PCI TG-DIO
User Manual

Soliton Technologies CO., LTD

www.soliton.com.tw

Copyright

Publication date: 08/15/2003 V1.6

目錄 :

功能介紹

PCI-TG DIO 是汎捷科技專為數位化輸出入自動控制所全新設計之 PCI 界面 I/O 卡。將硬體線路設計精簡化，達到低成本、高可靠度的要求。另齊全之軟體支援及即插即用(Plug & Play)機制，使用者無須做任何 Jumper 設定，即可簡單、迅速的使用本卡，非常適合於工業自動控制、學校教學實驗、遊戲機控制等用途。

卡片是採矮卡設計(Low Profile) ，可適用於各種高度之電腦機箱。二顆廣泛使用的 8255 IC，提供 48bit TTL 位準之數位輸出入線，除支援 Mode0，Mode1，Mode2 基本操作外，並支援 8255 於 Mode1 及 Mode2 下產生硬體中斷；同時也提供 Mailbox 功能可讓使用者更簡單直接存取各 8-bit 輸入及輸出界面。面板上有一標準 25 pin D-type 接頭，提供 24-bit I/O 信號；另 24-bit 則以排針座方式供使用者自定接頭。

產品內容

PCI-TG DIO 卡 ---------- 1 片
安裝光碟片 ---------- 1 片
25 Pin D-Type 連接線 --------- 1 條

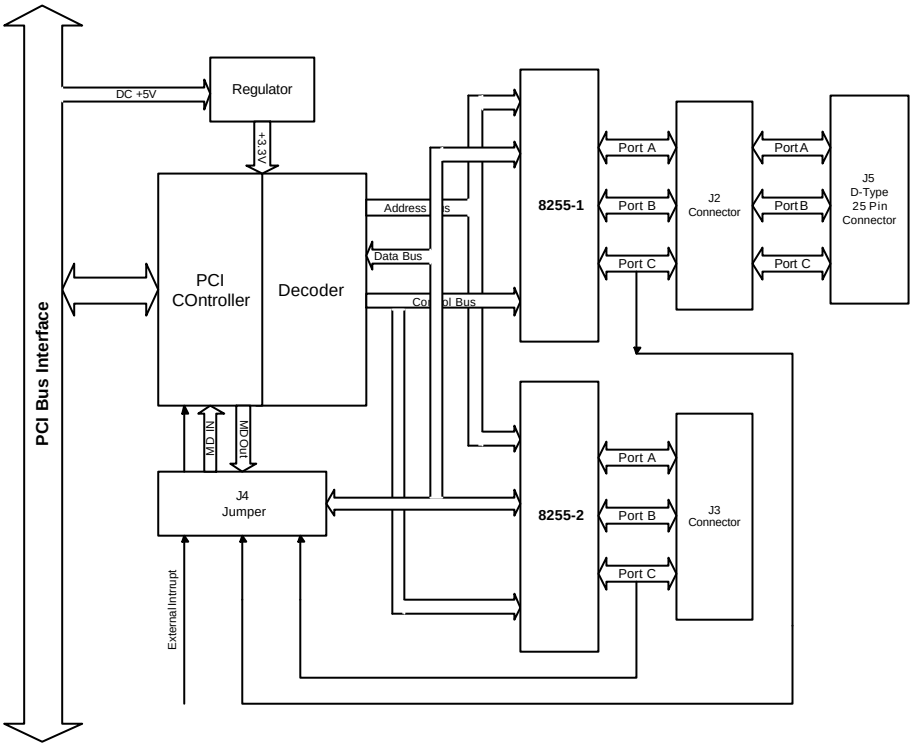
光碟片內容

光碟片目錄內容說明如下：

Driver	Windows 98/2000/XP 驅動程式
DLL	動態連結函數館
Library	靜態連結函數館
Include	Include file
Samples	範例程式
├── MSVC	Microsoft Visual C++ 6.0 範例程式
├── MSVB	Microsoft Visual Basic 6.0 範例程式
└── BCB	Borland C++ Builder 5.0 範例程式
Utility	測試程式
Documents	使用說明書
Linux	Linux 驅動程式及範例程式壓縮檔

硬體架構

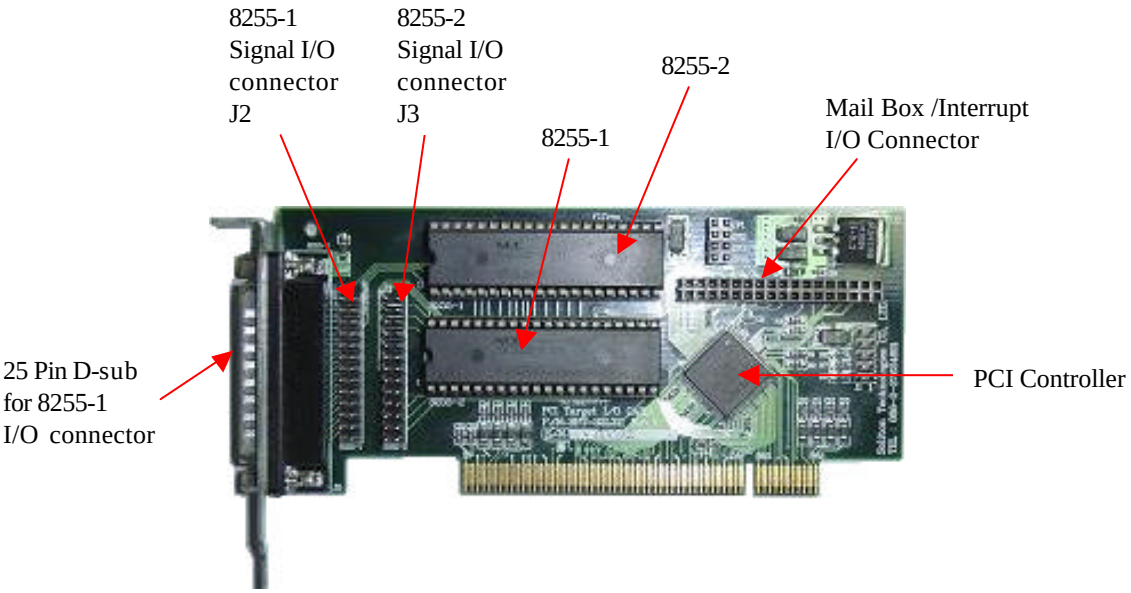
主要是由一 XILINX CPLD 及二個 8255 IC 所組成。XILINX CPLD 其功能主要是處理 PCI Bus 上的訊號，並將其解譯 (Decode) 成本地匯流排(Local Bus) 訊號以讀/寫二個 8255 的內部暫存器 (Register) 來完成 I/O 控制動作。



圖一. 系統方塊圖

從系統方塊圖可知，每一顆 8255 的 Port C 有二條中斷訊號線可用程式產生內部中斷 (Internal Interrupt) ；另一外接中斷訊號線 (External Interrupt)，提供使用者連接外部硬體中斷線路來進階使用中斷。所有中斷均可透過 Jumper 設定來使用。有關詳細 8255 中斷使用請參閱坊間 8255 的相關書籍。

硬體圖片



圖二. PCI-TG DIO 電路板

系統規格

- **48-bit TTL Digital I/O lines**
- **PCI Bus 2.2 Compliant**
- **8255 Mode 1, 2, 3 Support**
- **8 bits of Mailbox Input Direct Access**
- **8 bits of Mailbox output Direct Access**
- **Multi Card Operation Support**
- **Hardware Interrupt Support**
- **Plug & Play**

Driver Support

- **Windows 98/Me/2K/XP**
- **Linux**

Software

- **DLL and sample program provided**
- **Test Panel Utility**

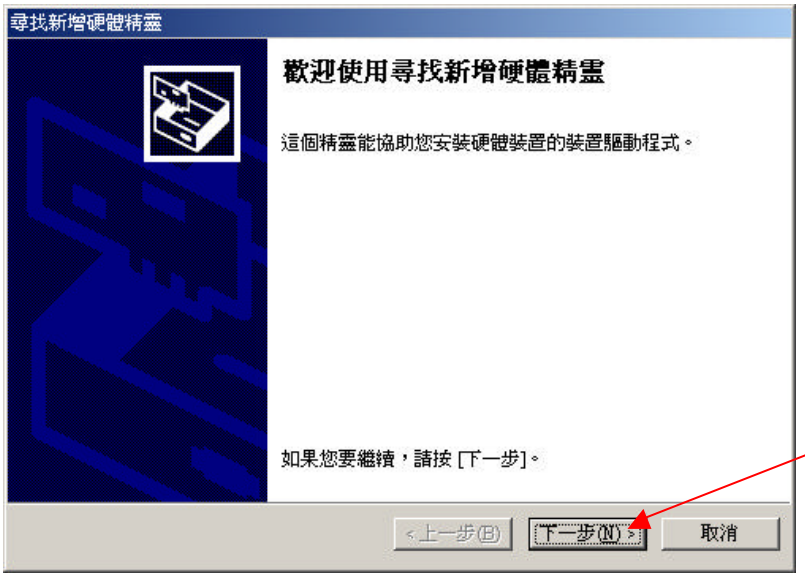
安裝硬體

在安裝卡片之前，請先將電腦關機，再將 AC 電源線從插座拔除並依下列步驟完成硬體安裝。

1. 打開電腦機箱蓋。
2. 從防靜電袋取出 PCI-TG DIO 卡片直接插在主機板上任何一個 PCI 插槽。
3. 確定插入定位後，以螺絲將卡片固定緊即可。
4. 將 AC 電源線插回插座，開啟電腦。

安裝驅動程式

裝妥硬體後重新開啟電腦電源，待 Windows 作業系統完成啟動載入後，系統會自動偵測到新硬體加入並出現一找到新硬體對話框，請依下列安裝程序完成驅動程式安裝。



Step 2.

Step1: 打開光碟機，放入隨 PCI-TG DIO 卡所附之光碟片。

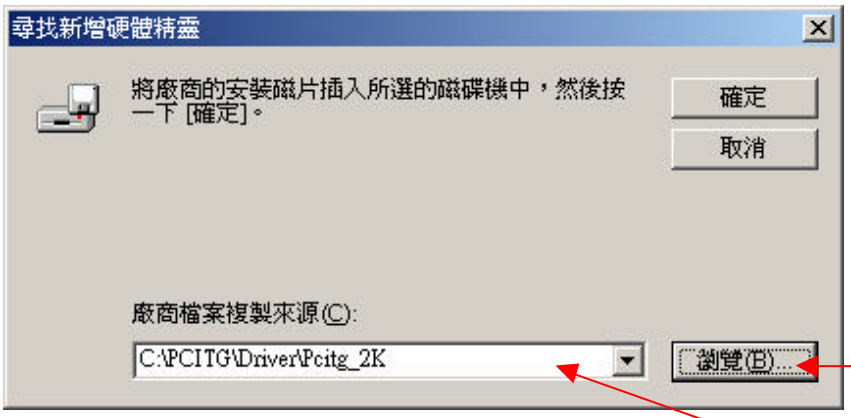
Step2: 點選 下一步 按鈕，使用尋找新增硬體精靈協助安裝。

Step3: 點選 搜尋適當的裝置驅動程式檔案。

Step4: 點選 下一步 按鈕。

Step5: 點選 指定位置。

Step6: 點選 下一步 按鈕。



Step7: 點選 瀏覽 按鈕。

Step8: 選擇您光碟機之磁碟代號及驅動程式的路徑。

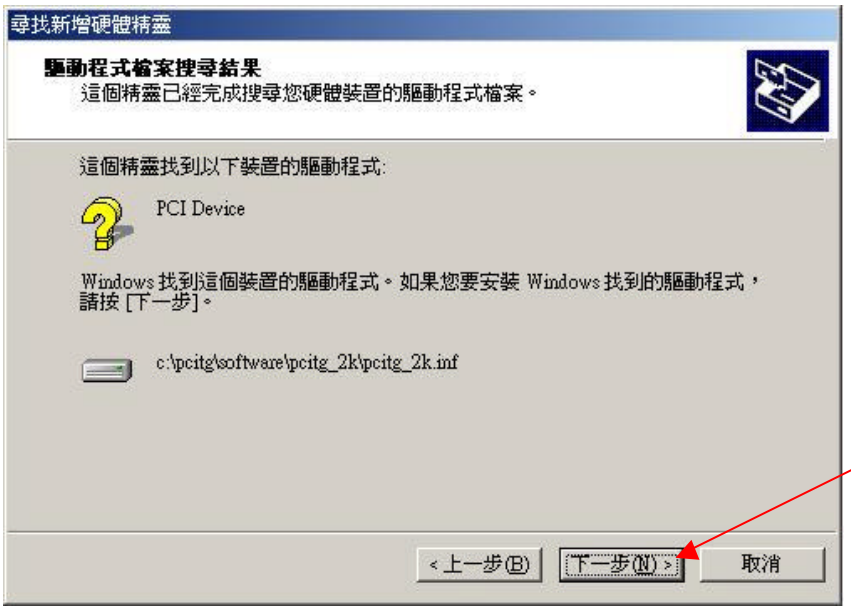
假設您光碟機之磁碟代號為 E:

Windows 2000/XP **作業系統使用者請使用下列路徑**

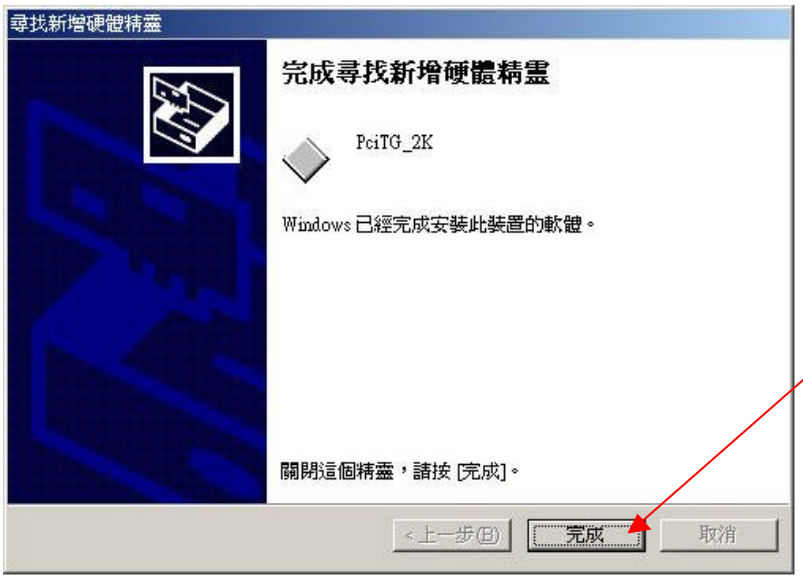
E:\PCI-TG DIO\Drivers\Win2K

Windows 98 **作業系統使用者請使用下列路徑**

E:\PCI-TG DIO\Drivers\Win9x



Step9: 電腦將會在光碟機內搜尋到對應之驅動程式，請點選 下一步 按鈕。

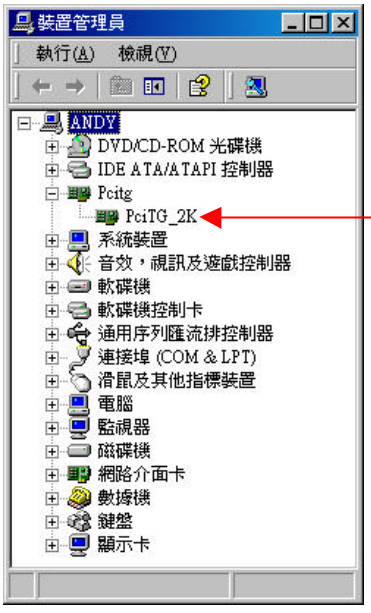


Step10: 電腦在完成驅動程式安裝後，出現安裝完成對話框，請點選 完成 按鈕。

檢查裝置管理員

在完成以上步驟後，請開啟裝置管理員檢查是否有 Pcitg 裝置，若有，表示您已成功安裝驅動程式。若是 Pcitg 裝置圖示(Icon)前有問號圖示，則表示與其他裝置有系統資源上衝突，請將 PCI-TG DIO 卡更換另一 PCI 插槽，重新安裝。

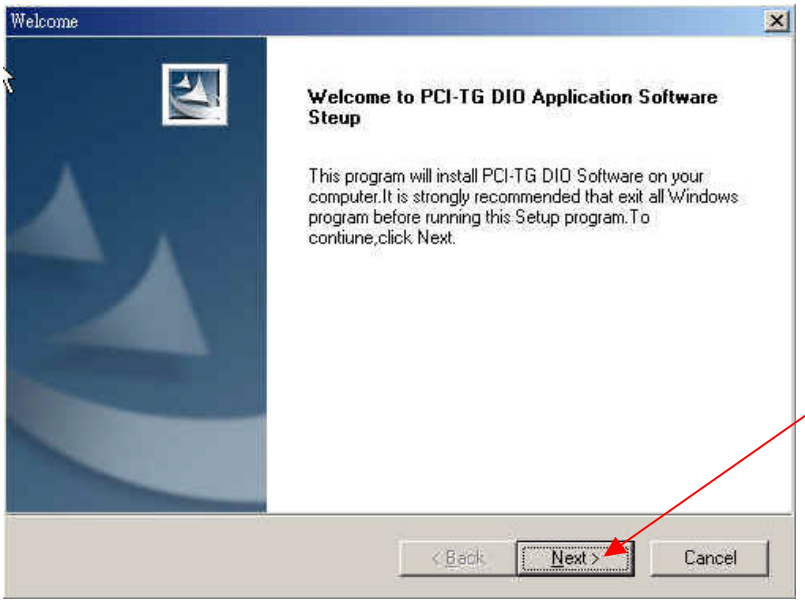
欲開啟裝置管理員，請點選電腦左下方 開始 按鈕，再選擇 設定 及 控制台。於控制台內快點二下 系統 圖示；在系統視窗內點選上方的 硬體 頁，並點選此頁內的 裝置管理員 即可開啟下方視窗。



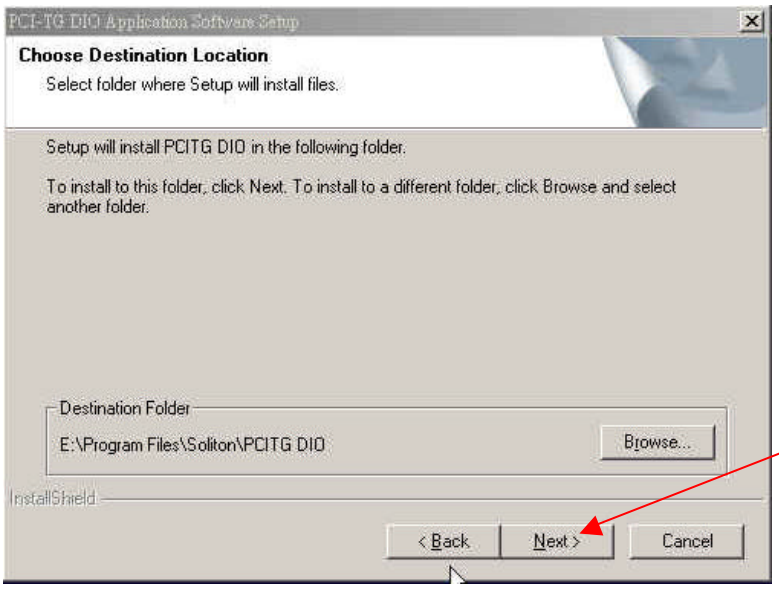
安裝應用軟體

應用軟體內容十分豐富，除了有二個測試軟體外也分別提供 Microsoft Visual Basic, Visual C++ 及 Borland C++ Builder 等語言範例程式。請依下列步驟完成軟體安裝。

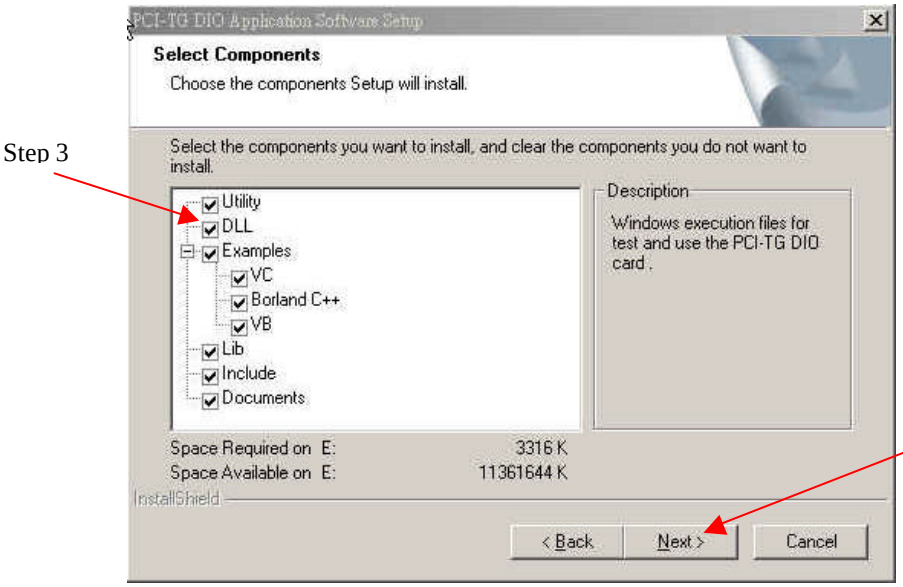
Step1: 打開光碟機，放入隨 PCI-TG DIO 卡所附之光碟片並執行 Tgdio_Setup.exe 安裝程式。隨即出現歡迎畫面，請點選 Next



Step 1

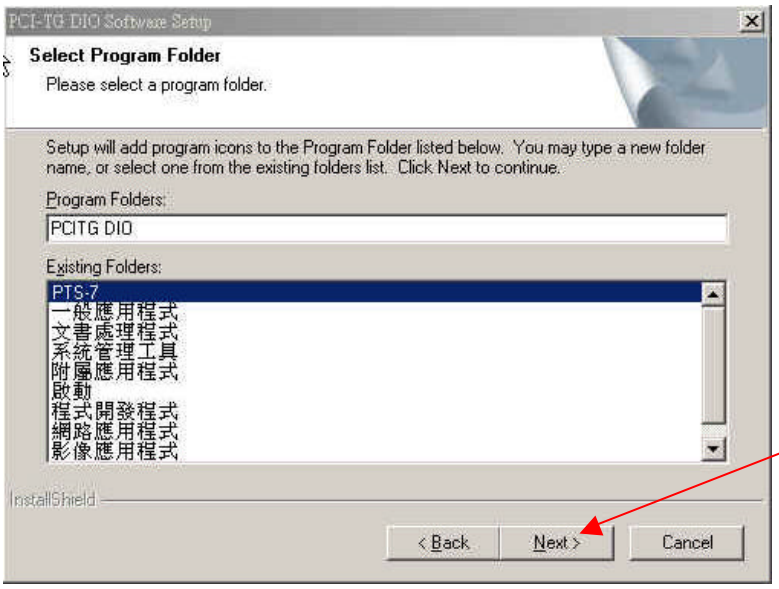


Step2: 選擇安裝的路徑及目錄名稱，可用系統內定之路徑或點選 Browse 變更路徑。之後請點選 Next 按鈕。

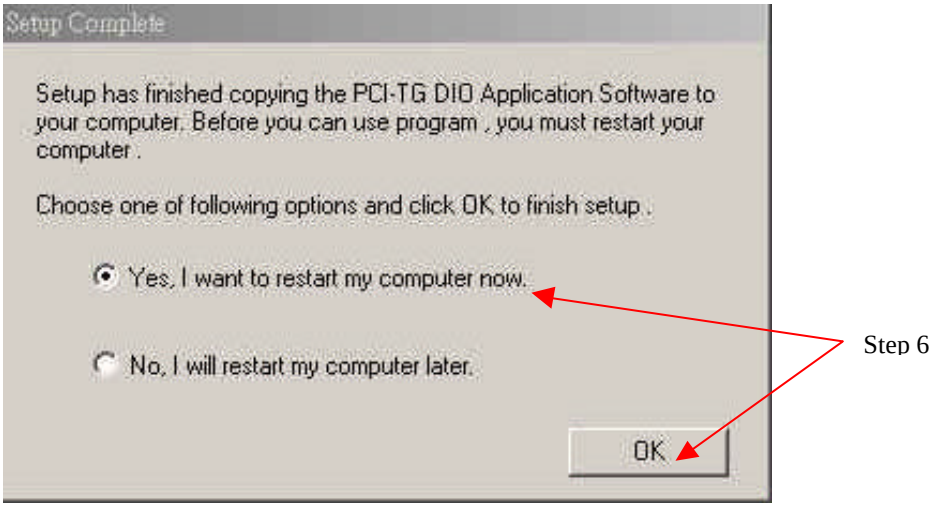


Step3: 勾選欲安裝的項目，因所需硬碟容量不大建議安裝所有項目。

Step4: 請點選 Next 按鈕。



Step5: 設定在 開始 / 程式集 內的路目錄名稱，可用系統內定之 PCITG DIO 即可。接著請點選 Next 按鈕。



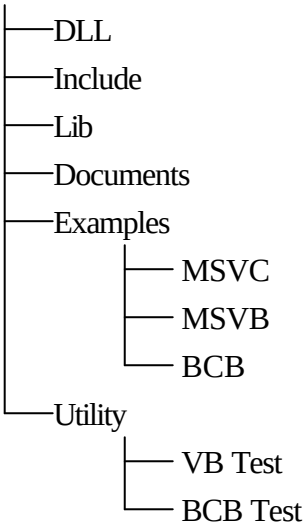
Step6: 安裝程式接著將會複製(Copy)檔案到您的電腦內。最後出現安裝完畢畫面並詢問是否要重新開機。請點選要重新開機選項再按 OK 按鈕。

目錄及檔案介紹

在完成軟硬體安裝後，系統將會安裝下列目錄在您的電腦硬碟中。路徑名稱會因個人設定有所不同，說明書均以系統內定之路徑為例說明。

系統內定之路徑(Default Installation Path)

C:\Program Files\Soliton\PCITG_DIO



DLL

Pcitglib.dll : **程式在執行中所需之動態連結函數館**

Library

Pcitglib.lib : VC **程式於** Link **時所需之連結函數館**

Pcitgbcb.lib : BCB **程式於** Link **時所需之連結函數館**

Include

Pcitglib.h : Include file

Samples

- MSVC　Microsoft Visual C++ 6.0 **範例程式**
- MSVB　Microsoft Visual Basic 6.0 **範例程式**
- BCB　Borland C++ Builder 5.0 **範例程式**

Utility

VB Test : VB_Panel **測試程式**

BCB Test : BCB_Panel **測試程式**

Documents

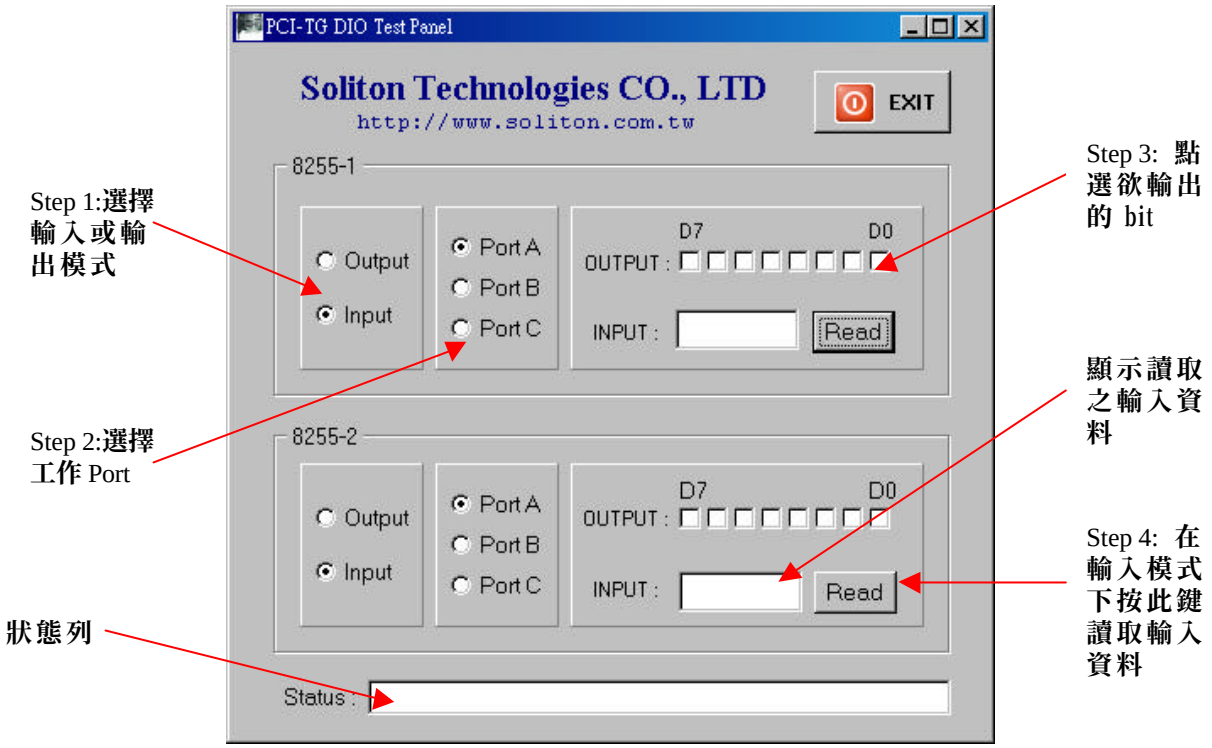
Pcitg_Manual.pdf : **使用說明書**

卡片功能測試

此處將以 BCB_Panel.exe 來執行 PCI-TG DIO 基本的輸出/輸出功能。請快點二下 c:\Program Files\Soliton\PCITG_DIO\BCB_Panel.exe 圖示開始執行程式。程式啟動後會先開啟 TG DIO 裝置，並將結果顯示在下方的狀態列(Status) 。圖三是程式開啟後的畫面。從畫面可知有二欄分別控制二顆 8255。

每個 8255 可讓使用者設成輸入或輸出模式(Input/Output Mode)。選擇你要控制 8255 的 Port，每顆 8255 共有三個 Port，分別是 A ,B 及 C。每個 Port 又有 8 位元(bits)。故每顆 8255 共有 3 Ports * 8 bits= 24-bit I/O。

若要輸出，只要點選 D7-D0 下方小方格即可立刻輸出位元資料。相反的若要讀取輸入資料，先點選 Input 將 8255 設成輸入模式再點選欲讀取的 Port，最後按下 Read 按鈕，讀入資料結果會顯示在 Read 左邊方格(EditBox)內。



圖三

pcitg_open

Syntax：

pcitg_t pcitg_open(DWORD *instance*)

Description：

Get a handle to a device(PCITG). When the function returns, pcitg_t will contain completed information for the PCITG board.

- *instance* is PCITG board ID or board Index to determine which PCITG board we want to control. The valid range is 0 to 31.

Return Value：

If successful, will return the PCITG handle of data structure pcitg_t .
Return 0 if fail.

Usage：

```
pcitg_t board0;
int boardid = 0;
board0 = pcitg_open(boardid);
```

Notes：

This function should be used to open a channel to a PCITG device and get a handle to the device driver before any PCI API functions that require a handle can be used.

pcitg_close

Syntax：

void pcitg_close(pcitg_t *board*)

Description：

Close the handle of PCITG board.

・*board* is a handle of PCITG board include the information of PCITG.

Return Value：

No return value .

Usage：

pcitg_close(board0);

Notes：

Be sure to call pcitg_close before your application exits.

pcitg_in

Syntax：

int pcitg_in(pcitg_t *board*, int *chip*, int *port*, unsigned long **value*)

Description：

Basic port Read routine.

To get data from port of 8255 of PCITG board.

- *board* is a handle of PCITG board include the information of PCITG.
- *chip* is a handle of 8255, indicate *the* base address of 8255.
- *port* are PortA, PortB, PortC, or ControlPort of 8255.
- **value* is a address of unsigned long parameter *value*, to store the input data.

Return Value：

Return 0 if successful; 1 on error.

Usage：

```
unsigned long data;
      :
// Set 8255 chip0 of PCI-TG DIO board to input mode
pcitg_out(board0, PCITG_CHIP0, PCITG_PORT_CTL, 0x9B);

// Read input data from port A of 8255 chip0 then print out the data
pcitg_in(board0, PCITG_CHIP0, PCITG_PORT_A, &data);
printf("Input data from 8255 port = 0x%.2X\n", data);
```

Notes：

Chip ? could be **PCITG_CHIP0 or PCITG_CHIP1**

Port ? could be one of the macro **PCITG_PORT_A, PCITG_PORT_B**, **PCITG_PORT_C, PCITG_PORT_CTL or PCITG_PORT_ALL. PCITG_PORT_ALL** means can read all ports in the same time.

pcitg_out

Syntax：

int pcitg_out(pcitg_t *board*, int *chip*, int *port*, unsigned long *value*)

Description：

Basic port write routine.

Output data to port of 8255 of PCITG board.

- *board* is a handle of PCITG board include the information of PCITG.
- *chip* is a handle of 8255, indicate *the* base address of 8255.
- *port* are PortA, PortB, PortC, or ControlPort of 8255.
- *value* output data value depends on user define.

Return Value：

Return 0 if successful; 1 on error.

Usage：

```
int m_outputdata = 0xAA;
// Set 8255 chip0 of PCI-TG DIO board to output mode
pcitg_out(board0, PCITG_CHIP0, PCITG_PORT_CTL, 0x80);

// To output data 0x55 on port A of 8255 chip0
pcitg_out(board0, PCITG_CHIP0, PCITG_PORT_A, 0x55);
// To output data 0xAA on port B of 8255 chip0
pcitg_out(board0, PCITG_CHIP0, PCITG_PORT_B, m_outputdata);
```

Notes：

chip ? could be **PCITG_CHIP0** or **PCITG_CHIP1**

port ? could be one of the macro **PCITG_PORT_A, PCITG_PORT_B**, **PCITG_PORT_C,PCITG_PORT_CTL or PCITG_PORT_ALL.**

PCITG_PORT_ALL means can read/write all ports in the same time.

pcitg_in2

Syntax：

int pcitg_in2(pcitg_t *board*, int *ports*, unsigned long **value*)

Description：

To do the multiple chip testing. You can use this function to get data from port of 8255 chip0 and chip1 of PCITG board.

- *board* is a handle of PCITG board include the information of PCITG.
- *ports* are PortA, PortB, PortC, or ControlPort of 8255.
- **value* is a address of unsigned long parameter *value*, to store the input data.

Return Value：

Return 0 if successful, -1 on error.

Usage：

```
unsigned long data;
pcitg_in2(board0, PCITG_PORT_A, &data);
```

Notes：

Input data of Chip0 are store in data bit7 – bit0
Input data of Chip1 are store in data bit15 – bit8

pcitg_out2

Syntax：

int pcitg_out2(pcitg_t *board*, int *ports*, unsigned long *value*)

Description：

To do the multiple chip testing. You can use this function to output data to port of 8255 chip0 and chip1 of PCITG board.

- *board* is a handle of PCITG board include the information of PCITG.
- *ports* are PortA, PortB, PortC, or ControlPort of 8255.
- *value* output data value depends on user define.

Return Value：

Return 0 if successful, -1 on error.

Usage：

pcitg_out2(board0, PCITG_PORT_A, 0x1234);

Notes：

Output data of Chip0 should set in data bit7 – bit0

Output data of Chip1 should set in bit15 – bit8

pcitg_mbin

Syntax：

int pcitg_mbin(pcitg_t *board*, unsigned long **value*)

Description：

The mbin means Mail Box register Input .You can use this function to get data from PCITG board and store the data to Mail Box.

・*board* is a handle of PCITG board include the information of PCITG.

・**value* is a address of unsigned long parameter *value*, to store the input data.

Return Value：

Return 0 if successful, 1 on error.

Usage：

```
unsigned long data;
pcitg_t board0;
board0 = pcitg_open(0);


// Read input data from mailbox of PCI-TG DIO then print out the data
pcitg_mbin(board0, &data)
printf("Input data from mailbox = 0x%.2X\n", data);
```

pcitg_mbout

Syntax：

int pcitg_mbout(pcitg_t *board*, unsigned long *value*)

Description：

The mbout means output data to Mail Box register. .You can use this function output data to PCITG board's Mail Box register.

- *board* is a handle of PCITG board include the information of PCITG.
- *value* output data value depends on user define.

Return Value：

Return 0 if successful, 1 on error.

Usage：

```
pcitg_t board0;
board0 = pcitg_open(0);

// Write output data to mailbox out port on PCI-TG DIO board 0
pcitg_mbout(board0, 0x78);
```

pcitg_intcin

Syntax：

int pcitg_intcin(pcitg_t *board*, unsigned long **value*)

Description：

Read the Interrupt control register (INT_CTL) status.

- *board* is a handle of PCITG board include the information of PCITG .
- **value* is a address of unsigned long parameter *value*, to store the input data.

Return Value：

Return 0 if successful, 1 on error.

Usage：

```
unsigned long m_statusdata;
pcitg_t board0;
board0 = pcitg_open(0);


pcitg_intcin(board0, &m_statusdata);
printf("Interrupt status = 0x%.2X\n", m_statusdata);
```

pcitg_intcout

Syntax：

int pcitg_intcout(pcitg_t *board*, unsigned long *value*)

Description：

Set the interrupt control register word.

- *board* is a handle of PCITG board include the information of PCITG .
- *value* output data value depends on user define.

Return Value：

Return 0 if successful, 1 on error.

Usage：

```
pcitg_t board0;
board0 = pcitg_open(0);

// Write data 0xB0 to interrupt control register
pcitg_intcout(board0, 0xB0);
```

pcitg_connect

Syntax：

int pcitg_connect(pcitg_t *board*, void (*sig_handler) (pcitg_t, int))

Description：

To connect user's interrupt process to main program and pass its handle to main program.

*sig_handler : UserISR

(UserISR will be executed on each interrupt)

* int : INT vector, will be one of the PCITG_IRQ_EXT, PCITG_IRQ_B0, PCITG_IRQ_A0, PCITG_IRQ_B1, PCITG_IRQ_A1

・*board* is a handle of PCITG board include the information of PCITG .

Return Value：

Return 0 if successful; 1 on error.

Usage：

```
// Define a data structure
typedef struct
{
    ULONG nSampleCount;
    int intvec;
} MY_EXTENSION, *PMY_EXTENSION;

pcitg_t board0;
board0 = pcitg_open(0);

// Enable Chip0 PC0, PC3, Chip1 PC0, PC3 Interrupt
pcitg_intcout(board0, 0x3F);

PMY_EXTENSION pExtension = NULL;
pExtension = (PMY_EXTENSION)malloc(sizeof(MY_EXTENSION));
pExtension->nSampleCount = 0;
pcitg_set_extension(board0, pExtension);

pcitg_connect(board0, UserISR);
```

pcitg_set_extension

Syntax：

int pcitg_set_extension(pcitg_t *board*, PVOID *pext)*

Description：

Associates a user defined data structure with a particular device.

- *board* is a handle of PCITG board include the information of PCITG .
- *pext* is the structure pointer of extension.

Return Value：

Return 0 if successful.

Usage：

```
// Define a data structure
typedef struct
{
        ULONG nSampleCount;
        int intvec;
} MY_EXTENSION, *PMY_EXTENSION;

PMY_EXTENSION pExtension = NULL;
pExtension = (PMY_EXTENSION)malloc(sizeof(MY_EXTENSION));
pExtension->nSampleCount = 0;
pcitg_set_extension(board0, pExtension);
```

Notes：

The pcitg_set_extension and pcitg_get_extension routines support a user defined device extension to hold variables and/or buffers associated with a particular device

pcitg_get_extension

Syntax：

PVOID pcitg_get_extension(pcitg_t *board*)

Description：

Returns a pointer to the user defined data structure.

・*board* is a handle of PCITG board include the information of PCITG .

Return Value：

A point by user defined.

Usage：

```
// User interrupt service routine
void UserISR(pcitg_t board, int intvec)
{
    PMY_EXTENSION pExtension =
    (PMY_EXTENSION)pcitg_get_extension(board);
    // Increase the interrupt counter
    pExtension->nSampleCount++;
    pExtension->intvec = intvec;

    // test multichip, in2, out2
    pcitg_out2(board, PCITG_PORT_C, 0);
    printf("ISR INT Count=[%d]
    Vector=[%d]\n",pExtension->nSampleCount, pExtension->intvec);

    // Output the interrupt count to 8255 port A
    pcitg_out(board,PCITG_CHIP0,PCITG_PORT_A,
    pExtension->nSampleCount);
    // Output the interrupt count to 8255 port B
    pcitg_out(board, PCITG_CHIP0,PCITG_PORT_B, intvec);
}
```

範例程式

此範例是以 C 語言所撰寫程式。其主要目的是使用 8255 基本輸出/輸入的 Mode0 模式工作。若要正確執行此程式在硬體上須使用一 26Pin 排線以一對一方式連接 PCI-TG DIO 板上的 J2 及 J3 插座。

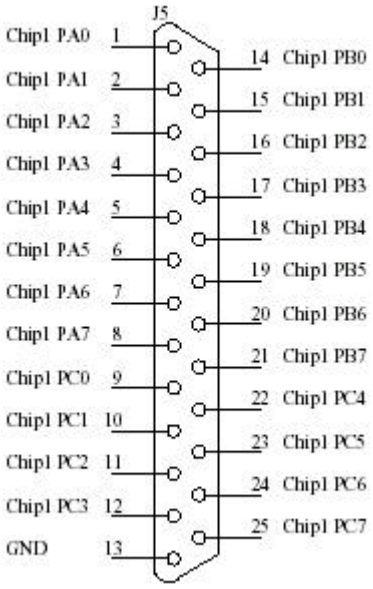
程式中共有三個迴圈分別從 PortA 、PortB 、 PortC 輸出及輸入。每個迴圈執行 256 次，也就是說從第一顆 8255(8255-1)輸出十六進制資料 0x00 到 0xFF 共 256 次，並從第二顆 8255(8255-2)讀回，再檢查是否有錯誤。其他 VC, VB, BCB 相關範例程式及進階使用請參考 **Examples** 目錄內容。

```
#include "stdafx.h"
#include "pcitglib.h"


int main(int argc, char* argv[])
{
	int board_index = 0;
	pcitg_t board0;
	unsigned long data;
	int execnt = 0;
	int sleeptime = 10;
	int status = 0;
	unsigned long x;
	int error = 0;
	int total_error = 0;

	if (argc > 1)
	{
		board_index = atoi(argv[1]);
	}

	board0 = pcitg_open(board_index);
	if (board0 == 0)
	{
		printf("ERROR opening Board[%d]: (%0x) returned from CreateFile\n",
			board_index, GetLastError());
		return 1;
	}
	else
	{
		printf("Board[%d] found, Device Handle = 0x%.8X\n", board_index,
			board0->hDevice);
	}

	// Set 8255-1 to output mode
	pcitg_out(board0, PCITG_CHIP0, PCITG_PORT_CTL, 0x80);
	// Set 8255-2 to input mode
```

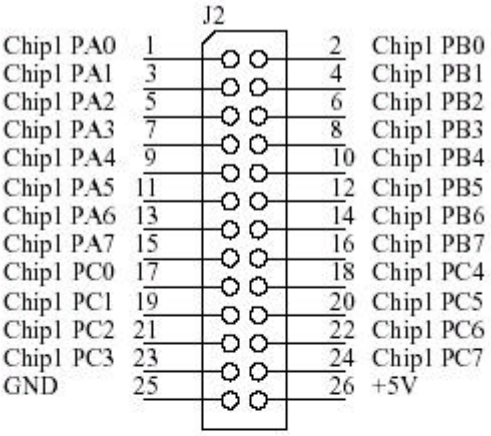
```
	pcitg_out(board0, PCITG_CHIP1, PCITG_PORT_CTL, 0x9B);

	// Looping for output data 0 to 255 on chip0 port A and read back from chip1 port A
	for (x=0; x<256; x++)
	{
		pcitg_out(board0, PCITG_CHIP0, PCITG_PORT_A, x);
		pcitg_in(board0, PCITG_CHIP1, PCITG_PORT_A, &data);
		if (x != data) error++;
	}
	if (error > 0) printf("Port A Check Error = %d\n", error);
	else printf("Port A Check Success!\n");
	total_error += error;

	error = 0;
	// Looping for output data 0 to 255 on chip0 port B and read back from chip1 port B
	for (x=0; x<256; x++)
	{
		pcitg_out(board0, PCITG_CHIP0, PCITG_PORT_B, x);
		pcitg_in(board0, PCITG_CHIP1, PCITG_PORT_B, &data);
		if (x != data) error++;
	}
	if (error > 0) printf("Port B Check Error = %d\n", error);
	else printf("Port B Check Success!\n");
	total_error += error;

	error = 0;
	// Looping for output data 0 to 255 on chip0 port C and read back from chip1 port C
	for (x=0; x<256; x++)
	{
		pcitg_out(board0, PCITG_CHIP0, PCITG_PORT_C, x);
		pcitg_in(board0, PCITG_CHIP1, PCITG_PORT_C, &data);
		if (x != data) error++;
	}
	if (error > 0) printf("Port C Check Error = %d\n", error);
	else printf("Port C Check Success!\n");
	total_error += error;


	pcitg_close(board0);
	if (total_error > 0) status = 1;
	return status;
}
```

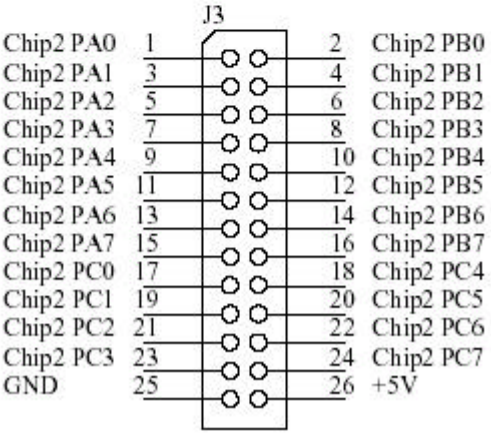
接頭接？圖

25 Pin D-Type Connector Pin Assignments

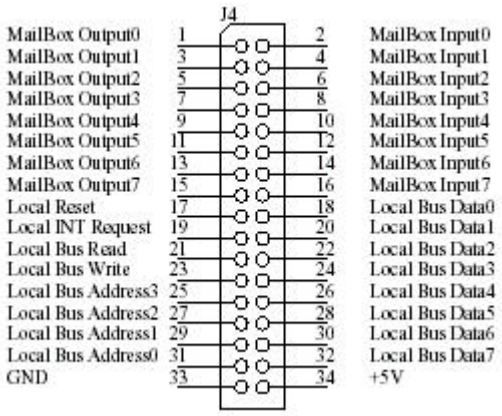


I/O Connector Pin Assignments





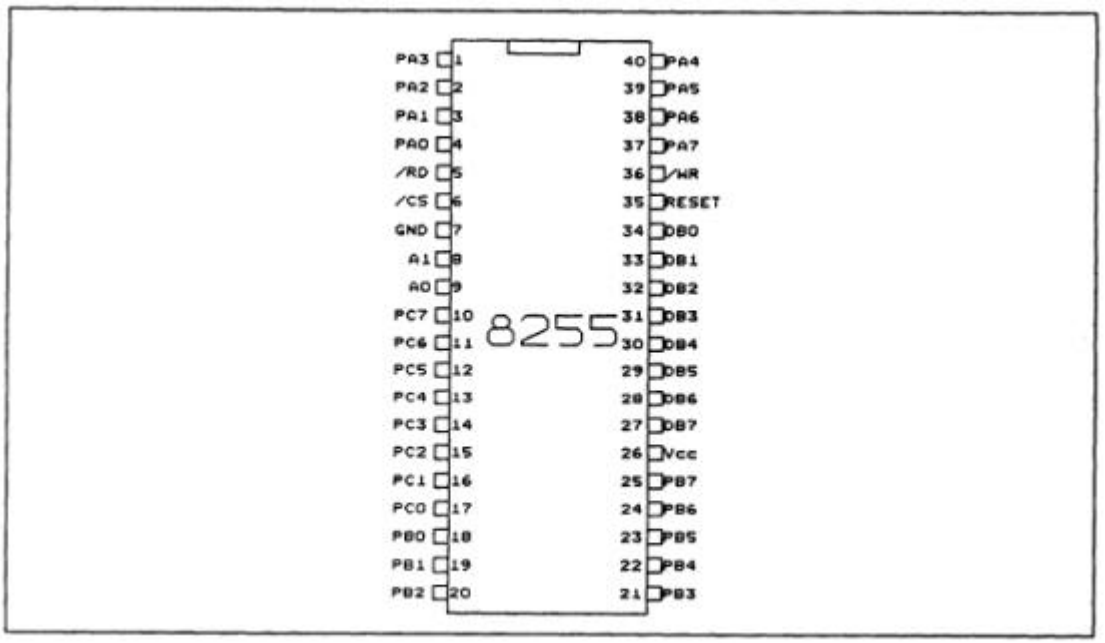
Expansion Connector Pin Assignments



附錄一

8255A I/O Controller IC 規格：

8255A 是當前最常用來作 I/O 控制的一顆 IC，它是利用 CMOS 技術所製成，其特點為省電並且能容忍較高電壓，但不適合高頻運作。



8255 腳位圖

8255 的各接腳功能如下所列：

1. PAO 到 PA7：接腳 4 到接腳 1 及接腳 40 到接腳 37
   A 埠，為 8bits 的 I/O 埠。
2. PBO 到 PB7：接腳 18 到接腳 25
   B 埠，為 8bits 的 I/O 埠。
3. PC0 到 PC3 及 PC4 到 PC7：接腳 17 到接腳 14 及接腳 13 到接腳 10
   C 埠，隨 8255 工作模式的不同，C 埠可作為單純的 I/O 或是作為 A 埠、B 埠的交握（Handshaking）控制信號的輸出入腳。
4. DB0 到 DB7：接腳 27 到接腳 34
   三態的資料匯流排，微電腦經由此匯流排，進行與 8255 的資料傳輸。
5. Vcc：接腳 26
   +5V 電源供應腳。
6. GND：接腳 7
   8255 接地腳。

7. REST：接腳 35

   8255 的重置腳，高態動作。8255 重置後會清除所有內部暫存器的值，並設定 A 埠、B 埠及 C 埠皆為輸入模式。

8. $\overline{\text{CS}}$：接腳 16

   晶片選擇線，低態動作。

9. $\overline{\text{RD}}$：接腳 5

   微電腦讀取 8255 內部資料控制腳，當 CS $\overline{\text{接腳}}$信號為 0，RD 接$\overline{\text{腳}}$$\overline{\text{信}}$號從 1 變為 0 時，由 8255 的 A1 及 A0 接腳信號所指定之暫存器的內容將被送到匯流排上。

10. $\overline{\text{WR}}$：接腳 36

    微電腦系統欲將資料寫入 8255 時，當 CS $\overline{\text{接腳}}$信號為 0 時，WR 接$\overline{\text{腳}}$$\overline{\text{信}}$號從 1 變為 0 時，8255 會將資料匯流排上的資料存入由 A1 及 A0 接腳信號所指定的內暫存器中。

1、 A1 及 A0：接腳 8 及接腳 9

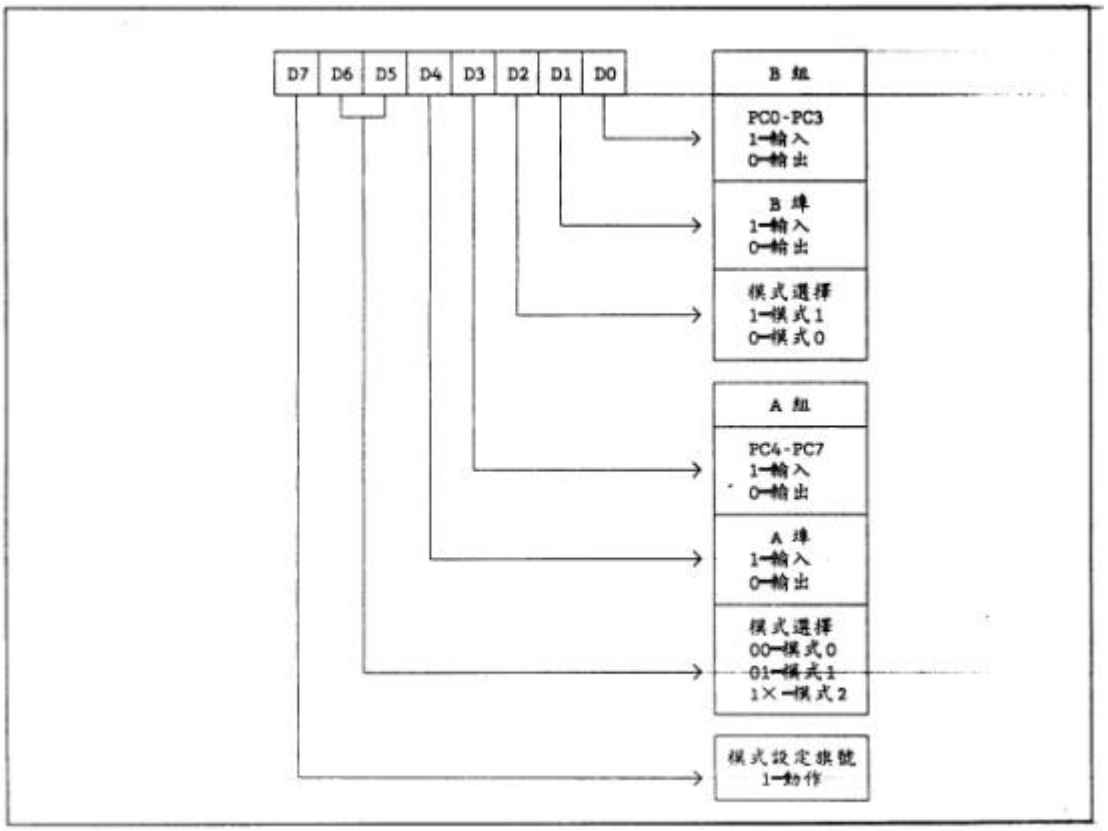
8255 有 4 個內部暫存器，分別是 A 埠暫存器、B 埠暫存器、C 埠暫存器及控制暫存器。當微電腦要讀寫 8255 的內部暫存器時，必須利用 A1 及 A0 指定要對那一個暫器進行讀寫動作。下表為 A1、A0 配合 $\overline{\text{RD}}$、$\overline{\text{WR}}$ 及 $\overline{\text{CS}}$ 的控制狀態表。

表 4-1 8255 控制狀態表

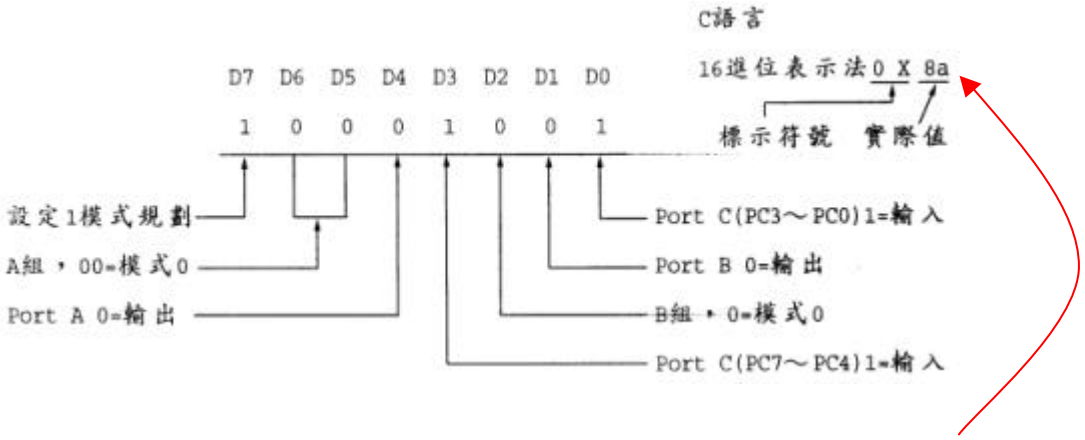
A1	A0	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{CS}}$	操 作 情 形
0	0	0	1	0	A 埠資料送到匯流排
0	1	0	1	0	B 埠資料送到匯流排
1	0	0	1	0	C 埠資料送到匯流排
0	0	1	0	0	匯流排資料存入 A 埠
0	1	1	0	0	匯流排資料存入 B 埠
1	0	1	0	0	匯流排資料存入 C 埠
1	1	1	0	0	匯流排資料存入控制暫存器
×	×	×	×	1	匯流排呈高阻抗
1	1	0	1	0	錯誤操作
×	×	×	1	0	匯流排呈高阻抗

當 8255 被重置後，會自行設定為工作在模式 0，並且 3 個 I/O 埠全部作為輸入使用。一個控制系統不一定要讓 8255 的 3 個 I/O 埠都作輸入使用，所以當要使用 8255 之前，必須先設定 8255 的內部控制暫存器，以決定 8255 要工作於那一種工作模式，每一個 I/O 埠是要作輸入或作輸出使用。下圖為 8255 控制暫存器的設定格式。

? **8255 內部控制暫存器說明:**



控制暫存器的設定格式



上例是將 8255A 設成模式 0，Port A、Port B 為輸出，Port C 為輸入。只要將 0x8A 寫入其控制暫存器即可。

利用 8255 內部控制暫存器的設定，8255 共有 3 種工作模式：

1. **模式 0：基本輸入／輸出**（Basic Input/Output）。
2. **模式 1：觸動式輸入／輸出**（Strobed Input/Output）。
3. **模式 2：觸動式雙向匯流排輸入／輸出**（Strobed Bidirectional Bus I/O）。

？ 8255 模式說明:

模式 0: 為最基本的 I/O 模式，其特性如下：

1. 任何埠都具有輸出及輸入功能。
2. 輸出時，各 Port 均有鎖定功能，能將信號鎖定在最後一次的輸出狀態上。
3. 輸入無鎖定功能，資料收到後，埠上的信號將不會保留最後一次輸入的狀態。
4. 有 2 個 8 位元埠（PA 和 PB）及 2 個 4 位元埠（PC0~PC3 和 PC4~PC7）可供利用。
5. 共可組成 16 種不同的輸出入狀態。

模式 1: 為觸動式輸入／輸出型態，又稱交握式傳輸（Hand shake)。在這種模式中，Port C 將被規劃當作 A 埠和 B 埠的交握控制訊號線，資料可從埠傳送到週邊，並等待週邊的知會訊號（ACK)，來告知是否收到。

例如要控制列表機時，第一筆資料送上列表機的傳輸線前，主機先會告知列表機並等待列表機回應（交握訊號)，等到回應出現，第一筆資料才開始傳送。

模式 1 特性如下：

1. 擁有 Port A 和 Port B 兩組 I/O 埠，C 被分為兩組（4bit)，分別作為 A 埠及 B 埠的控制線。

2. 當規劃為輸入時，週邊裝置傳到微處理機的資料必須由週邊控制電路產生觸動入訊號（Strobe Input，STB)，也就是回應訊號將資料鎖入。此時 8255 會自動產生輸入緩衝器已滿信號，通知週邊不要再送資料，直到 8255 允許才會繼續進行。

8255 模式 0 的 16 種工作情形

控制暫存器的內容	工 作 情 形			
D7 D6 D5 D4 D3 D2 D1 D0	PA0~PA7	PB0~PB7	PC0~PC3	PC4~PC7
1 0 0 0 0 0 0 0	輸出	輸出	輸出	輸出
1 0 0 0 0 0 0 1	輸出	輸出	輸入	輸出
1 0 0 0 0 0 1 0	輸出	輸入	輸出	輸出
1 0 0 0 0 0 1 1	輸出	輸入	輸入	輸出
1 0 0 0 1 0 0 0	輸出	輸出	輸出	輸入
1 0 0 0 1 0 0 1	輸出	輸出	輸入	輸入
1 0 0 0 1 0 1 0	輸出	輸入	輸出	輸入
1 0 0 0 1 0 1 1	輸出	輸入	輸入	輸入
1 0 0 1 0 0 0 0	輸入	輸出	輸出	輸出
1 0 0 1 0 0 0 1	輸入	輸出	輸入	輸出
1 0 0 1 0 0 1 0	輸入	輸入	輸出	輸出
1 0 0 1 0 0 1 1	輸入	輸入	輸入	輸出
1 0 0 1 1 0 0 0	輸入	輸出	輸出	輸入
1 0 0 1 1 0 0 1	輸入	輸出	輸入	輸入
1 0 0 1 1 0 1 0	輸入	輸入	輸出	輸入
1 0 0 1 1 0 1 1	輸入	輸入	輸入	輸入

模式 2: 稱為雙向觸動式 I/O 匯流排，在此種模式，A 埠可同時規劃為輸入及輸出，而 C 埠的五個位元作為閃控訊號。當 8255 處於模式 2 時，B 埠可獨立定義為模式 0 或 1，而 C 埠所剩下的非閃控訊號線的接腳，可視為另外的 3 個輸出或輸入位元，其特性如下：

1. 和第一種模式比較，它的軟體規劃較少，卻仍可達到相同的目的。
2. 微電腦由 A 埠寫入週邊的同時，也可利用 A 埠同時從週邊讀回一組資料。
3. 輸出及輸入均有閘門控制，可防止資料瞬間消失。

連絡方式

關於 PCI-TG DIO 卡在使用上如有任何問題，可先至汎捷科技網址查詢或歡迎來電詢問。

網址：http://www.soliton.com.tw
電話：03-656-6996
傳真：03-656-6883