第一章　簡介和安裝

AMIDA GPIB 控制卡的設計是完全相容於 IEEE 488.1 及 IEEE 488.2 標準，採用 PCI 介面，支援隨插即用(Plug & Play)規範，支援 Windows 95/98/Me/NT4.0/2000/XP 作業系統，驅動程式函式庫包含有 Visual C++、Borland C++ Builder、Labwindows、Visual Basic、Delphi 及 Labview 等視窗發展軟體工具。

此外，支援的函式庫提供 NI-like 的命令控制語法格式，方便使用者順利發展及移轉應用程式。本介面卡並提供交談式視窗 GPIB 介面程式，方便使用者可以在不需撰寫程式下直接控制儀器。

AMIDA GPIB 控制卡特性如下：

- 採用 PCI 介面，支援隨插即用規範。
- 相容於 IEEE 488.1 & IEEE 488.2 標準。
- 支援 Windows 95/98/Me/NT4.0/2000/XP 作業系統。
- 提供 NI-like 驅動程式函式庫。
- 支援 Visual C++、Borland C++ Builder、Labwindows、Visual Basic、Delphi 及 Labview 等視窗發展軟體工具。
- 提供交談式視窗 GPIB 介面程式

1.1 安裝及設定

1. 首先將驅動光碟片至於光碟機中，執行 Setup.exe 來安裝介

面卡所提供的工具程式與各種視窗發展工具的函式庫(**注意 AMIDA GPIB 介面卡不要插入電腦 PCI 擴充槽中)**。

2. 關閉電源，將 AMIDA GPIB 介面卡插於電腦 PCI 擴充槽中，鎖緊固定鐵片上之螺絲，然後重新開機。

3. Windows 95/98 作業系統安裝

Windows 95/98 作業系統支援隨插即用規範，若您的電腦尚未安裝有本介面卡，則開機後 Windows 作業系統會出現找到新的硬體“PCI Communication Device”之訊息視窗，然後跳出**新增硬體精靈**畫面如圖 1-1 所示：

圖 1-1

按“**下一步**”後出現圖 1-2 所示畫面，勾選“搜尋裝置的最適用的驅動程式”後

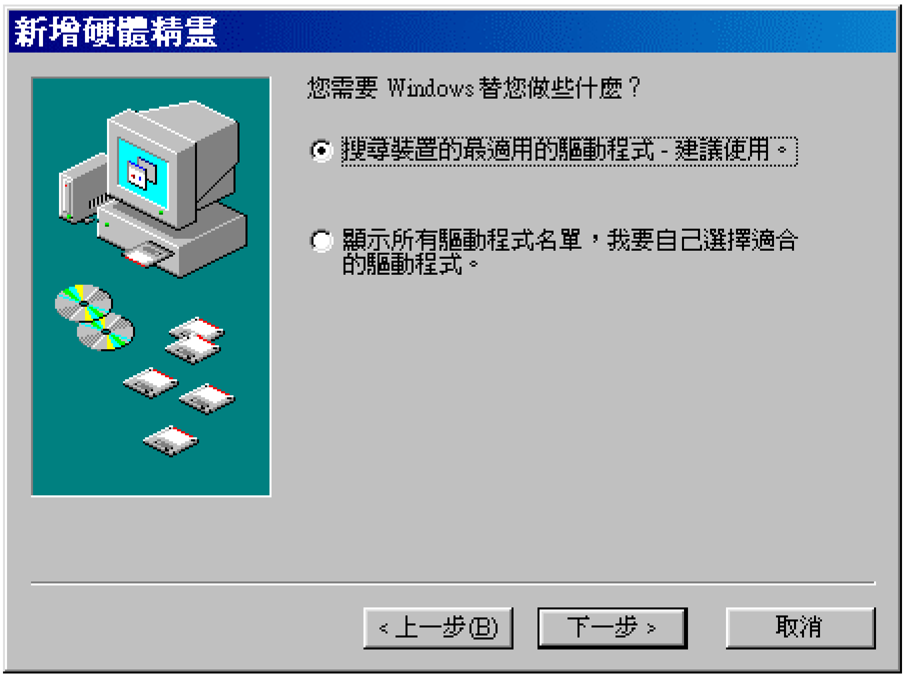

圖 1-2

按“下一步後，出現如圖 1-3 所示畫面，勾選

圖 1-3

“CD-ROM 光碟機”選項，並將驅動光碟片置入光碟機中，選取“下一步”按“Enter”鍵入，請稍後片刻，此時作業系

統會搜尋符合的裝置驅動程式，最後出現圖 1-4 畫面，

圖 1-4

選取”下一步” 按 “Enter” 鍵，作業系開始安裝驅動程式，完成後出現圖 1-5 所示，按 “Enter” 鍵後即完成驅動程式安裝。

圖 1-5

依上述步驟,介面卡的驅動程式已經正確地裝設於作業系統上。

4. Windows Me/2000/XP 作業系統安裝

上述各作業系統均支援隨插即用規範,若您的電腦尚未安裝有本介面卡，則開機後 Windows 作業系統會出現找到新的硬體“PCI Communication Device”之訊息視窗，驅動光碟片至於光碟機中，請稍後片刻，等待系統自動蒐尋後會跳出**新增硬體精靈**畫面如圖 1-6 所示：

圖 1-6

選取”下一步“按“Enter”鍵，作業系開始安裝驅動程式，完成後出現圖 1-7 所示，按“Enter”鍵後即完成驅動程式安裝。

圖 1-7

5. NT 4.0

Windows NT 4.0 版本作業系統是不支援隨插即用，所以作業系統是無法自動偵測本卡。使用者只要執行上述第一項所述執行 Setup.exe 即可自動設定好 NT 4.0 所需之驅動程式及各類函式庫。

1.2 目錄說明

安裝好軟體後，所有驅動程式函式庫及工具程式存於目錄 ..\Program Files\ATIGPIB 下，分別為

\ATIGPIB\Utlity\ 交談式工具

\ATIGPIB\Driver\ 視窗體發展工具函式庫

\ATIGPIB\Driver\VC Visual C++函式庫驅動程式及範例程式

\ATIGPIB\Driver\VB Visual Basi 函式庫驅動程式及範例程式

\ATIGPIB\Driver\BC Borland C++ Builder 函式庫驅動程式及範例程式

\ATIGPIB\Driver\ Delphi Borland Delphi 函式庫驅動程式及範例程式

\ATIGPIB\Driver\ LW Labwindows 函式庫驅動程式及範例程式

\ATIGPIB\Driver\ LV LabView 函式庫驅動程式及範例程式

1.3 功能測試

首先執行硬體測試程式 ATITEST.exe。於命令提示字元執行

C：\Program Files\ATIGPIB\Utility> **ATITEST**

此測試程式會測試介面卡是否安裝正確。

接著，準備一台具有 GPIB 介面的儀器(以 HP 33401A 多功能電表為例)連接於介面卡上，然後執行交談式 GPIB 介面程式 ATIGPIB.exe 來測試基本 GPIB 命令通訊是否正常。執行**開始|程式集|ATIGPIB**|ATIGPIB 或從命令提示字元鍵入

C：\ Program Files\ATIGPIB\utility>ATIGPIB　會出現如圖 1-9

所示畫面，執行選單 **開啟 | 開啟新裝置**，可得圖 1-10 之畫面，

圖 1-9

在'**主要位址**'選項設定欲控制之 GPIB 儀器位址，'**逾時時間**' 選項設 3 秒後按"**確定** "，回到圖 1-11 所示畫面，顯示已經成功地開啟一新裝置，接著即可對此裝置作動作。

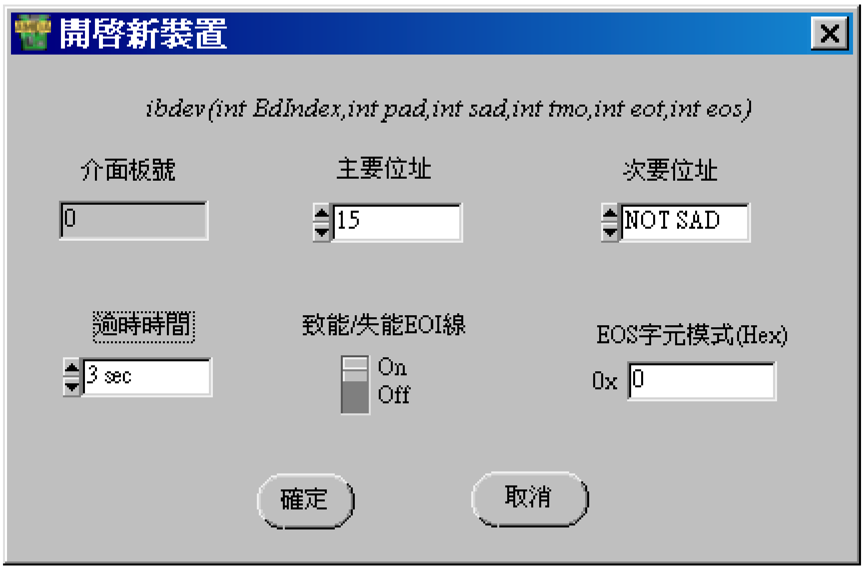

圖 1-10

圖 1-11

執行**寫入及讀出**按鈕，可得圖 1-12 畫面，查詢出 HP34401 儀器的型號資訊。

圖 1-12

第二章 GPIB 簡介

GPIB介面最早是由美商惠普 (Hewlett-Packard) 公司所發展出來，作為自己公司內部儀器間的連接介面，那時稱之為HPIB (Hewlett-Packard Interface Bus) 。1975年，美國電機電子工程協會 (IEEE) 依據HPIB為基礎，公佈了ANSI/IEEE Std 488.1-1975，稱為可程式化儀器之IEEE標準數位介面 (IEEE Standard Digital Interface for Programmable Instrument，簡稱IEEE 488-1975) ，它規範了連接器 (Connector) 和電纜線 (Cable) 間的電氣特性與機械特性，也定義出匯流排間資料傳輸交握 (Handshaking) 協定。1978年IEEE又對1975年所訂的標準做了第一次的修訂，稱為IEEE 488-1978。根據以上所述，可以得知HPIB、GPIB及IEEE 488等，指的都是同樣的標準 — IEEE 488-1978標準。

IEEE 488-1978只對硬體 (電氣、機械特性及匯流排協定等) 作了詳細地規範，在韌體程式上卻無明確地定義，例如命令格式的語法、參數型態的定義及結束字元的使用等，使得某些儀器間出現了GPIB相容性的問題。有鑑於此，惠普與太克 (Tektronix) 等儀器大廠，聯合於1987年重新訂定IEEE 488，將其分成了硬體上的標準－IEEE 488.1-1987及軟體上的標準－IEEE 488.2-1987。

IEEE 488.1-1987即為IEEE 488.1-1978，而IEEE 488.2-1987卻為新定的標準規範，命名為IEEE標準碼、格式、協定及共同命令(IEEE Standard Code， Formats， Protocols， and Common Commands) ，它定義了①控制命令的表頭格式，②參數的型態，③共同命令集及④狀態回報格式等。如此，在軟體程式設計有了新的標準，即可解決軟體相容性的問題。

2.1 GPIB匯流排的基本架構

2.1-1 裝置連接方式

IEEE 488介面，可將控制器 (電腦) 、量測儀器 (裝置) 等以並聯方式連接在一起，形成一自動量測系統。連接方式可分成線式(Linear) 串接與星形 (Star) 連接兩種方式。線性串接方式是以一台串聯著一台的方式，連接成一測試迴路。而星形連接方式，圖中，以一台裝置為中心，幅射連接至其他裝置。此種連接方式有個缺點，為受限於連接器重疊之數目，重疊數目最好不要超過三個，以免最下方的裝置的連接器承受太大的力量。

2.1-2 IEEE 488.1 電纜線長度的限制

GPIB匯流排為了要達到資料高速傳輸的目的，在IEEE 488.1裡對於電纜線的長度有以下的限制：

(1) 一個系統內，電纜線的總長度不得超過2公尺乘以連接裝置的數目，但最大長度不得超過20公尺。

(2) 一個系統內，所能連接裝置數目不得超過15個，而且其中至少要有2至3個裝置是在開機狀態。

2.1-3 IEEE 488 訊號線

IEEE 488匯流排是由16條訊號線與8條地線及隔離線所組成，如圖2-1所示，為IEEE 488連接器的接腳圖，圖中詳細定義24條匯流排所在的位置，其中16條訊號線依其功能可區分成三大類：

- 八條資料線 (Data Line)。
- 五條介面管理線 (Management Line)。
- 三條交握線 (Handshake Line)。

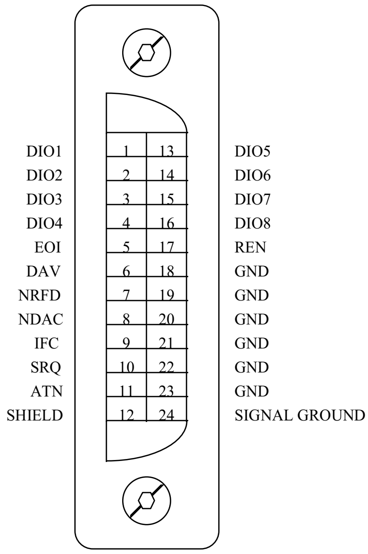

圖2-1　　IEEE 488 連接器之接腳圖

1. 八條資料線

IEEE 488.1標準裡定義之資料線共有八條，稱為DIO1、DIO2‧‧‧至DIO8，用來傳輸命令訊息 (介面訊息) 和資料訊息 (裝置相關訊息) 之用。。

2. 五條管理線

IEEE 488匯流排裡共定義有五條管理線，分別分別說明如下。

(1) IFC(Interface Clear)線

IFC管理線，為系統控制器重置裝置之用。

(2) ATN(Attention)線

ATN管理線，用來指示在匯流排八條資料線上的資料為命令訊息或則是資料訊息。

(3) REN(Remote Enable)線

REN管理線，是裝置用來切換裝置為遠程控制(Remote Control) 或是本地控制 (Local Control) 。所謂遠程控制乃是透過GPIB匯流排所傳送過來的裝置相關訊息進行控制，而本地控制則是利用裝置本身面板按鈕開關進行控制。

(4) EOI(End-or-Identify)線

EOI管理線，是裝置 (發話者) 所傳送的資料序列至最後位元組時，令EOI線動作 (拉至低準位) ，表示資料傳輸結束。相對地，收聽者在判別其EOI線被拉至低電位時，即停止讀取資料。EOI線可與NL (New Line) 字元及 CR (Carriage Return) 字元相互結合成結束符號。

(5) SRQ(Service Request)線

SRQ管理線，是裝置用來通知值班控制器，其需要控制器的服務。當SRQ管理線所處的電位為低準位時，表示裝置須要服務，此時裝置須一直保持低準位，直到其被值班控制器循序輪詢完後，才可將SRQ線電位拉高，解除服

務要求。因此，相對地值班控制器就必須隨時監測其SRQ信號，以判斷是否有裝置要求服務中斷。

3. 三條交握線

IEEE 488.1標準定義了三條交握線，作為裝置間的非同步資料傳輸， GPIB 匯流排使用此三條交握線間的互鎖交握技術(Inter Locking Handshake Scheme)，來確保裝置間所傳輸或接收的資料不會遺失。三條交握線定義如下：

(1) DAV(Data Valid)：資料有效線

(2) NRFD (Not Ready For Data)線：尚未備妥接收資料線

(3) NDAC(Not Data Accepted)：尚未接受資料

2.1-4 裝置角色的扮演

在IEEE 488匯流排裡，為了確保資料傳輸的正確性，根據裝置控制匯流排的時機不同，可區分成控制器 (Controller) 、發話者(Talker)、收聽者 (Listener) 三種角色，分述如下：

1. 控制器

一般來說，一個由GPIB介面所組成的測試系統，是由一台電腦與許多不同種類的儀器所組成，此時，電腦即稱為控制器。若在一系統裡有多台電腦連接在一起，那麼目前具有控制能力的電腦

即稱為值班控制器，值班控制器可經由控制權的移轉，將控制權轉移至空閒 (Idle) 的控制器 (電腦) 裡。一個裝置要能稱為控制器，需具備以下四點能力：

① 管理匯流排之通信連接管道。

② 回應裝置之服務要求中斷。

③ 送出 GPIB 命令訊息。

④ 移轉／接收匯流排控制權。

2. 發話者

大部份IEEE 488裝置都具備有發話者的功能，其所需具備的能力為：

① 可被控制器設定成具有發話的能力。

② 有能力向GPIB匯流排傳送資料。

③ 任一時刻匯流排上只能有一裝置發話。

3. 收聽者

大部份的IEEE 488裝置都具備有收聽者的功能，其所需具備的能力為：

① 可被控制器設定成具有收聽的能力。

② 具有由GPIB匯流排接收資料之能力。

③ 任何時刻，匯流排上最多可允許14個收聽者同時接收資料。

2.2 介面機能

IEEE 488.1裡所定義的介面機能，是用來產生、處理或接收介面訊息，以使介面訊息能夠正確地在裝置間傳遞，而達到通訊的目的。IEEE 488.1所定義的介面機能共有十種，但並不是所有裝置都需要實現這十種機能，完全依照裝置需求而定，但是裝置若實現其中某一機能時，就必需遵守該介面機能的通訊協定。

每個介面機能又可以進一步分成很多種的副機能 (Subset) ，在IEEE 488.1 裡的規格書的附錄裡有詳細地記載。因此，為了能顯示出某一裝置所具有的介面機能能力，標準裡建議裝置在其手冊或GPIB連接器附近，標記出如圖2-2所示之記號。在每一介面機能英文縮寫字母後面接上數字，用以表示該機能之層次(Level) ，亦即為副機能，此數字若為0表示無該機能。IEEE 488.1裡所定義的十大介面機能及其副機能為：

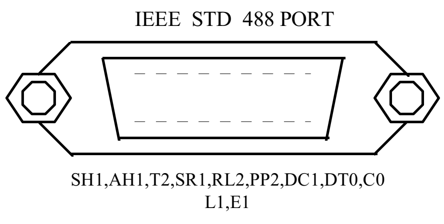

圖2-2 IEEE 488介面機能範例

1. SH，發送交握介面機能

發送交握介面機能主要的作用是使裝置具有一能有效地傳輸多線訊息的能力，其與一個或多個接受交握介面機能間的互鎖交握程序能夠確保每一多線訊息的非同步傳輸。

2. AH，接受交握介面機能

接受交握介面機能主要的作用是使得裝置具有一能有效地接收多線訊息的能力，其與一個或多個發送交握介面機能間的互鎖交握程序，能夠確保每一多線訊息的非同步傳輸。

3. T，發話者介面機能 (包含循序輪詢能力)

發話者介面機能主要的作用是使得裝置具有一能傳送裝置相關資料 (資料訊息，且包含循序輪詢時所回傳之狀態位元組暫存器之值) 至匯流排的能力；此能力只有在裝置被控制器設定成發話者才存在。

4. L，收聽者介面機能

收聽者介面機能主要的作用是使裝置具有一能接收裝置相關資料 (資料訊息，包括狀態位元) 的能力。此能力只有在裝置被控制器設定成收聽者才存在。

5. SR，服務要求介面機能

服務要求介面機能主要的作用是提供裝置具有一能以非同步方式向值班控制器要求服務，並且能以同步方式在循序輪詢時將狀態位元組暫存器的內含值送出。

6. RL，遠程與本地介面機能

遠程與本地介面機能主要的作用是提供裝置具有一選擇遠程與本地兩種輸入資料模式的能力。採用本地模式是允許裝置可以從面板上設定其操作程序；反之，採用遠程工作模式是透過IEEE 488匯流排而來的裝置相關訊息，來設定其操作程序。

7. PP，並行輪詢介面機能

並行輪詢介面機能主要的作用是使得裝置在值班控制器執行並行輪詢時，可以利用PPR (Parallel Poll Response) 訊息回應給控制器，使得控制器能具以分辨出在匯流排上是誰在要求服務。

並行輪詢和循序輪詢，雖然都可以用來回應裝置的服務要求，但是在程序上還是有些差異，例如：

① 實行並行輪詢時，事先值班控制器需先指定那一條DIO線為該裝置之對應線，而循序輪詢卻不需事先設定。

② 並行輪詢一次可以回應八個裝置的狀態 (八條DIO線) ，而循序輪詢，卻要依序輪詢每一次裝置的狀態位元，以判別其是否要求服務。

8. DC，裝置清除介面機能

裝置清除介面機能主要的作用是提供裝置具有被清除 (初始化) 的能力。

9. DT，裝置觸發介面機能

裝置觸發介面機能主要作用是一個或一群裝置具有一被觸發而執行其本身制定的基本操作能力。所謂一群指的是系統裡一部份的裝置或全部裝置。裝置觸發介面機能可藉著利用介面訊息GET來觸發裝置，以達同步的作用。

10. C，控制器介面機能

控制器介面機能主要的作用是提供裝置具有傳送裝置位址、通用命令和定址命令至其他裝置的能力。此外，此機能尚具有執行並行輪詢的能力，以便能決定是那一個裝置要求服務。

2.3 循序輪詢與並行輪詢

一般正常情況下，整個系統控制流程都是以控制器為主體，由其指派何者為發話者，何者為收聽者，從而進行資料的傳遞。雖然如此，還是會有很多的突發狀況發生，須要立刻通知控制器，以便作即時的處理。例如印表機的紙張用完了，須要馬上通知控制器以停止資料的列印；或則是時間計數器已達到預設值，也須要

立即知會控制器做適當的處理。要能達成上述的要求，裝置就必需用“中斷”的方式通知控制器，這樣控制器才能及時做些回應的措施。

IEEE 488.1 硬體規格裏定義有一訊號線稱為 SRQ 線。SRQ 線平常的準位為“H”準位，一旦裝置有急事要控制器馬上處理，就把此 SRQ 線的準位拉成“L”，以中斷控制器。而在控制器的程式設計中就必須要有中斷服務副程式，來執行相對應的工作。中斷服務副程式首要的工作就是確認到底是那一種裝置產生 SRQ 中斷，確認的方式有兩種，第一種方式稱為“循序輪詢”，第二種稱之為“並行輪詢”。採取循序輪詢時，控制器在接收到 SRQ 訊號後會將裝置的狀態位元組暫存器 (Status Byte Register) 值傳回，控制器檢查裝置所傳回的狀態位元組暫存器的位元六 (RQS 位元) 的值是否為 1，若為 1 表示此裝置要求中斷服務，則跳到相對應副程式執行相對應的工作；若不為 0 則控置器繼續呼叫下一個發話者，直到找到發送 SRQ 信號之裝置為止，如此逐一詢問，故稱呼此方法為循序輪循。至於並行輪循則是每一裝置分配一條 DIO 線，控制器一次讀取 8 條 DIO 線值即可得知發話者的狀態，從而判斷何者須要服務，由於一次可詢問八個裝置，故稱之為並行輪詢。一般來說，並行輪詢詢問那個裝置需要服

務，主要是根據 ist 狀態來判別，控制器可以隨時實行並行輪詢，並不一定要利用 SRQ 中斷才回應，只有循序輪詢一定要利用 SRQ 中斷動作才能啟始。

2.3-1　循序輪詢

循序輪詢動作的方式為當一裝置需要服務時，會送出SRQ信號 (SRQ訊號線為真，低電壓準位) 通知控制器，接收到SRQ信號的控制器，會依照順序逐一輪詢各個裝置，被呼叫的裝置即送出狀態位元組暫存器之值，控制器接收到此一狀態位元組資料後，加以檢查位元6之值，判斷值是否為1，若為1表示此裝置要求服務。

2.3-2　並行輪循

前述循序輪詢方式是控制器按照順序逐一詢問裝置是否要求服務，因此，若該要求服務的裝置，恰好在最後才被輪詢，那麼此舉勢必會延遲到整個系統時效上的需求，此時若改用並行輪詢會變得較有效率。

所謂並行輪詢是將八條DIO線分配給八個裝置，每條DIO線代表著該裝置的狀態，如此，可藉著資料的讀取，一次判斷八個裝置

的狀態，即可知那一裝置要求服務。此種分配裝置是使用那一條DIO線來表示其狀態的動作，稱之為構建 (Configure) ，並行輪詢中構建的方式有兩種，一種稱之為本地構建 (PP2) ，其構建方式為只要按照裝置使用手冊的說明將開關設定成所需的狀態即可。而另一種方式稱為遠程構建 (PP1) ，其構建方式為接受從控制器所下之介面訊息命令 (PPE) 來設定使用之模式。

當整個構建完畢之後，控制器令ATN及EOI線上之電壓準位同時變成“L” (即IDY介面訊息) ，以通知全部發話者現在已進入並行輪詢狀態，發話者在接收到此一訊號後，會將表示自己之狀態位元狀態 (狀態位元值之決定，於後途述) 送出至預先被分配的DIO線上 (PPR訊息)，整個動作如圖2-3所示。接著控制器依據三線交握協定，從匯流排上讀取DIO1～DIO8上之各個發話者所傳送來的狀態信號，即可得知裝置之狀態。

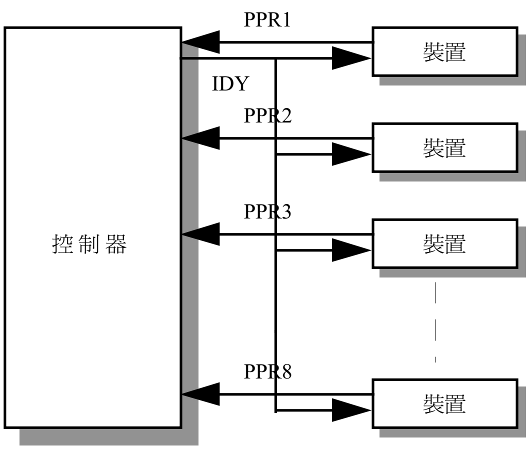

圖2-3　並行輪詢之訊息交換

圖2-4所示為並行輪詢採用遠程構建之程序圖，首先，在命令模式下對所欲構建之裝置定址為收聽者，而後送出如下命令。

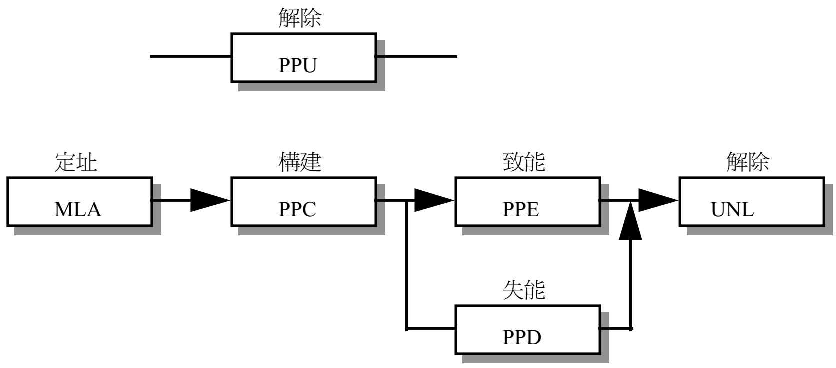

圖2-4 並行輪詢之遠程構建程序圖

(1) PPC：並行輪詢構建命令，通知裝置現已進入並行輪詢之遠程構建模式，其後需緊接著二次命令，PPE或PPD。

(2) PPE：並行輪詢致能命令。作用為將資料線分配給被選址之裝置，其命令之位元組合為

Y110SPPP

其中，PPP (DIO1至DIO3) 之組合值加一，即為裝置被分配的DIO線位置，可分成PPR1至PPR8訊息，如表2-1所示。，至於S位元稱為感測位元 (Sense Bit) ，在回應並行輪詢時，若代表裝置之狀態的位元ist本地訊息 (ist的產生在IEEE 488.1中允許裝置自行定義，但在IEEE 488.2裡卻有明

確的規範和感測位元S之極性一致時令該裝置所分配到之DIO線的電壓準位為”L”，如表2-2所示。

表2-1　PPR訊息

感測位元S	P3	P2	P1	PPR訊息	PPE訊息
0	0	0	0	PPR1	96
0	0	0	1	PPR2	97
0	0	1	0	PPR3	98
0	0	1	1	PPR4	99
0	1	0	0	PPR5	100
0	1	0	1	PPR6	101
0	1	1	0	PPR7	102
0	1	1	1	PPR8	103
1	0	0	0	PPR1	104
1	0	0	1	PPR2	105
1	0	1	0	PPR3	106
1	0	1	1	PPR4	107
1	1	0	0	PPR5	108
1	1	0	1	PPR6	109
1	1	1	0	PPR7	110
1	1	1	1	PPR8	111

至於感測位元的用法，舉例來說，假設某一計數器在並行輪詢組態時 (PPE介面訊息) 設定其S位元值為0，且為PPR1 (使用DIO線1) ，那麼此計數器可以在平常時設其ist值為1，等到其計數器達到預設值時，ist值轉變為0，因此

在接受並行輪詢時令DIO1線之電壓為“L”，以通知控制器其計數器值已達預設值了。

表2-2　ist本地訊息

ist	S	PPR訊息值
0	0	真
0	1	否
1	0	否
1	1	真

(3) PPD：並行輪詢失能命令，作用為取消分配給被選址之裝置的資料線，此即為解除遠程構建的方法，此外，也可使用通用命令PPU，將所有匯流排上之具有遠程構建能力的裝置全部解除。

(4) UNL：解除遠程構建狀態。

第三章 GPIB命令函式說明

要實現本介面卡具有控制器功能，須提供一函式庫供使用者能方便正確地控制符合IEEE 488.1及IEEE488.2的各類儀器，並支援Windows95/98/Me/NT4.0/2000/XP作業系統，此外提供NI-like之驅動程式函式集，方便使用者應用，所具備的函式庫可歸納四大類，分述如下：

＜１＞ 開啟：提供2個函式，用來新增／關閉裝置ID，供所有函式使用。

1. ibdev：開啟及初始化GPIB裝置
2. ibon1：將裝置上線或則下線

＜２＞ 結構：提供5個函式，用來增對裝置的各個參數作細部設定。

1. ibpad：設定裝置的主要位址
2. ibsad：設定裝置的次要位址
3. ibtmo：設定裝置的逾時時間
4. ibeos：設定裝置的命令結束字元模式
5. ibeot：設定裝置在I/O讀寫時是否自動地讓EOI線動作

＜３＞ 讀寫：提供2個函式，用來寫入命令至裝置及從裝置中讀取資料

1. ibwrt：寫入命令資料於裝置

2. ibrd：從裝置中讀取資料

＜4＞ 操作：提供九個函式，用來實現IEEE 488所提供GPIB介面機能，包括裝置清除、觸發、循序輪詢及並列輪詢等操作機能。

1. ibln：檢查裝置是否在GPIB滙流排線上(online)

2. ibclr：清除指定位址的裝置

3. ibtrg：觸發指定位址的裝置

4. ibloc：回到本地操作狀態

5. ibpct：將控制權移轉給另一具有控制器能力的裝置

6. ibrsp：執行循序輪詢操作

7. ibppc：建構並列輪詢

8. ibrpp：執行並列輪詢操作

9. ibwait：等待GPIB事件發生

3.1 函式說明

所有函式依特性可分成四大類共19函式：

開啟：

開啟新裝置　　　　ibdev

上線／下線	ibonl
開閉介面	ibclose

結構：

設定主要位址	ibpad
設定次要位址	ibsad
設定逾時時間	ibeot
設定結束字元模式	ibeos
致能／失能EOI線	ibeot

讀寫：

讀取資料	ibrd
寫入命令	ibwrt

操作：

檢查裝置是否上線	ibln
清除裝置	ibclr
觸發裝置	ibtrg
返回本地操作	ibloc
控制權移轉	ibpct
循序輪序	ibrsp
建構並列輪詢	ibppc

並列輪詢	ibrpp
等待事件成立	ibwait

ibdev

int ibdev(int board ID， int pad， int sad， int two， int eot inteos);

功用

開啟及初始化裝置，回傳一ID碼供後續程式使用。

參數

輸入

名稱	型別	說明
Board ID	整數	存取介面卡編號，目前只支援一塊介面卡，故此欄位保留，機定值為0。
Pad	整數	裝置主要位址，值0~31，詳細參閱ibpad函式說明。
Sad	整數	裝置之次要位址，值0x60 ~ 0x7E，詳細參閱ibsad函式說明，常用值為0。
Tmo	整數	裝置操作逾時時間，輸入數值為時間代碼，詳細參閱ibtmo函式說明，常用值設T3S。
Eot	整數	EOI線模式設定，0表EOI線不動作，1表EOI線動作，詳細參閱ibeot函式說明，常用值設1。
eos	整數	結束字元模式及EOS字元設定，詳細參閱ibeos函式說明，常用值設0。

傳回值

型別	說明
整數	裝置ID碼，開啟成功傳回正確碼，-1為失敗碼。

範例:

//開啟及初始化裝置,回傳”dev0”ID碼

```
dev0 = ibdev(0,22,0,T3s,1,0);
if ( dev < 0 )
   printf ("Device Open Error");
```

int ibonl (int ud， int v) ;

功用

致能/失能裝置，亦即使裝置ID碼有效/無效

參數

輸入

名稱	型別	說明
ud	整數	裝置ID碼
V	整數	用來決定裝置致能/失能 V=1致能裝置 V=0失能裝置

傳回值

型別	說明
整數	狀態變數ibsta值。

範例：

```
//開啟及初始化裝置,回傳"dev0"ID碼
 dev0 = ibdev(0,22,0,T3s,1,0);
 if ( dev < 0 )
     printf ("Device Open Error");
//"dev0"裝置失能
ibonl(dev0,0);
```

int ibpad (int ud， int v);

功用

改變裝置的主要位址，GPIB的定址模式，可分主要位址及次要位址兩種，一般常用為主要位址，可設定的範圍為0~30共31種變化，代表GPIB位址為0到31。

參數

輸入

名稱	型別	說明
ud	整數	裝置ID碼
V	整數	GPIB主要位址之值，範圍0~30

傳回值

型別	說明
整數	狀態變數ibsta值。

範例：

```
dev0 = ibdev(0,22,0,T3s,1,0);
if ( dev < 0 )
    printf ("Device Open Error");
ibpad(dev0,3); /*設定主要位址為3*/
```

int ibsad (int ud， int v);

功用

改變裝置的次要位址，值範圍為0x60~0x7E，其中0x60代表次要位址0，0x7E代表次要位址30，依以類推。若為0表忽略次要位址設定。

參數

輸入

名稱	型別	說明
ud	整數	裝置ID碼
V	整數	GPIB次要位址值，0表忽略次要位址，值範圍為0x60 ~ 0x7E

傳回值

型別	說明
整數	狀態變數ibsta值。

範例：

```
dev0 = ibdev(0,22,0,T3s,1,0);
if ( dev < 0 )
    printf ("Device Open Error");
ibpad(dev0,0x62); /* 設定次要位址為 3*/
```

int ibtmo (int ud，int v);

功用

設定裝置之逾時時間

參數

輸入

名稱	型別	說明
ud	整數	裝置ID碼
V	整數	逾時時間週期設定簡碼，相對應的時間如表3-1

傳回值

型別	說明
整數	狀態變數ibsta值。

範例：

```
dev0 = ibdev(0,22,0,T3s,1,0);
if ( dev < 0 )
  printf ("Device Open Error");

ibtmo(dev0,T10s); /*設定逾時時間為 10 秒*/
```

常數	V值	逾時時間
TNONE	0	無逾時時間
T10us	1	10us
T30us	2	30us
T100us	3	100us
T300us	4	300us
T1ms	5	1ms
T3ms	6	3ms
T10ms	7	10ms
T30ms	8	30ms
T100ms	9	100ms
T300ms	10	300ms
T1s	11	1s
T3s	12	3s
T10s	13	10s
T30s	14	30s
T100s	15	100s
T300s	16	300s
T1000s	17	1000s

表3-1

int ibeos (int ud，int v);

功用

設定字串結束字元(EOS)之值與模式，V值為0表EOS模式失能無效，值不為於0代表意義如下：

低位元組　EOS字元

高位元組　EOS模式設定，其中

Bit 2為1表示讀入資料字串等於EOS字元時會產生END結束旗號，表示已讀入完整的字串。反之Bit 2為0讀入資料時忽略EOS字元值。

Bit 3為1表示在寫出字串時，當寫出字元等於EOS字元時會立即跟隨送出EOI信號。反之，Bit 3為0，寫出字元等於EOS字元時，EOI線不會動作。

Bit 4為1表示，比較EOS字元全部八個bit，反之，Bit 4為0比較EOS字元較低7的位元。

參數

輸入

名稱	型別	說明
ud	整數	裝置ID碼
V	整數	EOS模式和字元設定，0表失能EOS模式和字元。

輸出]

型別	說明
整數	狀態變數ibsta值。

範例：

```
dev0 = ibdev(0,22,0,T3s,1,0);
if ( dev < 0 )
  printf ("Device Open Error");

 ibeos（dev0，0x140D）; /*EOS字元為0x0D(retum
                      碼)，比較EOS字元八位元，
                      且在讀入資料時，符合EOS字
                      元，表示讀入完整字串*/
```

int ibeot (int ud， int v);

功用

GPIB讀/寫操作時，V=1(致能)或V=0(失能)GPIB EOI 線自動動作。

參數

輸入

名稱	型別	說明
ud	整數	裝置ID碼
V	整數	V=1，致能EOI線自動動作 V=0，失能EOI線自動動作

傳回值

型別	說明
整數	狀態變數ibsta值。

範例：

```
dev0 = ibdev(0,22,0,T3s,1,0);
if ( dev < 0 )
  printf ("Device Open Error");

ibeot(dev0,0); /*設定失能 EOI 線自動動作*/
```

int ibwrt (int ud， void * wrtbuf， long count);

功用

寫入控制命令予所指定的裝置，ud為所指定的裝置，wrtbuf指向欲寫出之字串起始位址，忽略參數三count值，亦即本函式會將wrtbuf指標所指向之字串全部送出，實際筆數回傳至全域變數ibcntl中。

參數

輸入

名稱	型別	說明
ud	整數	裝置ID碼
wrtbuf	指標位址	欲寫出命令字串之起始位址
count	長整數	忽略。實際寫出之筆數，由wrtbuf所指之字串長度決定。

傳回值

型別	說明
整數	狀態變數ibsta值。

範例：

//開啟及初始化裝置,回傳"dev0"ID碼

```
dev0 = ibdev(0,22,0,T3s,1,0);
if ( dev < 0 )
  printf ("Device Open Error");
```

//寫入命令"*idn?"(裝置鑑別命令)於裝置 lcr 上,注意欄位三之值(35)可為任意值，系統忽略此欄位

```
ibwrt(lcr,"*idn?",35);
if ( ibsta & ERR )
     printf(lcr,"Device Write Error");
```

int ibrd (int ud， void * rdbuf， long count);

功用

從裝置中讀入資料於rdbuf所指向之字串緩衝器(buffer)中，其中ud為指定的裝置ID碼，忽略參數三，亦即會讀入完整的字串(根據END條件)，實際筆數回傳至全域變數ibcntl中。

參數

輸入

名稱	型別	說明
ud	整數	裝置ID碼
rdbuf	指標位址	指向欲讀入資料存放之字串緩衝器之起始位址
count	長整數	忽略。實際讀入之筆數，由END旗號決定。

傳回值

型別	說明
整數	狀態變數ibsta值。

範例：

```
dev0 = ibdev(0,22,0,T3s,1,0);
if ( dev < 0 )
printf ("Device Open Error");

ibwrt(lcr,"*idn?",35);
if ( ibsta & ERR )
            printf(lcr,"Device Write Errorr");
//存入資料存於 parser_buf 字串中，忽略參數三之值
 ibrd(lcr,parser_buf,40);
if ( ibsta & ERR )
   printf(lcr,"Device Read Error");
else
{      //根據回傳筆數 ibcntl，放入結束字元
      parser_buf[ibcntl-1]=0;
       printf("GPIB read(%d) : %s\n\r",ibcntl,parser_buf);     }
```

int ibclr (int ud);

功用

清除所指定的裝置，ibclr命令實現IEEE 488標準裏之SDC介面訊息機能，詳細參閱2.2節說明。

參數

輸入

名稱	型別	說明
ud	整數	裝置ID碼

傳回值

型別	說明
整數	狀態變數ibsta值。

範例：

```
dev0 = ibdev(0,22,0,T3s,1,0);
if ( dev < 0 )
  printf ("Device Open Error");

//清除裝置
ibclr(dev0);
```

int ibtrg (int ud);

功用

觸發所指定的裝置。ibtrg實現IEEE 488標準裏之GET

介面訊息機能，詳細參閱2.2節說明。

參數

輸入

名稱	型別	說明
ud	整數	裝置ID碼

傳回值

型別	說明
整數	狀態變數ibsta值。

範例：

```
dev0 = ibdev(0,22,0,T3s,1,0);
if ( dev < 0 )
  printf ("Device Open Error");

//觸發裝置
ibtrg(dev0);
```

int ibloc (int ud)；

功用

回復至本地操作狀態，亦即儀器面板按鍵可動作。

參數

輸入

名稱	型別	說明
ud	整數	裝置ID碼

傳回值

型別	說明
整數	狀態變數ibsta值。

範例：

```
dev0 = ibdev(0,22,0,T3s,1,0);
if ( dev < 0 )
  printf ("Device Open Error");

ibloc(dev0);
```

int ibln (int ud，int pad，int sad，shrot *listen);

功用

檢查裝置是否上線，裝置的指定是根據主要位址pad與次要位址sad而定，結果存於*listen變數上，值1表上線(存在於GPIB匯流排上)，0表裝置未上線。

參數

輸入

名稱	型別	說明
ud	整數	裝置ID碼，本版本忽略此欄位
pad	整數	裝置主要位址，設定值參照ibpad函式說明
sad	整數	裝置次要位址，設定方式參照ibsad函式說明。
listen	指標位址	指示裝置是否上線，1表上線，0表未上線

傳回值

型別	說明
整數	狀態變數ibsta值。

範例：

```
dev0 = ibdev(0,22,0,T3s,1,0);
if ( dev < 0 )
  printf ("Device Open Error");

ibln(dev0,3,0,&status); /*檢查裝置位址 3 使否上線*/
printf(“Staus = %d “,status);
```

int ibrsp (int ud，char * spr);

功用

執行循序輪詢，對象為所指定的裝置ud，狀態位元組資料則回應於spr所指之位元組，如果*spr值之bit 6為1表裝置要求服務。

參數

輸入

名稱	型別	說明
ud	整數	裝置ID碼
spr	指標位址	指向循序輪序回應位元組之位址

傳回值

型別	說明
整數	狀態變數ibsta值。

範例：

```
dev0 = ibdev(0,22,0,T3s,1,0);
if ( dev < 0 )
    printf ("Device Open Error");
```

//寫入命令”*sre 16”　，此命令為 IEEE488.2 必備命令之一，

設定裝置若輸出佇列有資料即產生序列輪詢(srq 動作)

```
ibwrt(dev0,"*sre 16",35);
if ( ibsta & ERR )
    printf("Device Write Error");
```

//寫入命令”*idn?”於裝置，裝置會回傳資料(裝置輸出佇列

含有資料)

```
ibwrt(dev0,"*idn?",35);
if ( ibsta & ERR )
    printf("Device Write Error");
```

//等待事件 RQS 或逾時(TIMO)成立

```
ibwait(dev0,RQS|TIMO);
   printf("ibsta = %x\n\r",ibsta);
```

//執方序行輪向，注意 bit6 是否為 1

```
ibrsp(lcr,&spr);
     printf("spr =%x,ibsta=%x\n\r",spr,ibsta);
```

//讀取裝置回傳資料，存於 parser_buf 字串中

```
ibrd(lcr,parser_buf,10);
if ( ibsta & ERR )
    report_errorr(lcr,"Device Read Error");
else
{
     parser_buf[ibcntl-1]=0;
     printf("GPIB read(%d) : %s\n\r",ibcntl,parser_buf);     }
```

//寫入命令”*sre 0”於裝置，清除裝置輸出佇列有資料時產生序列輪詢要求服務

```
ibwrt(lcr,"*sre 0",35);
if ( ibsta & ERR )
        printf("Device Write Error");
```

ibppc

int ibppc (int ud，int v);

功用

構建並列輪詢，指定裝置ud採用何條資料線來作為狀態傳回值。參數v用來設定致能(v不等於0)或失能(v=0)並列輪詢構建。致能並列輪詢構建，合法的v值為96到126之間，詳細的定義如表2-1所述。

參數

輸入

名稱	型別	說明
ud	整數	裝置ID碼
v	整數	設定致能/失能並列輪詢構建值，v=0表失能，致能之值如表2-1所述

傳回值

型別	說明
整數	狀態變數ibsta值。

範例：

```
dev0 = ibdev(0,22,0,T3s,1,0);
if ( dev < 0 )
      printf ("Device Open Error");
ibppc(dev0,96);//構建並列輪詢,值 96 表使用 DIO0
```

ibrpp

int ibrpp (int ud，char * ppr);

功用

執行並列輪詢，並列輪詢狀態值存於ppr所指向位元組中。注意的是ppr之資料所表示的是所有的並列輪詢資訊，每一位元代表某一裝置的狀態，1表ist=1，0表ist=0，至於那一位元代表那一裝置，由ibppc函式來構建，詳細請參閱第2.3節說明。

參數

輸入

名稱	型別	說明
ud	整數	裝置ID碼
ppr	指標位址	指向並列輪詢回應之狀態位元組位址

傳回值

型別	說明
整數	狀態變數ibsta值。

範例：

```
dev0 = ibdev(0,22,0,T3s,1,0);
if ( dev < 0 )
```

```
 printf ("Device Open Error");
 ibppc(dev0,96);
```

//執行並列輪詢，判別值 spr 之 bit 0，即為 ist 值

```
  ibrpp(dev0,&ppr);
   printf("PPR =%x\n\r",ppr);
```

int ibwait (int ud，int mask);

功用

等待GPIB事件成立。GPIB事件定義於mask中，其中mask=0，ibwait函式不等待直接更新GPIB事件狀態於ibsta全域變數中。其他值定義於下表中

代碼	位元位置	十六進制值	說明
TIMO	14	0 x 4000	逾時
END	13	0 x 2000	介面卡接到END或EOS訊息
RQS	11	0 x 800	裝置要求服務
CMPL	8	0 x 100	I/O動作完成

表3-2

參數

名稱	型別	說明
ud	整數	裝置ID碼
mask	整數	GPIB事件狀態之遮幕值，詳細定義如表3-2所示

傳回值

型別	說明
整數	狀態變數ibsta值。

範例：

```
dev0 = ibdev(0,22,0,T3s,1,0);
if ( dev < 0 )
      printf ("Device Open Error");
.
.
.
//等待事件 RQS 或逾時(TIMO)成立
ibwait(dev0,RQS|TIMO);
```

int ibpct (int ud);

功用

控制權轉移，將控制權轉移至具有控制器能力的裝置ud，注意執行此函式時所指定的裝置up須具備有控制器的能力，執行完後本介面卡進入idle狀態。

參數

輸入

名稱	型別	說明
ud	整數	裝置ID碼，欲移轉的裝置能力，須具備有控制器能力

傳回值

型別	說明
整數	狀態變數ibsta值。

範例：

```
 dev0 = ibdev(0,22,0,T3s,1,0);
if ( dev < 0 )
      printf ("Device Open Error");

ibpct(dev0);
```

int ibclose (void);

功用

應用程式執行結束時，最好呼叫此函數以便能順利釋放控制卡使用權。

參數

無輸入

傳回值

型別	說明
整數	狀態變數ibsta值。

範例：

```
dev0 = ibdev(0,22,0,T3s,1,0);
if ( dev < 0 )
      printf ("Device Open Error");
.
.
.
/ *程式最後結束時*/
  ibclose();
```

第四章　　應用程式介面說明

1、 Visual C++介面

- 驅動程式存放於\ATIGPIB\Driver\VC 目錄下。
- 將目錄下 vcgpib.LIB 及 atigpib.h 檔案複製於所工作之目錄。
- 應用程式將上述兩檔案連結於計劃(Project)中即可。
- 可參考\ATIGPIB\Driver\VC\testvc.DSW 範例。

2、 Borland C++ Builder 介面

- 驅動程式存於\ATIGPIB\Driver\BC 目錄下。
- 將目錄下之 bcgpib.LIB 及 atigpib.h 兩檔案複製於所工作之目錄。
- 應用程式將上述兩檔案連結於計劃(Project)中即可。
- 可參考\ATIGPIB\Driver\BC\bctest.bpr 範例。

3、 Labwindows CVI 介面

- 驅動程式存放於\ATIGPIB\Driver\LW 目錄下。
- 將目錄下之 lwgpib.LIB 及 atigpib.h 兩檔案複製於所工作之目錄。
- 應用程式將上述兩檔案連結於計劃(Project)中即可。
- 可參考\ATIGPIB\Driver\LW\lwtest.prj 範例。

4． Borland Delphi 介面

- 驅動程式存於\ATIGPIB\Driver\Delphi 目錄下。
- 將目錄下之 dpgpib.pas 檔案複製於所工作之目錄。
- 將目錄下之 ati48832.dll 複製於 Windows 系統目錄下(設定程式已經自動複製完成)
- 應用程式須將 dpgpib.pas 檔案連結於計劃中即可。
- 可參考\ATIGPIB\Driver\Delphi\dptest.dpr 範例。

5． Visual Basic 介面

- 驅動程式存於\ATIGPIB\Driver\VB 目錄下。
- 將目錄下之 gpib-32.bas 及 global.bas 兩檔案複製於所工作之目錄。
- 將目錄下之 ati48832.dll 複製於 Windows 系統目錄下(設定程式已經自動複製完成)
- 應用程式須將 gpib-32.bas 及 global.bas 兩檔案連結於計劃中即可。
- 可參考\ATIGPIB\Driver\VB \vbtest.vbp 範例。

6． Labview 5.0 介面

- 驅動程式存於\ATIGPIB\Driver\LV50 目錄下。
- 將目錄下之 lvgpib50.llb 檔案複製於所工作之目錄。

- 將目錄下之 lvgpib.dll 複製於 Windows 系統目錄下(設定程式已經自動複製完成)
- 可參考\ATIGPIB\Driver\LV50 \amidagpib.vi 範例。

7． Labview 6.0 介面

- 驅動程式存於\ATIGPIB\Driver\LV60 目錄下。
- 將目錄下之 lvgpib60.llb 檔案複製於所工作之目錄。
- 將目錄下之 lvgpib.dll 複製於 Windows 系統目錄下(設定程式已經自動複製完成)
- 可參考\ATIGPIB\Driver\LV60 \amidagpib.vi 範例。