IMP Series

驅動函式庫

參考手冊

版本 ：V.2.04

日期 ：2013.06

http://www.epcio.com.tw

目 錄

I. 介紹與說明

IMP Device Driver 可用來驅動利用 IMC (Intelligent Motion control Chip) 所設計開發的 IMP Series 運動控制平台。在 Windows XP/7 環境下提供驅動函式庫(IMCDriver.lib)與動態聯結函式庫(IMCDriver.dll)，使用者只需含入相對應的標頭檔 IMCDriver.h，呼叫相對應功能函式，即可驅動 IMP Series 運動控制平台。

驅動函式庫共有 154 個函式可供使用者呼叫，分成 9 大部份，分別驅動 IMP 運動控制平台上不同功能輸出或輸入：

▲ Global Control Interface 中斷及重置功能設定
▲ PGE Control Interface 設定位置脈波輸出控制
▲ Encode Counter Interface 設定編碼計數器輸入控制
▲ PCL Control Interface 設定硬體位置閉迴路
▲ Asynchrous Remote I/O Interface 設定遠端輸出入點控制
▲ Local I/O Control Interface 設定近端輸出入點控制
▲ ADC Control Interface 設定類比轉數位輸入控制
▲ DAC Control Interface 設定數位轉類比輸出控制
▲ Timer Control Interface 設定計時器控制

在函式使用上，有關驅動函式原型宣告及資料型態宣告部分均已定義於“IMCDriver.h”標頭檔內，相關常數部份則定義於”IMCDefine.h”標頭檔。使用者使用時必須含入此兩標頭檔內容。

範例程式部份乃使用 IMP Series 驅動函式庫所設計，主要針對各個功能模組作一使用說明，包含 PGE 模組脈波輸出，ENC 模組編碼輸入，PCL 硬體閉迴路控制，DAC 類比電壓輸出，ADC 類比電壓輸入，LIO 近端輸出入點控制，ARIO 遠端輸出入點控制，計時器及看門狗程式規劃等。

安裝程式會協助使用者把相關的檔案內容放至指定目錄中，使用者只需跟隨安裝步驟執行即可。

II. IMP 驅動函式庫（Device Driver Library）

II.1. Global Interface Control

II.1.1 IMC_GetInterruptCount()

void IMC_GetInterruptCount(WORD wCardIndex)	
Parameters	wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	中斷發生的計數值
Description	讀取中斷發生的計數值

II.1.2 IMC_OpenDevice()

BOOL IMC_OpenDevice(int nMode,WORD wCardIndex)	
Parameters	nMode 0：初始化 IMP 的 PCI 模式 1：初始化 IMP 的 Standalone 模式 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	true：Initial IMP 模組成功 false：Initial IMP 模組失敗
Description	初始化 IMP Series 運動控制平台。

II.1.3 IMC_CloseIfOpen()

void IMC_CloseIfOpen(WORD wCardIndex)	
Parameters	wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	結束 IMP 模組，本函式會關閉 IMP 模組內所有功能，若初始化時有設定中斷功能，亦會還原中斷向量。

II.1.4 IMC_GLB_ResetModule()

void IMC_GLB_ResetModule(int Module_no, WORD wCardIndex)	
Parameters	Module_no：Reset Module 編號 RESET_PGE：PGE Module RESET_ENC：Encoder counter Module RESET_RIO：Remote IO Module RESET_ADC：ADC Module RESET_LIO：Local IO Module RESET_PCL：PCL Module RESET_DAC：DAC Module RESET_ALL：All Modules

	RESET_TMR：Timer Control Modules wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	Reset 所選定的 IMP 模組，本函式為提供使用者以軟體設定方式重置 IMP 模組，IMP 可單獨重置各模組，亦可重置所有模組。

II.1.5 IMC_GLB_GetInterruptSource()

DWORD	***IMC_GLB_GetInterruptSource(IMCINT *source, WORD wCardIndex)***
Parameters	source 為提供使用者以讀取中斷訊號之 Module 編號；此為一結構變數，定義如下： typedef struct _IMC_INT { BYTE PGEINT; BYTE ENCINT; BYTE LDIOINT; BYTE TIMERINT; BYTE ADCINT; BYTE PCLINT; }IMCINT; 其中讀回值表發生中斷之功能模組編號，各狀態變數說明如下： source->PGEINT 表 PGE Module 觸發中斷之狀態 source->ENCINT 表 ENC Module 觸發中斷之狀態 source->LDIOINT 表 Local IO Module 觸發中斷之狀態 source->TIMERINT 表 Timer Module 觸發中斷之狀態 source-> ADCINT 表 ADC Module 觸發中斷之狀態 source-> PCLINT 表 PCL Module 觸發中斷之狀態 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	發生中斷之功能模組編號
Description	IMP 除了在 DAC Module，其它 Module 皆有不同的中斷訊號產生源，當硬體中斷產生時應先呼叫此函數判別中斷是由那一個 Module 所觸發，再至該 Module 所支援的 function 判斷中斷產生的源由。
See also	IMC_PGE_GetInterruptSource()， IMC_ENC_GetInterruptSource()， IMC_RIO_GetInterruptSource()， IMC_ADC_GetInterruptSource()， IMC_TMR_GetInterruptSource()， IMC_LIO_GetInterruptSource()。

II.1.6 IMC_GLB_SetInterruptMode()

void IMC_GLB_SetInterruptMode(int mode, WORD wCardIndex)	
Parameters	mode：interrupt mode INT_RISE_EDGE：interrupt rising edge trigger INT_FALL_EDGE：interrupt falling edge trigger INT_LEVEL_HIGH：interrupt level trigger high active INT_LEVEL_LOW：interrupt level trigger low active wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定中斷訊號產生時，其觸發中斷模式。

II.1.7 IMC_GLB_SetInterruptMask()

void IMC_GLB_SetInterruptMask(WORD MaskBit, WORD wCardIndex)	
Parameters	MaskBit：Mask Module 編號 IMC_PGE_INT_MASK：PGE Module IMC_ENC_INT_MASK：Encoder counter Module IMC_LIO_INT_MASK：Local IO Module IMC_TMR_INT_MASK：Timer Module IMC_ADC_INT_MASK：ADC IO Module IMC_PCL_INT_MASK：PCL Module IMC_ALL_INT_MASK：All Modules wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	Mask 所選定的 IMP 模組，本函式提供使用者以軟體設定方式遮閉 IMP 模組中斷功能，可單獨遮閉各模組或所有模組之中斷功能。

II.1.8 IMC_GLB_GetDeviceID()

int IMC_GLB_GetDeviceID(WORD wCardIndex)	
Parameters	wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	所使用的裝置名稱
Description	讀取目前所使用的裝置名稱

II.1.9 IMC_GLB_SetInterruptHostCPU()

void IMC_GLB_SetInterruptHostCPU(int Host, WORD wCardIndex)	
Parameters	Host：指定處理中斷的 CPU 0：Host PCI (中斷由 PC 處理) 1：Embedded IMC (中斷由 IMP 處理) wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定處理中斷服務程式的 CPU。

II.1.10 IMC_GLB_ResetPPC()

void	***IMC_GLB_ResetPPC(WORD wCardIndex)***
Parameters	wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	Reset CPU PowerPC440

II.1.11 IMC_GLB_SetInterruptPeriod()

void	***IMC_GLB_SetInterruptPeriod(WORD Period, WORD wCardIndex);***
Parameters	Period 設定中斷服務程式的中斷週期。 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定中斷服務程式的中斷週期。Interrupt Mode 為 Edge Mode 時才有意義

II.2. PGE Control Interface

II.2.1 IMC_PGE_GetInterruptSource()

DWORD IMC_PGE_GetInterruptSource(PGEINT *source, WORD wCardIndex)

Parameters

source 為提供 User 使用以讀取 PGE 中斷訊號之源由；此為一結構變數，定義如下：

```
typedef struct _PGE_INT
{
  BYTE FIFO0;
  BYTE FIFO1;
  BYTE FIFO2;
  BYTE FIFO3;
  BYTE FIFO4;
  BYTE FIFO5;
  BYTE FIFO6;
  BYTE FIFO7;
  BYTE CYCLE;
}PGEINT;
```

其中讀回值表發生中斷之中斷發生源訊號編碼，各狀態變數說明如下：

source->FIFO0：PGE channel 0 FIFO reach minimum stock
source->FIFO1：PGE channel 1 FIFO reach minimum stock
source->FIFO2：PGE channel 2 FIFO reach minimum stock
source->FIFO3：PGE channel 3 FIFO reach minimum stock
source->FIFO4：PGE channel 4 FIFO reach minimum stock
source->FIFO5：PGE channel 5 FIFO reach minimum stock
source->FIFO6：PGE channel 6 FIFO reach minimum stock
source->FIFO7：PGE channel 7 FIFO reach minimum stock
source->CYCLE：PGE Cycle Interrupt happened

wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5

Return value　發生中斷之中斷發生源訊號編碼

Description　讀取 PGE 中斷發生的原因，並清除中斷 Latch 值，等待下一次中斷發生。當硬體中斷發生後，可先經由 IMC_GLB_GetInterruptSource()判斷是否為 PGE 所發生，若是則呼叫本函式讀取中斷發生源。

See also　IMC_GLB_GetInterruptSource()

II.2.2 IMC_PGE_SetISRFunction()

void IMC_PGE_SetISRFunction(PGEISR myPGE_ISR, WORD wCardIndex)

Parameters　myPGE_ISR：User 自己撰寫的 PGE 中斷副程式之 Function

	Pointer wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定 User 自己撰寫的中斷副程式，但必須於呼叫於 IMC_OpenDevice()之前。

II.2.3 IMC_PGE_GetCurrentCommand()

void IMC_PGE_GetCurrentCommand(WORD pge_ch_no, long *pge_cmd, WORD wCardIndex)	
Parameters	pge_ch_no：指定 PGE channel 編號（0~7） pge_cmd：讀取的 Pulse command value。 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	讀取指定之 PGE channel 目前正在執行之 PGE pulse command 的值。

II.2.4 IMC_PGE_CheckFIFOEmpty()

void IMC_PGE_CheckFIFOEmpty(WORD pge_ch_no, WORD *flag, WORD wCardIndex)	
Parameters	pge_ch_no：指定 PGE channel 編號（0~7） flag：讀回 fifo 是否為空旗標值 0：FIFO not empty 1：FIFO empty wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	檢查目前指定之 PGE channel 之 FIFO 是否為空的狀態。

II.2.5 IMC_PGE_CheckFIFOFull()

void IMC_PGE_CheckFIFOFull(WORD pge_ch_no, WORD *flag, WORD wCardIndex)	
Parameters	pge_ch_no：指定 PGE channel 編號（0~7） flag：讀回 fifo 是否為滿旗標值 0：FIFO not full 1：FIFO full wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	檢查目前指定之 PGE channel 之 FIFO 是否為滿的狀態。

II.2.6 IMC_PGE_GetStockCount()

void IMC_PGE_GetStockCount(WORD pge_ch_no, WORD *wCount, WORD wCardIndex)	
Parameters	pge_ch_no：指定 PGE channel 編號（0~7）

	wCount：讀回 stock 內命令儲存筆數 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	讀取目前指定之 PGE channel FIFO 中儲存尚未被消化送出之命令筆數。

II.2.7 IMC_PGE_EnableOutABSwap()

void IMC_PGE_EnableOutABSwap(WORD pge_ch_no, WORD wSwap, WORD wCardIndex)	
Parameters	pge_ch_no：指定 PGE channel 編號（0~7） wInverse：指定輸出訊號是否對調 0：輸出 A/B 不對調 1：輸出 A/B 互相對調 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定 PGE channel 輸出脈波格式時；輸出腳 A 及 B 兩訊號線對調。Default 為不對調。

II.2.8 IMC_PGE_EnableOutAInverse()

void IMC_PGE_EnableOutAInverse(WORD pge_ch_no, WORD wInverse, WORD wCardIndex)	
Parameters	pge_ch_no：指定 PGE channel 編號（0~7） wInverse：指定輸出訊號是否反相 0：輸出不反相 1：輸出反相 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定 PGE channel 輸出脈波格式時；輸出腳 A 訊號線反相。Default 為不反相。

II.2.9 IMC_PGE_EnableOutBInverse()

void IMC_PGE_EnableOutBInverse(WORD pge_ch_no, WORD wInverse, WORD wCardIndex)	
Parameters	pge_ch_no：指定 PGE channel 編號（0~7） wInverse：指定輸出訊號是否反相 0：輸出不反相 1：輸出反相 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定 PGE channel 輸出脈波格式時；輸出腳 B 訊號線反相。Default 為不反相。

II.2.10 IMC_PGE_SetOutputFormat()

void IMC_PGE_SetOutputFormat(WORD pge_ch_no, WORD format, WORD wCardIndex)	
Parameters	pge_ch_no：指定 PGE channel 編號（0~7） format：表 PGE pulse output 格式設定值，可設定格式如下 PGE_FMT_PD：Pulse / Direction output format (default) PGE_FMT_CW：CW / CCW output format PGE_FMT_AB：Phase A / Phase B output format PGE_FMT_NO：Inhibit，no pulse output wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定 PGE channel 輸出脈波格式；當 IMP 輸出命令為脈波（pulse）型式時，可經由軟體設定不同型式之脈波輸出格式。

II.2.11 IMC_PGE_EnableStockInterrupt()

void IMC_PGE_EnableStockInterrupt(WORD FIFO_no, WORD wEnable, WORD wCardIndex)	
Parameters	FIFO_no：PGE FIFO channel 編號（0 ~ 7） wEnable：指定是否開啟中斷功能 0：關閉中斷功能 1：開啟中斷功能 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	開啟 PGE FIFO 最小儲存筆數中斷功能；選定的 PGE FIFOchannel 於固定時間讀取命令，若 FIFO 中剩餘之命令儲存筆數等於所設定的最小儲存筆數時，產生硬體 IRQ 中斷觸發。

II.2.12 IMC_PGE_EnableCycleInterrupt()

void IMC_PGE_EnableCycleInterrupt(WORD wEnable, WORD wCardIndex)	
Parameters	wEnable：指定是否開啟中斷功能 0：關閉中斷功能 1：開啟中斷功能 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	開啟 PGE 循環中斷功能，PGE 將在每個 IPO time 的固定週期自動產生硬體 IRQ 中斷觸發。

II.2.13 IMC_PGE_SetIPOTime()

void IMC_PGE_SetIPOTime(double ipotime, WORD wCardIndex)	
Parameters	ipotime：PGE IPO time，PGE 運作時之插值時間 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None

Description	設定 PGE 運作時的 cycle time，此函數會將 IPO time 轉換成為 PGE 內部的 Clock Divider 及 Clock Number 並寫入硬體，當啟動 PGE 控制功能後，IMC 將會固定時間自 FIFO 中讀取一筆命令，並經由 PGE 轉成脈波輸出。

II.2.14 IMC_PGE_SetStockThreshold()

void IMC_PGE_SetStockThreshold(WORD wThreshold, WORD wCardIndex)	
Parameters	wThreshold：最小 FIFO 儲存筆數設定值（1~63） wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定 PGE FIFO 中觸發中斷功能之最小儲存命令筆數，所有 PGE channel 均使用相同設定；設定後且經由 IMC_EnableStockInterrupt()函數啟動 FIFO 最小儲存筆數檢查中斷功能，則在 FIFO 剩餘筆數等於 wThreshold 時觸發硬體 IRQ 產生中斷。
See also	IMC_EnableStockInterrupt()

II.2.15 IMC_PGE_SendPulse()

void IMC_PGE_SendPulse(WORD pge_ch_no, long pulse, WORD wCardIndex)	
Parameters	pge_ch_no：指定 PGE channel 編號（0~7） pulse：pulse command value wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	將 pulse command 寫入指定之 PGE FIFO；每一筆 pulse command 所能送出的最大值與設定的 PGE Clock Number 數目相關。
See Also	IMC_PGE_SetClockNumber()，IMC_PGE_SetClockDivider()。

II.2.16 IMC_PGE_Start()

void IMC_PGE_Start(WORD wStart, WORD wCardIndex)	
Parameters	wStart：開啟或關閉 PGE Engine 0：關閉 1：開啟 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	啟動/停止 PGE 運作功能。

II.2.17 IMC_PGE_GetPulseCounter()

void IMC_PGE_GetPulseCounter(WORD ch, long* lCounter, WORD wCardIndex)	
Parameters	ch：指定 PGE channel 編號（0~7）

	lCounter：讀取的 Pulse Counter 值 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	PGE 實際輸出脈波數會記錄在 IMC 內部暫存器，透過本函式可讀取暫存器內的脈波值。
See Also	IMC_PGE_EnablePulseCounter()， IMC_PGE_DisablePulseCounter()， IMC_PGE_ClearPulseCounter()

II.2.18 IMC_PGE_EnablePulseCounter()

void	***IMC_PGE_EnablePulseCounter(WORD ch, WORD wEnable WORD wCardIndex)***
Parameters	ch：指定 PGE channel 編號（0~7） wEnable：開啟或關閉 Pulse Counter 功能 0：關閉 Pulse Counter 功能 1：開啟 Pulse Counter 功能 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	開啟 IMP 內部脈波輸出計數器之紀錄功能。
See Also	IMC_PGE_GetPulseCounter()，IMC_PGE_ClearPulseCounter()

II.2.19 IMC_PGE_ClearPulseCounter()

void	***IMC_PGE_ClearPulseCounter(WORD ch, WORD wClear, WORD wCardIndex)***
Parameters	ch：指定 PGE channel 編號（0~7） wClear：開啟或關閉清除功能 0：關閉清除功能 1：開啟清除功能 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	清除 IMP 內部脈波輸出計數器之值為零
See Also	IMC_PGE_GetPulseCounter()，IMC_PGE_EnablePulseCounter()

II.2.20 IMC_PGE_SetClockDivider()

void	***IMC_PGE_SetClockDivider(DWORD Divider, WORD wCardIndex)***
Parameters	Divider：PGE 內時脈除頻值，可設定範圍(0 ~ 2^{31}-1)，預設值為 0，除頻結果為 Fine Interpolation Clock Period。 Clock period = (divider + 1) * system clock period wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定 PGE 運作時，所使用的工作時脈週期(PGE Clock period)，其中 PGE Clock period 為(divider + 1)*System Clock (100MHz)。

II.2.21 IMC_PGE_GetClockDivider()

DWORD IMC_PGE_GetClockDivider(WORD wCardIndex)	
Parameters	wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	PGE 運作時，所使用的工作時脈週期(PGE Clock period)
Description	讀取 PGE 運作時，所使用的工作時脈週期(PGE Clock period)。

II.2.22 IMC_PGE_SetClockNumber()

void IMC_PGE_SetClockNumber(DWORD Number, WORD wCardIndex)	
Parameters	Number：每個插值時間(Interpolation time)所能送出的最大脈波數，可設定值範圍(0 ~ 2^{31}-1)，預設值為 0。 實際插值時間 IPO Time = (Number + 1) * Fine Interpolation Clock Period wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	此設定值將會影響每個 IPO Time 所能送出的最大脈波數。

II.2.23 IMC_PGE_GetClockNumber()

DWORD IMC_PGE_GetClockNumber(WORD wCardIndex)	
Parameters	wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	PGE 運作時，每個插值時間所能送出的最大脈波數
Description	讀取 PGE 運作時，每個插值時間所能送出的最大脈波數。

II.2.24 IMC_PGE_SetOutputFormat()

void IMC_PGE_SetOutputFormat(WORD Channel, WORD Format, WORD wCardIndex)	
Parameters	Channel：指定 PGE channel 編號（0~7） Format：表 PGE pulse output 格式設定值，可設定格式如下 0：Inhibit，no pulse output 1：Pulse / Direction output format (default) 2：CW / CCW output format 3：Phase A / Phase B output format wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定 PGE channel 輸出脈波格式。 PGE channel 的脈波(pulse)輸出格式，可經由軟體設定。

II.2.25 IMC_PGE_ClearPulseCounter()

void IMC_PGE_ClearPulseCounter(WORD Channel, WORD Clear, WORD wCardIndex)	
Parameters	Channel：指定 PGE channel 編號（0~5）

	Clear：IMP 內部脈波輸出計數器清除狀態： 0：不清除 1：清除 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	清除 IMP 內部脈波輸出計數器之值為零

II.2.26 IMC_PGE_GetIPOTime()

void IMC_PGE_GetIPOTime(WORD wCardIndex)	
Parameters	wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	PGE IPO time，PGE 運作時之插值時間
Description	讀取 PGE 運作時的 cycle time。

II.2.27 IMC_PGE_EraseFIFOCmd()

void IMC_PGE_EraseFIFOCmd(WORD Channel, WORD EraseNumber, WORD wCardIndex);	
Parameters	Channel：指定 PGE channel 編號（0~7） EraseNumber：欲刪除 FIFO 內之未執行命令筆數 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	本函式可移除 PGE FIFO 內已設定但未執行的命令。本函式可於命令輸出中途刪除 FIFO 內未執行之命令，最多一次可刪除 64 筆命令，但正執行中之該筆命令不受影響。

II.3. Encoder Counter Interface

II.3.1 IMC_ENC_GetInterruptSource()

DWORD	***IMC_ENC_GetInterruptSource(ENCINT *source, WORD wCardIndex)***
Parameters	source：為提供 User 使用以讀取 ENC 中斷訊號之源由；此為一結構變數，定義如下： typedef struct _ENC_INT { BYTE INDEX0; BYTE INDEX1; BYTE INDEX2; BYTE INDEX3; BYTE INDEX4; BYTE INDEX5; BYTE INDEX6; BYTE INDEX7; BYTE COMP0; BYTE COMP1; BYTE COMP2; BYTE COMP3; BYTE COMP4; BYTE COMP5; BYTE COMP6; BYTE COMP7; }ENCINT; 其中讀回值為發生中斷之發生源訊號編碼，各狀態變數說明如下： source->INDEX0：ENC channel 0 Index happened source->INDEX1：ENC channel 1 Index happened source->INDEX2：ENC channel 2 Index happened source->INDEX3：ENC channel 3 Index happened source->INDEX4：ENC channel 4 Index happened source->INDEX5：ENC channel 5 Index happened source->INDEX6：ENC channel 6 Index happened source->INDEX7：ENC channel 7 Index happened source->COMP0： ENC channel 0 compare value equal counter value source->COMP1： ENC channel 1 compare value equal counter value source->COMP2： ENC channel 2 compare value equal counter

	value source->COMP3： ENC channel 3 compare value equal counter value source->COMP4： ENC channel 4 compare value equal counter value source->COMP5： ENC channel 5 compare value equal counter value source->COMP6： ENC channel 6 compare value equal counter value source->COMP7： ENC channel 7 compare value equal counter value wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	發生中斷之中斷發生源訊號編碼
Description	讀取 Encoder counter channel 0～7 發出中斷的條件狀態，並清除中斷 Latch 值，等待下一次中斷發生。當硬體中斷發生後，可先經由 IMC_GLB_GetInterruptSource()判斷是否為 ENC 所發生，若是則呼叫本函式讀取中斷發生源。
See also	IMC_GLB_GetInterruptSource()

II.3.2 IMC_ENC_SetISRFunction()

void IMC_ENC_SetISRFunction(NCISR myENC_ISR, WORD wCardIndex)	
Parameters	myENC_ISR：User 自己撰寫的 ENC 中斷副程式之 Function Pointer wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定 User 自己撰寫的中斷副程式，但必須於呼叫於 IMC_OpenDevice()之前。

II.3.3 IMC_ENC_ReadCounter()

void IMC_ENC_ReadCounter(WORD enc_ch_no, long *lCounter, WORD wCardIndex)	
Parameters	enc_ch_no：encoder counter channel number（0~7） lCounter：讀取 Encoder counter 值 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	讀取 encoder counter channel 目前計數值。

II.3.4 IMC_ENC_ReadLatchCounter()

void IMC_ENC_ReadLatchCounter(WORD enc_ch_no, long *lLatch, WORD wCardIndex)	
Parameters	enc_ch_no：encoder counter channel number（0~7） lLatch：讀取 latch 的 Encoder counter 值

	wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	讀取因設定中斷條件而產生觸發儲存的 encoder counter value。

II.3.5 IMC_ENC_GetIndexStatus()

void IMC_ENC_GetIndexStatus(WORD enc_ch_no, WORD *wStatus, WORD wCardIndex)	
Parameters	enc_ch_no：encoder counter channel number（0~7）
	wStatus：讀取 Encoder 的 Index 狀態值
	wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	讀取目前 Encode Counter Index 訊號的 HIGH / LOW 狀態。

II.3.6 IMC_ENC_SetCounter()

void IMC_ENC_SetCounter(WORD enc_ch_no, long lValue, WORD wCardIndex)	
Parameters	enc_ch_no：encoder counter channel number（0~7）
	lValue：Encoder counter 初值
	wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定 Encoder counter 之初值。此函式可預先設定好 Encoder counter 的起始計數值。

II.3.7 IMC_ENC_SetComparator()

void IMC_ENC_SetComparator(WORD enc_ch_no, long lValue, WORD wCardIndex)	
Parameters	enc_ch_no：encoder counter channel number（0~7）
	lValue：Encoder counter 比較器設定值
	wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定 Encoder counter 比較值，當 counter 累積值等於設定比較值時，搭配 IMC_ENC_EnableCompareInterrupt()可產生一硬體中斷觸發訊號。此觸發訊號可用來觸發 LIO 快速送出一預先設定好的輸出命令。
See also	IMC_ENC_EnableComparatorInterrupt()

II.3.8 IMC_ENC_EnableIndexInterrupt()

void IMC_ENC_EnableIndexInterrupt(WORD enc_ch_no, WORD wEnable, WORD wCardIndex)	
Parameters	enc_ch_no：encoder counter channel number（0~7）
	wEnable：開啟或關閉中斷功能

	0：關閉中斷功能 1：開啟中斷功能 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	開啟 Encoder counter Index 中斷觸發功能。

II.3.9 IMC_ENC_EnableComparatorInterrupt()

void IMC_ENC_EnableComparatorInterrupt(WORD enc_ch_no, WORD wEnable, WORD wCardIndex)	
Parameters	enc_ch_no：encoder counter channel number（0~7） wEnable：開啟或關閉中斷功能 0：關閉中斷功能 1：開啟中斷功能 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	開啟 Encoder counter 比較值中斷觸發功能。
See also	IMC_ENC_SetComparator()

II.3.10 IMC_ENC_SetInputRate()

void IMC_ENC_SetInputRate(WORD enc_ch_no, WORD rate, WORD wCardIndex)	
Parameters	enc_ch_no：encoder counter channel number（0~7） rate：Encoder multiplier rate ENC_RATE_X0：Multiplier rate to be 4 (Default) ENC_RATE_X1：Multiplier rate to be 1 ENC_RATE_X2：Multiplier rate to be 2 ENC_RATE_X4：Multiplier rate to be 4 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定各 Encoder counter 訊號解碼倍率。Encoder 解碼倍率必須在 Encoder 輸入格式為 A/B Phase 時方為有效。本函式必須搭配 IMC_ENC_SetInputFormat()設定為 A/B Phase 輸入。
See also	IMC_ENC_SetInputFormat()

II.3.11 IMC_ENC_SetInputFormat()

void IMC_ENC_SetInputFormat(WORD enc_ch_no, WORD wFormat, WORD wCardIndex)	
Parameters	enc_ch_no：encoder counter channel number（0~7） wFormat：編碼器輸入訊號格式 ENC_FMT_AB：Input type is quadratic or A/B phase ENC_FMT_CW：Input type is CW / CCW ENC_FMT_PD：Input type is Pulse / Direction

	ENC_FMT_NO：Input Disable(Default) wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定各 Encoder counter 輸入訊號型態，此函式必需搭配硬體實際訊號設定，當輸入訊號為馬達編碼器迴授訊號時，請參考馬達或驅動器設定，當接一般手輪時請設定為 A/B Phase 輸入(內定為 A/B Phase 輸入)。

II.3.12 IMC_ENC_EnableInAInverse()

void IMC_ENC_EnableInAInverse(WORD enc_ch_no, WORD wInverse, WORD wCardIndex)	
Parameters	enc_ch_no：encoder counter channel number（0~7） wInverse：開啟或關閉輸入訊號反相 0：輸入訊號不反相 1：輸入訊號反相 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定各 encoder counter channel 輸入訊號之 inA 腳位反相。Default 無反相。

II.3.13 IMC_ENC_EnableInBInverse()

void IMC_ENC_EnableInBInverse(WORD enc_ch_no, WORD wInverse, WORD wCardIndex)	
Parameters	enc_ch_no：encoder counter channel number（0~7） wInverse：開啟或關閉輸入訊號反相 0：輸入訊號不反相 1：輸入訊號反相 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定各 encoder counter channel 輸入訊號之 inB 腳位反相。Default 無反相。

II.3.14 IMC_ENC_EnableInCInverse()

void IMC_ENC_EnableInCInverse(WORD enc_ch_no, WORD wInverse, WORD wCardIndex)	
Parameters	enc_ch_no：encoder counter channel number（0~7） wInverse：開啟或關閉輸入訊號反相 0：輸入訊號不反相 1：輸入訊號反相 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定各 encoder counter channel 輸入訊號 inC 腳位反相。

Default 無反相。

II.3.15 IMC_ENC_EnableInABSwap()

void IMC_ENC_EnableInABSwap(WORD enc_ch_no, WORD wSwap, WORD wCardIndex)	
Parameters	enc_ch_no：encoder counter channel number（0~7） wSwap：開啟或關閉輸入訊號 A/B 對調 0：輸入訊號不對調 1：輸入訊號對調 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定指定 encoder channel 輸入訊號之 inA 及 inB 腳位，在訊號進入 counter 前經訊號交換前置處理。Default 無交換。

II.3.16 IMC_ENC_SetCounterLatchMode()

void IMC_ENC_SetCounterLatchMode(WORD enc_ch_no, WORD mode, WORD wCardIndex)	
Parameters	enc_ch_no：encoder counter channel number（0~7） mode：Encoder counter latch 觸發模式設定 ENC_TRIG_FIRST：第一次滿足觸發條件即 latch 不再變動 ENC_TRIG_LAST：觸發條件滿足時即 latch value 且隨條件滿足即再 latch 新值 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定各 encoder counter latch 觸發模式。本函式必須搭配 IMC_ENC_SetIndexLatchEnable()，IMC_ENC_SetExternalLatch Enable()，設定觸發訊號源。則當觸發訊號動作時便可根據觸發模式 Latch Encoder counter 值。
See also	IMC_ENC_SetIndexLatchEnable()， IMC_ENC_SetExternalLatchEnable()

II.3.17 IMC_ENC_SetIndexLatchSource()

void IMC_ENC_SetIndexLatchSource(WORD enc_ch_no, WORD source, WORD wCardIndex)	
Parameters	enc_ch_no：encoder counter channel number（0~7） source:encoder latch condition,共有 8 種觸發源可做為 counter latch 的條件，設定時可同時取多個條件的聯集。 NO_TRIG_ENC： No trigger source selected INDEX0_TRIG_ENC：Encoder channel 0 Index signal INDEX1_TRIG_ENC：Encoder channel 1 Index signal INDEX2_TRIG_ENC：Encoder channel 2 Index signal INDEX3_TRIG_ENC：Encoder channel 3 Index signal

	INDEX4_TRIG_ENC：Encoder channel 4 Index signal INDEX5_TRIG_ENC：Encoder channel 5 Index signal INDEX6_TRIG_ENC：Encoder channel 6 Index signal INDEX7_TRIG_ENC：Encoder channel 7 Index signal wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定 Encoder 觸發訊號源，此觸發訊號源可用來 Latch Encoder counter 之值。本函式必須搭配 IMC_ENC_SetCounterLatchMode()使用
See also	IMC_ENC_SetExternalLatchEnable()， IMC_ENC_SetCounterLatchMode()

II.3.18 IMC_ENC_SetExternalLatchSource()

void	***IMC_ENC_SetExternalLatchSource(WORD enc_ch_no, WORD source, WORD wCardIndex)***
Parameters	enc_ch_no：encoder counter channel number（0~7） source：External Plus / Minus Limit Input Signal，共有 16 種觸發源可做為 counter latch 的條件，設定時可同時取多個條件的聯集。 NO_TRIG_ENC：No trigger source selected OTP0_TRIG_ENC：Plus Limit Channel 0 Input OTP1_TRIG_ENC：Plus Limit Channel 1 Input OTP2_TRIG_ENC：Plus Limit Channel 2 Input OTP3_TRIG_ENC：Plus Limit Channel 3 Input OTP4_TRIG_ENC：Plus Limit Channel 4 Input OTP5_TRIG_ENC：Plus Limit Channel 5 Input OTP6_TRIG_ENC：Plus Limit Channel 6 Input OTP7_TRIG_ENC：Plus Limit Channel 7 Input OTN0_TRIG_ENC：Minus Limit Channel 0 Input OTN1_TRIG_ENC：Minus Limit Channel 1 Input OTN2_TRIG_ENC：Minus Limit Channel 2 Input OTN3_TRIG_ENC：Minus Limit Channel 3 Input OTN4_TRIG_ENC：Minus Limit Channel 4 Input OTN5_TRIG_ENC：Minus Limit Channel 5 Input OTN6_TRIG_ENC：Minus Limit Channel 6 Input OTN7_TRIG_ENC：Minus Limit Channel 7 Input wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定 Encoder 觸發訊號源，此觸發訊號源可用來 Latch Encoder counter 之值。
See also	IMC_ENC_SetIndexLatchEnable()， IMC_ENC_SetCounterLatchMode ()

II.3.19 IMC_ENC_ClearCounter()

void IMC_ENC_ClearCounter(WORD enc_ch_no, WORD wClear, WORD wCardIndex)	
Parameters	enc_ch_no：encoder counter channel number（0~7） wClear：開啟或關閉清除功能 0：關閉清除功能 1：開啟清除功能 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	清除 Encoder counter value。

II.3.20 IMC_ENC_ClearLatchCounter()

void IMC_ENC_ClearLatchCounter(WORD enc_ch_no, WORD wClear, WORD wCardIndex)	
Parameters	enc_ch_no：encoder counter channel number（0~7） wClear：開啟或關閉清除功能 0：關閉清除功能 1：開啟清除功能 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	清除 Encoder Latch Counter value。

II.3.21 IMC_ENC_StartCounter()

void IMC_ENC_StartCounter(WORD enc_ch_no, WORD wStart, WORD wCardIndex)	
Parameters	enc_ch_no：encoder counter channel number（0~7） wStart：開啟或關閉編碼器功能 0：關閉編碼器功能 1：開啟編碼器功能 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	啟動 ENC Counter 記錄功能。

II.4. PCL Control

II.4.1 IMC_PCL_GetInterruptSource()

DWORD IMC_PCL_GetInterruptSource(PCLINT *source, WORD wCardIndex)

Parameters

source：為提供 User 使用以讀取 PCL 中斷訊號之源由；此為一結構變數，定義如下：

```
typedef struct _PCL_INT
{
  BYTE OV0;
  BYTE OV1;
  BYTE OV2;
  BYTE OV3;
  BYTE OV4;
  BYTE OV5;
  BYTE OV6;
  BOOL OV7;
}PCLINT;
```

其中讀回值為發生中斷之發生源訊號編碼，各狀態變數說明如下：

source->OV0：PCL channel 0 error counter overflow
source->OV1：PCL channel 1 error counter overflow
source->OV2：PCL channel 2 error counter overflow
source->OV3：PCL channel 3 error counter overflow
source->OV4：PCL channel 4 error counter overflow
source->OV5：PCL channel 5 error counter overflow
source->OV6：PCL channel 6 error counter overflow
source->OV7：PCL channel 7 error counter overflow

wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5

Return value

發生中斷之中斷發生源訊號編碼

Description

讀取 PCL 中斷發生的原因，並清除中斷 Latch 值，等待下一次中斷發生。當硬體中斷發生後，可先經由 IMC_GLB_GetInterruptSource()判斷是否為 PCL 所發生，若是則呼叫本函式讀取中斷發生源。

See also

IMC_GLB_GetInterruptSource()

II.4.2 IMC_PCL_SetISRFunction()

void IMC_PCL_SetISRFunction(PCLISR myPCL_ISR, WORD wCardIndex)

Parameters

myPCL_ISR：User 自己撰寫的 PCL 中斷副程式之 Function Pointer

	wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定 User 自己撰寫的中斷副程式，但必須於呼叫於 IMC_OpenDevice()之前。

II.4.3 IMC_PCL_ReadErrorCounter()

void	***IMC_PCL_GetErrorCounter(WORD channel, short *nError WORD wCardIndex)***
Parameters	channel：Error counter channel number 0 ~ 7 nError：Error counter 值 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	讀取各軸位置命令輸出與 Encoder 回授之命令誤差計數值。呼叫本函數時必須先啟動 IMC_PCL_EnableCloseLoop()
See also	IMC_PCL_EnableCloseLoop()

II.4.4 IMC_PCL_ReadErrorVoltage()

void	***IMC_PCL_GetErrorVoltage(WORD channel, short *nVoltage, WORD wCardIndex)***
Parameters	channel：Error counter channel number 0 ~ 7 nVoltage：Error voltage 值 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	讀取各軸位置控制迴路輸出之電壓命令值。呼叫本函數時必須先啟動 IMC_PCL_EnableCloseLoop()
See also	IMC_PCL_EnableCloseLoop()

II.4.5 IMC_PCL_SetErrorThreshold()

void	***IMC_PCL_SetErrorThreshold(WORD channel, int PlusThreshold, int MinusThreshold, WORD wCardIndex)***
Parameters	channel：Error counter channel number 0 ~ 7 nPlusThreshold：Error counter 正轉溢位值 nMinusThreshold：Error counter 反轉溢位值 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定各軸閉迴路位置誤差的溢位門檻值。當 Error counter 累積值等於設定之溢位門檻值時，PCL 將產生一硬體中斷觸發訊號，並將 Error counter 計數值歸零。

II.4.6 IMC_PCL_SetPGain()

void	***IMC_PCL_SetPGain(WORD channel, WORD wPgain, WORD wCardIndex)***

Parameters	channel：PCL channel number 0 ~ 7 wPgain：closed-loop Proportional gain (0~127) wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定閉迴路控制軸之 Proportion gain 值。閉迴路增益值可藉由設定一比例項(Kp）增益值，IMC 內定硬體會將設定的 Gain 值與位置誤差之乘積再除以 16，因此所輸出的比例補償值 = Kp / 16。

II.4.7 IMC_PCL_SetIGain()

void IMC_PCL_SetIGain(WORD channel, WORD wIgain, WORD wCardIndex)	
Parameters	channel：PCL channel number 0 ~ 7 wIgain：closed-loop Integral gain (0~127) wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定閉迴路控制軸之 Integral gain 值。閉迴路增益值可藉由設定一積分項(Ki）增益值，及一積分時間常數，IMC 硬體會在這固定的積分時間內將誤差值累加並與 Ki 值相乘後與比例及微分補償項相加。
See also	IMC_PCL_SetIClockDivider()

II.4.8 IMC_PCL_SetDGain()

void IMC_PCL_SetDGain(WORD channel, WORD wDgain, WORD wCardIndex)	
Parameters	channel：PCL channel number 0 ~ 7 wDgain：closed-loop Derivative gain (0~127) wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定閉迴路控制軸之 Derivative gain 值。閉迴路增益值可藉由設定一微分項(Kd)增益值，及一微分時間常數，IMC 硬體會在這固定的微分時間內將誤差值差分並與 Kd 值相乘後與比例及積分補償項相加。
See also	IMC_PCL_SetDClockDivider()

II.4.9 IMC_PCL_SetFGain()

void IMC_PCL_SetFGain(WORD channel, WORD wFgain, WORD wCardIndex)	
Parameters	channel：PCL channel number 0 ~ 7 wDgain：closed-loop Feed-forward gain (0~127) wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None

Description	設定閉迴路控制軸之 Feed-Forward gain 值。閉迴路增益值可藉由設定一前饋項(Kf)增益值，IMC 硬體會將命令值差分並與 Kf 值相乘後與回授補償(比例積分微分)項相加。

II.4.10 IMC_PCL_SetIClockDivider()

void	***IMC_PCL_SetIClockDivider(DWORD dwDivider, WORD wCardIndex)***
Parameters	dwDivider：PCL 迴路 I-Gain 積分取樣時間(2^{16} ~ 2^{32}) wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定閉迴路控制軸之 Integral gain 之積分取樣時間。設定範圍 2^{16}~2^{32} 個系統時脈時間。

II.4.11 IMC_PCL_SetDClockDivider()

void	***IMC_PCL_SetDClockDivider(DWORD dwDivider, WORD wCardIndex)***
Parameters	dwDivider：PCL 迴路 D-Gain 微分取樣時間 (0 ~ 2^{32}) wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定閉迴路控制軸之 Derivative gain 之微分取樣時間。設定範圍 0~2^{32} 個系統時脈時間。

II.4.12 IMC_PCL_ClearErrorCounter()

void	***IMC_PCL_ClearErrorCounter(WORD channel, WORD wClear, WORD wCardIndex)***
Parameters	channel：PCL channel number 0 ~ 7 wClear：開啟或關閉清除 Error Counter 功能 0：關閉清除 Error Counter 功能 1：開啟清除 Error Counter 功能 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	清除 error counter 計數值以及溢位狀態。

II.4.13 IMC_PCL_EnableCloseLoop()

void	***IMC_PCL_EnableCloseLoop(WORD channel, WORD wEnable, WORD wCardIndex)***
Parameters	channel：PCL channel number 0 ~ 7 wEnable：開啟或關閉 PCL 功能 0：關閉 PCL 功能 1：開啟 PCL 功能 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	開啟 PCL 硬體閉迴路控制功能。

II.4.14 IMC_PCL_SetErrorCounterMode()

void IMC_PCL_SetErrorCounterMode(WORD channel, WORD wMode, WORD wCardIndex)	
Parameters	channel：PCL channel number 0 ~ 7 wMode：Error Counter 運作模式 0：使用軟體閉迴路功能 1：使用硬體閉迴路功能 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定 PCL 閉迴路輸出控制模式，當設定軟體閉迴路模式時，硬體閉迴路只負責計算位置誤差，但不輸出命令至 DAC 模組。當設定為硬體閉迴路模式時，則計算後的命令會經由 PID+FF 後輸出至 DAC 模組。本函式設定完成後，必須呼叫 IMC_PCL_EnableCloseLoop()啟動硬體閉迴路控制功能。

II.4.15 IMC_PCL_SetFeedbackMode()

void IMC_PCL_SetFeedbackMode(WORD Channel, WORD Mode, WORD wCardIndex)	
Parameters	Channel：設定回授 channel 編號（0~7） Mode：設定回授模式： 0：回授編碼器訊號 1：回授 ADC 訊號 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定 PCL 回授訊號來源。

II.4.16 IMC_PCL_GetIClockDivider()

DWORD IMC_PCL_GetIClockDivider(WORD wCardIndex)	
Parameters	wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	讀取 PCL 的 I-Gain 積分取樣時間 ($2^{16} \sim 2^{32}$)
Description	讀取閉迴路控制軸之 Integral gain 之積分取樣時間。設定範圍 $2^{16} \sim 2^{32}$ 個系統時脈時間。

II.4.17 IMC_PCL_GetDClockDivider()

DWORD IMC_PCL_GetDClockDivider(WORD wCardIndex)	
Parameters	wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	讀取 PCL 的 D-Gain 微分取樣時間 ($0 \sim 2^{32}$)
Description	讀取閉迴路控制軸之 Derivative gain 之微分取樣時間。設定範圍 0~2^{32} 個系統時脈時間。

II.4.18 IMC_PCL_SetTemperatureCommand()

void IMC_PCL_SetTemperatureCommand(WORD Channel, float Voltage, WORD wCardIndex)	
Parameters	Channel：設定 PCL 溫度控制 channel 編號（0~7） Voltage：設定 PCL 溫度控制電壓值 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定 PCL 溫度控制電壓值

II.4.19 IMC_PCL_GetTemperatureCommand()

float IMC_PCL_GetTemperatureCommand(WORD Channel, WORD wCardIndex)	
Parameters	Channel：設定 PCL 溫度控制 channel 編號（0~7） wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	PCL 溫度控制電壓值
Description	讀取 PCL 溫度控制電壓值

II.4.20 IMC_PCL_SetFrictionCompensation()

void IMC_PCL_SetFrictionCompensation(WORD Channel, WORD Compensation, WORD wCardIndex)	
Parameters	Channel：設定 PCL 摩擦力補償 channel 編號（0~7） Voltage：設定 PCL 摩擦力補償值 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定 PCL 摩擦力補償值

II.4.21 IMC_PCL_GetFrictionCompensation()

WORD IMC_PCL_GetFrictionCompensation(WORD Channel, WORD wCardIndex);	
Parameters	Channel：讀取 PCL 摩擦力補償 channel 編號（0~7） wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	PCL 摩擦力補償值
Description	讀取 PCL 摩擦力補償值

II.4.22 IMC_PCL_EnablePlusOverflowInterrupt()

void IMC_PCL_EnablePlusOverflowInterrupt(WORD Channel, WORD Enable, WORD wCardIndex)	
Parameters	Channel：PCL channel 編號（0~7） Enable：狀態變數，可設定格式如下 0：Disable 1：Enable wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5

Return Value	None
Description	開啟指定 channel 的 error counter overflow 時產生中斷觸發之功能。當位置命令與 Encoder 位置的正誤差量超過 Error Counter 計數器所能容許範圍時，Error Counter 產生 Overflow 中斷通知，並自動輸出 0V 電壓命令值。

II.4.23 IMC_PCL_EnableMinusOverflowInterrupt()

void IMC_PCL_EnableMinusOverflowInterrupt(WORD Channel, WORD Enable, WORD wCardIndex)	
Parameters	Channel：PCL channel 編號（0~7） Enable：狀態變數，可設定格式如下 0：Disable 1：Enable wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	開啟指定 channel 的 error counter overflow 時產生中斷觸發之功能。當位置命令與 Encoder 位置的負誤差量超過 Error Counter 計數器所能容許範圍時，Error Counter 產生 Overflow 中斷通知，並自動輸出 0V 電壓命令值

II.5. Asynchrous Remote IO

II.5.1 IMC_ARIO_GetInputValue()

void	***IMC_ARIO_GetInputValue(WORD set, WORD slave, DWORD *value, WORD wCardIndex)***
Parameters	set：Remote IO set number selection RIO_SET0：Remote I/O Set 0 slave：Slave number selection in a set RIO_SLAVE0：DI0~ DI15 in the Slave0 (Slave number is 0~31) value：variable name to read back the Digital input data wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	對於 Remote IO 指定的 slave 讀取目前 digital input 的訊號狀態。

II.5.2 IMC_ARIO_SetOutputValue()

void	***IMC_ARIO_SetOutputValue(WORD set, WORD slave, DWORD value, WORD wCardIndex)***
Parameters	set：Remote IO set number selection RIO_SET0：Remote I/O Set 0 slave：Slave number selection in a set RIO_SLAVE0：DO0~ DO15 in the Slave0 (Slave number is 0~31) value：16 bits output data wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定指定之 slave 編號之 16 bit digital output 輸出訊號狀態值。

II.5.3 IMC_ARIO_SetClockDivider()

void	***IMC_ARIO_SetClockDivider(WORD set, WORD divider, WORD wCardIndex)***
Parameters	set：Remote IO set number selection RIO_SET0：Remote I/O Set 0 divider：Remote IO clock divider（0~255） wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定 Remote IO 傳輸資料的時脈頻率。換算後之傳輸頻率為 System clock 除以(2*divider +1)，內定設定值為 255。

II.5.4 IMC_ARIO_EnableSlaveControl()

void IMC_ARIO_EnableSlaveControl(WORD set, WORD slave, WORD wEnable, WORD wCardIndex)	
Parameters	set：Remote I/O set number selection RIO_SET0：Remote I/O Set 0 slave：Remote I/O slave number (0 ~ 31) wEnable：開啟或關閉 Slave 傳輸功能 0：關閉傳輸功能 1：開啟傳輸功能 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	開啟指定的 Remote IO Slave 功能。Slave 功能開啟後，I/O 模組才開始做傳送與接收。

II.5.5 IMC_ARIO_GetMasterStatus()

void IMC_ARIO_GetMasterStatus(WORD Set, WORD *Status, WORD wCardIndex)	
Parameters	Set：Remote I/O set number selection RIO_SET0：Remote I/O Set 0 Status：目前 Remote I/O Master 端資料傳輸的狀態；1 代表 Remote I/O 的資料傳輸訊號正常；0 則否 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	讀取目前 Remote I/O Master 端資料傳輸的狀態

II.5.6 IMC_ARIO_GetSlaveStatus()

void IMC_ARIO_GetSlaveStatus(WORD Set, DWORD *Status, WORD wCardIndex)	
Parameters	Set：Remote I/O set number selection RIO_SET0：Remote I/O Set 0 Status：目前 Remote I/O Slave 端資料傳輸的狀態；0 代表 Remote I/O Slave 端資料傳輸的訊號正常；1 則否(bit 0~bit 31 分別代表 Slave 0~Slave 31 的狀態) wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	讀取目前 Remote I/O Master 端是否收到來自 Slave 資料傳輸的狀態

II.6. Local IO Control

II.6.1 IMC_LIO_GetInterruptSource()

DWORD IMC_LIO_GetInterruptSource(LIOINT *source, WORD wCardIndex)

Parameters source：為提供 User 使用以讀取 LIO 中斷訊號之源由；此為一結構變數，定義如下：

```
typedef struct _LIO_INT
{
  BYTE OTP0;
  BYTE OTP1;
  BYTE OTP2;
  BYTE OTP3;
  BYTE OTP4;
  BYTE OTP5;
  BYTE OTP6;
  BYTE OTP7;
  BYTE OTN0;
  BYTE OTN1;
  BYTE OTN2;
  BYTE OTN3;
  BYTE OTN4;
  BYTE OTN5;
  BYTE OTN6;
  BYTE OTN7;
  BYTE HOME0;
  BYTE HOME1;
  BYTE HOME2;
  BYTE HOME3;
  BYTE HOME4;
  BYTE HOME5;
  BYTE HOME6;
  BYTE HOME7;
}LIOINT;
```

其中讀回值為發生中斷之中斷發生源訊號編碼，各狀態變數說明如下：

source-> OTP0：第 0 軸正極限觸發中斷
source-> OTP1：第 1 軸正極限觸發中斷
source-> OTP2：第 2 軸正極限觸發中斷
source-> OTP3：第 3 軸正極限觸發中斷
source-> OTP4：第 4 軸正極限觸發中斷
source-> OTP5：第 5 軸正極限觸發中斷
source-> OTP6：第 6 軸正極限觸發中斷

	source-> OTP7：第 7 軸正極限觸發中斷 source-> OTN0：第 0 軸負極限觸發中斷 source-> OTN1：第 1 軸負極限觸發中斷 source-> OTN2：第 2 軸負極限觸發中斷 source-> OTN3：第 3 軸負極限觸發中斷 source-> OTN4：第 4 軸負極限觸發中斷 source-> OTN5：第 5 軸負極限觸發中斷 source-> OTN6：第 6 軸負極限觸發中斷 source-> OTN7：第 7 軸負極限觸發中斷 source-> HOME0：第 0 軸 HOME 點觸發中斷 source-> HOME1：第 1 軸 HOME 點觸發中斷 source-> HOME2：第 2 軸 HOME 點觸發中斷 source-> HOME3：第 3 軸 HOME 點觸發中斷 source-> HOME4：第 4 軸 HOME 點觸發中斷 source-> HOME5：第 5 軸 HOME 點觸發中斷 source-> HOME6：第 6 軸 HOME 點觸發中斷 source-> HOME7：第 7 軸 HOME 點觸發中斷 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return value	發生中斷之中斷發生源訊號編碼
Description	讀取 LIO 中斷發生的原因，並清除中斷 Latch 值，等待下一次中斷發生。當硬體中斷發生後，可先經由 IMC_GLB_GetInterruptSource() 判斷是否為 LIO 所發生，若是則呼叫本函式讀取中斷發生源。
See also	IMC_GLB_GetInterruptSource()

II.6.2 IMC_LIO_SetISRFunction()

void IMC_LIO_SetISRFunction(LIOISR myLIO_ISR, WORD wCardIndex)	
Parameters	myLIO_ISR：User 自己撰寫的 LIO 中斷副程式之 Function Pointer wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定 User 自己撰寫的中斷副程式，但必須於呼叫於 IMC_OpenDevice()之前。

II.6.3 IMC_LIO_GetPlusLimitLDIInput()

void IMC_LIO_GetPlusLimitLDIInput(DWORD *input, WORD wCardIndex)	
Parameters	input：正極限 Channel 0~7 輸入狀態值 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	讀取正極限 LIO_OTP0 ~ LIO_OTP7 數位輸入訊號值。

II.6.4 IMC_LIO_GetPlusLimitStatus()

void IMC_LIO_GetPlusLimitStatus(WORD point, WORD *wStatus, WORD wCardIndex)	
Parameters	point：正極限輸入點編號 LIO_OTP0：第 0 軸正極限輸入點 LIO_OTP1：第 1 軸正極限輸入點 LIO_OTP2：第 2 軸正極限輸入點 LIO_OTP3：第 3 軸正極限輸入點 LIO_OTP4：第 4 軸正極限輸入點 LIO_OTP5：第 5 軸正極限輸入點 LIO_OTP6：第 6 軸正極限輸入點 LIO_OTP7：第 7 軸正極限輸入點 wStatus：正極限狀態值 0：無溢位 1：發生溢位 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	讀取設定軸是否超過正方向之行程極限。若是則機臺有可能發生撞機的危險，使用者應立即作緊急處理。

II.6.5 IMC_LIO_SetPlusLimitTriggerMode()

void IMC_LIO_SetPlusLimitTriggerMode(WORD wPoint, WORD wMode, WORD wCardIndex)	
Parameters	wPoint：正極限輸入點編號 LIO_OTP0：Plus Limit Point 0 LIO_OTP1：Plus Limit Point 1 LIO_OTP2：Plus Limit Point 2 LIO_OTP3：Plus Limit Point 3 LIO_OTP4：Plus Limit Point 4 LIO_OTP5：Plus Limit Point 5 LIO_OTP6：Plus Limit Point 6 LIO_OTP7：Plus Limit Point 7 wMode：Plus Limit Trigger Mode LIO_INT_NO：關閉觸發功能 LIO_INT_RISE：輸入上升緣觸發功能 LIO_INT_FALL：輸入下降緣觸發功能 LIO_INT_LEVEL：輸入轉態觸發功能 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	此函式可單獨設定各個正極限輸入點的中斷觸發功能。

II.6.6 IMC_LIO_EnablePlusLimitInterrupt()

void IMC_LIO_EnablePlusLimitInterrupt(WORD wPoint, WORD wEnable, WORD wCardIndex)	
Parameters	wPoint：正極限輸入點編號 LIO_OTP0：Plus Limit Point 0 LIO_OTP1：Plus Limit Point 1 LIO_OTP2：Plus Limit Point 2 LIO_OTP3：Plus Limit Point 3 LIO_OTP4：Plus Limit Point 4 LIO_OTP5：Plus Limit Point 5 LIO_OTP6：Plus Limit Point 6 LIO_OTP7：Plus Limit Point 7 wEnable：開啟或關閉正極限中斷功能 0：關閉正極限中斷功能 1：開啟正極限中斷功能 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	開啟正極限中斷觸發輸出。呼叫本函數前請先呼叫 IMC_LIO_SetPlusLimitTriggerMode() 設定中斷型式
See also	IMC_LIO_SetPlusLimitTriggerMode()

II.6.7 IMC_LIO_GetMinusLimitLDIInput()

void IMC_LIO_GetMinusLimitLDIInput(DWORD *wInput, WORD wCardIndex)	
Parameters	wInput：負極限 Channel 0~7 輸入狀態值 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	讀取負極限 LIO_OTN0~ LIO_OTN7 數位輸入訊號值。

II.6.8 IMC_LIO_GetMinusLimitStatus()

void IMC_LIO_GetMinusLimitStatus(WORD wPoint, WORD *wStatus, WORD wCardIndex)	
Parameters	wPoint：負極限輸入點編號 LIO_OTN0：第 0 軸負極限輸入點 LIO_OTN1：第 1 軸負極限輸入點 LIO_OTN2：第 2 軸負極限輸入點 LIO_OTN3：第 3 軸負極限輸入點 LIO_OTN4：第 4 軸負極限輸入點 LIO_OTN5：第 5 軸負極限輸入點 LIO_OTN6：第 6 軸負極限輸入點 LIO_OTN7：第 7 軸負極限輸入點 wStatus：負極限狀態值

	0：無溢位 1：發生溢位 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	讀取設定軸是否超過負方向之行程極限。若是則機臺有可能發生撞機的危險，使用者應立即作緊急處理。

II.6.9 IMC_LIO_SetMinusLimitTriggerMode()

void IMC_LIO_SetMinusLimitTriggerMode(WORD wPoint, WORD wMode, WORD wCardIndex)	
Parameters	wPoint：負極限輸入點編號 LIO_OTN0：第 0 軸負極限輸入點 LIO_OTN1：第 1 軸負極限輸入點 LIO_OTN2：第 2 軸負極限輸入點 LIO_OTN3：第 3 軸負極限輸入點 LIO_OTN4：第 4 軸負極限輸入點 LIO_OTN5：第 5 軸負極限輸入點 LIO_OTN6：第 6 軸負極限輸入點 LIO_OTN7：第 7 軸負極限輸入點 wMode：Minus Limit Trigger Mode LIO_INT_NO：關閉觸發功能 LIO_INT_RISE：輸入上升緣觸發功能 LIO_INT_FALL：輸入下降緣觸發功能 LIO_INT_LEVEL：輸入轉態觸發功能 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	此函式可單獨設定各個負極限輸入點的中斷觸發功能。

II.6.10 IMC_LIO_EnableMinusLimitInterrupt()

void IMC_LIO_EnableMinusLimitInterrupt(WORD wPoint, WORD wEnable, WORD wCardIndex)	
Parameters	wPoint：負極限輸入點編號 LIO_OTN0：第 0 軸負極限輸入點 LIO_OTN1：第 1 軸負極限輸入點 LIO_OTN2：第 2 軸負極限輸入點 LIO_OTN3：第 3 軸負極限輸入點 LIO_OTN4：第 4 軸負極限輸入點 LIO_OTN5：第 5 軸負極限輸入點 LIO_OTN6：第 6 軸負極限輸入點 LIO_OTN7：第 7 軸負極限輸入點 wEnable：開啟或關閉負極限中斷功能 0：關閉負極限中斷功能

	1：開啟負極限中斷功能 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	開啟負極限中斷觸發輸出。呼叫本函數前請先呼叫 IMC_LIO_SetMinusLimitTriggerMode()設定中斷型式
See also	IMC_LIO_SetMinusLimitTriggerMode()

II.6.11 IMC_LIO_GetHomeSensorLDIInput()

void IMC_LIO_GetHomeSensorLDIInput(DWORD *wInput, WORD wCardIndex)	
Parameters	wInput：HOME 點輸入 Channel 0~7 輸入狀態值 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	讀取 HOME 點輸入 LIO_HOME0 ~ LIO_HOME7 數位輸入訊號值。

II.6.12 IMC_LIO_GetHomeSensorStatus()

void IMC_LIO_GetHomeSensorStatus(WORD wPoint, WORD *wStatus, WORD wCardIndex)	
Parameters	wPoint：HOME 點編號 LIO_HOME0：第 0 軸 HOME 點輸入點 LIO_HOME1：第 1 軸 HOME 點輸入點 LIO_HOME2：第 2 軸 HOME 點輸入點 LIO_HOME3：第 3 軸 HOME 點輸入點 LIO_HOME4：第 4 軸 HOME 點輸入點 LIO_HOME5：第 5 軸 HOME 點輸入點 LIO_HOME6：第 6 軸 HOME 點輸入點 LIO_HOME7：第 7 軸 HOME 點輸入點 wStatus：HOME 點狀態值 0：無觸發 1：HOME 點觸發 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	讀取設定軸 HOME 點狀態。

II.6.13 IMC_LIO_SetHomeSensorTriggerMode()

void IMC_LIO_SetHomeSensorTriggerMode(WORD wPoint, WORD wMode, WORD wCardIndex)	
Parameters	wPoint：HOME 點編號 LIO_HOME0：第 0 軸 HOME 點輸入點 LIO_HOME1：第 1 軸 HOME 點輸入點 LIO_HOME2：第 2 軸 HOME 點輸入點

	LIO_HOME3：第 3 軸 HOME 點輸入點 LIO_HOME4：第 4 軸 HOME 點輸入點 LIO_HOME5：第 5 軸 HOME 點輸入點 LIO_HOME6：第 6 軸 HOME 點輸入點 LIO_HOME7：第 7 軸 HOME 點輸入點 wMode：Home Sensor Trigger Mode LIO_INT_NO：關閉觸發功能 LIO_INT_RISE：輸入上升緣觸發功能 LIO_INT_FALL：輸入下降緣觸發功能 LIO_INT_LEVEL：輸入轉態觸發功能 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	此函式可單獨設定各個 HOME 點的中斷觸發功能。

II.6.14 IMC_LIO_EnableHomeSensorInterrupt()

void IMC_LIO_EnableHomeSensorInterrupt(WORD wPoint, WORD wEnable)	
Parameters	wPoint：HOME 點編號 LIO_HOME0：第 0 軸 HOME 點輸入點 LIO_HOME1：第 1 軸 HOME 點輸入點 LIO_HOME2：第 2 軸 HOME 點輸入點 LIO_HOME3：第 3 軸 HOME 點輸入點 LIO_HOME4：第 4 軸 HOME 點輸入點 LIO_HOME5：第 5 軸 HOME 點輸入點 LIO_HOME6：第 6 軸 HOME 點輸入點 LIO_HOME7：第 7 軸 HOME 點輸入點 wEnable：開啟或關閉 HOME 點觸發中斷功能 0：關閉 HOME 點觸發中斷功能 1：開啟 HOME 點觸發中斷功能 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	開啟 HOME 點觸發中斷輸出。呼叫本函數前請先呼叫 IMC_LIO_SetHomeSensorTriggerMode()設定中斷型式
See also	IMC_LIO_SetHomeSensorTriggerMode()

II.6.15 IMC_LIO_SetServoOn()

void IMC_LIO_SetServoOn(WORD wChannel, WORD wCardIndex)	
Parameters	wChannel：伺服驅動開關接點編號 LIO_SVO0：第 0 軸伺服驅動開關接點 LIO_SVO1：第 1 軸伺服驅動開關接點 LIO_SVO2：第 2 軸伺服驅動開關接點 LIO_SVO3：第 3 軸伺服驅動開關接點

	LIO_SVO4：第 4 軸伺服驅動開關接點 LIO_SVO5：第 5 軸伺服驅動開關接點 LIO_SVO6：第 6 軸伺服驅動開關接點 LIO_SVO7：第 7 軸伺服驅動開關接點 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	開啟指定軸之伺服驅動接點功能。本接點可連接馬達驅動器的伺服驅動輸入接點，當呼叫本函式設定後，指定軸將可接受位置命令或速度命令之輸入。當呼叫初始化函式設定後，內定狀態為關閉伺服驅動功能。
See alse	IMC_LIO_SetServoOff()

II.6.16 IMC_LIO_SetServoOff()

void IMC_LIO_SetServoOff(WORD wChannel, WORD wCardIndex)

Parameters	wChannel：伺服驅動開關接點編號 LIO_SVO0：第 0 軸伺服驅動開關接點 LIO_SVO1：第 1 軸伺服驅動開關接點 LIO_SVO2：第 2 軸伺服驅動開關接點 LIO_SVO3：第 3 軸伺服驅動開關接點 LIO_SVO4：第 4 軸伺服驅動開關接點 LIO_SVO5：第 5 軸伺服驅動開關接點 LIO_SVO6：第 6 軸伺服驅動開關接點 LIO_SVO7：第 7 軸伺服驅動開關接點 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	關閉指定軸之伺服驅動接點功能。本接點可關閉馬達驅動器的伺服驅動輸入接點，當呼叫本函式設定後，指定軸之馬達驅動器將不再接受來自於控制器的位置命令或速度命令之輸入。當呼叫初始化函式設定後，內定狀態為關閉伺服驅動輸入功能。
See also	IMC_LIO_SetServoOn()

II.6.17 IMC_LIO_SetLedLightOn()

void IMC_LIO_SetLedLightOn(WORD wPoint, WORD wEnable, WORD wCardIndex)

Parameters	wPoint：LED 輸出接點編號 LIO_LED0：第 0 點 LED 輸出點 LIO_LED1：第 1 點 LED 輸出點 LIO_LED2：第 2 點 LED 輸出點 LIO_LED3：第 3 點 LED 輸出點 LIO_LED4：第 4 點 LED 輸出點 LIO_LED5：第 5 點 LED 輸出點 LIO_LED6：第 6 點 LED 輸出點

	LIO_LED7：第 7 點 LED 輸出點 wEnable：開啟或關閉 LED 輸出功能 0：關閉 1：開啟 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	開啟指定 LED 點輸出功能。本接點連接至 IMP 運動控制平台上 LED0~LED7 輸出接點。

II.6.18 IMC_LIO_SetLedTriggerSource()

void IMC_LIO_SetLedTriggerSource(WORD wPoint, WORD wSource, WORD wCardIndex)	
Parameters	wPoint：伺服驅動開關接點編號 LIO_LED0：第 0 點 LED 輸出點 LIO_LED1：第 1 點 LED 輸出點 LIO_LED2：第 2 點 LED 輸出點 LIO_LED3：第 3 點 LED 輸出點 LIO_LED4：第 4 點 LED 輸出點 LIO_LED5：第 5 點 LED 輸出點 LIO_LED6：第 6 點 LED 輸出點 LIO_LED7：第 7 點 LED 輸出點 wSource：LED Trigger Source LED_SOURCE_COMP0：觸發源由第 0 組編碼器比較器功能 LED_SOURCE_COMP1：觸發源由第 1 組編碼器比較器功能 LED_SOURCE_COMP2：觸發源由第 2 組編碼器比較器功能 LED_SOURCE_COMP3：觸發源由第 3 組編碼器比較器功能 LED_SOURCE_COMP4：觸發源由第 4 組編碼器比較器功能 LED_SOURCE_COMP5：觸發源由第 5 組編碼器比較器功能 LED_SOURCE_COMP6：觸發源由第 6 組編碼器比較器功能 LED_SOURCE_COMP7：觸發源由第 7 組編碼器比較器功能 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	此函式可單獨設定某一個點的觸發功能，觸發條件可同時設定多個編碼器比較器功能。

II.6.19 IMC_LIO_EnableLedTrigger()

void IMC_LIO_EnableLedTrigger(WORD wPoint, WORD wEnable, WORD wCardIndex)	
Parameters	wPoint：LED 輸出接點編號 LIO_LED0：第 0 點 LED 輸出點 LIO_LED1：第 1 點 LED 輸出點 LIO_LED2：第 2 點 LED 輸出點

	LIO_LED3：第 3 點 LED 輸出點 LIO_LED4：第 4 點 LED 輸出點 LIO_LED5：第 5 點 LED 輸出點 LIO_LED6：第 6 點 LED 輸出點 LIO_LED7：第 7 點 LED 輸出點 wEnable：開啟或關閉 LED 觸發功能 0：關閉 LED 觸發功能 1：開啟 LED 觸發功能 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	LED 輸出接點可以規劃為條件式硬體觸發輸出點，此函式可單獨啟動某一個點的條件式硬體觸發功能，初值設定所有點為關閉狀態。

II.6.20 IMC_LIO_SetLedTriggerValue()

void IMC_LIO_SetLedTriggerValue(WORD wPoint, WORD wValue, WORD wCardIndex)	
Parameters	wPoint：LED 輸出接點編號 LIO_LED0：第 0 點 LED 輸出點 LIO_LED1：第 1 點 LED 輸出點 LIO_LED2：第 2 點 LED 輸出點 LIO_LED3：第 3 點 LED 輸出點 LIO_LED4：第 4 點 LED 輸出點 LIO_LED5：第 5 點 LED 輸出點 LIO_LED6：第 6 點 LED 輸出點 LIO_LED7：第 7 點 LED 輸出點 wValue：條件式硬體觸發輸出狀態 0：條件式硬體觸發時輸出 Low 1：條件式硬體觸發時輸出 High wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	LED 輸出接點可規劃為條件式硬體觸發輸出點，此函式可單獨啟動某一點的條件式硬體觸發值，初值設定所有點為 0。

II.6.21 IMC_LIO_SetLedTriggerPeriod()

void IMC_LIO_SetLedTriggerPeriod(WORD wPeriod, WORD wCardIndex)	
Parameters	wPeriod：條件式硬體觸發輸出時間 0~65535 個 System Clock wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	LED 輸出接點可以規劃為條件式硬體觸發輸出點，此函式可設定條件式硬體觸發的輸出時間，初值設定為 0。

II.6.22 IMC_LIO_SetMotionEnable()

void	***IMC_LIO_SetMotionEnable(WORD wEnable, WORD wCardIndex)***
Parameters	wEnable：開啟或關閉位置命令及電壓命令輸出功能 0：關閉位置命令及電壓命令輸出功能 1：開啟位置命令及電壓命令輸出功能 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	開啟控制模組上位置命令及電壓命令輸出功能。本函式設定後，輸出功能將被開啟。當呼叫初始化函式設定後，內定狀態為關閉輸出功能。

II.6.23 IMC_LIO_EnableServoOnOff()

void	***IMC_LIO_EnableServoOnOff(WORD Channel, WORD Enable, WORD wCardIndex)***
Parameters	Channel：伺服驅動開關接點編號： LIO_SVO0：第 0 軸伺服驅動開關接點 LIO_SVO1：第 1 軸伺服驅動開關接點 LIO_SVO2：第 2 軸伺服驅動開關接點 LIO_SVO3：第 3 軸伺服驅動開關接點 LIO_SVO4：第 4 軸伺服驅動開關接點 LIO_SVO5：第 5 軸伺服驅動開關接點 LIO_SVO6：第 6 軸伺服驅動開關接點 LIO_SVO7：第 7 軸伺服驅動開關接點 Enable：LIO 狀態變數，可設定格式如下： 0：Disable 1：Enable wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	開啟或關閉指定軸之伺服驅動接點功能。本接點可連接馬達驅動器的伺服驅動輸入接點，當 Enable 本函式設定後，指定軸將可接受位置命令或速度命令之輸入。

II.6.24 IMC_LIO_SetServoTriggerMode()

void	***IMC_LIO_SetServoTriggerMode(WORD Channel, WORD Mode, WORD wCardIndex)***
Parameters	Channel：digital input 中斷觸發 LIO 接點編號 (LIO_SVO0~LIO_SVO7) Mode：LIO 中斷觸發型態： LIO_INT_RISE：Rising edge trigger (default) LIO_INT_FALL：Falling edge trigger LIO_INT_LEVEL：Level change trigger

	wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定 LIO 中具中斷型態之 Local digital input 接點，中斷觸發型態為上緣觸發或是下緣觸發或是轉態觸發。

II.6.25 IMC_LIO_EnablePlusLimit()

void IMC_LIO_EnablePlusLimit(WORD Channel, WORD Enable, WORD wCardIndex)	
Parameters	Channel：正極限輸入點編號 LIO_OTP0：第 0 軸正極限輸入點 LIO_OTP1：第 1 軸正極限輸入點 LIO_OTP2：第 2 軸正極限輸入點 LIO_OTP3：第 3 軸正極限輸入點 LIO_OTP4：第 4 軸正極限輸入點 LIO_OTP5：第 5 軸正極限輸入點 LIO_OTP6：第 6 軸正極限輸入點 LIO_OTP7：第 7 軸正極限輸入點 Enable：正極限狀態變數，可設定格式如下： 0：Disable 1：Enable wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	開啟設定軸正方向之行程極限輸入功能

II.6.26 IMC_LIO_EnableMinusLimit()

void IMC_LIO_EnableMinusLimit(WORD Channel, WORD Enable, WORD wCardIndex)	
Parameters	Channel：負極限輸入點編號 LIO_OTN0：第 0 軸負極限輸入點 LIO_OTN1：第 1 軸負極限輸入點 LIO_OTN2：第 2 軸負極限輸入點 LIO_OTN3：第 3 軸負極限輸入點 LIO_OTN4：第 4 軸負極限輸入點 LIO_OTN5：第 5 軸負極限輸入點 LIO_OTN6：第 6 軸負極限輸入點 LIO_OTN7：第 7 軸負極限輸入點 Enable：正極限狀態變數，可設定格式如下： 0：Disable 1：Enable wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	開啟設定軸負方向之行程極限輸入功能

II.6.27 IMC_LIO_EnableHomeSensor()

void IMC_LIO_EnableHomeSensor(WORD Channel, WORD Enable, WORD wCardIndex)	
Parameters	Channel：HOME 點編號 LIO_HOME0：第 0 軸 HOME 點輸入點 LIO_HOME1：第 1 軸 HOME 點輸入點 LIO_HOME2：第 2 軸 HOME 點輸入點 LIO_HOME3：第 3 軸 HOME 點輸入點 LIO_HOME4：第 4 軸 HOME 點輸入點 LIO_HOME5：第 5 軸 HOME 點輸入點 LIO_HOME6：第 6 軸 HOME 點輸入點 LIO_HOME7：第 7 軸 HOME 點輸入點 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	開啟設定軸之 HOME 點 Sensor 輸入功能

II.6.28 IMC_LIO_EnableLedLight()

void IMC_LIO_EnableLedLight(WORD Channel, WORD Enable, WORD wCardIndex)	
Parameters	Channel：LED 輸出接點編號 LIO_LED0：第 0 點 LED 輸出點 LIO_LED1：第 1 點 LED 輸出點 LIO_LED2：第 2 點 LED 輸出點 LIO_LED3：第 3 點 LED 輸出點 LIO_LED4：第 4 點 LED 輸出點 LIO_LED5：第 5 點 LED 輸出點 LIO_LED6：第 6 點 LED 輸出點 LIO_LED7：第 7 點 LED 輸出點 Enable：開啟或關閉 LED 輸出功能 0：Disable 1：Enable wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	開啟指定 LED 點輸出功能

II.6.29 IMC_LIO_GetLedLightOutput()

void IMC_LIO_GetLedLightOutput(DWORD *dwLDIState, WORD wCardIndex)	
Parameters	dwLDIState：所有 LED 的輸出狀態 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None

Description	讀取所有 LED 的輸出狀態

II.6.30 IMC_LIO_SetLedLightOutput()

void IMC_LIO_SetLedLightOutput(DWORD dwLDOState, WORD wCardIndex)	
Parameters	dwLDOState：所有 LED 的輸出狀態 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定所有 LED 的輸出狀態

II.6.31 IMC_LIO_GetLedLightStatus()

void IMC_LIO_GetLedLightStatus(WORD Channel, WORD *State, WORD wCardIndex)	
Parameters	Channel：LED 輸出接點編號 LIO_LED0：第 0 點 LED 輸出點 LIO_LED1：第 1 點 LED 輸出點 LIO_LED2：第 2 點 LED 輸出點 LIO_LED3：第 3 點 LED 輸出點 LIO_LED4：第 4 點 LED 輸出點 LIO_LED5：第 5 點 LED 輸出點 LIO_LED6：第 6 點 LED 輸出點 LIO_LED7：第 7 點 LED 輸出點 State：LED 輸出狀態 0：Disable 1：Enable wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	讀取指定 LED 點輸出狀態

II.6.32 IMC_LIO_EnablePrdy()

void IMC_LIO_EnablePrdy(WORD Enable, WORD wCardIndex)	
Parameters	Enable：啟動 Prdy 功能 0：關閉 Prdy 功能 1：開啟 Prdy 功能 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	開啟 Prdy 功能

II.6.33 IMC_LIO_GetEmgcStopStatus()

void IMC_LIO_GetEmgcStopStatus(WORD *pwStatus, WORD wCardIndex)	
Parameters	pwStatus：LIO 緊急停止狀態

	0：Disable 1：Enable wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	讀取 LIO 緊急停止狀態

II.7. ADC IO Control

II.7.1 IMC_ADC_GetInterruptSource()

DWORD IMC_ADC_GetInterruptSource(ADCINT *source, WORD wCardIndex)

Parameters

source 為提供 User 使用以讀取 ADC 中斷訊號之源由；此為一結構變數，定義如下：

```
typedef struct _ADC_INT
{
  BYTE COMP0;
  BYTE COMP1;
  BYTE COMP2;
  BYTE COMP3;
  BYTE COMP4;
  BYTE COMP5;
  BYTE COMP6;
  BYTE COMP7;
}ADCINT;
```

其中讀回值為發生中斷之中斷發生源訊號編碼，各狀態變數說明如下：

source->COMP0： ADC channel 0 comparator interrupt
source->COMP1： ADC channel 1 comparator interrupt
source->COMP2： ADC channel 2 comparator interrupt
source->COMP3： ADC channel 3 comparator interrupt
source->COMP4： ADC channel 4 comparator interrupt
source->COMP5： ADC channel 5 comparator interrupt
source->COMP6： ADC channel 6 comparator interrupt
source->COMP7： ADC channel 7 comparator interrupt

wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5

Return value: 發生中斷之中斷發生源訊號編碼

Description: 讀取 ADC 中斷發生的原因，並清除中斷 Latch 值，等待下一次中斷發生。當硬體中斷發生後，可先經由 IMC_GLB_GetInterruptSource()判斷是否為 ADC 所發生，若是則呼叫本函式讀取中斷發生源。

See also: IMC_GLB_GetInterruptSource()

II.7.2 IMC_ADC_SetISRFunction()

void IMC_ADC_SetISRFunction(ADCISR myADC_ISR, WORD wCardIndex)

Parameters

myADC_ISR：User 自己撰寫的 ADC 中斷副程式之 FunctionPointer

	wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定 User 自己撰寫的中斷副程式，但必須於呼叫於 IMC_OpenDevice()之前。

II.7.3 IMC_ADC_GetInputVoltage()

void IMC_ADC_GetInputVoltage(WORD channel, float *fVoltage, WORD wCardIndex)	
Parameters	channel：ADC channel selection 0 ~ 7 fVoltage：ADC channel 直流電壓輸入值 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	讀取指定的 ADC channel 輸入之直流電壓值，若 ADC 指定為「Unipolar」則有效值為 0 ~ 10 V 精度為 0.61 mV，若 ADC 設定為「Bipolar」則電壓有效值為 -5 V ~ +5 V。

II.7.4 IMC_ADC_SetCompareVoltage()

void IMC_ADC_SetCompareVoltage(WORD channel, float dfVoltage, WORD wCardIndex)	
Parameters	channel：ADC channel selection（0 ~ 7） dfVoltage：ADC channel compared voltage value (-5V ~ +5V) wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定 ADC channel 在 Bipolar 模式下輸入電壓比較值，本函式不提供 Unipolar 模式下電壓比較功能。設定本函式後必須再設定 IMC_ADC_SetCompareMode()函式，則當該 ADC channel 輸入電壓與比較型式條件成立後，可產生 ADC 中斷觸發訊號。
See also	IMC_ADC_SetCompareMode()。

II.7.5 IMC_ADC_SetCompareMode()

void IMC_ADC_SetCompareMode(WORD channel, WORD wMode, WORD wCardIndex)	
Parameters	channel：ADC channel selection（0 ~ 7） wMode：ADC channel 比較型式 ADC_COMP_NO：關閉 ADC 輸入電壓比較功能 ADC_COMP_RISE：ADC 輸入電壓由小到大，並通過比較值 ADC_COMP_FALL：ADC 輸入電壓由大到小，並通過比較值 ADC_COMP_LEVEL：ADC 輸入電壓值改變，並通過比較值 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定 ADC channel 電壓比較型式，則當比較條件成立便會觸發硬體中斷訊號。

See also	IMC_ADC_SetCompareVoltage()

II.7.6 IMC_ADC_SetConverterMode()

void	***IMC_ADC_SetConverterMode(WORD wMode, WORD wCardIndex)***
Parameters	mode：轉換模式設定 0：bipolar/differential converter mode 1：unipolar/differential converter mode 2：bipolar/single end converter mode 3：unipolar/single end converter mode wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定 ADC 電壓轉換模式為雙極性或單極性/差動式或單端式。

II.7.7 IMC_ADC_EnableChannel()

void	***IMC_ADC_EnableChannel(WORD channel , WORD wEnable, WORD wCardIndex)***
Parameters	channel：ADC channel number 0 ~ 7 wEnable：開啟或關閉指定 AD Channel 轉換功能 0：關閉 AD 轉換功能 1：開啟 AD 轉換功能 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	開啟 ADC channel 輸入電壓類比轉數位功能。本函式設定完成後必須呼叫 IMC_ADC_StartConverter()函式，啟動 ADC 轉換功能。
See also	IMC_ADC_StartConverter()

II.7.8 IMC_ADC_StartConverter()

void	***IMC_ADC_StartConverter(WORD wEnable, WORD wCardIndex)***
Parameters	wEnable：開啟或關閉 AD 轉換功能 0：關閉 AD 轉換功能 1：開啟 AD 轉換功能 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	啟動 ADC 進行類比/數位電壓值轉換，設定本函式後即開始進行類比/數位轉換。本函式必須配合 IMC_ADC_EnableChannel()函式使用。
See also	IMC_ADC_EnableChannel()

II.7.9 IMC_ADC_GetCompareVoltage()

float	***IMC_ADC_GetCompareVoltage(WORD Channel, WORD wCardIndex);***
Parameters	Channel ADC channel 編號（0 ~ 7）

	wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	ADC channel compared voltage value (-5V ~ +5V)
Description	讀取 ADC channel 在 Bipolar 模式下輸入電壓比較值。

II.7.10 IMC_ADC_GetCompareMode()

WORD IMC_ADC_GetCompareMode(WORD Channel, WORD wCardIndex)

Parameters	Channel　ADC channel 編號（0 ~ 7）
	wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	讀取 ADC channel 比較型式： ADC_COMP_NO：關閉 ADC 輸入電壓比較功能 ADC_COMP_RISE：ADC 輸入電壓由小到大，並通過比較值 ADC_COMP_FALL：ADC 輸入電壓由大到小，並通過比較值 ADC_COMP_LEVEL：ADC 輸入電壓值改變，並通過比較值
Description	讀取 ADC channel 電壓比較型式，則當比較條件成立便會觸發硬體中斷訊號。

II.8. DAC IO Control

II.8.1 IMC_DAC_SetOutputVoltage()

void IMC_DAC_SetOutputVoltage(WORD channel, float fVoltage, WORD wCardIndex)	
Parameters	channel：DAC channel number 0 ~ 7 fVoltage：類比輸出電壓 (-10V ~ 10 V) wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定 DAC channel 輸出電壓值。DAC 輸出電壓模式可呼叫 IMC_DAC_SelectSource()設定為軟體直接規劃。設定完成後便可呼叫本函式直接規劃 DAC 輸出。
See also	IMC_DAC_SelectSource()

II.8.2 IMC_DAC_SetTriggerVoltage()

void IMC_DAC_SetTriggerVoltage(WORD channel, float fVlotage, WORD wCardIndex)	
Parameters	channel：DAC channel number 0 ~ 7 fVoltage：類比輸出電壓 (-10V ~ 10 V) wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定 DAC channel 硬體觸發產生時之立即輸出電壓值。當 DAC 規劃為軟體命令模式時，可預先設定一硬體觸發電壓命令於 DAC 模組內，當觸發條件成立時則可由硬體立即把預先設定的命令輸出。
See also	IMC_DAC_SelectSource() ， IMC_DAC_SetTriggerSource() ， IMC_DAC_EnableTriggerOutput()

II.8.3 IMC_DAC_SetTriggerSource()

void IMC_DAC_SetTriggerSource(WORD channel, DWORD source, WORD wCardIndex)	
Parameters	channel：DAC channel number 0 ~ 7 source：DAC 硬體觸發源，以 bit 表示共 8 個中斷觸發源，可同時設定多種觸發源。以常數定義如下。 ENC0_TRIG_DAC：Encoder counter channel 0 comparator interrupt ENC1_TRIG_DAC：Encoder counter channel 1 comparator interrupt ENC2_TRIG_DAC：Encoder counter channel 2 comparator interrupt

	ENC3_TRIG_DAC：Encoder counter channel 3 comparator interrupt ENC4_TRIG_DAC：Encoder counter channel 4 comparator interrupt ENC5_TRIG_DAC：Encoder counter channel 5 comparator interrupt ENC6_TRIG_DAC：Encoder counter channel 6 comparator interrupt ENC7_TRIG_DAC：Encoder counter channel 7 comparator interrupt wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定 DAC channel 搭配觸發條件產生時,硬體立即輸出電壓值之功能，每一 DAC channel 可設定搭配多個觸發條件。設定完本函式後必須再設定 IMC_DAC_EnableTrigger Mode() 開啟觸發模式。本功能只能在 DAC 設定為軟體命令模式下使用，請參考 IMC_DAC_SelectSource()
See also	IMC_DAC_SelectSource()，IMC_DAC_EnableTriggerOutput()

II.8.4 IMC_DAC_SelectSource()

void IMC_DAC_SelectSource(WORD channel, WORD source, WORD wCardIndex)	
Parameters	channel：DAC channel number 0 ~ 7 source：DAC data source DAC_CMD_SOFT：Source from DAC output buffer DAC_CMD_PCL：Soruce from PCL error counter wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定 DAC channel 輸出之命令來源為軟體規劃或由硬體閉迴路(PCL)輸入。當命令源為 PCL 時則命令為 PCL 內部的位置誤差補償值，當設定為軟體規劃模式時可呼叫 IMC_DAC_SetOutputVoltage()，IMC_DAC_SetTriggerVoltage()設定輸出電壓命令
See also	IMC_DAC_SetOutputVoltage()，IMC_DAC_SetTriggerVoltage()

II.8.5 IMC_DAC_EnableChannel()

void IMC_DAC_EnableChannel(WORD channel , WORD wEnable, WORD wCardIndex)	
Parameters	channel：DAC channel number 0 ~ 7 wEnable：開啟或關閉指定 DA Channel 轉換功能 0：關閉 D/A 轉換功能 1：開啟 D/A 轉換功能

	wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	開啟 DAC channel 數位轉類比電壓輸出功能。設定完成後必須呼叫 IMC_DAC_StartConverter()函式，啟動 DAC 轉換功能。
See also	IMC_DAC_StartConverter()

II.8.6 IMC_DAC_StartConverter()

void IMC_DAC_StartConverter(WORD wStart, WORD wCardIndex)	
Parameters	wStart：開啟或關閉 DAC 轉換功能 0：關閉 DAC 轉換功能 1：開啟 DAC 轉換功能 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	啟動 DAC 進行數位/類比電壓輸出轉換，設定本函式後即開始進行數位/類比轉換。本函式必須配合 IMC_DAC_EnableChannel()函式使用。
See also	IMC_DAC_EnableChannel()

II.8.7 IMC_DAC_GetOutputVoltage()

void IMC_DAC_GetOutputVoltage(WORD Channel, float* fVoltage, WORD wCardIndex)	
Parameters	channel：DAC channel 編號 0 ~ 7 fVoltage：類比輸出電壓 (-10V ~ 10 V) wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	DAC channel 輸出電壓值
Description	讀取 DAC channel 輸出電壓值。

II.9. Timer Control

II.9.1 IMC_TMR_SetISRFunction()

void IMC_TMR_SetISRFunction(TMRISR myTMR_ISR, WORD wCardIndex)	
Parameters	myTMR_ISR：User 自己撰寫的 Timer 中斷副程式之 Function Pointer wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定 User 自己撰寫的中斷副程式，但必須於呼叫於 IMC_OpenDevice()之前。

II.9.2 IMC_TMR_GetInterruptSource()

DWORD IMC_TMR_GetInterruptSource(TMRINT *source, WORD wCardIndex)	
Parameters	source 為提供 User 使用以讀取 TMR 中斷訊號源;此為一結構變數，定義如下： typedef struct _TMR_INT { BYTE TIMER; }TMRINT; 讀回值表中斷發生源，其狀態變數說明如下： source->TIMER：Timer Index happened wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	讀取之 TMR 中斷源
Description	讀取 Timer 發出中斷的條件狀態，並清除中斷 Latch 值，等待下一次中斷發生。當硬體中斷發生後，可先經由 IMC_Get_InterruptSource() 判斷是否為 TMR 所發生，若是則呼叫本函式讀取中斷發生源。
See also	IMC_GLB_GetInterruptSource()

II.9.3 IMC_TMR_SetTimerClock()

void IMC_TMR_SetTimerClock(DWORD wClock, WORD wCardIndex)	
Parameters	wClock：計時器時脈數，可設定範圍 (0 ~ 2^{32} System Clock)
Return Value	None
Description	設定計時器之計時時脈數,配合呼叫 IMC_TMR_SetTimerEnable()及 IMC_TMR_SetTimerIntEnable()，計時終了將發出 Timer 中斷觸發功能。Timer 計時時脈數基本單位為系統時脈 System Clock (10ns)。
See also	IMC_TMR_SetTimerEnable()，IMC_TMR_SetTimerIntEnable()

II.9.4 IMC_TMR_SetTimer()

void IMC_TMR_SetTimer(float dfPeriod, WORD wCardIndex)	
Parameters	dfPeriod：計時器時間(μs)，可設定範圍 (0～2^{32} 毫秒) wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定計時器之計時時間，配合呼叫 IMC_TMR_SetTimerEnable() 及 IMC_TMR_SetTimerIntEnable()，計時終了將發出 Timer 中斷觸發功能。Timer 計時時間之單位為 μs。
See also	IMC_TMR_SetTimerEnable()，IMC_TMR_SetTimerIntEnable()

II.9.5 IMC_TMR_ReadTimerCount()

DWORD IMC_TMR_SetTimerClock(WORD wCardIndex)	
Parameters	wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	計時器目前之時脈數
Description	讀取計時器目前之計時時脈數，Timer 計時時脈數基本單位為系統時脈(10ns)。

II.9.6 IMC_TMR_SetTimerEnable()

void IMC_TMR_SetTimerEnable(WORD wEnable, WORD wCardIndex)	
Parameters	wEnable：開啟或關閉計時器功能 0：關閉計時器功能 1：開啟計時器功能 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	開啟 Timer 計時功能。呼叫本函式前請先設定 Timer 計時器值。
See also	IMC_TMR_SetTimer()

II.9.7 IMC_TMR_SetTimerIntEnable()

void IMC_TMR_SetTimerIntEnable(WORD wEnable, WORD wCardIndex)	
Parameters	wEnable：開啟或關閉計時器中斷功能 0：關閉計時器中斷功能 1：開啟計時器中斷功能 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	開啟 Timer 中斷觸發功能。呼叫本函式前請先設定 Timer 計時器值並開啟 Timer 計時功能
See also	IMC_TMR_SetTimer()，IMC_TMR_SetTimerEnable()

II.9.8 IMC_TMR_GetTimerEnable()

WORD IMC_TMR_GetTimerEnable(WORD wCardIndex)

Parameters	wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	0：關閉計時器功能 1：開啟計時器功能
Description	讀取 Timer 計時功能是否開啟。

II.9.9 IMC_TMR_GetTimerIntEnable()

WORD IMC_TMR_GetTimerIntEnable(WORD wCardIndex)

Parameters	wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	0：關閉計時器中斷功能 1：開啟計時器中斷功能
Description	讀取 Timer 計時中斷功能是否開啟。

II.9.10 IMC_ TMR _ReadTimerClock()

DWORD IMC_ TMR _ReadTimerClock(WORD wCardIndex);

Parameters	wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	讀取計時器時脈數(0 ~ 2^{32} System Clock)
Description	讀取計時器之計時時脈數。

II.9.11 IMC_WDG_EnableTimer()

void IMC_WDG_EnableTimer(WORD Enable, WORD wCardIndex);

Parameters	Enable：啟動 Watch Dog 功能 0：關閉 Watch Dog 功能 1：開啟 Watch Dog 功能 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	開啟 Watch Dog 功能

II.9.12 IMC_WDG_SetTimerClock()

void IMC_WDG_SetTimerClock(DWORD clock, WORD wCardIndex)

Parameters	clock：Watch Dog 時脈數，可設定範圍 (0 ~ 2^{32} System Clock) wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定 Watch Dog 之計時時脈數。

II.9.13 IMC_WDG_ReadTimerClock()

DWORD IMC_WDG_ReadTimerClock(WORD wCardIndex)

Parameters	wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	Watch Dog 目前之時脈數
Description	讀取 Watch Dog 目前之計時時脈數，Watch Dog 計時時脈數基本單位為系統時脈(10ns)。

II.9.14 IMC_WDG_SetTimer()

void IMC_WDG_SetTimer(DWORD Period, WORD wCardIndex)	
Parameters	Period：Watch Dog 計時器時間(μs)，可設定範圍(0 ~ 2^{32} 毫秒)。 wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定 Watch Dog 之計時時間。Watch Dog 計時時間之單位為 μs。一但 Watch Dog 計時終了時會產生硬體 reset 的訊號，如不想產生 reset 訊號，則在計時終了前可利用 IMC_WDG_RefreshTimer()使計數器重新計時

II.9.15 IMC_WDG_SetResetPeriod()

void IMC_WDG_SetResetPeriod(DWORD clock, WORD wCardIndex)	
Parameters	clock硬體 reset 訊號持續時間，單位為 10ns wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	設定在 Watch Dog 計時終了所產生硬體 reset 訊號的持續時間

II.9.16 IMC_WDG_RefreshTimer()

void IMC_WDG_RefreshTimer(WORD wCardIndex)	
Parameters	wCardIndex：欲控制的運動控制卡之編號，編號範圍 0~5
Return Value	None
Description	重置 Watch Dog 之計時時間。避免 Watch Dog 計時終了產生硬體 reset 訊號

Revision History

日期	版本	修改內容
2010/10/06	2.01	—P.16，修正 IMC_ENC_SetInputFormat() 參數定義。 —P.19，修正 IMC_ENC_StartCounter () 參數定義。 — P.36 ，修正 IMC_ADC_SetConverter Mode ()參數定義。 —修飾各函式文字說明敘述與排版對齊。
2011/03/15	2.02	—P.18，新增 IMC_ENC_SetExternalLatch Source()外部觸發訊號源至 16 種。
2013/04/30	2.03	—新增 4 項 IMC_GLB 函式 —新增 8 項 IMC_PGE 函式 —新增 9 項 IMC_PCL 函式 —新增 1 項 IMC_DAC 函式 —新增 2 項 IMC_ADC 函式 —新增 11 項 IMC_LIO 函式 —新增 3 項 IMC_TMR 函式 —新增 6 項 IMC_WDG 函式 —新增 4 項 IMC_ARIO 函式 —修正 2.02 版之錯誤定義
2013/06/18	2.04	—刪除 1 項 IMC_PCL 函式 —刪除 2 項 IMC_ARIO 函式 —修飾部份函式文字說明敘述