中文全功能繪圖介面程式庫

林立域・董俊良

摘 要

在今天國內的應用軟體，都是來自國外的英文軟體，雖然國內有許多軟體從業人員，利用各種方法將這些英文軟體改成可以出現中文字的”中文軟體”，但是仍有許多缺點：1.智慧財產權的問題；2.會喪失某些原有功能；3.亦無法達到在中文環境下使用的特殊功能，如繪圖與畫面編排等。因此，本人在兼顧前面三種情況下，利用介面軟體方式來解決這個問題，使得英文軟體能達到眞正本土化或中文化的目的。

一、前 言

以往的Microsoft BASIC 都有提供不少繪圖指令，但因受硬體設備之限制，以致於在單色螢幕上無法發揮其繪圖能力，我們很欣然地發現Quick BASIC 4.0支援了單色繪圖卡 (Hercules Card)之繪圖功能，但是此版本也產生了另一個問題，那就是與中文系統無法完全配合。例如，文數字的顯示緩慢，無法直接使用BASIC之繪圖指令畫圖等。

雖然寫程式或繪圖並不一定要使用中文系統，但是對大多數國人而言，中文仍然是彼此溝通的最佳工具，尤其在教學方面。經過研究結果發現QuickBASIC 4.0爲求程式執行迅速，採用直接將輸出入系統(BASIC INPUT OUTPUT SYSTEM簡稱BIOS)來作輸出工作。而中文系統則是利用BIOS來作爲輸出資料的管道。雖然中文系統會每隔一段時間去作一次資料比對工作，但是如此文字的顯示已變爲較爲緩慢，甚至造成無法顯示的弊端。要解決此問題，我採用系統呼叫的方式來強迫QuickBASIC 4.0將輸出資料送往BIOS處理。在此僅以倚天中文爲例[3]，將上述方法書寫成程式[6]。

不過有一點需要特別注意的是QuickBASIC 4.0推出兩個新的系統呼叫指令INTERRUPT及INTERRUPTX來取代較早版本的INT86及INT86X，同時它爲了能

相容以前書寫之程式，亦可使用INT86OLD及INT86XOLD 指令來作系統呼叫工作。以書寫此程式之經驗，我們曾先後嘗試了INTERRUPT、INTERRUPTX及INT86OLD、INT86XOLD兩組指令來呼叫BIOS之螢幕處理常式及DOS服務常式[2]，最後我們發現INT86OKD及INT86XOKD使用上較爲穩定，而INTERRUPT及INTERRUPTX有時會莫明奇妙的當機。

QuickBASIC 中文全功能繪圖介面程式庫(QBETS)，是針對倚天中文系統所提供的繪圖和特殊功能，而專門爲QuickBASIC程式語言所設計的一套程式庫(Library)。

QBETS程式庫提供了39個的函數功能，其中包括了

1.處理中文碼的功能

2.中文繪圖功能

3.鍵盤輸入功能

4.直接取得中文系統的訊息

5.處理文字畫面的功能

QBETS 程式庫中所有關於倚天中文系統的函數，均以『ET』爲函數名稱的開頭，其函數名稱和倚天資訊股份有限公司所發行的『倚天中文繪圖介面程式庫』系列所使用的名稱相同。因此，您只要熟悉本程式庫，將來對於所有同一系列的倚天中文繪圖介面程式庫都能駕輕就熟，不需要因程式語言的不同而重複去學習不同的介面程式庫。

二、磁片中的內容

在您的QBETS磁片中應其有下面的檔案：

QBETS.BI	繪圖介面程式庫的引含檔(INCLUDE FILE)
QBETS.QLB	常駐式的繪圖介面程式庫
QBETS.LIB	外部連結(LINK)的繪圖介面程式庫
ETTEXT.BI	關於本文模式的引含檔(INCLUDE FILE)
ETGRAPH.BI	關於繪圖模式的引含檔(INCLUDE FILE)
SAMPLE	存放範例程式的子目錄

三、如何安裝和使用QBETS程式庫

QBETS 中文繪圖介面程式庫是專爲QuickBASIC 4.X版本所設計的，因此依照QuickBASIC的特性就有兩種不同的使用方式：

首先，請將下列的程式拷貝到您的QuickBASIC系統磁片中，或是將它們放到您啓動QuickBASIC系統的目錄中。

QBETS.BI

QBETS.QLB

QBETS.LIB

ETTEXT.BI

ETGRAPH.BI

3-1 直接使用QBETS.QLB

您可以在每次啓動QuickBASIC系統時，同時載入QBETS.QLB 這個常駐式的程式庫。您只要在DOS系統下鍵入QB/L QBETS.QLB即可。

```
C>QB/L QBETS.QLB <enter>
```

這樣的話，您就可以直接在QuickBASIC的操作環境下，使用全部的QBETS所提供的函數功能。

3-2 產生一個可執行檔(.EXE)

您也許有一些程式，已經在QuickBASIC的操作環境下執行過，而且没有任何的錯誤(Bug)存在。這時候，您希望透過QBETS去產生這個程式的可執行檔 (因爲在您的程式中有使用到QBETS程式庫的功能)。您可以使用下面的方式：

```
C>BC TEST;
C>LINK TEST+QBETS.LIB,TEST.EXE,NUK,;
```

在上面的例子中，我們先使用BC去編譯(Compiler)TEST.BAS這個程式，得到TEST.OBJ檔後，再用連結程式(LINK)將TEST.OBJ和QBETS.LIB 連接起來，產生TEST.EXE這個可執行檔。

四、使用格式的說明

關於使用QBETS程式庫的函數，有下面幾點的說明：

1.使用QBETS程式庫時，必須在程式中指定載入QBETS.BI這個引含檔。

2.您可以使用下面的兩種方式去呼叫一個函數，

```
CALL etCheck
```

或

```
etCheck
```

3.所有的參數(Parameter)的傳遞都必須遵守原函數的定義，並且不可以省略其中部份參數的傳遞。

例如：您想畫出一條從座標(10，10) 到(200，100)的實線時，必須寫成

```
CALL etLine(10，10，200，100，1，""，&HFFFF)
```

不可以寫成

```
CALL etLine(10，10，200，100)
```

五、重要名詞解釋

輸入碼及內碼

倚天中文系統自1.50版起提供多內碼功能，爲方便使用者又多提供了『輸入碼』。『輸入碼』是指凡是透過BIOS的輸出動作(如：INT10H及 INT17H) 所使用的內碼系統。一般而言，您應將內碼及輸入碼設定成相同，但是在一些特殊的情況下您很可能需要將兩者設定成不相同的兩種碼。

例如：您使用一部具有中文字形的中文印表機，而該印表機提供之中文須使用BIG-5 碼，現在您有一份中文資料要列印，該資料是使用IBM 5550碼爲內碼。在以前您可能捨棄印表機之中文功能或是使用轉碼程式將資料檔轉換成BIG-5 碼爲內碼，因此中文系統的內碼須設爲IBM 5550碼，而印表機須使用BIG-5 碼，因此必須將輸入碼設定成BIG-5 碼，接著就可以進行列表動作了。待列表完畢，最好將內碼及輸入碼設定成原來的樣子。

ASCII 碼及掃描碼

在PC上，每一次的按鍵動作都會產生兩個碼，鍵盤在接受操作者的按鍵動作後，先送出一個ASCII碼，等到操作者放開按鍵後，再送出一個掃描碼(SCAN CODE)。

在很多情況下，您必須利用掃描碼去判斷鍵盤動作。例如：您想在程式執行時，偵測F1至F2其中某個功能鍵是否被按下，或是想判別某些組合鍵是否被按下等等，這時候就必須使用掃描碼，而不能使用ASCII碼了。

六、QBETS 函數功能說明

etCheck	檢查倚天中文是否存在記憶體中
etCircle	繪製一圓
etCls	清除螢光幕
etCodeIGet%	取得目前的中文輸入碼
etCodeISet	設定新的中文輸入碼
etCodeXGet%	取得目前的中文內碼
etCodeXSet	設定新的中文內碼
etCrtLGet%	取得目前的顯示識別碼
etCrtLSet	設定新的顯示識別碼
etCshGet	取得目前的游標大小
etCshSet	設定新的游標大小
etCspGet	取得目前游標位置
etCspSet	設定新的游標位置
etCtl	設定中文鍵盤及顯示幕的模式
etExist%	檢查倚天中文是否存在記憶體中
etGet	取得螢幕上的影像資料
etGrephlnq	取得螢幕繪圖模式下的資料
etKeyGet&	取得輸入鍵的資料
etKeyStatus%	取得移位鍵的狀態

etLine	繪製線段
etModify	轉換影像資料
etPaint	在螢幕上指定的位置著色
etPoint%	讀取指定位置點的資料
etPreset	消除一點
etPrint	透過BIOS顯示資料
etPrtLGet%	取得目前的印表機識別碼
etPrtLSet	設定新的印表機識別碼
etPset	繪製一點
etPut	將影像資料顯示在螢幕上
etScrDn	將螢幕往下捲動
etScrUp	將螢幕往上捲動
etTxetInq	取得螢幕本文模式下的資料
etVer%	取得倚天中文的版本號碼
etVPortClr	清除視窗中的內容
etVPortSet	設定視窗的範圍
etWrtChar	顯示帶有屬性的字元
etWrtFont	顯示特殊字型
etWrtStr	顯示帶有屬性的字串
Pause	設定延遲時間

etCheck

〔功能〕 檢查倚天中文系統是否已經載入。

〔格式〕 CALL etCheck

〔傳回值〕如果沒有載入倚天中文系統，系統將傳回找不到中文系統的訊息 "ET Chinese system not found!"

〔說明〕 如果已經載入倚天中文系統，系統將自動轉換到中文系統下的英數輸入模式

參考指令：etExist

etCircle

〔功能〕 繪製一圓、圓弧或一橢圓。

〔格式〕 CALL etCircle(X%，Y%，Radius%，Color%，Start!，End!，Aspect!)

X%，Y%	中心點座標
Radius%	半徑
Color%	顏色代碼
Start!	起始角度
End!	終止角度
Aspect!	縱橫比

〔傳回值〕 無

〔說明〕 1.在單色顯示器中，顏色代碼只能夠爲0或1。

2.Start!和End!分別爲田圓弧的起始角度和終止角度，並以角度(Degree)爲單位，範圍在0.0度到360.0度。若兩者均爲正值，則只畫出一圓弧；若是Start!或是End!爲負值，則除了畫出一圓弧外，並在標示負值的起始角度或終止角度與圓心之間加畫一直線。

3.Aspect!爲 ’縱橫比’，即Y軸長度除以X軸長度的比例，用來指定橢圓的形狀。當縱橫比<1時，將畫出一壓扁(橫向)的橢圓；當縱橫比>1時，將畫出拉長(縱向)的橢圓。

4.本函數的功能和BASIC中的CIRCLE指令相似。

etCls

〔功能〕 清除螢光幕，並將游標設定在螢幕左上角(1，1)的位置。

〔格式〕 CALL etCls

〔傳回值〕無

參考指令：etScrUp，etScrDn，etVportClr

etCodeIGet

〔功能〕 取得目前中文系統所使用的輸入碼。

〔格式〕 PRINT etCodeIGet%

〔傳回值〕傳回目前中文系統所使用的輸入碼，傳回值爲一整數。

〔說明〕 1.輸入碼的含意和用途，請參考前面『輸入碼和內碼』之說明，或是參考倚天中文系統使用手册第二章第四章的說明。

2.下表所列爲傳回值所代表的意義：

傳回值	意義
0	BIG-5 碼
1	通用碼(國家標準交換碼)
2	公會碼
3	倚天碼
4	王安碼
5	IBM 5550 碼

參考指令： etCodeISet，etCodeXGet，etCodeXSet

etCodeISet

〔功能〕 重新設定目前中文系統所使用的輸入碼。

〔格式〕 CALL etCodeISet(State%)

State% 新的輸入碼

〔傳回值〕無

〔說明〕 1.輸入碼的含意和用途，請參考前面『輸入碼和內碼』之說明，或

是參考倚天中文系統使用手册第二章第四章的説明。

2.下表所列爲傳回值所代表的意義：

傳回值	意義
0	BIG-5 碼
1	通用碼(國家標準交換碼)
2	公會碼
3	倚天碼
4	王安碼
5	IBM 5550 碼

參考指令： etCodelSet，etCodelGet，etCodeXSet

etCodelSet

〔功能〕 重新設定目前中文系統所使用的內碼。

〔格式〕 CALL etCodeXSet(State%)

State% 新的內碼

〔傳回值〕無

〔説明〕 1.內碼的含意和用途，請參考前面『輸入碼和內碼』之説明，或是參考倚天中文系統使用手册第二章第四章的説明。

2.下表所列爲內碼所代表的意義：

傳回值	意義
0	BIG-5 碼
1	通用碼(國家標準交換碼)
2	公會碼
3	倚天碼
4	王安碼
5	IBM 5550 碼

參考指令： etCodelGet，etCodelSet，etCodeXGet

etCrtLGet

〔功能〕 取得目前的，顯示識別碼。

〔格式〕 PRINT etCrtLGet%

〔傳回值〕本函數傳回目前的顯示識別碼，傳回值爲一整數。

參考指令：etCrtLSet

etCrtLSet

〔功能〕 重新設定顯示識別碼。

〔格式〕 CALL etCrtLSet(Value%)

Value% 新的顯示識別碼。

〔傳回值〕 無

〔說明〕 1.Value%爲欲設定顯示識別碼之ASCII碼。

例如： 重新設定顯示識別碼爲"|"

CALL etCrtLSet(124)

2.低於ET 1.60版的中文系統無法使用本函數。

參考指令：etCrtLGet

etCshGet

〔功能〕 取得目前游標的形狀及大小。

〔格式〕 CALL etCshGet(StartLine%,EndLine%)

StartLine% 游標起始線

EndLine% 游標終止線

〔傳回值〕無

〔說明〕 1.StartLine% 傳回目前游標起始線的設定值，EndLine% 傳回目前游標終止線的設定值。

2.有關游標起始線及游標終止線之含意，請參考etCshSet函數的說明。

參考指令：

etCshSet

etCshSet

〔功能〕 重新設定目前游標的形狀及大小。

〔格式〕 CALL etCshSet(StartLine%,EndLine%)

StartLine% 游標起始線

EndLine% 游標終止線

〔傳回值〕無

〔說明〕 1.StartLine%傳回目前游標起始線的設定值，EndLine%傳回目前游標終止線的設定值。

2.在中文畫面下，游標終止線的設定值中文系統並不使用，此時游標大小由游標起始線的設定值所控制。游標起始線表示游標的頂端，因此當游標起始值爲0時，游標的形狀最大。在24X24中文下，若游標起始線之值爲27或更大，則游標會消失；在其餘的中文版本下，若游標起始線之值爲15或更大，則游標會消失。

3.雖然中文系統並不理會游標終止線，但是爲了避免當您使用Ctrl-Alt-A鍵切換到純英文模式時，會使游標變成兩半，所以您應該讓游標起始線之值小於或是等於游標終止線之值。同時，若是游標起始線之值大於或是等於13，則在純英文的模式下游標會消失。

4.當您在倚天中文系統下使用QuickBASIC 4.X時，會發現游標在程式執行後往往會消失不見。這時候，可在程式中加上本函數再將游標重現。

參考指令：etCshGet

etCspGet

〔功能〕 取得目前游標的位置。

〔格式〕 CALL etCspGet(Row%,Cloumn%)

Row% 傳回的列座標

Column% 傳回的行座標

〔傳回值〕無

參考指令：etCspSet

etCspSet

〔功能〕 重新設定目前游標的位置。

〔格式〕 CALL etCspSet (Row%,Cloumn%)

Row% 指定的列座標

Column% 指定的行座標

〔傳回值〕 如果設定的行數或列數超出最大行數或最大列數，將傳回一錯誤訊息。

〔說明〕 1.行數和列數的設定，應遵守下面的規則：

$1 \leqq$ Row% $\leqq 24$

$1 \leqq$ Column% $\leqq 80$

2.本函數的功能和BASIC中的LOCATE命令相似。

參考指考： etCspGet

etCtl

〔功能〕 指定控制字元用以設定螢幕及鍵盤之模式。

〔格式〕 CALL etCtl(Option$)

Option$ 指定的控制字元

〔傳回值〕 無

〔說明〕 1.有關控制字元及其功能簡列於下表：

控制字元	切換動作及功能	同義鍵
'Q'	釋放 ET.COM	
'q'	釋放 ET.COM	
'A	純英文顯示	
'a'	純英文／中文顯示切換	Alt-Ctrl-A
'N'	英數輸入	
'n'	英數輸入	
'C'	中文輸入	
'c'	中文／英數輸入切換	Ctrl-Esc
'F'	全形輸入	
'f'	全形／半形輸入切換	Shift-Esc
'1'-'9'	選擇中文輸入法	Alt-Ctrl-1-9
'0'	內碼輸入	Alt-Ctrl-0
'-'	詞庫輸入	Alt-Ctrl- -

控制字元	切換動作及功能	同義鍵
'L'	不解釋顯示，印表機控制字元	
'l'	解釋／不解釋顯示，印表機控制字元	Alt-Ctrl-L
'T'	中式游標移動	
't'	中式／英式游標移動	Alt-Ctrl-T
'H'	顯示內碼	
'h'	顯示內碼／字形切換	Alt-Ctrl-H
'G'	圖形上捲	
'g'	圖形上捲／不上捲切換	Alt-Ctrl-G
'M'	不顯示狀態行	
'm'	顯示／不顯示狀態行切換	Shift-Esc

參考指令：etPreset,etDrawDot

etExist

〔功能〕 檢查倚天中文系統是否已經常駐在記憶體中。

〔格式〕 PRINT etExist%

〔傳回值〕如果已經載入倚天中文系統，本函數將傳回1，否則傳回 0。

〔說明〕 當您在中文系統的輸入模式下，可以按下Alt-Ctrl-A切換到純英文 (ASCII)的模式，這時候中文系統仍存在記憶體中，因此您還是可以使用本函數去檢查倚天中文系統是否已經常駐在記憶體中。

參考指令：etCheck

etGet

〔功能〕 將矩形區域中的圖形儲存到指定的陣列中。

〔格式〕 CALL etGet (X1%,Y1%,X2%,Y2%,Array)

X1%,Y1%矩形區域中左上角的座標點

X2%,Y2% 矩形區域中右下角的座標點

Array() 儲存圖形的陣列

〔傳回值〕無

〔說明〕 1.計自儲存圖形陣列之大小的公式如下：

陣列大小＝((X%＋7)＼8) *Y%／2

X%：矩形區域的寬度

Y%：矩形區域的高度

參考指令：

etGraphInq

〔功能〕 取得目前螢幕繪圖模式下的參數。

〔格式〕 CALL dtGraphInq (etGraphInfo)

etGraphInfo 爲一已定義好的記錄(Record)

〔傳回值〕本函數將以etGraphInfo 這個記錄傳回目前螢幕繪圖模式下的參數。

〔說明〕 etGrphInfo這個記錄已定義於etGraph.bi檔中，其代表意義分述如下：

```
TYPE etGraphInfo
   Rwidth AS INTEGER       螢幕實際寬度
   Rheight AS INTEGER      螢幕實際高度
   Px AS INTEGER           主畫面左上角的水平座標
   Py AS INTEGER           主畫面左上角的垂直座標
   Pwidth AS INTEGER       主畫面寬度
   Pheighy AS INTEGER      主畫面高度
   Cw AS INTEGER           ASCII 字元寬度
   Ch AS INTEGER           ASCII 字元高度
   Vx1 AS INTEGER          繪圖視窗左上角的水平座標
   Vy1 AS INTEGER          繪圖視窗左上角的垂直座標
   Vx2 AS INTEGER          繪圖視窗右下角的水平座標
   Vy2 AS INTEGER          繪圖視窗右下角的垂直座標
   Lx AS INTEGER           最後參考點的水平座標
   Ly AS INTEGER           最後參考點的垂直座標
   Pflag AS INTEGER        繪圖系統旗標
   MaxColor AS INTEGER     最大顏色值
   Color0 AS INTEGER       背景色
   Color1 AS INTEGER       前景色
   Reserved AS INTEGER     保留
END TYPE
```

參考指令：etGraphIng

etKeyGet

取得按鍵的ASCII碼及掃描碼。

Code& ＝etKeyGet&

傳回按鍵的ASCII 碼及掃描碼。

1.有關ASCII 碼及掃描碼之區別，請參考前面〝重要名詞解釋〞中的說明。

2.在BASIC中的INKEY$ 僅能取得按鍵的ASCII 碼，因此無無法判斷是否有功能鍵或特殊鍵被按下，本函數連同掃描碼一併取得，因此可判斷功能鍵或特殊鍵是否被按下。

3.掃描碼的對照表詳列於附錄中。

參考指令：etKeyStatus

etKeyStatus

〔功能〕 取得鍵盤上特殊鍵的移位狀態。

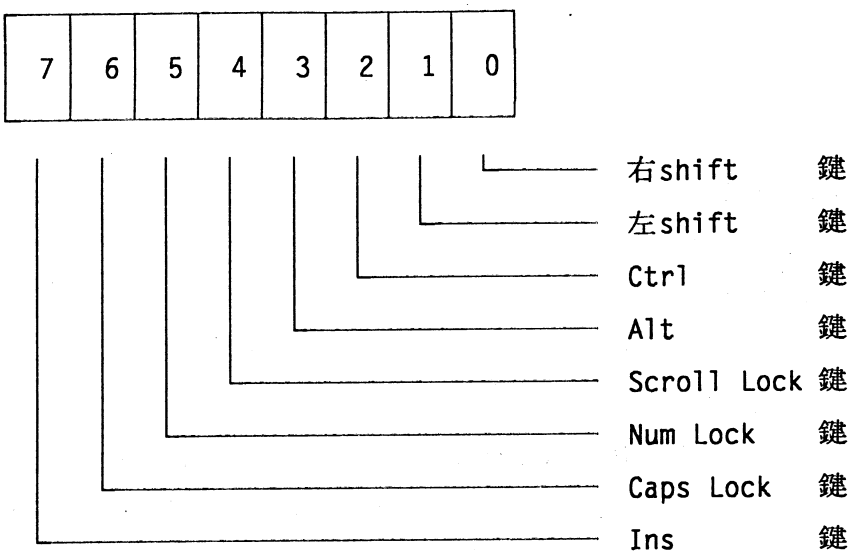
〔格式〕 Status%＝etKeyStatus%

〔傳回值〕無

〔說明〕 1.特殊鍵包括左Shift，右Shift，Ctrl，Alt，Sciroll，Lock，Num Lock，Caps Lock，及Ins等鍵。

2.上述八個鍵各由一個位元表示，若其中任一鍵被按下則其所屬之位元將爲1，否則爲0 。

它們的位元配置如下表所示：



參考指令：etKeyGet

etLine

〔功能〕 繪製一線段或矩型方塊。

〔格式〕 CALL etLine (X1%, Y1%, Y2%, Color%, State$, Style!)

X1%, Y1% 起始點座標

X2%, Y2% 終止點座標

Color% 顏色代碼

State% 狀態字串

Style! 線段樣式

〔傳回值〕無

〔說明〕 1.在單色顯示器中，顏色代碼只能夠爲0或1。

2.狀態字串定義如下：

空字串("")由(X1%, Y1%)至(2%, Y2%) 繪出一線段。

"B" or "b" 以(X1%, Y1%)和(2%, Y2%)兩個點爲對角線，畫出一長方形。

"B" or "bf" 以(X1%, Y1%)和(2%, Y2%) 兩個點爲對角線，畫

出一實心方塊。

3.線段樣式是由一組16位元的數值所定義，每個位元代表線段內的一個點。若位元值爲1 ，則該對應點會繪出；若位元值爲0 ，則該對應點不會繪出。此樣式值將反覆使用，直到整條線段繪完爲止。

例如：若線段樣式值設定爲&HE724，則繪出的線段將如下圖所示：

■■■□□■■■□□■□□■□□ &HE724

4.本函數的功能和BASIC中的LINE指令相似。

etModify

〔功能〕 將etGet 函數所儲存於陣列的圖形資料，進行放大、縮小、反相、旋轉或上下顛倒處理。

〔格式〕 CALL etModify (State%, Source(), Target(), Xlen%, Ylen%)

State% 運算狀態

Source() 儲存原始圖形的陣列

Target() 儲存目的圖形的陣列

Xlen% 目的圖形的寬度

Ylen% 目的圖形的高度

〔傳回值〕 無

〔說明〕 1.選擇圖形處理功能由參數State%決定，詳細說明如下：

State%	功能
0	旋轉 0 度
1	旋轉90 度
2	旋轉180度
3	旋轉270度
4	上下顛倒
5	黑白反相

上述的運算狀態可爲複合的方式，只要將狀態值相加即可。例如：希望將原始圖形旋轉270 度並作黑白反相處理，則運算狀態State%＝11

2. Source()爲原始圖形之儲存陣列；Target()爲經過處理後的圖形之儲存陣列，計算陣列大小的方式和etGet 函數相同。

   陣列大小＝（(X%＋7)＼8）*Y%／2

   X%：矩形區域的寬度

   Y%：矩形區域的高度

3. Xlen% 和Ylen% 分別爲目的圖形的寬度和高度，可和原始圖形的寬度和高度不同。例如：要將目的圖形放大兩倍，則只要將原始圖形的寬度和高度乘以2 即可。

參考指令：etGet, etPut

etPaint

〔功能〕 在指定的封閉區域內塗滿指定的圖案。

〔格式〕 CALL etPaint (X%, Y%, Pattern%, Boundary%)

X%, Y%, 指定的著色點

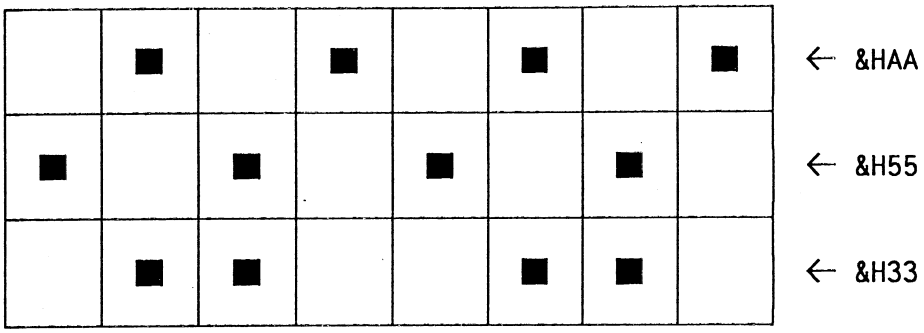
Pattern$ 圖案樣式字串

Boundary% 邊界顏色

〔傳回值〕無

〔說明〕 1.當您指定的區域不是一個封閉的區域時，則整個螢幕將被填滿。

2.著色的圖案樣式定義如下：



因此，Pattern$＝&HAA 55 33"

3.Bondary% 爲邊界條件，也就是塗色之終止區域的顏色。

4.本函數的功能類似BASIC 中的Paint指令。

etPaint

〔功能〕 讀取螢幕中指定座標點的顏色。

〔格式〕 Color% ＝etPoint (X%, Y%)

X%, Y%指定的座標點

〔傳回值〕傳回指定座標點的顏色代碼。

〔說明〕 1.本函數的功能類似於BASIC 中的POINT指令。

etPreset

〔功能〕 在指定的座標繪製一點。

〔格式〕 CALL etPreset (X%, Y%, Color%)

X%, Y% 指定的座標點

Color% 顏色代碼

〔傳回值〕無

〔說明〕 1.基本上，etPset和etPreset是兩個非常相似的函數，它們的功能都是在指定的座標上繪製一點，您也可以指定顏色代碼去產生一黑色的點，（也就是消除一指定點）。但是在一般的情況下，我們常會不去指定顏色代碼，使用etPset去繪製一點，而使用etPreset去消除一點。

2.在單色顯示器中，顏色代碼只能夠爲0或1。

參考指令：etPset, etDrawDot

etPaint

〔功能〕 在螢幕上顯示一指定的字串，加上顯示控制碼則可以顯示24×24字型及放大字型。

〔格式〕 Call etPrint (Strng$)

Strng$指定的字串

〔傳回值〕無

〔說明〕 1.本函數顯示的速度較原來QuickBASIC中的Print 指令快。

2.詳細的字型顯示控制方式，請參考倚天中文系統使用手册第七章第一節。

參考指令：etWrtStr

etPrtLGet

〔功能〕 取得目前設定的印表識別碼。

〔格式〕 PRINT etPrtLGet%

〔傳回值〕本函數傳回目前的印表識別碼，傳回值爲一整數。

參考指令：etPrtLSet

etPrtLSet

〔功能〕 重新設定印表識別碼。

〔格式〕 CALL etPrtLSet (Value%)

Value% 新的印表識別碼

〔傳回值〕本函數傳回目前的印表識別碼，傳回值爲一整數。

〔說明〕 Value% 爲欲設定印表識別碼之ASCII碼。

例：重新設定印表識別碼爲"!"

CALL etPrtLSet (33)

參考指令：etPrtLGet

etPset

〔功能〕 在指定的座標繪製一點。

〔格式〕 CALL etPset (X%, Y%, Color%)

X%, Y% 指定的座標點

Color% 顏色代碼

〔傳回值〕無

〔說明〕 1.基本上，etPset和etPreset是兩個非常相似的函數，它們的功能都是在指定的座標上繪製一點，您也可以指定顏色代碼去產生一黑色的點，（也就是消除一指定點）。但是在一般的情況下，我們通常不指，定顏色代碼，使用etPset去繪製一點，而使用etPreset去消除一點。

2.在單色顯示器中，顏色代碼只能夠爲0或1。

參考指令：etPreset

etPut

〔功能〕 將儲存的圖形陣列重新顯示在螢幕上。

〔格式〕 CALL etPut (X%, Y%, Array(), Option$)

X%, Y% 指定的座標點

Array() 儲存圖形的陣列

Option& 運算狀態

〔傳回值〕無

〔說明〕 1.計算儲存圖形陣列之大小的公式如下：

陣列大小＝ ((X%＋7)＼8) *Y%／2

X%：矩形區域的寬度

Y%：矩形區域的高度

2.運算狀態表示圖形顯示的方式，有以下五種：

PSET將儲存在陣列中的圖形，直接以點對點的方式顯示在螢幕上。

PRESET將儲存在陣列中的圖形，將每個點的資料經過反相後，再以點對點的方式顯示在螢幕上。

AND 將儲存在陣列中的圖形，與螢幕上現有的影像作AND 運算後，再顯示到螢幕上。

OR將儲存在陣列中的圖形，與螢幕上現有的影像作OR運算後，再顯示到螢幕上。

XOR 將儲存在陣列中的圖形，與螢幕上現有的影像作XOR 運算後，再顯示到螢幕上。若是使用此方式將同一圖形顯示在同一地方兩次，螢幕上將恢復原來之畫面。

3.下面的範例程式，將教您如何把BASIC 語言所使用的BLK 格式的圖形檔顯示在螢幕上。在這裡我們所使用的是，松崗電腦圖書資料有限公司所發行的『造型銀行[illegible]』的圖形檔作爲範例。

參考指令：etGet, etModify

etScrUn

〔功能〕 定義一個文字視窗，並將視窗中的內容向上捲動。

〔格式〕 CALL etScrDn (Row1%, Column1%, Row2%, Column2%, Num%, Attr%)。

Row1% Column1% 文字視窗的左上角座標

Row2%, Column2% 文字視窗的右下角座標

Num% 捲動列數

Attr% 空白行的屬性

〔傳回值〕如果設定的行數或列數超出最大行數或最大列數，將傳回一錯誤訊息。

〔說明〕 1.當您定義一個文字視窗，並將視窗中的內容向下捲動時，視窗底部的內容會消失，並在視窗的頂部填入空。

2.Attr% 爲上述空白行的屬性，而非文字視窗中文字的屬性。

3.若捲動列數超過視窗大小，或是捲動列數設定爲0 ，則整個文字視窗中的內容將會被完全清除。

參考指令：etScrDn

etTextInq

〔功能〕 取得目前螢幕本文模式下的參數。

〔格式〕 CALL etTextInq (etTextInfo)

etTextInfo爲一已定義好的記錄(Record)

〔傳回值〕本函數將以etTextInfo這個記錄傳回目前螢幕本文模式下的參數。

〔說明〕 1.etTextInfo這個記錄已定義於etText.bi 檔中，其代表意義分述如下：

```
TYPE etTextInfo
     Column AS INTEGER          螢幕寬度（行數）
     Row AS INTEGER             螢幕高度（列數）
     Vram AS INTEGER            Video RAM 位址
     Nattr AS INTEGER * 1       正常屬性
     Iattr AS INTEGER * 1       高亮度
     NDattr AS INTEGER * 1      不顯示
     Uattr AS INTEGER * 1       加底線
     Rattr AS INTEGER * 1       反相
     Battr AS INTEGER * 1       閃爍
     Uiattr AS INTEGER * 1      高亮度、加底線
     BRattr AS INTEGER * 1      閃爍、反相
     BUattr AS INTEGER * 1      閃爍、加底線
     BIattr AS INTEGER * 1      高亮度、閃爍
     Reserced AS INTEGER * 1    保留
END TYPE
```

參考指令：etGraphInq

etVer

〔功能〕 取得目前所使用的中文系統之版本號碼。

〔格式〕 PRINT etVer%

〔傳回值〕本函數傳回目前所使用的中文系統之版本號碼，傳回值爲一整數。

參考指令：etExist%

etVPortClr

〔功能〕 清除視窗中的內容，並將最後的參考點位置重置於視窗之中心位置。

〔格式〕 CALL etVPortClr

〔傳回值〕無

〔說明〕 1.上面所說的視窗，是指以etVPortSet函數所建立之繪圖視窗，如果未曾建立繪圖視窗，則本函數會清除整個螢光幕。

2.執行本函數之後，最後的參考點爲螢幕中心或是視窗中心，實際位置則依照您使用etVPortSrt函數的設定而決定。本函數的詳細用法請參考etVPortSet函數之說明。

參考指令：etVPortSet

etVPortSet

〔功能〕 設定一繪圖視窗。

〔格式〕 CALL etVPortSet (State%, X1%, Y1%, X2%, Y2%)

(State% 視窗型態

X1%, Y1% 視窗左上角的座標

X2%, Y2% 視窗右下角的座標

〔傳回值〕無

〔說明〕 1.經由本函數設定一繪圖視窗後，凡是繪於視窗外的圖形將不會被顯示出來，只顯示繪於視窗之內的圖形。文字資料則不受此限制。

2.『視窗型態』之設定值定義如下：

State%	功 能
0	直接設定視窗範圍
1	設定視窗，並繪出邊框

3.視窗範圍的設定應以不超過螢幕正常顯示的範圍爲原則，而螢幕的顯示範圍則因不同的中文版本而有所差異。

	飛碟、光電、閃電系列 (16x16)	霹靂系列 (24x24)
寬	640點	960點
高	382點	702點

您也可以使用etGraphInq函數去取得這些資料，它們分別存放在Pwidth和Pheight 這兩個變數中。

參考指令：etVPorClr

etWrtChar

〔功能〕 在螢幕上指定的位置輸出一個或數個帶有屬性的字元。

〔格式〕 CALL etWrtChar (Row%, Colunm%, Char$, Count%, Attr%)

Row% 指定的行數

Colunm% 指定的列數

Char$ 輸出的字元

Count% 字元顯示的個數

Attr% 輸出字元的屬性

〔傳回值〕無

〔說明〕 1.行數和列數的設定，應遵守下面的規則：

1＜Row% ＜24

1＜Column% ＜80

2.字元的輸出位置並不受目前游標位置的影響。

3.如果只需要輸出一個字元，千萬要記得將Count%的值設定爲 1；若將Count%設定爲0 ，則將輸出無限個字元，也就是無止境的運作下去。

4.本函數執行後，會將游標移到寫出第一個字元的位置，也就是(Row%, Column%) 的位置上。

5.屬性的定義和其顯像的方式如下：

Attr%	顯像方式
&H1	底線
&H7	正常
&H8	高亮度
&H70	反白
&H80	閃爍

其中，高亮度(&H8)和閃爍(&H80)並不能單獨使用，必須配合&H1，&H7，&H70，才有作用。

例如：您想使用高亮度(&H8)加上底線(&H1)的字體，則Attr%＝&H9 (&H8＋&h1)。

參考指令：etWrtStr

etWrtStr

〔功能〕 在螢幕上指定的位置顯示一字串，並且設定字串的屬性。

〔格式〕 CALL etWrtStr (State%, Row%, Colunm%, Strng$, Attr%)

State% 輸出型態

Row% 指定的行數

Colunm% 指定的列數

Strng$ 輸出的字串

Attr% 輸出字串的屬性

〔傳回值〕如果設定的行數或列數超出最大行數或最大列數，本函數將傳回一錯誤訊息。

〔說明〕 1.行數和列數的設定，應遵守下面的規則：

1＜Row%＜24

1＜Column%＜80

2.State%可設定輸出型態：

State%	輸出型態
0	顯示前後，游標位置不動。
1	顯示後，游標位置移到字串後面。

3.如果輸出字串中含有特殊字元，則本函數會去解釋它們而不會將它們顯示出來。這些特殊字元包括：Back space (&H08)，Carriage return (&H0D), Line feed (&H0A), Bell (&H07), 和顯示識別碼。

4.如果您要在螢幕上顯示24X24 的字型，也可以使用下面的方式：

```
Strng$=CHR$(27)+"T24,10,10,,3,S; QuickBASIC 4.0"
```

但是，當您使用這種方式顯示24X24 的字型時，本函數中的指定輸出位置和文字屬性都將失效。

5.屬性的定義和其顯像的方式如下：

Attr%	顯像方式
&H1	底線
&H7	正常
&H8	高亮度
&H70	反白
&H80	閃爍

其中，高亮度(&H8)和閃爍(&H80)並不能單獨使用，必須配合&H1，&H7，&H70，才有作用。

例如：您想使用高亮度(&H8)加上底線(&H1)的字體，則Attr%＝&H9 (&H8＋&h1)。

參考指令：etPrint

etWrtFont

〔功能〕 在上螢幕上指定的位置顯示特殊字型。

〔格式〕 CALL etWrtFont (FontType%, X%, Y%, Pitch%, xScale%, yScale% Strng$)

FontType% 特殊字型的綿號 X%, Y% 指定的座標

Pitch% 字距

xScale% 字的寬度

yScale% 字的高度

Strng$ 欲顯示的字串

〔傳回值〕無

〔說明〕 1.FontType% 爲特殊字型的編號，其定義如下。

FontType%	字型定義
1	中空字
2	立體字
3	陰影字
4	美術字
5	花紋字
6	特殊字
7	木雕字
8	古體字

2.字距(Pitch%)爲字與字之間的距離，以點爲單位。

3.使用這個函數時，您的倚天中文系統最好有24X24 的字型這樣顯示出來的特殊字型會比較好看。

Pause

〔功能〕 設定延犀時間。

〔格式〕 CALL Pause (Delay Time!)

〔傳回值〕無

〔說明〕 1.Delay Time! 爲欲設定的延遲時間，以秒爲單位。

程式範例：

```
DECLARE SUB title (test!)
DECLARE SUB ccurve (x1%, x2%, y1%, y2%, sinth!, costh!, ressq!)
DECLARE SUB kochflake (cos30!, resolution!, reduuctionfactor!, x%,
            y%, r!)
DECLARE SUB sphere (xc!, yc!, r!)
DECLARE SUB picture ()
DECLARE SUB ellipse (xc!, yc!, r!, a!, b!)


* * * * * * * * * * * * * * * * * * * * * * * * * * *
*   程式範例：C-曲線、雪片、球體運動  D E M D   *
* * * * * * * * * * * * * * * * * * * * * * * * * * *
         CLEAR,,3000                    '重新設定堆疊(stack)的大小
CALL etcheck                            '檢查倚天中文系統是否已經載入
CALL etcls                              '清除螢幕
CALL etvportset (1, 1, 1, 638, 371)     '設定繪圖視窗
test = 1
CALL title (test)
        CONST pi = 3.141593
        theta = -45
```

```
        SP = 0: colors = 3
        sinth = SIN (theta * pi/180)
        costh = COS (theta * pi/180)
        cos30 = COS (30 * pi/180)
        resolution = 4
        ressq = resolution * resolution
        x1% = 200:y1% = 250: x2% = 400:y2% = 250
        LET xc = 300:yc = 150:r = 100
        DEF fnrad (degrees) = degrees * pi/180
'畫出 C- 曲線
CALL ccurve (x1%, x2%, y1%, y2%, sinth!, costh!, ressq!)
CALL pause (5!)                         '畫面暫停5秒鐘
CALL etvportclr                         '清除視窗畫面
CALL etvportclr
CALL etvportset (1, 1, 1, 638, 371)
        test = 2
CALL title (test)
        resloution = 3:reductionfactor = 4
        x% = 300:y% = 200:r = 70:col = 0
'畫出雪片
CALL kochflake (cos30, rrsolution, reductionfactor, x%, y%, r)
CALL pause (5!)
CALL etvportclr                         '清除視窗畫面
        test = 3
CALL title (test)
CALL sphere (xc, y, r)                  '模擬3度空間球體
CALL picture                            '繪出球體運動
CALL pause (5!)
CALL etcls                              '清除螢幕
CALL etctl ("a")                        '切換爲英文模式
```

```
CLEAR                                   清除所有的變數及堆疊內容
END

SUB ccurve (x1%, x2%, y1%, y2%, sinth, costh, ressq)
DIM stackx1 (20), stackx2 (20), stacky1 (20), stacky2 (20)
DIM stackmx (20), stackmy (20)
        xdiff = x2% - x1%: ydiff = y2% - y1%
IF xdiff * xdiff + ydiff * ydiff < = ressq THEN
        CALL etline (x1%, y1%, x2%, 1, "", &HFFFF)  '畫線
EXIT SUB
ELSE
END IF
        stackmx (SP) = mx: stackmy (SP) =my
        SP = SP + 1
        stackx1 (SP) = x1%: stacky1 (SP) = y1%
        stackx2 (SP) = x2%: stacky2 (SP) = y2%
        xdir = xdiff/2/costh: ydir = ydiff/2/costh
        mx = x1% + xdir * costh - ydir * sinth
        mx = y1% + xdir * sinth + ydir * costh
        x2% = mx: y2 = my
'遞迴副程式
CALL ccurve (x1%, x2%, y1%, y2%, sinth, costh, ressq)
        x1% = mx: y1% = my: x2% = stackx2 (SP):y2% = stacky2 (SP)
'遞迴副程式
CALL ccurve (x1%, x2%, y1%, y2%, sinth, costh, ressq)
        x1% = stackx1 (SP): y1% = stacky1 (SP)
        x2% = stackx2 (SP): y2% = stacky2 (SP)
        SP = SP - 1
        mx = stankmx (SP): my = stackmy (SP)
END SUB
```

```
SUB ellipse (xc, yc, r, a, b)
DIM xxc% (360), yyc% (360)
LET xca% =xc + a:yc% = yc
FOR theta = 0 TO 360 STEP 1
        cosine = COS (fnrad(theta))
        sine = Sin (fnrad(theta))
        r = 1/SQR(cosine * cosine/(a* a) + sine * sine/(b * b))
        x = r * cosine
        y = r * sine
        xxc%(theta) = INT(xc + x)
        yyc%(theta) = INT(yc + y)
        IF theat = 0 THEN
        CALL etline(xca%,yx%,xxc%(theta),yyc%(theta),1,"" ,&HFFFF)
        ELSE
CALL etline(xxc%(theta-1),yyc%(theta-1), xxc%(theta), yyc%(theta)
        ,1, "",&HFFFF)
        END IF
NEXT theta
END SUB
SUB kochflake (cos30, resolution, reductionfactor, x%, y%, r)
DIM stackx1(20), stackx2(20), stacky1(20), stacky2(20), stackr(20)
DIM stackx(20), stacky(20), stacky3(20(, stacky4(20), stackcol(20)
IF r < resolution THEN
        EXIT SUB
ELSE
ENS IF
        stackxl(SP) = x1%: stackx2(SP) = x2%
        stackyl(SP) = y1%: stacky2(SP) = y2%
        stacky3(SP) = y3%: stacky4(SP) = y4%
        SP = SP + 1
```

```
        x1% = x% - r * cos30:x2% =x% + r * cos30:xinc = r*cos30/3
        y1% = y% - r:y2% =y% -r / 2: y3% = y% + r / 2: y4% = y%+r
CALL etline(x1%, y2% ,x%, y4%,1,"",&HFFFF)
CALL etline(x%, y4% ,x2%, y2%,1,"",&HFFFF)
CALL etline(x2%, y2% ,x1%, y2%,1,"",&HFFFF)
CALL etline(x%, y1% ,x1%, y3%,1,"",&HFFFF)
CALL etline(x1%, y3% ,x2%, y3%,1,"",&HFFFF)
CALL etline(x2%, y3% ,x%, y1%,1,"",&HFFFF)
        stackr(SP) = r: stackcol(SP) = col
        stackx(SP) = x%:stacky(SP) = y %
        r = r * reductionfactor
        col = (col + 1) MOD 3
        x% = x1% : y% = y2%
' 遞迴副程式
CALL kochflake(cos30, resolution, reductionfactor, x%, y%, r)
        x% = stackx(SP):y% = y4%
' 遞迴副程式
CALL kochflake(cos30, resolution, reductionfactor, x%, y%, r)
        x% = x2%: y% = y2%
' 遞迴副程式
CALL kochflake(cos30, resolution, reductionfactor, x%, y%, r)
        x% = stackx(SP): y% = y1%
' 遞迴副程式
CALL kochflake(cos30, resolution, reductionfactor, x%, y%, r)
        x% = x1%: y% = y3%
' 遞迴副程式
CALL kochflake(cos30, resolution, reductionfactor, x%, y%, r)
        x% = x2%: y% = y3%
' 遞迴副程式
CALL kochflake(cos30, resolution, reductionfactor, x%, y%, r)
```

```
        x% = stackx(SP): y% = stacky(SP)
' 遞迴副程式
CALL kochflake(cos30, resolution, reductionfactor, x%, y%, r)
        r = stackr(SP): x% =stackx(SP): y% = stacky(SP)
        col = stackcol(SP)
        SP = SP - 1
        x1% = stackx1(SP): x2% = stackx2(SP)
        y1% = stacky1(SP): y2% = stacky2(SP)
        y3% = stacky3(SP): y4% = stacky4(SP)
END SUB
SUB picture
x1% = 180: x2% = 420: y1% = 40: y2% = 252
x3% = x2%  + 1: x4% = x2% - x1% + x3%
x% = x2% - x1%: y% = y2% - y1%
size = ((x% + 7) / 8 ) * y% / 2
xlen% = x%
ylen% = y%
DIM source(size), sourcea(size), sourceb(size), sourcec(size)
DIM targeta(size), targetb(size), targetc(size)
CALL etget(x1%, y1%,x2%, y2%, source())        '取得球體區域面積
'將球體旋轉 90 度
CALL etmodify(1, source(), targeta(), xlen%, ylen%)'
'將球體旋轉 180 度
CALL etmodify(2, source(), targetb(), xlen%, ylen%)'
'將球體旋轉 270 度
CALL etmodify(3, source(), targetc(), xlen%, ylen%)'
FOR xx% = x1% TO 1 STEP -5
CALL etput (xx%, y1%, source(),"pset") '將球體重新顯示在螢幕上
NEXT xx%
FOR yy% = 48 TO 130 STEP 8
```

```
CALL etput (1, yy%, source(), "pset")  '將球體重新顯示在螢幕上
NEXT yy%
yy%  = yy% -16
FOR xx% = 1 TO 500 STEP 2
i = i + 1
SELECT CASE i
CASE 1
CALL etput(xx%, yy%, targeta(), "pset") '將球體重新顯示在螢幕上
CASE 2
CALL etput(xx%, yy%, targetb(), "pset") '將球體重新顯示在螢幕上
CASE 3
CALL etput(xx%, yy%, targetc(), "pset") '將球體重新顯示在螢幕上
CASE ELSE
END SELECT
IF i = 3 THEN
i = 0
ELSE
END IF
NEXT xx%
END SUB
SUB sphere (xc, yc, r)
CALL etvportset(0, 1, 1, 638, 339)
LET b = r
FOR a = 1 TO b + 1 STEP b / 5
        CALL ellipse(xc, yc, r, a, b)
NEXT a
LET a = b + 1
FOR b = 1 TO a + 1 STEP a / 5
        CALL ellipse(xc, yc, r, a, b)
NEXT b
```

```
END SUB
SUB title (test)
'繪圖模式下寫出中文字
CALL etwrtstr(1, 2, 2, "靖宇研究發展中心QuickBASIC 4.X " + "",&H7)
CALL etwrtstr(1, 2, 34, "中文全功能繪圖界面程式庫" + "", &H7)
SELECT CASE test
CASE 1
CALL etwrtstr(1, 23, 30, "圖型 : C-曲線 (C-Curve)" + "", &H7)
CASE 2
CALL etwrtstr(1, 23, 30, "圖型 : 雪片 (Snowflakes)" + "" &H7)
CASE 3
CALL etwrtstr(1, 24, 30, "圖型 : 模擬 3 度空間球體 " + "" ,&H7)
CASE ELSE
END SELECT
END SUB
```

參考文獻

1.Microsoft Co. press, "Microsoft DOS 3.3 Reference Menu", (1988).

2.Microsoft Co. press, "Microsoft Quck BASIC 4.0 Reference Menu", (1989).

3.倚天資訊，〝倚天中文系統使用手冊〞，台北(1988).

4.林立域，〝Quick BASIC 4.X程式設計〞，松崗，台北(1989).

5.林立域，〝大家來學Quick BASIC 〞，靖宇，台北(1990).

6.林立域，〝Quick BASIC 高等應用技法〞，靖宇，台北(1990).

7.林立域，〝中文全功能編排繪圖介面程式庫〞靖宇研發中心，台中(1989)