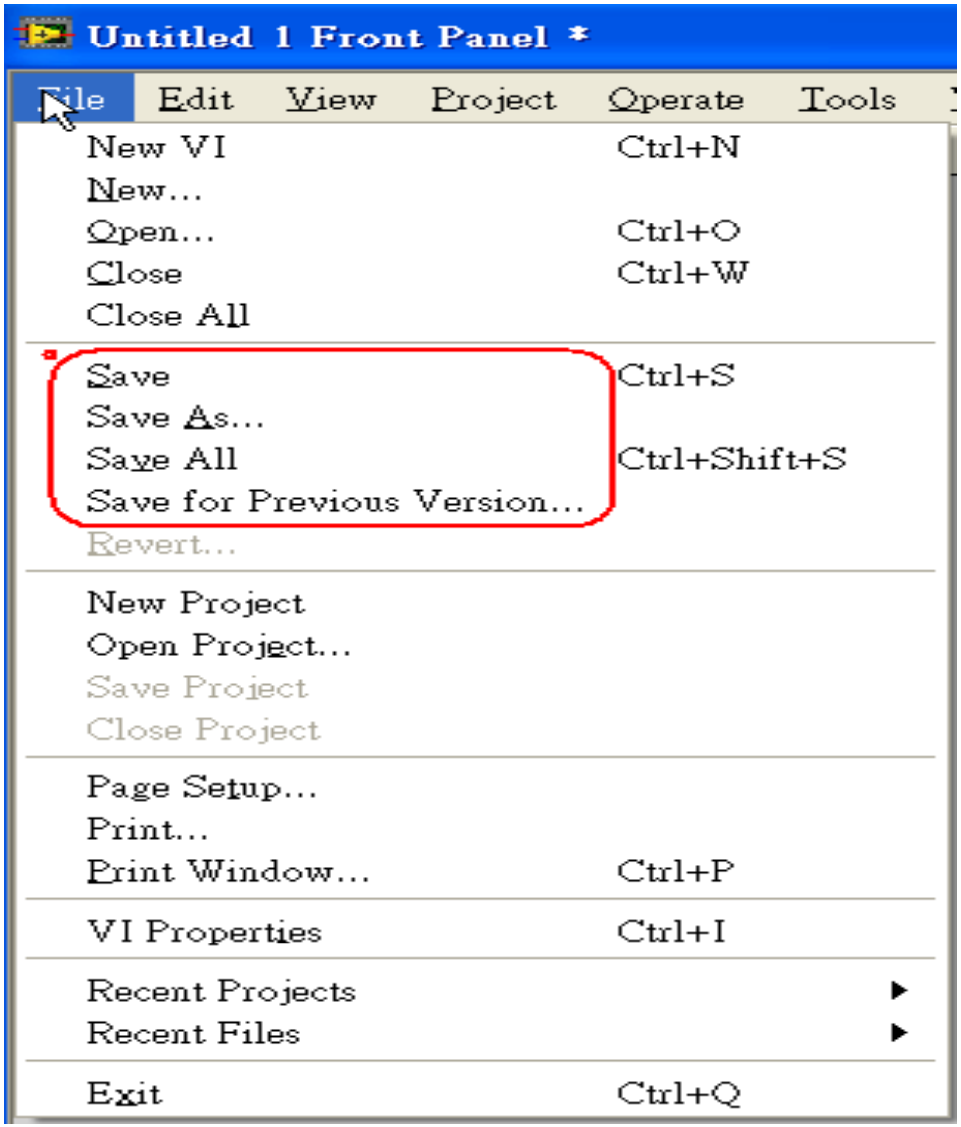
離開鍵 — 離開步進執行的過程，直接跳到最後一步驟

LabVIEW的偵錯工具

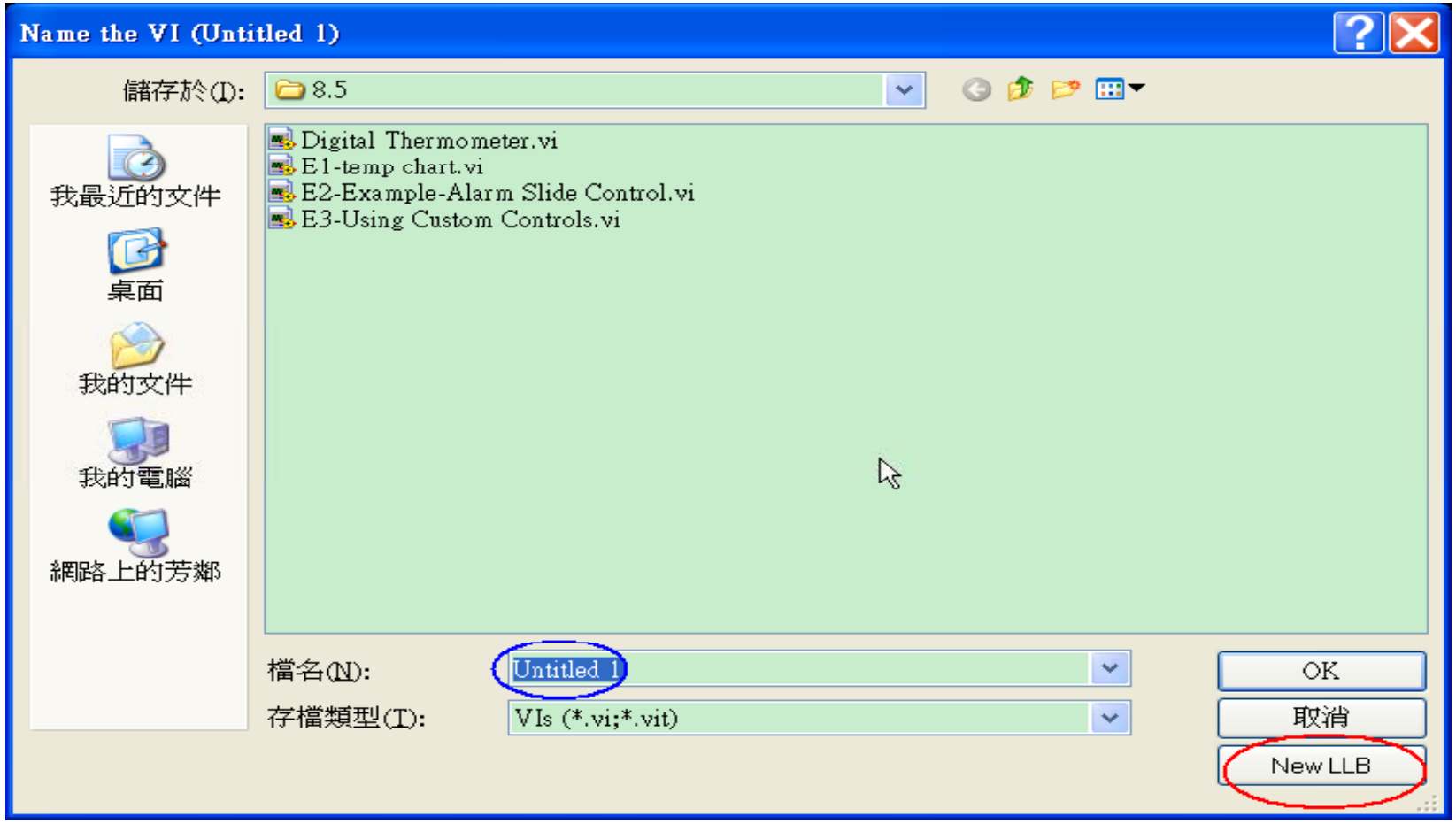
- 除了以上的除錯工具外，探測工具（**Probe tool**）也是一個很不錯的工具
  - 另外利用錯誤對話框也可以加速偵錯的時間



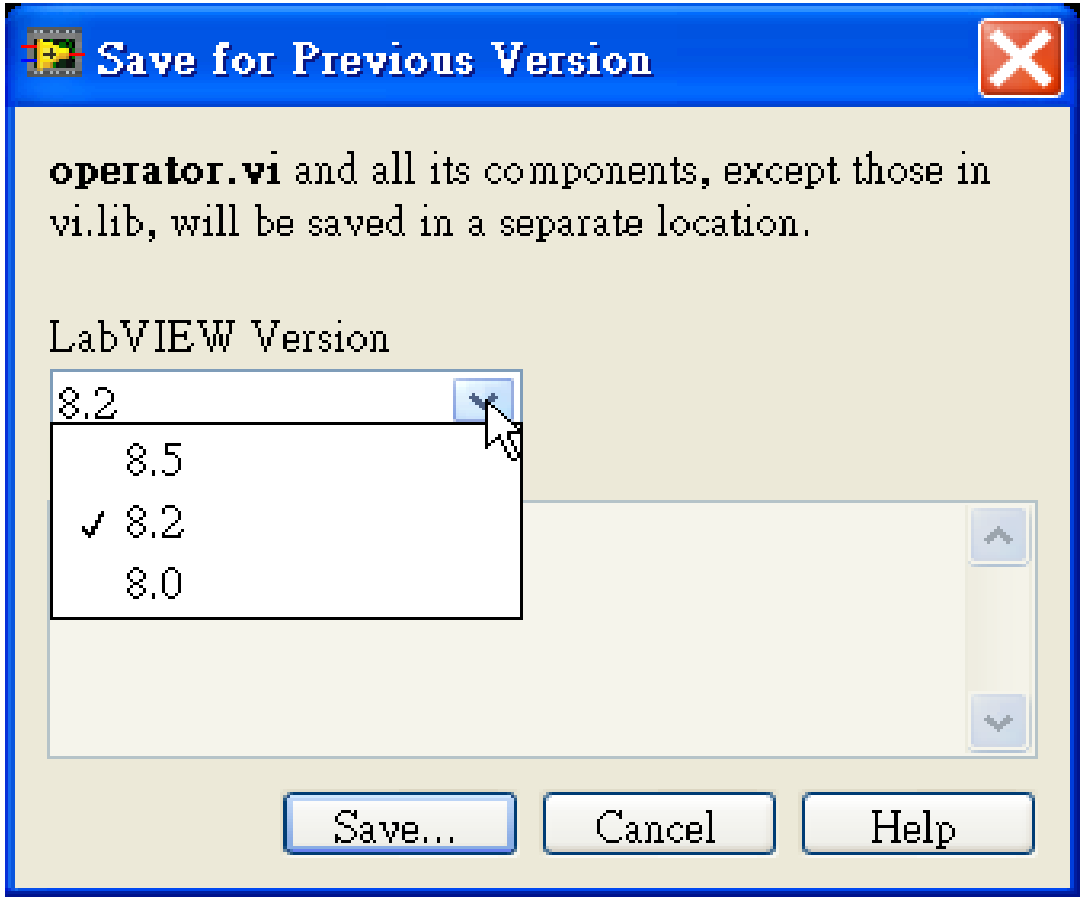
VI 程式儲存


Untitled 1 Front Panel *
File
Edit
View
Project
Operate
Tools
New VI Ctrl+N
New...
Open... Ctrl+O
Close Ctrl+W
Close All
Save Ctrl+S
Save As...
Save All Ctrl+Shift+S
Save for Previous Version...
Revert...
New Project
Open Project...
Save Project
Close Project
Page Setup...
Print...
Print Window... Ctrl+P
VI Properties Ctrl+I
Recent Projects
Recent Files
Exit Ctrl+Q

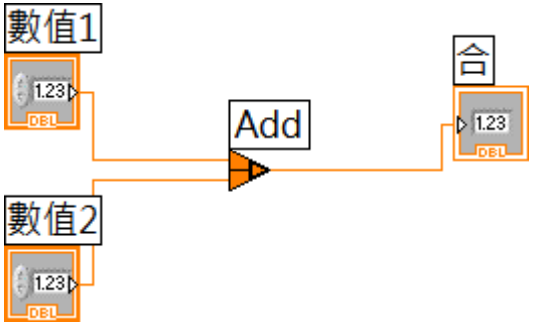
SAVE


Name the VI (Untitled 1)
儲存於(I):
8.5
Digital Thermometer.vi
E1-temp chart.vi
E2-Example-Alarm Slide Control.vi
E3-Using Custom Controls.vi
我最近的文件
桌面
我的文件
我的電腦
網路上的芳鄰
檔名(N):
Untitled 1
存檔類型(T):
VIs (*.vi;*.vit)
OK
取消
New LLB

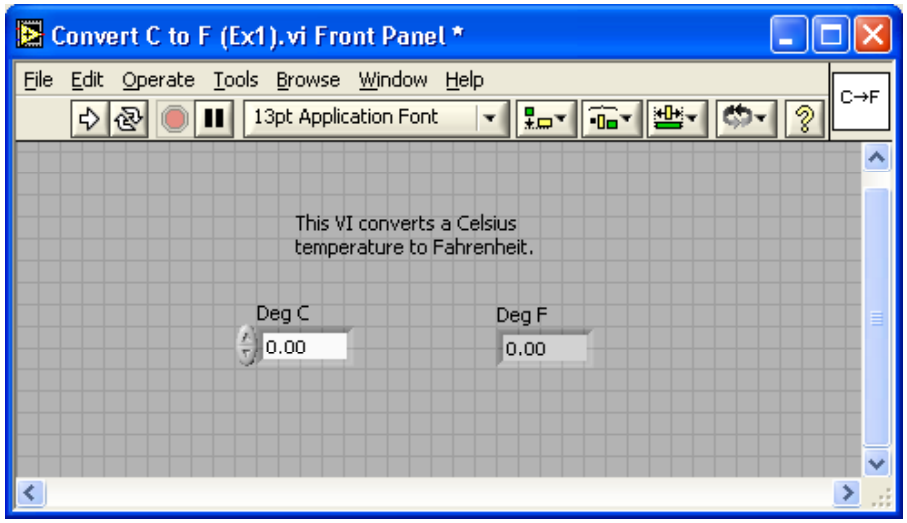
Save for Previous Version

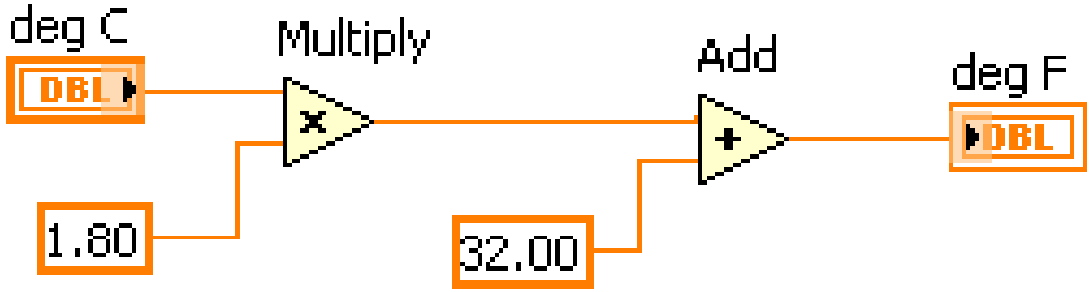


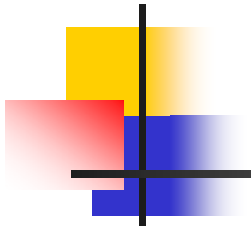
Exercise 1: Summing machine



Exercise 2 - Convert ˚C to ˚F







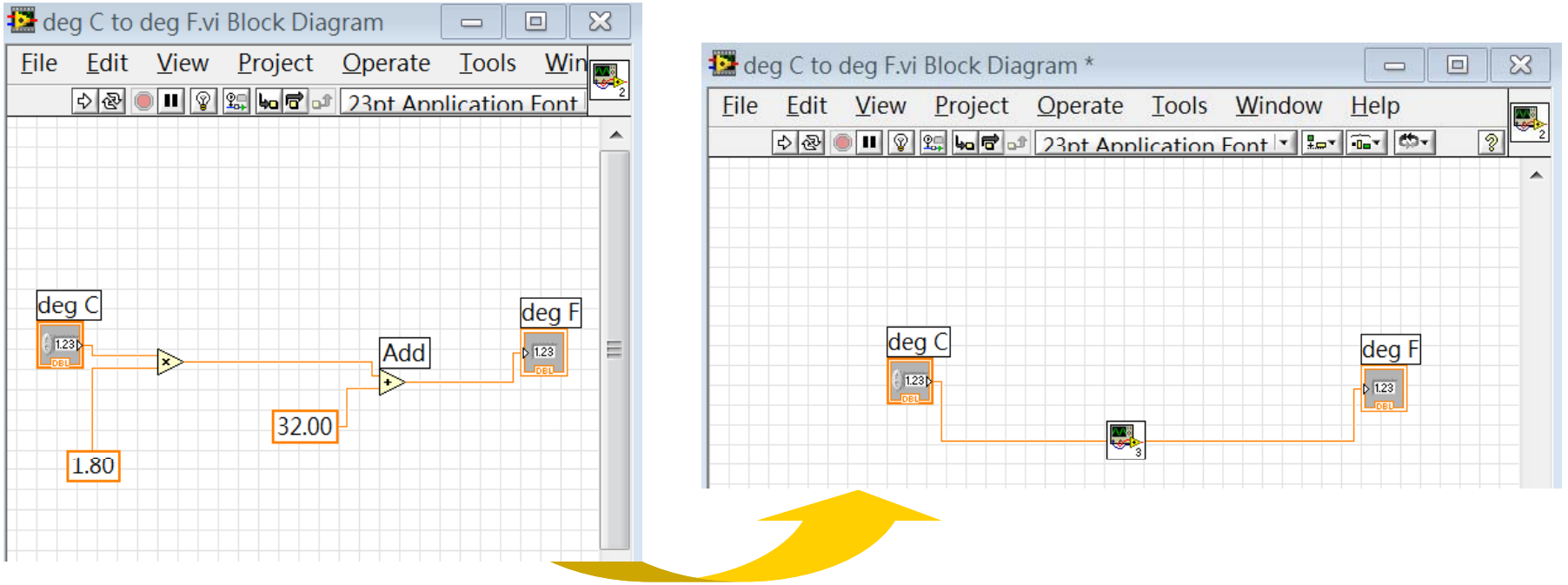
SubVI的建構

- 建立SubVI（附屬虛擬儀器）
- SubVI的圖像連結
- SubVI的應用

建立一個 SubVI

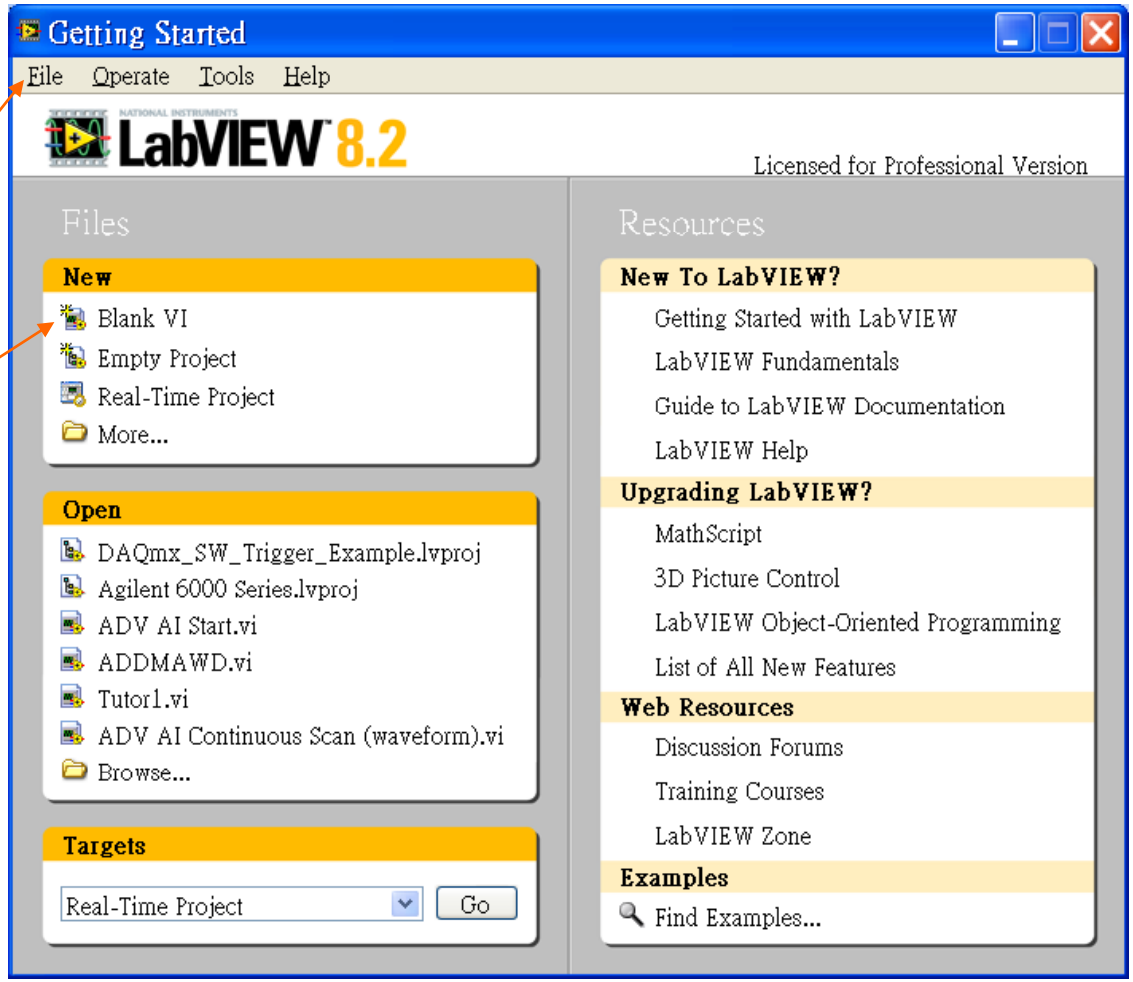
建立一個SubVI有兩種方式：

1. 直接在VI的Block Diagram中以《Select a VI》選取
2. 利用《Create SubVI》功能：
   - 首先選取想要建立SubVI的範圍（圖中虛線部份）
   - 選取Edit/Create SubVI

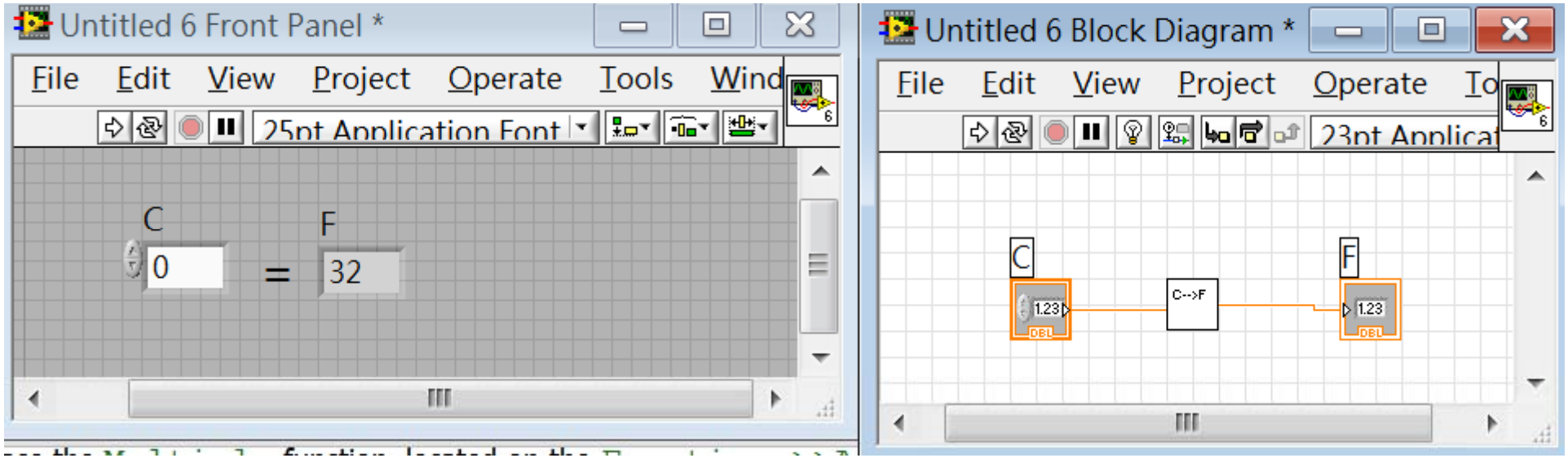


建立一個虛擬儀表: create .vi

這是LabVIEW 8.2的起始畫面，我們可以經由選取“File”裡面的“New VI”或畫面中的“Blank VI”來開始一個虛擬儀表的建立



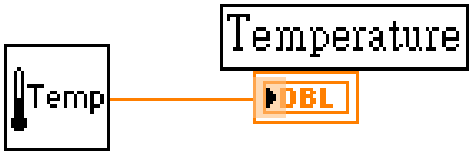
建立一個虛擬儀表: wiring terminals



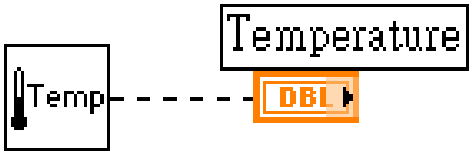
然後依序利用編輯技巧建立如上圖所示的VI

建立一個虛擬儀表：correct wiring

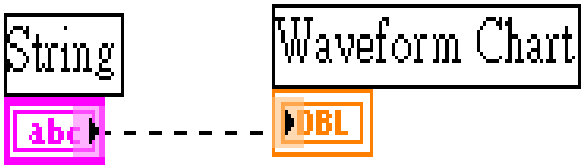
連線時應注意事項：



正確的連線



不正確的連線：Output接到Control（=Output）



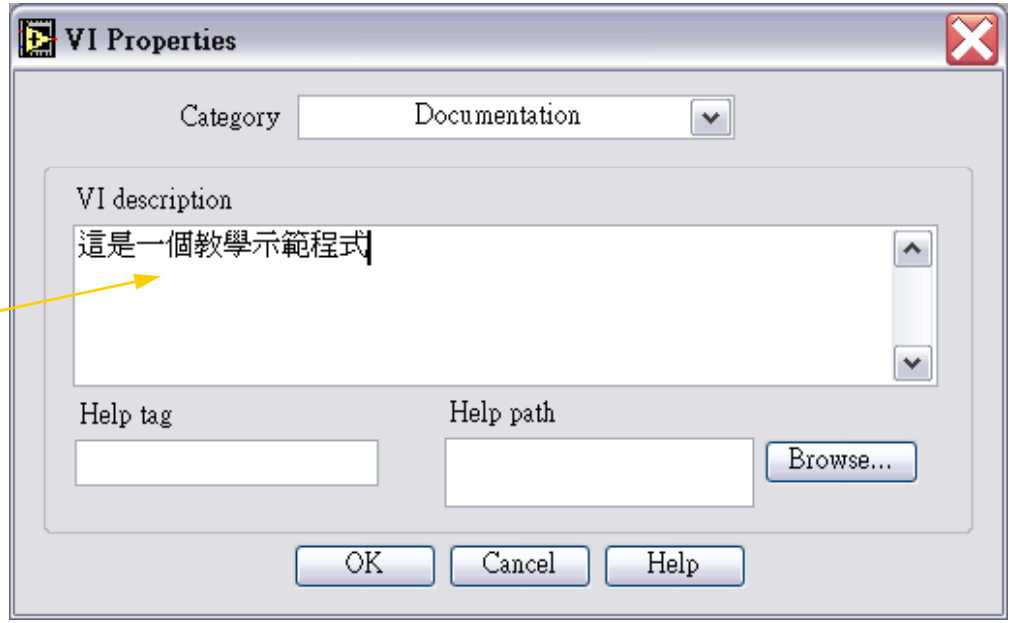
資料型態不符

建立一個虛擬儀表：documentation

先執行VI以確認功能的正確性

建立說明：選取*File》VI Properties* 開啟視窗，在目錄（Category）中選取文件（Documentation）

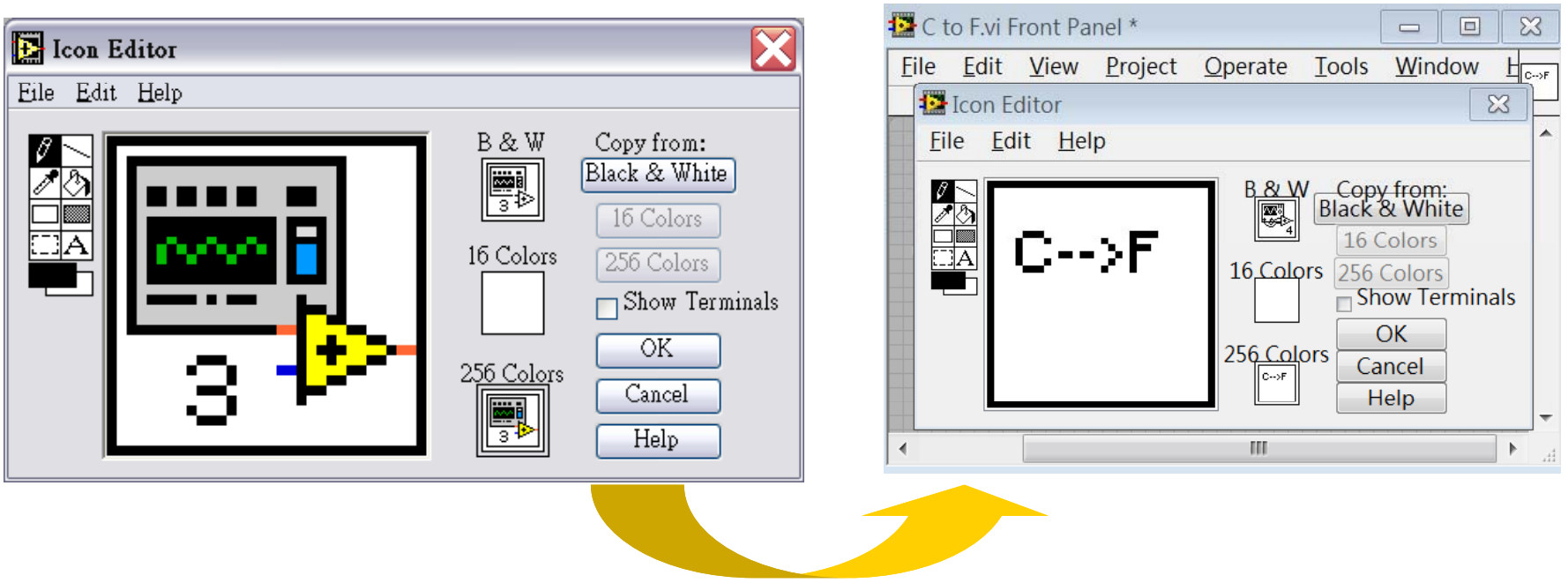
然後輸入對這VI的敘述



建立一個虛擬儀表：Icon edit

圖像（Icon）的編輯

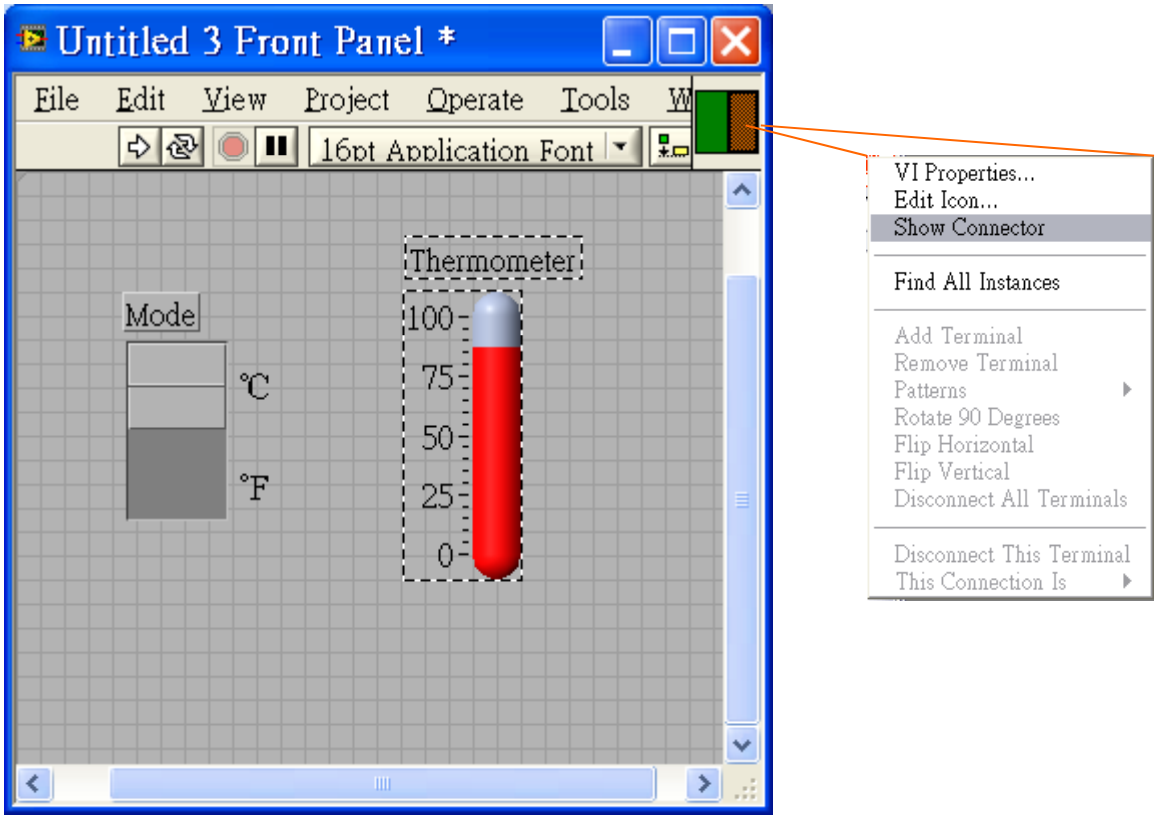
介紹如利用編輯工具來編輯Icon



建立一個虛擬儀表：icon edit

建立VI的圖像與物件的聯結

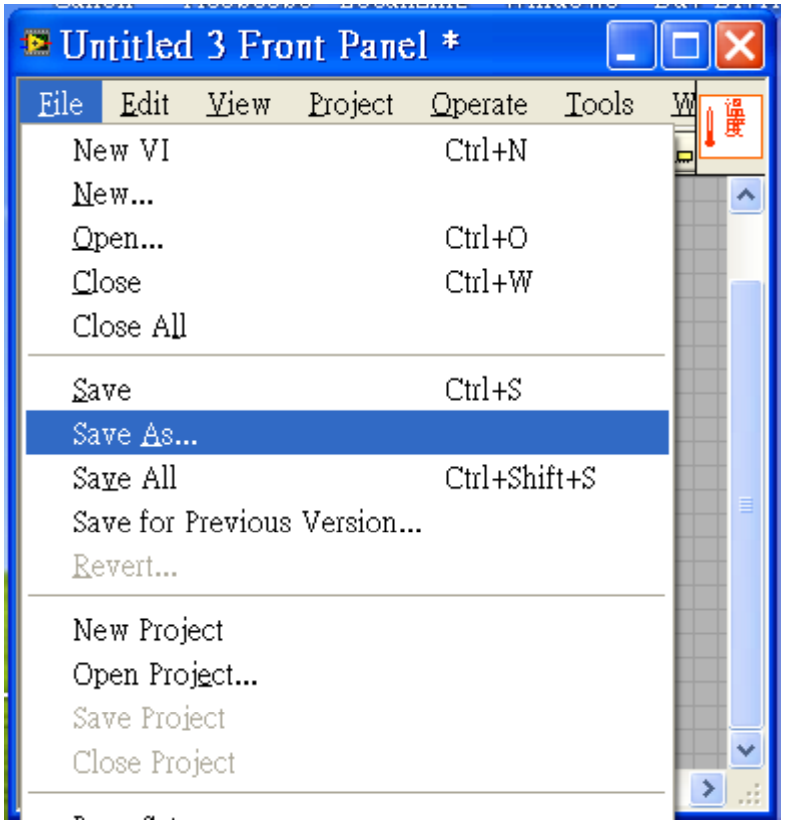
- 介紹如何利用連線工具來建立Icon的輸出入端點



建立一個虛擬儀表：儲存與讀取程式

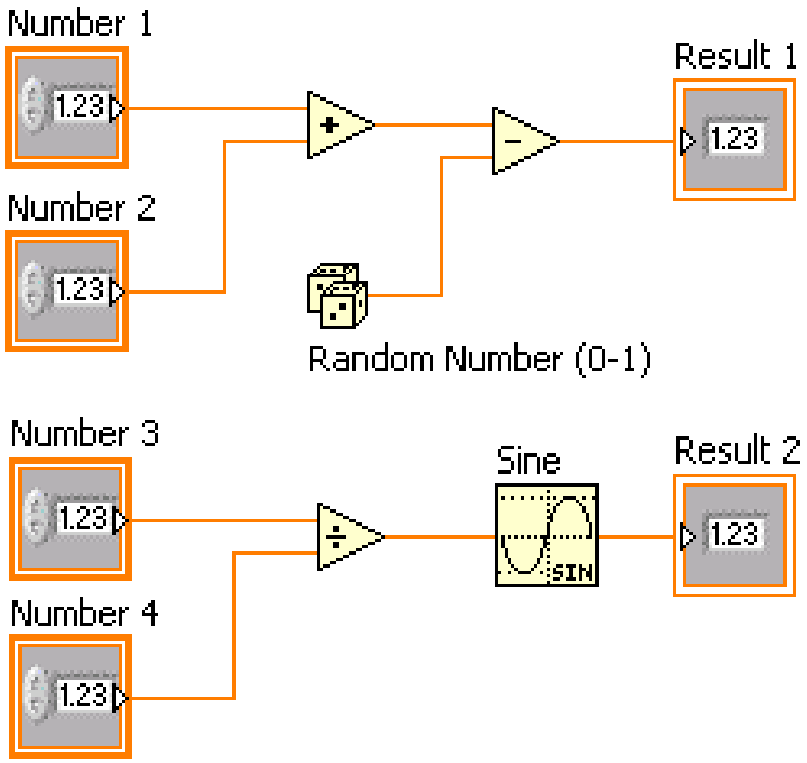
儲存與讀取VI

- 選取File/Save As來儲存VI



LabVIEW資料流程式的觀念

- 程式的執行是根據資料的流動順序而非由左至右
- 節點或SubVI的執行與否是根據所有的輸入端的資料是否到齊
- 節點會在執行完畢後提供資料給所有輸出端



LabVIEW使用上的小技巧（Tips）

- Tip 1－快速鍵
  - <Ctrl-R> 執行一個程式
  - <Ctrl-F> 尋找一個物件
  - <Ctrl-H> 叫出求助視窗
  - <Ctrl-B> 清除所有破碎的聯線
  - <Ctrl-W> 關閉Active視窗
  - <Ctrl-E> 可以交換顯示前置面版和圖示區
- Tip 2－按下**<shift>**加上滑鼠右鍵可以叫出工具面盤（Tools Palette）

建立VI的儀器面板（Front Panel）

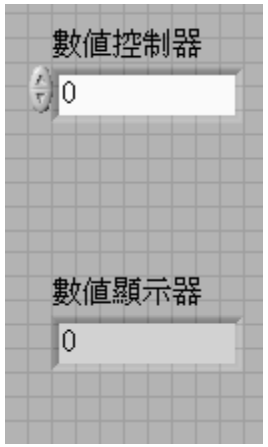
一個前置面板是由控制器與顯示器所組成

布林控制器

布林顯示器

數值控制器

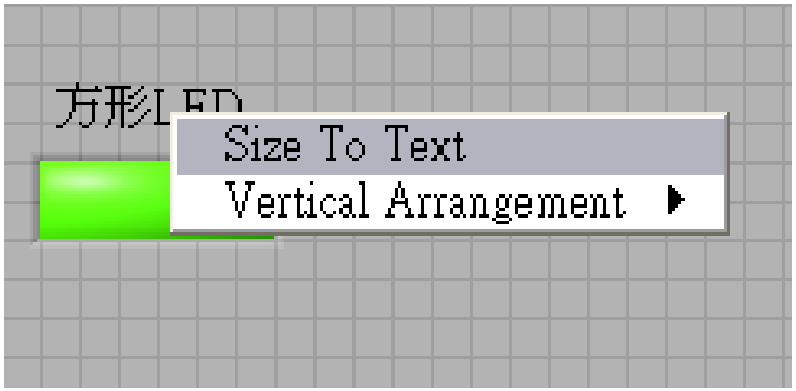
數值顯示器



如何叫出儀器面板物件的Properties

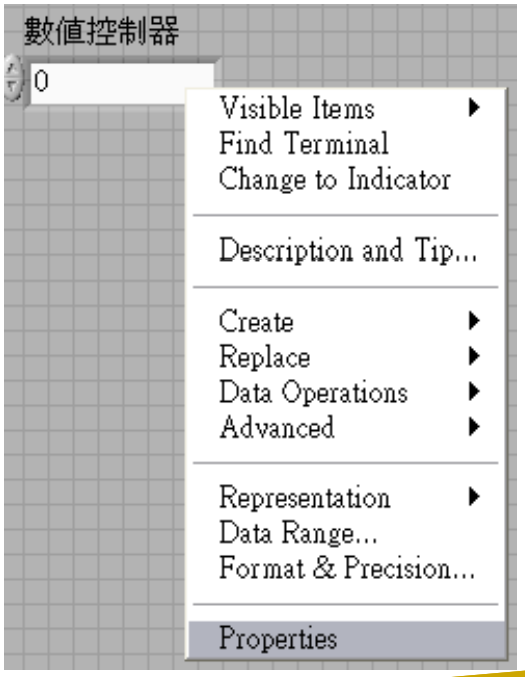
要叫出控制器的隨機式選單，只要將滑鼠對準它並按下滑鼠右鍵即可

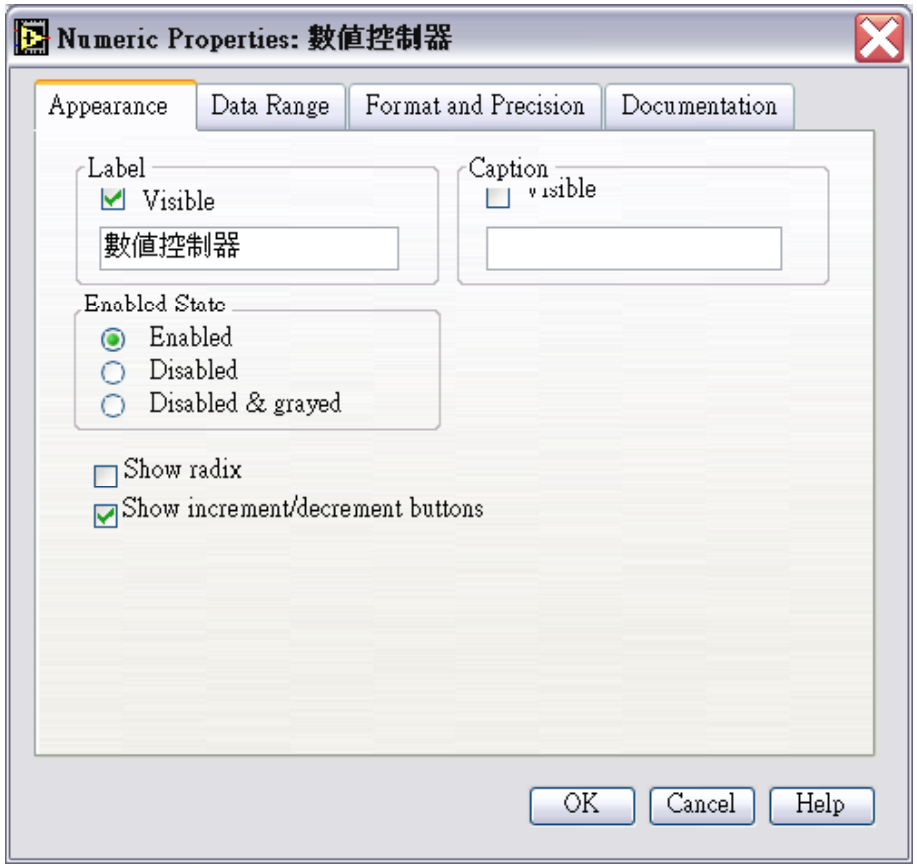
要叫出Label的隨機式選單，只要將滑鼠對準它並按下滑鼠右鍵即可



屬性頁面（Property Page）

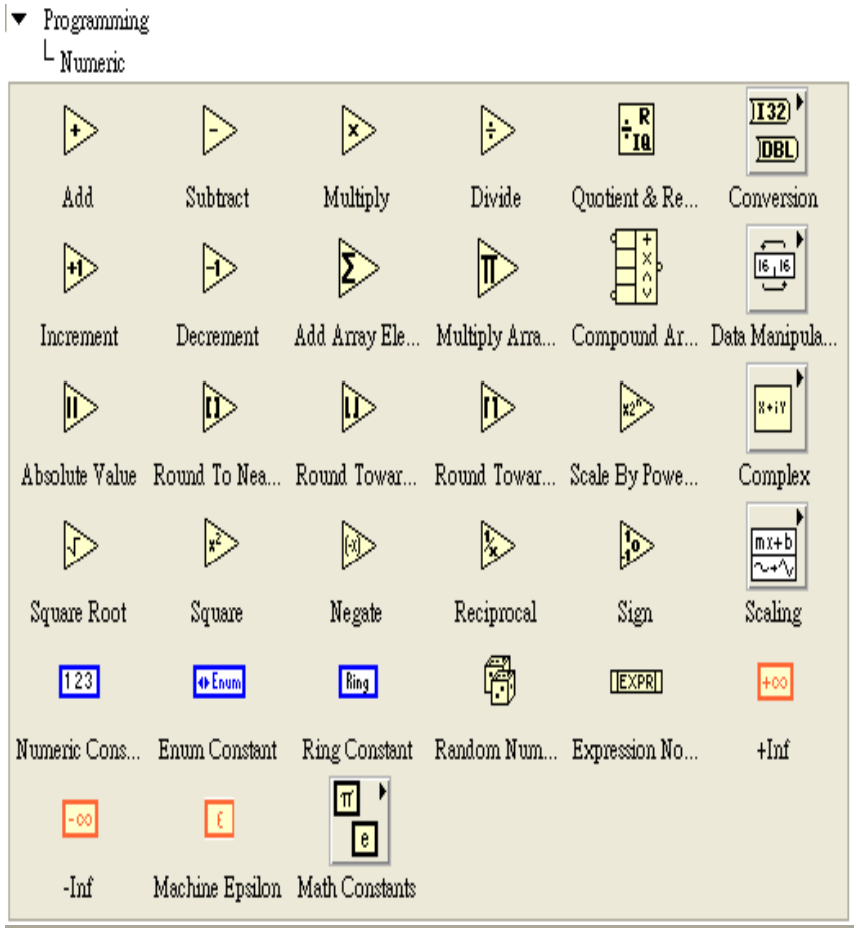
將滑鼠對準控制或顯示器
然後按下滑鼠右鍵並選擇
property即可叫出該物件的
屬性頁面





數值函數

- 一元函數
- 二元函數
- 多元函數
- 複合算術函數 ：可於Numeric或 Boolean面板上找到
- 運算式節點(Expression Node)
- 常數函數
- 進階函數



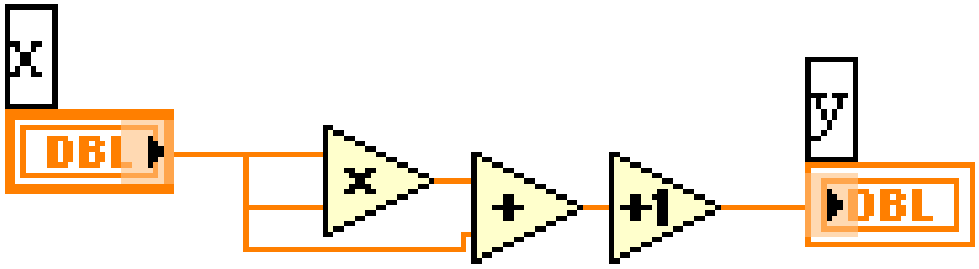
程式節點（Formula Node）

- 程式結點（ Formula Node）它提供一個可寫數學方程式的圖示空間，尤其當所要計算的方程式相當的複雜時，更能帶給我們方便，例如想完成下列方程式：

$$y = x^2 + x + 1$$

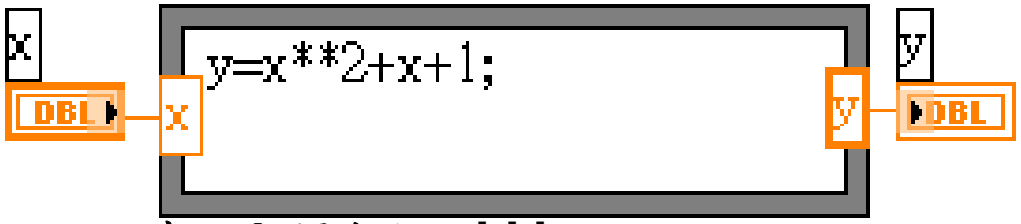
方法有二種：

1. VI方式



程式節點（Formula Node）

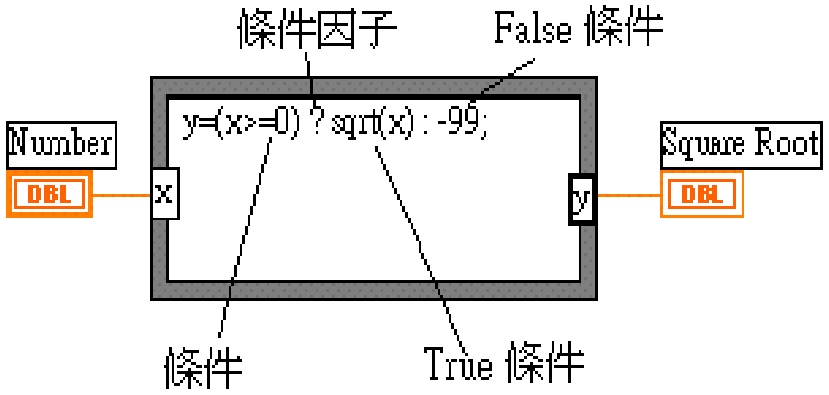
2. 程式結點（ Formula Node） VI方式：



注意：1 須自行add input, output
2 need ; at the end

- 程式結點除了上述的使用方式外，也可以做有條件的判斷

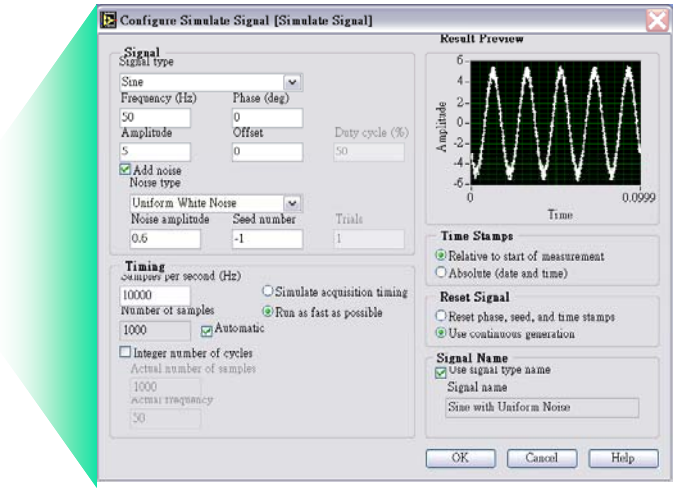
```
if (x >= 0) then
    y = sqrt(x)
    else
    y = -99
    end if
```



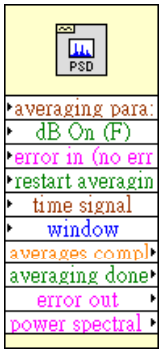
Express VIs, VIs and Functions

- Express VIs: 交談式的VIs提供可規劃的對話畫面
- Standard VIs: 模組化的VIs可透過連線的手段加以客製化
- Functions: 基本操作所需的元件不提供前置面版與圖示區

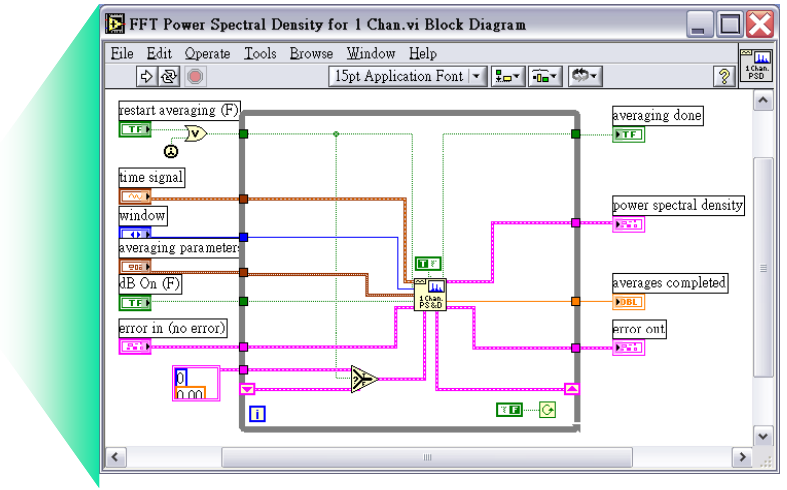
Express VIs



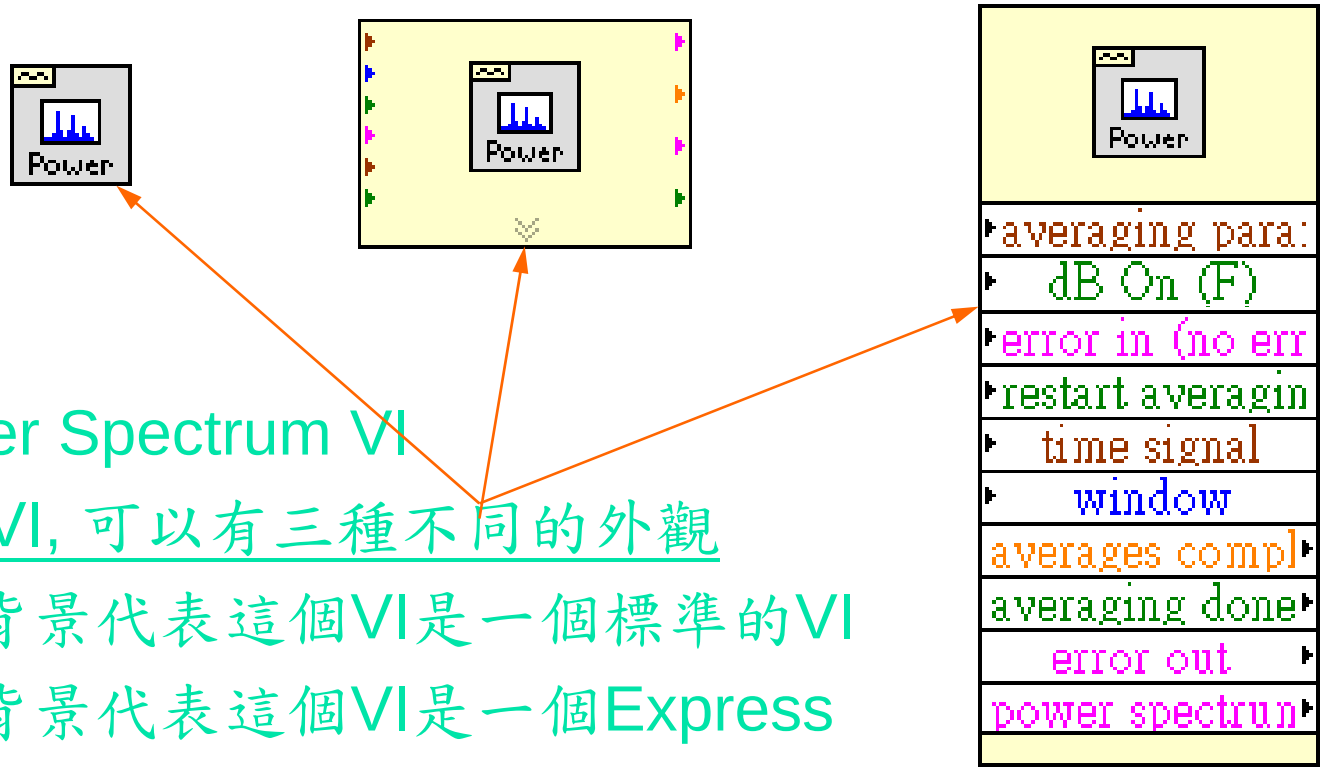
Standard VIs



Functions



圖示區VIs的外觀

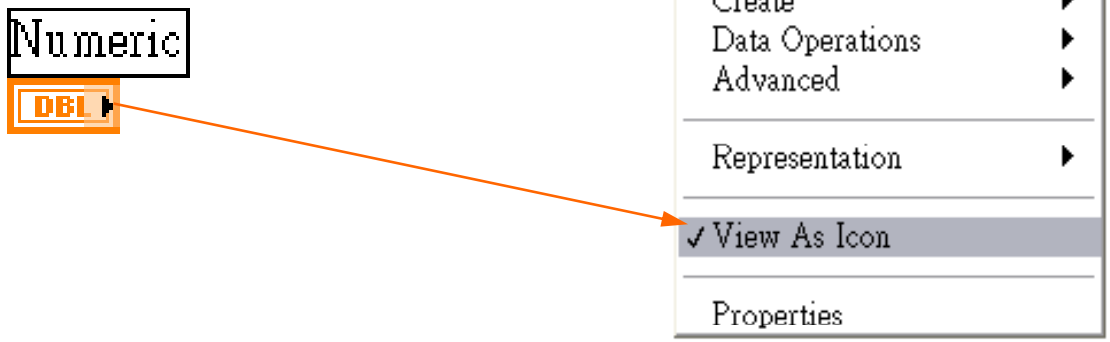


FFT Power Spectrum VI

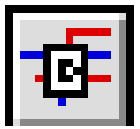
- 同樣的 VI, 可以有三種不同的外觀
- 黃色的背景代表這個VI是一個標準的VI
- 藍色的背景代表這個VI是一個Express VI

圖示區對應端的外觀

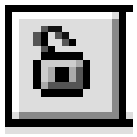
- 對應端是用來作為前置面版與圖示區之間交換資料用的一個窗口
- 使用者可以利用按滑鼠點選View As Icon來改變外觀



如何使用求助視窗

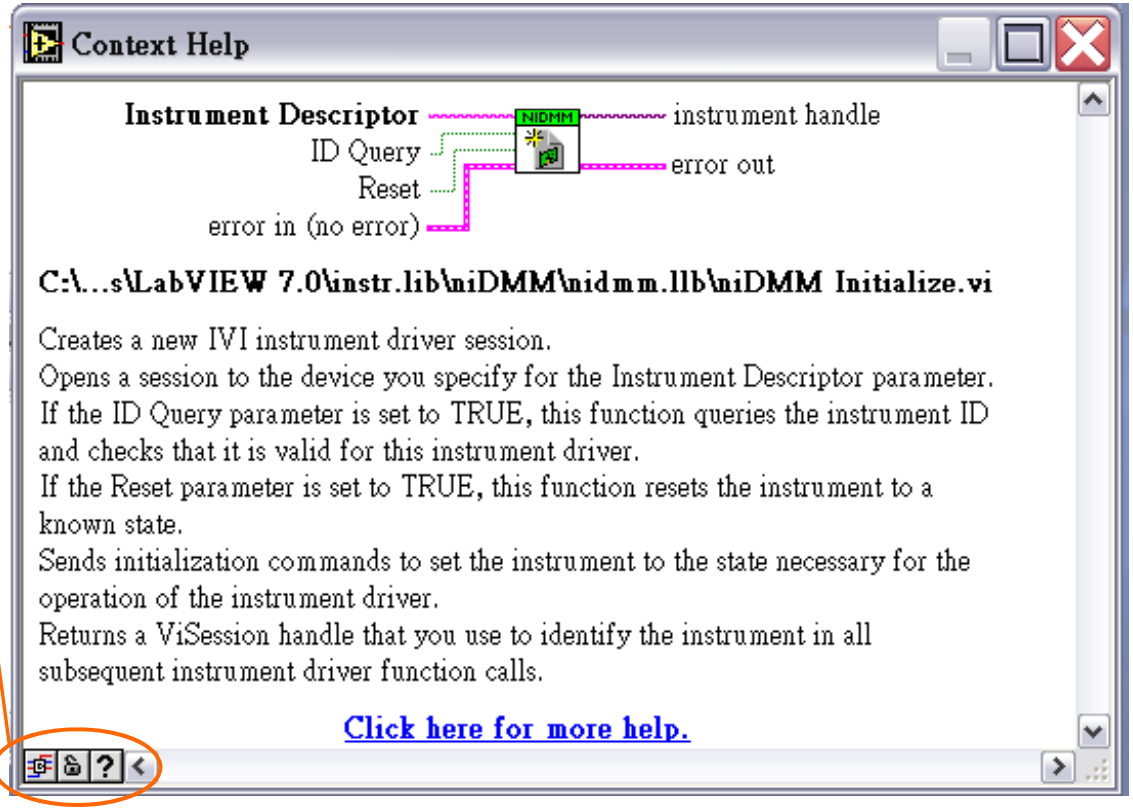


按下這個按鍵可以顯示SubVI的詳細路徑



按下這個按鍵可以鎖住求助視窗

按下這個按鍵可以顯示更詳細的說明



LabVIEW的偵錯工具

- 圖示區工具棒上的偵錯按鍵的應用
  - 介紹圖示區工具棒區中的相關除錯工具

執行提示鍵 — 可以清楚看到程式執行的順序與資料（Data）的狀態

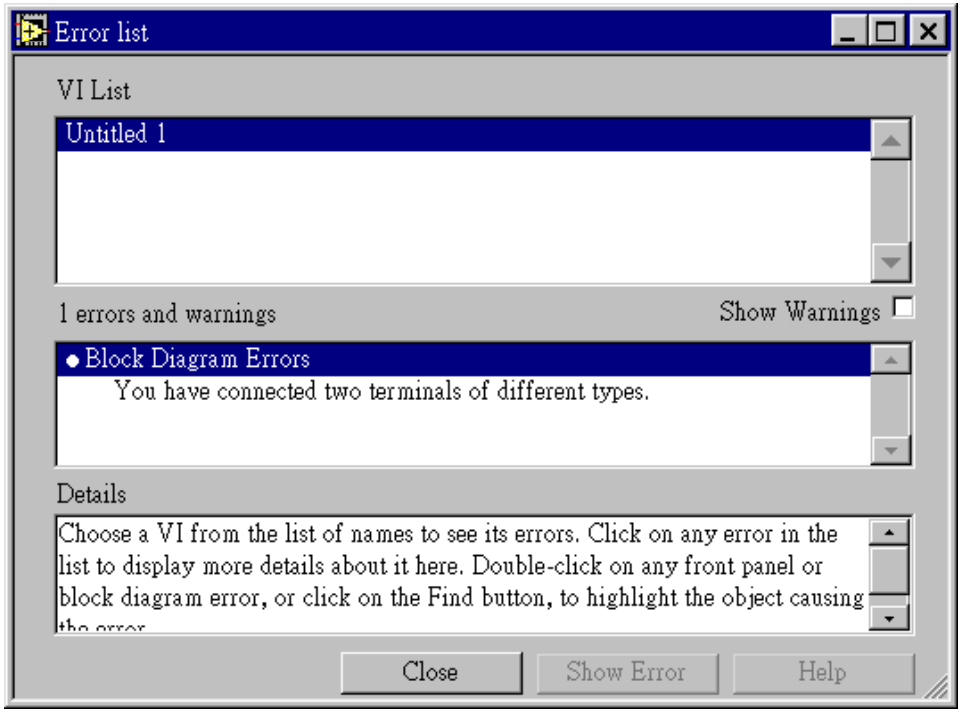
單步進入鍵 — 每按一次只執行一個步驟，當碰到SubVI或迴圈時則進入該SubVI或迴圈

單步越過鍵 — 和上一個鍵功能類似，唯一不同的是當碰到SubVI或迴圈時，不會進入該SubVI或迴圈而直接執行下一步驟

離開鍵 — 離開步進執行的過程，直接跳到最後一步驟

LabVIEW的偵錯工具

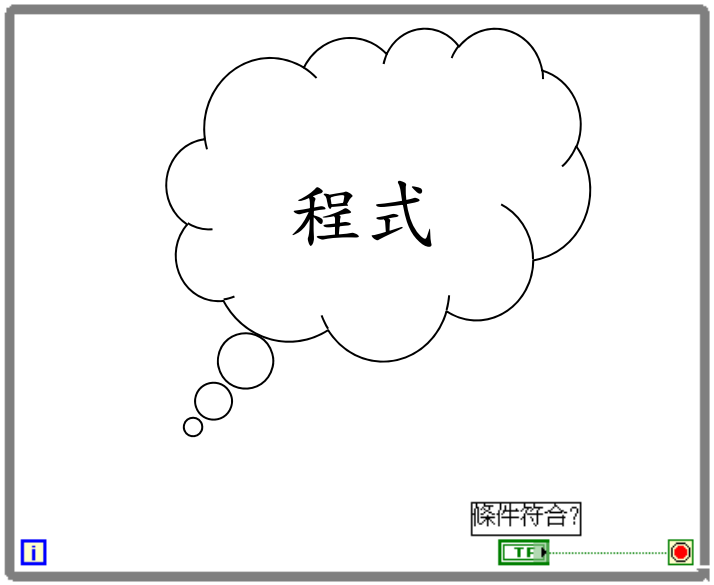
- 除了以上的除錯工具外，探測工具（Probe tool）也是一個很不錯的工具
- 另外利用錯誤對話框也可以加速偵錯的時間



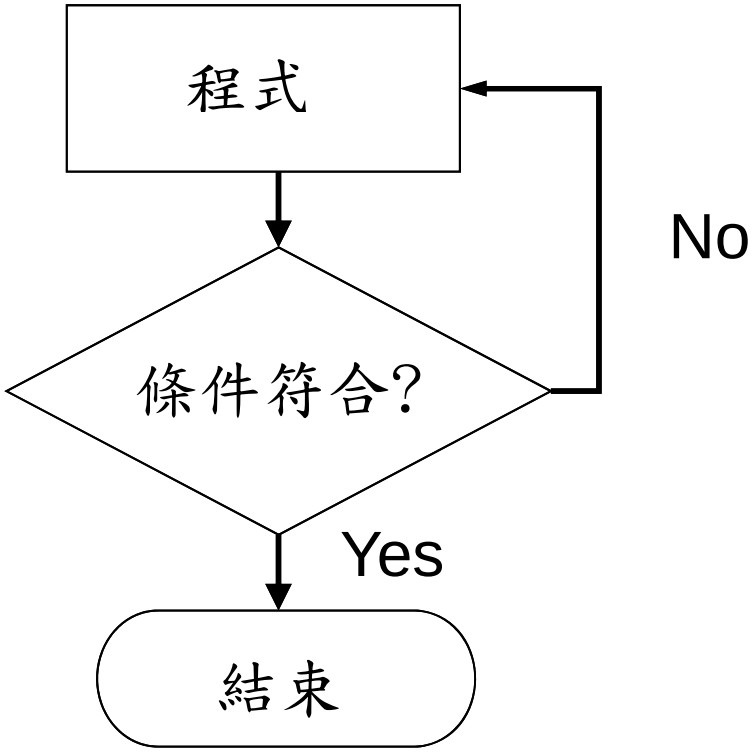
迴圈與圖形的使用

- 無限重複迴圈 （ While Loop ）
- 控制重複迴圈 （ For Loop ）
- 暫存器 （ Shift Register ）
- 記錄圖 （ Chart ）
- 數據圖 （ Graph ）

無限重複迴圈（While Loop）



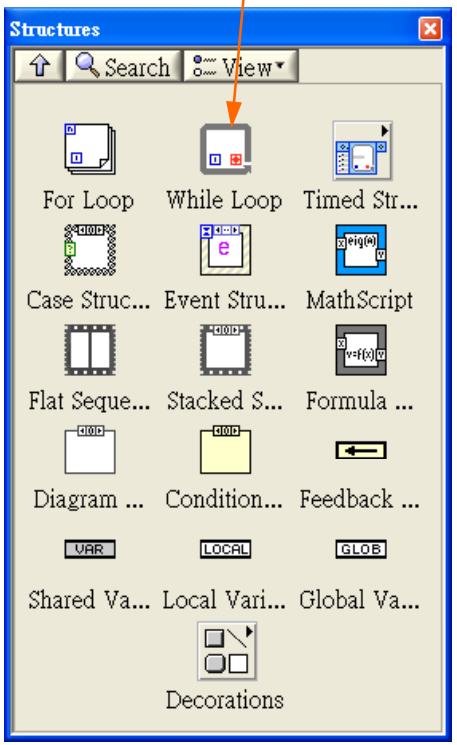
LabVIEW的While Loop



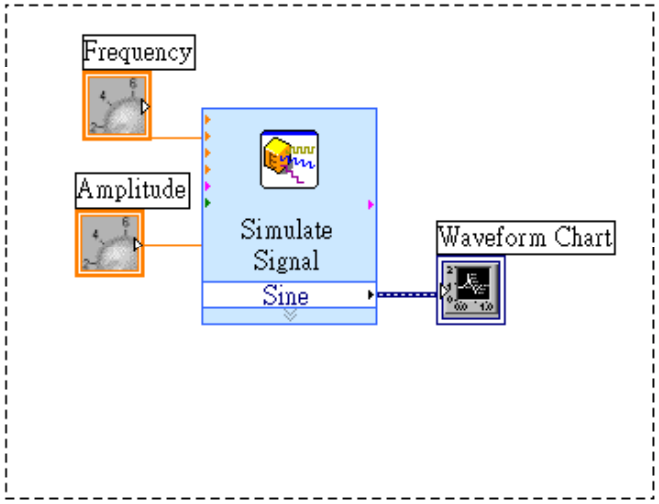
流程圖

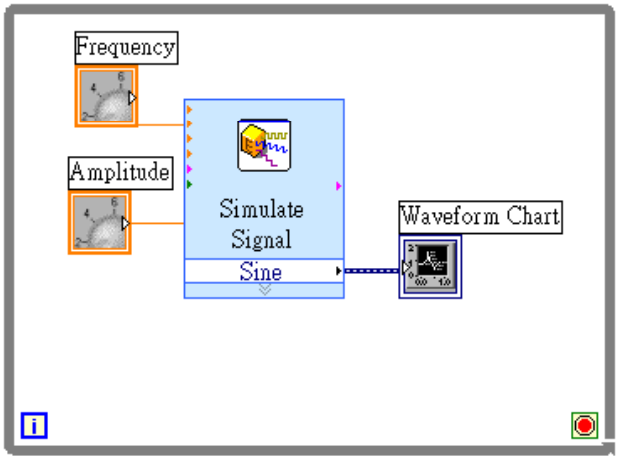
無限重複迴圈（While Loop）

選擇While Loop



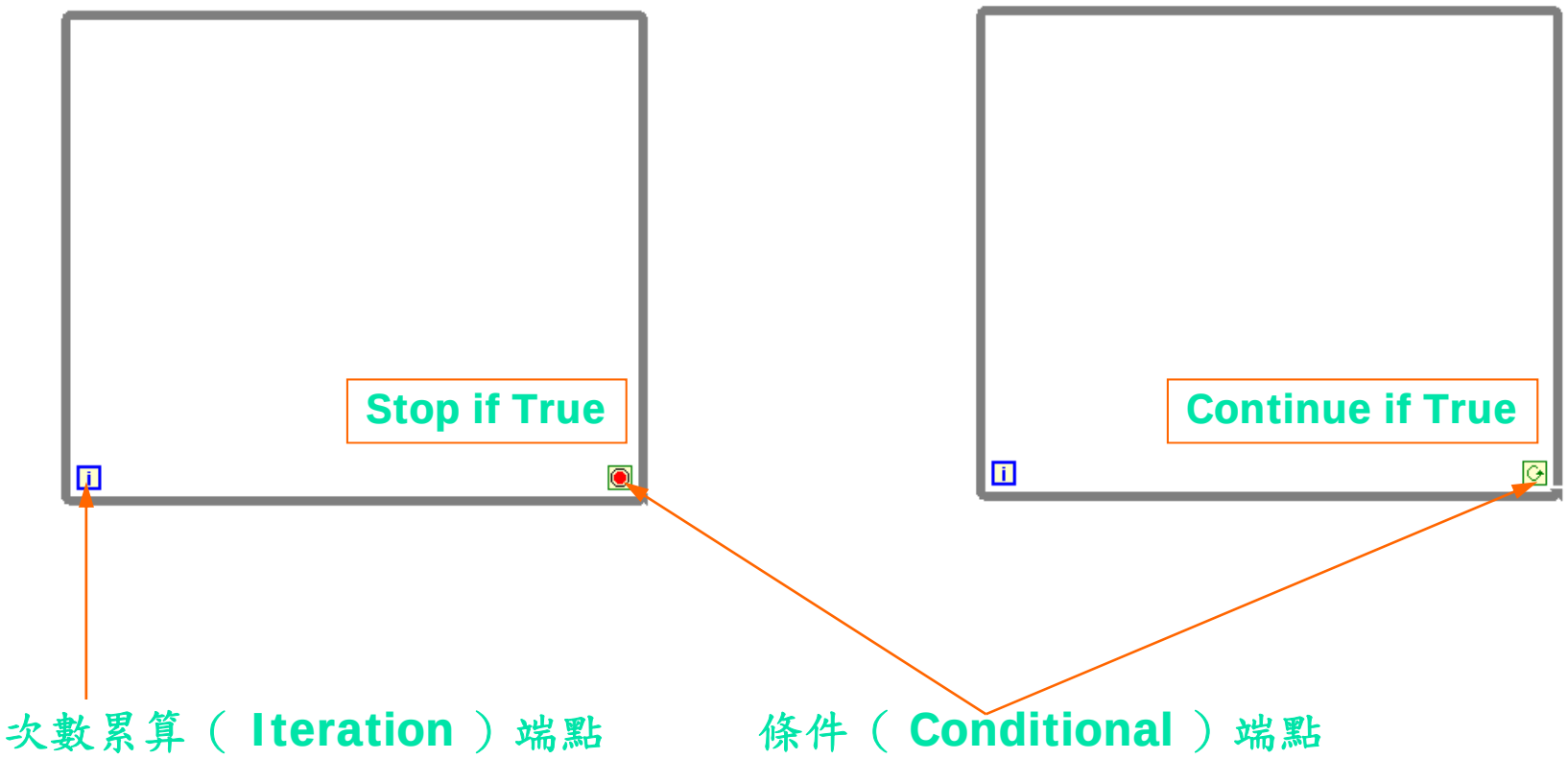
將需要重複執行的程式部分框入迴圈中





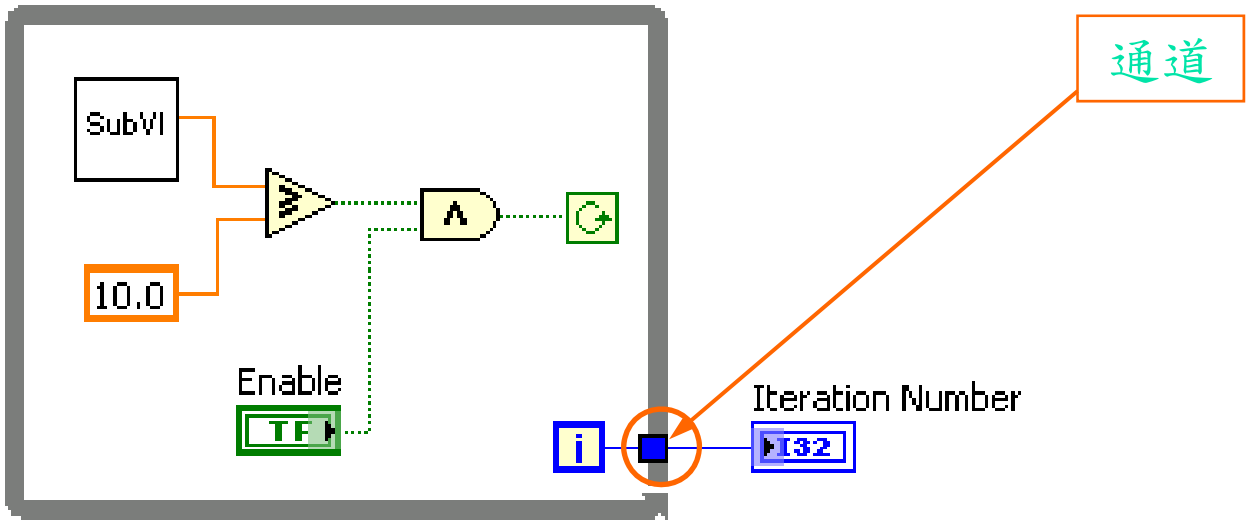
無限重複迴圈（While Loop）

- 設定While Loop的條件

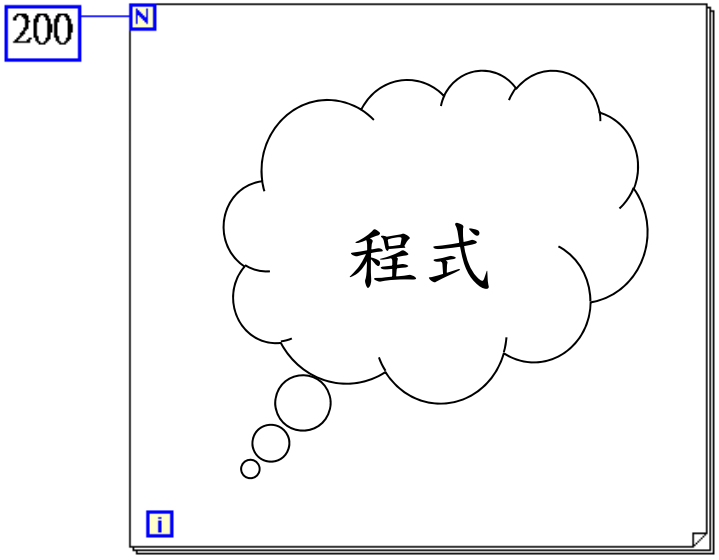


架構的通道（Structure Tunnels）

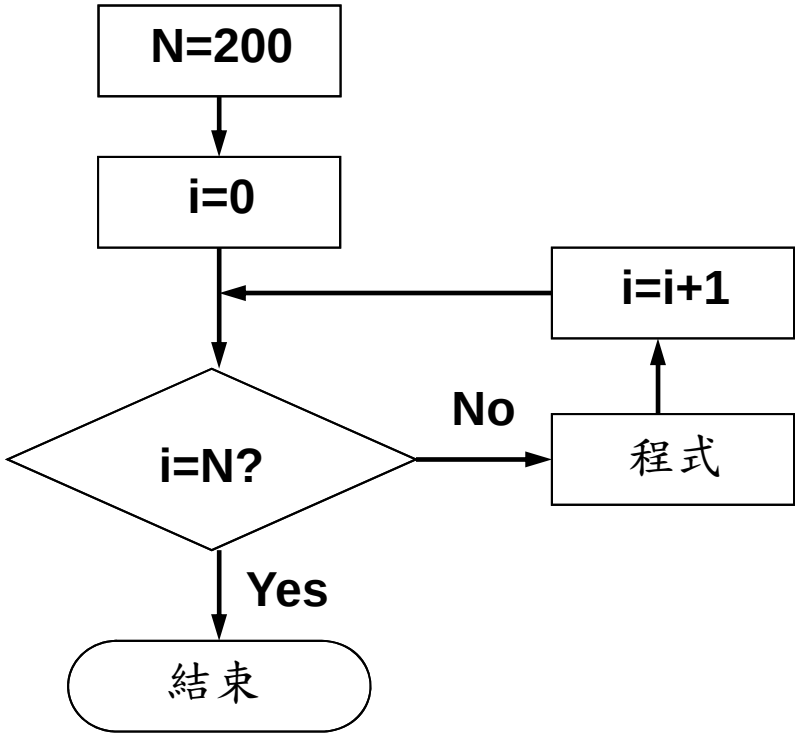
- 通道的功能是將資料放入架構或輸出到架構外
- 通道是以一個方塊的形式出現在架構的邊框（border）而顏色則是取決於資料的形態
- 當通道將資料放入一個迴圈時，這個迴圈會等到資料到達後才會執行
- 迴圈結束後通道內的資料才會被送出



控制重複迴圈（For Loop）



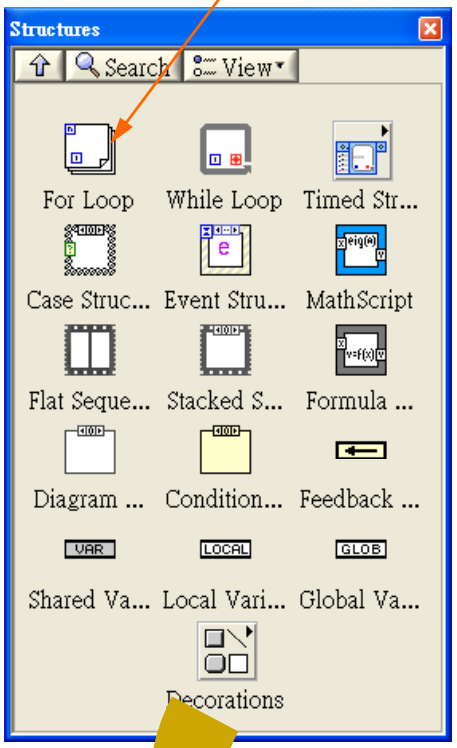
LabVIEW的For Loop



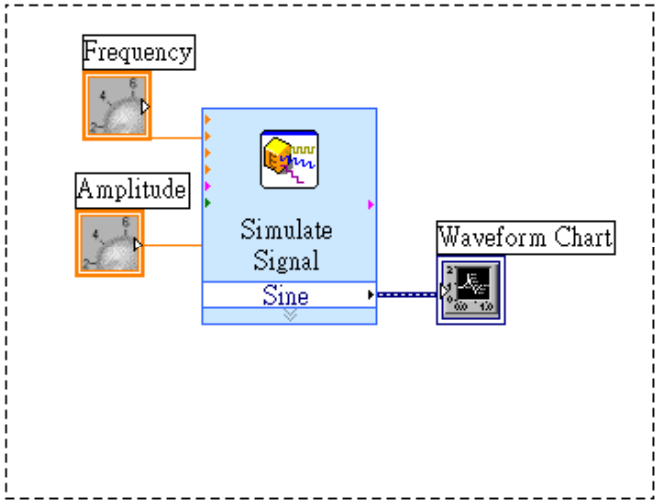
流程圖

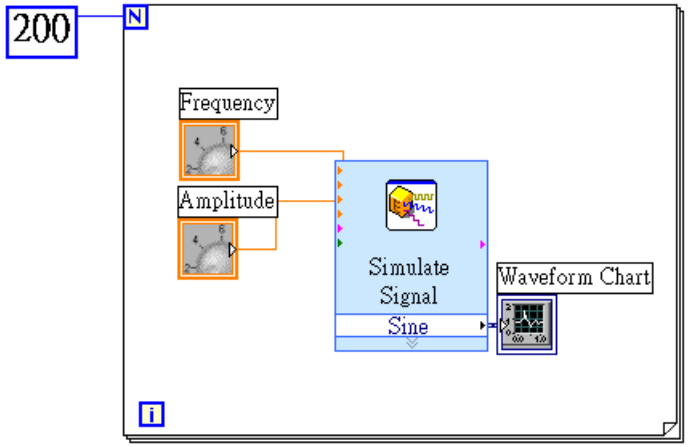
控制重複迴圈（For Loop）

選擇For Loop



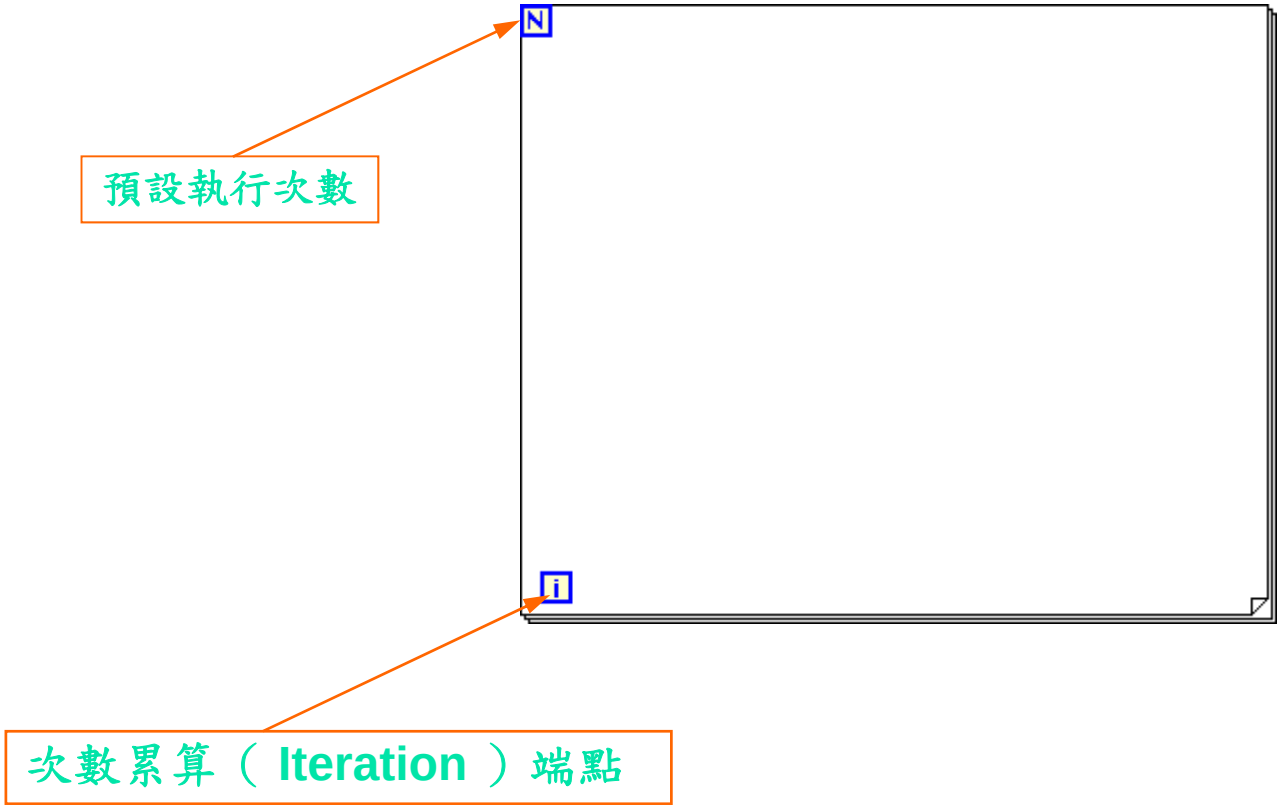
將需要重複執行預設次數的程式部分框入迴圈中





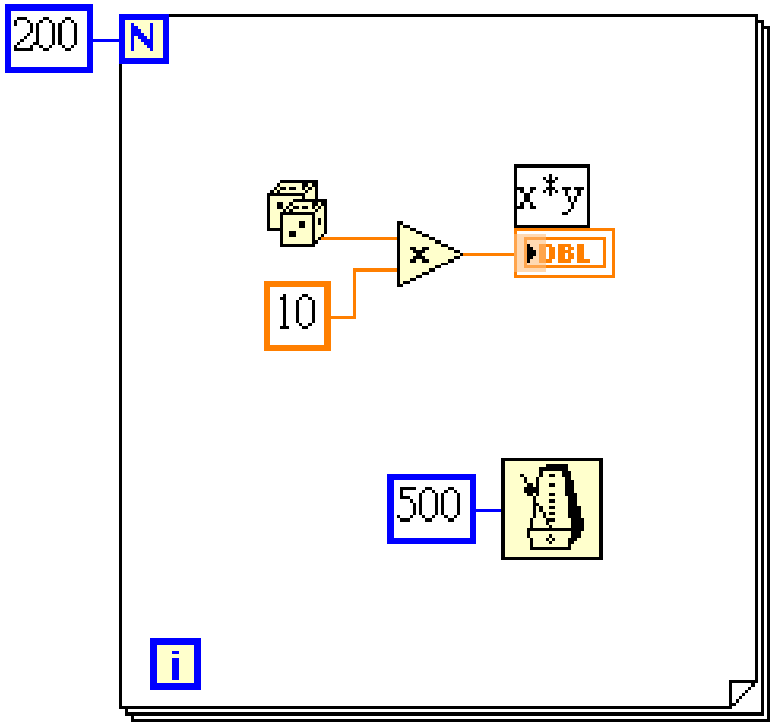
控制重複迴圈（For Loop）

- 設定For Loop的條件



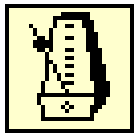
控制重複迴圈（For Loop）

- 在迴圈當中的程式將被執行N次
- 執行的次數是預先設定好的

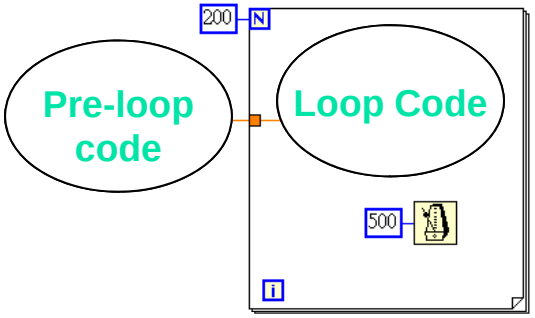


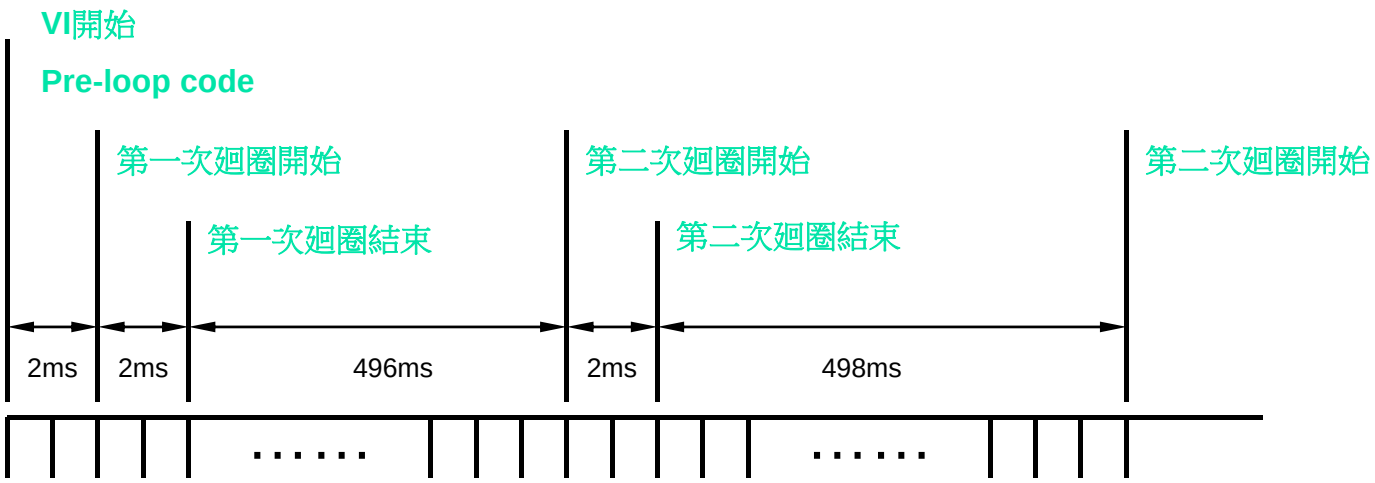
迴圈內時間的控制

Wait Until Next ms Multiple



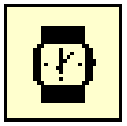
Functions»Time & Dialog palette





迴圈內時間的控制

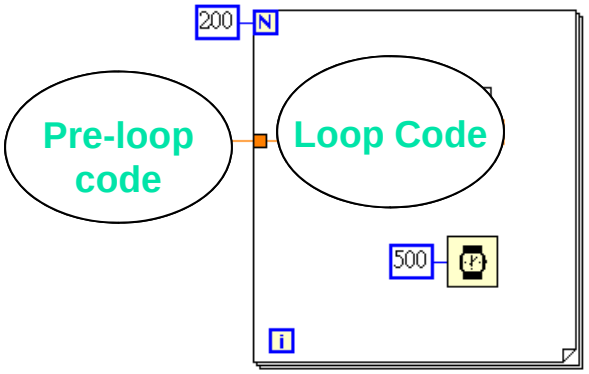
Wait (ms)

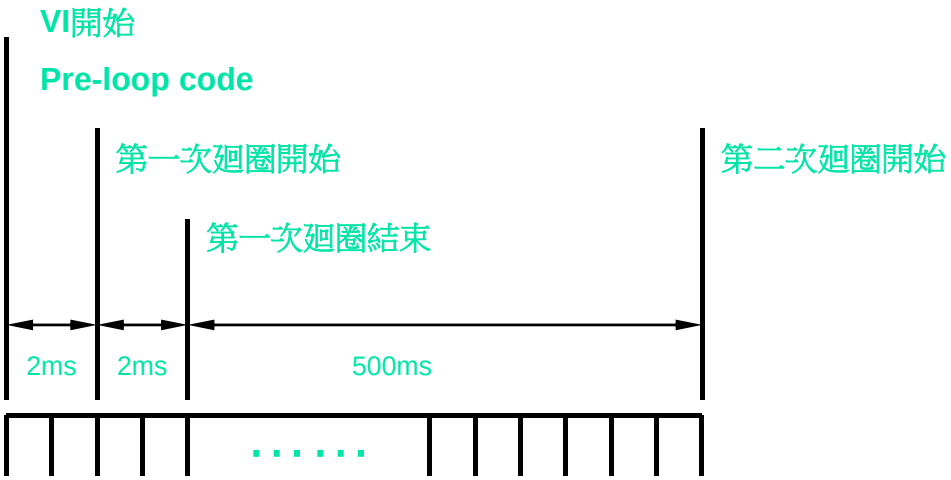


Functions»Time & Dialog palette

Time Delay

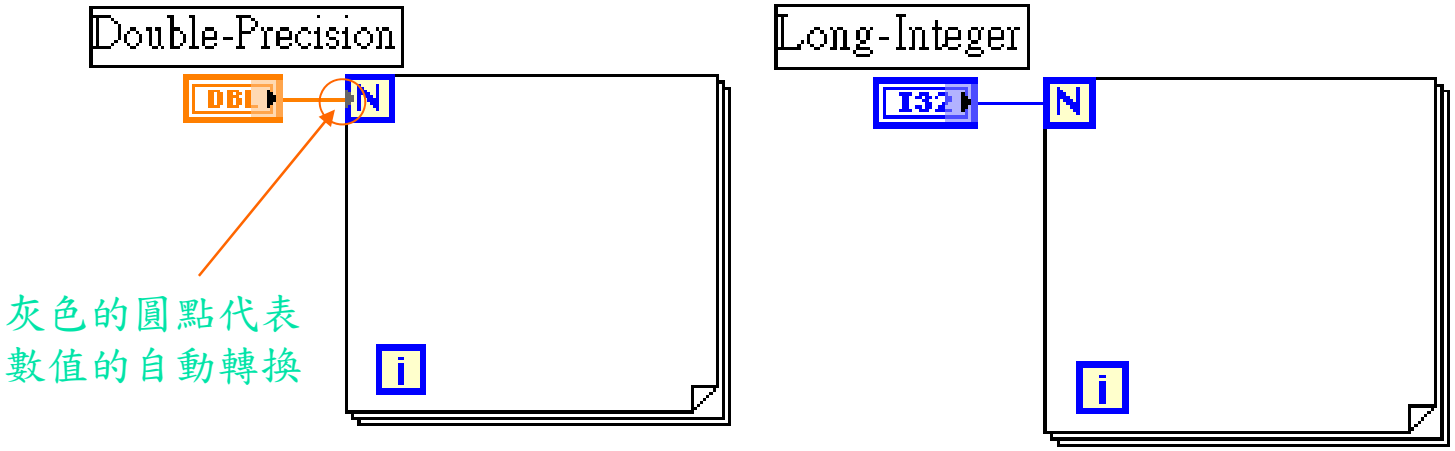
Functions»Time & Dialog palette





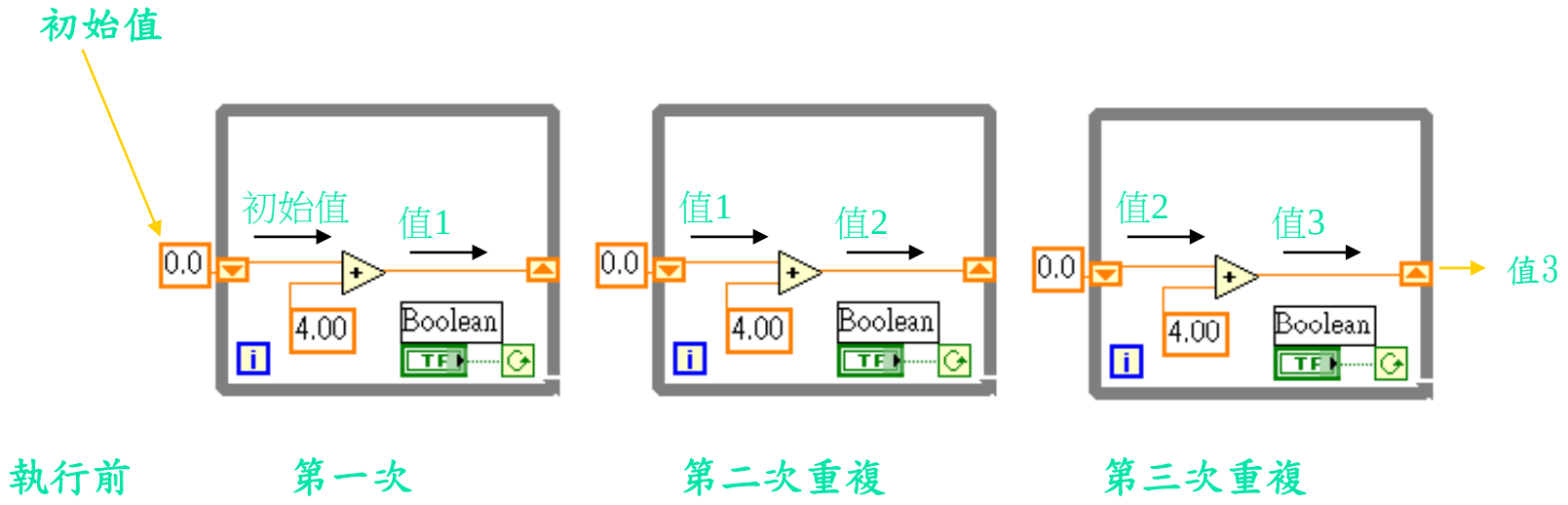
數值的轉換

- LabVIEW中的數值預設為double-precision (8 bytes) 或 long integer (4 bytes)
- LabVIEW會自動轉換數值為正確的形態
- For Loop count terminal 的形態是long integer (4 bytes)



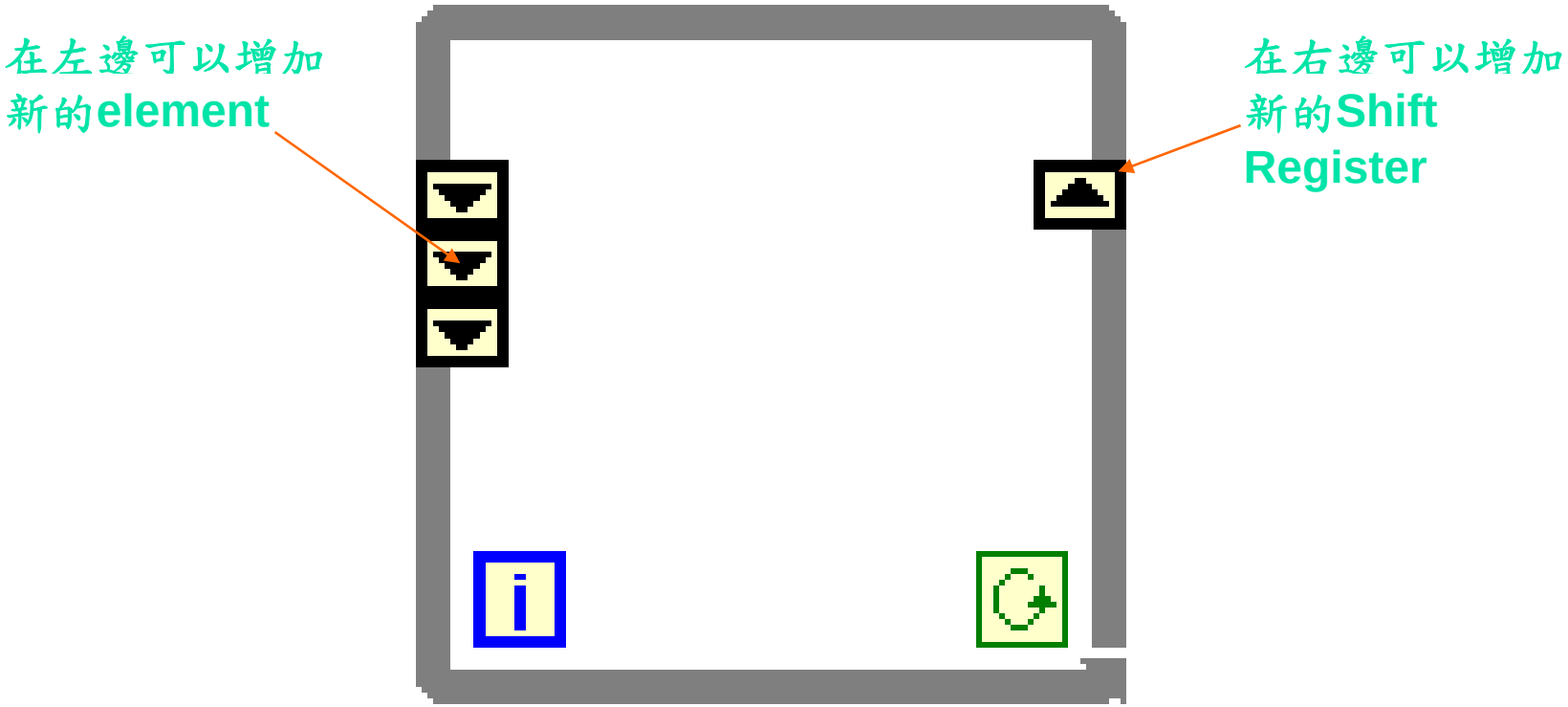
移位暫存器（Shift Register）

- 暫存器（ Shift Register ）：
  - 將滑鼠放在While Loop的Border上，按下右鍵選擇《Add Shift Register》
  - 資料流動的方式如下圖所示：



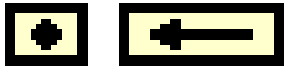
移位暫存器（Shift Register）

- 右邊的Terminal儲存這一個iteration完成的值
- 左邊的Terminal儲存下一個iteration的初始值



回授結點（Feedback Nodes）

Feedback Node



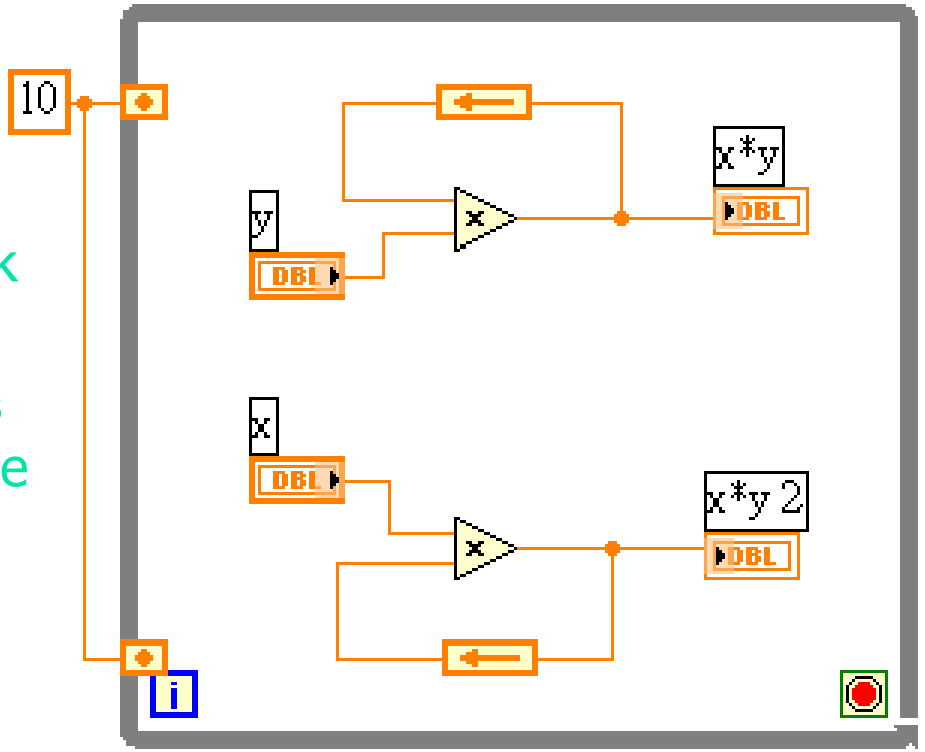
Functions»
Structures palette

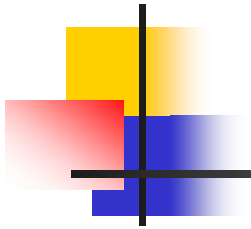
- 在 While Loop或For Loop內當你將一個 subVI、function或group of subVIs的輸出接到同一個subVI、function或group of subVIs的輸入端時，它會自動出現
- 當迴圈完成一個iteration時會自動儲存資料，並將資料送到下一個iteration，並且會自動轉換任何的資料型態

回授結點（Feedback Nodes）

使用Feedback Node有兩種方式：

1. 將輸出端直接連接到輸入端就會自動產生一個 feedback node
2. 從 **Functions»Structures** palette中選取feedback node

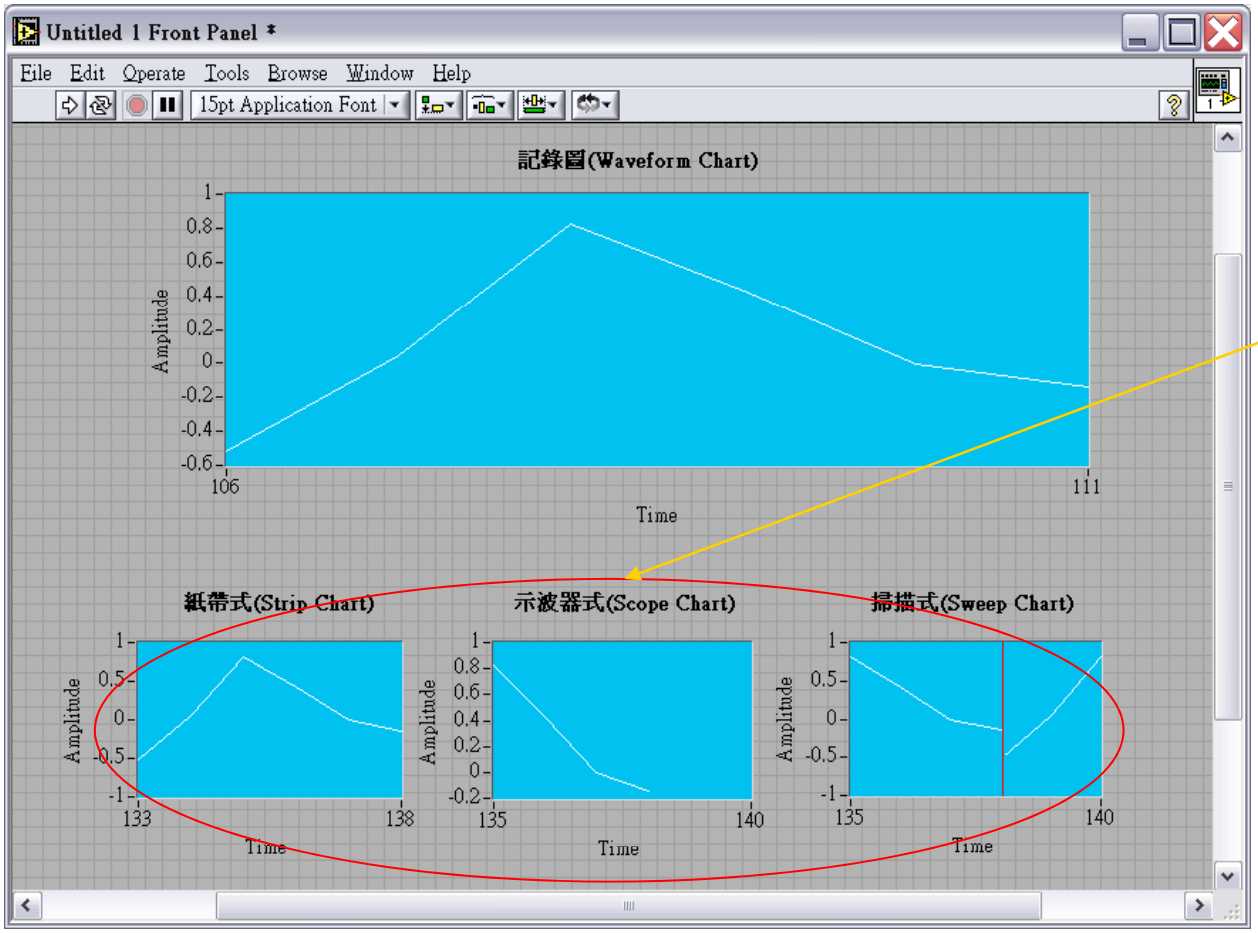




繪製資料（Plotting Data）

- 記錄圖（Waveform Chart）
- 數據圖（Waveform Graph）
  - XY Graph
- 強度圖（Intensity Graph）

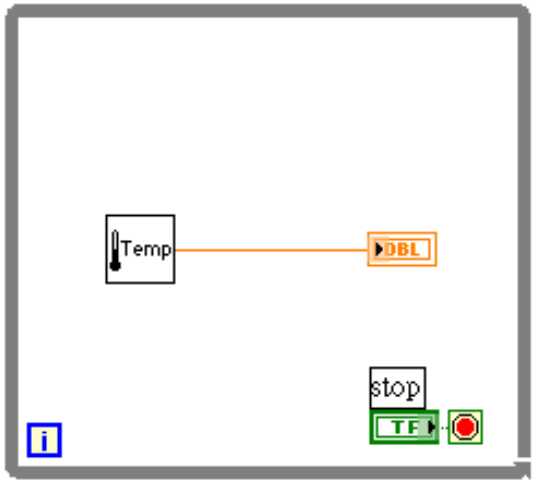
記錄圖（Waveform Chart）



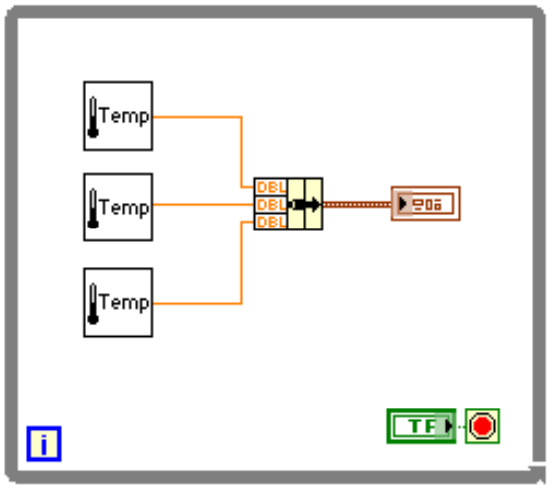
記錄圖的三種顯示模式

記錄圖（Waveform Chart）

- 繪製單筆資料



- 繪製多筆資料



記錄圖（Waveform Chart）

■改變記錄圖的屬性（Properties）

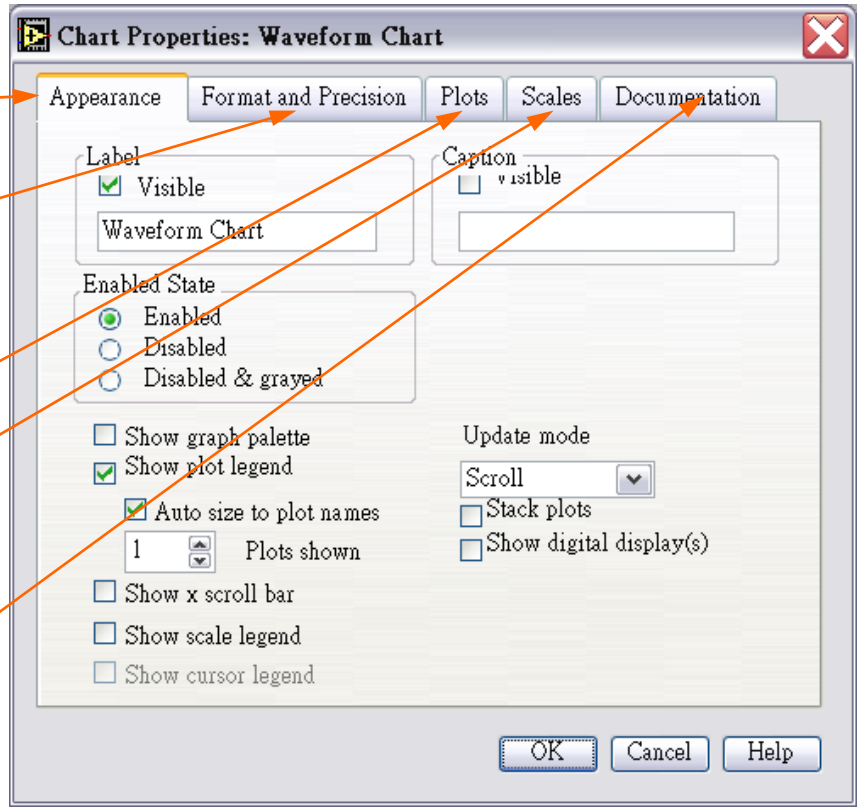
改變外觀

設定格式與軸線的精度

選擇資料線（ plot ）的型態

編輯刻度

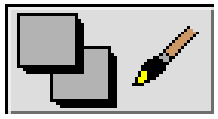
編輯說明文件



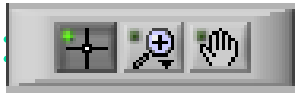
記錄圖（Waveform Chart）

- 記錄圖的進一步控制
  - 數值顯示（Digital Display）：

只要在記錄圖上點選彈出式選單的第一項：顯示項目（Visible Items），選擇數值顯示（Digital Display）即可。

然後可以利用文字工具  或調色盤  來改變字型與顏色。

  - 圖形調整板（Graph Palette）：

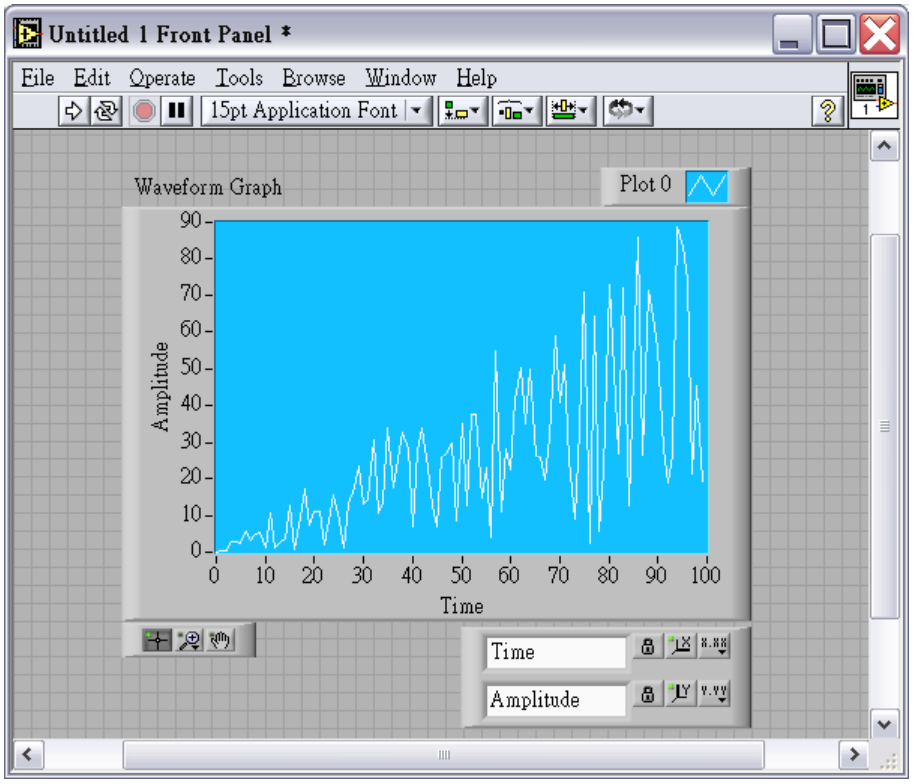


由左至右的三個按鍵分別為：固定鍵、放大/縮小鍵、及移動鍵。

放大/縮小鍵又可彈出一個附屬調整盤  ，上面六個鍵分別具備局部放大、回復、整體放大、縮小等功能。

數據圖（Waveform Graph）

- 繪製一組或多組陣列資料的2D顯示，每一組陣列就叫做一個Plot
- 數據圖與XY Graph的不同：
  - 數據圖示一組資料對應自己的索引
  - XY Graph則是一組資料對應另一組資料

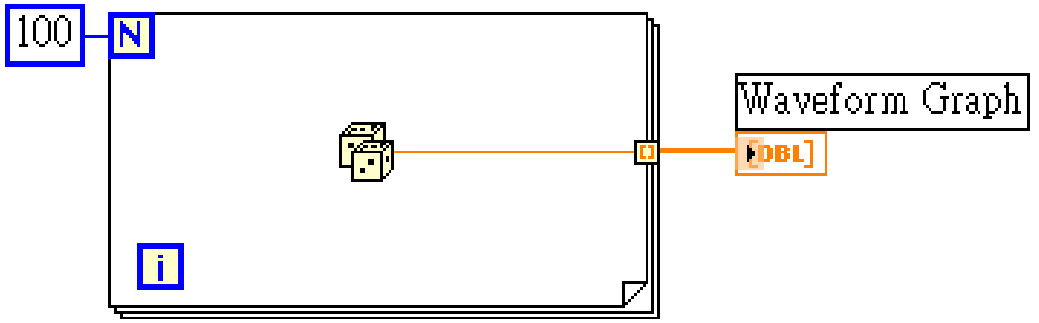


Single-Plot 數據圖

- 使用內建預設值

Initial X=0

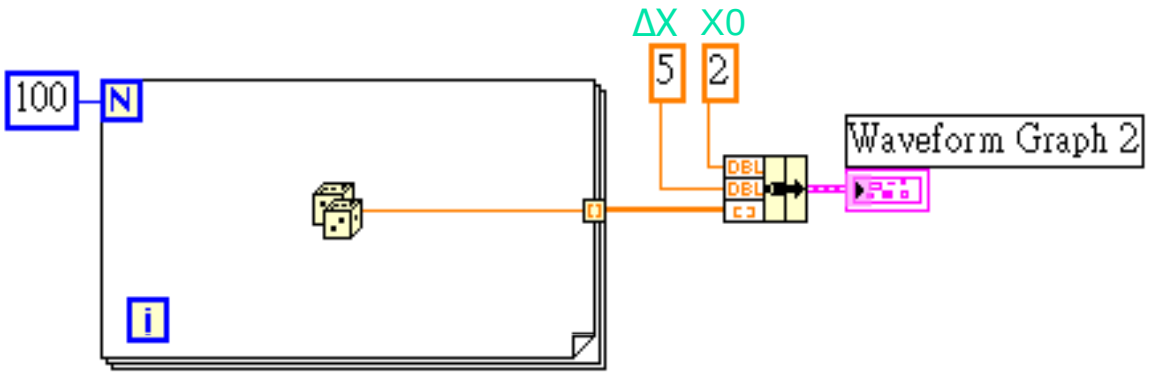
Delta X=1



- 使用者自訂值

Initial X=2

Delta X=5

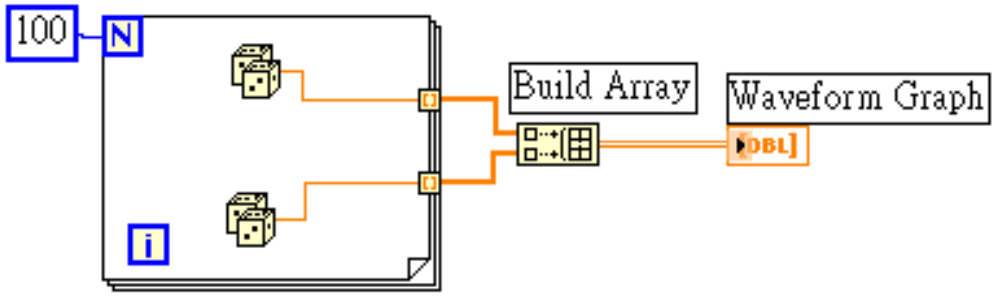


Multiple-Plot 數據圖

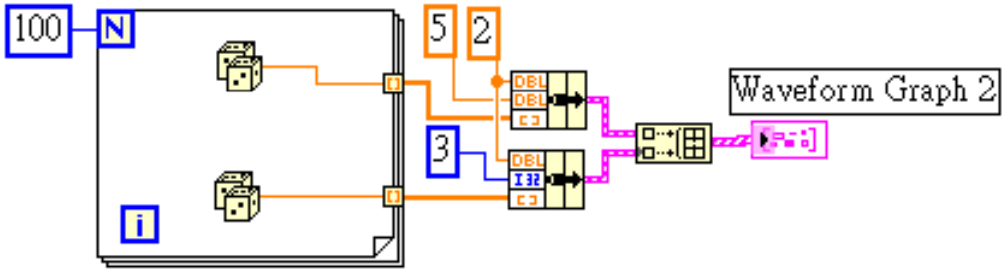
- 每一列代表單獨的Plot：

Initial X=0

Delta X=1

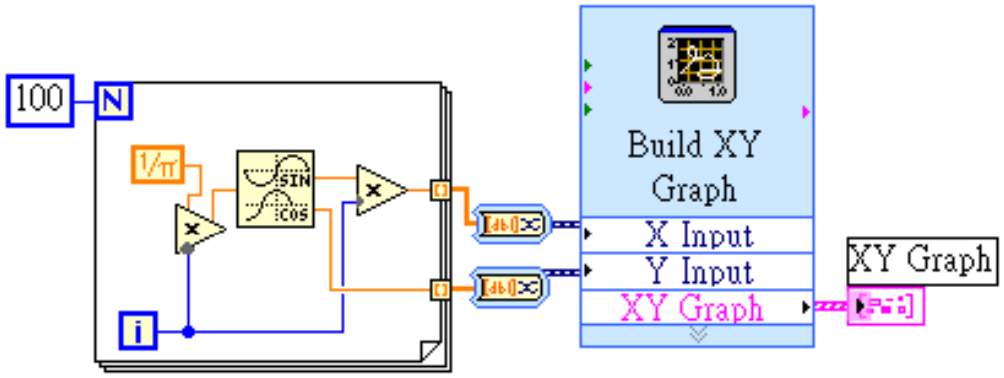


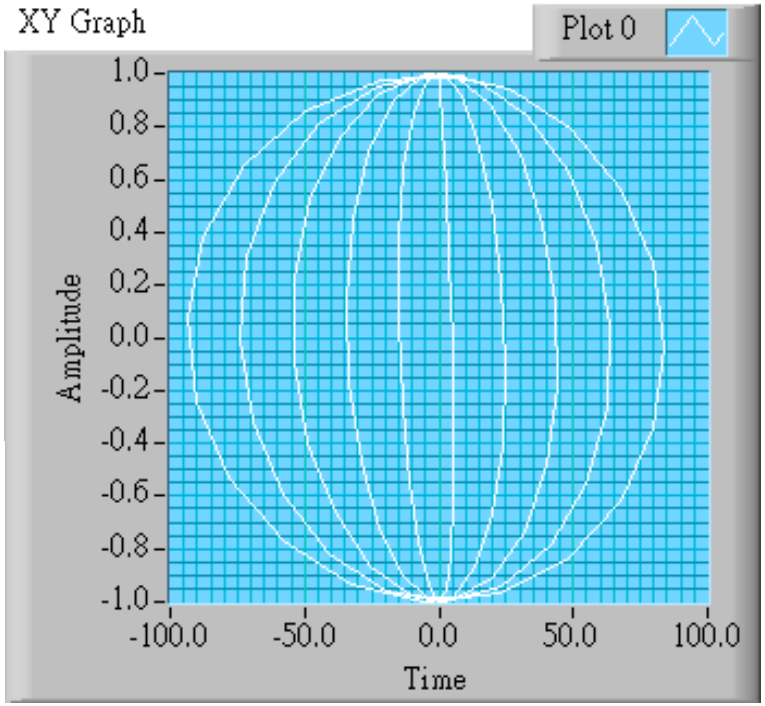
- 每一列代表單獨的Plot：
  但每一個Plot的X軸的間隔由
  使用者自行定義



XY Graph

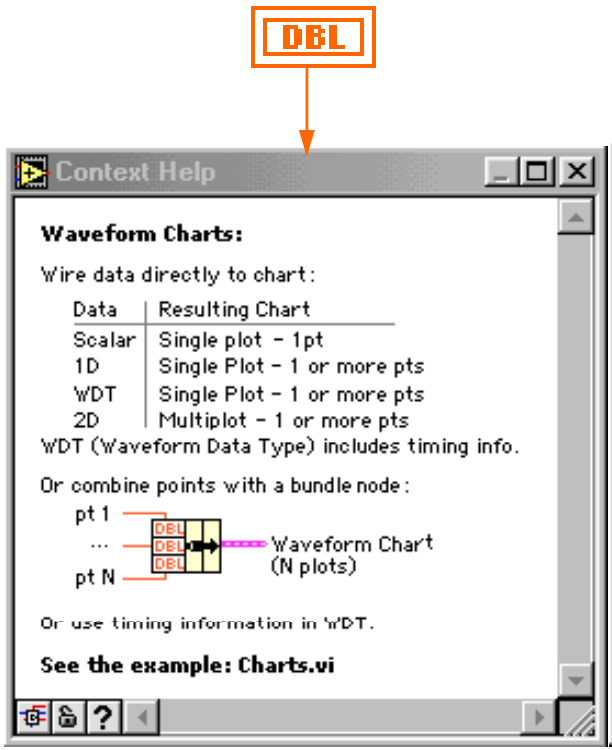
- 沒有一致的 X axis
- 由獨立的X 和 Y 陣列來定義資料點



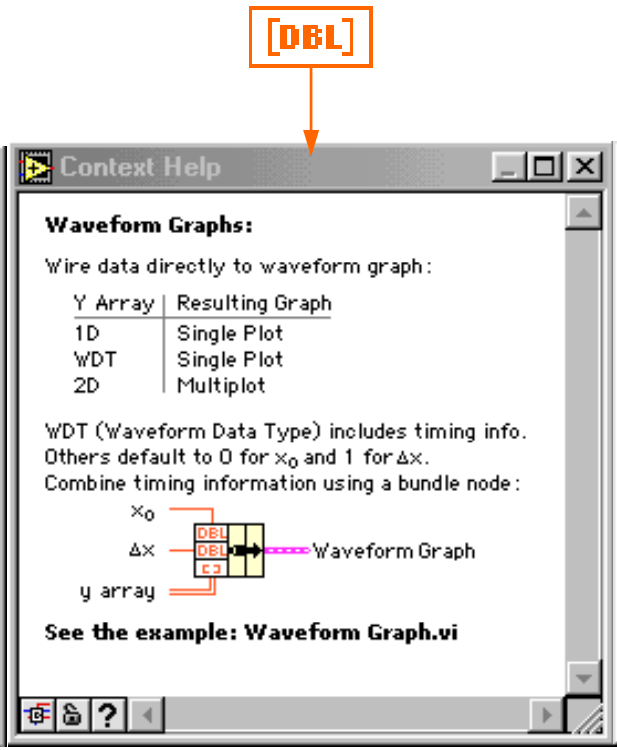


記錄圖與數據圖（結論）

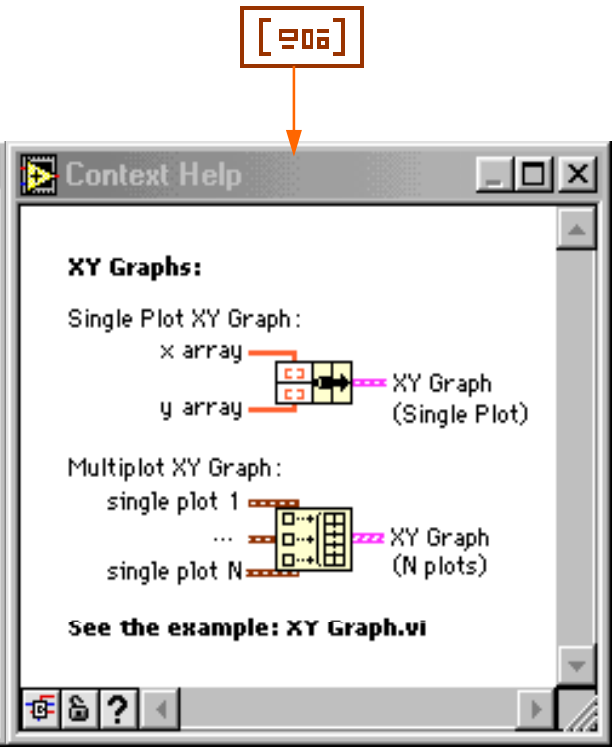
記錄圖

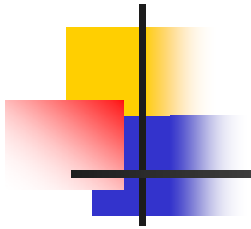


數據圖



XY Graph



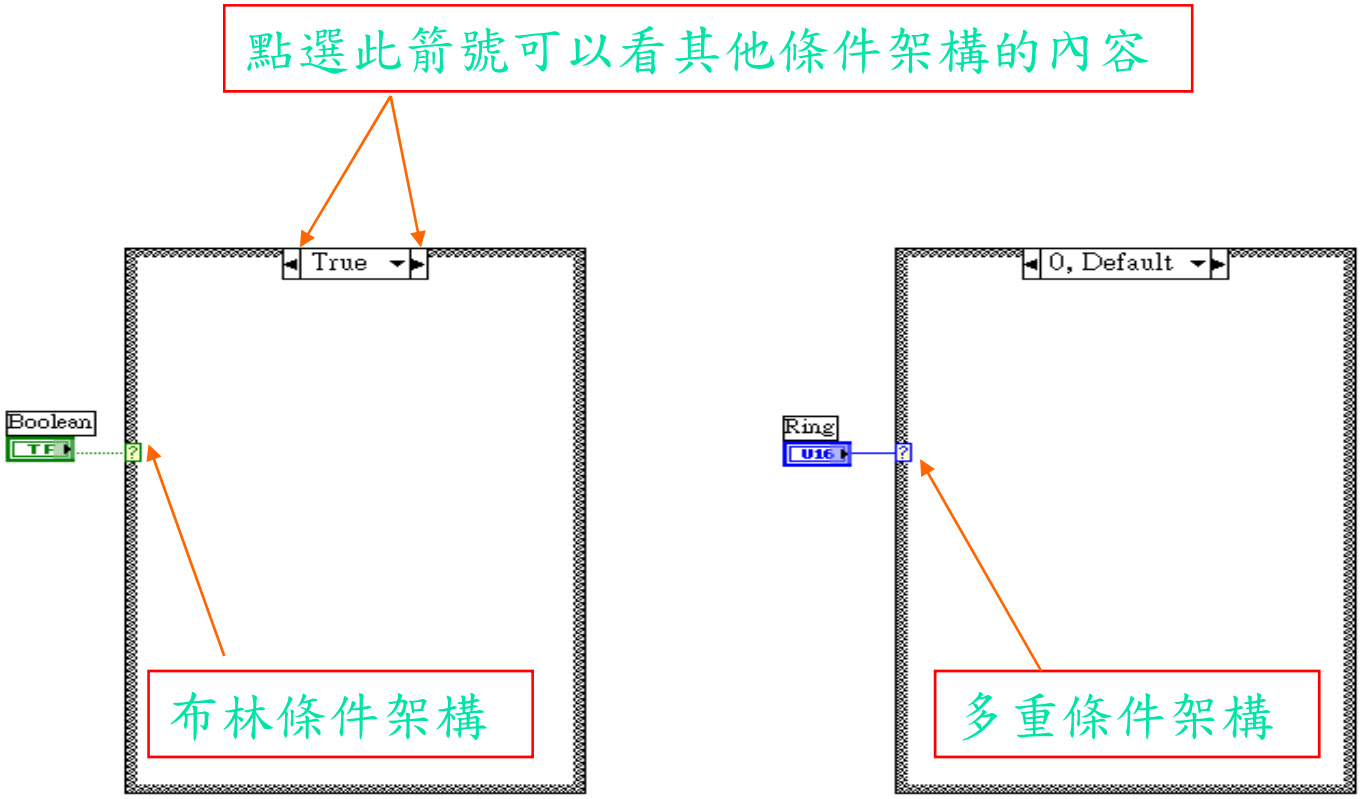


程式架構物件

- 條件架構（ Case Structure ）
- 程序架構（ Sequence Structure ）
- 程式結點（ Formula Node ）

條件架構（Case Structure）

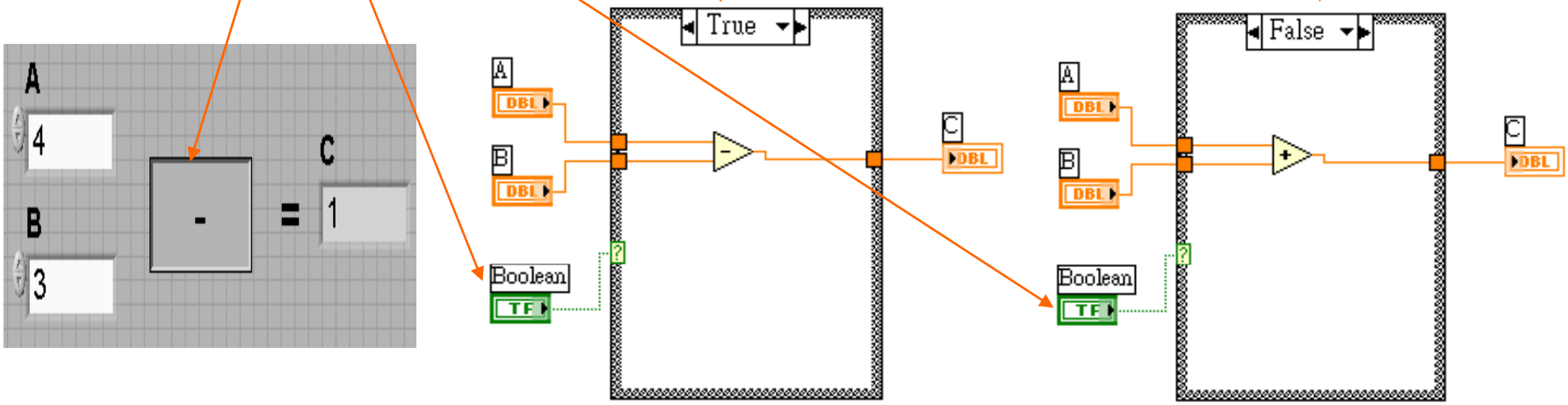
- 條件架構（Case Structure）是一有條件的執行架構



條件架構（Case Structure）

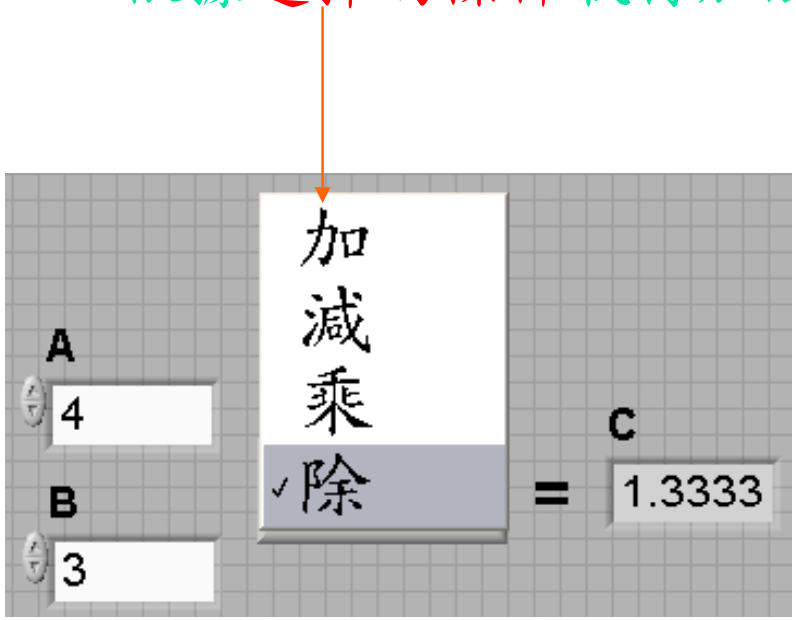
- 布林條件架構範例如下：

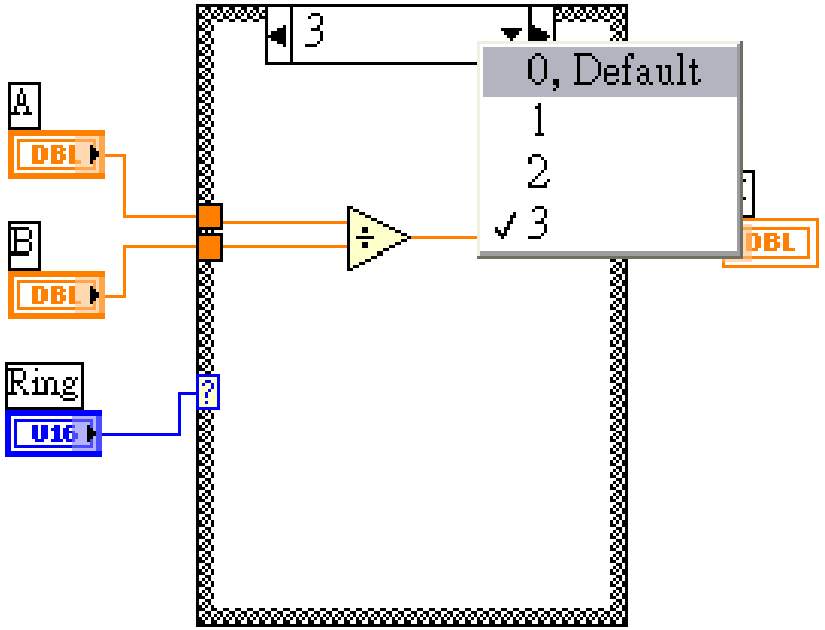
  當Boolean條件為《True》時執行減法，條件為《False》時執行加法



條件架構（Case Structure）

- 多重條件架構範例如下：
  - 依據選擇的條件執行加法、減法、乘法、除法等

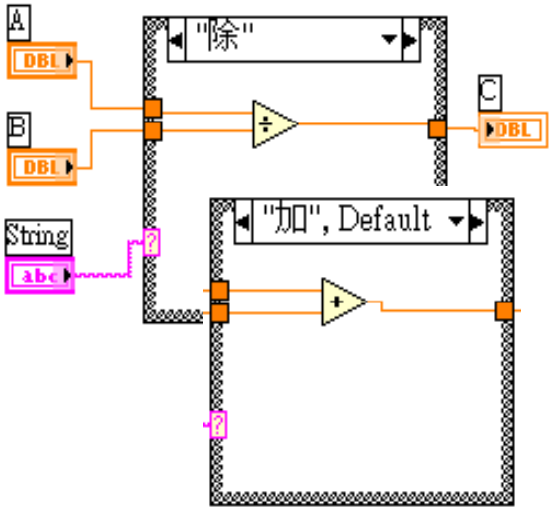




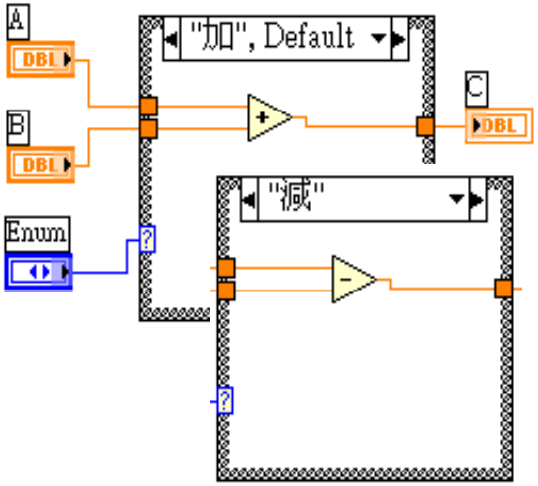
條件架構（Case Structure）

- 其他可以當做判斷條件的物件

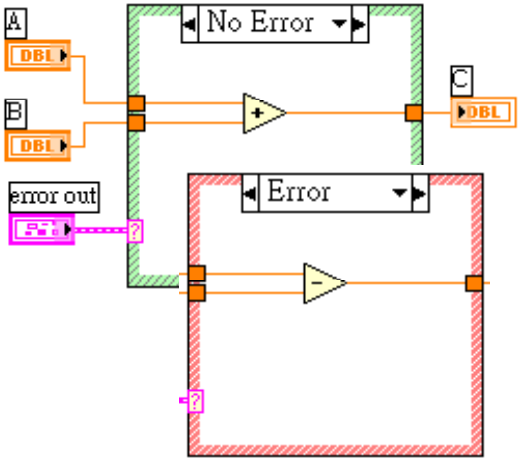
字串條件架構



Enum條件架構



Error條件架構



條件架構（Case Structure）

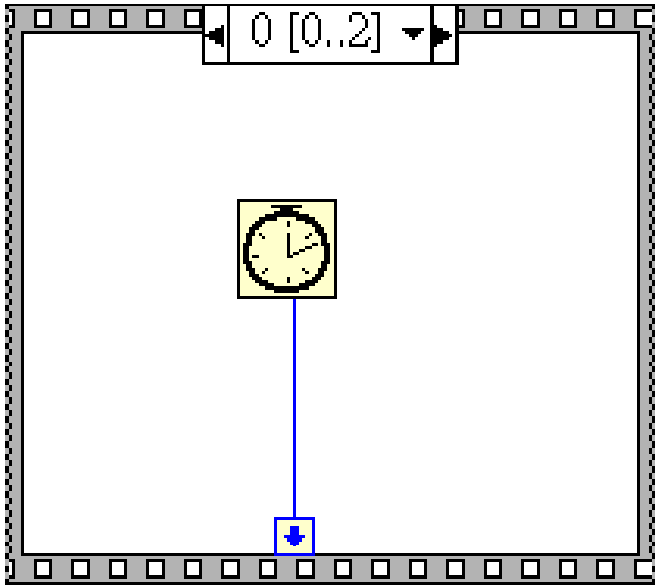
- 條件架構（Case Structure）的輸入端點不一定在每一層架構中都需給予定義或數值。
- 條件架構（Case Structure）的輸出端點在每一層架構中都需給予定義或數值。

程序架構（Sequence Structure）

- 程序架構（Sequence Structure）主要的功能是控制 VI 圖示區內執行的先後順序，舉例如下：
  - 現在設計一 VI，我們先給定一個值，然後產生亂數，直到亂數和我們設定的值相同才停止，計算其所需的時間。

Step 1：

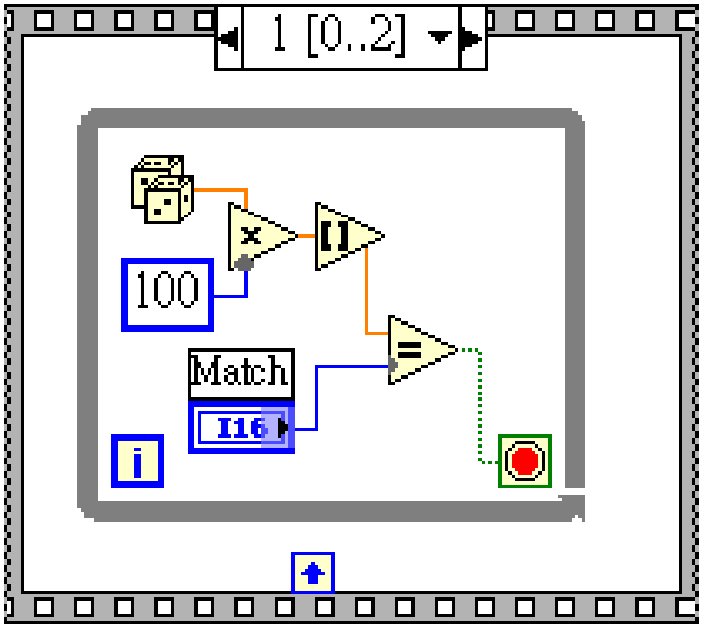
首先取得第一個時間的值，並且傳到後面。



程序架構（Sequence Structure）

Step 2：

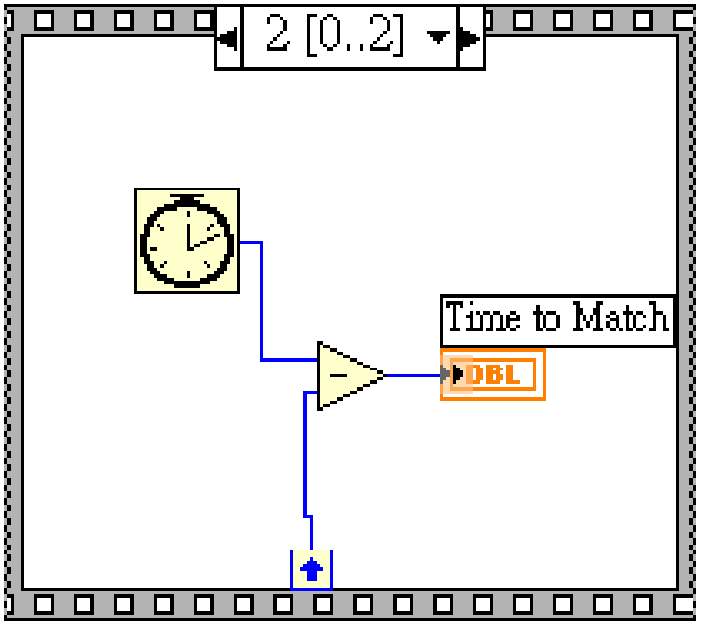
利用亂束產生器產生一個亂數值並利用進位函數取得一個整數值，再與我們設定的值相比較，直到相同為止。



程序架構（Sequence Structure）

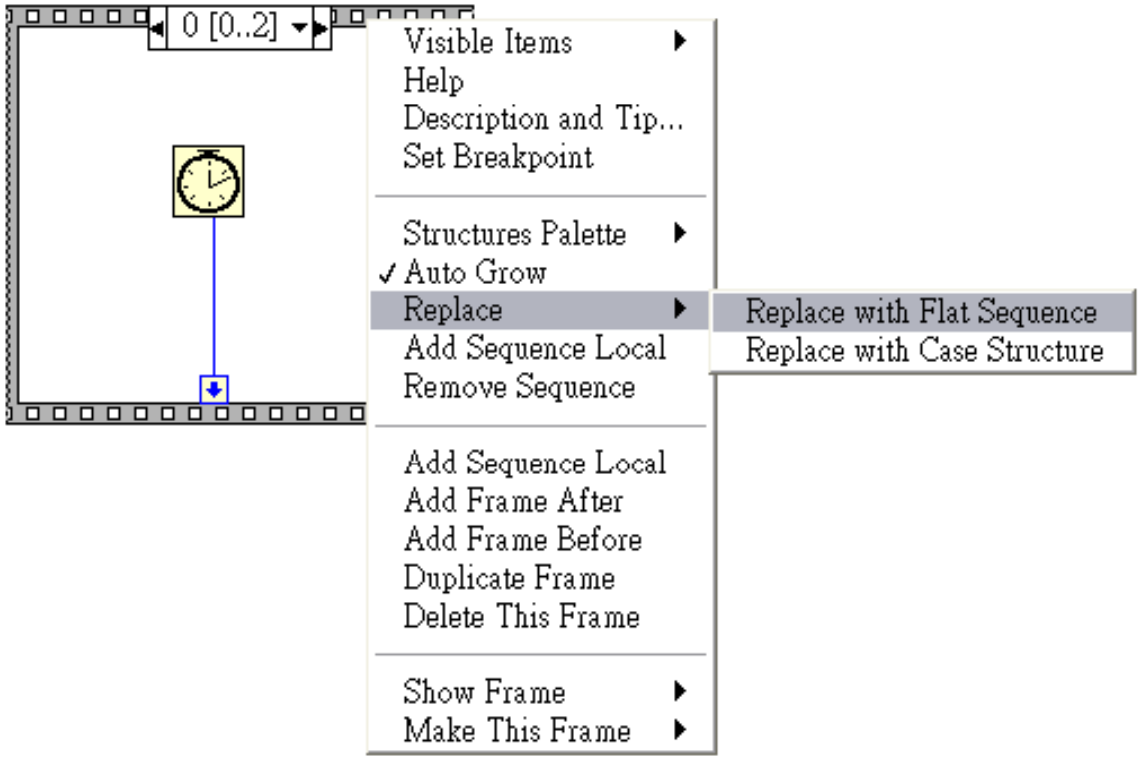
Step 3：

最後在取得當下的時間並與第一個Frame傳過來的時間值相減即可得《Time to Match》的時間。



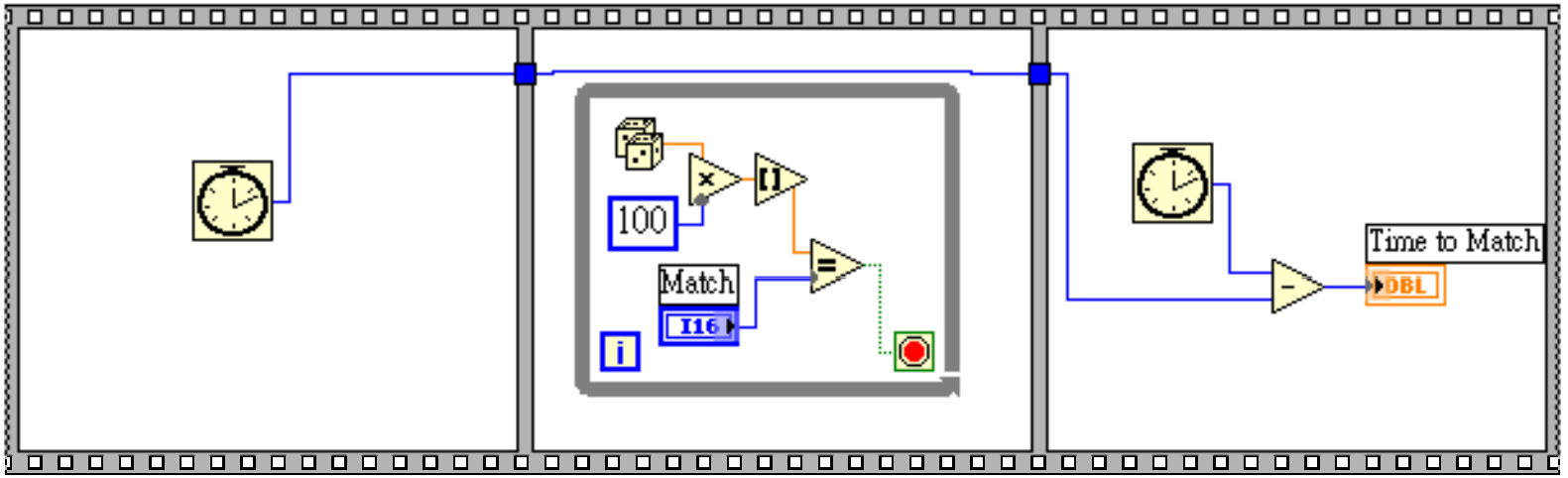
程序架構（Sequence Structure）

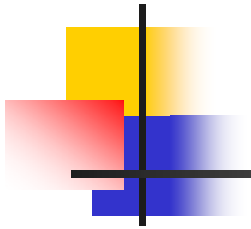
LabVIEW 7 以後的版本也提供另一種平面式的程序架構，可已經由下圖所示做轉換



程序架構（Sequence Structure）

這樣的程序架構可讓我們一目了然看到所有在架構中的程式





數列與數列集

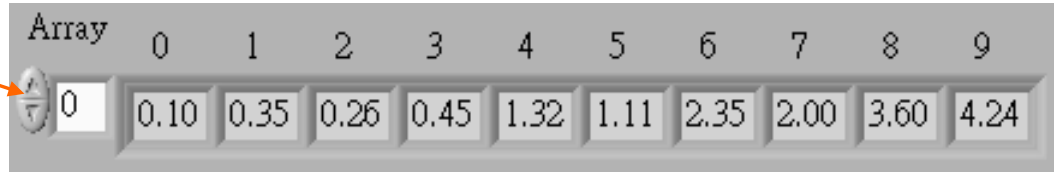
- 數列 （ Array ）
- 數列集 （ Cluster ）
- 數列運算
- 數列函數與運用
- 表格的使用

數列（Array）

- 數列（*Array*）是相同形式數據資料的集合 。
- 數列（*Array*）可為一維或多維，每維最多不超過 231-1個元素。
- 在數列中，透過索引可以找到元素所在的位置 。
- 索引值由 0 開始計算 。

1D Array

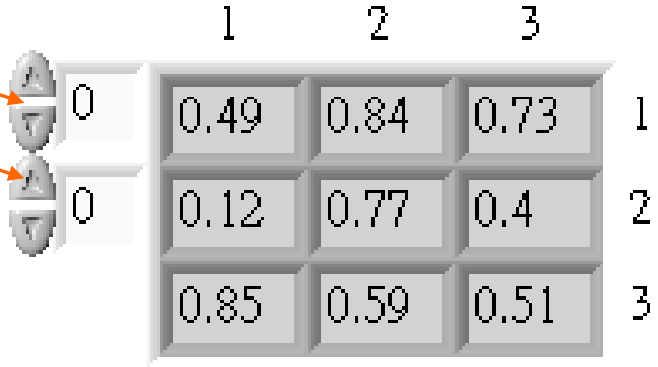
索引值



2D Array

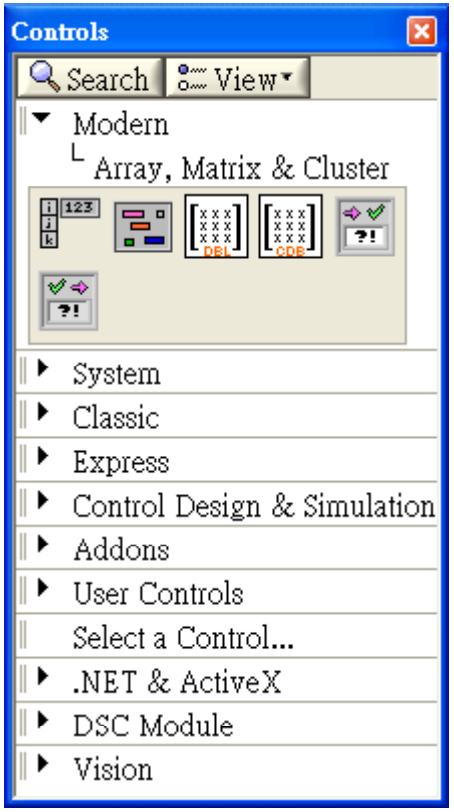
行索引值

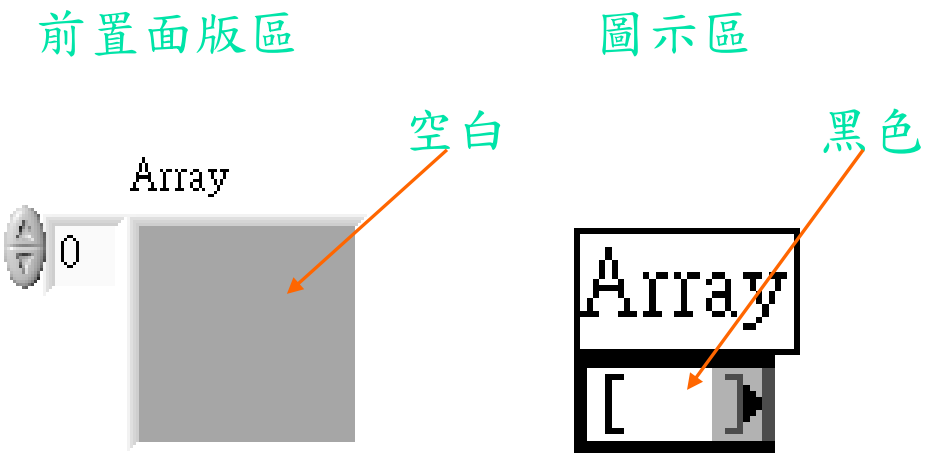
欄索引值



數列控制與索引建立

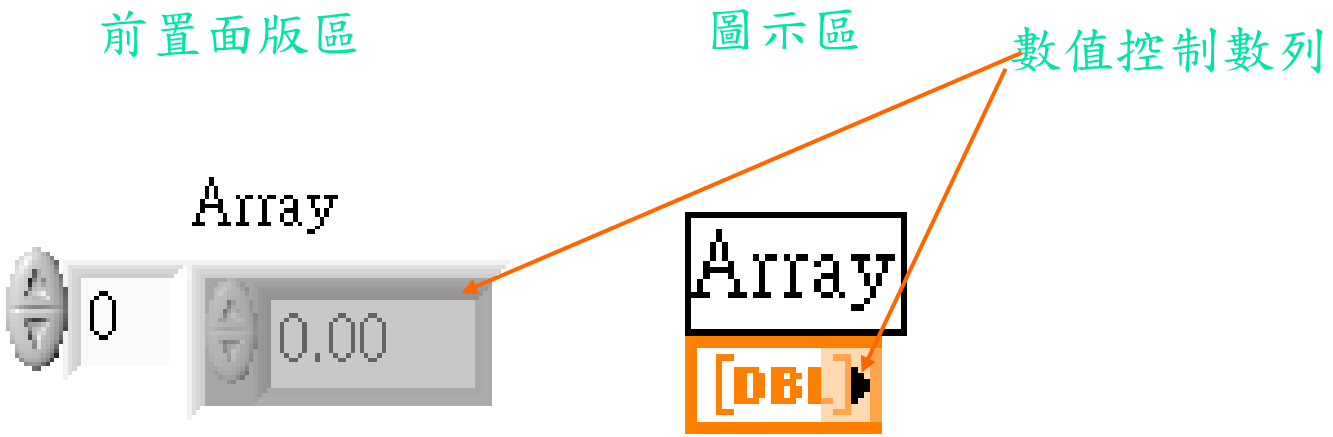
- 如右圖所示在控制面盤（Controls Palette）中選取數列與數列集（Array, Matrix & Cluster）Sub-Palette中的數列（Array）物件。





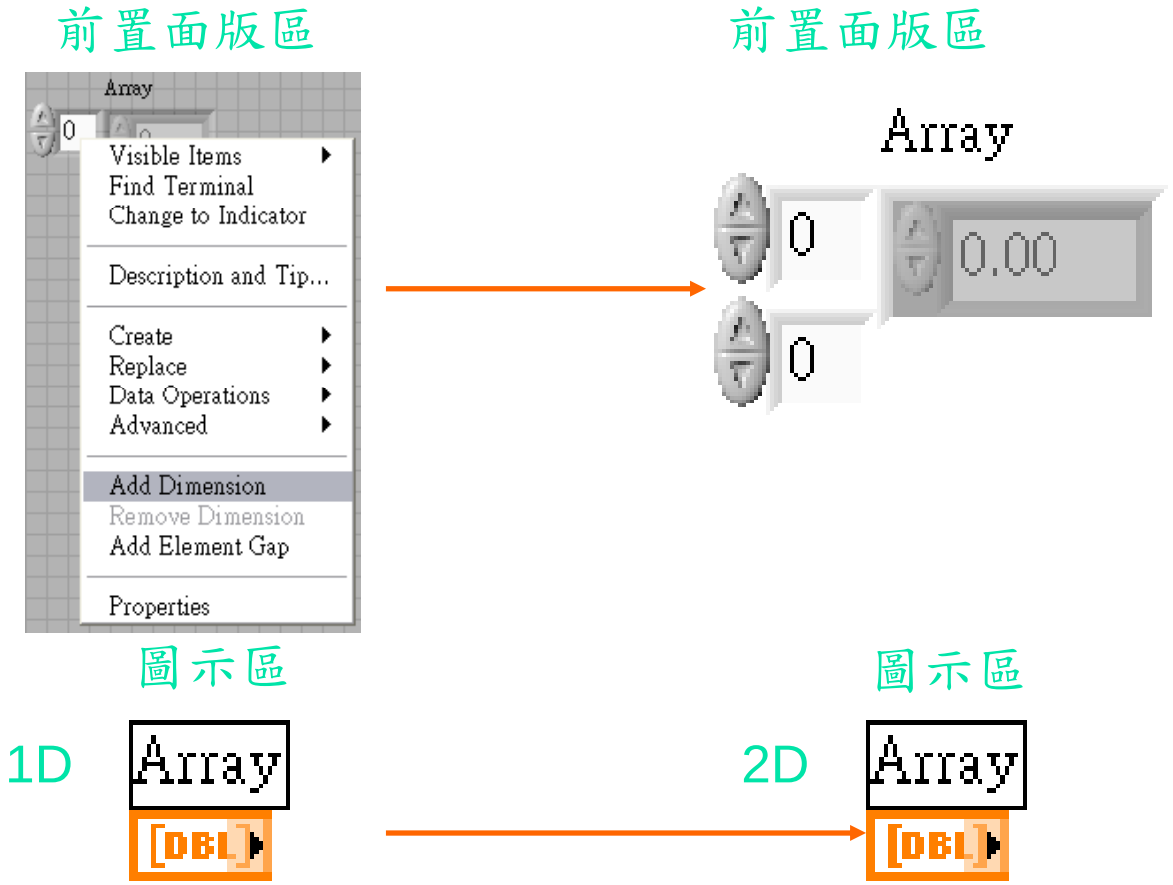
數列控制與索引建立

- 接下來再將想建立的物件（數值、布林、文字等）放入。



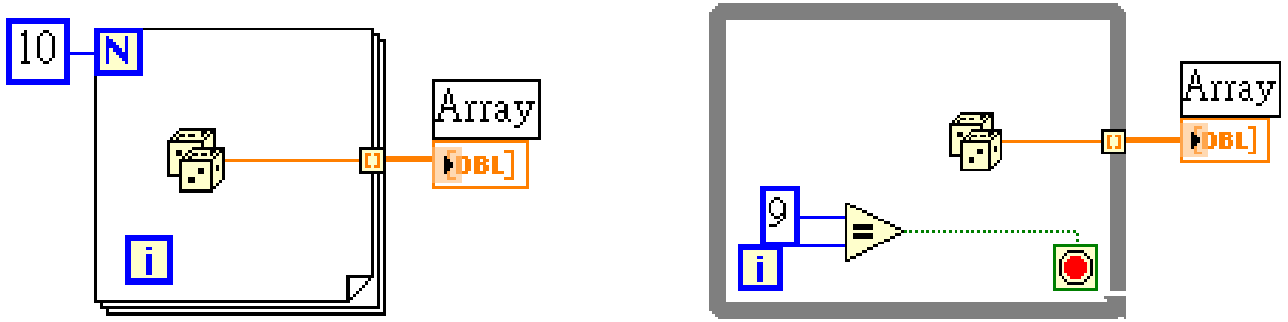
數列控制與索引建立

- 若要增加數列之索引維數 ，可如下圖所示：

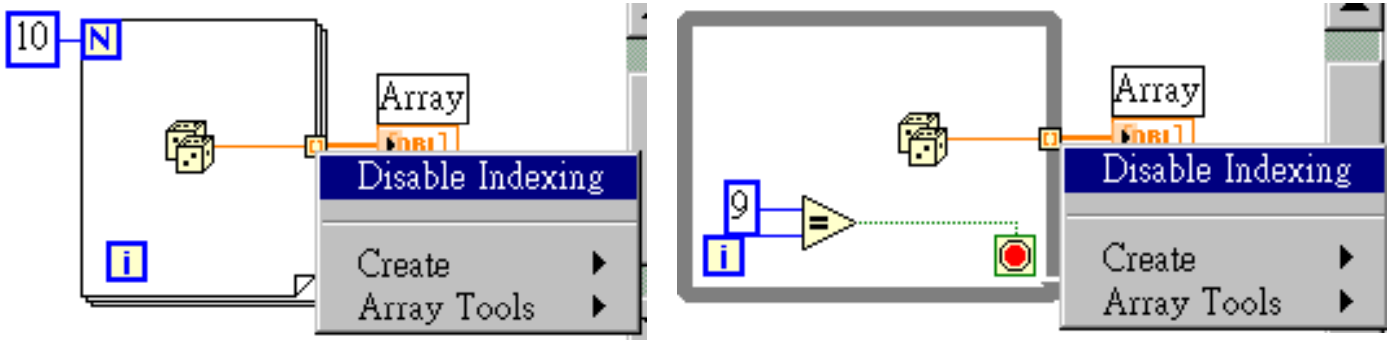


自動索引建立數列

- For Loop 與 While Loop 能在其迴圈範圍內自動建立數列，稱為**自動索引** (auto-indexing)

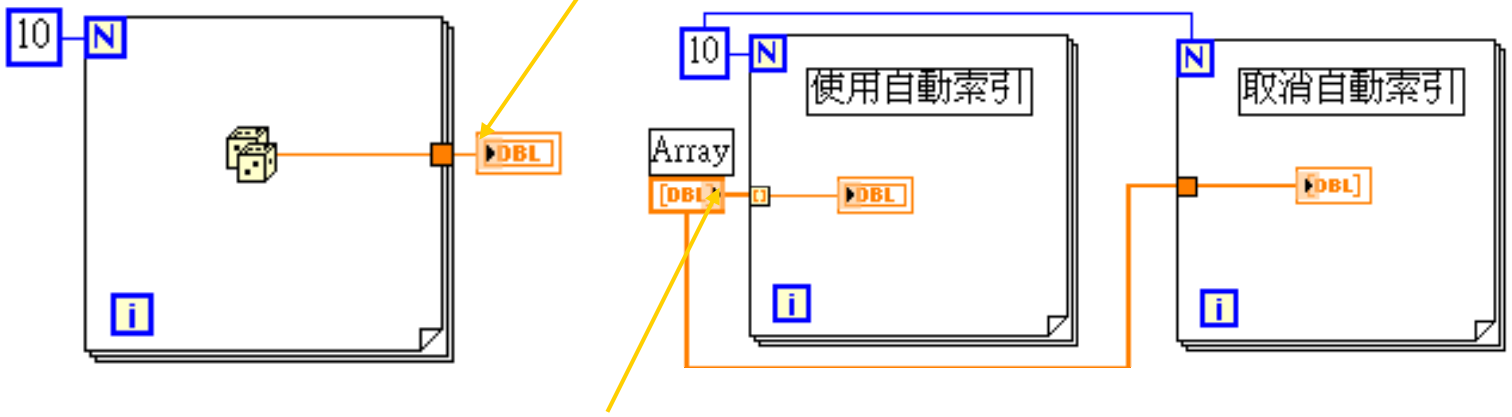


- 也可以經由選取 *Disable Indexing*，取消自動索引



自動索引建立數列

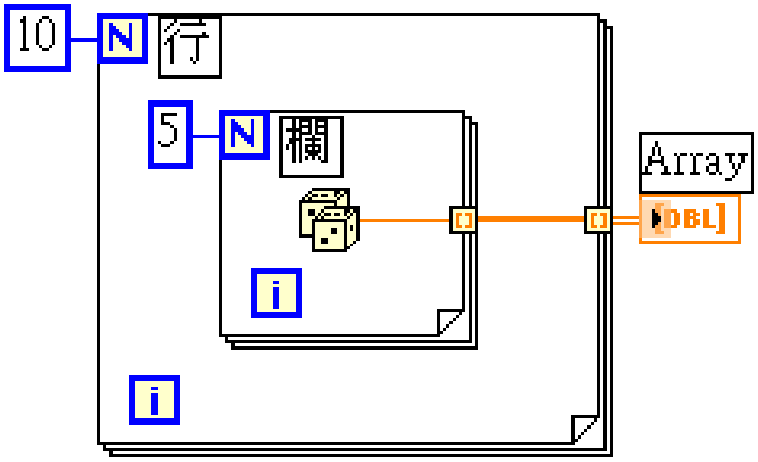
- 取消自動索引後只會傳出最後一個值

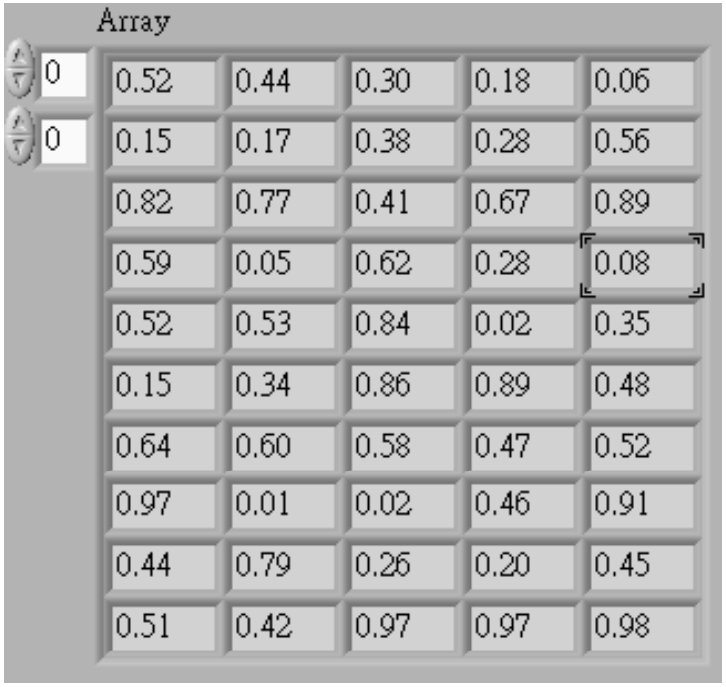


- 自動索引也會發生在由數列傳入迴圈時，當數列進入迴圈，**LabVIEW** 會將自動索引設成預設值
- 要注意的是這只在For Loop才會發生，While Loop不會將自動索引設為預設值

建立二維數列

- 一個二維數列需要兩個索引值定義元素，其中一個為欄指標(直)，一個為行指標(橫)，二者的起始值均為0。

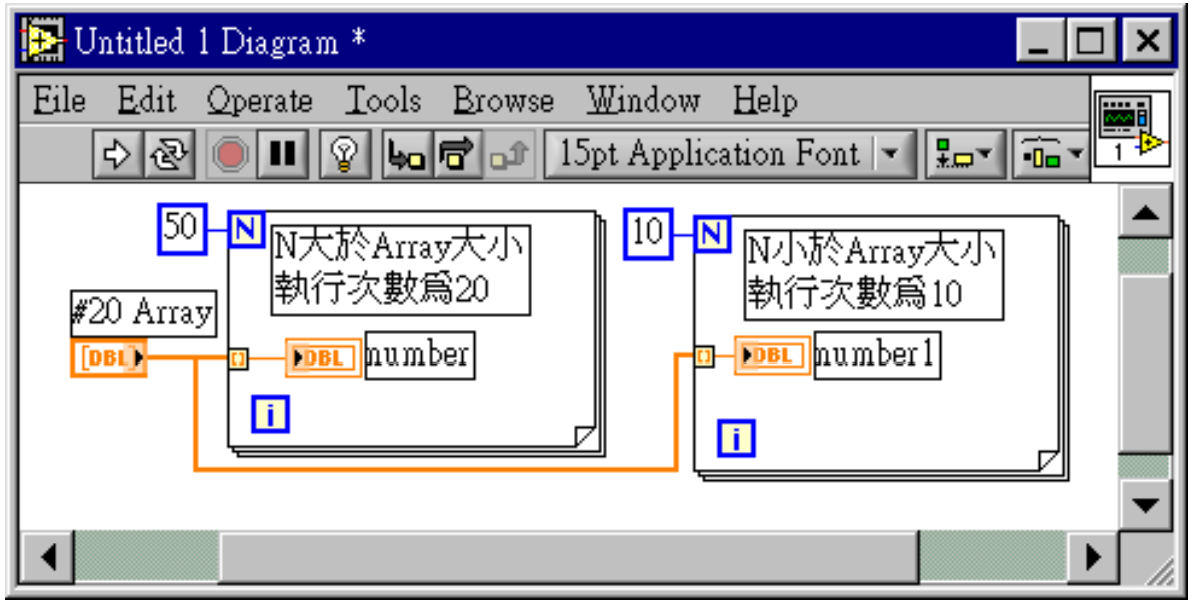




- 內圈產生欄元素
- 外圈產生行元素

自動索引決定 For Loop 迴圈執行次數

- 當一個數列進入一個 For Loop，使用自動索引時，**LabVIEW**會自動判斷並且以數列的大小與迴圈執行次數N去比較大小，迴圈執行次數以數目小的為準。



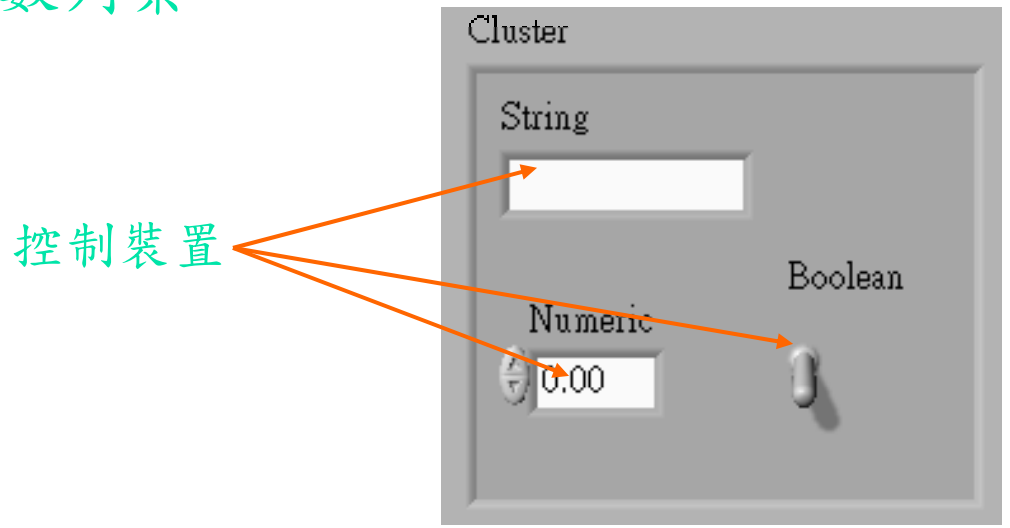
數列集（Cluster）

- 數列集（*Cluster*）與數列相似，只不過是另一種型態。數列集內可包含不同型式的資料，我們可以想像它就好比是許多電話纜線束集在一起。

- 數列集就像是一主幹，可有不同型態的輸入與輸出線。不過每條傳輸線的兩端資料型態必需是相同的。

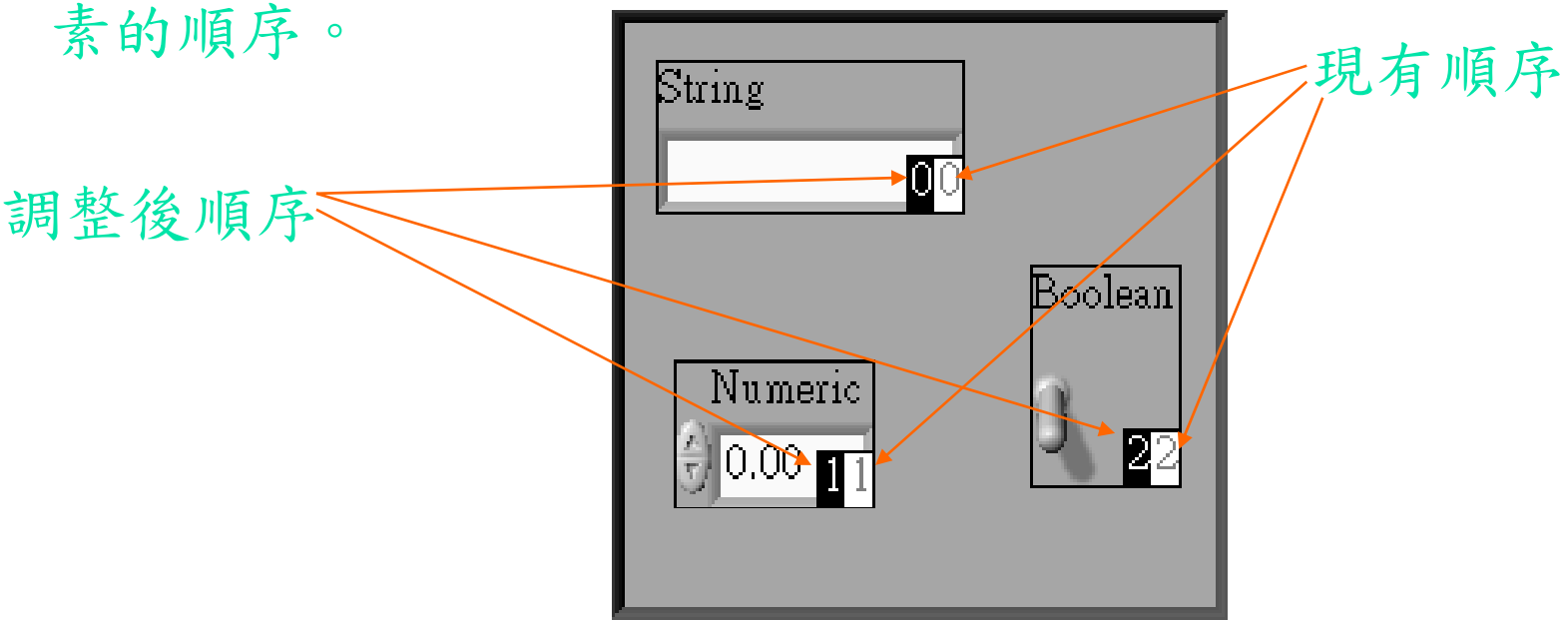
建立數列集

- 在前置面版中，選取 *Controls* 》 *Cluster* 中的 Cluster，會出現一數列集框。數列集框內可放任何的資料型態，數值、布林、字串列等。
- 有一點要特別注意的，在數列集框之中，必需全都是控制裝置或全都是指示裝置，不能將二者混合成為一個數列集。

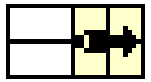
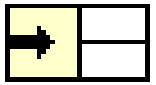


數列集順序

- 數列集內各元素的順序與位置無關，而與置入數列集之順序有關。
- 必要時可以在數列框上拉出一隨機選單，選取**調整順序**（*Reorder controls in Cluster*），來調整元素的順序。

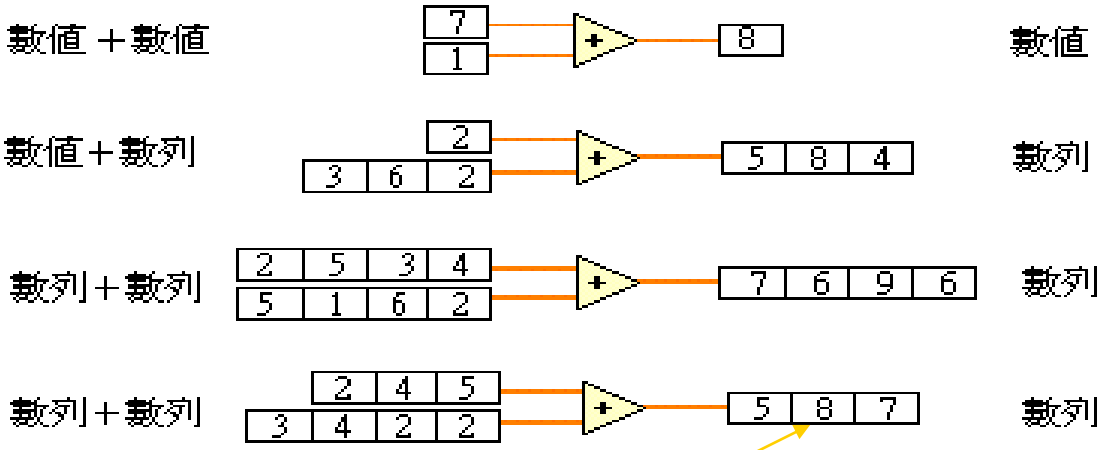


集束與解散數列集

- 在圖示區中若需建立數列集可利用**集束函數**  (*Functions》Cluster》Bundle*)，各元素可由左端輸入集束成一新的數列集。
- 解散數列集，可以利用**解散函數**  (*Functions》Cluster》Unbundle*)。它可將一個數列集拆成個別的元素，各元素的順序是由上而下。輸出端個數必需與輸入端元素個數相同，傳輸出的元素型式則以其原有的型式為準。

數列運算

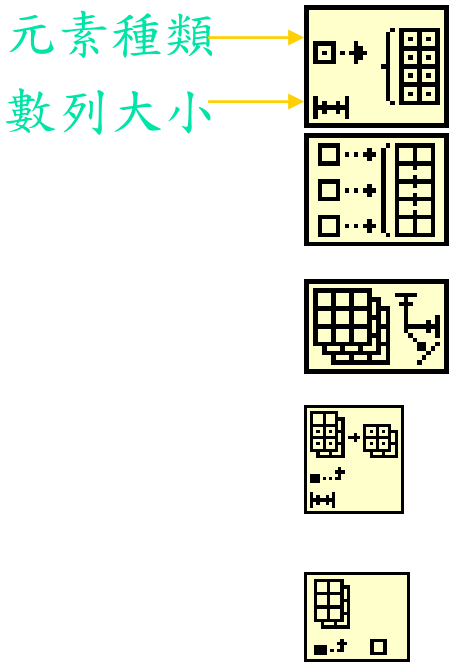
- 在 **LabVIEW** 中，加法、乘法、除法等函數，都可應用於數值與數列的運算，以加法為例，圖 7-16 表示純量加純量、純量加數列、數列加數列等不同的結果，請特別注意比較傳輸線的粗細。



- 當兩個大小不同的數列相加，其和數列之大小以較小的為準，較長數列的元素便被忽略，如上圖的第四項。

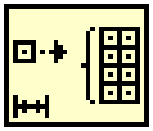
數列函數與運用

- 初始數列 （*Initialize Array*）
- 建立數列 （*Build Array*）
- 數列大小 （*Array Size*）
- 數列子集 （*Array Subset*）
- 數列索引 （*Index Array*）

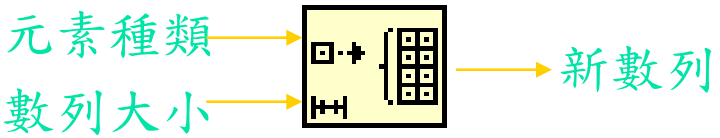


數列函數與運用

- 初始數列 （*Initialize Array*）

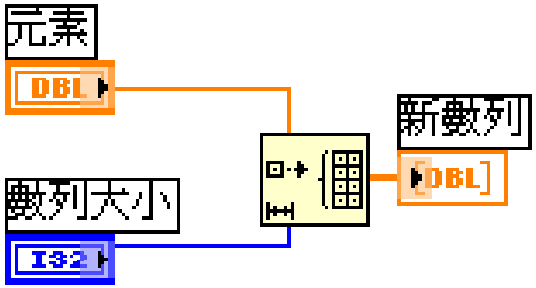


– 此函數是建立一新的數列，且設定此數列內容的起始值



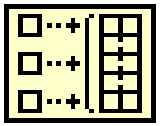
前置面版區

圖示區

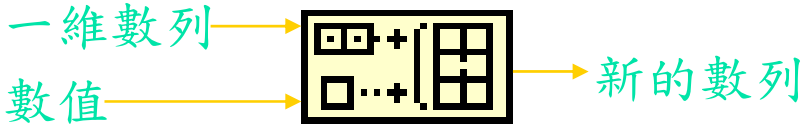


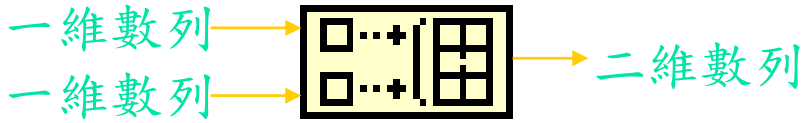
數列函數與運用

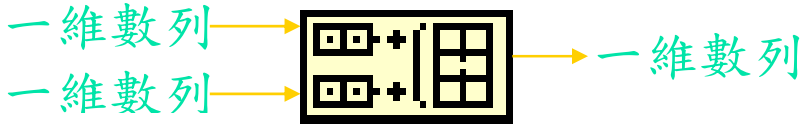
- 建立數列 （*Build Array*）



  - 此函數可以結合
    - a) 兩個以上的元素成為一個一維數列
    - b) 兩個以上的一維數列組合成一維或二維數列

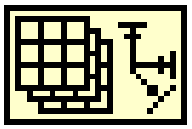






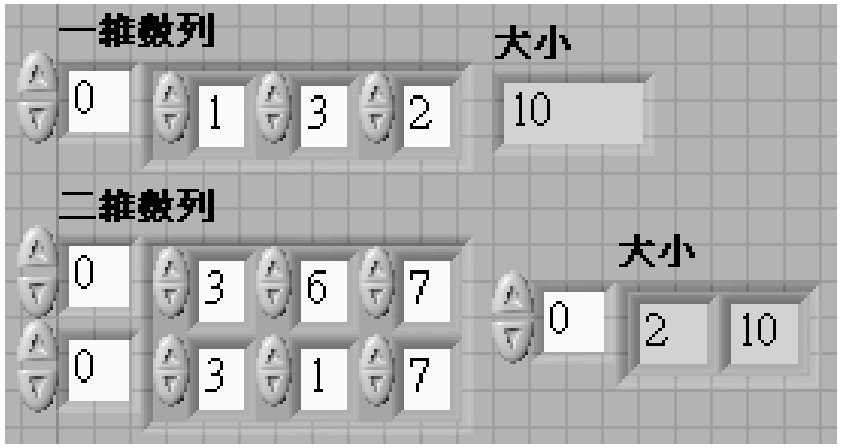
數列函數與運用

- 數列大小 （*Array Size*）

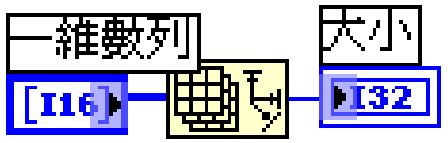


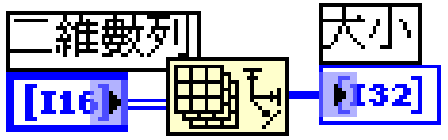
  - 此函數功能是決定輸入數列的大小
    a) 若輸入為一維數列時輸出為一個數值
    b) 若輸入為二維數列時輸出為一個數列

前置面版區



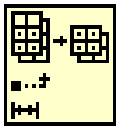
圖示區





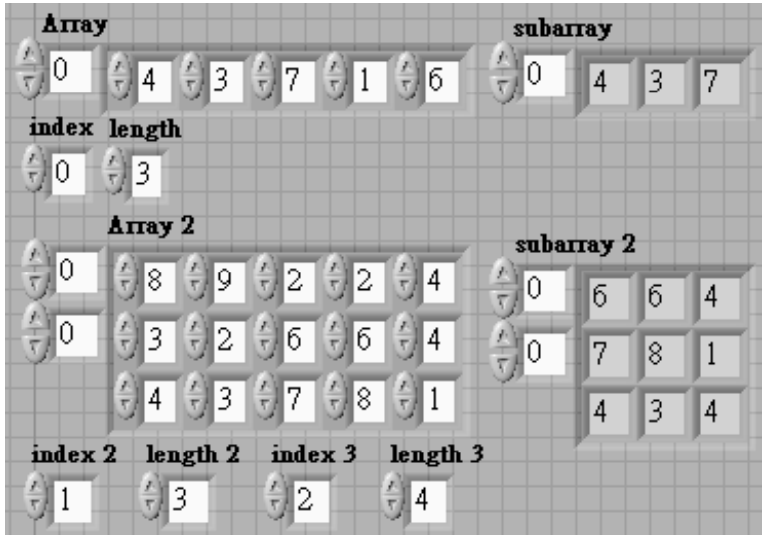
數列函數與運用

- 數列子集 （*Array Subset*）

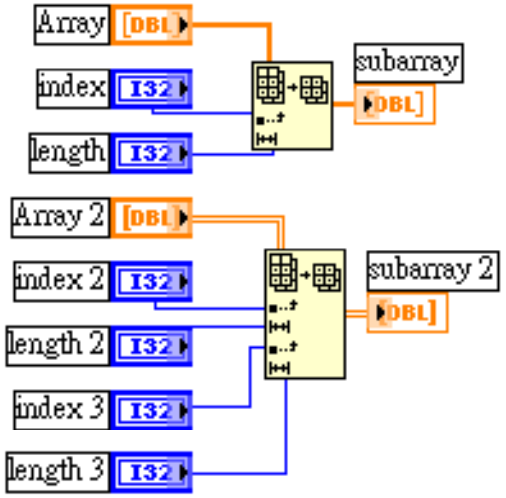


  - 此函數可以設定索引及長度，在一數列中取出一子數列
    a) 若輸入為一維數列時可取出一個一維數列
    b) 若輸入為二維數列時可取出一個二維數列

前置面版區

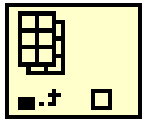


圖示區



數列函數與運用

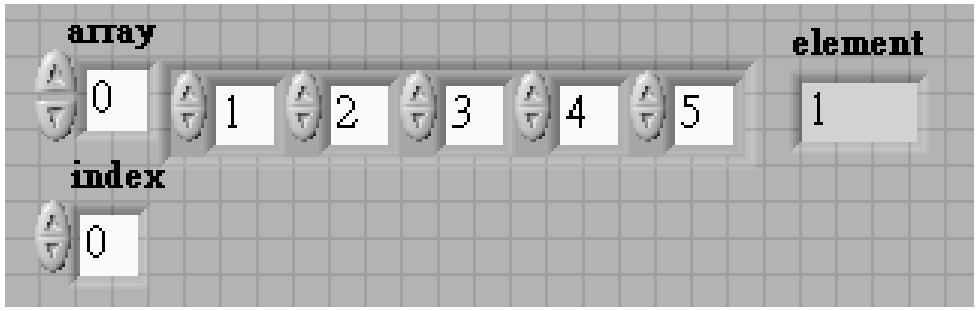
- 數列索引 （Index *Array*）



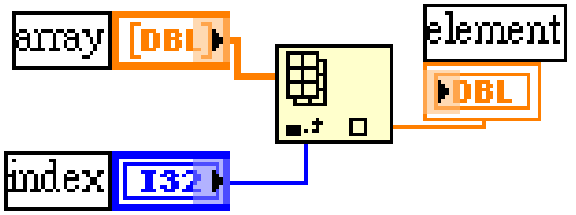
  - 此函數是依照輸入的數列及索引，取出該數列索引值內的資料

    a) 下例為取一維數列的第一個元素

前置面版區



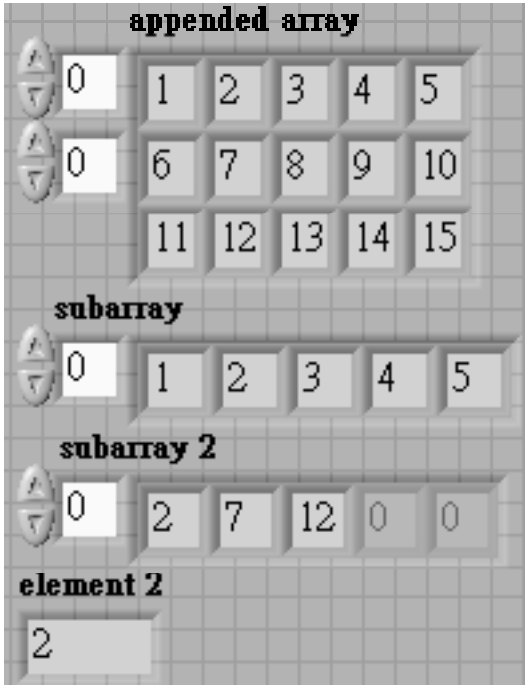
圖示區



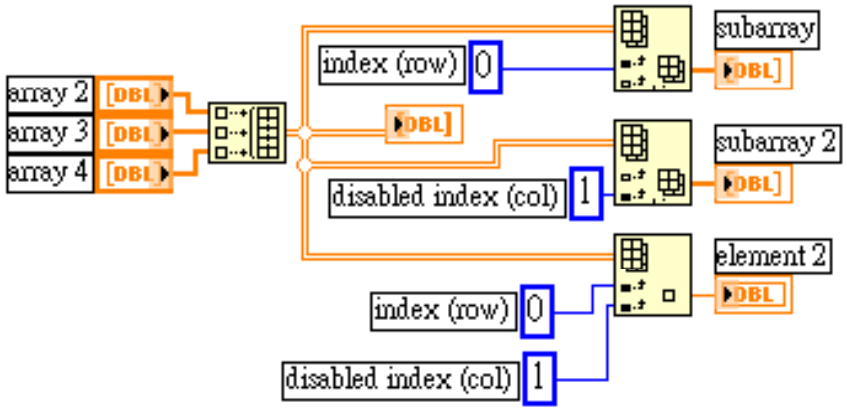
數列函數與運用

b) 下例為分別取二維數列的第一行數列、第二欄數列與第一行的第二個元素

前置面版區

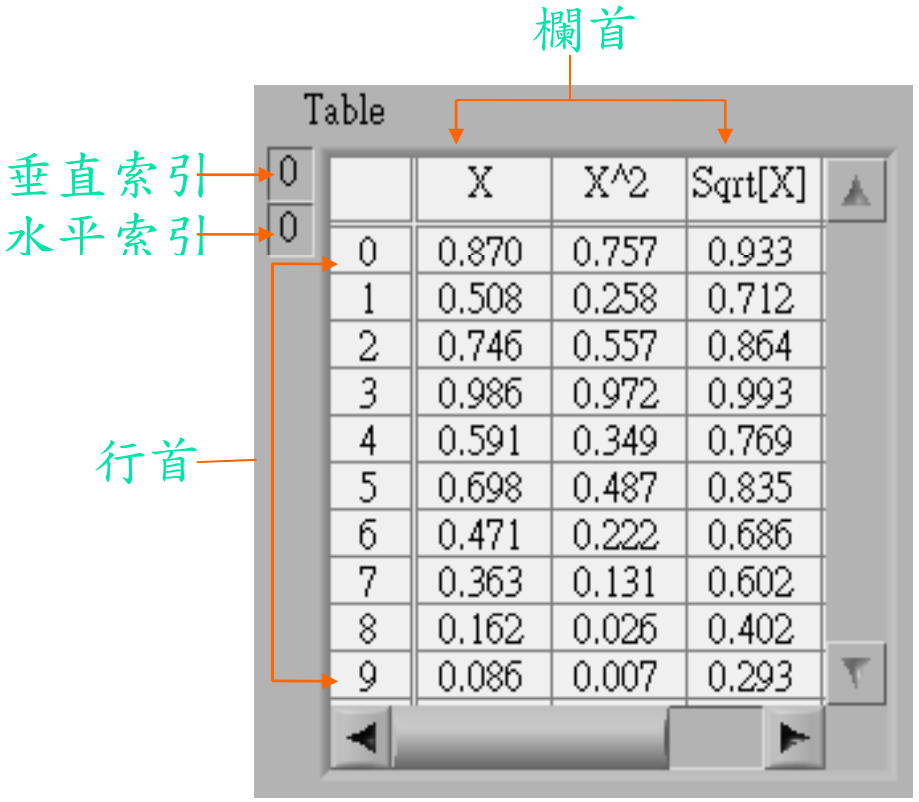


圖示區



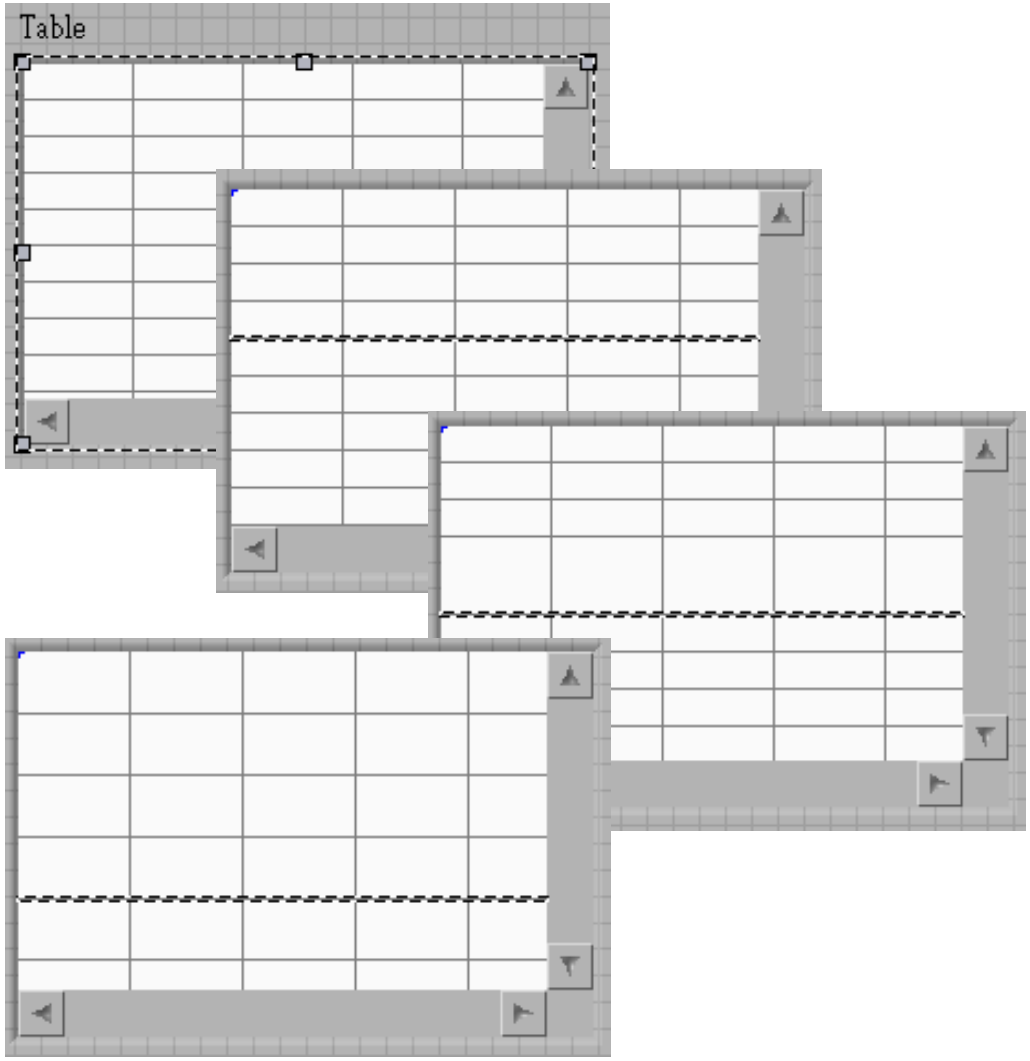
表格的使用

- 表格的特點
  - 表格可視為一個二維數列，每一個方格中可輸入數值或文字
  - 表格中的欄首與行首並非表格資料的一部份，它們是另一分開的資料，可藉由屬性節點來讀取與設定。



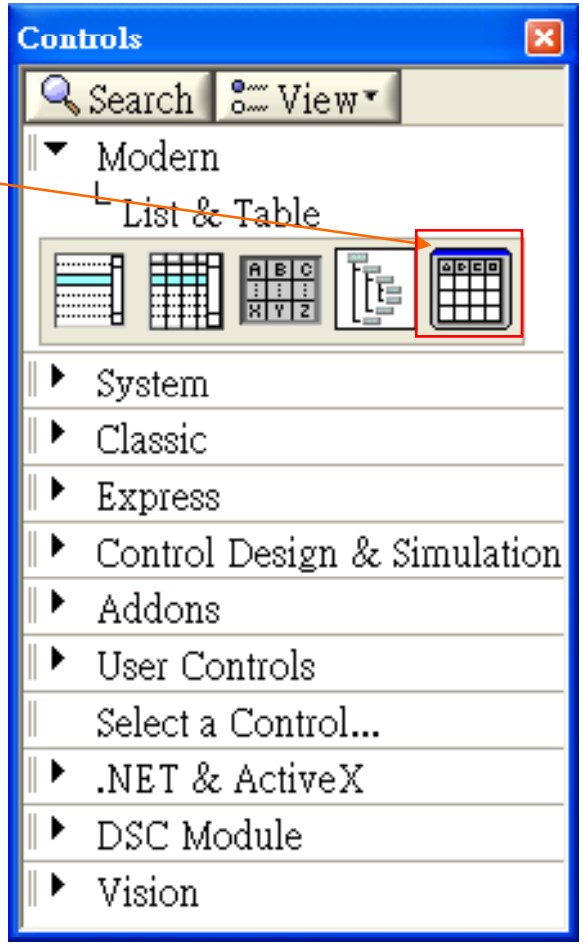
表格的使用

- 改變表格的大小
  - 利用定位工具可以任意改變表格的大小、欄寬與列高
  - 若再拖曳分格線時同時也按住<*Shift*>鍵，則可以讓每一欄寬或列高相等



表格的使用

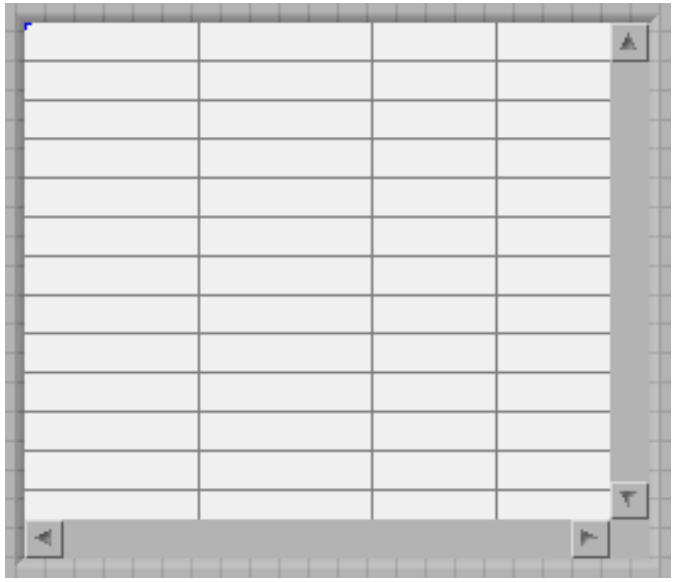
- 快速表格（Express Table）
  - Express Table是LabVIEW 7以後新增的一個物件，位置如右圖所示
  - 當我們點選它時其實是啟動一個叫做mergeTable.vi的程式



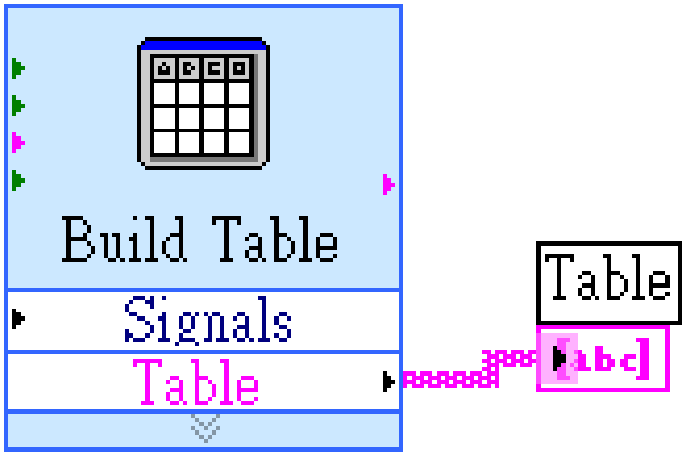
表格的使用

- Express Table可以將一個波形的資料轉換成文字型態，這其中當然包括每一點的資料及其相對應的時間

前置面版區



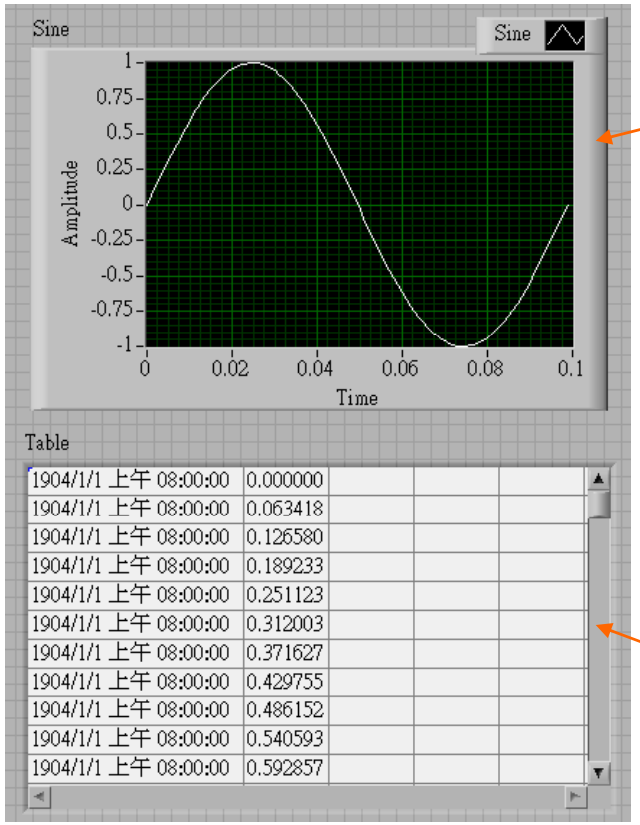
圖示區



表格的使用

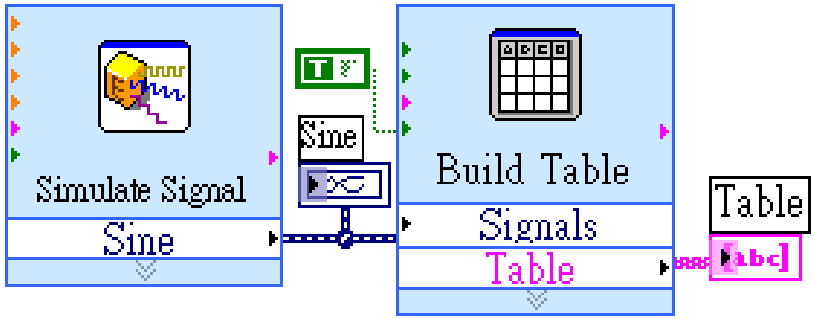
- Express Table的使用範例如下圖：

前置面版區

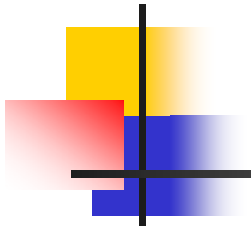


原始波形圖

圖示區



轉換完成的
資料表格

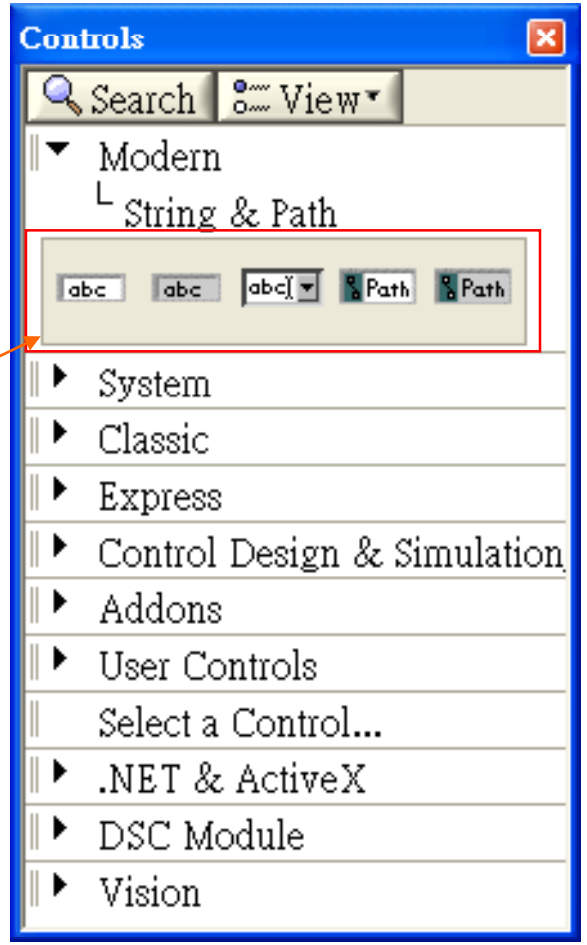


字串列與檔案存取

- 字串列
- 字串列函數
- 檔案存取

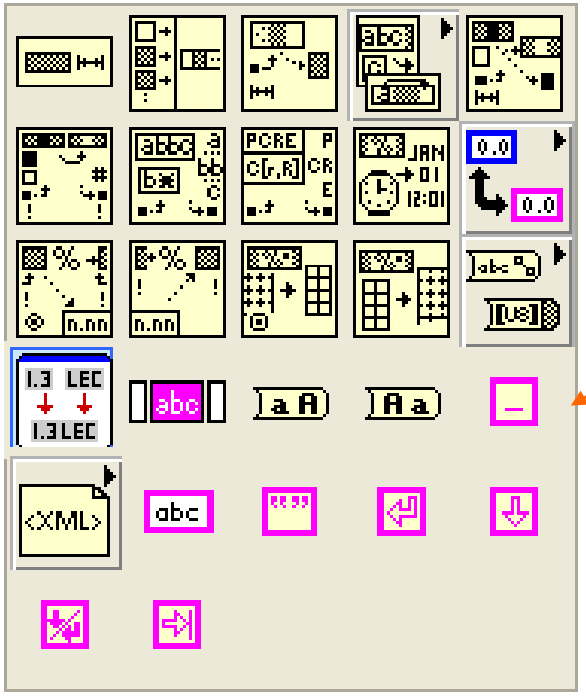
字串列

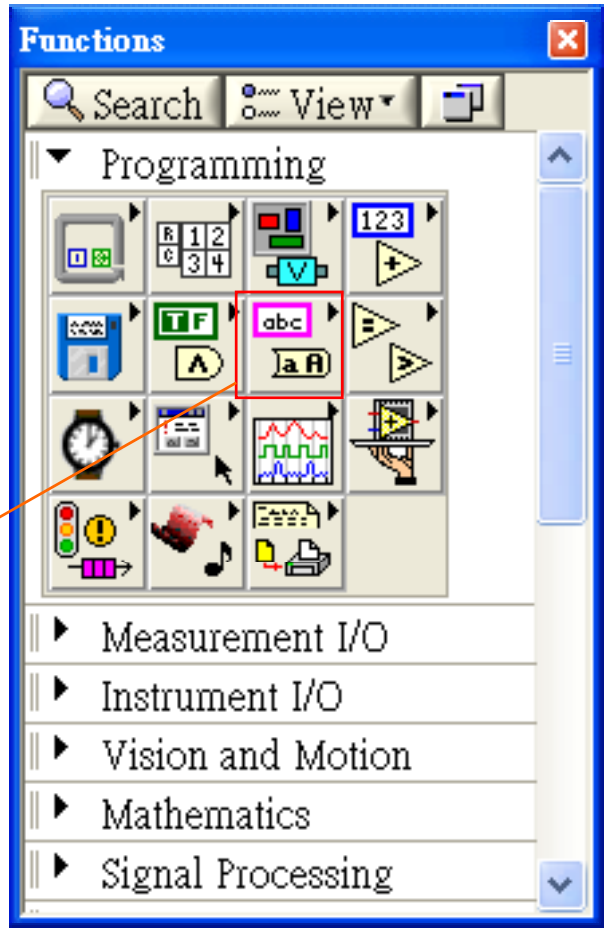
- 字串列（*String*）是指由ASCII 碼所組成的一個集合，在字串列中可輸入文字或數值。改變顯示框大小
- 字串列（*String*）物件可如右圖所示選取



字串列函數

- 字串列（String）函數可如右圖所示選取

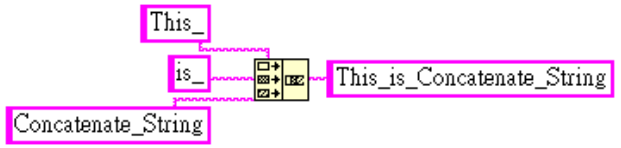




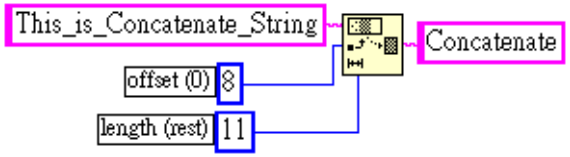
常用的字串列函數

- 字串列長度 （*String Length*）
  - 此函數是計算所輸入字串列的字元長度 。

- 組合字串列 （*Concatenate Strings*）
  - 此函數將所輸入的字串列或數列組成一個新的字串列。

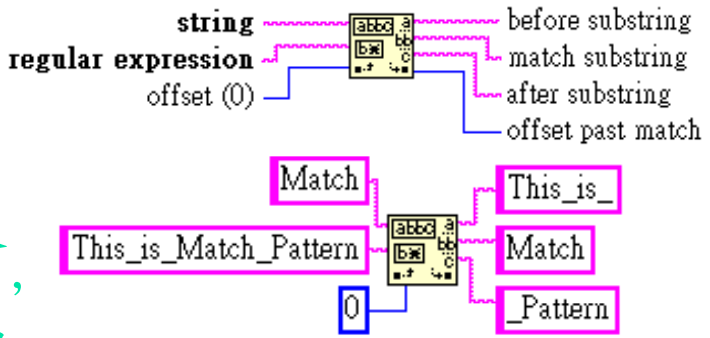


- 字串列子集 （*String Subset*）
  - 此函數可經由設定的初值(由offset指定)與字串列長度，擷取其中某一長度的字串列，與索引值相同，初值亦是由0開始。

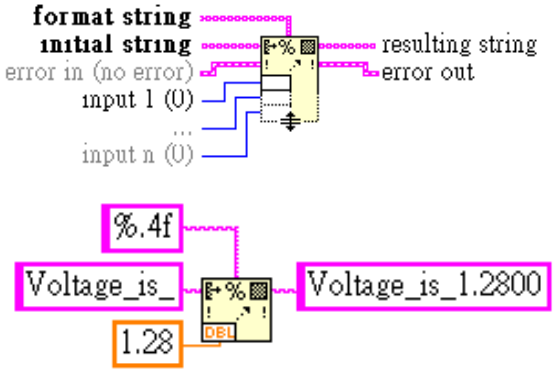


常用的字串列函數

- **文字配對** （*Match Pattern*）
  - 此函數可從一個字串列內找出一符合我們所需要的字串（由 regular expression 設定 ），當找到該字串後會將字串列分成三個部份。若沒找到，則輸出端 offset past match 會輸出值 "-1"。

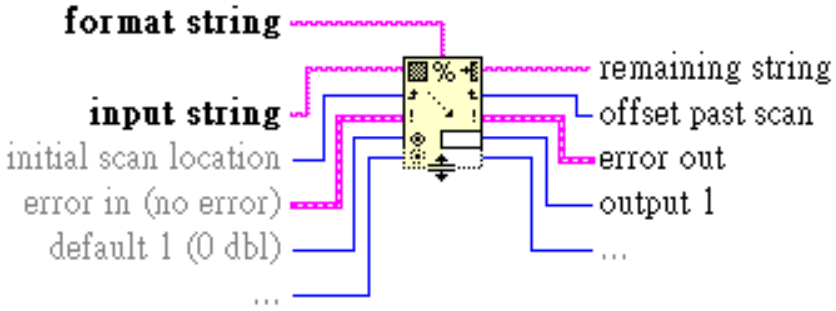


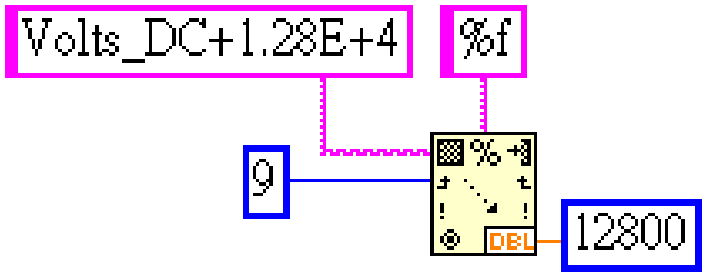
- **字串格式化** （*Format Into String*）
  - 此函數能將**敘述文字**（*argument*）項，轉換成特定的型式（由 format string 來設定 ），如修改資料型態、精確度，然後能與 initial string 項結成一新的數列。



常用的字串列函數

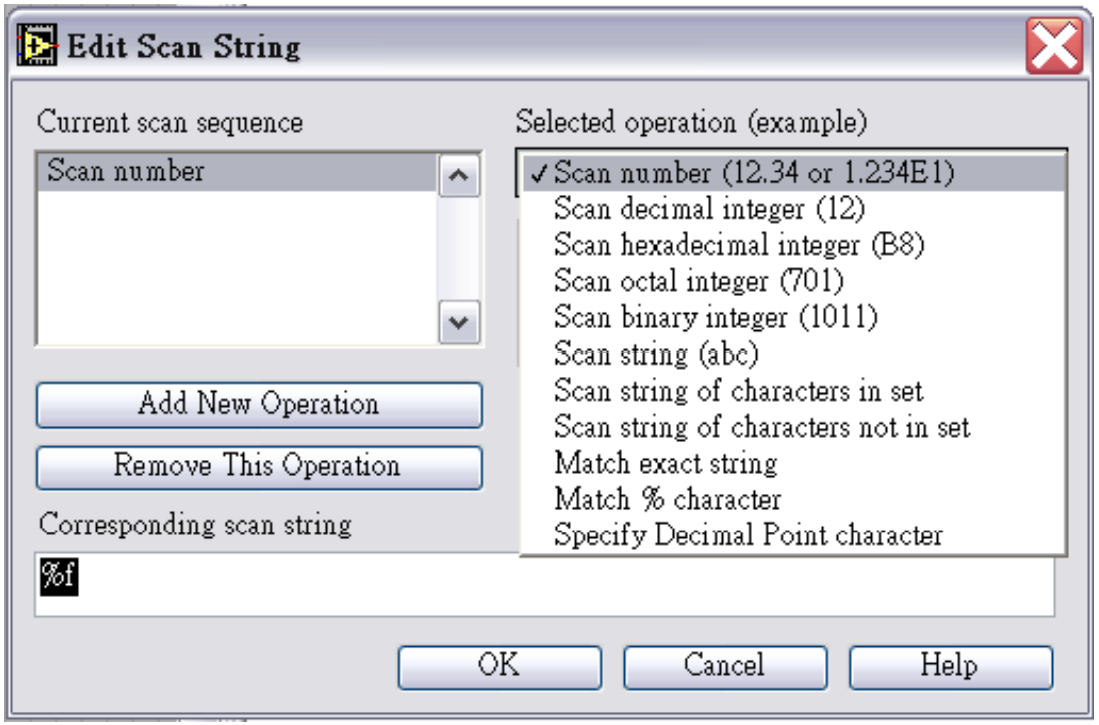
- 掃瞄字串（*Scan From String*）
  - 此函數的功能是將字串列的數值部分，透過format string設定，轉換成我們所設定的數值型式，以及利用initial search location（如同索引值）找出該字元的位置，然後輸出。





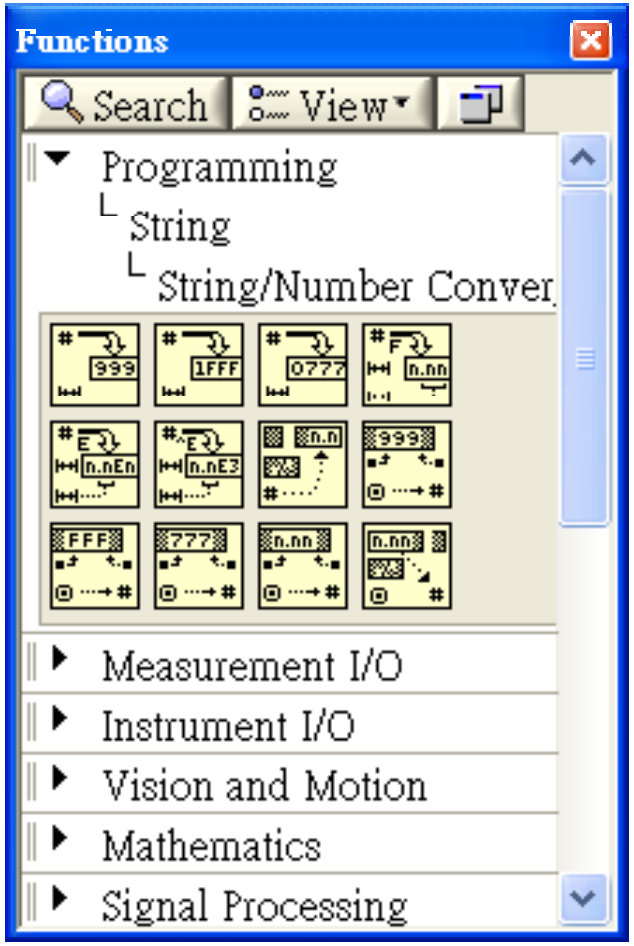
常用的字串列函數

- 以上字串格式化（*Format Into String*）與掃瞄字串（*Scan From String*）都有一個特定的設定對話框，也就是字串格式的設定，透過它可以改變字串型態、精確度以及轉換後數值的長度等



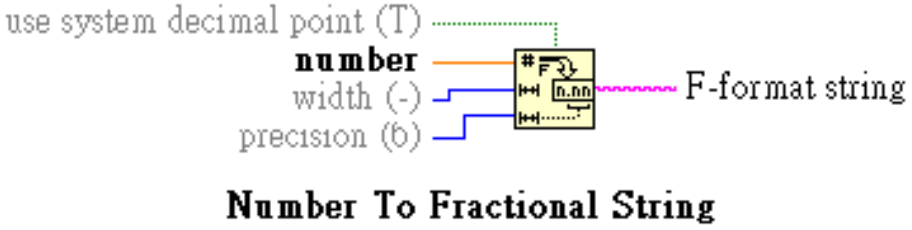
常用的字串列函數

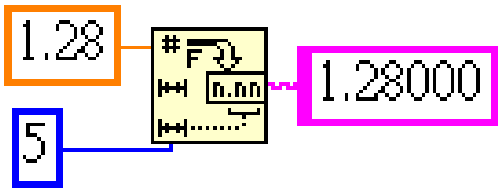
- 數字與字串列轉換的函數工具 （*String/Number Conversion*）



常用的字串列函數

- 數字轉換浮點字串（*Number To Fractional String*）
  - 此函數可改變數值的型態，
    然後將它轉換成字串列。

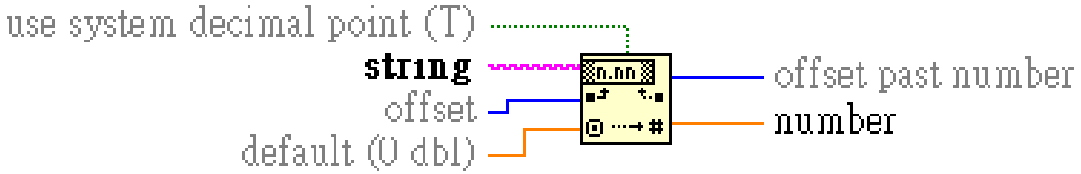


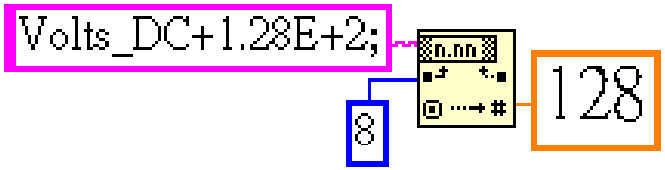


常用的字串列函數

- **浮點/指數字串轉換數字（*Fraction/Exp String to Number*）**

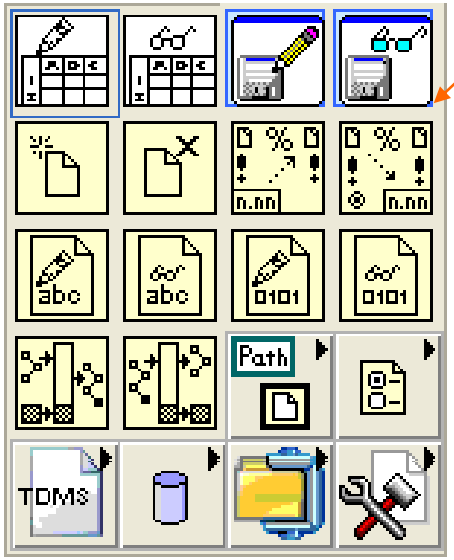
  此函數可將字串列內數字的部份轉換成一數值。

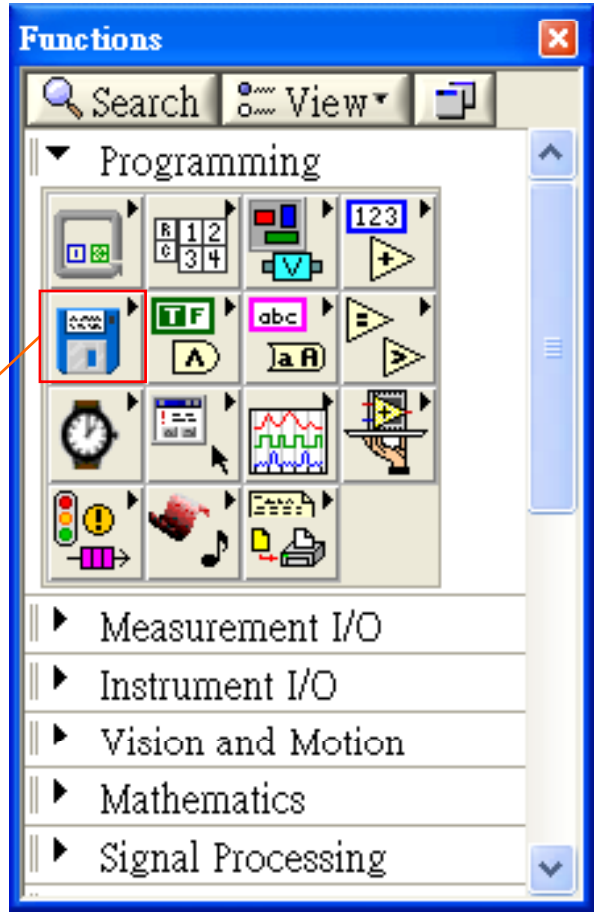




檔案存取

- 檔案存取三步驟：
  1. 開啟舊檔或新的檔案。
  2. 讀取或寫入檔案資料。
  3. 關閉檔案。





檔案存取

High-Level Utility file VIs：

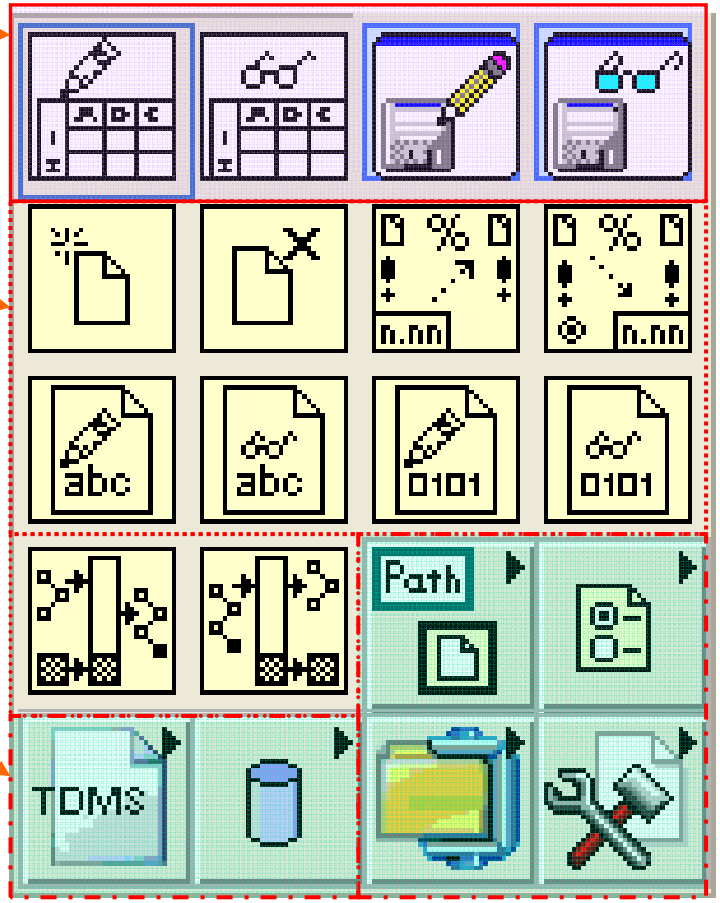
提供一些只需簡單設定即可使用的函數，功能較簡單

Intermediate file I/O VIs：

提供一些較具彈性與完整功能的函數，設定上較複雜但幾乎可以處理大部分的檔案存取

Advanced file functions：

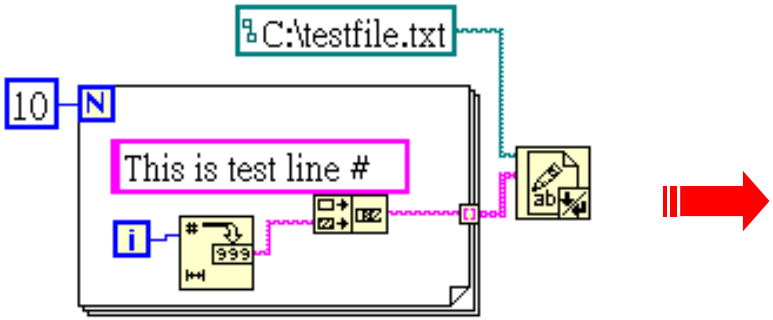
提供一些複雜但更強的函數

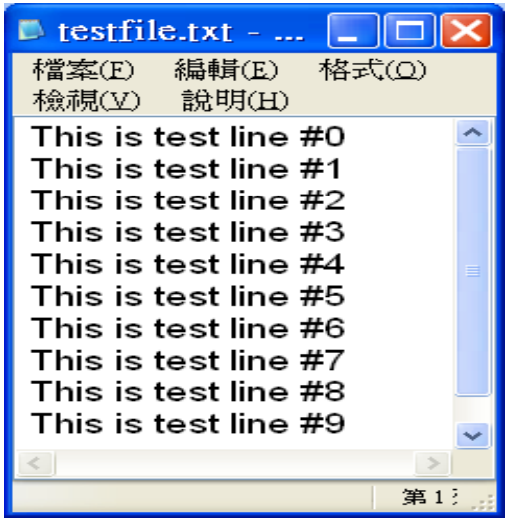


Intermediate file I/O

- 寫入到文字檔案（Write to Text File VI）

  此函數的功能是將字串或字串列寫入一文字檔

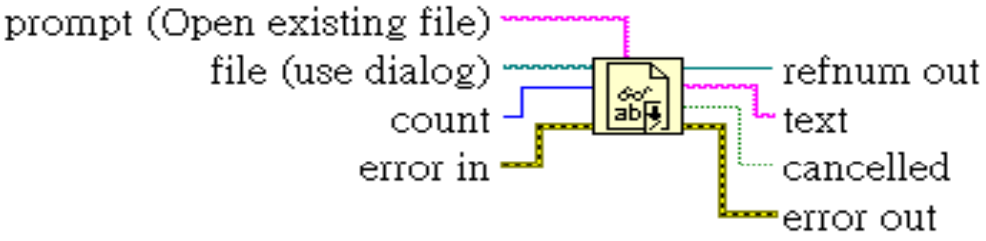




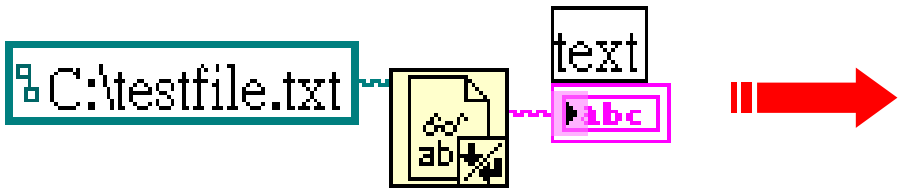
Intermediate file I/O

- 從檔案讀出字串（Read From Text File VI）

此函數的功能是讀取文字檔內的特定數量的字元或數行的文字



圖示區



前置面版區

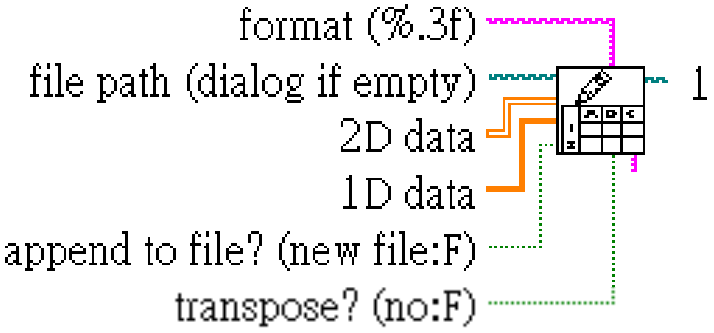
text

This is test line #0
This is test line #1
This is test line #2
This is test line #3
This is test line #4
This is test line #5
This is test line #6
This is test line #7
This is test line #8
This is test line #9

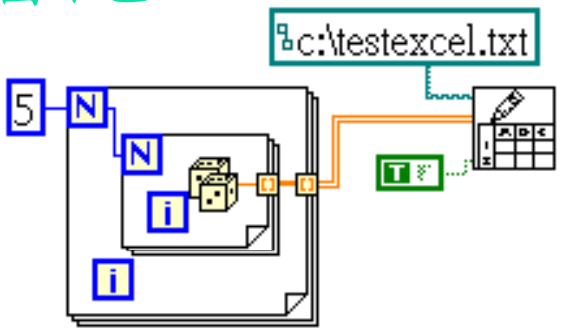
高階檔案存取函數

- 寫入試算表檔案（Write To Spreadsheet File VI）

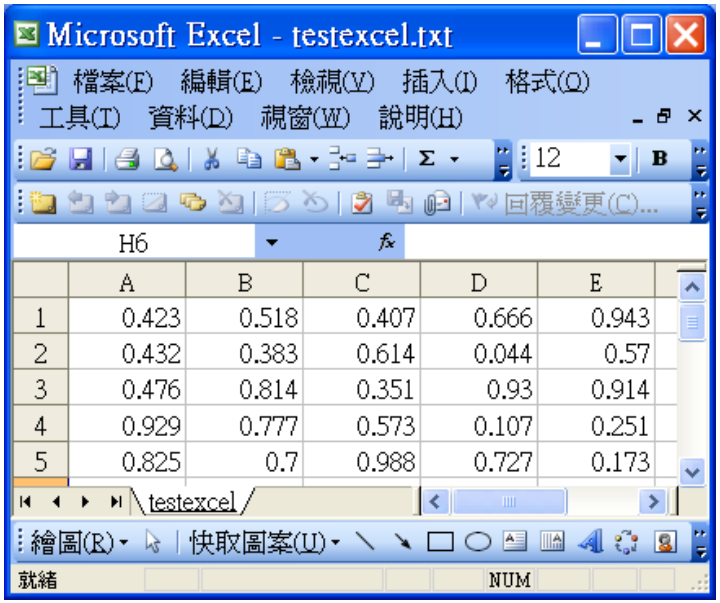
此函數是將一維或二維的單精度數值數列轉換成一字串列，然後將此字串儲存於一新的表格式文字檔內或增添於舊檔中



圖示區



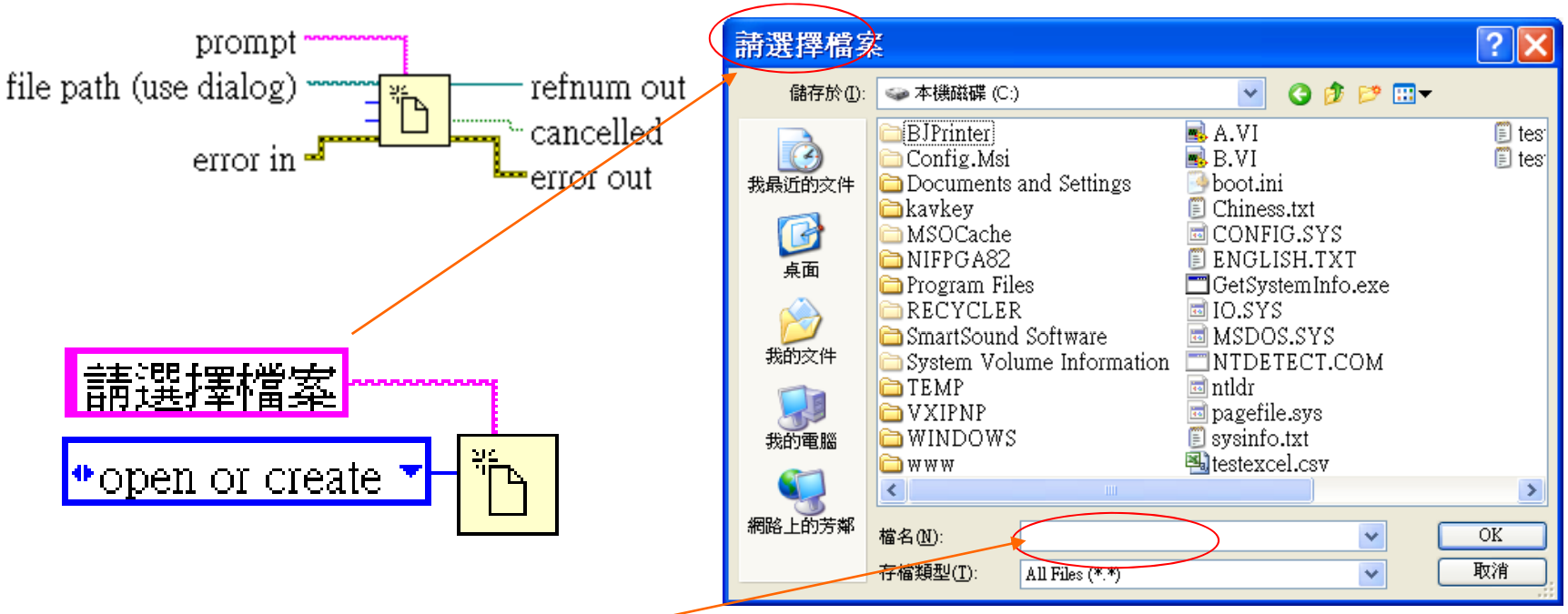
EXCEL



低階檔案存取函數

- **開啟/建立/代換檔案（Open/Create/Replace File VI）**

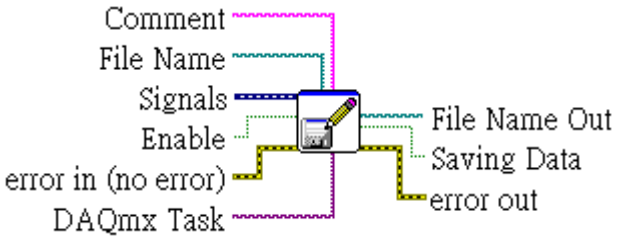
此函數包含開啟舊檔、建立新檔、覆蓋舊檔等功能



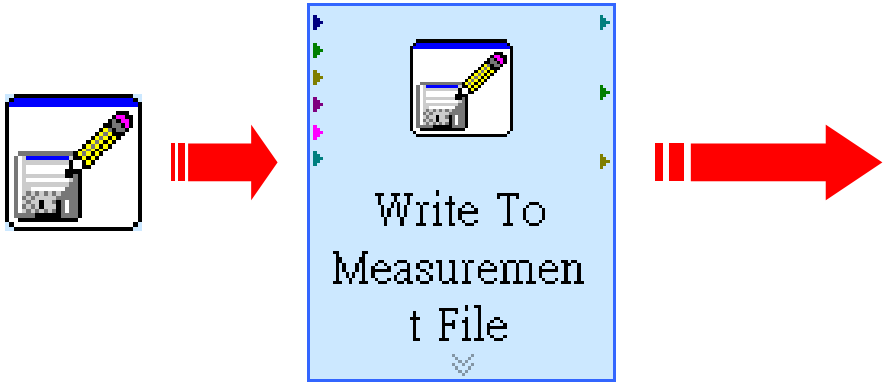
File path如果不指定檔案路徑或名稱，會出現一對話框（如上圖）要求輸入檔案路徑或名稱

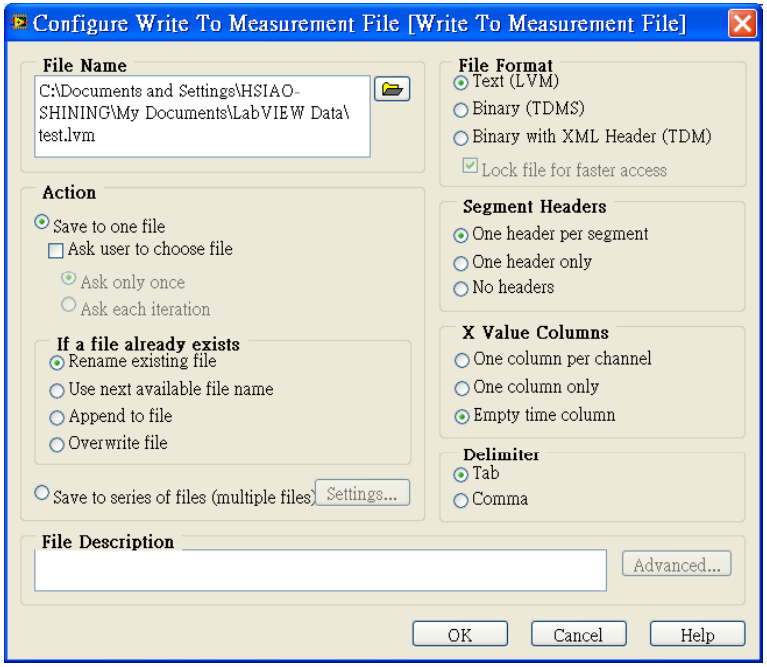
檔案存取（Write To Measurement File）

Write To Measurement File：



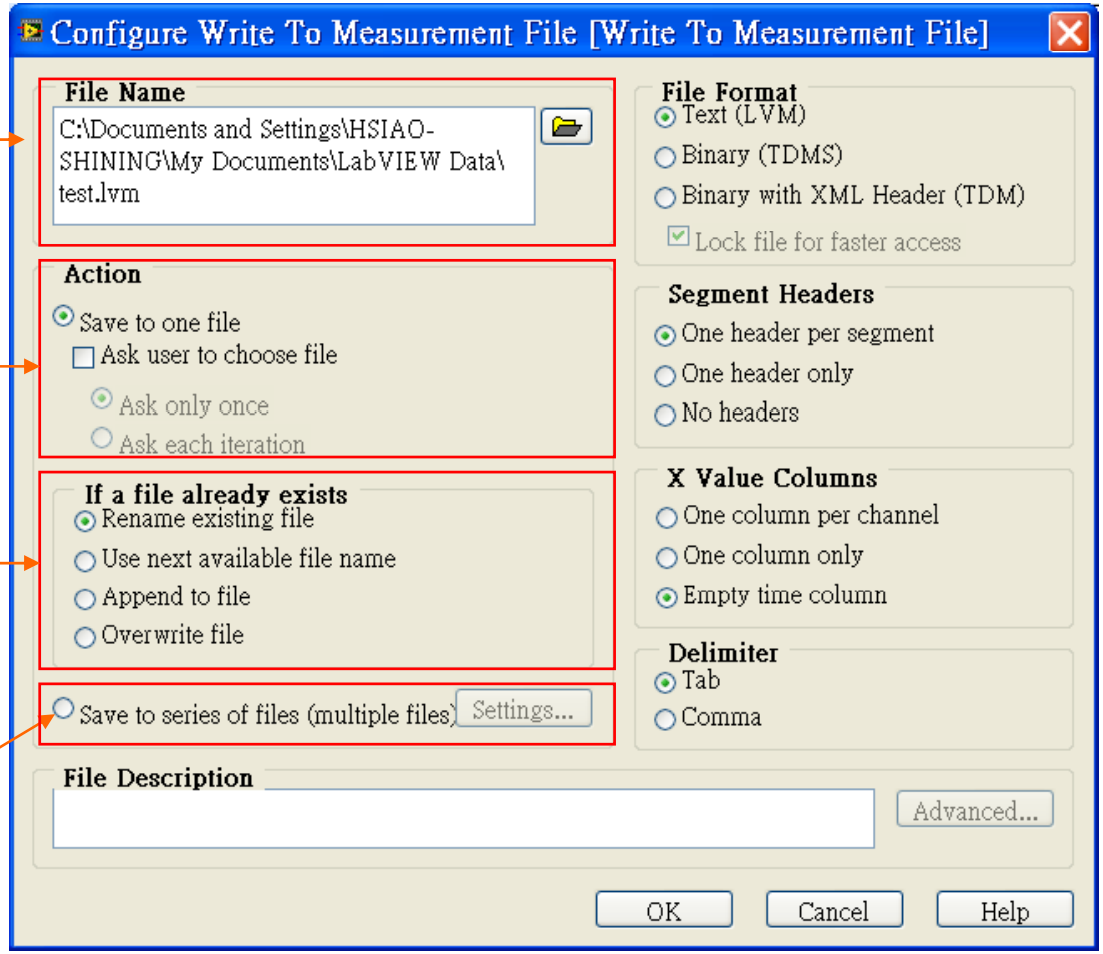
LabVIEW 8.2的新增功能，當將它拖曳到圖示區時會出現如下圖所示的Express VI並彈出一個對話框



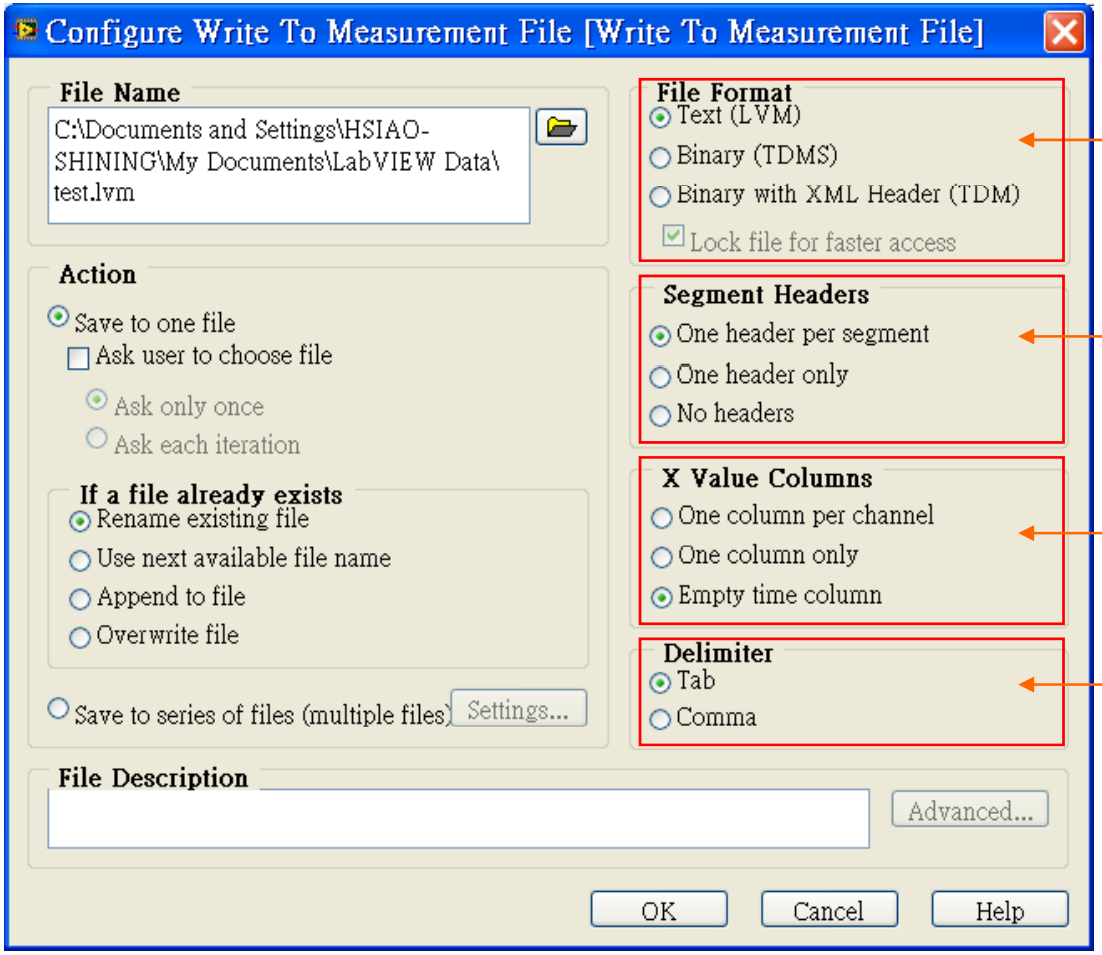


檔案存取（Write To Measurement File）

- File Name：使用者可以在這裡定義檔案的路徑與名稱
- Save to one file：使用者可以在這裡定義存檔的動作
- If a file already exists：當檔名已經存在十的動作可在此設定
- 這裡可以設定多檔名儲存



檔案存取（Write To Measurement File）

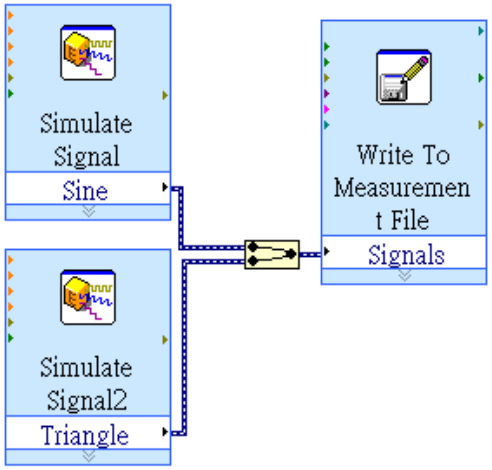


- File Format：使用者可以在這裡定義檔案的格式
- Segment Headers：使用者可以在這裡定義表頭
- X Value Columns：定義X軸的存檔格式
- 設定分隔字元

檔案存取（Write To Measurement File）

Write To Measurement File：

- 包含了開啟（open），寫入（write），關閉（close）和 錯誤訊息處理（error handling functions）
- 可以處理與接受包含< tab>或<comma>分隔符號
- 可以接受動態的資料型態



Microsoft Excel - test.lvm

	A	B	C	D
1	LabVIEW Measurement			
2	Writer_Versio	0.92		
3	Reader_Versio	1		
4	Separator	Tab		
5	Multi_Heading	Yes		
6	X_Columns	No		
7	Time_Pref	Relative		
8	Operator	HSIAO-SHINING		
9	Date	2007/7/12		
10	Time	24:24.5		
11	***End_of_Header***			
12				
13	Channels	2		
14	Samples	100	100	
15	Date	2007/7/12	2007/7/12	
16	Time	24:24.5	24:24.5	
17	X_Dimension	Time	Time	
18	X0	0.00E+00	0.00E+00	
19	Delta_X	0.001	0.001	
20	***End_of_Header***			
21	X_Value	Sine	Triangle	Comment
22		0	0	
23		0.063418	0.0404	
24		0.12658	0.0808	
25		0.189233	0.1212	
26		0.251123	0.1616	
27		0.312003	0.202	
28		0.371627	0.2424	
29		0.429755	0.2828	
30		0.486152	0.3232	
31		0.540593	0.3636	
32		0.592857	0.404	