高整合性運動控制器軟體設計

作者: 黃美燕

摘要

本文主要介紹與說明二代控制器架構對系統及軟體之效益，如 PC 傳送命令至 EPCIO 板的 FIFO Stack 將降低作業系統的即時需求且在提高軌跡規劃插值點精度時不增加 CPU 負擔；控制軟體可彈性規畫控制迴路應用硬體或軟體控制法則；EPCIO 板 latch 多中斷源機制可提高軟體處理多中斷源效益；最後提供一功能強大的即時運動控制程式庫以提供使用者一簡單好用的使用介面。

壹、前言

工業用伺服馬達控制器不斷地推陳出新，從開迴路的馬達控制到精密的位置控制；工控器的性能隨著應用場合的改變而有不同程度的需求，由陽春型升級到豪華配備。工控器講求是高穩定性、高精密度、響應速度快與彈性化的介面支援。最簡單的馬達控制板所具備的功能有速度控制、位置控制與少數的 I/O（輸入/輸出）控制點。市場上現在流行的是四軸卡，即一片控制卡可以同時控制四軸馬達，卡上通常會提供一些 I/O 點供使用者應用。遇到四軸以上或者 I/O 點需求較多的場合，則可選用更高軸數或多卡配置，同時添購泛用型 I/O 卡。從成本的角度來分析，國產與進口的控制卡的性能相去不遠，價格卻相差甚多。目前國內有多家廠商提供物價廉的四軸卡，二片四軸卡比一片八軸卡成本低。使用者在成本的壓力會選用二片四軸卡搭配泛用型 I/O 卡，在一陣東補西填的拼湊下系統變複雜了，也為以後的系統維護埋下深遠的隱憂。可否找到一片整合多軸且同時擁有彈性化的 I/O 介面規劃能力的控制卡？EPCIO 是一個高整合性的控制器核心 ASIC，它提供了一連串的精采創作，為伺服馬達控制卡設計者提供了一條康莊大道。EPCIO 板提供六軸伺服馬達定位控制、九軸 Encoder 計數器、最多七百六十八點的分散式數位輸出入控制、八組 ADC 控制及八組 DAC 電路。其中 EPCIO 板的 FIFO 設計將提高補間控制(contouring motion control)精度;DAC 提供選擇輸入來源以彈性應用硬體或軟體控制法則;另 EPCIO 板可 latch 多個中斷事件，控制軟體再藉讀取 EPCIO Latch 中斷源之暫存器，判斷發生的中斷源，以及時處理多個即時性的中斷事件。本文主要介紹第二代控制器架構對系統及軟體效能影響，並介紹工研院機械所控制軟體架構，最後針對此 EPCIO 控制器提出一運動控制程式庫，提供使用者使用 EPCIO 板之一介面；使用者不需深入運動控制中複雜的軌跡規劃、定位控制、即時多工環境，透過此函式庫直接呼叫函式即可快速開發整合系統。

貳、本文

一、第一代控制器 CPU 送命令至 Motion Card 架構-Single Buffer

在第一代控制器，CPU 送命令至 Motion Card 的方式為每一固定時間(在此稱 DDA Time)，送 DDA 命令至 Motion Card，Motion Card 則在此 DDA Time 將命令平均送至 Motor Drive。圖 1 為軌跡規畫所得的速度-時間圖，DDA 命令則為每一 DDA Time 所規畫的馬達進給量，即速度-時間圖中 1、2、3…之面積。使用時，Motion Card 在每一 DDA Time 中斷 CPU，CPU 需即時將 DDA 命令寫入 Motion Card。

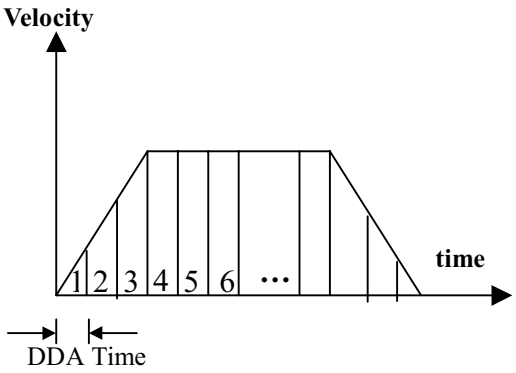


圖 1、軌跡規畫所得的速度-時間圖及 DDA 命令

二、 Single Buffer 中 CPU 寫入命令的延遲時間對馬達轉動的影響

當控制板對 CPU 發出中斷，要求 CPU 寫入 DDA 命令，從發出中斷至真正寫入 DDA 命令，其延遲時間必須很短，否則反應在馬達轉動上會造成速度不連續現象，如圖 2(A)，CPU 寫入命令的延遲時間很短，因此馬達轉動速度較為連續；但如圖 2(B)，CPU 寫入命令的延遲時間較長，因此馬達轉動速度會反應出不連續現象。因此使用 Single Buffer 架構對 OS 即時性需求相當高，以一般伺服系統而言，需達 3ms 的即時性需求。

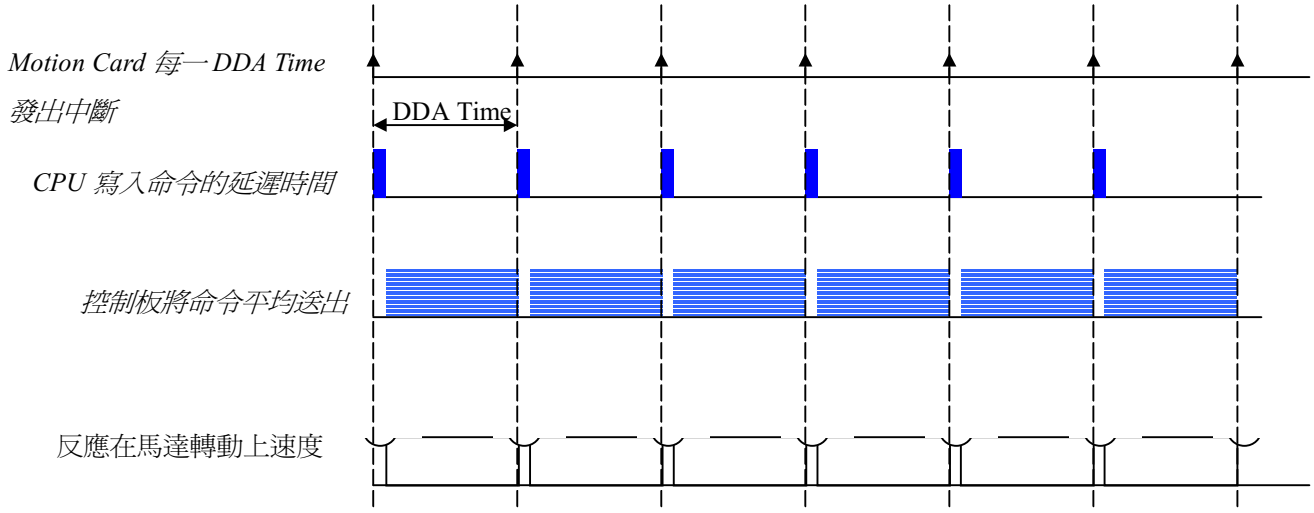


圖 2(A) 、 CPU 寫入命令短延遲時間對馬達轉動的影響

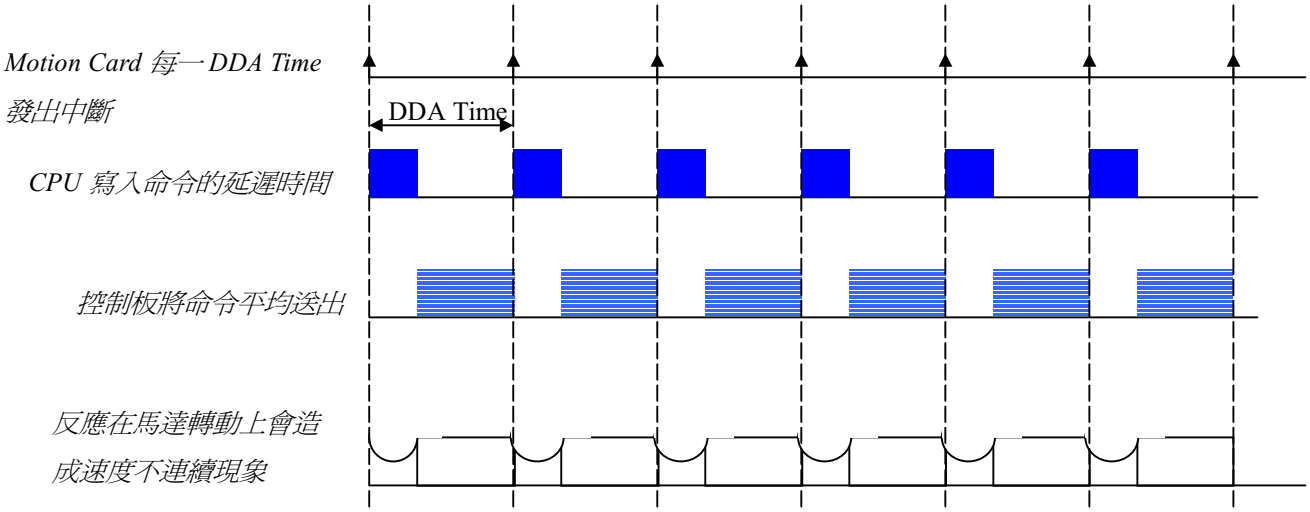
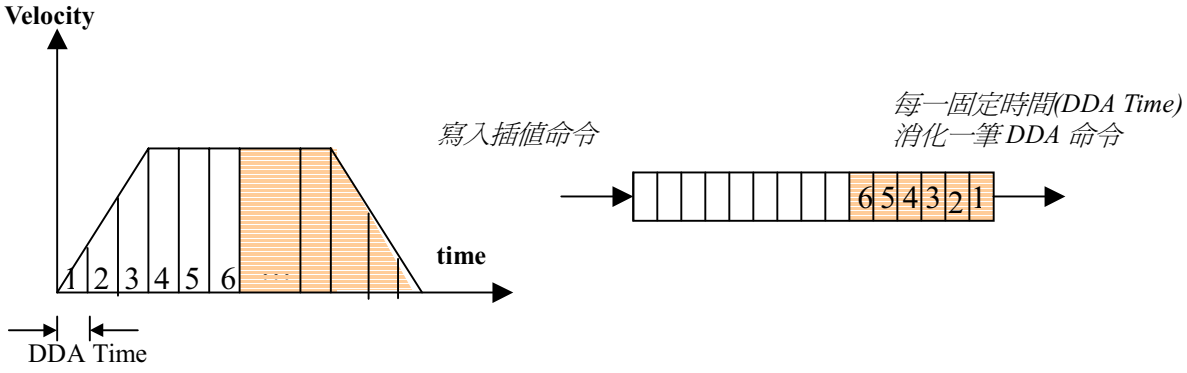


圖 2(B) 、 CPU 寫入命令長延遲時間對馬達轉動的影響

三、第二代控制器 CPU 送命令至 EPCIO 板架構-FIFO

第二代控制器 EPCIO 板，將 CPU 送命令架構改善為 FIFO，FIFO 為 PC 寫命令至 EPCIO 板的緩衝區，其最大效用可降低填入 DDA 命令的即時需求，從 EPCIO 要求 CPU 送出命令到 CPU 真正寫入命令，可有一緩衝時間；且 CPU 一次填入多筆命令，可提高運動插值點的精確度，並降低 CPU 負擔，提高系統效能；以下介紹 FIFO 運作原理及使用方法。

FIFO(First In First Out)為 EPCIO 板存放 DDA 命令的緩衝區，當 User 不斷填入新的命令，FIFO 依序由較先進入之命令送出，其最大可存放筆數為 63 筆；使用者由 PC 端寫入 DDA 命令，FIFO 則於每一 DDA 時間消化一筆命令，其寫入與消化命令不需有同步關係；如圖 3 為軌跡規劃所得的速度-時間圖，DDA 命令為每一固定時間規劃的馬達進給量(如 1、2、3…之面積)。當寫入 FIFO 之命令比送出命令還快，FIFO 可能出現滿(Overflow)的狀態，或當送出比寫入還快，則出現空(Underflow)的狀態。因此上層軟體必須讀取 FIFO 狀態，當 FIFO 為滿，則不再丟 DDA Pulse 命令，以確保命令正確性，或當 FIFO 接近空的狀況，則填入新的命令以防止命令發生不連續現象，造成馬達轉動不連續。



軌跡規劃速度-時間圖

圖 3、FIFO 的寫入與送出

(一) FIFO 使用方式

1. 寫入 FIFO 命令與 FIFO 消化命令無同步關係 – 藉由判斷 FIFO 狀態

軟體直接運算軌跡規劃插值點，並送至 FIFO Stack，此時上層軟體只管計算出固定時間馬達進給量，且判斷 FIFO 不為滿的狀態就送出 DDA 命令。這樣的方式，上層軟體架構單純，但佔大量 CPU 時間 Polling FIFO 狀態，且當外界裝置有中斷需求則 CPU 負擔更大。

2. 利用最小庫存中斷機制監督當剩餘筆數

FIFO 提供監督最小庫存中斷機制，使用者可設定最小庫存數目，監督當剩餘筆數小於設定的最小庫存數，則發出中斷，通知 CPU 丟命令至 FIFO。

(二) FIFO 效益

1. FIFO 監督最小庫存數機制降低 OS 的即時需求且 CPU 一次填入命令筆數影響中斷 CPU 頻率

1.1 軟體每次填進一筆命令且最小庫存數為零

為了讓上層軟體對周邊裝置的狀態有較即時的監控，較好的設計為填入命令與 FIFO 送出命令有一同步關係，先以軟體每次填進一筆命令且最小庫存中斷為零討論，上層軟體每次填進一筆命令，待 FIFO 拿走這一筆命令而為空的狀態即發出中斷，CPU 再填入下一筆命令，如圖 4(B)，這樣的架構相當 FIFO 每消化一筆命令便發出中斷，其即時需求為上層軟體需在小於 DDA Time 時間內填入新的命令。

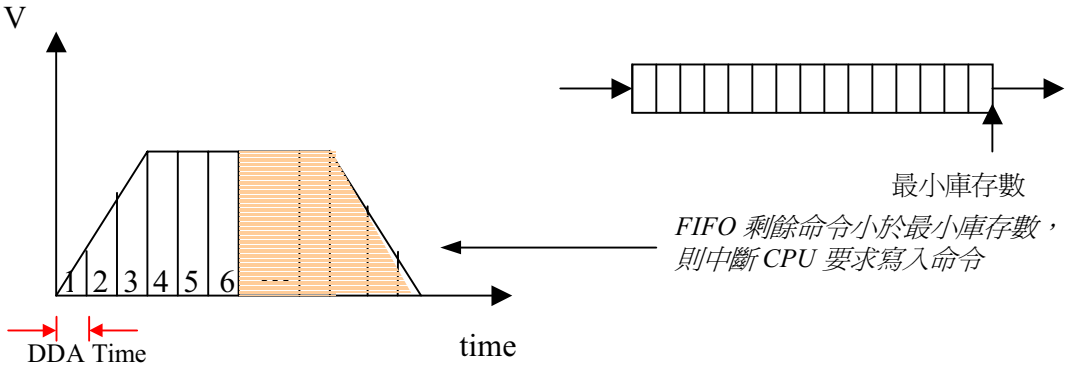


圖 4(A)、FIFO 的寫入與送出有同步關係

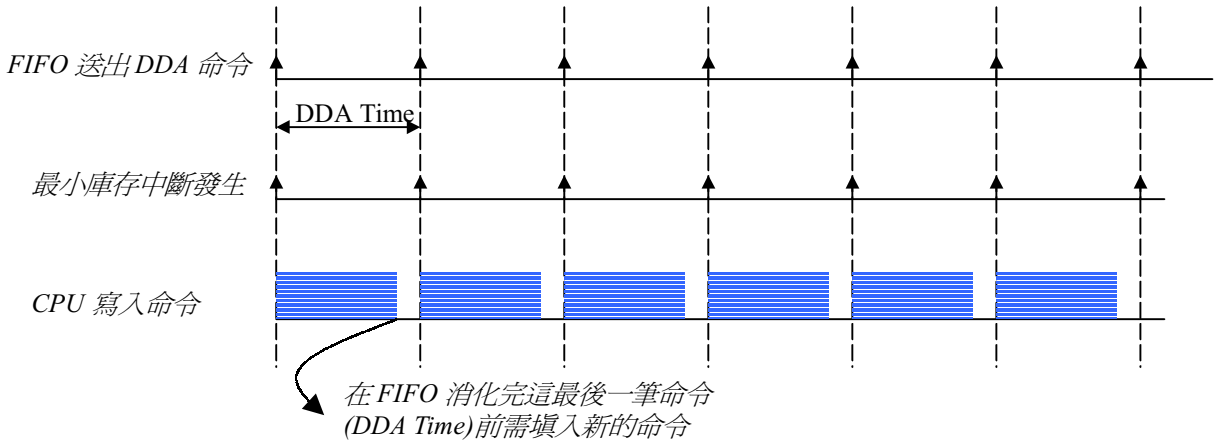


圖 4(B)、FIFO 的寫入與送出有同步關係

1.2 設定最小庫存數 n 且 CPU 一次填入命令筆數為 N

既然 FIFO 有監督最小庫存數並發出中斷要求 CPU 填入命令功能，若增加最小庫存數，CPU 則有更充裕時間填入新的命令，其即時需求程度也可降低。以下舉一實例說明，將最小庫存數 n 設為 3，上層軟體每次寫 N(=4)筆命令，則當最小庫存數為 3 時發出中斷，要求 CPU 填入下 4 筆命令，此時 FIFO 庫存數為 N+n(=7)筆，見圖五(A)，而當 FIFO 忠實的於每一 DDA Time 消化一筆命令，其庫存數也將由 6、5、4 減至 3，此時 FIFO 則又發出最小庫存中斷；如圖 5(B)，這樣的架構相當 FIFO 每消化軟體寫下的 N(=4)筆命令，碰到最小庫存數就發出中斷，因此 FIFO 對 CPU 發出中斷時間為 N 個 DDA Time 時間；而發出中斷時，FIFO 還有 n(=3)筆命令庫存，其即時需求為上層軟體需在小於 n+1(=4)個 DDA Time 時間內填入新的命令。

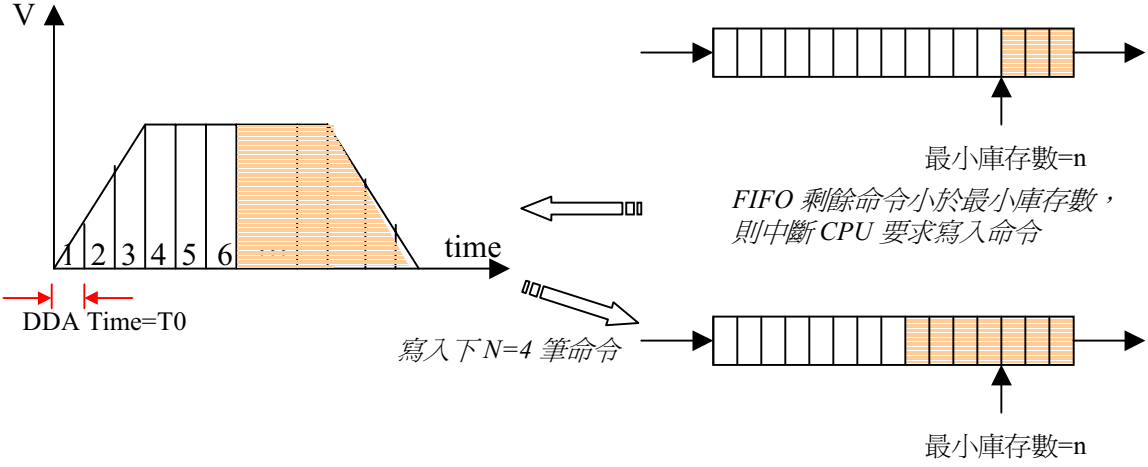


圖 5(A)、監督最小庫存數 3 並發出中斷要求

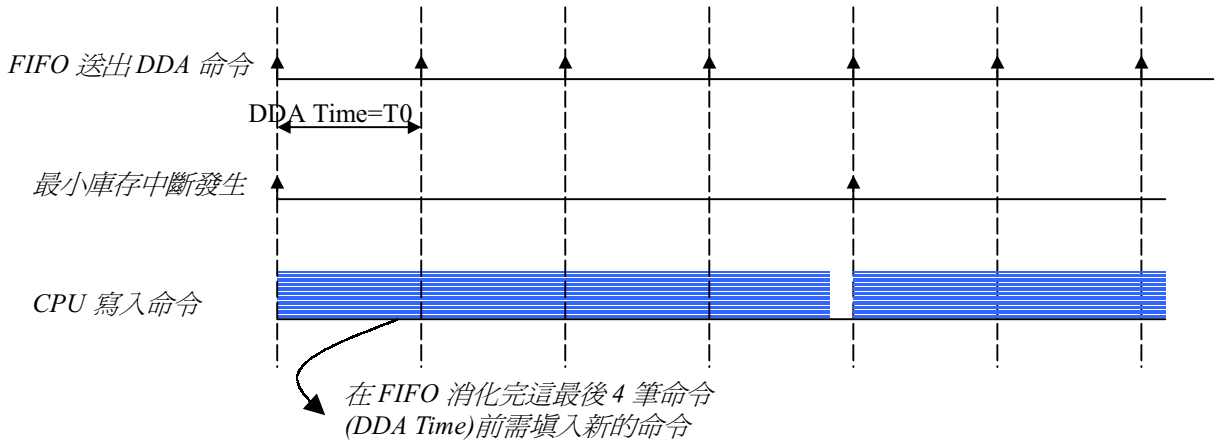


圖 5(B)、寫入 FIFO 即時需求

2. FIFO 可提高插值精度，但不會因中斷發生頻率太高對 CPU 造成負擔

FIFO 另一項值得一提的是可提高軌跡規劃插值精度(contouring trajectory resolution)，當插值點精度提高，則可降低定位誤差，軌跡運動之追蹤性也會變好；尤其圓的補間最為顯著，如圖 6，插值點越多，誤差愈小，運動的實際軌跡也愈逼近指令軌跡。

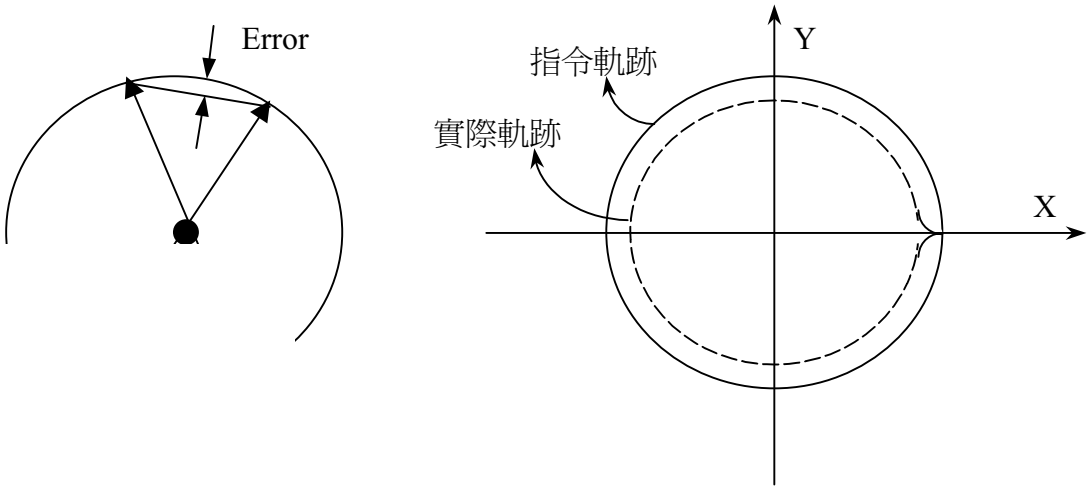


圖 6、插值點數與誤差、運動的實際軌跡關係

因此只要 CPU 運算速度夠快則上層軟體可將插值精度提高，在第一代

控制器架構裡，若將插值點數提高，則對 CPU 產生中斷頻率也提高，對 CPU 將產生更大負擔，在第二代控制器裡此狀況得以改善，如圖七，將圖五的軌跡規劃插值點數提高一倍(此時 DDA Time 為圖 5DDA time 的 1/2)，當 CPU 被中斷，一次寫入 8 筆 DDA 命令，FIFO 則每次消化 8 筆命令才中斷 CPU 一次，因此 CPU 中斷時間為 8 個 DDA Time，與圖 5 中斷時間相同(為 4T0)，但卻提高插值精度。由此可知，利用 FIFO 一次填入多筆命令，可提高插值精度，但不造成對 CPU 更大的負擔。

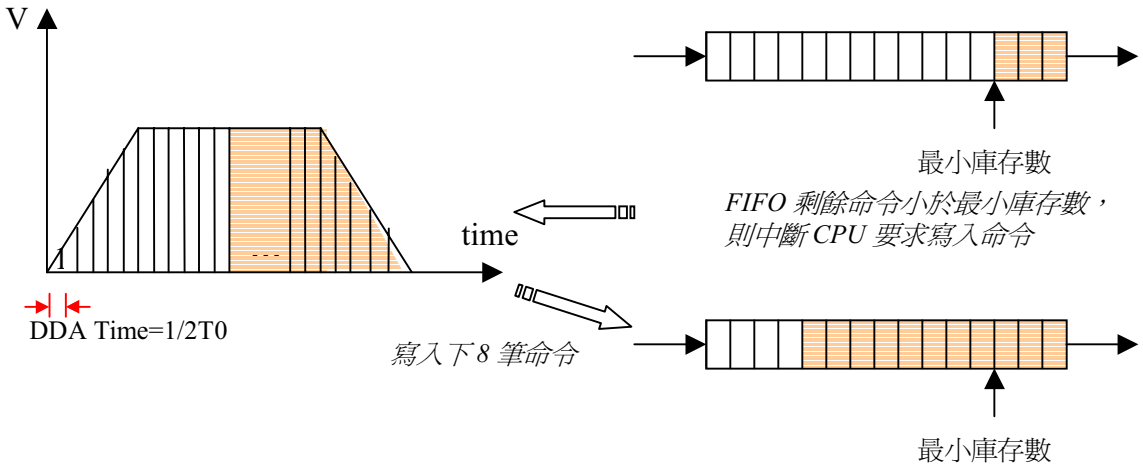


圖 7(A)、提高插值精度

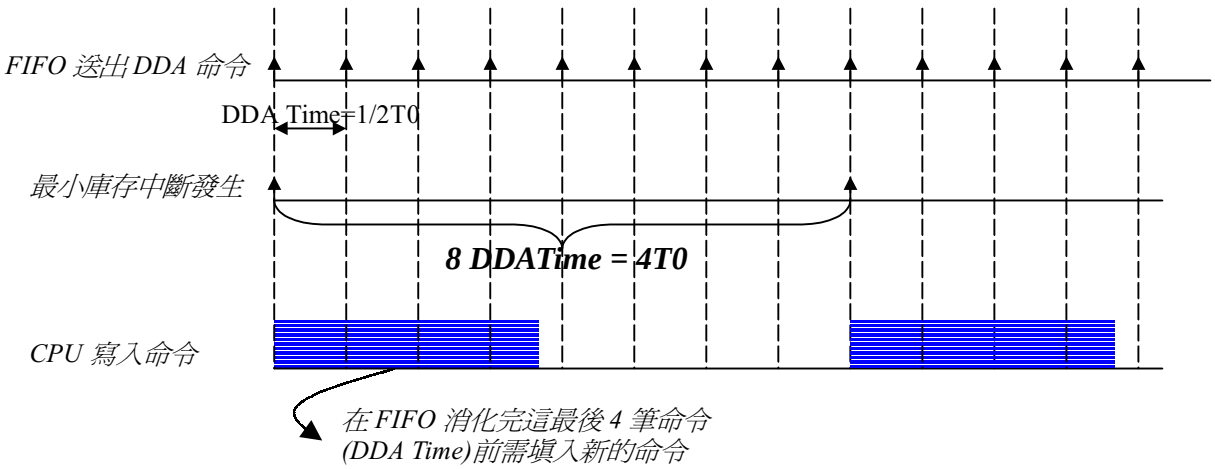


圖 7(B)、提高插值精度

四、EPCIO 控制軟體可彈性規畫各軸送出命令型式

EPCIO 板送出命令至馬達 Driver 命令含 Pulse 命令及速度命令，Pulse 命令輸出可規劃為 10~15bit，速度命令為 16bit。使用者根據系統需求，可自行規劃各軸送出命令型式；當規劃為 Pulse 命令輸出，如圖 8，使用者寫入每一 DDA 時間馬達進給量，再由硬體的 DDA(Digital Differential Analysis)，將命令平均送出。

當規劃 EPCIO 板送出命令至馬達 Driver 為速度命令輸出，軟體可選擇 DAC 輸入來源以決定應用硬體或軟體控制法則，當選擇應用硬體控制法則，即由 EPCIO 板做位置 Close Loop，則如圖 9，使用者可寫入每一 DDA 時間馬達進給量，由硬體的 DDA(Digital Differential Analysis)及位置補償命令透過 DAC 送出電壓命令至馬達 Driver，其中 P Gain 設定直接由軟體設定；又當選擇應用軟體控制法則，即直接由軟體送出速度命令或使用軟體自行訂定的控制法則，如圖 10，透過 User 自定的控制法則，直接將速度命令送至 EPCIO 板，並由 EPCIO 板的 DAC 輸出。值得一提的是，EPCIO 可規劃 6 軸 Pulse 命令輸出及 8 軸速度命令輸出以達到 14 軸的最大控制軸數。

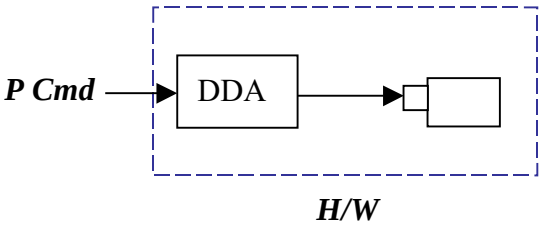


圖 8、軟體寫 Pulse 命令由 EPCIO 送出 Pulse train 至 Motor Drive

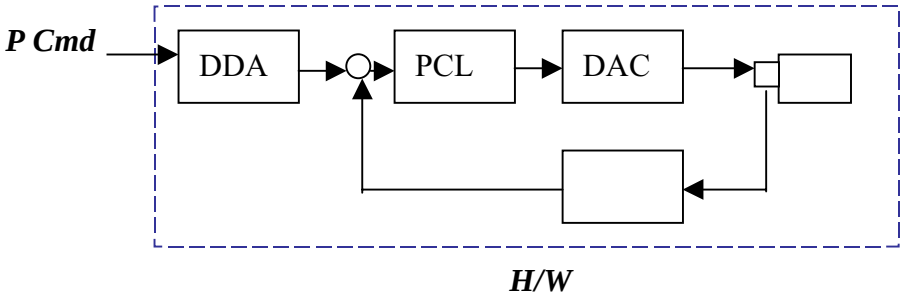


圖 9、軟體寫 Pulse 命令由 EPCIO 送出速度命令至 Motor Drive

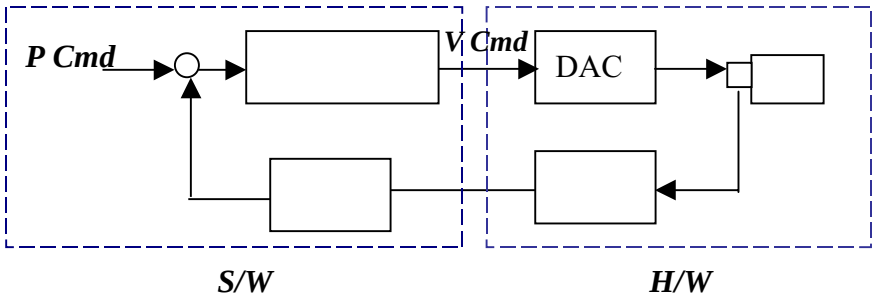


圖 10、軟體寫速度命令由 EPCIO 送出速度命令至 Motor Drive

五、EPCIO 控制軟體處理多個中斷事件

控制系統涵蓋運動控制及 I/O 控制系統，在複雜的控制系統中，常需即時性的中斷需求，如要求 PC 下達命令、Error Counter Overflow、Over Travel、緊急停止、I/O 狀態變化等。EPCIO 板設計多個條件狀態發出中斷源；當這些中斷事件發生，對 CPU 發出中斷，CPU 需即時處理這些中斷源事件。以 ISA Bus 為例，PC 提供使用者使用的中斷號碼，通常只剩 2~3 個，又中斷事件的優先權皆高，控制軟體若要處理外界多中斷源，需達到相當高的 Hard Real Time 等級需求。因此軟體欲即時處理多中斷源需求，將是一大挑戰；EPCIO 板提供使用者選擇使用 PC 一個中斷向量號碼，並由 EPCIO 板 latch 多個中斷事件，控制軟體可讀取 EPCIO Latch 中斷源之暫存器，判斷發生的中斷源，再依其不同優先權處理多個即時性的中斷事件。

EPCIO 板發出中斷源條件如下：

I. Channel 0-6Error counter overflow

II. 第 0-8 軸各寫一 Preset 值，當各軸 Encoder 值大於或等於此 Presets 值，則產生中斷源。

III. 第 0-8 軸有 index latch，則產生中斷源。

IV. ADC module 提供 User 可寫入一 Preset 值，當 AD 轉換的值與 Preset 值比較，符合設定條件，則發出中斷。

>> ADC channel 0-7 voltage value 由小於至大於或等於 Preset value

由大於至小於或等於 Preset value

以上兩者皆可

>> ADC 完成一個 channel 轉換

>> ADC 指定多個 channel 完成轉換

V. Local IO 點前 8 點可發出中斷

>> Timer interrupt

>> Local IO DI0-DI7 interrupt

Falling edge/ Rising edge/ Both

>> Local Double Function DI0-DI6 interrupt

Falling edge/ Rising edge/ Both

VI. Remote IO 點，3 個 Slave 的各前 4 點可發出中斷

>> Slave 0 DI[0:3] is Falling edge/ Rising edge/ Both

>> Slave 1 DI[0:3] is Falling edge/ Rising edge/ Both

>> Slave 2 DI[0:3] is Falling edge/ Rising edge/ Both

VII. Master 與 Slave 傳輸失敗，發出中斷

>> Master-Slave transmission failure

>> Slave not ready

控制系統依照需求設定以上相關的中斷條件，當中斷源發生，軟體需判斷發生的中斷源，盡快完成 EOI 讓系統可接受下一個中斷源且針對個別中斷發生即時處理必要事件。

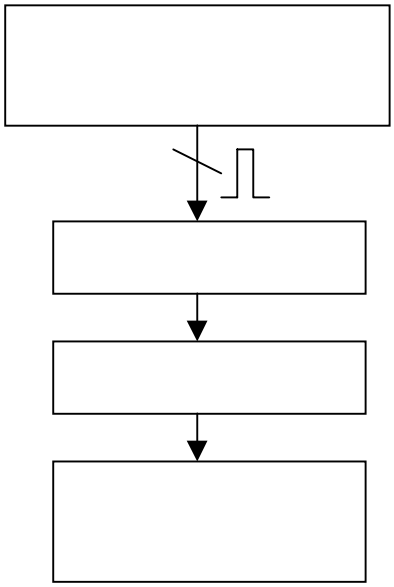


圖 11、EPCIO 控制軟體處理多個中斷事件

六、EPCIO 運動控制軟體架構

搭配 EPCIO 板，運動控制器軟體需涵蓋解譯、軌跡規劃、控制迴路維護及即時處理中斷源事件，工研院機械所根據運動控制的 Hard Real Time 需求，發展一即時性多工架構，因應不同特定功能，將其切分為獨立高度模組化的軟體模組(如解譯、運動控制、程序控制、定位控制等)，每一模組均可獨立執行其特定功能，且獨立運作不具相依性。在多工機制下，這些獨立模組依其功能有各自的優先權及執行一次之循環時間。

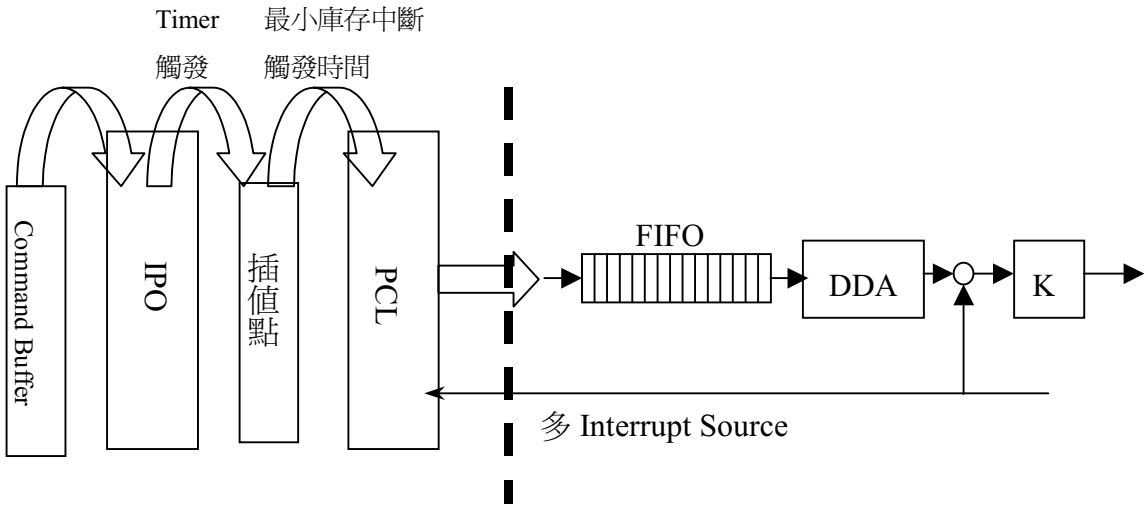


圖 12、EPCIO 運動控制軟體架構

圖 12 為機械所提出的運動控制運動架構，其中運動控制模組(IPO SIC)主要負責軌跡規劃，其輸入命令為運動指令，如點對點、直線、圓弧等，輸出則為每個 DDA 時間馬達的進給 PULSE 命令，其觸發方式採用純軟體的 Timer+Event 方式，其循環時間需大於或等於計算一筆插值點所需的時間，且須小於或等於 DDA Time，以確保有足夠快的時間產生新的運動插值點。而其將每次運算的運動插值點放在一 Queue Buffer，PCL SIC 受到 EPCIO FIFO 監督最小庫存中斷觸發，則將 IPO 放在 Queue 的插值點拿出 4 點，寫至 EPCIO 的 FIFO Stack，由前文分析，其插值精確度也增進 4 倍。

定位控制模組(PCL SIC)負責在 EPCIO 板發出 DDA 中斷時，將運動控制模組所得的 DDA pulse 命令，輸出至 EPCIO 板的 FIFO，且當 EPCIO 對 CPU 發出多個中斷信號，定位控制模組根據中斷源處理各個不同中斷事件。

七、EPCIO 即時運動控制程式庫

針對 EPCIO 板強大的功能，本部門提供一強大的即時運動控制程式庫，如圖 13，此函式庫提供使用者使用 EPCIO 板之一介面；使用者不需深入運動控制中複雜的軌跡規劃、定位控制、即時多工環境，透過此函式庫直接呼叫函式即可在圖控軟體或應用系統快速開發整合系統。此函式庫在運動控制上提供點對點、直線、圓、圓弧、螺旋等軌跡的定位控制；在規劃上接受梯形/S 型加減速曲線、進給速度設定、軟體過行程保護，另針對系統需求可設定連續路徑及動態強制調整速度。

在定位控制部分，提供使用者設定可接受定位誤差範圍、定位確認、齒輪齒隙、間隙補償等功能。

此函式庫在運動控制上有極強大的功能，也提供使用者相當大的設定彈性，當使用者輸入不合理或不予接受的設定值，各函數提供相對應的返回值回應使用者，但若使用者忽略相關設定，此函式庫也提供合理的預設值。其提供的函式及使用變數如(一)、(二)所列。

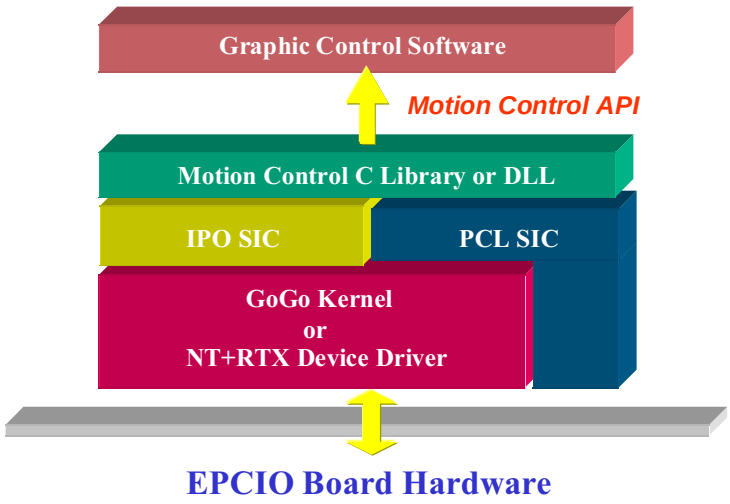


圖 13、運動控制函式庫架構

(一) 運動控制函式庫 Function List

指令功能	*函　　式*
設定/讀取 絕對、相對座標值	MCC_set_abs
	MCC_set_inc
	MCC_get_pos_type
直線補間差值	MCC_line
	int MCC_line_x
	int MCC_line_y
	int MCC_line_z
圓補間差值	MCC_circle_xy
	MCC_circle_yz
	MCC_circle_zx
	MCC_circle_xyuvw
	MCC_circle_xyuvw
	MCC_circle_zxuvw
圓弧補間差值	MCC_arc_xy
	MCC_arc_yz
	MCC_arc_zx
	MCC_arc_xyuvw
	MCC_arc_yzuvw
	MCC_arc_zxuvw
點對點同動補間	MCC_ptp
	MCC_ptp_x
	MCC_ptp_y
	MCC_ptp_z
	MCC_ptp_uvw
點對點不同動補間	MCC_ptp_asyn
連續、單步、脈衝 JOG	MCC_jog_c
	MCC_jog_i
	MCC_jog_s
運動暫停、持續、棄置功能	MCC_motion_hold
	MCC_motion_conti

	MCC_motion_abort
運動延遲	MCC_motion_delay
直線、圓弧、圓的進給速度設定	MCC_set_fspd
點對點進給速度設定	MCC_set_ptp_spd
連續路徑設定 / 取消	MCC_enable_quit
	MCC_disable_quit
	MCC_check_quit
定位確認設定 / 取消	MCC_enable_in_pos
	MCC_diable_in_pos
確認運動位置已進入運動命令目的點的定位誤差範圍內	MCC_check_in_pos

一般功能	***函　　式***
設定/讀取 Line、arc、ptp 同動運動各軸一致的加、減式速型式	MCC_set_acc_type
	MCC_get_acc_type
	MCC_set_dec_type
	MCC_get _dec_type
設定/讀取 ptp 不同動運動中各軸加減速型式	MCC_set_ptp_asyn_acc_type
	MCC_get_ptp_asyn_acc_type
	MCC_set_ptp_asyn_decc_type
	MCC_get_ptp_asyn_dec_type
設定/讀取 line, arc 及 ptp 同動運動中加、減速時間	MCC_set_acc_step
	MCC_get_acc_step
	MCC_set_dec_step
	MCC_get _dec_step
設定/讀取 不同步點對點運動的各軸加、減速時間	MCC_set_ptp_asyn_acc_step
	MCC_get_ptp_asyn_acc_step
	MCC_set_ptp_asyn_dec_step
	MCC_get_ptp_asyn_dec_step

設定/取消/確認 過行程檢查	MCC_set_check_OT
	MCC_get_check_OT
設定、讀取 DDA 時間	MCC_set_dda
	MCC_get_dda
DDA 最大速度設定	MCC_set_max_dda_spd
	MCC_get_max_dda_spd
設定/讀取 DDA 最大加速度	MCC_set_max_dda_acc
	MCC_get_max_dda_acc
設定/讀取 定位比例增益 (P gain)	MCC_set_P_gain
	MCC_get_P_gain
齒輪齒隙、間隙補償	MCC_set_comp
設定/取消/確認 運動空跑	MCC_enable_dry_run
	MCC_disable_dry_run
	MCC_check_dry_run
原點復歸	MCC_go_home
	MCC_check_homef
設定/讀取 line、circle、arc 速度強制	MCC_set_over_spd
	MCC_get_over_spd
設定/讀取 點對點同動、不同動運動的速度強制	MCC_set_ptp_ospd
	MCC_get_ptp_ospd
設定/讀取 各軸定位誤差範圍	MCC_set_inpos_tol
	MCC_get_inpos_tol
設定/取消 手輪運動	MCC_MPG_set
	MCC_MPG_disable
檢視目前機器是否已停止運動	MCC_check_stop

系統功能	*函　　式*
啟動運動控制程式庫的功能	MCC_init_motion
關閉運動控制程式庫功能	MCC_close_motion

清除系統中的錯誤記錄	MCC_clear_error
讀取目前機器運作是否發生錯誤或錯誤代碼	MCC_get_errcode
反應新設定的系統參數值	MCC_update_param

其它	***函　　式***
讀取各軸目前位置之直角座標值	MCC_get_cpos
讀取各軸目前位置之馬達座標值	MCC_get_ppos
讀取各軸 error counter 的值	MCC_get_errcnt

（二）可使用變數

1. 機構參數 **MCC_MACH_PARAM** *epcio_axis_param*[6];

內含：

type[6]：各軸運動型態(平移軸、轉動軸、主軸或未使用)。

pos2encoder_dir[6]：方向調整參數。

PPR[6]：馬達軸心每旋轉一圈迴授至位置閉迴路的 pulse 數。

RPM[6]：馬達最大安全轉速。

pitch[6]：導螺桿間隙值。

gear_ratio[6]：齒輪比。

high_limit[6]：邏輯原點至正方向極限開關（limit switch）的距離，單位為 mm。

low_limit[6]：邏輯原點至負方向極限開關（limit switch）的距離，單位為 mm。

HOT_offset[6]：正方向過行程軟體保護值。

LOT_offset[6]：負方向過行程軟體保護值。

sersor_on[6]：設定各軸 home sensor ON 時的訊號邏輯狀態。

index_on[6]: 設定各軸 index ON 時的訊號邏輯狀態。

home_dir[6]：設定各軸歸原點時第二、第三相位的運動方向；其中 bit1 為第二相位方向，設定值「0」表正方向「1」表負方向，bit2 則為第三相位的運動方向，設定值「0」表正方向「1」表負方向。

2. 運動參數

cart_max_speed : 設定最大速度，單位為 mm。

參、結論

隨著 CPU 運算速度的快速進展，EPCIO 板在 PC-Based 的軟體技術效益勢必發揮的更加淋漓盡致，當 CPU 運算速度更加提昇，伴隨 FIFO 機制，軌跡規劃的插值點精度將更加提昇；又在複雜控制系統的多即時性的中斷需求，也因 EPCIO latch 多中斷源機制而提高軟體處理多中斷源效益。最後提供一功能強大的即時運動控制程式庫以提供使用者一簡單好用的使用介面，更是我們致力追求的。

肆、參考文獻

1. “系統核心軟體設計報告書”，工研院機械所，范維如著，1997 年 12 月
2. 機電整合與運動控制-原理與單軸平台實例，高立圖書，施慶隆、李文猶編著，1997 年 6 月
3. ”EPCIO ASIC 設計規劃書”，工研院機械所，賴振國、陳文泉編著，1999 年 4 月
4. “運動控制程式庫設計報告書”，工研院機械所，黃美燕著，1999 年 6 月