凌阳科技

實驗指導書

SPCE3200 精簡開發板——eCos 平臺篇

（圖形介面實驗）

V1.0　2008-04-28

凌陽單晶片技術資料

http://www.unsp.com

版權聲明

前 言

本教材是結合 SPCE3200 教學實驗平臺而設計，與課堂教學內容結合緊密。本書主要針對嵌入式作業系統 eCos 的應用，其中安排的實驗均是 SPCE3200 教學實驗平臺在嵌入式作業系統 eCos 下的功能使用，且附有範例程式。實驗內容淺顯易懂，讀者通過這些實驗的學習，可以逐步掌握凌陽 SPCE3200 嵌入式教學平臺在嵌入式作業系統 eCos 下的編程方法。

實驗共分五部分，基本內容如下：

第一章：eCos 基礎實驗（主要內容爲 RedBoot 的使用，典型設備的使用，多線程操作，多線程同步，定時告警等實驗）；

第二章：圖形介面實驗（主要內容爲 eCos 下的 MicroWindow GUI 的基礎應用實驗）；

第三章：檔案系統和網路應用實驗（主要內容爲 eCos 下的檔案系統和網路協定棧的應用實驗）；

第四章：多媒體實驗（主要內容爲音頻播放、視頻採集、圖像編解碼等實驗）。

第五章：eCos 綜合實驗（主要內容爲應用 GUI、檔案系統及網路等元件，進行綜合實驗聯繫）。

本書爲第二部分圖形介面試驗。本書中的所有實驗範例代碼均經過除錯。實驗時按照硬體連接說明進行連接後，程式可直接下載運行，使讀者達到節省時間、快速入門的目的。

由於編者水準有限，編寫時間倉促，書中難免有所錯漏，敬請讀者和專家指正。

目 錄

Tel:010-62981113-2944/2940/2942/2945 http://www.unsp.com
E-mail:unsp@sunplus.com.cn

2 圖形介面實驗

实验十一 MicroWin 環境配置

【實驗目的】

1、 瞭解嵌入式 GUI——MicroWin 的體系架構和消息機制；

2、 掌握 MicroWin 的開發環境的建立。

【實驗設備】

1、 裝有 Windows 系統和 S+core IDE 集成開發環境的 PC 機一台；

2、 凌陽 SPCE3200 嵌入式精簡開發板一套；

3、 本實驗用到開發板的介面有： TFT LCD 液晶介面。

【實驗要求】

在 eCos 下編譯 MicroWin 並生成庫，編譯 MicroWin 的例副程式並燒錄到 SPCE3200 開發板上，觀察程式運行效果。

【實驗原理】

1、 MicroWin 的體系結構

MicroWin（以下簡稱 MicroWin）是著名的開放式源碼的嵌入式 GUI 軟體，可以運行在多種嵌入式作業系統或裸機下，eCos 系統完全支援 MicroWin。

MicroWin 使用分層機構的設計方法，允許改變不同的層來適應實際的應用。在最底一層，提供了螢幕、滑鼠、觸控屏和鍵盤等的驅動，使程式能存取實際的硬體設備和其他用戶定制設備。在中間一層，有一個輕巧的圖形引擎，提供了繪製線條，區域填充、繪製多邊形、裁剪和使用顏色样式的方法。在最上一層，提供了不同的 API 給圖形應用程式使用。這些 API 可以提供或不提供桌面和視窗外型。目前，MicroWin 支援 ECMA APIW 和 Nano-X API。這些 API 提供了 Win32 和 X 視窗系統的緊密相容性，使得別的應用程式可以很容易就能移植到 MicroWin 上。

1) 設備驅動層

設備驅動介面在 device.h 中定義。一個 MicroWin 程式通常有螢幕、滑鼠和鍵盤驅動。中間層的設備無關圖形引擎核心函數將直接調用設備驅動來執行硬體相關操作。這樣，我們在 MicroWin 中改變或者增加設備時不會影響整個系統的工作。

2) 設備無關的圖形引擎層

設備無關的圖形引擎層包含 MicroWin 的核心圖形函數，這些函數通過調用螢幕、滑鼠和鍵盤驅動程式來跟硬體打交道。用戶應用程式不直接調用核心圖形引擎函數。核心圖形引擎函數獨立于應用程式 API 有幾個原因。其一，在客戶/伺服器样式下，核心圖形函數常常駐留在伺服器上。其二，由於速度上的原因，核心圖形函數使用內部文本字體和位元映射格式，這與標準 API 資料結構有區別。其三，核心圖形函數常常使用指標，不使用標識。

在 MicroWin 中，核心圖形函數遵守 GdXXX 的命名方式，它們只關心圖形輸出，不負責視窗管理。另外，所有的裁剪和顏色轉換都在這一層處理。下面是 MicroWin 的核心模組的檔案組成：

devdraw.c： 核心圖形函數(線、圓、多邊形、文本、位元映射、顏色轉換)

devclip.c： 核心裁剪函數（devclip2.c 是新的 y-x-banding 運算法則、devclip1.c 是舊的）

devrgn.c： 相交/聯合/相減異或生成區域動態配置的函數

devmouse.c： 滑鼠指標更新和裁剪的核心函數

devkbd.c： 核心鍵盤處理函數

devpalX.c： 提供 1、2、4 和 8 位色調色板支援

3) 應用介面層

目前，MicroWin 提供兩種不同的應用程式介面——MicroWin API 和 Nano-X。兩種 API 都在核心圖形引擎層和設備驅動層之上運行。

- MicroWin API

MicroWin API 在設計上儘量順從歐洲 ECMA APIW 標準。目前，支援大部分的圖形繪製和裁剪工作、支援自動繪製視窗標題欄、支援視窗的拖動。MicroWin API 使用消息機制，允許編寫應用程式時無需考慮系統最終是否使用視窗管理器。MicroWin API 不符合客戶/伺服器样式。

- Nano-X API

Nano-X API 是模仿 David Bell 寫的 mini-x 伺服器開發出來的。它具有仿照 X 視窗系統的 Xlib API，但函數以 GrXXX 的方式命名，而不是以 X 爲首碼。Nano-X API 使用客戶/伺服器样式，但不提供視窗的自動裝飾、標題欄和用戶視窗移動。

MicroWin API 的根源程式是 mwin/win*.c，而 Nano-X API 的根源程式是 nanox/srv*.c。在 eCos 下我們選用 MicroWin API 進行圖形程式的開發。

2、 MicroWin 開發環境建立的步驟

MicroWin 需要依賴於 eCos 作業系統來運行，但同時又獨立於 eCos。所以，建立 MicroWin 在 eCos 下的圖形程式開發環境可以分爲兩個步驟：

1) 生成 eCos 庫

首先依次打開 eCos 源碼包中的檔案夾 “packages” → “hal” → “score” → “spce3200” → “current” → “src” ，再打開檔案 “lcd_support.c” ， 並做如下修改：

- 把 “TFT_VER_BACK” 的值修改爲 18；

Tel:010-62981113-2944/2940/2942/2945 http://www.unsp.com
E-mail:unsp@sunplus.com.cn

- 把“TFT_VER_SYNC”的值修改爲 1

然後打開eCos Config Tools使用net範本建立ecos工程，並做如下修改，如 圖 2.1和圖 2.2所示：

- 使能“I/O sub-system”→“Serial device drivers”→“Hardware serial device drivers”；
- 使能“I/O sub-system”→“Serial device drivers”→“Termios compatible TTY drivers”→“Termios TTY channel #0”；
- 將“Termios TTY channel #0 device”的值修改爲“/dev/serial”；
- 將“eCos HAL”→“SCORE architecture”→“Sunplus SPCE3200”→“LCD type”的值改成“SPCE3200_EM_BOARD”。

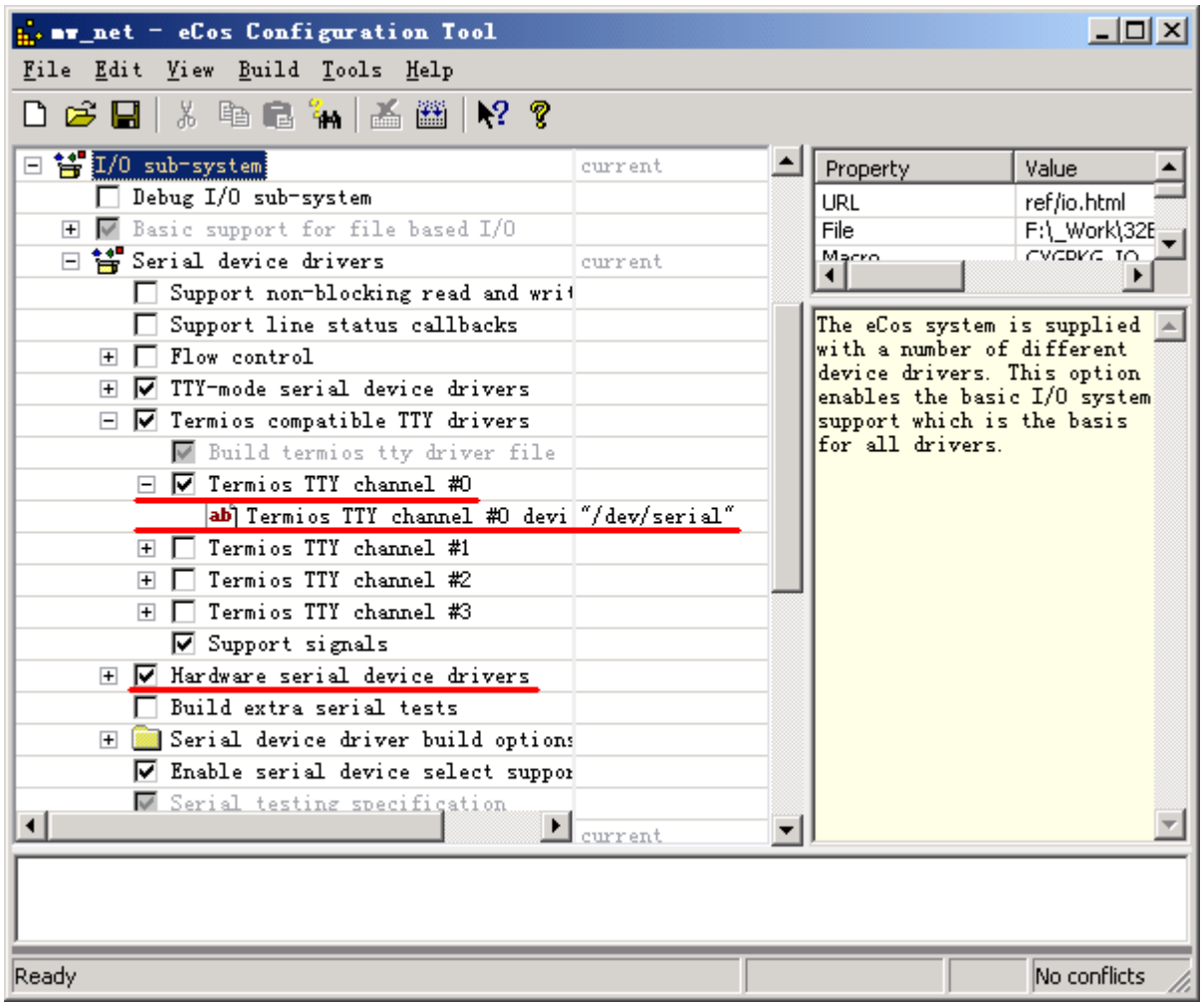

圖 2.1 eCos 配置 1

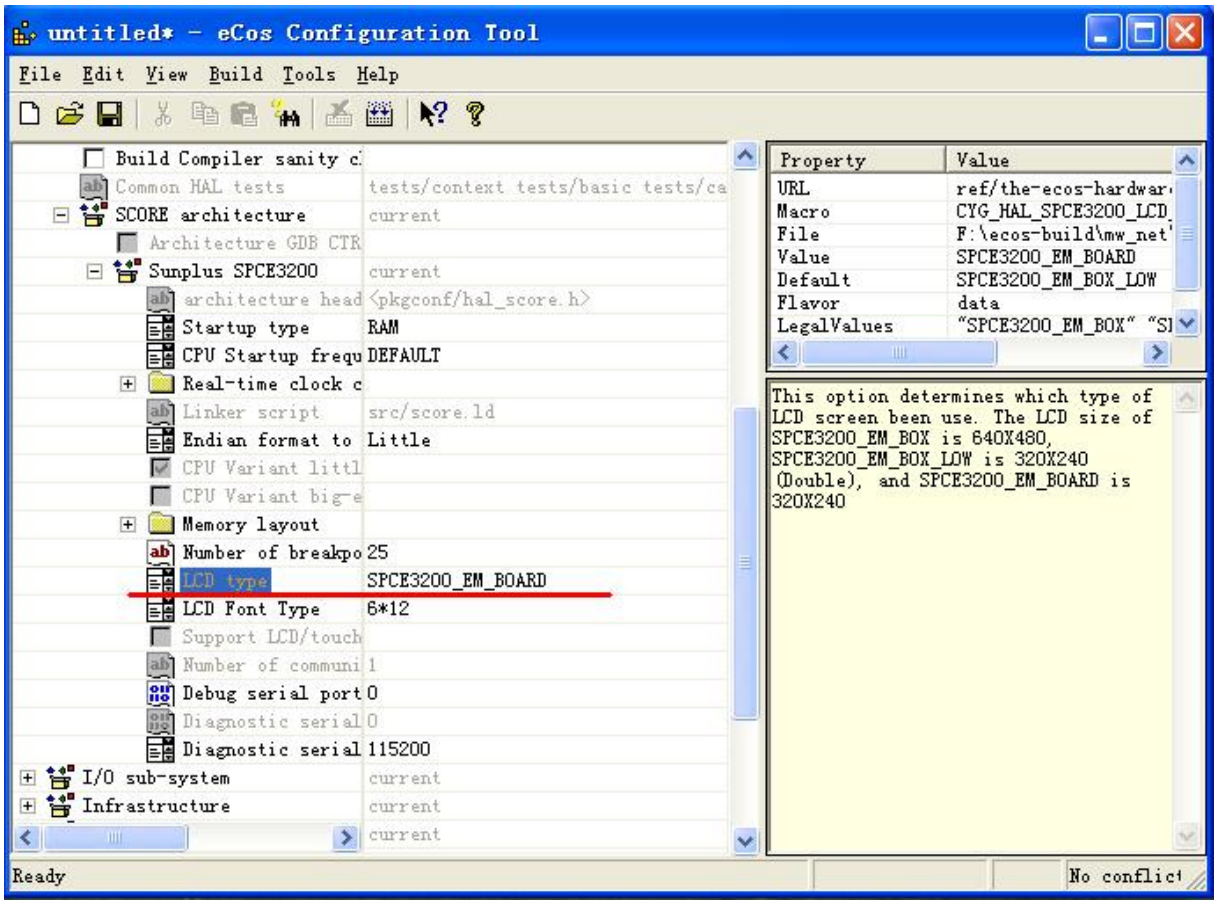

圖 2.2 eCos 配置 2

然後將此配置保存，例如保存到 f:\ecos-build\mw_net 下命名爲 mw_net.ecc（此路徑在後面編譯 MicroWin 時會使用到）。

最後按 F7 編譯，等待出現“build finished”編譯成功。

2) 生成 MicroWin 庫

打開“ecos源碼路徑\packages\services\MicroWin-0.91\src\Arch.rules”檔案，找到如 圖 2.3所示的代碼段，將常數ECOS_PREFIX 的值修改爲第一步操作中生成的eCos庫的安裝路徑（本例中爲/ecos-f/ecos-build/mw_net/mw_net_install）。

Tel:010-62981113-2944/2940/2942/2945 http://www.unsp.com
E-mail:unsp@sunplus.com.cn

```
ifeq ($(ARCH), ECOS)
    COMPILER = gcc
    CXX_COMPILER = g++
    TOOLSPREFIX = $(SCORETOOLSPREFIX)
    ECOS_PREFIX = /ecos-f/ecos-build/mw_net/mw_net_install
    INCLUDEDIRS += -I$(ECOS_PREFIX)/include
    DEFINES += -D__ECOS -DUNIX=1
    CFLAGS += -mel -Wall -ffunction-sections -fdata-sections
    ifeq ($(OPTIMIZE), Y)
        CFLAGS += -O2
    else
        CFLAGS += -O0
    endif
    ifeq ($(DEBUG), Y)
        CFLAGS += -g
    endif
    LDFLAGS += -nostdlib
endif
```

圖 2.3 修改 ECOS_PREFIX

在 cygwin 中使用“cd”命令修改當前工作目錄至“ecos 源碼路徑\packages\services\MicroWin-0.91\src\”下，然後執行“make”命令，即可開始編譯 MicroWin 庫。（注意：eCos 源碼的存放目錄最好不要包含空格和中文，否則編譯過程中可能會出現錯誤，如果是利用安裝在了 C 盤或 D 盤的 Program File 目錄下，則在“cd”進入“Program File”目錄時一定要使用“cd progra~1”這種格式進入該檔案夾，否則編譯會出錯）

等待編譯成功結束後，可以在“ecos 源碼路徑\packages\services\MicroWin- 0.91\src\lib”目錄下找到 MicroWin 的程式馆。

至此，MicroWin 的開發環境即建立完成。可以使用 MicroWin 的程式馆並配合“ecos 源碼路徑\packages\services\MicroWin-0.91\src\include”目錄下的标头档案來開發圖形程式。

3、 生成 MicroWin 例副程式

MicroWin 的源碼中提供了一些例副程式以便用戶可以快速的體驗或測試 MicroWin 系統。得到例副程式有兩種方法，下面將分別介紹。

1、 生成 MicroWin 的示例程式方法一

打開“ecos源碼路徑\packages\services\MicroWin-0.91\src\config”檔案，找到如 圖 2.4所示代碼段，並將MICROWINDEMO的值修改爲“Y”；

```
#############################################################################
#
# Demos to build
#
#############################################################################
MICROWINDEMO                = N
NANOXDEMO                   = N
```

圖 2.4 Demo 編譯選擇

在 cygwin 中使用“cd”命令修改當前工作目錄（注意不能有中文和空格）至“ecos 源碼路徑\packages\services\MicroWin- 0.91\src\”下，然後執行“make”命令，即可開始編譯 MicroWin 庫，同時將編譯 Demo 程式。

等待編譯成功結束後，可以在“ecos 源碼路徑\packages\services\MicroWin- 0.91\src\bin”目錄下找到 app_load.elf 檔案，該檔案即爲例副程式生成的可執行程式檔案。

2、 生成 MicroWin 的例副程式方法二

如果已經成功編譯 MicroWin 庫，可在 cygwin 下使用“cd”命令進入“ecos 源碼路徑\packages\services\MicroWin-0.91\src\demos\mwin\”目錄下，並執行“make”命令，等待編譯成功結束後，便可以在“ecos 源碼路徑\packages\services\MicroWin- 0.91\src\bin”目錄下找到 app_load.elf 檔案。

【實驗步驟】

1、 按照實驗原理中有關建立 MicroWin 開發環境的說明編譯 MicroWin 庫；

2、 任選生成 MicroWin 的例副程式的兩種方法的其中一種得到 app_load.elf 檔案；

3、 使用 RedBoot 將 app_load.elf 檔案下載到 SPCE3200 開發板上，實現脫機運行；

4、 觀察 MicroWin 例子的運行結果，並可通過觸摸 LCD 螢幕來操作這些例子。

Tel:010-62981113-2944/2940/2942/2945 http://www.unsp.com
E-mail:unsp@sunplus.com.cn

实验十二　窗口實驗

【實驗目的】

1、 掌握 IDE 下建立 MicroWin 應用程式的方法；

2、 瞭解 MicroWin 應用程式的框架；

3、 學會使用 MicroWin 建立一個簡單的應用程式。

【實驗設備】

1、 裝有 Windows 系統和 S+core IDE 集成開發環境的 PC 機一台；

2、 凌陽 SPCE3200 嵌入式精簡開發板一套；

3、 本實驗用到開發板的介面有：TFT LCD 液晶介面。

【實驗要求】

使用 MicroWin API 建立一個視窗，並在視窗中以不同的顏色、不同的背景色和不同的位置輸出若干字串。

【實驗原理】

1、 窗口

視窗是MicroWin系統最主要的內容，是應用程式與用戶之間交互的介面，也是MicroWin系統管理應用程式的基本單位。一個典型的MicroWin應用程式視窗的基本組成如 圖 2.5所示。

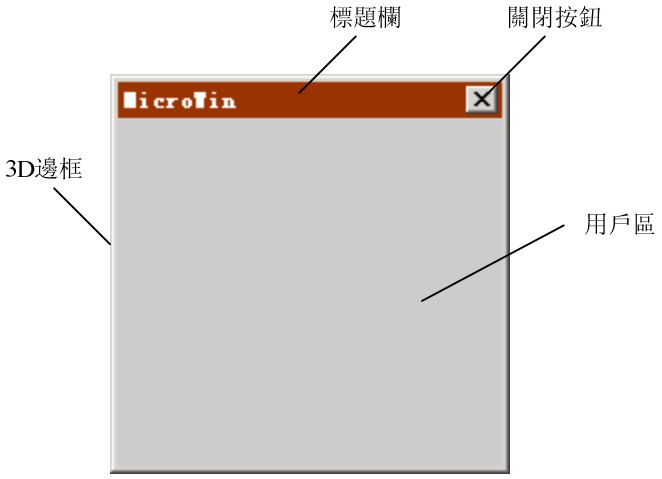

圖 2.5 MicroWin 窗口的基本組成

編寫一個 MicroWin 應用程式首先應創建一個或多個視窗，隨後應用程式的運行過程既是視窗內部、視窗與視窗之間、視窗與系統之間進行資料處理與資料交換的過程。

2、 MicroWin 的消息機制

MicroWin 系統是消息驅動系統，消息傳遞是 MicroWin 中最基本的通訊機制，任何應用程式的動作或者用戶操作都與某種消息相關聯。MicroWin 應用程式需要圍繞消息或事件來設計處理函數。所謂消息是描述事件發生的資訊。例如當按下觸控屏上的某一點時，系統就會發出一條特定的消息，標識滑鼠按鍵事件的發生。MicroWin 程式的執行順序取決於事件發生的順序，程式的執行是由順序產生的消息驅動的。程式師可以針對消息型別編寫相應的程式以處理接收的消息，或者發出其他消息以驅動其他程式。

MicroWin 系統中的一個消息包含有一個約定的消息號以及兩個參數：wParam 和 lParam。消息被存儲在應用程式的消息佇列中,應用程式通過調用函數 GetMessage()獲取佇列中的消息，從而得知系統發生的某些事件或得知用戶的某種操作,如視窗的創建與消除事件分別對應 WM_CREAT 和 WM_DESTROY 消息、WM_CHAR 消息代表用戶通過鍵盤輸入了字元、WM_LBUTTONDOWN 消息代表滑鼠左鍵被按下等等。在通常情況下,每個消息都對應於一個用視窗，在獲取消息後,應用程式通過調用 DispatchMessage 函數將消息分派到所對應的視窗進行處理。當視窗建立時,該視窗所對應的各種消息的處理函數（稱爲視窗函數）必須同時被定義,所以系統才能將消息傳遞給這個視窗函數來處理。

消息傳遞機制允許核心的 API 通過處理各種事件的消息來實現各種功能，如視窗的創建，繪製，移動等。通常情況下，MicroWin 系統提供一個統一的預設的消息處理函數 DefWindowsProc 來對大多數的消息進行預設的處理，這樣就使得所有視窗的動作在行爲上具有大部分的一致性，當某一視窗需要對某些消息進行特殊的操作時，用戶可以通過改寫處理程式來滿足要求。

MicroWin 的消息系統由 3 個部分組成：

1) 消息佇列

MicroWin 系統本身維護了一個系統消息佇列，應用程式可以從這個消息佇列中獲取消息，然後分派給對應的視窗。

2) 消息迴圈

MicroWin 應用程式中包含了一段稱作“消息迴圈（也稱消息泵）”的代碼，用來從消息佇列中檢索消息並把他們分發到相應的視窗函數中。正是這個消息迴圈使得一個應用程式能夠回應外部的各種事件，所以消息迴圈往往是一個 MicroWin 應用程式的核心部分。

3) 視窗函數

最終爲了處理各種消息，MicroWin 應用程式必須爲每個視窗定義一個相應的視窗函數。視窗函數從形式上看是一個 switch 語句結構，用以處理由消息迴圈發送到該視窗的各種消息。視窗函數是一種回調函數（Callback Function），也就是說，它是由 MicroWin 系統負責調用的，而應用程式本身不調用它。

MicroWin 系統中的消息從發生到被處理一般有 5 個步驟：

1) 系統發生了一個事件；

2) MicroWin 系統把事件翻譯爲對應的消息，並把它放到消息佇列中；

3) 應用程式從消息佇列中獲取消息，然後把它封裝在 MSG 結構中；

4) 應用程式通過消息迴圈把消息分派給對應的視窗函數；

5) 視窗函數負責最終處理這個消息。

MicroWin 的消息可分爲佇列消息(Queued Messages)和非佇列消息(Non-Queued Messages)兩種：

1) 佇列消息(Queued Messages)

佇列消息會先保存在消息佇列中，然後由消息迴圈從此佇列中取消息並分發到各視窗回調函數進行處理。佇列消息可以由 MicroWin 提供的 PostMessage 函數發出。PostMessage 函數在將消息填入消息佇列之後立即返回。

2) 非佇列消息(NonQueued Messages)

非佇列消息會繞過系統消息佇列直接發送到視窗函數被處理，此時，相當於直接調用了目標視窗的視窗函數。非佇列消息可以由 MicroWin 提供的 SendMessage 函數發出。SendMessage 函數將等待目標視窗處理完該消息後才返回。

消息分發流程如 圖 2.6所示，其中消息 1 爲非佇列消息，其直接送給視窗的回調函數並等待執行完畢後返回；消息 2 爲佇列消息，其產生後先被保存到系統的消息佇列中後返回，再由消息迴圈送給視窗函數。

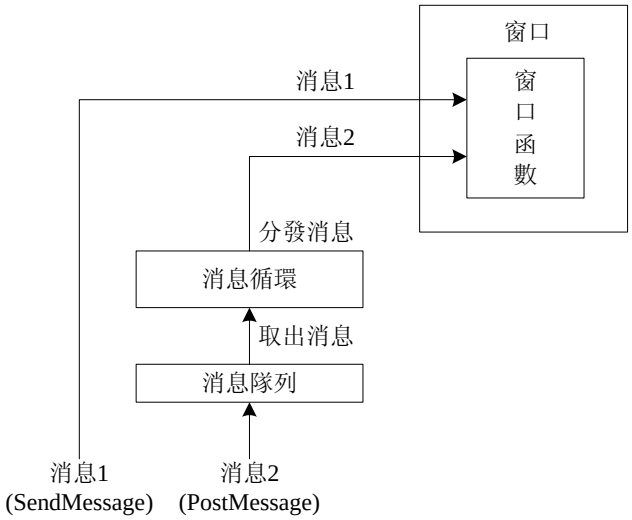

圖 2.6 消息分發流程

3、 MicroWin 程式框架

進行 MicroWin 程式設計，實際上是在進行一種物件導向的程式設計（OOP）。使用 MicroWin 編程與在 Window 下圖形編程非常類似，讀者可以找一些有關 Window 下圖形編程資料參考。

1) WinMain 函數

WinMain 函數是所有 MicroWin 應用程式的入口，類似於 C 語言中的 main 函數。WinMain 函數是 MicroWin 應用程式的起點，其功能是完成一系列的定義和初始化工作，並產生消息迴圈。

WinMain 函數是整個程式運行的核心，它實現以下功能：

- 註冊視窗類，建立視窗及執行其他必要的初始化工作；
- 進入消息迴圈，根據從消息佇列接收到的消息，調用相應的處理程式；
- 當消息迴圈檢索到 WM_QUIT 消息時終止程式運行。

WinMain 函數的完整形式如下：

```
int WINAPI WinMain(    HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine,
                       int nCmdShow)
```

其中，hInstance 和 hPrevInstance 是由系統傳遞的兩個實例控制碼值。實例控制碼在 MicroWin 系統中並不被支援，但是爲了相容 ECMA APIW，MicroWin 系統保留了實例控制碼的概念，通常實例控制碼始終爲 NULL。lpCmdLine 是一個字串指標，代表了應用程式的啓動參數，MicroWin 也並不支援該參數，該參數始終爲 NULL。nCmdShow 是由系統傳遞的用於控制顯示方式的值，應用程式應該根據這個值的狀態對主表單的顯示方式進行設置，在 MicroWin 系統中，該值始終爲 SW_SHOW，表示應用程式應該顯示它的表單。

2) 建立視窗

MicroWin 程式建立視窗的步驟爲，首先必須註冊一個視窗類，用來定義一類具有某些公共屬性的視窗。然後再用這個視窗類創建一個視窗的實例，並讓其顯示在顯示器上。

應用程式可以使用 RegisterClass 函數註冊一個視窗類。該函數的函數原型和功能描述如下：

【函數原型】ATOM WINAPI RegisterClass(CONST WNDCLASS *lpWndClass);

【功　　能】註冊一個視窗類

【參　　數】lpWndClass： 視窗類資訊結構體指標

【返 回 值】TRUE：註冊成功；FALSE：註冊失敗

【頭 文 件】使用本函數需要包含”window.h”

註冊視窗類函數中需要用到一個 WNDCLASS 結構體，在調用 RegisterClass 函數之前，必須首先填充這個結構體以便指定視窗類的屬性。WNDCLASS 結構體定義如下：

```
typedef struct _WNDCLASS {
    UINT style;
    WNDPROC lpfnWndProc;
    int cbClsExtra;
    int cbWndExtra;
    HANDLE hInstance;
    HICON hIcon;
```


Tel:010-62981113-2944/2940/2942/2945 http://www.unsp.com
E-mail:unsp@sunplus.com.cn

```
    HCURSOR hCursor;
    HBRUSH hbrBackground;
    LPCTSTR lpszMenuName;
    LPCTSTR lpszClassName;
} WNDCLASS;
```

WNDCLASS 結構體各成員的描述如下：

- style ：視窗類的風格。使用此視窗類創建的視窗都將繼承這個屬性。視窗類的風格可以組合使用，常用的可選的值有：
  - CS_DBLCLKS：窗口支援雙擊事件
  - CS_HREDRAW：當視窗的寬度改變時，重畫整個視窗
  - CS_VREDRAW：當視窗的高度改變時，重畫整個視窗
  - CS_NOCLOSE：視窗沒有關閉按鈕
- lpfnWndProc：視窗函數指標。使用此視窗類創建的視窗均由此函數指標所指向的函數來進行消息處理；
- cbClsExtra：在申請視窗類結構體時，額外申請位元組的數量；
- cbWndExtra：在創建視窗時，額外申請位元組的個數；
- hInstance：視窗實例控制碼；
- hIcon：設置視窗類圖示控制碼，MicroWin 系統不關心該屬性；
- hCursor：設置視窗類游標控制碼；
- hbrBackground：設置視窗的背景色畫刷控制碼；
- lpszMenuName：設置視窗的功能表控制碼，MicroWin 不關心該屬性；
- lpszClassName：視窗類名，該參數是一個字串標識，用來唯一標識一個視窗類。在接下來創建視窗時需要使用。

註冊視窗類之後，便可以使用 CreateWindowEx 函數來創建一個視窗。CreateWindowEx 的函數原型和功能描述如下：

【函數原型】HWND CreateWindowEx(DWORD dwExStyle, LPCTSTR lpClassName,
LPCTSTR lpWindowName, DWORD dwStyle,
int x, int y, int nWidth, int nHeight,
HWND hWndParent, HMENU hMenu,
HINSTANCE hInstance, LPVOID lpParam);

【功　　能】創建一個視窗

【參　　數】dwExStyle： 視窗擴展風格，目前版本的 MicroWin 系統不支援，直接填 0 即可

lpClassName： 視窗類名，必須是一個已經註冊過的視窗類

lpWindowName： 視窗名稱，會被顯示在標題欄

dwStyle： 視窗風格，樣式說明見 表 2.1

x： 窗口左上角横坐標

y： 窗口左上角縱坐標

nWidth： 窗口寬度

nHeight： 窗口高度

hWndParent :該視窗的父視窗控制碼

hMenu： 該視窗使用的功能表的控制碼

hInstance： 應用實例控制碼

lpParam： 視窗私有資料，一般爲 NULL

【返 回 值】創建成功返回窗口控制碼，創建失敗返回 NULL；

【頭 文 件】使用本函數需要包含”window.h”

表 2.1 常用視窗樣式

樣式	說明
WS_BORDER	帶有邊框的視窗
WS_CAPTION	帶有標題欄的視窗
WS_HSCROLL	帶水準捲軸的視窗
WS_VSCROLL	帶垂直捲軸的窗口
WS_OVERLAPPEDWINDOW	創建一個標準視窗（帶有邊框、標題欄、關閉按鈕）
WS_POPUP	彈出式窗口
WS_POPUPWINDOW	帶有邊框的彈出式視窗
WS_CHILD	指示該視窗是其他視窗的字視窗
WS_VISIBLE	窗口可見
WS_DISABLED	窗口初始不可用

3) 顯示視窗

視窗創建完成後，可以使用 ShowWindow 函數設置視窗的顯示方式。ShowWindow 函數還可以用來隱藏/顯示指定的視窗，它的函數原型和功能描述如下：

【函數原型】BOOL WINAPI ShowWindow(HWND hwnd, int nCmdShow);

【功 能】設置指定視窗的顯示方式。

Tel:010-62981113-2944/2940/2942/2945 http://www.unsp.com
E-mail:unsp@sunplus.com.cn

【參　　數】hwnd： 窗口控制碼

nCmdShow： 視窗的顯示方式。在 MicroWin 系統中該參數爲 0 表示將視窗隱藏，不爲 0 表示將視窗顯示出來

【返 回 值】TRUE：設置成功；FALSE：設置失敗

【頭 文 件】使用本函數需要包含”window.h”

顯示視窗後，應用程式常常調用 UpdateWindow 函數更新並繪製用戶區，使應用程式得以立即顯示。UpdateWindow 函數的函數原型和功能描述如下：

【函數原型】BOOL WINAPI UpdateWindow(HWND hwnd);

【功　　能】重繪視窗的用戶區，並向視窗發送 WM_PAINT 消息。

【參　　數】hwnd： 窗口控制碼

【返 回 值】TRUE：操作成功；FALSE：操作失敗

【頭 文 件】使用本函數需要包含”window.h”

4) 消息迴圈

WinMain 函數的最後一步工作是構造一個消息迴圈，以便從消息佇列中取出消息並分發給各個視窗進行處理。消息迴圈的常見格式如下：

```
MSG msg;
……
while(GetMessage(&msg, NULL, 0, 0))          // 從消息佇列獲取一條消息
{
    TranslateMessage(&msg);                  // 將消息的虛擬鍵轉換爲字元資訊
    DispatchMessage(&msg);                   // 將消息分發給指定的視窗函數
}
```

應用程式依靠這個消息迴圈不斷從消息佇列中獲取消息，並將消息分發給擁有該消息的視窗的視窗函數進行處理。

5) 視窗函數

視窗函數定義了應用程式對接收到的不同消息的回應，其中包含了應用程式對各種可能接收到的消息的處理過程。通常，視窗函數由一個或多個 switch 語句組成。每一條 case 語句對應一種消息，當應用程式接收到一個消息時，相應的 case 語句所定義的程式被執行。

視窗函數的一般形式如下：

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message,
                                    WPARAM wParam, LPARAM lParam);
{
    Switch(message)
    {
        case …:
            …
            break;
        case WM_DESTROY:
            PostQuitMessage(0);
            break;
        default:
            return DefWindowProc(hWnd, message, wParam, lParam);
    }
    return 0;
}
```

在視窗函數中程式師只需根據視窗可能收到的消息增加 case 語句並編寫相應的處理程式即可。在視窗函數中一般都有對 WM_DESTROY 消息的處理，該消息是關閉視窗時發出的。一般情況下，應用程式應當在對這個消息的處理程式中調用 PostQuitMessage 函數來回應它。PostQuitMessage 函數的函數原型和功能描述如下：

【函數原型】void PostQuitMessage(int nExitCode);

【功　　能】向應用程式發出 WM_QUIT 消息，退出消息迴圈。

【參　　數】nExitCode：退出碼，MicroWin 系統不關心此參數

【返 回 值】無

【頭 文 件】使用本函數需要包含”window.h”

另外，視窗函數一般不可能針對所有的消息都編寫對應的 case 語句，所以，在視窗函數中一般必須有下面的語句，爲未定義處理程式的消息提供預設的處理：

```
default:
    return DefWindowProc(hWnd, message, wParam, lParam);
```


Tel:010-62981113-2944/2940/2942/2945 http://www.unsp.com
E-mail:unsp@sunplus.com.cn

DefWindowProc 函數是系統提供的預設視窗函數，以保證所有發送到視窗的消息均得以處理。

6) MicroWin 系統中常見的視窗消息

MicroWin 程式所有的動作都是通過消息實現的，下面是常用到的消息。

- WM_CREATE：創建視窗時會發送這個消息，一些初始化的工作可以在這個消息中完成，比如創建子視窗、按鈕、文本框等。
- WM_PAINT：無論何時當視窗的某一部分需要更新時系統就會發送這個消息給視窗。視窗函數中的代碼最好能使 MicroWin 程式在回應 WM_PAINT 消息時完成所有顯示區域的繪製功能。因爲程式必須在收到 WM_PAINT 消息時就更新整個顯示區域，如果在程式的其他部分也繪製的話，會導致程式碼重複，或者導致系統發送 WM_PAINT 消息時有一些元素無法更新。
- WM_LBUTTONDOWN：滑鼠左鍵在視窗內單擊產生這個消息，lparam 參數的高位元組存放的是滑鼠的 Y 座標、lpram 參數的低位元組存放的是滑鼠的 X 座標。
- WM_COMMAND：由字視窗（控制元等）發送來的消息。當控制元與用戶交互時通常會向父視窗發送一個 WM_COMMAND 消息，wparam 參數的低位元組存放的是控制元的 ID 號，父視窗可以根據這個消息完成用戶交互處理。

上面是建立一個 MicroWin 程式的基本框架，每一個 MicroWin 程式碼中都包括上面幾部分中的大部分。很少人真正記住上面代碼的全部寫法，通常，編寫 MicroWin 程式時往往先複製一個現有的程式，然後再做相應的修改。您可以按照此習慣自由使用本書附帶光碟中的程式。

4、 在視窗上輸出文字

在視窗上輸出文字屬於 MicroWin 的繪圖工作。在視窗的顯示區域繪圖需要依賴於 MicroWin 系統中的設備環境 DC。有關設備環境的概念在後面的章節中會有詳細介紹，這裏讀者只需要瞭解輸出文字（繪圖）的步驟即可。

一般情況下，在 MicroWin 系統中輸出文字的基本步驟爲：

- 在視窗函數中增加對 WM_PAINT 消息的處理；
- 在 WM_PAINT 消息中使用 BeginPaint 函數獲取設備環境
- 使用 SetTextColor 和 SetBkColor 函數設置文字的顏色和背景顏色
- 使用 DrawText 或 TextOut 函數進行文字輸出
- 使用 EndPaint 函數釋放設備環境

使用 BeginPaint 函數獲取設備環境控制碼並進行繪圖的的典型用法如下：

```
HDC hdc;                                    // 設備環境控制碼
```

```
    PAINTSTRUCT psd;                        // 重繪結構體，供系統使用
    switch(message)
    {
        case WM_PAINT:
            hdc = Beginpaint(hWnd, &psd);   // 獲得設備環境控制碼
            ……                              // 這裏可以使用設備環境控制碼 hdc 進行繪圖操作
            EndPaint(hWnd, &psd);           // 釋放設備環境
            break;
        ……
    }
```

後面的章節將詳細介紹包括 BeginPaint 和 EndPaint 在內的各種繪圖函數，本節只介紹與文字輸出相關的 API。常用的與輸出文字相關的 API 如下：

【函數原型】BOOL TextOut(HDC hdc, int x, int y, LPCSTR lpszString, int cbString)

【功　　能】主要用於向表單輸出文本

【參　　數】hdc：設備環境控制碼

x：文本顯示的左上角水準座標

y：文本顯示的左上角垂直座標

lpszString：要顯示的字串指標

cbString：顯示字元數,如小於 0 則顯示整串,以'\0'結尾

【返 回 值】函數執行成功則返回 TRUE,失敗返回 FALSE

【頭 文 件】使用本函數需要包含”window.h”

【函數原型】int WINAPI DrawTextA(HDC hdc, LPCSTR lpString, int nCount,

LPRECT lpRect, UINT uFormat)

【功　　能】在指定的矩形內輸出帶有格式的 ASCII 文本

【參　　數】hdc：設備環境控制碼

lpString：要顯示的字串指標

lpRect：指向 RECT 結構的指標，標識了字串輸出的矩形區域

uFormat：字串輸出格式。格式說明見 表 2.2

Tel:010-62981113-2944/2940/2942/2945 http://www.unsp.com
E-mail:unsp@sunplus.com.cn

【返 回 值】如果函數調用成功，返回值是正文的高度；如果函數調用失敗，返回值是 0。

【頭 文 件】使用本函數需要包含”window.h”

【函數原型】COLORREF WINAPI SetTextColor(HDC hdc, COLORREF crColor)

【功　　能】設定文本顏色。

【參　　數】hdc：設備環境控制碼

crColor：文本顏色。可以通過宏 RGB(r,g,b)產生，如 RGB(255,0,0)表示紅色

【返 回 值】如果成功，返回設備的舊顏色值。失敗，則返回值爲 CLR_INVALID。

【頭 文 件】使用本函數需要包含”window.h”

【函數原型】COLORREF WINAPI　SetBkColor(HDC hdc, COLORREF crColor)

【功　　能】設置當前的背景色，如果沒有要設置的顏色則用最接近的顏色代替。

【參　　數】hdc：設備環境控制碼

crColor：文本的背景色，可以通過宏 RGB(r,g,b)產生，如 RGB(255,0,0)表示紅色

【返 回 值】成功，返回舊背景色的顏色值，失敗，返回 CLR_INVALID

【頭 文 件】使用本函數需要包含”window.h”

表 2.2 常用文本輸出格式說明

格式	說明
DT_LEFT	文本左對齊
DT_CENTER	文本居中
DT_RIGHT	文本右對齊
DT_WORDBREAK	當文本寬度超出給定矩形寬度時，自動按單詞爲單位換行
DT_SINGLELINE	即使文本寬度超出給定矩形寬度也不換行
DT_CALCRECT	不輸出文本，只計算輸出文本所占的寬度和高度

5、 本實驗原理

在本實驗中，使用上面敍述的 MicroWin 應用程式的框架建立一個最簡單的視窗。創建視窗在 WinMain 函數中進行，幾乎所有的 MicroWin 的應用程式的 WinMain 函數都可以採用這個範本，參考代碼如下：

```
int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine,
```

```
                                int nCmdShow)
    {
        TCHAR szAppName[]    = "Window_CLASS";
        TCHAR szAppTitle[] = "Window Demo";
        HWND hwnd;
        MSG msg;
        WNDCLASS wndclass;
        //===============註冊視窗類===============//
        wndclass.style = CS_DBLCLKS | CS_HREDRAW | CS_VREDRAW; //  視窗類的風格
        wndclass.lpfnWndProc = (WNDPROC)WndProc;        //  視窗過程函數指標
        wndclass.cbClsExtra = 0;                        //  創建視窗類時，額外申請位元組的個
數
        wndclass.cbWndExtra = 0;                        //  創建視窗時，額外申請位元組的個數
        wndclass.hInstance = hInstance;                 //  視窗實例控制碼
        wndclass.hIcon = 0;                             //  設置視窗類圖示控制碼
        wndclass.hCursor = 0;                           //  設置視窗類游標控制碼
        wndclass.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);//設置視窗的背景色
        wndclass.lpszMenuName = NULL;                   //  設置視窗的功能表控制碼
        wndclass.lpszClassName = szAppName;             //  窗口類名
        RegisterClass(&wndclass);                       //  註冊視窗類
        //===============創建窗口===============//
        hwnd = CreateWindowEx(0L, szAppName,            //  窗口類名
                              szAppTitle,               //  窗口標題
                              WS_OVERLAPPEDWINDOW | WS_VISIBLE,//  視窗樣式
                              0, 0,                     //  窗口左上角座標
                              320, 240,                 //  窗口大小
                              NULL,                     //  父表單控制碼
                              NULL,                     //  主菜單控制碼
                              NULL,                     //  實例控制碼
                              NULL);                    //  窗口參數
        //===============顯示視窗===============//
```


Tel:010-62981113-2944/2940/2942/2945 http://www.unsp.com
E-mail:unsp@sunplus.com.cn

```
    ShowWindow(hwnd, nCmdShow);                    // 設置視窗顯示方式
    UpdateWindow(hwnd);                            // 立即繪製窗口
    //===============消息迴圈===============//
    while(GetMessage(&msg, NULL, 0, 0))
    {
        TranslateMessage(&msg);                    // 轉換消息
        DispatchMessage(&msg);                     // 投遞消息
    }
    return msg.wParam;
}
```

在視窗中輸出若干字串的功能需要在視窗函數中對 WM_PAINT 消息的處理中實現。視窗函數的參考代碼如下：

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam,
                                    LPARAM lParam)
{
    PAINTSTRUCT ps;
    HDC hdc;
    RECT rect;
    switch(message)
    {
        case WM_PAINT:
            hdc=BeginPaint(hWnd, &ps);             // 得到設備環境控制碼
            rect.top = rect.left = 0;              // 定義一個矩形區域
            rect.bottom = rect.right = 200;
            SetBkColor(hdc, RGB(0, 255, 0));       // 設置背景色爲綠色
            SetTextColor(hdc, RGB(255, 0, 0));     // 設置文本色爲紅色
            TextOut(hdc, 100, 50, "Hello MicroWin World", -1);//輸出文本

            SetBkColor(hdc, RGB(0, 0, 255));       // 設置背景色爲藍色
            SetTextColor(hdc, RGB(0, 255, 0));     // 設置文本色爲綠色
```

```
            DrawText(hdc, "welcom come into MicroWin World", -1, &rect, 0);
                                                    // 輸出文本
            rect.top += 100;                        // 修改矩形的大小
            SetBkColor(hdc, RGB(255, 255, 255));    // 設置背景色爲白色
            SetTextColor(hdc, RGB(0, 255, 0));      // 設置文本色爲綠色
            DrawText(hdc, "Best Regards", -1, &rect, DT_CENTER);//輸出文本

            SetBkColor(hdc, RGB(128, 128, 128));    // 設置背景色爲灰色
            SetTextColor(hdc, RGB(0, 0, 0));        // 設置文本色爲黑色
            DrawTextA(hdc, "1234567890", -1, &rect, DT_RIGHT);// 輸出文本
            EndPaint(hWnd, &ps);                    // 釋放設備環境
            break;
        case WM_DESTROY:
            PostQuitMessage(0);                     // 發送退出消息迴圈消息
            break;
        default:
            return DefWindowProc(hWnd, message, wParam, lParam);  // 系統預設視窗函數
    }
    return 0;
}
```

【實驗步驟】

1、 打開 S+core IDE，使用“Score IDE MicroWin Project”範本採用預設設置新建一個工程；

2、 工程嚮導會生成 APPWinMain.c 檔案，其中已經包含了 WinMain 函數和一個預設的視窗函數，在視窗函數中增加對 WM_PAINT 消息的處理，編寫輸出文本的程式；

3、 把 实验十一中的“default_install”檔案夾拷貝到新建的工程目錄下；

4、 修改、編譯（Rebuild All）直到沒有任何錯誤；

5、 利用 RedBoot 下載程式到開發板上脫機運行；

6、 觀察螢幕上的輸出結果。

Tel:010-62981113-2944/2940/2942/2945 http://www.unsp.com
E-mail:unsp@sunplus.com.cn

【範例路徑】

在大學計畫網站（score.unsp.com）或者 32 位元嵌入式開發系統光碟都會提供本實驗的參考程式，路徑如下：

\Example_eCos\2.GUI_Exa\ex12_Window

实验十三　繪圖實驗

【實驗目的】

1、 瞭解 MicroWin 的設備無關繪圖引擎特性；

2、 掌握 MicroWin 下進行基本繪圖的方法。

【實驗設備】

1、 裝有 Windows 系統和 S+core IDE 集成開發環境的 PC 機一台；

2、 凌陽 SPCE3200 嵌入式精簡開發板一套；

3、 本實驗用到開發板的介面有：TFT LCD 介面。

【實驗要求】

使用 MicroWin 的繪圖函數在視窗上繪製四個填充圖形：第一個是用灰色填充的紅邊矩形；第二個是黑色邊框的空心橢圓；第三個是用藍色填充的黑邊扇形；第四個是用綠色填充的黑邊多邊形。

【實驗原理】

MicroWin 爲圖形顯示和繪圖操作提供了強大的 GDI（Graphics Device Interface）函數，使得應用程式無需考慮特殊的硬體設置。GDI 函數集合提供了包括基本繪圖、選取操作在內的一系列 API 函數，應用程式可以調用這些函數方便的完成圖形繪製、文字輸出、選取填充等操作。

1、 設備描述表

設備描述表是用來描述 GDI 輸出設備的輸出位置、輸出屬性等資訊的一種結構。MicroWin 的圖形引擎採用設備描述表來確定任何設備的 GDI 輸出的位置和形象的屬性。應用程式無需也不能存取設備描述表，但是，應用程式可以使用設備描述表的控制碼來間接地存取設備描述表及其屬性。應用程式對圖形的操作均參照設備描述表中的屬性執行，因此可以將設備描述表看成圖形的“輸出範本”，依靠這塊範本，當應用程式調用 GDI 函數輸出圖形和文字時，不必關心諸如背景顏色、字體等問題。

2、 圖形刷新

圖形刷新是繪圖過程中必須考慮的問題，圖形刷新包括刷新的請求、系統對刷新請求的回應以及具體的刷新方法。

首先考慮這樣一種實際情況：應用程式在視窗內繪製了一個橢圓，然後顯示了一個用來選擇顏色的列表框，用戶在列表框上選擇填充橢圓內部的顏色。但是，列表框覆蓋了橢圓的一部分，現在的問題是，當用戶完成顏色選擇操作並關閉列表框後，應用程式將如何恢復橢圓被覆蓋部分的顏色

Tel:010-62981113-2944/2940/2942/2945　http://www.unsp.com
E-mail:unsp@sunplus.com.cn

和形狀。

MicroWin 應用程式的大部分用戶操作都集中在視窗內，因此上述情況可能頻繁出現。在視窗大小調整、視窗移動或被其他物件覆蓋後，都必須刷新視窗內用戶區的內容，以恢復用戶區應有的顯示形態。但是，MicroWin 系統並不總是記錄視窗中需要保存的內容，這樣做既不現實也沒有必要，系統只在幾種情況下自動刷新。因此，應用程式必須具有及時處理刷新請求和刷新圖形的功能。

MicroWin 系統通常通過發送 WM_PAINT 消息通知應用程式刷新其介面內容。當用戶區的內容需要刷新時，系統在應用程式的消息佇列中加入該消息，以通知視窗函數執行刷新處理。

一般情況下，系統向應用程式消息佇列發送 WM_PAINT 消息的情況有三種可能：視窗移動後需要刷新、被覆蓋區域需要刷新以及物件穿越後需要刷新。

視窗移動後的刷新可以理解爲下列事件的發生，這時系統將向應用程式發送 WM_PAINT 消息：

- 用戶區移動或顯示
- 用戶視窗大小改變
- 程式通過捲軸滾動視窗

被覆蓋區域的刷新發生在下面的事件產生時。此時，系統將試圖保存被覆蓋的區域，以備以後刷新。此後如果系統不能有效刷新，則向應用程式發送 WM_PAINT 消息：

- 下拉式功能表關閉，並需要恢復被覆蓋的部分
- 其他表單關閉時，需要恢復被覆蓋的部分

對於這種情況，一個視窗被另一個視窗覆蓋的區域被稱爲無效區域。MicroWin 系統爲每個視窗建立了一個 PAINTSTRUCT 結構，該結構中包含了包圍無效區域的一個最小矩形的結構 RECT，這個矩形稱爲無效矩形。應用程式可以根據這個無效矩形執行刷新操作。

物件穿越後需要刷新的 情況，MicroWin 系統會自動完成刷新任務，應用程式不必考慮。如游標穿過用戶區或者圖示拖過用戶區等。

爲了執行有效的刷新，應用程式必須全面分析系統可能發送的刷新請求，並根據不同的情況進行分別處理。常用的 MicroWin 應用程式刷新視窗的方法如下：

- 在記憶體中保持一個顯示輸出的副本，當需要重繪視窗時，將副本拷貝到相應的視窗中。該方法適用於刷新位元映射等複雜圖形
- 記錄曾經發生過的事件，在視窗需要刷新時重新調用視窗執行這個事件
- 重新繪製圖形，一般對於簡單圖形常採用重新繪製圖形方法執行刷新。在應用程式中，將圖形繪製處理程式放在消息 WM_PAINT 回應模組中，一旦程式接收到刷新請求即可重繪圖形

3、 設備環境

獲取設備環境（又稱設備上下文）是應用程式輸出圖形的先決條件。設備環境是與設備描述表對應的一個結構。應用程式必須指定繪圖操作使用的設備環境，系統才能使用設備環境所規定的輸出屬性進行繪圖。

常用的兩種獲取設備環境的方法是調用函數 BeginPaint 或 GetDC。

1) 調用 BeginPaint 函數

應用程式回應 WM_PAINT 消息進行繪圖刷新時，主要通過調用 BeginPaint 函數獲取設備環境。BeginPaint 函數的函數原型和功能描述如下：

【函數原型】HDC WINAPI BeginPaint(HWND hWnd, LPPAINTSTRUCT lpPaint);

【功　　能】獲取指定視窗的設備環境

【參　　數】hWnd：窗口控制碼

lpPaint ： 用於標識視窗無效區域的結構體，系統將填寫該結構體以便標識需要刷新的無效區域，並爲後續過程提供進一步的處理

【返 回 值】設備環境控制碼

【頭 文 件】使用本函數需要包含”window.h”

由 BeginPaint 函數獲取的設備環境必須用 EndPaint 函數釋放，二者必須配對使用。EndPaint 函數的函數原型和功能描述如下：

【函數原型】void EndPaint(HWND hWnd, PAINTSTRUCT &ps);

【功　　能】釋放視窗的設備環境

【參　　數】hWnd：窗口控制碼

ps：用於標識視窗無效區域的結構體

【返 回 值】無

【頭 文 件】使用本函數需要包含”window.h”

典型的使用 BeginPaint 函數和 EndPaint 函數進行繪圖操作的程式片段如下：

```
case WM_PAINT:                  // WM_PAINT 消息處理程式
    HDC hdc;
    PAINTSTRUCT ps;
    hdc = BeginPaint(hWnd, &ps);  // 獲取設備環境, hWnd 爲當前視窗控制碼
    ... ...                     // 使用 hdc 進行繪圖操作
    EndPaint(hWnd, &ps);        // 結束繪圖
    break;
```

2) 調用 GetDC 函數

如果應用程式的繪圖工作並非由 WM_PAINT 消息引起，則需要調用 GetDC 函數獲取設備環境。

Tel:010-62981113-2944/2940/2942/2945 http://www.unsp.com
E-mail:unsp@sunplus.com.cn

GetDC 函數的函數原型和功能描述如下：

【函數原型】HDC WINAPI GetDC(HWND hWnd);

【功　　能】獲取指定視窗的設備環境

【參　　數】hWnd：窗口控制碼

【返 回 值】設備環境控制碼

【頭 文 件】使用本函數需要包含”window.h”

GetDC 函數可以獲取指定視窗的設備環境而無需 PAINTSTRUCT 結構體，這是因爲該函數並不一定在 WM_PAINT 消息的處理函數中使用，所以無需標識視窗的無效矩形。由該函數獲取的設備環境必須由 ReleaseDC 函數釋放，二者也必須配對使用。ReleaseDC 函數的函數原型和功能描述如下：

【函數原型】void ReleaseDC(HWND hWnd);

【功　　能】釋放視窗的設備環境

【參　　數】hWnd：窗口控制碼

【返 回 值】無

【頭 文 件】使用本函數需要包含”window.h”

兩種獲取設備環境的方法的區別如 表 2.3所示。

表 2.3 BeginPaint 與 GetDC 函數的區別

	BeginPaint/EndPaint 函數	GetDC/ReleaseDC 函數
適用場合	只用于 WM_PAINT 圖形刷新時獲取設備環境	使用較爲廣泛
操作區域	操作區域爲無效區域	操作區域爲整個用戶區

4、 坐標系統

MicroWin 系統中絕大多數的 API 都使用視窗的相對座標（不包括標題欄和 3D 外框的區域）進行繪圖，這個坐標系統“用戶坐標系統”。使用 GetClientRect 函數可以獲得指定視窗的用戶區域，以便將來在視窗的用戶區內進行繪圖。

另外，應用程式還可以選擇使用“視窗坐標系統”，此時的座標是包含標題欄和 3D 外框在內的區域。使用 GetWindowRect 函數可以獲得指定視窗的視窗區域，此時可以在包括標題欄在內的區域內進行繪圖。

GetClientRect 和 GetWindowRect 函數使用了 RECT 結構體來保存區域資訊，RECT 結構體的定義如下：

```
typedef struct {
    MWCOORD   left;
```

```
    MWCOORD   top;
    MWCOORD   right;
    MWCOORD   bottom;
} RECT;
```

其中 left 和 top 可以確定矩形的左上角座標，right 和 bottom 可以確定矩形的右下角座標。

MicroWin 系統還提供了 ClientToScreen 和 ScreenToClient 函數實現客戶座標與螢幕座標之間的轉換。

與坐標系統相關的函數如下：

【函數原型】BOOL WINAPI GetClientRect(HWND hWnd, LPRECT lpRect);

【功　　能】獲取指定視窗的用戶區域

【參　　數】hWnd： 窗口控制碼

lpRect： 用於保存用戶區域的矩形結構體。該結構體的左上角座標爲 0，右下角座標標識了用戶區域的寬度和高度

【返 回 值】成功返回 TRUE，失敗返回 FALSE

【頭 文 件】使用本函數需要包含”window.h”

【函數原型】BOOL WINAPI GetWindowRect(HWND hWnd, LPRECT lpRect);

【功　　能】獲取指定視窗的視窗區域

【參　　數】hWnd： 窗口控制碼

lpRect： 用於保存視窗區域的矩形結構體。該結構體標識了視窗在螢幕上的左上角和右下角的座標

【返 回 值】成功返回 TRUE，失敗返回 FALSE

【頭 文 件】使用本函數需要包含”window.h”

【函數原型】BOOL WINAPI ScreenToClient(HWND hWnd, LPPOINT lpPoint);

【功　　能】將螢幕上某一點的座標轉換爲某個視窗的用戶區座標

【參　　數】hWnd： 窗口控制碼

lpPoint： 待轉換的螢幕座標，函數執行後該結構體將保存轉換後的用戶區座標

【返 回 值】成功返回 TRUE，失敗返回 FALSE

【頭 文 件】使用本函數需要包含”window.h”

【函數原型】BOOL WINAPI ClientToScreen(HWND hWnd, LPPOINT lpPoint);

Tel:010-62981113-2944/2940/2942/2945 http://www.unsp.com
E-mail:unsp@sunplus.com.cn

【功　　能】將某個視窗的用戶區座標轉換爲螢幕座標

【參　　數】hWnd：窗口控制碼

lpPoint：待轉換的用戶區座標，函數執行後該結構體將保存轉換後的螢幕座標

【返 回 值】成功返回 TRUE，失敗返回 FALSE

【頭 文 件】使用本函數需要包含”window.h”

5、 顏色的設置與應用

MicroWin系統使用 32 位元無符號整數表示顏色，其中低三位元組分別表示紅、綠、藍三個顏色值，每一個顏色值的範圍是 0~255，如 圖 2.7所示。

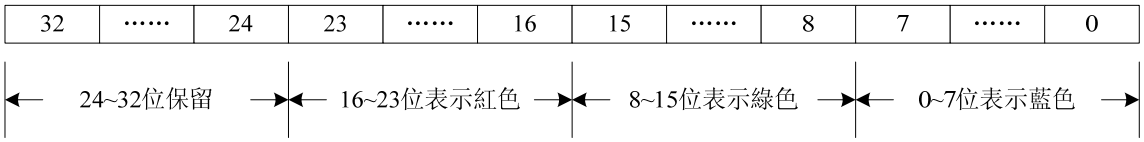

圖 2.7 32 位元無符號整數表示顏色

使用宏 RGB 可以定義繪圖的顏色，其形式爲：

RGB(nRed, nGreen, nBlue)

其中 nRed、nGreen 和 nBlue 分別表示紅色值、綠色值和藍色值，例如 RGB(255, 0, 0)表示紅色，RGB(0, 255, 0)表示綠色，RGB(0, 0, 255)表示藍色。在下面的繪圖操作中，一般都需要使用 RGB 巨集來指定繪圖使用的顏色。

6、 繪圖工具

MicroWin 系統使用畫筆和畫刷進行繪圖操作。畫筆的功能是繪製線條；畫刷則用於指定區域的填充。

1)　畫筆

畫筆的操作包括創建畫筆、將畫筆選入設備環境和刪除畫筆等操作。應用程式使用畫筆控制碼對畫筆進行標識和操作，設備環境也需要用戶指定一個畫筆控制碼以便確定線條的樣式。畫筆控制碼的型別爲 HPEN，可以使用它來定義一個畫筆控制碼。

應用程式可以使用 GetStockObject 函數獲取 MicroWin 系統內置的四種畫筆，這四種畫筆分別是：WHITE_PEN、BLACK_PEN、DC_PEN 和 NULL_PEN。GetStockObject 函數的函數原型和功能描述如下：

【函數原型】HGDIOBJ WINAPI GetStockObject(int nObject);

【功　　能】得到系統定義物件（畫筆，畫刷，字體等）的控制碼

【參　　數】nObject：指定的系統物件的型別

【返 回 值】成功，返回對象的控制碼；失敗，返回 NULL

【頭 文 件】使用本函數需要包含”window.h”

例如，可以使用下面的代碼來獲取系統的黑色畫筆：

```
HPEN hPen;                                    // 定義畫筆控制碼
hPen = (HPEN)GetStockObject(BLACK_PEN);   // 獲取系統黑色畫筆
```

除了系統定義的畫筆之外，用戶還可以使用 CreatePen 函數創建新的畫筆。CreatePen 函數的函數原型和功能描述如下：

【函數原型】HPEN WINAPI CreatePen(int nPenStyle, int nWidth, COLORREF crColor);

【功　　能】按照指定的樣式，寬度和顏色創建一個畫筆

【參　　數】nPenStyle：指定畫筆的樣式。MicroWin 系統僅支援 PS_SOLID（實線）和 PS_NULL（無線條）兩種樣式

nWidth：指定畫筆寬度（邏輯寬度），MicroWin 系統將忽略此參數

crColor：指定畫筆顏色，可以使用 RGB(r, g, b)宏來產生一個顏色

【返 回 值】成功，返回畫筆的控制碼；失敗，返回 NULL

【頭 文 件】使用本函數需要包含”window.h”

選擇或創建畫筆後，必須調用 SelectObject 函數將其選入設備環境。使用該函數後，應用程式將使用該畫筆繪圖，直至選入另外的一種畫筆爲止。SelectObject 函數的函數原型和功能描述如下：

【函數原型】HGDIOBJ WINAPI　SelectObject(HDC hdc, HGDIOBJ hObject);

【功　　能】選擇一個物件到指定的設備環境中，該物件將替換先前的同型別物件

【參　　數】hdc：設備環境控制碼

hObject：被選對象的控制碼

【返 回 值】如果選擇的物件不是區域並且函數成功執行，則返回被替換的物件的控制碼；如果選擇的物件是區域並且函數成功執行，則返回 表 2.4所示的值的其中一個；如果函數執行失敗則返回NULL

【頭 文 件】使用本函數需要包含”window.h”

表 2.4 SelectObject 函數返回值及說明

返回值	說明
SIMPLEREGION	區域包含一個簡單的矩形
COMPLEXREGION	區域是一系列矩形的集合
NULLREGION	區域爲空

Tel:010-62981113-2944/2940/2942/2945　http://www.unsp.com
E-mail:unsp@sunplus.com.cn

當不再使用當前畫筆時，需要使用 DeleteObject 函數刪除畫筆，以免佔用記憶體空間。DeleteObject 函數的函數原型和功能描述如下：

【函數原型】BOOL WINAPI DeleteObject(HGDIOBJ hObject);

【功　　能】刪除一個物件並釋放所有與該物件有關的資源，在物件被刪除後，指定的控制碼同時失效

【參　　數】hObject： 被刪除對象的控制碼

【返 回 值】刪除成功返回 TRUE，如果物件已經被設備環境選中

【頭 文 件】使用本函數需要包含”window.h”

2)　畫刷

畫刷的創建與應用與畫筆很相似，操作畫刷也包括創建、選入設備環境和刪除。畫刷控制碼的型別爲 HBRUSH，與畫筆一樣，應用程式需要使用畫刷控制碼對畫刷進行操作。

應用程式可以使用GetStockObject函數來獲取MicroWin系統內置的七種畫刷，畫刷的詳細樣式可以參考 表 2.5。

表 2.5 畫刷樣式及說明

樣式	說明	樣式	說明
BLACK_BRUSH	黑色畫刷	LTGRAY_BRUSH	亮灰色畫刷
DKGRAY_BRUSH	深灰色畫刷	NULL_BRUSH	空畫刷
GRAY_BRUSH	灰色畫刷	WHITE_BRUSH	白色畫刷
HOLLOW_BRUSH	虛畫刷		

例如，可以使用下面的代碼來獲取系統的灰色畫刷：

```
HBRUSH hBrush;                                  // 定義畫刷控制碼
hBrush = (HBRUSH)GetStockObject(GRAY_PEN);      // 獲取系統灰色畫刷
```

除了系統定義的畫刷之外，用戶還可以使用 CreateSolidBrush 函數創建一個具有指定顏色的單色畫刷。CreateSolidBrush 函數的函數原型和功能描述如下：

【函數原型】HBRUSH WINAPI CreateSolidBrush(COLORREF crColor);

【功　　能】用指定的顏色創建一個單色畫刷

【參　　數】crColor： 指定畫刷顏色，可以使用 RGB(r, g, b)宏來產生一個顏色

【返 回 值】成功，返回畫刷的控制碼；失敗，返回 NULL

【頭 文 件】使用本函數需要包含”window.h”

與畫筆一樣，選擇或創建畫刷後，必須調用 SelectObject 函數將其選入設備環境。當不再使用當前畫刷時，同樣需要使用 DeleteObject 函數刪除畫筆，以免佔用記憶體空間。

7、 常用繪圖函數

MicroWin 系統提供了多種繪圖函數，下面列出常用的幾種，更多的函數可以參考有關手冊。

【函數原型】COLORREF WINAPI SetPixel(HDC hdc, int x, int y, COLORREF crColor);

【功　　能】將指定座標處點的顏色設置爲指定顏色

【參　　數】hdc： 設備環境控制碼

x： 指定點的 x 座標

y： 指定點的 y 座標

crColor： 指定點的顏色，可以使用 RGB(r, g, b)宏來產生一個顏色

【返 回 值】函數執行成功則返回設置的顏色的 RGB 值，該值可能與 crColor 有所不同，這是因爲系統沒有找到與指定顏色匹配的顏色；如果函數執行失敗則返回-1

【頭 文 件】使用本函數需要包含”window.h”

【函數原型】COLORREF WINAPI GetPixel(HDC hdc, int x, int y);

【功　　能】獲取指定座標處點的顏色

【參　　數】hdc： 設備環境控制碼

x： 指定點的 x 座標

y： 指定點的 y 座標

【返 回 值】函數執行成功則返回指定點的顏色的 RGB 值；如果函數執行失敗則返回-1

【頭 文 件】使用本函數需要包含”window.h”

【函數原型】BOOL WINAPI MoveToEx(HDC hdc, int x, int y, LPPOINT lpPoint);

【功　　能】將繪圖點移動到指定位置，並可以選擇性的得到移動之前的點的位置

【參　　數】hdc： 設備環境控制碼

x： 新位置的 x 座標

y： 新位置的 y 座標

lpPoint ： 指向用於保存舊位置座標的 POINT 結構體變數的指標，如果該參數爲 NULL 則不保存舊位置的座標

Tel:010-62981113-2944/2940/2942/2945 http://www.unsp.com
E-mail:unsp@sunplus.com.cn

【返 回 值】函數執行成功則返回 TRUE；函數執行失敗則返回 FALSE

【頭 文 件】使用本函數需要包含”window.h”

【函數原型】BOOL WINAPI LineTo(HDC hdc, int x, int y);

【功　　能】從當前繪圖點繪製一條線段，並且不包含當前點

【參　　數】hdc：設備環境控制碼

x：結束點的 x 座標

y：結束點的 y 座標

【返 回 值】函數執行成功則返回 TRUE；函數執行失敗則返回 FALSE

【頭 文 件】使用本函數需要包含”window.h”

【函數原型】BOOL WINAPI Rectangle(HDC hdc, int nLeft, int nTop, int nRight, int nBottom);

【功　　能】繪製矩形，用當前的畫筆繪製矩形邊框，並用當前畫刷進行填充

【參　　數】hdc：設備環境控制碼

nLeft：矩形左上角的 x 座標

nTop：矩形左上角的 y 座標

nRight：矩形右下角的 x 座標

nBottom：矩形右下角的 y 座標

【返 回 值】函數執行成功則返回 TRUE；函數執行失敗則返回 FALSE

【頭 文 件】使用本函數需要包含”window.h”

【函數原型】BOOL WINAPI Ellipse(HDC hdc, int nLeft, int nTop, int nRight, int nBottom);

【功　　能】繪製橢圓。該橢圓內切于一個左上角爲(nLeft, nTop)右下角爲(nRight, nBottom)的矩形，用當前的畫筆繪製橢圓邊框，並用當前畫刷進行填充

【參　　數】hdc：設備環境控制碼

nLeft：外接矩形左上角的 x 座標

nTop：外接矩形左上角的 y 座標

nRight：外接矩形右下角的 x 座標

nBottom：外接矩形右下角的 y 座標

【返 回 值】函數執行成功則返回 TRUE；函數執行失敗則返回 FALSE

【頭 文 件】使用本函數需要包含”window.h”

【函數原型】BOOL WINAPI Arc(HDC hdc, int nLeft, int nTop, int nRight, int nBottom

int nXStart, int nYStart, int nXEnd, int nYEnd);

【功　　能】繪製橢圓弧。該橢圓內切于一個左上角爲(nLeft, nTop)右下角爲(nRight, nBottom)的矩形，使用一條由點(nXStart, nYStart)到橢圓中心線所連接的假像線與橢圓的交點爲起始點，使用一條由點(nXEnd, nYEnd)到橢圓中心線所連接的假像線與橢圓的交點爲終止點，逆時針方向在橢圓線上畫一段弧，如 圖 2.8所示

【參　　數】hdc：設備環境控制碼

nLeft：橢圓外接矩形左上角的 x 座標

nTop：橢圓外接矩形左上角的 y 座標

nRight：橢圓外接矩形右下角的 x 座標

nBottom：橢圓外接矩形右下角的 y 座標

nXStart：弧線起始點的 x 座標

nYStart：弧線起始點的 y 座標

nXEnd：弧線終止點的 x 座標

nYEnd：弧線終止點的 y 座標

【返 回 值】函數執行成功則返回 TRUE；函數執行失敗則返回 FALSE

【頭 文 件】使用本函數需要包含”window.h”

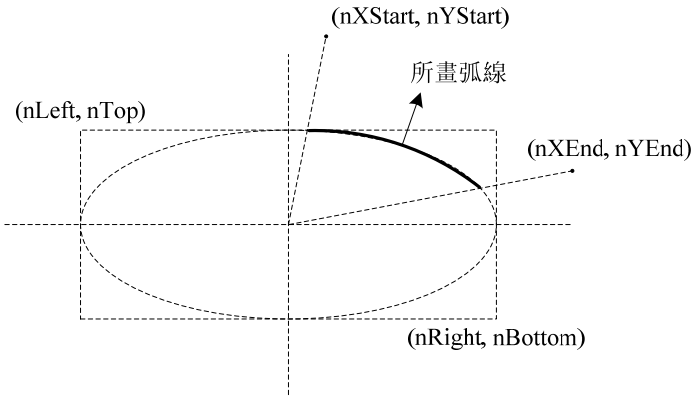

圖 2.8 Arc 函數功能示意圖

【函數原型】BOOL WINAPI Pie(HDC hdc, int nLeft, int nTop, int nRight, int nBottom

int nXStart, int nYStart, int nXEnd, int nYEnd);

【功　　能】繪製扇形區域。扇形由橢圓的一段圓弧及兩個弧端點到橢圓圓心的連線（半徑）所組成，使用當前畫筆繪製扇形邊框，並使用當前畫刷填充，如 圖 2.9所示

Tel:010-62981113-2944/2940/2942/2945 http://www.unsp.com
E-mail:unsp@sunplus.com.cn

【參　　數】hdc：設備環境控制碼

nLeft：橢圓外接矩形左上角的 x 座標

nTop：橢圓外接矩形左上角的 y 座標

nRight：橢圓外接矩形右下角的 x 座標

nBottom：橢圓外接矩形右下角的 y 座標

nXStart：弧線起始點的 x 座標

nYStart：弧線起始點的 y 座標

nXEnd：弧線終止點的 x 座標

nYEnd：弧線終止點的 y 座標

【返 回 值】函數執行成功則返回 TRUE；函數執行失敗則返回 FALSE

【頭 文 件】使用本函數需要包含”window.h”

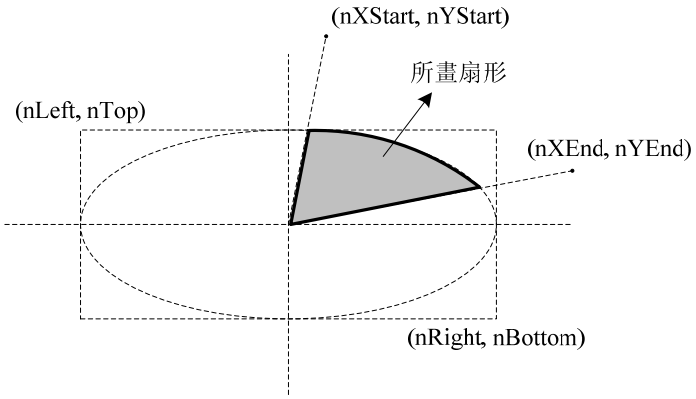

圖 2.9 Pie 函數功能示意圖

【函數原型】BOOL WINAPI Polygon(HDC hdc, CONST POINT *lpPoints, int nCount);

【功　　能】繪製多邊形

【參　　數】hdc：設備環境控制碼

lpPoints：多邊形頂點陣列

nCount：多邊形頂點個數

【返 回 值】函數執行成功則返回 TRUE；函數執行失敗則返回 FALSE

【頭 文 件】使用本函數需要包含”window.h”

【函數原型】BOOL WINAPI DrawDIB(HDC hdc,int x,int y,PMWIMAGEHDR pimage);

【功　　能】繪製 DIB 位元映射。位元映射資料可以使用 MicroWin 源碼目錄下的 \src\bin\convbmp.exe 工具獲得

【參　　數】hdc：設備環境控制碼

x：圖片左上角的 x 座標

y：圖片左上角的 y 座標

pimage：圖片資料

【返 回 值】函數執行成功則返回 TRUE；函數執行失敗則返回 FALSE

【頭 文 件】使用本函數需要包含”window.h”

【其他說明】使用實驗參考程式中附帶的圖形介面工具ConvBMP.exe可以將BMP檔案轉換爲DIB資料檔案。如 圖 2.10所示，首先點擊“Add”按鈕添加一個或多個BMP檔案，然後選擇一個輸出檔案，接著可以使用“Up”、“Down”、“Remove”等按鈕調整待轉換的BMP檔案列表，最後單擊“Convert”即可完成轉換，轉換後的資料保存在輸出檔案中。

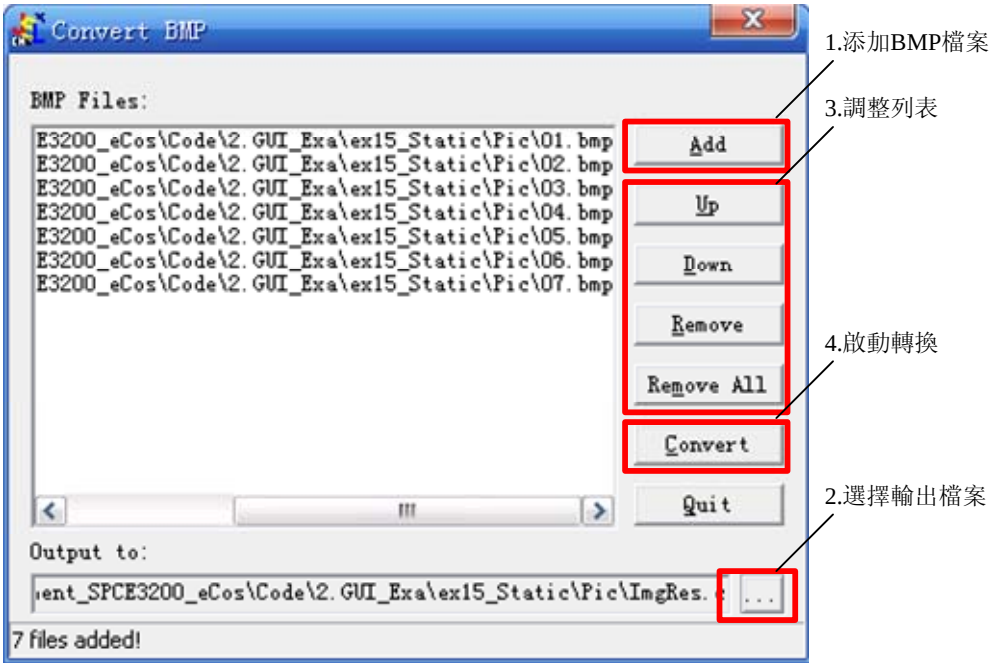

圖 2.10 ComvBMP 工具使用示意圖

8、 本實驗的實驗原理

在本實驗中，主要通過在WM_PAINT消息的處理函數中使用MicroWin系統提供的繪圖API進行圖形繪製。WM_PAINT消息的處理函數的程式流程如 圖 2.11所示。

Tel:010-62981113-2944/2940/2942/2945 http://www.unsp.com
E-mail:unsp@sunplus.com.cn

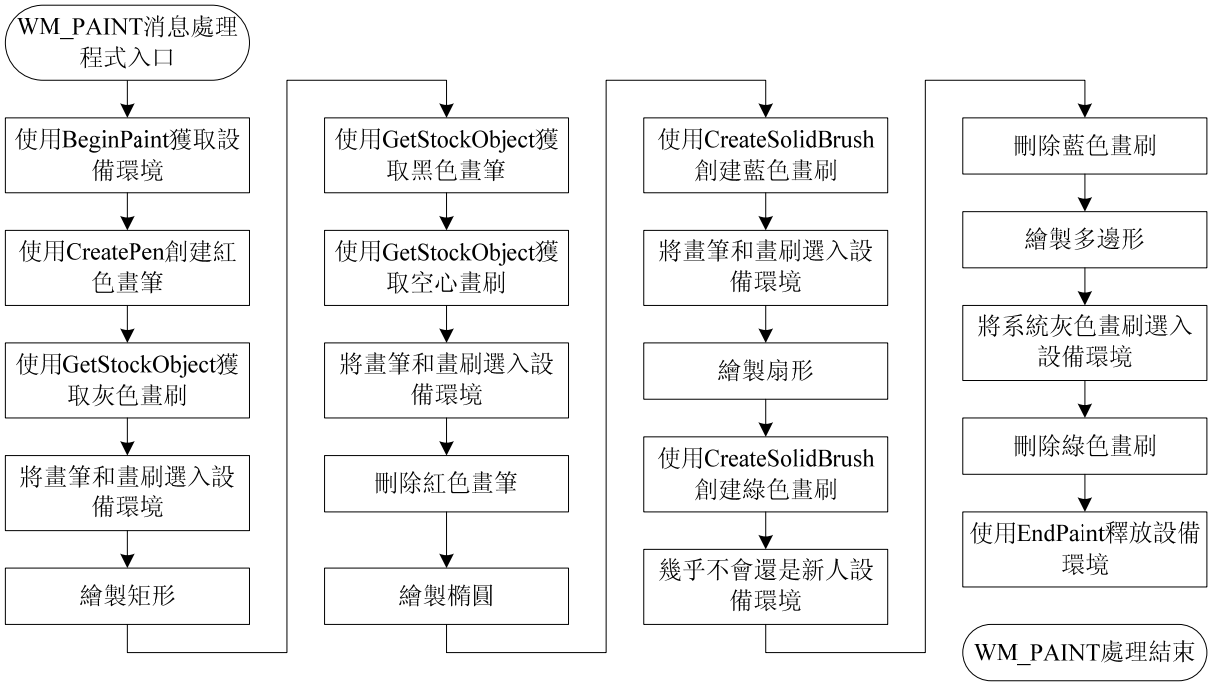

圖 2.11 實驗十三 WM_PAINT 消息處理程式流程圖

【實驗步驟】

1、 打開 S+core IDE，使用“Score IDE MicroWin Project”範本採用預設設置新建一個工程；

2、 工程嚮導會生成 APPWinMain.c 檔案，其中已經包含了 WinMain 函數和一個預設的視窗函數，在視窗函數中增加對 WM_PAINT 消息的處理，編寫輸出文本的程式；

3、 把 实验十一中的“default_install”檔案夾拷貝到新建的工程目錄下；

4、 修改、編譯（Rebuild All）直到沒有任何錯誤；

5、 利用 RedBoot 下載程式到開發板上並脫機運行，觀察 LCD 螢幕上繪製的圖案；

【範例路徑】

在大學計畫網站（score.unsp.com）或者 32 位元嵌入式開發系統光碟都會提供本實驗的參考程式，路徑如下：

\Example_eCos\2.GUI_Exa\ex13_Paint

实验十四　靜態控制元實驗

【實驗目的】

1、 掌握 MicroWin 下控制元的作用及其使用方法；

2、 掌握靜態控制元的使用方法

【實驗設備】

1、 裝有 Windows 系統和 S+core IDE 集成開發環境的 PC 機一台；

2、 凌陽 SPCE3200 嵌入式精簡開發板一套；

3、 本實驗用到開發板的介面有：TFT LCD 介面。

【實驗要求】

創建一個主視窗，並在主視窗內放置一個靜態控制元用於顯示圖像。爲主視窗設置一個計時器，在計時器中完成圖像的替換刷新，從而實現動畫效果。

【實驗原理】

1、 控制元概述

控制元是 MicroWin 圖形用戶介面的主要組成部分之一，用戶通過操作控制元物件完成與應用程式之間的交互。控制元的使用集中體現了 MicroWin 系統对象導向的特點。

控制元也是視窗，它具有通用的視窗屬性。可以使用ShowWindow()函數和MoveWindow()函數等視窗管理函數來顯示或隱藏控制元、改變控制元的位置、尺寸以及進行其他操作。在MicroWin系統中，控制元都是採用CreateWindow()或CreateWindowEx()函數來創建的，這兩個函數在創建控制元的時候爲控制控制元的風格提供了許多靈活性。用CreateWindow()或CreateWindowEx()函數來創建控制元，必須指定控制元的視窗類，每個控制元都應屬於某個視窗類，這種視窗類也像主視窗的視窗類那樣，在應用程式中定義並註冊。在更多的情況下，用戶只需要使用系統提供的預定義的控制元視窗類即可。MicroWin系統提供的標準控制元主要包括：靜態控制元、按鈕控制元、文本框控制元、下拉清單控制元、列表框控制元、進度條控制元和捲軸控制元等。MicroWin系統同時爲每個控制元提供了一個API函數用來對這類控制元視窗進行註冊，在使用某種控制元之前必須首先使用對應的API函數進行視窗類註冊。本書附錄中的 MicroWin提供六種常用的控制元，各個控制元對應的類名以及註冊函數API名稱如 表 3.1 所示。

表 3.1列出了系統預定義的控制元視窗及他們的類名和註冊 API。

控制元通過向其父視窗發送WM_COMMAND消息與應用程式進行通信。WM_COMMAND的wParam參數的低字包含了控制元子視窗的標識ID，高字代表了消息通知碼，用以通知父視窗用戶在子視窗內所進行的操作；lParam參數則代表了子視窗的控制碼。不同的控制元發送給父窗口的通知

碼是不一樣的。本書附錄中的 表 3.3列出了不同的控制元發送給父視窗的通知代碼。

父視窗可通過調用函數SendMessage向指定的子視窗控制元發送消息，以指示其動作。不同的控制元可以接受的消息也是不同的，附錄中的 表 3.4列出了MicroWin系統下各個控制元的控制消息。

2、 靜態控制元簡介

靜態控制元是一種包含文本或圖形的小視窗。應用程式通常使用靜態控制元標記其他控制視窗或分割不同組別的控制元。靜態控制元一般不接收用戶輸入也不發出消息，但是，應用程式可以通過設置靜態控制元的樣式使其能夠回應用戶輸入，向應用程式發送消息。

3、 靜態控制元的創建

MicroWin 提供的靜態控制元的類名爲：“STATIC”，在使用之前必須使用靜態控制元註冊函數對其進行註冊，靜態控制元註冊函數的函數原型如下：

int WINAPI MwRegisterStaticControl(HINSTANCE hInstance);

其中，hInstance 表示當前視窗的實例控制碼，一般爲 NULL；返回值表示是否註冊成功，非零則表示成功，否則表示失敗。失敗的原因可能是因爲已經註冊過，或者系統沒有足夠的記憶體。

在註冊靜態控制元之後，便可以使用 CreateWindow 函數創建它，如下：

```
hWnd = CreateWindow(“STATIC”,                    // 預定義靜態控制元類名
                    NULL,                        // 靜態控制元一般無標題
                    WS_CHILD | WS_VISIBLE | ..., // 靜態控制元樣式，樣式說明見 表 2.6
                    X, Y,                        // 靜態控制元在視窗中的位置
                    nWidth, nHeight,             // 靜態控制元的寬度和高度
                    hWndParent,                  // 擁有該靜態控制元的視窗控制碼
                    (HMENU)ID,                   // 靜態控制元標識
                    hInstance,                   // 應用程式當前實例控制碼，一般爲 0
                    NULL);                       // 靜態控制元私有資料
```

控制元的視窗屬性必須具有WS_CHILD樣式，一般都具有WS_VISIBLE樣式，同時，可以使用“|”選擇多個 表 2.6所示的擴展樣式。

表 2.6 靜態控制元常用樣式及說明

樣式	說明
SS_LEFT	文字左對齊
SS_CENTER	文本居中
SS_RIGHT	文本右對齊

SS_ICON	在靜態控制元上顯示圖示
SS_GROUPBOX	使靜態控制元作爲一個具有標題的分組框
SS_LEFTNOWORDWRAP	文本左對齊顯示，並且當文本長度超過靜態控制元寬度時不進行自動換行
SS_BITMAP	在靜態控制元上顯示位元映射（同 SS_ICON）
SS_NOTIFY	當有用戶輸入（如單擊）時強制使靜態控制元向父視窗發出消息
SS_CENTERIMAGE	位元映射居中（同時設置 SS_BITMAP 或 SS_ICON 時有效）
SS_REALSIZEIMAGE	不對位元映射進行自動縮放

注：SS_LEFT、SS_CENTER、SS_RIGHT、SS_ICON、SS_GROUPBOX、SS_LEFTNOWORDWRAP 和 SS_BITMAP 只能選擇其中一種樣式，不能使用“|”選擇多個。

如果用戶爲靜態控制元選擇了 SS_ICON 或 SS_BITMAP 樣式，則 CreateWindow 函數的最後一個參數(靜態控制元私有資料指標)將用來表示需要顯示的 DIB 位元映射。用戶可以在 CreateWindow 時爲其指定 DIB 位元映射，或者通過一個已經存在的靜態控制元發送 STM_SETIMAGE 來設置位元映射。

4、 靜態控制元與應用程式之間的消息傳遞

應用程式創建文本框控制元後，可通過接收控制元發出的消息得知用戶的輸入，並可通過向靜態控制元發送消息對其進行操作。

1) 靜態控制元向應用程式發送消息

一般情況下靜態控制元不發送資訊。但在實際應用中，常需要靜態控制元能夠回應用戶的輸入，向應用程式發送控制元消息。這時，應用程式在創建靜態控制元的時候需加入SS_NOTIFY樣式。該樣式允許靜態控制元向其父視窗發送WM_COMMAND消息，該消息的lParam參數保存了靜態控制元的控制碼；wParam參數的低位元組爲控制元標識，高位元組爲靜態控制元動作的消息通知碼。常用的通知碼及其說明可以參考 表 2.7。

表 2.7 靜態控制元常用通知碼及說明

通知碼	說明
STN_CLICKED	單擊靜態控制元
STN_DBLCLK	雙擊靜態控制元
STN_ENABLE	靜態控制元被使能
STN_DISABLE	靜態控制元被禁止

2) 應用程式向靜態控制元發送消息

應用程式對靜態控制元的操作通過調用函數SendMessage或PostMessage向其發送各種消息來完成。表 2.8列出了MicroWin支援的常用的靜態控制元控制消息及說明。

表 2.8 靜態控制元常用控制消息及說明

控制消息	說明
STM_SETICON	設置靜態控制元顯示的位元映射 參數：wParam： 新的 DIB 位元映射資料指標 返回值：舊的 DIB 位元映射數據指標
STM_GETICON	獲取靜態控制元當前顯示的位元映射 參數：無 返回值：DIB 位元映射數據指標
STM_SETIMAGE	同 STM_SETICON
STM_GETIMAGE	同 STM_GETICON

5、 本實驗的實驗原理

在本實驗中，首先使用 MicroWin 應用程式框架創建一個主視窗，並在 WM_CREATE 消息中使用 CreateWindow 創建一個靜態控制元。由於需要顯示動畫，所以在創建靜態控制元時爲其指定 SS_BITMAP 樣式。

接下來需要使用計時器每隔一定時間更新靜態控制元中的圖片，從而實現動畫效果。應用程式可以調用 SetTimer 函數來爲某個表單指定一個計時器，並設置定時週期。定時時間到之後，表單會接收到 WM_TIMER 消息，在此消息中可以完成需要週期性處理的動作，比如本實驗中需要更新圖片。當不再使用計時器時，可以調用 KillTimer 函數將計時器刪除。與計時器有關的函數及他們的功能描述如下：

【函數原型】UINT WINAPI SetTimer(HWND hwnd, UINT idTimer,

UINT uTimeout, TIMERPROC lpTimerFunc);

【功　　能】爲指定視窗創建一個計時器

【參　　數】hwnd： 窗口控制碼

idTimer： 計時器編號

uTimeout： 定時時間（單位：ms）

lpTimerFunc： 當發生計時器事件時需要執行的函數

【返 回 值】計時器編號

【頭 文 件】使用本函數需要包含”window.h”

【說　　明】如果 lpTimerFunc 參數爲 NULL，則當發生計時器事件時將向 hwnd 表單發送 WM_TIMER 事件；如果 lpTimerFunc 不爲 NULL，則當發生計時器事件時將執行由 lpTimerFunc 指

定的函數。

【函數原型】BOOL WINAPI KillTimer(HWND hwnd, UINT idTimer);

【功　　能】刪除指定視窗的計時器

【參　　數】hwnd : 窗口控制碼

idTimer : 計時器編號

【返 回 值】函數執行成功則返回 TRUE；函數執行失敗則返回 FALSE

【頭 文 件】使用本函數需要包含”window.h”

在WM_TIMER消息中，即可通過向靜態控制元發送STM_SETIMAGE消息來更新圖片，從而達到動畫效果。本實驗程式的消息處理函數的流程如 圖 2.12所示。

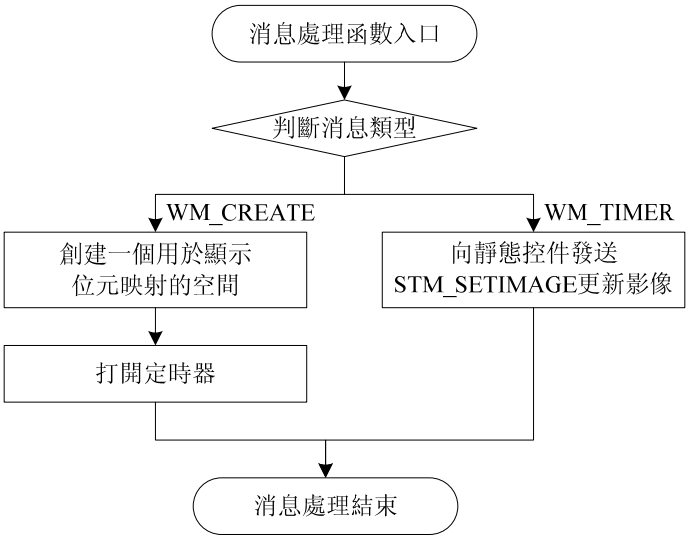

圖 2.12 實驗十四消息處理程式流程圖

【實驗步驟】

1、 打開 S+core IDE，使用“Score IDE MicroWin Project”範本採用預設設置新建一個工程；

2、 使用 convbmp 工具轉換一組連續動作的 bmp 圖像檔案，並將轉換得到的*.c 檔案添加至工程；

3、 在工程嚮導生成的 APPWinMain.c 檔案中按照實驗原理的描述編寫程式；

4、 把 实验十一中的“default_install”檔案夾拷貝到新建的工程目錄下；

5、 修改、編譯（Rebuild All）直到沒有任何錯誤；

6、 利用 RedBoot 下載程式到開發板上並脫機運行；

7、 觀察螢幕上的動畫效果。

Tel:010-62981113-2944/2940/2942/2945 http://www.unsp.com
E-mail:unsp@sunplus.com.cn

【範例路徑】

在大學計畫網站（score.unsp.com）或者 32 位元嵌入式開發系統光碟都會提供本實驗的參考程式，路徑如下：

\Example_eCos\2.GUI_Exa\ex14_Static

实验十五　按鈕控制元實驗

【實驗目的】

1、 掌握按鈕控制元的使用方法

【實驗設備】

1、 裝有 Windows 系統和 S+core IDE 集成開發環境的 PC 機一台；

2、 凌陽 SPCE3200 嵌入式精簡開發板一套；

3、 本實驗用到開發板的介面有： TFT LCD 介面。

【實驗要求】

創建一個主視窗，利用按鈕控制元在主視窗內放置一對單選按鈕、一對複選按鈕、四個組框，並將這些控制元如 圖 2.13所示排列。

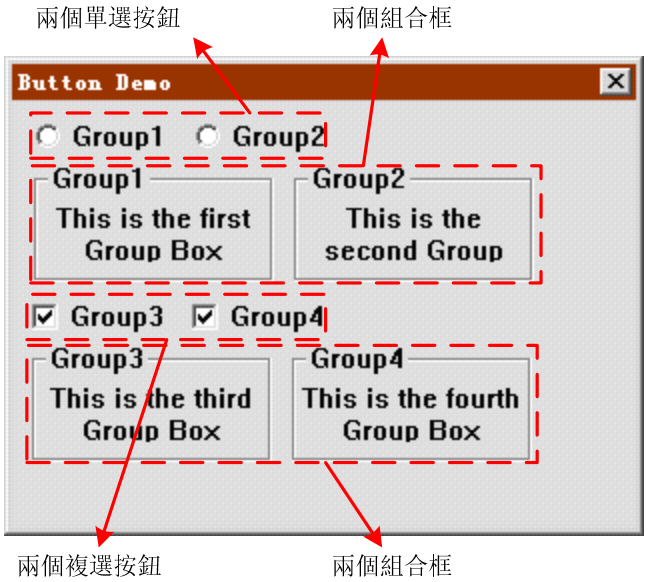

圖 2.13 實驗十五主表單控制元排列示意圖

其中，四個組框初始狀態下不顯示，當點擊“Group1”單選按鈕時，Group1 組框顯示，當點擊“Group2”單選按鈕時，Group2 組框顯示，同時Group1 組框隱藏，從而實現Group1 和Group2 不能同時選中的效果；當選中“Group3”複選按鈕時Group3 組框顯示，選中“Group4”複選按鈕時Group4 組框顯示，從而實現複選按鈕可以同時選中多個的效果。實驗效果如 圖 2.14所示。

Tel:010-62981113-2944/2940/2942/2945 http://www.unsp.com
E-mail:unsp@sunplus.com.cn

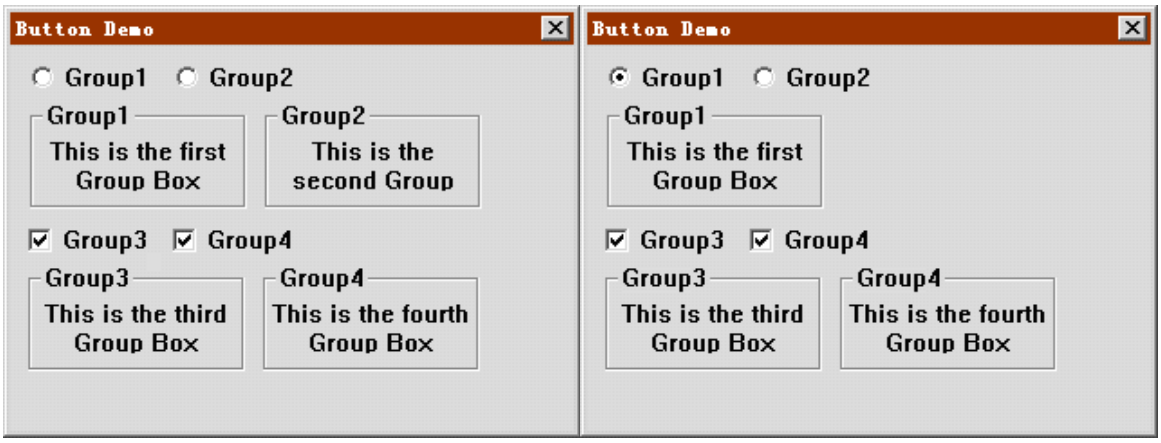

圖 2.14 單選按鈕按下時的示意圖

【實驗原理】

1、 按鈕控制元簡介

按鈕控制元是 MicroWin 應用程式中最常用的控制元之一。按鈕控制元的型別比較豐富，其中主要有普通按鈕、單選按鈕、複選按鈕和組框按鈕等。

1) 普通按鈕與預設普通按鈕

普通按鈕和預設普通按鈕都是最常用的按鈕，其外觀爲矩形條，並可在用戶單擊時產生按下和抬起的動畫效果，按鈕上可設置文本或位元映射等。該型別按鈕的作用是幫助用戶觸發指定動作。當用戶單擊按鈕時，應用程式立即執行相應動作。

2) 單選按鈕與自動單選按鈕

單選按鈕的外形爲按鈕文本和其左側的小圓框，當單選按鈕被選中時，該項的圓框將加點顯示。單選按鈕所包含的各項之間一般存在互斥性，即用戶只能選擇其中某個選項。這種互斥性通常由應用程式來控制，單選按鈕本身不具備自動選中或取消選中的能力。

自動單選按鈕與普通單選按鈕的區別在於：當用戶選擇自動單選按鈕時，系統可自動消除其他單選按鈕的選中旗标，以保證互斥性，而不需要應用程式來控制。

3) 複選按鈕

複選按鈕的外形爲按鈕文本和其左側的小方框，當複選按鈕被選中時，該項的方框將加叉號顯示。

複選按鈕常用來顯示一組選項供用戶選擇。與單選按鈕不同，其各項之前不存在互斥性，用戶可同時選擇其中一個或多個選項。

4) 組框

組框的外形爲左上角包含文字的矩形框。組框是一種 特殊的按鈕形式，雖然它屬於按鈕類控制元，但是它既不處理滑鼠和鍵盤的輸入，也不向其父視窗發送消息，其主要作用在於將控制元分隔成不同的組並加以說明。

2、 按鈕控制元的創建

MicroWin 提供的按鈕控制元的類名爲：“BUTTON”，在使用之前必須使用按鈕控制元註冊函數對其進行註冊，按鈕控制元註冊函數的函數原型如下：

int WINAPI MwRegisterButtonControl(HINSTANCE hInstance);

其中，hInstance 表示當前視窗的實例控制碼，一般爲 NULL；返回值表示是否註冊成功，非零則表示成功，否則表示失敗。失敗的原因可能是因爲已經註冊過，或者系統沒有足夠的記憶體。

在註冊按鈕控制元之後，便可以使用 CreateWindow 函數創建它，如下：

```
hWnd = CreateWindow(“BUTTON”,                    // 預定義按鈕控制元類名
                    NULL,                        // 按鈕控制元一般無標題
                    WS_CHILD | WS_VISIBLE | ..., // 按鈕控制元樣式，樣式說明見 表 2.9
                    X, Y,                        // 按鈕控制元在視窗中的位置
                    nWidth, nHeight,             // 按鈕控制元的寬度和高度
                    hWndParent,                  // 擁有該按鈕控制元的視窗控制碼
                    (HMENU)ID,                   // 按鈕控制元標識
                    hInstance,                   // 應用程式當前實例控制碼，一般爲 0
                    NULL);                       // 按鈕控制元私有資料
```

控制元的視窗屬性必須具有WS_CHILD樣式，一般都具有WS_VISIBLE樣式，同時，可以使用“|”選擇多個 表 2.9所示的擴展樣式。

表 2.9 按鈕控制元常用樣式及說明

樣式	說明
BS_PUSHBUTTON	普通按鈕控制元
BS_DEFPUSHBUTTON	預設的普通按鈕控制元
BS_CHECKBOX	複選按鈕控制元
BS_AUTOCHECKBOX	自動複選按鈕控制元
BS_RADIOBUTTON	單選按鈕控制元
BS_AUTORADIOBUTTON	自動單選按鈕控制元

BS_GROUPBOX	組框控制元
BS_TEXT	表示按鈕上存在文本
BS_ICON	表示按鈕上帶有圖示（同 BS_BITMAP）
BS_BITMAP	表示按鈕上帶有位元映射（同 BS_ICON）
BS_CENTER	按鈕上的文本居中
BS_LEFT	按鈕上的文本左對齊
BS_RIGHT	按鈕上的文本右對齊
BS_USERBUTTON	用戶定義按鈕

注：BS_PUSHBUTTON、BS_DEFPUSHBUTTON、BS_CHECKBOX、BS_AUTOCHECKBOX、BS_RADIOBUTTON、BS_AUTORADIOBUTTON 和 BS_GROUPBOX 只能選擇其中一種樣式，不能使用“|”選擇多個。

如果用戶爲按鈕控制元選擇了 BS_ICON 或 BS_BITMAP 樣式，則 CreateWindow 函數的最後一個參數(按鈕控制元私有資料指標)將用來表示需要顯示的 DIB 位元映射。用戶可以在 CreateWindow 時爲其指定 DIB 位元映射，或者通過一個已經存在的按鈕控制元發送 BM_SETIMAGE 來設置位元映射。

3、 按鈕控制元與應用程式之間的消息傳遞

應用程式創建按鈕控制元後，可通過接收控制元發出的消息得知用戶的輸入，並可通過向按鈕控制元發送消息對其進行操作。

1) 按鈕控制元向應用程式發送消息

與靜態控制元一樣，當用戶與按鈕控制元交互時，按鈕控制元將向其父視窗發送 WM_COMMAND 消息，該消息的 lParam 參數保存了按鈕控制元的控制碼；wParam 參數的低位元組爲控制元標識，高位元組爲按鈕控制元動作的消息通知碼。MicroWin 目前僅支援 BN_CLICKED 消息。當用戶單擊按鈕時，控制元將發送該消息給父視窗。

2) 應用程式向按鈕控制元發送消息

應用程式對按鈕控制元的操作通過調用函數SendMessage或PostMessage向其發送各種消息來完成。表 2.10列出了MicroWin支援的常用的按鈕控制元控制消息及說明。

表 2.10 按鈕控制元常用控制消息及說明

控制消息	說明
BM_GETCHECK	獲取單選按鈕或複選按鈕的選中狀態 參數：無

	返回值：0 表示未選中；1 表示選中
BM_SETCHECK	設置單選按鈕或複選按鈕的選中狀態 參數： wParam：選中狀態（0 表示取消選中，1 表示選中） 返回值：無
BM_GETSTATE	獲取按鈕狀態 參數：無 返回值：非零表示按鈕處於按下狀態，爲零表示沒有被按下
BM_SETSTATE	設置按鈕狀態 參數： wParam：非零則表示將按鈕設置爲按下狀態，否則取消按鈕的按下狀態 返回值：無
BM_GETIMAGE	設置按鈕控制元顯示的位元映射 參數： wParam : 新的 DIB 位元映射資料指標 返回值：舊的 DIB 位元映射數據指標
BM_SETIMAGE	獲取按鈕控制元當前顯示的位元映射 參數：無 返回值：DIB 位元映射數據指標

4、 本實驗的實驗原理

在本實驗中，首先使用 MicroWin 應用程式框架創建一個主視窗，並在 WM_CREATE 消息中按照實驗要求創建各個按鈕控制元。這裏建議選擇 BS_AUTORADIOBUTTON 和 BS_AUTOCHECKBOX 創建單選按鈕和複選按鈕，這樣系統可以幫助我們完成單擊之後的選中或取消選中的效果。

當用戶單擊單選按鈕或複選按鈕時，我們需要在WM_COMMAND消息中根據各自的選中狀態來修改四個組框的顯示或隱藏狀態。顯示/隱藏效果的實現可以通過調用ShowWindow函數來實現。本實驗程式的消息處理函數的流程如 圖 2.15所示。

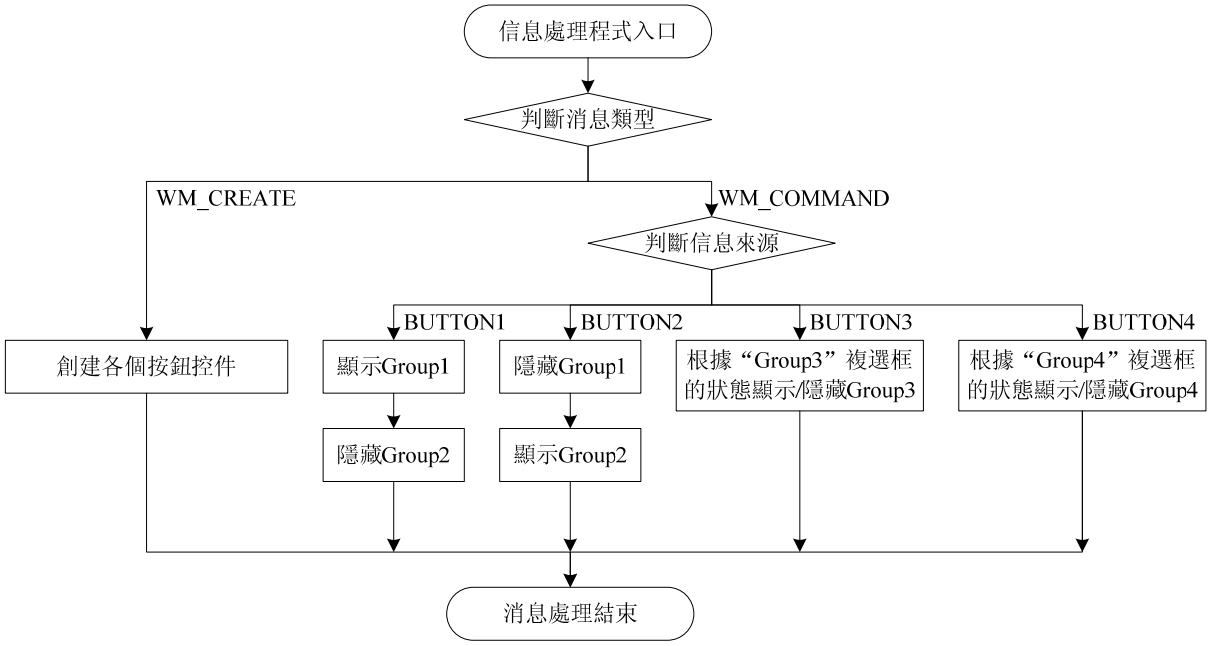

圖 2.15 實驗十五消息處理程式流程圖

【實驗步驟】

1、 打開 S+core IDE，使用“Score IDE MicroWin Project”範本採用預設設置新建一個工程；

2、 在工程嚮導生成的 APPWinMain.c 檔案中按照實驗原理的描述編寫程式；

3、 把上個實驗中的“default_install”檔案夾拷貝到新建的工程目錄下；

4、 修改、編譯（Rebuild All）直到沒有任何錯誤；

5、 利用 RedBoot 下載程式到開發板上脫機運行；

6、 點擊單選按鈕和複選按鈕，觀察實驗現象。

【範例路徑】

在大學計畫網站（score.unsp.com）或者 32 位元嵌入式開發系統光碟都會提供本實驗的參考程式，路徑如下：

\Example_eCos\2.GUI_Exa\ex15_Button

实验十六　文本框控制元實驗

【實驗目的】

1、 掌握文本框控制元的使用方法；

【實驗設備】

1、 裝有 Windows 系統和 S+core IDE 集成開發環境的 PC 機一台；

2、 凌陽 SPCE3200 嵌入式精簡開發板一套；

3、 本實驗用到開發板的介面有： UART 介面，TFT LCD 介面。

【實驗要求】

使用MicroWin製作一個“乘法器”。如 圖 2.16所示，用戶可以通過串口類比鍵盤在“Multiplier”框中輸入一個數，然後單擊“X”按鈕，再在“Multiplier”框中輸入另一個數位，單擊“X”按鈕後在“Result”框可以看到兩次輸入的數值相乘運算之後的結果，再次在“Multiplier”框中輸入數位並單擊“X”按鈕後，在“Result”中可以看到本次輸入與上次的結果累乘之後的結果，依次類推……

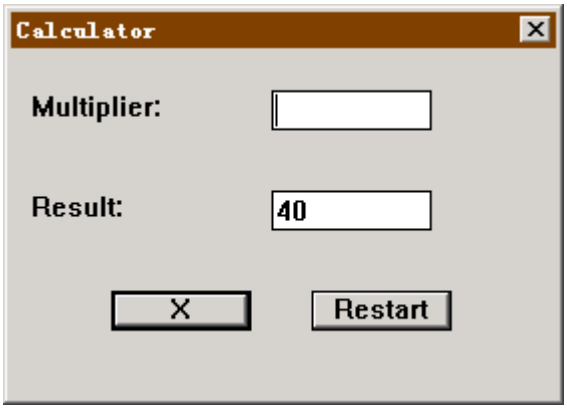

圖 2.16 利用文本框控制元編制“乘法運算器”介面

【實驗原理】

1、 文本框控制元簡介

文本框控制元的主要作用在於接收用戶的鍵盤輸入。用戶可在文本框控制元中編輯輸入文本。文本框控制元是 MicroWin 應用程式中一種重要的交互工具，也是最常用的交互控制元之一。

文本框控制元通過選擇相應的樣式可具有以下功能特點：

- 可自動轉換用戶輸入，如大小寫轉換等
- 可使用某個字元（如*）代替顯示用戶的輸入

Tel:010-62981113-2944/2940/2942/2945 http://www.unsp.com
E-mail:unsp@sunplus.com.cn

■ 可支援多行編輯

2、 文本框控制元的創建

MicroWin 提供了兩種文本框控制元，它們的類名分別是：“EDIT”和“MEDIT”。其中，“EDIT”只能用來編輯單行文本，而“MEDIT”可以設置多行編輯樣式，從而用來編輯多行文本。

在創建文本框控制元之前，需要首先對文本框控制元進行註冊，針對“EDIT”和“MEDIT”兩種不同的文本框，MicroWin 系統提供了兩個不同的函數來進行註冊，他們分別是：

int WINAPI MwRegisterEditControl(HINSTANCE hInstance); // 用於註冊“EDIT”文本框控制元

int WINAPI MwRegisterMEditControl(HINSTANCE hInstance);// 用於註冊“MEDIT”文本框控制元

其中，hInstance 表示當前視窗的實例控制碼，一般爲 NULL；返回值表示是否註冊成功，非零則表示成功，否則表示失敗。失敗的原因可能是因爲已經註冊過，或者系統沒有足夠的記憶體。

在註冊文本框控制元之後，可以使用 CreateWindow 函數創建一個單行或多行的文本框，如下：

創建單行文本框：

```
hWnd = CreateWindow(“EDIT”,                    // 預定義文本框視窗類名
                    NULL,                      // 文本框一般無標題
                    WS_CHILD | WS_VISIBLE | ...,  // 樣式，常用樣式說明見 表 2.11
                    X, Y,                      // 文本框在視窗中的位置
                    nWidth, nHeight,           // 文本框的寬度和高度
                    hWndParent,                // 擁有該文本框的視窗控制碼
                    (HMENU)ID,                 // 文本框標識
                    hInstance,                 // 應用程式當前實例控制碼，一般爲 0
                    NULL);                     // 私有數據
```

創建多行文本框：

```
hWnd = CreateWindow(“EDIT”,                    // 預定義文本框視窗類名
                    NULL,                      // 文本框一般無標題
                    WS_CHILD | WS_VISIBLE | ES_MULTILINE ...,
                                               // 樣式，常用樣式說明見 表 2.11
                    X, Y,                      // 文本框在視窗中的位置
                    nWidth, nHeight,           // 文本框的寬度和高度
                    hWndParent,                // 擁有該文本框的視窗控制碼
```

```
                    (HMENU)ID,              // 文本框標識
                    hInstance,              // 應用程式當前實例控制碼，一般爲 0
                    NULL);                  // 私有數據
```

表 2.11 文本框控制元常用樣式及說明

樣式	說明
ES_LEFT	文本框內的文字左對齊
ES_CENTER	文本框內的文字居中對齊
ES_RIGHT	文本框內的文字右對齊
ES_MULTILINE	文本框支援多行編輯（僅對“MEDIT”控制元有效）
ES_UPPERCASE	自動將輸入轉換爲大寫
ES_LOWERCASE	自動將輸入轉換爲小寫
ES_PASSWORD	將輸入替換成“*”顯示
ES_AUTOVSCROLL	當用戶輸入超出視窗時，視窗自動垂直滾動
ES_AUTOHSCROLL	當用戶輸入超出視窗時，視窗自動水準滾動
ES_NOHIDESEL	當文本框失去焦點時仍保持被選中文本的狀態
ES_READONLY	設置文本框唯讀
ES_WANTRETURN	設置該屬性後在按下回車時文本框將自動換行，而不是將其做爲一個字元處理
ES_NUMBER	文本框只接受數位

注：ES_CENTER 和 ES_RIGHT 在目前版本的 MicroWin 中沒有實現，其效果與 ES_LEFT 相同。

3、 文本框控制元與應用程式之間的消息傳遞

應用程式創建文本框控制元後，可通過接收控制元發出的消息得知用戶的請求，並可通過向文本框控制元發送消息對其進行操作。“EDIT”控制元和“MEDIT”控制元的消息是一致的，下面列出的表格對二者基本都適用，個別區別會在表格的備註中標明。

1) 文本框控制元向應用程式發送消息

當文本框控制元的狀態發生變化時，會通過向父表單發送WM_COMMAND消息通知應用程式來處理。文本框控制元發送的這個WM_COMMAND消息的lParam參數保存了文本框的控制碼；wParam參數的低位元組爲控制元標識，高位元組爲文本框控制元動作的消息通知碼。常用的通知碼及其說明可以參考 表 2.12。

表 2.12 文本框控制元常用通知碼及說明

通知碼	說明	通知碼	說明
EN_SETFOCUS	文本框獲得焦點	EN_MAXTEXT	用戶輸入達到允許的最大位元組數
EN_KILLFOCUS	文本框失去焦點	EN_HSCROLL	文本框的內容水準滾動
EN_CHANGE	文本框的內容發生變化	EN_VSCROLL	文本框的內容垂直滾動
EN_UPDATE	文本框的內容被更新		

2) 應用程式向文本框控制元發送消息

應用程式對文本框控制元的操作通過調用函數SendMessage或PostMessage向其發送各種消息來完成。表 2.13列出了MicroWin支援的常用的文本框控制元控制消息及說明。

表 2.13 文本框控制元常用控制消息及說明

控制消息	說明
EM_LIMITTEXT	設置文本框中文本的最大長度。當用戶輸入的文本超出最大長度時僅保留最大長度之前的字元 參數： wParam：最大長度（wParam 應小於等於 3000） 返回值：無
EM_SETPASSWORDCHAR	設置新的密碼字元。當文本框應用 ES_PASSWORD 樣式時，用戶輸入的字元將使用密碼字元顯示。文本框的密碼字元預設爲“*” 參數： wParam：新的密碼字元 返回值：無
EM_SETREADONLY	更改文本框的唯讀屬性 參數： wParam：爲 0 時去除文本框的唯讀屬性，非 0 時文本框唯讀 返回值：無
EM_GETPASSWORDCHAR	得到文本框當前使用的密碼字元 參數： lParam：用於保存密碼字元 返回值：無

EM_SETLIMITTEXT	同 EM_LIMITTEXT
EM_GETLIMITTEXT	得到文本框當前的最大限制長度 參數： lParam： 用於保存最大限制長度 返回值：無

4、 本實驗的實驗原理

在本實驗中，與其他的 MicroWin 實驗一樣，首先使用 MicroWin 應用程式框架創建一個主視窗，並在 WM_CREATE 消息中完成在視窗上創建控制元的操作。

程式的主要功能在“X”按鈕的單擊事件中完成，這裏需要完成文本框內容的讀取，並轉換爲數位，然後進行乘法運算，並輸出到保存結果的文本框中進行顯示。獲取文本框中用戶輸入的文本可以使用 GetWindowText 函數，更新文本框中的結果可以使用 SetWindowText 函數。

程式的消息處理函數的流程如 圖 2.17所示。

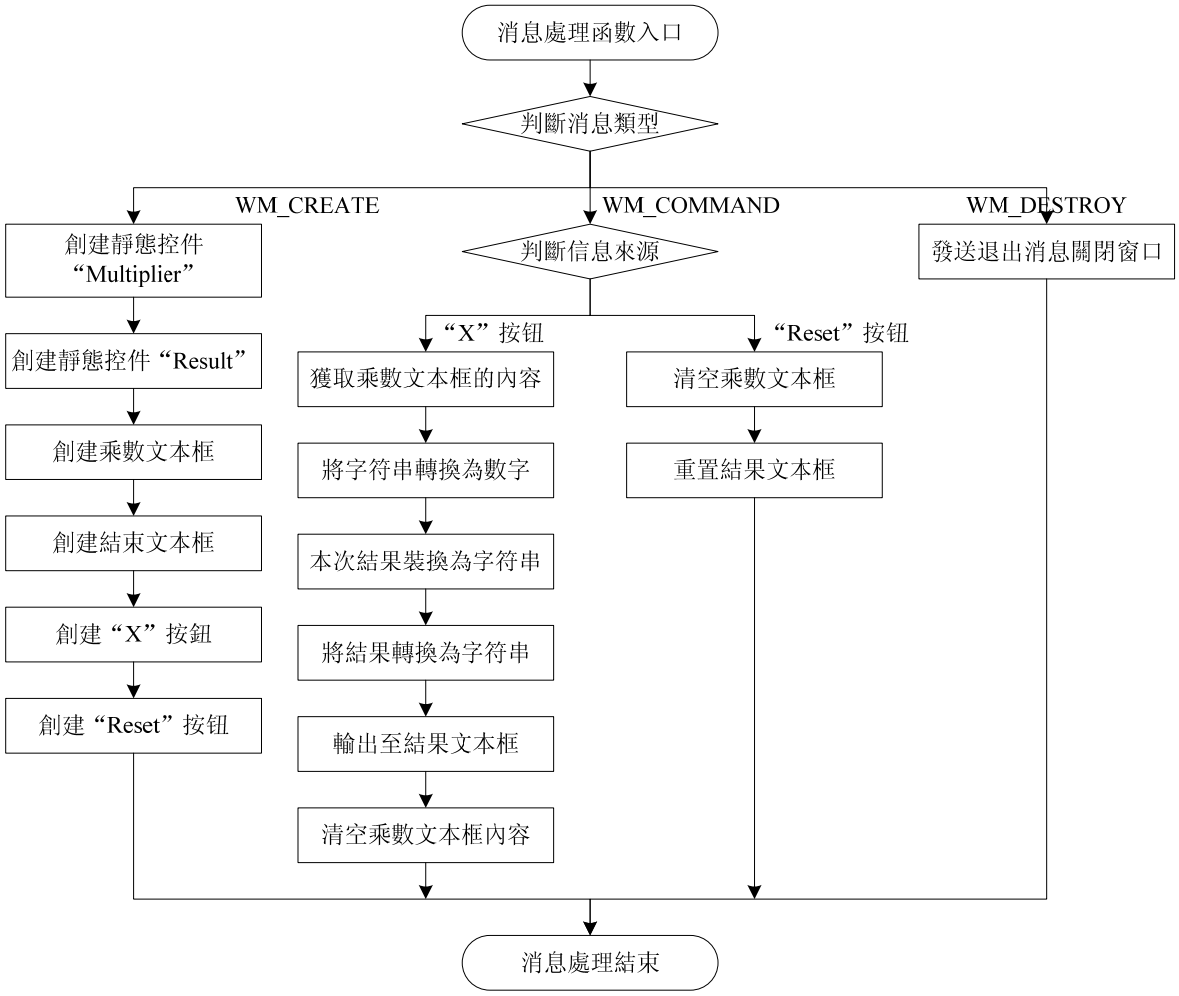

圖 2.17 實驗十六消息處理程式流程圖

Tel:010-62981113-2944/2940/2942/2945 http://www.unsp.com
E-mail:unsp@sunplus.com.cn

【實驗步驟】

1、 打開 S+core IDE，使用“Score IDE MicroWin Project”範本採用預設設置新建一個工程；

2、 在工程嚮導生成的 APPWinMain.c 檔案中按照實驗原理的描述編寫程式；

3、 把上個實驗範例中的“default_install”檔案夾拷貝到新建的工程目錄下；

4、 修改、編譯（Rebuild All）直到沒有任何錯誤；

5、 使開發板上的撥碼開關 S2 的 3 和 4 都置於 ON 的狀態；

6、 使用串口線連接開發板和 PC，並在 PC 端打開串口除錯助手或超級終端軟體，設置串列傳輸速率 115200，資料位元 8 位元，停止位 1 位，無校驗位；

7、 利用 RedBoot 下載程式到開發板上脫機運行；

8、 在串口除錯助手或超級終端軟體中輸入數位，可以看到在 Multiplier 文本框中出現輸入的字元，然後按下“X”按鈕，Result 文本框中將顯示結果。

【範例路徑】

在大學計畫網站（score.unsp.com）或者 32 位元嵌入式開發系統光碟都會提供本實驗的參考程式，路徑如下：

\Example_eCos\2.GUI_Exa\ex16_Edit

实验十七　列表框控制元實驗

【實驗目的】

1、 掌握列表框控制元的使用方法；

【實驗設備】

1、 裝有 Windows 系統和 S+core IDE 集成開發環境的 PC 機一台；

2、 凌陽 SPCE3200 嵌入式精簡開發板一套；

3、 本實驗用到開發板的介面有：TFT LCD 介面。

【實驗要求】

創建一個主視窗，並在主視窗上放置兩個列表框控制元，然後添加兩個按鈕“ADD”和“DEL”，如 圖 2.18所示。左側列表框中初始有一些字串，當點擊ADD按鈕時，實現將左側列表框中被選中的項添加至右側的列表框中；當點擊DEL按鈕時，實現將右側列表框中被選中的項刪除。

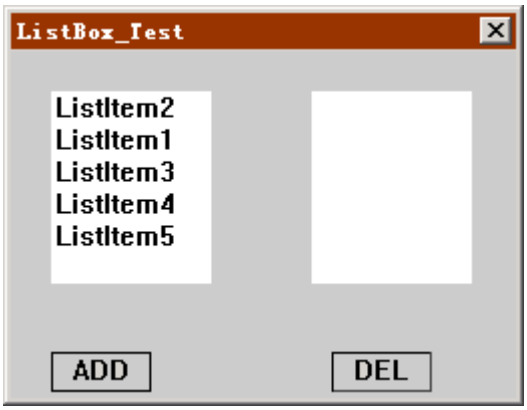

圖 2.18 實驗十七主表單介面

【實驗原理】

1、 列表框控制元簡介

列表框控制元常用于集中顯示同種型別的一系列內容以供用戶選擇。列表框一般具有如下特點：

- 可提供大量的可選項（需要時自動顯示捲軸）
- 可設置單選（單個選項）或多選（多項選擇）功能
- 單選時，單擊列表框，被選的項以反色顯示表示被選中；再次單擊該選項，恢復爲非選中狀態

Tel:010-62981113-2944/2940/2942/2945 http://www.unsp.com
E-mail:unsp@sunplus.com.cn

2、 列表框控制元的創建

MicroWin 提供的列表框控制元的類名爲：“LISTBOX”，在使用之前必須使用列表框控制元註冊函數對其進行註冊，列表框控制元註冊函數的函數原型如下：

int WINAPI MwRegisterListboxControl(HINSTANCE hInstance);

其中，hInstance 表示當前視窗的實例控制碼，一般爲 NULL；返回值表示是否註冊成功，非零則表示成功，否則表示失敗。失敗的原因可能是因爲已經註冊過，或者系統沒有足夠的記憶體。

在註冊列表框控制元之後，可以使用 CreateWindow 函數創建它：

```
hWnd = CreateWindow(“LISTBOX”,                  // 預定義列表框窗口類名
                    NULL,                       // 列表框一般無標題
                    WS_CHILD | WS_VISIBLE | ..., // 樣式，常用樣式說明見 表 2.14
                    X, Y,                       // 列表框在視窗中的位置
                    nWidth, nHeight,            // 列表框的寬度和高度
                    hWndParent,                 // 擁有該列表框的窗口控制碼
                    (HMENU)ID,                  // 列表框標識
                    hInstance,                  // 應用程式當前實例控制碼，一般爲 0
                    NULL);                      // 私有數據
```

表 2.14 列表框控制元常用樣式及說明

樣式	說明
LBS_NOTIFY	列表框可向父視窗發出消息
LBS_SORT	按字母順序排列表項
LBS_MULTIPLESEL	允許選擇多項
LBS_STANDARD	標準樣式，即同時具有 LBS_NOTIFY，LBS_SORT，WS_VSCROLL，WS_BORDER 樣式

3、 列表框控制元與應用程式之間的消息傳遞

應用程式創建列表框控制元後，可通過接收控制元發出的消息得知用戶的請求，並可通過向列表框控制元發送消息對其進行操作。

1) 列表框控制元向應用程式發送消息

當用戶與列表框交互時，即當列表框控制元的狀態發生變化時，會通過向父表單發送WM_COMMAND消息通知應用程式來處理。該消息的lParam參數保存了列表框控制元的控制碼；wParam參數的低位元組爲控制元標識，高位元組爲列表框控制元動作的消息通知碼。常用的通知碼及其說明可以參考 表 2.15。

表 2.15 列表框控制元常用通知碼及說明

通知碼	說明	通知碼	說明
LBN_ERRSPACE	記憶體不足，無法添加新項	LBN_SETFOCUS	列表框得到焦點
LBN_SELCHANGE	用戶的選擇發生改變	LBN_KILLFOCUS	列表框失去焦點

2) 應用程式向列表框控制元發送消息

應用程式對列表框控制元的操作通過調用函數SendMessage或PostMessage向其發送各種消息來完成。表 2.16列出了MicroWin支援的常用的列表框控制元控制消息及說明。

表 2.16 列表框控制元常用控制消息及說明

控制消息	說明
LB_ADDSTRING	向列表框添加項 參數： lParam：新添加的項的名稱字串指標 返回值：新添加的項在列表中的索引值（小於零表示添加失敗）
LB_DELETESTRING	刪除指定項 參數： wParam：需要刪除的項在列表中的索引值 返回值：無
LB_FINDSTRING	在列表中查找以指定字串開始的項 參數： wParam：開始查找的項索引值 lParam：要查找的項名稱字串指標 返回值：項在列表中的索引值（小於零表示查找的字串不存在）
LB_GETCOUNT	得到列表框當前的項數 參數：無 返回值：當前項數
LB_GETCURSEL	獲取當前被選中的項的索引值 參數：無 返回值：項索引值（小於零表示沒有項被選中）
LB_GETSEL	獲取指定的項的選中狀態 參數：

Tel:010-62981113-2944/2940/2942/2945 http://www.unsp.com
E-mail:unsp@sunplus.com.cn

	wParam：需要獲取狀態的項的索引值 返回值：非零表示該項被選中，否則表示未被選中
LB_GETSELCOUNT	獲取多選列表框中選中的項數 參數：無 返回值：被選中的項數
LB_GETTEXT	獲取指定項的名稱 參數： wParam：項的索引值 lParam：用於保存項名稱的緩衝區指標 返回值：無
LB_GETTEXTLEN	獲取指定項的名稱的長度 參數： wParam：項的索引值 返回值：項的名稱的長度（單位：位元組）
LB_GETTOPINDEX	獲取列表框中第一項的索引值 參數：無 返回值：項的索引值
LB_INSERTSTRING	在列表框的指定位置添加項 參數： wParam：新添加的項的位置 lParam：新添加的項的名稱字串指標 返回值：新添加的項在列表中的索引值（小於零表示添加失敗）
LB_RESETCONTENT	清空列表框 參數：無 返回值：無
LB_SETSEL	設置多選列表框中指定項的選中狀態 參數： wParam：-1 表示反選該項 0 表示取消該項的選擇 1 表示選中該項

	lParam：需要修改狀態的項的索引值 返回值：無
LB_SETCURSEL	設置單選列表框中的指定項爲當前選中的項 參數： wParam：新的被選中的項的索引值 返回值：舊的被選中的項的索引值
LB_SETTOPINDEX	設置列表框中顯示的第一項的索引值 參數： wParam：項的索引值 返回值：無

應用程式向列表框控制元發送消息時應注意以下問題：

a) 索引號的使用

應用程式通過索引值操作指定項，其中第一項的預設索引號爲 0。例如，在列表框的第 3 個位置插入項的語句爲：

SendMessage(hwnd, LB_INSERTSTRING, 2, (LPARAM)szNewItem);

應用程式常通過項列表框發送 LB_GETCURSEL 消息獲取當前選中的項的索引號，形式爲：

nIndex = SendMessage(hwnd, LB_GETCURSEL, 0, 0); // nIndex 即爲當前被選中的項的索引值

b) 多選列表框消息發送

當應用程式設置列表框的樣式爲 LBS_MULTIPLESEL 時，用戶可在該列表框中選取多個項。當用戶選擇列表框中的多個項時，應用程式可通過向列表框發送 LB_GETSELCOUNT 消息獲取選中的項數，其形式爲：

nCount = SendMessage(hwnd, LB_GETSELCOUNT, 0, 0); // nCount 即爲當前被選中的項的數量

此外，應用程式還可以向列表框發送 LB_GETSEL 消息獲取指定項的選中狀態，以此來判斷哪些項被選中，其形式爲：

nSelect = SendMessage(hwnd, LB_GETSEL, nIndex, 0); // nSelect 非零即表示 nIndex 項被選中

4、 本實驗的實驗原理

在本實驗中，與其他的 MicroWin 實驗一樣，首先使用 MicroWin 應用程式框架創建一個主視窗，並在 WM_CREATE 消息中在視窗上創建兩個列表框控制元和兩個按鈕控制元。

在創建完畢控制元之後，還需要初始化第一個列表框，即向其添加一些條目，以便後續操作。初始化工作可以在 WM_CREATE 消息中，創建列表框之後來完成。

本實驗程式的消息處理函數的流程如 圖 2.19所示。

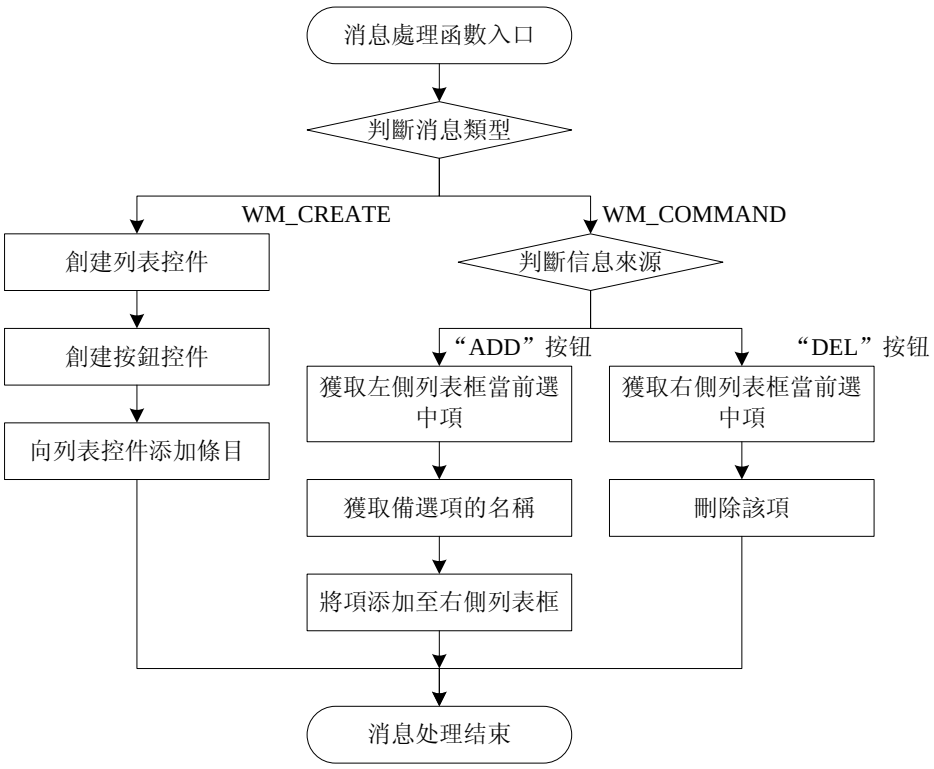

圖 2.19 實驗十七消息處理程式流程圖

【實驗步驟】

1、 打開 S+core IDE，使用“Score IDE MicroWin Project”範本採用預設設置新建一個工程；

2、 在工程嚮導生成的 APPWinMain.c 檔案中按照實驗原理的描述編寫程式；

3、 把上個實驗範例中的“default_install”檔案夾拷貝到新建的工程目錄下；

4、 修改、編譯（Rebuild All）直到沒有任何錯誤；

5、 利用 RedBoot 下載程式到開發板上脫機運行；

6、 通過點擊“ADD”和“DEL”按鈕操作列表框，觀察實驗現象。

【範例路徑】

在大學計畫網站（score.unsp.com）或者 32 位元嵌入式開發系統光碟都會提供本實驗的參考程式，路徑如下：

\Example_eCos\2.GUI_Exa\ex17_ListBox

实验十八　下拉式列示方塊控制元實驗

【實驗目的】

1、 掌握下拉式列示方塊控制元的使用方法；

【實驗設備】

1、 裝有 Windows 系統和 S+core IDE 集成開發環境的 PC 機一台；

2、 凌陽 SPCE3200 嵌入式精簡開發板一套；

3、 本實驗用到開發板的介面有：TF LCD 介面。

【實驗要求】

創建一個主視窗，並在主視窗上放置一個下拉式列示方塊控制元，然後添加一個用於顯示文本的靜態控制元。如 圖 2.20所示。編寫程式使靜態控制元的顯示內容隨著下拉式列示方塊控制元中的選擇的變化而變化。

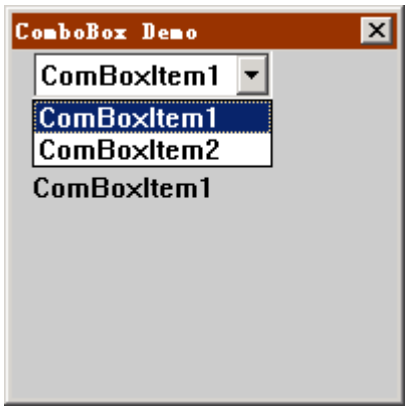

圖 2.20 實驗十八主表單介面

【實驗原理】

1、 下拉式列示方塊控制元簡介

下拉式列示方塊是兩種預定義的視窗的組合樣式。在 MicroWin 系統中使用單一控制元往往不能完全滿足與用戶交互的需要，最常見的下拉式列示方塊例子是列表框控制元與靜態文本或文本框相關聯，實現讓用戶從預定列表中選擇一個值並可以編輯的效果。MicroWin 系統提供的預定義下拉式列示方塊便是這種組合。下拉式列示方塊中的列表框常以隱藏的形式出現在文本框下，當用戶單擊文本框右側的箭頭時將彈出列表框以供選擇，並在用戶完成選擇之後再次隱藏。

2、 下拉式列示方塊控制元的創建

Tel:010-62981113-2944/2940/2942/2945 http://www.unsp.com
E-mail:unsp@sunplus.com.cn

MicroWin 提供的下拉式列示方塊控制元的類名爲：“COMBOBOX”，在使用之前必須使用下拉式列示方塊控制元註冊函數對其進行註冊，下拉式列示方塊控制元註冊函數的函數原型如下：

int WINAPI MwRegisterComboboxControl(HINSTANCE hInstance);

其中，hInstance 表示當前視窗的實例控制碼，一般爲 NULL；返回值表示是否註冊成功，非零則表示成功，否則表示失敗。失敗的原因可能是因爲已經註冊過，或者系統沒有足夠的記憶體。

另外，由於下拉式列示方塊實際是列表框和文本框的組合，所以，在創建下拉式列示方塊之前還必須使用列表框和文本框的註冊函數將二者註冊。

在註冊列表框控制元和相關控制元之後，便可以使用 CreateWindow 函數創建它：

```
hWnd = CreateWindow(“COMBOBOX”,                  // 預定義下拉式列示方塊窗口類名
                    NULL,                        // 下拉式列示方塊一般無標題
                    WS_CHILD | WS_VISIBLE | ..., // 樣式，常用樣式說明見 表 2.17
                    X, Y,                        // 下拉式列示方塊在視窗中的位置
                    nWidth, nHeight,             // 下拉式列示方塊的寬度和高度
                    hWndParent,                  // 擁有該下拉式列示方塊的窗口控制碼
                    (HMENU)ID,                   // 下拉式列示方塊標識
                    hInstance,                   // 應用程式當前實例控制碼，一般爲 0
                    NULL);                       // 私有數據
```

需要注意的是，創建下拉式列示方塊生成的控制碼不能操作下拉式列示方塊中的任一部分，只能操作整個下拉式列示方塊。

表 2.17 下拉式列示方塊控制元常用樣式及說明

樣式	說明
CBS_SIMPLE	下拉式列示方塊中的列表框可見
CBS_DROPDOWN	下拉式列示方塊由列表框和文本框組成，列表框平時不可見
CBS_DROPDOWNLIST	下拉式列示方塊由列表框和靜態文本組成，列表框平時不可見
CBS_AUTOHSCORLL	文本框中自動水準滾動
CBS_SORT	列表框中各項按字母順序排列

注：CBS_SIMPLE、CBS_DROPDOWN 和 CBS_DROPDOWNLIST 只能選擇其中一種樣式，不能使用“|”選擇多個。

3、 下拉式列示方塊控制元與應用程式之間的消息傳遞

應用程式創建下拉式列示方塊控制元後，可通過接收控制元發出的消息得知用戶的請求，並可通過向下拉式列示方塊發送消息對其進行操作。

1) 下拉式列示方塊控制元向應用程式發送消息

當用戶與下拉式列示方塊交互時，即當下拉式列示方塊的狀態發生變化時，下拉式列示方塊會通過向父表單發送WM_COMMAND消息通知應用程式來處理。該消息的lParam參數保存了下拉式列示方塊的控制碼；wParam參數的低位元組爲控制元標識，高位元組爲下拉式列示方塊動作的消息通知碼。常用的通知碼及其說明可以參考 表 2.18。

表 2.18 下拉式列示方塊控制元常用通知碼及說明

通知碼	說明	通知碼	說明
CBN_SELCHANGE	下拉式列示方塊中列表框的選擇發生改變	CBN_EDITCHANGE	下拉式列示方塊中的文本框中的內容發生改變
CBN_SETFOCUS	下拉式列示方塊得到焦點	CBN_DROPDOWN	下拉式列示方塊中的列表框將顯示
CBN_KILLFOCUS	下拉式列示方塊失去焦點	CBN_CLOSEUP	下拉式列示方塊中的列表框將隱藏

2) 應用程式向下拉式列示方塊控制元發送消息

應用程式對下拉式列示方塊的操作通過調用函數SendMessage或PostMessage向其發送各種消息來完成，對下拉式列示方塊的操作實際是對下拉式列示方塊中各成員的操作。 表 2.19列出了MicroWin支援的常用的列表框控制消息及說明。

表 2.19 下拉式列示方塊控制元常用控制消息及說明

控制消息	說明
CB_SHOWDROPDOWN	顯示列表框 參數： wParam：是否顯示列表框（0：隱藏，1：顯示） 返回值：無
CB_ADDSTRING	向下拉式列示方塊中的列表框添加項 參數： lParam：新添加的項的名稱字串指標 返回值：新添加的項在列表中的索引值（小於零表示添加失敗）
CB_DELETESTRING	刪除下拉式列示方塊中的列表框中的指定項 參數：

Tel:010-62981113-2944/2940/2942/2945 http://www.unsp.com
E-mail:unsp@sunplus.com.cn

	wParam : 需要刪除的項在列表中的索引值 返回值：無
CB_INSERTSTRING	在下拉式列示方塊中的列表框的指定位置添加項 參數： wParam：新添加的項的位置 lParam：新添加的項的名稱字串指標 返回值：新添加的項在列表中的索引值（小於零表示添加失敗）
CB_FINDSTRING	在下拉式列示方塊中的列表框中查找以指定字串開始的項 參數： wParam：開始查找的項索引值 lParam : 要查找的項名稱字串指標 返回值：項在列表中的索引值（小於零表示查找的字串不存在）
CB_RESETCONTENT	清空下拉式列示方塊中的列表框 參數：無 返回值：無
CB_SETCURSEL	設置下拉式列示方塊中的列表框的指定項爲當前選中項，並在文本框中顯示 參數：wParam：新的被選中的項的索引值 返回值：被選中的項的索引值
CB_GETCURSEL	獲取下拉式列示方塊中的列表框當前被選中的項的索引值 參數：無 返回值：項索引值（小於零表示沒有項被選中）
CB_GETCOUNT	得到下拉式列示方塊中的列表框當前的項數 參數：無 返回值：當前項數
CB_GETLBTEXT	獲取下拉式列示方塊中的列表框指定項的名稱 參數： wParam：項的索引值 lParam：用於保存項名稱的緩衝區指標 返回值：無
CB_GETLBTEXTLEN	獲取下拉式列示方塊中的列表框指定項的名稱的長度

	參數：wParam：項的索引值 返回值：項的名稱的長度（單位：位元組）
CB_LIMITTEXT	設置下拉式列示方塊中的文本框的字串的最大長度 參數：wParam： 最大長度（單位：位元組） 返回值：無

應用程式可以通過向下拉式列示方塊發送 CB_GETCURSEL 獲取列表框的當前選擇項，並通過發送 CB_GETLBTEXT 來獲得當前選擇的文本，如下所示。

```
int nCurSel;
char szTitle[255];
nCurSel = SendMessage(hComboBox, CB_GETCURSEL, 0, 0);
if(nCurSel >= 0)
{
    SendMessage(hComboBox, CB_GETLBTEXT, nCurSel, (LPARAM)szTitle);
}
```

另外，用戶也可以通過調用 GetWindowText 函數來獲取下拉式列示方塊當前選擇的文本，如下所示。

```
char szTitle[255];
GetWindowText(hComboBox, szTitle, 255);
```

4、 本實驗的實驗原理

在本實驗中，與其他的 MicroWin 實驗一樣，首先使用 MicroWin 應用程式框架創建一個主視窗，並在 WM_CREATE 消息中在視窗上創建一個下拉式列示方塊控制元和一個用於顯示文本的靜態控制元。在創建完畢控制元之後，需要向下拉式列示方塊添加一些條目，以便後續操作。

本實驗程式的消息處理函數的流程如 圖 2.21所示。

Tel:010-62981113-2944/2940/2942/2945 http://www.unsp.com
E-mail:unsp@sunplus.com.cn

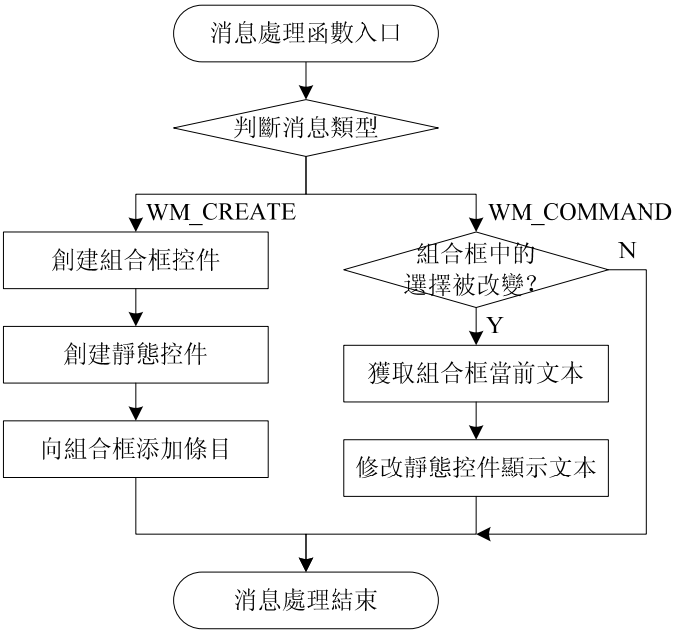

圖 2.21 實驗十八消息處理程式流程圖

【實驗步驟】

1、 打開 S+core IDE，使用“Score IDE MicroWin Project”範本採用預設設置新建一個工程；

2、 在工程嚮導生成的 APPWinMain.c 檔案中按照實驗原理的描述編寫程式；

3、 把上個實驗範例中的“default_install”檔案夾拷貝到新建的工程目錄下；

4、 修改、編譯（Rebuild All）直到沒有任何錯誤；

5、 利用 RedBoot 下載程式到開發板上脫機運行；

6、 在下拉式列示方塊中選擇不同的條目，觀察靜態控制元中的文本的變化。

【範例路徑】

在大學計畫網站（score.unsp.com）或者 32 位元嵌入式開發系統光碟都會提供本實驗的參考程式，路徑如下：

\Example_eCos\2.GUI_Exa\ex18_ComboBox

实验十九　進度條控制元實驗

【實驗目的】

1、 掌握進度條控制元的使用方法；

【實驗設備】

1、 裝有 Windows 系統和 S+core IDE 集成開發環境的 PC 機一台；

2、 凌陽 SPCE3200 嵌入式精簡開發板一套；

3、 本實驗用到開發板的介面有：TFT LCD 介面。

【實驗要求】

創建一個主視窗，並在主視窗上放置 10 個按鈕控制元，所有按鈕標題均爲“N”，然後在視窗下面放置一個進度條控制元，用來記錄被按下的按鈕的比例。當用戶單擊某個按鈕時，將按鈕標題更改爲“Y”，表示此按鈕以按下，並使進度條增加一個百分比，以便表示此時被按下的按鈕占總按鈕的百分比。如果用戶單擊了一個已經單擊過的按鈕，則進度條不變。如 圖 2.22所示。

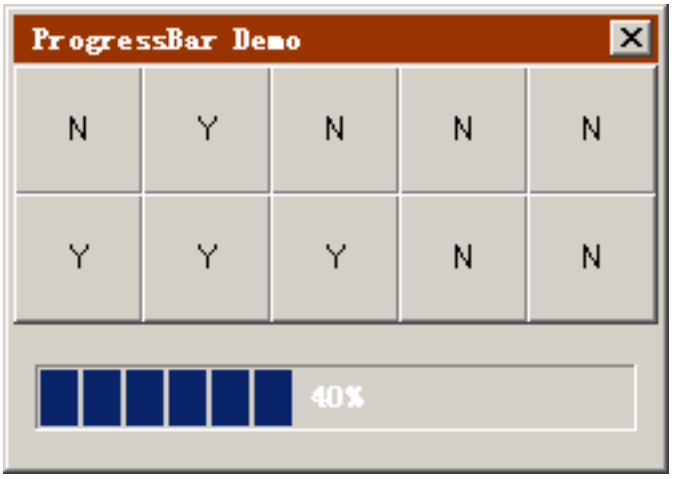

圖 2.22 利用進度條控制元編制“按鈕按下比例統計”介面

【實驗原理】

1、 進度條控制元簡介

進度條控制元是一個可以用來指示某個操作的進度的視窗。它包含了一個矩形區域，在這個矩形區域內有一個一般被填充系統高亮顏色的可以指示進度的條形圖形（進度條）。圖 2.23列出了MicroWin支援的兩種進度表示方式。

Tel:010-62981113-2944/2940/2942/2945 http://www.unsp.com
E-mail:unsp@sunplus.com.cn

圖 2.23 MicroWin 支援的兩種進度表示方式

進度條控制元通過選擇相應的樣式可具有以下功能特點：

- 進度條可設置爲連續填充的矩形或塊狀矩形
- 可選擇是否顯示以百分比方式顯示的進度資訊
- 支援水準或垂直方向的進度條樣式

2、 進度條控制元的創建

MicroWin 提供的進度條控制元的類名爲：“PROGBAR”，在使用之前必須使用進度條控制元註冊函數對其進行註冊，進度條控制元註冊函數的函數原型如下：

int WINAPI MwRegisterProgressBarControl(HINSTANCE hInstance);

其中，hInstance 表示當前視窗的實例控制碼，一般爲 NULL；返回值表示是否註冊成功，非零則表示成功，否則表示失敗。失敗的原因可能是因爲已經註冊過，或者系統沒有足夠的記憶體。

在註冊進度條控制元之後，便可以使用 CreateWindow 函數創建它，如下：

```
hWnd = CreateWindow(“PROGBAR”,                  // 預定義進度條控制元類名
                    NULL,                       // 進度條控制元一般無標題
                    WS_CHILD | WS_VISIBLE | ..., // 進度條控制元樣式，樣式說明見 表 2.20
                    X, Y,                       // 進度條控制元在視窗中的位置
                    nWidth, nHeight,            // 進度條控制元的寬度和高度
                    hWndParent,                 // 擁有該進度條控制元的視窗控制碼
                    (HMENU)ID,                  // 進度條控制元標識
                    hInstance,                  // 應用程式當前實例控制碼，一般爲 0
                    NULL);                      // 進度條控制元私有資料，必須爲 NULL
```

表 2.20 進度條控制元常用樣式及說明

樣式	說明
PBS_NOTIFY	設置此樣式後，如果進度條超出爲其設定的範圍，則會向其父視窗發出通告消息
PBS_SMOOTH	設置此樣式後將使用連續的矩形來表示進度，即 圖 2.23中的第二個進度樣式；否則將使用塊狀矩形表

	示進度，即 圖 2.23中的第一個進度樣式
PBS_VERTICAL	設置此樣式後，進度條將按照垂直方向增長（從下到上）；否則將按照水準方向增長（從左到右）
PBS_HIDETEXT	設置此樣式後，將不顯示百分比資訊

3、 進度條控制元與應用程式之間的消息傳遞

應用程式創建進度條控制元後，可通過接收控制元發出的消息得知控制元的狀態，並可通過向進度條控制元發送消息對其進行操作。

1) 進度條控制元向應用程式發送消息

當進度條控制元的值超過了它的範圍，並且進度條控制元被設置了 PBS_NOTIFY 樣式，則會向父表單發送 WM_COMMAND 消息通知應用程式來處理。進度條控制元發送的這個 WM_COMMAND 消息的 lParam 參數保存了進度條的控制碼；wParam 參數的低位元組爲控制元標識，高位元組爲進度條動作的消息通知碼。進度條控制元的消息通知碼有兩個：PBN_REACHMAX 和 PBN_REACHMIN。其中，當進度條控制元的值超過其上限時，將發送 PBN_REACHMAX 消息；當進度條控制元的值低於其下限時將發送 PBN_REACHMIN 消息。

2) 應用程式向進度條控制元發送消息

應用程式對進度條控制元的操作通過調用函數SendMessage或PostMessage向其發送各種消息來完成。表 2.21列出了MicroWin支援的進度條控制元控制消息及說明。

表 2.21 進度條控制元控制消息及說明

控制消息	說明
PBM_SETRANGE	設置進度條控制元的值的範圍 參數： wParam： 進度條控制元值的範圍的一個端點值 lParam： 進度條控制元值的範圍的另一個端點值 返回值：無 說明：wParam 和 lParam 指定了進度條控制元的值的範圍。進度條會根據設置的新的範圍自動調整進度位置，如果進度條的當前值超出了新設定的範圍，它會將當前位置調整到端點處。
PBM_SETSTEP	設置進度條每次增長的步長 參數： wParam： 步長值 返回值：無 說明：此步長值是指當控制元接收到 PBM_STEPIT 消息時增加的距離。

Tel:010-62981113-2944/2940/2942/2945 http://www.unsp.com
E-mail:unsp@sunplus.com.cn

	該值可以爲負數，表示進度反向增長。
PBM_SETPOS	設置進度條的當前位置 參數： wParam：進度條的新位置 返回值：無 說明：如果 wParam 的值超出了進度條的範圍，且進度條具有 PBS_NOTIFY 樣式，則控制元會向父窗口發送 PBN_REACHMAX 或 PBN_REACHMIN 通告消息。
PBM_DELTAPOS	使進度條從當前位置增加指定的值 參數： wParam：增加的值 返回值：無 說明：如果進度條增加 wParam 之後超出了進度條的範圍，且進度條具有 PBS_NOTIFY 樣式，則控制元會向父窗口發送 PBN_REACHMAX 或 PBN_REACHMIN 通告消息。
PBM_STEPIT	使進度條增加 PBM_SETSTEP 指定的步長值 參數：無 返回值：無 說明：如果進度條增加步長值之後超出了進度條的範圍，且進度條具有 PBS_NOTIFY 樣式，則控制元會向父窗口發送 PBN_REACHMAX 或 PBN_REACHMIN 通告消息。
PBM_GETRANGE	得到進度條控制元當前的範圍 參數： wParam：用於保存進度條控制元的值的範圍的最小值 lParam：用於保存進度條控制元的值的範圍的最大值 返回值：無
PBM_GETPOS	得到進度條當前的位置 參數：無 返回值：當前位置
PBM_SETBARCOLOR	設置進度條的顏色 參數： lParam：進度條的顏色

	返回值：無
PBM_SETBKCOLOR	設置進度條的背景顏色 參數： lParam：進度條的背景顏色 返回值：無

4、 本實驗的實驗原理

在本實驗中，與其他的 MicroWin 實驗一樣，首先使用 MicroWin 應用程式框架創建一個主視窗，並在 WM_CREATE 消息中創建按鈕和進度條等控制元。

程式的主要功能在按鈕的單擊事件中完成。當發生按鈕單擊事件後，程式使用 GetWindowText 函數獲取被單擊的按鈕的標題，判斷該按鈕是否曾經被按下過，如果沒有被按下，則修改按鈕的標題爲“Y”將其表示爲已按下，並向進度條控制元發送消息使其按照按下按鈕占所有按鈕的百分比來進行顯示。

程式的消息處理函數的流程如 圖 2.24所示。

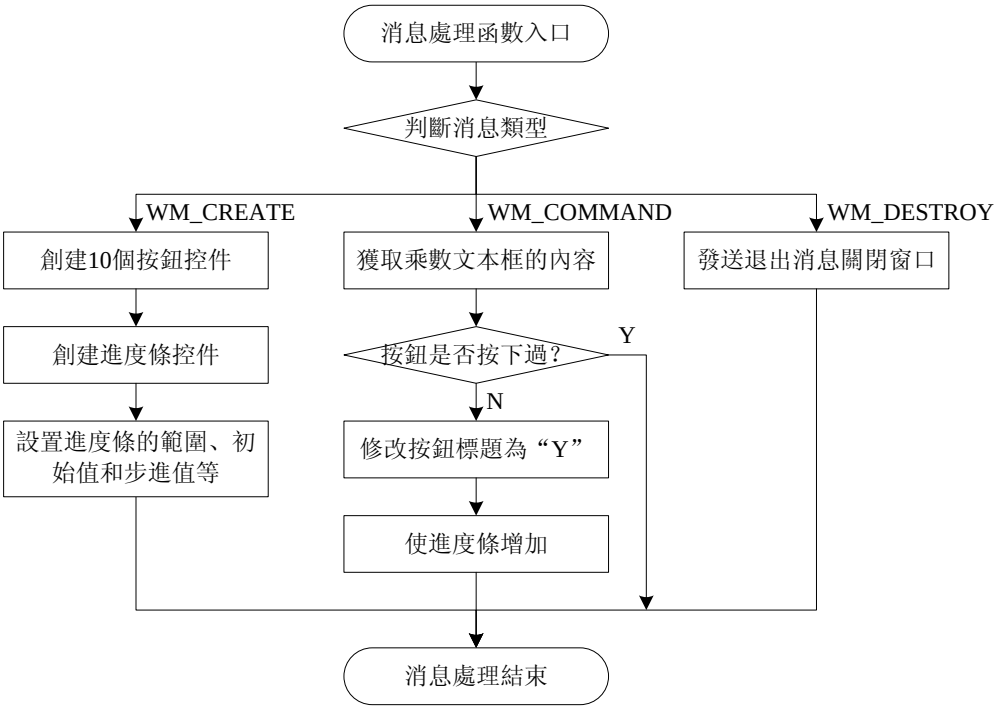

圖 2.24 實驗十九消息處理程式流程圖

【實驗步驟】

1、 打開 S+core IDE，使用“Score IDE MicroWin Project”範本採用預設設置新建一個工程；

2、 在工程嚮導生成的 APPWinMain.c 檔案中按照實驗原理的描述編寫程式；

Tel:010-62981113-2944/2940/2942/2945 http://www.unsp.com
E-mail:unsp@sunplus.com.cn

3、 把上個實驗範例中的“default_install”檔案夾拷貝到新建的工程目錄下；

4、 修改、編譯（Rebuild All）直到沒有任何錯誤；

5、 利用 RedBoot 下載程式到開發板上脫機運行；

6、 點擊主視窗上的按鈕，觀察進度條的變化。

【範例路徑】

在大學計畫網站（score.unsp.com）或者 32 位元嵌入式開發系統光碟都會提供本實驗的參考程式，路徑如下：

\Example_eCos\2.GUI_Exa\ex19_ProgressBar

3 附錄

控制元速查

MicroWin提供六種常用的控制元，各個控制元對應的類名以及註冊函數API名稱如 表 3.1所示。

表 3.1 MicroWin 系統常用控制元的類名及註冊函數

類名	控制元視窗類註冊 API	說明
STATIC	MwRegisterStaticControl	顯示靜態文本或圖像的子視窗（靜態控制元）
BUTTON	MwRegisterButtonControl	按鈕子視窗（按鈕控制元）
EDIT MEDIT	MwRegisterEditControl MwRegisterMEditControl	接收用戶輸入的文本輸入子視窗（文本框控制元）
COMBOBOX	MwRegisterComboboxControl	選擇列表框的子視窗（下拉清單控制元）
LISTBOX	MwRegisterListboxControl	字串列表的子視窗（列表框控制元）
PROGBAR	MwRegisterProgressBarControl	表示進度資訊的子視窗（進度條控制元）

各個控制元的常用樣式如 表 3.2所示。每個控制元的樣式大多數都可以使用“|”進行組合使用。

表 3.2 控制元和相應的樣式

控制元	通知碼	說明
靜態控制元	SS_LEFT	文字左對齊
	SS_CENTER	文本居中
	SS_RIGHT	文本右對齊
	SS_ICON	在靜態控制元上顯示圖示
	SS_GROUPBOX	使靜態控制元作爲一個具有標題的分組框
	SS_LEFTNOWORDWRAP	文本左對齊顯示，且不進行自動換行
	SS_BITMAP	在靜態控制元上顯示位元映射（同 SS_ICON）
	SS_NOTIFY	靜態控制元可以向父視窗發出消息
	SS_CENTERIMAGE	位元映射居中
	SS_REALSIZEIMAGE	不對位元映射進行自動縮放
按鈕控制元	BS_PUSHBUTTON	普通按鈕控制元
	BS_DEFPUSHBUTTON	預設的普通按鈕控制元
	BS_CHECKBOX	複選按鈕控制元

Tel:010-62981113-2944/2940/2942/2945 http://www.unsp.com
E-mail:unsp@sunplus.com.cn

	BS_AUTOCHECKBOX	自動複選按鈕控制元
	BS_RADIOBUTTON	單選按鈕控制元
	BS_AUTORADIOBUTTON	自動單選按鈕控制元
	BS_GROUPBOX	組框控制元
	BS_TEXT	表示按鈕上存在文本
	BS_ICON	表示按鈕上帶有圖示（同 BS_BITMAP）
	BS_BITMAP	表示按鈕上帶有位元映射（同 BS_ICON）
	BS_CENTER	按鈕上的文本居中
	BS_LEFT	按鈕上的文本左對齊
	BS_RIGHT	按鈕上的文本右對齊
	BS_USERBUTTON	用戶定義按鈕
文本框控制元	ES_LEFT	文本框內的文字左對齊
	ES_CENTER	文本框內的文字居中對齊
	ES_RIGHT	文本框內的文字右對齊
	ES_MULTILINE	文本框支援多行編輯（僅對“MEDIT”控制元有效）
	ES_UPPERCASE	自動將輸入轉換爲大寫
	ES_LOWERCASE	自動將輸入轉換爲小寫
	ES_PASSWORD	將輸入替換成“*”顯示
	ES_AUTOVSCROLL	當用戶輸入超出視窗時，視窗自動垂直滾動
	ES_AUTOHSCROLL	當用戶輸入超出視窗時，視窗自動水準滾動
	ES_NOHIDESEL	當文本框失去焦點時仍保持被選中文本的狀態
	ES_READONLY	設置文本框唯讀
	ES_WANTRETURN	按下回車時文本框自動換行
	ES_NUMBER	文本框只接受數位
列表框控制元	LBS_NOTIFY	列表框可向父視窗發出消息
	LBS_SORT	按字母順序排列表項
	LBS_MULTIPLESEL	允許選擇多項
	LBS_STANDARD	標準樣式
下拉式列示	CBS_SIMPLE	下拉式列示方塊中的列表框可見

	CBS_DROPDOWN	下拉式列示方塊由列表框和文本框組成，列表框平時不可見
	CBS_DROPDOWNLIST	下拉式列示方塊由列表框和靜態文本組成，列表框 平時不可見
	CBS_AUTOHSCORLL	文本框中自動水準滾動
	CBS_SORT	列表框中各項按字母順序排列
進度條控制元	PBS_NOTIFY	進度條可向父視窗發出消息
	PBS_SMOOTH	使用連續的矩形表示進度
	PBS_VERTICAL	進度條垂直方向增長
	PBS_HIDETEXT	不顯示百分比資訊

控制元通過向父視窗發送WM_COMMAND消息來通知自身狀態。該消息的lParam參數保存控制元的控制碼；wParam參數的低位元組爲控制元標識，高位元組爲控制元動作的通知碼。常用通知碼如 表 3.3所示。

表 3.3 控制元和相應的通知碼

控制元	通知碼	說明
靜態控制元	STN_CLICKED	單擊靜態控制元
	STN_DBLCLK	雙擊靜態控制元
	STN_ENABLE	靜態控制元被使能
	STN_DISABLE	靜態控制元被禁止
按鈕控制元	BN_CLICKED	單擊按鈕控制元
文本框控制元	EN_SETFOCUS	文本框獲得焦點
	EN_KILLFOCUS	文本框失去焦點
	EN_CHANGE	文本框的內容發生變化
	EN_UPDATE	文本框的內容被更新
	EN_MAXTEXT	用戶輸入達到允許的最大位元組數
	EN_HSCROLL	文本框的內容水準滾動
	EN_VSCROLL	文本框的內容垂直滾動
列表框控制元	LBN_ERRSPACE	記憶體不足，無法添加新項
	LBN_SELCHANGE	用戶的選擇發生改變
	LBN_SETFOCUS	列表框得到焦點

Tel:010-62981113-2944/2940/2942/2945 http://www.unsp.com
E-mail:unsp@sunplus.com.cn

	LBN_KILLFOCUS	列表框失去焦點
下拉式列示方塊控制元	CBN_SELCHANGE	下拉式列示方塊中列表框的選擇發生改變
	CBN_SETFOCUS	下拉式列示方塊得到焦點
	CBN_KILLFOCUS	下拉式列示方塊失去焦點
	CBN_EDITCHANGE	下拉式列示方塊中的文本框中的內容發生改變
	CBN_DROPDOWN	下拉式列示方塊中的列表框將顯示
	CBN_CLOSEUP	下拉式列示方塊中的列表框將隱藏
進度條控制元	PBN_REACHMAX	進度條達到上限
	PBN_REACHMIN	進度條達到下限

各個控制元的常用通知碼如 表 3.4所示。一般父視窗通過調用SendMessage或PostMessage函數向控制元發送控制消息。

表 3.4 控制元和相應的控制消息

控制元	通知碼	說明
靜態控制元	STM_SETICON	設置靜態控制元顯示的位元映射
	STM_GETICON	獲取靜態控制元當前顯示的位元映射
	STM_SETIMAGE	同 STM_SETICON
	STM_GETIMAGE	同 STM_GETICON
按鈕控制元	BM_GETCHECK	獲取單選按鈕或複選按鈕的選中狀態
	BM_SETCHECK	設置單選按鈕或複選按鈕的選中狀態
	BM_GETSTATE	獲取按鈕狀態
	BM_SETSTATE	設置按鈕狀態
	BM_GETIMAGE	設置按鈕控制元顯示的位元映射
	BM_SETIMAGE	獲取按鈕控制元當前顯示的位元映射
文本框控制元	EM_LIMITTEXT	設置文本框中文本的最大長度
	EM_SETPASSWORDCHAR	設置新的密碼字元
	EM_SETREADONLY	更改文本框的唯讀屬性。
	EM_GETPASSWORDCHAR	得到文本框當前使用的密碼字元
	EM_SETLIMITTEXT	同 EM_LIMITTEXT
	EM_GETLIMITTEXT	得到文本框當前的最大限制長度

列表框控制元	LB_ADDSTRING	向列表框添加項
	LB_DELETESTRING	刪除指定項
	LB_FINDSTRING	在列表中查找以指定字串開始的項
	LB_GETCOUNT	得到列表框當前的項數
	LB_GETCURSEL	獲取當前被選中的項的索引值
	LB_GETSEL	獲取指定的項的選中狀態
	LB_GETSELCOUNT	獲取多選列表框中選中的項數
	LB_GETTEXT	獲取指定項的名稱
	LB_GETTEXTLEN	獲取指定項的名稱的長度
	LB_GETTOPINDEX	獲取列表框中第一項的索引值
	LB_INSERTSTRING	在列表框的指定位置添加項
	LB_RESETCONTENT	清空列表框
	LB_SETSEL	設置多選列表框中指定項的選中狀態
	LB_SETCURSEL	設置單選列表框中的指定項爲當前選中的項
	LB_SETTOPINDEX	設置列表框中顯示的第一項的索引值
下拉式列示方塊控制元	CB_SHOWDROPDOWN	顯示列表框
	CB_ADDSTRING	向下拉式列示方塊中的列表框添加項
	CB_DELETESTRING	刪除下拉式列示方塊中的列表框中的指定項
	CB_INSERTSTRING	在下拉式列示方塊中的列表框的指定位置添加項
	CB_FINDSTRING	在下拉式列示方塊中的列表框中查找以指定字串開始的項
	CB_RESETCONTENT	清空下拉式列示方塊中的列表框
	CB_SETCURSEL	設置列表框的指定項爲當前選中項，並顯示
	CB_GETCURSEL	獲取下拉式列示方塊中的列表框當前被選中的項的索引值
	CB_GETCOUNT	得到下拉式列示方塊中的列表框當前的項數
	CB_GETLBTEXT	獲取下拉式列示方塊中的列表框指定項的名稱
	CB_GETLBTEXTLEN	獲取下拉式列示方塊中的列表框指定項的名稱的長度
	CB_LIMITTEXT	設置下拉式列示方塊中的文本框的字串的最大長度
進度條控制元	PBM_SETRANGE	設置進度條控制元的值的範圍
	PBM_SETSTEP	設置進度條每次增長的步長

Tel:010-62981113-2944/2940/2942/2945 http://www.unsp.com
E-mail:unsp@sunplus.com.cn

	PBM_SETPOS	設置進度條的當前位置
	PBM_DELTAPOS	使進度條從當前位置增加指定的值
	PBM_STEPIT	使進度條增加 PBM_SETSTEP 指定的步長值
	PBM_GETRANGE	得到進度條控制元當前的範圍
	PBM_GETPOS	得到進度條當前的位置
	PBM_SETBARCOLOR	設置進度條的顏色
	PBM_SETBKCOLOR	設置進度條的背景顏色