PISO-CAN200/400
PISO-CAN100U/200U/400U/800U
PEX-CAN200i
PCM-CAN100/200/200P

使用者手冊

產品保固

凡泓格科技股份有限公司產品從購買即日起，若無任何材料性缺損保固一年。

免責聲明

凡使用本系列產品除品質量所造成的損害，泓格科技股份有限公司不承擔任何法律責任。泓格科技股份有限公司有義務提供本系列產品可靠而詳盡的資料，但保留修改權利，且不承擔使用者非法利用資料對第三方所造成侵害構成的法律責任。

版權

商標

手冊中所涉及所有公司商標，商標名稱以及產品名稱分別屬於該商標或名稱的擁有者所有。

目錄

1 產品資訊

1.1 簡介

CAN (Controller Area Network；控制器區域網路) 是一種串列式通訊協定，特別適合使用在主系統或子系統下提供更完整的智慧型網路設備如感應器及驅動器。它提供高安全等級及有效率的分散式即時控制，更具備了偵錯和優先權判別的機制。PISO-CAN、PEX-CAN與PCM-CAN使用獨立的CAN 控制器，每個CAN 卡可以有兩個或四個獨立的CAN匯流排通訊埠，端子部分使用 5-pin的螺釘接線端子或 9-pin D-sub公頭接線端子。它可以是主/從端介面，並且應用在不同的CAN 應用。另外，這些CAN卡使用NXP SJA1000T CAN控制器與82C250/251 或TJA1042 收發器，提供匯流排仲裁與錯誤偵測。這幾種CAN卡的差異在於電腦的插槽介面，一些是PCI介面、一些是PCI Express介面及一些是PCI-104 介面，要得知CAN卡詳細的特色與規格，請參考 1.2 節與 1.3 節。

1.2 特色

- PCI 匯流排介面
- 2500Vrms 光隔離保護
- 1/2/4/8 個獨立的 CAN 通訊埠
- 相容 CAN 2.0A 與 CAN 2.0B 的規範
- CAN bus 可程式傳輸速率高達 1 Mbps
- 可跳線選擇 120Ω(歐姆)終端電阻
- 直接由 CAN 控制器的記憶體存取資料
- PISO-CAN200/400
  - 33MHz 32bit 5V 隨插即用 PCI v2.1 匯流排
  - PCI 卡介面支援 5V PCI bus
  - 3 kV_{DC} 隔離保護
  - 具有 2/4 個獨立的 CAN 通道
- PISO-CAN100U200U400U/800U
  - 符合 PCI v2.2 32-bit 33MHz
  - Universal PCI 卡、支援 5V 與 3.3V PCI 匯流排
  - 3 kV_{DC} 隔離保護
  - 具有 1/2/4/8 個獨立的 CAN 通道
- PEX-CAN200i
  - 32-bit, 33MHz, X1 PCI Express 匯流排
  - PCI Express R1.0 規格
  - 3 kV_{DC} 隔離保護
  - 具有 2 個獨立 CAN 通道
- PCM-CAN100/200/200P
  - 符合 PCI104 規格
  - 9-pin D-sub 公連接端子
  - 1 kV_{DC} 隔離保護
  - 具有 1/2 個獨立 CAN 通道
- 驅動程式支援 Windows 2000/X/P/7 和 Linux 2.6.x ~ 3.2.20 環境

1.3 硬體規格

1.3.1 PCM-CAN100/200/200P

模組名稱	PCM-CAN100-D	PCM-CAN200-D	PCM-CAN200P-D
匯流排介面			
種類	PCI-104		PC/104-Pluse
CAN 介面			
控制器	NXP SJA1000T 搭配 16MHz 震盪器		
收發器	NXP 82C250		
通道數	1	2	
接頭	9 針公/母座 D-Sub	9 針公座 D-Sub	
	(CAN_L, CAN_SHLD, CAN_H, 其餘腳位空接)		
鮑率(bps)	10 k, 20 k, 50 k, 125 k, 250 k, 500 k, 800 k, 1 M (允許使用者自訂義鮑率)		
終端電阻	跳線設定 120 Ω 終端電阻		
電源			
功耗	250 mA @ 5 V		
機構			
尺寸	91mm x 22mm x 96mm (寬 x 長 x 高)		
環境			
工作溫度	0 ~ 60 ℃		
儲存溫度	-20 ~ 70 ℃		
濕度	相對溼度 5 ~ 85% RH, 無結露		

1.3.2 PEX-CAN200i

模組名稱	PEX-CAN200i-D	PEX-CAN200i-T
匯流排介面		
類型	PCIe x 1	
CAN 介面		
控制器	NXP SJA1000T 搭配 16MHz 震盪器	
收發器	NXP 82C250	
通道數	2	
接頭	9 針公座 D-Sub	5 針螺絲端子
	(CAN_L, CAN_SHLD, CAN_H, 其餘腳位空接)	
鮑率(bps)	10 k, 20 k, 50 k, 125 k, 250 k, 500 k, 800 k, 1 M (允許使用者自訂義鮑率)	
終端電阻	跳線設定 120 Ω 終端電阻	
電源		
功耗	100 mA @ 12 V, 100 mA @ 3.3 V	
機構		
尺寸	120mm x 22mm x 85mm (寬 x 長 x 高)	
環境		
工作溫度	0 ~ 60 ℃	
儲存溫度	-20 ~ 70 ℃	
濕度	相對溼度 5 ~ 85% RH, 無結露	

1.3.3 PISO-CAN200/200U

模組名稱	PISO-CAN200-D PISO-CAN200-T		PISO-CAN200U-D	PISO-CAN200U-T
匯流排介面				
類型	PCI 介面, 5 V 訊號, 33 MHz, 32 位元, 隨插即用		Universal PCI 介面, 支援 3.3 V 與 5 V 訊號, 33 MHz, 32 位元, 隨插即用	
CAN 介面				
控制器	NXP SJA1000T 搭配 16MHz 震盪器			
收發器	NXP 82C250			
通道數	2			
接頭	9 針公座 D-Sub	5 針螺絲端子	9 針公座 D-Sub	5 針螺絲端子
	(CAN_L, CAN_SHLD, CAN_H, 其餘腳位空接)			
鮑率(bps)	10 k, 20 k, 50 k, 125 k, 250 k, 500 k, 800 k, 1 M (允許使用者自訂義鮑率)			
終端電阻	跳線設定 120 Ω 終端電阻			
電源				
功耗	250 mA @ 5 V			
機構				
尺寸	126mm x 22mm x 85mm (寬 x 長 x 高)			
環境				
工作溫度	0 ~ 60 ℃			
儲存溫度	-20 ~ 70 ℃			
濕度	相對溼度 5 ~ 85% RH, 無結露			

1.3.4 PISO-CAN400/400U

模組名稱	PISO-CAN400U-D	PISO-CAN400U-T	PISO-CAN400U-D	PISO-CAN400U-T
匯流排介面				
類型	PCI 介面, 5 V 訊號, 33 MHz, 32 位元, 隨插即用		Universal PCI 介面, 支援 3.3 V 與 5 V 訊號, 33 MHz, 32 位元, 隨插即用	
CAN 介面				
控制器	NXP SJA1000T 搭配 16MHz 震盪器			
收發器	NXP 82C250			
通道數	4			
接頭	9 針公座 D-Sub	5 針螺絲端子	9 針公座 D-Sub	5 針螺絲端子
	(CAN_L, CAN_SHLD, CAN_H, 其餘腳位空接)			
鮑率(bps)	10 k, 20 k, 50 k, 125 k, 250 k, 500 k, 800 k, 1 M (允許使用者自訂義鮑率)			
終端電阻	跳線設定 120 Ω 終端電阻			
電源				
功耗	300 mA @ 5 V			
機構				
尺寸	126mm x 22mm x 85mm (寬 x 長 x 高)			
環境				
工作溫度	0 ~ 60 ℃			
儲存溫度	-20 ~ 70 ℃			
濕度	相對溼度 5 ~ 85% RH, 無結露			

1.3.5 PISO-CAN100U

模組名稱	PISO-CAN100U-D	PISO-CAN100U-T
匯流排介面		
類型	Universal PCI 介面，支援 3.3 V 與 5 V 訊號, 33 MHz, 32 位元，隨插即用	
CAN 介面		
控制器	NXP SJA1000T 搭配 16MHz 震盪器	
收發器	NXP 82C250	
通道數	1	
接頭	9 針公座 D-Sub	5 針螺絲端子
	(CAN_L, CAN_SHLD, CAN_H，其餘腳位空接)	
鮑率(bps)	10 k, 20 k, 50 k, 125 k, 250 k, 500 k, 800 k, 1 M (允許使用者自訂義鮑率)	
終端電阻	跳線設定 120 Ω 終端電阻	
電源		
功耗	225 mA @ 5 V	
機構		
尺寸	126mm x 22mm x 85mm (寬 x 長 x 高)	
環境		
工作溫度	0 ~ 60 ℃	
儲存溫度	-20 ~ 70 ℃	
濕度	相對溼度 5 ~ 85% RH，無結露	

1.3.6 PISO-CAN800U

匯流排介面	
類型	Universal PCI 介面，支援 3.3 V 與 5 V 訊號, 33 MHz, 32 位元，隨插即用
CAN 介面	
控制器	NXP SJA1000T 搭配 16MHz 震盪器
收發器	NXP TJA1042
通道數	8
接頭	37 針公座 D-Sub x 2
鮑率 (bps)	10 k, 20 k, 50 k, 125 k, 250 k, 500 k, 800 k, 1 M (允許使用者自訂義鮑率)
終端電阻	跳線設定 120 Ω 終端電阻
電源	
功耗	800 mA @ 5 V
機構	
尺寸	193mm x 22mm x 93mm (寬 x 長 x 高)
環境	
工作溫度	0 ~ 60 ℃
儲存溫度	-20 ~ 70 ℃
濕度	相對溼度 5 ~ 85% RH，無結露

1.4 產品檢查清單

除了手冊外，包裝內含如下項目:

- □ 一塊 PISO-CAN 或 PEX-CAN200 或 PCM-CAN 系列 CAN 卡
- □ ADP-9 板卡 (僅適用於 PISO-CAN400/PISO-CAN400U)
- □ 軟體光碟

首先，建議使用者先閱讀產品發表聲明檔案，檔案中提供下列重要資訊：

- □ 驅動程式、工具軟體及範例程式路徑。
- □ 如何安裝驅動程式與工具軟體
- □ 診斷程式路徑
- □ 常見問題與解答

注意！

假如這些任何配件有任何遺失或損害，請聯繫當地的代理商。請保留出貨的相關配件和盒子，以方便您未來要寄送或存放產品.

2 硬體配置

這一節將說明CAN網路中，PISO-CAN、PEX-CAN及PCM-CAN系列的硬體設定、線路連接與終端電阻的架構。

2.1 板卡元件分佈圖

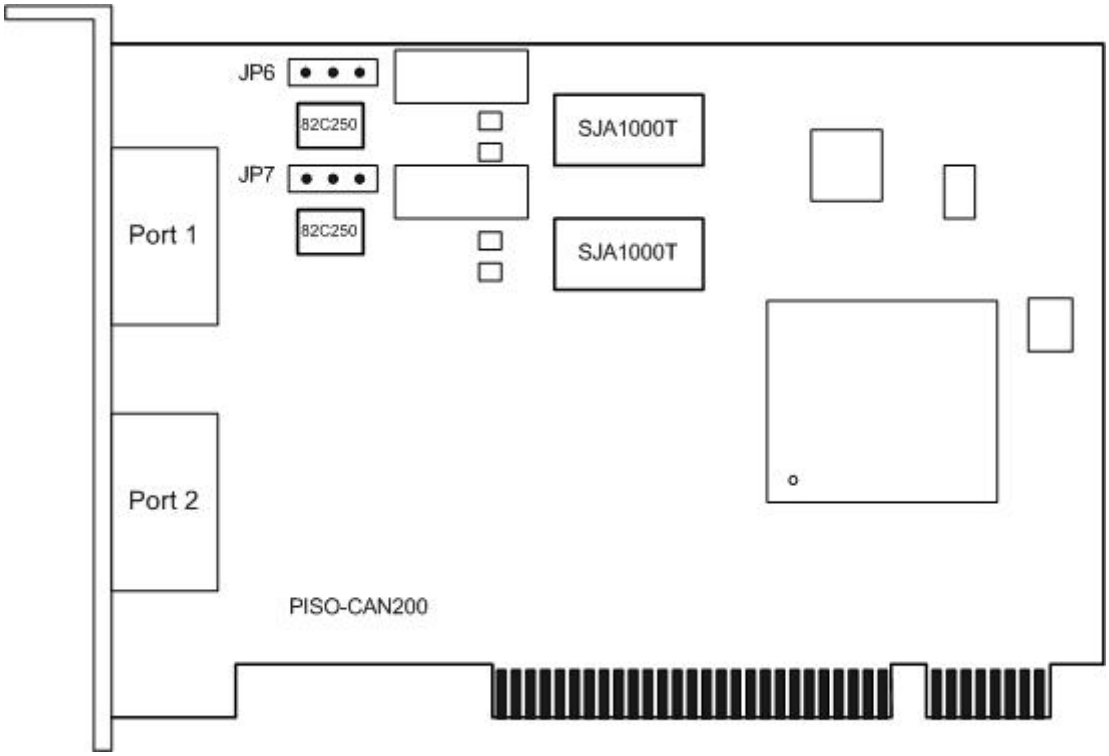


圖 2.1　PISO-CAN200 板卡元件分佈圖

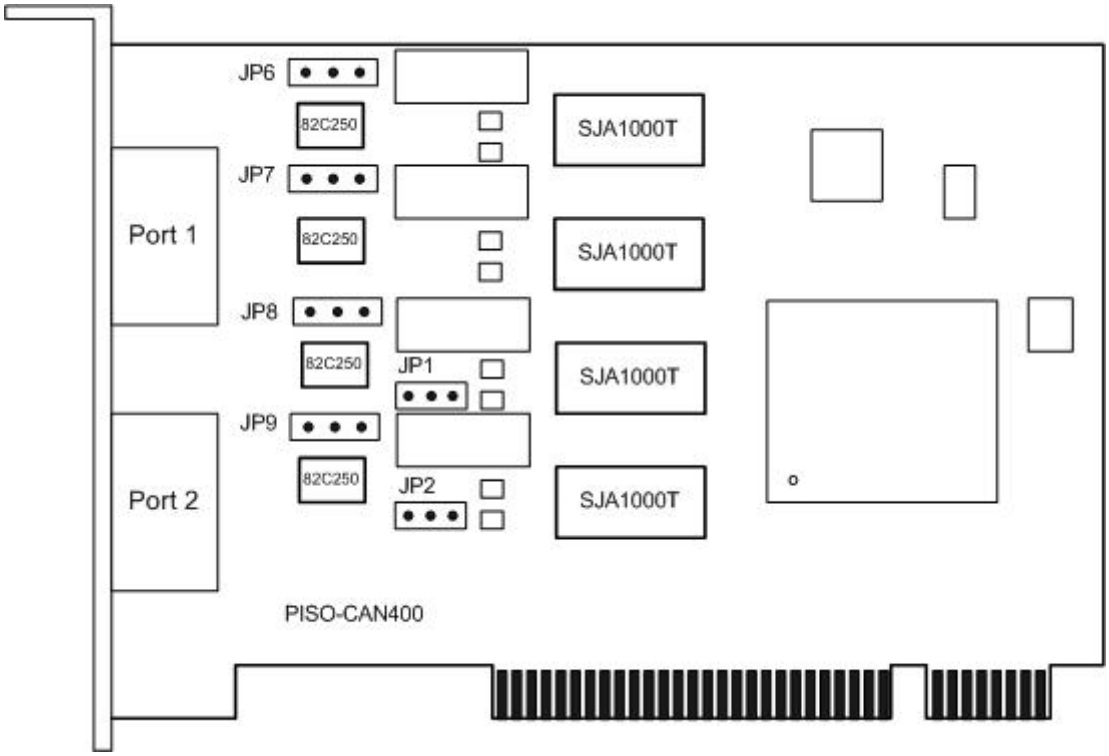


圖 2.2　PISO-CAN400 板卡元件分佈圖

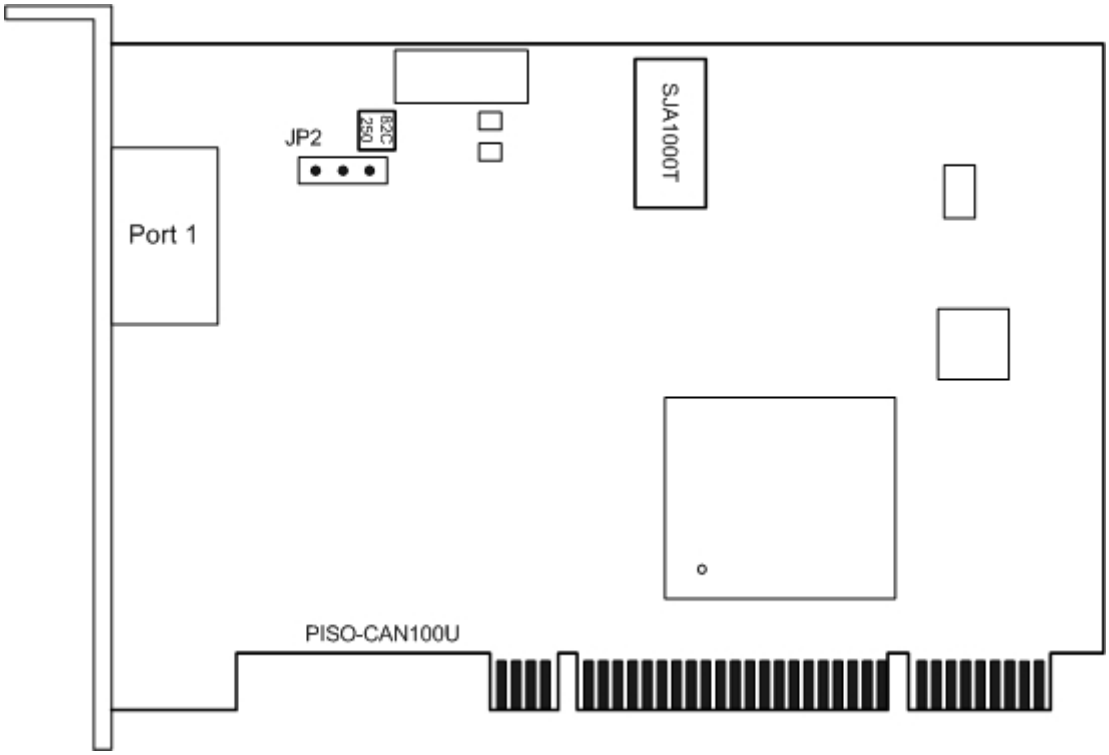


圖 2.3　PISO-CAN100U 板卡元件分佈圖

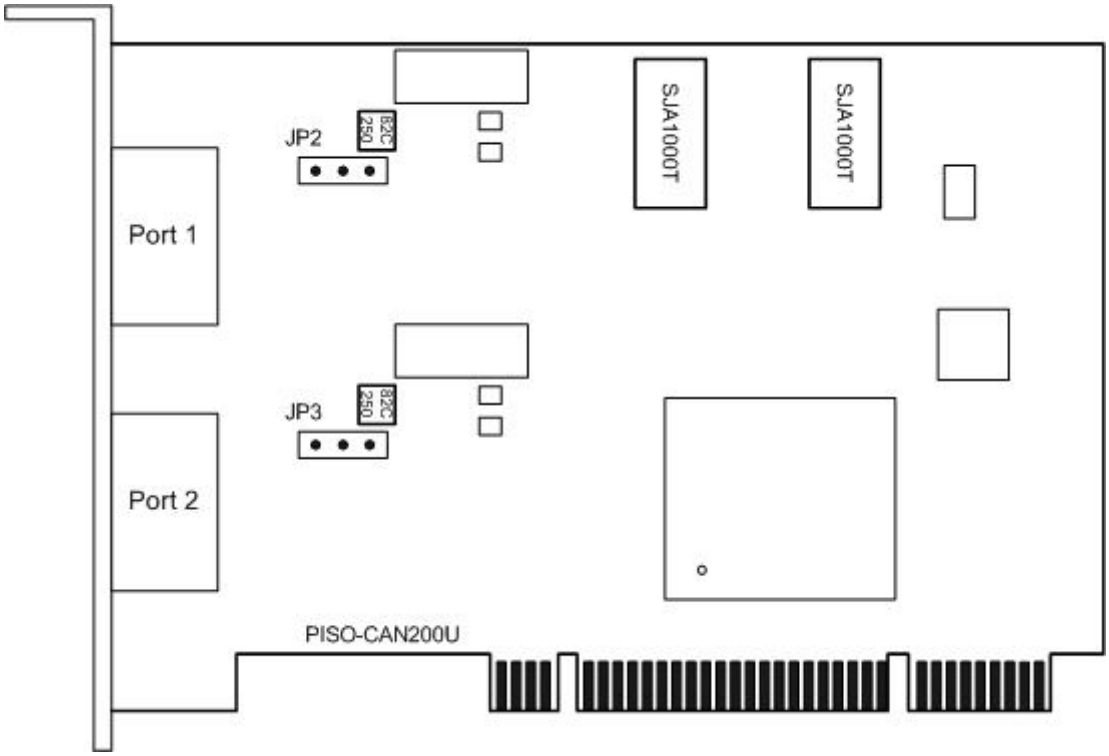


圖 2.4　PISO-CAN200U 板卡元件分佈圖

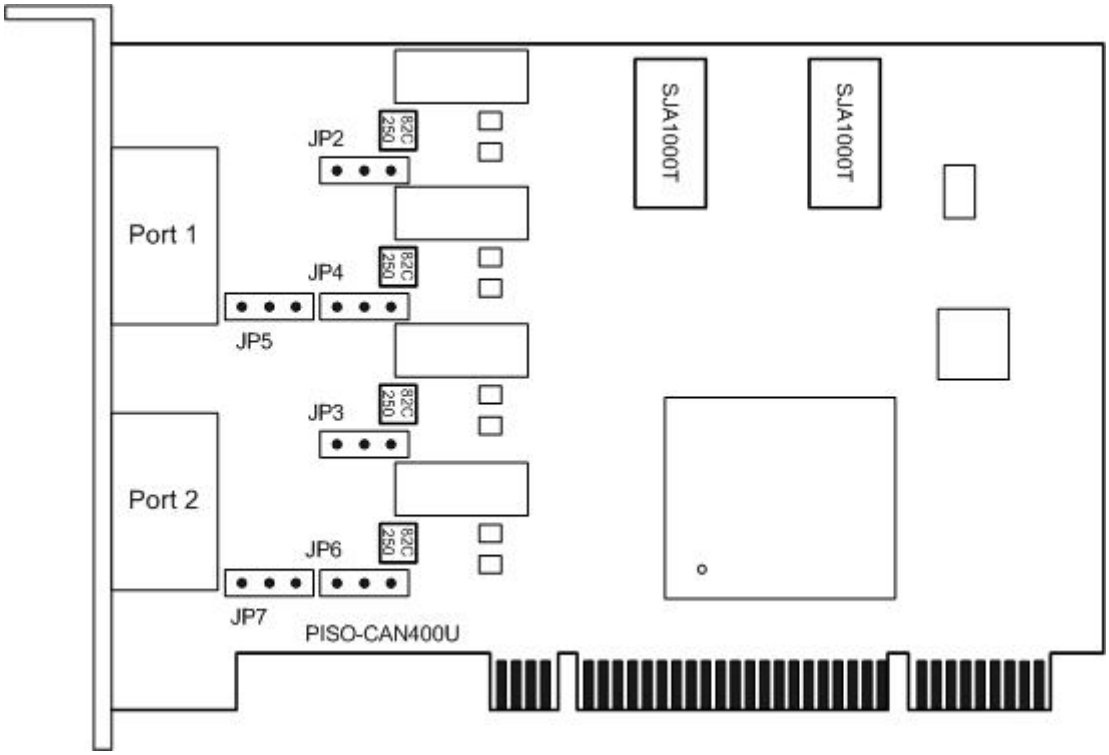


圖 2.5　PISO-CAN400U 板卡元件分佈圖

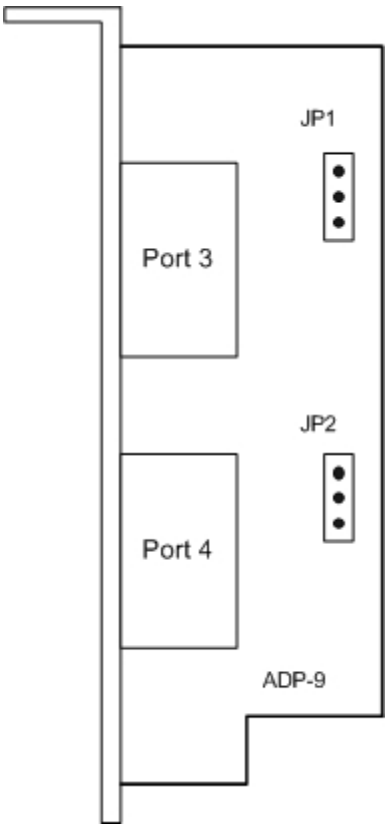


圖 2.6　ADP-9 板卡元件分佈圖(適用於 PISO-CAN400/PISO-CAN400U)

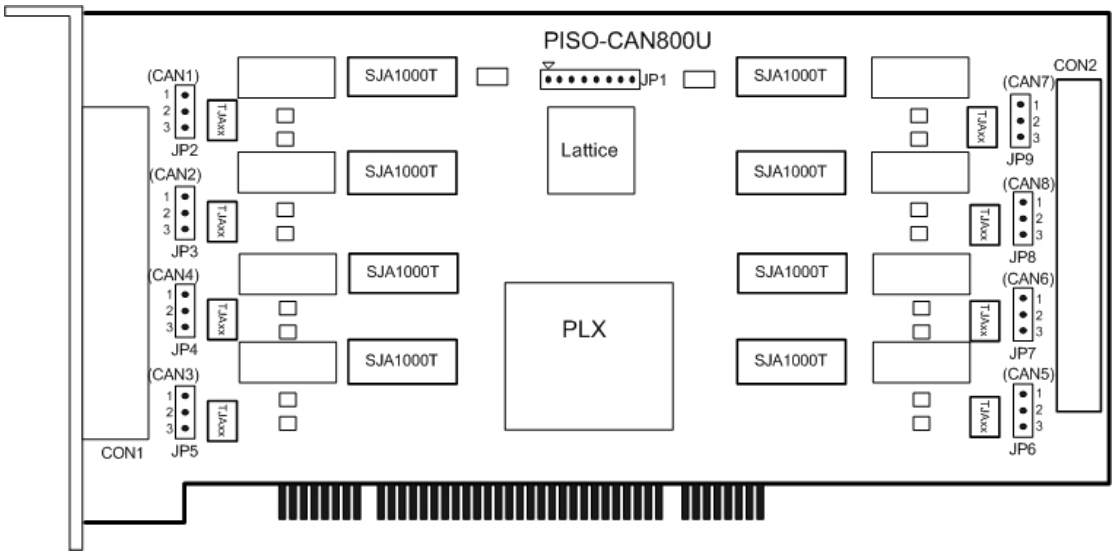


圖 2.7　PISO-CAN800U 板卡元件分佈圖

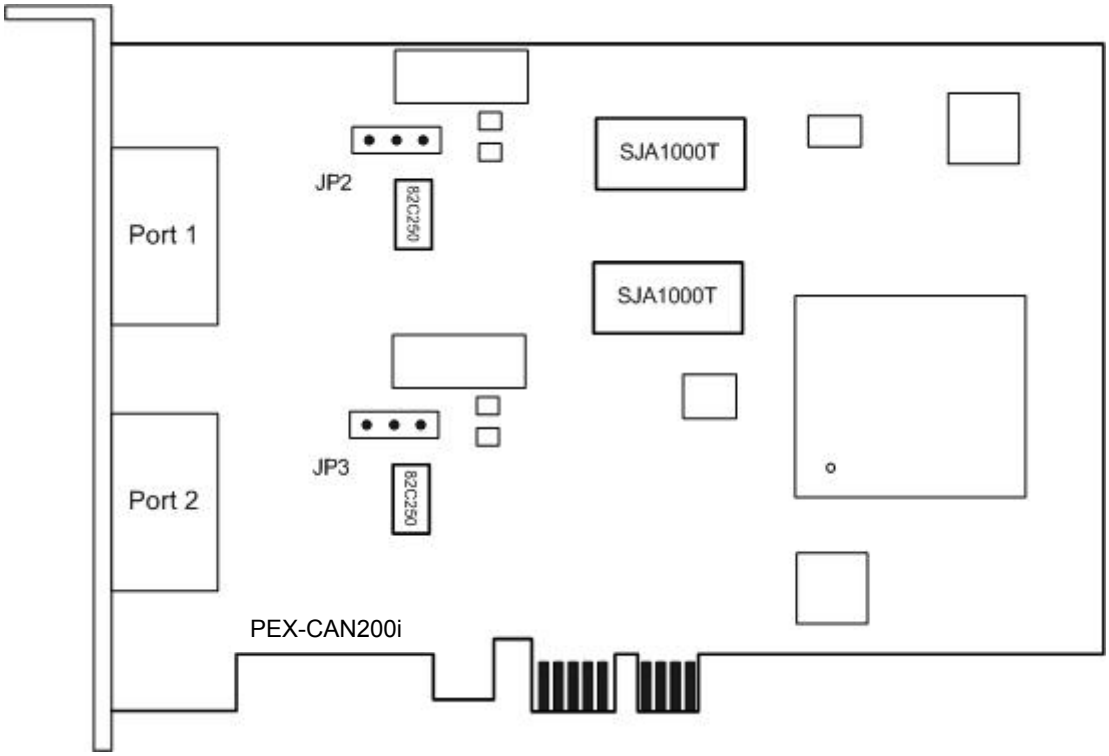


圖 2.8　PEX-CAN200i 板卡元件分佈圖

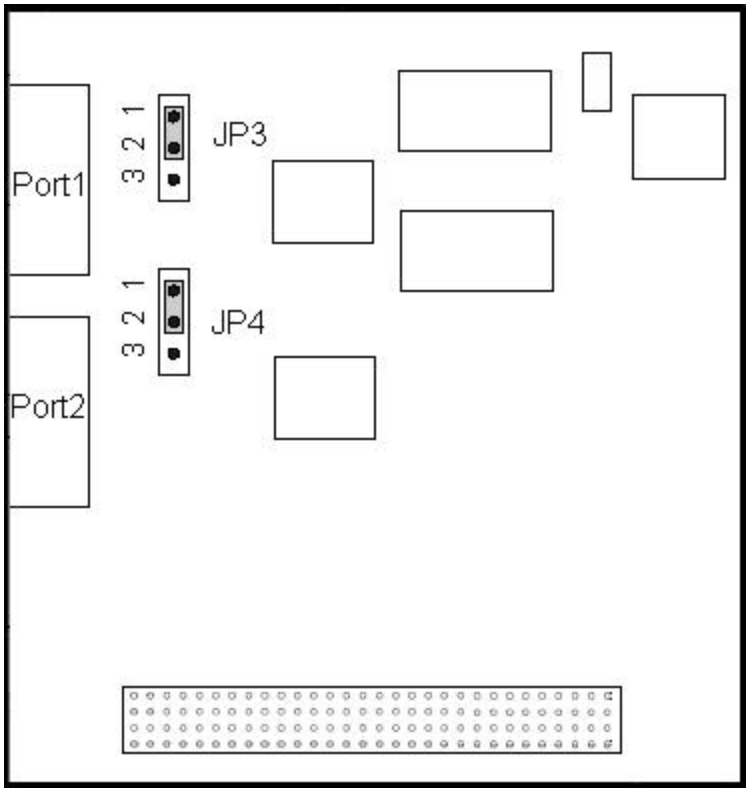


圖 2.9 PCM-CAN200 板卡元件分佈圖

2.2 跳線選擇

PISO-CAN200/400

表 2.1 跳線選擇

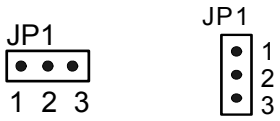
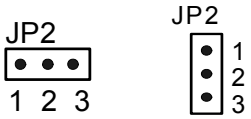
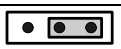
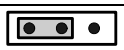
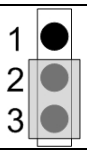
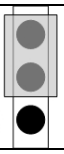
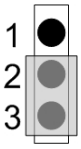
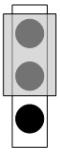
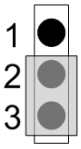
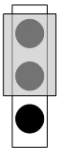
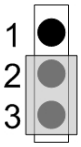
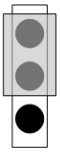
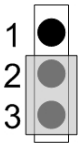
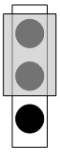
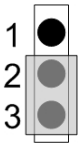
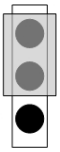
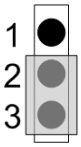
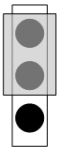
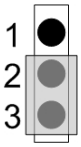
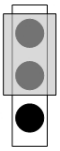
跳線	描述	狀態	
JP1	CAN Port 3 端子連接 PISO-CAN400 板卡與 ADP-9 板卡。	 Pin1: CAN_L Pin2: CAN_H Pin3: Shield	
JP2	CAN Port 4 端子連接 PISO-CAN400 板卡與 ADP-9 板卡。	 Pin1: CAN_L Pin2: CAN_H Pin3: Shield	
JP6	Port 1 終端電阻(120Ω)選擇	開啟 	關閉 
JP7	Port 2 終端電阻(120Ω)選擇 ()		
JP8	Port 3 終端電阻(120Ω)選擇		
JP9	Port 4 終端電阻(120Ω)選擇		

表 2.2 跳線選擇

跳線	描述	狀態	
JP5	CAN Port 3 端子連接 PISO-CAN400U 板卡與 ADP-9 板卡。	 Pin1: CAN_L Pin2: CAN_H Pin3: Shield	
JP7	CAN Port 4 端子連接 PISO-CAN400U 板卡與 ADP-9 板卡。	 Pin1: CAN_L Pin2: CAN_H Pin3: Shield	
JP2	Port 1 終端電阻(120Ω)選擇	開啟 	關閉 
JP3	Port 2 終端電阻(120Ω)選擇 (PISO-CAN200U/400U 使用)		
JP4	Port 3 終端電阻(120Ω)選擇 (PISO-CAN400U 使用)		
JP6	Port 4 終端電阻(120Ω)選擇 (PISO-CAN400U 使用)		

PISO-CAN800U

Table 2.3 Jumper Selections

Jumper	Description	Status	
		Enable	Disable
JP2	Port 1 terminator resister(120Ω) selection		
JP3	Port 2 terminator resister(120Ω) selection		
JP4	Port 4 terminator resister(120Ω) selection		
JP5	Port 3 terminator resister(120Ω) selection		
JP6	Port 5 terminator resister(120Ω) selection		
JP7	Port 6 terminator resister(120Ω) selection		
JP8	Port 8 terminator resister(120Ω) selection		
JP9	Port 7 terminator resister(120Ω) selection		

PEX-CAN200i-D/T

表 2.4 跳線選擇

跳線	描述	狀態	
		開啟	關閉
JP2	Port 1 終端電阻(120Ω)選擇	 1 2 3	 1 2 3
JP3	Port 2 終端電阻(120Ω)選擇	 1 2 3	 1 2 3

PCM-CAN100/200/200P

表 2.3 顯示了每個模組的相對應開關與訊號設定。

表 2.5:旋轉開關設定

開關位置	插槽模組	CLK	ID 選擇	INT
0 or 4 or 8	1	CLK0	IDSEL0	INTA
1 or 5 or 9	2	CLK1	IDSEL1	INTB
2 or 6	3	CLK2	IDSEL2	INTC
3 or 7	4	CLK3	IDSEL3	INTD

表 2.6 跳線選擇

跳線	說明	狀態	
		開啟	關閉
JP3	Port 1 終端電阻(120Ω)選擇	 3 2 1	 3 2 1
JP4 (適用於 PCM-CAN200)	Port 2 終端電阻(120Ω)選擇	 3 2 1	 3 2 1

2.3 腳位定義

PISO-CAN200/400-T, PISO-CAN100U/200U/400U-T和 PEX-CAN200i-T 配備有 1/2/4 套 **5 針螺釘接線端子**，PISO-CAN200/400-D, PISOCAN100U/200U/400U-D、PEX-CAN200i-D和PCM-CAN100/200 配備有 1/2/4 套 **9 針D-sub 公頭接線端子**，PISO-CAN800U配備有 2 套 37 針**D-sub 母頭接線端子，搭配使用CA-9-3715D/CA-9-3705 轉換線**, 使用者可以將37針母頭接線端子轉成多個9針公頭接線端子, 用於CAN bus的電線連接，而接線端子的腳位定義詳細的說明如下：

2.3.1 5 針螺釘接線端子

CAN 匯流排接口的 5-pin 螺釘接線端子如圖 2.8 所示，且對應的腳位定義如表 2.6 所示。

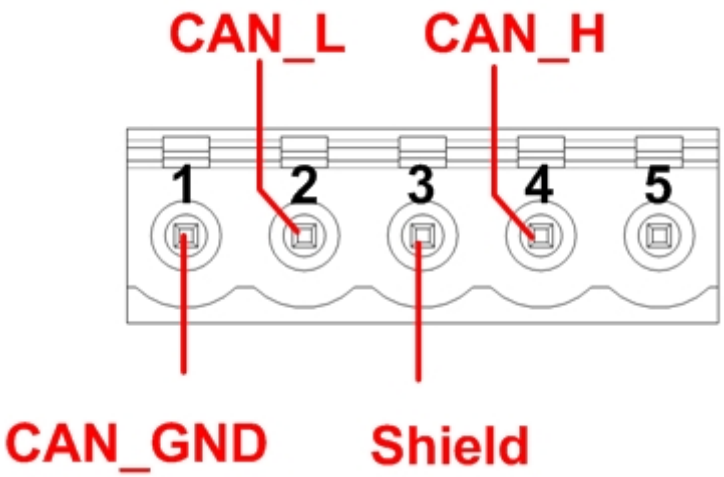


圖 2.9　5-pin 螺釘接線端子

表 2.6　5-pin 螺釘接線端子腳位定義

5-pin 螺釘接線端子腳位定義	
1	CAN_GND
2	CAN_L
3	CAN_SHLD
4	CAN_H
5	保留

2.3.2 9 針 D-sub 公頭接線端子

CAN 匯流排接口的 9-pin D-sub 公頭接線端子如圖 2.9 所示，且對應的腳位定義如表 2.7 所示。

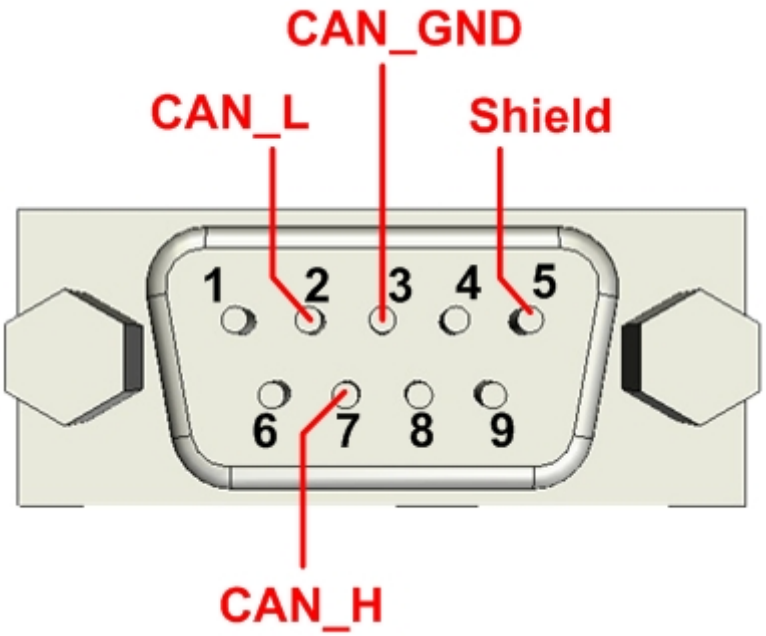


圖 2.10 9-pin D-sub 公頭接線端子

表 2.7 9-pin D-sub 公頭接線端子腳位定義

D-sub 公頭接線端子腳位定義	
1	保留
2	CAN_L
3	CAN_GND
4	保留
5	CAN_SHLD
6	保留
7	CAN_H
8	保留
9	保留

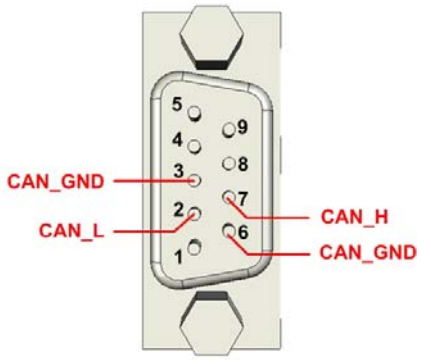
2.3.3 37 針 D-sub 母頭接線端子

PISO-CAN800U (CON1) 轉 DB-9 接腳定義

(使用 CA-9-3715D/CA-9-3705)

Pin Assignment Name	Terminal No.		Pin Assignment Name
CAN1_GND	19	37	CAN1_L
CAN1_H	18	36	N.C.
CAN1_GND	17	35	N.C.
N.C.	16	34	N.C.
N.C.	15	33	CAN2_GND
CAN2_L	14	32	CAN2_H
N.C.	13	31	CAN2_GND
N.C.	12	30	N.C.
N.C.	11	29	N.C.
CAN4_GND	10	28	CAN4_L
CAN4_H	09	27	N.C.
CAN4_GND	08	26	N.C.
N.C.	07	25	N.C.
N.C.	06	24	CAN3_GND
CAN3_L	05	23	CAN3_H
N.C.	04	22	CAN3_GND
N.C.	03	21	N.C.
N.C.	02	20	N.C.
N.C.	01		

37-Pin Female D-Sub Connector_CAN (CON1)



DB-37 to Male DB-9 Connector_CAN

PISO-CAN800U (CON2) 轉 DB-37 接腳定義

CON2			
1	DB-37_Pin01	2	DB-37_Pin20
3	DB-37_Pin02	4	DB-37_Pin21
5	DB-37_Pin03	6	DB-37_Pin22
7	DB-37_Pin04	8	DB-37_Pin23
9	DB-37_Pin05	10	DB-37_Pin24
11	DB-37_Pin06	12	DB-37_Pin25
13	DB-37_Pin07	14	DB-37_Pin26
15	DB-37_Pin08	16	DB-37_Pin27
17	DB-37_Pin09	18	DB-37_Pin28
19	DB-37_Pin10	20	DB-37_Pin29
21	DB-37_Pin11	22	DB-37_Pin30
23	DB-37_Pin12	24	DB-37_Pin31
25	DB-37_Pin13	26	DB-37_Pin32
27	DB-37_Pin14	28	DB-37_Pin33
29	DB-37_Pin15	30	DB-37_Pin34
31	DB-37_Pin16	32	DB-37_Pin35
33	DB-37_Pin17	34	DB-37_Pin36
35	DB-37_Pin18	36	DB-37_Pin37
37	DB-37_Pin19	38	N.C.
39	N.C.	40	N.C.

Pin Assignment Name	Terminal No.		Pin Assignment Name
CAN5_GND	19	37	CAN5_L
CAN5_H	18	36	N.C.
CAN5_GND	17	35	N.C.
N.C.	16	34	N.C.
N.C.	15	33	CAN6_GND
CAN6_L	14	32	CAN6_H
N.C.	13	31	CAN6_GND
N.C.	12	30	N.C.
N.C.	11	29	N.C.
CAN8_GND	10	28	CAN8_L
CAN8_H	09	27	N.C.
CAN8_GND	08	26	N.C.
N.C.	07	25	N.C.
N.C.	06	24	CAN7_GND
CAN7_L	05	23	CAN7_H
N.C.	04	22	CAN7_GND
N.C.	03	21	N.C.
N.C.	02	20	N.C.
N.C.	01		

37-Pin Female D-Sub Connector_CAN (CON2)

2.4 硬體安裝

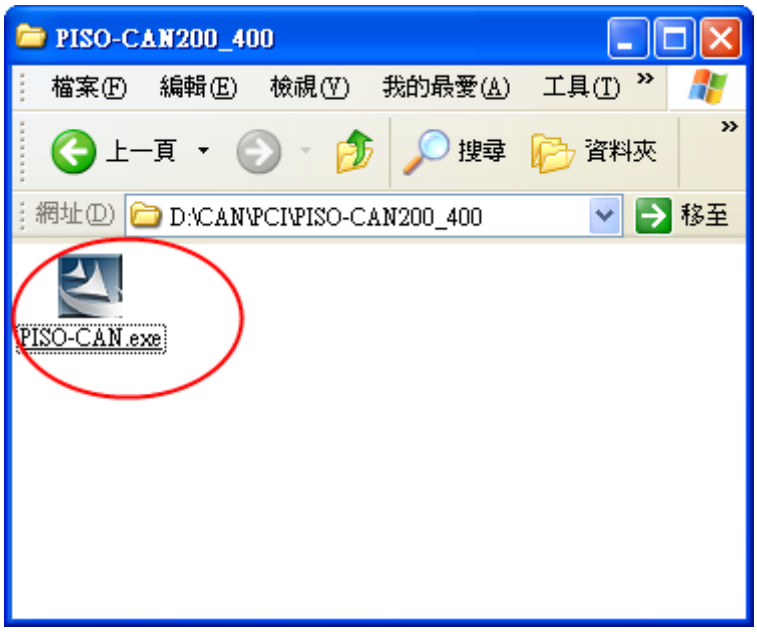
1. 在 PISO-CAN/PEX-CAN/PCM-CAN 上更改跳線設定，以符合您特定的要求。
2. 關閉電源並且移除電腦上蓋。
3. 安裝 PISO-CAN、PEX-CAN 或 PCM-CAN 系列的 CAN 卡，至適合的空 PCI 插槽。
4. 裝回電腦上蓋。
5. 安裝 CAN bus 接線至 5-pin 螺釘接頭或 9-pin D-sub 接頭。
6. 當硬體安裝完成後，請開啟電腦電源。

3 軟體安裝

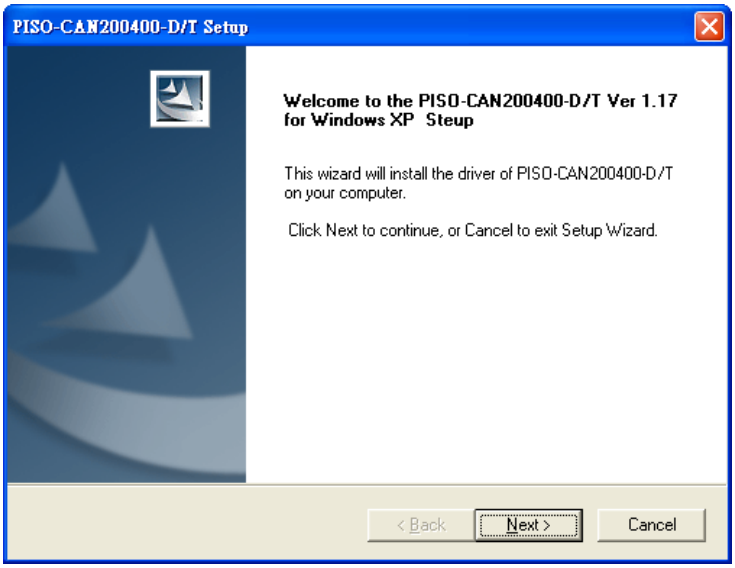
PISO-CAN 或 PCM-CAN 驅動程式可用於 Windows 2K/XP 之環境，使用者可以從軟體光碟的 “\CAN\PCI\PCM_PISO-CAN_series\driver” 路徑中，找到並執行 “PISO-CAN.exe” 來開始安裝驅動程式。

安裝PISO-CAN 或 PCM-CAN 卡驅動程式

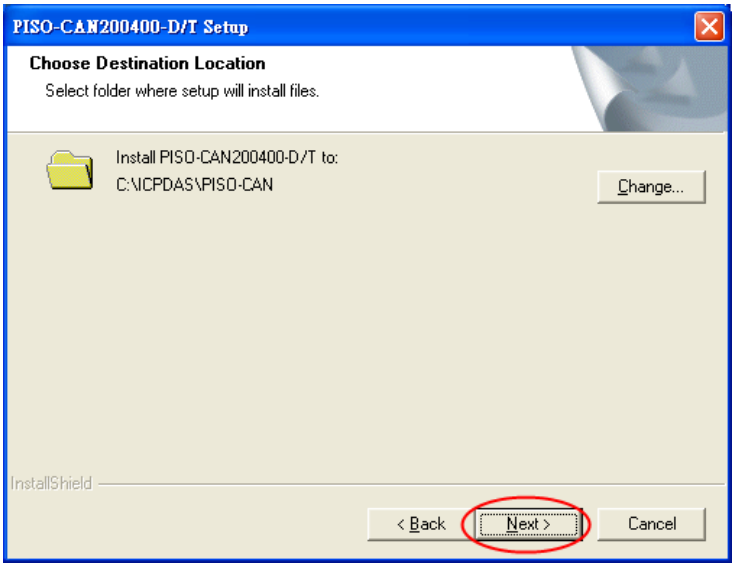
Step 1: 放置產品光碟至光碟機並且於路徑 “\CAN\PCI\PCM_PISO-CAN_series\driver\win_2k_xp_7” (例如:作業系統為 Windows 2000/XP/7)下找到 “PISO-CAN.exe” 並且執行。



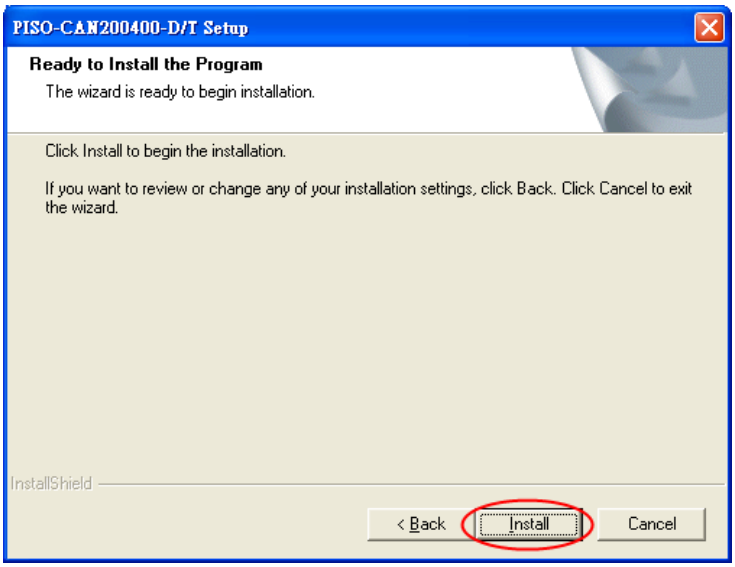
Step 2: 點擊「Next」開始安裝 PISO-CAN。



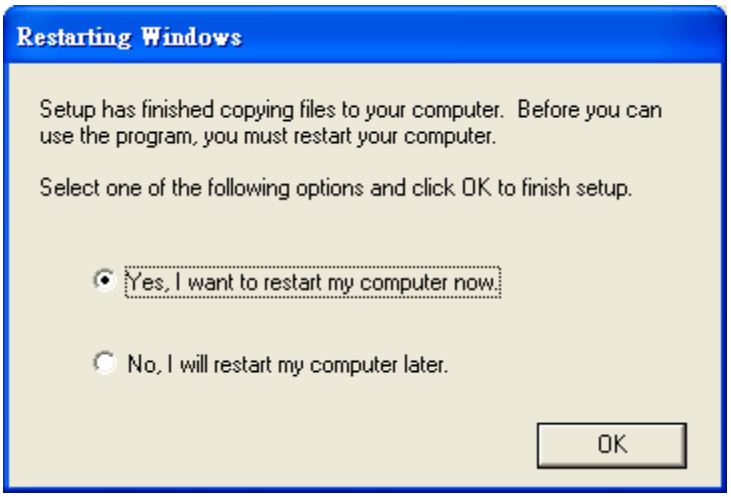
Step 3: 選擇 PISO-CAN 安裝資料夾，並點擊「Next」繼續安裝。



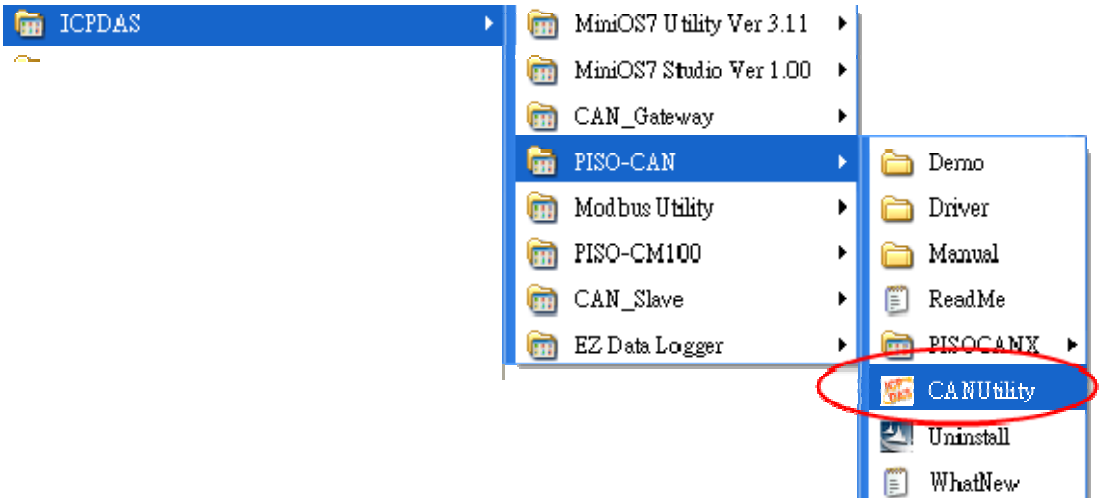
Step 4: 點擊「Install」繼續安裝。



Step 5: 最後，重新啟動電腦以完成安裝。

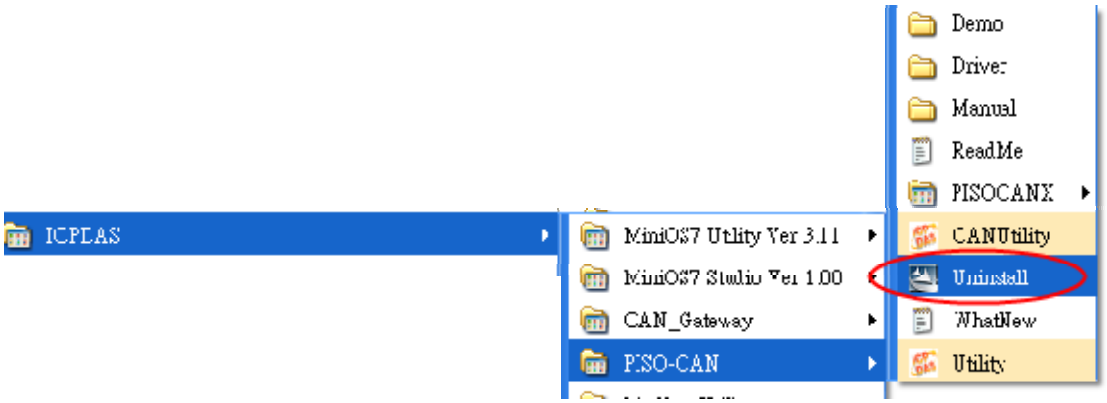


當安裝完成後，在 Windows “開始”啟動列中，可找到 PISO-CAN 資料夾，如下圖。



移除 PISO-CAN 驅動程式

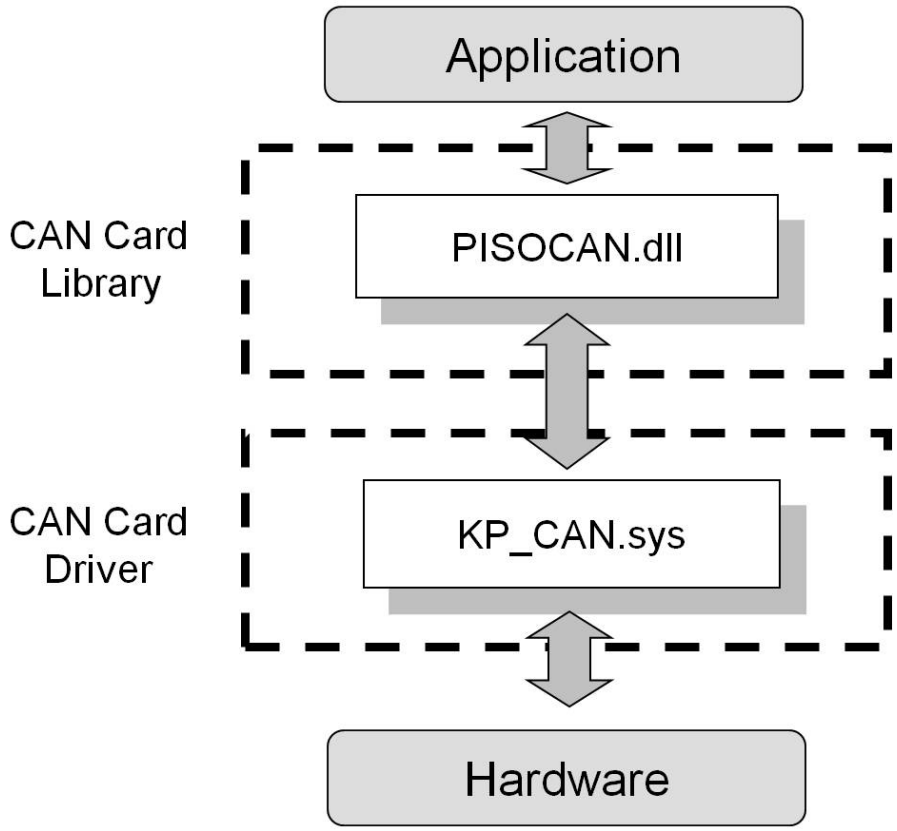
如果使用者要移除 PISO-CAN 驅動程式，請點擊「Uninstall」，如下圖。



4 DLL 驅動程式安裝

Windows DLL 驅動程式

DLL 驅動程式是用於 Windows 2000/XP/7 環境中，能呼叫 PISO-CAN、PEX-CAN 和 PCM-CAN 系列的函式庫，其應用架構如下圖所示。使用者的應用程式可以使用 VB、VC、Delphi 及 Borland C++ Builder…等開發工具在使用者模式呼叫 “PISOCAN.DLL” 驅動程式。DLL 驅動程式會將函式呼叫的指令傳送至 KP_CAN.sys，以存取硬體系統，如下圖所示。



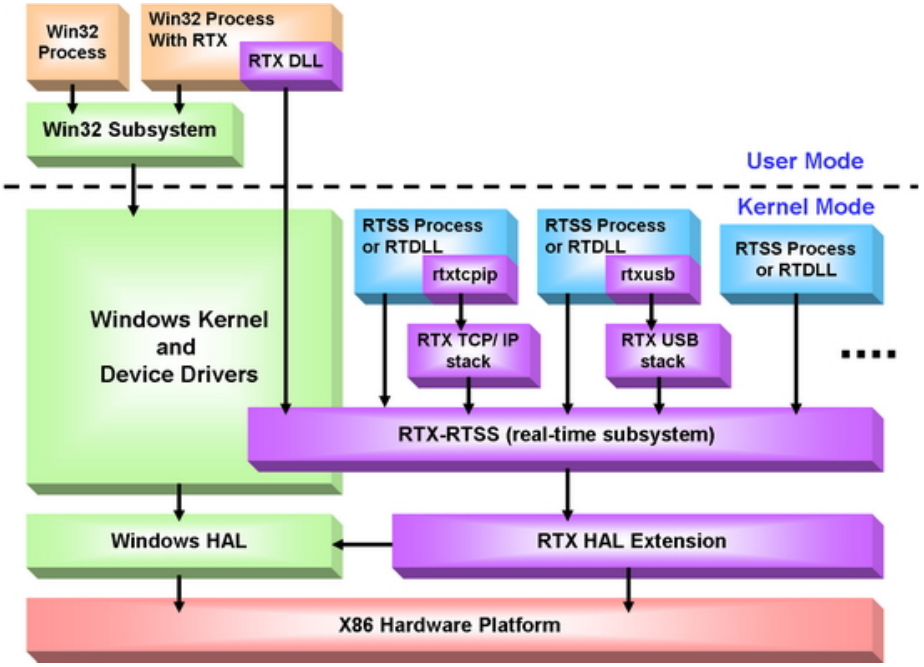
RTX 驅動程式

為了滿足用戶應用 RTX 系統，泓格提供了 PISO-CAN 系列 CAN 卡的 RTX 驅動程式，如果用戶想 CAN 通信接口結合起來至他們的時間關鍵型(time-critical)系統中，PISO-CAN 系列的 CAN 卡 RTX 驅動程式可以幫助他們輕鬆快速地做到這一點。此外，RTX 驅動程式 API 的名稱和參數都是與 Windows 驅動程序相同的，如果用戶之前使用的是 Windows 驅動程序，他們將不需要付出更多的努力學習如何使用 RTX 驅動程式的 API。RTX 驅動程式增加了 PISO-CAN 系列 CAN 卡的附加價值，並滿足用戶獲得高度的實時系統。由於高性價比和高實時性的特點，PISO-CAN 系列 CAN 卡將更廣泛的使用在 CAN Bus 的應用環境上。

特點：

1. RTX 驅動程式的 API 的名稱和參數與 Windows 驅動程序相同。如果用戶先前已使用 PISO-CAN 系列 CAN 卡的話，將不需要學習新的 API 的使用方式
2. 如果 PISO-CAN 系列 CAN 卡可以得到獨立的 IRQ，它將支持中斷功能
3. 直接 I/ O 控制和高實時性
4. 支持在 Windows2000 SP4 和 Windows XP SP2 操作系統
5. 支持 RTX 8.1 和 2011 版本

備註:
在執行 PISO-CAN 系列 RTX 的 API 之前，用戶需要先執行 “pisocan_rtx.rtss” 文件。



4.1 DLL 函式定義與描述

表 4.1 與表 4.2 列出在 PISO-CAN、PEX-CAN 或 PCM-CAN **(以下簡稱 PISO-CAN)**中，提供的所有函式，且每一個函式的詳細資訊會在接下來的小節介紹。然而，為了使描述更為簡化與清楚，分別以 **[input]** 和 **[output]** 表示輸入與輸出參數的函式屬性，如下表所示。

關鍵字	函式呼叫前，由使用者設定參數？	函式呼叫後，從參數取得資料？
[input]	是	否
[output]	否	是

表 4.1 DLL 函式定義

函式定義	章節
WORD CAN_GetDllVersion();	4.1.1
int CAN_TotalBoard();	4.1.2
int CAN_GetBoardInf(BYTE BoardNo, DWORD *dwVID, DWORD *dwDID, DWORD *dwSVID, DWORD *dwSDID, DWORD *dwIrqNo);	4.1.3
int CAN_GetCardPortNum(BYTE BoardNo, BYTE *bGetPortNum);	4.1.4
int CAN_ActiveBoard(BYTE wBoardNo)	4.1.5
int CAN_CloseBoard(BYTE wBoardNo);	4.1.6
int CAN_BoardIsActive(BYTE BoardNo);	4.1.7
int CAN_Reset(BYTE BoardNo, BYTE Port);	4.1.8
int CAN_Init(BYTE wBoardNo, BYTE Port);	4.1.9
int CAN_Config(BYTE BoardNo, BYTE Port, ConfigStruct *CanConfig);	4.1.10
int CAN_ConfigWithoutStructure(BYTE BoardNo, BYTE Port, DWORD AccCode, DWORD AccMask, BYTE BaudRate, BYTE BT0, BYTE BT1);	4.1.11
int CAN_EnableRxIrq(BYTE BoardNo ,BYTE Port);	4.1.12
int CAN_DisableRxIrq(BYTE BoardNo, BYTE Port);	4.1.13
int CAN_RxIrqStatus(BYTE BoardNo, BYTE Port, BYTE *bStatus);	4.1.14
int CAN_InstallIrq(BYTE BoardNo);	4.1.15
int CAN_RemoveIrq(BYTE BoardNo);	4.1.16
int CAN_IrqStatus(BYTE BoardNo, BYTE *bStatus);	4.1.17
int CAN_Status(BYTE BoardNo, BYTE Port, BYTE *bStatus);	4.1.18
int CAN_SendMsg(BYTE BoardNo, BYTE Port, PacketStruct *CanPacket);	4.1.19
int CAN_SendWithoutStruct(BYTE BoardNo, BYTE Port, BYTE Mode, DWORD Id, BYTE Rtr, BYTE Dlen, BYTE *Data)	4.1.20
int CAN_RxMsgCount(BYTE BoardNo, BYTE Port);	4.1.21
int CAN_ReceiveMsg(BYTE BoardNo, BYTE Port, PacketStruct *CanPacket);	4.1.22
int CAN_ReceiveWithoutStruct(BYTE BoardNo, BYTE Port, BYTE *Mode, DWORD *Id, BYTE *Rtr, BYTE *Dlen, BYTE *Data, LONGLONG *MsgTimeStamps);	4.1.23
int CAN_ClearSoftBuffer(BYTE BoardNo, BYTE Port);	4.1.24
int CAN_ClearDataOverrun(BYTE BoardNo, BYTE Port);	4.1.25
void CAN_OutputByte(BYTE BoardNo, BYTE Port, WORD wOffset, BYTE bValue);	4.1.26
BYTE CAN_InputByte(BYTE BoardNo, BYTE Port, WORD wOffset);	4.1.27
LONGLONG CAN_GetSystemFreq(void);	4.1.28
Int CAN_InstallUserIsr(BYTE BoardNo, void(*UserISR)(BYTE BoardNo));	4.1.29
Int CAN_RemoveUserIsr(BYTE BoardNo);	4.1.30
int CAN_BusErrorCode(BYTE BoardNo, BYTE Port, BYTE *bErrorCode);	4.1.31

表 4.2 回傳碼的說明

回傳碼	錯誤名稱	備註
0	CAN_NoError	正常
1	CAN_DriverError	驅動程式錯誤
2	CAN_ActiveBoardError	板卡無法啟動
3	CAN_BoardNumberError	板卡編號超出最大板卡數量(7)
4	CAN_PortNumberError	通訊埠編號超出最大通訊埠數量
5	CAN_ResetError	CAN 晶片硬體重新啟動錯誤
6	CAN_SoftResetError	CAN晶片軟體重新啟動錯誤
7	CAN_InitError	CAN晶片初始化錯誤
8	CAN_ConfigError	CAN 晶片配置錯誤
9	CAN_SetACRError	設定接受碼暫存器錯誤
10	CAN_SetAMRError	設定接受遮罩暫存器錯誤
11	CAN_SetBaudRateError	設定鮑率錯誤
12	CAN_EnableRxIrqFailure	CAN 晶片的接收中斷功能啟用失敗
13	CAN_DisableRxIrqFailure	CAN 晶片的接收中斷功能關閉失敗
14	CAN_InstallIrqFailure	PCI 板卡的 IRQ 安裝失敗
15	CAN_RemoveIrqFailure	PCI 板卡的 IRQ 移除失敗
16	CAN_TransmitBufferLocked	CAN 晶片的傳送緩衝器被鎖住
17	CAN_TransmitIncomplete	先前的傳輸尚未完成
18	CAN_ReceiveBufferEmpty	CAN晶片的RXFIFO是空的
19	CAN_DataOverrun	CAN晶片的RXFIFO沒有足夠的空間，造成資料遺失
20	CAN_ReceiveError	資料接收未完成
21	CAN_SoftBufferIsEmpty	驅動程式的軟體緩衝器是空的
22	CAN_SoftBufferIsFull	驅動程式的軟體緩衝器是滿的
23	CAN_TimeOut	函式無回應與超時
24	CAN_InstallIsrError	使用者 ISR 安裝失敗

4.1.1 CAN_GetDllVersion

- **說明:**

  取得 PISOCAN.dll 驅動程式的版本資訊

- **語法:**

  WORD CAN_GetDllVersion(viod)

- **參數:**

  無

- **回傳:**

  DLL 版本資訊。例如，如果傳回 101(hex)，表示驅動程式版本為 1.01。

4.1.2 CAN_TotalBoard

- **說明:**

  取得所有安裝在 PCI bus 的 CAN 板卡數量。

- **語法:**

  int CAN_TotalBoard(void)

- **參數:**

  無。

- **回傳:**

  回傳板卡數量。

4.1.3 CAN_GetBoardInf

- **說明:**

  取得 PISO-CAN 板卡的資訊，包含供應商名稱，設備名稱及中斷編號。

- **語法:**

  int CAN_GetBoardInf(BYTE BoardNo, DWORD *dwVID, DWORD *dwDID, DWORD *dwSVID,DWORD *dwSDID, DWORD *dwSAuxID, DWORD *dwIrqNo)

- **參數:**

  BoardNo: [input] PISO-CAN 板卡編號。
  *dwVID: [output]板卡的供應商名稱。
  *dwDID: [output]板卡的設備名稱。
  *dwSVID: [output]板卡的子供應商名稱。
  *dwSDID: [output]板卡的子設備名稱。
  *dwSAuxID: [output]板卡的子輔助名稱。
  *dwIrq: [output]板卡的邏輯中斷編號。

- **回傳:**

  CAN_NoError：正常。

  CAN_DriverError： 核心驅動程式無法開啟。

  CAN_BoardNumberError： “BoardNo”超出目前所有板卡編號。

4.1.4 CAN_GetCardPortNum

- **Description:**

  取得 PISO-CAN 卡的 CAN 通訊埠編號。

- **Syntax:**

  int CAN_GetCardPortNum(BYTE BoardNo, BYTE *bGetPortNum)

- **Parameter:**

  BoardNo: [input] PISO-CAN 板卡編號 (0~7)。
  * bGetPortNum: [output] CAN 卡的通訊埠編號。

- **Return:**

  CAN_NoError: 正常。

  CAN_DriverError: 核心驅動程式無法開啟。

  CAN_BoardNumberError: “BoardNo”值超出目前所有板卡編號。

4.1.5 CAN_ActiveBoard

- **說明:**

  啟動設備。**在使用其他 PISO-CAN 板卡的功能之前，必須呼叫此函式一次。**

- **語法:**

  int CAN_ActiveBoard(BYTE BoardNo)

- **參數:**

  BoardNo: [input] PISO-CAN 板卡編號 (0~7)。.

- **回傳:**

  CAN_NoError: 正常

  CAN_BoardNumberError: “BoardNo”值超出目前所有板卡編號。

  CAN_ActiveBoardError: 此板卡無法啟動或核心驅動程式無法開啟。

4.1.6 CAN_CloseBoard

- **說明:**

  停止並關閉核心驅動程式且釋放電腦端的設備資源。在離開使用者應用程式之前，這個函式必須被呼叫一次。

- **語法:**

  int CAN_CloseBoard(BYTE BoardNo)

- **參數:**

  BoardNo: [input] PISO-CAN 板卡編號(0~7)。

- **回傳:**

  CAN_NoError: 正常。

  CAN_ActiveBoardError:板卡未啟動。

  CAN_BoardNumberError: “BoardNo”值超出目前所有板卡編號。

4.1.7 CAN_BoardIsActive

- **說明:**

  檢查該版卡是否已經啟動。

- **語法:**

  int CAN_BoardIsActive(BYTE BoardNo)

- **參數:**

  BoardNo: [input] PISO-CAN 板卡編號 (0~7)。

- **回傳:**

  0: 表示板卡未啟動。

  1: 表示板卡已啟動。

4.1.8 CAN_Reset

- **說明:**

  重新啟動 CAN 控制器。

- **語法:**

  int CAN_Reset(BYTE BoardNo, BYTE Port)

- **參數:**

  BoardNo: [input] PISO-CAN 板卡編號 (0~7)。

  Port: [input] CAN 通訊埠編號 (1~8)。

- **回傳:**

  CAN_NoError: 正常。

  CAN_DriverError: 核心驅動程式無法開啟。

  CAN_BoardNumberError: “BoardNo”不正確，或超出目前所有板卡編號。

  CAN_PortNumberError: “Port”編號不正確。

  CAN_ActiveBoardError: 此板卡未被啟動。

4.1.9 CAN_Init

- **說明:**

  初始化 CAN 控制器。

- **語法:**

  int CAN_Init(BYTE BoardNo, BYTE Port)

- **參數:**

  BoardNo: [input] PISO-CAN 板卡編號 (0~7)。

  Port: [input] CAN 通訊埠編號 (1~8)。

- **回傳:**

  CAN_NoError: 正常。

  CAN_DriverError: 核心驅動程式無法開啟。

  CAN_BoardNumberError: “ BoardNo”不正確，或超出目前所有板卡的編號。

  CAN_PortNumberError: “Port”編號不正確。

  CAN_ActiveBoardError: 此板卡未被啟動。

  CAN_InitError: CAN 控制器初始化失敗。

4.1.10 CAN_Config

- **說明:**

  CAN 控制器組態。在呼叫此函式後，CAN 控制器將進入操作模式。

- **語法:**

  int CAN_Config(BYTE BoardNo, BYTE Port,ConfigStruct *CanConfig);

- **參數:**

  BoardNo: [input] PISO-CAN 板卡編號(0~7).

  Port: [input] CAN 通訊埠編號(1~8)

  *ConfigStruct: [input]“ConfigStruct”指標的結構定義如下：

```
typedef struct config
{
    BYTE AccCode[4];
    BYTE AccMask[4];
    BYTE BaudRate;
    BYTE BT0, BT1;
} ConfigStruct;
```

  AccCode[4]: CAN 控制器接受碼。

  AccMask[4]: CAN 控制器接受遮罩。

  BaudRate: 0→使用者定義(必須設定 BT0、BT1)； 1→10Kbps；
  2→20Kbps； 3→50Kbps；4→125Kbps； 5→250Kbps；
  6→500Kbps；7→800Kbps；8→1Mbps。

  BT0, BT1: 使用者定義鮑率 (若“BaudRate=0”時使用)。例如，
  BT0=0x04、 BT1=0x1C、CAN 控制器的鮑率設定便是

100Kbps。更詳細的鮑率設定，請參考 SJA1000 CAN 控制器的手冊。

- **回傳:**

  CAN_NoError: 正常。

  CAN_DriverError: 核心驅動程式無法開啟。

  CAN_BoardNumberError: “BoardNo”不正確，或超出目前所有板卡編號。

  CAN_PortNumberError: “Port”編號不正確。

  CAN_ActiveBoardError: 此板卡未被啟動。

  CAN_SoftResetError: CAN 控制器軟體重新啟動錯誤。

  CAN_SetACRError: 設定 CAN 控制器接受碼發生錯誤。

  CAN_SetAMRError: 設定 CAN 控制器接受遮罩發生錯誤。

  CAN_SetBaudRateError: 設定 CAN 控制器鮑率發生錯誤。

  CAN_ConfigError: CAN 控制器進入操作模式失敗。

4.1.11 CAN_ConfigWithoutStructure

- **說明:**

  此函式與“**CAN_Config**”相同，但不使用“ConfigStruct”結構型態。提供這個功能是因為部份程式開發平台在編譯 PISOCAN.lib 時，結構位址會有不一樣的分配方式。因此，如果使用者使用“CAN_Config”卻無法正確配置 CAN 卡的時候，就可以使用“CAN_ConfigWithoutStruct”函式來代替。

- **語法:**

  int CAN_ConfigWithoutStructure(BYTE BoardNo, BYTE Port,
  DWORD AccCode, DWORD AccMask, BYTE BaudRate,
  BYTE BT0, BYTE BT1);

- **參數:**

  BoardNo: [input] PISO-CAN 板卡編號(0~7)。

  Port: [input] CAN 通訊埠編號(1~8)。

  AccCode: CAN 控制器接收碼，低位元對應至 ACR[3]暫存器，高位元對應至 ACR[0]暫存器。

  AccMask: CAN 控制器接收遮罩，低位元對應至 AMR[3]暫存器，高位元對應至 AMR[0]暫存器。

  BaudRate: 0→使用者定義(必須設定 BT0、BT1)； 1→10Kbps；
  2→20Kbps； 3→50Kbps；4→125Kbps；5→250Kbps；
  6→500Kbps；7→800Kbps； 8→1Mbps。

  BT0, BT1: 使用者定義鮑率(若“BaudRate=0”時使用)。例如，
  BT0=0x04、 BT1=0x1C，CAN 控制器的鮑率設定便是 100Kbps。更詳細的鮑率設定，請參考 SJA1000 CAN 控制器的手冊。

- **回傳:**

  CAN_NoError: 正常。

  CAN_DriverError: 核心驅動程式無法開啟。

  CAN_BoardNumberError: “BoardNo”不正確，或超出目前所有板卡編號。

  CAN_PortNumberError: “Port”編號不正確。

  CAN_ActiveBoardError: 此板卡未被啟動。

  CAN_SoftResetError: CAN 控制器軟體重新啟動錯誤。

  CAN_SetACRError: 設定CAN控制器接受碼發生錯誤。

  CAN_SetAMRError: 設定CAN控制器接受遮罩發生錯誤。

  CAN_SetBaudRateError: 設定CAN控制器鮑率發生錯誤。

  CAN_ConfigError: CAN控制器進入操作模式失敗。

4.1.12 CAN_EnableRxIrq

- **說明:**

  啟用 CAN 控制器的接收中斷功能。

- **語法:**

  int CAN_EnableRxIrq(BYTE BoardNo, BYTE Port)

- **參數:**

  BoardNo: [input] PISO-CAN 板卡編號(0~7)。

  Port: [input] CAN 通訊埠編號(1~8)。

- **回傳:**

  CAN_NoError: 正常。

  CAN_DriverError: 核心驅動程式無法開啟。

  CAN_BoardNumberError: “BoardNo”不正確，或超出目前所有板卡編號。

  CAN_PortNumberError: “Port”編號不正確。

  CAN_ActiveBoardError: 此板卡未被啟動。

  CAN_EnableRxIrqFailure: 啟用接收中斷功能失敗。

4.1.13 CAN_DisableRxIrq

- **說明:**

  停用 CAN 控制器的接收中斷功能。

- **語法:**

  Int CAN_DisableRxIrq(BYTE BoardNo, BYTE Port)

- **參數:**

  BoardNo: [input] PISO-CAN 板卡編號(0~7)。

  Port: [input] CAN 通訊埠編號(1~8)。

- **回傳:**

  CAN_NoError: 正常。

  CAN_DriverError: 核心驅動程式無法開啟。

  CAN_BoardNumberError: “BoardNo”不正確，或超出目前所有板卡編號。

  CAN_PortNumberError: “Port”編號不正確。

  CAN_ActiveBoardError: 此板卡未被啟動。

  CAN_DisableRxIrqFailure:停用接收中斷功能失敗。

4.1.14 CAN_RxIrqStatus

- **說明:**

  取得 CAN 控制器接收中斷功能的狀態。

- **語法:**

  int CAN_RxIrqStatus(BYTE BoardNo, BYTE Port, BYTE *bStatus)

- **參數:**

  BoardNo: [input] PISO-CAN 板卡編號(0~7)。

  Port: [input] CAN 通訊埠編號(1~8)。

  *bStatus:[output] 0→停用接收中斷；

  1→啟用接收中斷。

- **回傳:**

  CAN_NoError: 正常。

  CAN_DriverError: 核心驅動程式無法開啟。

  CAN_BoardNumberError: “BoardNo”不正確, 或超出目前所有板卡編號。

  CAN_PortNumberError: “Port”編號不正確。

  CAN_ActiveBoardError: 此板卡未被啟動。

4.1.15 CAN_InstallIrq

- **說明:**

  對 PISO-CAN board 板卡啟用或啟動 IRQ 。在呼叫此函式之前，必須先呼叫 “**CAN_EnableRxIrq**” 函式。

- **語法:**

  int CAN_InstallIrq(BYTE BoardNo)

- **參數:**

  BoardNo: [input] PISO-CAN 板卡編號 (0~7).

- **回傳:**

  CAN_NoError: 正常。

  CAN_DriverError: 核心驅動程式無法開啟。

  CAN_BoardNumberError: “BoardNo” 不正確，或超出目前所有板卡編號。

  CAN_ActiveBoardError: 此板卡未被啟動。

  CAN_InstallIrqFailure: 啟用或啟動 IRQ 失敗。

4.1.16 CAN_RemoveIrq

- **說明:**

  停用或停止 PISO-CAN 板卡的 IRQ。在呼叫此函式後，板卡上的所有 CAN 控制器的 IRQ 功能將被停用。

- **回傳:**

  int CAN_RemoveIrq(BYTE BoardNo)

- **參數:**

  BoardNo: [input] PISO-CAN 板卡編號(0~7)。

- **回傳:**

  CAN_NoError: 正常。

  CAN_DriverError: 核心驅動程式無法開啟。

  CAN_BoardNumberError: “BoardNo”不正確，或超出目前所有板卡編號。

  CAN_ActiveBoardError: 此板卡未被啟動。

  CAN_RemoveIrqFailure: 停用或停止 IRQ 失敗。

4.1.17 CAN_IrqStatus

- **說明:**

  取得 PISO-CAN 板卡的 IRQ 狀態。

- **語法:**

  int CAN_IrqStatus(BYTE BoardNo, BYTE *bStatus)

- **參數:**

  BoardNo: [input] PISO-CAN 板卡編號 (0~7)。

  *bStatus:[output] 0→停用 IRQ。

  1→啟用 IRQ。

- **回傳:**

  CAN_NoError: 正常。

  CAN_DriverError: 核心驅動程式無法開啟。

  CAN_BoardNumberError: “BoardNo”不正確，或超出目前所有板卡編號。

  CAN_ActiveBoardError: 此板卡未被啟動。

4.1.18 CAN_Status

- **說明:**

  取得 CAN 控制器的狀態。

- **語法:**

  int CAN_Status(BYTE BoardNo, BYTE Port,BYTE *bStatus)

- **參數:**

  BoardNo: [input] PISO-CAN 板卡編號(0~7)。

  *bStatus:[output] CAN 控制器的狀態值。

表 4.3 “bStatus” 的位元說明。

位元	名稱	值	狀態
bit 7	匯流排狀態	1	匯流排關閉
		0	匯流排開啟
bit 6	錯誤狀態	1	錯誤
		0	正常
bit 5	傳輸狀態	1	傳輸
		0	閒置
bit 4	接收狀態	1	接收
		0	閒置
bit 3	傳輸完成狀態	1	完成
		0	未完成
bit 2	傳輸緩衝器狀態	1	釋放
		0	鎖住
bit 1	資料溢出狀態	1	溢出
		0	不存在
bit 0	接收緩衝器狀態	1	滿的/未滿的
		0	空的

- **回傳:**

  CAN_NoError: 正常。

  CAN_BoardNumberError: “BoardNo” 不正確，或超出目前所有板卡編號。

  CAN_PortNumberError: “Port” 編號不正確。

  CAN_ActiveBoardError: 此板卡未被啟動。

4.1.19 CAN_SendMsg

- **說明:**

  立即發送 CAN 訊息。

- **語法:**

  int CAN_SendMsg(BYTE BoardNo, BYTE Port, PacketStruct *CanPacket)

- **參數:**

  BoardNo: [input] PISO-CAN 板卡編號(0~7)。

  Port: [input] CAN 通訊埠編號(1~8)。

  *CanPacket: [input] “CanPacket” 結構定義如下：

```
typedef struct packet
{
    LONGLONG MsgTimeStamps;
    BYTE    mode;
    DWORD id;
    BYTE    rtr;
    BYTE    len;
    BYTE    data[8];
} PacketStruct;
```

  MsgTimeStamps:此參數在本函式中沒有作用。

  mode: 0→ 11-bit 的識別碼； 1 → 29-bit 的識別碼。

  id: 識別碼。

  rtr: 遠端傳輸請求。

  len: 資料長度。

data[8]: 資料位元組。

- **回傳:**

  CAN_NoError: 正常。

  CAN_BoardNumberError: “BoardNo”不正確，或超出目前所有板卡編號。

  CAN_PortNumberError: “Port”編號不正確。

  CAN_ActiveBoardError: 此板卡未被啟動。

  CAN_TransmitBufferLocked: CAN 晶片的傳輸緩衝器被鎖住。

  CAN_TransmitIncomplete: 尚未完成傳輸。

  CAN_ConfigError: 通訊埠尚未配置成功。

4.1.20 CAN_SendWithoutStruct

- **說明:**

  此函式與“**CAN_SendMsg**”函式相同，但不使用“PacketStruct”結構型態。如果使用者在部份應用程式開發環境，如.NET 2003，使用“CAN_SendMsg”，卻無法正確發送 CAN 訊息，此時便可用“CAN_SendWithoutStruct”函式代替。

- **語法:**

  int CAN_SendWithoutStruct(BYTE BoardNo, BYTE Port, BYTE Mode, DWORD Id, BYTE Rtr, BYTE Dlen, BYTE *Data)

- **參數:**

  BoardNo: [input] PISO-CAN 板卡編號(0~7)。

  Port: [input] CAN 通訊埠編號(1~8)。

  Mode: 0→ 11-bit 的識別碼；1 → 29-bit 的識別碼。

  Id: 識別碼。

  Rtr: 遠端傳輸請求。

  Dlen: 資料長度。

  *Data: 資料位元組。

- **回傳:**

  CAN_NoError: 正常。

  CAN_BoardNumberError: “BoardNo”不正確。

  CAN_PortNumberError: “Port”編號不正確。

  CAN_ActiveBoardError: 此板卡未被啟動。

  CAN_TransmitBufferLocked: CAN 晶片的傳輸緩衝器被鎖住。

  CAN_TransmitIncomplete: 傳輸尚未完成。

  CAN_ConfigError: 通訊埠尚未配置成功。

4.1.21 CAN_RxMsgCount

- **說明:**

  取得 CAN 控制器的 RXFIFO 或軟體緩衝器(4KBytes)裡面，可用的 CAN 訊息數量。在呼叫 **“CAN_EnableRxIrq”** 和 **“CAN_InstallIrq”** 函式後，所得到的 CAN 訊息數量是軟體緩衝區內部的，否則就是 CAN 控制器 RXFIFO 內部的。

- **語法:**

  int CAN_RxMsgCount(BYTE BoardNo, BYTE Port);

- **參數:**

  BoardNo: [input] PISO-CAN 板卡編號 (0~7)。

  Port: [input] CAN 通訊埠編號(1~8)。

- **回傳:**

  CAN 訊息的數量。

  注意. 如果 “*BoardNo*” 或 “Port” 參數不正確，回傳值將一直是 “0” 。

4.1.22 CAN_ReceiveMsg

- **說明:**

  從 CAN 控制器的 RXFIFO 或軟體緩衝器中，取得接收訊息。在呼叫 **“CAN_EnableRxIrq”**和 **“CAN_InstallIrq”**後，訊息不是在軟體緩衝器內，就是在 CAN 控制器的 RXFIFO 內。

  注意! 如果使用者的電腦進入“待機模式”或“睡眠模式”，此函式將不能接收任何訊息。

- **語法:**

  int CAN_ReceiveMsg(BYTE BoardNo, BYTE Port, PacketStruct *CanPacket)

- **參數:**

  BoardNo: [input] PISO-CAN 板卡編號(0~7)。

  Port: [input] CAN 通訊埠編號(1~8)。

  *CanPacket: [output]“CanPacket”結構定義如下：

```
typedef struct packet
{
    LONGLONG MsgTimeStamps;
    BYTE   mode;
    DWORD id;
    BYTE   rtr;
    BYTE   len;
    BYTE   data[8];
} PacketStruct;
```

  MsgTimeStamps: 此參數是紀錄當 CAN 訊息從 SJA1000 接收起來時的

時間標記，在 Windows 98/Me/NT4 環境中，是用系統時脈計數器來記錄此時間，而在 Windows 2000 /XP 環境中，則以 100-ns 為單位的系統時間中斷計數器記錄時間。其中系統時脈計數器是在電腦開機後，系統就會開始計數。如果在 64 位元的 SJA1000 FIFO 裡有接收並儲存 1 個以上的 CAN 訊息時，這幾筆 CAN 訊息的時間標記可能會很接近。

mode: 0→ 11-bit 的識別碼；1 → 29-bit 的識別碼。

id: 識別碼。

rtr: 遠端傳輸請求。

len: 資料長度。

data[8]: 資料位元組。

- **回傳:**

CAN_NoError: 正常。

CAN_BoardNumberError: “BoardNo”不正確, 或超出目前所有板卡編號。

CAN_PortNumberError: “Port”編號不正確。

CAN_ActiveBoardError: 此板卡未被啟動。

CAN_ConfigError: 通訊埠尚未配置成功。

CAN_ReceiveBufferEmpty: CAN 控制器的 RXFIFO 是空的。

CAN_SoftBufferIsEmpty: 軟體的 RX 緩衝器是空的。

CAN_SoftBufferIsFull: 軟體的 RX 緩衝器是滿的。

4.1.23 CAN_ReceiveWithoutStruct

- **說明:**

  此函式與“**CAN_ReceiveMsg**”函式相同,但不使用“PacketStruct”結構型態。提供這個功能是因為部份程式開發平台在編譯 PISOCAN.lib 時,結構位址會有不一樣的分配方式,如 .NET 2003。因此,如果使用“CAN_ReceiveMsg”函式,卻無法正確的接收 CAN 訊息,那麼可以使用“CAN_ReceiveWithoutStruct”函式來代替。

- **語法:**

  int CAN_ReceiveWithoutStruct(BYTE BoardNo, BYTE Port, BYTE *Mode, DWORD *Id, BYTE *Rtr, BYTE *Dlen, BYTE *Data, DWORD *H_MsgTimeStamps, DWORD *L_MsgTimeStamps)

- **參數:**

  BoardNo: [input] PISO-CAN 板卡編號 (0~7)。

  Port: [input] CAN 通訊埠編號(1~8)。

  *Mode: 0→ 11-bit 的識別碼;1 → 29-bit 的識別碼。

  *Id: 識別碼。

  *Rtr: 遠端傳輸請求。

  *Dlen: 資料長度。

  *Data: 資料位元組

  *H_MsgTimeStamps, *L_MsgTimeStamps: 這些參數是紀錄當 CAN 訊息從 SJA1000 接收起來時的時間標記,在 Windows 98/Me/NT4 環境中,是用系統時脈計數器來記錄此時間,而在 Windows 2000/XP 環境中,則以 100-ns 為單位的系統時間中斷計數器記錄時間。參數“*H_MsgTimeStamps”是時間標記的高位元 DWORD 而參數 *L_MsgTimeStamps 是時間標記的低位元 DWORD。其

中系統時脈計數器是在電腦開機後，系統就會開始計數。如果在 64 位元的 SJA1000 FIFO 裡有接收並儲存 1 個以上的 CAN 訊息時，這幾筆 CAN 訊息的時間標記可能會很接近。

- **回傳:**

  CAN_NoError: 正常。

  CAN_BoardNumberError: “ BoardNo” 不正確, 或超出目前所有板卡編號。

  CAN_PortNumberError: “Port” 編號不正確。

  CAN_ActiveBoardError: 此板卡未被啟動。

  CAN_ConfigError: 通訊埠尚未成功啟動。

  CAN_ReceiveBufferEmpty: CAN 控制器的 RXFIFO 是空的。

  CAN_SoftBufferIsEmpty: 軟體的 RX 緩衝器是空的。

  CAN_SoftBufferIsFull: 軟體的 RX 緩衝器是空的。

4.1.24 CAN_ClearSoftBuffer

- **說明:**

  清除 PISOCAN.DLL 驅動程式的軟體緩衝器。

- **語法:**

  int CAN_ClearSoftBuffer(BYTE BoardNo, BYTE Port)

- **參數:**

  BoardNo: [input] PISO-CAN 板卡編號(0~7) 。

  Port: [input] CAN 通訊埠編號(1~8)。

- **回傳:**

  CAN_NoError: 正常。

  CAN_BoardNumberError: BoardNo”不正確,或超出目前所有板卡編號。

  CAN_PortNumberError: 通訊埠不正確。

4.1.25 CAN_ClearDataOverrun

- **說明:**

  清除 CAN 控制器的資料溢位狀態位元。

- **語法:**

  int CAN_ClearDataOverrun(BYTE BoardNo, BYTE Port)

- **參數**

  BoardNo: [input] PISO-CAN 板卡編號 (0~7) 。

  Port: [input] CAN 通訊埠編號 (1~8) 。

- **回傳:**

  CAN_NoError: 正常。

  CAN_BoardNumberError: “BoardNo” 不正確，或超出目前所有板卡編號。

  CAN_PortNumberError: 通訊埠不正確。

  CAN_ActiveBoardError: 此板卡未被啟動。

  CAN_ConfigError: CAN 控制器進入操作模式失敗。

4.1.26 CAN_OutputByte

- **說明:**

  將資料寫入 CAN 晶片 (SJA1000) 。

- **語法:**

  void CAN_OutputByte(BYTE BoardNo, BYTE Port, WORD wOffset, BYTE bValue)

- **參數:**

  BoardNo: [input] PISO-CAN 板卡編號 (0~7)。

  Port: [input] CAN 通訊埠編號(1~8)。

  wOffset: [input] 位址偏移量。

  bValue: [input] 資料位元組。

- **回傳:**

  無.

4.1.27 CAN_InputByte

- **說明:**

  從 CAN 晶片讀取資料(SJA1000)。

- **語法:**

  BYTE CAN_InputByte(BYTE BoardNo, BYTE Port, WORD wOffset)

- **參數:**

  BoardNo: [input] PISO-CAN 板卡編號 (0~7)。

  Port: [input] CAN 通訊埠編號(1~8) 。

  wOffset: [input] 位址偏移量。

- **回傳:**

  CAN 晶片的資料位元組。

4.1.28 CAN_GetSystemFreq

- **說明:**

  取得時脈頻率。此函式對於計算接收訊息的時間標記來說很有用。

- **語法:**

  LONGLONG CAN_GetSystemFreq(void)

- **參數:**

  無。

- **回傳:**

  在 Windows 98/Me/NT4 回傳的是時脈頻率，而在 Windows 2000/XP 則回傳 10000000。

4.1.29 CAN_InstallUserIsr (適用於 Windows 2000/XP)

- **說明:**

  這個函式允許使用者請求中斷服務程序(ISR)，當使用者將自定義的 ISR 放到此函式中後，接收 CAN 訊息的中斷將觸發此 ISR。

- **語法:**

  int CAN_InstallUserIsr(BYTE BoardNo,
  void(*UserISR)(BYTE BoardNo))

- **參數:**

  BoardNo: [input] PISO-CAN 板卡編號(0~7)

  (*UserISR)(BYTE BoardNo): [input] 此指標指向一個函式，格式為“void XXX(BYTE BoardNo)”。其中“XXX”是使用者的 ISR 的函式名稱，而參數“BoardNo”表示產生中斷訊號的板卡編號。

- **回傳:**

  CAN_NoError: 正常。

  CAN_DriverError: 核心驅動程式無法打開。

  CAN_BoardNumberError: “BoardNo”值不正確，或超出目前所有板卡的編號。

  CAN_ActiveBoardError: 此板卡未被啟動。

  CAN_InstallIrqFailure: 啟用或啟動 IRQ 失敗。

  CAN_InstallIsrError: 啟用或啟動 ISR 失敗。

4.1.30 CAN_RemoveUserIsr (only for Windows 2000/XP)

- **說明:**

  當使用者不需要 ISR 功能時，可呼叫此函式移除 ISR。

- **語法:**

  Int CAN_RemoveUserIsr(BYTE BoadNo) 。

- **參數:**

  BoardNo: [input] PISO-CAN 板卡編號(0~7) 。

- **回傳:**

  CAN_NoError: 正常。

  CAN_DriverError: 核心驅動程式無法打開。

  CAN_BoardNumberError: “BoardNo”值不正確，或超出目前所有板卡編號。

  CAN_ActiveBoardError: 此板卡未被啟動。

  CAN_RemoveIrqFailure: 禁用或停止 IRQ 失敗。

4.1.31 CAN_BusErrorCode

- **說明:**

  取得 CAN 控制器 Error Code Capture(ECC)暫存器的數值。

- **語法:**

  int CAN_BusErrorCode(BYTE BoardNo, BYTE Port,BYTE *bErrorCode)

- **參數:**

  BoardNo: [input] PISO-CAN 板卡編號(0~7)。

  Port: [input] CAN 通訊埠編號(1~8)。

  *bErrorCode:[output] CAN 控制器的 Error code capture 暫存器數值。

表 4.4 Error code capture 暫存器的位元說明。

Bit	SYMBOL	NAME	VALUE	FUNCTION
ECC.7(1)	ERRC1	Error Code 1	-	-
ECC.6(1)	ERRC0	Error Code 0	-	-
ECC.5(2)	DIR	Direction	1	RX; error occurred during reception
			0	TX; error occurred during transmission
ECC.4(2)	SEG4	Segment 4	-	-
ECC.3(2)	SEG3	Segment 3	-	-
ECC.2(2)	SEG2	Segment 2	-	-
ECC.1(2)	SEG1	Segment 1	-	-
ECC.0(2)	SEG0	Segment 0	-	-

備註:

1. ECC.7 及 ECC.6 暫存器的位元說明請參考表 4.5
2. ECC.4 ~ ECC.0 暫存器的位元說明請參考表 4.6

Table 4.5 ECC.7 及 ECC.6 暫存器的位元說明

BIT ECC.7	BIT ECC.6	功能
0	0	Bit error
0	1	Form error
1	0	Stuff error
1	1	Other type of error

Table 4.6　ECC.4 ~ ECC.0 暫存器的位元說明

BIT ECC.4	BIT ECC.3	BIT ECC.2	BIT ECC.1	BIT ECC.0	功能
0	0	0	1	1	Start of frame
0	0	0	1	0	ID.28 to ID.21
0	0	1	1	0	ID.20 to ID.18
0	0	1	0	0	Bit SRTR
0	0	1	0	1	Bit IDE
0	0	1	1	1	ID.17 to ID.13
0	1	1	1	1	ID.12 to ID.5
0	1	1	1	0	ID.4 to ID.0
0	1	1	0	0	Bit RTR
0	1	1	0	1	Reserved bit 1
0	1	0	0	1	Reserved bit 0
0	1	0	1	1	Data length code
0	1	0	1	0	Data field
0	1	0	0	0	CRC sequence
1	1	0	0	0	CRC delimiter
1	1	0	0	1	Acknlwledge slot
1	1	0	1	1	Acknowledge delimiter
1	1	0	1	0	End of frame
1	0	0	1	0	Intermission
1	0	0	0	1	Active error flag
1	0	1	1	0	Passive error flag
1	0	0	1	1	Tolerate dominant bits
1	0	1	1	1	Error delimiter
1	1	1	0	0	Overload flag

- **回傳:**

  CAN_NoError: 正常。

  CAN_BoardNumberError: “BoardNo” 不正確，或超出目前所有板卡編號。

  CAN_PortNumberError: “Port” 編號不正確。

4.2 應用流程圖

在這一節，我們將顯示 PISO-CAN/PEX-CAN /PCM-CAN 板卡在發送與接收 CAN 訊息的操作程序。 圖 4.1 介紹 “CAN 訊息發送”程序。 圖 4.2 與 4.3 分別代表中斷模式及輪循模式的 “CAN 訊息接收” 程序。為了能正確且容易的經由 CAN 網路，發送及接收 CAN 訊息，須遵循 PISO-CAN/PEX-CAN/PCM-CAN 板卡的操作規則。更詳細的資訊，請參考第 5 章的範例程式。

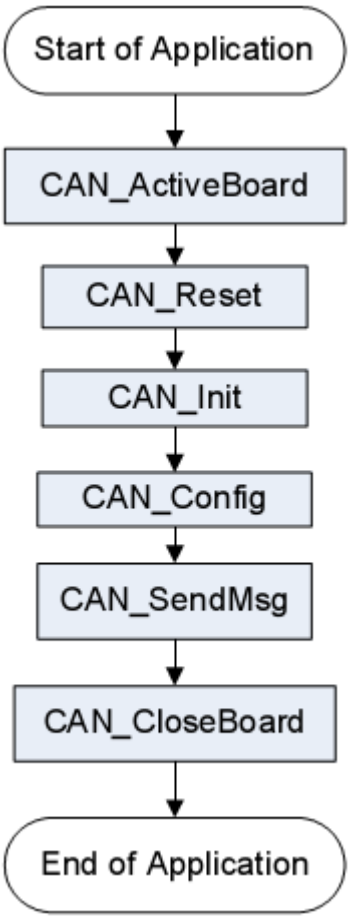


圖 4.1 “CAN 訊息發送” 流程圖

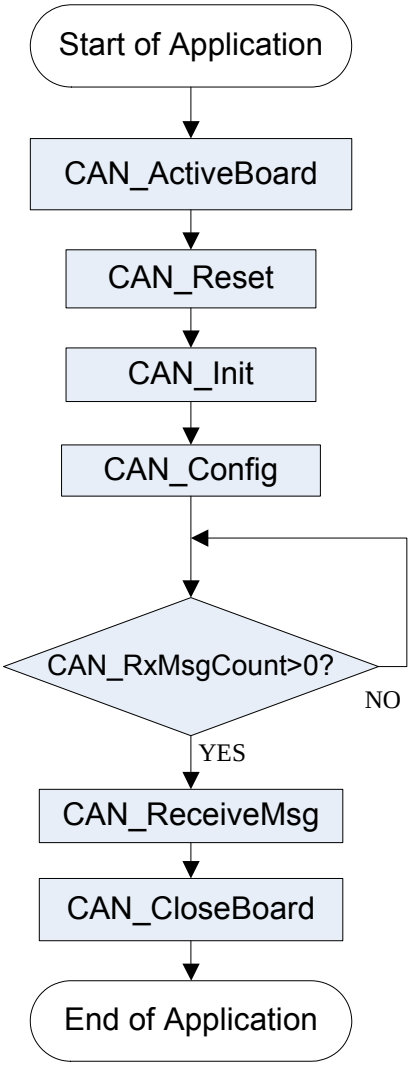


圖 4.2 “CAN 訊息接收”流程圖

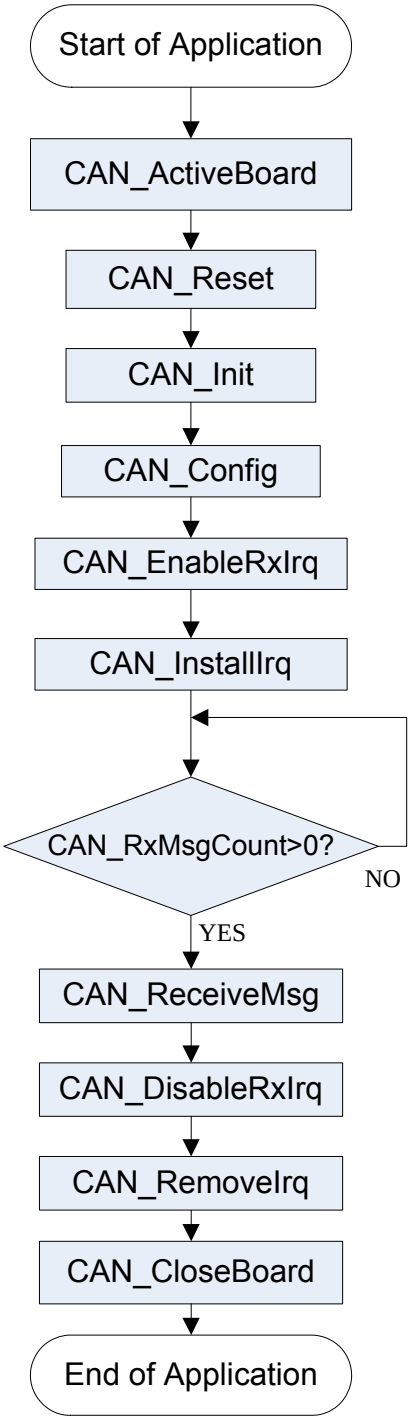


圖 4.3 “用 IRQ 接收 CAN 訊息” 流程圖

5 範例程式(適用於 Windows)

如果 DLL 驅動程式未正確地安裝，範例程式將不能正常的運作。在安裝 DLL 驅動程式期間，安裝精靈將對作業系統註冊正確的核心驅動程式，並且複製 DLL 驅動程式及範例程式到作業系統(windows 2000、XP、7)的相對應位置。在安裝完成後，針對不同的開發環境，其相關的範例程式、開發元件庫(library)及標頭檔(Header file)介紹如下：

```
|--\Demo                          →示範程式
   |--\BCB3                       →適用於 Borland C++ Builder 3
   |      |--\CAN.H               →標頭檔
   |      |--\PISOCAN.LIB         →BCB 的連結庫
   |
   |
   |--\Delphi4                    →適用於 Delphi 4
   |      |--\CAN.PAS             →宣告檔
   |
   |--\VC6                        →適用於 Visual C++ 6
   |      |--\CAN.H               →標頭檔
   |      |--\PISOCAN.LIB         →VC6 的連結庫
   |
   |--\VB6                        →適用於 Visual Basic 6
   |      |--\CAN.BAS             →宣告檔
   |
   |--\C#,Net                     →適用於 C#.Net
   |      |--\ PISOCAN_Net.DLL    →動態連結函式庫
   |
   |--\VB,Net                     →適用於 VB.Net
   |      |--\ PISOCAN_Net.DLL    →動態連結函式庫
```

範例程式清單：

TxRxCAN_NoIRQ:　傳送與接收 CAN 訊號。
TxRxCAN_IRQ:　用 IRQ 傳送與接收 CAN 訊號。

範例程式簡介

TxRxCAN_NoIRQ:

Demo1 的例子用於啟動 PISO-CAN/PEX-CAN /PCM-CAN 板卡。此示範程式是設計由同一塊 PISO-CAN/PEX-CAN/PCM-CAN 板卡的 Port 1 發送出 CAN 訊息，並由 Port 2 立即接收 CAN 訊息。在執行此程式前，使用者需先將 CAN 的 Port1 與 Port2 之間的接線連接。在此範例中，使用者可以輸入 CAN 訊息到 Port1 的訊息框，並且點擊「Send」發送至 Port2。如果在 Port2 的訊息框點擊「Receive」，由 Port2 接收的訊息將會顯示在“文字”方塊中。 螢幕截圖顯示如下。請注意，如果 Port2 顯示一個警告訊息如 CAN 資料溢位，這表示在 64 位元組的 RXFIFO CAN 緩衝器被其他訊息覆蓋而無法讀取。這意味著訊息從 CAN bus 接收時，錯誤的邏輯運算(and/or)，導致部份訊息遺失。接著使用者可點擊「Clear Overrun」清除控制器內的 RXFIFO 緩衝器溢位狀態。

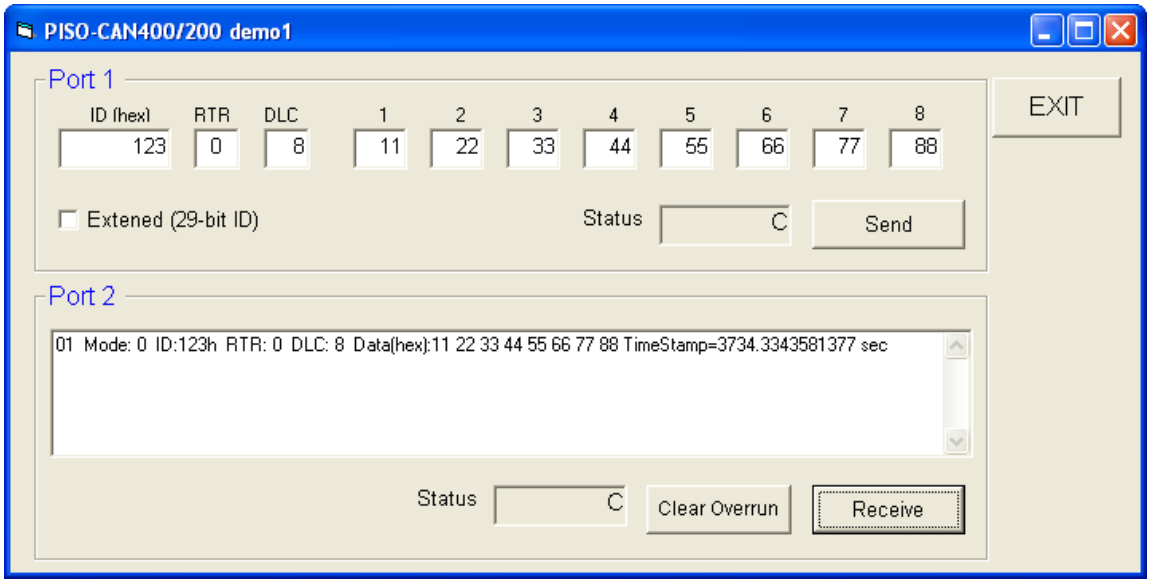


圖 5.1: Demo1 的樣式

TxRxCAN_IRQ:

在 demo 2，我們提供一個實例示範如何經由 “port 1” 發送出 CAN 訊號，且以中斷模式從 “port 2” 接收 CAN 訊號。在此範例中，使用者可以輸入 CAN 訊息到 port1 的訊息框，並且點擊「Send」發送出 CAN 訊息。 同時，port 2 以中斷模式接收 CAN 訊息。如下圖所示，port 2 可以自動地接收 CAN 訊息，並且儲存在軟體緩衝器的 4K byte 裡。當使用者點擊「Receive」，所有儲存在 4K bytes 緩衝器裡的訊息，將全部顯示在文字編輯區域，如下圖所示。

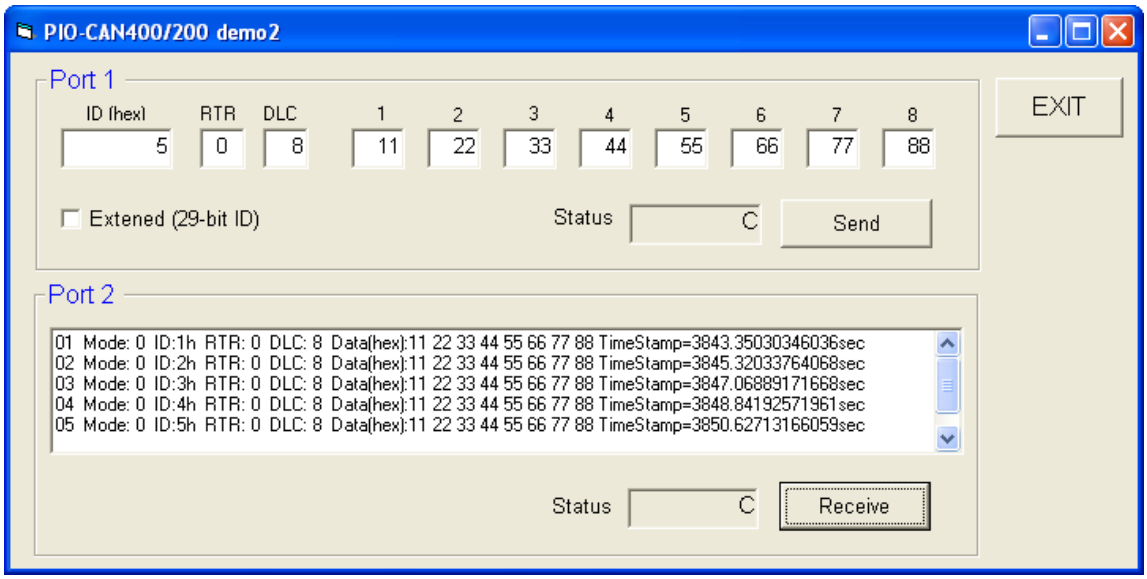


圖 5.2: Demo2 的樣式

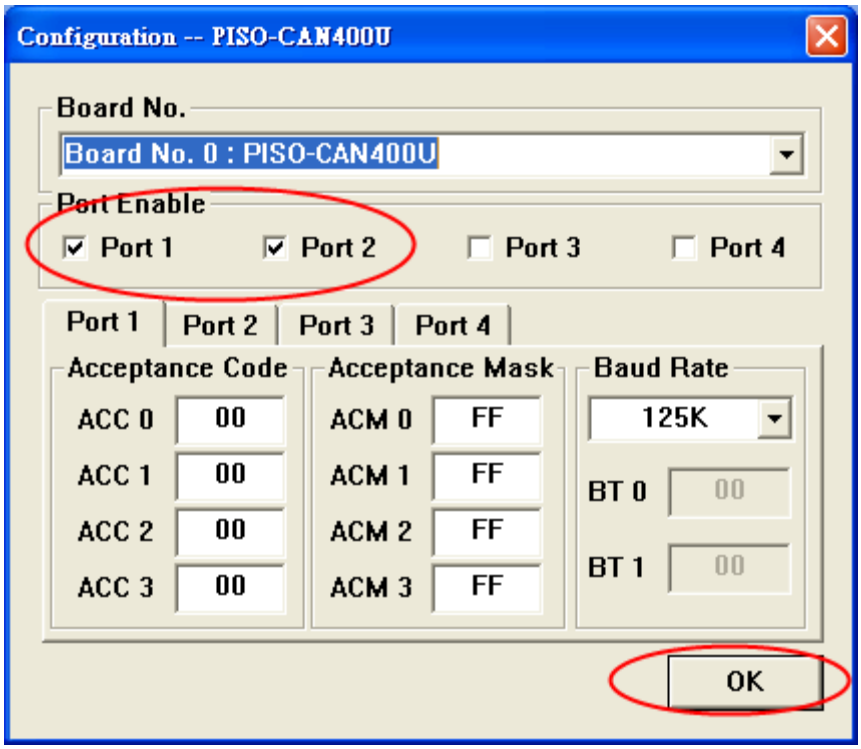
6 CANUtility 工具軟體(適用於 Windows)

針對 PISO-CAN、PEX-CAN 或 PCM-CAN，我們提供一個友善的 CAN bus 工具軟體，讓使用可以在 CAN 網路上輕鬆的發送及接收 CAN 訊息。此工具軟體可以在 CAN 網路上監控訊息或測試設備，而且支援數種功能，如發送 CAN 訊息、接收 CAN 訊息、儲存 CAN 訊息、循環傳輸、等等。其操作方式將在接下來的小節一一介紹。

(1) CAN 配置對話框

請點擊「Board No.」選單，選擇已裝在電腦上，並想要使用的版卡型號。

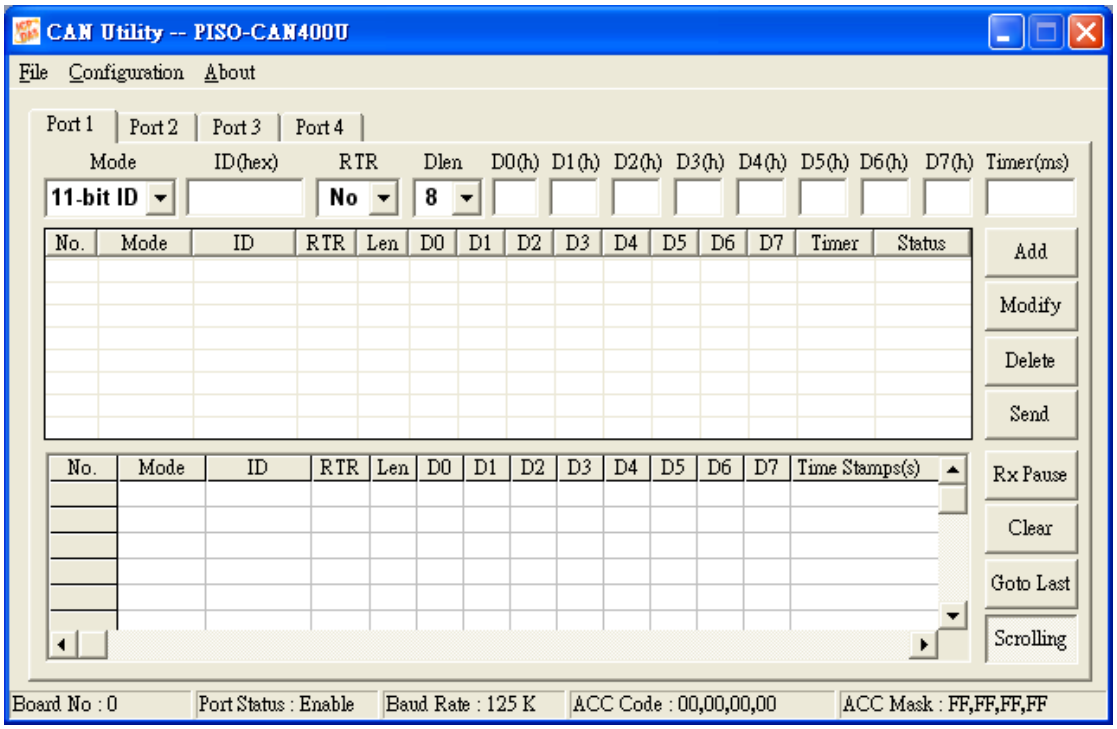
勾選通訊埠啟用複選框來啟動 CAN 通訊埠，接著點選 CAN 通訊埠標籤。根據每個 CAN 通訊的要求，使用者需設定適當的鮑率、接受碼及接受遮罩。鮑率的選項有 8 種，分別為 10K、20K、50K、125K、250K、500K、800K 及 1M。使用者也可以選擇使用者定義模式，藉由 BT0 及 BT1 來設定特殊的鮑率，但前提是，使用者需要了解 SJA1000。接著，點擊「OK」儲存設定。



(2) 主對話框

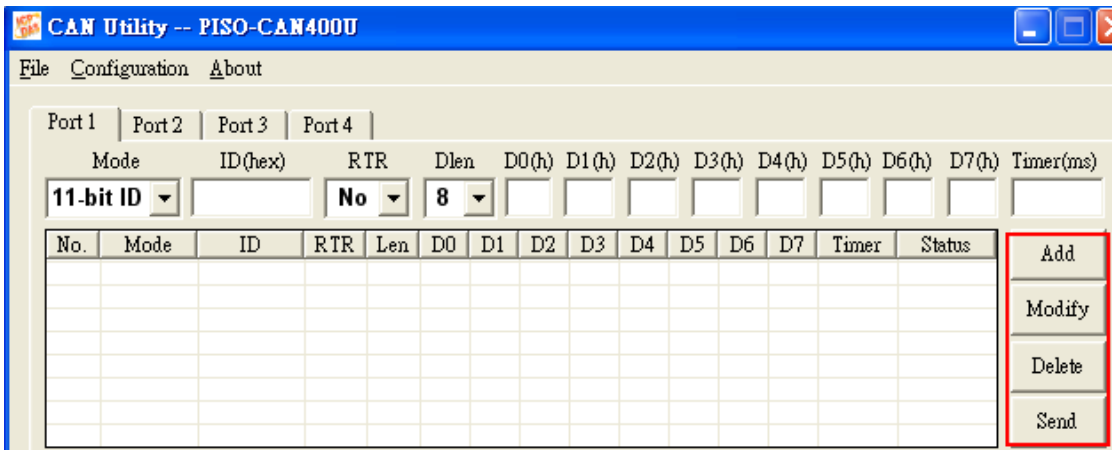
CAN 工具軟體的主要對話框如下圖所示。會有 1 個頁籤、2 個頁籤、4 個頁籤與 8 個頁籤的情況分別代表 1 個埠(PCM-CAN100)、2 個埠(PISO-CAN 200/200U, PEX-CAN200i, PCM-CAN200), 4 個埠(PISO-CAN400/400U)及 8 個埠(PISO-CAN800U)。在主對話框的底端，狀態列顯示被選擇的通訊埠的 5 種參

數，分別是板卡編號、通訊埠狀態、鮑率、接受碼、接受遮罩。



(3) CAN 傳輸功能

CAN 通訊埠傳輸的部份頁面如下圖所示，其中傳輸清單包含了下列 4 種功能。

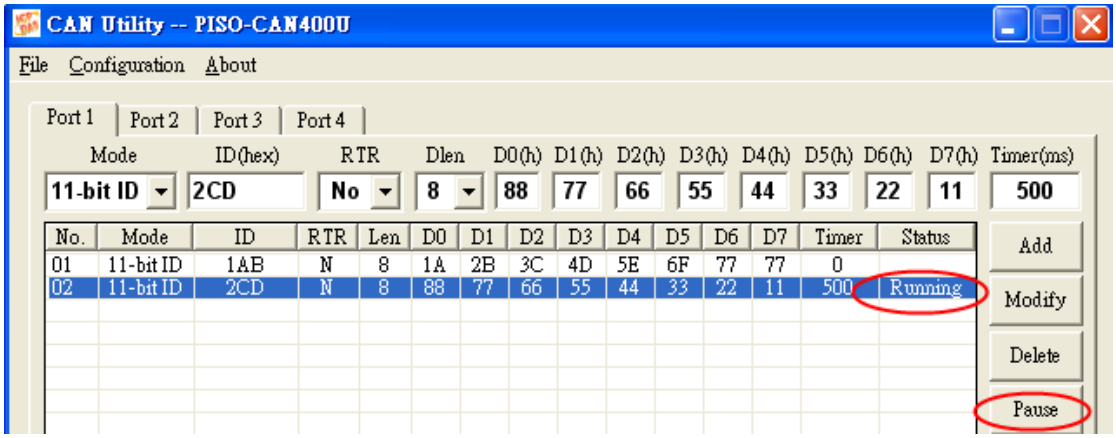


Add 按鈕: 使用者可以在文字編輯區內輸入 CAN 的訊息內容，接著點擊「Add」新增一筆訊息至傳輸清單。傳輸清單最多可容納 20 筆訊息。在新增訊息到傳輸清單後，可使用「Send」發送訊息至 CAN 網路。

Modify 按鈕:傳輸清單中，如果使用者想修改 CAN 的訊息內容，首先選擇欲修改的 CAN 訊息。接著該訊息的資訊顯示在傳輸清單上方的文字編輯區內。使用者可以直接在文字輯區內修改訊息。最後點擊「Modify」儲存修改設定。

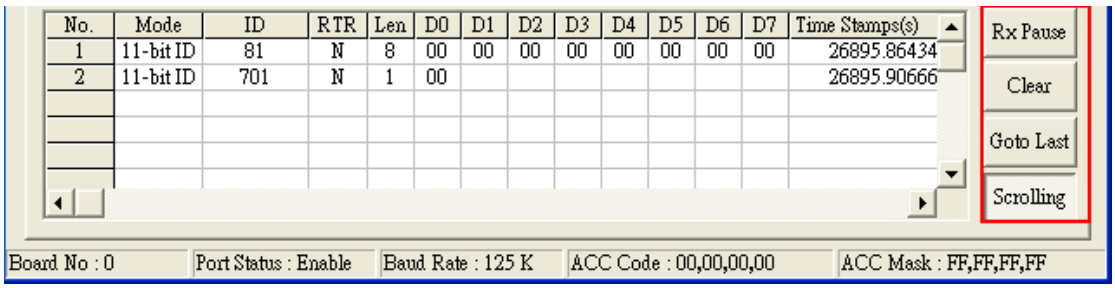
Delete 按鈕:在傳輸清單中，若有一些訊息未使用，可以選取該訊息並點擊「Delete」來刪除訊息。

Send 按鈕:從傳輸清單中選擇一筆 CAN 訊息，點擊「Send」發送時，會從所選擇的通訊埠發送訊息一次。如果 CAN 訊息的 Timer 參數不為 0，訊息發送的時間將取決於 Timer 參數週期。在這例子中，傳輸清單中的 CAN 訊息狀態顯示為 “Running” ，且在「Send」按鈕上的文字改變為 “Pause” 。如果使用者想停止訊息的傳輸，請再一次點擊此按鈕。最多允許 5 筆 CAN 訊息，同時從同一個 CAN 通訊埠循環發送訊息。



(4) CAN 接收功能

下圖顯示所選擇的 CAN 通訊埠的接收部份，接收清單包含了下列 4 種功能。



Rx Pause 按鈕: 點擊此按鈕來停止特定 CAN 通訊埠的訊息接收，再點一次便可繼續接收訊息。

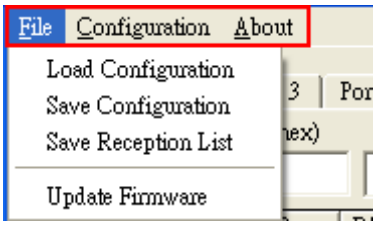
Clear 按鈕: 點擊此按鈕可刪除所有顯示在接收列表的 CAN 訊息。

Goto Last 按鈕: 點擊此按鈕可直接跳到最後接收的訊息欄位。

Scrolling 按鈕: 當此按鈕被按下時，接受清單將自動捲動至最後一筆訊息。如果這個按鈕被按起來，接受清單將停止捲動，但還是持續從 CAN 通訊埠接收訊息。此按鈕的預設狀態是被按下的。

(5) 功能選項

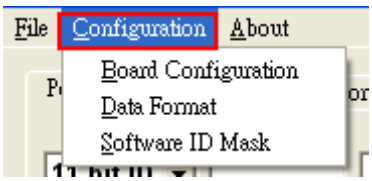
CANUtility 軟體工具有三個功能選項。



File 項目:

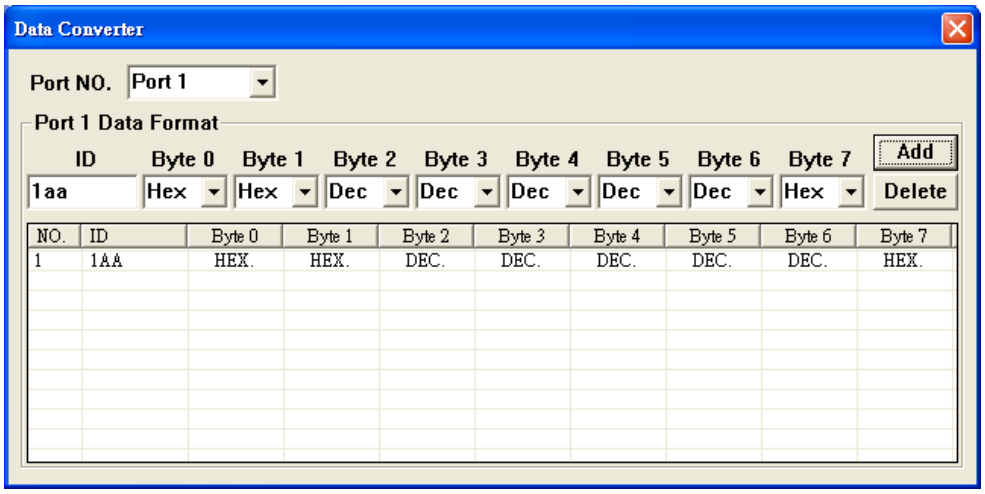
- **Load Configuration:** 如果使用者在使用 CANUtility 之前，已有儲存配置設定，可點擊此功能載入原有的設定。
- **Save Configuration:** 此功能會將每一個 CAN 通訊埠的傳輸清單、資料格式清單及 ID 遮罩清單儲存至一個 “.txt” 文字檔中。
- **Save Reception List:** 此功能用於儲存接收清單中的訊息。每一個不同 CAN 通訊埠的接受清單中，除了沒有訊息的之外，其餘的資料被儲存至 “.txt” 文字檔中。例如，如果使用者想將接收清單中的資料儲存到 “test.txt” 檔。一般來說，當使用者用 PISO-CAN400 時，資料會被儲存到四個 “.txt” 檔，text_port01.txt 、 text_port02.txt 、 text_port03.txt 、 與 text_port04.txt。如果 Port 2 的接受清單沒有資料，那麼就不會產生 text_port02.txt 這個檔案。
- **Update Firmare:** 更新 CAN 板卡的韌體。此功能只適用於 PISO-CM100/U、 PISO-CPM100/U 及 PISO-DNM100/U，而不能用於 PISO-CAN、PEX-CAN 及 PCM-CAN。

Configuration 項目:

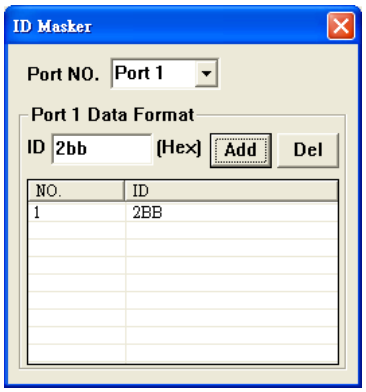


- **Board Configuration:** 使用者點擊「Board Configuration」重新配置 CAN 板卡。詳細資訊請參考本節的 “(1) CAN 配置對話框”。
- **Data Format:** 使用者可以對特定 ID 設定顯示指定的格式 (如十六進制、十進制或 ASCII)，設定對話框如下圖所示。例如，以 “ID=0x1AA” CAN 訊息的 Byte2 到 Byte6 的資料格式設定為十進制。然後，接受清單就會以十進制顯示 “ID=0x1AA” 的 Byte2 到 byte 6 資料格式，而其他位元組以十六進制表示。若有訊息沒配置資料格式，將一律以十六進制表示。在此對話框中，使用者最多可

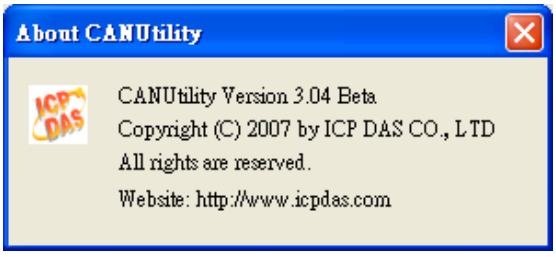
配置 20 組不同的 ID 訊息。



- **Software ID Mask:** 如果不想顯示接受表中的特定 ID 訊息，可使用此遮罩功能。如下圖所示，在 ID 遮罩清單中，使用者最多可設定 20 筆不同 ID 訊息。設定完成之後，如果 CAN 通訊埠接收的訊息與 ID 遮罩清單中所設定的一樣，則 CAN 訊息不會顯示在接受清單中。



About: 顯示關於 CANUtility 工具軟體的版本及泓格科技網址的資訊。



7 附錄

7.1 接受濾波器

四個 8 位元接受暫存器(AC0、AC1、AC2 及 AC3)與接受遮罩暫存器(AM0、AM1、AM2 及 AM3)用於各種息訊的濾波器。這些暫存器用於控制 4 位元組的濾波器，它可以檢查 CAN 訊息的特定位元並且決定 CAN 板卡是否接收該訊息。訊息濾波器的基本概念如圖 A.1 所示。接受碼暫存器主要用於決定 CAN 可以接受何種 ID 訊息，而接受遮罩暫存器主要用於決定 ID 訊息中，哪些位元須使用接受碼暫存器來檢查。如果接受遮罩中任一位元設為“0”，表示 ID 訊息中相同位置的位元需要檢查。

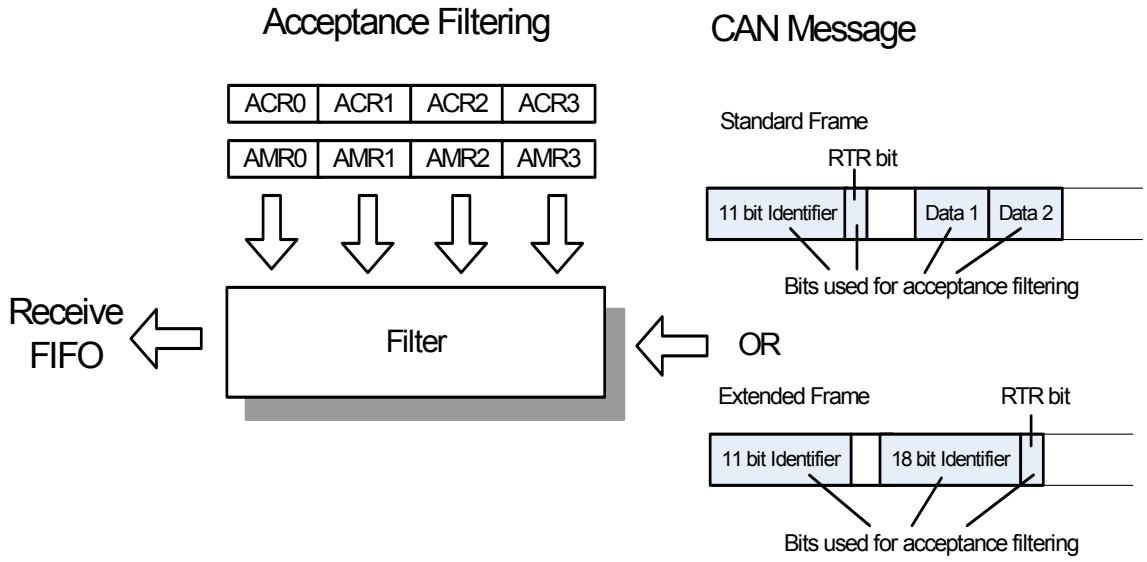


圖 A.1 接受濾波器

Example 1:

假設**有**一個標準幀(**Standard Frame**)的訊息，其接受碼暫存器(ACRn) 和接受遮罩暫存器(AMRn)設定如下。

n	0	1 (較高 4 位元)	2	3
ACRn	01xx x010	xxxx	xxxx xxxx	xxxx xxxx
AMRn	0011 1000	1111	1111 1111	1111 1111
接受訊息 (ID.28..ID.18 RTR)	01xx x010 xxxx			

("x"=忽略, 只有 ACR1 與 AMR1 較高的 4 位元被使用)

在這個例子中，ACR0 和 AMR0 用於訊息 ID 中最高的 8 位元；ACR1 和 AMR1 中較高的 4 位元用於 ID 中較低的 3 位元以及 RTR 位元；ACR1 和 AMR1 中較低的 4 位元未使用；ACR2 和 AMR2 用於 CAN 訊息的第一個位元組；ACR3 和 AMR3 用於 CAN 訊息的第二個位元組。因此，此 CAN 訊息不管是否為遠端傳輸請求訊息，只要該訊息 ID 的格式為 "01xx x010 xxx" 就會被接受。(x 代表 "忽略")

Example 2:

假設有一個**延伸幀(Extended Frame)**的訊息，其接受碼暫存器(ACRn)和接受遮罩暫存器(AMRn)設定如下。

n	0	1	2	3(較高 6 位元)
ACRn	1011 0100	1011 000x	1100 xxxx	0011 0xxx
AMRn	0000 0000	0000 0001	0000 1111	0000 0111
接受訊息 (ID.28..ID.0 RTR)	1011 0100	1011 000x	1100 xxxx	0011 0x

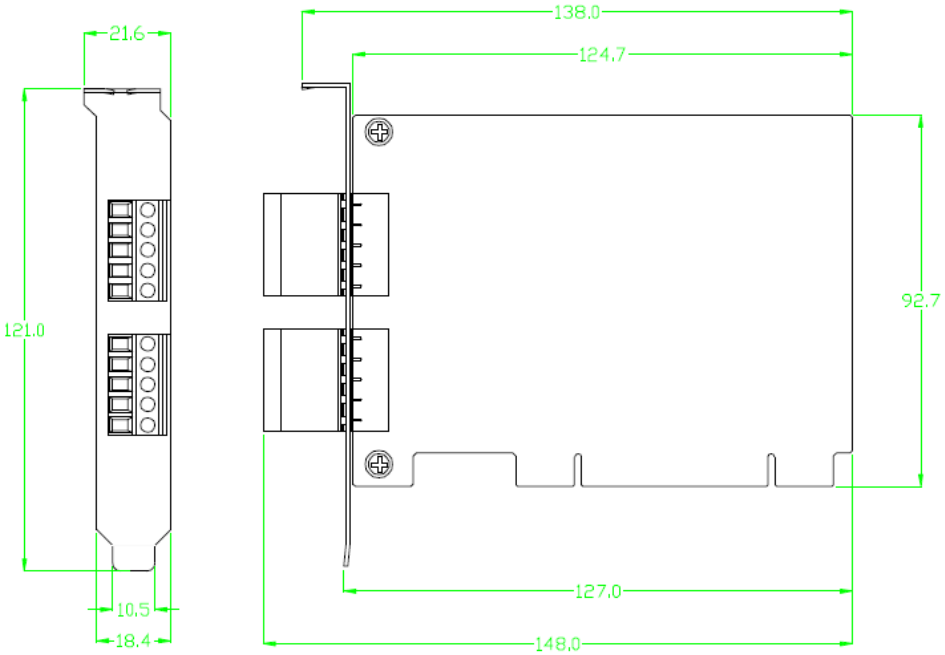
("x"=忽略，只有 ACR3 和 AMR3 較高的 6 位元被使用)

在這個例子中，AMR3 和 AMR3 較低的 2 位元未使用。接受碼與接受遮罩的其他位元，全部用於 29 位元的訊息 ID 及 RTR 位元。因此，不管 CAN 訊息是否為 RTR(遠端傳輸請求)，只要訊息 ID 格式如 "1011 0100 1011 000x 1100 xxxx 0011 0x "(x 代表 "忽略")就會被接受。

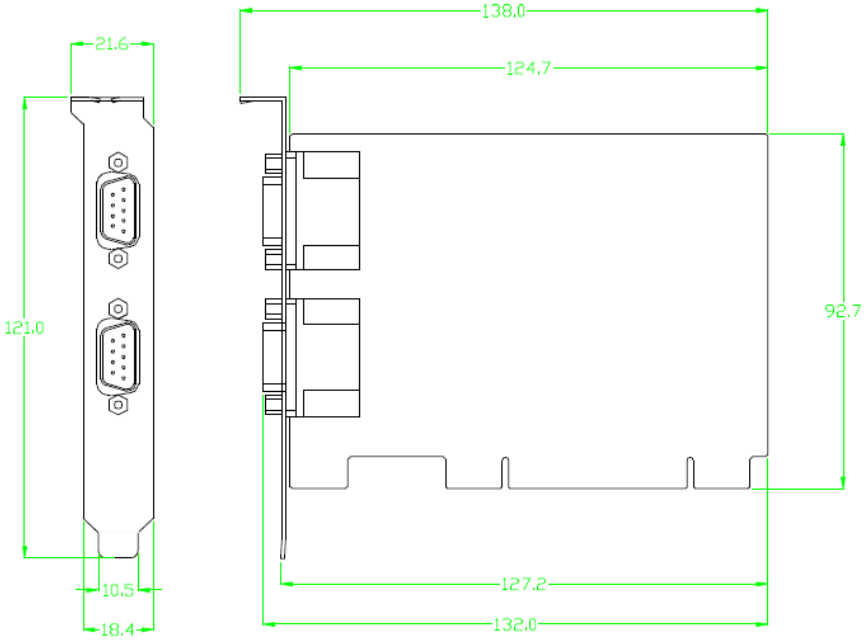
8 尺寸圖

8.1 PISO-CAN200/400

PISO-CAN200/400-D/T



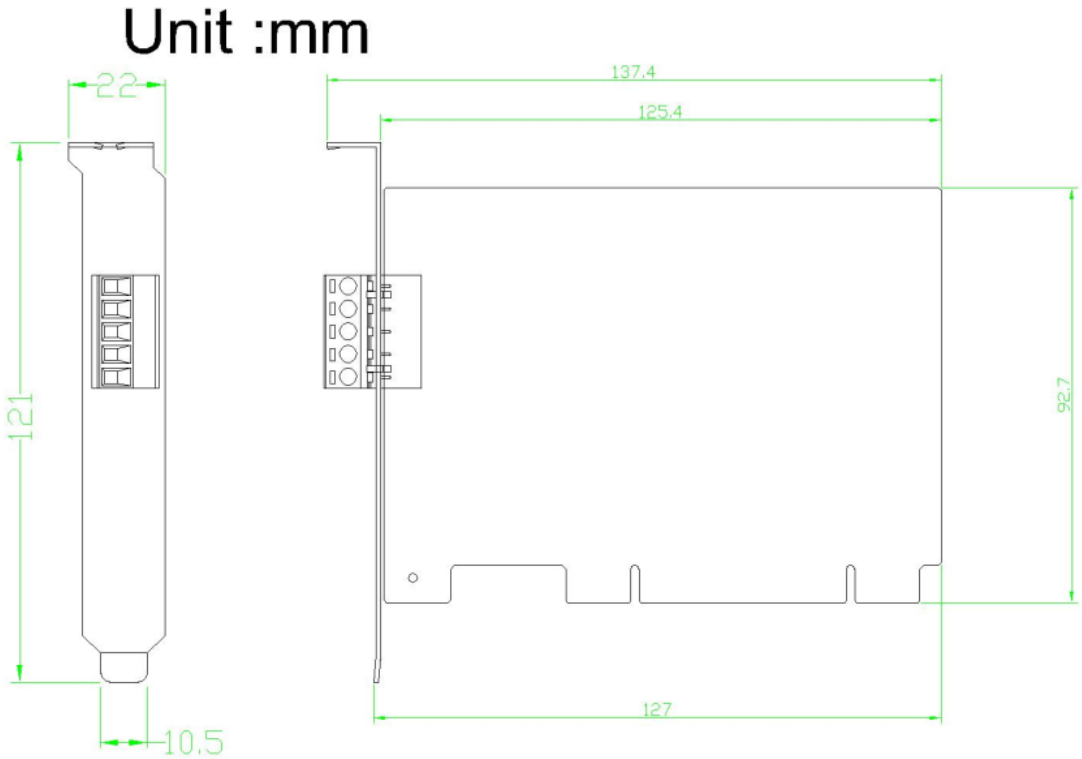
PISO-CAN200/400-T



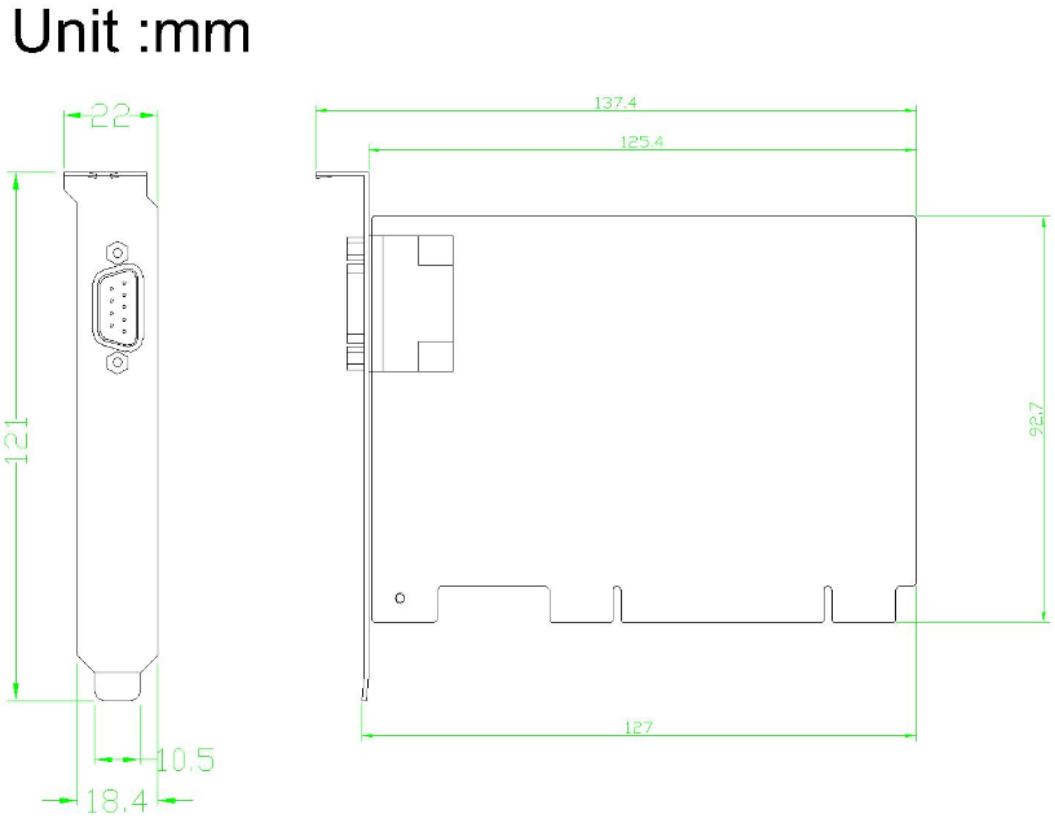
PISO-CAN200/400-D

8.2 PISO-CAN100U/200U/400U/800U

PISO-CAN100U-D/T

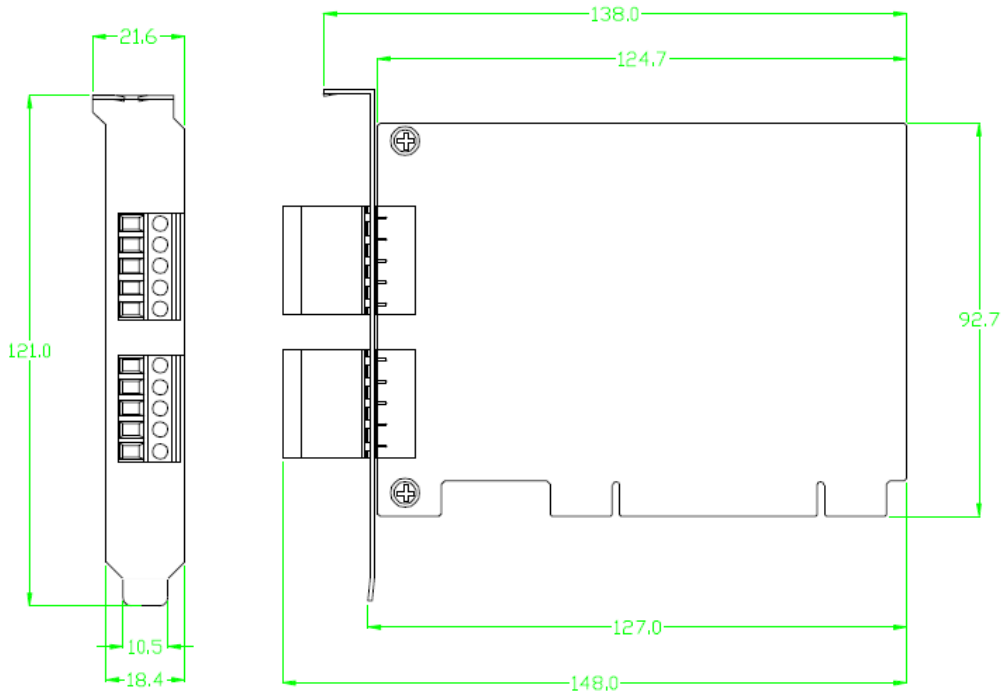


PISO-CAN100U-T

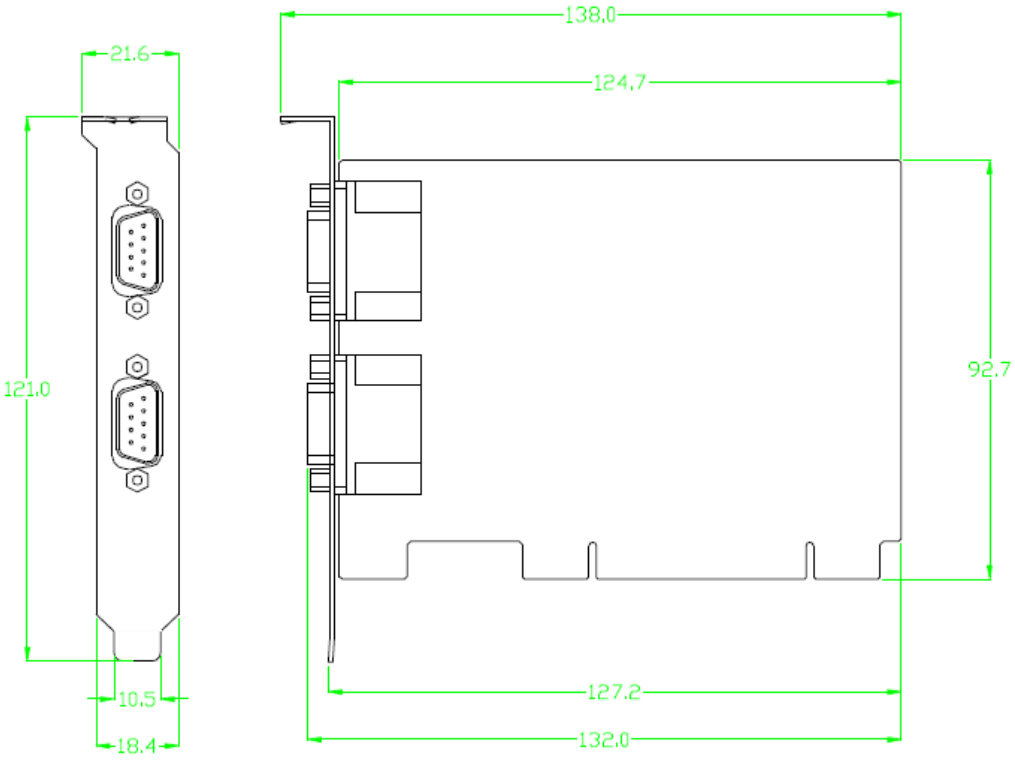


PISO-CAN100U-D

PISO-CAN200U/400U-D/T

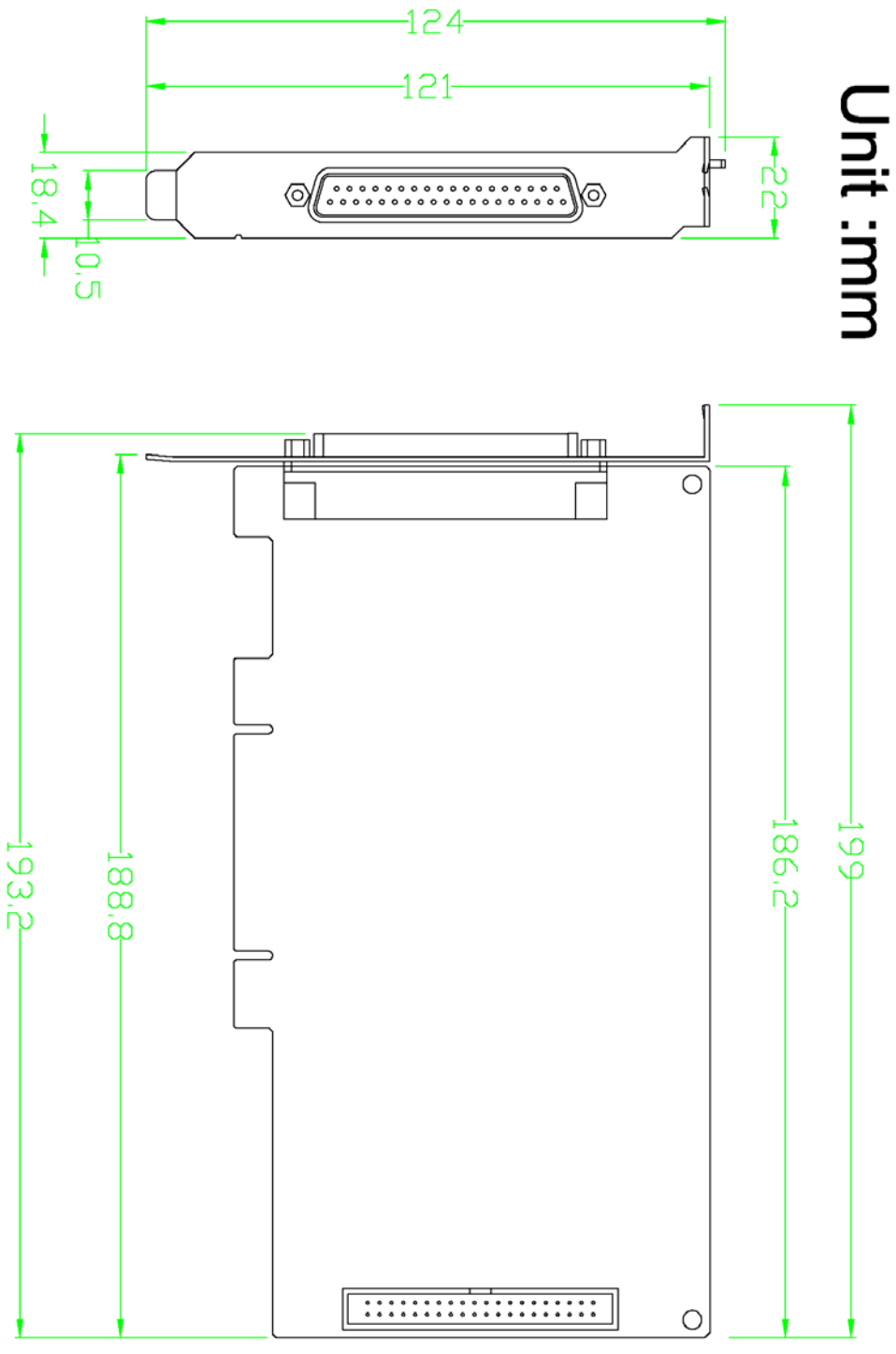


PISO-CAN200U/400U-T



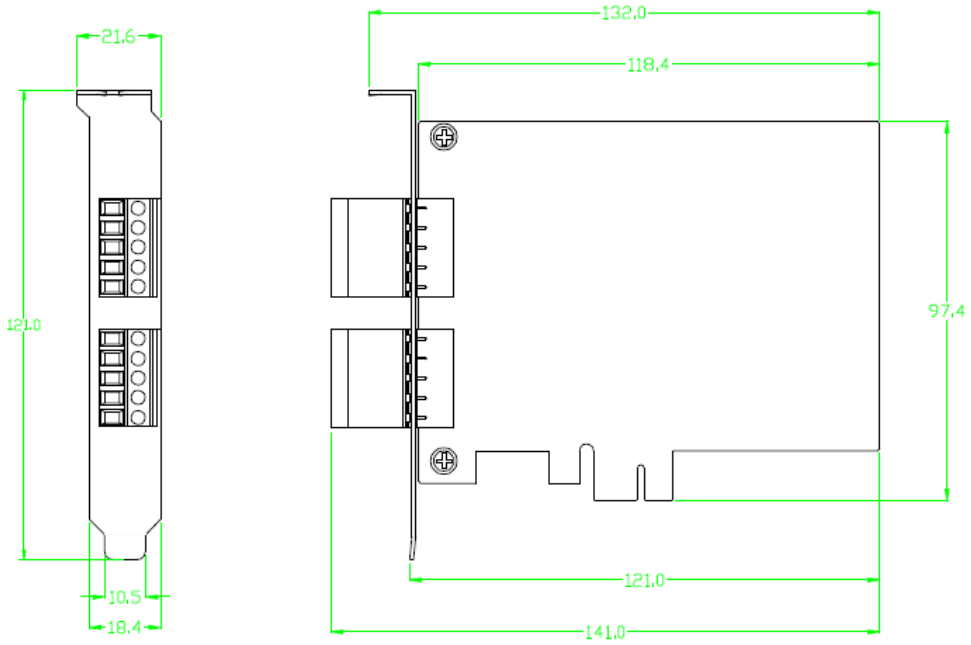
PISO-CAN200U/400U-D

PISO-CAN800U-D/T

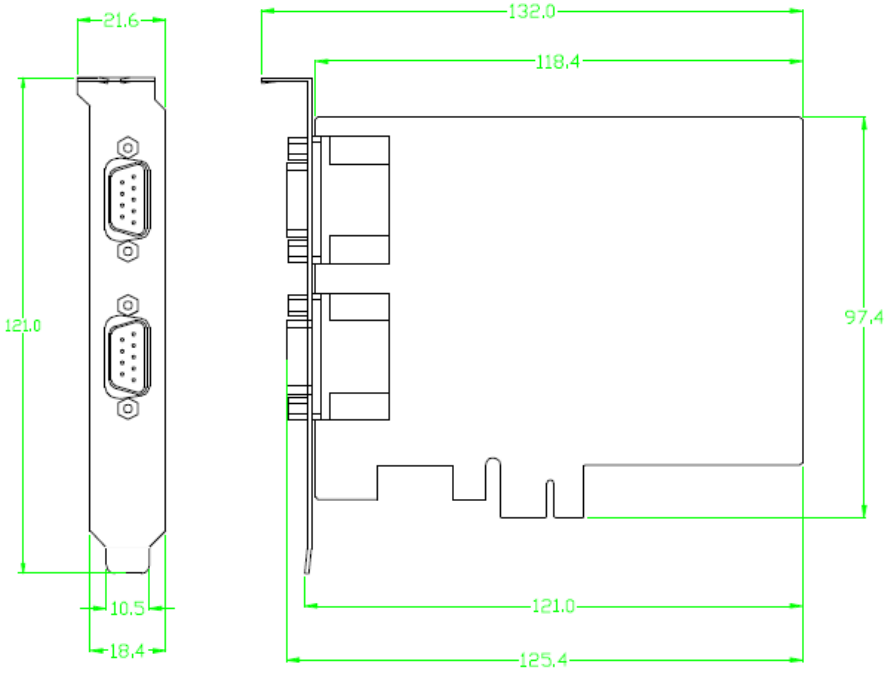


8.3 PEX-CAN200i

PEX-CAN200i-D/T



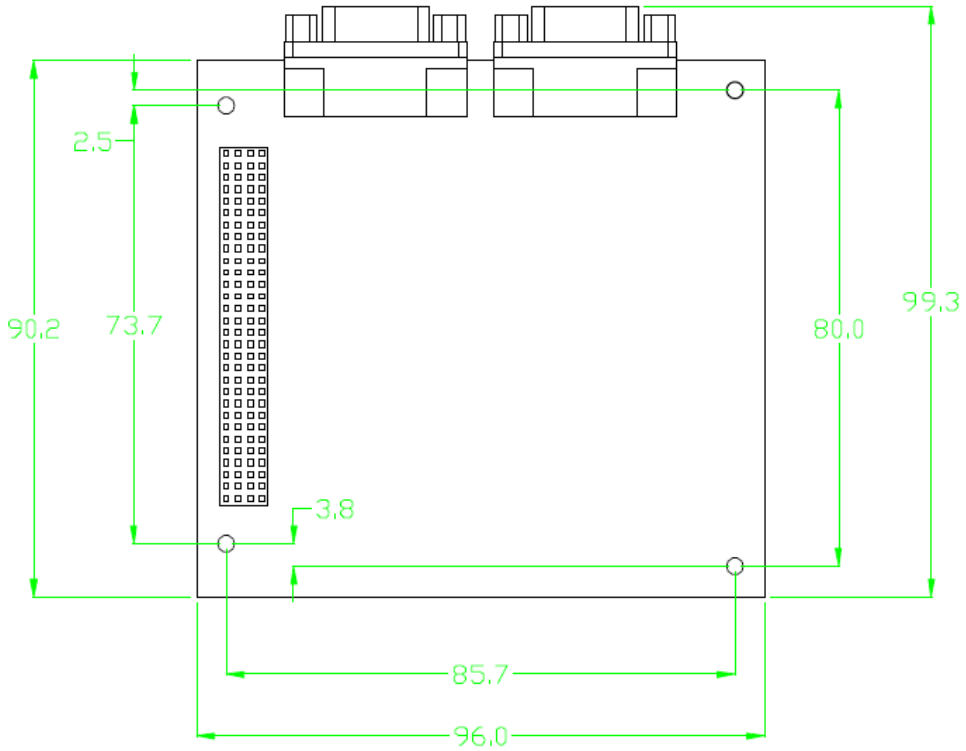
PEX-CAN200i-T



PEX-CAN200i-D

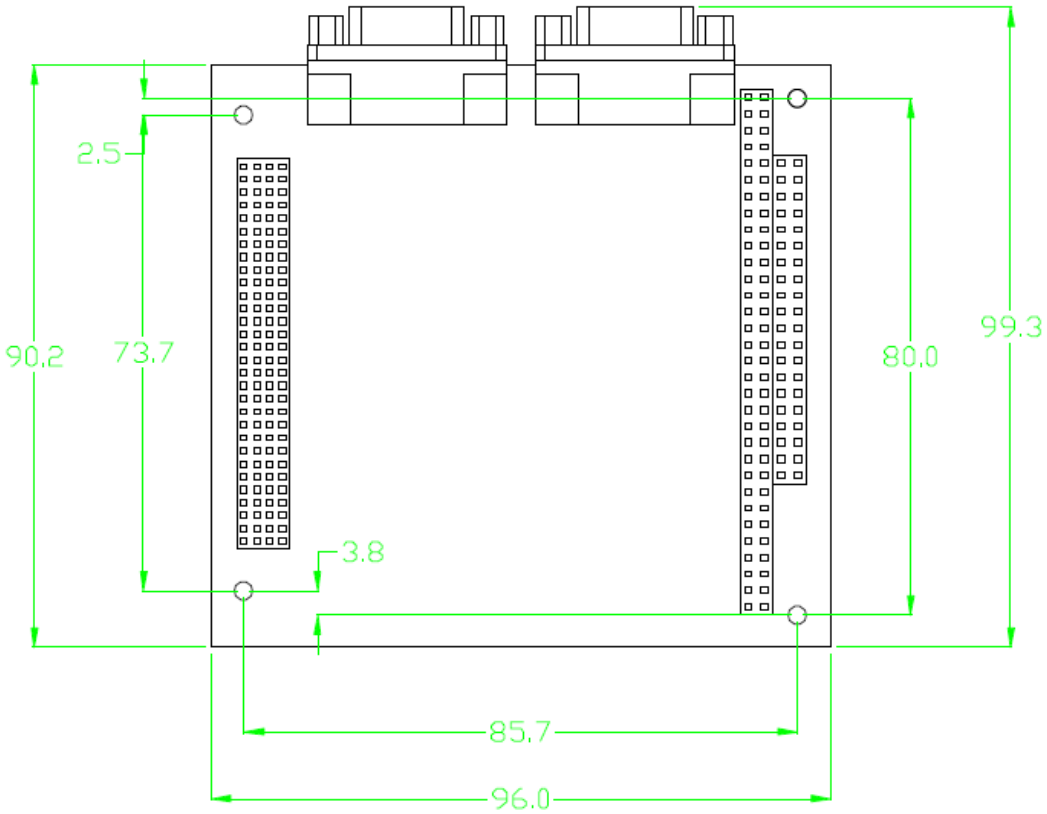
8.4 PCM-CAN100/200

PCM-CAN100/200-D



Unit :mm

PCM-CAN200P-D



Unit :mm