運動控制器軟體介面與架構概論

■工研院機械所　廖素貞

關鍵詞

使用者介面　User Interface
描述語言　Description Language
控制元件　Control Component
圖控介面　Graphic Control Interface

摘要

本文將運動控制器軟體介面可分成三種形式：函式呼叫(function call)、描述語言方式(descrption language)以及圖控介面(graphic control interface)，並有詳細的介紹。以單機之運動控制器的角度，探討單處理器與多處器運動控制器的運動即時性和同步問題。

前言

當您選擇運動控制器發展您的應用系統時，您的考量因素是什麼？當然是控制卡的規格、價格、穩定性、售後維護以及使用的方便性等因素，然而在產品研發的過程中時間就是金錢，如何在最短的時間內開發完成產品是一個很重要的因素，因此我們利用運動控制器開發應用系統時，運動控制器軟體使用的介面是否簡單、易學習，是縮短應用系統研發時間的關鍵。不論是國內或國外的運動控制器也都提供多種的使用介面讓顧客有更多的選擇，這些使用介面的方式可能是函式呼叫或描述語言的方式，也可能是圖形化的介面，在本篇文章中我們將探討目前市售的運動控制器其使用的介面方式。

一套運動控制器通常包含主機板、運動控制卡、IO 卡，而有些運動控制卡本身就單獨有 CPU 來負責整個運動控制器的運動軌跡計算和迴路控制，這樣的架構筆者稱它爲多處理器運動控制器，在發展應用系統時要選擇單處理器架構或多處理器架構？在文章中亦會有詳細的介紹這兩種架構供您參考。

壹、使用者介面（User Interface）

當運動控制卡提供簡易的使用介面，可以讓使用者很快就熟悉如何使用該運動控制卡，如此可以縮短產品發展的時間，而一般運動控制卡提供的介面形式大致可分爲三種形式包括：函式呼叫(function call)、描述語言(description language)、圖控介面方式，以下我們針對這三種介面方式來介紹說明：

一、函式呼叫(Function Call)

函式呼叫的介面乃是指使用者透過呼叫運動控制器所提供的運動控制函式，配合控制流程來發展使用者需求的應用系統。這些函式包括直線、點對點、圓弧運動、原點復歸(home)、手動程式(Jog)、加減速曲線／時間設定、參數設定等。Delta Tau、Acroloop、MEI、Galil、MIRL 等公司的運動控制器皆有提供此種形式的介面。圖 1 爲機械所之 PMC32-MCCL 運動函式庫的使用範例。

```
#include   “mccl_fun.h”
#include   “mccl_type.h”

main()
{
MCC_initial(0x240,0xb4000,_PMC32_600_);          //運動控制器初起化
MCC_set_fspd(100);     //設定運動速度
MCC_set_abs();         //設定座標型態爲絕對座標
MCC_line_x(150);       //X 軸移動到 150mm 位置
……
MCC_close();           //關閉運動函式庫
}
```

圖 1 函式呼叫介面範例，運動函式庫爲機械所開發的 PMC32-MCLL

這些運動函式通常包裝成 DLL(Dynamic Library Link)或函式庫(function library)的形式；DLL 是一種可在程式執行時載入的碼，使用在 Windows 作業系統，大部分應用系統人機介面之發展工具（如 Visual C/C++、Borland C++ Builder、Visual Basic 等）皆支援 DLL 形式，因爲支援 DLL 之發展工具會提供公用程式(如 BCB 的 implib)將 DLL 轉換成發展工具可接受的函式庫。而函式庫是使用在 DOS 作業系統環境，開發時必須將函式庫連結(link)到發展環境。

大部分運動函式都支援 C/C++編譯(compiler)環境，因此對於會 C 程式語言的使用者，很容易接受函式呼叫方式。

函式呼叫的形式使用起來的感覺就像在程式語言設計，對一般沒有程式語言設計經驗的人來說會有較高的進入門檻，事實上這些函式可加以包裝成後面我們會探討的圖形(Graphic)介面形式。

二、描述語言(Descroption Language)

描述語言的介面乃是由運動控制器的廠商提供一套淺顯易懂語法的直譯器(Interpreter)給使用者，基本上這些語法和指令都很容易學習（至少比學習任一種程式語言要容易的），可以在短時間內就熟悉。使用者只要依照該語法撰寫自己的應用系統就可以了，請參考圖 2。

```
INSTRUCTION          INTERPRETATION
#INITIAL             Program label
HMXY                 Drive X and Y to home
BGXY                 Start Motion
AMXY                 Wait until completion
X0=0                 Define starting position as zero
Y0=0                 Define starting position as zero
#LOOP                Label
                     :
                     :
JP   #LOOP           Repeat the process
EN                   End of program
```

圖 2 描述語言範例，Galil Motion Control,Inc 運動控制卡之指令和語法

描述語言並沒有統一的語法，各廠家的運動控制器皆有各自一套的指令和語法，所以如果您是第一次使用某廠家的運動控制器就一定得先熟悉該運動控制器的指令和語法。前面所談的函式呼叫介面，就不一定有如此需求。

三、圖控介面(Graphic Control Interface)

圖控介面顧名思義就是以圖形化的方式來開發應用系統，大致可分爲幾種形式（一）CAD-to-motion 將電腦輔助設計(CAD,Computer Auxiliary Design)的繪圖軟體產生的幾何圖形資料（如 AutoCAD 之.DXF、HP-GL 之.PLT、G-Code）轉換成運動控制器可接受的運動命令。Parker、Galil 公司皆有提供此形式的工具(toolkit)。圖 3 爲 Parker 公司 6000 Series Software 其中的一套軟體工具

CompuCAM，它就是 CAD-to-motion 的介面形式。

圖 3 CAD-to-motion 範例，Parker 公司之軟體工具 CompuCAM

（二）流程圖（flow charts）　使用者以類似像畫流程圖的方式來設計開發自己的運動控制應用。將控制器功能包裝成許多運動控制區塊(motion control block)系統，選取所需的控制區塊並依系統流程把各控制區塊連接，並把設計完成的流程圖檔案經過轉換下載（download）至運動控制器。圖 4 爲 Parker 公司控制器所提供的一套類似流程圖的工具 Motion Builder。

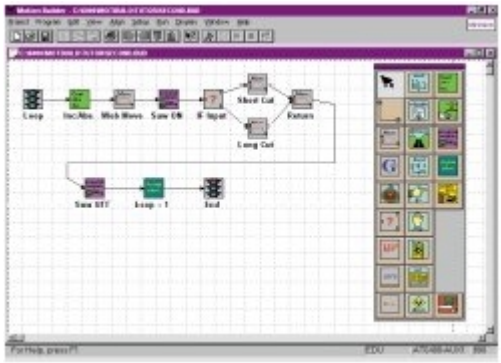

圖 4 流程圖介面範例，Parker 之軟體發展工具 Motion Builder

（三）VI 元件　VIs（Virtual Instruments）是開發工具 LabView 的元件名稱，LabView 是一套圖形化 icon-based 的發展工具，將運動控制器提供的功能函式包裝成 VI 的形式載入到 LabView 開發工具來設計發展使用者應用系統的人機介面。Galil、Parker、NI、Delta Tau 等公司的運動控制器皆支援 LabView 發展環境。圖 5 爲機械所之 PMC32-MCCL 運動函式庫包裝成 VI 元件使用範例。

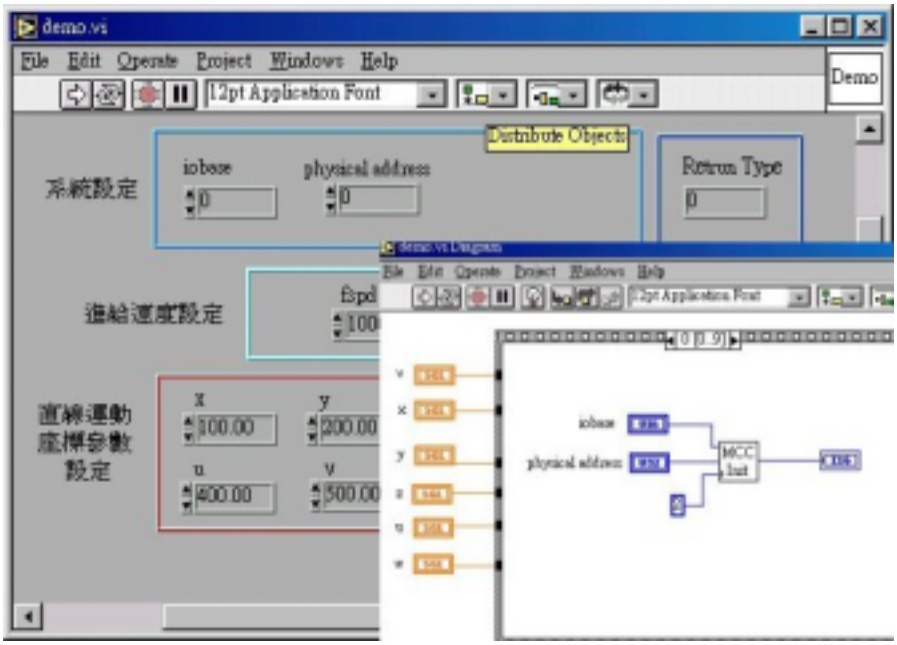

圖 5 機械所 PMC32-MCCL 運動函式庫之 VI 元件使用範例

（四）控制元件(component)　　將運動控制器的功能包裝成控制元件(component)，這些控制元件可能是 VBX、OCX、ActiveX 或 OLE 等，VBX 是 Visual Basic 的控制元件，ActiveX 控制元件可以被不同種類的軟體開發工具使用（如 Visual C/C++、Borland C++ Builder 等）。使用的方式是將控制元件載入到發展工具中，選取所需的元件，透過設定或讀取該元件的屬性（property）完成某項功能，也可以在該元件產生事件（event）時撰寫處理程式碼。Parker、Galil 公司有提供控制元件形式的介面。圖 6 爲 Galil 公司利用一套該公司所提供的 Active X Tool Kit 將運動功能包裝成 VBX 控制元件，載入到 Visual Basic 環境所設計的範例。

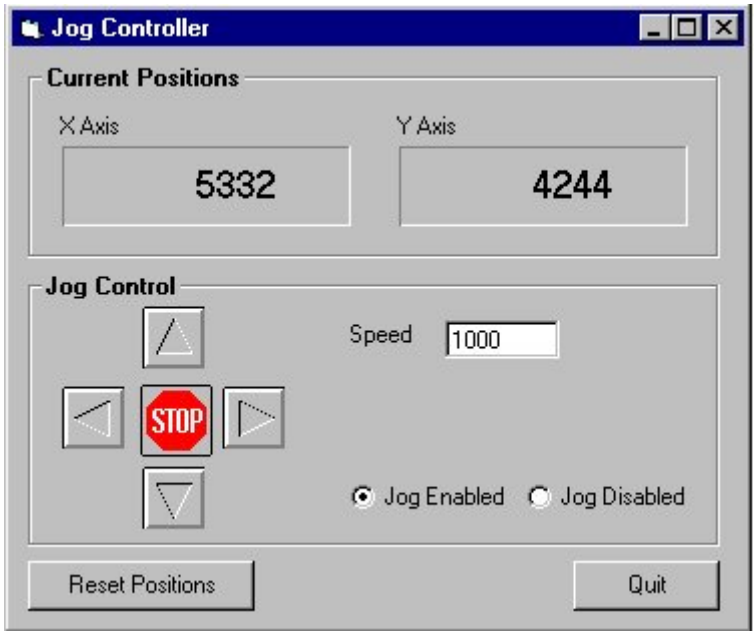

圖 6 控制元件，資料取自 Galil 公司之使用範例

圖控介面已是目前的趨勢，使用者無需花太多的時間學習發展工具，他的方便性大大的縮短了系統研發的時間，達到 Time to Market。

目前大部分的運動控制器，也都同時提供了不同的使用介面，讓使用者有更多的選擇，但在價格方面也會有區分。也就是說相同的硬體控制模組，會因爲不同的軟體介面包裝而改變了產品的價值。

貳、單處理器與多處理器之運動控制器

如果一套運動控制器只用一個 CPU(Center Process Unit)來處理所有事情，包括讀取輸入資料、顯示輸出資料、中斷處理、計算運動路徑以及人機介面流程的執行等，將它定義爲單處理器運動控制器。如果控制器中除了主機板上的 CPU 外，在控制器的運動控制卡也有獨立一個 CPU 來運作，在這裡我把它定義爲多處理器之運動控制器。本文都是以單機的狀況來討論，不包含分散式的控制系統。

本章節與第一章節的使用介面，並沒有直接相關性，因筆者有機會同時接觸此兩種架構的控制器，故藉此談談兩種架構的特質與運動的即時性。以下就針對這兩種架構來討論：

一、單處理器運動控制器

由於整個運動控制只有一個處理器，因此這個處理器負責了所有的任務。運動控制器中運動控制模組，會將欲執行的路徑（如點對點、直線、圓弧運動）切割成數個小線段，即所謂的插補（interpolation）運算，再針對每一小線段去計算輸出資料（如圖 7），這些輸出資料的單位爲脈衝（pulse）。運動控制模組每個插補點輸出必須是連續，才能保證運動路徑的平滑性，所以插補點計算是一個具即時性、優先權高的模組，通常以中斷處理的機制完成。

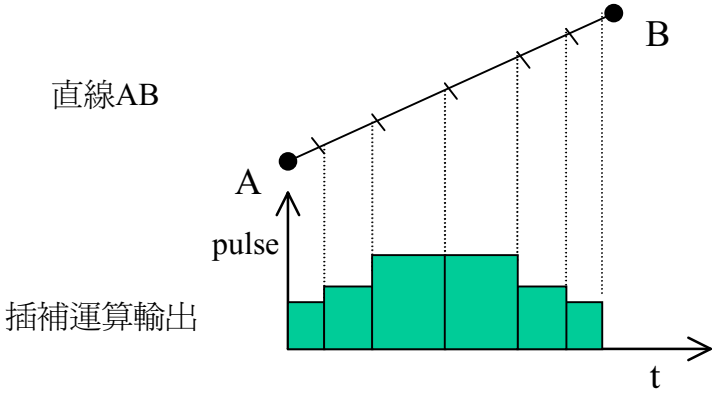

圖 7 路徑規劃與輸出脈衝

在 DOS 作業系統時代使用者可以很容易掌握運動控制模組的即時性，因爲使用者可以直接存取或規劃硬體的資源，當中斷信號發生時使用者可直皆攔截而進行處理，就算同時使用很多中斷，使用者也可以很明確知道各中斷的優先順序以及可能被延遲的時間。而在 Windows 作業系統環境，所有的中斷信號和記憶體存取，都必須透過系統核心管理程式來運作。由於作業系統所要處理的中斷信號很多，不只是使用者設定的中斷信號，還包括鍵盤、滑鼠等週邊裝置所發出的中斷信號，對於所有中斷信號作業系統會一視同仁，將每個發生的中斷依先後順序放在一個佇列（Queue）中，而依序處理各個中斷服務程式，如此的機制造成使用所設定的中斷處理程式可能無法很快被處理，因爲會被其他週邊產生的中斷信號給延遲了，而這延遲的時間使用者亦無法掌握，因此可能產生運動控制器之運動路徑不連續。通常 Windows 作業系統在不加裝任何即時功能的子系統下，在最壞的狀況（worest case）下，可保證的中斷反應時間爲數百毫秒(ms)。大部分運動控制系統可接受的即時性大約在數毫秒到數十秒的範圍，甚至有些要求是在數百微秒(us)，因此數百毫秒的即時(RealTime)性保證是很難被接受，不過目前亦有許多解決的辦法：

解決方法一：在 Windows 作業系統下另外加裝具有增強即時功能的子系統（如 VenturCom 公司所推出的 RTX）。

解決方法二：發展一套「純」Windows 作業系統的運動函式庫，即在 Windows 作業系統下不加裝任何應用程式及周邊設備，如此可以減少不必要的中斷信號產生，而提升系統的即時性。
解決方法三：選擇具有以硬體完成路徑規劃功能的運動控制卡，系統只要下達欲執行的運動命令（如點對點、直線運動），無需進行插補計算，因此大大減低即時性的需求。
方法一的方式無形中就增加了產品的成本，方法二限制使用者使用作業系統的方式，倘若使用者沒遵循限制，可能就要自行負責後果。方法三由於是利用硬體完成插補計算，所以運動功能的規格易被限制，比較沒有彈性，較難完成複雜性高的插補計算。

二、多處理器運動控制器

多處理器的運動控制器，基本上有兩個 CPU，一個爲控制器的主處理器負責控制系統的人機介面包括讀取輸入資料、顯示輸出資料以及控制流程的執行，另一個爲負責運動控制卡的軌跡插補計算和伺服迴路控制。由於運動控制卡本身具有獨立運作的能力，因此運動卡只要透過傳輸埠外接個教導盒（Teach Box）或顯示面板就可變成嵌入式（Embeded）系統。

多處理器運動控制器之運動控制卡，本身有 CPU 專門負責所有運動軌跡的插補計算，即時性容易掌握，不受控制器作業系統的影響，並可接受複雜的計算，所以迴路控制功能多以軟體來實現，如此可提供多種不同的控制法則給使用者選用，甚至開放讓使用者自行撰寫控制法則，下載到運動控制卡；因此多處理器運動控制器很適用於較複雜的控制系統。由於多處理器運動控制器比單處理器運動控制器多了一個 CPU，所以在硬體元件和電路設計會比較多且複雜，所以在硬體成本價格方面比單處理器控制器高。

介面方式

兩個 CPU 必須透過一個介面進行溝通，這個介面可能是一些標準的串列或並列的傳輸埠（port），也可能是可雙向讀寫的記憶區塊（Dual Port RAM）。事實上大部分的控制器都會同時提供這兩種溝通介面，不過在執行的期間通常只會選擇一種介面。因爲這兩個 CPU 各自獨立運作，且可能會資料格式不一致（如 Intel 和 TI 公司之 CPU 的浮點資料格式就不同）因此溝通時的同步機制及資料轉換是很重要的，否則可能會產生資料被覆蓋（overwrite）或資料錯誤的現象，甚至會產生死結（dead lock）的狀況。

一般標準的串並傳輸介面，對溝通介面的信號都有明確的定義，透過介面

信號的狀態，可得知要傳送資料或接收資料以及傳送資料是否成功。但若使用者是使用 Dual Port RAM 做溝通介面時，使用者必須自行規劃溝通的機制。Dual Port RAM 的使用就像一般記憶體（memory）一樣，通常我們會把它規劃成幾個區塊，包括系統共用區、輸入資料區、輸出資料區、使用者區以及保留區，分述如下：

系統共用區：此區的資料包括系統狀態以及同步機制所需求的變數，如命令輸入旗號（flag）、系統命令的代碼。

輸入資料區：此區的資料通常包括系統環境和程式設定的參數、運動命令所需的資料。

輸出資料區：此區的資料通常是顯示目前運動狀態，如運動是否停止、原點復歸是否完成、錯誤代碼以及目前運動的位移值等。

使用者區：有些運動控制卡可能有提供給使用者撰寫程式（如控制法則）的功能，使用者就可藉由此區定義資料參數。

保留區：供未來發展系統程式擴充功能之用。

下達命令形式

從控制器下達命令到運動控制卡通常有兩種形式：循序或批次的方式，分述如下：

循序式：控制器下達一筆運動命令到運動控制卡，會等運動控制卡接受命令後，再下達另外一筆命令。通常運動控制卡會有運動命令緩衝區，可以暫存數筆命令。此種下達命令的形式，控制器與運動控制卡的溝通頻繁，因此同步機制的處理會較複雜。

批次式：在控制器端先完成運動控制程式（運動程式的形式可能是圖形介面或描述語言），轉譯成運動控制卡的命令，再下載到運動控制卡，由於所有控制程式一起下載，因此控制器與運動控制卡的溝通沒有像循序式那樣頻繁，同步問題自然就減少，但此種形式，控制器端要提供一套轉譯器的工具給使用者。

即時性

單處理器運動控制器一個 CPU 要負責太多工作，因此在複雜的作業系統下會有即時性問題，而多處理器運動控制器之運動控制卡可以專心的進行軌跡插補計算，因此比較沒有即時性的問題。

結論

在這競爭激烈、技術日益精進的市場裡，工程師有很多新的技術要研發和處理

應用系統整合的問題，他們不需要也不想花費太多時間在學習一套工具或程式設計；因此易懂易學習的使用介面是未來趨勢。單處理器與多處器之運動控制器的介紹，希望有助於在選擇控制器時，另一個思考的層面。

相關參考資料

[1] Galil：www.galilmc.com
[2] Parker：www.compumotor.com
[3] Delta Tau：www.deltatau.com
[4] Acroloop：www.acroloop.com
[5] MEI：www.motioneng.com