

LabVIEW 的基礎練習

一、目的

藉由簡單的練習對LabVIEW程式設計有初步的認識。

二、實驗操作

本實驗分成兩大部分：第壹部分是 LabVIEW 程式的簡單練習，共有 13 個實做練習。

第壹部分、LabVIEW 程式設計基礎篇

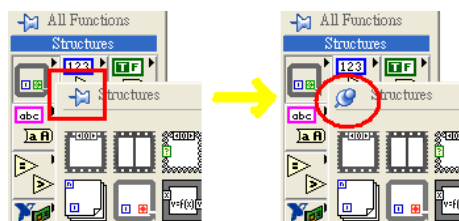
既然已經上過 LabVIEW 的第一堂課，相信同學們都對於 LabVIEW 的工作環境有一些基本的認識了，那麼廢話不多說，就從實做例題開始練習吧！

Tip1 ... 鍵盤捷徑

Ctrl+N	建立一個新的 VI	Ctrl+B	將程式方塊圖中所有壞線移除
Ctrl+R	執行目前的 VI	Ctrl+T	將人機介面與程式方塊圖左右並列
Ctrl+W	關閉目前的 VI	Ctrl+E	在人機介面/程式方塊圖之間切換
Ctrl+H	顯示隱藏 Help 視窗	Shift+滑鼠右鍵	呼叫工具面版

Tip2 ... 圖釘

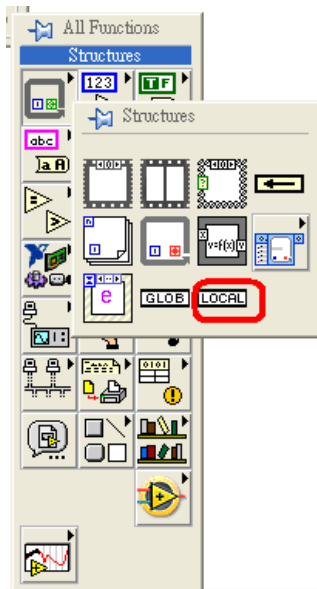
按右鍵可分別在人機介面及程式方塊圖上呼叫出控制面板(Control)與函數面版(Function)，但若經常使用到某個子面版，想將它固定在螢幕上，則可以在此面版左上角的圖釘上點一下使它立起，即可擁有此固定的視窗，並可將它移動至你想要的位置，不想要時也可以關閉它。



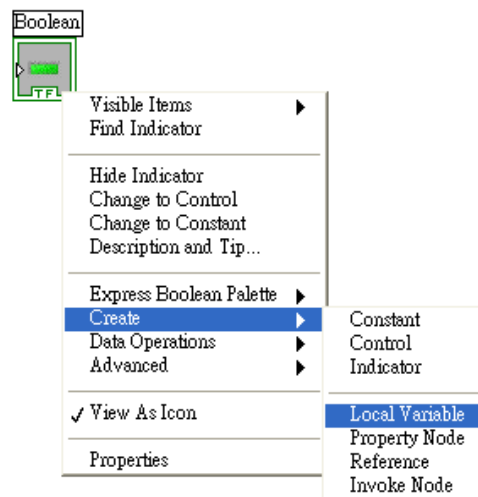
Tip3 ... 區域變數(Local Variable)

當一個變數必須同時出現在其他許多小程序時，就要用到分身變數。產生方法如下：

(1)



(2)



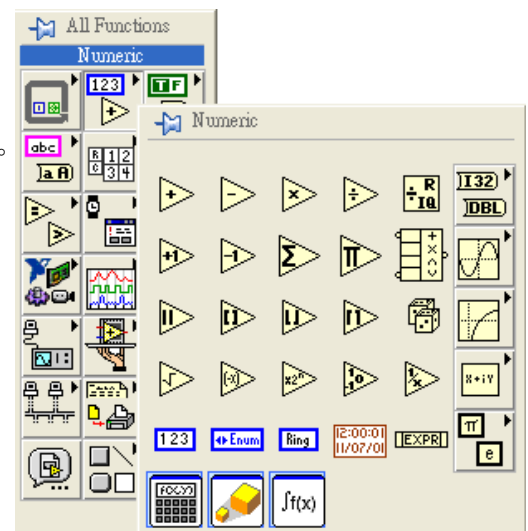
(3)

若要同時有好幾個分身變數
 →已產生的**local V.**
 →**Ctrl+滑鼠**拉出
 →即產生另一個**local V.**

◀ 實做練習一 ▶ 複習數值元件(Numeric)的使用

請寫出一程式，運算任意 x 值經過下列運算，並顯示其結果 y 。

$$y = \frac{1}{\sqrt{3x^2 + 5}}$$

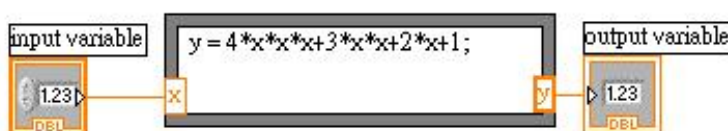


Tip 4 公式節點...

倘若你的公式很冗長，程式方塊必定會很複雜，所以 LabVIEW 中有一個方便的公式節點，讓你更輕鬆地表達數學公式。Functions> Programming>Structures>Formula Node

選擇公式節點後在方框內鍵入數學式，並選擇 Add Input 與 Add Output。利用接線的方式將數值連結到所增加的輸入與輸出點上，注意在每個方程式的敘述結尾處都要加上一個分號。

例：公式 $y = 4x^3 + 3x^2 + 2x + 1$

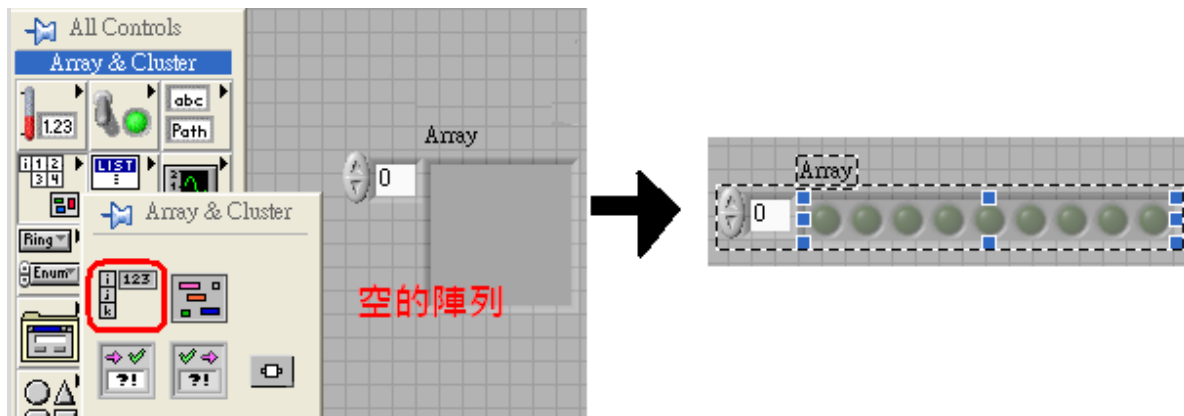


◀ 實做練習二 ▶ 複習布林元件(Boolean)的使用

請寫出一程式，將任意一個十進位的數值轉換成布林二進位陣列。

(十進位數值使用整數，布林二進位陣列使用圓形 LED 燈泡)

產生一個空的陣列，再從控制面板加入 Round LED，並往右拖曳成多個 Round LED 陣列



Functions > Programming > Boolean > Number To Boolean Array

複習 LabVIEW 程式結構(Sequence、Case、For Loop、While Loop)

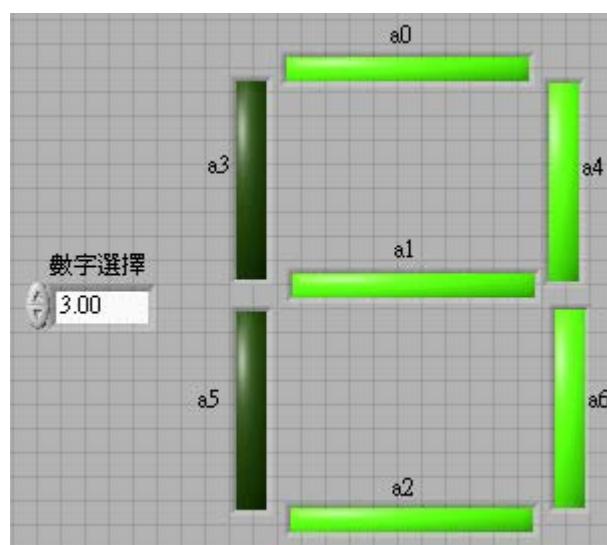
◀ 實做練習三 ▶ 七段顯示器(0~9) --- Using 條件結構(Case)

我們使用條件結構(Case)，讓選擇七段顯示器該亮出選定的數字來～

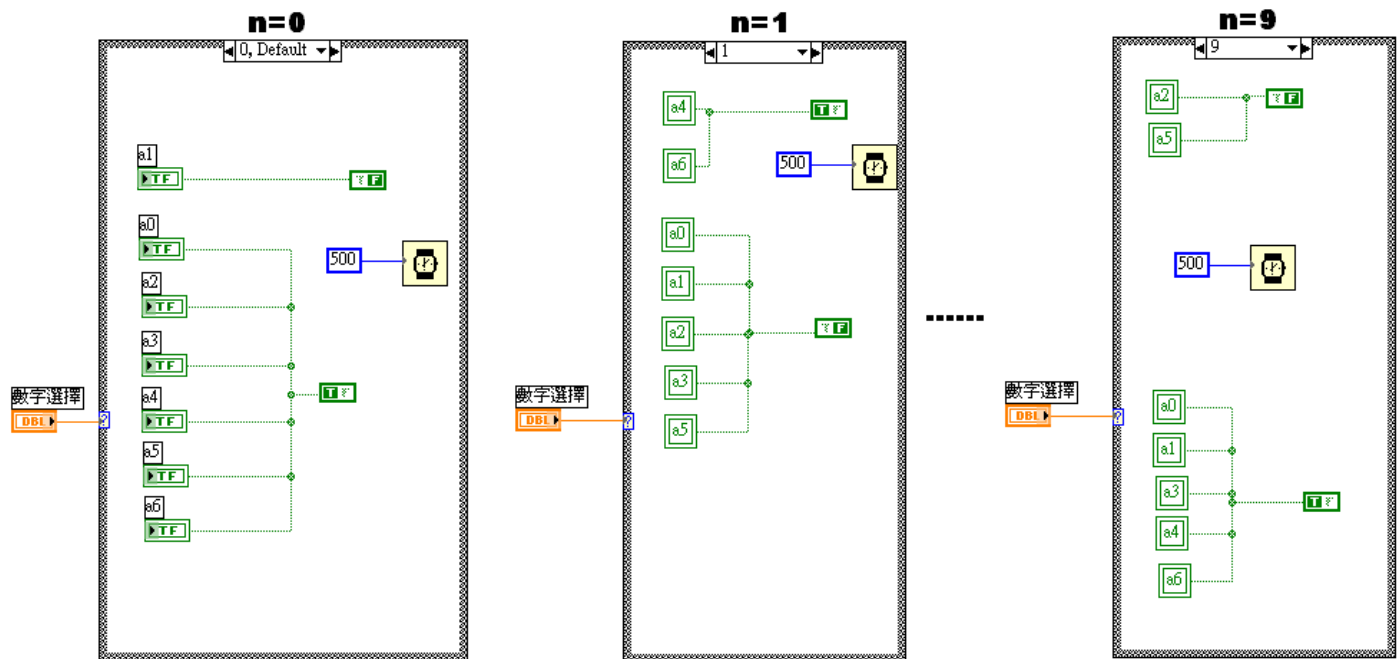
◆ 程式說明：

程式執行順序由數字選擇元件來決定，若選擇 0 的話，則會執行第一層的迴圈(0,Default)，人機介面上會由布林燈泡形成一個 0 的數字；若選擇 1 的話，則會執行第二層的迴圈(1)，人機介面上會由布林燈泡形成一個 1 的數字。以此類推，即可顯示由 0~9 的數字。請練習！

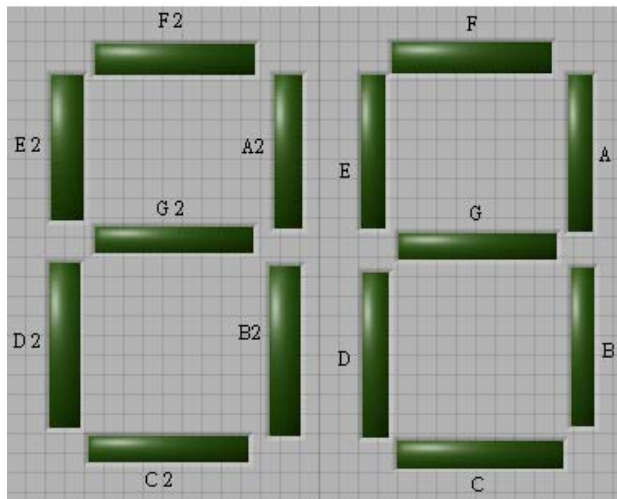
人機介面面版區



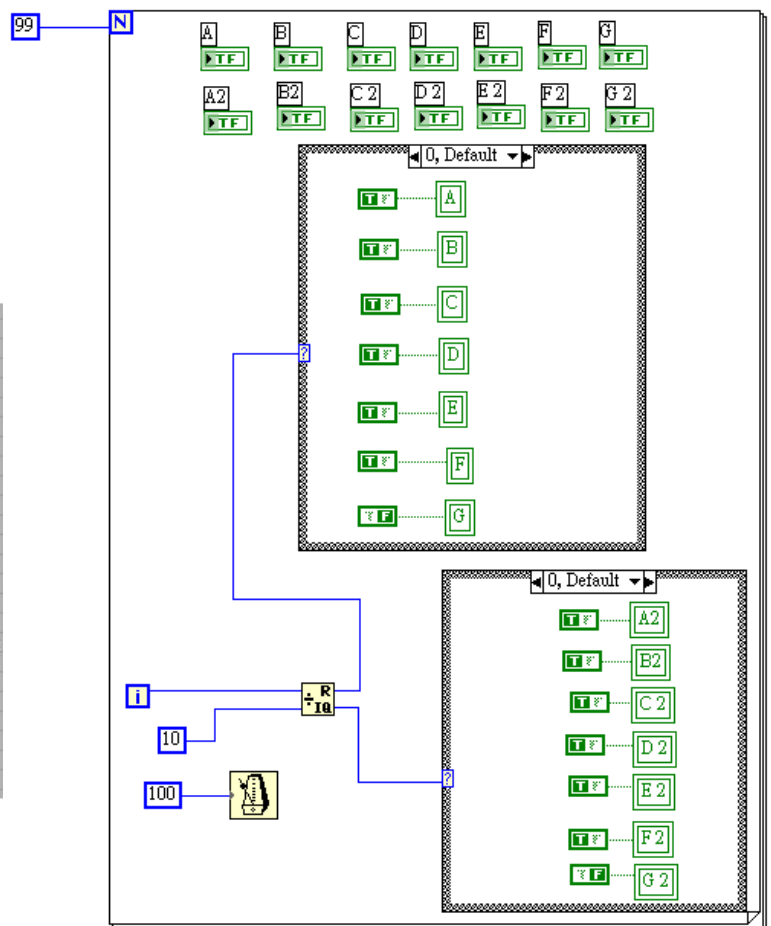
程式方塊編輯區



◀ 實做練習四 ▶ 七段顯示器(0~9) --- Using 次數迴圈(For Loop)



人機介面面版區



程式方塊編輯區

◆ 程式說明：

迴圈執行次數索引值 N 會從 0 往上執行共 99 次，每一次的次數索引值會先跟 10 做除法運算。例如 8

除以 10 會於 8，所以個位數會顯示 8，而十位數則為 0。倘若索引值數到 15 時，15 除以 10 會餘 5，而商數等於 1，那麼個位數會變成 5，十位數則顯示 1。以此類推。

.....

Tip5 ... 時間函數

(1). Wait (ms) : Functions>All Functions>Wait (ms)

功能：讓 VI 在等待特定時間(毫秒)後才繼續執行

milliseconds to wait —  — millisecond timer value

Wait (ms)

(2). Wait Until Next ms Multiple : Functions>All Functions>Wait Until Next ms Multiple

功能：能夠使 LabVIEW 在等待輸入時間的整倍數後才繼續執行 VI，讓迴圈以特定的時間間隔來執行，或者是使兩個動作能夠同步進行

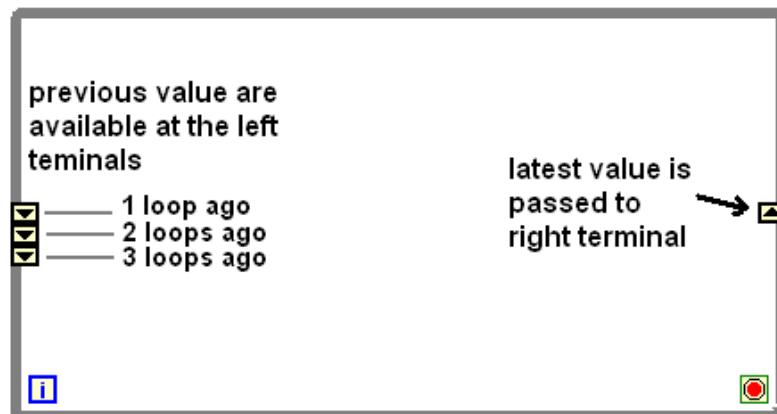
millisecond multiple —  — millisecond timer value

Wait Until Next ms Multiple

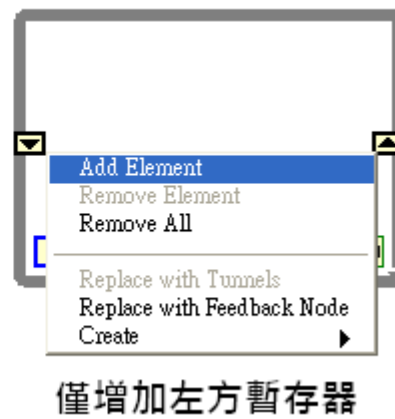
.....

Tip6 ... 移位暫存器

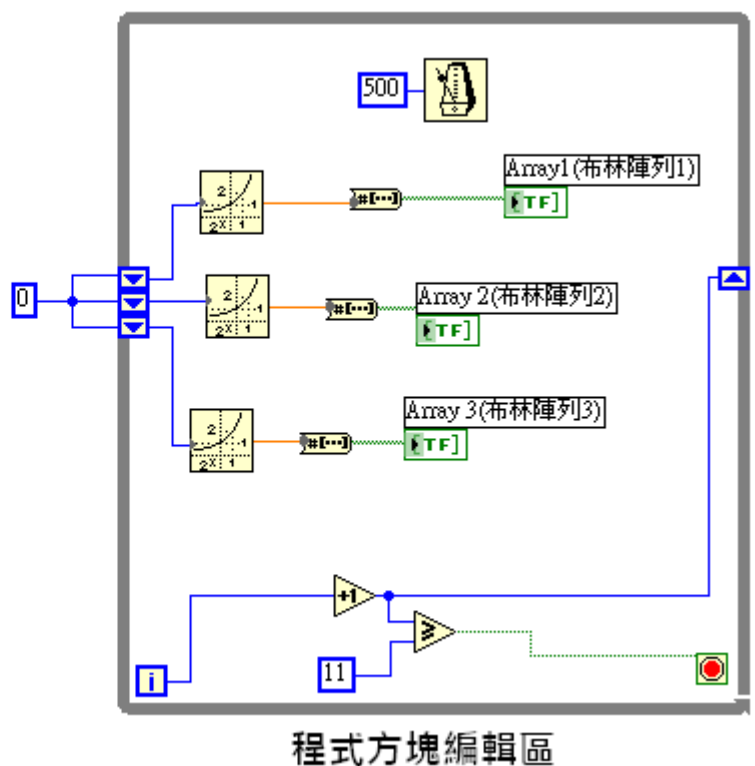
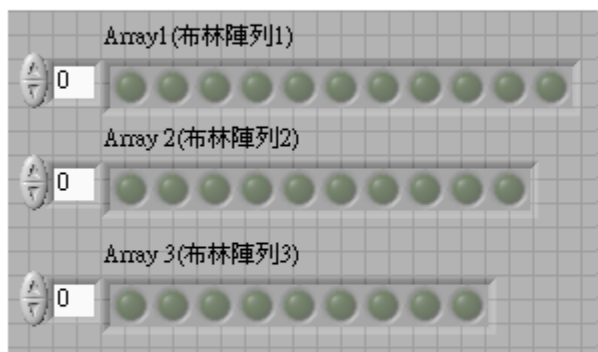
可作為 For Loop 及 While Loop 使用，是一種特殊型態的變數，用來作為迴圈的本次與下次之間的數值轉移。右邊的暫存器接收並儲存了第一次迴圈完成的資料，並且在下一次迴圈完成前，將資料送至左邊的暫存器中。資料運作順序說明如下圖所示。



增加移位暫存器的方法：



◀ 實做練習五 ▶ 由左往右的跑馬燈 --- Using 條件迴圈(While Loop) + 移位暫存器(Shift Register)



◆ 程式說明：

i 值從 0 開始執行， $0+1=1$ ，將此結果值 1 送至右邊移位暫存器，並且判斷 $1 \geq 11$ 是 False，所以繼續進行下一迴圈，並在下一迴圈開始前將數值 1 送至左邊最上面的移為暫存器。

接下來執行 $i=1$ ， $1+1=2$ ，此結果值 2 被送至右邊移位暫存器，並且判斷 $2 \geq 11$ 是 False，所以繼續進行下一迴圈，並在下一迴圈開始前將數值 2 送至左邊最上面的移為暫存器，而原本在左上暫存器中的 1 則被送至左中的暫存器中。以此類推，當執行到 $i=10$ 時， $10+1=11$ ， $11 \geq 11$ 被判斷為 True，程式執行就會停止。左邊移位暫存器中的數字先被運算為 2 的 x 次方，再轉為布林二進位。可參考下面十進位轉二進位之算法。 2^0 代表第一個燈， 2^1 代表第二個燈， 2^2 代表第三個燈...

2^0 代表左一， 2^1 代表左二	十進位	二進位（注意！燈的編號和此數字排列反向）
2^2 代表左三		

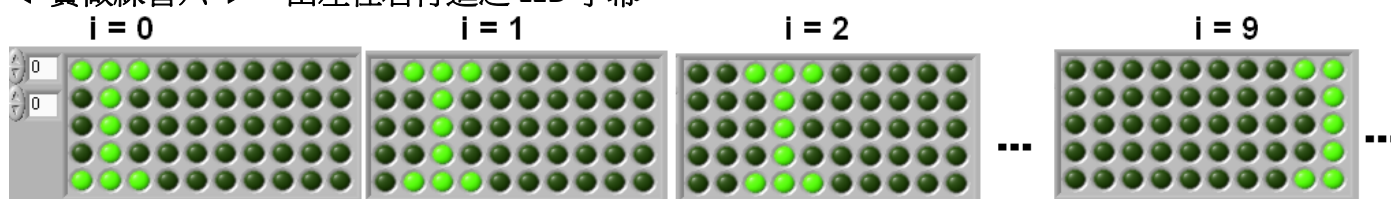
2^0	1	1	
$1 \times 2^1 + 0 \times 2^0$	2	10	
$1 \times 2^1 + 1 \times 2^0$	3	11	
$1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$	4	100	
$1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$	5	101	
$1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$	6	110	
$1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$	7	111	

整個動作會由第一排的第一個燈開始亮，等亮到第一排的第二個時，第二排的第一個燈也會亮起。Loop 外面有一個整數值 0，它在此扮演的角色是起始你的暫存器內之數值。若少了”起始”這個動作的話，下一次執行時，移位暫存器將會包含前面執行剩下來的數值。

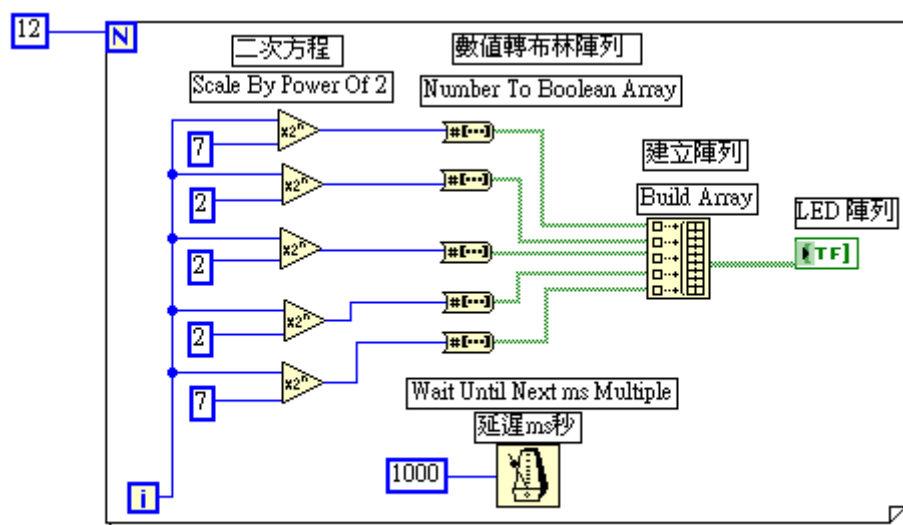


2 的 x 次方：Functions > Mathematics > Elementary & Special Functions > Power of 2

◀ 實做練習六 ▶ 由左往右行進之 LED 字幕



由左往右跑的 LED 字幕



程式方塊編輯區



以 2 為底之指數函數：Functions>Programming>Numeric>Scale by Power of

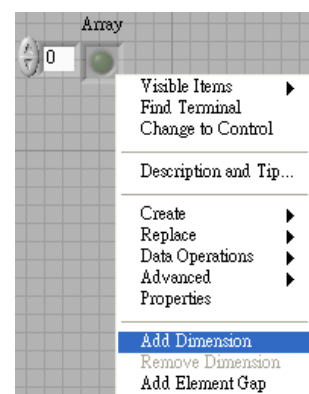
2 (ex~ $7 \times 2^0 = 7$)



建立陣列：Functions > Programming > Array > Build Array

在此建立之陣列是二維陣列，所以產生之 LED 陣列要先將它轉成二維陣列：

對新產生之陣列，按右鍵，選擇”Add Dimension”，接著在右下角拖曳成你要的陣列大小。



◆ 程式說明：

LED 亮燈顯示為字母 I，起始時，第一行及第五行皆是亮第 1、2、3 顆燈（111=7，起始表示為 7×2^0 ），第二、三、四行皆亮第二顆燈（10=2，起始表示為 2×2^0 ），i 值往上增加為 1 時，數值由起使之 $\times 2^0$ ，變為 $\times 2^1$ ，於是亮燈往右走一個。以此類推當 i=11 時，完成循環字幕之動作。

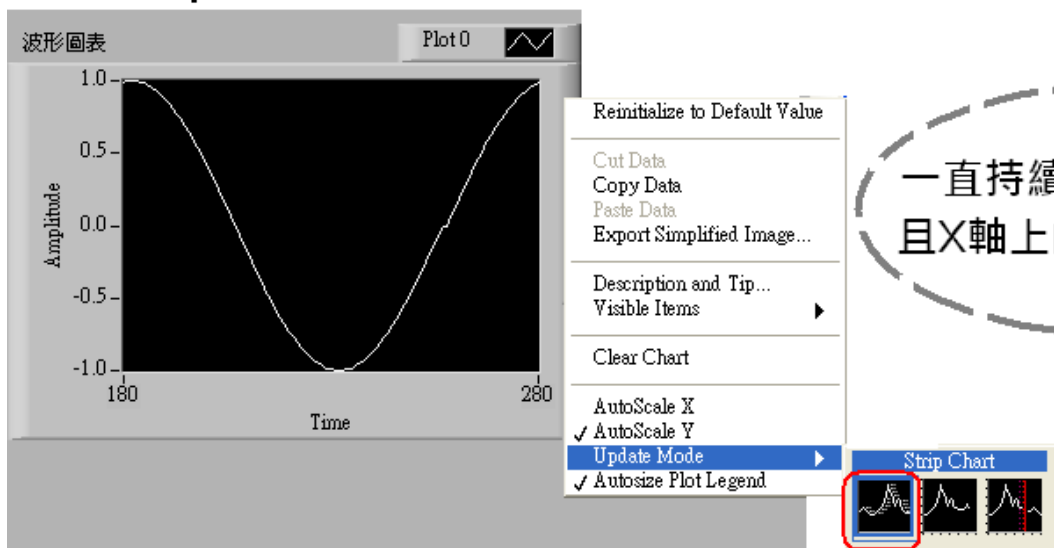
繪製圖表 (Waveform Chart) 與 圖形(Waveform Graph)

圖表(Waveform Chart)

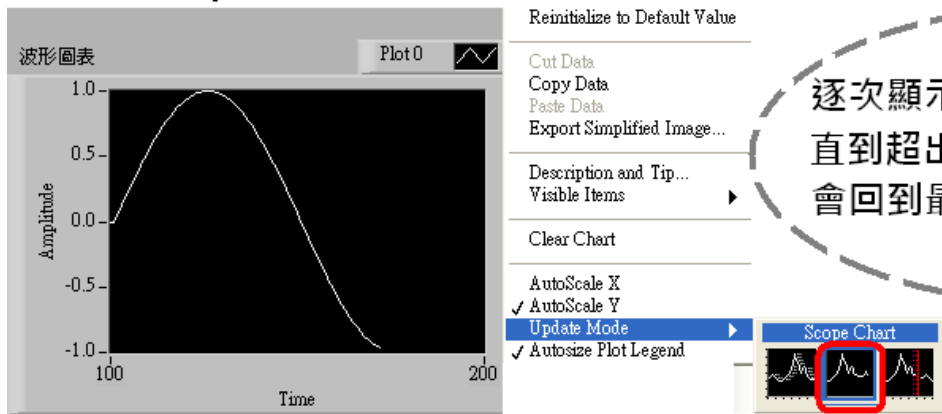
繪製的動作是將 X 軸對 Y 軸的數值用圖形顯示，通常 Y 軸代表繪製的資料數值，而 X 軸為時間。波形圖表的顯示元可以允許單點或多點資料的顯示。此顯示元大多使用在迴圈之內，用來持續並顯示先前的資料及繪製新的資料，使其有連續並更新資料的顯示能力。（通常每個 Y 軸數值的產生受限於迴圈的疊代，所以 X 軸代表著疊代的次數。）雖然 LabVIEW 只有一種圖表顯示元，但對於連續資料確有三種不同的更新模式。

圖表更新模式

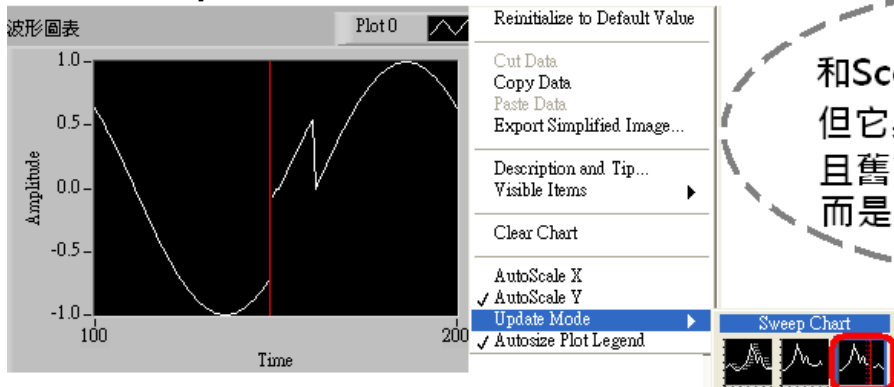
Strip Chart



Scope Chart



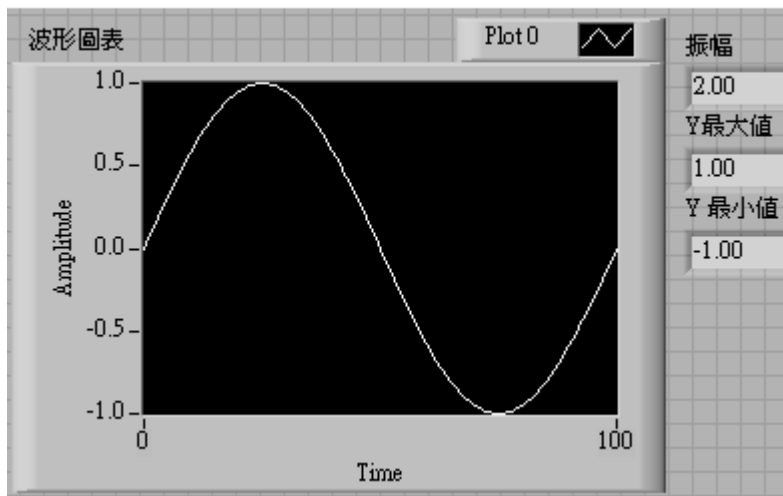
Sweep Chart



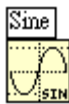
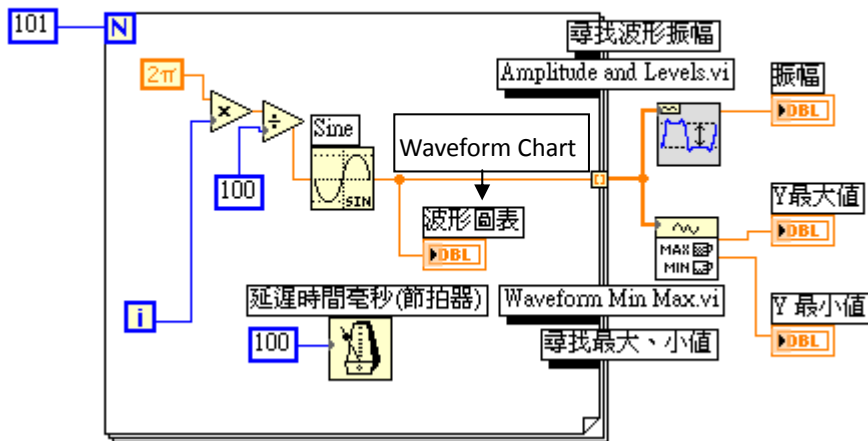
和Scope顯示波形的方式很相似，
但它具有一掃瞄線，
且舊有的波形不會馬上消失，
而是被新的數據所覆蓋。

◀ 實做練習七 ▶ 單點繪製圖表 - 繪製一個正弦函數波形

人機介面版面區



程式方塊編輯區



正弦函數：Functions > Mathematics > > Elementary & Special Functions > Trigonometric > Sine



尋找波形振幅：Function>Programming>Waveform>Analog waveform>Waveform Measurements>Amplitude and Levels



尋找最大、最小值：Function > Programming>Waveform > Analog waveform > Waveform Min Max

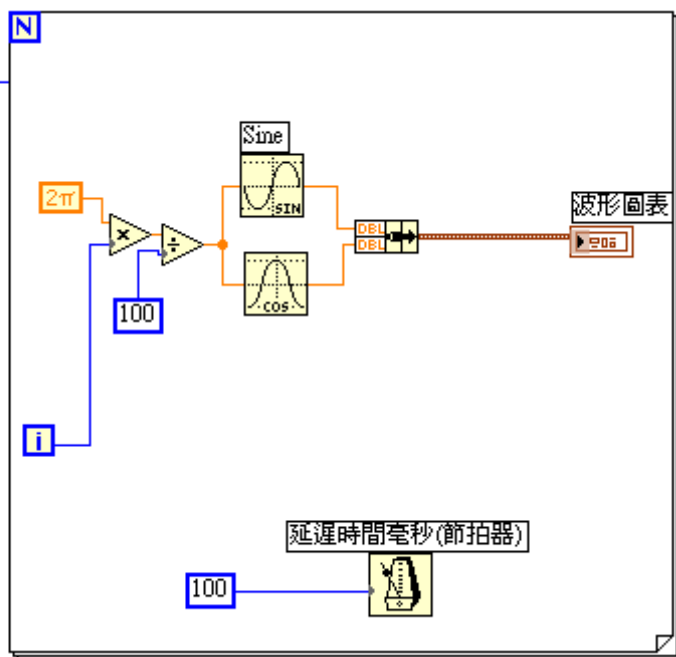
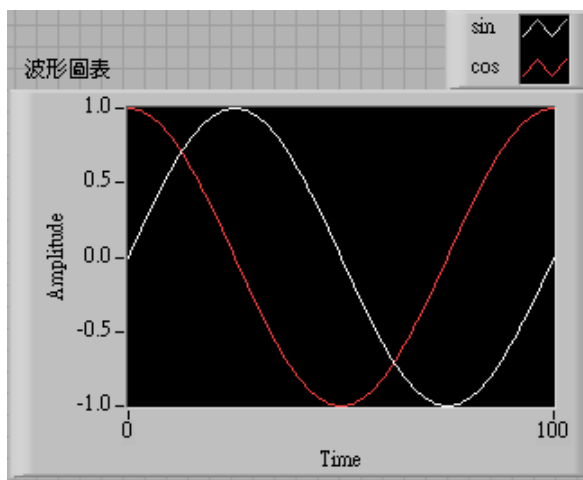
◆ 程式說明：

由於 LabVIEW 的角度是弧度量，所以設計程式時必須要用弧度量的單位計算。例如：當 For Loop 迴圈執行至第 51 次時，即 $i=50$ ，依照程式計算 $2\pi \times 50 \div 100$ 恰等於 π ， $\sin \pi = 0$ ，所以在此設計 $N=101$ ，恰使圖形上只出現一個完整 sine 波。在此範例中，單點輸入圖表，僅需簡單地將一個產生值(或說量測值)與圖表的程式方塊圖做連接，每次完成一個迴圈，可以輸入一組之資料點進入圖表。

◀ 實做練習八 ▶ 多點繪製圖表 - 同時繪製一個正弦函數+餘弦函數波形

人機介面版面區

程式方塊編輯區



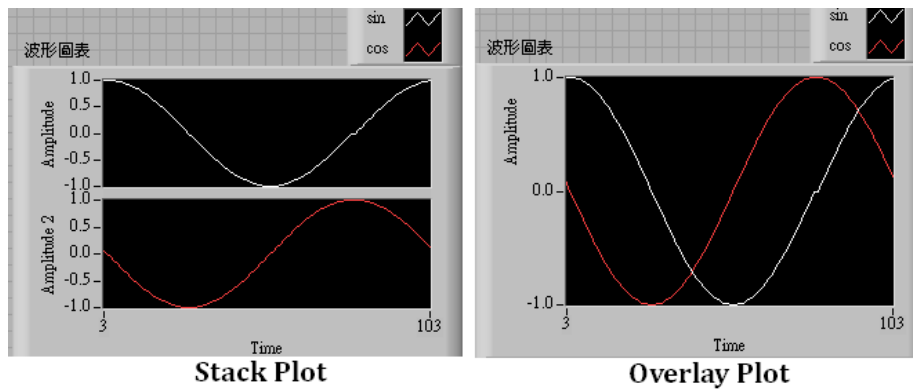
 Bundle(組合綜合集)將資料源連接在一起：Functions>Programming>Cluster, Class,& Variant> Bundle

◆ 程式說明：

波形圖表亦可繪製一組以上的資料，然而你不能將數個程式方塊圖的資料源共同連結在單一的圖表接點上。必須先使用 **Bundle** 功能，將所有資料源連結在一起。

Tips for Chart

- 清除圖表 --清除圖表內舊的資料點。對著圖表的圖示按右鍵跳出『彈出選單』>Data Operations> Clear Chart
- 捲軸 -- 圖表的捲軸功能，可以隱藏或顯示出來，你可以藉由捲軸的功能來觀看圖表上舊的資料點。
：彈出選單>Properties>Appearance>(勾選)Show x scroll bar
- 圖表顯示長度 -- 一個圖表所暫存的資料點預設為 1024 點，若要暫存更長或更短的點數時，可更改 Chart History Length，但更改暫存點數並非影響圖表螢幕所顯示點數的多寡。暫存的資料點數上限為 100,000 點。
：彈出選單>Chart History Length
- 堆積與覆蓋繪圖 -- 多點繪製圖表時，可選擇多個資料點繪製在同一個 Y 軸上，稱為 overlaid 繪製;或是每個繪製點有其個別的 Y 軸使尺度，稱為 stacked 繪製。
：彈出選單>Overlay Plots / Stack Plots

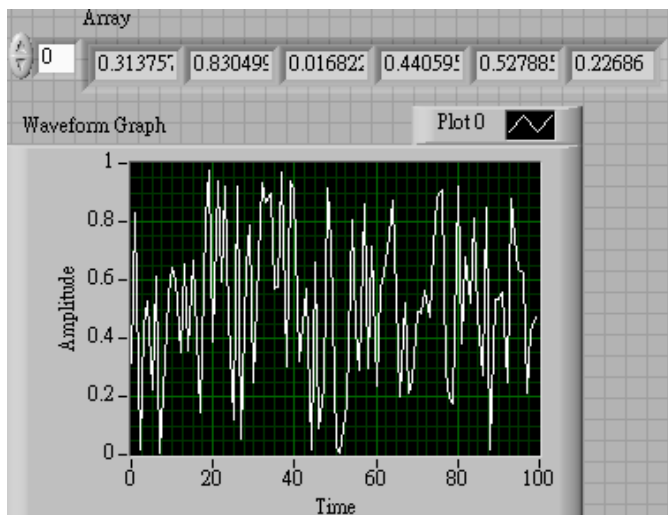


圖形(Graph)

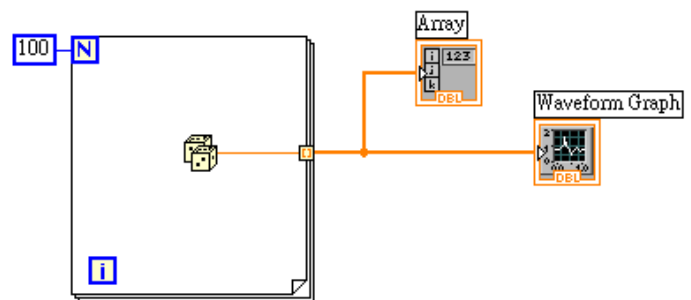
不同於圖表能每次一一繪製資料點的功能，圖形繪製資料點的方式是預先產生完成整比資料列，再顯示曲線，所以圖形每次繪製曲線時並無法與先前的資料並列。LabVIEW 提供多種相當好用的圖形繪製：波形圖形(Waveform graph)、XY 圖形(XY graph)、強度圖形(Intensity Graph)…等等。以下將介紹波形圖形與 XY 圖形。

(1) 波形圖形(Waveform Graph)

◀ 實做練習九▶ 單點繪製的波形圖形



人機介面面版

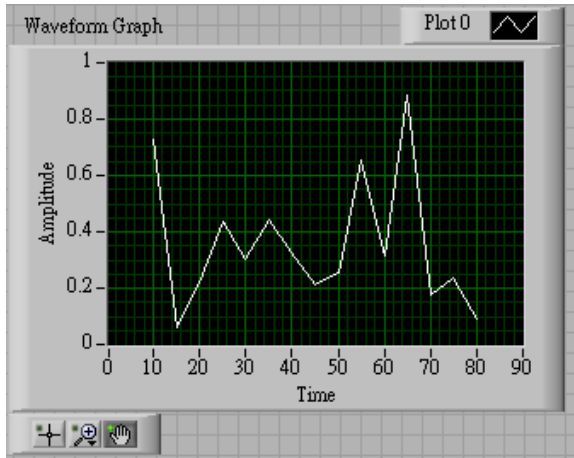


程式方塊編輯區

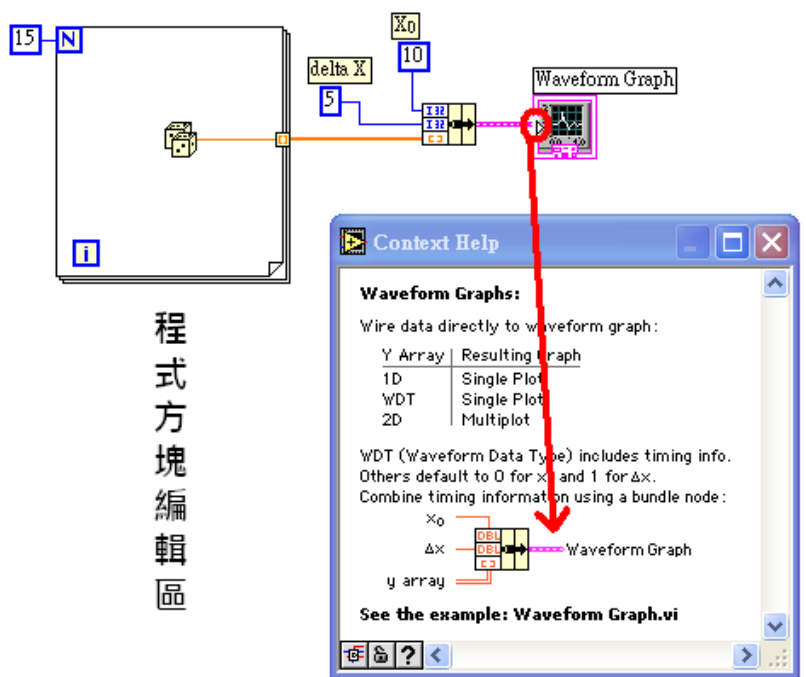
◆ 程式說明：

利用 For Loop 及亂數產生資料，在人機介面上方處可以看到資料的型態是一維的數值陣列，我們直接將資料輸入波形圖形的接點上（資料列即形成圖形之 Y 軸數值）。這種方法，預設值是假設 X 軸的初始值為零($X_0 = 0$)，且 X 軸的差值（集每次 X 軸上的增加量）為 1($\Delta X = 1$)。

若你想改變圖形上的時間尺度(X 軸)，需要將 X_0 、 ΔX 與資料列一起聚集成叢集，並將此叢集接線至圖形的接點上。你可以仿造下面程式方塊編輯區的做法。

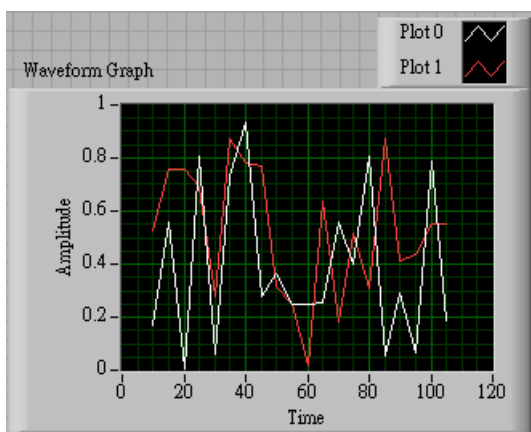


人機介面面版

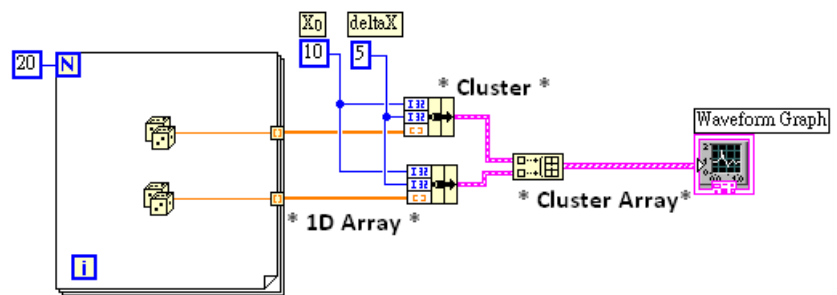


程式方塊編輯區

◀ 實做練習十 ▶ 多點繪製的波形圖形



人機介面面版



程式方塊編輯區

◆ 程式說明：

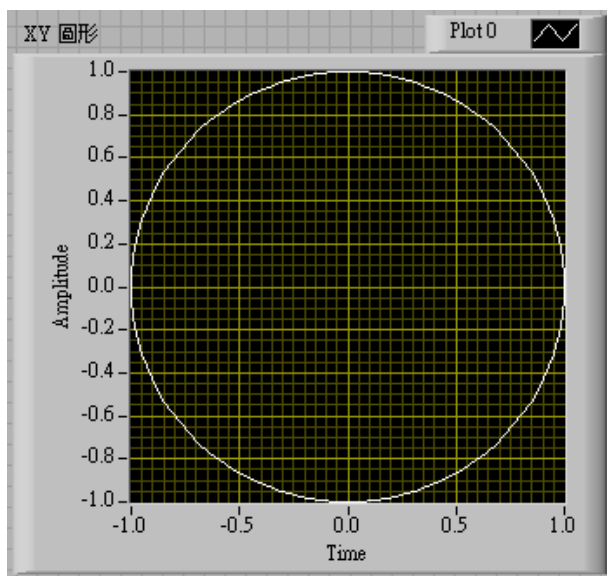
若要在波形圖形上繪製多於一個以上的資料列，只要在單點繪製圖形上將資料列增加維度即可。在這個範例中，我們做的就是兩個 1D 的陣列，建造為一個 2D 陣列。只不過這兩個陣列，我們為了自訂 X_0 與 ΔX 值，所以將它們各自轉成 Cluster。所以其實，我們是建造一個 2D 的 Cluster Array，再將其輸入波形圖形的程式方塊接點。

❖ 波形圖形是將 2D 陣列分離成兩個列向量，然後分別繪製成兩條曲線。它的預設是繪製列向量，如果你的資料是以行向量的方式整理，你必須在繪製圖形之前先將資料陣列做轉置(Transpose)！（轉置是將列向量的數值與行向量的數值相互置換，例如：轉置一個 2×10 的陣列，就會得到一個 10×2 的陣列）。關於此點，在後面儲存檔案時，也會應用得到！

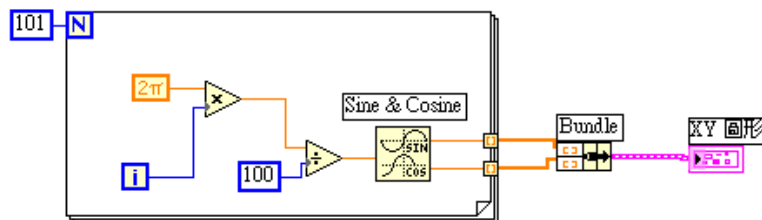
(2) XY 圖形(XY Graph)

XY 圖形上的每一個點位置是由個別的 XY 陣列所決定。

◀ 實做練習十一 ▶ 單點繪製的 XY 圖形 -- 描繪出一個圓形圖



人機介面面板



程式方塊編輯區

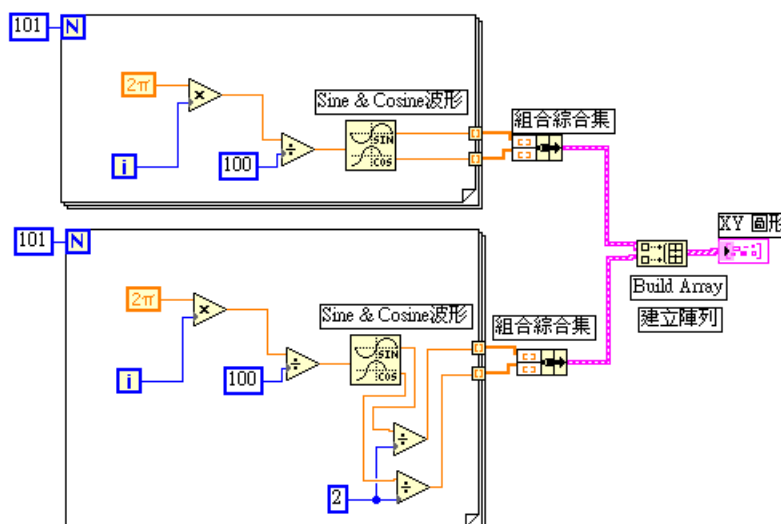
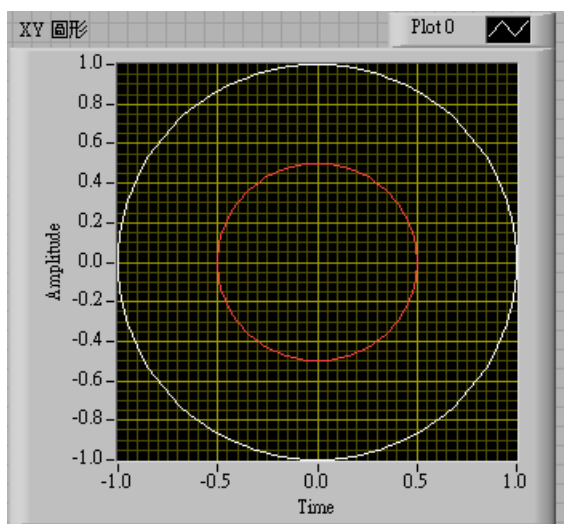


(正弦與餘弦函數) : Functions>Mathematics>Elementary&Special Functions>Trigonometric> Sine&Cosine

◇ 程式說明 :

當 $i=0$ 時, $\sin 0=0$ 、 $\cos 0=1$; $i=25$ 時, $\sin 0.5\pi=1$ 、 $\cos 0.5\pi=0$; $i=50$ 時, $\sin \pi=0$ 、 $\cos \pi=-1$; $i=75$ 時, $\sin 1.5\pi=-1$ 、 $\cos 1.5\pi=0$ 。在此範例中, 以正弦函數的值為 X 軸、餘弦函數的值為 Y 軸, 以上四個 i 值分別在 XY 圖形中產生 : (0,1)、(1,0)、(0,-1)、(-1,0) 四個點。以此類推, 共有 101 個點可繪出如上圖所示的圓圈。

◀ 實做練習十二 ▶ 多點繪製的 XY 圖形 -- 在 XY 圖形上顯示出兩個圓



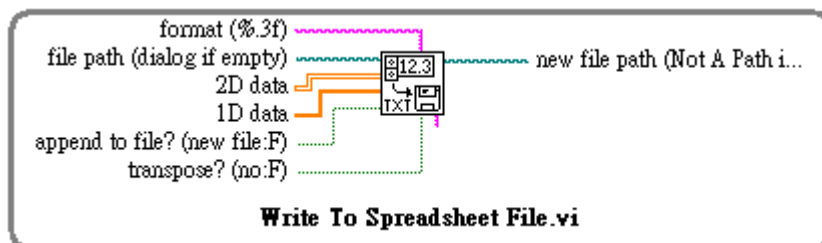
◇ 程式說明 :

此例與練習十一很相似，只是將其中一個波形除以二，再以建立陣列合併就可以繪出兩個同心圓了！

檔案 I/O 函數

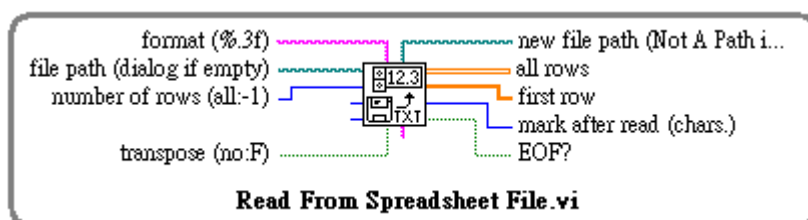
檔案 I/O 的操作，可在磁碟檔案中取出資訊或存入資訊。在 LabVIEW 中有許多處理檔案存取的函數，可由 **Function > File I/O** 中選取。在 **File I/O** 子面板中，第一行是屬於高階的函數，只需要一些簡單的參數設定即可，但功能亦較簡單。第二行是屬於較低階的函數，功能較多，所需的設定亦較複雜。檔案函數需要一個檔案路徑的輸入，此路徑名稱看起來像是一個字串；若你並未將檔案路徑接線，檔案函數將會彈出一個對話框，要求你選取或輸入一個檔案名稱。當被呼叫的時候，檔案函數會執行三個步驟，第一：開啓舊檔或建立一個新的檔案。第二：讀取或寫入檔案資料。第三：關閉檔案。在此，我們僅介紹高階檔案存取函數中的” Write to spreadsheet. vi” 及” Read from spreadsheet. vi” 兩種較簡易的存與取函數。這兩種 VI 建立/讀取的檔案僅是一般的文字檔，你可以用一般可開啓文字檔的程式來瞧瞧資料內容長得是什麼樣。

● Write To Spreadsheet File



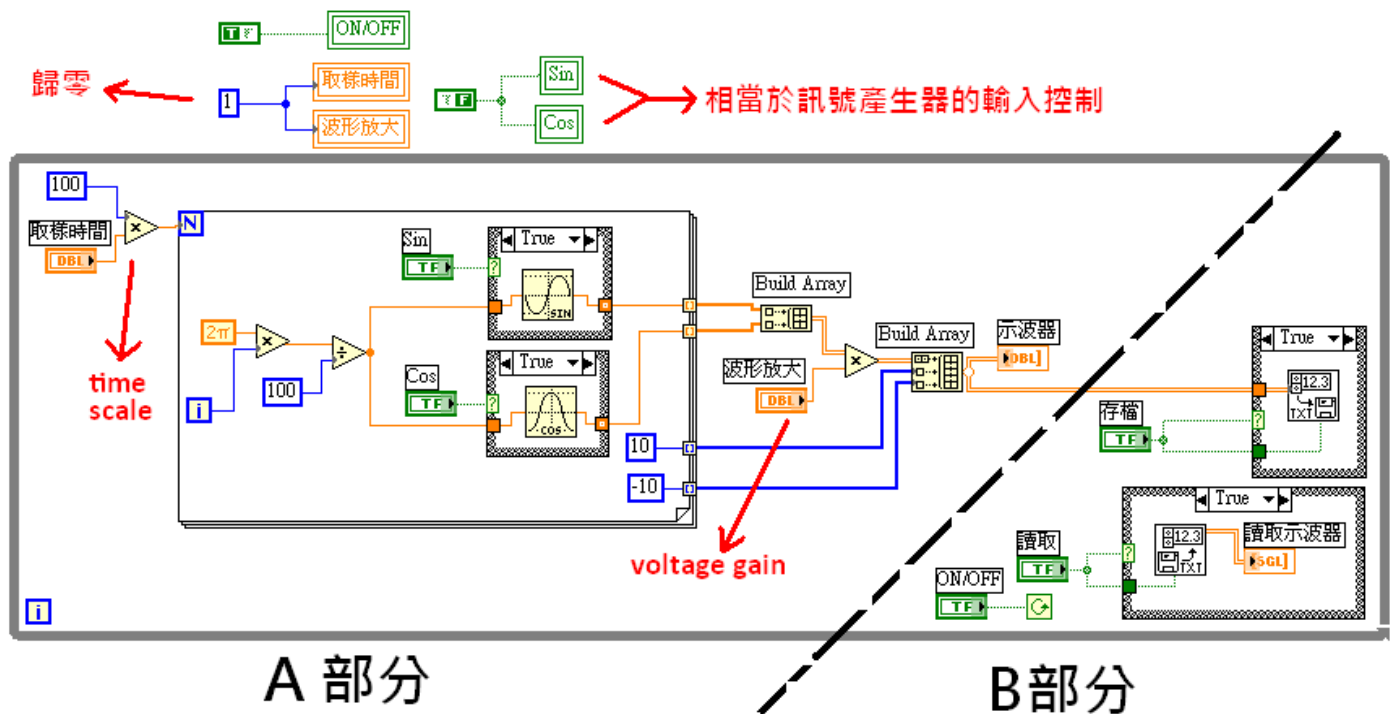
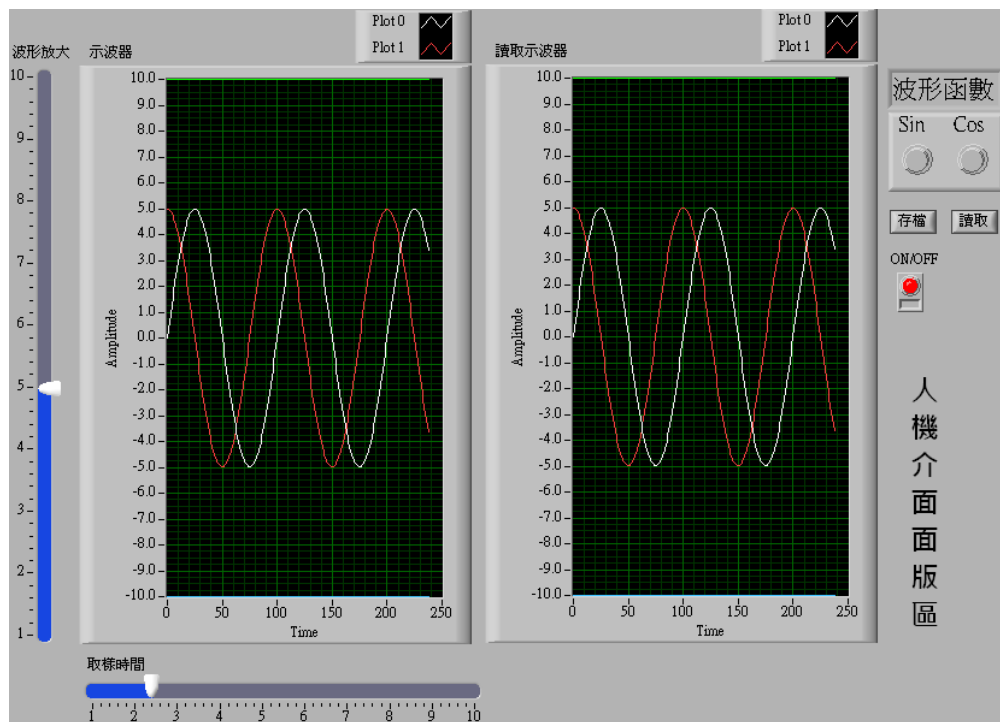
Write To Spreadsheet File 轉換 2D 或 1D 陣列的數字成為文字字串，然後將此字串列儲存於一新的表格格式文字檔案中或增添於一舊檔中，利用一般編輯文書軟體(Excel, WordPad, Origin ...)，就可以讀取檔案內容。我們可在寫入檔案前設定是否轉置此資料陣列，獲釋選擇覆蓋或增添資料。在 LabVIEW 中，資料陣列是以列(row)定義，而一般我們文書編輯軟體(ex: Excel)讀取時，則通常定義為行(column)的讀取方式。所以應該注意你在運用時是否需要做 2D 陣列的轉置。

● Read From Spreadsheet File



Read From Spreadsheet File 可由表格式文字檔內經由 offset 的設定，讀取特定的列數(row)，並轉換成數值的二維數列。

◀ 實做練習十三 ▶ 存檔與讀檔的應用



◆ 程式說明：

程式開始執行時，選擇欲產生函數波形即可顯示在示波圖形上。選擇波形放大和取樣時間將可調整波形。若按下存檔按鈕(此按鈕按下表示 True)即可將波形資料儲存於選擇之檔案內。若按下讀取按鈕(按下表示 True)，且你選擇的讀取檔案是剛剛你存檔時的檔案，那麼你將會在讀取示波器上得到將相同的波形。程式方塊圖分成 A 部分與 B 部分，A 部分是模擬示波器的顯示，B 部分的程式內容則是為了達成存檔與讀檔的功能。

你可以試試看，直接將剛剛儲存的檔案，匯入 Excel 或 Origin，將資料繪製成圖，比對看看是否和人機介面中的讀取示波器顯示的一樣。你也可以試著不輸入布林函數到程式方塊編輯區內存檔的

Transpose(轉置)，然後再將檔案匯入 Excel 或 Origin，看看有何差別。

資料與例題來源：

1. LabVIEW 程式設計入門 ， 徐瑞隆 編著 ， 新文京出版
2. LabVIEW 基礎篇 ， 蕭子健、儲昭偉、王智昱 編譯 ， 高立圖書出版
3. LabVIEW 入門手冊 ， 林踐 編著 凡異出版
4. <http://www.phys.ncku.edu.tw/ultracold/paper/aroma.pdf>
5. http://intelligent.mech.yzu.edu.tw/course/experiment/experiment_handout/MEexperiment_2006.doc
6. <http://www.gauss.com.tw/logic/ch1/1-2.htm>
7. <http://www.cinelformacao.com/labview/files/usb6008/usb6008intro.htm>
8. <http://www.pt.ntu.edu.tw/hmchai/PTcomputer/hLabVIEW/LabVIEWindex.htm>