圖控式語言 — LabVIEW在運動控制之設計發展

■工研院機械所　　張雅玲

前言：

1986 年 NI 推出 LabVIEW 之後，便開啓了虛擬儀控之大門。虛擬儀控讓使用者能利用標準的電腦來建構自己的儀控系統，而在自動化測試/控制的領域之中，圖控式軟體(LabVIEW)因爲具備高彈性且高效率的特性，並且有漂亮的人機介面，相信在未來的生產環境之中勢必會成爲應用上的趨勢。

本文所要介紹的是利用 NI 所開發的圖控式軟體 LabVIEW 和工研院控制器發展部所研發的 PMC32-6000 PCI-Bus 的 DSP 運動控制卡，在 PC-Based 的基礎之下，介紹以人機介面來控制禾宇的 XYZ 三軸平台的實例而作之說明。

壹、LabVIEW 之簡介

LabVIEW(**Lab**oratory **V**irtual **I**nstrument **E**ngineering **W**orkbench)是由 National Instrument 所發展的應用軟體。爲一套專爲資料擷取、儀器控制與資料分析而設計的革命性圖控程式語言，它提供了一個嶄新的程式設計方法，只需將所謂的虛擬儀表(Virtual Instrument)物件以流程圖的方式加以連接組合，便可完成所需要的系統；更由於它的簡單、易學因而大大地縮短了研發時間與經費及增加了生產力。而此篇的內容主要是針對運動控制方面的設計而做一系列的介紹，從基礎的 LabVIEW 軟體的介紹進而擴展至應用層面的說明。以下開始首先針對 LabVIEW 這一套圖控式的語言做一介紹。

LabVIEW 所撰寫的程式稱之爲虛擬儀表，其主要可分爲三個部分：前置面板(Front Panel)、程式方塊流程圖(Block Diagram)、圖像和連接器(Icon/Connection)。LabVIEW 圖控式的程式語言，取代了以往的文字模式開發軟體，且使用了所謂資料流(dataflow)的編輯方式，所以使其程式的執行更是清晰。以下我們開始針對這三部份分別做說明。

1、 前置面板(Front Panel)

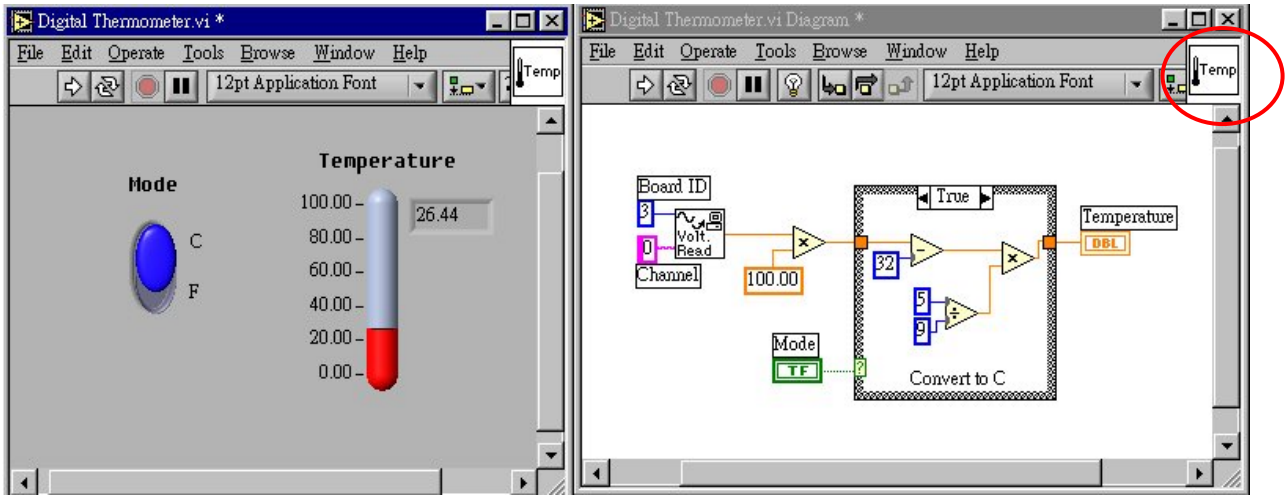
所謂的前置面板爲使用者所操作的人機介面面板，圖一是利用 LabVIEW 所撰寫的華氏和攝氏轉換範例程式。圖一左邊爲前置面板，可以看做是一真正的儀表，可輸入控制的地方稱爲控制器(Controls)而資料輸出地方爲顯示器(Indicators)，LabVIEW 提供了各式種類的控制器和顯示器，像是按鈕、圖表或是一些較容易瞭解的控制元件。利用控制元和顯示元來建立前置面板，當你在前置面板建立了一個元件後，便會在程式方塊流程圖中出現所謂的 Terminal。所以前置面板相當於一般儀測系統上的輸入數值的設定與輸出儀表板的組成，不僅可以模擬開關、設定起始值與臨界值(選用 Control)，同時輸出結果亦能顯示在此面板上(選用 Indicator)。

2、 程式方塊流程圖

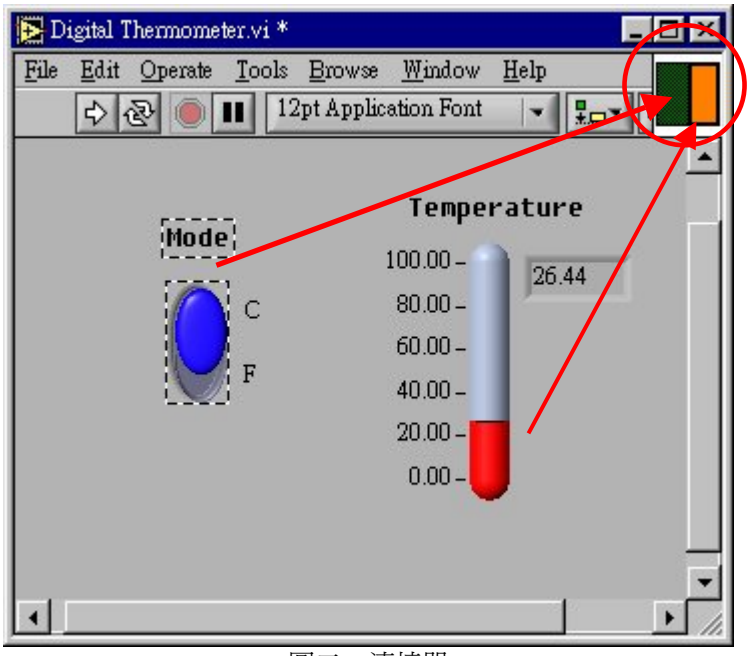
圖一右邊所顯示爲程式流程圖，每一個前置面板都附有程式方塊流程圖，就是 VI 的程式，我們用拉取元件的方式來建立程式方塊流程圖而組成部分表示程序的節點(Node)，如 For Loop 或是圖一所顯示的 Case。在程式方塊流程圖中，我們利用線條將所有的組成部分連接起來，使得資料經由連線來傳輸。圖一右爲程式方塊流程圖的範例了，利用副程式(Demo Voltage Read.vi)來讀取電壓值，再利用 Mode 的選擇鈕來選擇溫度的單位，當將 Mode 選擇爲 C 時，則爲攝氏，若爲 F 時，則爲華氏。此時的 Case 便會依照使用者的選擇而執行程式。所以程式方塊圖相當於傳統程式語言的原始碼，由於 LabVIEW 提供的是繪圖形的設計環境，因此程式設計的流程即爲自然，其內容也非常容易瞭解且除錯簡單，因此，它的功能如同傳統硬體儀測系統內的線路圖。

3、 圖像和連接器

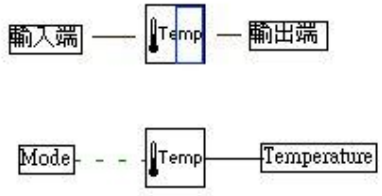
當我們將一個 VI 完成後，製作成圖像(如圖一右上角或副程式 Demo Voltage Read.vi 的圖像所示)和連接器(如圖二右上角所示)，便可在其它的 VI 程式流程圖中使用，如 Demo Voltage Read。製作完圖像和連接器就必須要決定輸入和輸出，這就如副程式中的變數，也相當於前置面板的控制器和顯示器，下面圖三就是說明如何將 Digital Thermometer 製作成圖像和連接器。



圖一、LabVIEW 範例



圖二、連接器



圖三、圖像、連接器示意圖

貳、LabVIEW 的工作環境(視窗、選單和工具)介紹

LabVIEW 軟體系統是經由一些相關的檔案環境所組成的，檔案環境如下所示。

1、 Window 95

LabVIEW 支援許多不同的作業系統，在視窗環境下，LabVIEW 程式集內有許多應用程式，按下 LabVIEW 的 icon 即可啓動 LabVIEW。

2、 其它檔案和目錄

LabVIEW 是使用檔案和目錄來建立 Vis 所需的資訊，這些檔案和目錄說明如下：

(1) vi.lib 目錄

包括 Vis 的函式庫。vi.lib 目錄一定要和 LabVIEW 使用相同的目錄，不可任意更改目錄名稱。因爲 LabVIEW 會首先尋找此目錄，如果改變此目錄名稱，我們將會有許多控制和函式庫功能無法使用。

(2) example 目錄

提供了許多 LaVIEW 程式功能的範例。使用者可以從此目錄中了解 LabVIEW 的使用方法。

(3) cintools 目錄

包含連接外部 C 語言應用程式至 LabVIEW 的檔案。

(4) menus 目錄

提供全部的選單資訊。

(5) help 目錄

包含所有 LabVIEW 相關的 Help 檔案，Vis 和 VI 函式庫被放置於此目錄中。

(6) BASSCLASS.LLB 函式庫

此檔案功能包含我們將使用在 LabVIEW 路徑下的路徑下的 VIs 函式庫。

(7) user.lib 目錄

可以將使用者所製作的函式庫放置於此目錄中，以便開發程式中可以直接在 Function 選單中選取。

(8) instr.lib 目錄

您可以將想在您可以將想在 Function 選單中出現的儀器動作程式，放在此。

3、 工具列

前置面板和程式方塊流程圖視窗的上方均包含有工具列可用來控制 VI 的按鈕和狀態顯示，兩視窗的工具列並非同時有效。下圖四爲前置面板工具列，圖五爲程式方塊流程圖的工具列，皆位於面板之最上方。

圖四、前置面板工具列

圖五、流程方塊圖工具列

4、 下拉式功能表

在 LabVIEW 視窗最上端的選單工具列有幾向下拉式功能表，下拉式功能表包含需多常用的應用選項，像是 Open 、Save….如圖六所示。

圖六、下拉式功能表

5、 功能板(Palettes)

LabVIEW 提供圖形化、可移動的功能板來幫助 Vis 的產生和運作，此三種功能板分別爲 Tools、Controls 和 Functions。Tools 中的工具可以用來建立、修正和除錯 VIs，如圖七所示。Controls 可以來增加前置面板的控制器和顯示器，而 Controls 功能板只能在前置面板的視窗上取得，如圖八所示。利用來建立程式方塊流程圖，每一個選項皆有利於圖像的選擇，如圖九所示。

圖七、Tool

圖八、Control

圖九、Function

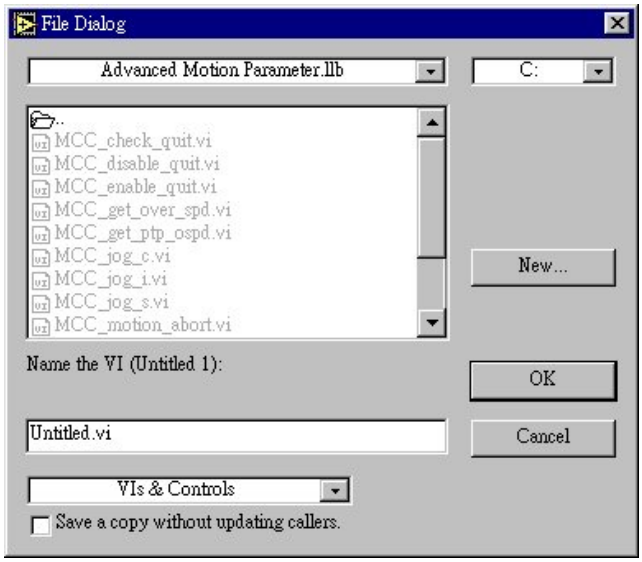
6、 VI 函式庫

我們可以從 VI 函式庫的特別檔案讀取或儲存 Vis(通常副檔名爲.LLB)，下圖 Advanced Motion Parameter.LLB 函式庫就是一個範例，使用 VI 函式庫的好處如下：

(1) 在 VI 函式庫中，可使用高達 225 個自原來設定 VI 的檔名。

(2) VI 函式庫可先壓縮 VIs 在存入硬碟空間在存入硬碟空間(在讀取檔案時可解壓縮)。

(3) 因爲有多種的 VIs 儲存在一個檔案中，比較容易在電腦與電腦之間傳輸。



圖十、VI 函式庫

7、 移動 VI 跨越平台

我們能將 Vis 從一平台轉移到另一平台(譬如從 LabVIEW for Macintosh 至 LabVIEW for Windows)，LabVIEW 能自動轉換及重新編譯新平台上的 Vis，我們能轉移在函式庫中的 Vis，並且能在函式庫中使用長檔名，其檔名可程到新平台所限制的檔名長度。

參、爲何選擇 LabVIEW 作爲發展軟體

一、LabVIEW 之特色

LabVIEW 是一套專爲資料擷取、儀器控制與資料分析而設計的革命性圖控程式語言，它提供了一個嶄新程式設計方法，只需將所謂的虛擬儀表(VIs)物件以流程圖的方式加以連接組合，便可完成所需的系統，更由於它的製作簡單、易學因而大大地縮短了研發時間與經費更可增加了生產力，也可以建立一個 stand-alone 的執行程式或是 shared libraries，如 DLL，這是因爲 LabVIEW 是一個 32-bit 的編譯器。

二、整合 LabVIEW 到其他的應用軟體

爲了確保 LabVIEW 程式碼可以輕易地和不同的程式語言及企業工具整合，在 LabVIEW 6i 版本中可產生 32 位元的動態連結資料庫(DLL)或是任何 VI 的共享資料庫，有了這些工具就可以輕易的整合這些動態連結資料庫及共享資料庫到其他的程式編譯環境，像是 Microsoft Visual Basic、Microsoft Visual C++、或 Measurement Studio 之中。

在 LabVIEW 6i 的版本中，編輯器(Application Builder)產生共享資料庫(如 DLLs)附加在可執行程式及安裝器(Installer)中，可以較以往更快捷更有效率的產生可執行程式或共享資料庫(及他們的安裝器)，因爲應用程式在建立的過程中已經最佳化，編輯器所產生的安裝器可以提供更爲簡便的方法，展開量測與自動化監控的所有軟體組件。

三、LabVIEW 圖控式程式語言的特性和傳統儀測系統之比較

虛擬儀表和傳統儀測系統的比較如下表一所示。

系統	傳統儀測系統	虛擬儀表
1	儀測內容由製造者定義	儀測內容由使用者定義
2	具有特定功能需獨立使用且具有限的連接性	應用導向的系統，方便與網路、週邊及其他應用連接
3	以硬體線路爲主	以軟體程式爲主
4	價格通常昂貴	價格低廉，可再用性高
5	再開發與維修費用高	因採用軟體"線路"，故大附件低發展與維修費用
6	封閉式裝置且有固定功能	開放式裝置，功能可調適
7	功能提昇，演進緩慢	快速性能演進

表一、虛擬儀表和傳統儀測之比較

四、LabVIEW 之好處

利用 LabVIEW 來發展控制系統的好處可以分爲以下八種，接下來針對這八種一一做說明。

1、建立屬於自己的解決方案

利用 LabVIEW，您可以建立 VI，而不必在編寫煩人的程式。也可以非常快速地製作出人機介面，讓軟體系統具有互動式控制的功能。如果要使用某種功能，只需很直覺的將各種區塊圖形加以組合即可。

2、製作人機介面

只要從控制選單中選取所需之物件，便可以在 VI 的人機介面中顯示各種系統控制功能以及資料，包括數值顯示、各種量測數值、量測計、恆溫裝置、儲存槽、發光二極體、圖表、圖形等等。而完成 VI 之後便可以執行，並利用控制面板上的各種功能，例如按下開關、移動滑板，將圖形放大，或只利用鍵盤來輸入數值等，來操控系統。

3、建構圖型式的方塊圖程式

在編寫虛擬儀控的程式時，只需要將區塊圖形組合起來就可以，無須像以往編寫傳統的程式一般，

擔心各種語法的問題。您可以在功能選單中選取各種物件(圖示)，然後用線將其連接在一起，就可以將資料從某各區塊中傳到另一個區塊中。可以使用的區塊功能包括簡單的運算功能，常用的進階擷取和分析功能，以及網路和檔案輸出/輸入之控制等，應有盡有。

4、資料流向程式編寫

LabVIEW 所使用的是一個具有專利且被稱爲 G 的資料流向程式編寫模組，讓您跳脫以文字爲主之語言直線型架構方式。由於 LabVIEW 中的執行命令是根據各區塊中的資料流動，而不是根據文字行號順序來決定，因此您可以製作一些可同時執行的區塊。由於 LabVIEW 是一個多工以及多緒的系統，因此可以執行多個執行緒和 VI。

5、模組化以及層級式的架構

LabVIEW 採用了模組化的設計，因此每個 VI 都可以單獨執行，或只當作是其他 VI 的一部份來使用。您可以將自己的 VI 製作成一個圖示，並且設計一個 VI 和子 VI 的層級架構，以便進行修改，交換以及配合其他的 VI 而使用，滿足您多變的應用需求。

6、圖控的編譯器

在許多的應用中，執行速度是非常重要的。而在各種圖控程式編寫系統的編譯器中，LabVIEW 的編譯器是唯一可產生最佳化的編碼，使其行速度可以和編譯器過的 C 語言程式相抗衡的系統。利用內建的概括器(Profiler)，您可以對編碼中執行速度非常重要的部分進行分系和最佳化處理，這樣就可以提高圖控程式編寫的速度，而不會影響程式執行的速度。

7、連結性

LabVIEW 具有各種的 VI 程式庫，可以和其他的應用系統連結。您可以在 LabVIEW 中呼叫任何一種動態連結程式庫(DLL)或只共享的程式庫；也可以在其他的語言中呼叫 LabVIEW 程式碼，利用 LabVIEW ActiveX 包容器(container)，您可以嵌入 ActiveX 的控制功能以及文件，或只將編寫在 LabVIEW 的程式中。

8、從其他程式語言來呼叫程式碼

在 Windows 的環境下有兩個方式來呼叫外部程式碼，一個是使用 Call Library Function 來呼叫動態連結程式庫(DLL)，另一個適用程式碼介面節點(CIN)來直接呼叫外部語言所建立的可執行程式。這兩個函數從 Advanced 面板上都可以找的到。在 Window95/98/NT 的環境下，可以呼叫三十二位元的 DLL。

肆、虛擬儀表 VI 的產生

一、如何產生 VI

之前我們已經談過了一些關於 LabVIEW 環境的基本知識了，現在所要說明的是如何真正的製作一個屬於自己的 VI。

1、人機介面

在開始程式時，需要將控制元與顯示元放在人機介面上，以定義使用者輸入與程式的輸出。所以這時只要將所需的元件到控制面板上選取並利用標籤來加以註記名稱。並可利用工具列的 Font 來加以改變文字的字體、格式、大小和顏色。

2、程式方塊流程圖

缺乏程式支援的使用者介面是不會好到哪裡去的。您可以利用程式方塊圖中放上函數，子 VI 即結構來建立真正的程式。利用選取函數面板來建構自己的程式區，便可完成一個虛擬儀表了。

3、圖像/連接器

當一 VI 要作爲 SubVI 時，便需有一圖像代表。SubVI 需有聯結器來傳遞資料。每個虛擬儀表在其視窗的右上角均有顯示內定的圖像。而連結器是附屬虛擬儀表與外界連結的介面。若要定義連結器方格，先在圖像上點一下後，如何將一設計結果(虛擬儀表)定義成一附屬虛擬儀表，可將一設計好的虛擬儀表(VI)當作附屬虛擬儀表(SubVI)以便在其他場所中使用。選擇”Function”功能表裡”Select a VI…”並在目錄中選用所要的附屬虛擬儀表。

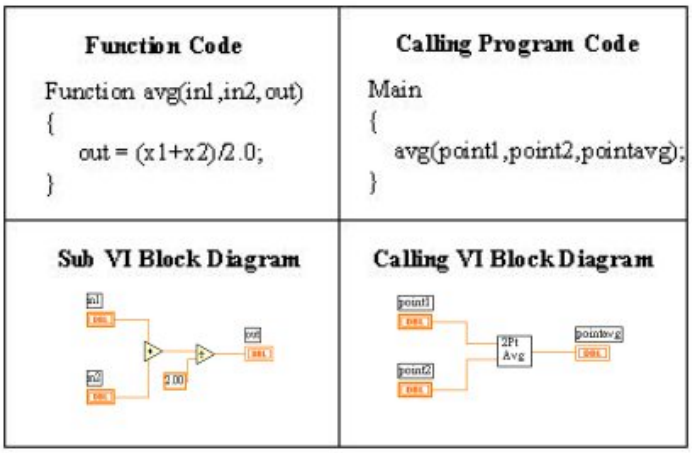
二、編輯技巧

當了解了如何建立一個虛擬儀表之後，就要關心如何設計程式的技巧了，建議可以依照下列的四個基本方針：

(1) 採用由上而下設計方式來架構你的藍圖。
(2) 製作一張需求清單。
(3) 以模組化程式的設計方式來設計 VI。
(4) 撰寫程式，儘量將各模組設計成 SubVI。

1、 建立一個附屬的虛擬儀表(SubVI)

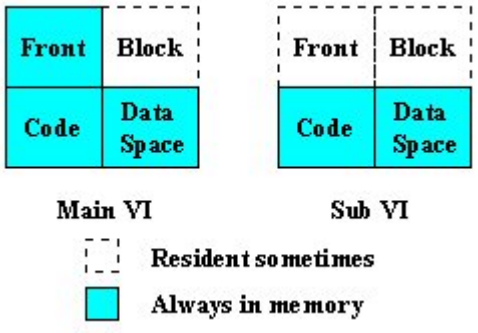
附屬虛擬儀表類似於函式或是子程式。以下圖十一的程式碼和程式方塊流程來說明附屬虛擬儀表和子程式的相似處。如圖十一所示，當將函式 avg()編輯爲一個子 VI，而利用圖像/連接器將其編輯成如右下角所見的 2PtAvg 方塊，所以當我們在另一程式中需要用到此函式時，只需要將這 icon 連結進入，並依照所規定的參數來輸入，便可執行此函式。



圖十一、SubVI 與子程式示意圖

2、 SubVI 可以降低記憶體的使用

以下說明爲何建議開發者能利用 SubVI 來編輯應用程式，其主要的原因可以從下圖十二所示。首先說明下圖各代表的意思。Front：爲前置面板。Block：爲程式方塊圖。Code：圖形執行碼；爲程式流程圖 Compile 後的機械碼。而 Data Space：爲 Control/Indication 的數值、預設值、程式方塊圖中的常數值和動態定義的資料。由圖十二可以了解到當開啓一個 MainVI 的時候，其 Front Panel、Code、和 Data Space 爲常態性的佔據記憶體的位置，而當將一個程式編輯成 SubVI 的時候，這時可以看見記憶體只被 Code 和 Data Space 所佔據。所以爲了記憶體的降低，建議儘量將其編成 SubVI 形式。



圖十二、VI 記憶體

3、 記憶體、效率的管理

記憶體的管理是在編寫 LabVIEW 程式的一大課題，當你有一大型的陣列或是對時間要求較高的程式時，對於記憶體的管理就不能不仔細了解。以下是一些對於記憶體管理上必須注意的幾項要求。

(1) 使用正確的資料型態。

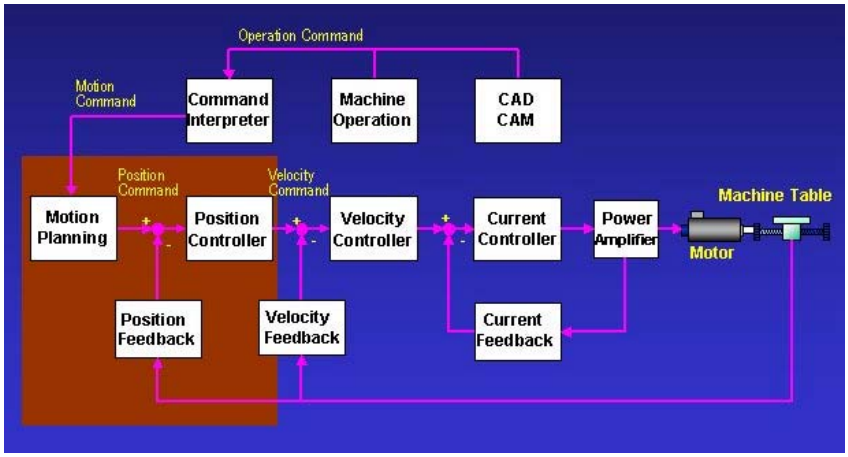
(2) 廣域變數將會使用大量的記憶體。

(3) 避免強制不確定的資料型態(在接點上的灰色點)。強制的意思是連接不同型態的資料，因爲對於 LabVIEW 而言接受不同的資料型態則必須對資料進行複製的動作，所以會加重了記憶體的使用。

伍、運動控制模組

運動控制軟體模組爲一提供使用者開發上層應用軟體環境，使用者可搭配 ASIC-Based 運動控制模組與 DSP-Based 運動控制模組進行即時多工函式庫聯結，並可在圖控軟體或應用程式下直接呼叫相關函式，以達到快速開發整合系統之目的。本範例程式是利用工研院控發部所開發的 DSP 運動控制卡(PMC32-6000)來撰寫應用程式。

而運動控制方塊圖如圖十三所示。

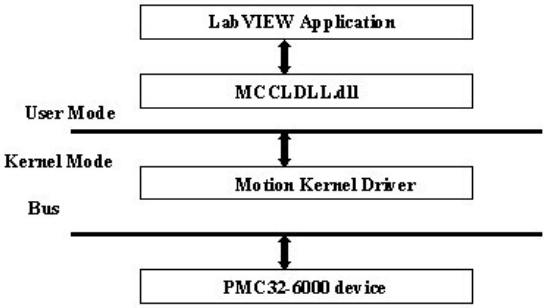


圖十三、Control System Block Diagram

陸、LabVIEW 製作運動控制程式之方法

以往 LabVIEW 都被當作 G 程式編寫發展環境而使用於製程控制以及工廠自動化方面。針對基本的製程監視以及控制等應用，LabVIEW 也結合了像 Automation Symbols 工具箱、PID 控制工具箱，以及 PLC 驅動程式等加值型軟體，提供最佳的解決方案。

在發展一套完整的系統，可以使用 LabVIEW 控制您的系統，並利用互動式的圖形人機介面來展現成果。LabVIEW 適用於多種平台，可以搭配 Window NT/98/3.1、Mac OS、Sun、HP-UX 以及 Concurrent PowerMAX 來使用，呼叫所謂動態連結檔的形式來製作運動控制程式，圖十四、十五爲應用程式和 PMC32-6000 板子之間溝通的示意圖。



圖十四、PMC32-6000 與 LabVIEW 之溝通

柒、範例介紹

接下來要介紹的範例是利用雕刻機來開發運動控制系統的介紹。處理開發者需要的控制系統是一個功能強大、具有彈性的且可以擴展的應用程式。選用 LabVIEW 是由於它的快速開發環境，以及整合功能。接下來便開始針對此範例程式的開發環境做簡單的介紹說明。

一、開發環境介紹

1、開發軟體

我們利用的開發軟體爲 LabVIEW5.0，程式環境爲 Windows 95 作業系統。

2、硬體

(1) 運動控制卡(PMC 32-6000，PCI-Bus，6 軸定位控制及 256 點遠端 IO 控制卡)

由工研院機械所控發部所開發的 DSP 6 軸控制卡，這爲 DSP-Based 運動及輸出入整合控制模組。其系統效益如下：以 DSP 爲核心之智慧型運動控制模組，可簡化工業控制系統之運動控制設計時程，縮短 time-to-market。與 PC-Based 工業控制器搭配使用時，可節省可節省 IPC 模組之處理時間，使得 IPC 具有更多時間來處理人性化人機介面與複雜且彈性之加工需求。而可應用的範圍有：PC-Based 工業控制器、工具機控制、產業機械控制、半導體設備定位控制等。

(2) 轉接板(EPCIO-601-2)

爲運動控制卡之 DDA(Digital Differential Analyzer)、Encoder、LIO 及 D/A 等信號輸出/輸入之轉接板，搭配 Panasonic MINAS AC 伺服器*XX 系列使用之專用轉接板，並且保留彈性亦可與其它形式之伺服驅動器(步進馬達驅動器)搭配使用。

(3) 雕刻機

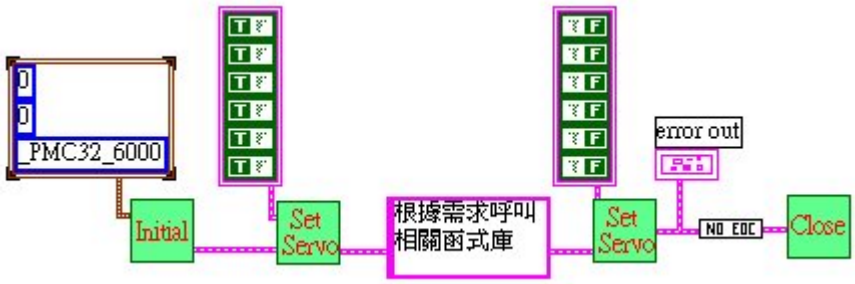
雕刻機是由禾宇精密公司所製造的 XYZ 萬用移動平台，規格如表二所示。

加工範圍	340mm230mm
移動距離	Z 軸 50mm(Max)
機械構造	X/Y/Z 軸
滑動構造	X/Y 軸日本 THK 線性軸承　　Z 軸日本 THK 交叉型滑軌
驅動方式	X/Y 軸伺服馬達　　Z 軸步進馬達
移動速度	50mm/sec 以上
極限控制	X/Y 軸雙極限開關　　Z 軸單極限開關
機械規格	W=420mm　D=570mm　H=430mm
機械重量	25KG
使用電力	AC110/220

表二

3、運動函式庫

使用的運動控制函式庫爲 MCCL3.0(Motion Control C Library)。爲即時運動函式庫，將其包裝成 DLL 的形式，提供給使用者來建構應用程式，由於它是一個 DLL，使用者可選擇一套熟悉的發展工具來撰寫應用系統的人機介面。而我們所使用的應用軟體爲 LabVIEW5.0。而使用 PMC32-MCCL 運動函式庫的流程如下圖十五所示，在呼叫函式庫中任何函式之前，必須先執行啓動運動控制函式庫的功能函式，透過 MCC_initial()傳入的參數來設定使用者目前使用的卡別，若選擇的爲 ISA-Bus 板子，則需要傳入該卡使用的 IO 位址和 dual port RAM mapping 的實際位址，因爲我們所使用的是 PCI 介面，所以在插入硬體卡時作業系統會自動配置好。當系統配置好之後便可以呼叫 MCC_set_servo_status()函式方式來設定各軸 Servo 狀態，使用者亦可藉由硬體開關方式來設定 Servo 的狀態，當使用者以硬體方式設定 Servo 狀態時則軟體設定將會無效。當這些設定完成後，便可根據需求呼叫相關函式。當完成控制程式之後，需呼叫 MCC_close()來關閉運動控制函式庫的功能。呼叫此函式後 PC 會釋放與 PMC32-6000 mapping 之記憶空間，因此在應用系統結束前要呼叫此函式來釋放 PC 資源。



圖十五、PMC-MCCL 使用流程圖

而範例程式中所會運用的函式庫如下所示：

(1) 指令功能

■絕對座標值設定選取	MCC_set_abs()
■單步 JOG 功能	MCC_jog_s()
■直線進給速度設定	MCC_set_spd()
■直線運動設定	MCC_line()
■連續路徑取消	MCC_disable_quit()
■定位確認取消	MCC_disable_in_pos()

(2) 一般功能

■原點復歸	MCC_go_home()
■檢視是否完成回 Home 動作	MCC_check_homef()
■檢視機器始停止運動	MCC_check_stop()

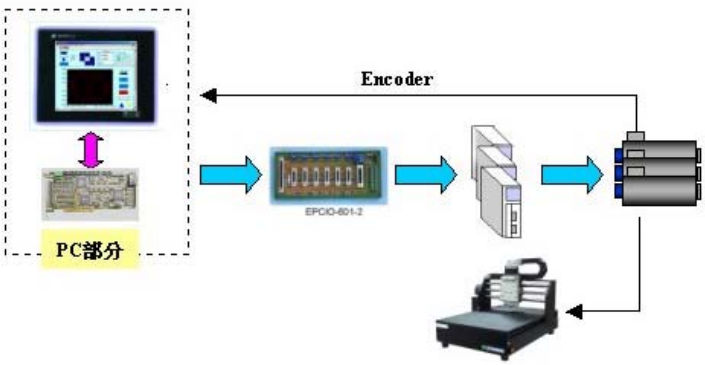
(3) 系統功能

■啓動及關閉運動函式庫	MCC_initial()，MCC_close()
■系統參數值的設定更新	MCC_update_param()
■伺服狀態設定	MCC_set_servo_status()
■讀取馬達讀取馬達 Encoder 值	MCC_get_encoder()

二、設計階段

1、硬體架構與控制器的設計

此範例採用 DSP-Based 的 PMC32-6000PCI Bus 做爲控制介面，並在 PC 上設計一人機介面，經由 EPCIO-601-2 轉接板將訊號傳送給驅動器來驅動馬達控制 XYZ 平台，並從馬達的 Encoder 取得回授資料，形成一迴路控制。如圖十六所示

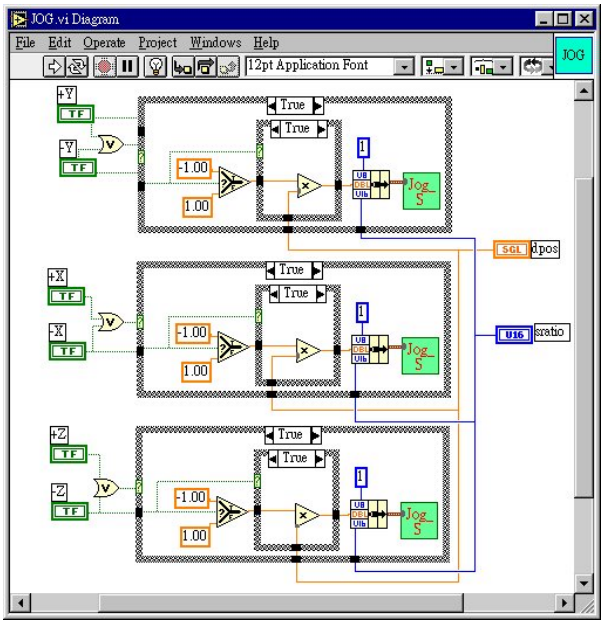


圖十六、硬體架構圖

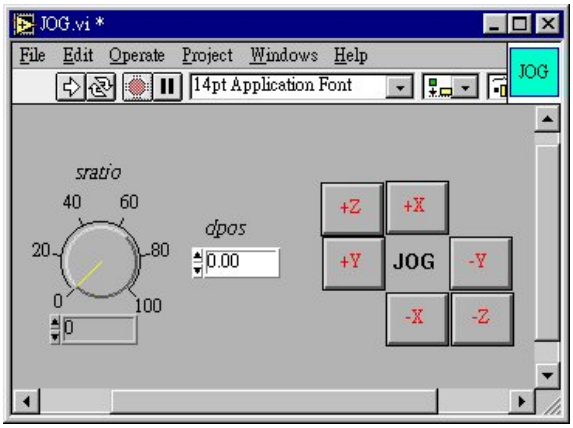
2、雕刻機的模擬程式共分爲以下幾個部分：

(1) JOG 功能

使用者在應用系統的開始或執行過程中可能需要利用手動的方式來帶動機台，而 PMC32-MCCL 提供的手動程式包括粗調、微調、和細調三種模式。而我們所利用的則是細調 — MCC_jog_s 的函式，這函式是針對短行程而設計的，其帶動的行程單位是爲 mm。而爲了應用程式的記憶體的降低和程式管理的簡化，所以將此項功能當成是一個 SubVI 來建構。而程式區如圖十七所示，將將 XYZ 的手動模式編寫成一個個的控制元，當使用者按下人機介面圖十八中+X、-X、+Y、-Y、+Z、-Z 中的任何一個按鍵，便會依照指示執行程式，亦可利用 *dpos* 控制元來改變帶動距離或是 *sratio* 旋鈕來改變運動的速度百分比。



圖十七、JOG 程式方塊圖

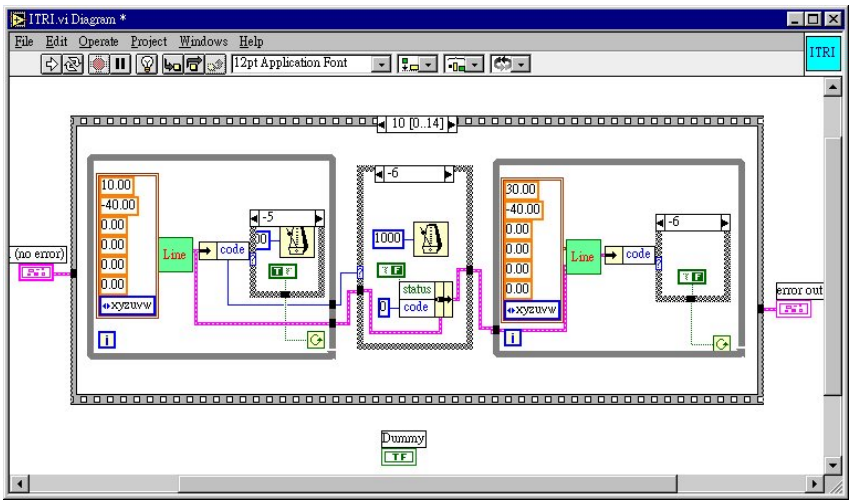


圖十八、JOG 人機介面

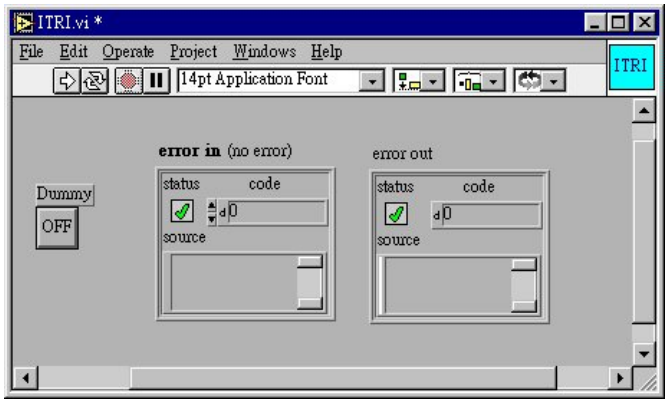
(2) 雕刻 ITRI Mark

而利用 XYZ 三軸平台來模擬雕刻工研院 LOGO。運用到的函式包括了：直線 MCC_line、MCC_check_stop，這兩個函式。而 MCC_check_stop 的功能在於檢視目前機器是否已停止運動，以確保運動緩衝區中以無指令。而其程式區塊和人機介面如下圖十九、二十所示。

針對程式方塊，由於 ITRI LOGO 是由一連串的直線運動函式所執行出來，所以在程式撰寫上首先先建立一個 Frame，再將直線運動函式放入，而當一筆直線函式到下一筆函式中間以 Case 來建構，這是爲了確保每一筆命令皆能確實的進入運動緩衝區之中，所建立的一個保護的程式。



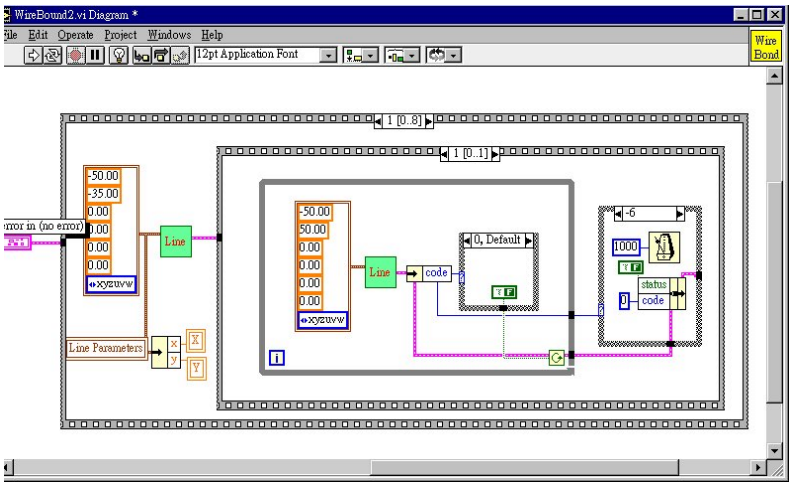
圖十九、ITRI 程式流程圖



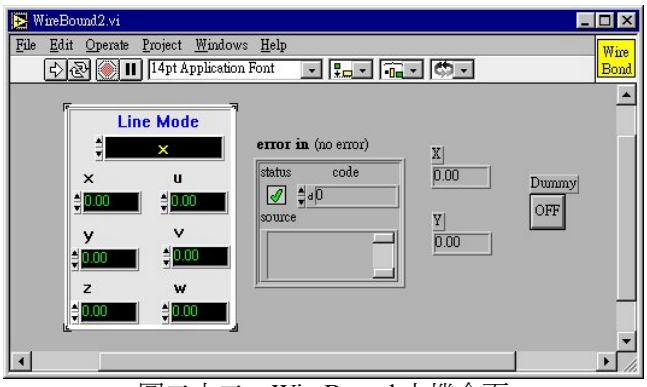
圖二十、ITRI 人機介面

(3) WireBound 功能

除了 ITRI 的雕刻功能外，另外一個程式爲 WireBound 的模擬程式。其程式流程圖和人機介面如圖二十一和圖二十二，方法如上所示。



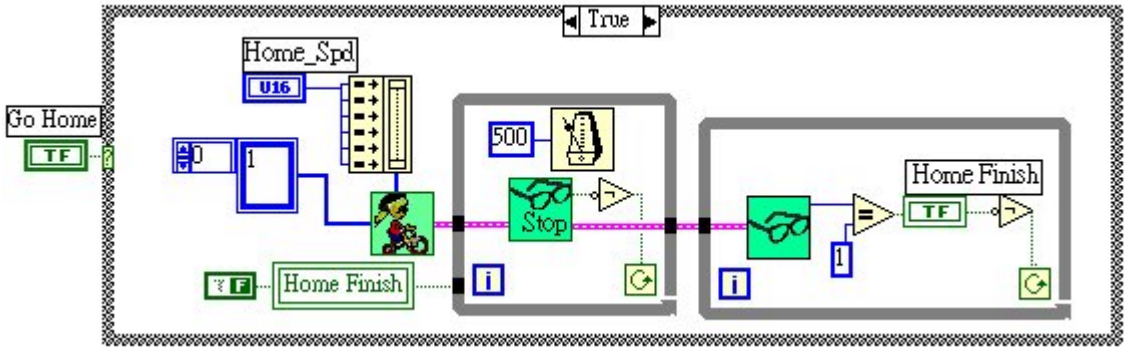
圖二十一、WireBound 程式方塊圖



圖二十二、WireBound 人機介面

(4) Home 功能

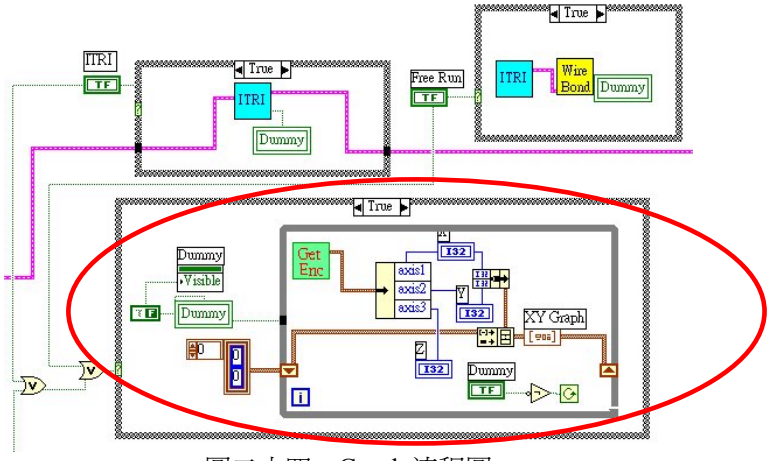
除了 JOG 功能之外，回歸原點亦是機台的重要動作，對於一切的運動控制而言，我們必須要提供一個原點作爲起始點，這時候的回 Home 功能就提供了這樣需求。PMC32-MCCL 原點復歸具有可任意指定原點復歸順序的彈性，而在呼叫此函式前需設定原點參數，可透過 MCC_update_param()函式來，設定的參數有設定的參數有 Home Sensor 的邏輯準位，Index 的邏輯準位，和進行原點復歸的方向設定。而在進行原點復歸中還運用了兩個參數，一個爲 MCC_check_stop()，另一爲 MCC_check_homef()，函式 MCC_check_homef()爲檢視機器是否已完成原點復歸的程序。程式流程如下圖二十三所示。當使用者執行了 Go Home 的按鍵，程式便開始執行回 Home 的動作，當回 Home 結束後程式中的 Home Finish 的 LED 燈便會亮起，這時候則離開迴圈回到了主程式中。



圖二十三、Go Home 流程圖

(5) Graph 功能

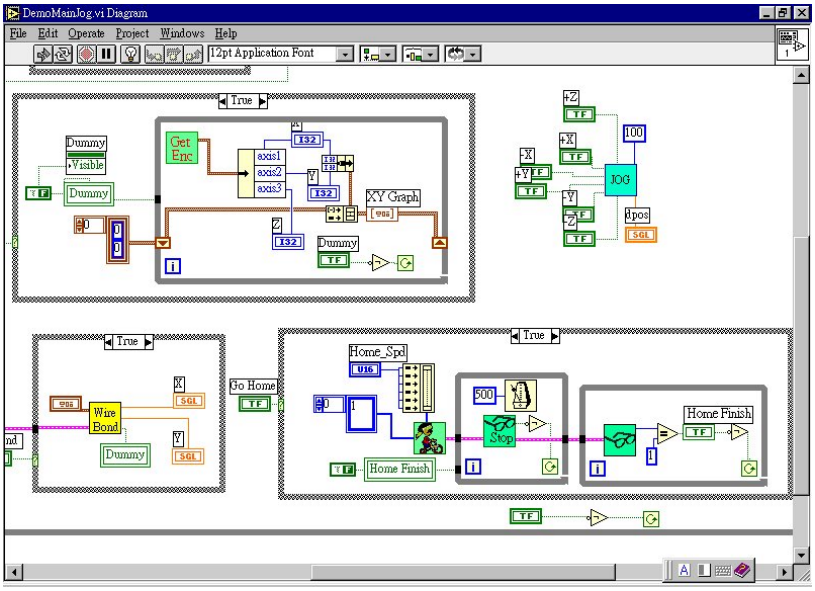
除了以上的運動函式庫之外，爲了讓人機介面上能顯示運動軌跡圖，所以在程式方塊中又加上了繪圖的功能。利用的函式爲 MCC_get_encoder()來讀取目前各軸的 Encoder 值。再利用讀回的數值繪於 Graph 之上，程式流程圖如下圖二十四所示。



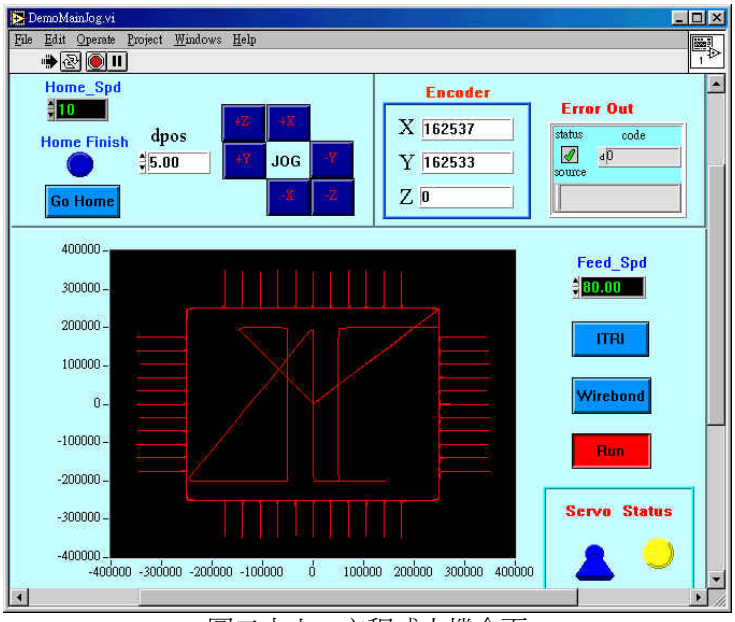
圖二十四、Graph 流程圖

(6) 主程式

當將所有的副程式編輯完之後，就必須編寫主程式將這些副程式放入其中了，主程式的程式方塊圖如圖二十五所示。由於我們已經將部分的運動程式(ITRI、WireBound、JOG)編寫成副程式，所以接下來在主程式之中只要呼叫各個副程式便可執行。由圖中可知我們將各個模擬程式放入 Case 之中，只要程式一開始執行，使用者只要在人機介面中(如圖二十六所示)的控制元件 ITRI、WireBound 或是 Demo 這些控制元件啓動，便可開始執行模擬程式。當開始執行任何一個模擬程式時，這時的便會開始讀取 Encoder 值，並利用下圖中的將其繪圖出來，顯示至 XY Graph 之中，如人機介面中所見的圖形。使用者亦可根據需求執行 JOG 或是或是 Go Home 功能，當程式執行完成後，只需將 Servo Status 中的控制元啓動便可結束此模擬程式。



圖二十五、主程式流程圖

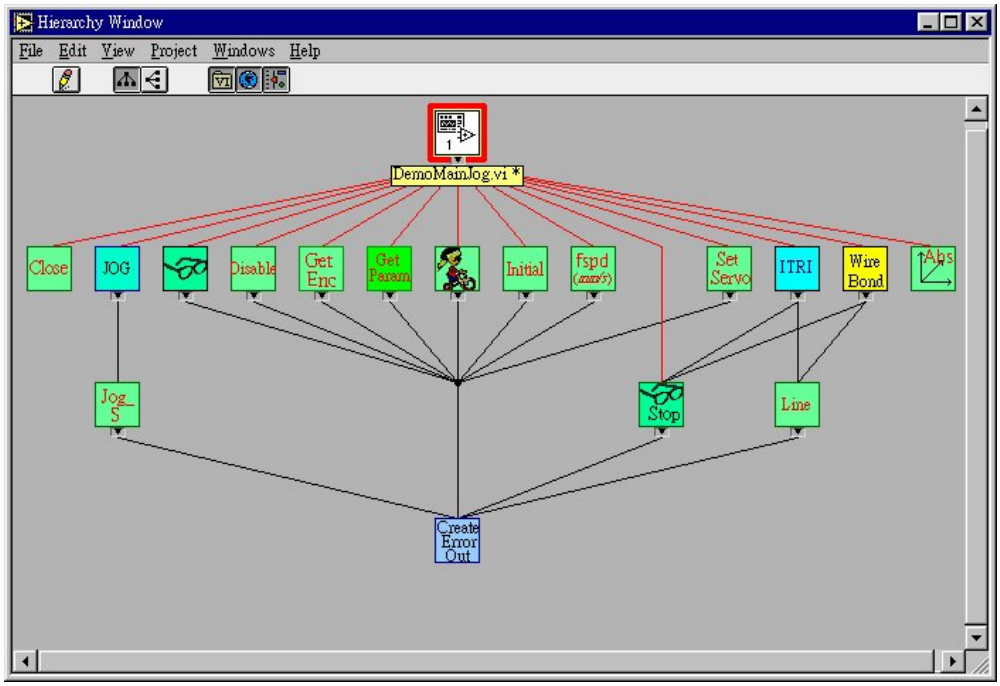


圖二十六、主程式人機介面

捌、結語

從以上的說明和介紹，應該可以對開發控制系統有一定的了解了。而最後我們可以從 LabVIEW 中的層次視窗圖中可以清楚知道這個範例程式的層次架構，如圖二十七所示，將所包含的副程式及所有呼叫到的函式庫，從最上層的主程式 DemoMainJog.vi 一一的將此層次架構起來，讓使用者可以清楚知道在此程式中所呼叫到的函式庫及副程式，而各個副程式又呼叫到的函式庫都可以在這裡清楚的表示出來。

本系統以 PMC32-6000 的 DSP 控制卡結合 LabVIEW 圖控軟體應用於 XYZ 平台之控制，建立一控制系統。而由這個範例程式可以看出圖控軟體高彈性、高效率與具有漂亮人機介面的特性，所以運用在控制器方面都有其優勢。雖然視窗多工作業環境會影響迴路控制時間，但以現今電腦的運算速度，皆可以大幅的改善。所以利用 LabVIEW 來開發應用軟體還是有其便利性和可靠性的。



圖二十七、層次架構

玖、參考資料

1、PC-Based 系統整合革命 LabVIEW 6i 在網際網路上即時量測之應用/劉建昇、蕭子健

2、LabVIEW 在電腦整合自動化的應用實例/張永慶、洪子瑜

3、圖控式語言-LabVIEW/謝勝治

4、LabVIEW 基礎篇/蕭子健、儲昭偉、王智昱

5、以 LabVIEW 爲架構人機介面發展系統設計/彭昺嘉

6、www.epcio.com.tw網站