Lập trình ghép nối trên WINDOWS~

Bởi:

Khoa CNTT ĐHSP KT Hưng Yên

Lập trình ghép nối trên WINDOWS

Trong các phần trước chúng ta sử dụng ngắt để lập trình cho các thiết bị ngoài bằng môi trường Dos và viết chương trình là một việc khó khăn hơn. Hơn thế nữa các chương trình viết ra rất khó đọc và khó sử dụng đối với người mới bắt đầu hoặc là những người không hiểu sâu về phần cứng và hệ thống trong nó. Bây giờ chúng ta bắt đầu tìm cách lập trình sử dụng ngay các hàm của Window có hỗ trợ để lập trình ghép nối và điều khiển thiết bị ngoài,trong phần này chúng ta sử dụng ngôn ngữ Vb để lập trình với các thiết bị như SoundCard,VGAcard,...

Tệp *.DLL và cách tiếp cận.

Tệp DLL trong Windows:

Tệp tin *.DLL (Dymantic Link librany) Thư viện liên kết động, đôi khi còn gọi là các hàm thư viện của Windows (Window Function librany). Thay vì chúng ta phải viết toàn bộ ứng dụng bằng tay thì bây giờ chúng ta chỉ việc gọi các chức năng đã có sẵn trong tệp tin *.DLL. Đặc tính liên kết dộng hoàn toàn tương phản với với một khái niệm khác đó là liên kết tĩnh cụ thể là việc đóng gói sử dụng các hình ảnh liên kết tĩnh. Mặc dù hiện nay Unix cũng cung cấp các thư viện dùng chung (tương tự *.DLL), nhiều nhà cung cấp Unix các liên kết tĩnh vì ứng dụng hoàn toàn tương tự các thành phần cần thiết, việc cài đặt một thư viện hay ứng dụng mới không thể làm hỏng các chương trình cài đặt trước đó. Các nhà cung cấp không cần phải lo nắng tới việc khi cài đặt thì các phầm mềm khác có ảnh hưởng tới phần mềm của mình hay không. Khi sử dụng các tệp *.DLL có những ưu điểm sau đây.

* Tệp *.EXE có kích thước giảm đi đáng kể do phần lớn các mã nằm trong DLL.

* Một thư viện DLL có thể sử dụng đồng thời cho nhiều ứng dụng khác nhau, nhưng lại chỉ cần nạp một lần vào bộ nhớ trước khi chạy

* Các chương trình thể hiện tính Modul rõ hơn vì có sự thay đổi trong DLL thì các chương trình gọi thương không bị thay đổi.

Ngoài ra còn có một số ưu điểm khác

Một là: Dung lượng ổ đĩa được tiết kiệm

Hai là: Tiêt kiệm bộ nhớ vì nó có thể dùng cho nhiều ứng dụng khác nhau tức là nó áp dụng *kĩ thuật chia sẻ*. Windows cố gắng nạp DLL vào bộ nhớ Heap và khi chương trình nào cần sử thì sẽ ánh xạ địa chỉ vị trí của nó sang cho chương trình cần dùng. Phấn lớn các DLL có thể chia sẻ được nhưng có một số thì không thể chia sẻ mà phải sử dụng riêng

Ba là: Việc sửa chữa lỗi xảy ra tỏ ra dễ dàng hơn vì phấn nào sảy ra lỗi thì phần đó được sửa chữa. Các DLL có lỗi là duy nhất có thể sửa chữa nó mà kkhông cần phải viết lại toàn bộ ứng dụng

Cách tiếp cận Với DLL của Windows

Cơ chế bảo vệ file của Windows

Khi tiếp cận với các thơ viện một vấn đề quan trọng cần phải biết đến là cơ chế bảo vệ File Của window. Chức năng bảo vệ tập tin của window (WFP: Windows File Protected) là bảo vệ các DLL hệ thống khỏi phải sửa chữa hay xóa bỏ các tác nhân không được phép. Các ứng dụng không thể thay thế các DLL hệ thống, chỉ có các Package nâng cấp hệ điều hành, chẳng hạn như là SERVICE PACK mới có thể làm điều đó. Các DLL hệ thống chỉ được nâng cấp bỏi SERVICE PACK được gọi là DLL được bảo vệ (Protection DLL). Trong Window 2000 có khoảng 2.800 DLL được bảo vệ. Nếu ta thử sao chép một DLL được bảo vệ trong thư mục hệ thống (Win\System32) bằng một DLL cùng tên nhưng khác phiên bản thì mọi việc tưởng diễn ra êm dẹp và lại không hề báo lỗi. Nhưng sau đó Window 2000 lại thay thế nó bằng bản gốc ban đầu.

Mỗi khi DLL được đặt vào trong thư mục hệ thống thì Window nhận được một sự thay đổi thư mục thì nó lập tức kiểm tra xem có DLL được bảo vệ nào bị thay đổi hay không. Nếu đúng có sự thay đổi thì nó kiểm tra xem có chữ kí số phù hợp lệ hay không. Nếu không hợp lệ thì Window tự sao chép DLL gốc từ thư mục Win\System32\dllCache vào thư mục Window\SYStem32. WFP bảo vệ các DLL hệ thống khỏi sự thay đổi của các thành phần của phần mềm cài đặt. Ngay các sản phẩm của MicroSoft Như Office và Visual Studio cũng không thể nâng cấp các DLL được bảo vệ trong thư mục hệ thống.

Vấn đề xung đột DLL

Khi mới tiếp xúc với các DLL thì một trong số các khó khăn gặp phải là vấn đề xung đột DLL:Sau khi cài đặt một phần mềm mới với một số thư viện liên kết động DLL nào đó. Một hoặc vài ứng dụng có sẵn trên máy không làm việc được nữa.

Vấn đê xung đột DLL trên Windows9x gây nên bởi một số yếu điểm trong việc bảo vệ DLL: một chương trình cài đặt nào đó không kiểm tra phiên bản trước khi sao chép các DLL vào thư mục của hệ thống. Thí dụ nếu một chương trình cài đặt so sánh phiên bản hiện thời của MFC42.DLL và không gây ra vấn đề gỉ. Tuy nihên một chương trình cài đặt lại không làm thao tác này. Bản DLL cũ được sao chép đè nên bản DLL mới hơn. Hậu quả là khi chương trình yêu cầu một đọan mã trong bản mới hơn, nó sẽ không tìm được đoạn mã đó. Vấn đề này xảy ra rất phổ biến với những người sử dụng win9x, đặc biệt là khi Download các phần mềm miễn phí hoặc là khi soa chép các chương trình từ người quen. Các chương trình chuyên nghiệp ngày nay không gây nên vấn đề này bời lẽ chúng luôn kiểm tra chước khi ghi đè nên DLL.

Các DLL mới luôn được coi là tương thích với phiên bản cũ, nhưng điều này không phải bao giờ cũng đúng. Vấn đề xung đột DLL cũng có thể xảy ra khi một DLL được cài đặt nhưng bản thân nó lại chứa một lỗi mới. Mặc dù đây là nguyên nhân rất ít gặp nhưng đã có trường hợp xảy ra trong thực tế.

Cách tạo và sử dụng tệp *.DLL trong BASIC và DELPHI

Các DLL riêng

Thường thì các chương trình trên Window 9x xho những ứng dụng cụ thể ta sẽ càm thấy thiếu một số hàm nào đó. Khi đó cách giải quyết tốt nhất là viết ra hàm bằng một ngôn ngữ khác, chẳng hạn như ngôn ngữ C. Sau đó thiết lập một hàm DLL. Đây chính là DLL riêng (Private). Có thể định nghĩa: *DLL riêng là các DLL được cài đặt trong một ứng dụng xác định và chỉ có ứng dụng đó sử dụng.* Chẳng hạn ta quan tâm đến chương trình Maypp.exe. Ta đã kiểm tra Myapp.exe với Msvcrt.dll phiên bản 1.0 và với Mayapp.dll phiên bản 2.0. Ta muốn bảo rằng Mayapp.exe luôn được sử dụng Msvcrt.dll phiên bản x.x và MA.dll phiên bản 2.0. Để làm được việc đó phần mềm cài đặt của ta sao chép Mayapp.exe, Msvcrt.dll phiên bản 1.0 và Sa.dll phiên bản 2.0 vào thư mục /Myapp. Sau đó ta lưu ý Window 98/2000 rằng Myapp.exe sẽ dùng các DLL riêng đó. Khi Myapp chạy trên một hệ thống Windows 98/2000, nó tìm trong thư mục /myapp các DLL riêng trước khi tìm trong cá thư nục và đường dẫn của hệ thống. Các Service Pack tương lai nâng cáp Msvcrt.dll sẽ không thể làm hỏng Myapp vì nó không sử dụng phiên bản chung của Msvcrt.dll. Các ứng dụng khác có cài đặt các phiên bản khác của DLL không thể ảnh hưởng tới Myapp, bời lẽ Myapp có phiên bản sử dụng riêng MA.dll.

Các DLL riêng còn được gọi là các DLL cạnh nhau (Side to side), bởi lẽ một bản riêng của DLL được sử dụng trong ứng dụng ứng dụng khác. Nếu ta chạy WorđPa và mypp đồng thời thì hai bản Msvcrt.dll được nạp vào vào bộ nhớ (do đó mà có thuật ngữ "cạnh nhau"), ngay cả khi WordPad và Myapp cùng dùng chung phiên bản của Msvcrt.dll.

Có hai cách tiếp cận để có được DLL riêng. Nếu ta đang viết một ứng dụng mới hoặc một bộ phận mới, ta đặt cho mỗi phần một phiên bản duy nhất. Ứng dụng của ta biết phải nạp bản riêng DLL dùng chung nhờ thông tin phiên bản của nó. Cách tiép cận thứ hai là bảo vệ các ứng dụng có sẵn, Giả sử C:\Myapp\Myapp.exe là một ứng dụng đã có mà ta muốn bảo vệ khỏi rủi ro trong những lần nâng cấp của các DLL sau cũng như của Service Pack. Ta chỉ cần sao chép các DLL định biến thành các DLL riêng của Myapp vào thư mục \Myapp và tạo ra một tệp rỗng trong thư mục đó tên là "Myapp.exe local". bây giờ khi Myapp chạp và tìm File.local nó sẽ tìm kiếm trong thư mục hiện thời các DLL và COM service trước khi tìm đến đường dẫn chuẩn. Nếu ứng dụng của ta mà bị lỗi do Service Pack nâng cấp, ta tạo một chương trình cài đặt với file.local và các DLL mà ta cần cung cấp chúng cho khách hàng.

Cả cách tiếp cận chỉ đình phiên bản (cho các ứng dụng đang viết) và cho local (các ứng dụng đã có) đều có một số đặc tính như sau:

* Các DLL trong thư mục ứng dụng đươc nạp thay vì các DLL hệ thống

* Ta không thể đổi hướng 20 DLL đươck liệt kê trong HKEYLOCAL_MACHINE\ SYSTEM\CurrentControlSet\ControlSession manager\ KnownDLLs. Phần lớn chúng không thể chạy cạnh nhau vì cần duy trì các trạng thái không phụ thuộc vào tiến trình Vídụ: kernel32, user32 và ole32 không thê bị đổi hướng bởi lẽ chúng có trạng thái (các đối tượng kernel, các handle cửa sổ...) cần duy trì xuyên suốt các tiến trình. Trong hệ điều hành tương lai một số các DLL này sẽ được sửa lại và chạy cạnh nhau, khi đó danh sách sẽ được rút bớt.

* Để giải quyết các xung đột trong các ứng dụng hiện có phải xác định xem: DLL nào cần được bảo vệ.

Các tiếp cận chỉ định cần phải được thực hiện bởi chính tác giả của bộ phận hay ứng dụng. Các nhà quản lí các chương trình cài đặt có thể tạo ra các file.local rỗng để thực hiện cách tếp cận thứ hai. Các hệ điều hành hiện nay hầu hết có các cơ chế bảo vệ.

Tệp Port.DLL

Điều đặc biệt khó khăn khi làm việc trong môi trường Windows là tiếp cận đến giao diện của máy tính PC. Thực tế cho thấy có một biện pháp hiệu quả nhất là tạo ra một tệp DLL có khả năng sử dụng trong nhiều ứng dụng. Một tệp khác được giới thiệu trong

phần sau có tên quy ước là 8255.DLL. Được viết bằng C^{++} các tệp này được viết trong một ngôn ngữ khác, tùy theo kinh nghiêm của người lập trình.

Tệp DLL được đề cập đến nhiều lần trong phàn này là đượ viết dưới dạng ngôn ngữ Delphi có quy ước là Port.dll để chỉ ró đối tượng ứng dụng là các cổng. Tệp DLL này được thực hiện chức năng mở rộng của ngôn ngữ để dùng cho các ngôn ngữ lập trình khác nhau. Sau khi tạo ra (hoặc kiếm được) thì phải sao chép vào trong thư mục hệ thống của Windows để các chương trình đều có thể sử dụng được. Tùy theo cách lựa chọn, ta cũng có thể đặt tệp DLL này vào trong thư mục chương trình nào đó của chương trình điều hành (EXE)

Những nhiệm vụ đặt ra khi viết tệp PORT.DLL là:

Mở giao diện

Truyền dữ liệu theo cách nối tiếp

Tiếp cận đến các đường dẫn ở giao diện

Nhập và xuất ra các cổng

phát khoảng thời gian để cho có thời lượng chính xác đến Mili giây

phát khoảng thời gian để cho có thời quét chính xác đến Micro giây

Truy cập tới card âm thanh

Truy cập qua cổng trò chơi

Tệp DLL có thể được sử dụng trong các hệ thống có thể lập trình hoàn toàn khác nhau. Ngoài ngôn ngữ được lập trình trong phần này, các hàm có thể được viết bằn ngôn ngữ C. Vì vậy một chương trình được viết ra một lần có thể được chuyển giao dễ dàng sang các hệ hthống lập trình khác. Ngoài ra, ta có thể trao đổi với thư viện DLL trong các Macro của Word hoặc Exel. Việc viết ra một thư viện DLL chung để truy cập tới phàn cứng đã được dự tính. Ở một mức độ nào đó có phần trái ngược với cách tư duy của nhà thiết kế phần mềm khoi xây dựng Windows. Trong đó tất cả các thao tác truy cạp nên phần cứng đều tiến hành thông qua các tệp đệm (Driver). Một tệp đệm luôn đi theo một thiết bị hoàn toàn xác định. Đối với ứng dụng ghép nối máy tính không chuyên nghiệp thì không thể hy vọng đến sự giúp đỡ của các tệp này. Lý do việc viết ra các tệp này thường rất tốn kém và thường được viết ra bởi các hãng lớn.

Trong DOS, mỗi ngôn ngữ lập trình đều có các lệnh dùng cho cổng, mà thường gọi tắt là lệnh cổng (GWBASIC là INP và OUT, còn trong TurboPascal là PORT [...],...).

Để trao đổi trực tiếp trên toàn bộ phần cứng của máy tính PC.Trong một số hệ điều hành thì chúng ta thâm nhập phần cứng phải thông qua các hàm các thư viện các dịch vụ của hệ điều hành và có sự bảo vệ của hệ thống nên việc truy cập trực tiếp vào phần cứng càng trở nên khó khăn. Trong các hệ điều hành thiên hướng mạng thì việc truy cập đó lại càng bị bó hẹp

Gọi tệp *.DLL trong VisualBasic

Hình 4-10: Tạo một Module

Việc sử dụng một tệp tin DLL có thể được chỉ ra ở đây thông qua một thí dụ đơn giản mà không bổ sung thêm gì cho phần cứng. Loa của máy tính được điều khiển thông qua các khối, các khối này có thể điều khiển qua cá lệnh cổng. Loa được điểu khiển hoặc là bằng bộ định thời để xuất ra âm thanh có tần số nhất định hoặc ta có thể điều khiển trực tiếp thông qua đường dẫn xuất ra vi mạch ghép nối ngoại vi lập tình được PPI (Programable Peripheral Interface) loại 8255 trong máy tính pc ta cũng có thể tạo ra các âm thanh theo các cách thay đổi trạng thái logic một đường dẫn bằng một chuỗi các tac dộng liên tục: Bật và tắt. Các phương phápthử nghiệm dược nêu ra ở đây một mặt đẻ làm quen với các khái niệm cơ bản về một tệp DLL, mặt khấc đểkhảo sát tiến trình thời gian trong Windows.

Loa được điều khiển qua bít 1 của cổng B trên vi mạch 8255. Vi mạch này chiếm các địa chỉ 60h (96 dec) trong vùng vào/ra của PC, cổng B nằm ở địa chỉ 97. Các nối ra cổng luôn có độ rộng 8 bít, vì thế 8 đường dẫn có thể chuyển cùng một lúc. Nhưng chỉ có đường dẫn bit 1 được phép thay đổi, bởi vì cổng B của mạch PPI, còn điều khiển nhiều đường dẫn khác. Do đó trước hết, trạng thái của cổng chỉ được đọc để thay đổi chỉ một bit. Nếu cảm thấy kho khăn trong việc tìm hiểu cách xử lí từng bit, ta nên chọn cách tiếp cận với cách chương trình dùng làm thí dụ được giới thiệu trong các chương trình sau; Ngoài ra việc truy nhập nên ccác giáo diện từ bên ngoài có phần đơn giản hơn.

Để tiếp cận với các địa chỉ cổng riêngbiệt trên máy tính PC tệp DLL giới thhiệu hai hàm cụ thể.

Out Port ADR.DAT 'xuất dữ liệu ra một địa chỉ ra tệp

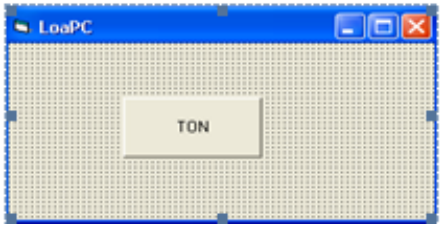
Input port ADR 'đọc dữ liệu từ địa chỉ ADR

Trong Visual Basic outport có thể được kết nói như là Sub (procedure) ngược lại inport phải là một hàm. Các phàn tử của tệp DLL đợc chỉ định bằng lệnh khai báo (Declare) để việc chuyển giao dữ liệu giữa VIsualBasic và têp DLL, có thể vận hành đúng thì tất cả các thông số giao ByVal cần được khai báo,nghĩa là như một tham trị- ngược với vieecj chuyển giao một địa chỉ so sánh trong hệ diều hành Windows 95/95 32 bit phải dùng các chữ *in khi khai báo* trong thư viện DLL, Tất cả ccác hàm chỉ được đặt tên bằng các chữ cái viết in và tên này phải được giữ nguyên trong chương trình được gọi. những lời khai báo toàn cục phải được khai báo trong một Modul riêng. Modul này được kết nối trong Project TON,

Declare sub OUTPORT Lib "PORT.DLL" (ByVal Adr as Integer, Byval dat as Integer)

Declare Fuction OUTPOR Lib "PORT.DLL" (ByVal Adr as Integer, Byval as Integer) as Integer

Declare Sub DELAY Lib "PORT.DLL" (ByVal thoigian As Integer)



Hình 4-11: cửa sổ chương trình

Bây giờ Inport và Outport (Trong đoạn chương trình viết bằng chữ In !) có thể được sử dụng trong một Project chung. Ngoài ra thủ tục Delay cũng đã được khai báo và sẽ được sử dụng trong phần dưới đây. Lần xuất âm tần đẩu tiên cần phải được tạo ra trong vòng lặp nhanh với 100 xung vuông góc ở loa, và tàn số của âm thanh phải nằm trong vùng nghe được. Việc quản lí từng bit khi xuất ra cổng bằng lệnh outport được tiến hành bằng cách sử dụng các hàm logic ALD và OR để chỉ thay đổi một đường dẫn của cổng được đọc vào bằng lệnh Inport. Các hàm này sẽ còn được giải thích chi tiết hơn trong các phần sau. Chương trình nên sử dụng một khuôn mẫu đơn giản với một phím nhấn TON. Đường dẫn loa dược tắt/bật 100 lần nhờ vậy ta có thể nghe chẳng hạn ở một máy tính tính PC với xung Nhịp 200Mhz chỉ một loại âm thanh thì thời gian chờ bổ sung cần phải được điền vào. Ở đây thời gian chở được tạo trong vòng lặp đếm với 10000 vòng lặp

Private Sub Command1_Clic()

```
Dim i, t As Integer

For i = 1 To 100

OUTPORT 97, (INTPORT(97) Or 2)

For t = 1 To 1000

Next t

OUTPORT 97, (INTPORT(97) And 2)

Next

For t = 1 To 1000

Next t

End Sub
```

Khi nhấn vào phín "TON" ta sẽ nghe thấy âm thanh phất ra ở loa. Độ cao của tần số âm thanh phụ thuộc vào máy tính được sử dụng. Điểm đáng chú ý là; âm thanh được tạo ra có chất lương không cao; đôi khi ta thấy nhiều tiếng ồn hoặc tiếng lạo xạo nguyên nhân là tiến trình sử dụng thời gian (Time Characteristic) của Windows.

Thời gian chờ được tạo ra qua vòng lặp trễ thường không bao giờ có thể đông đều bởi vì Windows còn phải hoàn thành nhiều nhiêm vụ chẳng hạn Windows còn phải quan sát chuột hoặc các quá trình khác đang diễn ra đổng đồng thời cần được xử lí do đó người ta thờng nói rằng Window không có khả năng thời gian thực, rằng không thể điều khiển các quá trình diễn biến nhanh bằng Windows một cách tin cậy. Tất nhiên là nhận xét này mang tính tương đối bởi vì nnhanh đến thế nào và tin cậy đến mức nào còn là một ranh giới chưa rõ ràng. Có thể khẳng định rằng khôngthể tạo ra một âm thanh trong chẻo bằng những chương trình đã dẫn ra làm thí dụ trên.

Đương nhiên là vòng lặp đếm không phải là giải pháp được lựa chon trước tiên khi ta quan tâm đến thời giam trễ Window đẫ cung cấp một phương tiện tốt hơn đẻ nhận được thời gian trễ đến tưng mili giây thông qua việc truy nhập tới hàm Delay của DLL việc sử dụng hàm delay theo cách này cho phép cải thiện chất lượng âm thanh được xuất ra đấng tiếc là tần số ccaco nhất của âm thanh được xuất ra chỉ cỡ 500 Hz khi ta thay đổi trạng cổng từng ms

Với đoạn chương trình này âm thanh được xuất ra nghe rõ ràng hơn tuy chất lượng chư so sánh với âm thanh được tọa ra từ các vi mạch. Các kết quả nhận được từ DLL Realime (true) còn được cải thiện nhiều hơn. Nhưng ở đây ta có một ấn tượng rõ ràng

khả năng thời giam thực của Window có thể đi xa hơn. Muốn khảo sát chi tiết hơn, ta cần đến một giao động để có thể quan sát trọng thái đường dẫn, chẳng hạn ở giao diện cổng COM của cổng nối tiếp

ví dụ

```
Private Sub Command1_Click()

For n = 1 To 100

OUTPORT 97, (INPORT(97) Or 2)

DELAY 1

OUTPORT 97, (INPORT(97) Or 253)

DELAY 1

Next n

End Sub
```

Bên cạnh hàm DELAY là dùng cho khoảng thời gian là mini giây, trong DLL còn có hàm giây trễ với khoảng thời gian là micro giây.

Còn một vấn đề cần quan tâm đến là *việc gọi hàm DLL*. Tất cả các lời gọi DLL cần được khai báo trong modul Basic bên ngaòi có tyê là PORT.PAS sau đó, thư viện có htể nạp vào project mới, mà không đòi hói sự quan tâm nhiều hơn đến các khai báo. Tròn tệp PORT.DLL phải được đặt trong thư mục Window hoặc phải đặt trong thư mục có chứa chương trình exe cần chạy

Đoạn chương trình sau đây là PORT.BAS với tấ cả các khai báo dùng trong VB5

```
Declare Function OPENCOM Lib "Port" (ByVal a$) As Integer

Declare Sub CLOSECOM Lib "Port" ()

Declare Sub SENBYTE Lib "Port" (ByVal b$)

Declare Function READBYTE Lib "Port" () As Integer

Declare Sub DTR Lib "Port" (ByVal b$)

Declare Sub RTS Lib "Port" (ByVal b$)
```

```
Declare Sub TXD Lib "Port" (ByVal b$)

Declare Function CTS Lib "Port" () As Integer

Declare Function DSR Lib "Port" () As Integer

Declare Function RI Lib "Port" () As Integer

Declare Function DCD Lib "Port" () As Integer

Declare Sub DELAY Lib "Port" (ByVal b$)

Declare Sub TIMEINIT Lib "Port" ()

Declare Sub TIMEINITUS Lib "Port" ()

Declare Function TIMEREAD Lib "Port" () As Long

Declare Function TIMEREADUS Lib "Port" () As Long

Declare Sub DELAYUS Lib "Port" (ByVal l As Long)

Declare Sub READTIME Lib "Port" (ByVal l As Boolean)

Declare Sub OUTPORT Lib "Port" (ByVal a%, ByVal b%)

Declare Function INPORT Lib "Port" (ByVal p%) As Integer

Declare Function JOYX Lib "Port" () As Long

Declare Function JOYY Lib "Port" () As Long

Declare Function JOYZ Lib "Port" () As Long

Declare Function JOYW Lib "Port" () As Long

Declare Function JOYBUTTOM Lib "Port" () As Long

Declare Function SOUNDSETRATE Lib "Port" (ByVal a$) As_ Integer

Declare Function SOUNDSETRATE Lib "Port" () As Integer

Declare Function SOUNDBUSY Lib "Port" () As Boolean
```

Declare Function SOUNDIS Lib "Port" () As Boolean

Declare Sub SOUNDIN Lib "Port" (ByVal buff$, ByVal size%)

Declare Sub SOUNDOUT Lib "Port" (ByVal buff$, ByVal size%)

Declare Function SOUNDBYTES Lib "Port" () As Integer

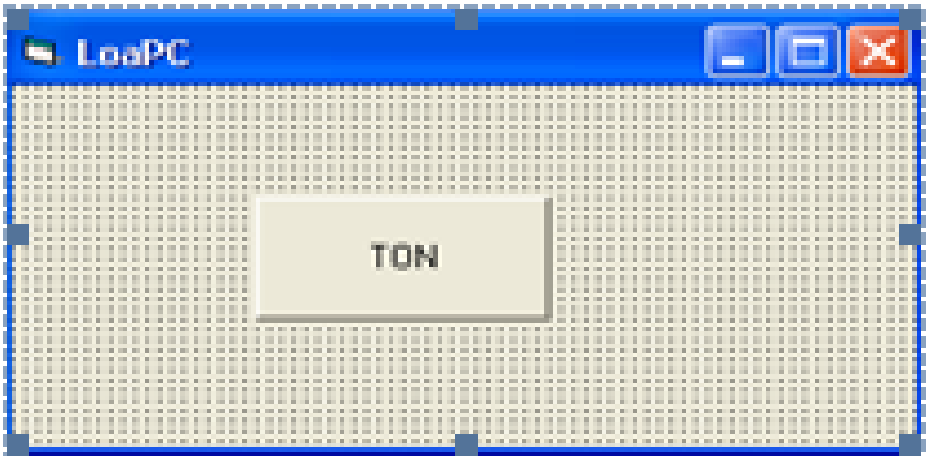
Declare Function SOUNDBYTES Lib "Port" (ByVal b%) As Integer

Declare Sub SOUNDCAPIN Lib "Port" ()

Declare Sub SOUNDCAPOUT Lib "Port" ()

Theo cách tương tự, Các hàm DLL có thể được khai báo trong VBA (VisualBasic Application). Trên nguyên tắc ccó thể chạy thử tất cả các chương trình có thể chạ trên Word Hoặc Excel, các chương trình có thể khởi độnng như là cá marco, các khai báo có thể có trong VisualBasic. Thí du sau đây cho phép tạo ra âm thanh từ loa của PC. Các thí dụ viết bằng VisualBasic được giới thiệu sau đây.

Trong Word 97 một User_Form có thể chứa các phần tử điều khiển riêng, cũng có thể được sử dụng trong một Macro. Khi đó, trước từ "Declare" luôn là từ khóa "Private". Theo cách này, tất cả các hàm và thủ tục được tạo liên kết với Private



Hình 4-12: Một macro trong Word 97