LỜI GIỚI THIỆU

Trong các máy tính thế hệ hiện nay có một số thiết bị ngoài thông dụng như: Màn hình, bàn phím, chuột, máy in…với các thiết bị ngoài đó máy tính đều có khối ghép nối tương ứng và chúng được tích hợp luôn trên một bo mạch gọi là main board. Tuy nhiên máy tính không chỉ dừng lại với thiết bị ngoại vi nói trên mà có những yêu cầu cao hơn, như kết nối với các máy móc trong công nghiệp…và đã được các nhà sản xuất lưu tâm tới và họ để trống vô số các con đường có thể ghép nối với bus của máy tính như: RS232, LPT, COM, USB, các khe PCI …Đây chính là con đường ai muốn nghiên cứu mở rộng phạm vi kết nối của máy tính kết hợp sử dụng với bộ vi điều khiển có thể lập trình được.

Mô đun *"Lập trình ghép nối máy tinh"* là một mô đun chuyên môn của học viên ngành sửa chữa máy tính. Mô đun này nhằm trang bị cho học viên các trường công nhân kỹ thuật và các trung tâm dạy nghề những kiến thức về kỹ thuật lập trình, kỹ thuật ghép nối các thiết bị ngoại vi với máy tính...với các kiến thức này học viên có thể áp dụng trực tiếp vào lĩnh vực sản xuất cũng như đời sống. Mô đun này cũng có thể làm tài liệu tham khảo cho các cán bộ kỹ thuật, các học viên của các nghành khác quan tâm đến lĩnh vực này.

Mặc dù đã có những cố gắng để hoàn thành giáo trình theo kế hoạch, nhưng do hạn chế về thời gian và kinh nghiệm soạn thảo giáo trình, nên tài liệu chắc chắn còn những khiếm khuyết. Rất mong nhận được sự đóng góp ý kiến của các thầy cô cũng như các bạn học sinh, sinh viên và những ai có nhu cầu sử dụng tài liệu này.

Hà Nội, 2013

Tham gia biên soạn
Khoa Công Nghệ Thông Tin
Trường Cao Đẳng Nghề Kỹ Thuật Công Nghệ
Địa Chỉ: Tổ 59 Thị trấn Đông Anh – Hà Nội
Tel: 04. 38821300
Chủ biên: Nguyễn Kim Dung

Mọi góp ý liên hệ: Phùng Sỹ Tiến – Trưởng Khoa Công Nghệ Thông Tin
Mobible: 0983393834
Email: tienphungktcn@gmail.com – tienphungktcn@yahoo.com

MỤC LỤC

ĐỀ MỤC **TRANG**

LỜI GIỚI THIỆU.....1
BÀI MỞ ĐẦU: TỔNG QUAN VỀ NGÔN NGỮ LẬP TRÌNH TRUYỀN THÔNG.....5
Mục tiêu.....5
A. LÝ THUYẾT.....5
B. CÂU HỎI VÀ BÀI TẬP.....23
BÀI 1: CÁC CÂU LỆNH VÀ ĐỐI TƯỢNG TRONG NGÔN NGỮ LẬP TRÌNH.....24
Mã bài : MĐ38-02.....24
Giới thiệu.....24
Mục tiêu.....24
A. LÝ THUYẾT.....24
B. CÂU HỎI VÀ BÀI TẬP.....38
BÀI 2: LẬP TRÌNH THIẾT BỊ ẢO.....39
Mã bài: MĐ38-03.....39
Giới thiệu.....39
Sau khi đã tìm hiểu sâu về kỹ thuật lập trình Visual Basic đặc biệt là thư viện các hàm, ta có thể tạo ra các thiết bị ảo có mức độ phức tạp vừa phải. Các thiết bị này có thể đặt riêng lẻ hoặc trong khuôn khổ của một chương trình lớn. Để tiến hành các thiết bị ảo ta cần đến một số tệp để hình thành các điều khiển tùy biến được kết hợp với nhau theo cách dùng chia sẻ. Vấn đề đặt ra là làm thế nào để sử dụng các điều khiển này cho thích hợp với từng đồ án cụ thể.....39
Mục tiêu.....39
A. LÝ THUYẾT.....40
B. CÂU HỎI VÀ BÀI TẬP.....54
BÀI 3: LẬP TRÌNH QUA CỔNG NỐI TIẾP.....54
Mã bài: MĐ38-04.....54
Giới thiệu.....54
Mục tiêu.....54
A. LÝ THUYẾT.....54

Article I. 6 Bits 79
B. CÂU HỎI VÀ BÀI TẬP.....85
BÀI 4: LẬP TRÌNH QUA CỔNG SONG SONG.....92
Mã bài: MĐ41-05.....92
Giới thiệu.....92
Mục tiêu.....92
A. LÝ THUYẾT.....92
BÀI 5: LẬP TRÌNH QUA CÁC MẠCH GHÉP NỐI ĐA NĂNG.....127
Mã bài: MĐ41-06.....127
Mặc dù số lượng các đường dẫn được sử dụng ở giao diện nối tiếp bị hạn chế, kỹ thuật ghép nối máy tính của giao diện nối tiếp đã được đề cập tương đối kỹ trong chương trước đã học, có thể đem phối hợp các ứng dụng với nhau để tạo ra một giao diện (khối ghép nối đa năng) dùng cho những ứng dụng mới khác nhau.....127
Mục tiêu.....127
A. LÝ THUYẾT.....127
B. CÂU HỎI VÀ BÀI TẬP.....134
Câu 1: Trình bày cách xây dựng phần cứng và điều khiển?.....134
TÀI LIỆU THAM KHẢO.....139

MÔ ĐUN: LẬP TRÌNH GHÉP NỐI MÁY TÍNH

❖ **Mã mô đun:** MĐ38

❖ **Vị trí, ý nghĩa, vai trò của mô đun:**

- Vị trí:

+ Môn đun được bố trí sau khi học xong môn học/mô đun: Lắp ráp và cài đặt máy tính, Đo lường và điều khiển máy tính, Kiến trúc máy tính, Sửa chữa máy tính.

- Tính chất:

+ Là môn đun chuyên ngành tự chọn.

- Ý nghĩa, vai trò của mô đun:

+ Mô đun này giúp chúng ta có thể tạo ra được mối liên hệ cần thiết giữa máy tính và thế giới bên ngoài

+ Chương trình điều hành hệ thống ghép nối của mô đun có khả năng đảm nhận việc thu thập thông tin từ bên ngoài và điều khiển các thiết bị ghép nối với máy tính

+ Giúp cho người đọc có kỹ năng về lập trình đối tượng và tạo ra được các mạch điện tử để ghép nối với máy tính có nhiều ứng dụng trong thực tiễn.

❖ **Mục tiêu của mô đun:**

- Lập trình truyền thông qua cổng nối tiếp và song song
- Xây dựng kế hoạch và thiết kế các chương trình điều khiển ghép nối máy tính.
- Lập trình hoàn chỉnh trên môi trường phát triển với ngôn ngữ hỗ trợ lập trình ghép nối.
- Có ý thức tự giác, tính kỷ luật cao, tinh thần trách nhiệm trong học tập.
- Bình tĩnh, tự tin trong các công việc liên quan ghép nối máy tính..

Mã bài	Tên bài	Thời lượng			
		Tổng số	Lý thuyết	Thực hành	Kiểm tra
MĐ38-01	Bài mở đầu: Tổng quan về ngôn ngữ lập trình truyền thông	04	03	10	0
MĐ38-02	Bài 1: Các câu lệnh và đối tượng trong ngôn ngữ lập trình	10	03	05	02
MĐ38-03	Bài 2: Lập trình thiết bị ảo	10	04	05	01
MĐ38-04	Bài 3: Lập trình qua cổng nối tiếp	12	06	06	
MĐ38-05	Bài 4: Lập trình qua cổng song song	12	06	05	01
MĐ38-06	Bài 5: Lập trình qua các mạch ghép	12	06	06	

	nối đa năng				

BÀI MỞ ĐẦU: TỔNG QUAN VỀ NGÔN NGỮ LẬP TRÌNH TRUYỀN THÔNG

Mã bài : MĐ38-01

❖ **Giới thiệu**

Bài mở đầu nhằm giới thiệu cho học sinh một cách tổng quan về ngôn ngữ được sử dụng để lập trình giao tiếp với máy tính, việc lập trình Visual có thể hiểu là dùng các điều khiển có sẵn, chúng ta sẽ dùng chuột với những thao tác để lấy những điều khiển cần dùng từ hộp công cụ đưa vào Form để thiết kế chương trình, xác lập các thuộc tính cho chúng và sau đó là viết lệnh cho những điều khiển đó.

❖ **Mục tiêu**

- Nhận biết được tổng thể của ngôn ngữ lập trình truyền thông
- Trình bày chính xác các điều khiển truyền thông, đặc tính, các sự kiện
- Sử dụng gọi được các hàm API trong lập trình truyền thông và một số ứng dụng trong lập trình truyền thông.
- Cẩn thận, tự giác, chính xác...
- Tuân thủ, đảm bảo an toàn cho người và thiết bị
- Có tinh thần trách nhiệm cao trong học tập và làm việc

❖ **Nội dung chính**

A. LÝ THUYẾT

1. Giới thiệu về ngôn ngữ truyền thông

Mục tiêu :

- Hiểu được khái niệm về ngôn ngữ lập trình Visual Basic

- Thực hiện được một số thao tác cơ bản trên phần mềm Visual Basic 6

1.1. Giới thiệu Visual Basic

Visual Basic là một ngôn ngữ lập trình cấp cao 32 bit được sử dụng để viết các chương trình chạy trong môi trường Windows. Visual Basic sử dụng kiểu lập trình Visual hay RAD(Rapid Application Development) trong đó việc tạo các cửa sổ, các Điều khiển và cách ứng xử của các cửa sổ cũng như các Điều khiển được thực hiện một cách dễ dàng nhanh chóng chỉ bằng các thao tác với mouse không cần phải khai báo, tính toán với nhiều câu lệnh phức tạp.

Visual Basic là một ngôn ngữ lập trình theo kiểu hướng đối tượng. Nó khác với kiểu lập trình cũ là kiểu Top Down.

- Lập trình Top Down: chương trình được bố trí và thực thi từ trên xuống. Với kiểu

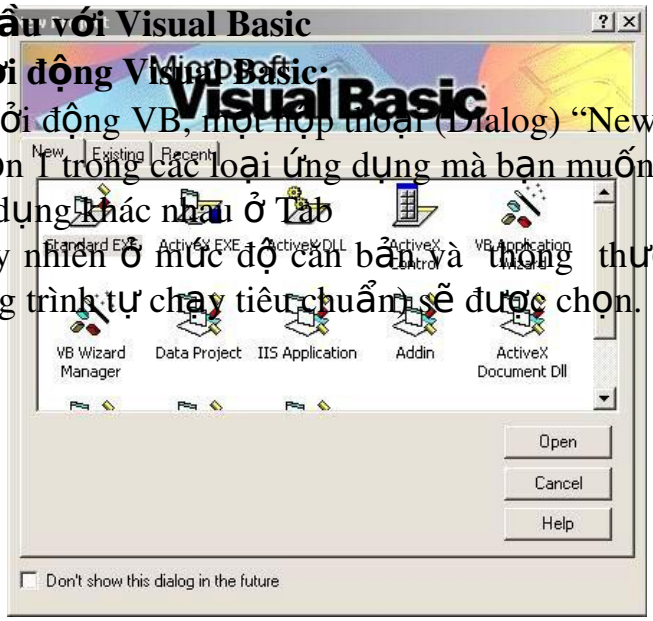
lập trình này, việc bố trí sẽ trở nên rất khó khăn đối với các chương trình lớn.

- Lập trình hướng đối tượng OOP (object-oriented programming): Các thành phần được phân thành các đối tượng (Object) và viết cách ứng xử riêng cho mỗi đối tượng sau đó kết hợp chúng lại tạo thành chương trình.

1.2. Bắt đầu với Visual Basic

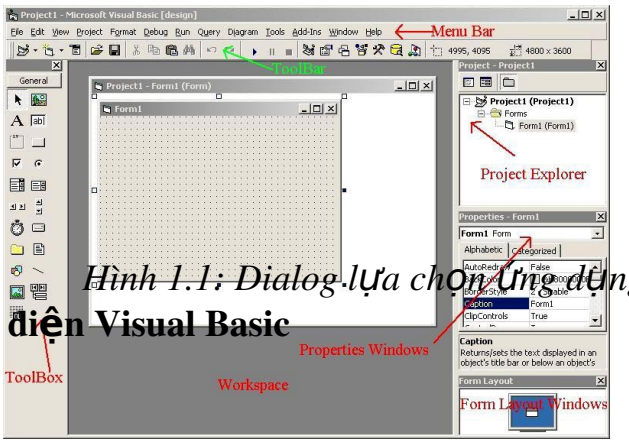
1.2.1. Khởi động Visual Basic

Sau khi khởi động VB, một hộp thoại (Dialog) "New Project" xuất hiện cho phép lựa chọn 1 trong các loại ứng dụng mà bạn muốn tạo. VB6 cho phép tạo 13 loại ứng dụng khác nhau ở Tab "New", tuy nhiên ở mức độ căn bản và thông thường, Standard EXE (một loại chương trình tự chạy tiêu chuẩn) sẽ được chọn.



Hình 1.1: Dialog lựa chọn ứng dụng

1.2.2. Giao diện Visual Basic



Hình 1.2: Giao diện Visual Basic

+ Giao diện cơ bản của VB bao gồm các thành phần sau:

- ✓ MenuBar: các trình đơn của VB.
- ✓ Toolbar: một số chức năng cơ bản của chương trình.
- ✓ Toolbox: chứa các Điều khiển (Control) thông dụng.
- ✓ Project Explorer: hiển thị các thành phần của ứng dụng đang thực hiện.
- ✓ Properties Window: Cửa sổ hiển thị các đặc tính (Properties) thiết kế của các

+ Điều khiển.

- ✓ Form Layout Window: xem trước hoặc thay đổi vị trí Form khi thực thi ứng dụng.
- ✓ Workspace: vùng làm việc của chương trình…
- ✓ Ngoài ra giao diện VB còn chứa rất nhiều các thành phần khác.

Để hiển thị thành phần nào bạn chọn trình đơn "View" và click chọn thành phần bạn muốn hiển thị.

Tuy nhiên, với những thành phần được giới thiệu trên đã đủ để giúp bạn xây dựng các ứng dụng của VB. Các thành phần trên sẽ được giới thiệu đầy đủ hơn trong các phần sau của tài liệu này.

Chú ý: Do VB là một ngôn ngữ lập trình theo kiểu hướng đối tượng (tuy chưa thật sự đầy đủ ý nghĩa), vì vậy để có thể làm việc với VB trước hết bạn phải biết cơ bản về khái niệm "Đối tượng"-Object.

2. Các điều khiển truyền thông

Mục tiêu :

- Trình bày chính xác các điều khiển truyền thông, đặc tính, các sự kiện

- Sử dụng được các đối tượng để tạo các Form

Các Điều khiển trong VB là một dạng của Đối tượng. Đó là những công cụ có sẵn giúp cho việc tạo giao diện của ứng dụng trở nên dễ dàng và nhanh chóng hơn, đây chính là đặc trưng của kiểu lập trình VISUAL hay RAD đã được đề cập ở trên. Các Điều khiển cũng bao gồm các thành phần của Đối tượng. Vì vậy, từ bây giờ khái niệm Điều khiển có thể hiểu đồng nhất với Đối tượng.

2.1. Textbox

Textbox hay còn gọi là edit field or edit control, hiển thị thông tin trong thời gian thiết kế hay lúc chương trình đang thực thi. Việc truy xuất thông tin của Textbox được thực hiện thông qua Properties "Text"

Biểu tượng	Hình ảnh	Properties	Method	Event	Ghi chú
ab|	Text1	**Name**: Tên đối tượng **Alignment**:canh lề 0:canh trái 1: canh phải 2: canh giữa **Backcolor**: màu nền **BorderStyle**: Viền 0: không viền 1: có viền nổi **Enable**: cho phép hoặc cấm sử dụng (true, false) **Height**: Chiều cao **Font**: Font của chữ trong Text **Forecolor**: Màu của chữ **Index**: chỉ số đối tượng (mảng đối tượng) **Left**: Khoảng cách lề trái **Lock**: cho phép hoặc cấm sửa thông tin (true hoặc false) **MultiLine**: True: cho phép xuống dòng Flase: không xuống dòng **ScrollBar**: Các thanh trượt 0: không sử dụng 1: Thanh trượt ngang 2: Thanh trượt dọc 3: Cả 2 thanh trượt **Text**: Nội dung của Textbox **Top**: Khoảng cách lề trên **Visible**: ẩn hiện đối tượng. **Width**: Chiều rộng Textbox	**Move** **Left,Top,[Width], [Height]** : di chuyển VD: Text1.Move 100, 200, 300,400 La: Di chuyển Text1 đến vị trí cách lề trái 100, lề trên 200, rộng 300, cao 400 **Chú ý: có thể bỏ các giá trị trong [ ] như Width hay Height** **Refresh**: làm tươi Textbox. **SetFocus**: Làm cho Textbox nhận Focus, và trở thành đối tượng được chú ý bởi Windows.	**Change**: Xảy ra khi thay đổi nội dung Text. **Click**: Xảy ra khi Click mouse vào Text. **DblClick**: Xảy ra khi Click đôi mouse vào Text. **GotFocus**: Xảy ra khi Text nhận Focus. **KeyDown**: Xảy ra khi Text đang có Focus và 1 phím được nhấn xuống. **KeyPress**: Xảy ra khi Text đang có Focus và 1 phím được nhấn . **KeyUp**: Xảy ra khi Text đang có Focus và 1 phím được thả. **LostFocus**: Xảy ra khi mất Fucos **MouseDown**: Xảy ra khi nhấn mouse xuống Text. **MouseMove**: Xảy ra khi có sự di chuyển mouse trên Text. **MouseUp**: Xảy ra khi thả mouse lên Text.	

2.2. CommandButton

Là các nút nhấn được sử dụng để bắt đầu, ngắt hoặc dừng một qúa trình nào đó

Biểu tượng	Hình ảnh	Properties	Method	Event	Ghi chú
		**Name**: Tên đối tượng **Caption**: Phần chữ hiển thị trên nút nhấn. **Backcolor**: màu nền **Default**: true: Enter =nhấn Button false: bỏ chức năng này. **Enable**: cho phép hoặc cấm sử dụng (true, false) **Height**: Chiều cao Button **Font**: Font của chữ Caption **Forecolor**: Màu của chữ **Index**: chỉ số đối tượng (mảng đối tượng) **Left**: Khoảng cách lề trái **Picture**: hình chèn vào Button. Chỉ có tác dụng khi Style=1 **Style**: 0-Standard : mặc định. 1-Graphic: cho phép chèn hình **Top**: Khoảng cách lề trên **Visible**: ẩn hiện đối tượng. **Width**: Chiều rộng Textbox	**Move Left,Top,[Width], [Height]** : di chuyển VD: Command1.Move 100, 200, 300,400 La: Di chuyển Command1 đến vị trí cách lề trái 100, lề trên 200, rộng 300, cao 400 **Chú ý: có thể bỏ các giá trị trong [ ] như Width hay Height** **Refresh**: làm tươi CommandButton. **SetFocus**: Làm cho CommandButton nhận Focus, và trở thành đối tượng được chú ý bởi Windows.	**Click**: Xảy ra khi Click mouse vào CommandButton. **GotFocus**: Xảy ra khi CommandButton nhận Focus. **KeyDown**: Xảy ra khi CommandButton đang có Focus và 1 phím được nhấn xuống. **KeyPress**: Xảy ra khi CommandButton đang có Focus và 1 phím được nhấn . **KeyUp**: Xảy ra khi CommandButton đang có Focus và 1 phím được thả. **LostFocus**: Xảy ra khi mất Focus. **MouseDown**: Xảy ra khi nhấn mouse xuống CommandButton. **MouseMove**: Xảy ra khi có sự di chuyển mouse trên CommandButton. **MouseUp**: Xảy ra khi thả mouse lên CommandButton.	

2.3. PictureBox

Là một điều khiển được sử dụng chứa các loại file hình tiêu chuẩn. Đặc

biệt
ta còn có thể vẽ lên các Form

Biểu tượng	Hình ảnh	Properties	Method	Event	Ghi chú
		**Name**: Tên đối tượng **AutoRedraw**:cho phép tự vẽ lại true:các nét vẽ trên PictureBox không mất khi thay đổi đối tượng. false: ngược lại **Backcolor**: màu nền **BorderStyle**: Viền 0: không viền 1: có viền nổi **DrawStyle**: quy định loại nét vẽ **DrawWidth**: độ lớn nét vẽ **Enable**: cho phép hoặc cấm sử dụng (true, false) **FillColor**: Màu tô **FillStyle**: Cách tô màu. **Height**: Chiều cao **Font**: Font của chữ trong Text **Forecolor**: Màu vẽ **Index**: chỉ số đối tượng (mảng đối tượng) **Left**: Khoảng cách lề trái **Picture**: hình hiển thị trong Form **ScaleMode**: đơn vị đo	**Circle**: Vẽ đường tròn trên PictureBox. VD: Picture1.Circle (100,200), 50 vẽ hình tròn tâm (100,200), bán kính 50. Trên Picture1 **Cls**: Xoá các hình vẽ trên PictureBox, không xóa hình nền. **Move Left,Top,[Width],[Height]** : di chuyển VD: Picture1.Move 100, 200, 300,400 La: Di chuyển Picture1 đến vị trí cách lề trái 100, lề trên 200, rộng 300, cao 400 **Chú ý: có thể bỏ các giá trị trong [ ] như Width hay Height** **Refresh**: làm tươi PictureBox. **Line (x1,y1)-(x2,y2), [color]** : Vẽ đường thẳng trong Picture từ (x1,y1) đến (x2,y2) **Point (x,y)** : lấy màu tại điểm (x,y)	**Click**: Xảy ra khi Click mouse vào PictureBox. **DblClick**: Xảy ra khi Click đôi mouse vào PictureBox. **GotFocus**: Xảy ra khi Form nhận Focus. **KeyDown**: Xảy ra khi Form đang có Focus và 1 phím được nhấn xuống. **KeyPress**: Xảy ra khi PictureBox đang có Focus và 1 phím được nhấn . **KeyUp**: Xảy ra khi Form đang có Focus và 1 phím được thả. **LostFocus**: Xảy ra khi mất Focus **MouseDown**: Xảy ra khi nhấn mouse xuống Picture **MouseMove**: Xảy ra khi có sự di chuyển mouse trên Picture. **MouseUp**: Xảy ra khi	

	Top: Khoảng cách lề trên	**Pset (x,y) , color** : chấm thả mouse lên Picture . điểm tại (x,y)
	Visible: ẩn hiện đối tượng.	**SetFocus**: Làm cho PictureBox nhận Focus, và trở thành đối tượng được chú ý bởi Windows.
	Width: Chiều rộng Textbox	

2.4. Form

Form hay biểu mẫu, là đối tượng quan trọng khi viết các ứng dụng. Form được xem là nền của giao diện, một ứng dụng phải có ít nhất một Form. Cũng giống như các Đối tượng khác, Form cũng có các Properties, Method và Event riêng, tuy nhiên Form có còn có 1 đặc điểm quan trọng khác là tất cả các Đối tượng nằm trên Form được coi là 1 thành phần của Form đó, như vậy có thể dung dấu "."để truy xuất các đối tượng trên Form. Vd: trên Form1 có 1 nút lệnh Command1, để đặt giá trị Left của Command1 là 100 ta sử dụng lệnh: Command1.Left=100 hay Form1.Command1.Left=100.

- Từ khoá "Me": từ khoá Me được sử dụng thay cho tên 1 Form nào đó trong 1 câu lệnh nếu câu lệnhđược viết ngay trong Form đó. Ví dụ trên có thể viết lại: Me.Command1.Left=100.

Properties	Method	Event	Ghi chú
Name: Tên đối tượng **AutoRedraw**:cho phép tự vẽ lại **true**:các nét vẽ trên Form không mất khi thay đổi đối tượng. **false**: ngược lại **Backcolor**: màu nền **BorderStyle**: Viền 0: không viền, không di chuyển, không thay đổi kích thước được. 1: có viền nổi,có thể di chuyển nhưng không thay đổi kích thước được. 2: dạng đầy đủ nhất. (mặc định). 3,4,5…Không quan trọng lắm. **Caption**: Tiêu đề sẽ xuất hiện trên Form khi Form xuất hiện. **DrawStyle**: quy định loại nét vẽ **DrawStyle**: quy định loại nét vẽ **Enable**: cho phép hoặc cấm sử dụng (true, false) **FillColor**: Màu tô khi vẽ **FillStyle**: Cách tô màu. **Font**: Font của chữ trong Text **Forecolor**: Màu nét vẽ	**Circle**: Vẽ đường tròn trên Form. VD: Form1.Circle (100,200), 50 vẽ hình tròn tâm (100,200), bán kính 50. Trên Form1 **Cls**: Xoá các hình vẽ trên Form, không xóa hình nền. **Move Left,Top,[Width], [Height]** : di chuyển VD: Form1.Move **100, 200, 300,400** Là: Di chuyển Form1 đến vị trí cách lề trái 100, lề trên 200, rộng 300, cao 400 **Chú ý: có thể bỏ các giá trị trong [ ] như Width hay Height** **Line (x1,y1)-(x2,y2), [color]** : Vẽ đường thẳng trong Form từ (x1,y1)	**Load**: Form được load lên. **Click**: Xảy ra khi Click mouse vào Form. **DblClick**: Xảy ra khi Click đôi mouse vào Form. **GotFocus**: Xảy ra khi Form nhận Focus. **KeyDown**: Xảy ra khi Form đang có Focus và 1 phím được nhấn xuống. **KeyPress**: Xảy ra khi Form đang có Focus và 1 phím	

	đến (x2,y2)	được nhấn .
	Refresh: làm tươi Form.	
Height: Chiều cao	**Point (x,y)** : lấy màu tại điểm (x,y)	
Icon: biểu tượng của Form khi xuất hiện.	**Pset (x,y) , color** : chấm điểm tại (x,y)	**KeyUp**: Xảy ra khi Form đang có Focus và 1 phím được thả.
Index: chỉ số đối tượng (mảng đối tượng)	**SetFocus**: Làm cho Form nhận Focus, và trở thành đối tượng được chú ý bởi Windows.	
Left: Khoảng cách lề trái		**LostFocus**: Xảy ra khi mất Fucos
MaxButton: cho phép hoặc cấm nút lệnh phóng to Form		**MouseDown**: Xảy ra khi nhấn mouse xuống Form.
MiDiChild: True: Form là một Form con Flase: Form là một Form độc lập.		**MouseMove**: Xảy ra khi có sự di chuyển mouse trên Form.
		MouseUp: Xảy ra khi thả mouse lên Form.
Picture: hình nền trên Form		**Unload**: Khi Form được tắt.
Top: Khoảng cách lề trên		
Visible: ẩn hiện đối tượng.		
Width: Chiều rộng Textbox		
...		

Ngoài ra còn rất nhiều Control khác thường được sử dụng khi thiết kế các ứng dụng với VB như :

- Image: cũng được sử dụng hiển thị hình ảnh như Form nhưng không thể vẽ được lên đối tượng này. Đặc biệt kích thước hình ảnh chèn vào có thể thay đổi theo kích thước của đối tượng Image nếu thuộc tính Stretch=true.
- Label: thường được sử dụng làm các nhãn, các đề tựa
- Check box: mang hai giá trị true hoặc false khi control này được đánh dấu chọn hoặc không chọn.
- OptionButton: Nút lựa chọn
- ComboBox: danh sách các lựa chọn dạng xổ xuống.
- ListBox: cũng chứa danh sách các lựa chọn nhưng dạng khung
- Frame: khung bao cho một nhóm control có đặc điểm chung nào đó.
- HScrollbar: thanh trượt ngang.
- VScrollbar: thanh trượt đứng.
- Nhóm DriveListBox, DirListBox, FileListBox: sử dụng truy xuấtđến các đườngdẫn trên đĩa.
- Timer: là bộ định thì, thời gian định thì chứa trong Properties Interval

…

Các Properties, Method, Event của các Control trên sẽ được giới thiệu trong từng các ví dụ có liên quan.

3. Cách gọi và viết các DLL

Mục tiêu :

- Sử dụng được các hàm API trong lập trình truyền thông và một số ứng dụng trong lập trình truyền thông.

- Trình bày được vấn đề xung đột các DLL

*- Tạo và sử dụng được các tệp *.DLL trong BASIC và DELPHI*

DLL là các thư viện liên kết động chứa các hàm và thủ tục mà ta có thể sử dụng để bổ sung cho những hàm còn thiếu của một ngôn ngữ lập trình.

Có hai loại DLL là Windows API DLL và Third-Party DLL

- Windows API DLL là những tập tin DLL đã được cài sẵn theo cáchệ điều hành
- Windows. Các tập tin Windows API DLL có những hàm, thủ tục được bổ sung một số chức năng mà VB chưa có.

Ngoài các Windows API DLL, các chương trình trên Windows có thể phải sử dụng các DLL khác (do các công ty hay cá nhân khác Microsoft phát triển) gọi là cácThird-Party DLL. Không như cácWindows API DLL , các Third-Party DLL cần được cài lên đĩa cứng trước khi sử dụng lần đầu. các Third-Party DLL thường được tạo ra bằng ngôn ngữ C.

Việc sử dụng các DLL có nhiều ưu điểm so với các thư viện tĩnh (thường gọi là Package):

- DLL tiết kiệm chỗ trống trên đĩa.
- DLL tiết kiệm bộ nhớ bằng cách sử dụng kỹ thuật chia sẻ hay còn gọi là ánh xạ bộ nhớ.
- Việc gỡ rối (Debug) trở nên dễ dàng hơn bởi các lỗi được cô lập trong DLL duy nhất.
- DLL luôn tỏ ra hiệu quả khi độ an toàn của nó được đảm bảo.

Khai báo DLL

Để có thể sử dụng các hàm, thủ tục trong một DLL, trước hết phải khai báo các hàm, thủ tục đó. Công thức khai báo chung trong VB là:

[Public| Private] Declare Sub|Function name Lib "Libname" [Alias "aliasname"]

vd: Public Declare Function PortIn Lib "io.dll" (ByVal Port As Integer) As Byte

Trong đó :

- Public : sử dụng toàn cục
- PortIn: tên hàm
- Io.dll: tên DLL

3.1. Tệp *.DLL và cách tiếp cận

3.1.1. 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. Phần 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

3.1.2. Cách tiếp cận với DLL của Windows

+ Cơ chế bảo vệ file của Windows:

Khi tiếp cận với các thư 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 bởi 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.

3.1.3. 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 bởi 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.

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

3.2.1. 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, bởi 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 WordPa 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 tiếp 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 được 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.

3.2.2. 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

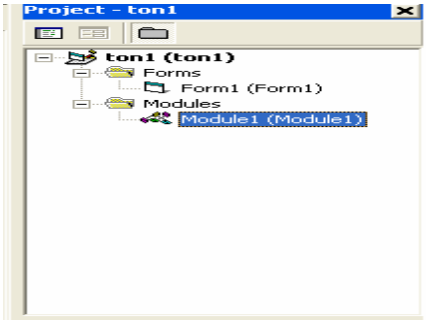
Tệp DLL có thể được sử dụng trong các hệ thống có thể lập trình hoàn toàn khác nhau. Ngoài ngôn ngữ được lập trình trong phần này, các hàm có thể được viết bằn ngôn ngữ C. Vì vậy một chương trình được viết ra một lần có thể được chuyển giao dễ dàng sang các hệ hthống lập trình khác. Ngoài ra, ta có thể 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

3.2.3. Gọi tệp *.DLL trong VisualBasic

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



Hình 1.3: Tạo một Module

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 pháp thử nghiệm dược nêu ra ở đây một mặt đẻ làm quen với các khái niệm cơ bản về một tệp DLL, mặt khấc để khảo sát tiến trình thời gian trong Windows.

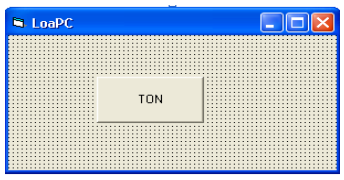
Loa được điều khiển qua bít 1 của cổng B trên vi mạch 8255. Vi mạch này chiếm các địa chỉ 60h (96 dec) trong vùng vào/ra của PC, cổng B nằm ở địa chỉ 97. Các nối ra cổng luôn có độ rộng 8 bít, vì thế 8 đường dẫn có thể chuyển cùng một lúc. Nhưng chỉ có đường dẫn bit 1 được phép thay đổi, bởi vì cổng 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 tệp 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 1.4: 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 đầu 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 đã cung cấp một phương tiện tốt hơn để nhận được thời gian trễ đến từng 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 đáng 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 ngoài có tên là PORT.PAS sau đó, thư viện có thể nạp vào project mới, mà không đòi hỏi sự quan tâm nhiều hơn đến các khai báo. Trong tệp PORT.DLL phải được đặt trong thư mục Window hoặc phải đặt trong thư mục có chứa chương trình exe cần chạy

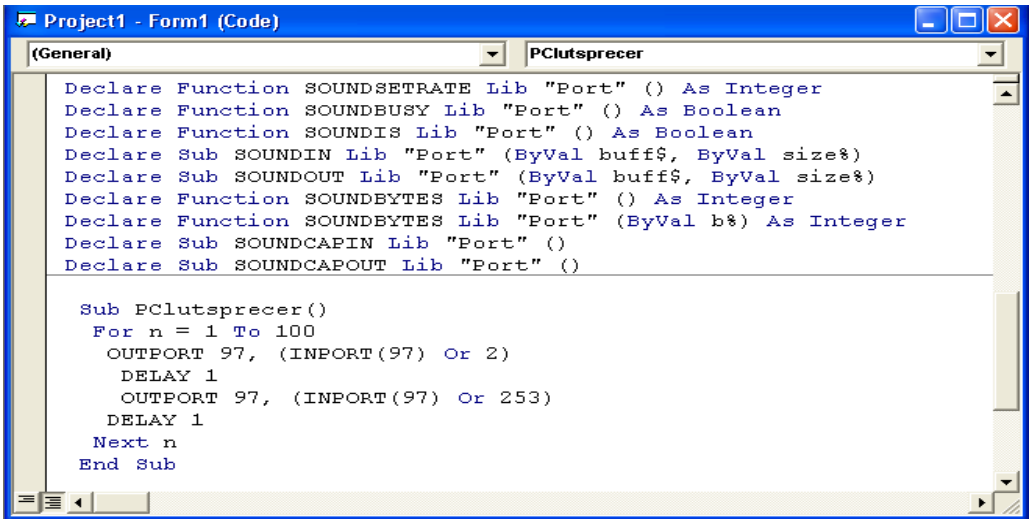
Đoạn chương trình sau đây là PORT.BAS với tất cả các khai báo dùng trong VB5:

```
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 1.5: Một macro trong Word 97

B. CÂU HỎI VÀ BÀI TẬP

Câu 1 : Trình bày một số đặc điểm của ngôn ngữ lập trình truyền thông ?

Câu 2 : Nêu một số đặc tính và sự kiện của các điều khiển của ngôn ngữ lập trình truyền thông ?

Câu 3 : Trình bày cách gọi và viết các DLL ?

Câu 4 : Trình bày một số ứng dụng của ngôn ngữ lập trình truyền thông ?

Câu 5 : Nêu ý nghĩa các thuộc tính của các điều khiển truyền thông ?

Câu 6 : Viết chương trình giới thiệu cơ cấu tổ chức của Trường THPT ?

Hướng dẫn viết chương trình :

+ Thiết kế Form chương trình

Tạo Form, thiết kế cho các điều khiển, xác lập các thuộc tính cho Form và 13 điều khiển

+ Viết và giải thích code của chương trình

* Viết code cho điều khiển có tên CmdADCD

```
Private Sub CmADSD_Click()
TxtHienthi = « CÔ NGUYỄN THỊ A »
End sub
```

* Viết code cho điều khiển có tên CmdBTCD

```
Private Sub CmADSD_Click()
TxtHienthi = « CÔ NGUYỄN THỊ B »
End sub
```

* Viết code cho điều khiển có tên CmdCTCD

```
Private Sub CmADSD_Click()
TxtHienthi = « CÔ NGUYỄN THỊ C »
End sub
```

…..

Câu 7 : Viết chương trình xem lịch của tháng ?

Hướng dẫn viết chương trình :

+ Thiết kế Form chương trình

- Tạo được form theo mẫu đề bài đưa ra

- Vào menu Project, chọn Components…hộp thoại Components hiện ra Click chọn Microsoft Calendar Control 8.0 Click nút Apply, điều khiển Calendar sẽ được đưa vào hộp công cụ

- Đưa điều khiển CommanButton vào trong Form, xác lập các thuộc tính cho Form và nút CommanButton

+ Viết Code cho điều khiển có tên Nutthoat

```
Private Sub Nutthoat_Click()
        End
End sub
```

BÀI 1: CÁC CÂU LỆNH VÀ ĐỐI TƯỢNG TRONG NGÔN NGỮ LẬP TRÌNH

Mã bài : MĐ38-02

❖ **Giới thiệu**

Bài này nhằm giới thiệu cho người học các kiến thức cơ bản về hằng, biến và cách khai báo biến, vận dụng các kiến thức đó người học có thể luyện tập làm một số bài tập cơ bản và nâng cao

❖ **Mục tiêu**

- Tạo khai báo chính xác hằng, biến.

- Viết đúng các câu lệnh trong ngôn ngữ lập trình, câu lệnh điều kiện, câu lệnh vòng lặp

- Thao tác và vận dụng tốt các đối tượng căn bản và truyền thông, các thuộc tính và sự kiện của các đối tượng.

- Tinh thần tương trợ, giúp đỡ nhau trong học tập.

❖ **Nội dung chính**

A. LÝ THUYẾT

1. Cách khai báo hằng biến

Mục tiêu:

- Trình bày được các khai báo hằng và biến

1.2. Khai báo biến

Được dùng để lưu trữ tạm thời các giá trị trong quá trình tính toán của chương trình. Giá trị mà ta lưu trữ trong biến có thể là những số nguyên, số thực, hay con chữ cái..mà ta gọi là kiểu biến. Vì là thành phần lưu trữ tạm thời nên biến sẽ tự động mất đi khi kết thúc chương trình hay thậm chí khi kết thúc một câu lệnh.

Trong VB, biến được khai báo theo cấu trúc:

Dim Tên Biến as Kiểu Biến.

+ Tên Biến phải:

- ✓ Dài không quá 255 ký tự
- ✓ Phải bắt đầu bằng chữ cái.
- ✓ Không có khoảng trắng hay ký hiệu +,-,*,/… trong tên biến.
- ✓ Không được trùng với từ khoá (keywords)của VB
- ✓ Không nên đặt tên trùng nhau.
- ✓ Có sự phân biệt giữa chữ viết HOA và chữ viết thường.

Kiểu biến	Kích thước biến	Khoảng giá trị
Byte	1 byte	0 to 255
Boolean	2 bytes	**True** or **False**
Integer	2 bytes	-32,768 to 32,767
Long (long integer)	4 bytes	-2,147,483,648 to 2,147,483,647
Single (single-precision floating-point)	4 bytes	-3.402823E38 to -1.401298E-45 for negative values; 1.401298E-45 to 3.402823E38 for positive values
Double (double-precision floating-point)	8 bytes	-1.79769313486231E308 to -4.94065645841247E-324 for negative values; 4.94065645841247E-324 to 1.79769313486232E308 for positive values
Currency (scaled integer)	8 bytes	-922,337,203,685,477.5808 to 922,337,203,685,477.5807
Decimal	14 bytes	+/-79,228,162,514,264,337,593,543,950,335 with no decimal point; +/-7.9228162514264337593543950335 with 28 places to the right of the decimal; smallest non-zero number is +/-0.0000000000000000000000000001
Date	8 bytes	January 1, 100 to December 31, 9999
Object	4 bytes	Any **Object** reference
String (variable-length)	10 bytes + string length	0 to approximately 2 billion
String (fixed-length)	Length of string	1 to approximately 65,400
Variant (with numbers)	16 bytes	Any numeric value up to the range of a **Double**
Variant (with characters)	22 bytes + string length	Same range as for variable-length **String**
User-defined (using **Type**) Kiểu dữ liệu do người dùng quy định.	tuỳ theo quy định riêng.	Phụ thuộc vào khoảng giá trị của kiểu dữ liệu được sử dụng.

Chú ý:

+ Biến phải được sử dụng đúng kiểu và đúng khoảng giá trị.

Ví dụ: Dim x as integer, nếu gán x="abc" sẽ xuất hiện lỗi vì x đã được khai báo
là một số nguyên nên không thể gán cho các ký tự.

+ Trong VB, biến có thể được khai báo như sau: Dim TenBien, tức không cần khai báo Kiểu biến, hoặc Dim TenBien as Variant khi đó kiểu biến là Variant tức tuỳ ý. VD: Dim x hay Dim x as Variant khi đó ta có thể gán x=1 hay x="abc"…vì ở đây x không có kiểu biến nhất định nên trình dịch tự hiểu kiểu của x khi nó được sử dụng.

+ Trong VB, một biến có thể được sử dụng mà không cần khai báo trước, tuy nhiên để tránh gặp lỗi khi lặp trình nên khai báo đầy đủ biến trước khi sử
dụng. Tốt nhất nên sử dụng từ khoá Option Explicit ở đầu chương trình.

Giới hạn sử dụng: nếu biến khai báo bên trong 1 đoạn chương trình với từ khoá Dim phía trước, biến đó chỉ có tác dụng bên trong đoạn chương trình đó, biến như vậy gọi là biến cục bộ(Local). Ngược lại nếu được khai báo trong phần Generalvới từ khoá Public thay cho Dim, biến sẽ có tác dụng trong toàn khối chương trình ta gọi là biến toàn cục(Global).

Nếu 1 biến toàn cục và 1 biến cục bộ trùng tên thì biến có tác dụng là biến cục bộ.

2. Các câu lệnh

Mục tiêu:

- Nắm vững được các câu lệnh vòng lặp và rẽ nhánh

- Sử dụng được các câu lệnh cơ bản trong lập trình

+ **Do…Loop:** là một vòng lặp vô tận. các câu lệnh bên trong Do…Loop sẽ được thực hiện lặp đi lặp lại mãi cho đến khi gặp lệnh Exit Do.

```
Do
[DoEvents]
Các câu lệnh
Loop
```

Để tránh bị treo máy do rơi váo vòng lặp vô tận, có thể sử dụng Doevents

+ Do While "điều kiện" …Loop: Trong khi "điều kiện" còn đúng thì còn thực hiện các câu lệnh bên trong Do và Loop

```
Do While     " điều kiện"
[Các câu lệnh]
[Các câu lệnh]
Loop
VD:   Dim Num
Num=0
Do While Num<10
Msgbox  "Xin Chao lan: "+str(Num),VBOKOnly,"DoLoop"
Num=Num+1
Loop
```

Kq: Hộp thoại sẽ xuất hiện 10 lần cho đến khi nào giá trị cúa Num>=10 thì thôi.

+ Do Until "điều kiện"…Loop": Thực hiện các câu lệnh bên trong cho đến khi "điều kiện" đã thỏa thì dừng.

```
Do Until      " điều kiện"
[Các câu lệnh]
[Các câu lệnh]
Loop
```

Chú ý: Ta có thể nhận thấy sự khác biệt cơ bản giữa Do While và Do Until là ở chổ: trong khi "điều kiện" trong Do While là điều kiện để thực hiện chương trình thì "điều kiện " trong Do Until lại là điều kiện để dừng chương trình.

+ Các Vòng lặp Do…Loop khác:

```
Do
[các câu lệnh]
[Exit Do]
[Các câu lệnh]
Loop [{While | Until} "điều kiện" ]
```

Cấu trúc này cũng giống như 2 cấu trúc trên tuy nhiên sự khác biệt là ở đây các câu lệnh thực hiện trước khi kiểm tra "điều kiện", nghĩa là luôn có ít nhất một lần các câu lệnh được thực hiện bất chấp điều kiện đúng hay sai.

Chú ý: Trong các vòng lặp Do…Loop, để thoát khỏi vòng lặp ta cò thể sử dụng câu lệnh

Exit Do.

+ Vòng lặp For…Next: Lặp lại một quá trình nào đó theo một số lần đã được chỉ định.

For counter = start To end [Step step]
[các câu lệnh]
Next [counter]
VD: Dim i
For i=1 to 10 Step 1
MsgBox "Xin Chao lan:"+str(i),VBOKOnly,"ForNext"
Next i

Kq: Hộp thoại sẽ xuất hiện 10 lần cho đến khi nào giá trị cúa i=10 thì thôi.

Chú ý : có thể sử dụng câu lệnh Exit For để thoát khỏi vòng lặp For…Next

+ Câu lệnh rẽ nhánh If…then…Else: Nếu điều kiện đúng thì thực hiện các câu lệnh theo sau Then, nếu không sẽ thực hiện các câu lệnh theo sau Else.

If "điều kiện" Then [các câu lệnh] [Else các câu lệnh khác]
hoặc
If "điều kiện" Then
[các câu lệnh]
Else
[các câu lệnh khác]
End if
VD: Dim x as Single ' khai báo biến x với kiểu là Single
x=Rnd(1)*10 ' cho x một giá trị ngẫu nhiên từ 0-10, hàm Rnd(1) trả ra một 'giá trị
ngẫu nhiên từ 0 đến 1.
If X>5 then MsgBox "So Lon",VBOKOnly,"IfThen" Else MsgBox "So
nho",VBOKOnly,"IfThen"

+ Sử dụng Elseif: Đây là dạng điều kiện lồng trong điều kiện.

If điều kiện Then
[các câu lệnh]
…..
[ElseIf điều kiện-n Then
[các câu lệnh elseif] ...
[Else

[các câu lệnh khác]]
End If

+ Cấu trúc Select case: Cũng là một cấu trúc rẽ nhánh sử dụng như là một kiểu lựa chọn.

Select Case tên biến
[Case giá trị-n
[các câu lệnh-n]] ...
[Case Else
[các câu lệnh khác]]
End Select
+ Ví dụ: Dim i
i=Rnd(1)*3
Select case i
Case 0 : MsgBox "So 0",VBOKOnly,"SelectCase"
Case 1 : MsgBox "So 1",VBOKOnly,"SelectCase"
Case 2 : MsgBox "So 2",VBOKOnly,"SelectCase"
Case 3 : MsgBox "So 3",VBOKOnly,"SelectCase"
End Select

+ With Đối tượng: khi muốn truy xuất cùng lúc nhiều Properties của 1 đối tượng, người ta sử dụng With đối tượng:

+ Ví dụ: Ví dụ sau thay đổi cùng lúc các thuộc tính Border, Alignment, Font, BackColor…của một đối tượng TextBox tên là Text1.

With Text1
.BorderStyle=1
.Alignment=2
.BackColor=QBColor(12)
.Font="VNI-Times"
End With

+ Bẫy lỗi (Error trapping): bẫy lỗi là việc đoán trước các lỗi có thể xẩy ra để hướng chương trình vào một lối khác, có thể tránh được lỗi đó.

On Error Goto Line: Nếu có lỗi sẽ nhảy đến dòng Line ngay trong đoạn chương trình.

+ VD:On Error Goto Loi
Open "TESTFILE" For Output As #1
[các câu lệnh khác]
Loi:
[các câu lệnh xử lí lỗi]

On Error Resume Next: Nếu có lỗi thì bỏ qua, nhảy đến câu lệnh tiếp theo sau. Dùng câu lệnh này thường không giải quyết lỗi triệt để, dễ làm nảy sinh lỗi liên hoàn rất khó xử lí.

Khi có lỗi xảy ra, giá trị nhận dạng lỗi sẽ được lưu tự động vào biến "Err", như vậy để nhận dạng lỗi và tìm ra cách giải quyết thích hợp chỉ cần tham khảo biến "Err".

3. Các đối tượng cơ sở và truyền thông

Mục tiêu:

- Trình bày được một số đối tượng cơ bản sử dụng trong chương trình

Lập trình hướng đối tượng là một khái niệm khó. Tuy nhiên với mức độ trung bình ta chỉ cần hiểu các thành phần cơ bản nhất của khái niệm này nhằm giúp ta xây dựng được các ứng dụng trong VB.

Đối tượng (Object) là một vật thực hữu, mỗi đối tượng có 1 tên gọi riêng.Ví dụ "Cat " là một đối tượng, nó thuộc lớp mèo (khái niệm lớp-Class ở đây không được đề cập). Mỗi đối tượng có các thuộc tính (Properties), Các hoạt động hay Phương thức (Method) và các ứng xử hay sự kiện (Event). Ví dụ "Chân", "mắt", "màu lông",… là các Properties của Cat, còn khả năng chạy, nhảy…là các Method của Cat, "Khát nước" là một Event của Cat.

Truy xuất đối tượng là việc đọc (GET), và đặt (SET) các Properties của đối tượng hay gọi các Method của đối tượng. Cú pháp:

Tên Đối tượng.Properties hoặc Đối tượng.Method

Ví dụ : bạn muốn biết màu lông của Cat, bạn dùng: Biến=Cat.màulông, nghĩa là màu lông của Cat đã được gán cho Biến, giá trị của Biến cũng chính là giá trị của màu lông.

Bây giờ nếu bạn muốn Cat phải chạy, bạn gọi: Cat.chạy vì "chạy" là một Method của Cat.

Như vậy có thể hiểu Method là các mệnh lệnh mà đối tượng có khả năng thực hiện khi được yêu cầu. Khác với các Properties ta chỉ có thể gọi các Method lúc chương trình đang thực thi (Runtime).

Riêng Event là các sự kiện xảy ra đối với đối tượng và các đáp ứng của đối tượng đối với sự kiện đó, các đáp ứng chính là code được người lập trình viết, đó cũng chính là nội dung quan trọng nhất khi lập trình. Phần này được sẽ được trình bày cụ thể trong các ví dụ thiết kế ứng dụng.

+ Ví dụ viết chương trình xuất ra chữ Hello

- Một số đối tượng cơ bản được sử dụng trong chương trình :

Control	Thuộc tính	Giá trị
CommandButtion	Name Caption Vị trí	cmdShow &Show Bên trong FraHello
CommandButtion	Name Caption Vị trí	cmdClear &Clear Bên trong FraHello

TextBox	Name Aligment Font Lock Text Vị trí	Txt -Hello 3-center Time New Roman True Bên trong FarHello
Frame	Name Caption Vị trí	frastyle select Type Bên trong FraCal
Label	Name Caption Vị trí	LblPlus + Bên trong FraCal
…	…	…

4. Các thuộc tính và sự kiện

Mục tiêu:

- Nắm vững được các thuộc tính và sự kiện trong các đối tượng

- Biết được các lỗi truyền thông

4.1. Các thuộc tính

+ Settings:

Xác định các tham số cho cổng nối tiếp. Cú pháp:

MSComm1.Settings = ParamString

MSComm1: tên đối tượng

ParamString: là một chuỗi có dạng như sau: "BBBB,P,D,S"

BBBB: tốc độ truyền dữ liệu (bps) trong đó các giá trị hợp lệ là:

110	2400	38400
300	9600 (măc định)	56000
600	14400	188000
1200	19200	256000

P: kiểm tra chẵn lẻ, với các giá trị:

Giá trị	Mô tả
O	Odd (kiểm tra lẻ)
E	Even (kiểm tra chẵn)
M	Mark (luôn bằng 1)
S	Space (luôn bằng 0)
N	Không kiểm tra

D: số bit dữ liệu (4, 5, 6, 7 hay 8), mặc định là 8 bit

S: số bit stop (1, 1.5, 2)

+ Ví dụ:

MSComm1.Settings = "9600,O,8,1" sẽ xác định tốc độ truyền 9600bps, kiểm tra parity chẵn với 1 bit stop và 8 bit dữ liệu.

+ CommPort:

Xác định số thứ tự của cổng truyền thông, cú pháp:

MSComm1.CommPort = PortNumber

PortNumber là giá trị nằm trong khoảng từ 1 ? 99, mặc định là 1.

+ Ví dụ:

MSComm1.CommPort = 1 xác định sử dụng COM1

+ PortOpen:

Đặt trạng thái hay kiểm tra trạng thái đóng / mở của cổng nối tiếp. Nếu dùng thuộc tính này để mở cổng nối tiếp thì phải sử dụng trước 2 thuộc tính Settings và CommPort. Cú pháp:

MSComm1.PortOpen = True | False

Giá trị xác định là True sẽ thực hiện mở cổng và False để đóng cổng đồng thời xoá

nội dung của các bộ đệm truyền, nhận.

+ Ví dụ: Mở cổng COM1 với tốc độ truyền 9600 bps

MSComm1.Settings = "9600,N,8,1"

MSComm1.CommPort = 1

MSComm1.PortOpen = True

+ Các thuộc tính nhận dữ liệu:

Input: nhận một chuỗi ký tự và xoá khỏi bộ đệm. Cú pháp:

InputString = MSComm1.Input

Thuộc tính này kết hợp với InputLen để xác định số ký tự đọc vào. Nếu InputLen = 0 thì sẽ đọc toàn bộ dữ liệu có trong bộ đệm.

InBufferCount: số ký tự có trong bộ đệm nhận. Cú pháp:

Count = MSComm1.InBufferCount

Thuộc tính này cùng dược dùng để xoá bộ đệm nhận bằng cách gán giá trị 0.

MSComm1.InBufferCount = 0

InBufferSize: đặt và xác định kích thước bộ đệm nhận (tính bằng byte). Cú pháp:

MSComm1.InBufferCount = NumByte

Giá trị măc định là 1024 byte. Kích thước bộ đệm này phải đủ lớn để tránh tình trạng mất dữ liệu.

+ Ví dụ: Đọc toàn bộ nội dung trong bộ đệm nhận nếu có dữ liệu

```
MSComm1.InputLen = 0
If MSComm1.InBufferCount <> 0 Then
InputString = MSComm1.Input
End If
```

+ Các thuộc tính xuất dữ liệu:

Bao gồm các thuộc tính Output, OutBufferCount và OutBufferSize, chức năng của các thuộc tính này giống như các thuộc tính nhập.

+ CDTimeout:

Đặt và xác định khoảng thời gian lớn nhất (tính bằng ms) từ lúc phát hiện sóng mang

cho đến lúc có dữ liệu. Nếu quá khoảng thời gian này mà vẫn chưa có dữ liệu thì sẽ gán thuộc tính CommEvent là CDTO (Carrier Detect Timeout Error) và tạo sự kiện OnComm.

Cú pháp:

```
MSComm1.CDTimeout = NumTime
```

+ DSRTimeout:

Xác định thời gian chờ tín hiệu DSR trước khi xảy ra sự kiện OnComm.

+ CTSTimeout:

Đặt và xác định khoảng thời gian lớn nhất (tính bằng ms) đợi tín hiệu CTS trước khi đặt thuộc tính CommEvent là CTSTO và tạo sự kiện OnComm. Cú pháp:

```
MSComm1.CTSTimeout = NumTime
```

+ CTSHolding:

Xác định đã có tín hiệu CTS hay chưa, tín hiệu này dùng cho quá trình bắt tay bằng phần cứng (cho biết DCE sẵn sàng nhận dữ liệu), trả về giá trị True hay False.

+ DSRHolding:

Xác định trạng thái DSR (báo hiệu sự tồn tại của DCE), trả về giá trị True hay False.

+ CDHolding:

Xác định trạng thái CD, trả về giá trị True hay False.

+ DTREnable:

Đặt hay xoá tín hiệu DTR để báo sự tồn tại của DTE. Cú pháp:

```
MSComm1.DTREnable = True | False
```

+ RTSEnable:

Đặt hay xoá tín hiệu RTS để yêu cầu truyền dữ liệu đến DTE. Cú pháp:

```
MSComm1.RTSEnable = True | False
```

+ NullDiscard:

Cho phép nhận các ký tự NULL (rỗng) hay không (= True: cấm). Cú pháp:

MSComm1.NullDiscard = True | False

+ SThreshold:

Số byte trong bộ đệm truyền làm phát sinh sự kiện OnComm. Nếu giá trị này bằng 0

thì sẽ không tạo sự kiện OnComm. Cú pháp:

MSComm1.SThreshold = NumChar

+ HandShaking:

Chọn giao thức bắt tay khi thực hiện truyền dữ liệu. Cú pháp:

MSComm1.HandShaking = Protocol

Các giao thức truyền bao gồm:

Protocol	Giá trị	Mô tả
ComNone	0	Không bắt tay (mặc định)
ComXon/Xoff	1	Bắt tay phần mềm (Xon/Xoff)
ComRTS	2	Bắt tay phần cứng (RTS/CTS)
ComRTSXon/Xoff	3	Bắt tay phần cứng và phàn mềm

4.2. Các sự kiện

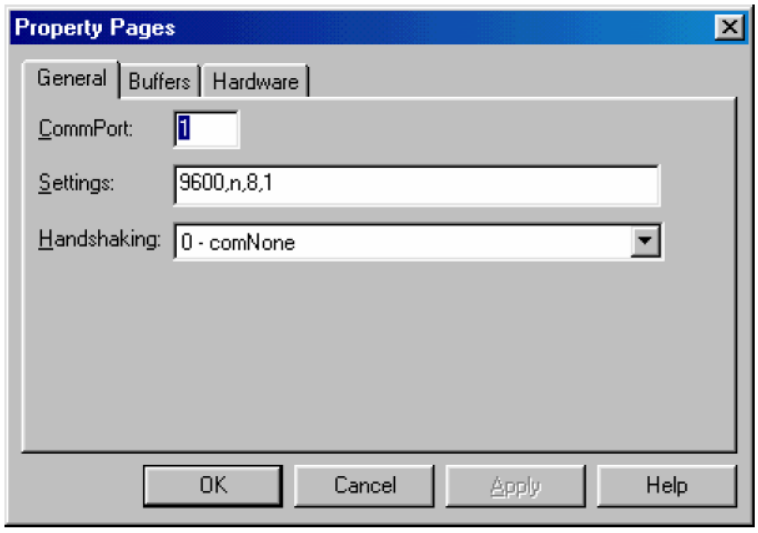
Sự kiện	Giá trị	Mô tả
ComEvSend	1	Đã truyền ký tự
ComEvReceive	2	Khi có ký tự trong bộ đệm nhận
ComEvCTS	3	Có thay đổi trên CTS (Clear To Send)
ComEvDSR	4	Có thay đổi trên DSR (Data Set Ready)
ComEvCD	5	Có thay đổi trên CD (Carrier Detect)
ComEvRing	6	Phát hiện chuông
ComEvEOF	7	Nhận ký tự kết thúc file

+ Sự kiện OnComm

Sự kiện OnComm xảy ra bất cứ khi nào giá trị của thuộc tính CommEvent thay đổi.

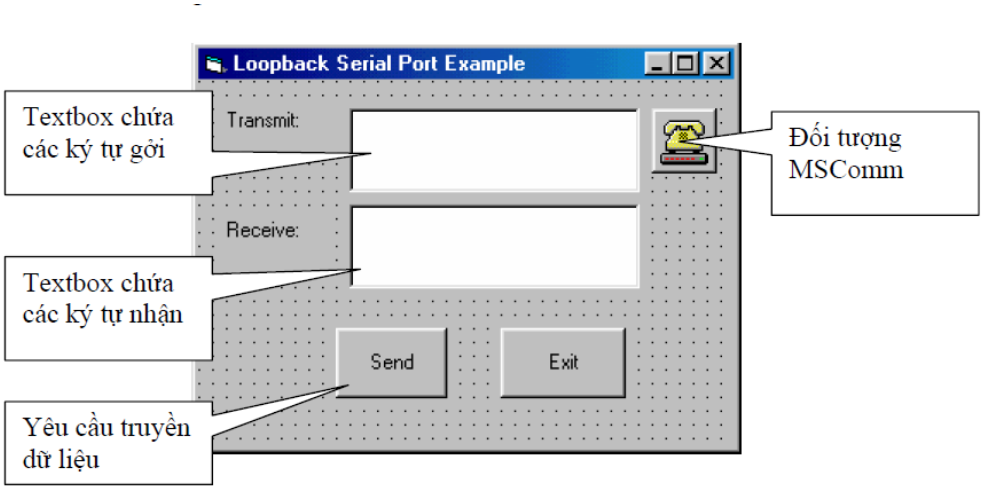
Các thuộc tính RThreshold và SThreshold = 0 sẽ cấm sự kiện OnComm khi thực hiện nhận hay gởi dữ liệu. Thông thường, SThreshold = 0 và RThreshold = 1.

Một chương trình truyền nhận đơn giản thực hiện bằng cách nối chân TxD với RxD của cổng COM1 (loopback). Phương pháp này dùng để kiểm tra cổng nối tiếp. Thuộc tính cơ bản của cổng nối tiếp:



Hình 2.1: Các thuộc tính cơ bản của MSComm

Cửa sổ chương trình thực thi:



Hình 2.2: Cửa sổ chương trình loopback

+ Các lỗi truyền thông:

Lỗi	Giá trị	Mô tả
ComBreak	1001	Nhận tín hiệu Break
ComCTSTO	1002	Carrier Detect Timeout
ComFrame	1004	Lỗi khung
ComOver	1006	Phần cứng không đọc ký tự trước khi gởi ký tự kế
ComCDTO	1007	Carrier Detect Timeout
ComRxOver	1008	Tràn bộ đệm nhận
ComRxParity	1009	Lỗi parity
ComTxFull	1010	Tràn bộ đệm truyền

5. Cách viết mã chương trình

Mục tiêu:

- Nắm được cấu trúc của một chương trình VB

- Biết cách sử dụng cửa sổ mã lệnh

❖ **Cấu trúc một chương trình trong VB:**

Private/Public Function/Sub TenChuongTrinh/TenEvent(Biến tham chiếu, tham trị)

Dim [Biến1] as Kiểubiến1

Dim [Biến2] as Kiểubiến2

…

Lệnh 1

Lệnh 2

….

Lệnh n

End sub

Mã chương trình được nhập vào trong cửa sổ mã, chúng ta thử viết một chương trình Quay xổ số, hầu hết các đối tượng mà chúng ta đã tạo ra đều "biết" phải làm việc như thế nào khi chương trình chạy, nên chúng sẵn sàng nhận sự nhập vào của người dùng và tự động xử lý các dữ liệu này.

Tính hoạt động vốn có của các đối tượng là một trong những lợi điểm mạnh nhất của Visual Basic - sau khi các đối tượng đã đặt trên một biểu mẫu một lần và các đặc tính đã được đặt, chúng sẵn sàng chạy mà không cần lập trình bổ xung

Tuy nhiên, "phần thịt" của chương trình trò chơi "Quay xổ số", tính toán các số ngẫu nhiên, hiển thị các số này trong các hộp và phát hiện sự trúng số vẫn còn bị chương trình bỏ sót. Loogic tính toán này có thể được bổ sung vào trong ứng dụng bằng cách sử dụng các mã lệnh chương trình được giải thích rõ ràng cho thấy chương trình sẽ làm gì trên từng bước đi

Bởi vì chương trình được điều khiển bằng các nút Spin(quay) và End (kết thúc) nên sẽ liên kết mã dùng cho trò chơi với các nút đó.

Cửa sổ mã là một cửa sổ đặc biệt trong môi trường lập trình mà ta sử dụng để nhập vào và soạn thảo các lệnh trong Visual Basic

5.1. Đọc đặc tính trong các bảng

Trong bài này chúng ta đã đặt các đặc tính cho chương trình Quay xổ số theo từng bước một. Sau này, các chỉ thị để đặt các đặc tính sẽ được giới thiệu trong khuôn mẫu bảng trong trường hợp việc thiết lập là đặc biệt khó khăn.

Ở đây là các đặc tính mà ta đã đặt như vậy trong chương trình Quay xổ số lập trình trong khuôn mẫu bảng, giống như trong một trường hợp mà sau này mà

Sau này vẫn còn đề cập đến.

Đối tượng	Đặc tính	Đặt
Command1	Tiêu đề (caption)	"Spin"
Command2	Tiêu đề	"End"
Label1, Lable2	BorderStyle	1 - Single cố định (Fixed Single)

Label3	Alignment (Căn lề) Font Tiêu đề	2 - Center Times New Roman, Bold, cỡ 24
Label4	Tiêu đề Font foreColor (màu chữ)	"Quay xổ số" Arial, Bold, cỡ 20 Màu tía (purple) tối (&H008000808)
Image1	Picture (Bức tranh) Kích thước ảnh (Stretch) Ẩn viện	"Wb6Sbs\BH_Ch19\coins.wmf" True False

Trong các bước sau, chúng ta sẽ nhập vào mã chương trình cho Quay xổ số trong cửa sổ mã.

5.2. Sử dụng cửa sổ mã

Nhấn kép lên nút lệnh End trên biểu tượng (Form). Cửa sổ mã xuất hiện

Lệnh End làm dừng việc thực hiện một chương trình

Nếu như cửa sổ nhỏ hơn cửa sổ đã chỉ ra trên hình 20-13, thì ta thay đổi kích thước nó bằng chuột. Kích thước chính xác là không quan trọng bởi vậy vì cửa sổ mã bao gồm các dải cuộn mà ta có thể sử dụng để kiểm tra các lệnh chương trình dài.

B. CÂU HỎI VÀ BÀI TẬP

Câu 1: Trình bày một số đặc điểm của biến, kiểu của biến và cách khai báo biến?

Câu 2: Trình bày đặc điểm về hằng, cách khai báo hằng?

Câu 3: Trình bày cú pháp, cách viết của câu lệnh điều kiện và câu lệnh vòng lặp?

Câu 4: Viết chương trình tìm năm âm lịch?

Câu 5: Viết chương trình từ điển Anh Việt - Việt Anh?

Câu 6: Viết chương trình khi chúng ta nhập vào một số nguyên dương thì màn hình sẽ in ra tất cả các số nguyên tố từ 2 cho đến số vừa nhập?

Câu 7: Viết chương trình kiểm tra 1 số có phải là một số nguyên tố hay không?
Câu 8: Viết chương trình tính n giai thừa?
Câu 9: Viết chương trình tìm ước số chung lớn nhất của 2 số?
Câu 10: Viết chương trình học tập có dùng MENU?

BÀI 2: LẬP TRÌNH THIẾT BỊ ẢO

Mã bài: MĐ38-03

❖ **Giới thiệu**

Sau khi đã tìm hiểu sâu về kỹ thuật lập trình Visual Basic đặc biệt là thư viện các hàm, ta có thể tạo ra các thiết bị ảo có mức độ phức tạp vừa phải. Các thiết bị này có thể đặt riêng lẻ hoặc trong khuôn khổ của một chương trình lớn. Để tiến hành các thiết bị ảo ta cần đến một số tệp để hình thành các điều khiển tùy biến được kết hợp với nhau theo cách dùng chia sẻ. Vấn đề đặt ra là làm thế nào để sử dụng các điều khiển này cho thích hợp với từng đồ án cụ thể.

❖ **Mục tiêu**

- Xây dựng được chương trình hiển thị số, máy phát tín hiệu, dao động ký nhớ số.
- Xây dựng được các chương trình điều khiển số
- Tự tin trong lập trình ghép nối máy tính

❖ **Nội dung chính**

A. LÝ THUYẾT

1. Các thiết bị hiển thị số

Mục tiêu:

- Biết được cấu tạo của bộ hiển thị số

- Viết được chương trình minh họa việc sử dụng bộ hiển thị số

Chúng ta bắt đầu với một loại thiết bị ảo đơn giản nhất: thiết bị hiện thị số, chẳng hạn như vôn kế hiển thị số. Các thiết bị này có thể xây dựng một cách đơn giản bằng một trường nhãn (Label – filed). Cần lưu ý là hình vẽ được in ra dưới dạng ảnh đen trắng, khi nhìn trên màn hình máy tính ảnh còn đẹp hơn thế nhiều. Ở phía bên trái, các trường nhãn đã được sử dụng để tạo ra các bộ hiển thị trên ba cửa sổ tương ứng với điện áp hiện tại, điện áp trung bình và điện áp bình phương trung bình (RMS: Root Mean Square)

Tận cùng phía bên phải là khả năng lựa chọn vùng đo được thiết kế theo kiểu các nút rađiô. Khi sử dụng nhóm nút 3D thay cho các nút rađiô thông thường, ta nhận được một ảnh gần như nổi của phím nhấn. Nhờ các phần tử 3D của phiên bản chuyên nghiệp của Visual Basic các thiết bị ảo có thể được thiết kế không tốn nhiều thời gian, các thiết bị này phản ứng một cách mềm dẻo và nhờ vậy trông gần giống như một thiết bị thật sự.

Trước hết ta đặt một trường nhãn lên một biểu mẫu (form) và đặt cho trường này một tên gọi thích hợp, chẳng hạn "Bộ chỉ thị số" hoặc "Volt_meter". Sau đó ta đặt mầu cho nền, chẳng hạn màu đen, còn chữ màu xanh lục, vàng hoặc đỏ. Các màu này là những màu thường gặp của điốt phát quang (LED), cũng là màu của các LED bảy thanh dùng làm bộ chỉ thị số. Đối với các bộ chỉ thị số ở giữa, trước hết ta đặt một phần tử mặt chỉ thị (panel) 3D, sau đó đặt đặc tính, chẳng hạn BevelWidth = 1, BorderWidth = 3, Bevellnter = Insert, BevelOuter = Raised. Tiếp theo ta lựa chọn phần tử và đặt một trường nhãn, trong mặt chỉ thị

Sau khi, bằng cách này chúng ta có được những ảnh nổi cần thiết, ta cần phải sử dụng các kiểu dữ liệu và các hàm sau đây, để hiểu rõ hơn chúng ta bắt đầu với việc tìm hiểu một môđun, có tên quy ước là "Measverar.bas"

```
Type DigiMeastype
        ADRes As Single
        'Độ phân giải của bộ biến đổi AD, thí dụ 10V/2048
        Curval As Single    'giá trị đo hiện tại'
        He_so As Single
        Format As String    'khuôn mẫu hiện thị'
        Don_vi As String    'Đơn vị thí dụ 'V''
        Max As Single       'giá trị cực đại dùng cho hiện thị'
End Type
Sub digimeasdisplay
        (display As Label value As Integer, digimeas As Digimeastype)
        Dim Gia_stri As single
```

```
        Gia_stri value * digimeas.ADRes
        IF Abs(Gia_stri)<digimeas.max
        Then
                Gia_stri = Gia_stri * digimeas.He_so
                Display = Format(Gia_stri, digimeas.format) + digimeas.Don_vi
        Else
                If display = "---------"
                Then
                ------------------------------
        Else
                Display = "----------"
                End If
        End If
        Digimeas_CurVal = value End Sub
End Sub
```

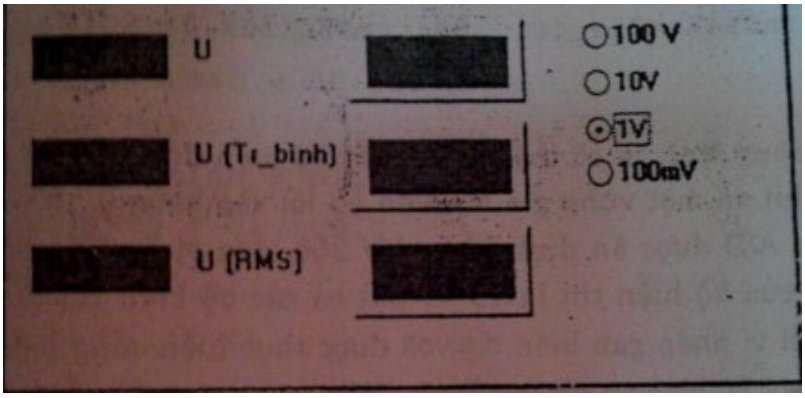
Cấu trúc DigiMeas Type cần được điền đầy bằng một loạt các giá trị, để ấn định các đặc điểm chi tiết của một thiết bị hiển thị số. Trước hết phải ấn định độ phân giải của bộ biến đổi A/D. Nếu chúng ta muốn sử dụng một bộ biến đổi A/D 8 bít với dải đo từ 0 đến 10V, thì độ phân dải phải đạt 10V/256. Lớp tiếp theo của cấu trúc được sử dụng ở bên trong để lưu trữ các thiết bị đo trong qua trình sử dụng. Tiếp theo nữa là một hệ số nhân, mà thông thường vẫn đặt là 1. Khuôn mẫu được lựa chọn là xâu khuôn mẫu dùng để định dạng bộ hiển thị, sau đó là đơn vị đo, cần được ấn định như một xâu và cuối cùng là vùng hiển thị cực đại.

Chương trình con digimeasDisplay lựa chọn một trường nhãn dùng cho bộ hiện thị, sau đó là giá trị đo, giống như nó được cung cấp từ bộ biến đổi A/D và một biến thuộc kiểu digimeas Type. Sau đó, giá trị lối ra của bộ biến đổi A/D được chuyển đổi sang một đại lượng vật lý (chẳng hạn điện áp và ký hiệu là V). Nếu như giá trị đo này lại nhỏ hơn giá trị cực đại thì được định dạng theo xâu khuôn mẫu (format) và cho hiển thị. Nếu như xảy ra tình trạng vùng đo cho phép bị vượt quá thì chương trính sẽ không hiển thị kết quả mà xuất ra các ký tự"-------" hoặc " " (xem hình 3-3), vì chương trình con digimeasDisplay được gọi nhiều lần kế tiếp nhau.

Chương trình được viết ra có tên là "Digivolt.mak" minh họa việc sử dụng bộ hiển thị số. Biểu mẫu của chương trình dùng làm thí dụ được mô tả trên hình 3-1 và 3-2. Ngoài ra biểu mẫu còn chứa một bộ định thời gian (Timer) để đọc và hiển thị các giá trị một cách tuần hoàn sau những thời gian cố định

Bởi vì các chương trình được đề cập đến ở đây đều mang tính minh họa, nên cả hai kiểu hiển thị số đều được tích hợp trong một biểu mẫu và được điều khiển bởi các dữ liệu giống nhau. Dòng hiển thị cao nhất chỉ ra giá

trị đo hiện tại, dòng ở giữa là giá trị trung bình và dưới cùng là giá trị bình phương trung bình RMS.



Hình 3.3: Bộ hiển thị số khi vùng đo bị vượt quá

Phần khai báo của biểu mẫu chứa các định nghĩa sau:

```
Dim digivolt As DigiMeastype
Dim digivoltTr_binh As DigiMeastype
Dim digivolrRMS As DigiMeastype

Dim rmsBuffer (0 To 50) As Single
Dim rmsBuffer (0 To 50) As Single
```

Ba khai báo đầu tiên định nghĩa các biến dùng cho các bộ hiển thị riêng lẻ, hai biến cuối cùng là cần có đối với việc tính các giá trị trung bình để lưu trữ trung gian. Khi nạp biểu mẫu, các biến digiVolt.digiVoltTr_binh.digiVoltRMS được điền đầy bằng các giá trị:

```
Sub Form_load ()
Dim mGia_tri As Single
        Digivolt.ADRes = 10# / 256
        Digivolt.Curval = 0#
        Digivolt.He_so = 1
        Digivolt.format = “00.00”
        Digivolt.Don_vi=”V”
        Digivolt.max=10
        digivoltTr_binh=digivolt
        digivoltRMS=digivolt

        Timer1.Enterval=1000
        Timer1.Enabled=True
        Call Tr_binhGia_triRek(mGia_tri, 0#, mwBuffer(0),50,1)
        Call rmsRek(mGia_tri, 0#, rmsBuffer(0), 50, 1)
End Sub
```

Từ các dòng lệnh chúng ta có thể nhận thấy bộ biến đổi A/D có độ phân giải bằng 8 bít và một vùng giá trị điện áp lối vào bằng 0…10 V. Từ đó độ phân giải A/D được ấn định bằng 10/256, đơn vị là "V", giá trị cực đại tức thời của bộ hiển thị là 10V. Tất cả các bộ hiển thị được xử lý đồng thời. Bởi vì phép gán biến digivoltRMS. Tiếp đó bộ phát khoảng thời gian với độ dài mỗi khoảng bằng 100 ms được thiết lập và kích hoạt. Cuối cùng bộ đếm dùng cho giá trị trung bình được khởi tạo.

Các nút rađio được sử dụng cho việc lựa chọn vùng đo. Các nút rađiô có tên trong chương trình tương ứng với vùng hiển thị (100 V:U100; 10V; 1V:U1 mV:U0100). Mỗi lần nhấn một nút rađiô có thể chuyển đổi vùng hiển thị cho tất cả các bộ hiển thị. Muốn thế các thủ tục sự kiện sau đã được định nghĩa:

```
Sub 0100_Click
Digivolt.Don_vi="V"
Digivolt.He_so = 1
        Digivolt.format = "000.0"
        Digivolt.max=100
        digivoltTr_binh=digivolt
        digivoltRMS=digivolt
End Sub
Sub 010_click()
Digivolt.Don_vi="V"
Digivolt.He_so = 1
        Digivolt.format = "00.00"
        Digivolt.max=10
        digivoltTr_binh=digivolt
        digivoltRMS=digivolt
End Sub
Sub U1_click()
Digivolt.Don_vi="V"
Digivolt.He_so = 1
        Digivolt.format = "0.000"
        Digivolt.max=1
        digivoltTr_binh=digivolt
        digivoltRMS=digivolt
End Sub
Sub U0100_click()
Digivolt.Don_vi="mV"
Digivolt.He_so = 1000
        Digivolt.format = "0.00"
        Digivolt.max=1
        digivoltTr_binh=digivolt
        digivoltRMS=digivolt
```

```
End Sub
```

Các thủ tục sự kiện này thật đơn giản, chỉ có đơn vị, khuôn mẫu hiển thị và vùng điều khiển và giá trị của hệ số được định nghĩa. Thí dụ đối với vùng hiển thị "1V" He_so=1; đơn vị ="V";format="0.000" và một giá trị cực đại max=1 đã được lựa chọn. Ở vùng hiển thị "100 mV" He_so phải được đặt lên 1.000, nhờ vậy mà các giá trị đo được hiển thị đúng. Sau đó, khi hiển thị giá trị đo tiếp theo các thông số hiển thị được cập nhật cho phù hợp với vùng hiển thị mới được thiết lập.

Việc đọc vào và hiển thị một giá trị đo được tiến hành qua thủ tục sự kiện của bộ định thời:

```
Sub Timer1_Timer()
Dim X As Integer
Dim mGia_tri As Single
Dim rmsGia_tri As Single
        X=readAD()
        Call Tr_binhgia_triRek(mGia_tri, X * 1#, mwBuffer(0), 50, 0)
        Call rmsrek(rmsGia_tri, X * 1#, rmsBuffer(0), 50, 0)
        Call digimeasDisplay(voltmeter(1), Int (mGia_tri).digivoltTr_binh)
        Call digimeasDisplay(vlotmeter(2), Int (rmsGia_tri).digivoltRMS)
        Call digimeasDisplay(voltmeterB(0), X, digivolt)
        Call digimeasDisplay(voltmeaterB(1),
Int(mGia_tri).digivoltTr_binh)
        Call digimeasDisplay(voltmeterB(2), Int(rmsGia_tri).digivoltRMS)
End Sub
```

Hàm read AD đọc một giá trị đo từ bộ biến đổi A/D. Tiếp đó giá trị trung bình và giá trị bình phương trung bình (RMS) được tính và được lưu trữ trong các biến mGia_tri hoặc rmsGia_tri. Tiếp theo các giá trị đo được hiển thị bằng việc gọi chương trình digimeasDisplay. Hàm readAD trong chương trình minh họa này chỉ là một hàm giả định.

```
Function readAD () As Interger
        readAD = 256 * Rnd
End Function
```

Chúng ta phải lập trình hàm này cho thích hợp với bộ biến đổi A/D đã được phép nối với máy tính PC để thực hiện các phép đo.

2. Máy phát tính hiệu hình sin

Mục tiêu:

- Trình bày được cấu tạo của máy phát tín hiệu hìn Sin

- Viết được chương trình mô tả việc sử dụng máy phát tín hiệu hình Sin

Chúng ta tìm hiểu một thí dụ khác để tạo ra một máy phát tín hiệu hình sin, có thể tạo ra các tín hiệu có dạng tương ứng với các hàm số khác nhau, với biên độ và tần số có thể đặt được. Thông thường các dạng tín hiệu sau cần được tạo ra trên một máy phát tín hiệu hoặc bộ tạo hàm:

+ Hàm sin
+ Hàm côsin
+ Hàm tang
+ Hình răng cưa
+ Hình tam giác
+ Hình chữ nhật

Trong đó, tần số, biên độ, offset, pha và tỷ số độ rộng xung/ trống(Ty_so xung/ trong: tỷ số giữa độ kéo dài và độ trống của xung) nếu cần đều có thể thay đổi được. Ngoài ra, việc hạn chế lối ra bằng các giới hạn phía trên và phía dưới có thể biến đổi được và là bật(cho phép) và cấm được. Khi đó, tín hiệu được tính năng trực quan (làm cho nhìn thấy được) cần được chuẩn hóa lên một dải biên độ và tần số chuẩn.

Sau đây chúng ta sẽ tiếp tục tìm hiểu nguyên tắc tạo ra tín hiệu và mặt bằng người dùng và cuối cùng là vấn đề thời gian thực. Khi dùng phần tử điều khiển của phiên bản chuyên nghiệp, một mặt máy với các phím bấm trông như nó có thể được thiết lập. Khi đó có rất nhiều thông số liên quan đến các phần tử 3 chiều (kỹ thuật 3D) như: Shadow Color và Shadow Style trong 3D Frame, Outline, RoundedCorner Aligment, ……có thể được lựa chọn. Mặt máy có thể được thể hiện bằng cách sử dụng các điều khiển tùy biến (Custom-control) bổ xung, như núm phím gạt điều chỉnh nhưng đối với nhiều ứng dụng một mặt.

Trước hết chúng ta tìm hiểu việc tạo ra dạng tín hiệu. Muốn vậy, trong mô đun "measverar" một loạt các hàm dùng cho các kiểu tín hiệu khác nhau đã được bổ sung vào, các kiểu này cung cấp giá trị lối ra thích hợp đối với một thời điểm cho trước và các thông số của máy phát tín hiệu.

Các thông số của máy phát tín hiệu được tóm tắt bên trong một cấu trúc SigGen Type:

```
Type SigGenType
        Amplitude As Single
        Frequency As Single
        Offset As Single
        Phase As Single
        Ty_so_xung_trong As Single 'tỷ số độ rộng/ độ trống xung
        Ulimit As integer                   0: Khong_han_che;
                                            1: co_han_che;
        Llimit As integer
                                            0: Khong_han_che;
                                            1: co_han_che;
```

```
        Ulim As Single              'giá trị giới hạn phía trên'
        Llim As Single              'giá trị giới hạn phía dưới'
End Type
```

Tần số và pha không nên viết trực tiếp trong cấu trúc này mà với các chương trình con sau:

```
Sub setfrequency (f As Single, siggen As SigGentype)
Dim ph As Single
    If (f < .0001) Then f=.001
    Ph=GetPhase(siggen)
    Siggen.frequency=f
    Call setphase (ph, siggen)
End sub
Sub setphase (p As Single, siggen As SigGentype)
    Siggen.Phase=(1#-(p/360#))/siggen.frequency
End sub
```

Để tạo ra một tín hiệu hình sin, coooossin và tang, có thể xem việc tạo ra tín hiệu hình sin được mô tả như một đại diện:

```
    Function GetPhase (siggen As SigGentype) As Single
        GetPhase=360#*(1#-(sigger.frequency*siggen.phase)
    End Function
```

Các tín hiệu hình sin được tạo ra bằng các hàm sau:

```
        Function sinusgent ( t as single, siggen, as siggentype)
        Dim as single
            Y=sin((t+siggen.phase) *siggen.frequency82*3.141428)
            Y= siggen.amplitude*y+siggen.offset
            Sinusgen=limit(y, siggen)
        End function
```

Các tín hiệu răng cưa được tạo ra bằng hàm sau:

```
    Function Rangcuagen(t as single, siggen as siggentype)
    Dim y as single
    Dim anzperiod as long
    Dim tnew as single
        If siggen.frequency=0 then siggen.frequency .0001
        Tnew = t + siggen.phase
        Anzperiod = int (tnew*siggen.frequency)
        Tnew=(tnew-(anzperiod/siggen.frequency))
        Y=tnew*siggen.frequency
        Y=siggen.amplitude*2*(y-5)+siggen.offset
        Rangcuagen = limit(y, siggen)
End function
```

Trước hết tần số được hạn chế đến một giá trị lớn hơn 0 để tránh phải thực hiện phép chia cho số 0. Tiếp đó thời gian chuyển giao được chuyển đổi

sang một giá trị thời gian trong khoảng kéo dài của chu kỳ(nhỏ hơn giá trị một chu kỳ). Với giá trị mới này của thời gian, tín hiệu răng cưa được tạo ra. Theo cách tương tự các tín hiệu hình tam giác và hình chữ nhật(xung vuông) được tạo ra:

```
Function tamgiacgen
        (t as single, siggen as siggentype)
Dim y as single
Dim anzperiod as long
Dim tnew as single
        If siggen.frequency=0 then siggen.frequency
        If siggen.ty_so_xung_trong=0 then
                Siggen.ty_so_xung_trong=0.0001
        End if
If siggen.ty_so_xung_trong=1 then
Siggen.ty_so_xung_trong=1-0.0001
Endif tnew=t+siggen.phase
Anzperiod=int(tnew*siggen.frequency)
If (tnew<(siggen.ty_so_xung_trong/siggen.frequency))
Then
Y=tnew*siggen.frequency/siggen.ty_so_xung_trong
Else
Y=(1#-tnew*siggen.ferquency)/(1#-siggen.ty_so_xung_trong)
End if
Y=siggen.amplitude*2*(y-0.5-siggen.offset)
                Tam_giacgen=limin(y,siggen)
End function
Function vuonggocgen
        (t as single, siggen as siggentype)
Dim y as single
Dim anzperiod as long
Dim tnew as single
        If siggen.frequency=0 then
        Siggen.frequency=0.0001
        End if
        if siggen.ty_so_xung_trong=1 then
        siggen.ty_so_xung_trong=1-0.0001 then
        end if
        tnew=t+siggen.phase
        anzperiod=int (tnew*siggen.frequency)
        tnew=(tnew-(anzperiod/siggen.frequency))
        if (tnew<(siggen.ty_so_xung_trong/siggen.frequency))
        then
```

```
        y=0
        else
        y=1
        end if
        y=siggen.amplitude*2*(y-0.5)+siggen.offset
        vuong_gocgen=limit(y,siggen)
end function
```

Tùy thuộc vào loại tín hiệu được lựa chọn (biến signaltype) mà các chương trình con tương ứng sẽ được gọi và các giá trị tính toán được trong trường signal được tạm thời lưu trữ. Việc gọi chương trình con Displaysignal lam hiện lên tín hiệu được tính toán trên dao động ký ở góc phía trên –bên trái của biểu mẫu.

Việc lựa chọn một loại tín hiệu được tiến hành bằng một dãy các nút trong nhóm "kiểu tín hiệu". Dãy nút này được định nghĩa là mảng điều khiển trong đó đặc trưng của từng kiểu tín hiệu. Chẳng hạn nút "tam giác" có chỉ số là 4, phù hợp với giá trị của hằng số tam giác. Khi ta nhấn một nút, thủ tục sau đây được thực hiện.

```
sub signalform_click
        (index as integer, value as integer)
        Signally=index showsig
End sub
```

Thủ tục này đặt ngay kiểu tín hiệu, sau đó gọi chương trình con Showsig. Khi ta thử lựa chọn các kiểu tín hiệu khác nhau, kế tiếp nhau, ta sẽ khẳng định là chỉ có biểu mẫu của tín hiệu thay đổi chứ không phải các thông số khác như chẳng hạn tần số hoặc biên độ.

Việc thiết lập các thông số khác được tiến hành nhờ một nút điều chỉnh kiểu phím gạt và nếu cần nhờ một hoặc nhiều nút. Ở đây chúng ta sẽ hạn chế trên một số điểm cơ bản về thủ tục sự kiện, bạn đọc quan tâm đến vấn đề này có thể tìm đọc trong các tài liệu đề cập chi tiết đến chủ đề sự kiện trong Visual Basic. Tần số và biên độ có thể được thay đổi bằng các nút từng bước đến 10 phần trăm (0,1 Hz hoặc 0,1 V) và hoàn toàn bằng số. Bằng một phím điều chỉnh theo kiểu gạt có thể tiến hành thiết lập chính xác từ 0,1 đến 10,0. Giá trị được đặt là tích của giá trị đã được đặt bằng phím gạt (chẳng hạn 2,7) và của giá trị được ghi ở bên cạnh nút đã được kích hoạt, chẳng hạn 0,1 Hz, như vậy sẽ nhận được giá trị: 2,7*0,1 Hz = 0,27 Hz. Để thể hiện việc hiển thị kết quả, chỉ có giá trị liên quan đến các phím điều chỉnh kiểu gạt được thể hiện. Độ phân giải của phím điều chỉnh kiểu gạt đã được lựa chọn sao cho đúng bằng 0,01. Khi phím gạt được thay đổi thì tín hiệu được thể hiện tính trực quan và giá trị trong từng bộ hiển thị cũng thay đổi trong chốc lát, khi nhấn một nút chỉ có giá trị trong bộ chỉ thị số (ngoại trừ nút Upperlimit và Lowerlimit).

Khi nhấn, chẳng hạn nút 1 Hz thủ tục sự kiện sau đây được thực hiện:

```
Sub f1_Click (Value As Integer)
        Freqfactor =1
Call setFrequency (freqFactor * res * freqBar.Value.SigGen)
Frequency = Format (SigGen.frequency,   "0000.000")
End Sub
```

Trước hết, tần số của máy phát tín hiệu SigGen bị thay đổi và tiếp theo là bộ chỉ thị số (trường text frequency) được hiện lên. Các thủ tục sự kiện dùng cho các nút tiếp theo để lựa chọn vùng tần số được viết ra theo cách hoàn toàn tương tự

Khi dịch chuyển thanh cuộn, thủ tục sau đây được thực hiện:

```
Sub FreBar_Change ()
        Call setfrequency (res * FreqBar.Value, DispSig)
        Call setfrequency (freqfactor * res * freqBar.Value.SigGen)
        Frequency = Format (SigGen.frequency,   "0000.0000")
        Showsig
End Sub
```

Thủ tục sự kiện dùng cho thanh cuộn để thiết lập biên độ, offset và tác dụng hạn chế có dạng hoàn toàn tương tự. Tất nhiên là khi thiết lập mức độ hạn chế còn phải kiểm tra xem tính năng này đã chắc chắn được kích hoạt

```
Sub LLimBar_Change
If SigGen. LLimit = 1 Then
DispSig.LLim = LlimBar * res
SigGen.LLim = LlimBar * res *ampfactor
        LowerLimit = Format (SigGen.Llim,   "000.0000")
 ShowSig
        End If
End Sub
```

Các giá trị của Offset, Upperlimit và Lowerlimit thông thường được tính toán theo cách nhân giá trị đã được thiết lập của từng phím điều khiển gạt với vùng giá trị của biên độ. Từ đó suy ra nếu ta thay đổi vùng giá trị của biện độ từ 10V sang 1V thì các giá trị hiện thời của Amplitude (biên độ), Offset, Upperlimit, Lowerlimit được chuyển đổi sang một giá trị nhỏ hơn 10 lần và diễn biến này cũng nhìn thấy ngay được trong từng bộ chỉ thị số. Thủ tục sự kiện chẳng hạn sẽ có đối với nút 10V, có thể viết như sau:

```
Sub a10_Click (Value As Integer)
        Ampfactor = 10
SigGen.Amplitude = ampfactor * res * ampBar.Value
Amplitude = Format (SigGen.Amplitude,   '0000.0000')
Call UpdateAllDependents
End Sub
```

Chương trình con UpdateAlldependents cập nhật offset, Lowerlimit và Upperland bằng cách gọi các thủ tục sự kiện tương ứng:

```
Sub UpdateAllDependents ()
        Call LlimBar_Change
        Call UlimBar_Change
        Call OffsetBar_Change
End Sub
```

Thủ tục sự kiện OffsetBar Change, chẳng hạn có thể viết như sau:

```
Sub OffsetBar_Change ()
        DispSig.Offset = OffsetBar * res
        SigGen.Offset = OffsetBar * res * ampFactor
        Offset = Format (SigGen.Offset,    "000.0000")
        showSig
End Sub
```

Khi tính độ dịch chuyển offset đối với máy phát tín hiệu hình sin SigGen, giá trị của thanh cuộn Offsetbar được nhân với vùng giá trị của biên độ AmpHe_so. Việc tạo ra tín hiệu tiến hành qua thủ tục của một bộ định thời Timer 1

```
Sub Timer1_Timer
Dim t As Single
Dim y As Single
        t  = Th_gian()
        y = SignalGen(t, siggen, signaltyp)
        DA (y)
End Sub
```

Hàm SigGen được định nghĩa trong môđun "Measveras.bas" như sau:

```
Function SignalGen
        (t As Single, siggen As SigGeritype, sigtyp As Integer)
Dim y As Single
Select Case (sigtyp)
Case Sinus y = SinusGen (t, siggen)
Case Cosinus y = CosinusGen(t, siggen)
Case Tangens y = TangensGen(t, siggen)
Case Rangcua y = RangcuaGen(t, siggen)
Case Tam_giac y = Tam_giacGen(t, siggen)
Case Vuong_goc y = Vuong_gocGen(t, sigger)
End Function
```

Chương trình con D/A cần được làm cho thích ứng với card biến đổi D/A. Do không có một card nào được phép có vùng điện áp lối ra từ - 100 V đến 100 V với độ phân giải 1 đến 10mV nên ta phải làm thích ứng máy phát tín hiệu theo cách này với phần cứng của ta. Cũng bằng các nút dùng cho dải biên độ ta có thể điều khiển một bộ khuếch đại có thể lập trình được ở phía sau bộ biến đổi D/A để làm tăng vùng các giá trị

3. Dao động ký nhớ số

Mục tiêu:

- Viết được chương trình mô tả việc sử dụng dao động ký nhớ số

Ở đây một lần nữa lại xuất hiện một vấn đề là Windows tỏ ra rất hạn chế về mặt thời gian thực do chưa áp dụng kỹ thuật lập trình đặc biệt (các bộ đếm thiết bị ảo). Chương trình dùng làm thí dụ trên hình đã được sử dụng bộ định thời trên Windows để đọc vào các giá trị đo một cách tuần hoàn. Khi không có một chương trình Windows khác cùng được kích hoạt, các điều khiển có cơ hội dành lấy khoảng thời gian dài hơn và nhờ vậy các quá trình xảy ra tương đối chậm có thể được thu thập dữ liệu một cách chính xác. Khi sử dụng Shareware-Timer, ta có thể quản lý các số liệu đo lường trong một khoảng thời gian quét ngắn hơn. Đối với các yêu cầu cao hơn ta nên chọn giải pháp sử dụng các card thu thập dữ liệu đo lường , có được từ các nhà cung cấp phần cứng máy tính, và có thể làm việc cùng với Visual Basic. Nhờ vậy có thể thu thập dữ liệu đo lường đến cỡ vài chục kHz với độ chính xác cao.

4. Điều khiển số

Mục tiêu:

- Viết được chương trình về điều khiển số

Lĩnh vực điều khiển số rất rộng vì thế vượt ra ngoài phạm vi trình bày của phần này, ngay cả đến việc bàn luận về các thiết bị ảo trong điều khiển số chỉ xin giới thiệu một chương trình dùng làm ví dụ cho một bộ điều khiển P (Hình ảnh minh họa chúng ta tham khảo trong cuốn sách "Lập trình ghép nối máy tính trong Window" chỉ ra mặt thiết bị ảo của chương trình này với tổng cộng 3 bộ điều chỉnh P

Với mỗi phím gạt ở trên cùng giá trị cần có trong vùng 0…10 được thiết lập và với mỗi phím gạt ở phía dưới cùng, hệ số P ở dưới cùng. Hai bộ hiện thị kia trong từng bộ điều chỉnh chỉ ra giá trị hiện thời và độ lệch của giá trị này với giá trị cần có

Trong một môđun có tên là "Regler.bas" ta đặt những dòng khai báo dùng để bổ sung thêm một bộ điều chỉnh P, cụ thể là:

```
Type PD_khienType
P As Single
SollGia_tri As Single
isGia_tri As Single          'giá trị tạm thời hiện tại'
setGia_tri As Single         'giá trị được đặt mới'
ulim As Single               'giới hạn trên'
As Single                    'giới hạn dưới'
End Type
```

Trong cấu trúc dữ liệu này còn có một hàm Pregler:

```
Function Pregler (hienthoi As Single, P As PreglerType)
        As single
Dim stell As Single
Dim x As Single
P.isGia_tri = hienthoi
X = (P.SollGia_tri - P.is Gia_tri)
Stell = x * p.p
If stell>p.ulim Then stell = p.ulim
If stell<p.llim Then stell = p.llim
P.setGia_tri = stell
Pregler = p.setGia_tri
End Function
```

Hàm này thực hiện việc tính độ chênh lệch giữa giá trị đang là và phải là rồi nhân giá trị này với hệ số P của bộ điều khiển P. Giá trị sau khi tính toán còn bị hạn chế bởi giới hạn trên và dưới sau đó xuất ra thành giá trị được đặt.

Mặt trước của mỗi bộ điều khiển là một khung nhóm (Gruppe frame), mỗi khung chứa hai thanh cuộn và một dãy các trường text. Mỗi một trong số các phần tử điều khiển này là bộ phận của một lớp các bộ điều khiển. Vì thế ta có thể dễ dàng tăng thêm số lượng của các bộ điều khiển P, bằng cách nhấn vào một khung nhóm và sao chép. Sau đó, khi ta nhấn vào Form (biểu mẫu) và thực hiện lệnh "chèn vào" thì một khung nhóm cho thích hợp, ngoài ra trong phần khai bào của Form:

```
Const res = 1
Const anzp = 3
Dim hienthoiderte (anzp) As Single
Dim preg (anzp) As preglerType
```

Phải làm thích ứng các hằng số anzp với số lượng các bộ điều khiển P cần có

Khi dịch chuyển thanh cuộn để thiết lập "giá trị cần có", thủ tục sự kiện sau đây được gọi:

```
Sub ParamBarSoll_Change (Index As Integer)
        Preg(Index).SollGia)_tri = parambarsolll(Index)*res
        SollGia_triP(Index) = Formats(preg(Index).SollGia_tri, "00.00")
End Sub
```

Thủ tục này làm thay đổi giá trị cần có SoollGia_tri của bộ điều khiển với chỉ số index tương ứng và cập nhật giá trị cần có được hiện thị. Tương tự, việc dịch chuyển thanh cuộn ParamBar - P sẽ làm thay đổi hệ số P_He_so của một bộ điều khiển và cập nhật giá trị tương ứng cần hiển thị:

```
Sub PraramBarP_Change (Index As Integer)
        pReg(Index) = Formats(pReg(Index).P, "00.00")
```

```
End Sub
```

Cứ sau một khoảng thời gian đã ấn định thì thủ tục sự kiện sau lại được gọi:

```
Sub Timerl_Timer()
Dim hienthoi As Single
Dim stell As Single
Dim Index As Integer
For Index = 0 To anzp -1
Hienthoi = AD(Index)
Stell = Pregler (hienthoi, pReg(Index))
isGia_triP(Index) = Formats(hienthoi, "00.00")
DeltaP(Index) = Formats(pReg(Index).SollGia_tri-hienthoi, "00.00")
      Call DA(Index, Stell)
Next Index
End Sub
```

Đối với mỗi bộ điều khiển P, thủ tục này đọc vào giá trị đang có bằng một chương trình con AD, chương trình con này tính giá trị cần đặt SetGia_tri và xuất ra bằng chương trình con AD. Ngoài ra, độ chênh lệch giữa giá trị đang có và giá trị cần có cũng được hiện thị. Các chương trình con AD và DA lại được làm thích ứng với bộ biến đổi A/ D cũng như bộ biến đổi D/ A. Các giá trị điện áp ở lối ra và lối vào được chuẩn hóa trong vùng từ -10 V đến 10 V. Trong chương trình dùng làm thí dụ , cả hai hàm lại được chọn la Dummy, các hàm này được ghép với một khoảng đều chỉnh giả định dưới dạng một mạch PTI đơn giản

Ở đây, vấn đề khoảng thời gian của bộ định thời được xem xét lại một lần nữa. Cần chú ý là: ở một bộ điều khiển số, thời gian quét cần phải được duy trì chính xác đến mức có thể, vì đây là một thông số quan trọng cần được quan tâm đến trong giai đoạn thiết kế chương trình. Tất nhiên là một bộ điều khiển P, thời gian quét không không làm thay đổi thông số của bộ điều khiển. Ngoài ra nên chú ý là một bộ điều khiển số trên thực tế thể hiện phản ứng giống như một bộ điều khiển analog, chừng nào mà thời gian quét nhỏ hơn (khoảng 5 đến 10 lần) so với các hằng số thời gian cơ bản của khoảng điều chỉnh. Trong trường hợp như vậy, một giá trị thời gian quét không cố định thực tế không ảnh hưởng đến quá trình điều khiển, chừng nào mà thời gian quét luôn thỏa mãn điều kiện đã được đặt ra. Đương nhiên là đối với một bộ điều khiển hai điểm hoặc ba điểm đơn giản thì kết luận này vẫn đúng

Như vậy, với bộ điều khiển P đã được trình bày trên cơ sở của một bộ định thời Windows ta có thể tạo ra các khoảng thời gian nhỏ nhất đến khoảng 100 đến 200 mili giây. Qua đó hàng loạt các khoảng điều khiển chậm có thể được điều khiển. Ngoài ra, tốc độ quét hầu như không phụ thuộc vào số lượng bộ điều khiển P, bởi vì quá trình cập nhật số liệu trê màn hình và chạy các thuật toán điều khiển diễn ra rất nhanh. Với một bộ định thời hoạt động

theo kiểu chia sẻ (Shareware) có độ phân giải cao, tốc độ quét có thể nhỏ đi đáng kể (ít nhất là một vài mili giây, khi bỏ qua thời gian cập nhật kết quả hiện thị). Tất nhiên là trong mỗi trường hợp, chỉ có một chương trình điều khiển được chạy

Điểm cuối cùng cần lưu ý là lập trình tạo ra các thiết bị ảo là một bài toán khó, việc trình bày dựa trên những hiểu biết ở mức đơn giản về Visual Basic còn khó khăn hơn. Những hy vọng không vì thế mà mất đi tính hấp dẫn vốn có của các thiết bị ảo, nhất là khi các thiết bị ảo đạt đến mức "đẹp hơn thật"

B. CÂU HỎI VÀ BÀI TẬP

Câu 1: Trình bày các thiết bị hiển thị số?
Câu 2:Trình bày nguyên lý hoạt động của máy phát tín hiệu hìn sin?
Câu 3: Nêu một số đặc điểm của dao động ký nhớ số?
Câu 4: Xây dựng chương trình hiển thị số?
Câu 5: Xây dựng chương trình điều khiển số?

BÀI 3: LẬP TRÌNH QUA CỔNG NỐI TIẾP

Mã bài: MĐ38-04

❖ Giới thiệu

Bài này nhằm giới thiệu cho người học những nội dung sau:

- Cổng nối tiếp
- Xuất trực tiếp ra dữ liệu số
- Cổng nối tiếp RS232
- Truyền dữ liệu nối tiếp và đồng bộ

❖ Mục tiêu

- Lập trình điều khiển thiết bị qua cổng nối tiếp
- Truyền được dữ liệu qua cổng nối tiếp và đồng bộ.
- Tự tin trong lập trình ghép nối máy tính

❖ Nội dung chính

A. LÝ THUYẾT

1. Cổng nối tiếp

Mục tiêu:

- Trình bày được cấu trúc của cổng nối tiếp
- Nêu được các đặc tính kỹ thuật của chuẩn RS-232

1.1. Cấu trúc cổng nối tiếp

Cổng nối tiếp được sử dụng để truyền dữ liệu hai chiều giữa máy tính và ngoại vi, có các ưu điểm sau:

- Khoảng cách truyền xa hơn truyền song song.

- Số dây kết nối ít.
- Có thể truyền không dây dùng hồng ngoại.
- Có thể ghép nối với vi điều khiển hay PLC (Programmable Logic Device).
- Cho phép nối mạng.
- Có thể tháo lắp thiết bị trong lúc máy tính đang làm việc.
- Có thể cung cấp nguồn cho các mạch điện đơn giản

Các thiết bị ghép nối chia thành 2 loại: DTE (Data Terminal Equipment) và DCE (Data Communication Equipment). DCE là các thiết bị trung gian như MODEM còn DTE là các thiết bị tiếp nhận hay truyền dữ liệu như máy tính, PLC, vi điều khiển, … Việc trao đổi tín hiệu thông thường qua 2 chân RxD (nhận) và TxD (truyền). Các tín hiệu còn lại có chức năng hỗ trợ để thiết lập và điều khiển quá trình truyền, được gọi là các tín hiệu bắt tay (handshake). Ưu điểm của quá trình truyền dùng tín hiệu bắt tay là có thể kiểm soát đường truyền.

Tín hiệu truyền theo chuẩn RS-232 của EIA (Electronics Industry Associations).

Chuẩn RS-232 quy định mức logic 1 ứng với điện áp từ -3V đến -25V (mark), mức logic 0 ứng với điện áp từ 3V đến 25V (space) và có khả năng cung cấp dòng từ 10 mA đến 20 mA.

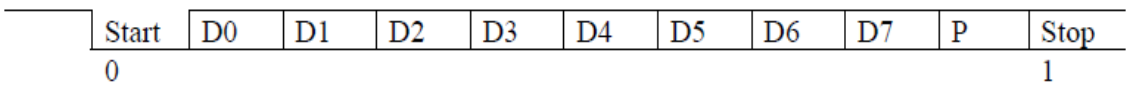
Ngoài ra, tất cả các ngõ ra đều có đặc tính chống chập mạch.

Chuẩn RS-232 cho phép truyền tín hiệu với tốc độ đến 20.000 bps nhưng nếu cáp truyền đủ ngắn có thể lên đến 115.200 bps.

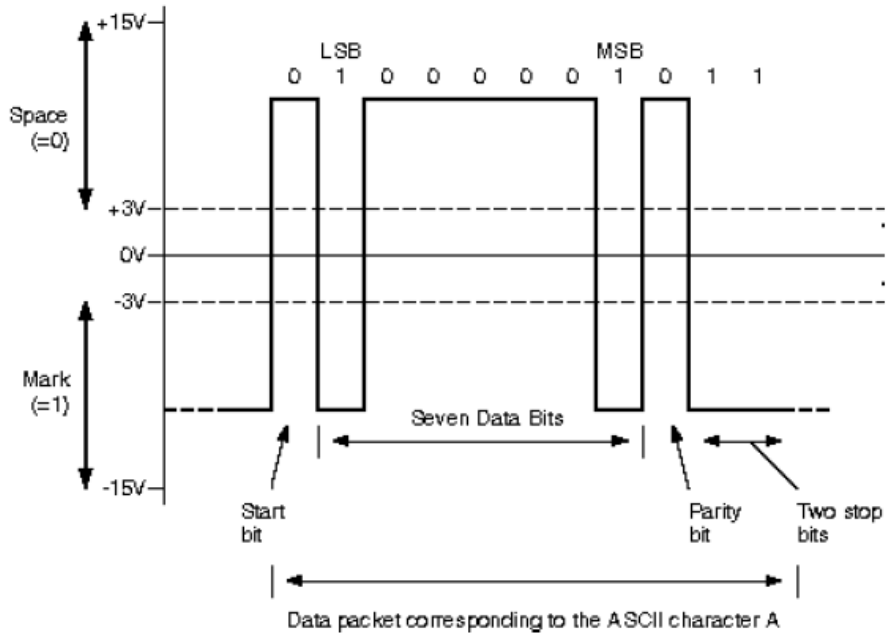
Các phương thức nối giữa DTE và DCE:

- Đơn công (simplex connection): dữ liệu chỉ được truyền theo 1 hướng.
- Bán song công (half-duplex): dữ liệu truyền theo 2 hướng, nhưng mỗi thời điểm chỉ được truyền theo 1 hướng.
- Song công (full-duplex): số liệu được truyền đồng thời theo 2 hướng.

Định dạng của khung truyền dữ liệu theo chuẩn RS-232 như sau:



Khi không truyền dữ liệu, đường truyền sẽ ở trạng thái mark (điện áp -10V). Khi bắt đầu truyền, DTE sẽ đưa ra xung Start (space: 10V) và sau đó lần lượt truyền từ D0 đến D7 và Parity, cuối cùng là xung Stop (mark: -10V) để khôi phục trạng thái đường truyền. Dạng tín hiệu truyền mô tả như sau (truyền ký tự A):



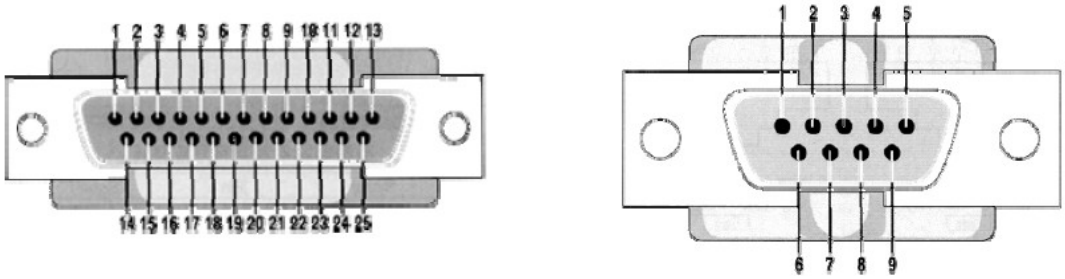
Hình 4.1: Tín hiệu truyền của ký tự 'A'

Các đặc tính kỹ thuật của chuẩn RS-232 như sau:

Chiều dài cable cực đại	15m
Tốc độ dữ liệu cực đại	20 Kbps
Điện áp ngõ ra cực đại	± 25V
Điện áp ngõ ra có tải	± 5V đến ± 15V
Trở kháng tải	3K đến 7K
Điện áp ngõ vào	± 15V
Độ nhạy ngõ vào	± 3V
Trở kháng ngõ vào	3K đến 7K

Các tốc độ truyền dữ liệu thông dụng trong cổng nối tiếp là: 1200 bps, 4800 bps, 9600 bps và 19200 bps.

+ Sơ đồ chân:



Hình 4.2: Sơ đồ chân cổng nối tiếp

Cổng COM có hai dạng: đầu nối DB25 (25 chân) và đầu nối DB9 (9 chân) mô tả như hình 4.2. Ý nghĩa của các chân mô tả như sau:

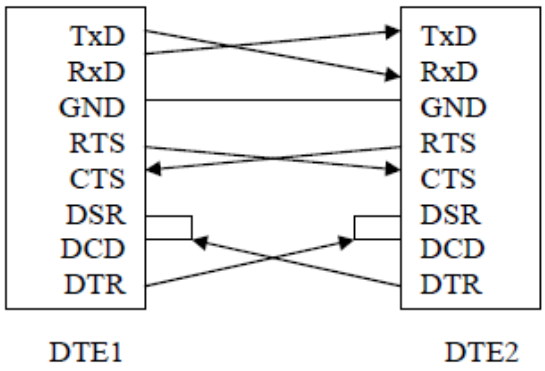
D25	D9	Tín hiệu	Hướng truyền	Mô tả
1	-	-	-	Protected ground: nối đất bảo vệ
2	3	TxD	DTE→DCE	Transmitted data: dữ liệu truyền
3	2	RxD	DCE→DTE	Received data: dữ liệu nhận
4	7	RTS	DTE→DCE	Request to send: DTE yêu cầu truyền dữ liệu
5	8	CTS	DCE→DTE	Clear to send: DCE sẵn sàng nhận dữ liệu
6	6	DSR	DCE→DTE	Data set ready: DCE sẵn sàng làm việc
7	5	GND	-	Ground: nối đất (0V)
8	1	DCD	DCE→DTE	Data carier detect: DCE phát hiện sóng mang
20	4	DTR	DTE→DCE	Data terminal ready: DTE sẵn sàng làm việc
22	9	RI	DCE→DTE	Ring indicator: báo chuông
23	-	DSRD	DCE→DTE	Data signal rate detector: dò tốc độ truyền
24	-	TSET	DTE→DCE	Transmit Signal Element Timing: tín hiệu định thời truyền đi từ DTE
15	-	TSET	DCE→DTE	Transmitter Signal Element Timing: tín hiệu định thời truyền từ DCE để truyền dữ liệu
17	-	RSET	DCE→DTE	Receiver Signal Element Timing: tín hiệu định thời truyền từ DCE để truyền dữ liệu
18	-	LL		Local Loopback: kiểm tra cổng
21	-	RL	DCE→DTE	Remote Loopback: Tạo ra bởi DCE khi tín hiệu nhận từ DCE lỗi
14	-	STxD	DTE→DCE	Secondary Transmitted Data
16	-	SRxD	DCE→DTE	Secondary Received Data
19	-	SRTS	DTE→DCE	Secondary Request To Send
13	-	SCTS	DCE→DTE	Secondary Clear To Send
12	-	SDSRD	DCE→DTE	Secondary Received Line Signal Detector
25	-	TM		Test Mode
9	-			Dành riêng cho chế độ test
10	-			Dành riêng cho chế độ test
11				Không dùng

1.2. Truyền thông giữa hai nút

Các sơ đồ khi kết nối dùng cổng nối tiếp:

Khi thực hiện kết nối như trên, quá trình truyền phải bảo đảm tốc độ ở đầu phát và thu giống nhau. Khi có dữ liệu đến DTE, dữ liệu này sẽ được đưa vào bộ đệm và tạo ngắt.

Ngoài ra, khi thực hiện kết nối giữa hai DTE, ta còn dùng sơ đồ sau:



Hình 4.3: Kết nối trong truyền thông nối tiếp dùng tín hiệu bắt tay

Khi DTE1 cần truyền dữ liệu thì cho DTR tích cực, tác động lên DSR của DTE2 cho biết sẵn sàng nhận dữ liệu và cho biết đã nhận được sóng mang của MODEM (ảo). Sau đó, DTE1 tích cực chân RTS để tác động đến chân CTS của DTE2 cho biết DTE1 có thể nhận dữ liệu. Khi thực hiện kết

nối giữa DTE và DCE, do tốc độ truyền khác nhau nên phải thực hiện điều khiển lưu lượng. Quá trinh điều khiển này có thể thực hiện bằng phần mềm hay phần cứng. Quá trình điều khiển bằng phần mềm thực hiện bằng hai ký tự Xon và Xoff.

Ký tự Xon được DCE gởi đi khi rảnh (có thể nhận dữ liệu). Nếu DCE bận thì sẽ gởi ký tự Xoff. Quá trình điều khiển bằng phần cứng dùng hai chân RTS và CTS. Nếu DTE muốn truyền dữ liệu thì sẽ gởi RTS để yêu cầu truyền, DCE nếu có khả năng nhận dữ liệu (đang rảnh) thì gởi lại CTS.

2. Xuất trực tiếp ra dữ liệu số

Mục tiêu:

- Trình bày được các thanh ghi có thể truy xuất trực tiếp ra dữ liệu số

Các cổng nối tiếp trong máy tính được đánh số là COM1, COM2, COM3, COM4 với các địa chỉ như sau:

Tên	Địa chỉ	Ngắt	Vị trí chứa địa chỉ
COM1	3F8h	4	0000h:0400h
COM2	2F8h	3	0000h:0402h
COM3	3E8h	4	0000h:0404h
COM4	2E8h	3	0000h:0406h

Giao tiếp nối tiếp trong máy tính sử dụng vi mạch UART với các thanh ghi cho trong bảng sau:

Offset	DLAB	R/W	Tên	Chức năng
0	0	W	THR	Transmitter Holding Register (đệm truyền)
	0	R	RBR	Receiver Buffer Register (đệm thu)
	1	R/W	BRDL	Baud Rate Divisor Latch (số chia byte thấp)
1	0	R/W	IER	Interrupt Enable Register (cho phép ngắt)
	1	R/W	BRDH	Số chia byte cao
2		R	IIR	Interrupt Identification Register (nhận dạng ngắt)
		W	FCR	FIFO Control Register
3		R/W	LCR	Line Control Register (điều khiển đường dây)
4		R/W	MCR	Modem Control Register (điều khiển MODEM)
5		R	LSR	Line Status Register (trạng thái đường dây)
6		R	MSR	Modem Status Register (trạng thái MODEM)
7		R/W		Scratch Register (thanh ghi tạm)

Các thanh ghi này có thể truy xuất trực tiếp kết hợp với địa chỉ cổng (ví dụ như thanh ghi cho phép ngắt của COM1 có địa chỉ là BACOM1 + 1 = 3F9h.

+ IIR (Interrupt Identification):

IIR xác định mức ưu tiên và nguồn gốc của yêu cầu ngắt mà UART đang chờ phục vụ. Khi cần xử lý ngắt, CPU thực hiện đọc các bit tương ứng để xác định nguồn gốc của ngắt. Định dạng của IIR như sau:

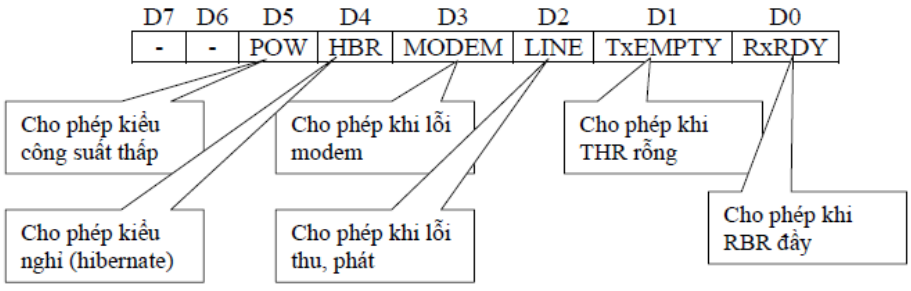
D7 D6	D5	D4	D3	D2 D1	D0
00: không có FIFO 11: cho phép FIFO	Cho phép FIFO 64 byte (trong 16750)	-	1: ngắt time-out (trong 16550)	Xác định nguồn gốc ngắt	0: có ngắt 1: không ngắt

D2	D1	Ưu tiên	Tên	Nguồn	D2 – D0 bị xoá khi
0	0	4	Đường truyền	Lỗi khung, thu đè, lỗi parity, gián đoạn khi thu	Đọc LSR
0	1	3	Đệm thu	Đệm thu đầy	Đọc RBR
1	0	2	Đệm phát	Đệm phát rỗng	Đọc IIR, ghi THR
1	1	1	Modem	CTS, DSR, RI, RLSD	Đọc MSR

(mức 1 ưu tiên cao nhất)

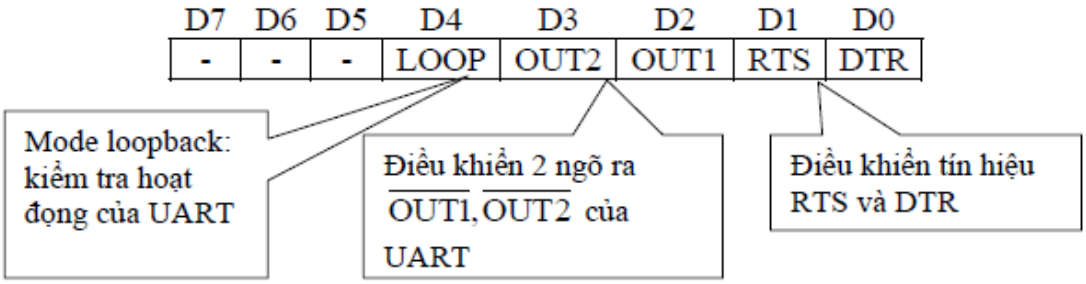
+ IER (Interrupt Enable Register):

IER cho phép hay cấm các nguyên nhân ngắt khác nhau (1: cho phép, 0: cầm ngắt)

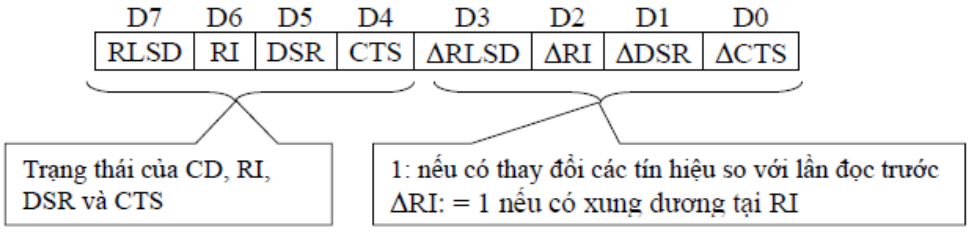


+ MCR (Modem Control Register):

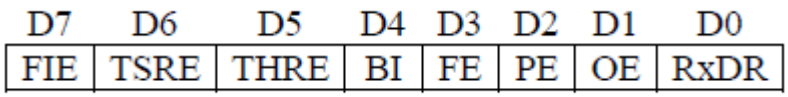
+



MSR (Modem Status Register):



+ LSR (Line Status Register):



FIE: FIFO Error – sai trong FIFO
TSRE: Transmitter Shift Register Empty – thanh ghi dịch rỗng (=1 khi đã phát 1 ký tự và bị xoá khi có 1 ký tự chuyển đến từ THR.
THRE: Transmitter Holding Register Empty (=1 khi có 1 ký tự đã chuyển từ THR – TSR và bị xoá khi CPU đưa ký tự tới THR).

+ LCR (Line Control Register):

D7	D6	D5	D4	D3	D2	D1	D0
DLAB	SBCB	PS2	PS1	PS0	STB	WLS1	WLS0

DLAB (Divisor Latch Access Bit) = 0: truy xuất RBR, THR, IER, = 1 cho phép đặt bộ chia tần trong UART để cho phép đạt tốc độ truyền mong muốn.

UART dùng dao động thạch anh với tần số 1.8432 MHz đưa qua bộ chia 16 thành tần số 115,200 Hz. Khi đó, tuỳ theo giá trị trong BRDL và BRDH, ta sẽ có tốc độ mong muốn.

Ví dụ: như đường truyền có tốc độ truyền 2,400 bps có giá trị chia 115,200 / 2,400 = 48d = 0030h ? BRDL = 30h, BRDH = 00h.

Một số giá trị thông dụng xác định tốc độ truyền cho như sau:

Tốc độ (bps)	**BRDH**	**BRDL**
1,200	00h	60h
2,400	00h	30h
4,800	00h	18h
9,600	00h	0Ch
19,200	00h	06h
38,400	00h	03h
57,600	00h	02h
115,200	00h	01h

SBCB (Set Break Control Bit) =1: cho phép truyền tín hiệu Break (=0) trong khoảng thời gian lớn hơn một khung

PS (Parity Select):

PS2	**PS1**	**PS0**	**Mô tả**
X	X	0	Không kiểm tra
0	0	1	Kiểm tra lẻ
0	1	1	Kiểm tra chẵn
1	0	1	Parity là mark
1	1	1	Parity là space

STB (Stop Bit) = 0: 1 bit stop, =1: 1.5 bit stop (khi dùng 5 bit dữ liệu) hay 2 bit stop (khi dùng 6, 7, 8 bit dữ liệu). WLS (Word Length Select):

WLS1	WLS0	Độ dài dữ liệu
0	0	5 bit
0	1	6 bit
1	0	7 bit
1	1	8 bit

+ Một ví dụ khi lập trình trực tiếp trên cổng như sau:

```
.MODEL SMALL
.STACK 100h
.DATA
    Com1 EQU 3F8h
    Com_int EQU 08h
    Buffer DB 251 DUP(?)
    Bufferin DB 0
    Bufferout DB 0
    Char DB ?
    Seg_com DW ? ;
    Off_com DW ?
    Mask_int DB ?
    Msg DB 'Press any key to exit$’
.CODE
    Main PROC
    MOV AX,@DATA
    MOV DS,AX
    MOV AH,35h
    MOV AL,Com_int
    INT 21h
```

```
MOV Seg_com,ES ;
MOV Off_com,BX
PUSH DS
MOV BX,CS
MOV DS,BX
LEA DX,Com_ISR
MOV AH,35h ;
MOV AL,Com_int
INT 21h
POP DS
MOV DX,Com1+3 ;
MOV AL,80h ;
OUT DX,AL ; truyền dữ liệu
MOV DX,Com1 ;
MOV AL,0Ch
OUT DX,AL
MOV DX,Com1+1
MOV AL,00h ;
OUT DX,AL ;
MOV DX,Com1+3 ; LCR = 0000 0011B
MOV AL,03h ; DLAB = 0, SBCB = 0
OUT DX,AL ; PS = 000
; STB = 0
; WLS = 11
MOV DX,Com1+4 ;
MOV AL,03h ; MCR = 0000 0011b ? DTR=RTS = 1
OUT DX,AL ;
MOV DX,21h ;
IN AL,DX ; D7 – D0
MOV Mask_int,AL ; =0: cho phép, =1:
AND AL,0EFh ; = 1110 1111b
OUT DX,AL ;
MOV AL,01h ; IER = 0000 0001b
MOV DX,Com1+1 ;
OUT DX,AL
MOV AH,09h
LEA Dx,Msg
INT 21h
Lap:
MOV AH,0Bh
INT 21h
CMP AL,0FFh
```

```
JE Exit
MOV AL,bufferin
CMP AL,bufferout
JE Lap
MOV AL,buffer[bufferout]
MOV char,AL
INC bufferout
MOV AL,bufferout
CMP AL,251
JNE Next
MOV bufferout,0
Next:
MOV DL,char ;
MOV AH,02h
INT 21h
MOV AL,char ;
MOV DX,Com1
OUT DX,AL
JMP Lap
Exit:
MOV AL,Mask_int
OUT 21h,AL ;
MOV DX,Off_com
MOV BX,Seg_com
MOV DS,BX
MOV AH,35h ;
MOV AL,Com_int
INT 21h
MOV AH,4Ch
INT 21h
Main ENDP
Com_ISR PROC
MOV DX,Com1+5 ;
IN AL,DX
AND AL,1 ;
JZ exit_ISR
MOV DX,Com1
IN AL,DX
MOV buffer[bufferin],AL
INC bufferin
MOV AL,bufferin
CMP AL,251
```

```
        JNE Exit_ISR
        MOV bufferin,0
        Exit_ISR:
        MOV AL,20h ;
        OUT 20h,AL
        IRET
        Com_ISR ENDP
    END Main
```

3. Cổng nối tiếp RS232

Mục tiêu:

- Trình bày được quá trình truyền thông nối tiếp qua cổng RS232

- Trình bày được cấu tạo của cổng nối tiếp RS232

- Viết được chương trình hiển thị được địa chỉ của cổng truyền thông

Chuńg ta thấy răng việc truyền thông nôí tiếp đoì hỏi rất nhiều thao tác phải thực hiện. Ta phải chuyển một byte dư liệu từ dạng song song thành dạng nôí tiếp (bởi vì hâù hết các hệ thống sô đều làm việc vơí các dư liệu ở dạng song song). Tiếp theo ta phải tạo ra một lơì tin theo đuńg định dạng cho trươć băng cách thêm các bit Start, Stop, Parity phu hợp. Sau đo ta mơí truyền dư liệu đi dươí dạng nôí tiếp. Cổng nôí tiếp của maý PC co một vi mạch chuyên dung để điều khiển truyền thông nôí tiếp. Do đo, khi sử dụng cổng RS-232 của maý PC để truyền thông, tất cả công việc của chuńg ta la gửi byte dư liệu cần truyền ra thanh ghi dư liệu của vi mạch nay. Sau đo mọi thao tác của qua trình truyền thông kể trên se được vi mạch thực hiện dựa theo nhưng thiết lập trong qua trình khởi tạo.

3.1. Qua trình truyền một byte dư liệu

Trong trạng thaí rôĩ, gia trị logic trên đươ ng truyền luôn băng 1. Để baó việc băt đâù truyền dư liệu, bên gửi đưa gia trị logic 0 lên đươ ng truyền trong khoảng thơì gian băng độ daì một bit. Bit đo gọi la bit Start. Khi truyền vơí tôć độ 300 baud, một bit co độ daì la 3,3 ms, trong khi vơí tôć độ 9600 baud thi co độ daì 0,1 ms.

Ngay sau bit Start, bên truyền gửi tiếp 8 bit dư liệu kê tiếp nhau, băt đâù băng bit LSB. Tiếp sau đo bên truyền se gửi tiếp một bit co gia trị logic 1 lên đươ ng truyền va duy tri trong khoảng thơì gian ít nhất la độ daì một bit. Ngay sau đo hoặc sau một khoảng thơì gian bât ky, bit Start tiếp theo se được gửi để băt đâù truyền một byte mơí.

3.2. Cổng nôí tiếp RS 232

Phần cưńg

Nhưng thuộc tính của phần cưńg

Những thiết bị sử dụng cáp nối tiếp cho việc truyền thông được chia làm hai loại. Đó´la` DCE (Data Communications Equipment) va` DTE (Data Terminal Equipment) DCE la`những thiết bị được sử dụng như một modem của chúng ta, bộ tiếp hợp TA, maý ve˜ ... trong khi đo´DTE lại được sử dụng như maý tính hoặc Teminal của chúng ta.

Những phaí minh vê` cổng nối tiếp của EIA (Electronics Industry Association) trong đo´tiêu biểu la`chuẩn RS 232C. No´đưa ra nhiều thông sô´ như:

A "Space" (logic 0) ở giữa +3V va`+12V.

A "Mark" (logic 1) ở giữa -3V va`-12V.

Vuùng giữa +3V va`-3V la`không xaćđịnh.

Liệt kê ở trên chỉ la`một phần của danh saćh của chuẩn EIA. Trong đo´bao gồm cả Line Capacitance, Maximum Baud Rates, ... Để biết chi tiết hơn xin tham khảo chuẩn EIA RS-232C. Tuy nhiên no´thật thu´vị để ghi nhơ´rằng Chuẩn RS -232C chỉ ro˜a maximum baud rate of 20,000 BPS!, điều ma`làm cho no´kha´ chậm trươć những tiêu chuẩn của ngaỳ nay. Một tiêu chuẩn mơí, RS -232D gần đây đa˜đươc phaí hành.

Cổng nối tiếp co´hai loại ỎsizeO˜ đo´la`bộ nối D-Type 25 chân va`bộ nối D-Type 9 chân, cả hai loại naỳ đều co´chung một đặc điểm khać hẳn vơí cổng maý in la`chô˜nối vơí maý in ở maý PC la`ổ căm, trong khi ở caćcổng nối tiếp lại la` phićh căm nhiều chân. Bên dươí la`bảng kết nối chân cho bộ kết nối 9 chân va` 25 chân D-Type.

D-Type-25 Pin No	D-Type-9 Pin No	Abbreviation	Full Name
Chân 2	Chân 3	TD	Transmit Data
Chân 3	Chân 2	RD	Receive Data
Chân 4	Chân 7	RTS	Request To Send
Chân 5	Chân 8	CTS	Clear To Send
Chân 6	Chân 6	DSR	Data Set Ready
Chân 7	Chân 5	SG	Signal Ground
Chân 8	Chân 1	CD	Carrier Detect
Chân 20	Chân 4	DTR	Data Terminal Ready
Chân 22	Chân 9	RI	Ring Indicator

Hình 4.4:Bảng D Type 9 Pin and D Type 25 Pin Connectors

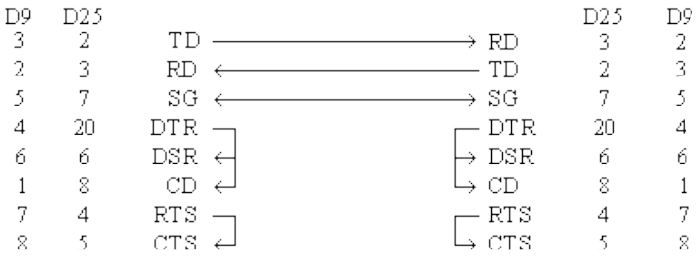
+ Chươć năng của caćchân

Chư˜viết tăt	*Tên đây đủ*	Chươć năng

TD	Transmit Data	Serial Data Output (TXD) - Đâù ra của dư liệu
RD	Receive Data	Serial Data Input (RXD) - Dư liệu được nhập vaò
CTS	Clear to Send	Baó răng Modem săn sang trao đổi dư liệu.
DCD	Data Carrier Detect	Khi naò modem phaí hiện ra một ỎCarrierO˜tư` một modem kêt thuć khać của the phone line, thi`Line nay trở thanh tích cực.
DSR	Data Set Ready	Thông baó vơí UART răng the modem săn sang thiêt lập một môí liên kêt .
DTR	Data Terminal Ready	Đây la`sự đôí lập vơí DSR. Baó vơí the Modem răng the UART săn sang để liên kêt .
RTS	Request To Send	Thông baó cho the Modem răng the UART săn sang để trao đổi dư liệu.
RI	Ring Indicator	Goes active when modem detects a ringing signal from the PSTN.

Null Modems

Một Null Modem được sử dụng để nôí cho hai DTE cung nhau. Nhưng modem nay thương được sử dụng như một cać để nôí mạng cho nhưng tro` chơi hoặc để chuyển giao giưã cać file maý tính sử dụng giao thức Zmodem Protocol, Xmodem Protocol, ... Điêu nay cưng co´thể được sử dụng vơí nhiêu Microprocessor Development Systems (hệ thông phaí triển bộ vi xử ly).



Hình 4.5: Sơ đồ`nôi dây Null Modem

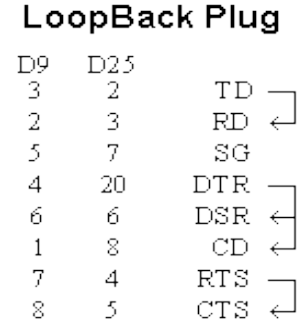
Trên đây la`phương phaṕ ưu tiên của việc nôí dây của một Null Modem. No´chỉ yêu câu 3 dây (TD, RD & SG) để măć được xuyên thẳng qua vi`vậy ảnh hưởng lơń đến chi phi´để sử dụng chạy caṕ dai. Nguyên ly´của thao tać thi`đơn giản co´ly´. Mục tiêu la`laṁ cho maý tính cho răng no´la`một modem hơn la`một computer khać. Any data transmitted from the first computer must be received by the second thus TD is connected to RD. Điêu thư´hai maý tính phải co´cưng cơ câú như vậy thi`RD được nôí tơí TD. Baó hiệu răng Signal Ground (SG) cưng phải được nôí sao cho cả hai grounds phổ biên tơí môĩ maý tính.

Data Terminal Ready (DTR) lặp lại khi chân Data Set Ready (DSR) va` Data Carrier Detect (DCD) co´mặt trên cả hai (on both computers) maý tính. Khi

chân the Data Terminal Ready ở mức tích cực thi`chân the Data Set Ready va` chân Data Carrier Detect ngay lập tức trở thành tích cực (active). Vaò thơi điểm nay maý tính cho răng the Virtual Modem săn sang được nôí va`phat hiện ra the carrier của modem khac.

Va`vấn đê`cân lo lăng bây giơ`la`chân the Request to Send va`chân Clear To Send. Trong khi cả hai maý tính giao thiệp vơí nhau ở cung một tôc độ, vi` vậy việc điêu khiển luông la`không cân thiêt vơí hai tuyên nay vi`chung co´thể kêt nôí cung nhau trên môĩ maý tính. Khi maý tính muôn gửi dư~liệu, no´xac nhận sự co´mặt của chân the Request to Send ở mức cao va`khi đo´no´moc nôí vơí chân the Clear to Send, luc nay ngay lập tức maý tính nhận được câu trả lơi răng no´co´thể gửi dư~liệu va`no´thực hiện ngay.

Chu´y´răng the ring indicator se~không kêt nôí tơí bât ky`caí gi`of each end. Đương nay chỉ sử dụng để chỉ cho maý tính biêt răng co´một ringing signal đang sử dụng đương dây the phone . Trong khi chung ta không co´một modem để kêt nôí tơí đương dây the phone thi`đương nay được ngưng kêt nôí.



Hình 4.6: Sơ đô`nôi dây Loopback Plug

Loopback plug thiêt bị nay co´thể trở nên vô cung dê~sử dụng khi viêt nhưng chương trình truyên thông sử dụng cổng nôí tiêp RS232. No´co´thể nhận va`truyên nhiêu tuyên đương cung nhau, vi`thê´ma`mọi thư´được truyên ra ngoai của cổng nôí tiêp thi`ngay lập tức nhận được bởi cung cổng đo´. Nêu chung ta nôí thiêt bị nay vơí cổng nôí tiêp nạp vao Terminal Program, thi`bât cư´ caí gi`chung ta đanh maý se~ngay lập tức được hiện lên trên man hình (displayed on the screen).

Xin chu´y´răng thiêt bị nay chưa được dự định cho việc sử dụng vơí nhưng chương trình Chẩn đoan (Diagnostic Programs) va`se~co´le~không lam việc. Bởi vi`nhưng chương trình ma`chung ta yêu câu khac nhau se~baó cho Loop Back plug caí ma`co´thể thay đổi tư`chương trình nay đên chương trình khac.

Tôc độ DTE / DCE

Chung ta đa~noí tom tăt vê`DTE va`DCE. Một thiêt bị đâu cuôí dư~liệu (Data Terminal Device) tiêu biểu la`một maý tính va`một thiêt bị truyên thông dư~liệu (Data Communications Device) tiêu biểu la`một Modem. Ngươi ta

thương nhắc đến tốc độ của DTE to DCE hoặc DCE to DCE. DTE to DCE là tốc độ giữa modem và máy tính của chúng ta, đôi khi được đề cập đến như là tốc độ của thiết cuối của chúng ta. DTE to DCE cần phải chạy ở một tốc độ nhanh hơn tốc độ của DCE to DCE. DCE to DCE là sự kết nối giữa các modem, đôi khi được gọi là tốc độ the line speed.

Hầu hết mọi người ngày nay có những modem với tốc độ 28,8K hoặc 33,6K. Bởi vậy chúng ta cần phải chờ đợi tốc độ của the DCE to DCE cũng như tốc độ của modem là 28,8K hoặc 33,6K. Suy cho cùng vì tốc độ cao của modem nên chúng ta mong muốn tốc độ của the DTE to DCE sẽ đạt đến khoảng 115,200 BPS (Maximum Speed of the 16550a UART). Những chương trình truyên thông mà chúng ta sử dụng đã đặt tốc độ cho DCE to DTE. Tuy nhiên, chúng chỉ có tốc độ 9,6 KBPS, 14,4 KBPS ... và coi như nó là tốc độ modem của chúng ta.

Những modem ngày nay có thể nén dữ liệu vào trong chúng (Data Compression). Điều này cũng như rất nhiều PK-ZIP nhưng phần mềm trong modem của chúng ta có thể nén và giải nén dữ liệu. Khi đưa ra đúng cách thức chúng ta có thể mong đợi việc nén số truyền với tỷ lệ 1:4 hoặc thậm chí còn cao hơn. Tỷ lệ nén dữ liệu 1:4 là rất tiêu biểu cho việc nén dữ liệu của những file văn bản. Nếu chúng ta chuyển những file văn bản đó ở 28,8K (DCE-DCE), thì khi modem nén nó chúng ta thực sự đang chuyển 115,2 KBPS giữa những computers và như vậy tốc độ của DCE-DTE là 115,2 KBPS. Như vậy do là lý do tại sao tốc độ của the DCE-DTE cần phải cao hơn tốc độ kết nối của modem.

Vài nhà sản xuất modem đã trích dẫn một tỷ lệ nén cực đại là 1:8. Để làm ví dụ cho lời trích dẫn đó họ đưa ra một modem mới với tốc độ 33,6 KBPS khi đó chúng ta có thể có một sự chuyển đổi cực đại 268,800 BPS giữa modem and UART. Nếu chúng ta chỉ có a 16550a nhưng chúng ta có thể làm 115,200 BPS tops, then you would be missing out on a extra bit of performance. Buying a 16C650 should fix your problem with a maximum transfer rate of 230,400 BPS.

Tuy nhiên, hãy khoan lạm dụng modem của chúng ta nếu chúng ta không có những tốc độ mong muốn. Đó là những tỷ lệ nén cực đại. Trong vài trường hợp cá biệt nếu chúng ta cố gắng gửi cho một file nén, modem của chúng ta có thể mất nhiều thời gian hơn nén nó, vì vậy chúng ta có tốc độ truyền chậm hơn tốc độ kết nối của modem. Nếu điều này xảy chúng ta nên cố gắng tắt việc nén dữ liệu của chúng ta lại. Lúc này cần phải cố định trên những modem mới hơn. Một vài file nén dễ dàng hơn những file khác vì vậy bất kỳ file nào mà nén đơn giản thì tự nhiên sẽ có một tỷ lệ nén cao hơn.

Điều khiển Luồng (Flow Control)

Như vậy nếu tốc độ của DTE to DCE là nhanh hơn gấp vài lần tốc độ của DCE to DTE the PC có thể gửi dữ liệu tới modem của chúng ta tại 115,200BPS. Sớm hay muộn dữ liệu sẽ bị mất khi bộ đệm bị tràn, trường hợp này điều khiển luồng sẽ được sử dụng. Điều khiển luồng có hai dạng cơ bản, phần cứng (hardware) hoặc phần mềm (software).

Điêù khiển luông phân mêm (Software flow control), đôi khi được biểu thị như Xon/Xoff sử dụng hai dạng ky´tự Xon va`Xoff. Xon thương cho biết bởi nhưng ky´tự của the ASCII 17 trong khi đo´ky´tự the ASCII 19 được sử dụng cho Xoff. Nhưng modem chỉ co´một bộ đệm nhỏ vi`thê´khi maý tính Ởphủ đâỳO˜no´ Modem gửi một ky´tự Xoff để baó cho maý tính dưng công việc gửi dư˜liệu. Khi modem cho nhiêù dư˜liệu hơn, no´gửi một ky´tự Xon va`maý tính se˜gửi nhiêù dư˜liệu hơn. Kiểu điêù khiển luông thê´nay co´nhiêù lợi thê´răng no´ không yêu câù bât ky`Ởbước điện baó naò như nhưng ky´tự được gửi qua nhưng đương TD/RD. Tuy nhiên môĩ ky´tự yêu câù liên kêt chậm mât 10 bits điêù đo´ co´thể laìm chậm việc truyên thông lại.

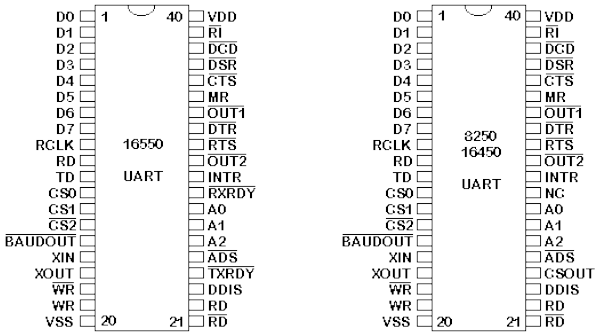
Điêù khiển luông phân cưng (Hardware flow control) cuñg được biêt như điêù khiển luông RTS/CTS flow control. No´sử dụng hai dây trong caṕ nôí tiêṕ của chuńg ta hơn la`truyên thêm nhưng ky´tự trong đương dư˜liệu của chuńg ta.

Điêù khiển luông Phân cưng cuñg được biêt như điêù khiển luông RTS / CTS. Vi`vậy điêù khiển luông phân cưng (hardware flow control) se˜không laìm chậm việc truyên thông lại như việc sử dụng Xon-Xoff does. Khi maý tính muôń gửi dư˜liệu no´se˜điêù khiển hoạt động của đương the Request to Send. Nêú modem co´ở phoàng cho dư˜liệu naỳ, thi`modem se˜băng việc điêù khiển hoạt động của đương the Clear to Send va`maý tính băt đâù gửi dư˜liệu. Nêú modem không co´ở phoàng thi`no´se˜không gửi tín hiệu cho Clear to Send.

The UART (8250 and Compatibles)

UART stands for Universal Asynchronous Receiver/Transmitter. Its the little box of tricks found on your serial card which plays the little games vơí modem va`nhưng thiêt bị kêt nôí khać. Đa sô´cać card se˜co´the UART's tích hợp vaò trong nhưng mạch điện tử chip khać ma`cuñg co´thể điêù khiển cổng song song của chuńg ta, cổng games, floppy hoặc ổ điã cưńg (hard disk drives) va`tiêu biểu la`nhưng thiêt bị surface mount. The 8250 series, bao gôm the 16450, 16550, 16650, & 16750 UARTS la`nhưng kiểu thương thâý trên PC của chuńg ta. Vê`sau chuńg ta se˜xem xeť nhưng kiểu khać, nhưng kiểu ma`co´thể được sử dụng trong nhưng thiêt bị homemade của chuńg ta va`projects.

Hình 4.7: Những sơ đồ`Chân cho 16550, 16450 & 8250 UARTs



The 16550 la`chip tương thích với the 8250 & 16450. Chỉ khac ở hai chân24 va`29. Chân 24 trên 8250 la`việc lựa chọn chip ở ngoai ma`chức năng chỉ la`việc Ỏchỉ baóO˜tới nêu chip hoật động hoặc không. Chân 29 không được kêt nôi trên the 8250/16450 UARTs. The 16550 đưa vào hai chân mới trong no´. Đo´la`Transmit Ready va`Receive Ready ma`co´thể thực thi với DMA (Direct Memory Access). Những chân nay co´hai kiểu thao tac khac nhau. Mode 0 hô˜ trợ việc chuyển giao đơn DMA trong khi mode 1 hô˜trợ Multi-transfer DMA.

Mode 0 cuñg được gọi la`mode 16450. Mode nay được lựa chọn khi bộ đệm FIFO được vô hiệu hoa´qua bit 0 của the FIFO Control Register hoặc khi bộ đệm the FIFO được cho phep nhưng DMA Mode Select = 0. (Bit 3 của FCR) Trong mode nay RXRDY la`tich cực ở mức thâp khi ít nhât một characters (Byte) co´mặt trong the Receiver Buffer. RXRDY se˜không hoạt động ở mức cao khi không co´nhiêu characters tôn tại trong the Receiver Buffer. TXRDY se˜hoạt động ở mức thâp khi không co´characters trong the Transmit Buffer. No´se˜không hoạt động ở mức cao sau khi characters/byte đâu tiên được tải vao trong the Transmit Buffer. Mode 1 la`khi bộ đệm FIFO được kich hoạt va`the DMA Mode Select = 1. Trong mode 1, RXRDY se˜hoạt động ở mức thâp khi the trigger level la`reached hoặc khi 16550 Time Out xảy ra va`se˜quay trở lại trạng thai không hoạt động khi không co´characters trong the FIFO. TXRDY se˜được kich hoạt khi không co´characters co´mặt bên trong the Transmit Buffer va`se˜không được kich hoạt khi the FIFO Transmit Buffer la`hoan toan Full.

Chân	**Tên**	*a. Lời ghi chu´*
Chân 1:8	D0:D7	Data Bus
Chân 9	RCLK	Receiver Clock Input. Tân sô´đâu vao nay cân phải cân băng với the receivers baud rate * 16
Chân 10	RD	Nhận dư˜liệu (Receive Data)
Chân 11	TD	Truyên dư˜liệu (Transmit Data)
Chân 12	CS0	Chip Select 0 - Active High
Chân 13	CS1	Chip Select 1 - Active High
Chân 14	nCS2	Chip Select 2 - Active Low
Chân 15	nBAUDOUT	Baud Output - Output from Programmable Baud

		Rate Generator. Frequency = (Baud Rate x 16)
Chân 16	XIN	Đâù vaò External Crystal Input – Sử dụng cho Baud Rate Generator Oscillator
Chân 17	XOUT	Đâù ra External Crystal Output
Chân 18	nWR	Write Line – Inverted (Đảo)
Chân 19	WR	Write Line - Not Inverted (không đảo)
Chân 20	VSS	Kêt nôí tơí Common Ground
Chân 21	RD	Read Line - Inverted
Chân 22	nRD	Read Line - Not Inverted
Chân 23	DDIS	Vô hiệu hoa´bộ phận điều khiển (Driver Disable). Chân nay rơi vaò mưć thâp khi CPU đọc tư`UART. Co´thể kêt nôí tơí Bus Transceiver trong trương hợp bus dư˜liệu co´dung lượng cao.
Chân 24	nTXRDY	Transmit Ready
Chân 25	nADS	Xung địa chỉ (Address Strobe). Sử dụng nêú tín hiệu không ổn định trong suôt qua´trình đọc hoặc ghi cycle
Chân 26	A2	Address Bit 2
Chân 27	A1	Address Bit 1
Chân 28	A0	Address Bit 0
Chân 29	nRXRDY	Receive Ready
Chân 30	INTR	Interrupt Output
Chân 31	nOUT2	User Output 2
Chân 32	nRTS	Request to Send
Chân 33	nDTR	Data Terminal Ready
Chân 34	nOUT1	User Output 1
Chân 35	MR	Master Reset
Chân 36	nCTS	Clear To Send
Chân 37	nDSR	Data Set Ready
Chân 38	nDCD	Data Carrier Detect
Chân 39	nRI	Ring Indicator
Chân 40	VDD	+ 5 Volts

Hình 4.8: Bảng Pin Assignments for 16550A UART

Tât cả cać chân của UARTs đêù thiích hợp vơí TTL. Bao gôm TD, RD, RI, DCD, DSR, CTS, DTR va`RTS ma`tât cả cać giao diện trong đo´la`serial plug của chuńg ta, typically a D-type connector. Vi`vậy RS232 Level Converters (ma`chuńg ta se˜noí cụ thể sau) đa˜đươc sử dụng. Caí nay thông thương la`the DS1489 Receiver va`the DS1488 as the PC has +12 and -12 volt rails ma`co´thể

sử dụng bởi những thiết bị nay. Trình chuyển đổi The RS232 se˜chuyển đổi tín hiệu the TTL vaò trong RS232 Logic Levels.

The UART yêu câù một Clock để chạy. Nếu chuńg ta xem xeť card nôí tiếp của chuńg ta a common crystal tìm thấy cuñg la`a 1.8432 MHZ hoặc a 18.432 MHZ Crystal. The crystal bên trong được kêt nôí tơí chân XIN-XOUT của the UART sử dụng thêm một sô´thành phần ma`giuṕ đơ˜the crystal để khởi động oscillating. Clock nay se˜ được sử dụng cho chương trình the Programmable Baud Rate Generator la`nhưng giao diện trực tiếp bên trong mạch chuyển đổi thơì gian (the transmit timing circuits) nhưng không trực tiếp bên trong mạch receiver thơì gian (the receiver timing circuits). Đôí vơí việc kêt nôí ngoaì nay được laìm tư`chân 15 (BaudOut) đến chân 9 (Receiver clock in). Chu´y´răng tín hiệu clock se˜ở tại Baudrate*16.

Nêú chuńg ta thực sự nghiêm tuć trong việc nghiên cứu tìm hiểu vê` 16550 UART xuć tiến sử dụng trong PC của chuńg ta, thi`haỹ đê`xuất việc downloading một bản sao của trang tính dư˜liệu PC16550D tư` National Semiconductors Site. Trang tính dư˜liệu (Data sheets) thi`sẵn co´trong dạng mâũ .PDF vi`thê´chuńg ta se˜cân Adobe Acrobat Reader để đọc nhưng điêù đo.Texas Instruments co´released the 16750 UART ma`co´64 Byte FIFO's. Trang tính dư˜ liệu cho TL16C750 săn co´để dung trong Texas Instruments Site.

Types of UARTS (For PC's) (deleted).

Registers của cổng nôi tiếp

Port Addresses & IRQ's

Tên	**Địa chỉ**	**IRQ**
COM 1	3F8	4
COM 2	2F8	3
COM 3	3E8	4
COM 4	2E8	3

Hình 4.9: Bảng 3 Standard Port Addresses

Trên la`bảng standard port addresses. Chuńg laìm việc trong đa sô´cać PC. Nêú chuńg ta tình cơ`hay may măń sở hưũ một IBM P/S2 ma`co´micro-channel bus, thi`chuńg ta mong đợi một sự thiêtlập khać của địa chỉ va`IRQ.Giống như cổng LPT, dư˜liệu cơ sở cho cać cổng COM co´thể đọc tư`Vuṇg Dư˜liệu BIOS (BIOS Data Area).

Start Address	**Function**
0000:0400	COM1's Base Address
0000:0402	COM2's Base Address
0000:0404	COM3's Base Address
0000:0406	COM4's Base Address

Hình 4.10: Bảng COM Port Addresses in the BIOS Data Area

Trên la` bảng cho thấy địa chỉ ma` chuńg ta co´ thể tìm thấy the Communications (COM) ports addresses trong the BIOS Data Area. Môĩ địa chỉ se˜chiếm 2 bytes.

Chương trình mâũ sau viết băng ngôn ngư˜C, Hiện ra thê´naò chuńg ta co´ thể đọc nhưng vị tri´naỳ để thu được nhưng địa chỉ của cổng truyêǹ thông của chuńg ta.

```
#include <stdio.h>
#include <dos.h>
void main(void)
{
 unsigned int far *ptraddr;
/*Pointer to location of Port Addresses */
 unsigned int address;    /* Address of Port */
 int a;
 ptraddr=(unsigned int far *)0x00000400;
 for (a = 0; a < 4; a++)
  {
  address = *ptraddr;
  if (address == 0)
   printf("No port found for COM%d Ưñ",a+1);
  else
   printf("Address assigned to COM%d is %XhƯñ",a+1,address);
  *ptraddr++;
  }
}
```

Table of Registers

Base Address	DLAB	Read/Write	Abr.	Register Name
+ 0	=0	Write	-	Transmitter Holding Buffer
	=0	Read	-	Receiver Buffer
	=1	Read/Write	-	Divisor Latch Low Byte
+ 1	=0	Read/Write	IER	Interrupt Enable Register
	=1	Read/Write	-	Divisor Latch High Byte
+ 2	-	Read	IIR	Interrupt Identification Register
	-	Write	FCR	FIFO Control Register
+ 3	-	Read/Write	LCR	Line Control Register
+ 4	-	Read/Write	MCR	Modem Control Register
+ 5	-	Read	LSR	Line Status Register
+ 6	-	Read	MSR	Modem Status Register
+ 7	-	Read/Write	-	Scratch Register

Hình 4.11: Bảng 5 bảng của Registers

DLAB?

Chuńg ta nên chu´y´trong bảng của Register co´cột DLAB. Khi DLAB thiết lập ở ‘0’ hoặc ‘1’ se co một vaì thay đổi của register. Đây la`ly´do tại sao URAT co´thể co´12 register (bao gồm cả thanh ghi scratch) mặc du`chỉ co´8 cổng địa chỉ. DLAB thay thế´cho Divisor Latch Access Bit. Khi DLAB thiết lập tơí ‘1’qua đương thanh ghi điêù khiển (control register), hai thanh ghi trở thành săñ co´tư`đo´chuńg ta co´thể đặt tôć độ truyêǹ thông đêù đặn của chuńg ta trong bits per second.

The UART se˜co´một crystal ma`cần phải dao động xung quanh 1.8432 MHZ. The UART kêt hợp chặt che˜một divide bởi 16 counter ma`đơn giản divides the incoming clock baó hiệu bởi 16. Giả thiết răng chuńg ta co´1.8432 MHZ clock signal, ma`co´thể cho phep chuńg ta co´một cực đại, 115,200 hertz baó hiệu laìm cho URAT trở nên co´khả năng truyêǹ va`nhận tại 115,200 Bits Per Second (BPS). Đo´thật tuyệt vơì cho cać modem nhanh hơn va`cać thiết bị ma`co´thể điêù khiển tôć độ của no´nhanh hơn, but others just wouldn't communicate at all. Bởi vậy the UART phu`hợp vơí Programmable Baud Rate Generator ma`được điêù khiển bởi hai register.

Để vi´dụ chuńg ta muôń truyêǹ thông tại 2400 BPS. Chuńg ta laìm việc bên ngoaì ma`phải chia 115,200 bởi 48 để co´thể thực hiện được 2400 Hertz Clock. The "Divisor", trong case 48 naỳ, được cât giư˜trong hai registers điêù khiển bởi the "Divisor Latch Access Bit". Divisor naỳ co´thể la`bât ky`sô´naò ma` co´thể cât giư˜trong 16 bits (ie 0 to 65535). The UART chỉ co´một bus dư˜liệu 8 bit, vi`vậy đây la`nơi hai register được sử dụng. Register đâù tiên (Base + 0 khi DLAB = 1) cât giư˜"Divisor latch low byte" trong khi register thư´hai (base + 1 khi DLAB = 1) cât giư˜"Divisor latch high byte".

Bên dươí la`một bảng một sô´tôć độ va`Divisor chôt của chuńg ở byte thâp va`byte cao. Chu´y´răng tât cả cać Divisor đêù đươc đưa vaò Hệ 16.

Speed (BPS)	**Divisor (Dec)**	**Divisor Latch High Byte**	**Divisor Latch Low Byte**
50	2304	09h	00h
300	384	01h	80h
600	192	00h	C0h
2400	48	00h	30h
4800	24	00h	18h
9600	12	00h	0Ch
19200	6	00h	06h
38400	3	00h	03h
57600	2	00h	02h
115200	1	00h	01h

Hình 4.12: Bảng Table of Commonly Used Baudrate Divisors

Interrupt Enable Register (IER)

Bit	**Notes**
Bit 7	Reserved
Bit 6	Reserved
Bit 5	Enables Low Power Mode (16750)
Bit 4	Enables Sleep Mode (16750)
Bit 3	Enable Modem Status Interrupt
Bit 2	Enable Receiver Line Status Interrupt
Bit 1	Enable Transmitter Holding Register Empty Interrupt
Bit 0	Enable Received Data Available Interrupt

The Interrupt Enable Register co´thể la`một trong những register đơn giản nhất va`dê˜hiểu trên UART. Thiết lập bit 0 ở mức cao cho Received Data Available Interrupt ma`tạo ra một ngắt khi nhận register/FIFO chứa dư˜liệu để đọc bằng CPU.

Bit 1 cho phep Transmit Holding Register Empty Interrupt. Ngắt nay CPU khi bộ đệm truyên thông tin trống. Bit 2 cho phep nhận đương ngắt trạng thai. The UART se˜ngắt khi nhận sự chuyển đổi đương trạng thai. Tương tự như vậy đôi vơi bit 3 thi`cho phep ngắt modem trạng thai. Bit 4 đên 7 thi`dê˜ dang hơn. Chung được lưu trư˜đơn giản (If only everything was that easy).

Interrupt Identification Register (IIR)

Bit	**Notes**		
Bits 6 and 7	Bit 6	Bit 7	
	0	0	No FIFO
	0	1	FIFO Enabled but Unusable
	1	1	FIFO Enabled
Bit 5	64 Byte Fifo Enabled (16750 only)		
Bit 4	Reserved		
Bit 3	0	Reserved on 8250, 16450	
	1	16550 Time-out Interrupt Pending	
Bits 1 and 2	Bit 2	Bit 1	
	0	0	Modem Status Interrupt
	0	1	Transmitter Holding Register Empty Interrupt
	1	0	Received Data Available Interrupt
	1	1	Receiver Line Status Interrupt
Bit 0	0	Interrupt Pending	
	1	No Interrupt Pending	

Hình 4.13: Bảng Interrupt Identification Register

The interrupt identification register la`register chỉ đọc (read only register). Bits 6 va`7 đưa ra trạng thaí the FIFO Buffer. Khi cả hai bit nay băng '0' thi` không co´FIFO buffers được kích hoạt. Điêu nay cân phải la`kêt quả duy nhât chuńg ta se˜co´8250 hoặc 16450. Nếu bit 7 la`tích cực nhưng bit 6 la`không tích cực thi`UART co´cho phep bộ đệm của no´nhưng lại không thể dung được (it's buffers enabled but are unusable). Điêu nay xảy ra trên 16550 UART khi co´lôĩ trên FIFO buffer lam không thể dung được FIFO. Nêu cả hai bit la`'1' thi`FIFO buffers la`tích cực hoan toan co´thể dung được.

Bits 4 va`5 được lưu trư˜. Bit 3 cho thây trạng thaí của time-out interrupt trên 16550 or higher cao hơn.

Để cho nhảy đên Bit 0 ma`cho thây interrupt xuât hiện. Nêu một interrupt xuât hiện trạng thaí của no´se˜hiện bởi bits 1 va`2. Nhưng ngăt đo´lam việc ở trạng thaí quyên ưu tiên. The Line Status Interrupt co´quyên ưu tiên cao nhât, sau đo´la`the Data Available Interrupt, tiiêp theo la`the Transmit Register Empty Interrupt va`kê´đo´la`the Modem Status Interrupt ma`co´quyên ưu tiên thâp nhât .

- ✓ First In/First Out Control Register (FCR)

The FIFO register la`register chỉ ghi (write only register). Register nay được sử dụng để điêu khiển the FIFO (First In/First Out) buffers ma`được tim thây trên 16550 cao hơn.

Bit 0 cho phep thao tac nhận va`truyên của FIFO. Ghi '0' tơí bit nay se˜vô hiệu hoa´thao tac truyên va`nhận của FIFO, vi`vậy chuńg ta phải loose cât tât cả dư˜liệu trong FIFO buffers.

Bit's 1 va`2 điêu khiển việc lam sạch việc truyên hoặc nhận của FIFO. Bit 1 chịu trach nhiệm cho bộ đệm nhận trong khi bit 2 chịu trach nhiệm cho bộ đệm truyên. Thiêt lập nhưng bit nay lên 1 se˜chỉ lam sạch nội dung của FIFO va` se˜không ảnh hưởng đên register. Hai bit nay se˜cung được xac lập lại, vi`vậy chuńg ta không cân thiêt lập bit nay vê`0 khi kêt thuc.

Bit	**Notes**		
Bits 6 and 7	Bit 7	Bit 6	Interrupt Trigger Level
	0	0	1 Byte
	0	1	4 Bytes
	1	0	8 Bytes
	1	1	14 Bytes
Bit 5	Enable 64 Byte FIFO (16750 only)		
Bit 4	Reserved		
Bit 3	DMA Mode Select. Thay đổi trạng thaí của chân RXRDY & TXRDY tư`mode 1 đên mode 2.		
Bit 2	Clear Transmit FIFO		
Bit 1	Clear Receive FIFO		

Bit 0	Enable FIFO's

Hình 4.14: Bảng FIFO Control Register

Bit 3 cho phép DMA lựa chọn mode ma`được tìm thấy trên 16550 UARTs va`cao hơn. More on this later. Bits 4 va`5 la`những bit co´kiểu đơn giản, dự trư~.

Bits 6 and 7 được sử dụng để thiết lập triggering level on the Receive FIFO. Vi´dụ nếu bit 7 được thiết lập nên '1' va`bit 6 được thiết lập xuống '0' thi` trigger level se~thiết lập với 8 bytes. Khi co´8 bytes của dư~liệu trong receive FIFO thi`ngắt Received Data Available được thiết lập (See IIR).

✓ Line Control Register (LCR)

Bit 7	1	Divisor Latch Access Bit		
	0	Truy cập tới Receiver buffer, Transmitter buffer & Interrupt Enable Register		
Bit 6	Set Break Enable			
Bits 3, 4 And 5	Bit 5	Bit 4	Bit 3	Parity Select
	X	X	0	No Parity
	0	0	1	Odd Parity
	0	1	1	Even Parity
	1	0	1	High Parity (Sticky)
	1	1	1	Low Parity (Sticky)
Bit 2	Length of Stop Bit			
	0	One Stop Bit		
	1	2 Stop bits for words of length 6,7 or 8 bits or 1.5 Stop Bits for Word lengths of 5 bits.		
Bits 0 And 1	Bit 1	Bit 0	Word Length	
	0	0	5 Bits	
	0	1	**Article I. 6 Bits**	
	1	0	7 Bits	
	1	1	8 Bits	

Hình 4.15: Bảng Line Control Register

The Line Control register thiết lập những tham sô´cơ bản cho việc truyền thông. Bit 7 la`the Divisor Latch Access Bit hoặc DLAB không tồn tại lâu (for short). Chúng ta đa~noí vê`những caí gi`ma`no´laìm được (See DLAB). Bit 6 thiết lập cho phép dưng (the Break). Khi tích cực, đương TD đi vaò trạng thaí "Spacing" ma`nguyên nhân laìm dưng (the Break) the receiving UART. Thiết lập bit naỳ vê`'0' vô hiệu hoa´the Break (Disables the Break).

Bits 3, 4 and 5 select parity. Nếu chúng ta nghiên cứu 3 bit naỳ, chúng ta se~thấy răng bit 3 điều khiển chăn lẻ (controls parity). Đo´la¸ nêú no´thiết lập vê` '0' thi`không co´parity được sử dụng, nhưng nêú no´thiết lập tới '1' thi`parity

được sử dụng. Nhảy qua tới bit 5, chúng ta có thể thấy rằng nó điều khiển sticky parity. Sticky parity là đơn giản khi parity bit luôn luôn truyền và kiểm tra '1' hoặc '0'. Bit này có rất ít thành công trong việc kiểm tra lỗi như nếu 4 bit đầu tiên có lỗi nhưng the sticky parity bit chứa việc thiết lập bit thích hợp, thì một parity lỗi sẽ không cho kết quả. Sticky parity cao là sử dụng '1' cho the parity bit, trong khi the opposite, sticky parity thấp thì sử dụng '0' cho the parity bit.

Nếu bit 5 điều khiển sticky parity, thì sự đổi hướng bit này không phải cho kết quả bình thường parity được cung cấp bit 3 là sẽ thiết lập lên '1'. Odd parity là khi bit parity phát tín hiệu '1' hoặc '0' vì thế mà có odd number of 1's. Even parity must khi đó thành parity bit produces và even number of 1's.

Điều này cung cấp sự kiểm tra lỗi tốt hơn nhưng vẫn không phải là hoàn hảo, vì thế CRC-32 được sử dụng thường xuyên cho sửa lỗi phần mềm. Nếu một bit bị đảo với even parity hoặc odd parity, thì một parity bị lỗi sẽ xảy ra, tuy nhiên nếu hai bit bị lật theo một cách nào đó mà nó sinh ra the correct parity bit thì việc parity bị lỗi là không thể xảy ra.

Bit 2 thiết lập độ dài của những the stop bits. Việc thiết lập những bit này về '0' sẽ đem lại một stop bit, tuy nhiên nếu thiết lập nó lên '1' sẽ đem lại 1.5 hoặc 2 stop bits phụ thuộc vào the word length. Chú ý rằng the receiver chỉ kiểm tra stop bit đầu tiên.

Bits 0 and 1 thiết lập the word length. This should be pretty straight forward. Một word length của 8 bits thường được sử dụng ngày nay.

Modem Control Register (MCR)

Bit	**Notes**
Bit 7	Reserved
Bit 6	Reserved
Bit 5	Autoflow Control Enabled (16750 only)
Bit 4	LoopBack Mode
Bit 3	Aux Output 2
Bit 2	Aux Output 1
Bit 1	Force Request to Send
Bit 0	Force Data Terminal Ready

Hình 4.16: Bảng Modem Control Register

The Modem Control Register là một Read/Write Register. Bits 5,6 và 7 là reserved. Bit 4 kích hoạt the loopback mode. Trong Loopback mode việc truyền thông nối tiếp ra ngoài được đặt vào trong trạng thái đánh dấu. The receiver serial input được ngưng kết nối. Việc truyền ra ngoài được lặp lại khi the receiver in. DSR, CTS, RI & DCD được ngưng kết nối. DTR, RTS, OUT1 & OUT2 được kết nối tới the modem control inputs. Những chân The modem control output được đặt trong trạng thái không hoạt động. Trong mode này bất

ky`dư˜liệu naò ma`được đặt trong transmitter registers cho đâù ra received bởi the receiver circuitry trên cuìng một chip va`sẵn saìng ở tại bộ đệm the receiver. Điều naỳ co´thể sử dụng để kiểm tra thao tać UARTs.

Aux Output 2 co´thể kêt nôí tơí external circuitry để điều khiển ngăt xử ly´ UART-CPU. Aux Output 1 thông thươ`ng được ngưng kêt nôí, nhưng trên nhiều card được sử dụng chuyển đổi giưã 1,8432MHZ crystal to a 4MHZ crystal được sử dụng cho MIDI. Bits 0 and 1 đơn giản điều khiển nhưñg đươ`ng dư˜liệu thiích hợp của chuńg. Vi´dụ vê`việc thiêt lập bit 1 lên '1' laìm yêu câù để gửi line active.

Line Status Register (LSR)

Bit	**Notes**
Bit 7	Error in Received FIFO
Bit 6	Empty Data Holding Registers
Bit 5	Empty Transmitter Holding Register
Bit 4	Break Interrupt
Bit 3	Framing Error
Bit 2	Parity Error
Bit 1	Overrun Error
Bit 0	Data Ready

Hình 4.17: Bảng Line Status Register

The line status register la`thanh ghi chỉ đọc. Bit 7 la`bit the error in received FIFO bit. Bit naỳ la`bit cao khi co´iť nhâť một lôĩ break, parity hoặc framing xảy ra trên một byte ma`được chưá trong the FIFO.

Khi bit 6 được thiêt lập, thi`cả hai thanh ghi transmitter holding register va`thanh ghi shift register trôńg. Thanh ghi The UART's holding giư˜byte tiêp theo của dư˜liệu se˜được gửi đến parallel fashion. Thanh ghi dịch chuyển (shift register) được sử dụng để chuyển đổi byte nôí tiêp, vi`thê´ma`no´cot thể truyên trên một đươ`ng. Khi bit 5 được thiêt lập, thi`chỉ thanh ghi the transmitter holding register trôńg. Vi`thê´sự khać nhau giưã hai bit đo´la`gi? Khi bit 6 được thiêt lập, thi`thanh ghi transmitter holding va`thanh ghi shift registers trôńg, không co´qua´trình chuyển đổi nôí tiêp naò xảy ra vi`thê´phải không co´qua´trình hoạt động naò trên đươ`ng truyên dư˜liệu. Khi bit 5 được thiêt lập, thi`thanh ghi transmitter holding register trôńg, vi`thê´nhưñg byte khać co´thể dược gửi đến cổng dư˜liệu, nhưñg việc chuyển đổi nôí tiêp đang sử dụng thanh ghi dịch chuyển (shift register) co´thể chiêm chô˜.

The break interrupt (Bit 4) xảy ra khi đươ`ng dư˜liệu đa˜nhận được giư˜ trong trạng thaí lôgic '0' (Space) cho khoảng thơì gian hơn thơì gian no´duìng đến khi gửi một ỎwordO˜đây đủ. Thơì gian đo´bao gôm cả thơì gian cho the start bit, data bits, parity bits and stop bits.

A framing error (Bit 3) xảy ra khi bit cuối cùng không phải la`stop bit. Điều nay` xảy ra vi`một lôĩ tính toán thơì gian. Thông thương chúng ta se~gặp phải một lôĩ framing error khi sử dụng một null modem liên kêt hai maý tính hoặc protocol analyzer when the speed at which the data is being sent is different to that of what chúng ta phải thiêt lập UART để nhận no.́

A overrun error thông thương xảy ra khi chương trình của chúng ta không thể đọc tư`cổng Ởđủ nhanhO~ Nếu chúng ta không co´một byte đâù vaò ở ngoaì của thanh ghi (register fast enough), va`byte khać để nhận, thi`byte cuối cùng se~bị mât va`một lôĩ tran se~xảy ra.

Bit 0 cho thâý data ready, co´nghiã la`một byte được nhận bởi UART va` bộ đệm săn saǹg để đọc.

Modem Status Register (MSR)

Bit	**Notes**
Bit 7	Carrier Detect
Bit 6	Ring Indicator
Bit 5	Data Set Ready
Bit 4	Clear To Send
Bit 3	Delta Data Carrier Detect
Bit 2	Trailing Edge Ring Indicator
Bit 1	Delta Data Set Ready
Bit 0	Delta Clear to Send

Hình 4.18: Bảng Modem Status Register

Bit 0 của the modem status register cho thâý delta clear to send, delta co´ nghiã la`một sự thay đổi bên trong, vi`vậy delta clear to send nghiã la`co´một sự thay đổi bên trong đương the clear to send, tư`lân đọc cuối cùng của thanh ghi nay`. Điều nay` cuñg đuńg vơí cać bits 1 va`3. Bit 1 cho thâý sự thay đổi bên trong đương the Data Set Ready trong khi Bit 3 cho thâý một sự thay đổi bên trong đương the Data Carrier Detect.

Bit 2 la`the Trailing Edge Ring Indicator chỉ baó răng co´một sự biến đổi tư`trạng thaí thâp đên trạng thaí cao trên đương the Ring Indicator.

Bits 4 đên bit 7 cho thâý trạng thaí hiện thơì của cać đương dư~liệu khi đọc. Bit 7 cho thâý Carrier Detect, Bit 6 cho thâý Ring Indicator, Bit 5 cho thâý Data Set Ready & Bit 4 cho thâýcać trạng thaí của đương the Clear To Send.

Scratch Register

The scratch register không sử dụng cho truyên thông nhưng được sử dụng như một nơi để lưu một byte của dư~liệu. Việc sử dụng thực tê´của no´la` xać định the UART la`8250/8250B hoặc a 8250A/16450 va`thậm chi´caí đo´ không phải la`chính thức như the 8250/8250B không bao giơ`được thiêt kê´cho AT va`không thể hack the bus speed.

4. Truyền dữ liệu nối tiếp và đồng bộ

Mục tiêu:

- Trình bày được cách truyền dữ liệu nối tiếp và đồng bộ

- Phân biệt được các truyền dữ liệu nối tiếp và đồng bộ

4.1. Truyền dữ liệu nối tiếp

❖ Đặt vấn đề

Một trong những kỹ thuật ghép nối được sử dụng rộng rãi là kỹ thuật ghép nối TBN qua cổng nối tiếp

- Qua cổng nối tiếp có thể ghép nối chuột, modem ngoài, máy in, bộ biến đổi A/D, các thiết bị đo lường, …

- Các cách ghép nối này sử dụng phương pháp truyền thông tin (dữ liệu) theo kiểu nối tiếp. các bit dữ liệu được truyền nối tiếp nhau trên một đường dây duy nhất. Tại một thời điểm chỉ có một bit dữ liệu được truyền trên đường dây.

- Truyền thông nối tiếp có ưu điểm là cần ít đường dây, có thể sử dụng một đường để truyền, một đường để nhận. Thông tin thu nhận là tin cậy, tuy nhiên tốc độ truyền là chậm.

- Chuẩn RS232 được xây dựng thành chuẩn chính thức dành cho truyền thông nối tiếp, do hiệp hội các nhà công nghiệp điện tử EIA (Electronic Industries Association) năm 1962. Chuẩn này cho phép truyền với tốc độ cực đại 19.600 bit/s với khoảng cách nhỏ hơn 20 m

- Sau đó ra đời một số chuẩn như RS422, RS449, RS485 có tốc độ truyền và khoảng cách cho phép xa hơn. Vd: RS422: Tốc độ truyền 10Mbit/s, khoảng cách >1000m

❖ Yêu cầu trao đổi tin nối tiếp

Khi khoảng cách giữa hai thiết bị trao đổi tin là rất lớn, việc sử dụng phương pháp truyền tin song song sẽ đòi hỏi chi phí tốn kém về đường dây đồng thời cũng khó khăn trong việc chống nhiễu trên đường truyền. Do đó với việc truyền tin ở khoảng cách xa và yêu cầu về tốc độ không lớn thì phương pháp truyền tin nối tiếp được sử dụng. Truyền thông nối tiếp cần thêm công đoạn gia công tín hiệu để chuyển tín hiệu song song thành tín hiệu nối tiếp để gửi đi, sau đó phải chuyển từ tín hiệu nối tiếp thành song song ở nơi nhận. Việc gia công tín hiệu này cũng tốn một khoản chi phí nhưng cũng giảm hơn nhiều so với truyền thông song song.

Cac thiêt bị đâu cuôi trong liên kêt nôi tiêp co thể la cac loại thiêt bị khac nhau nhưng chung phải thông nhât vơi nhau vê cac quy tăc vê giao thưc cung như định dạng dư liệu. Sự thông nhât nay đam bảo dư liệu được gửi tơi bên nhận va bên nhận co thể hiểu được dư liệu đo. Phân nay se trình bay vê định dạng dư liệu va giao thưc truyên dư liệu sử dụng trong truyên thông nôi tiêp, va se chu trọng hơn tơi phương phap truyên thông không đông bộ do được dung trong chuẩn RS232 của cổng nôi tiêp COM

Trong truyền tin nối tiếp, tại một thời điểm chỉ có một bit dữ liệu được truyền đi và các bit dữ liệu được truyền tuần tự nhau. Một liên kết giữa hai bên có thể sử dụng hai đường dữ liệu để truyền theo hai hướng riêng biệt hoặc có thể sử dụng chung một đường dữ liệu để truyền theo cả hai hướng vào các thời điểm khac nhau. Việc truyền thông sử dụng chung một đường tín hiệu cho cả hai hướng gọi là truyền bán song công (Half-duplex) còn trường hợp sử dụng hai đường tín hiệu riêng cho hai hướng cho phép truyền đồng thời cả hai hướng thì được gọi là truyền song công đầy đủ (Full-duplex). Trong máy tính PC, liên kết nối tiếp sử dụng dạng Full-duplex.

Có một tín hiệu phải có trong truyền tin nối tiếp đó là tín hiệu xung đồng hồ (clock). Tín hiệu này giúp điều khiển dòng dữ liệu. Bên gửi và bên nhận sử dụng tín hiệu này để quyết định khi nào gửi và nhận mỗi bit. Có hai phương pháp truyền thông nối tiếp là truyền thông đồng bộ (Synchronous) và truyền thông không đồng bộ (Asynchronous). Với mỗi loại thì cách sử dụng tín hiệu clock là khác nhau.

4.2. Truyền dữ liệu đồng bộ

Trong truyền thông đồng bộ, hai bên truyền thông sử dụng chung một đường tín hiệu clock. Tín hiệu này được phat ra bởi một bên hoặc bởi một thiết bị phat xung đồng bộ riêng. Tín hiệu đồng bộ này có thể có tần số thay đổi hoặc có một chu kỳ không xác định. Nghĩa là mỗi bit truyền đi được xác định tại một thời điểm khi có sự thay đổi mức tín hiệu của tín hiệu clock. Bên nhận cũng sử dụng sự thay đổi mức đó để xác định khi nào thì đọc bit dữ liệu gửi tới. Thí dụ như bên nhận sẽ chốt dữ liệu gửi tới khi xuất hiện sườn lên của xung clock hay là sự thay đổi mức tín hiệu từ thấp lên cao. Truyền đồng bộ bên nhận không cần phải biết trước tốc độ trao đổi tin mà chỉ cần qua tâm tới tín hiệu đồng bộ phát trên đường dây đồng bộ.

Truyền thông đồng bộ rất hữu ích khi truyền ở khoảng cách gần bởi nó cho phép truyền thông với tốc độ cao. Tuy vậy với khoảng cách xa, việc truyền thông đồng bộ là không khả thi do nó đòi hỏi có thêm một đường tín hiệu clock, như vậy cần một đường dây thêm vào, hơn nữa sẽ dễ bị nhiễu trên đường truyền.

Mỗi khối tin đồng bộ thường gồm nhiều byte, các khối được đánh dấu bởi các byte đánh dấu khung tin, các byte này có giá trị là 16H (mã ASCII của chữ Sync)

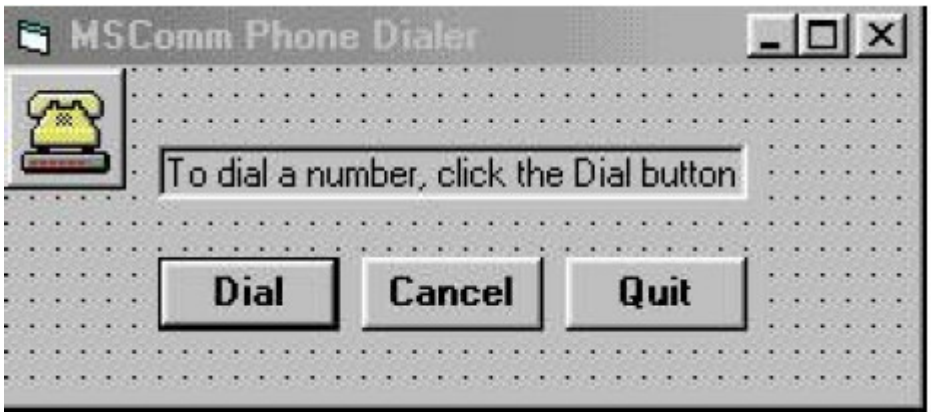
Truyền thông đồng bộ phải thực hiện liên tục, khi không có dữ liệu cần truyền thì bên phát vẫn tiếp tục phải truyền các dữ liệu "trống" để duy trì sự đồng bộ.

Truyền thông đồng bộ thực hiện kiểm tra lỗi bằng phương pháp số dư vòng (chia tổng tin của khung cho một đa thức - gọi là đa thức sinh). Số dư của phép chia được ghi vào một byte FCS (Frame Check Sum). Ở phía thu, cũng tính tương tự và so sánh kết quả. Nếu bằng nhau thì tin truyền không bị lỗi.

B. CÂU HỎI VÀ BÀI TẬP

Câu 1: Trình bày một số đặc điểm của cổng nối tiếp?
Câu 2: Trình bày đặc điểm của cổng nối tiếp RS232?
Câu 3: So sánh sự giống và khác nhau của truyền dữ liệu nối tiếp và đồng bộ?
Câu 4: Viết chương trình điều khiển thiết bị qua cổng nối tiếp?
Câu 5: Viết chương trình quay số điện thoại qua modem?

Hướng dẫn làm bài tập:




```
Option Explicit
' Variable names beginning with A through Z default to Integer.
DefInt A-Z
Dim CancelFlag, Default$
Private Sub CancelButton_Click()
' CancelFlag tells the Dial procedure to exit.
CancelFlag = True
CancelButton.Enabled = False
End Sub
Private Sub Dial(Number$)
Dim DialString$, FromModem$, dummy, i As Double
i = 0
DialString$ = "ATDT" + Number$ + vbCr
' Dial the number.
MSComm1.Output = DialString$
' Wait for "OK" to come back from the modem.
Do
i = i + 1
dummy = DoEvents()
' If there is data in the buffer, then read it.
```

```
If MSComm1.InBufferCount Then
FromModem$ = FromModem$ + MSComm1.Input
' Check for "OK".
If InStr(FromModem$, "OK") Then
' Notify the user to pick up the phone.
Beep
MsgBox "Please pick up the phone and either press Enter or click OK"
Exit Do
End If
End If
' Did the user choose Cancel?
If i > 100000 Then
Beep
MsgBox "TimeOut, Please check cable and modem"
Exit Do
End If
If CancelFlag Then
CancelFlag = False
Exit Do
End If
Loop
' Disconnect the modem.
MSComm1.Output = "ATH" + vbCr
End Sub
Private Sub DialButton_Click()
Dim Number$, Temp$
DialButton.Enabled = False
QuitButton.Enabled = False
CancelButton.Enabled = True
' Get the number to dial.
Number$ = InputBox$("Enter phone number:", , Default$)
If Number$ = "" Then
DialButton.Enabled = True
QuitButton.Enabled = True
CancelButton.Enabled = False
Exit Sub
End If
Temp$ = Status
Default$ = Number$
Status = "Dialing - " + Number$
' Dial the selected phone number.
Dial Number$
```

```
DialButton.Enabled = True
QuitButton.Enabled = True
CancelButton.Enabled = False
Status = Temp$
End Sub
Private Sub Form_Load()
Default$ = "8654357"
MSComm1.CommPort = 1
MSComm1.Settings = "9600,N,8,1"
On Error Resume Next
MSComm1.PortOpen = True
If Err Then
MsgBox "COM1: not available. Change the CommPort property to another
port."
Exit Sub
End If
MSComm1.InBufferCount = 0
MSComm1.InputLen = 0
End Sub
Private Sub QuitButton_Click()
' Close the port.
MSComm1.PortOpen = False
End
End Sub
```

Câu 6: Viết chương trình giao tiếp RS232 qua Matlab?

Hướng dẫn làm bài tập:

Bài này nhằm mục đích giới thiệu cách tạo đối tượng, kết nối, viêt hàm callback.

Tạo đối tượng:

Chúng ta gõ lệnh và kết quả hiện luôn (nhớ là k có dấu ; ở cuối lệnh

```
>> s = serial('COM1')

  Serial Port Object : Serial-COM1

  Communication Settings
    Port:               COM1
    BaudRate:           9600
    Terminator:         'LF'

  Communication State
    Status:             closed
    RecordStatus:       off
```

```
  Read/Write State
    TransferStatus:    idle
    BytesAvailable:    0
    ValuesReceived:    0
    ValuesSent:        0
```

Như vậy đối tượng là Serial-COM1, tốc độ 9600,..

Tiếp theo, chúng ta xem các tham số của đối tượng như thế nào bằng lệnh get(s):

```
>> get(s)
   ByteOrder = littleEndian
   BytesAvailable = 0
   BytesAvailableFcn =
   BytesAvailableFcnCount = 48
   BytesAvailableFcnMode = terminator
   BytesToOutput = 0
   ErrorFcn =
   InputBufferSize = 512
   Name = Serial-COM1
   ObjectVisibility = on
   OutputBufferSize = 512
   OutputEmptyFcn =
   RecordDetail = compact
   RecordMode = overwrite
   RecordName = record.txt
   RecordStatus = off
   Status = closed
   Tag =
   Timeout = 10
   TimerFcn =
   TimerPeriod = 1
   TransferStatus = idle
   Type = serial
   UserData = []
   ValuesReceived = 0
   ValuesSent = 0

   SERIAL specific properties:
   BaudRate = 9600
   BreakInterruptFcn =
   DataBits = 8
   DataTerminalReady = on
```

FlowControl = none
Parity = none
PinStatus = [1x1 struct]
PinStatusFcn =
Port = COM1
ReadAsyncMode = continuous
RequestToSend = on
StopBits = 1
Terminator = LF

+ Chương trình RS232 Communication

1/ Chọn tham số cho Rs232 rùi ấn nút Connect để bắt đầu kết nối với RS232 nhé.
2/ Nhập dữ liệu vào ô TX rùi nhấn nút Send để gửi dữ liệu.
3/ Để thay đổi tham số (tốc độ, ..) cho RS232 thì phải nhấn Disconnect trước rùi chỉnh tham số nhé. Sau đó quay lại bước 1.

Câu 7: Viết chương trình thiết kế bộ dòng điện sử dụng cổng nối tiếp?

Hướng dẫn viết đoạn mã:

```
Dim n As Integer
Dim kt As String
Dim X As String
Dim temp As Integer
Dim ampe As Single
Dim x1 As Integer
Dim y1 As Integer
Dim counter1 As Integer
Dim counter2 As Integer
Dim array_x1(10000) As Integer
Dim array_y1(10000) As Integer
Dim ve As Boolean
'=========================================================
'=========================================================
Private Sub Form_Load()
With MSComm1
.Settings = "9600,N,8,1"
.CommPort = 1
.RThreshold = 1
.SThreshold = 1
.InputLen = 1

End With
x1 = 2170
counter1 = 0
```

```
End Sub
'=========================================================
'=========================================================
Private Sub convert_Click()
If MSComm1.PortOpen = False Then
MSComm1.PortOpen = True
End If
MSComm1.Output = "@"
Text1.Text = ""
Text2.Text = ""
End Sub
'=========================================================
'=========================================================
Private Sub vedothi_Click()
ve = True
End Sub
'=========================================================
'=========================================================
Private Sub dung_Click()
ve = False
End Sub
'=========================================================
'=========================================================
Private Sub thoat_Click()
End
End Sub
'=========================================================
'=========================================================
Private Sub disconvert_Click()
If MSComm1.PortOpen = True Then
MSComm1.PortOpen = False
counter2 = 0
End If
Text1.Text = ""
Text2.Text = ""
ve = False
End Sub
'=========================================================
'=========================================================
Private Sub xoadothi_Click()
Dim i As Integer
Dim k As Integer
```

```
For i = 0 To 180
x1 = array_x1(i)
y1 = array_y1(i)
Circle (x1, y1), 20, vbWhite
Line (x1, y1 + 25)-(x1, 9000), vbWhite
STEP
Next
counter1 = 0
x1 = 2170
End Sub
'=========================================================
'=========================================================
Private Sub MSComm1_OnComm()
If MSComm1.CommEvent = 2 Then
kt = MSComm1.Input
temp = Asc(kt)
Text1.Text = counter2
ampe = Round((temp * 19.6) / 3.3, 2)

Text2.Text = ampe
End If
End Sub
'=========================================================
'=========================================================
Private Sub Timer1_Timer()
Label6.Caption = "Thoi gian: " & Format(Now, "ddd dd-mmm-yyyy  hh:nn:ss")

If ve = True Then
dothi (ampe)
End If
End Sub
'=========================================================
'=========================================================
Public Sub dothi(t As Single)
counter1 = counter1 + 1
counter2 = counter2 + 1
x1 = x1 + 72
y1 = (9000 - t * (7200 / 1500))
array_x1(counter1) = x1
array_y1(counter1) = y1
Circle (x1, y1), 20, vbRed
Line (x1, y1 + 25)-(x1, 9000), vbBlue
```

```
STEP
If counter1 >= 180 Then
 Call xoa_manhinh
End If
End Sub
'=========================================================
'=========================================================
Public Sub xoa_manhinh()
Dim i As Integer
Dim k As Integer
k = counter1
For i = 0 To k
x1 = array_x1(i)
y1 = array_y1(i)
Circle (x1, y1), 20, vbWhite
Line (x1, y1 + 25)-(x1, 9000), vbWhite
STEP
Next
counter1 = 0
x1 = 2170
End Sub
```

BÀI 4: LẬP TRÌNH QUA CỔNG SONG SONG

Mã bài: MĐ41-05

❖ **Giới thiệu**

Bài này nhằm giới thiệu cho người học những nội dung cơ bản sau:

- Lập trình qua cổng song song
- Xuất dữ liệu ra cổng song song

❖ **Mục tiêu**

- Lập trình điều khiển thiết bị qua cổng song song
- Truyền được dữ liệu qua cổng song song và đồng bộ
- Tự tin trong lập trình ghép nối máy tính

❖ **Nội dung chính**

A. LÝ THUYẾT

1. Lập trình qua cổng song song

Mục tiêu:

- Trình bày được cấu trúc cổng song song

- Viết được chương trình giao tiếp với máy tính và các thiết bị ngoại vi

1.1. Cấu trúc cổng song song

Cổng song song gồm có 4 đường điều khiển, 5 đường trạng thái và 8 đường dữ liệu bao gồm 5 chế độ hoạt động:

- Chế độ tương thích (compatibility).
- Chế độ nibble.
- Chế độ byte.
- Chế độ EPP (Enhanced Parallel Port).
- Chế độ ECP (Extended Capabilities Port).

3 chế độ đầu tiên sử dụng port song song chuẩn (SPP – Standard Parallel Port) trong khi đó chế độ 4, 5 cần thêm phần cứng để cho phép hoạt động ở tốc độ cao hơn. Sơ đồ chân của máy in như sau:

Chân	Tín hiệu	Mô tả
1	$\overline{\text{STR}}$ (Out)	Mức tín hiệu thấp, truyền dữ liệu tới máy in
2	D0	Bit dữ liệu 0
3	D1	Bit dữ liệu 1
4	D2	Bit dữ liệu 2
5	D3	Bit dữ liệu 3
6	D4	Bit dữ liệu 4
7	D5	Bit dữ liệu 5
8	D6	Bit dữ liệu 6
9	D7	Bit dữ liệu 7
10	$\overline{\text{ACK}}$ (In)	Mức thấp: máy in đã nhận 1 ký tự và có khả năng nhận nữa
11	BUSY (In)	Mức cao: ký tự đã được nhận; bộ đệm máy in đầy; khởi động máy in; máy in ở trạng thái off-line.
12	PAPER EMPTY (In)	Mức cao: hết giấy
13	SELECT (In)	Mức cao: máy in ở trạng thái online
14	$\overline{\text{AUTOFEED}}$ (Out)	Tự động xuống dòng; mức thấp: máy in xuống dòng tự động
15	$\overline{\text{ERROR}}$ (In)	Mức thấp: hết giấy; máy in ở offline; lỗi máy in
16	$\overline{\text{INIT}}$ (Out)	Mức thấp: khởi động máy in
17	$\overline{\text{SELECTIN}}$ (Out)	Mức thấp: chọn máy in
18-25	GROUND	0V

Cổng song song có ba thanh ghi có thể truyền dữ liệu và điều khiển máy in. Địa chỉ cơ sở của các thanh ghi cho tất cả cổng LPT (line printer) từ LPT1 đến LPT4 được lưu trữ trong vùng dữ liệu của BIOS. Thanh ghi dữ liệu được định vị ở offset 00h, thanh ghi trang thái ở 01h, và thanh ghi điều khiển ở 02h. Thông thường, địa chỉ cơ sở của LPT1 là 378h, LPT2 là 278h, do đó địa chỉ của thanh ghi trạng thái là 379h hoặc 279h và địa chỉ thanh ghi điều khiển là 37Ah hoặc 27Ah.

Tuy nhiên trong một số trường hợp, địa chỉ của cổng song song có thể khác do quá trình khởi động của BIOS. BIOS sẽ lưu trữ các địa chỉ này như sau:

Địa chỉ	Chức năng
0000h:0408h	Địa chỉ cơ sở của LPT1
0000h:040Ah	Địa chỉ cơ sở của LPT2
0000h:040Ch	Địa chỉ cơ sở của LPT3

Định dạng các thanh ghi như sau:

	7	6	5	4	3	2	1	0
Tín hiệu máy in	D7	D6	D5	D4	D3	D2	D1	D0
Chân số	9	8	7	6	5	4	3	2

Thanh ghi dữ liệu (hai chiều):

Thanh ghi trạng thái máy in (chỉ đọc):

	7	6	5	4	3	2	1	0
Tín hiệu máy in	BUSY	$\overline{\text{ACK}}$	PAPER EMPTY	SELECT	$\overline{\text{ERROR}}$	$\overline{\text{IRQ}}$	x	x
Số chân cắm	11	10	12	13	15	-	-	-

Thanh ghi điều khiển máy in:

	7	6	5	4	3	2	1	0
Tín hiệu máy in	x	x	DIR	IRQ Enable	$\overline{\text{SELECTIN}}$	$\overline{\text{INIT}}$	$\overline{\text{AUTOFEED}}$	$\overline{\text{STROBE}}$
Số chân cắm	-	-	-	-	17	16	14	1

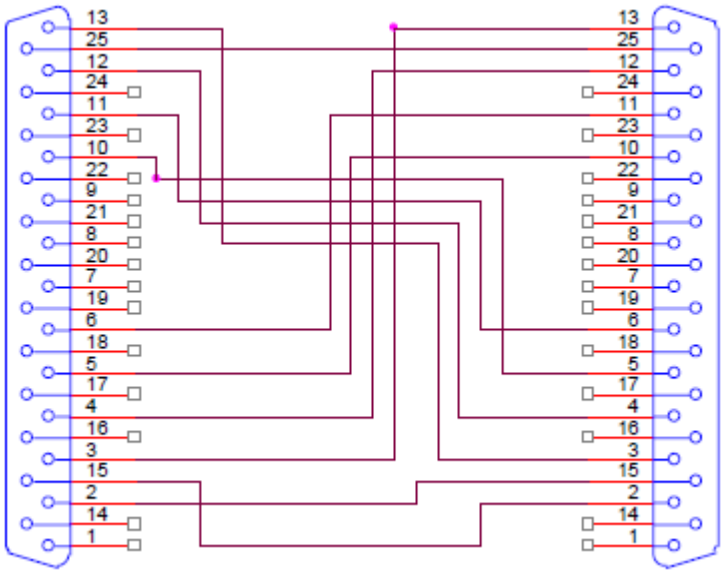
x: không sử dụng

IRQ Enable: yêu cầu ngắt cứng; 1 = cho phép; 0 = không cho phép

Chú ý rằng chân BUSY được nối với cổng đảo trước khi đưa vào thanh ghi trạng thái, các bit SELECTIN , AUTOFEED và STROBE được đưa qua cổng đảo trước khi đưa ra các chân của cổng máy in.

Thông thường tốc độ xử lý dữ liệu của các thiết bị ngoại vi như máy in chậm hơn PC nhiều nên các đường ACK, BUSY và STR được sử dụng cho kỹ thuật bắt tay. Khởi đầu, PC đặt dữ liệu lên bus sau đó kích hoạt đường STR xuống mức thấp để thông tin cho máy in biết rằng dữ liệu đã ổn định trên bus. Khi máy in xử lý xong dữ liệu, nó sẽ trả lại tín hiệu

ACK xuống mức thấp để ghi nhận. PC đợi cho đến khi đường BUSY từ máy in xuống thấp (máy in không bận) thì sẽ đưa tiếp dữ liệu lên bus.



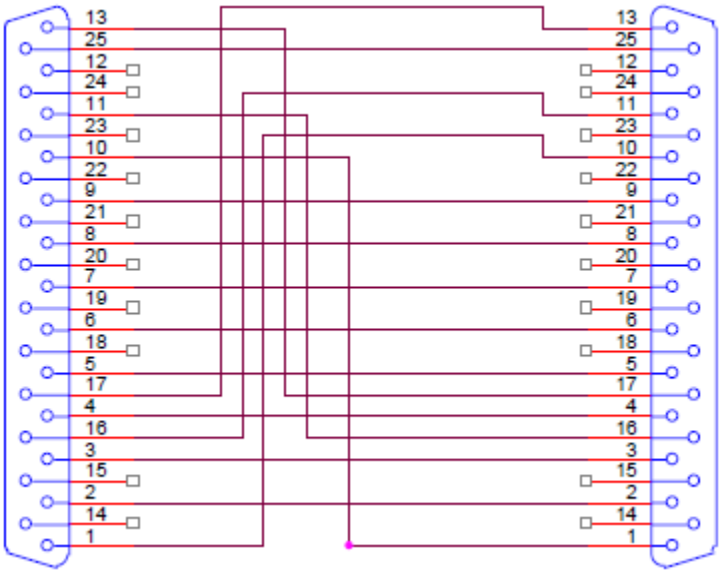
1.2. Giao tiếp với thiết bị ngoại vi

1.2.1. Giao tiếp với máy tính

Quá trình giao tiếp với cổng song song dùng 2 chế độ: chế độ chuẩn SPP và chế độ mở rộng. Việc giao tiếp ở chế độ chuẩn mô tả như sau:
Hình 5.1: Trao đổi dữ liệu qua cổng song song giữa 2 PC dùng chế độ chuẩn
Sơ đồ chân kết nối mô tả như sau:

PC1		PC2	
Chức năng	**Chân**	**Chân**	**Chức năng**
D0	2	15	$\overline{\text{ERROR}}$
D1	3	13	SELECT
D2	4	12	PAPER EMPTY
D3	5	10	$\overline{\text{ACK}}$
D4	6	11	BUSY
BUSY	11	6	D4
$\overline{\text{ACK}}$	10	5	D3
PAPER EMPTY	12	4	D2
SELECT	13	3	D1
$\overline{\text{ERROR}}$	15	2	D0
GND	25	25	GND

Ngoài ra, việc kết nối giữa 2 máy tính sử dụng cổng song song có thể dùng chế độ mở rộng, chế độ này cho phép giao tiếp với tốc độ cao hơn.



Hình 5.1: Trao đổi dữ liệu qua cổng song song giữa 2 PC dùng chế độ mở rộng

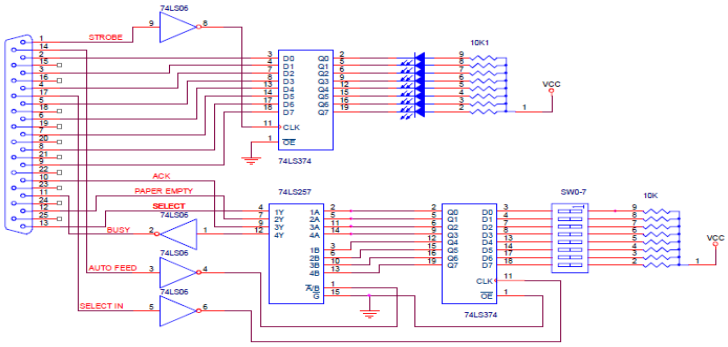
Sơ đồ chân kết nối mô tả như sau:

PC1		PC2	
Chức năng	**Chân**	**Chân**	**Chức năng**
D0	2	2	D0
D1	3	3	D1
D2	4	4	D2
D3	5	5	D3
D4	6	6	D4
D5	7	7	D5
D6	8	8	D6
D7	9	9	D7
SELECT	13	17	$\overline{\text{SELECTIN}}$
BUSY	11	16	$\overline{\text{INIT}}$
$\overline{\text{ACK}}$	10	1	$\overline{\text{STROBE}}$
$\overline{\text{SELECTIN}}$	17	13	SELECT
$\overline{\text{INIT}}$	16	11	BUSY
$\overline{\text{STROBE}}$	1	10	$\overline{\text{ACK}}$

1.2.2. Giao tiếp thiết bị khác

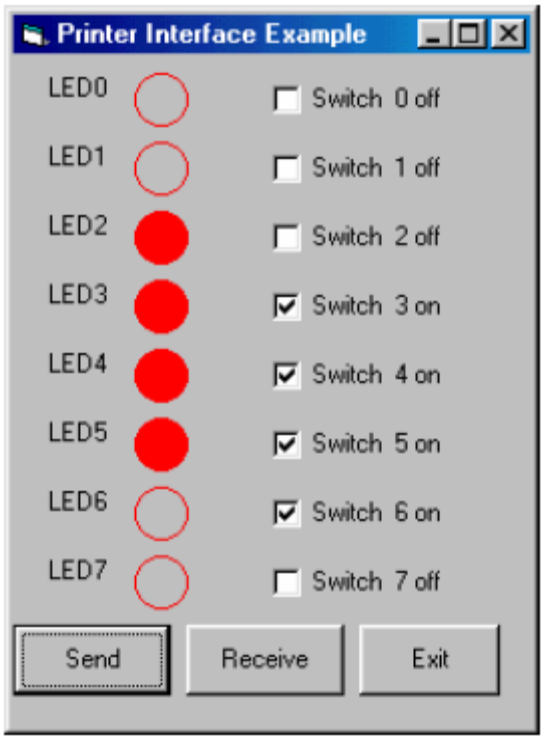
Quá trình giao tiếp với các thiết bị ngoại vi có thể thực hiện thông qua chế độ chuẩn.

Để đọc dữ liệu, có thể dùng một IC ghép kênh 2 ?? 1 74LS257 và dùng 4 bit trạng thái của cổng song song còn xuất dữ liệu thì sử dụng 8 đường dữ liệu D0 – D7.



Hình 5.2: Mạch giao tiếp đơn giản thông qua cổng máy in

Giao diện:



Hình 5.3: Giao diện của chương trình giao tiếp với cổng máy in

Chương trình giao tiếp trên VB sử dụng thư viện liên kết động để trao đổi dữ liệu với cổng máy in. Thư viện IO.DLL bao gồm các hàm sau:

- Hàm PortOut: xuất 1 byte ra cổng Private Declare Sub PortOut Lib "IO.DLL" (ByVal Port As Integer, ByVal Data As Byte) Port: địa chỉ cổng, Data: dữ liệu xuất
- Hàm PortWordOut: xuất 1 word ra cổng Private Declare Sub PortWordOut Lib "IO.DLL" (ByVal Port As Integer, ByVal Data As Integer)
- Hàm PortDWordOut: xuất 1 double word ra cổng Private Declare Sub PortDWordOut Lib "IO.DLL" (ByVal Port As Integer, ByVal Data As Long)
- Hàm PortIn: nhập 1 byte từ cổng, trả về giá trị nhật Private Declare Function PortIn Lib "IO.DLL" (ByVal Port As Integer) As Byte
- Hàm PortWordIn: nhập 1 word từ cổng Private Declare Function PortWordIn Lib "IO.DLL" (ByVal Port As Integer) As Integer
- Hàm PortDWordIn: nhập 1 double word từ cổng Private Declare Function PortDWordIn Lib "IO.DLL" (ByVal Port As Integer) As Long

Chương trình nguồn:

```
VERSION 5.00
Begin VB.Form Form1
    Caption = "Printer Interface Example"
    ClientHeight = 4665
    ClientLeft = 60
    ClientTop = 345
    ClientWidth = 3585
    LinkTopic = "Form1"
    ScaleHeight = 4665
    ScaleWidth = 3585
    StartUpPosition = 3 'Windows Default
Begin VB.CommandButton cmdReceive
    Caption = "Receive"
    Height = 495
    Left = 1200
    TabIndex = 18
    Top = 3960
    Width = 1095
End
Begin VB.CheckBox chkSW
```

```
        Height = 375
        Index = 7
        Left = 1800
        TabIndex = 17
        Top = 3480
        Width = 1575
End
Begin VB.CheckBox chkSW
        Height = 375
        Index = 6
        Left = 1800
        TabIndex = 16
        Top = 3000
        Width = 1575
End
Begin VB.CheckBox chkSW
        Height = 375
        Index = 5
        Left = 1800
        TabIndex = 15
        Top = 2520
        Width = 1575
End
Begin VB.CheckBox chkSW
        Height = 375
        Index = 4
        Left = 1800
        TabIndex = 14
        Top = 2040
        Width = 1575
End
Begin VB.CheckBox chkSW
        Height = 375
        Index = 3
        Left = 1800
        TabIndex = 13
        Top = 1560
        Width = 1575
End
Begin VB.CheckBox chkSW
        Height = 375
        Index = 2
```

```
        Left = 1800
        TabIndex = 12
        Top = 1080
        Width = 1575
End
Begin VB.CheckBox chkSW
        Height = 375
        Index = 1
        Left = 1800
        TabIndex = 11
        Top = 600
        Width = 1575
End
Begin VB.CheckBox chkSW
        Height = 375
        Index = 0
        Left = 1800
        TabIndex = 10
        Top = 120
        Width = 1575
End
Begin VB.CommandButton cmdExit
        Caption = "Exit"
        Height = 495
        Left = 2400
        TabIndex = 9
        Top = 3960
        Width = 975
End
Begin VB.CommandButton cmdSend
        Caption = "Send"
        Height = 495
        Left = 0
        TabIndex = 8
        Top = 3960
        Width = 1095
End
Begin VB.Label lblLED
        BackStyle = 0 'Transparent
        Caption = "LED7"
        Height = 375
        Index = 7
```

```
        Left = 240
        TabIndex = 7
        Top = 3480
        Width = 1095
End
Begin VB.Label lblLED
        BackStyle = 0 'Transparent
        Caption = "LED6"
        Height = 375
        Index = 6
        Left = 240
        TabIndex = 6
        Top = 3000
        Width = 975
End
Begin VB.Label lblLED
        BackStyle = 0 'Transparent
        Caption = "LED5"
        Height = 375
        Index = 5
        Left = 240
        TabIndex = 5
        Top = 2520
        Width = 975
End
Begin VB.Label lblLED
        BackStyle = 0 'Transparent
        Caption = "LED4"
        Height = 375
        Index = 4
        Left = 240
        TabIndex = 4
        Top = 2040
        Width = 975
End
Begin VB.Label lblLED
        BackStyle = 0 'Transparent
        Caption = "LED3"
        Height = 375
        Index = 3
        Left = 240
        TabIndex = 3
```

```
        Top = 1560
        Width = 975
End
Begin VB.Label lblLED
        BackStyle = 0 'Transparent
        Caption = "LED2"
        Height = 375
        Index = 2
        Left = 240
        TabIndex = 2
        Top = 1080
        Width = 975
End
Begin VB.Label lblLED
        BackStyle = 0 'Transparent
        Caption = "LED1"
        Height = 375
        Index = 1
        Left = 240
        TabIndex = 1
        Top = 600
        Width = 975
End
Begin VB.Label lblLED
        BackStyle = 0 'Transparent
        Caption = "LED0"
        Height = 375
        Index = 0
        Left = 240
        TabIndex = 0
        Top = 120
        Width = 975
End
Begin VB.Shape shpLED
        BorderColor = &H000000FF&
        FillColor = &H000000FF&
        FillStyle = 0 'Solid
        Height = 375
        Index = 7
        Left = 840
        Shape = 3 'Circle
        Top = 3480
```

```
        Width = 375
End
Begin VB.Shape shpLED
        BorderColor = &H000000FF&
        FillColor = &H000000FF&
        FillStyle = 0 'Solid
        Height = 375
        Index = 6
        Left = 840
        Shape = 3 'Circle
        Top = 3000
        Width = 375
End
Begin VB.Shape shpLED
        BorderColor = &H000000FF&
        FillColor = &H000000FF&
        FillStyle = 0 'Solid
        Height = 375
        Index = 5
        Left = 840
        Shape = 3 'Circle
        Top = 2520
        Width = 375
End
Begin VB.Shape shpLED
        BorderColor = &H000000FF&
        FillColor = &H000000FF&
        FillStyle = 0 'Solid
        Height = 375
        Index = 4
        Left = 840
        Shape = 3 'Circle
        Top = 2040
        Width = 375
End
Begin VB.Shape shpLED
        BorderColor = &H000000FF&
        FillColor = &H000000FF&
        FillStyle = 0 'Solid
        Height = 375
        Index = 3
        Left = 840
```

```
        Shape = 3 'Circle
        Top = 1560
        Width = 375
End
Begin VB.Shape shpLED
        BorderColor = &H000000FF&
        FillColor = &H000000FF&
        FillStyle = 0 'Solid
        Height = 375
        Index = 2
        Left = 840
        Shape = 3 'Circle
        Top = 1080
        Width = 375
End
Begin VB.Shape shpLED
        BorderColor = &H000000FF&
        FillColor = &H000000FF&
        FillStyle = 0 'Solid
        Height = 375
        Index = 1
        Left = 840
        Shape = 3 'Circle
        Top = 600
        Width = 375
End
Begin VB.Shape shpLED
        BorderColor = &H000000FF&
        FillColor = &H000000FF&
        FillStyle = 0 'Solid
        Height = 375
        Index = 0
        Left = 840
        Shape = 3 'Circle
        Top = 120
        Width = 375
End
End
        Attribute VB_Name = "Form1"
        Attribute VB_GlobalNameSpace = False
        Attribute VB_Creatable = False
        Attribute VB_PredeclaredId = True
```

```
        Attribute VB_Exposed = False
        'IO.DLL
        Private Declare Sub PortOut Lib "IO.DLL" (ByVal Port
        As Integer, ByVal Data As Byte)
        Private Declare Function PortIn Lib "IO.DLL" (ByVal
        Port As Integer) As Byte
        'Variable
        Private BA_LPT As Integer
        Private Sub cmdExit_Click()
End
End Sub
        Private Sub cmdReceive_Click()
        Dim n As Integer
        Dim n1 As Integer
        Dim i As Integer
        PortOut BA_LPT + 2, &H8 'SELECTIN = 1
        PortOut BA_LPT + 2, 0 'SELECTIN = 0
        n1 = PortIn(BA_LPT + 1) 'Doc 4 bit thap
        n1 = n1 / &H10 'Dich phai 4 bit
        PortOut BA_LPT + 2, 2 'AUTOFEED=1
        n = PortIn(BA_LPT + 1) 'Doc 4 bit cao
        n = n And &HF0
        n = n + n1
        For i = 0 To 7
        chkSW(i).Value = n Mod 2
        If chkSW(i).Value = 0 Then
                chkSW(i).Caption = "Switch " & Str(i) &
                " off"
        Else
                chkSW(i).Caption = "Switch " & Str(i) &
                " on"
        End If
                n = Fix(n / 2)
                Next i
End Sub
        Private Sub cmdSend_Click()
        Dim t As Integer
        Dim i As Integer
        Dim s As String
        t = 0
        For i = 0 To 7
        t = t + (2 ^ i) * (1 - shpLED(i).FillStyle)
```

```
        Next i
        PortOut BA_LPT, t
        PortOut BA_LPT, 1 'STROBE = 1
        PortOut BA_LPT, 0 'STROBE = 0
End Sub
        Private Sub Form_Load()
        BA_LPT = &H378
        PortOut BA_LPT + 2, 0
End Sub
        Private Sub lblLED_Click(Index As Integer)
        shpLED(Index).FillStyle = 1 -
        shpLED(Index).FillStyle
End Sub
```

2. Xuất dữ liệu ra cổng song

Mục tiêu:

- Nắm được cấu tạo và nguyên tắc hoạt động của bộ tạo hàm

- Viết được chương trình tạo ra một bảng giá trị

- Viết được chương trình điều khiển mô tơ

Giao diện máy in có khả năng xuất ra 8 bit dữ liệu cùng một lúc hay như thường nói là xuất ra 8 bit dữ liệu dưới dạng song song. Thông qua giao diện máy in có thể điều khiển, chẳng hạn các mạch logic hoặc các bộ đệm công suất (chịu dòng lớn). trái ngược với việc xuất ra nối tiếp, việc xuất ra qua cổng song song diễn ra với một lệnh cổng đơn giản và do nên mà cực kỳ nhanh. Đặc tính này mở ra khả năng tạo ra tín hiệu analog tần số thấp nhưng chất lượng cao.

2.1. Một bộ tạo hàm

Một bộ tạo hàm tạo ra tín hiệu điện áp xoay chiều với tần số biến đổi và dạng của đường cong lựa chọn được. để có được thiết bị này bằng mọt máy tính PC, thay cho việc tạo ra các tín hiệu số ta cần phải tạo ra một tín hiệu analog. Để chuyển đổi một tín hiệu số thành một tín hiệu analog ta cần có một bộ biến đổi số/ tương tự chẳng hạn như loại ZN426 của công ty ferranti (Hoa kỳ). Hình 10-1 chỉ ra cách đấu nối ở cổng máy in. Vi mạch có chứa một nguồn điện áp so sánh bên trong bằng 2,55v. Do vậy mà 256 bậc điện áp, mỗi bậc có giá trị bằng 10 mV có thể được phân giải và điện áp lối ra đạt tới giá trị từ 0 V đến 2,55 V.

Đoạn chương trình 10-1 chỉ ra một chương trình nhằm tạo lập một máy phát tín hiệu âm tần hình sin. Chương trình tạo ra một bảng giá trị, khi chương trình khởi động, bảng này được xếp vào các ô bằng các giá trị của hàm. Trong quá trình xuất ra, một bảng hình sin đã chuẩn bị sẵn được truy cập trực tiếp. Tốc độ xuất ra có thể thiết lập được qua một thanh ghi dịch trong khoảng từ 10 us đến 1 ms theo từng giá trị dùng làm mốc. việc xuất ra

tiến hành theo chu kỳ bằng năm giây một lần. trong khoảng thời gian này máy tính không phản ứng trước các tác động nhập vào khác.

```
Unit   PLTsinus;
Interface
Uses       PORTINC,
       Windows, Messages, Sysutils, Classes, Graphics,
            Controls, Forns, Dialogs, stdctrls;
Type
    Tforml = classs(ttorm)
     scrollBarl: TscrollBar;
     Editl: Tedit;
     Buttonlp: Tbutton;
     Procedure  FormCreate( sender: Tobject);
     Procedure  ButtonlClick( Sender: Tobject);
     Procedure  ScrollBarkchange (Sender: Tobject);
  End;
Var;
   Forml : Tforml;
   Bang : Array [0..255] of  Byte;    { Bang: Bản}
   BA, Thoi_gian: Word;
Implementation
 {$R * .DFM}
Procedure  Tforml. FormCreate( Sender: Tobject );
Var n : Integer;
Begin
   OpenCom ( Pchar ('LPT1:'));
   BA  : = $378;
    For    n  : =  0 to  255 do
       Bang    [n]   : = round (127.5 + 127.5*sin  ( n/128*  Pi));
   Thoi_gian   : =  10;
End;
Procedure Tfoml. ButtonlClick (Sender: Tobject );
Var  t : Dword;
       n : byte;
begin
   Thoi_gian    : = ScrollBarl. Position;
    T : =  0;   n :  = 0;
   RealTime (true) ;
   TimeInitus;
While TimeReadus  < 5000000 do begin
         OutPort  (BA,Bang inll;
         Inc (n) ; t := t – thoi_gian;
```

```
        While  TimeReadus < 0 đo ;
    End ;
    RealTime (false) ;
End ;

Procedure Tforml. SorollBảlChange (Sender : Tobject) ;
Begin
Editl. Text := floattostr (scrollbarl. Position) + 'us'
End ;
End.
```

2.2. Điều khiển những máy móc đơn giản

Giao diện máy in có tổng cộng mười hai đường dẫn xuất ra và năm đường dẫn nhập vào. Nhờ vậy có thể điều khiển môtơ và kiểm tra các công tắc. Khi ta sử dụng các môtơ bước theo cách điều khiển 4 bit (so sánh mục 3.3), thì có thể điều khiển ba môtơ cùng lúc. Các môtơ này có thể vận hành các mô hình chuyển động đơn giản.

Được điều khiển qua bộ đệm Darlington loạei UNL2803. Ngoài ra còn có ba công tắc được đấu vào, các công tắc này có thể được máy tính kiểm tra.

. Các thủ tục "Motor1" đến "Motor3" thực hiện thao tác một bước "sang trái" hoặc "sang phải" theo hướng đã chỉ định. Ba công tắc S1 đến S3 dùng làm công tắc tận cùng dùng cho môtơ M1 đén M3 và được kiểm tra bằng hàm "công_tắc1" đến "công_tắc3". Như vậy, một trạng thái ban đầu xác định của tất cả các môtơ có thể đạt được bằng thủ tục "Vị_trí_Null".

Việc điều khiển chuyển động thực diễn ra trong thủ tục "CH_DONG1", trong đó chỉ ra một thí dụ rất đơn giản. Tất cả các môtơ có thể được điều khiển riêng lẻ hoặc đồng thời. Thời gian chờ ngắn được quy định qua lệnh Delay sẽ ấn định tốc độ của môtơ. Khi đó tốc độ cực đại được cần phải được giữ đúng, nhờ vậy không có bước nào bị bỏ qua. Môtơ chịu tải lớn cần phải điều khiển chậm hơn.

```
Unit LPT1;
Interface
Une PORTINC,
        Windows, Messages, Sysutils, Classes, Graphics, Controls, Forms,
        Dialogs, Sdctrls;

Type
        Tfom1 = class (tfom)
                Button1 : Tbuton;
                Button2 : Tbuton;
                Procedure FormCreate (Sender: Tobject);
```

```
                Procedure ButtonlClick (Sender: Tobject);
                Procedure Button2Click (Sender: Tobject);
        End;
Var
        Form : Trorml;
        Buocl, Buoc1, Buoc3 : Byte;
Const   BA = $378;                      {NPT1}
                Rechts = true;
                Links = false;
Implementation
{$R *.DFM}
Procdere Motor1 (rechts ; Boonlean);
Begin
        If rechts then begin
                Buoc1 := buoc1*2;
                If  buoc1 = 16 then buoc1 := 1
        End else begin
                Buoc1 := Buoc1 div 2
                If  Buoc1 = 0 then Buoc1 := 8;
        End;
        Out Port (Ba, Buoc1 + 16 * Buoc2);
End;
Procedure Motor2 (rechts : Boolean);
Begin
        If rechts then begin
                Buoc2 := Buoc2 * 2;
                If  Buoc2 = 16 then Buoc2 :=1
        End else begin
                Buoc2 := Buoc2 div 2;
                If Buoc2 = 0 then Buoc2 := 8;
        End;
        outPort (BA, Buoc1+16 * Buoc2);
end;
procedure Motor3 (rechts : Boolean);
begin
        if rechts then begin
                Buoc3 := Buoc3 * 2;
                If Buoc3 = 16 then Buoc3 :=1
        End else begin
                Buoc3 := Buoc3 div 2;
                If Buoc3 = 0 then Buoc3 := 8;
        End;
```

```
        OutPort (BA+2, Buoc3 XOR 11)
End;
Function Cong_tac1 : Boolean;          {Cong_tac : Công tắc }
Begin
        If (Inport (BA+1) and 8) = 0 then Cong_tac1 := true
Else Cong_tac1 := false;
End;

Function Cong_tac2 : Boolean;
Begin
        If (Inport (BA+1) and 16) = 0 then Cong_tac2 := true
Else Cong_tac2 := false;
End;

Function Cong_tac3 : Boolean;
Begin
        If (Inport (BA+1) and 32) = 0 then Cong_tac3 := true
Else Cong_tac3 := false;
End;

Procedure NullPosition;
Begin
        Repeat
                Mororl (links);
                Delay (5);
        Until  Cong_tac1;
Repeat
                Moror2 (links);
                Delay (5);
        Until  Cong_tac2;
Repeat
                Moror3 (links);
                Delay (5);
        Until  Cong_tac3;
End;
Procedure Chuyen_dong1;        Chuyen_dong : chuyển động
Var N : Word;
Begin
        For n := 1 to 100 do begin         { 100 buoc }
                Motorl ( rechts );
                Motor2  ( rechts );        { tất cả các môtơ }
                Motor3  ( rechts );
```

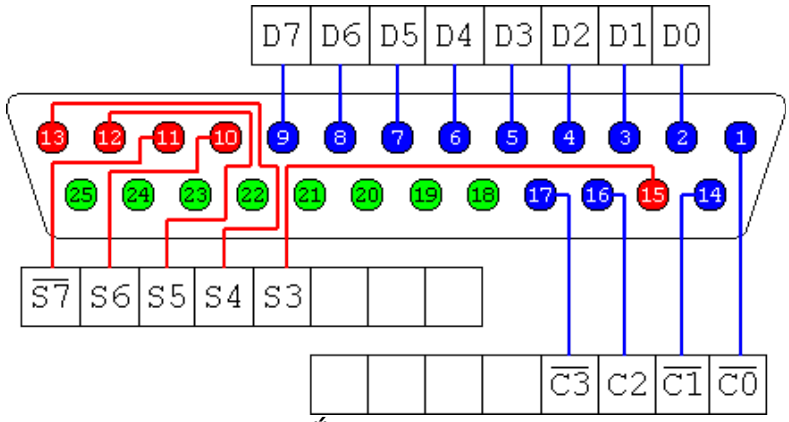
```
            Delay ( 5);                 { nhanh }
        End;
        For n := 1to 100 do begin       { 100 buoc }
            Motorl  rechts );           {chỉ môtơ 1  }
            Delay (10);                 { cham }
        End;
        Delay ( 1000);                  { tr_thái đứng 1s }
        For n :=1 to 100 do begin       { 100 buoce }
            Motor1 ( links ) ;
            Motor2 ( links ) ;          { tất cả các môtơ }
            Motor3 ( links ) ;
            Delay (5);                  { nhanh}
        End;
        For n:= 1 to 100 begin          {100 bước }
            Motorl ( links);            { chỉ môtơ 1 }
            Delay (10);                 { chậm }
        End;
        Delay (1000);                   { tr_thái đứng 1s }
End;

Procedure Tforml. FormCreate ( Sender : Tobject);
Begin
        openCom (Pchar ( 'LPT1:'));
        Buoc1 := 2;
        Buoc2 := 2;
        Buoc3 := 2;
        Uot Port (BA,17);               { D0 = 1, D4 = 1 }
        Out Port (BA+2, 1 XOR 11 ) ;    { Strobe = 1 }
End;
Procedure Tforml. ButtonlClick ( Sender : Tobject );
Begin
        nullPosition;
end;
Procedure Tforml. ButtonlClick ( Sender : Tobject );
Begin
        Chuyen_dong1;
End;
End.
```

2.3. Ghép nối song song qua cổng máy in

❖ **Giới thiệu chung**

Cổng máy in là giao diện thường được sử dụng nhiều nhất trong các ứng dụng ghép nối máy tính đơn giản, do tính phổ cập và đơn giản trong việc ghép nối và điều khiển cộng với yêu cầu tối thiểu về thiết bị phần cứng thêm vào. Cổng này cho phép đưa vào tới 13 bit và đưa ra 12 bit song song, trong đó có 4 đường điều khiển, 5 đường báo trạng thái và 8 đường dữ liệu. Trong hầu như bất kỳ PC nào ta cũng có thể tìm thấy cổng máy in ở phía sau. Đầu



nối này có dạng DB 25 chân (giắc cái – female).

Các cổng song song gần đây được chuẩn hoá theo chuẩn IEEE 1284 đưa ra năm 1994. Chuẩn này mô tả 5 chế độ hoạt động của cổng máy in như sau:

1. Chế độ tương thích (Compatibility mode)
2. Chế độ Nibble
3. Chế độ Byte
4. Chế độ EPP
5. Chế dộ ECP

Chế độ cớ sở (hay còn gọi là Centronics mode) được biết đến từ lâu. Chế độ này chỉ cho phép đưa dữ liệu theo một chiều ra (output), với tốc độ tối đa 150kB/s. Muốn thu dữ liệu (input) ta phải chuyển sang chế độ Nibble hay Byte. Chế độ Nibble có thể cho phép đưa vào 4 bit song song một lần. Chế độ Byte sử dụng tính năng song song hai hướng của cổng máy in để đưa vào một byte.

Để đưa ra một byte ra máy in (hoặc các thiết bị khác) trong chế độ cơ sở, phần mềm phải thực hiện các bước sau:

(1) Viết dữ liệu ra cổng máy in (ghi vào thanh ghi dữ liệu)

(2) Kiểm tra máy in có bận không, nếu máy in bận, nó sẽ không chấp nhận bất cứ dữ liệu nào, do đó dữ liệu ghi ra lúc đó sẽ bị mất

(3) Nếu máy in không bận, đặt chân Strobe (chân 1) xuống thấp (mức 0), để báo với máy in là đã có dữ liệu trên đường truyền (chân 2 - 9)

(4) Sau đó chờ 5 microgiây và đặt chân Strobe lên cao (mức 1).

Chế độ mở rộng (EPP) và nâng cao (ECP) sử dụng các thiết bị phần cứng tích hợp thêm vào để thực hiện và quản lý việc đối thoại với thiết bị ngoài. Ở chế độ này để cho phần cứng kiểm tra trạng thái máy in bận, tạo xung strobe và thiết lập sự bắt tay thích hợp. Do đó chỉ cần sử dụng một lệnh vào

ra để trao đổi dữ liệu nên giúp tăng tốc độ thực hiện. Khi đó cổng này có thể đưa dữ liệu ra với tốc độ 1 – 2 MB/s. Ngoài ra chế độ ECP còn hỗ trợ sử dụng kênh DMA và có thêm bộ đệm FIFO.

❖ **Cấu trúc cổng máy in**

Chuẩn IEEE 1284 đưa ra 3 đầu nối dùng cho cổng máy in. Dạng A (DB25) có thể thấy ở hầu hết các máy PC, dạng B (36 chân) thường thấy ở máy in, và dạng C, 36 chân, giống dạng B nhưng nhỏ hơn, có các thuộc tính điện tốt hơn và có thêm 2 đường tín hiệu dành cho các thiết bị đời mới sau này.

Số hiệu chân (DB25)	Tên	Hướng (In/Out)	Thanh ghi	Mô tả
1	nStrobe	In/Out	Control	Byte được in
2	Data 0	Out	Data	Đường dữ liệu D0 - D7
3	Data 1	Out	Data	
4	Data 2	Out	Data	
5	Data 3	Out	Data	
6	Data 4	Out	Data	
7	Data 5	Out	Data	
8	Data 6	Out	Data	
9	Data 7	Out	Data	
10	nAck	In	Status	Xác nhận (Acknowledge)
11	Busy	In	Status	Máy in bận
12	Paper-Out / Paper-End	In	Status	Hết giấy (Paper Empty)
13	Select	In	Status	Lựa chọn (Select)
14	nAuto-Linefeed	In/Out	Control	Tự nạp giấy (Auto Feed)
15	nError / nFault	In	Status	Lỗi
16	nInitialize	In/Out	Control	Đặt lại máy in
17	nSelect-Printer / nSelect-In	In/Out	Control	
18 - 25	Ground	Gnd		

nXXXX: Tích cực ở mức thấp

Bảng sơ đồ chân của cổng máy in

Tín hiệu ra của cổng máy in thường ở các mức logic TTL.

Address	Cæng
378h - 37Fh	LPT 1
278h - 27Fh	LPT 2

Khi khởi động BIOS gán địa chỉ cho các cổng máy in và lưu thông tin địa chỉ này trong bộ nhớ ở địa chỉ cho ở bàng dưới:

Địa chỉ bắt đầu	Mô tả
0000:0408	Địa chỉ cơ bản cổng LPT1
0000:040A	Địa chỉ cơ bản cổng LPT2
0000:040C	Địa chỉ cơ bản cổng LPT3
0000:040E	Địa chỉ cơ bản cổng LPT4

Chương trình ví dụ đọc thông tin địa chỉ của các cổng máy in có trong máy tính:

```
#include <stdio.h>
#include <dos.h>

void main(void)
{
 unsigned int far *ptraddr;  /* Pointer to location of Port Addresses
*/
 unsigned int address;       /* Address of Port */
 int a;

 ptraddr=(unsigned int far *)0x00000408;

 for (a = 0; a < 3; a++)
   {
    address = *ptraddr;
    if (address == 0)
                printf("No port found for LPT%d \n",a+1);
    else
                printf("Address assigned to LPT%d is
%Xh\n",a+1,address);
    *ptraddr++;
   }
}
```

Các thanh ghi của cổng máy in:

+ Thanh ghi dữ liệu (Data Register)

§Þa chØ	Tªn	Read/Write	Sè hiÖu bit	M« t¶
Base + 0	Data Port	Write	Bit 7	Data 7
			Bit 6	Data 6
			Bit 5	Data 5
			Bit 4	Data 4
			Bit 3	Data 3
			Bit 2	Data 2
			Bit 1	Data 1
			Bit 0	Data 0

Địa chỉ cơ sở (Base address) thường gọi là cổng dữ liệu (Data port) hay Thanh ghi dữ liệu (Data Register) thường sử dụng để đưa dữ liệu ra các chân tín hiệu (Chân 2 – 9). Thanh ghi này thường là thanh ghi chỉ ghi. Nếu ta đọc dữ liệu ở cổng này ta sẽ thu được giá trị mà ghi ra gần nhất. Nếu cổng máy in là hai chiều thì ta có thể thu giữ liệu vào từ cổng này.

+ Thanh ghi trạng thái (Status Register):

Địa chỉ	Tên	Read/Write	Số hiệu bit	Mô tả
Base + 1	Status Port	Read Only	Bit 7	Busy
			Bit 6	Ack
			Bit 5	Paper Out
			Bit 4	Select In
			Bit 3	Error
			Bit 2	IRQ (Not)
			Bit 1	Reserved
			Bit 0	Reserved

Thanh ghi trạng thái là thanh ghi chỉ đọc. Bất kỳ dữ liệu nào viết ra cổng này đều bị bỏ qua. Cổng trạng thái được tạo bởi 5 đường tín hiệu vào (Chân 10, 11, 12, 13, 15), một bit trạng thái ngắt IRQ và 2 bit để dành. Chú ý rằng bit 7 (Busy) là đầu vào tích cực thấp, nghĩa là khi có một tín hiệu +5V ở chân 11, bit 7 sẽ có giá trị logic 0. Tương tự với bit 2 (nIRQ) nếu có giá trị 1 có nghĩa là không có yêu cầu ngắt nào xuất hiện.

+ Thanh ghi điều khiển (Control Register):

Địa chỉ	Tên	Read/Write	Số hiệu bit	Mô tả
Base + 2	Control Port	Read/Write	Bit 7	Unused
			Bit 6	Unused
			Bit 5	Enable Bi-Directional Port
			Bit 4	Enable IRQ Via Ack Line
			Bit 3	Select Printer
			Bit 2	Initialize Printer (Reset)
			Bit 1	Auto Linefeed
			Bit 0	Strobe

Thanh ghi điều khiển được dự định là chỉ ghi. Khi một máy in được nối với máy tính, 4 đường điều khiển sẽ được sử dụng. Đó là các đường Strobe, Auto Linefeed, Inittialize và Select Printer, tất cả đều là đầu ra đảo trừ đường Initialize.

Bit 4 và 5 là các bit điều khiển nội. Bit 4 cho phép ngắt và bit 5 cho phép chế độ vào ra 2 chiều. Đặt bit 5 cho phép thu đữ liệu vào qua đường Data 0 – 7.

+ Thanh ghi điều khiển mở rộng ECR (Extended Control Register):

§Þa chØ	Bit	Function	
Base + 402H	7:5	*Selects Current Mode of Operation*	
		000	Standard Mode
		001	Byte Mode
		010	Parallel Port FIFO Mode
		011	ECP FIFO Mode
		100	EPP Mode
		101	Reserved
		110	FIFO Test Mode
		111	Configuration Mode
	4	ECP Interrupt Bit	
	3	DMA Enable Bit	
	2	ECP Service Bit	
	1	FIFO Full	
	0	FIFO Empty	

+ EPP - Enhanced Parallel Port

Cổng song song nâng cao (EPP) đa˜được thiết kê´bởi sự liên kết giữa các hãng Intel, Xircom & Zenith Data Systems. Cổng EPP ban đâu được thiết kê´ theo chuẩn va`sau đo´la`chuẩn IEEE 1284 ra đơi năm 1994. EPP co´hai chuẩn: EPP 1.7 va`EPP 1.9. Co´một vai sự khac nhau giữa các chuẩn nay ma`chuńg co´ nhưng ảnh hưởng tơí các thao tac xử ly´của thiết bị. Vấn đê`nay se˜coǹ được noí đến trong phân sau. EPP co´tôc độ truyên dư˜liệu theo tiêu chuẩn la`tư` 500KB/s tơí 2MB/s. Điêu nay cho phep các thiết bị phân cưńg tại các cổng tạo ra tín hiệu băt tay (tín hiệu moc nôí, hội thoại) chẳng hạn như tín hiệu *stroble*, để phân mêm xử ly´chuńg, vi´dụ như của Centronics.

EPP được sử dụng rộng raĩ hơn ECP. EPP khac vơí ECP ở chô˜cổng EPP phat ra các tín hiệu điêu khiển va`điêu khiển tât cả qua´trình truyên dư˜liệu tư`no´tơí thiết bị ngoại vi. Bên cạnh đo´thi`ECP lại yêu câu thiết bị ngoại vi co´ sự "hội thoại" trở lại bởi một tín hiệu moc nôí. Điêu nay la`không mêm dẻo cho việc thiết lập một liên kết logic va`như vậy cân co´một bộ điêu khiển chuyên dụng hoặc một chip ngoại vi ECP.

EPP Hardware Properties (các đặc trưng phân cưńg EPP)

Khi sử dụng chê´độ EPP, một tập các tac vụ khac nhau (co´tên tương ưńg) được săp xêp trên môĩ đương dây tín hiệu. Các tín hiệu nay được chỉ ra trong bảng 4. Chuńg sử dụng các tên chung trong SPP va`EPP trong các bảng mô tả vê`cổng song song va`các tai liệu. Điêu nay co´thể lam cho no´rât cưńg nhăc để chỉ ro˜chính xac nhưng gi`đang xảy ra. Mặc du`tât cả các tai liệu ở đây đêu se˜sử dụng tên theo EPP.

Pin	SPP Signal	EPP Signal	IN/OUT	Function
1	Strobe	Write	Out	Mức thấp thể hiện một chu kỳ ghi, mức cao chỉ định là đang đọc
2-9	Data 0-7	Data 0-7	In-Out	Data Bus. Hai chiều
10	Ack	Interrupt	In	Interrupt Line. Ngắt xuất hiện ở sườn dương của xung
11	Busy	Wait	In	Used for handshaking. A EPP cycle can be started when low, and finished when high.
12	Paper Out / End	Spare	In	Spare - Not Used in EPP Handshake
13	Select	Spare	In	Spare - Not Used in EPP Handshake
14	Auto Linefeed	Data Strobe	Out	Khi ở mức thấp, chỉ định là đang truyền dữ liệu (data)
15	Error / Fault	Spare	In	Spare - Note used in EPP Handshake

16	Initialize	Reset	Out	Reset - Tích cực thấp
17	Select Printer	Address Strobe	Out	Khi ở mức thấp, chỉ định đang truyền đại chỉ
18-25	Ground	Ground	GND	Ground

Bảng 1 Sự xăp xếp cac chân của EPP.

Cac tín hiệu Paper Out, Select va`Error không được xac định trong tập cac tín hiệu băt tay của EPP. Cac tín hiệu nay co´thể được sử dụng tuy`y´theo sự định nghiã của ngươì sử dụng. Trạng thai của cac được tín hiệu nay co´thể được xac định tại bât ky`thơì điểm nao theo sự xăp xếp tín hiệu của thanh ghi trạng thai. Đang tiêc la`không co´đâu ra thưà. Điêu nay co´thể trở nên phưc tạp cho việc xac định trạng thai tại một thơì điểm nao đo´của cho ky`truyên/nhận thông tin.

Các thanh ghi trong chế độ EPP

Chế độ EPP có một tập các thanh ghi mới, trong đó có 3 thanh ghi đã có từ chế độ SPP

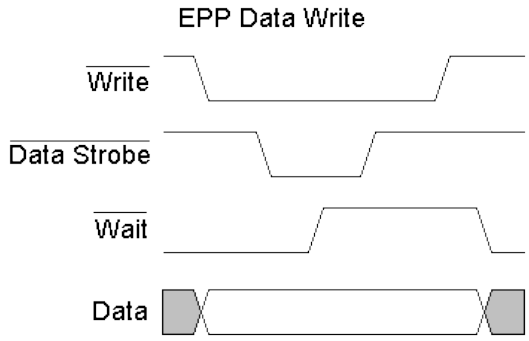
Address	Port Name	Read/Write
Base + 0	Data Port (SPP)	Write
Base + 1	Status Port (SPP)	Read
Base + 2	Control Port (SPP)	Write
Base + 3	Address Port (EPP)	Read/Write
Base + 4	Data Port (EPP)	Read/Write
Base + 5	Undefined (16/32bit Transfers)	-
Base + 6	Undefined (32bit Transfers)	-
Base + 7	Undefined (32bit Transfers)	-

Quá trình bắt tay của EPP

Theo trình tự thực hiện một chu ky`truyên dư liệu hợp khi sử dụng EPP, chung ta phải theo thư´tự băt tay của EPP. Do phân cưng laìm tât cả mọi việc nên cac tín hiệu băt tay nay chỉ được sử dụng cho phân cưng của chung ta ma` không được sử dụng cho phân mêm như trong trương hợp vơí SPP. Để khởi

tạo cho một chu ky`EPP, phân mêm chỉ cân thực hiện một thao tác vaò/ra để khởi tạo cho thanh ghi EPP. Chi tiêt vê`vấn đê`nay se~noí cụ thể sau.

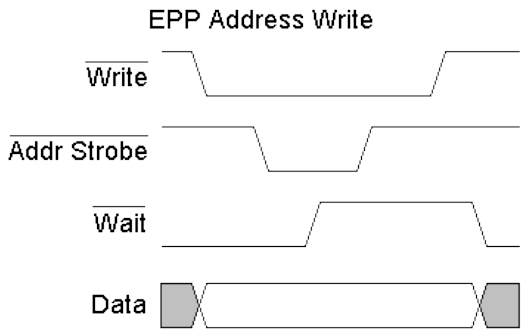
EPP Data Write Cycle



Hình 1. Enhanced Parallel Port Data Write Cycle.

1. Chương trình ghi dữ liệu vào thanh ghi dữ liệu EPP (Base+4)
2. /Write được xoá về 0. (Cho biết đang có một thao tác ghi)
3. Dữ liệu được đặt lên đường truyền dữ liệu (2 – 9).
4. /Data Strobe được kích hoạt nếu /Wait đang ở mức thấp (Sẵn sàng bắt đầu một chu kỳ mới)
5. Máy tính chờ tín hiệu xác nhận thể hiện bởi /Wait chuyển sang mức cao
6. Ngừng kích hoạt /Data Strobe
7. Chu kỳ ghi dữ liệu EPP kết thúc

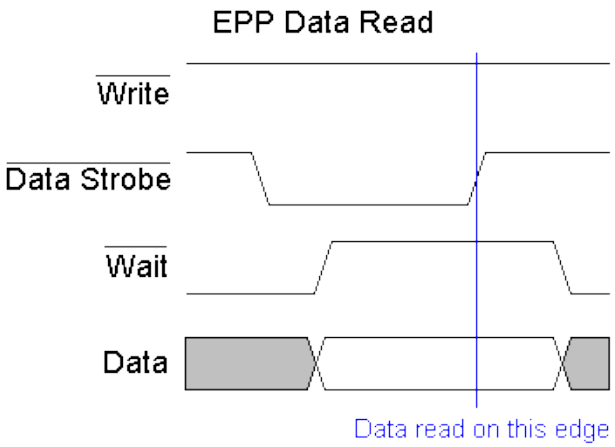
Quá trình gửi địa chỉ EPP (Address Write Cycle)



Hình 2. Enhanced Parallel Port Address Write Cycle.

1. Chương trình ghi giá trị địa chỉ vầothnh ghi địa chỉ EPP (Base+3)
2. /Write được xoá về 0. (Cho biết qúa trình ghi)
3. Giá trị địa chỉ được đặt lên đường truyền dữ liệu (2 – 7).
4. /Address Strobe được kích hoạt nếu /Wait đang ở mức thấp (Sẵn sàng bắt đầu)
5. Máy tính chờ tín hiệu xác nhận vứng với /Wait đặt lên mức cao (TBN đã đọc địa chỉ xong)
6. Tín hiệu /Address Strobe ngừng tích cực
7. Chu kỳ gửi địa chỉ EPP

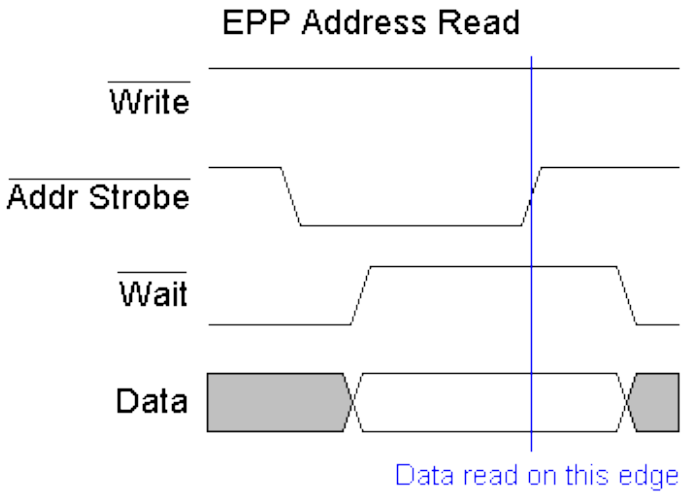
EPP Chu kỳ đọc dữ liệu



Hình 3. Enhanced Parallel Port Data Read Cycle.

1. Chương trình ra lệnh đọc thanh ghi dữ liệu EPP (Base+4)
2. /Data Strobe được kích hoạt nếu /Wait đang ở mức thấp(Sẵn sàng một chu kỳ mới)
3. Máy tính chờ tín hiệu xác nhận (/Wait chuyển sang mức cao)
4. Dữ liệu được đọc từ các chân tín hiệu của cổng
5. Ngừng kích hoạt tín hiệu /Data Strobe
6. Kết thúc chu kỳ đọc dữ liệu

EPP Address Read Cycle



Hình 4. Enhanced Parallel Port Address Read Cycle.

1. Program reads EPP Address Register (Base+3)
2. nAddr Strobe is asserted if Wait is Low (OK to start cycle)
3. Host waits for Acknowledgment by nWait going high
4. Data is read from Parallel Port Pins
5. nAddr Strobe is de-asserted
6. EPP Address Read Cycle Ends

Chu´y: *Nếu sử dụng EPP 1.7 (trước IEEE 1284) tín hiệu Strobes cho dữ liệu va`địa chỉ co´thể được dùng để xac nhận sự băt đâu của một chu ky`rôĩ của trạng thai đợi. EPP 1.9 se˜chỉ băt đâu một chu ky`đợi ở mức thâp. Cả EPP 1.7 va`EPP 1.9 chuyển tín hiệu đợi (strobe) lên mức cao để kêt thuc chu ky`.*

Các thanh ghi sử dụng trong chế độ EPP

Cổng EPP cũng co´một tập cać thanh ghi mơí. Tuy nhiên co´3 thanh ghi la` đa˜co´trươć trong cổng song song chuẩn. Bảng sau cho thấy cać thanh ghi đa˜co´ va`cać thanh ghi mơí.

Address	Port Name	Read/Write
Base+0	Data Port (SPP)	Write
Base+1	Status Port (SPP)	Read
Base+2	Control Port (SPP)	Write
Base+3	Address Port (EPP)	Read/Write
Base+4	Data Port (EPP)	Read/Write
Base+5	Undefined (16/32bit Transfers)	-
Base+6	Undefined (32bit Transfers)	-
Base+7	Undefined (32bit Transfers)	-

Bảng 2: EPP Registers

Như ta co´thể thấy, 3 thanh ghi đâu la`giống hệt cać thanh ghi trong tập thanh ghi của cổng song song chuẩn va`chức năng cũng la`giống. Vi`thê´nêu ta sử dụng một EPP ta co´thể đưa dư˜liệu ra thanh ghi dư˜liệu (Base+0) theo kiểu giống như ta co´thể đưa dư˜liệu ra nêu sử dụng SPP (Standard Parallel Port). Nêu ta đa˜kêt nôi vơí một maý in va`sử dụng chê´độ phu`hợp, sau đo´ta phải kiểm tra xem cổng co´bận không, tiêp theo ta co´thể baó (strobe) va`kiểm (Ack) tra thông qua việc ghi/đọc thanh ghi điêù khiển va`trạng thaí.

Nêú muôń truyên thông vơí một thiêt bị tương thích EPP thì tât cả công việc ta phải làm là gửi dư˜liệu ra thanh ghi dư˜liệu EPP (EPP Data Register) tại địa chỉ Base+4 va`cổng máy in se˜sinh ra tât cả cać tín hiệu băt tay cân thiêt. Tương tự như vậy, nêú muôń gửi một địa chỉ tơí thiêt bị, ta sử dụng thanh ghi địa chỉ EPP (EPP Address Register) tại địa chỉ Base+3.

Cả thanh ghi địa chỉ (Address Register) va`dữ liệu (Data Register) đêù co´ thể đọc va`ghi, do đó để đọc dư˜liệu tư`thiêt bị ta có thể sử dụng cùng một thanh ghi. mặc dù, card maý in phải khởi phát một chu ky`đọc vơí tín hiệu *Data Strobe* hoặc *Address Strobe* đâù ra. Thiêt bị ngoài vẫn co´thể đưa ra tín hiệu yêu cầu đọc qua đương tín hiệu yêu câù ngăt va`ISR (chương trình con phục vụ ngăt) sẽ thực hiện công việc đọc.

Cổng trạng thaí co´một sô´thay đổi nhỏ. Bit 0 la`để dự trư˜đôí vơí tập thanh ghi của SPP thi`giơ`đây no´la`Bit Time-out EPP. Bit nay se˜được lập khi xuât hiện một Time-out EPP. Sự kiện nay xảy ra khi đương tín hiệu *nWait* la` không được xac nhận trở lại trong khoảng 10us (gia´trị nay tuy`thuộc vao cổng khać nhau) của tín hiệu IOR hoặc IOW đa˜được xac nhận. Cać tín hiệu IOR va` IOW la`cać tín hiệu đọc va`ghi thiêt bị (I/O Read va`I/O Write) trên bus ISA.

Chê´độ EPP co´giản đô`thời gian rât giông vơí giản đô`thời gian của bus ISA. Khi thực hiện một chu ky`đọc, cổng phải đảm nhận trach nhiệm điêù khiển phu`hợp cac tín hiệu hội thoại Read/Write va`trả lại dư~liệu như trong chu ky` bus của ISA. Tât nhiên qua´trình nay không đông thơì vơí chu ky`bus ISA, vi` thê´cổng sử dụng tín hiệu điêù khiển IOCHRDY (I/O Channel Ready) trên bus ISA để cho biêt trạng thai đợi cho đên khi hoan thanh chu ky`bus. Bây giơ`ta co´ thể tưởng tượng răng nêú một qua´trình đọc hoặc ghi EPP được băt đâù nêú như không co´thiêt bị ngoại vi naò nôí vaò thi`se~ra sao? Cổng se~không bao giơ` nhận được một tín hiệu xac nhận (nWait) vi`thê´ma`để co´được một yêu câù cho trạng thai đợi, may tính phải thực hiện một vong lặp kiểm tra. , do đó nó duy trì việc gửi tín hiệu yêu cầu và chờ kết thúc trạng thái “wait”, và máy tính sẽ bị treo. Vi`vậy ma`EPP thực hiện một kiểu kiểm tra watchdog ma`thơì gian time out la`xâp xỉ 10uS.

Ba thanh ghi: Base+5, Base+6 va`Base+7 co´thể được sử dụng cho cac thao tac đọc/ghi 32 bits dư~liệu nêú như cổng co´hô~trợ cho no´. Điêù nay co´thể lam giảm cac thao tac vaò/ra của ta. Cổng song song co´thể chi truyên dư~liệu 8 bits tại một thơì điểm cho nên bât ky`một word 16 hay 32 bits được ghi tơí cổng song song se~được chia thanh cac byte va`được gửi qua 8 bits (đương) dư~liệu của cổng song song.

Lập trình cổng máy in trong chế độ EPP.

EPP chỉ co´2 thanh ghi chinh va`một cơ`trạng thai time-out, chung ta co´ thể thiêt lập chung nhưng gi?

Trươc khi ta co´thể băt đâù bât ky`một chu ky`EPP băng việc đọc va`ghi tơí thanh ghi dư~liệu va`thanh ghi địa chỉ thi`cổng phải được câu hinh một cach đung đăn cho chê´độ lam việc của no´ trong trạng thai tự do, cổng EPP cân phải co´cac tín hiệu nAddress Strobe, nData Strobe, nWrite va`nReset ở trạng thai không tich cực (ở mưc cao - high level). Một vai cổng yêu câù ta phải thiêt lập cac tín hiệu nay trươc khi thực hiện một chu ky`bus EPP. Vi`vậy nhiệm vụ đâù tiên của chung ta la`khởi tạo một cach thủ công cac tín hiệu nay băng việc sử dụng cac thanh ghi của SPP. Cụ thể la`ghi gia´trị xxxx 0100 tơí thanh ghi điêù khiển để khởi tạo.

Trên một vai card, nêú cổng song song được đặt trong chê´độ ngược lại, thi`một chu ky`ghi EPP se~không thể thực hiện được. Vi`vậy no´se~tự biêt phải đặt cổng vaò chê´độ hợp lệ trươc khi sử dụng EPP. Xoa´bits 5 của thanh ghi điêù khiển co´thể lam cho việc lập trình trở nên thu´vị hơn ma`không lam pha´ vơ~sự phat triển chương trình.

Bit time-out của EPP The EPP: Khi bit nay được lập, cổng EPP co´thể không đảm bảo đung chưc năng của no´. Một sự kiện chung la`luôn luôn đọc gia´ trị 0FFh tư`cả chu ky`địa chỉ va`chu ky`dư~liệu. Bit nay nên được xoa´để cac thao tac được tin cậy va`no´phải luôn được kiểm tra.

B. CÂU HỎI VÀ BÀI TẬP

Câu 1: Trình bày cấu trúc cổng song song?

Câu 2: Trình bày các giao tiếp với máy tính và thiết bị ngoại vi của cổng song song?

Câu 3: So sánh sự khác nhau của truyền dự liệu qua cổng song song và đồng bộ?

Câu 4: Viết chương trình đọc thông tin địa chỉ của các cổng máy in có trong máy tính?

Hướng dẫn viết đoạn mã bằng ngôn ngữ C:

```
include <stdio.h>
#include <dos.h>
void main(void)
{
 unsigned int far *ptraddr;  /* Pointer to location of Port Addresses
*/
 unsigned int address;       /* Address of Port */
 int a;
 ptraddr=(unsigned int far *)0x00000408;
 for (a = 0; a < 3; a++)
   {
    address = *ptraddr;
    if (address == 0)
          printf("No port found for LPT%d \n",a+1);
    else
          printf("Address assigned to LPT%d is
%Xh\n",a+1,address);
     *ptraddr++;
    }
}
```

Câu 5: Viết chương trình hẹn giờ cho 3 bóng đèn 220V-40W riêng lẻ?

Hướng dẫn viết đoạn mã bằng ngôn ngữ C:

```
#include<stdio.h>
#include<conio.h>
#include<time.h>
#include<dos.h>
#define a 0x80
#define b 0x40
#define c 0x20
void main()
```

```
    {
    clrscr();
    char
    time[10],ngay[10],d1b[6],d2b[6],d3b[6],d1t[6],d2t[6],d3t[6],dt[6],ds[6];
    int n; long b1,t1,b2,t2,b3,t3;
    _strdate(ngay);
    printf("\n\tChuong trinh dieu khien bat tat bong den 22V 40W\t %s",ngay);
    printf("\n\t-----------------------------------------------");
    printf("\n\t1.Duong Vu Kien\t\t4.Nguyen Van Hien");
    printf("\n\t2.Vu Thuy Hang\t\t5.Phung Thi Bich Lien");
    printf("\n\t3.nong Le Thuy\t\t6.Pham Thanh Thong");
    printf("\n\t-----------------------------------------------");
    printf("\n\t\tBan hay lua chon phuong thuc hen gio:\n");
    printf("\n\t1.Hen gio theo dong ho he thong");
    printf("\n\t2.Hen gio theo thoi gian dinh san");
    printf("\n\t-----------------------------------------------");
    printf("\n");
    scanf("%d",&n);
if(n==1)
    {goto gioht;}
    if(n==2)
    {goto giobt;}
    else goto exit;
    gioht:
    {
    printf("Ban hay nhap vao thoi gian:");
    printf("\nBat den 1:");
    fflush(stdin);
    gets(d1b);
    printf("\tTat den 1:");
    gets(d1t);
    printf("Bat den 2:");
    gets(d2b);
    printf("\tTat den 2:");
    gets(d2t);
    printf("Bat den 3:");
    gets(d3b);
    // printf("Bat ca 3 den:");
    // gets(ds);
    printf("Tat den 3:");
    gets(d3t);
    do
```

```
    {
    gotoxy(50,23);
    _strtime(time);
printf("Gio hien tai:%s",time);
    if((time[3]==d1b[0])&(time[4]==d1b[1])&(time[6]==d1b[3])&(time[7]==d1
b[4])
    )
    {
    outportb(0x378,(inportb(0x378))|(0x80));
    }
    if((time[3]==d1t[0])&(time[4]==d1t[1])&(time[6]==d1t[3])&(time[7]==d1t[
4]))
    {
    outportb(0x378,(inportb(0x378))&(0x7f));
    }
    if((time[3]==d2b[0])&(time[4]==d2b[1])&(time[6]==d2b[3])&(time[7]==d2
b[4])
    )
    {
    outportb(0x378,(inportb(0x378))|(0x40));
    }
    if((time[3]==d2t[0])&(time[4]==d2t[1])&(time[6]==d2t[3])&(time[7]==d2t[
4]))
    {
    outportb(0x378,(inportb(0x378))&(0xbf));
    }
    if((time[3]==d3b[0])&(time[4]==d3b[1])&(time[6]==d3b[3])&(time[7]==d3
b[4])
    )
    {
    outportb(0x378,(inportb(0x378))|(0x20));
}
    f((time[3]==d3t[0])&(time[4]==d3t[1])&(time[6]==d3t[3])&(time[7]==d3t[4
]))
    {
    outportb(0x378,(inportb(0x378))&(0xdf));
    }
    if((time[3]==dt[0])&(time[4]==dt[1])&(time[6]==dt[3])&(time[7]==dt[4]))
    {
    outportb(0x378,0);
    }
    if((time[3]==ds[0])&(time[4]==ds[1])&(time[6]==ds[3])&(time[7]==ds[4]))
```

```
    {
    outportb(0x378,0xff);
    }
    }
    while(!kbhit());
    goto exit2;
    }
    giobt:
    {
    printf("\nBan hay nhap vao thoi gian den 1 sang:");
    scanf("%ld",&b1);
    // printf("\nBan hay nhap vao thoi gian den 1 tat:");
    // scanf("%ld",&t1);
    printf("\nBan hay nhap vao thoi gian den 2 sang:");
    scanf("%ld",&b2);
    // printf("\nBan hay nhap vao thoi gian den 2 tat:");
    // scanf("%ld",&t2);
printf("\nBan hay nhap vao thoi gian den 3 sang:");
    scanf("%ld",&b3);
    printf("\nBan hay nhap vao thoi gian den tat:");
    scanf("%ld",&t3);
    printf("\nBat dau chuong trinh hen gio.Bam phim bat ki de ket thuc
    chuong trinh");
    do
    {
    outportb(0x378,a);
    delay(b1);
    // outportb(0x378,!a);
    // delay(t1);
    outportb(0x378,b);
    delay(b2);
    // outportb(0x378,!b);
    // delay(t2);
    outportb(0x378,c);
    delay(b3);
    outportb(0x378,0);
    delay(t3);
    }
    while(!kbhit());
    goto exit2;
    }
    exit:
```

```
    clrscr();
    printf("\nKhong co lua chon thu %d. Bam phim bat ki de ket thuc
    chuong trinh !!",n);
    outportb(0x378,0);
    getch();
    exit2:
clrscr();
    printf("\nChuong trinh hen gio ket thuc !!!");
    outportb(0x378,0);
    getch();
    getch();
}
```

BÀI 5: LẬP TRÌNH QUA CÁC MẠCH GHÉP NỐI ĐA NĂNG

Mã bài: MĐ41-06

❖ **Giới thiệu**

Mặc dù số lượng các đường dẫn được sử dụng ở giao diện nối tiếp bị hạn chế, kỹ thuật ghép nối máy tính của giao diện nối tiếp đã được đề cập tương đối kỹ trong chương trước đã học, có thể đem phối hợp các ứng dụng với nhau để tạo ra một giao diện (khối ghép nối đa năng) dùng cho những ứng dụng mới khác nhau

❖ **Mục tiêu**

- Xây dựng Phần cứng và cách điều khiển
- Thiết lập chương trình đo lường
- Kiểm tra hoạt động của các vi mạch.
- Tự tin trong lập trình ghép nối máy tính

❖ **Nội dung chính**

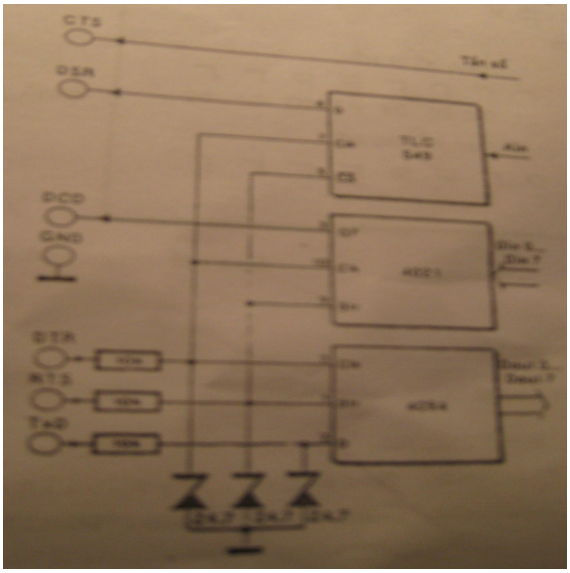
A. LÝ THUYẾT

1. Xây dựng Phần cứng và cách điều khiển

Mục tiêu:

- Viết được đoạn chương trình cơ sở dùng cho giao diện đa năng nối tiếp

Hình 6.1 chỉ ra sơ đồ khối của một giao diện với tám lối ra số, tám lối vào số, một lối vào analog và một lối và đếm. Toàn bộ ba thanh ghi dịch dùng để nhập vào và xuất ra có thể được điều khiển bằng một đường dẫn giữ nhịp và đồng bộ (strobe) chung.



Hình 6.1: Một giao diện nối tiếp đa năng

Việc nhập vào và xuất ra có thể được thực hiện đồng thời, bởi vì một đường dẫn giữ nhịp chung được sử dụng cho tất cả các vi mạch. Thủ tục "Trao đổi" trong đoạn chương trình 6.1 đảm nhận việc truyền dữ liệu theo hai hướng. Mỗi lời gọi lại phục hồi trạng thái lối ra của vi mạch 4094 và đồng thời đọc ra các trạng thái của các lối vào số và bộ biến đổi A/D. Tất cả dữ liệu sắp xếp trong các biến toàn cục Dout (lối ra sô), Din (lối vào số) và Ain (lối vào analog)

Ngoài ra, lối vào tự do CTS được sử dụng để đo tần số. Ở đây thời gian mở cổng được thiết lập cố định là 100 ms, vì vậy có thể nhận được độ phân giải về tần số bằng 10 Hz. Khi ta đấu vào trước đó một bộ biến đổi điện áp/ Tần số, thì giao diện có thể được sử dụng với hai lối vào analog.

Đoạn chương trình 8-1: Chương trình cơ sở dùng cho giao diện đa năng nối tiếp

```
Unit Unil
Interface
Uses PORTINC
    Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
Dialogs, ExtCtrls, StdCtrls;
Type
    Tform1 = class (TForm)
        CheckBox1 : TcheckBox;
        CheckBox2 : TcheckBox;
    CheckBox3 : TcheckBox;
    CheckBox4 : TcheckBox;
    CheckBox5 : TcheckBox;
```

```
        CheckBox6 : TcheckBox;
        CheckBox7 : TcheckBox;
CheckBox8 : TcheckBox;
            Labell : TLabel;
        Label2 : TLabel;
        Label3: TLabel;
        Label4 : TLabel;
        Label5 : TLabel;
        Label6 : TLabel;
        Label7 : TLabel;
        Label8 : TLabel;
        Timer1 : TTimer;
        Label9 : TLabel;
        CheckBox10 : TCheckBox;
        CheckBox11 : TCheckBox;
        CheckBox12 : TCheckBox;
        CheckBox13 : TCheckBox;
        CheckBox14 : TCheckBox;
        CheckBox15 : TCheckBox;
        CheckBox16 : TCheckBox;
        Label10 : TLabel;
        Edit1 : TEdit;
        Label11 : TLabel;
        Edit2 : TEdit;
        Label12 : TLabel;
        Procedure FromCreate(Sender : TObject);
        Procedure Timer1TTimer(Sender : TObject);
        End;
        Var
            Form1: TForm1;
            Dout, Din, Ain: Byte;
        Implementation
        {$R *.DFM}
        Procedure Trao_doi;
        Var Vi_tri, Vi_triAD, r, m : Integer;
        Begin
            RTS(1);
            Delayus(20);
            RTS(0);
            Vi_tri := 1;
            Vi_triAD := 128;
            Din := 0;
```

```
        Alin := 0;
        For n:=1 to  8 do begin
                If ((Dout AND Vi_tri) > 0) then
                TXD (1)              {xuất ra dữ liệu}
                Else TXD (0);
                If DCD = 1           { đọc dữ liệu}
                        Then Din := Din + Vi_tri;
                If DSR = 1 then      {đọc A/D}
                        Alin := Alin + Vi_triAD;
                DTR (1);             {Clock tới}
                Delayus (20);        {Gây trễ}
                DTR (0);             {Clock tắt}
                Vi_tri := Vi_tri *2;
                Vi_triAD :=Vi_triAD div 2;
                End;
                RTS(1);              {Strobe tới}
                Delayus(20)          {Gây trễ}
                RTS(0);              {Strobe}
        End;
Function Tan_so:Integer;
Var Bo_dem, Cu, Nhap_vao:Integer;          Cu:Cũ
Begin
        Bo_dem:=0;
        ReCuime     (true);
        TimeInitus;
        Cu:=CTS;
        While TimeReadus<(100000) do begin
                Nhap_vao:=CTS;
                If Nhap_vao>Cu then Bo_dem:=Bo_dem +1;
                Cu:=Nhap_vao;
        End;
        Tan_so:=Bo_dem;
        ReCuime (false);
End;
Procedure TForm1.FormCreate (Sender: TObject);
Begin
        OpenCom (pchar (‘com2:9600, N, 8, 1’))
        Timer1.Interval :=100;
        Timer1.Enabled :=true;
End;
Procedure TForm1.Timer1Timer(Sender: TObject);
Var Gia_tri: Word;
```

```
        Dien_ap: Real;
Begin
        Dout :=0;
        If CheckBox1.Checked Then Dout := Dout +1;
        If CheckBox2.Checked Then Dout := Dout +2;
        If CheckBox3.Checked Then Dout := Dout +4;
If CheckBox4.Checked Then Dout := Dout +8;
If CheckBox5.Checked Then Dout := Dout +16;
If CheckBox6.Checked Then Dout := Dout +32;
If CheckBox7.Checked Then Dout := Dout +64;
If CheckBox8.Checked Then Dout := Dout +128;
Trao_doi;
CheckBox9.Checked := ((Din And 1)>0);
CheckBox10.Checked := ((Din And 2)>0);
CheckBox11.Checked := ((Din And 4)>0);
CheckBox12.Checked := ((Din And 8)>0);
CheckBox13.Checked := ((Din And 16)>0);
CheckBox14.Checked := ((Din And 32)>0);
CheckBox15.Checked := ((Din And 64)>0);
CheckBox16.Checked := ((Din And 128)>0);
Dien_ap :=Alin/255*5;
Edit1.Text :=FloatToStrF (Dien_ap, ffGeneral,3,2) +' V';
Edit2.Text :=FloatToStrF (Tan_so*10, ffGeneral,4,0) +' Hz';
End;
End;
```

2. Thiết lập chương trình đo lường

Mục tiêu:

- Biết được các phần mềm sử dụng trong phòng thí nghiệm điện tử
- Sử dụng được phần mềm COMPACT vào đo lường các thiết bị

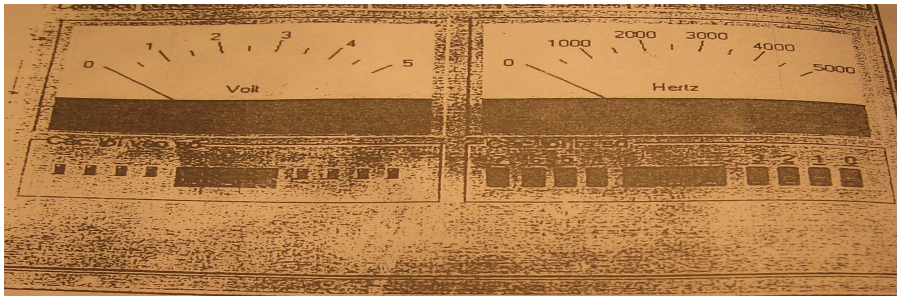
Để có được những phần mềm sử dụng trong phòng thí nghiệm điện tử như một công cụ, nhiều công ty đã viết ra các phần mềm mô phỏng hoặc điều hành thực sự một mạch ghép nối và giới thiệu tính năng trên nhiều tạp chí chuyên ngành. Có thể kể ra hàng loạt các phần mềm quen thuộc như: Daisy, DASYLab, DaqView, DIAdem, ChartView, LogView, WaveView, v… v…Việc mua trực tiếp các phần mềm loại này thường tốn kém về kinh phí. Nhưng việc tham khảo tính năng cũng như giao diện của phần mềm có thể cho ta những thông tin rất bổ ích khi đánh giá cũng như khi trực tiếp viết ra các phần mềm ghép nối

Dưới đây xin giới thiệu phần mềm COMPACT, được viết bằng ngôn ngữ Delphi. Bằng phần mềm COMPACT, tất cả các lối vào/ra của khối ghép

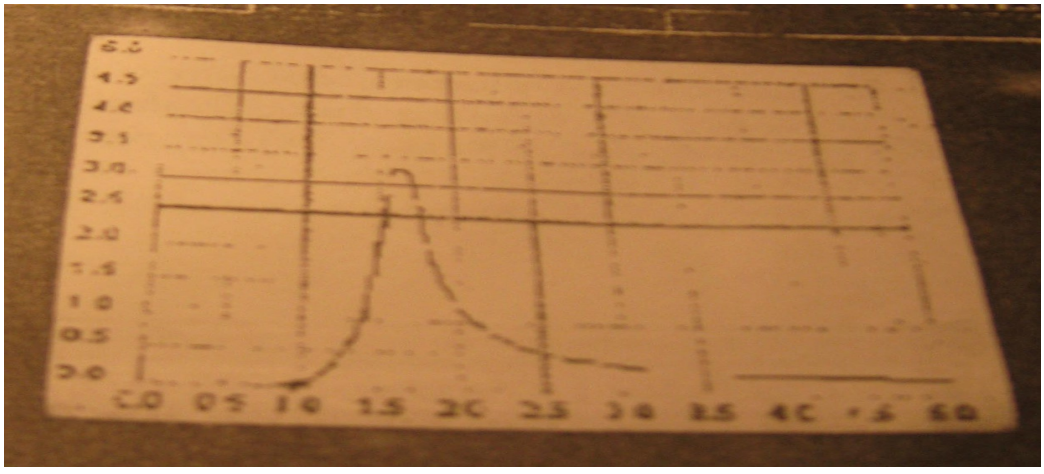
nối đa năng đều được điều khiển và cho phép thu thập các số liệu đo lường một cách linh hoạt. Người sử dụng có thể lựa chọn cổng COM khi khởi động chương trình.

Sau khi khởi động chương trình, trên màn hình hiện lên một thực đơn với nhiều chức năng. Trong thực đơn con "Overview" ta nhận được khả năng truy nhập trực tiếp lên tất cả các lối vào và lối ra. Cả hai lối vào analog dùng cho điện áp và tần số đều được mô tả trên các dụng cụ chỉ kim. Ngoài ra, trong trường hợp đo điện áp cũng xuất hiện một bộ hiển thị số, chỉ ra số byte được bộ biến đổi A/D cung cấp. Cả hai máy đo đều có thể sử dụng trực tiếp cho các phép thử và tìm kiếm lỗi

Ngoài ra các lối vào analog, ta cũng có được khả năng truy cập lên các lối vào/ra số của giao diện đa năng. Tám lối vào số đều được hiển thị bằng các LED ảo (vì chỉ là ảnh của một LED trên màn hình) và đồng thời qua bộ hiển thị số như là byte giữa 0 và 255. Ta cũng có thể giám sát trực tiếp các trạng thái của một mạch điện số. Các đường dẫn lối ra số được thay đổi bằng các phím gạt ảo. Trong một bộ chỉ thị số ta quan sát trạng thái của một mạch điện số. Các đường dẫn lối ra số được thay đổi bằng các phím gạt ảo. Trong một bộ chỉ thị số ta quan sát thấy trạng thái chung của cổng



Hình 6.4: Quan sát trực tiếp tất cả các lối vào và lối ra.



Hình 6.6: Máy tự ghi xy.

Chức năng 'overview" về cơ bản là tương ứng với chưng trình UNI1.EXE đã được giới thiệu trước đây . trong khi các khản năng bổ sung thêm được thể hiện ra các thiết bị ghi khác nhau của chương trình.

Thiết bị ghi TY vẽ ra môt hoặc hai đại lượng lôi vào analog theo thời gian. Nhờ vậy mà việc quan sat sự thay đổi theo thời gian của các đại lượng này trở lên đơn giản hơn. Dải đo có thể lữa chon được môt trong giới hạn rộng từ một giây đến 24h.

Đa số các đại lượng analong đều có sự phù thuộc qua lai.vì thế việc sử dụng một máy ghi XY tỏ ra là đặc biệt thuận tiện trong phần mềm COMPACT, điện áp được biểu diễn theo tần số.cách mô tả này đặc biệt thích hợp đối với các phép đo theo sự thay đổi của tần số.Khi đó tín hiệu từ một bộ tạo hàm(máy phát chức năng) điều khiển đồng thời lối vào tần số của giao diện đa năng và lối vào của một mạch cần khảo sát.tín hiệu lối ra của mạch kiểm tra được đưa đến lối vào analog của khối ghép nối của một bộ chỉnh lưu dùng trong đo lường. dãi tần số đáng quan tâm có thể được quét bằng tay, để vẽ ra quá trình diễn biến khi thay đổi tần số.

Thiết bị ghi theo bit (bit-plotter), đoi khi còn gọi là bộ phận tích logic, tỏ ra là thích hợp với các ứng dụng thuần tuy số. cho đến 8 trạng thái số có thể được quan sát trực tiếp trong một khỏng thời gian lựa chon được. Thiết bị này ở đây là thiết bị ảo vì được tạo ra bằng một hàm hocawcj một số hàm. Có khả năng hổ trợ chẳng hạn cho việc kiểm tra lỗi trong các mạch số.

Một chức năng thuận tiện khác của chương trình là bộ định thời (Timer). Bộ định thời cho phép đo chính xác thời gian ở các xung, được đưa đến lối vào CTS. Ở đây cũng có thể đấu vào một công tắc kiểu nhấn phím lên nguồn +5 V.

3. Kiểm tra hoạt động của các vi mạch

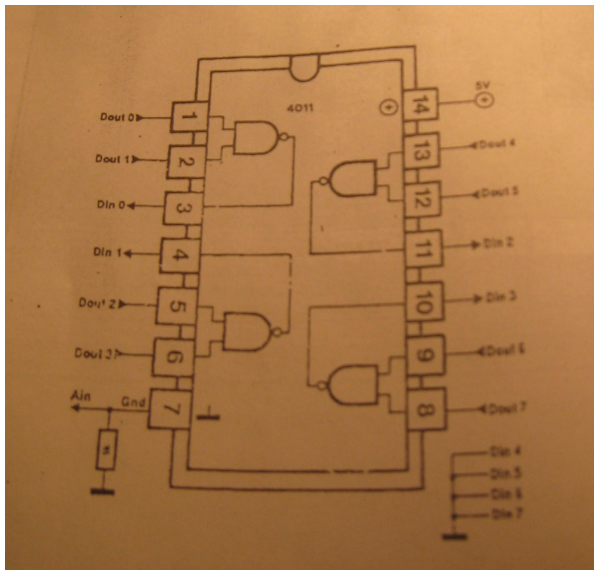
Mục tiêu:

- Viết được chương trình kiểm tra dùng cho một vi mạch CMOS loại 4011

Khối ghép nối đa năng có đủ các lối vào và lối ra để có thể kiểm tra các khả năng làm việc bình thường của các vi mạch số đơn giản. Đặc tính này có thể được minh họa thông qua thí dụ về vi mạch NAND kép bốn, loại 4011 (xem hình 6.9). Nguyên tắc kiểm tra là so sánh bảng chân lý của vi mạch với các trạng thái đo được trên vi mạch. Cụ thể là: qua khối ghép nối, máy tính PC tạo ra các trạng thái lối vào xác định và đọc các trạng thái lối ra tương ứng của vi mạch. Ngoài ra, dòng tiêu thụ của vi mạch thử cũng được đo để kiểm

tra, bởi vì các vi mạch CMOS hỏng thường có dòng tiêu thụ lớn hơn bình thường.

Đoạn chương trình 6.2 chỉ ra một thí dụ về một chương trình có thể được dùng để kiểm tra. Tám lối ra số của khối ghép nối (giao diện) được đếm tiến theo hệ nhị phân. Mỗi một trong số 256 trạng thái lối vào có thể của vi mạch cần kiểm tra đều được tạo ra một lần, để kiểm tra trạng thái đáp ứng ở các lối ra vi mạch. Thủ tục "Trao đổi" cần được gọi theo cách lặp lại hai lần kế tiếp nhau, bởi vì trong lần gọi đầu tiên câu trả lời đọc được là ứng với trạng thái vừa có trước đấy.



Hình 6.9: Kiểm tra một vi mạch CMOS bằng giao diện đa năng

Những phép tính tốn kém thời gian để suy ra trạng thái đứng của lối ra có thể được né tránh, khi ta làm việc với một bảng so sánh. Bảng này được tạo ta bằng cách nhấn phím "New" khi một vi mạch được đưa vào làm đối tượng kiểm tra. Một bảng với đầy đủ các dữ liệu cần thiết có thể được lưu trữ và nạp giống như một tệp nhị phân, chẳng hạn với tên là CD4011.BIN. Phương pháp kiểm tra này có thể dễ dàng mở rộng ra cho các vi mạch khác, trong đó ta sắp xếp các tệp dữ liệu cụ thể thích hợp với vi mạch đó. Với một đế cắm vi mạch thích hợp ta có thể kiểm tra các vi mạch thuộc những loại khác nhau. Mỗi sai lệch khỏi diễn biến bình thường được nhận biết đều dẫn đến một thông báo lỗi. Trong số đó phải kể cả trường hợp dòng tiêu thụ vượt quá 0,2 mA.

B. CÂU HỎI VÀ BÀI TẬP

Câu 1: Trình bày cách xây dựng phần cứng và điều khiển?

Câu 2: Nêu phương pháp thiết lập chương trình đo lường?

Câu 3: Nêu phương pháp kiểm tra hoạt động của các vi mạch?
Câu 4: Viết chương trình điều khiển giao diện đa năng?
Câu 5: Viết chương trình kiểm tra dùng cho một vi mạch CMOS loại 4011?

Hướng dẫn viết đoạn mã:

Chương trình kiểm tra dùng cho một vi mạch CMOS loại 4011.

```
Unit ICtest;
Interface
Uses PORTNC;
        Windows, Messages, SysUtils, Classes, Graphics,
        Controls, Forms, Dialogs, Extctrls, StdCtrls, Menus;
Type
        TForm1 = class (TForm)
        Edit1 : TEdit;
        Button1 : TButton;
        Button2 : TButton;
        Timer1 :TTimer;
        OpenDialog1 : TOpenDialog;
        SaveDialog1 : TSaveDialog;
        MainMenu1 : TMainMenu;
        File1 : TMenuItem;
        SaveAs1 : TMenuItem;
        Open1 : TmenuItem;
        New1 : TmenuItem;
        Procedure FormCreate (Sender : Tobject);
        Procedure Button3click (Sender : Tobject);
        Procedure Button2click (Sender : Tobject);
        Procedure Button1click (Sender : Tobject);
        Procedure Timer1Timer (Sender : Tobject);
        Procedure Open1Click (Sender : Tobject);
        Procedure SaveAsClick (Sender : Tobject);
End;
Var
        Form1 : Tform1;
        Dout, Din, Alin: byte;
        n: integer;
        Lci_ra : Array [0..255] of Byte;
        Testfile : File of byte;
        Ten_tep : string;
Implementation
{SR *.DFM}
Procedure Trao_doi;
Var Vi_tri, Vi_triAD, n, m : Integer;
```

```
Begin
        RTS (1);                        { Strobe an}
        Delayus (20);                   { Gây trễ }
        RTS (0);                        { Strobe cắt}
        Vi_tri := 1;
        Vi_triAD := 129;
        Din := 0;
        Ain := 0;
        For n:=1 to 8 do begin
                If ((Dout AND Vi_tri)>0) then
                        TXD (1)         {xuất ra dữ liệu}
                        Else TXD (0);
                If DCD = 1              { đọc dữ liệu}
                        Then Din := Din + Vi_tri;
                If DSR = 1 then         { đọc A/D}
                        Ain := Ain + Vi_triAD;
                DTR (1);                        {Clock tới}
                Delayus (20);                   { Gây trễ}
                DTR (0);                        { Clock tắt}
                Vi_tri := Vi_tri * 2;
                Vi_triAD := Vi_triAD div 2;
        End;
        RTS (1);                        {Clock tới}
        Delayus (20);                   { Gây trễ}
        RTS (0);                        { Clock tắt}

End;
Procedure Tform1.Formcreate(Sender:Tobject);
Begin
        Opencom(pchar('com2:9600, n, 2, 8, 1'));
End;
Proceduer Tform1.Button3click(sender : Tobject);
Var n: integer;
Begin
(Ten_tep := OpenDialog.FileName;}
Ten_tep := 'TEST.BIN';
AssignFile (Testfile, Ten_tep);
{$I-} Rewrite (Testfile) {$I+};
For n:=0 to 255 do bein;
        Write (Testfile, Loi_ra[n]);
End;
Closefile (Testfile);
```

```
End;
Procedure Tform1.Buttom2Click(Sender:Tobject);
Var Gia_tri: Word;
        Dien_ap, Fehler: Real;
Begin
Timer1.Enable := false;
Fehler := 0;
Edit1.Text := 'Text ...';
For n:=1 to 255 do begin
        Dout := n;
        Trao_doi;
        Delay (1);
        Trao_doi;
        If (Ain>10) or (Din <> Loi_ra[n]) then begin
                Fehler := Fehler + 1;
                Dien_ap := Ain/255*5;
                Edit1.Text := FloatToStr(N) + ',  '  +
                floatToStr(Din) + '<>' + FloatToStr (Loi_ra[n])
                +'  '+ FloatToStrF(Dien_ap, ffGeneral, 3, 2) + 'mA';
        End;
End;
If Fehler = 0 then Edit1.Text := 'Test O.K';
Timer1.Interval := 2000;
Timer1.Enabled := true;
End;
Procedure Tform1.ButtonClick(Sender : Tobject);
Var n:integer;
Begin
{Ten_tep := OpenDialog.FileName;}
Timer1.Enable := false;
For n:=0 to 255 do begin;
        Dout := n;
        Trao_doi;
        Delay (1);
        Trao_doi;
        Loi_ra[n] := Din;
End;
Edit1.text:= 'Fertig';
Timer1.Enable := true;
End;
Procedure Tform1.Timer1Timer(Sender: Tobject);
Begin
```

```
        Edit1.Text := 'bereit';
        Timer1.Enabled := false
End;
Procedure Tform1.Open1Click(Sender:Tobject);
Begin
        OpenDialog1.FileName := '*.bin';
        OpenDialog1.Execute;
        Ten_tep := OpenDialog1.FileName;
        If Ten_tep > ' ' then begin
                AssignFile (Testfile, Ten_tep);
                {$I-} Reset (Testfile) {$I+};
                For n:=0 to 255 do read (Testfile, Loi_ra[n]);
                Closefile (Testfile);
        End;
End;
Procedure Tform1.SaveAslClick(Sender : Tobject);
Var n:integer;
Begin
        SaveDialog1.FileName := '*.bin';
        OpenDialog1.Execute;
        If Ten_tep > ' ' then begin
                AssignFile (Testfile, Ten_tep);
                {$I-} Rewrite (Testfile) {$I+};
                For n:=0 to 255 do write (Testfile, Loi_ra[n]);
                Closefile (Testfile);
        End;
        End; End.
```

TÀI LIỆU THAM KHẢO

1. Ngô Diên Tập. *Kỹ thuật ghép nối máy tính.* Nhà xuất bản: Khoa học và Kỹ thuật, năm 2005.

2. Ngô Diên Tập. *Lập Trình Ghép Nối Máy Tính Trong Windows*. Nhà xuất bản: Khoa học và Kỹ thuật, năm 2005.

3. Trần Quang Vinh. *Nguyên lý phần cứng và kỹ thuật ghép nối máy tính*. NXB Giáo dục

4. Phạm Hữu Khang. *Giáo trình nhập môn lập trình VB6*. NXB Lao động - Xã hội, năm 2005

5. Phạm Hữu Khang. *Kỹ xảo lập trình VB6*. NXB Lao động - Xã hội, năm 2004