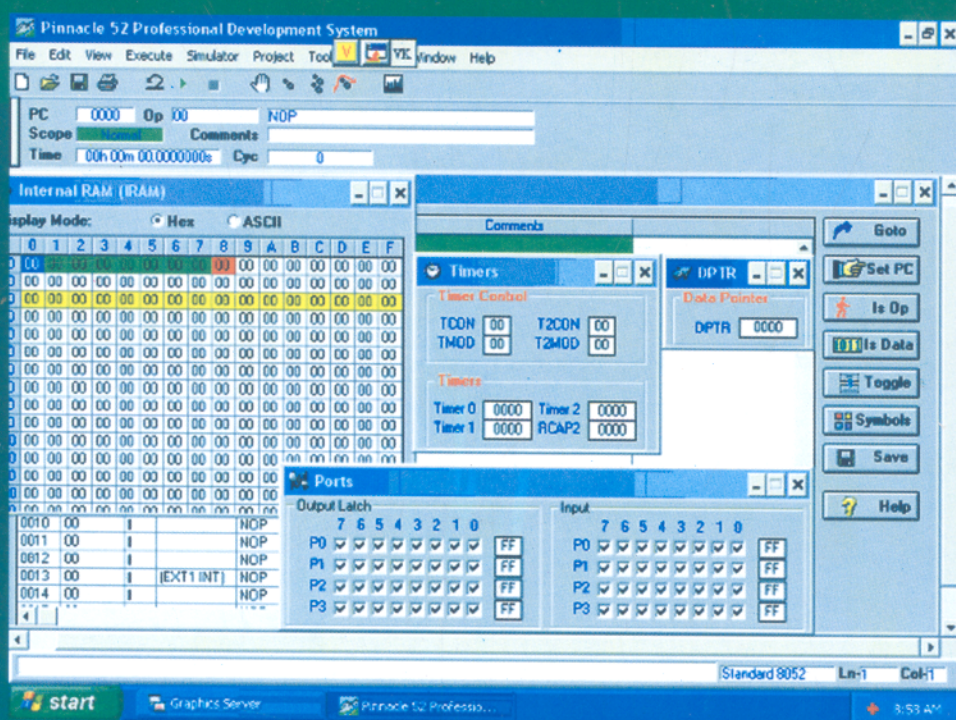


TS. NGUYỄN MẠNH GIANG

CẤU TRÚC-LẬP TRÌNH-GHÉP NỐI VÀ ỨNG DỤNG VI ĐIỀU KHIỂN

TẬP HAI
GHÉP NỐI VÀ ỨNG DỤNG VĐK 8051/8052



NHÀ XUẤT BẢN GIÁO DỤC

TS. NGUYỄN MẠNH GIANG

CẤU TRÚC - LẬP TRÌNH - GHÉP NỐI VÀ ỨNG DỤNG VI ĐIỀU KHIỂN

TẬP HAI
GHÉP NỐI VÀ ỨNG DỤNG VDK 8051/8052

NHÀ XUẤT BẢN GIÁO DỤC

LỜI NÓI ĐẦU

Đây là tập hai của bộ sách “Cấu trúc, lập trình, ghép nối và ứng dụng vi điều khiển”. Tài liệu này giới thiệu về ghép nối với các thiết bị ngoài (hiển thị, bàn phím, nhớ ngoài, cổng mở rộng (song song và nối tiếp)) và các hệ VDK dùng trong đo lường, điều khiển tự động và truyền thông.

Tiếp theo tập một, tập hai gồm 8 chương. Trong mỗi chương, đều có mô tả các thiết bị ngoài, cách ghép nối và lập trình với các ví dụ để chạy thử mô phỏng trên PINNACLE và PROTEUS.

Chương 11 Ghép nối và ứng dụng đèn LED.

Chương 12 Ghép nối và ứng dụng đèn LED ma trận.

Chương 13 Ghép nối với màn hình tinh thể lỏng (LCD).

Chương 14 Ghép nối với bàn phím.

Chương 15 Ghép nối với bộ nhớ ngoài của vi điều khiển 8051/8052.

Chương 16 Ghép nối mở rộng cổng song song cho vi điều khiển.

Chương 17 Ghép nối mở rộng cổng nối tiếp cho vi điều khiển.

Chương 18 Ghép nối với hệ vi điều khiển.

Trong tài liệu này cũng giới thiệu cách dùng phần mềm Proteus trong thiết kế như vẽ sơ đồ nguyên lý của hệ VDK, trong chạy mô phỏng các ví dụ với tệp đuôi .HEX sau khi hợp dịch và chạy mô phỏng (từng bước và cả chương trình) trên Proteus.

Tài liệu được biên soạn từ cơ bản tới chuyên sâu, cho người tự học, công nhân học nghề, sinh viên cao đẳng, đại học, cao học và nhà nghiên cứu trong các lĩnh vực tự động hoá, điện tử, vật lý kỹ thuật, cơ tin và thông tin.

Tùy trình độ và mục đích sử dụng, người đọc có thể bỏ qua hay chỉ chú ý tới phần mình quan tâm. Vì trong thời đại thông tin, mọi người có thể truy cập mạng Internet, nên cách trình bày chỉ tóm tắt ý chính và cơ bản, người đọc có thể tra cứu thêm về các vi mạch và ứng dụng trên mạng Internet.

Tác giả xin chân thành cảm ơn các Ban lãnh đạo, các đồng nghiệp ở Trường Đại học Bách Khoa Hà Nội, Đại học Sư phạm Kỹ thuật Nam Định, Viện Nghiên cứu Kỹ thuật hạt nhân Hà Nội, nơi tác giả đã công tác và cộng tác về sự khích lệ, cung cấp tài liệu và trao đổi chuyên môn.

Tác giả xin chân thành cảm ơn Ban Giám đốc, Ban lãnh đạo Công ty Cổ phần Sách Đại học - Dạy nghề của Nhà xuất bản Giáo dục đã giúp đỡ để tài liệu sớm ra mắt bạn đọc.

Tác giả mong được bạn đọc góp ý cho tài liệu được hoàn thiện hơn. Thư góp ý xin gửi về Công ty Cổ phần Sách Đại học - Dạy nghề (HEVOBCO), Nhà xuất bản Giáo dục, 25 Hàn Thuyên, Hà Nội, điện thoại 04 8.264974.

TÁC GIẢ

CHƯƠNG 11

GHÉP NỐI VÀ ỨNG DỤNG ĐÈN LED

Mục đích của chương:

- Hiểu được nguyên tắc hoạt động, cách mắc và lập trình cho các đèn hiển thị LED đơn.
- Hiểu được nguyên tắc hoạt động, cách mắc và lập trình cho các đèn hiển thị LED 7 đoạn.

Trong quá trình sử dụng thiết bị số, trong đó có máy vi tính, vi xử lý, vi điều khiển, chúng ta cần nhìn thấy được nội dung của dữ liệu (địa chỉ, dữ liệu, mã lệnh) dưới dạng các con số tự nhiên (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) hay các chữ cái (a, b, c, d,... z).

Có các loại hiển thị sau:

- *Đèn nóng sáng dây tóc thế nuôi thấp (5V) hay đèn diot phát quang*: khi có dòng điện nhỏ chạy qua (cỡ mA). Loại LED đơn này đơn giản, rẻ tiền, nhưng chỉ biểu diễn nội dung từng bit có giá trị 0 (tắt) hay 1 (sáng). Nếu nhiều bit, các đèn này biểu diễn thành các số nhị phân, khó hình dung thành các số thập phân.

- *Đèn diot (LED) 7 đoạn (thanh)*: (a, b, c, d, e, f, g) bố trí thành hình chữ số 8. Tùy các thanh sáng (có dòng điện chạy qua), ta có các chữ số hay chữ khác nhau. Có hai loại diot là:

+ *Anôt chung*: Các anôt nối với nhau và nối với nguồn nuôi sẽ phát sáng khi các katôt riêng rẽ của từng thanh có mức logic 0.

+ *Katôt chung*: Các katôt nối với nhau và nối với đất chung, sẽ phát sáng khi các anôt riêng rẽ của từng thanh có mức logic 1.

Đèn LED 7 thanh này hiển thị trực giác được các chữ số và các ký tự, nhưng có nhược điểm là phải giải mã (bằng vi mạch, ví dụ 7447 hay bằng chương trình) thành các con số dạng BCD (Binary Coded Decimal- số thập phân mã hoá thành dạng nhị phân).

- *Đèn LED nhiều đoạn*: Các đoạn được bố trí theo các đường ngang, dọc và chéo của 4 hình chữ nhật ghép lại. Loại này biểu diễn được chữ và con số một cách tinh tế hơn LED 7 đoạn nhưng đắt tiền và điều khiển phức tạp.

- *Đèn ma trận diot (ma trận LED)*: Gồm nhiều diot có kích thước rất nhỏ (coi như một điểm), sắp xếp thành các ma trận 5 x 7 theo 7 hàng và 5 cột. Cũng có các loại anôt chung hay katôt chung như đèn LED 7 đoạn. Tùy điều khiển các dòng điện chạy qua, các diot điểm này phát sáng, tạo thành các điểm của bức tranh của các con số thập phân hay các chữ a, b, c, d,... z. Đèn ma trận diot này biểu diễn đẹp các chữ hay số nhưng đắt tiền, phải xác định các mã của chữ hay số theo từng ma trận với các hàng và cột và đưa ra biểu diễn với tốc độ thích hợp với mắt người để quan sát.

- *Màn hình tinh thể lỏng LCD (Liquit Crystal Diot)*: gồm nhiều diot trong một khối tinh thể lỏng (thường có hai hàng, nhiều cột). LCD này có nguyên tắc sáng tối tương tự ma trận diot, có thể biểu diễn được các số, chữ và hình ảnh (graph). Loại LCD thông minh này có ưu điểm hơn LED 7 đoạn là biểu diễn các chữ giống chữ in, hình ảnh rõ nét, có độ phân giải cao.

Các LCD này được dùng rộng rãi trong các thiết bị số tiên tiến như tivi, đầu đĩa, đồng hồ, các thiết bị điện tử trong gia đình (lò sấy, bếp điện, lò nung,...).

Hiện nay, còn có loại màn hình tinh thể lỏng cỡ lớn (kích thước từ nhiều centimét tới mét), có nhiều hàng và cột, có thể biểu diễn được cả hình ảnh của một bức tranh lớn hay cả một thông báo gồm nhiều hàng chữ và số. Vì giá thành của loại LCD này còn đắt nên chưa thông dụng.

11.1. ĐÈN HIỂN THỊ LED RỜI

11.1.1. Cấu tạo và cách mắc

Đèn dây tóc nóng sáng và đèn diot phát sáng LED (Luminescence Emittet Diot) sẽ sáng khi có dòng điện đủ lớn (cỡ mA) chạy qua. Do đó, hai đèn này sáng dùng để chỉ thị:

- *Giá trị logic 1 (cỡ 3V–5V)*: Khi đèn sáng vì có dòng điện chạy qua.
- *Giá trị logic 0 (cỡ 0V–0,2V)*: Khi đèn tắt vì không có dòng điện chạy qua.

Đèn dây tóc nóng sáng có cấu tạo đơn giản nhưng tiêu tốn điện năng (dòng điện chạy qua lớn mới đủ sáng) nên ít dùng. Đèn diot dùng nhiều hơn vì tiêu tốn ít điện năng và có nhiều loại (hình 11.1) với màu sắc khác nhau (xanh, đỏ, tím, vàng...).

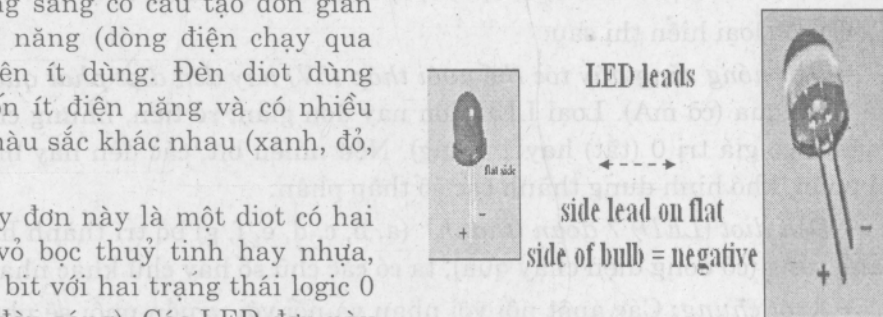
Đèn LED rời hay đơn này là một diot có hai cực, đặt trong một vỏ bọc thủy tinh hay nhựa, dùng để chỉ thị từng bit với hai trạng thái logic 0 (đèn tắt) và logic 1 (đèn sáng). Các LED đơn này được mắc với nguồn nuôi 5V qua một điện trở hạn chế dòng điện cỡ $1k\Omega - 3k\Omega$ hay qua một mạch logic đảo colectơ hở (ví dụ vi mạch 7403) để tăng công suất dòng điện qua diot. Khi đó, đèn sáng chỉ thị logic 0 và đèn tắt chỉ thị logic 1. Đèn có hai cực anôt và katôt nên mắc với nguồn điện qua một điện trở hạn chế dòng (cỡ hàng trăm tới hàng ngàn ôm).

Đèn LED này sử dụng đơn giản nhưng chỉ thị con số dạng nhị phân, người dùng khó hình dung ra con số dạng thập phân dưới dạng nhiều chữ số tự nhiên (0, 1, 2, 3, 4, 5, 6, 7, 8, 9).

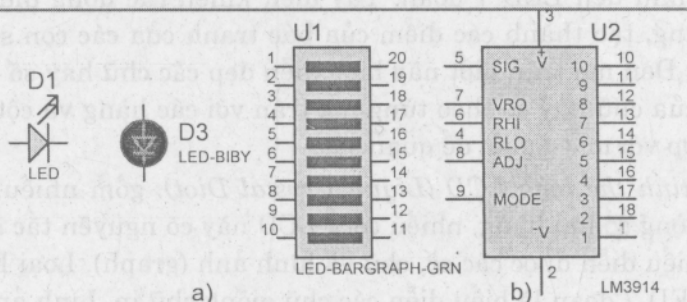
11.1.2. Các loại LED rời

Hiện có nhiều hãng điện tử chế tạo các loại đèn LED với kích cỡ, màu sắc và mục đích sử dụng khác nhau.

Trong chương trình mô phỏng PROTEUS, có mô phỏng một số loại LED như ở hình 11.2.



Hình 11.1. Một số loại đèn LED



Hình 11.2. Các loại LED đơn (a) và vi mạch điều khiển đèn công suất (b)

Hình 11.2 giới thiệu ba loại LED: loại phát quang, loại chỉ thị tròn và loại thanh. Vi mạch LN3914 có trên hình 11.2 là vi mạch điều khiển LED công suất.

11.1.3. Lập trình điều khiển LED rời

Nói chung, các đèn hiển thị LED rời có thể mắc với cổng P0, P1, P2 và P3 của vi điều khiển (VĐK) thuộc họ MCS – 51. Có thể mắc trực tiếp hay qua một điện trở hạn chế dòng điện hoặc một cổng logic để tăng công suất.

Ví dụ 11.1a giới thiệu chương trình đưa dữ liệu cố định ra các cổng Pi dùng lệnh gộp @WritePs < 1, 2 > với :

- Thông số 1 là giá trị của dữ liệu (dạng HEX) đưa vào thanh ghi chứa A.
- Thông số 2 là chỉ số của cổng VDK.

Phần sau của chương trình trên làm đèn LED sáng dần từ phải qua trái (dùng lệnh RLC A) cho tới khi C = 1. Có thể thay lệnh RLC A bằng lệnh RL A để dịch và lặp lại một số lần (thay bằng lệnh MOV R0, #N lần và DJNZ R0, LAP).

Ví dụ 11.1b là chương trình của ví dụ 11.1a dạng đuôi. LST sau khi hợp dịch trong PINNACLE.

Từ ví dụ trên, ta có thể viết chương trình cho LED đơn sáng dần dịch về bên phải (dùng lệnh RRC A) hay bên trái (dùng lệnh RLC A).

; Ví dụ 11.1a: Chương trình dùng lệnh gộp hiển thị trên LED rời		
macro @WritePs <1,2>	; lệnh gộp ghi dữ liệu	lap: call delay
mov a, #11	; dữ liệu	rlc a
mov P12, a	; ghi ra cổng P12	mov P3, a
endmac	; kết thúc lệnh gộp	jnc lap
ORG 00H		sjmp \$; kết thúc chờ
@WritePs 55h, 0	; ghi 55h ra P0	delay: mov r0, #250
call delay		del1: mov r1, #200
@WritePs 0AAh, 1	; ghi 0AAh ra P1	del2: djnz r1, del2
call delay		djnz r0, del1
@WritePs 00h, 2	; ghi 0AAh ra P1	ret
call delay		end
@WritePs 01h, 3	; ghi 0AAh ra P1	
; Ví dụ 11.1b: Dạng đuôi. LST của chương trình dùng lệnh gộp hiển thị trên LED đơn		
1 0000	; ví dụ 11.1a. LEDR, hiển thị trên LED rời	
6 0000	ORG 00H	
7 0000	MACLEVEL 00017	
0000	mov a, #55h	; dữ liệu
7 0002	mov P0, a	; ghi ra cổng P0
7 0004	MACLEVEL 0000	
8 0004	call delay	
9 0007	MACLEVEL 0002	
9 0007	mov a, #0AAh	; dữ liệu
9 0009	mov P1, a	; ghi ra cổng P1
9 000B	MACLEVEL 0000	

10	000B	call delay	
11	000E	MACLEVEL 0003	
11	000E	mov a, #00h	; dữ liệu
11	0010	mov P2, a	; ghi ra cổng P2
11	0012	MACLEVEL 0000	
12	0012	call delay	
13	0015	MACLEVEL 0004	
13	0015	mov a, #01h	; dữ liệu
13	0017	mov P3, a	; ghi ra cổng P3
13	0019	MACLEVEL 0000	
14	0019	lap: call delay	
15	001C	rlc a	
16	001D	mov P3, a	
17	001F	nc lap	
18	0021	sjmp \$	; kết thúc chờ
19	0023	delay: mov r0, #250	
20	0025	del1: mov r1, #200	
21	0027	del2: djnz r1, del2	
22	0029	djnz r0, del1	
23	002B	ret	

11.1.4. Lập trình điều khiển LED rời trong các biển quảng cáo

Với ví dụ 11.1 trên, chúng ta có thể viết chương trình đưa hiển thị như mong muốn:

- Cho tất cả các đèn nối với một cổng nào đó sáng khi đưa A = #0FFh.
- Cho mỗi đèn sáng lần lượt từ trái qua phải dùng lệnh SETB C và dịch phải qua C với lệnh RRC A.
- Cho mỗi đèn sáng lần lượt từ phải qua trái dùng lệnh SETB C và dịch trái qua C với lệnh RLCA.
- Cho hai đèn sáng dồn từng đèn từ hai đầu vào giữa với các giá trị đưa trực tiếp vào thanh ghi A như 10000001B, 01000010B, 00100100B, 00011000B.
- Cho các đèn sáng dồn từ hai đầu vào giữa với các giá trị A là 10000001B, 11000011B, 11100111B, 11111111B.
- Cho các đèn sáng dồn từ giữa ra hai đầu với các giá trị A là 00011000B, 00111100B, 01111110B, 11111111B.

Có ba cách viết chương trình hiển thị động (đèn sáng chạy) trên các LED của các biển quảng cáo là:

- Đưa các giá trị (0 cho tắt và 1 cho sáng) trực tiếp ra các cổng.
- Thành lập một bảng dữ liệu, rồi đọc bảng để đưa ra cổng (ví dụ 11.2).
- Dùng lệnh logic và vòng lặp (ví dụ 11.1 và ví dụ 11.3), còn có thể dùng lệnh ORL.

; Ví dụ 11.2: Chương trình sáng dồn vào giữa	; Ví dụ 11.3: Chương trình sáng dồn 16 bit
org 00h lb: mov dptr, #bang lb1: mov a, #00h movc a, @a+dptr	macro @sangdon <1, 2, 3> mov r0, #1h mov r2, #01h mov pl1, #2

<pre> mov p0, a lcall delay inc dptr cjne a, #0ffh, lb1 sjmp lb delay: mov r7, #25 nhan0: mov R6, #20 nhan1: mov r5, #250 djnz r5, \$ djnz r6, nhan1 djnz r7, nhan0 ret bang: DB 00000000B ; hay 00H DB 10000001B ; hay 81H DB 11000011B ; hay 0C3H DB 11100111B ; hay 0E7H DB 11111111B ; hay 0FFH DB 00011000B ; hay 18H DB 00111100B ; hay 3CH DB 01111110B ; hay 7EH DB 11111111B ; hay 0FFH DB 00000000B ; hay 00H end </pre>	<pre> mov pl3, r2 lapl4: acall delay mov a, r0 rl a mov r0, a add a, r2 mov r2, a mov pl3, a acall delay mov a, r2 cjne a, #0ffh, lapl4 endmac org 00h main: @sangdon 0, 00h, 2, 1 ; sáng cổng P2 @sangdon 2, 0ffh, 0, 2 ; sáng cổng P0 @sangdon 2, 00h, 0, 3 ; sáng cổng P0 sangdon 0, 0ffh, 2, 4 ; sáng cổng P2 ljmp main delay: mov r7, #25 nhan0: mov r6, #20 nhan1: mov r5, #250 djnz r5, \$ djnz r6, nhan1 djnz r7, nhan0 ret end </pre>
--	--

Trong ba cách trên, cách đưa dữ liệu trực tiếp không thể dùng cho một bảng dữ liệu dài. Cách dùng bảng là thông dụng nhất, cho mỗi trường hợp, nhất là cách hiển thị khó, không có giải thuật xác định. Cách dùng lệnh với giải thuật đôi khi ngắn gọn. Hiển thị dồn (sang trái hay sang phải). Còn có giải thuật khác là dùng lệnh logic ORL giá trị đưa ra hiển thị cũ x0 và giá trị dịch một bit y0 (bởi lệnh SETB C và RLC A).

Đèn LED rời trên, ngoài chỉ thị giá trị của thanh ghi hay ô nhớ của VDK, còn hay dùng để quảng cáo hay điều khiển đèn giao thông tại các ngã đường. Khi đó, phải mắc đèn công suất lớn với các tầng khuếch đại công suất dùng tranzito công suất.

Các chương trình trên có thể thay đổi thời gian trễ Delay để thời gian sáng hay tắt tùy ý. Với đèn điều khiển giao thông, có lúc phải cho hai đèn sáng đồng thời (đỏ, vàng) để chỉ sự chuẩn bị chuyển luồng giao thông hay chuyển đèn (đỏ sang xanh hay xanh sang vàng rồi đỏ). Người ta còn đưa thời gian đếm ngược từ thời gian sáng tối đưa cho mỗi đèn về 0 và hiển thị số đếm dạng thập phân cho các đèn để người đi đường biết và chuẩn bị đi hay dừng tại các giao lộ (xem phần 11.2 cho hiển thị LED 7 đoạn).

11.1.5. Thiết kế mạch LED đơn và chạy chương trình trên mạch mô phỏng

Trong thiết kế và chạy mô phỏng sơ đồ điện tử, mới xuất hiện chương trình mô phỏng PROTEUS có cả mô phỏng mạch với vi điều khiển các loại như MCS-51 của INTEL, PIC của Microchip, AVR của ATMEL, MC68H11 của Motorola v.v... Chương trình trên còn mô phỏng được các mạch số và tương tự thay cho Workbench và Circuit maker hay Protel.

Nhược điểm cơ bản của PROTEUS là không hợp dịch được các ngôn ngữ khác nhau trên giao diện, mà chỉ chạy mô phỏng được chương trình dạng đuôi.HEX. Muốn chạy được mạch VDK các loại, ta phải:

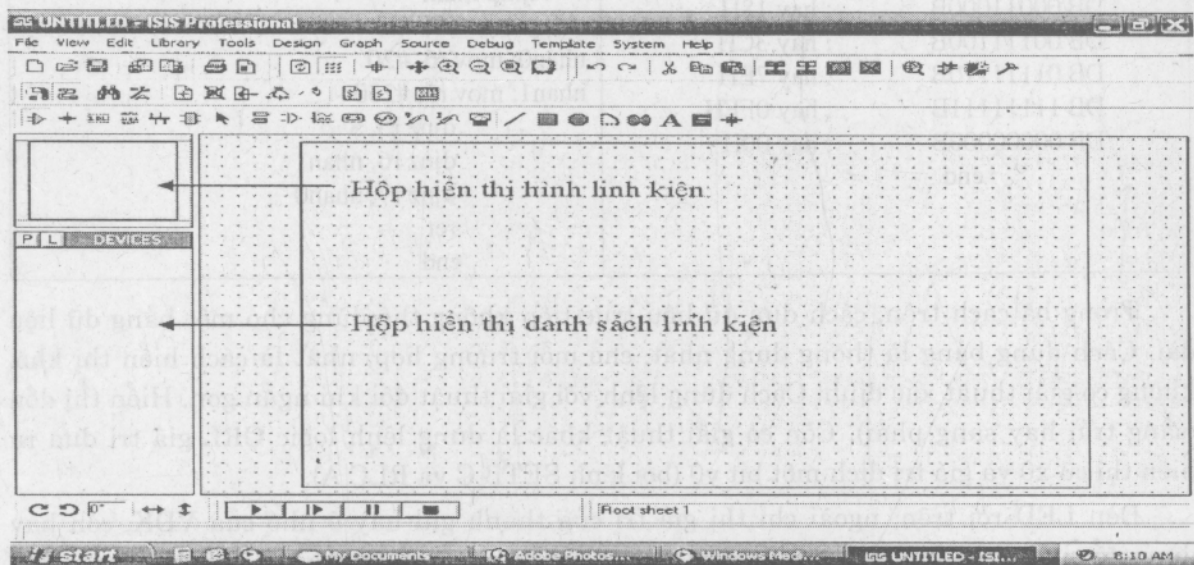
- Thiết kế mạch nguyên lý với chương trình ISIS.
- Viết chương trình và hợp dịch thành file. HEX với các chương trình hợp dịch tương ứng với ngôn ngữ đó bằng cách vào tệp Labcenter/Proteus/Tool/ASEM5I

Ví dụ: - PINNACLE cho họ MCS-51 của INTEL hay ASEM51.
 - AVR STUDIO cho AVR của ATMEL.
 - MPASM cho PIC của Microchip.

Ngoài các hợp ngữ, còn các ngôn ngữ bậc cao (ngôn ngữ C, C++, PASCAL, C51 và R51 cũng dịch ra tệp đuôi. HEX để chạy trên PROTEUS.

1. Chương trình PROTEUS

PROTEUS có giao diện và 4 thanh công cụ như hình 11.3, 11.4 và 11.5.

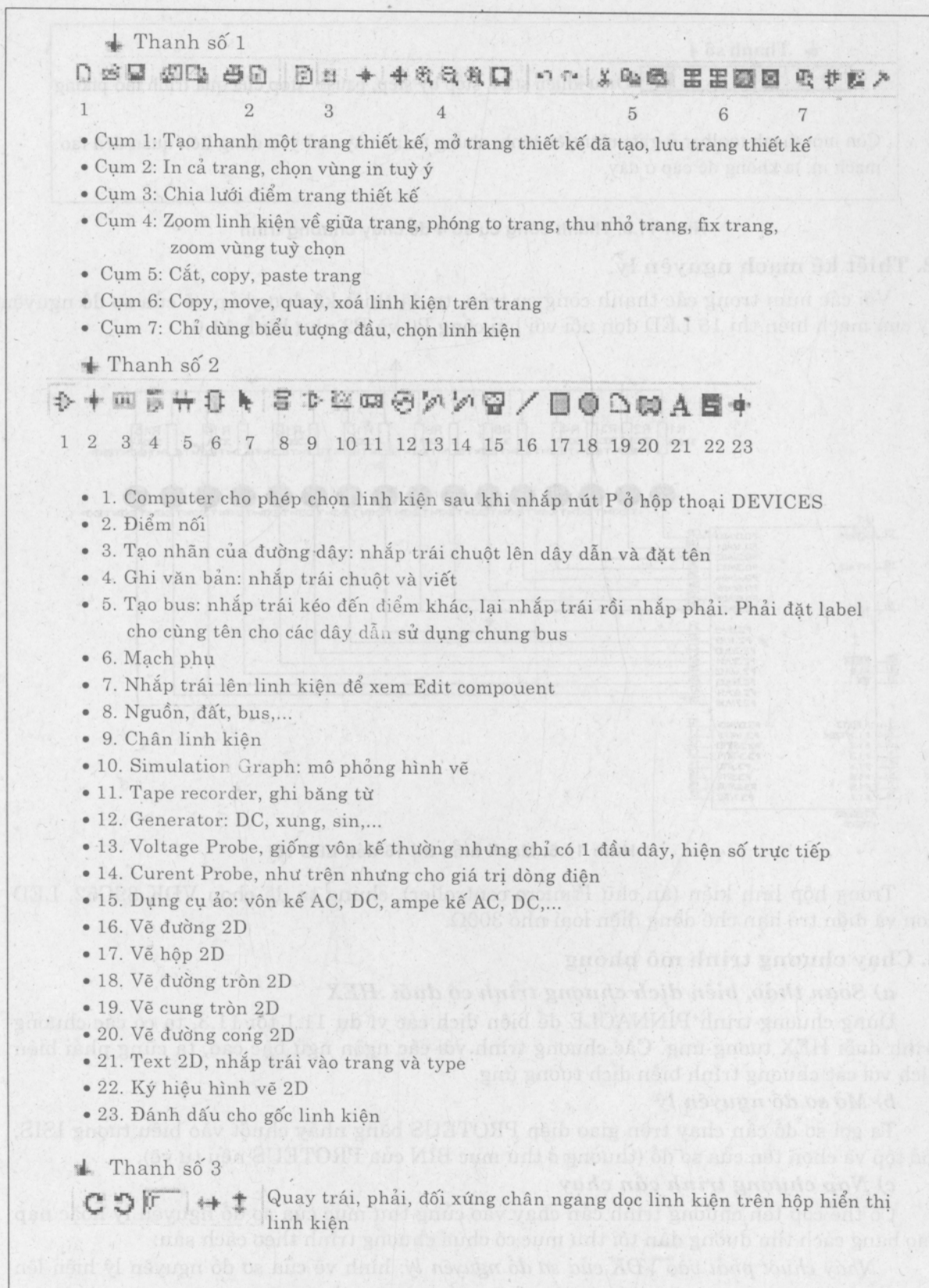


Hình 11.3. Giao diện của PROTEUS

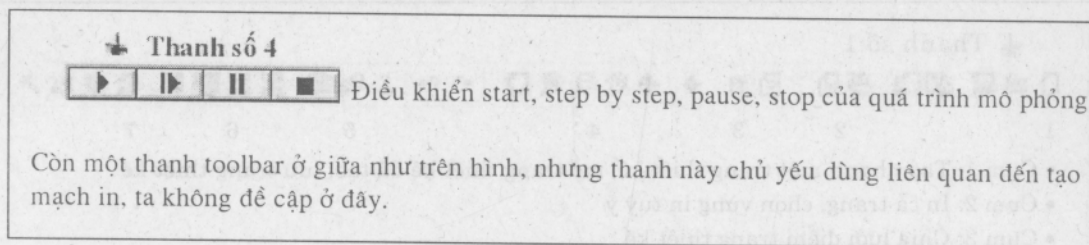
Công dụng của các thanh công cụ như sau:

- Thanh công cụ số 1 dùng để tạo trang, lưu giữ, sao chép, phóng to, thu nhỏ, xem và in v.v... các trang thiết kế.
- Thanh công cụ số 2 dùng để chọn các vi mạch ra danh sách linh kiện, vẽ các đường dây, ghi các nhãn v.v...
- Thanh công cụ số 3 dùng để quay 0° , 90° , 180° các linh kiện và tên nhãn.
- Thanh công cụ số 4 dùng để chạy chương trình mô phỏng.

Tên và công dụng từng nút trong nhóm các thanh ghi trên trình bày ở các hình vẽ ở trên.



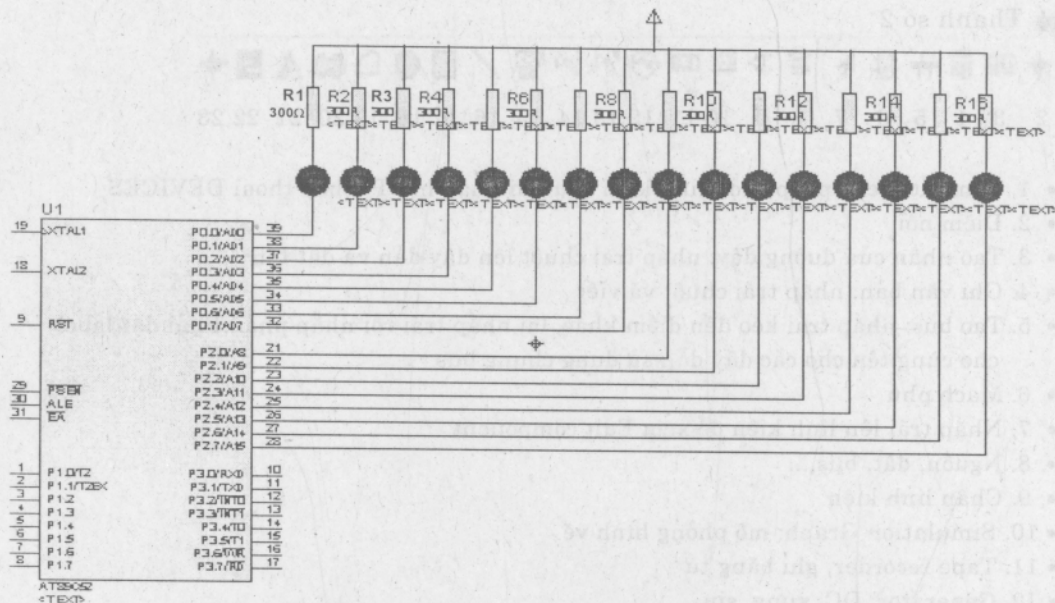
Hình 11.4. Các thanh công cụ 1, 2 và 3 của giao diện PROTEUS



Hình 11.5. Thanh công cụ số 4 để chạy chương trình

2. Thiết kế mạch nguyên lý

Với các nút trong các thanh công cụ trên, ta đã thiết kế được bản vẽ của sơ đồ nguyên lý của mạch hiển thị 16 LED đơn nối với hai cổng P0 và P2 như hình 11.6.



Hình 11.6. Mạch hiển thị 16 đèn LED rời

Trong hộp linh kiện (ấn chữ P/micro controller), chúng ta đã chọn VDK 89C52, LED đơn và điện trở hạn chế dòng điện loại nhỏ 300Ω.

3. Chạy chương trình mô phỏng

a) Soạn thảo, biên dịch chương trình có đuôi .HEX

Dùng chương trình PINNACLE để biên dịch các ví dụ 11.1 tới 11.3, ta có các chương trình đuôi HEX tương ứng. Các chương trình với các ngôn ngữ bậc cao, ta cũng phải biên dịch với các chương trình biên dịch tương ứng.

b) Mở sơ đồ nguyên lý

Ta gọi sơ đồ cần chạy trên giao diện PROTEUS bằng nháy chuột vào biểu tượng ISIS, mở tệp và chọn tên của sơ đồ (thường ở thư mục BIN của PROTEUS nếu tự vẽ).

c) Nạp chương trình cần chạy

Có thể copy tên chương trình cần chạy vào cùng thư mục của sơ đồ nguyên lý hoặc nạp vào bằng cách tìm đường dẫn tới thư mục có chứa chương trình theo cách sau:

- Nháy chuột phải vào VDK của sơ đồ nguyên lý: hình vẽ của sơ đồ nguyên lý hiện lên màu đỏ.

- Nháy chuột trái vào VDK của sơ đồ nguyên lý: hiện lên giao diện của chạy chương

trình (tránh nhấn chuột phải lần hai vì sơ đồ nguyên lý bị xoá đi, phải hồi phục bằng nút nhấn nút trở về mũi tên ngược chiều trên thanh công cụ số 1).

– *Nhấn chuột trái vào ô chương trình*: để tìm đường dẫn tới chương trình cần chạy trong PINNACLE.

Nhấn chuột trái vào tên chương trình cần chạy: khi đó, tên chương trình sẽ hiện vào hộp tên của chương trình với cả đường dẫn. Nếu có tên chương trình vào cùng thư mục với sơ đồ nguyên lý, ta không phải tìm tên chương trình theo đường dẫn.

– *Kiểm tra tần số nhịp của giao diện có hợp với tần số của VDK*: giá trị mặc định là 12MHz, nếu tần số nhịp không hợp, phải sửa lại cho đúng.

– *Trở về sơ đồ nguyên lý*: nhấn chuột trái vào chữ OK để trở về sơ đồ nguyên lý.

d) Chạy chương trình

Dùng thanh công cụ số 4 để chạy chương trình bằng cách nhấn chuột vào các nút tương ứng sau:

- | | |
|---|----------------------------------|
| – <i>Nút số 1 hình tam giác</i> : | chạy cả chương trình. |
| – <i>Nút số 2 hình một vạch và tam giác</i> : | chạy từng lệnh của chương trình. |
| – <i>Nút số 3 hình hai vạch thẳng</i> : | ngừng chạy chương trình. |
| – <i>Nút số 4 hình chữ nhật</i> : | kết thúc chạy chương trình. |

Trong khi chạy chương trình, trên các đường dây tín hiệu, có các chấm đỏ, xanh tương ứng với các mức logic 1 hay 0.

Có thể chương trình không chạy hay báo lỗi vì lý do khác nhau: tốc độ xung nhịp không hợp, quá thời gian, lệnh sai v.v...

Nếu chạy chương trình hiển thị trên 16 LED ở trên, ta sẽ thấy các LED sẽ nhấp nháy như quy luật giải thuật của chương trình. Nếu không như mong muốn, giải thuật không đúng, phải sửa lại.

Một sơ đồ nguyên lý ở trên, có thể cho chạy mô phỏng thử nhiều chương trình phục vụ cho nó.

Với cách thiết kế và chạy chương trình theo chương trình mô phỏng PROTEUS, ta có thể thực hành được về thiết kế và lập trình cho hệ VDK, trước khi chạy trên sơ đồ lắp ráp.

Với sơ đồ nguyên lý trên, ta có thể vẽ được sơ đồ mạch in với phần mềm ARES của PROTEUS.

Sau khi làm mạch in, lắp linh kiện, nuôi mạch bằng nguồn điện nuôi, nạp chương trình vào bộ nhớ PROM với máy nạp chương trình (Programmer), chương trình có thể chạy trên sơ đồ lắp ráp sau khi đã chạy tốt trên chương trình mô phỏng.

11.2. HIỂN THỊ TRÊN MỘT ĐÈN LED 7 ĐOẠN

Đèn LED rời có nhược điểm là chỉ biểu diễn được số nhị phân (logic 0 – đèn tắt và 1 – đèn sáng) không biểu diễn trực tiếp được các chữ và các con số. Muốn biểu diễn đèn LED rời thành chữ hay số, ta phải mắc nhiều đèn thành chữ hay số đó. Như vậy, sẽ tốn nhiều đèn và công suất điện nuôi lớn.

Đèn LED 7 đoạn (hay 7 thanh sáng) ra đời để khắc phục được nhược điểm trên.

11.2.1. Cấu tạo và hoạt động của đèn LED 7 đoạn

Mỗi đèn LED 7 đoạn có 7 thanh a, b, c, d, e, f và chấm h (để biểu diễn số thập phân), được sắp xếp như hình 11.7a. Mỗi thanh là một đèn LED đơn, được mắc theo:

– *Anôt chung*: (hình 11.7b), các anôt của các đèn nối chung với nhau và với cực +5V của nguồn điện nuôi, còn các cực âm là các katôt a, b, c, d, e, f, g, h.

– *Katôt chung*: (hình 10.7c), các katôt của các đèn nối chung với nhau và với đất, các anôt được nối với các lối vào a, b, c, d, e, f, g và h.

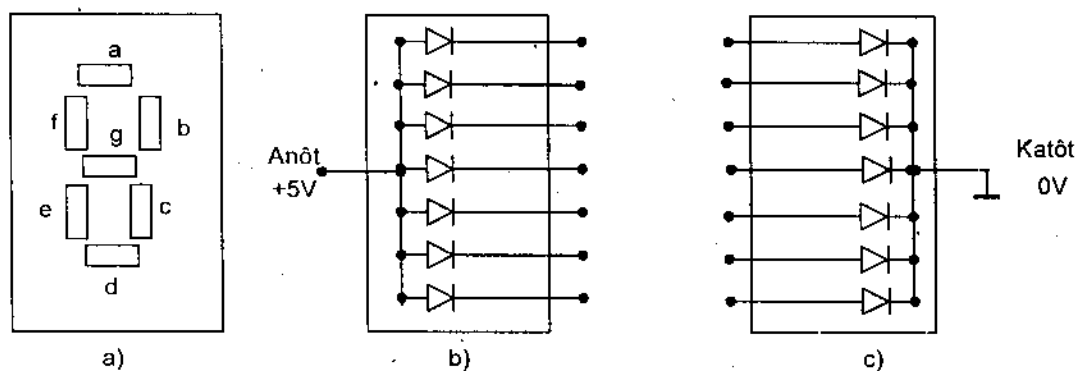
Như vậy, trong cả hai trường hợp (mắc anôt chung hay katôt chung), các đầu còn lại a, b, c, d, e, f, g và h của đèn đều nối với các lối ra của cổng vi điều khiển có giá trị của mã 7 đoạn.

Khi các thanh diot có dòng điện chạy qua (có thêm điện trở hạn chế dòng cỡ 200Ω), thanh phát sáng và tùy mã tương ứng của các lối ra a, b, c, d, e, f, g và h ta có hình sáng của các chữ số 0 tới 9 hay chữ từ A tới Z (bảng 11.1).

Bảng 11.1 được thành lập bằng cách:

– Mắc anôt chung hay katôt chung, để từ đó có lối ra từ a tới g có mức logic 0 (cho mắc anôt chung) hay 1 (cho mắc katôt chung). Các mã của cùng một số bù nhau (tổng = 0FFh).

– Các thanh sáng từ a tới g hiện hình các chữ số từ 0 (các thanh a tới f sáng, trừ thanh e tắt) tới 9 (các thanh a, b, c, d, f, g sáng, trừ thanh e tắt).



Hình 11.7. LED 7 thanh (a), có anôt chung (b) hay katôt chung (c)

Bảng 11.1. Mã 7 đoạn cho các số từ 0 tới 9

Số	Mã BCD	D = h	D6 = g	D5 = f	D4 = e	D3 = d	D2 = c	D1 = b	D0 = a	Mã 7đ A chung	Mã 7đ K chung
0	0000	0	0	1	1	1	1	1	1	C0	3FH
1	0001	0	0	0	0	0	0	1	1	F9	06H
2	0010	0	1	0	1	1	0	1	1	A4	5BH
3	0011	0	1	0	0	1	1	1	1	B0	4FH
4	0100	0	1	1	0	1	1	1	0	99	66H
5	0101	0	1	1	0	1	1	0	1	92	6DH
6	0110	0	1	1	1	1	1	0	1	82	7DH
7	0111	0	0	0	0	0	1	1	1	F8	07H
8	1000	0	1	1	1	1	1	1	1	80	7FH
9	1001	0	1	1	0	1	1	1	1	90	6FH
Blank	10H	0	0	0	0	0	0	0	0	0H	0H

11.2.2. Các loại đèn LED 7 thanh

Từ hình 11.8, ta thấy có các đèn LED đơn có các loại sau:

– Đèn LED 7 BCD: có 4 lối vào ABCD (có sẵn mạch giải mã 7 thanh bên trong) nên khi hiển thị, không cần mạch giải mã 7 thanh bên ngoài với sơ đồ nguyên lý ở hình (11.9).

– Đèn LED 7 số (digital): có 8 lối vào a, b, c, d, e, f, g, h, không cần nguồn nuôi ngoài (như anôt chung hay katôt chung) mà dùng mức logic 1 của lối ra của cổng VDK (có khi còn qua mạch công suất) để làm sáng thanh sáng. Loại đèn này cũng cần giải mã 7 thanh.

– Đèn LED 7 thanh mắc anôt chung: (nối nguồn nuôi ngoài) hay katôt chung (nối đất), có 8 lối vào a, b, c, d, e, f, g, h).

– Các đèn LED 7 thanh theo nhóm: hai đèn, 4 đèn (hình 11.8b), 6 đèn và 8 đèn (hình 11.8c).

– Đèn 16 thanh sáng: đèn có 16 thanh sáng.

Hãng TEXAS Instrument đã chế tạo LED 16 thanh TIL 308 và TIL 308A có dấu chấm thập phân DP (Decimal Point) ở bên trái còn TIL 309, TIL 309A ở bên phải.

Với LED 7 thanh sáng có những đặc điểm sau:

– Không có sẵn sơ đồ giải mã bên trong nên có các lối vào a, b, c, d, f, g, h. Do đó, khi hiển thị, cần vi mạch giải mã bên ngoài (ví dụ 74LS46, 74LS47 hay 4511, 4543) hay giải mã bằng chương trình.

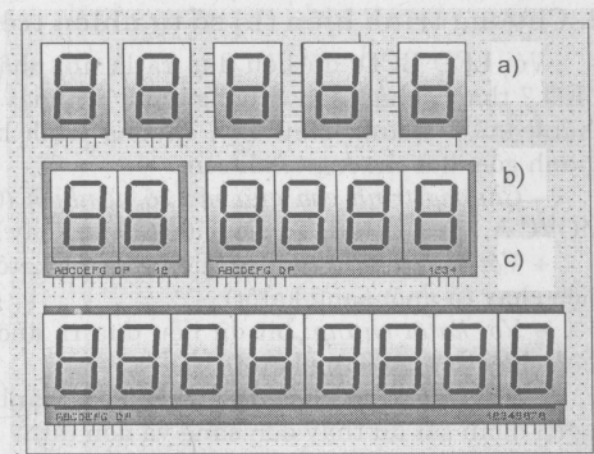
– Ngoài các lối vào dữ liệu (từ a tới h) cổng có các lối vào điều khiển sau:

+ /E: để cho phép đưa dữ liệu ra.

+ /LT: (Lampe Test– thử đèn), dùng để thử đèn 7 thanh sáng.

+ /BI/RBO: (Blanc Input – vào trắng/Ripple Blanc Output – đầu ra làm trắng số thập phân).

+ /RBI: (Ripple Blanc Input – đầu vào làm trắng số thập phân), dùng để xóa dấu chấm thập phân khi không chỉ thị số thập phân khi /RBI = 0.



Hình 11.8. Các loại đèn LED 7 thanh

11.2.3. Thiết kế, lập trình hiển thị một đèn LED

1. Thiết kế, lập trình hiển thị không mạch giải mã

Để hiển thị một chữ số (từ 0 tới 9) hay một ký tự chữ cái (từ A tới Z), mỗi đèn LED có thể mắc trực tiếp (không qua mạch giải mã hay bộ đếm công suất) như sau:

– Đèn LED BCD: 4 lối vào A, B, C, D mắc với 4 bit lối ra của cổng VĐK.

– Đèn LED số: các lối vào a, b, c, d, e, f, g, h mắc với 7 bit lối ra của cổng VĐK.

– Đèn LED anôt chung hay katôt chung: các lối vào a, b, c, d, e, f, g, h mắc với 7 bit lối ra của cổng VĐK.

Hình 11.9 giới thiệu cách mắc các đèn LED nhiều thanh sáng không có mạch giải mã như sau:

– Đèn LED BCD mắc với 4 bit của cổng P1.

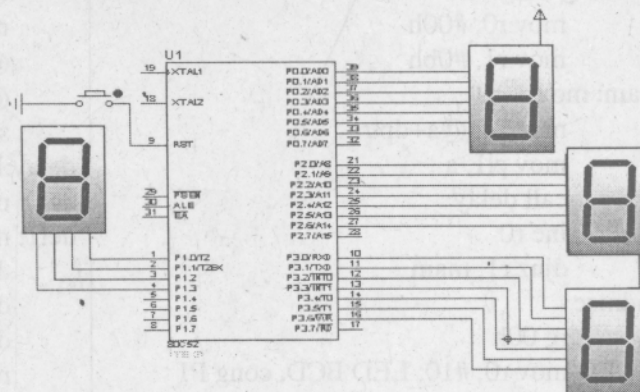
– Đèn LED anôt chung mắc với 8 bit của cổng P0.

– Đèn LED katôt chung mắc với 8 bit của cổng P2.

– Đèn LED mắc với 8 bit của cổng P3, có bảng mã giống như mắc katôt chung.

Như vậy:

– Chỉ có LED BCD chỉ cần 4 bit (A, B, C, D) của cổng ra vì có mạch giải mã bên trong.



Hình 11.9. Cách mắc các loại LED nhiều thanh

– Các đèn LED khác (anôt chung, katôt chung và LED số) đều cần 8 bit (chưa kể lối vào điều khiển dấu thập phân (DP trong TIL, /RBI trong các LED khác) và cần giải mã 2 – 7 (Binary – 7 segment). Bảng mã của LED 7 số cũng là bảng mã của LED 7 katôt chung.

2. Chương trình hiển thị số tự nhiên trên LED 7 thanh

Với LED BCD, dữ liệu đưa ra là nhị phân không cần chương trình giải mã. Còn các LED 7 thanh khác, cần chương trình con giải mã các số tự nhiên (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) thành mã 7 thanh. Như vậy, chương trình hiển thị không có mạch giải mã cho LED 7 thanh gồm hai chương trình con:

– *Chương trình con giải mã số tự nhiên (0 tới 9)*: bằng cách đọc bảng dữ liệu bởi lệnh `MOVC A, @A + DPTR`. Tùy loại anốt chung hay katốt chung, ta có hai bảng mã khác nhau vì:

+ *Với anốt chung*: bit dữ liệu đưa ra katốt, phải là logic 0 cho thanh sáng diot (dòng điện chạy từ anốt sang katốt).

+ *Với katốt chung*: bit dữ liệu đưa ra anốt, phải là logic 1 cho thanh sáng diot (dòng điện chạy từ anốt sang katốt).

– *Chương trình con hiển thị*: có các hành động đưa dữ liệu ra cổng, gọi trễ (để mắt người quan sát đủ thấy đèn sáng và đèn tắt).

Ví dụ 11.4 giới thiệu chương trình hiển thị trên các đèn LED các loại khác nhau theo sơ đồ mắc ở hình 11.9.

Sau khi hợp dịch trong Pinnacle, ghi vào sơ đồ mô phỏng ở hình 11.9 và nhấn nút chạy chương trình (nút thứ nhất của thanh công cụ số 4), theo dõi các đèn LED 7 thanh, ta thấy mỗi đèn sẽ hiển thị các số từ 0 tới 9 rồi trở về 0.

Ví dụ 11.5 giới thiệu chương trình đưa mã 7 đoạn từ ô nhớ ra hiển thị trên LED 7 đoạn. Trước đó có đoạn chương trình ghi và biến đổi các mã BCD thành mã 7 đoạn để ghi vào các ô nhớ.

3. Chương trình hiển thị ký tự

Hiển thị ký tự trên một LED 7 thanh cũng tương tự hiển thị số tự nhiên, chỉ có điều khác là:

– Các số tự nhiên có sẵn bảng mã số tự nhiên – mã 7 thanh (anốt chung hay katốt chung), trong khi hiển thị ký tự cần lập bảng mã cho chuỗi ký tự. Ví dụ, dòng chữ “CHAO – BAN”.

– Hiển thị cũng tương tự hiển thị số là đọc bảng dữ liệu (bởi lệnh `MOVC A, @A + DPTR`) rồi đưa A ra cổng Pi mong muốn.

Ví dụ 11.6a giới thiệu chương trình hiển thị chuỗi “CHAO BAN” đưa dữ liệu trực tiếp qua cổng P0 mắc với đèn LED 7 có katốt chung và chương trình 11.6b bằng đọc bảng dữ liệu.

; Ví dụ 11.4: Chương trình hiển thị các số tự nhiên (0 tới 9) trên các loại LED 7 <pre> macro @sangled7 mov r0, #00h mov r1, #0bh main: mov a, r0 movc a, @a+dpnr mov pl1, a call delay inc r0 djnz r1, main endmac org 00h mov r0, #10; LED BCD, cổng P1 mov a, #00h lap: mov P1, a call delay inc a cjne a, #0ah, lap </pre>	<pre> mov P1, #00h mov dpnr, #bangma1; anốt chung, P0 @sangled7 0 mov dpnr, #bangma2 @sangled7 2; katốt chung, cổng P2 @sangled7 3; LED số, cổng P3 sjmp \$ delay:MOV R7, #200 del0: mov r6, #50 del1: mov r5, #20 djnz r5, \$ djnz r6, del1 djnz r7, del0 ret bangma1: db 0c0h, 0f9h, 0a4h, 0b0h, 99h, 92h, 82h, 0f8h, 80h, 90h, 0c0h bangma2: db 03fh, 06h, 05bh, 04fh, 066h, 06dh, 7dh, 07h, 07fh, 06fh, 03fh end </pre>
--	---

; Ví dụ 11.5: Chương trình hiển thị dữ liệu trong ô nhớ dùng lệnh gộp giải mã	
MACRO @GIAIMA <1, 2, 3> mov a, l1 ; l1 - địa chỉ chứa dữ liệu anl a, #0fh ; lọc 4 bit thấp movc a, @a+dptr ; đọc mã 7 đoạn mov l2, a ; gửi vào địa chỉ nhớ l2 mov a, l1 ; l1 - địa chỉ chứa dữ liệu anl a, #0f0h ; lọc 4 bit cao swap a ; trao đổi 4 bit thấp movc a, @a+dptr ; đọc mã 7 đoạn mov l3, a ; gửi vào địa chỉ nhớ l3 ENDMAC org 00h Mov dptr, #bangma; địa chỉ bảng Mov r0, #30h Mov @r0, #25h	@GIAIMA 30h, 40h, 41h mov r0, #40h mov a, @r0 mov P0, a inc r0 mov a, @r0 mov P0, a Sjmp \$ bangma: db 0c0h, 0f9h, 0a4h, 0b0h, 99h, db 92h, 82h, 0f8h, 80h, 90h ; anốt chung ; (db 3fh, 06h, 5bh, 4fh, 66h, 6dh, 7dh, 07h, 7fh, 6fh, 00h; katốt chung delay: mov r0, #250 del1: mov r1, #200 del2: djnz r1, del2 djmp r0, del1 ret end

; Ví dụ 11.6a: Chương trình hiển thị “CHAO-BAN” đưa dữ liệu trực tiếp	; Ví dụ 11.6b: Chương trình hiển thị “CHAO-BAN” dùng lệnh đọc bảng dữ liệu
macro@WritePs <1, 2> ; lệnh dữ liệu mov a, #l1 ; dữ liệu mov P12, a ; ghi ra cổng P12 endmac ; kết thúc lệnh gộp ORG 00H @WritePs 039h, 0 ; ghi chữ “C” ra P0 call delay @WritePs 74h, 0 ; ghi chữ “H” ra P0 call delay @WritePs 5Ch, 0 ; ghi chữ “A” ra P0 call delay @WritePs 5Ch, 0 ; ghi chữ “O” ra P0 call delay @WritePs 40h, 0 ; ghi chữ “-” ra P0 call delay @WritePs 7Ch, 0 ; ghi chữ “B” ra P0 call delay @WritePs 5Ch, 0 ; ghi chữ “A” ra P0 call delay @WritePs 54h, 0 ; ghi chữ “N” ra P0 call delay sjmp \$; kết thúc chờ delay: mov r7, #250 del1: mov r6, #20 del2: mov r5, #20h djmp r5, \$ djmp r6, del2 djmp r7, del1 ret end	org 00h mov dptr, #bangma main: mov r0, #00h mov a, r0 movc a, @a+dptr djmp kt mov P0, a call delay inc dptr ljmp main kt: mov a, #00h mov P0, a call delay bangma: db 39h, 74h, 5ch, 3fh, 40h, 7ch db 5ch, 54h, 00h delay: mov r7, #250 del1: mov r6, #20 del2: mov r5, #20h djmp r5, \$ djmp r6, del2 djmp r7, del1 ret end <i>Chú ý:</i> Chuỗi ký tự kết thúc bởi mã 00h. Khi đọc tới mã này, chương trình nhảy tới nhãn KT để kết thúc.

11.3. THIẾT KẾ, LẬP TRÌNH HIỂN THỊ HỆ NHIỀU ĐÈN LED

Muốn hiển thị một chuỗi số (hay ký tự), ta phải dùng nhiều đèn LED nhiều thanh, mỗi đèn hiển thị một con số hay một ký tự.

11.3.1. Các phương án thiết kế hệ hiển thị nhiều LED

Trong các bảng hiển thị dạng số hay chữ khác, ta phải chọn số LED nhiều thanh bằng số các chữ số hay số các ký tự cần biểu diễn đồng thời. Trong các bảng số hay chữ chạy (xem ở dưới), ta có thể dùng số LED nhiều thanh, ít hơn số chữ cần hiển thị.

Yêu cầu đặt ra là thiết kế hệ hiển thị sao cho:

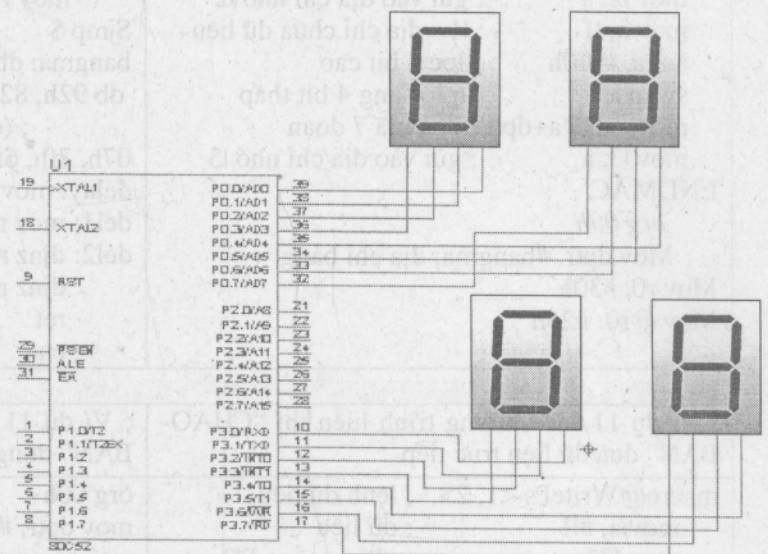
- Tốn ít cổng VDK.
- Điều khiển hiển thị đơn giản.

Có nhiều cách ghép nối nhiều đèn LED nhiều thanh cho hệ hiển thị trên.

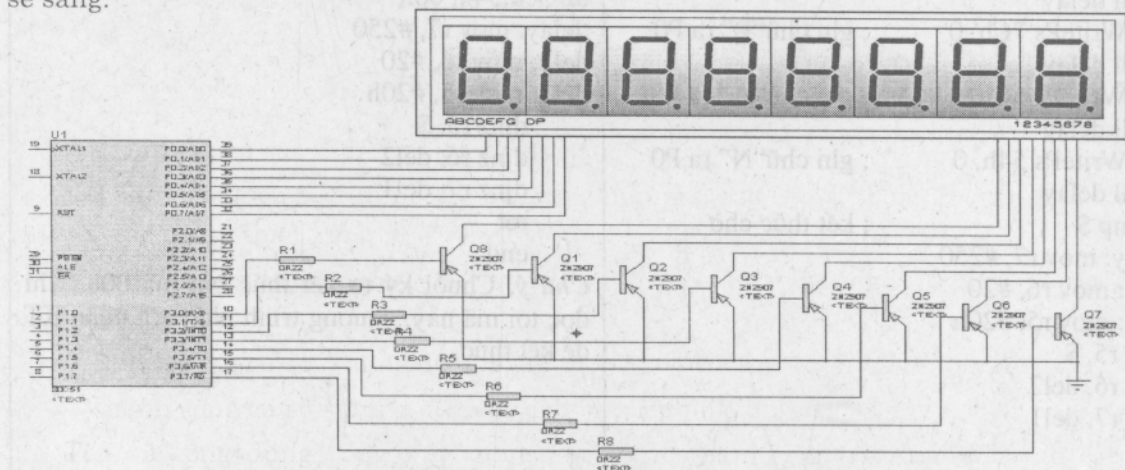
– *Hệ nhiều LED có đường dữ liệu nối trực tiếp với cổng:* (hình 11.10), hệ loại này, mỗi cổng chỉ ghép nối với 2 LED. Như vậy, hệ biểu diễn 8 chữ số (hay ký tự) cần tới 4 cổng.

– *Hệ có đường dây dữ liệu D0 – D7 chung cho mọi đèn:* (hình 11.11), hệ này cần thêm một cổng để điều khiển LED sáng thông qua các tranzito mắc nối tiếp với anôt hay katôt chung. Hệ này cần 2 cổng cho 8 LED và thường được dùng nhất. Chương trình đưa dữ liệu ra hiển thị có hai hành động:

- + Đưa dữ liệu mã 7 đoạn ra đường dây D0 – D7 qua cổng P0.
- + Đưa tín hiệu điều khiển tranzito từ cổng P2 cho từng tranzito và đèn LED tương ứng sẽ sáng.



Hình 1.10. Hệ 4 LED BCD



Hình 11.11. Cách mắc 8 đèn 7 đoạn thông dụng

– Hệ có vi mạch chốt dữ liệu: (hình 11.12), hệ này có đường dây dữ liệu chung đi vào các đèn qua vi mạch chốt dữ liệu và điều khiển đèn sáng bởi lối vào điều khiển chốt dữ liệu. Hệ này dùng cho các đèn LED số (không có lối vào điều khiển LED sáng như các LED có anôt hay katôt chung).

Hình 11.12 có 4 đèn LED số với 4 vi mạch chốt 74LS373. Có hai lối vào điều khiển chốt với /OE nối đất và LI nối với bit của cổng điều khiển P2.

Chương trình đưa dữ liệu ra hiển thị có hai hành động:

– Đưa dữ liệu mã 7 đoạn ra đường dây D0–D7 qua cổng P0.

– Đưa tín hiệu điều khiển chốt dữ liệu từ cổng P2 cho từng chốt và đèn LED nối với chốt đó sẽ sáng.

11.3.2. Hệ hiển thị nhiều LED

Trong hệ hiển thị nhiều LED có chung một đường dây dữ liệu, cần có mạch điều khiển cho LED sáng. Có thể nêu các loại mạch hiển thị sau:

– Điều khiển ghi dữ liệu vào vi mạch chốt dữ liệu: đây là trường hợp của nhiều LED BCD hay LED số. Dữ liệu được ghi vào chốt ở một trong hai lối vào, lối vào /OE với tín hiệu logic 0 hay lối vào LT với logic 1. Các tín hiệu điều khiển này có thể đi ra từ bộ giải mã địa chỉ hay trực tiếp từ một cổng của VĐK. Khi có dữ liệu ghi vào chốt, đèn LED tương ứng tự động sáng theo dữ liệu hiển thị.

– Điều khiển LED anôt hay katôt chung sáng: đây là trường hợp LED có anôt hay katôt chung với:

+ Tín hiệu logic 1 vào bazơ của tranzito ngược: tạo một dòng điện đi từ các lối ra a, b, c, d, e, f, g và h của VĐK hay vi mạch giải mã qua tranzito xuống đất.

+ Tín hiệu logic 0 vào bazơ của tranzito thuận: tạo một dòng điện đi từ nguồn điện nuôi của anôt chung vào các lối ra a, b, c, d, e, f, g và h của VĐK hay vi mạch giải mã qua tranzito xuống đất.

Nhờ có mạch điều khiển này ta có thể:

– Tiết kiệm được số cổng đáng kể đưa dữ liệu ra. Nếu có nhiều LED, ta phải mắc cổng ra của VĐK qua một mạch tăng công suất (ví dụ 2803) và ba trạng thái cho các dòng dữ liệu vào LED.

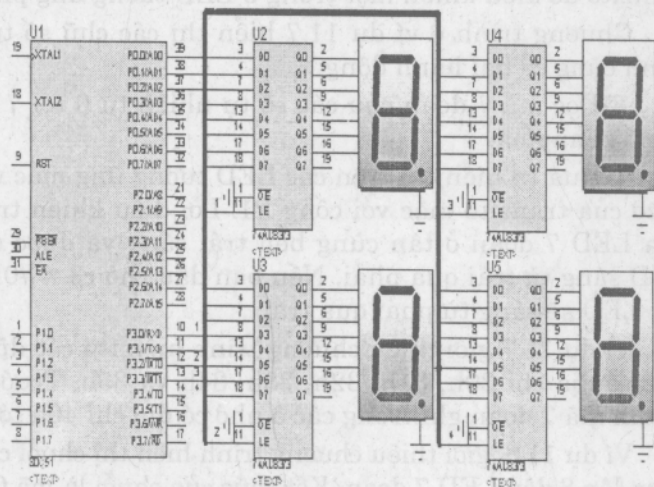
– Điều khiển đèn LED sáng theo quy luật mong muốn như chớp/tắt, hiển thị chạy, hiển thị dồn v.v... (xem ở dưới).

– Có khi để tiết kiệm cổng VĐK hơn nữa, người ta dùng thanh ghi dịch nối tiếp – song song để điều khiển LED (xem ở dưới).

– Giữa hai lần điều khiển LED sáng phải có thời gian trễ delay: Thời gian trễ delay này phải đủ lớn để mắt người có thể quan sát được dữ liệu trên đèn sáng, nhưng phải đủ nhỏ để người quan sát không cảm thấy các đèn nhấp nháy.

– Các dữ liệu đưa ra lần lượt và lặp lại nhiều lần: để mắt người quan sát có cảm giác các đèn sáng liên tục mà không cảm thấy nhấp nháy, thời gian lặp lại này phải nhỏ hơn 1/25 giây.

– Sau mỗi lần đèn LED sáng, phải điều khiển đèn tắt để chống lém.



Hình 11.12. Hệ nhiều LED số

11.3.3. Chương trình hiển thị cho nhiều đèn LED

Hình 11.11 giới thiệu cách mắc tối đa 8 đèn LED 7 đoạn với hai cổng của vi điều khiển: cổng P0 đưa dữ liệu ra song song cho 8 đèn LED và cổng P1 có 8 bit nối với 8 cực bazơ của 8 tranzito để điều khiển một trong 8 LED tương ứng phát sáng khi có dữ liệu đưa ra.

Chương trình ở ví dụ 11.7 hiển thị các chữ số từ 0 tới 7 trên 8 LED 7 đoạn. Chương trình cũng có hai hành động:

- Đọc mã 7 đoạn của các số tự nhiên từ 0 tới 7 có sẵn trong bảng mã với địa chỉ đầu tiên là bangma.
- Đưa ra hiển thị trên các LED tương ứng mắc với cổng P0 và tín hiệu điều khiển vào bazơ của tranzito mắc với cổng P1. Lỗi điều khiển trong r3 (có 1 bit bằng 0) có d7 = 0 cho đèn LED 7 đoạn ở tận cùng bên trái sáng và được dịch phải (RRC A) để điều khiển đèn LED sáng từ trái qua phải. Nếu ban đầu cho r3 = #0Eh và dùng lệnh dịch trái (RLC A) thì các LED sẽ sáng từ phải qua trái.

Ví dụ 11.7 giới thiệu chương trình hiển thị các dữ liệu dạng BCD được chứa trong các ô nhớ có địa chỉ 30h, 31h, 32h, 33h, 34h và 35h. Trước khi hiển thị, phải biến đổi mã BCD thành mã 7 đoạn ghi trong các ô nhớ có địa chỉ 40h tới 45h, bằng lệnh gộp GIAIMA.

Ví dụ 11.8. giới thiệu chương trình hiển thị chuỗi chữ APECHANOI2006 trong bảng mã 7 đoạn lên 8 đèn LED 7 đoạn. Kết thúc của chuỗi là mã 00h. Muốn biểu diễn đầy đủ chuỗi ký tự trên, ta phải có số đèn LED 7 đoạn bằng số ký tự cần biểu diễn (13 đèn LED 7 đoạn).

; Ví dụ 11.7: Chương trình hiển thị dữ liệu trong các ô nhớ 30h tới 35h	; Ví dụ 11.8: Chương trình hiển thị chữ “APEC HA NOI 2006: trong bảng mã
<pre>Org 00h Ljmpmain quetequP1 led7equP0 Org 100h Main: mov r0, #30h mov a, #00h lap0: mov @r0, a; ghi các số 0-7 vào các ô nhớ inc r0 inc a cjne a, #9, lap0 mov dptr, #bangma call giaima lap2: call hienthi sjmp lap2 giaima: mov r0, #30h mov r1, #40h lap1: Mov a, @r0 Movc a, @a+dptr; đọc mã 7 đoạn Mov @r1, a; gửi vào ô nhớ địa chỉ nhớ r1 inc r0 inc r1</pre>	<pre>quetequP1 led7equP0 org00h mov dptr, #bangma ; địa chỉ bảng mã mov r3, #07h ; bit d7= 0 lap: clr a ; xoá a movc a, @a+dptr ; đọc bảng jz kt mov led7, a ; đưa dữ liệu ra cổng P0 mov quet, r3 ; đưa điều khiển LED7 lcall delay ; gọi trễ mov quet, #0ffh ; xoá đèn LED7 mov a, r3 ; tạo điều khiển mới rr a ; dịch phải a mov r3, a ; ghi lại vào r3 inc dptr ; tăng địa chỉ mới sjmp lap ; trở về lap kt: sjmp \$ delay: mov r7, #250 ; trễ del1: mov r6, #20 del2: mov r5, #20h djnz r5, \$ djnz r6, del2 djnz r7, del1</pre>

<pre> mov a, r1 cjne a, #4Ah, lap1 ret hienthi: mov r3, #01111111b mov r0, #40h lap3: mov a, @r0 mov led7, a mov quet, r3 lcall delay mov quet, #0ffh ; xoá đèn LED7 mov a, r3 ; tạo điều khiển mới rr a ; dịch phải mov r3, a ; ghi lại vào r3 inc r0 mov a, r0 cjne a, #49h, lap3 ret delay: mov r7, #250; trễ del1: mov r6, #20 del2: mov r5, #20h djnz r5, \$ djnz r6, del2 djnz r7, del1 ret bangma: db 0C0h, 0F9h, 0A4h, 0B0h, 99h db 92h, 82h, 0F8h end </pre>	<pre> ret bangma: db 77h, 73h, 79h, 39h, 76h, 77h, 54h 3fh, 06h, 5bh, 3fh, 3fh, 7dh, 00h end </pre> <i>Chú ý:</i> Ví dụ 11.8 khác với các ví dụ ở trên là dùng lệnh <i>movc a, @a+dptr</i> và <i>inc dptr</i> (tăng dần địa chỉ của phần tử của bảng) để đọc tuần tự các phần tử của bảng (theo dptr vì A = 0) có địa chỉ đầu là <i>bangma</i> trong khi các ví dụ giải mã ở trên đọc phần tử bảng (theo giá trị của A vì dptr = bangma) bởi lệnh <i>movc a, @a+dptr</i>.
---	--

11.3.4. Hiển thị của một số thiết bị số thông dụng

Các thiết bị số nói chung, vi điều khiển nói riêng tích lũy dữ liệu dưới dạng nhị phân để tiết kiệm dung lượng nhớ. Nhưng đưa biểu diễn cho người điều hành quan sát lại phải hiển thị dưới dạng một con số thập phân gồm nhiều chữ số tự nhiên (từ 0 tới 9). Chúng ta phải viết chương trình để biến đổi mã nhị phân thành mã 7 đoạn (nếu dùng LED 7 đoạn) hay mã ma trận LED (nếu dùng các ma trận diot) để hiển thị. Ta sẽ xét hai trường hợp, hiển thị một byte nhị phân với số thập phân tới 256D và hai byte nhị phân với số thập phân tới 65536.

1. Đồng hồ điện tử số

Trong đời sống, chúng ta cần dùng đồng hồ chỉ thời gian thực. Người ta đã dùng vi điều khiển, trong đó có bộ đếm dao động của thạch anh làm đồng hồ đếm thời gian rồi chỉ thị thời gian hay số đếm về giờ, phút, giây lên các LED 7 đoạn tương ứng. Trong chương 8 về Timer của tập I, ta đã xét về Timer, ở đây, chúng ta xét tiếp về giải mã 7 đoạn và hiển thị trên các đèn LED 7 đoạn cho số đếm thời gian (giờ, phút, giây).

Trong chương trình của ví dụ 11.9 ta thấy:

- Dùng các lệnh gộp đếm, giải mã và hiển thị.
- Đếm thập phân cho giây, phút, giờ: số đếm chứa trong mỗi ô nhớ dạng BCD nén (ghi

trong các ô nhớ 30h, 31h và 32h) hay nói khác đi, mỗi ô nhớ chứa hai chữ số BCD nén của giây, chục giây (tới 60 giây), phút, chục phút (tới 60 phút) và giờ (chục giờ).

– *Giải mã BCD – 7đoạn*: sẽ giải mã 2 chữ số BCD nén trong một ô nhớ hay hai nibble khi dùng lệnh tráo hai nibble trong thanh ghi a (SWAP A). Như vậy, chương trình con giải mã phải dùng ba lệnh gộp giaima cho ba ô nhớ chứa các giá trị của giây, phút và giờ rồi ghi vào các ô nhớ từ 40h tới 45h.

– *Hiển thị*: chỉ dùng có một lệnh gộp với địa chỉ ô nhớ đầu chứa mã 7 đoạn của giây (40h) tới giá trị cạnh ô nhớ chứa mã của chục giờ (45h) để hiển thị ra 6 đèn LED 7 đoạn cho chục giờ, giờ, chục phút, phút, chục giây và giây.

Sau khi dịch bằng chương trình biên dịch PINNACLE ta sẽ có toàn văn bản của chương trình đồng hồ đếm thời gian có hiển thị thời gian trên 6 LED 7 đoạn cho chục giờ, giờ, chục phút, phút, chục giây và giây.

Trong chương trình của ví dụ 11.10 ta thấy chương trình giải mã phức tạp vì một byte nhớ chứa hai số BCD nén. Khắc phục nhược điểm trên, chương trình của ví dụ 11.11 dùng 6 lệnh đếm gộp cho 10 giây, 6 chục giây, 10 phút, 6 chục phút, giờ, chục giờ.

Trong chương trình của ví dụ 11.10 ta thấy:

– Dùng các lệnh gộp đếm còn giải mã và hiển thị là những chương trình con.

– *Đếm thập phân cho giây, chục giây, phút, chục phút, giờ và chục giờ*: số đếm chứa trong mỗi ô nhớ dạng BCD không nén (ghi trong các ô nhớ 30h, 31h, 32h, 33h, 34h và 35h) hay nói khác đi, mỗi ô nhớ chứa một chữ số BCD không nén của giây, chục giây, phút, chục phút, giờ và chục giờ.

– *Giải mã BCD – 7 đoạn*: sẽ giải mã 1 chữ số BCD không nén trong một ô nhớ hay một nibble nên không dùng lệnh tráo hai nibble trong thanh ghi a (SWAP A). Như vậy, chương trình con giải mã dùng cho sáu ô nhớ chứa các giá trị của giây, chục giây, phút, chục phút, giờ và chục giờ rồi ghi vào 6 ô nhớ từ 40h tới 45h.

– *Hiển thị*: chỉ dùng có một chương trình con với địa chỉ ô nhớ đầu chứa mã 7 đoạn của giây (40h) tới giá trị cạnh ô nhớ chứa mã của chục giờ (45h) để hiển thị ra 6 đèn LED 7 đoạn cho chục giờ, giờ, chục phút, phút, chục giây và giây.

So sánh hai chương trình của các ví dụ 11.9 và 11.10 ta thấy:

– Chương trình của ví dụ 11.9 đơn giản về việc đếm (chỉ 3 lệnh gộp) nhưng lại phức tạp trong giải mã.

– Chương trình của ví dụ 11.10 phức tạp về việc đếm (tới 6 lệnh gộp) nhưng lại đơn giản trong giải mã với một chương trình con.

– Cả hai chương trình giống nhau về hiển thị: một lệnh gộp hay một chương trình con.

; Ví dụ 11.9: Đồng hồ đếm BCD nén và hiển thị trên LED 7 đoạn	
MACRO @dem <1, 2, 3 > ; mov a, l1 add a, #1 da a mov l1, a cjne a, #12, l3 ENDMAC MACRO @GIAIMA <loai, nho1, nho2> mov a, l1 anl a, #0fh movc a, @a+dptr mov l2, a	mov sp, #70h mov dptr, #ma7doan clr tf0 mov tmod, #01h clr tf0 mov th0, #high (65536 - 5000) mov tl0, #low (65536 - 5000) mov ie, #82h ; cho phép ngắt Timer0 setb tr0 main4: mov gio, #00h main3: mov phut, #00h main2: mov giay, #00h

<pre> mov a, 11 anl a, #0f0h swap a movc a, @a+dptr mov l3, a ENDMAC MACRO @HIENTHI< dchigiay, dchigio+1> mov a, #01111111B; mã quét mov r0, #11; địa chỉ giây hthi1: mov led7, @r0 mov quet, a lcall delay mov quet, #0ffh inc r0 rr a mov a, r0 cjne a, #12, hthi1; địa chỉ giờ ENDMAC rg 0h jmp reset org 0bh jmp TimerInt0 quet equ P1 led7 equ P0 HBYTE equ 3bh LBYTE equ 3ah bdn equ 38h gio equ 32h phut equ 31h giay equ 30h org 100h Reset: </pre>	<pre> main1: mov bdn, #00h lcall giaima main0: @HIENTHI 40h, 46h mov a, bdn cjne a, #20, main0 @dem giay, 60h, main1 @dem phut, 60h, main2 @dem gio, 24h, main3 ljmp main4 ; chương trình con giải mã giaima: @GIAIMA giay, 40h, 41h @GIAIMA phut, 42h, 43h @GIAIMA gio, 44h, 45h ret ; chương trình con ngắt Timer0Int TimerInt0: clr tr0 inc bdn mov th0, #high (65536 - 5000) mov tl0, # low (65536 - 5000) setb tr0 reti Delay: mov r7, #20 Del: mov r6, #20 djnz r6, \$ djnz r7, Del ret ma7doan: db 0c0h, 0F9h, 0A4h, 0B0h db 99h, 92h, 82h, 0F8h, 80h, 90h end </pre>
; Ví dụ 11.10: Đồng hồ hiển thị đếm BCD không nén, hiển thị trên LED7 đoạn	
<pre> org 0h jmp reset org 0bh jmp TimerInt0 macro @dem <1, 2, 3 >; 1. loại đếm, ; 2. số sánh, 3. nhãn mov a, 11 add a, #1 da a mov l1, a cjne a, #12, l3 endmac quet equ P1 led7 equ P0 bdn equ 38h chucgio equ 35h </pre>	<pre> @dem phut, 10h, main3 @dem chucphut, 6h, main4 @dem gio, 10h, main5 @dem chucgio, 24h, main6 ljmp main7 TimerInt0: ; ngắt TimerInt0 sau 50msec inc bdn clr tr0 mov th0, #high (65536 - 4000) mov tl0, # low (65536-4000) clr tf0 setb tr0 reti gmá: ; chương trình con giải mã mov r5, #8; số lần biến đổi mã mov r0, #30h </pre>

<pre> gio equ 34h chucphut equ 33h phut equ 32h chucgiay equ 31h giay equ 30h org 100h Reset: mov sp, #70h mov dptr, #ma7doan clr tf0 mov tmod, #01h mov th0, #high (65536-4000) khởi tạo trễ Timer0 50ms mov tl0, #low (65536-4000) mov ie, #82h ; cho phép ngắt Timer0 setb tr0 main7: mov chucgio, #00h main6: mov gio, #00h main5: mov chucphut, #00h main4: mov phut, #00h main3: mov chucgiay, #00h main2: mov giay, #00h main1: mov bdn, #00h lcall gma main0: lcall hthi mov a, bdn cjne a, #20, main0 @dem giay, 10h, main1 @dem chucgiay, 6h, main2 </pre>	<pre> mov r1, #40h gma0: mov a, @r0 movc a, @a+dptr mov @r1, a inc r0 inc r1 djnz r5, gma0 ret hthi: ; chương trình con hiển thị mov a, #01111111B ; mã quét mov r0, #40h hthi1: mov led7, @r0 mov quet, a lcall delay mov quet, #0ffh inc r0 rr a mov a, r0 cjne a, #46h, hthi1 ret Delay: mov r7, #20 Del: mov r6, #20 djnz r6, \$ djnz r7, Del ret ; khai báo mã 7 đoạn từ số 0 tới số 9 ma7doan: db 0c0h, 0F9h, 0A4h, 0B0h, 99h, 92h, 82h, 0F8h, 80h, 90h end </pre>
--	--

2. Hiển thị một byte nhị phân trên LED 7 đoạn

Dung lượng của một byte nhị phân được hiển thị ra LED 7 đoạn được lập trình theo hai giai đoạn:

- Biến đổi số nhị phân (B) thành số dạng BCD không nén hay dạng HEX bằng chương trình.
- Biến đổi số dạng BCD không nén (hay dạng HEX) của các số tự nhiên (từ 0 tới 9) thành dạng mã 7 đoạn bằng đọc bảng mã theo các giá trị của A dạng BCD sang dạng mã 7 đoạn bằng lệnh `MOVC A, @A + DPTR`.
- Hiển thị mã 7 đoạn lên các LED 7 đoạn.

Ví dụ 11.11 giới thiệu chương trình hiển thị các số nhị phân của một byte chứa số nhị phân của 252D ra các số thập phân 252 trên các đèn LED 7 đoạn.

Chương trình dùng ba đoạn chương trình:

- Đoạn biến đổi một byte nhị phân của số FCH ra số thập phân 252D, rồi ghi vào trong các ô nhớ 10h, 11h và 12h.

– *Đoạn giải mã*: biến đổi số thập phân 252D đã lưu trữ trong các ô nhớ 10h, 11h và 12h (hàng đơn vị) thành các mã 7 đoạn bằng bảng mã có địa chỉ ban đầu là bangma. Mỗi số thập phân hàng trăm, chục, đơn vị có các mã A4h, 92h và A4h được ghi trong các vùng nhớ có địa chỉ là 30h, 31h và 32h.

– *Đoạn hiển thị:* lệnh này đọc các mã trong các ô nhớ có địa chỉ là 30h, 31h và 32h để hiển thị trên ba LED 7 đoạn. Số lần lặp lại và thời gian trễ có thể điều chỉnh để mắt nhìn rõ nhất.

3. Hiển thị hai byte nhị phân

Hai byte nhị phân thường được tích lũy số đếm, số thời gian ở hai thanh ghi TH, TL của vi điều khiển. Số thập phân tối đa chứa trong hai byte nhị phân là số thập phân 65536, tức chứa tới hàng chục ngàn (TTHOU), ngàn (THOU), hàng trăm (HUN), hàng chục (TEN) và hàng đơn vị (UNIT).

Chương trình hiển thị dữ liệu trong hai byte nhị phân cũng tương tự như một byte nhị phân đã xét nhưng nó phức tạp hơn vì có tới 5 con số thập phân.

Cũng có 3 công việc với 3 đoạn chương trình là :

– *Biến đổi số nhị phân ra 5 con số thập phân* của hàng chục ngàn, ngàn, trăm, chục và đơn vị. Vì có 5 đoạn biến đổi nên không dùng phép chia cho 10 đơn giản như trường hợp một byte mà dùng các phép trừ liên tiếp với 10000D = 2710H.

$$\begin{aligned} 1000D &= 3E8H \\ 100D &= 64H \\ 10D &= 0AH \end{aligned}$$

Ta dùng lệnh gộp chung cho cách biến đổi trên và ghi các số biến đổi vào các ô nhớ liên tiếp dành cho hàng chục ngàn (TTHOU), hàng ngàn (THOU), hàng trăm (HUN), hàng chục (TEN) và hàng đơn vị (UNIT) như ví dụ 11.12. Có thể dùng 5 phép chia cho 10D để biến đổi.

– *Giải mã số thập phân sang số mã 7 đoạn:* tương tự chương trình ở trên, ghi vào các dải nhớ tương ứng với các địa chỉ đầu 30h, 40h, 50h, 60h và 70h.

– *Hiển thị các mã trên ra các LED 7 đoạn:* tương tự chương trình ở trên.

4. Hiển thị thời gian đếm lùi ở đèn giao thông

Tại các nút giao thông trong thành phố hiện đại, người ta mắc các đèn :

– LED đơn to để điều khiển các phương tiện giao thông với màu xanh (cho đi qua), vàng (chuẩn bị) và đỏ (cấm đi qua).

– Các LED 7 đoạn nhỏ, màu xanh để chỉ thị thời gian đếm lùi của các phương tiện được phép đi qua và các LED 7 đoạn nhỏ, màu đỏ để chỉ thị thời gian đếm lùi của các phương tiện không được phép đi qua.

Do đó, trong các hộp đèn phải có bộ đếm lùi thời gian từ thời gian đặt sẵn (thường thời gian của màu xanh là 20sec, thời gian của màu vàng là 5sec và thời gian của màu đỏ = thời gian xanh + vàng = 25sec).

; Ví dụ 11.11: Hiển thị dữ liệu của một byte nhớ trên LED 7 đoạn

hang equ P0 cotd equ P3 org 00h mov dptr, #bangma mov a, #0fch ; so 252d mov b, #10 div ab mov 12h, b ; 12h chứa số đơn vị mov b, #10	hthi: mov r2, #250 mov r0, #30h mov r3, #00000100b mov r4, #3 lap2: mov a, @r0 mov hang, a mov cotd, r3 lcall delay mov cotd, #00
---	---

ab mov 10h, a ; 10h chứa số hàng trăm mov 11h, b ; 11h chứa số hàng chục mov R0, #10h mov r1, #30h mov r2, #00h lap1: mov a, @R0 movc a, @a+dptr mov @R1, a inc R1 inc R0 inc r2 mov a, R2 cjne a, #3, lap1	inc r0 mov a, r3 rr a mov r3, a djnz r4, lap2 djnz r2, hthi sjmp \$ bangma: db 0c0h, 0F9h, 0A4h, 0B0h, 99h db 92h, 82h, 0F8h, 80h, 90h delay: mov r5, #1 de: mov r6, 50 djnz r6, \$ djnz r5, de ret end
---	--

; Ví dụ 11.12: Hiển thị hai byte BIN ra đèn LED 7 đoạn	
MACRO @BINBCD ; <1.NHAN, 2.LBYTE, 3.HBYTE, 4.BAC> 11: CLR C MOV A, LBYTE SUBB A, #12 MOV R0, A MOV A, HBYTE SUBB A, #13 JC KT MOV HBYTE, A MOV A, R0 MOV LBYTE, A INC I4 JMP 11 KT: ENDMAC MACRO @GIAIMA <1,2> MOV A, 11 ; nạp ô nhớ địa chỉ 11 ANL A, #0FH MOVC A, @A+DPTR; đọc mã 7 theo A MOV I2, A ; nạp vào ô nhớ có địa chỉ I2 ENDMAC ORG 00h LJMP MAIN LBYTE DS 1 HBYTE DS 1 TTHOU EQU 30H THOU EQU 31H HUN EQU 32H TEN EQU 33H	BIN_0: MOV @R0, #0 INC R0 DJNZ R1, BIN_0 @BINBCD BTD_TT, 10H, 27H, TTHOU @BINBCD BTD_T, 0E8H, 03H, THOU @BINBCD BTD_HUN, 64H, 00H, HUN @BINBCD BTD_TEN, 0AH, 00H, TEN BTD_DV: MOV A, LBYTE MOV UNIT, A POP PSW MOV R7, #6H ; nạp số lần (số LED) MOV R0, #30H ; nạp địa chỉ BCD đầu MOV R1, #40H ; nạp địa chỉ đầu của mã MOV A, #00H MOV DPTR, #MA7DOAN; địa chỉ bảng @GIAIMA30H, 40H, 5 ; mã ở 34h vào 44h @GIAIMA 31H, 41H, 4; mã ở 33h vào 43h @GIAIMA 32H, 42H, 3; mã ở 32h vào 42h @GIAIMA 33H, 43H, 2; mã ở 33h vào 43h @GIAIMA 34H, 44H, 1; mã ở 34h vào 44h HIENTHI: MOV A, #1111110B MOV R0, #40H ; gán địa chỉ của TTHOU MOV B, #10000000B; điều khiển đèn LED HT1: MOV A, B MOV LED7, @R0; đọc mã ra P0 MOV A, B ; gửi B vào A MOV QUET, A ; đưa điều khiển LED LCALL DELAY ; gọi trễ MOV R0, #TTHOU INC R0 ; tăng địa chỉ nhớ mã

UNIT EQU 34H LED7 EQU P0 QUET EQU P1 ORG 100H MAIN: MOV LBYTE, #0FCH; ghi số 0FCH MOV HBYTE, #0FFH; ghi số 0FFH BIN_BCD: PUSH ACC PUSH PSW SETB RS0 CLR RS1 MOVR1, #5	RR A ; dịch phải A để điều khiển MOV B, A ; gửi A sang B MOV A, R0 ; gửi R0 sang A CJNE R0, #45H, HT1; so sánh địa chỉ cuối LJMP HIENTHI ; hiển thị lại DELAY: MOV R7, #8H DEL: MOV R6, #200 DEL1: MOV R5, #200 DJNZ R5, \$ POP ACC DJNZ R6, DEL1 DJNZ R7, DEL RET MA7DOAN: DB 0C0H, 0F9H, 0A4H, 0B0H DB 09H, 92H, 82H, 0F8H, 80H, 90H ; anốt chung END
---	---

Trong tập một của tài liệu này đã trình bày cách mắc và chương trình điều khiển đèn giao thông hiển thị LED 7 đoạn (Chương 7). Tại đây giới thiệu chương trình hiển thị với sự hiển thị thời gian đếm lùi trên các đèn Led 7 đoạn.

Ví dụ 11.13 giới thiệu chương trình điều khiển đèn giao thông với đếm lùi có hiển thị trên các đèn LED 7 đoạn. Thực chất chương trình trên gồm:

- Đếm lùi thời gian thực bằng ngắt do Timer với số đếm khác nhau định sẵn trong các ô nhớ.
- Giải mã ma trận diot các số đếm lùi.
- Hiển thị số đếm trên các đèn ma trận diot.
- Điều khiển các LED đơn to cho các màu và kích hoạt hay ngừng đếm lùi.

Các nhiệm vụ trên, ta đã xét ở các phần trước. Chương trình của ví dụ 11.12 chỉ có phần đếm và đóng mạch các đèn LED đơn to mà chưa có phần giải mã và hiển thị trên các diot ma trận. Người dùng sẽ tự động chèn các chương trình con cho giải mã và hiển thị vào trước vòng lặp chờ (nhãn WAIT1). Vì chỉ đếm từ 10 giây với 2 chữ số nên cần hai đèn Led 7 đoạn và cần 2 lối ra để điều khiển chúng. Hai lối ra P1.6 và P1.7 được dùng cho mục đích này. Còn lại, dùng một cổng nữa cho dữ liệu. Để tiện sử dụng, chương trình giới thiệu cả các lệnh gộp mà không gọi từ thư viện chương trình MYLIB.MAC.

Chương trình dùng :

- Hai lệnh gộp là @DEM và @ChopXanh để dùng cho chương trình chính có 6 đoạn sử dụng chúng với các thông số đếm (20, 4 cho 10 giây và 3 giây) và nhãn khác nhau.
- Chương trình phục vụ ngắt cho Timer0 với thời gian cơ bản là 50000 micro giây.

Theo trình tự của chương trình ta có:

- Đóng mạch các đèn Xanh 1 và Đỏ 2 trong 10 giây.
- Chớp/tắt đèn Xanh 1, đóng mạch Đỏ 2 trong $0,3 \times 5 = 3$ giây.
- Đóng mạch các đèn Vàng 1, Đỏ 2 trong 2 giây.
- Đóng mạch các đèn Xanh 2 và Đỏ 1 trong 10 giây .
- Chớp/tắt đèn Xanh 2, đóng mạch Đỏ 2 trong $0,3 \times 5 = 3$ giây.
- Đóng mạch các đèn Vàng 2, Đỏ 1 trong 2 giây.

So với ví dụ 7.5 và 7.6 của tập I, chương trình này chính xác và gọn nhẹ hơn.

; Ví dụ 11.13: Điều khiển đèn giao thông với các đèn Xanh, Đỏ, Vàng

<pre> macro @Dem <1,2,3> mov P1, #1 mov Count, #12 l3: jbc TimeOutflag, l4 sjmp l3 endmac macro @ChopXanh <1, 2, 3, 4, 5> Chopxanh1: mov r5, #6 l2: cpl l3 mov Count, #1 l4: jbc TimeOutflag, l5 sjmp l4 l5: djnz r5, l2 endmac org 00h jmp Reset org 0bh jmp Timer0Int Xanh1 equ P1.0 Vang1 equ P1.1 Do1 equ P1.2 Xanh2 equ P1.3 Vang2 equ P1.4 Do2 equ P1.5 org 20h TimeOutflag equ 20h HalfSecond equ 30h Count equ 31h ; dùng ngắt Timer0, chế độ 1 với thời gian ; là 50000 microsec ; các số đếm lùi là 0,5 x 20 = 10sec và 2 sec ; chớp đèn vàng 6 lần, mỗi lần 0,5sec </pre>	<pre> org 100h Reset: mov tmod, #00000001b mov th0, #high (65535-50000) mov tl0, #low (65535-50000) mov ie, #10000010b mov HalfSecond, #500/50 setb tr0 clr TimeOutflag Mainloop: @Dem 100001b, 20, Wait1, Chopxanh1 @ChopXanh 1, Loop1, Xanh1, Wait2, \$1 @Dem 100010b, 4, Wait3, \$2 \$2: @Dem 001100b, 20, Wait4, Chopxanh2 @ChopXanh 2, Loop2, Xanh2, Wait5, \$3 @Dem 010100b, 4, Wait6, \$4 \$4: jmp MainLoop Timer0Int: push acc mov th0, #high (65535-50000) mov tl0, #low (65535-50000) djnz HalfSecond, NotHalfSecond mov HalfSecond, #500/50 dec Count mov a, Count jnz Return setb TimeOutflag NotHalfSecond: Return: pop acc reti end </pre>
--	--

11.4. HIỂN THỊ MỘT CHUỖI SỐ HAY KÝ TỰ VỚI MẠCH GIẢI MÃ

11.4.1. Hiện thị với vi mạch giải mã 7 đoạn

1. Cách mắc mạch

Mạch giải mã 7 đoạn như 7446, 7447 thuộc họ TTL hay 4511, 4543 thuộc họ CMOS biến đổi một mã BCD có 4 lối vào ABCD sang mã 7 đoạn có 8 lối ra a, b, c, d, e, f, g, h. Khi đó, trong chương trình, chỉ cần một lệnh đưa dữ liệu ra cổng của vi điều khiển mắc với vi mạch giải mã 7447 và đèn LED 7 đoạn.

Cần hiển thị một chuỗi số hay ký tự ta có cách mắc như hình 11.13 như sau:

– Một mạch giải mã dữ liệu chung cho đường dây dữ liệu chung cho một đèn LED 7 đoạn, mắc với một cổng của vi điều khiển.

– Một mạch điều khiển các LED 7 đoạn mắc với một cổng khác của vi điều khiển.

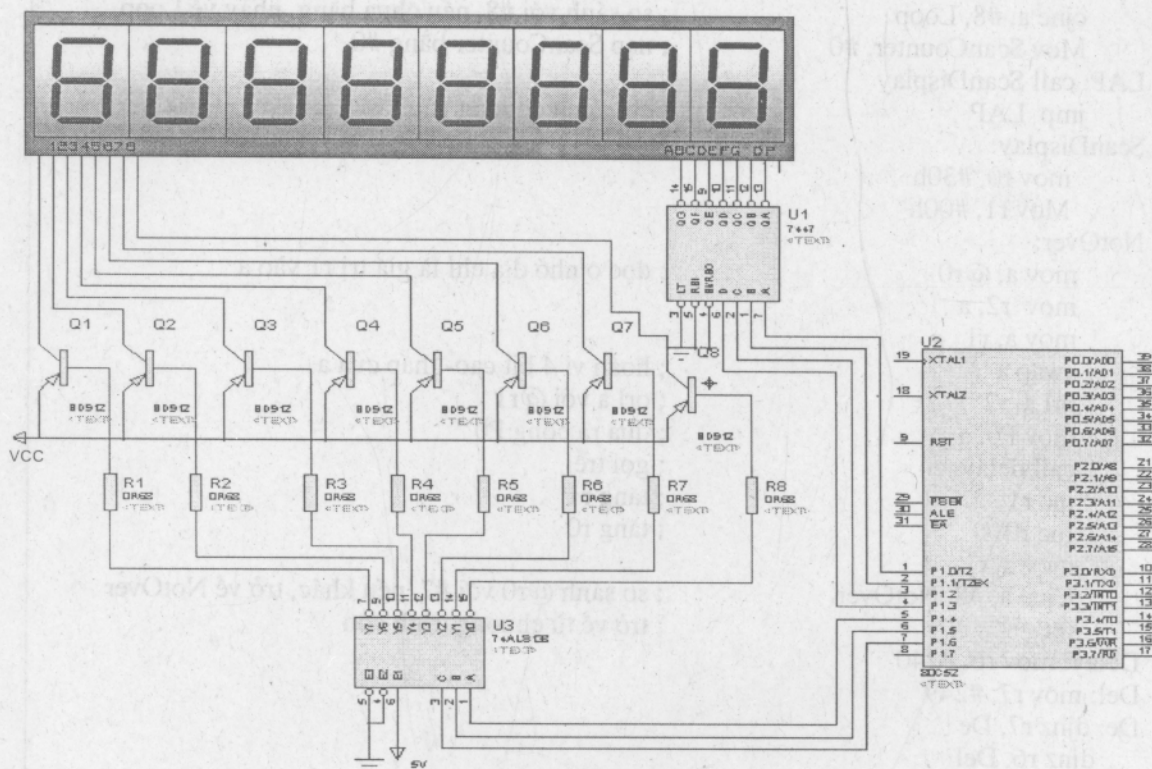
Hình 11.15 chỉ khác là có thêm vi mạch giải mã 74LS47 và 74LS138.

Chương trình hiển thị cho hình 11.15 không cần chương trình con giải mã và chương trình hiển thị cũng đơn giản hơn

– 4 bit P10 –P13: dành cho các lối vào DCBA để biểu diễn các số BCD. Bốn bit này được nối với 4 lối vào của vi mạch giải mã 7447.

– 3 bit P14 –P16: dành cho các lối vào CBA của mạch giải mã 3 –874LS138. Nếu dùng vi mạch giải mã 4–16 (ví dụ 74154) cần dùng cả 4 lối vào P14–P17. Các lối ra của các mạch giải mã này được nối với các cực bazơ của các tranzito để điều khiển LED sáng.

– 1 bit P17 nối với lối vào h hay dp của đèn LED cho chấm sáng.



Hình 11.13. Sơ đồ hiển thị dùng các vi mạch giải mã

2. Chương trình

Chương trình của ví dụ 11.14 có các phần sau:

- Ghi các số tự nhiên từ 0 tới 7 vào các ô nhớ RAM bên trong có địa chỉ từ 30h tới 37h.
- Cổng P0 hiển thị các số từ 0 tới 7 trong các ô nhớ từ 30h tới 37h với sự điều khiển hay quét các LED 7 đoạn bằng điều khiển các bit P14 tới P16 bởi các số tự nhiên từ 0 tới 7.
- Lệnh swap a hoán vị giá trị điều khiển các tranzito (0 tới 7) lên các lối ra P14 tới P16.
- Lệnh orl a, @r1 dùng để cộng các bit cao và thấp của a hay nói khác đi, giá trị tổng của a sẽ đưa ra cổng P1 là 4 bit thấp của dữ liệu (các số tự nhiên từ 0 tới 7) và các bit cao P14–P16 của điều khiển các LED qua vi mạch giải mã 74138 (từ 0 tới 7).
- Quan sát thấy chương trình hiện lên cổng P0 các số 00h, 11h, 22h, 33h, 44h, 55h, 66h và 77h rồi lại lặp lại.

; Ví dụ 11.14: Chương trình hiển thị các ô nhớ trên 8 đèn LED 7 đoạn

```

DisplayBuffer equ 30h
ScanCounter equ 38h
    org 00h                ; bắt đầu chương trình chính
    jmp Reset
Reset:                      ; chương trình chính
    Mov SP, #70h           ; bắt đầu vùng Stack từ địa chỉ nhớ 70h
    Mov r0, #DisplayBuffer ; bắt đầu địa chỉ nhớ đếm cho hiển thị (37h)
    Mov a, #0h
Loop: mov @r0, a            ; nạp #8 vào ô nhớ
    inc r0
    inc a                  ; tăng a
    cjne a, #8, Loop       ; so sánh với #8, nếu chưa bằng, nhảy về Loop
    Mov ScanCounter, #0    ; nạp ScanCounter bằng #0
LAP: call ScanDisplay
    jmp LAP                ; chờ ngắt chương trình
ScanDisplay:
    mov r0, #30h
    Mov r1, #00h
NotOver:
    mov a, @r0             ; đọc ô nhớ địa chỉ là giá trị r1 vào a
    mov r2, a
    mov a, r1
    swap a                 ; hoán vị 4 bit cao- thấp của a
    orl a, r2              ; orl a với @r1
    mov P0, a              ; đưa ra cổng P0
    call delay             ; gọi trễ
    inc r1                 ; tăng r1
    Inc r0                 ; tăng r0
    mov a, r1
    Cjne a, #8, NotOver    ; so sánh @r0 với #7, nếu khác, trở về NotOver
    ret                   ; trở về từ chương trình con
Delay: mov r6, #240
Del: mov r7, #249
De: djnz r7, De
    djnz r6, Del
    ret
end                        ; kết thúc chương trình dịch

```

11.4.2. Đồng hồ thời gian hiển thị với LED 7 đoạn và các vi mạch giải mã 7 đoạn và 74138

Vi điều khiển thường dùng làm đồng hồ đếm thời gian (xem tập một). Các giá trị về thời gian giây, chục giây, phút, chục phút, giờ, chục giờ được đếm tới 10 hoặc 6, có hiệu chỉnh thập phân rồi ghi trong các ô nhớ. Các số tự nhiên trong các ô nhớ phải được giải mã thành mã 7 đoạn rồi đưa ra hiển thị trên các đèn LED chục giờ, giờ, chục phút, phút, chục giây và giây. Để đếm được giây, chúng ta dùng Timer0 ở chế độ ngắt chương trình với đơn vị thời gian là 40 micro giây. Ví dụ 11.15 giới thiệu chương trình đếm và hiển thị trên LED 7 đoạn thời gian trên với các vi mạch giải mã 7447 và 74138.

; Ví dụ.11.15: Chương trình hiển thị đồng hồ thời gian trên các đèn LED 7 đoạn

<pre> MACRO @DEM <1, 2> mov a, l1 add a, #1 da a mov l1, a cjne a, #l2, NotoverFlow mov l1, #0 ENDMAC DisplayBuffer equ 30h ScanCounter equ 38h OneSecondCounter equ 39h Second equ 40h TSecond equ 41h Minute equ 42h TMinute equ 43h Hourse equ 44h THourse equ 45h org 00h ; bắt đầu chương trình chính jmp Reset org 0bh ; bắt đầu phục vụ ngắt Timer0Int jmp Timer0Int Reset: Mov SP, #70h ; vùng Stack Mov ScanCounter, #0; xoá ScanCounter Mov Hourse, #00h; nạp ô nhớ giờ = 12h Mov Minute, #0 ; xoá ô nhớ phút Mov Second, #0 ; xoá ô nhớ giây ***** ; use Timer 0, Mode 1 ; ngắt sau 4000 microsec ***** mov tmod, #01 ; chế độ 1 mov th0, #high (65536-4000) ; mov tl0, #low (65536-40000) ; setb rs0 clr rs1 mov ie, #10000010b ; cho ngắt Timer0 setb tr0 ; đóng mạch Timer0 Clock: @DEM Second, 10h @DEM TSecond, 6h @DEM Minute, 10h @DEM TMinute, 6h @DEM Hourse, 12h @DEM THourse, 00h NotOverFlow: ret \$1: call ScanDisplay ; gọi hiển thị sjmp \$1 ; chờ ngắt chương trình </pre>	<pre> Timer0Int: mov th0, #high (65536-4000) ; mov tl0, #low (65536-40000) ; Push acc ; lưu giữ thanh ghi a push psw setb rs0 clr rs1 djnz OneSecondCounter, NotOneSecond mov OneSecondCounter, #1000/4 call Clock NotOneSecond: pop psw pop acc reti ScanDisplay: mov r0, #40h Mov r1, #00h NotOver: mov a, @r0; đọc ô nhớ địa chỉ r1 vào a mov r2, a mov a, r1 swap a ; hoán vị 4 bit cao- thấp của a orl a, r2 ; orl a với @r1 mov P0, a ; đưa ra cổng P0 call delay inc r1 ; tăng r1 Inc r0 ; tăng r0 mov a, @r0 Cjne a, #7, NotOver ; so sánh @r0 với #7, ret ; trở về từ chương trình con Clock: @DEM Second, 10h @DEM TSecond, 6h @DEM Minute, 10h @DEM TMinute, 6h @DEM Hourse, 12h @DEM THourse, 00h NotOverFlow: ret Delay: mov r6, #40 Del: mov r7, #249 \$2: djnz r7, \$2 djnz r6, Del ret end ; kết thúc chương trình dịch </pre>
--	--

được nhận biết nếu được ấn đồng thời. Rollover N phím sẽ nhận tất cả các phím được nhấn đồng thời, từ phím đầu tiên tới phím cuối cùng.

Bảng 11.2. Mã lệnh của chế độ hoạt động

D7	D6	D5	Chức năng	Hành động
0	0	0	Xác lập chế độ	Chọn số vị trí hiển thị, điểm vào bên trái hay bên phải và loại quét bàn phím.
0	0	1	Nhịp (Clock)	Chương trình ghi Clock bên trong và xác lập thời gian quét và chống rung của phím
0	1	0	Đọc FIFO	Chọn loại đọc FIFO và địa chỉ đọc
0	1	1	Đọc hiển thị	Chọn loại đọc hiển thị và địa chỉ đọc
1	0	0	Ghi hiển thị	Chọn loại ghi hiển thị và địa chỉ ghi
1	0	1	Cấm ghi hiển thị	Cho phép nửa byte nhấp nháy
1	1	0	Xoá	Xoá hiển thị hay FIFO
1	1	1	Kết thúc ngắt	Xoá tín hiệu IRQ tới vi xử lý

Bảng 11.3. Các bit DD của lời điều khiển bộ chế độ

DD	Chức năng
00	Hiển thị 8 bit với lối vào bên trái
01	Hiển thị 16 bit với lối vào bên trái
10	Hiển thị 8 bit với lối vào bên phải
11	Hiển thị 16 bit với lối vào bên phải

Bảng 11.4. Các bit MMM của lời điều khiển bộ chế độ

MMM	Chức năng
000	Bàn phím được mã với khoá (lockout) 2 phím
001	Bàn phím được giải mã với khoá (lockout) 2 phím
010	Bàn phím được mã với quay vòng (rollover) N phím
011	Bàn phím được giải mã với phím quay vòng (rollover) N phím
100	Ma trận sensor được mã hoá
101	Ma trận sensor được giải mã
110	Bàn phím được chốt (strobe), quét hiển thị được mã hoá
111	Bàn phím được chốt (strobe), quét hiển thị được giải mã

b) Lệnh Clock dạng 001PPPPP

Lệnh này ghi chương trình cho bộ chia của nhịp bên trong. Mã PPPPP là một thang chia nhịp xung vào ổ chân (CLK) để có tần số mong muốn gần với 100kHz. Một nhịp xung vào 1MHz đòi hỏi 01010B cho PPPPP.

c) Lệnh Đọc FIFO dạng 010Z0AA

Lệnh này chọn địa chỉ của phím (keystroke) từ bộ đệm FIFO bên trong. Các bit vị trí AA chọn vùng của FIFO mong muốn từ 000 tới 111 và Z chọn tự tăng của địa chỉ. Ở chế độ hành động bình thường, lời điều khiển này chỉ được dùng với hành động của ma trận sensor của 8278.

d) Lệnh Đọc hiển thị, dạng 011ZAAAA

Lệnh này chọn địa chỉ đọc của một trong các vị trí của RAM hiển thị để đọc ra cổng dữ liệu. AAAA là địa chỉ của ô nhớ được đọc và Z chọn chế độ tự tăng. Lệnh này được dùng nếu thông tin được tích trong RAM hiển thị cần được đọc.

e) Lệnh Ghi hiển thị, dạng 100ZAAAA

Lệnh này chọn địa chỉ ghi dữ liệu cho hiển thị. AAAAA ghi địa chỉ để ghi dữ liệu từ cổng dữ liệu. Z chọn tự tăng, nghĩa là chuỗi ghi qua cổng dữ liệu là chuỗi vị trí của hiển thị.

f) Lệnh Cấm ghi hiển thị, dạng 1010WWBB

Lệnh này cấm ghi hoặc là nửa vùng của RAM hiển thị. W bên trái cấm ghi 4 bit bên trái của vùng RAM hiển thị. W bên phải cấm ghi 4 bit bên phải của vùng RAM hiển thị. Các chữ BB có chức năng tương tự, trừ chúng nhấp nháy (tắt) hoặc một nửa của các chân ra.

g) Lệnh Xóa, dạng 1100CCFA

Lệnh này xóa hiển thị, FIFO hoặc cả hai hiển thị và FIFO. Bit F xóa FIFO và trạng thái của RAM hiển thị và xác lập con trỏ địa chỉ về 000. Nếu các bit CC là 00 hay 01, tất cả các vùng của RAM hiển thị trở nên bằng 0000000, nếu CC = 10, tất cả các vùng trở nên bằng 00100000, và nếu CC = 11, tất cả các vùng trở nên bằng 11111111.

h) Lệnh kết thúc Ngắt, dạng 111E0000

Lệnh này được phát để xóa chân IRQ về 0 trong chế độ ma trận sensor. Nếu E bằng 1, chế độ lỗi đặc biệt được dùng. Trong chế độ lỗi đặc biệt này, thanh ghi trạng thái chỉ rằng có nhiều phím bị đóng đã xảy ra.

3. Chương trình hiển thị của vi mạch 8279

Ví dụ 11.16 giới thiệu chương trình hiển thị của 8279.

; Ví dụ 11.16: Chương trình hiển thị cho 1 LED 7 đoạn với 8279.	
Org 00h	
MASK equ 80h	; 10000000B, hiển thị đèn đầu tiên.
HTHI equ 30h	
Mov dptr, #201h	; địa chỉ 8279 cho lời điều khiển
Mov a, #00111110b	; clock của vi mạch
Mov @dptr, a	; ghi vào 8279
Mov a, #00h	; xác lập chế độ
Mov @dptr, a	
Mov a, #11000001b	; xóa hiển thị
Mov @dptr, a	
ORL, MASK	
Mov @dptr, a	; đưa điều khiển
Mov a, HTHI	
Dec dpl	; địa chỉ 200h của 8279 cho dữ liệu
Mov @dptr, a	
Sjmp \$	
end	

11.5. CÁCH GHÉP NỐI LED 7 ĐOẠN TIẾT KIỆM CỔNG CỦA VI ĐIỀU KHIỂN

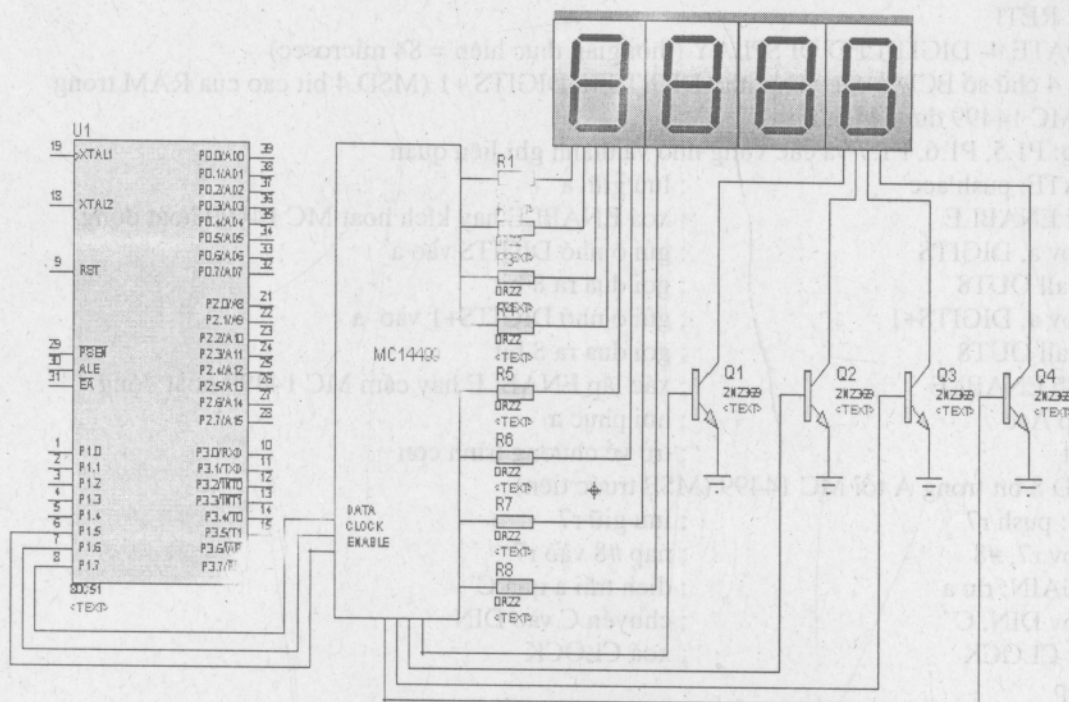
Ở phần trên, chúng ta đã tiết kiệm cổng vi điều khiển bằng dùng một cổng nối với hai vi mạch giải mã 7447 và 74138. Ở phần này, chúng ta xét cách ghép nối khác để tiết kiệm cổng, đó là dùng các vi mạch chuyên dùng và phương pháp địa chỉ để điều khiển LED.

11.5.1. Ghép nối LED 7 đoạn với vi mạch MC14499

1. Mắc vi mạch với LED 7 đoạn

Hãng Motorola đã chế tạo vi mạch MC 14499 để nối với 4 LED 7 đoạn. Trao đổi tin giữa vi điều khiển với MC 14499 theo kiểu nối tiếp (hình 11.15) qua các đường dây Data (chân số 5), xung nhịp qua đường dây Clock (chân số 13). Vi mạch được lựa chọn bằng đường dây /Enable (chân số 12). Như vậy, trong MC 14499 có:

- Bộ dao động bên trong mắc với tụ điện có dung lượng 0,15 μ F.
- Thanh ghi nhận tin nối tiếp từ lối vào Data để có tin song song.
- Bộ giải mã tin BCD thành mã 7 đoạn để đưa ra các lối ra a, b, c, d, e, f, g và dp.
- Bộ đếm tự động 4 bit cho các lối ra D1, D2, D3, D4 để điều khiển 4 đèn LED 7 đoạn.



Hình 11.15. Mạch hiển thị dùng MC14499

2. Chương trình hiển thị

Với mạch như hình 11.15, ta có chương trình hiển thị như ví dụ 11.17. Chương trình gồm:

- Các lệnh khởi phát chương trình chính.
- Chương trình phục vụ ngắt ngoài khi có tín hiệu đưa ra hiển thị bằng cách gọi chương trình con UPDATE.
- Chương trình con trao đổi 8 bit tín hiệu nối tiếp OUT8.

; Ví dụ 11.17: Chương trình hiển thị cho 4 LED 7 đoạn qua MC 14499.

DIN EQU P1.7

CLOCK EQU P1.6

ENABLE EQU P1.5

DIGITS DS 2

ICOUNT DS 1

Org 00h

```

Ljmp Main
Org 03h
Ljmp EXOISR
Main:
Mov ICOUNT, #50          ; nạp #50 vào ICOUNT
SETB EA                  ; cho phép ngắt toàn bộ
SETB EX0                  ; cho phép ngắt ngoài
SJMP $                   ; chờ ngắt
;*****
EXOISR: DJNZ ICOUNT, EXIT
      MOV ICOUNT, #50
      ACALL UPDATE
EXIT: RETI
; UPDATE 4- DIGIT LED DISPLAY (thời gian thực hiện = 84 microsec)
; Vào: 4 chữ số BCD ở các vùng nhớ DIGITS và DIGITS +1 (MSD 4 bit cao của RAM trong
; Ra: MC 14499 được hiển thị
; Dừng: P1.5, P1.6, P1.7 và các vùng nhớ và thanh ghi liên quan
UPDATE: push acc          ; lưu giữ a
      Clr ENABLE          ; xoá ENABLE hay kích hoạt MC 14499 hoạt động
      Mov a, DIGITS        ; gửi ô nhớ DIGITS vào a
      Acall OUT8           ; gọi đưa ra 8 bit
      Mov a, DIGITS+1      ; gửi ô nhớ DIGITS+1 vào a
      Acall OUT8           ; gọi đưa ra 8 bit
      Setb ENABLE          ; xác lập ENABLE hay cấm MC 14499 hoạt động
      Pop Acc              ; hồi phục a
      Ret                  ; trở về chương trình con
; SEND 8 bit trong A tới MC 14499 (MSB trước tiên)
OUT8: push r7             ; lưu giữ r7
      Mov r7, #8           ; nạp #8 vào r7
      AGAIN: rlc a          ; dịch trái a qua C
      Mov DIN, C            ; chuyển C vào DIN
      Clr CLOCK            ; xoá CLOCK
      Nop
      Nop
      Setb CLOCK           ; xác lập CLOCK
      Djnz r7, AGAIN        ; giảm r7, nếu khác 0, trở về AGAIN
      Pop r7               ; hồi phục r7
      Ret                  ; trở về chương trình con
end

```

11.5.2. Điều khiển LED 7 đoạn với đường dây địa chỉ

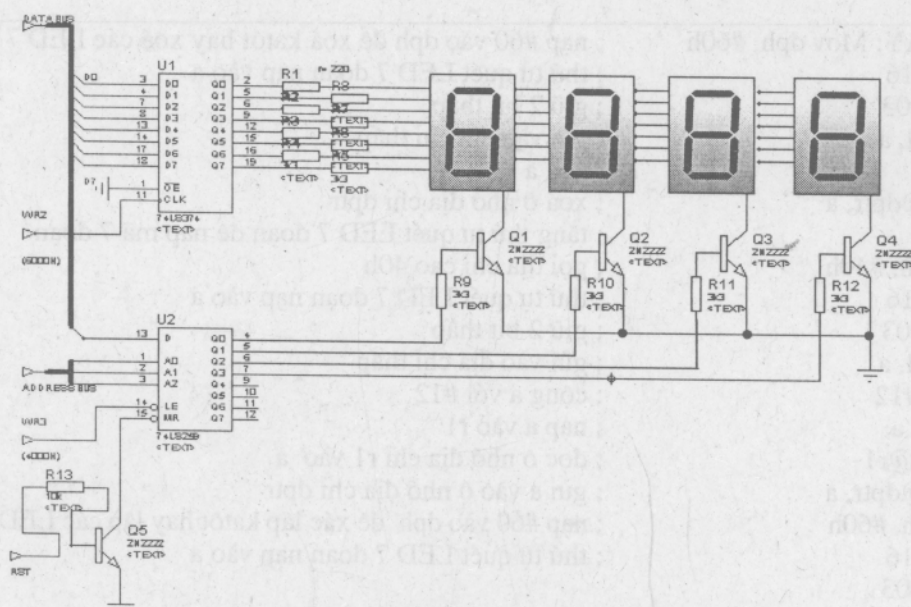
1. Mạch

Hình 11.16 giới thiệu cách điều khiển đèn LED 7 đoạn bằng các đường dây địa chỉ A0, A1 và A2.

Vì mạch thanh ghi đệm dữ liệu 74LS374 dùng để chốt dữ liệu mã 7 đoạn đưa ra các LED khi có địa chỉ vùng 4000h.

Vì mạch giải mã dữ liệu 74LS259 dùng để đưa tín hiệu điều khiển ra một trong các LED 7 đoạn khi có địa chỉ vùng 6000h và dữ liệu D0 = 1.

Tín hiệu RS dùng để xoá vi mạch 74LS259 khi ấn nút RESET của board vi điều khiển.



Hình 11.16. Sơ đồ hiển thị LED theo địa chỉ

2. Chương trình

Ví dụ 11.18 giới thiệu chương trình con điều khiển hiển thị 4 LED 7 đoạn của hình 11.16.

Chương trình hiển thị trên 4 đèn LED 7 đoạn nội dung của 4 ô nhớ có địa chỉ 12h, 13h, 14h và 15h. Các ô nhớ này đã chứa các mã 7 đoạn do giải mã 7 đoạn từ các ô nhớ chứa mã BCD.

Ô nhớ có địa chỉ 16h ghi thứ tự quét các LED 7 đoạn. Chương trình có 3 đoạn:

- Xóa katốt để xóa hay tắt LED 7 đoạn.
- Đưa mã 7 đoạn hay dữ liệu vào LED có địa chỉ 40xx tương ứng.
- Xác lập katốt để hiển thị dữ liệu vừa đưa ra địa chỉ 40xxh.

Mạch hiển thị trên đòi hỏi có bộ nhớ ngoài mà địa chỉ được gọi thông qua con trỏ dữ liệu DPTR với:

- Gọi địa chỉ $dptr = 400xh$ (trong đó $x = 0, 1, 2, 3$) từ lối ra giải mã WR2 để chốt dữ liệu 7 đoạn (hay xung CLOCK) từ đường dây dữ liệu vào vi mạch đệm 74LS374 bởi hai lệnh `MOV A, @R1` (với $R1 =$ địa chỉ nhớ bên trong chứa mã 7 đoạn) và `MOVX @DPTR, A`.

- Gọi địa chỉ $dptr = 600xh$ (trong đó $x = 0, 1, 2, 3$) từ lối ra của bộ giải mã WR3 để điều khiển tranzito mắc với các katốt của các đèn LED bởi hai lệnh `SETB Acc.0` và `MOVX @DPTR, A`.

Cách mắc với đường dây địa chỉ DPTR này để điều khiển các LED 7 đoạn này tuy tiết kiệm được một cổng của vi điều khiển nhưng phải tốn mạch giải mã các tín hiệu WR2, WR3 và thêm bộ giải mã địa chỉ 74LS259.

Chương trình con ở ví dụ 11.18 cũng tốn nhiều lệnh hơn để chuyển dữ liệu từ bộ nhớ trong ra bộ nhớ ngoài và đưa điều khiển các LED mắc với địa chỉ của bộ nhớ ngoài.

; Ví dụ 11.18: Chương trình hiển thị LED 7 đoạn điều khiển bằng đường dây địa chỉ

; địa chỉ vùng nhớ
; 12 : LED1 (mã 7 đoạn)
; 13 : LED2 (mã 7 đoạn)
; 14 : LED3 (mã 7 đoạn)
; 15 : LED3 (mã 7 đoạn)
; 16 : thứ tự quét LED 7 đoạn

DISPLAY: Mov dph, #60h	; nạp #60 vào dph để xoá katôt hay xoá các LED 7 đoạn
Mov a, 16	; thứ tự quét LED 7 đoạn nạp vào a
Anl a, #03	; giữ 2 bit thấp
Mov dpl, a	; gửi vào địa chỉ thấp
Clr a	; xoá a
Movx @dptr, a	; xoá ô nhớ địa chỉ dptr
Inc 16	; tăng thứ tự quét LED 7 đoạn để nạp mã 7 đoạn
Mov dph, #40h	; gọi địa chỉ cao 40h
Mov a, 16	; thứ tự quét LED 7 đoạn nạp vào a
Anl a, #03	; giữ 2 bit thấp
Mov dpl, a	; gửi vào địa chỉ thấp
Add a, #12	; cộng a với #12
Mov r1, a	; nạp a vào r1
Mov a, @r1	; đọc ô nhớ địa chỉ r1 vào a
Movx @dptr, a	; gửi a vào ô nhớ địa chỉ dptr
Mov dph, #60h	; nạp #60 vào dph để xác lập katôt hay lập các LED 7 đoạn
Mov a, 16	; thứ tự quét LED 7 đoạn nạp vào a
Anl a, #03	
Mov dpl, a	; gửi vào địa chỉ thấp
Setb acc.0	; xác lập bit acc.0
Movx @dptr, a	; gửi a có acc.0 = 1 vào ô nhớ có địa chỉ là dptr.
Ret	; trở về từ chương trình con
end	

11.6. HIỂN THỊ CHỮ/SỐ TRONG CÁC BIỂN QUẢNG CÁO

Trong đời sống, cần nhiều biển quảng cáo hiển thị các chữ và số. Người ta dùng các đèn LED 7 đoạn để hiển thị chữ và số bằng cách đọc dữ liệu trong các bộ nhớ ROM của hệ vi điều khiển. Để gây ấn tượng cho người quan sát, trong các biển quảng cáo này, các chữ hay số có thể nhấp nháy, hiện dần từ trái sang phải hay từ phải sang trái, chữ chạy từ trái sang phải hay chạy từ phải sang trái, hiện/tắt kiểu dồn từ hai đầu vào, từ giữa ra v.v ...

1. Hiện chữ/số nhấp nháy

Thông thường, các chữ/số hiện lên với thời gian trễ đủ lớn để mắt người có thể nhận biết và lặp lại với thời gian đủ nhỏ để mắt người nhận thấy như chữ hiện lên một cách liên tục.

Ngược lại, chúng ta thực hiện các chữ/số hiện lên nhấp nháy bằng cách bật sáng và tắt các đèn hiển thị với các thời gian trễ đủ lớn để mắt có thể thấy rõ các đèn tắt. Do đó, trong chương trình có thêm thời gian tắt đủ lớn (có thể bằng thời gian sáng hay khác đi).

Ví dụ 11.19 giới thiệu chương trình chớp/tắt chuỗi chữ *HANOI* trên 8 LED 7 đoạn với các thời gian sáng là DELAY và thời gian tắt là 3x delaytat.

; Ví dụ 11.19: Hiển thị nhấp nháy chuỗi chữ -HA-NOI-	
led7 equ P0	jnc maina; nhảy khi c = 0, sau 7 lần dịch phải
quét equ P2	djnz r4, main; lặp 250 lần
org 00h	lcall delaytat
mov dptr, #bangma ; địa chỉ đầu bảng	lcall delaytat
mainb: mov r4, #250	lcall delaytat
main: mov r3, #01111111b; điều khiển LED	ljmp mainb
mov r2, #00h ; biến của bảng	delay: mov r6, #01h
maina: mov a, r2	de2: mov r7, #50h
movc a, @a+dptr ; đọc bảng	djnz r7, \$
mov led7, a ; đưa ra hàng cho LED	djnz r6, de2

mov quet, r2 ; điều khiển LED 7 đoạn	ret
lcall delay ; thời gian sáng	delaytat: mov r6, #00h
mov quet, #0FFh	de2a: mov r7, #00h
mov led7, #0FFh tắt LED	djnz r7, \$
setb c ; xác lập c	djnz r6, de2a
inc r2 ; tăng địa chỉ bảng	ret
mov a, r3 ; r3 gửi vào a	bangma: db 0bfh, 089h, 088h, 0bfh,
rrc a ; dịch phải qua c	db 0c8h, 0c0h, 0f9h, 0bfh
mov r3, a ; đưa trở về r3	end

2. Hiện dẫn/tắt chữ/số từ trái sang phải

Thông thường, các chương trình hiển thị đều cho các chữ/số đều hiện dần từ trái sang phải, vì đọc bảng từ địa chỉ đầu tới cuối, theo thứ tự của chuỗi ghi trong bảng. Vì thời gian hiện một chữ quá ngắn, nhưng lặp lại nhiều lần đảm bảo thời gian cỡ 20msec nên mắt người quan sát thấy chữ hiện liên tục. Nếu cho thời gian trễ của chữ/số delay lớn lên tới mức lớn hơn, mắt người nhận biết được chữ/số hiện dần từng chữ từ trái qua phải theo thứ tự ghi trong bảng.

Ví dụ 11.20 giới thiệu chương trình chuỗi chữ *HANOI* trên 8 LED 7 đoạn chạy từ trái sang bên phải.

3. Hiện dần chữ/số từ phải sang trái

Nếu đọc chữ/số cuối của bảng trước rồi dần lên các chữ ở đầu bảng bằng cách:

- Cho địa chỉ dptr chỉ địa chỉ của bảng mã.
- Biến địa chỉ r2 = #7h để dptr+#r2 là địa chỉ của chữ/số ở cuối bảng để đọc chữ/số này trước.
- Cho địa chỉ giảm dần bằng lệnh djnz r2, maina để đọc các chữ/số tiếp theo tới khi r2 = #0, việc đọc sẽ lặp lại từ chữ cuối bảng.

Ví dụ 11.21 giới thiệu chương trình chuỗi chữ *HANOI* trên 8 LED 7 đoạn hiện lên từ bên phải với chữ cuối cùng (dấu -), chữ sát cuối (I) v.v... tới chữ đầu tiên bên trái (dấu -).

; Ví dụ 11.20: Chương trình hiện dần chữ -HA-NOI- từ trái sang phải	
led7 equ P0 quet equ P2 org 00h main: mov dptr, #bangma mov r4, #250 maina: mov r3, #11111110b mov r2, #00h setb c mainb: mov a, r2 movc a, @a+dptr mov led7, a mov quet, r3 lcall delay mov quet, #0FFh inc r2	mov a, r3 rrc a mov r3, a jc mainb djnz r4, maina ljmp main delay: mov r6, #01h de2: mov r7, #50h djnz r7, \$ djnz r6, de2 ret bangma: db 0bfh, 089h, 088h, 0bfh, 0c8h, 0c0h, 0f9h, 0bfh end

; Ví dụ 11.21: Chương trình chữ -HA-NOI- hiện dần từ phải sang trái	
led7 equ P0 quet equ P2 org 00h main: mov dptr, #bangma	mov a, r3 rlc a mov r3, a djnz r2, maina; giảm r2, về maina, nếu r2 khác 0

<pre> mov r4, #250 mainb: mov r3, #1111110b mov r2, #7 ; để đọc từ địa chỉ dptr + r2 maina: mov a, r2 movc a, @a+dptr mov led7, a mov quet, r3 lcall delay mov led7, #0FFh mov a, r3 </pre>	<pre> djnz r4, mainb sjmp main delay: mov r6, #01h de2: mov r7, #50h djnz r7, \$ djnz r6, de2 ret bangma: db 0bfh, 089h, 088h, 0bfh, 0c8h, 0c0h, 0f9h, 0bfh end </pre>
---	--

4. Hiện dẫn/tắt chữ/số kiểu dồn

Giống như LED đơn, ta có thể hiện dẫn chữ/số theo kiểu dồn từ ngoài vào hay từ giữa ra. Khi đó, ta phải lập bảng dữ liệu tương ứng theo yêu cầu hiển thị. Tại chương 12 chúng ta còn xét về các kiểu hiển thị động này cho ma trận diot và có thể áp dụng cho hiển thị động của LED đơn và LED nhiều thanh.

CÂU HỎI, BÀI TẬP VÀ THỰC HÀNH

I. Câu hỏi

- Đặc điểm của LED đơn.
- Đặc điểm của LED 7 đoạn.
- Sự cần thiết của hiển thị các dữ liệu trong các thiết bị điện tử số trong đó có vi điều khiển.
- So sánh sự hiển thị của LED đơn với LED 7 đoạn.
- Các cách mắc của LED đơn, LED 7 đoạn trong các thiết bị điện tử số.
- Các cách mắc nhiều LED nhiều thanh.
- Các cách mắc các LED nhiều thanh để tiết kiệm cổng của VĐK.
- Cách sử dụng vi mạch MC14499 cho trao đổi tín nối tiếp với 4 LED 7 thanh.
- Hiển thị động là gì? Kể và nêu nguyên tắc điều khiển các LED trong bảng quảng cáo.

II. Bài tập

- Thành lập bảng mã 7 đoạn cho các chuỗi chữ "Hoan nghênh quý khách tới tham trường CDSP".
- Thiết kế hệ hiển thị dùng 6 LED 7 thanh.
- Thiết kế hệ hiển thị dùng 8 LED 7 thanh với vi mạch MC1499.
- Thiết kế hệ hiển thị dùng 16 LED 7 thanh với vi mạch MC1499.
- Thiết kế hệ hiển thị dùng 32 LED 7 thanh với vi mạch MC1499.
- Thành lập bảng mã 7 đoạn cho các chuỗi chữ "Chúc mừng xuân 2007".
- Thành lập bảng mã 7 đoạn cho các chuỗi chữ "HOAN NGHENGH QUY KHACH TOI THAM TRUONG SPKT NAM DINH".
- Lập trình cho các ví dụ của chương trong PINNACLE.
- Lập trình cho các bài tập 3, 4, 5.
- Lập trình cho các bài tập 6, 7.

III. Thực hành

- Cho chạy mô phỏng các ví dụ của chương.
- Cho chạy mô phỏng các bài tập 3, 4, 5, 6 và 7.

CHƯƠNG 12

GHÉP NỐI VÀ ỨNG DỤNG ĐÈN LED MA TRẬN

Mục đích của chương:

- Hiểu được nguyên tắc hoạt động, cách mắc và lập trình cho một ma trận LED.
- Hiểu được nguyên tắc hoạt động, cách mắc và lập trình cho các vi mạch LED ma trận.
- Hiểu được nguyên tắc hoạt động, cách mắc và lập trình cho các LED ma trận điều khiển bằng xung và màu sắc khác nhau.
- Hiểu được cách mắc và lập trình cho hiển thị động trên các LED ma trận trong các biển quảng cáo.

Hiển thị bằng LED 7 hay 16 thanh có ưu điểm là hiển thị đơn giản, giá thành hạ, chương trình đơn giản, nhưng có nhược điểm là các số và chữ không đẹp, không tinh tế, độ phân giải không cao, đặc biệt có nhiều chữ khó hiển thị và chữ không rõ, dễ nhầm với các chữ khác.

Do đó, ma trận diot (dot matrix) đã được chế tạo ra để thay thế LED 7 đoạn, tuy rằng giá thành đắt hơn LED 7 đoạn. Đặc biệt là các vi mạch Led thông minh, có thể tích hợp tới 4 hay 8 ma trận LED được dùng rộng rãi trong các thiết bị số hiện đại, trong các bảng chỉ dẫn và các biển hiển thị dữ liệu, các bảng quảng cáo với các chữ và số màu sắc khác nhau với hiển thị động (dịch trái, dịch phải, chữ chạy v.v...).

12.1. CẤU TẠO, HOẠT ĐỘNG, CÁCH MẮC VÀ CÁC LOẠI MẠCH CỦA MA TRẬN LED

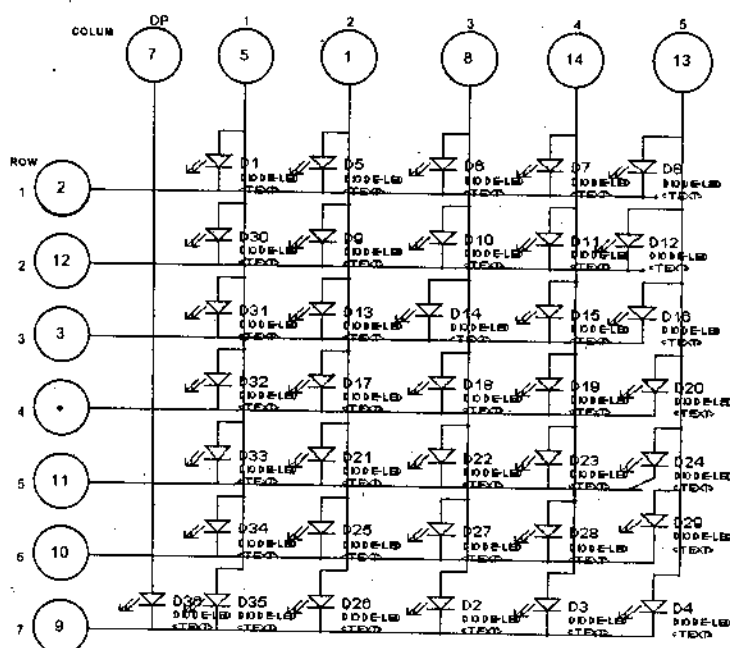
12.1.1. Cấu tạo ma trận LED

Khác với đèn LED 7 đoạn gồm 7 diot dưới dạng thanh mắc thành số 8, ma trận diot gồm nhiều diot kích thước nhỏ (được coi như một điểm sáng nên còn có tên diotled). Các diot điểm này thường được mắc thành dạng ma trận 5 x 7 (5 cột, 7 hàng), hay 8 x 8 kể cả một hàng và hai cột phân cách giữa các chữ hay số.

Hình 12.1 giới thiệu một vi mạch ma trận diot TIL 305 của hãng Texas Instrument.

Vi mạch bên có các chỉ số như sau:

- Các số ngoài vòng tròn là các chỉ số cột hay hàng.
- Các số trong vòng tròn là các chỉ số chân của vi mạch.
- DP (Decimal Point - chấm thập phân) để chỉ các số sau dấu chấm này là số thập phân.



Hình 12.1. Ma trận diot TIL 305 5 x 7

- Với TIL 308 có DP ở bên trái.
- Với TIL 309 có DP ở bên phải.
- Các vi mạch này chuyên dùng để hiển thị các số nguyên và thập phân.

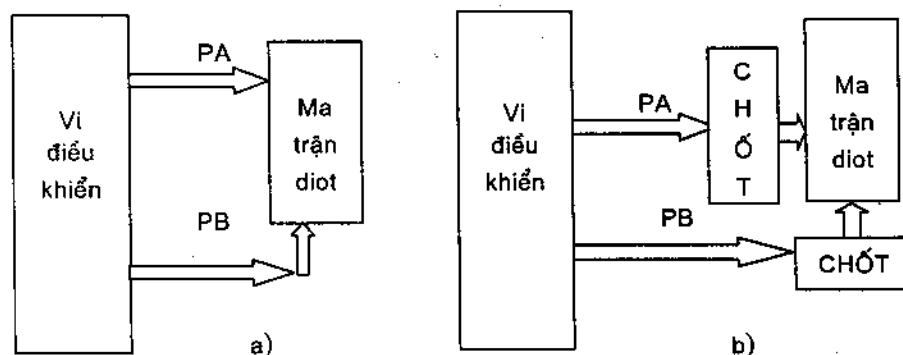
Như vậy, một ma trận 5x7 diot có :

- 5 cột C1 – C5 với các tín hiệu vào điều khiển logic 1.
- 7 hàng R1 – R7 với các lối vào điều khiển logic 0.

Mỗi diot sẽ có dòng điện chạy qua hay sẽ phát sáng khi các lối vào C1 – C5 có mức logic là 1 (cỡ 3,8 tới 5V) và khi một trong các lối vào hàng có mức logic 0.

12.1.2. Cách mắc ma trận LED

Từ cấu tạo và nguyên tắc hoạt động ở trên, ta phải có cách mắc các Led của ma trận thích hợp. Với mạch hiển thị có một ma trận LED được nối với VDK qua hai cổng song song, các cột của LED mắc với cổng cột C1 – C5 và các hàng của ma trận mắc với cổng hàng R1 – R7. Có thể mắc trực tiếp các cổng của VDK với các cổng cột, hàng của ma trận hoặc qua các vi mạch chốt 74LS374 và giải mã có chốt 74LS 259.



Hình 12.2. Mạch điện nối ma trận LED: không có (a) và có thanh ghi chốt (b)

Hình 12.2. giới thiệu nguyên tắc mắc một ma trận diot với VDK:

- Các đường dây cột nối với một cổng của VDK, ví dụ, P0 và các tín hiệu đưa ra điều khiển có mức logic 1.
- Các đường dây hàng nối với cổng khác, ví dụ, P2 và các tín hiệu đưa ra điều khiển diot sáng có mức logic 0.

Hình 12.3 giới thiệu cách mắc ma trận diot với VDK 8052:

- Các anốt của các diot được nối với nhau thành các đường dây cột của ma trận LED, được nối với các tranzito khi có tín hiệu ra ở mức logic 1.
- Các hàng của ma trận LED được nối với nhau thành các đường dây hàng của ma trận LED, mắc với cổng P2 qua điện trở hạn chế dòng cỡ 200 ôm.

- Nếu cần chỉ thị DP, lối ra P0.5 sẽ nối với chân DP (chân số 7 của vi mạch TIL 305).

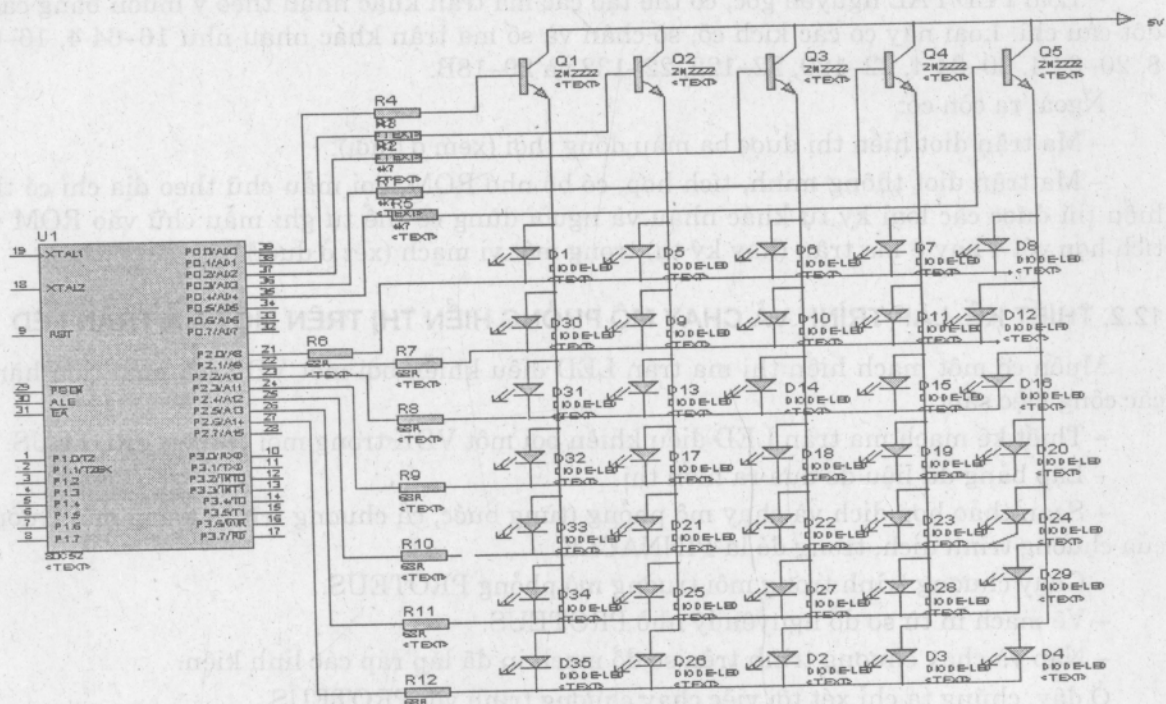
Nếu điều khiển như trên mà các diot chưa đủ sáng hay quá chói, ta phải điều chỉnh các điện trở hạn chế dòng mắc nối tiếp cho sáng vừa đủ để quan sát.

12.1.3. Hoạt động của ma trận LED

Một diot nằm ở cột thứ i và hàng thứ j sẽ phát sáng khi có đồng thời hai điều kiện:

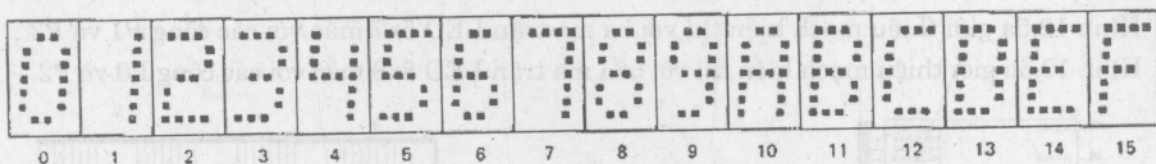
- Có tín hiệu ra anốt theo cột $C_i = 1$.
- Có tín hiệu ra theo hàng $R_j = 0$ để đóng mạch diot hay có dòng điện chạy qua diot từ anốt sang katốt.

Nói khác đi, diot tại toạ độ ij chỉ sáng khi có dòng chạy qua hay có đồng thời (trùng phùng) điện thế logic 1 ở hàng PAi và điện thế logic 1 ở cột PBj.



Hình 12.3. Cách mắc ma trận diot 5 cột x 7 hàng với VĐK.

Hình 12.4 giới thiệu các diot sáng của các ma trận diot của các ký tự (chữ), các số HEX từ 0 tới F.



Hình 12.4. Các số HEX biểu diễn trên ma trận diot 5 x 7

12.1.4. Các loại mạch ma trận LED

Người ta đã chế tạo các vi mạch ma trận LED với kích thước khác nhau và màu sắc khác nhau và có thể có 1, 4 hay 8 LED ma trận trên một vi mạch.

Có nhiều hãng điện tử chế tạo và bán các vi mạch ma trận diot nhưng nổi tiếng nhất là ba hãng Texas Instrument, Hewlet Packart và Siemen.

Vi mạch TIL 305 (hình 12.1) hiển thị màu đỏ, có dòng điện qua các diot là 10mA tới 100mA với ma trận 5 x 7 (5 cột, 7 hàng).

Trong thực tế, người ta hay dùng các ma trận 8 x 8 với 3 cột và 1 hàng trống, không sử dụng, để phân cách các số hay ký tự trong một dòng và nhiều dòng số/chữ.

Các mạch TIL dùng để hiển thị chữ số HEX và thập phân, có chấm sáng ở bên trái (TIL 308) hay bên phải (TIL 309), có kích thước ma trận (cỡ chữ số) 7,62mm (TIL 305), 6,9mm (TIL 308 và 309) có bên trong các mạch chốt, giải mã và đệm ra (driver). Hình 12.4 giới thiệu hiển thị các số HEX.

Trong chương trình mô phỏng PROTEUS có các loại ma trận diot sau:

- Loại 5 x 7 có 4 màu là đỏ, xanh, xanh lá cây và da cam.
- Loại 8 x 8 có 4 màu là đỏ, xanh, xanh lá cây và da cam.

– Loại PLD/PAL nguyên gốc, có thể tạo các ma trận khác nhau theo ý muốn bằng cách đốt cầu chì. Loại này có các kích cỡ, số chân và số ma trận khác nhau như 16–64 4, 16–64 8, 20–64 4, 20–84 4, 22–120, 22–122, 22–132 và 29–18B.

Ngoài ra còn có:

- Ma trận diot hiển thị được ba màu đồng thời (xem ở dưới).
- Ma trận diot thông minh, tích hợp, có bộ nhớ ROM, gọi mẫu chữ theo địa chỉ có thể hiển thị được các loại ký tự khác nhau và người dùng có thể tự ghi mẫu chữ vào ROM và tích hợp với 4 hay 8 ma trận (hay ký tự) trong một vi mạch (xét ở dưới).

12.2. THIẾT KẾ, LẬP TRÌNH VÀ CHẠY MÔ PHỎNG HIỂN THỊ TRÊN MỘT MA TRẬN LED

Muốn có một mạch hiển thị ma trận LED điều khiển bởi một VĐK, ta phải tiến hành các công việc sau:

- Thiết kế mạch ma trận LED điều khiển bởi một VĐK trong môi trường PROTEUS.
- Lập bảng dữ liệu để đưa ra hiển thị.
- Soạn thảo hợp dịch và chạy mô phỏng (từng bước, cả chương trình) trong môi trường của chương trình dịch, trong đó là PINNACLE.
- Chạy chương trình trong môi trường mô phỏng PROTEUS.
- Vẽ mạch in từ sơ đồ nguyên lý nhờ PROTEUS.
- Nạp và chạy chương trình trên sơ đồ mạch in đã lắp ráp các linh kiện.

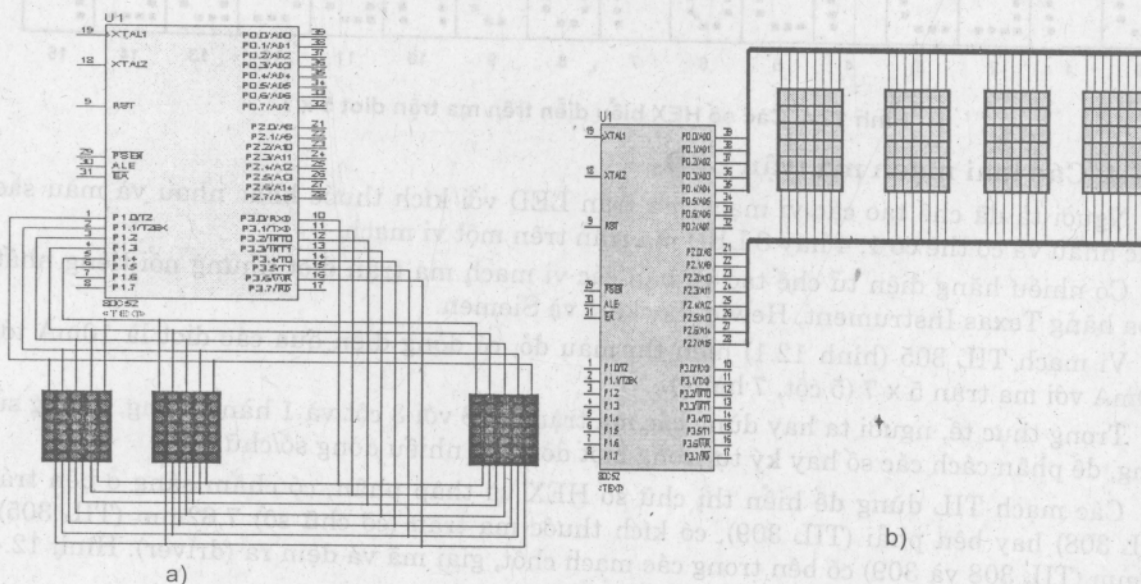
Ở đây, chúng ta chỉ xét tới việc chạy chương trình với PROTEUS.

12.2.1. Thiết kế mạch hiển thị ma trận LED

Với chương trình mô phỏng PROTEUS, ta có thể vẽ được sơ đồ nguyên lý và sơ đồ lắp ráp mạch hiển thị trên một ma trận LED.

Hình 12.5a giới thiệu mạch hiển thị với ba ma trận LED 5x7 mắc với các cổng P1 và P3.

Hình 12.5b giới thiệu mạch hiển thị với bốn ma trận LED 8x8 mắc với các cổng P0 và P2.



Hình 12.5. Mạch hiển thị ba ma trận LED 5x7 (a) và bốn ma trận LED 8x8 (b)

Chú ý: Khi chạy chương trình, thấy không hiển thị đúng chữ mong muốn là do nối sai dây nối với các cổng, ta phải nối lại dây sau khi xoay vi mạch đi 90 độ, đổi đường dây (dùng mũi tên ngang) hay đảo chiều (dùng mũi tên đứng) ở thanh công cụ số 3 của PROTEUS.

12.2.2. Lập bảng dữ liệu cho hiển thị

Các chữ số HEX được biểu diễn dưới dạng ma trận LED sáng như ở hình 12.4. Muốn hiển thị các ký tự hay số tự nhiên gồm nhiều chữ số, ta phải lập mô hình hiển thị của các diot trên giấy bằng cách điền các giá trị của ma trận với:

- Giá trị 0 cho diot tắt.
- Giá trị 1 cho diot sáng 1 theo mô hình mong muốn.

Ví dụ, ta muốn ma trận diot sáng hình chữ H. Từ đó, ta có bảng với 7 hàng và 5 cột tương ứng như bảng 12.1.

Bảng 12.1. Bảng hiển thị chữ H

Cột Hàng	C1	C2	C3	C4	C5
R1	1	0	0	0	1
R2	1	0	0	0	1
R3	1	0	0	0	1
R4	1	1	1	1	1
R5	1	0	0	0	1
R6	1	0	0	0	1
R7	1	0	0	0	1
Hàng	7FH	08H	08H	08H	7FH

Có hai cách trình bày dạng dữ liệu của bảng:

– *Dạng HEX*: mỗi phần tử của bảng là một số dạng HEX tương ứng với giá trị HEX của một hàng cho mỗi cột có địa chỉ là một giá trị của DPTR. Như vậy, mỗi hàng có 5 phần tử cho 5 cột và mỗi hàng của dạng bảng này phản ánh dữ liệu của một ma trận hay của một ký tự hay chữ số.

Các giá trị 7FH, 08H, 08H, 08H, 7FH chính là dạng HEX của bảng dữ liệu cho chữ H.

Cách biểu diễn bảng theo dòng ma trận này ngắn hơn, số dòng của bảng chính là số ma trận hay số ký tự hoặc chữ số của dữ liệu và chữ H được biểu diễn theo hình thẳng đứng, giống như khi hiển thị trên các LED của ma trận.

Như vậy, sau mỗi dòng hay sau 5 cột của một ký tự hoặc chữ số, địa chỉ của cột tăng lên 5 đơn vị. Trong lập trình, chúng ta phải chú ý tới đặc điểm này để chuyển sang phần tử của cột của ma trận LED kế bên.

– *Dạng nhị phân*: mỗi phần tử của bảng là một số dạng nhị phân (0 hay 1). Tại mỗi hàng hay mỗi dòng có 7 (8 kể cả phần tử phân cách luôn luôn tắt) giá trị nhị phân tương ứng với điểm sáng hay tắt tại vị trí đó. Với mỗi dòng dữ liệu này tương ứng với một cột có 7 phần tử của một hàng. Với ma trận 5 cột, cần 5 dòng dữ liệu để biểu diễn một ký tự hay một số.

Bảng dữ liệu dạng nhị phân này dài, vì cần tới 5 dòng cho 5 cột hay nói khác đi, mỗi ký tự hay chữ số cần tới 5 dòng của bảng và chữ H được biểu diễn theo kiểu nằm ngang.

Cách biểu diễn này có ưu điểm là đơn giản trong lập trình vì chỉ cần tăng địa chỉ (bởi lệnh INCDPTR) cho các dữ liệu tiếp theo, kể cả phần tử của đèn ma trận LED kế bên.

Đôi khi trong bảng dữ liệu còn thêm dòng đầu và cuối có các phần tử có giá trị 0FFH.

Việc kết thúc bảng dữ liệu người ta thường đưa vào thêm một phần tử hay cả dòng có giá trị 00H (hay một giá trị quy ước) để kết thúc việc đọc bảng bằng lệnh MOVC A, @A+DPTR.

Có thể có cách lập bảng khi đưa đồng thời các tín hiệu của 5 cột, còn quét 8 tín hiệu

theo các hàng. Cách này tốn thời gian quét (8 lần cho 8 hàng). Do đó, người ta dùng cách quét cho 5 cột để tiết kiệm thời gian quét.

Bảng các mã để hiển thị thường được ghi ở cuối chương trình hay có thể ghi ở một địa chỉ bất đầu riêng (với khai báo org xxxh và bangma:) ở trong chương trình hay trong một bộ nhớ ROM ngoài. Chương trình hiển thị bảng chính là chương trình đọc bảng trên với lệnh `MOVC A, @A + DPTR`.

Một phần tử của bảng dữ liệu của LED 7 đoạn tương ứng với 5 hay 8 phần tử của bảng dữ liệu của LED ma trận. Do đó, bảng dữ liệu của LED ma trận này sẽ dài và phức tạp hơn vì có số phần tử bằng $5 \times 7 \times$ số ký tự, trong khi bảng dữ liệu cho LED 7 đoạn có số phần tử bằng số ký tự.

Chú ý:

– Với văn bản chữ thường tiếng Việt, không phải chữ in, nếu có chữ hoa hay các chữ k, h, l, v.v... (cần hai ma trận chồng lên nhau), các ký tự này phải được hiển thị trên hai ma trận chồng nhau và ma trận ở mỗi hàng chỉ dùng biểu diễn một nửa ký tự đó.

– Với văn bản chữ thường tiếng Việt, không phải chữ in, nếu có chữ có đuôi xuống dòng dưới, ví dụ chữ g, f, y, v.v... cũng cần hai ma trận chồng lên nhau để hiển thị tương tự chữ hoa.

12.2.3. Lập trình hiển thị một ma trận LED

Chương trình hiển thị dữ liệu cho một ma trận LED chính là chương trình đưa các mã hay các dữ liệu của số hay ký tự ra cổng nối với hàng và cột của ma trận theo thiết kế đã có (cách mắc ma trận với các cổng của VDK).

Có ba cách đưa dữ liệu ra cổng:

- Đưa trực tiếp theo từng hàng và quét 5 cột cho ma trận 5×7 .
- Đưa theo trực tiếp theo ma trận và quét 5 cột cho ma trận 5×7 .
- Đưa liên tiếp theo cách đọc bảng dữ liệu lần lượt theo từng hàng và quét 5 cột cho ma trận 5×7 .

Do đó, có ba dạng chương trình hiển thị.

1. Chương trình hiển thị dạng đưa dữ liệu trực tiếp theo từng hàng

Chương trình dạng này đưa lần lượt 5 giá trị mã của hiển thị theo hàng và quét cột với giá trị logic 1 (dùng lệnh dịch trái hay phải 5 lần cho 5 cột), sau đó có thời gian trễ để quan sát và cần xóa hiển thị để chống lém (thanh sáng mờ mờ, tắt không tối hẳn).

Ví dụ 12.1a giới thiệu chương trình đưa hiển thị chữ H ra ma trận mắc với cổng P1 và P3 của VDK.

Ví dụ 12.1b giới thiệu chương trình viết dưới dạng lệnh gộp, đưa hiển thị chữ H ra ma trận mắc với cổng P1 và P3 của VDK.

; Ví dụ 12.1a: Chương trình hiển thị dữ liệu đưa mã trực tiếp theo từng hàng	Ví dụ 12.1b: Chương trình hiển thị dữ liệu khi đưa mã trực tiếp theo từng hàng, viết theo lệnh gộp
<pre> ORG 00H LJMP MAIN ORG 30H MAIN: MOV P3, #7FH MOV P1, #01H CALL DELAY ; gọi trễ MOV P1, #00H; tắt cột, chống lém MOV P3, #08H MOV P1, #02H </pre>	<pre> MACRO @HIENH <1,2> MOV P1, #2 MOV P3, #14 CALL DELAY ; gọi trễ MOV P3, #00H; tắt cột, chống lém ENDMAC ORG 00H LJMP MAIN ORG 30H </pre>

CALL DELAY MOV P1, #00H MOV P3, #08H MOV P1, #04H CALL DELAY MOV P1, #00H MOV P3, #08H MOV P1, #8H CALL DELAY MOV P1, #00H MOV P3, #7FH MOV P1, #10H CALL DELAY MOV P1, #00H SJMP MAIN DELAY: MOV R7, # 5 DELO: MOV R6, # 200 DEL1: DJNZ R6, DEL DJNZ R7, DELO RET END	MAIN: @HIENH 3, 7FH, 1, 01H ; cổng P3, cổng P1 @HIENH 3, 08H, 1, 02H @HIENH 3, 08H, 1, 04H @HIENH 3, 08H, 1, 08H @HIENH 3, 7FH, 1, 10H SJMP MAIN DELAY: MOV R7, # 5 DELO: MOV R6, # 200 DEL1: DJNZ R6, DEL DJNZ R7, DELO RET END <i>Chú ý:</i> - Chương trình dùng lệnh gộp 12.1b ngắn gọn hơn chương trình viết theo từng lệnh ở 12.1a. - Có thể dùng chương trình dùng lệnh gộp trên cho bất kỳ cách mắc nào vì chỉ cần thay tên cổng mắc với VĐK. - Có thể thay đổi thời gian trễ sao cho thích hợp với sự quan sát ký tự hiển thị bằng mắt.
--	--

2. Chương trình hiển thị dạng đưa dữ liệu trực tiếp theo từng ma trận

Với các ký tự có dữ liệu cố định theo hàng ngang hay cột thẳng đứng, ví dụ, chữ H, I, E, F, L, T, U ta có thể đưa dữ liệu theo hàng và cột kiểu ma trận.

Ví dụ 12.2 giới thiệu chương trình đưa dữ liệu kiểu ma trận cho chữ H với 8 hàng và 8 cột.

Ta có thể dùng chương trình trên để hiển thị các chữ khác kiểu ma trận đã kê ở trên.

3. Chương trình hiển thị dạng đọc bảng dữ liệu

Chương trình loại này có hai động tác cơ bản là đọc bảng và đưa hiển thị dữ liệu.

a) Đọc bảng dữ liệu

Với bảng LED ma trận, các phần tử của bảng được đọc vào thanh ghi chứa A bằng lệnh MOVC A, @A + DPTR trong đó:

- Đối với bảng dạng nhị phân: cho A = 0, DPTR tăng dần cho tới giá trị dptr (với dph, dpl) cuối.

- Đối với bảng dạng HEX:

+ Trong một hàng: DPTR = không đổi, A = 1 tới 5.

+ Giá trị của hàng sau: DPTR + 5 (hay 8 nếu có các cột của phần tử phân cách).

; Ví dụ 12.2: Chương trình hiển thị theo kiểu ma trận dữ liệu	
ORG 30H LOOP: MOV P2, #00000000B MOV P1, #11001100B CALL DELAY_1S MOV P2, #11100111B MOV P1, #11111100B CALL DELAY_1S LJMP LOOP	DELAY_1S: MOV R1, #1 LOOP1: MOV R2, #5 LOOP2: MOV R3, #255 LOOP3: DJNZ R3, LOOP3 DJNZ R2, LOOP2 DJNZ R1, LOOP1 RET END

Địa chỉ cuối của bảng = địa chỉ đầu + số dòng của bảng dạng HEX.

hay DPTR cuối = bangma + số dòng của bảng dạng HEX.

Có hai loại bảng : ngắn (DPH = 00H, DPL < 255D) và dài (DPH khác 0).

Trong chương trình đọc bảng theo địa chỉ tăng dần, kết thúc việc đọc bảng theo một trong hai cách sau:

- Số dòng của bảng dạng HEX.
- Địa chỉ cuối của bảng tính theo công thức trên.

b) Hiển thị dữ liệu

Sau khi đọc dữ liệu của bảng vào thanh ghi chứa A, việc hiển thị có hai hành động:

- Đưa dữ liệu ở A ra cổng song song nối với hàng của LED ma trận.
- Đưa tín hiệu điều khiển cột của LED ma trận để LED có hàng và cột tương ứng phát sáng.

Chương trình bảng đọc bảng dữ liệu này gồm các hành động sau:

- Gán địa chỉ đầu bảng mã cho DPTR (bằng lệnh MOV DPTR, #bangma).
- Tính số phần tử bảng hay địa chỉ cuối của bảng để kết thúc đọc bảng.
- Đọc các phần tử của bảng bằng lệnh MOVC A, @A + DPTR theo DPTR tăng dần hay A tăng dần cho tới khi kết thúc (theo số dòng hay địa chỉ cuối).
- Sau mỗi lần hiển thị một cột đều có thời gian trễ thích hợp (bằng lệnh LCALL DELAY).
- Trước khi đọc phần tử tiếp theo bởi vòng lặp, phải tăng địa chỉ DPTR và giảm số lần đọc.
- Kết thúc đọc bảng giảm số phần tử về 0 hay so sánh địa chỉ DPTR với địa chỉ DPTR cuối.

- Bảng mã được kê khai sau chương trình chính hay riêng rẽ (để ghi trong ROM rời).

So với hiển thị đèn LED 7 đoạn, chương trình hiển thị của ma trận LED có các đặc điểm sau:

- Đưa hiển thị dữ liệu đồng thời ra các hàng và quét lần lượt từng cột (từ trái sang phải hoặc từ phải sang trái) theo bảng mã (mã theo PB, bảng 11.1).
- Hoặc đưa hiển thị đồng thời các cột và quét lần lượt từng hàng theo bảng mã theo hàng (theo PA), cách này không dùng vì tốn thời gian quét (7 lần so với 5 lần ở trên).
- Lệnh so sánh địa chỉ (gửi vào a) với địa chỉ cuối là *cjne a, cuoi, lap* để đọc phần tử tiếp theo của bảng hay kết thúc việc đọc bảng mã.
- Lệnh đọc một phần tử của bảng là *movc a, @a+dptr*.
- Giữa mỗi lần đưa hết mã của cả ma trận LED, sẽ lại lặp lại một số lần (thường 250 lần) với thời gian trễ cỡ 20ms, để mắt có thể nhìn với cảm giác cả ma trận sáng một cách liên tục khi chưa hiển thị hình ma trận mới.
- Có thể có loại ma trận LED nhiều màu điều khiển bởi các cột đỏ cotd, cột xanh cotx, nên có thể hiển thị ma trận theo màu xanh, đỏ hay da cam (điều khiển đồng thời hai cột cotd, cotx).

- Có các loại ma trận LED kích thước nhỏ (ma trận điểm) hoặc rất lớn (biển quảng cáo) hay hộp chữ gồm nhiều ma trận diot mắc nối tiếp. Khi đó, yêu cầu có sự phân kênh cho nhiều lối ra cho các cột nếu quét theo cột (hoặc các hàng nếu quét theo các hàng) và chương trình đưa ra là chương trình đọc nối tiếp của một bảng mã có dung lượng lớn.

Do đó:

- Địa chỉ của mỗi hàng sẽ tăng lên 5 cho 5 cột. Nếu kể cả cột trống giữa các ký tự, mỗi hàng có 8 cột cho 5 cột chữ/số và 2 cột cho khoảng cách giữa các chữ/số.

– Địa chỉ *dptr* cho phần tử cuối cùng của bảng sẽ là địa chỉ đầu + số phần tử của bảng hay *bangma + số phần tử của bảng*. Chúng ta cần tính địa chỉ này để kết thúc việc đọc bảng bởi lệnh so sánh *dptr* với địa chỉ của phần tử cuối cùng của bảng.

Ví dụ 12.3 giới thiệu chương trình hiển thị trên một LED ma trận với lệnh gộp cho các số tự nhiên (từ 0 tới 9).

Ví dụ 12.4 giới thiệu chương trình hiển thị trên một LED ma trận cho các ký tự từ A tới G và dùng ngắt thời gian.

Cả hai chương trình của các ví dụ 12.3 và 12.4 có điểm chung là:

- Bảng mã được kê khai theo từng hàng, mỗi hàng có 5 phần tử cho 5 cột.
- Lệnh ghi địa chỉ đầu bảng mã *movc dptr, #bangma*.
- Công thức tính địa chỉ cuối của bảng là *cuoi = bangma + số dòng của bang ma*.
- Cần đưa 5 (hay 7 hoặc 8 nếu kể cả cột phân cách các ký tự hàng cho 5 cột cho một ký tự hay số vì mỗi ký tự hay số có 5 cột).
- Thời gian trễ có thể điều chỉnh để dễ quan sát.
- Số lần lặp lại hay thời gian lặp lại có thể dùng vòng lặp hay ngắt thời gian.

; Ví dụ 12.3: Chương trình hiển thị số tự nhiên	
<pre>org 00h ljmp main hang equ P0 ; hàng nối cổng P0 cotd equ P2 ; cột nối cổng P0 cuoi equ 30h ; địa chỉ chứa địa chỉ cuối main: mov cotd, #00h main0: mov dptr, #bangma mov a, # bangma add a, #35 ; số mã của bảng (5x7) mov cuoi, a ; địa chỉ cuối maina: mov r2, #250 ; số lần lặp cho một chữ mainb: mov r1, #00h ; số phần tử cho một chữ mov r3, #00000001b; điều khiển cột mainc: mov a, r1 movc a, @a+dptr ; đọc bảng mov hang, a ; đưa ra hàng mov cotd, r3 ; điều khiển cotd call delay ; gọi trễ mov cotd, #00 ; xoá cột inc r1 ; tăng cột cho chữ mov a, r3 rl a ; dịch điều khiển cột</pre>	<pre>mov r3, a cjne r1, #5, mainc ; so với 5 cột djnz r2, mainb ; lặp lại r2 lần mov a, dpl add a, #5 mov dpl, a ; địa chỉ mới của hàng mov a, dpl cjne a, cuoi, maina ljmp main Delay: mov r5, #1; delay time = 5msec Del: mov r6, #50 djnz r6, \$ djnz r5, Del ret bangma: db 0c1h, 0bah, 0b6h, 0aeh, 0c1h, 0ffh; mã số 0 db 0eeh, 0deh, 080h, 0feh, 0feh, 0ffh; mã số 1 db 0b0h, 0b6h, 0b6h, 0b6h, 086h, 0ffh; mã số 2 db 0b6h, 0b6h, 0b6h, 0b6h, 080h, 0ffh; mã số 3 db 087h, 0f7h, 0f7h, 080h, 0f7h, 0ffh; mã số 4 db 086h, 0b6h, 0b6h, 0b6h, 0b0h, 0ffh; mã số 5 db 080h, 0b6h, 0b6h, 0b6h, 0b0h, 0ffh; mã số 6 end</pre>

12.2.4. Hợp dịch và chạy mô phỏng cho chương trình hiển thị

Các chương trình ở ví dụ 12.3 và 12.4 được:

- Đưa vào chương trình PINNACLE để hợp dịch và chạy thử từng bước và cả chương trình.
- Chạy mô phỏng trong PROTEUS theo các bước sau:

: Ví dụ 12.4. Chương trình hiển thị trên ma trận LED các ký tự A, B, C, D, E, F, G	
<pre> ScanLinecounter equ 30h hang equ P0 ; hàng nối cổng P0 cotd equ P2 ; cột nối cổng P0 cuoi equ 40h ; ô nhỏ chứa địa chỉ cuối org 00h ljmp main org 0bh jmp Timer0Int org 100h mov ScanLinecounter, #00h mov dptr, #bangma mov cotd, #00000001b; d, khiến cột mov tmod, #02h mov ie, #10000010b mov th0, #256-200 mov tl0, #256-250 setb tr0 Loop2: mov r3, #7 Loop1: mov r5, #25 call delay mov a, dpl add a, #5 mov dpl, a mov a, dph adde a, #0 mov dph, a djnz r3, Loop1 jmp Loop2 ; chương trình ngắt thời gian TimerInt0: push acc push psw </pre>	<pre> mov hang, #0ffh inc ScanLinecounter clr c mov a, ScanLinecounter xrl a, #5 jnz X1 mov ScanLinecounter, a cpl c X1: mov a, cotd rlc a mov cotd, a mov a, ScanLinecounter movc a, @a+dptr ; đọc bảng mov hang, a ; đưa ra hàng pop psw pop acc reti sjmp main Delay: ; r5 x (20mS) mov r6, #40; delay time = 5msec Del: mov r7, #249 djnz r7, \$ djnz r6, Del djnz r5, Delay ret bangma: db 03h, 6dh, 6eh, 6dh, 03h ; A db 049h, 036h, 036h, 00h, 00h ; B db 05dh, 03eh, 03eh, 03eh, 041h ; C db 063h, 05dh, 03eh, 03eh, 00h ; D db 03eh, 03ch, 03ch, 03ch, 00h ; E db 07eh, 07ch, 07ch, 07ch, 00h ; F db 05h, 036h, 036h, 03eh, 041h ; G end </pre>

- + Vào chương trình LabCenterELECTRONIC/PROTEUS PROFESSIONAL.
- + Vào chương trình ISIS của PROTEUS.
- + Vào thư mục BIN để mở sơ đồ MOTDOT.ISIS đã thiết kế như ở hình 12.5 cho ma trận 5x7 hay 8x8. Chú ý là tên các cổng VDK phải đúng với cổng của hàng và cột trong chương trình.
- + Vào giao diện Edit Component để tìm chương trình cần chạy theo các bước sau:
 - Nháy chuột phải vào sơ đồ, sơ đồ sẽ chuyển từ màu đen sang màu đỏ.
 - Nháy chuột trái vào sơ đồ, sơ đồ sẽ chuyển từ màu đỏ sang giao diện Edit Component cần tìm. Nếu nháy nhầm chuột phải, sơ đồ sẽ bị xoá mất, phải nháy chuột trái vào biểu tượng Undo (mũi tên ngược chiều kim đồng hồ) để có lại sơ đồ.

- Nháy chuột trái vào mục Program File để tìm tên của chương trình trong PINNACLE và nháy tên của chương trình có đuôi.HEX.
- Nháy chuột trái vào ô OK.
- Nháy chuột trái ở vùng ngoài sơ đồ để chuyển từ màu đỏ sang đen.
- + Chạy mô phỏng chương trình: nháy chuột trái vào nút tam giác của thanh công cụ số 4.

Quan sát chữ hiển thị trên ma trận LED. Nếu không đúng, xem lại sơ đồ có mắc đúng tên cổng và chương trình có viết đúng không? Lập lại các động tác trên cho tới khi hiển thị đúng như mong muốn.

+ Nháy vào nút hình chữ nhật trên thanh công cụ số 4 để dừng chạy mô phỏng chương trình với sơ đồ đã thiết kế.

Chú ý là một sơ đồ có thể chạy nhiều chương trình (viết đúng cho chương trình) cũng như một chương trình có thể chạy với nhiều sơ đồ (thiết kế đúng như chương trình).

12.3. ĐIỀU KHIỂN HIỂN THỊ THEO MÀU SẮC VÀ THEO XUNG TÍN HIỆU

12.3.1. Điều khiển hiển thị theo màu sắc

Hiện nay, người ta đã chế tạo trong một ma trận diot, có chứa hai ma trận diot màu xanh và màu đỏ với các hệ chân ra cotx, cotd khác nhau.

Hình 12.6 giới thiệu hai ma trận 8x8 diot hai màu có katôt và anôt chung.

Như vậy, mỗi vi mạch, có hai ma trận diot cho hai màu cơ bản là đỏ và xanh lá cây.

Nếu nối các cột của các ma trận này với các lối ra điều khiển từ lối ra của cổng song song của vi điều khiển và tùy chương trình điều khiển, các ma trận diot hiển thị số hay chữ với màu khác nhau. Mỗi ma trận diot có thể có ba màu là đỏ (điều khiển cotd), xanh (điều khiển cotx) và da cam (điều khiển đồng thời hai cột cotd, cotx) hay màu da cam là tổng hợp của hai màu đỏ và xanh lá cây.

Có thể cho hiển thị hai nửa ký tự với :

- Nửa trên xanh bằng cách điều khiển màu xanh.
- Nửa dưới đỏ bằng cách điều khiển màu đỏ.
- Nửa dưới da cam bằng cách điều khiển cả hai màu đỏ và xanh lá cây.

Ví dụ 12.5 giới thiệu chương trình điều khiển bằng chữ CDSPKT màu đỏ (điều khiển cotd), rồi lập lại chương trình với điều khiển cột cotx để đèn màu xanh.

Ví dụ 12.6 giới thiệu chương trình điều khiển bằng chữ CDSPKT màu da cam với điều khiển đồng thời cả hai cột cotd, cotx.

Ví dụ 12.7 giới thiệu chương trình điều khiển hiển thị các chữ hai màu, nửa trên màu đỏ với 4 hàng của cột (4 bit cao với lệnh ORRL A, #0F0H), nửa dưới màu xanh cho 4 bit thấp của hàng với lệnh ORL A, #0Fh.

Từ các ví dụ về hiển thị màu cơ bản trên, ta có thể hiển thị các màu phức tạp khác, dành cho người dùng tự viết như những bài tập ứng dụng.

12.3.2. Điều khiển cột với tín hiệu xung

Một số loại ma trận LED trước kia được điều khiển theo mức điện áp với lệnh điều khiển:

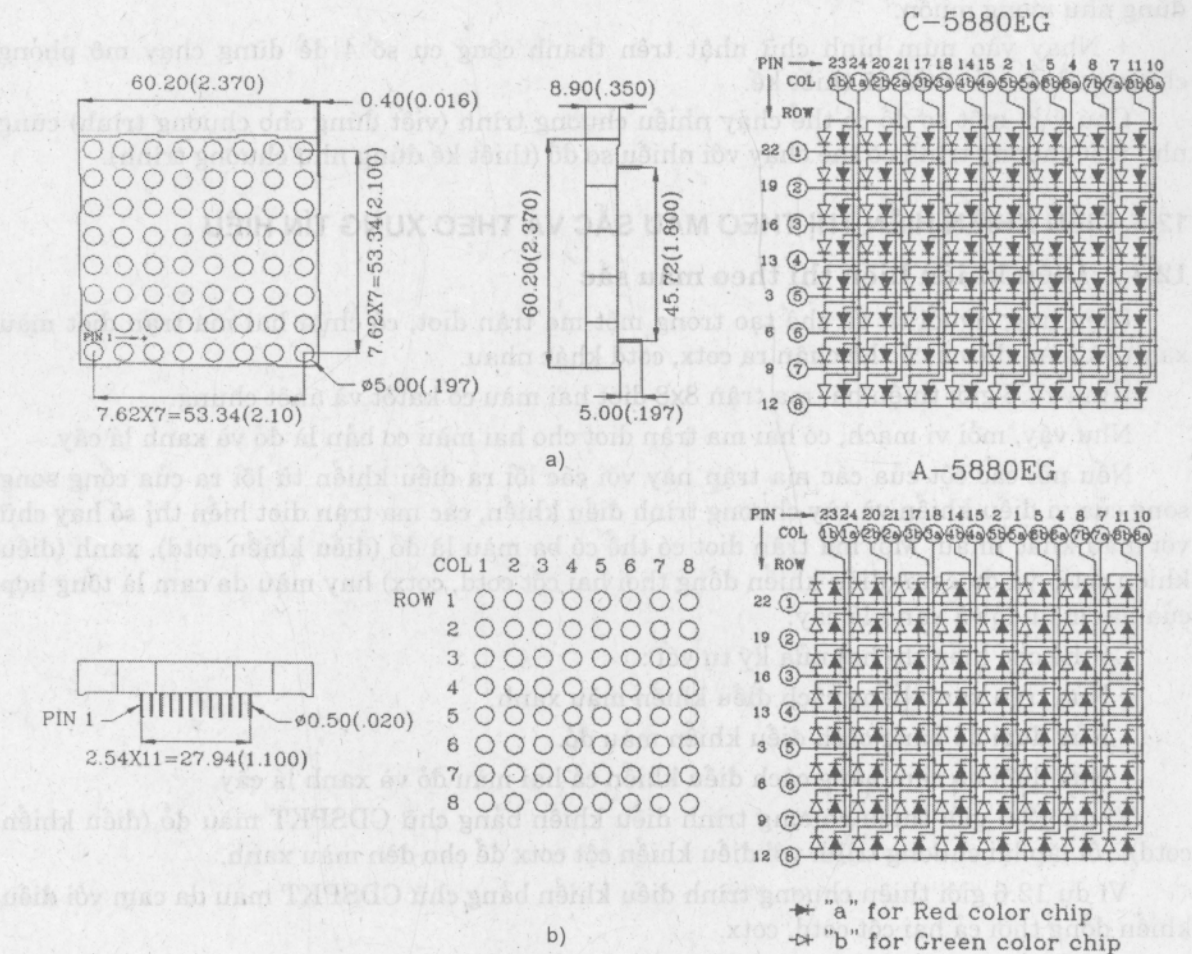
MOV COTD A, trong đó A có giá trị xác định.

Một số loại ma trận LED được chế tạo gần đây được điều khiển theo sườn của tín hiệu (xung) với sự xoá (CLR bit) hay xác lập (SETB bit) của một bit điều khiển nối với cổng ra

song song của vi điều khiển. Muốn vậy, trước khi đưa tín hiệu điều khiển với lệnh điều khiển ta phải xoá hay xác lập để chuẩn bị sườn xung thích hợp. Khi đưa sườn xung điều khiển, chỉ cần xác lập hay xoá bit đó.

Ví dụ 12.8 giới thiệu chương trình điều khiển ma trận LED theo sườn của xung điều khiển.

Trong các chương trình điều khiển LED ma trận màu, có thể có loại LED ma trận điều khiển bằng sườn tín hiệu điều khiển.



Hình 12.6. Sơ đồ chân và nguyên lý của ma trận diot 8x8 hai màu có katốt (a) và anốt (b) chung

; Ví dụ 12.5: Chương trình SPKT xanh, đỏ	
hang equ p3 ; hàng kết nối với port 3	main:
cotx equ p0 ; cột xanh kết nối với port 0	@hthi maina, mainb, mainc, cotdo; hien do
cotd equ p2 ; cột đỏ kết nối với port 2	@hthi maini, mainh, maing, cotx; hien xanh
macro @hthi <1, 2, 3, 4>	sjmp main
hthil4: mov dptr, #bangma	delay: mov r5, #1
l1: mov r2, #1	de: mov r6, #50
l2: mov r1, #00	djnz r6, \$
mov r3, #00000001b ; ma quet	djnz r5, de
l3: mov a, r1	ret

<pre> movc a, @a+dptr mov hang, a mov l4, r3 lcall delay mov l4, #00h inc r1 ; tăng lên để lấy byte kế mov a, r3 ; dịch chuyển mã quét rl a mov r3, a cjne r1, #7, l2 djnz r2, l3 mov a, dpl add a, #8 mov dpl, a inc dptr mov a, dpl cjne a, #0a1h, l1 endmac org 00h mov cotx, #0 ; tắt quét nếu có kết nối mov cotd, #0 ; tắt quét nếu có kết nối </pre>	<pre> bangna: db 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh; du lieu tat db 0C1H, 0BEH, 0BEH, 0BEH, 0DDH, 0ffh ; ma chu C db 080H, 0BEH, 0BEH, 0BEH, 0C1H, 0ffh ; ma chu D db 0CDH, 0B6H, 0B6H, 0B6H, 0D9H, 0ffh ; ma chu S db 080H, 0BEH, 0B7H, 0B7H, 0CFH, 0ffh ; ma chu P db 080H, 0E7H, 0DBH, 0BDH, 0FEH, 0ffh ; ma chu K db 0BFH, 0BFH, 080H, 0BFH, 0BFH, 0ffh ; ma chu T db 080H, 0DFH, 0EFH, 0F7H, 080H, 0ffh ; ma chu N db 080H, 0BEH, 0BEH, 0BEH, 0C1H, 0ffh ; ma chu D db 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh ; du lieu tat end </pre>
<pre> ; Ví dụ 12.6: Chương trình hiển thị chuỗi SPKT màu da cam hang equ p3 ; hàng kết nối với port 3 cotx equ p0 ; cột xanh kết nối với port 0 cotd equ p2 ; cột đỏ kết nối với port 2 org 00h mov cotx, #0 ; tắt quét nếu có kết nối mov cotd, #0 ; tắt quét nếu có kết nối main: mov dptr, #bangma maina: mov r2, #1 mainb: mov r1, #00 mov r3, #00000001b ; mã quét mainc: mov a, r1 movc a, @a+dptr mov hang, a mov cotd, r3 ; điều khiển cột đỏ mov a, dpl cjne a, #0a1h, maina sjmp main delay: mov r5, #1 de: mov r6, #50 djnz r6, \$ djnz r5, de ret bangna: db 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh ; du lieu tat db 0C1H, 0BEH, 0BEH, 0BEH, 0DDH, 0ffh ; ma chu C db 080H, 0BEH, 0BEH, 0BEH, 0C1H, 0ffh ; ma chu D </pre>	<pre> mov cotx, r3 ; điều khiển cột xanh lcall delay mov cotd, #00h; xoá cột đỏ mov cotx, #00h; xoá cột xanh inc r1 ; tăng lên để lấy byte kế mov a, r3 ; dịch chuyển mã quét rl a mov r3, a cjne r1, #7, nainb djnz r2, mainc mov a, dpl add a, #8 mov dpl, a inc dptr db 0CDH, 0B6H, 0B6H, 0B6H, 0D9H, 0ffh ; ma chu S db 080H, 0BEH, 0B7H, 0B7H, 0CFH, 0ffh ; ma chu P db 080H, 0E7H, 0DBH, 0BDH, 0FEH, 0ffh ; ma chu K db 0BFH, 0BFH, 080H, 0BFH, 0BFH, 0ffh ; ma chu T db 080H, 0DFH, 0EFH, 0F7H, 080H, 0ffh ; ma chu N db 080H, 0BEH, 0BEH, 0BEH, 0C1H, 0ffh ; ma chu D db 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh ; du lieu tat end </pre>

; Ví dụ 12.7: Chương trình hiển thị chuỗi SPKT, 4 hàng trên đỏ, 4 hàng dưới xanh	
<pre> lap equ 100 hàng equ P3 cotx equ P0 cotd equ P2 org 00h mov cotx, #00h mov cotd, #00h main4: mov r0, #00h main3: mov r2, #250 main2: mov r1, 00h mov r3, #8 mov r4, #1 main1: mov a, r1 movc a, @a+dptr main0: push acc orl a, #0fh mov hàng, a mov cotd, r4 lcall delay mov cotd, #00h pop acc orl a, #0f0h mov hàng, a ljmp main4 </pre>	<pre> mov cotx, r4 lcall delay mov cotx, #00 mov a, r4 rl a mov r4, a inc r1 djnz, main1 inc r1 djnz r3, main1 djnz r2, main2 inc r0 cjne r0, #40h, main3 delay: mov 75h, #1 del2: mov 76h, #50 del1: djnz 76h, del1 djnz 75h, del2 ret bangma: db 0b3h, 06dh, 06dh, 06dh, 09bh, 0ffh; chữ S db 001h, 0edh, 0edh, 0edh, 0f3h, 0ffh; chữ P db 001h, 0efh, 0d7h, 0bbh, 07dh, 0ffh; chữ K db 0fdh, 0fdh, 001h, 0fdh, 0fdh, 0ffh; chữ T end </pre>
; Ví dụ 12.8: Chương trình điều khiển đọc và hiển thị số từ 0 - 9 kiểu xung	
<pre> cot equ hang equ p0 org 00h ljmp main main: mov dptr, #100h nhan2: mov r2, #150 nhan1: mov r1, #00 setb pl.1 ; tín hiệu quét nhan0: clr pl.0 mov a, r1 movc a, @a+dptr mov hàng, a setb pl.0 ; tạo xung call delay10ms clr pl.1 inc r1 cjne r1, #5, nhan0 djnz r2, nhan1 mov r4, #05 </pre>	<pre> cjne a, #7, nhan2 jmp main x1: inc dptr djnz r4, x1 mov a, dpl bangma: db 0c1h, 0bah, 0b6h, 0aeh, 0c1h ; ma so 0 db 0eeh, 0deh, 080h, 0feh, 0feh ; ma so 1 db 0b0h, 0b6h, 0b6h, 0b6h, 086h; ma so 2 db 0b6h, 0b6h, 0b6h, 0b6h, 080h; ma so 3 db 087h, 0f7h, 0f7h, 080h, 0f7h ; ma so 4 db 086h, 0b6h, 0b6h, 0b6h, 0b0h ; ma so 5 db 080h, 0b6h, 0b6h, 0b6h, 0b0h ; ma so 6 db 0bfh, 0b7h, 0b7h, 080h, 0f7h ; ma so 7 db 080h, 0b6h, 0b6h, 0b6h, 080h ; ma so 8 db 086h, 0b6h, 0b6h, 0b6h, 080h ; ma so 9 delay10ms: mov r5, #20 del1: mov r6, #050h djnz r6, \$ djnz r5, del1 ret end </pre>

12.4. MA TRẬN LED TÍCH HỢP VÀ THÔNG MINH

Hiện nay, nhu cầu sử dụng các ma trận LED rất cao ở các thiết bị số, ở công sở, ở các biển quảng cáo. Do đó có nhiều hãng điện tử chế tạo và bán các vi mạch ma trận LED tích hợp và thông minh.

Gọi là tích hợp vì trên một vi mạch có thể chứa tới 4 hay 8 ma trận LED đơn, có cả các bộ nhớ RAM, nhớ ROM, mạch điều khiển ra công suất (driver).

Gọi là vi mạch thông minh vì không phải lập bảng, đọc bảng và đưa hiển thị dữ liệu của một ma trận các số tự nhiên (từ số 0 tới 9), các ký tự (từ A tới Z) và các ký hiệu khác mà trong bộ nhớ ROM của vi mạch LED có sẵn bộ ký tự, số tự nhiên và các ký hiệu khác. Chỉ cần đưa vào mã ASCII, là địa chỉ của bộ nhớ ROM, sẽ đọc ra mã của ma trận LED để hiển thị. Trên lưng của vi mạch ma trận LED có các LED hiển thị trực tiếp nội dung cần hiển thị. Nói khác đi, vi mạch ma trận LED tích hợp có cả mạch giải mã, mạch điều khiển và các diot LED hiển thị.

Dưới đây, chúng ta sẽ trình bày ba loại vi mạch tích hợp và thông minh trên là vi mạch có 1 ma trận LED, 4 ma trận LED và 8 ma trận LED.

12.4.1. Ma trận LED tích hợp và thông minh một ma trận LED

Vi mạch này chỉ có một ma trận LED, chỉ hiển thị một ma trận, cho một ký tự hay chữ số.

Chúng ta giới thiệu ba loại ma trận LED đơn thông minh của hãng Siemen.

1. Vi mạch PDP

Loại này có hai loại:

- PDP 1165, có màu đỏ, hiệu suất cao.
- PDP 1167, có màu xanh, hiệu suất rất cao.

Vi mạch này là một ma trận 8x8 có thể lập trình cho các cột và lời điều khiển (qua các đường dây địa chỉ A0 – A3), bên trong có chứa bộ nhớ RAM và mạch Driver đưa tín hiệu ra.

Chân của vi mạch như bảng 12.2.

Sơ đồ khối của vi mạch như hình 12.7.

Bảng 12.2. Chân của PDP 1165/1167

Rn				
1.	RST	Resets the System Active low,	11. V _{CC}	Flus 5 volts power pin
2.	CLK _{OUT}	Clock output for daisy chaining	12. D0	Data Bus Bit 0 (LSB)
3.	WR	Writes data into the display. Active low.	13. D1	Data Bus Bit 1
4.	CE	Chip Enable- Active low,	14. D2	Data Bus Bit 2
5.	A0	Address Input (LSB)	15. D3	Data Bus Bit 3
6.	A1	Address Input	16. D4	Data Bus Bit 4
7.	A2	Address Input (MSB)	17. D5	Data Bus Bit 5
8.	A3	Address input for control words.	18. D6	Data Bus Bit 6
9.	CLK _{IN}	Clock Input far daisy chaining	19. D7	Data Bus Bit 7 (MSB)
10.	R _{DIM}	Controls bringhtness through R _{EXT}	20. GND	Ground

Vi mạch PDP 1165/1167 khi dùng phải nối với:

- Vi mạch nhớ và giải mã địa chỉ cao A4 – A15 của VDK để đưa tín hiệu chọn địa chỉ vào /CE.
- Các địa chỉ A0 – A3 nối với các đường dây địa chỉ của VDK để chọn cột cho ma trận LED.

– Các đường dây dữ liệu D0 – D7 nối với cổng ra của VDK (cần một cổng) hay qua vi mạch chốt (74LS373).

Sơ đồ khối (hình 12.7) có các khối sau:

– *Khối logic vào (input logic)*: gồm các khối đệm dữ liệu, logic điều khiển và logic giải mã địa chỉ.

– *Khối nhớ hiển thị (display memory)*: khối nhớ RAM 8x8 bit cho cột và hàng và thanh ghi điều khiển.

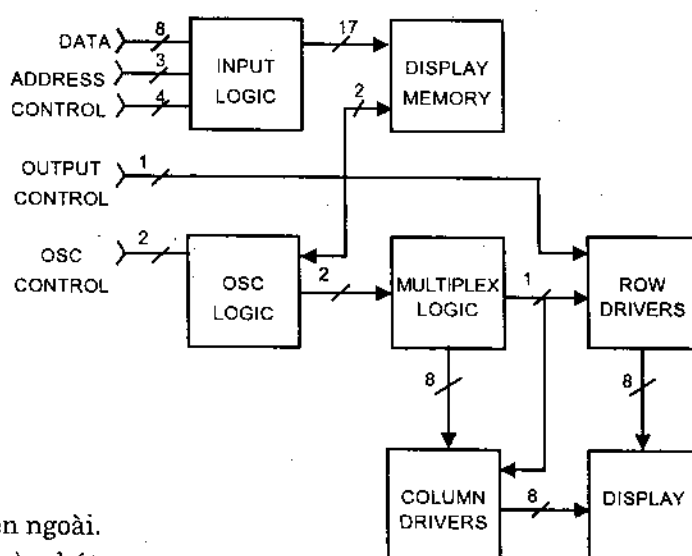
– *Khối logic dao động (OSC)*: phát nhịp cho dao động bên trong và bên ngoài.

– *Logic dồn kênh (logic multiplex)*: phát sơ đồ dồn kênh cho driver cột và hàng, điều khiển độ sáng, nhấp nháy.

– *Driver cột (column driver)*: điều khiển cột và độ cao của mỗi LED.

– *Driver hàng (driver row)*: điều khiển, có R_{dim} điều khiển độ sáng.

– *Hiển thị (display)*: gồm 64 LED nối thành 8 nhóm để tạo nên cột và hàng.



Hình 12.7. Sơ đồ khối của PDP-1165/67

2. Vi mạch DLR và DLG

Vi mạch này có các loại sau:

- DLR5735 màu đỏ, katốt chung.
- DLR5736 màu đỏ, anốt chung.
- DLG 5735 màu xanh, katốt chung.
- DLR5736 màu xanh, anốt chung.

Các vi mạch trên đều là chữ đơn có chiều cao 0,69" (17,5mm), không có drivers CMOS ở bên trong.

Các vi mạch trên đều có 12 chân, dùng mã ASCII và EBCDIC.

3. Vi mạch DLO, DLG

Có các loại :

- DLO4135/7135 màu đỏ hiệu suất cao, chữ đơn, DLO4135/7 cao 0,43" còn DLO7135 cao 0,68".
- DLG4137/7137 màu xanh lá cây.

Chúng đều là ma trận 5x7 thông minh, có bộ nhớ, giải mã và driver ở bên trong.

Bộ chữ gồm 96 mã ASCII, có chữ thường và chữ hoa và có bộ phát chữ.

Vi mạch trên có 16 chân với tên như ở bảng 12.3.

Bảng 12.3. Chân của DLO/DLG Pin Functions - DL04135/DLG4137

Pin	Function	Pin	Function
1	$\overline{LT}$ Lamp	9	D0 Data LSB
2	WR Write	10	D1 Data
3	BL1 Brightness	11	D2 data
4	BL0 Brightness	12	D3 Data
5	No Pin	13	D4 data
8	No Pin	14	D5 Data
1	$\overline{CE}$ Chip Enable	15	D6 Data MSB
8	I GND	16	+Vcc

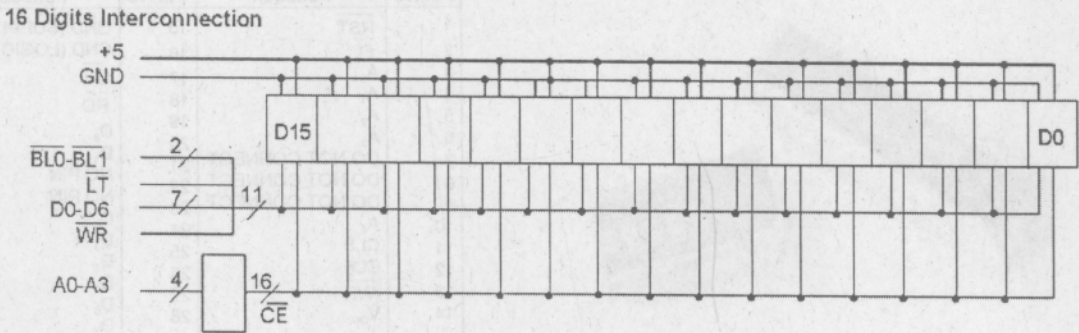
Có mạch dồn kênh và driver, có 4 mức điều khiển độ sáng (điều khiển bởi /BL1, /BL0) và thủ đèn (/LT).

Tương thích với đường dây của VĐK, chọn vi mạch bởi /CE và ghi dữ liệu song song bởi tín hiệu lệnh /WR của VĐK.

Có thể lựa chọn theo tọa độ X/Y.

Chỉ cần một nguồn nuôi 5V.

Hình 12.8 giới thiệu cách nối 16 ma trận để hiển thị 16 chữ số.



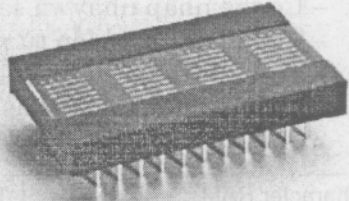
Hình 12.8. Sơ đồ nối 16 chữ số dùng vi mạch DLO/DLG

12.4.2. Ma trận LED tích hợp và thông minh bốn ma trận LED

1. Loại HDSP

Nhóm vi mạch này có 4 chữ 5x7, cao 6,9mm (0,27") có các loại sau:

- HDSP-2490 màu đỏ.
- HDSP-2491 màu vàng.
- HDSP-2492 màu đỏ hiệu suất cao.
- HDSP-2493 màu xanh lá cây hiệu suất cao.



2. Vi mạch loại SCF

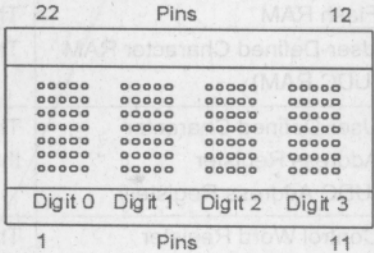
Có các loại sau:

- SCF 5740 chuẩn màu đỏ.
- SCF 5742 chuẩn màu đỏ, hiệu suất cao.
- SCF 5744 chuẩn màu xanh lá cây, hiệu suất cao.

Các vi mạch trên có chiều cao 0,270", gồm 4 ma trận, có lối vào nối tiếp.

Hình dáng của các vi mạch SCF như hình 12.9.

Vi mạch có thể mắc nối tiếp với cổng nối tiếp của VĐK hay qua một bit của cổng song song và nối tầng nhiều vi mạch SCF.



Hình 12.9. Hình dáng của các vi mạch SCF

12.4.3. Ma trận LED tích hợp và thông minh tám ma trận LED

1. Vi mạch loại HDSP-2100

- Có các loại:
- HDSP-210X
 - HDSP-211X
 - HDSP-250X như ở bảng 12.4.

Bảng 12.4. Các vi mạch loại HDSP-2100

	ALGaAs	High Efficiency			
Font Height	Red	Red	Orange	Yellow	Green
0,2 inches	HDSP-2107	HDSP-2112	HDSP-2110	HDSP-2111	HDSP-2123
0,27 inches	HDSP-2504	HDSP-2502	HDSP-2500	HDSP-2501	HDSP-2503

Hình dáng và tên các chân của các vi mạch loại này như ở hình 12.10.

Pin.No	Function	Pin No.	Function
1	RST	15	GND (SUPPLY)
2	FL	16	CND (LOGIC)
3	A ₀	17	CE
4	A ₁	18	RD
5	A ₂	19	D ₀
6	A ₃	20	D ₁
7	DO NOT CONNECT	21	NO PIN
8	DO NOT CONNECT	22	NO PIN
9	DO NOT CONNECT	23	D ₂
10	A ₄	24	D ₃
11	CLS	25	D ₄
12	CLK	26	D ₅
13	WR	27	D ₆
14	V _{DD}	28	D ₇

Hình 12.10. Hình dáng (a) và tên các chân (b) của HDSP-2100

Tính năng chủ yếu của HDSP-2100 như sau:

- Có thể lắp ráp (Stackable) theo phương X (HDSP-21XX) và phương X, Y (HDSP-250X).
- Có giải mã ASCII 128 ký tự chứa trong ROM với bảng mã như của LED ma trận đơn thông minh hay 4 ma trận LED thông minh.
- Có các chức năng định chương trình.
- Người dùng có thể xác định 16 ký tự (ngoài bảng mã ASCII).
- Có thể nhấp nháy và sáng theo nhiều mức khác nhau.
- Thiết bị điều khiển từ xa.
- Thử được thiết bị.
- Có bộ nhớ RAM với đặc tính như ở bảng 12.5.

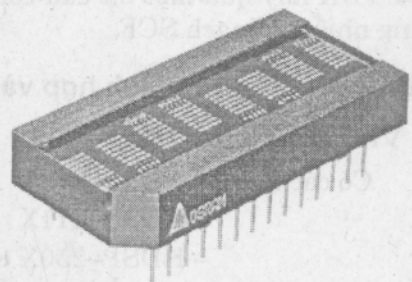
Bảng 12.5. Đặc tính của nhớ RAM của HDSP-21000

Character RAM	This RAM stores either ASCII character data or a UDC; RAM address.
Flash RAM	This is a 1 x 8 RAM which stores Flash data.
User-Defined Character RAM (UDC RAM)	This RAM stores the dot pattern for custom characters.
User-Defined Character Address Register (UDC Address Register)	This register is used to provide the address to the UDC RAM when the user is writing or reading a custom character.
Control Word Register	This register allows the user to adjust the display brightness, flash individual characters; blink, self test.; or clear the display.

2. Vi mạch loại HDP-2000

Loại này có các vi mạch sau:

- HDSP2110S màu đỏ.
- HDSP2111S màu vàng.
- HDSP2112S màu đỏ, hiệu suất cao.
- HDSP2113S màu xanh lá cây.
- HDSP2114S màu xanh lá cây, hiệu suất cao.
- HDSP2115S màu da cam.



Hình 12.11. Hình dáng vi mạch HDSP

Các vi mạch trên có chữ kích thước 0,200", ma trận 5x7, có lối vào song song.

Hình dáng vi mạch như ở hình 12.11.

Có thể quan sát được ký tự hiển thị cách 2,5m.

Các tính năng khác của vi mạch loại này cũng tương tự vi mạch loại HDSP-2100 như có nhớ RAM, nhớ ROM (bộ mã 128 mã ASCII) có thể ghi 16 ký tự của người dùng, có thể lập trình cho hoạt động như độ sáng, nhấp nháy v.v... với thanh ghi điều khiển.

12.5. HỆ HIỂN THỊ TRÊN NHIỀU MA TRẬN LED

12.5.1. Các phương án mắc mạch hiển thị nhiều ma trận LED

Một hệ hiển thị chữ hay số đều phải dùng nhiều ma trận LED để hiển thị cho từng chữ số hay từng ký tự. Do đó, ta phải biết ghép nối nhiều ma trận LED với VDK để hiển thị một chuỗi số hay chuỗi ký tự như mong muốn.

Với LED 7 đoạn, ta chỉ cần hai cổng của VDK để ghép nối cho 8 LED 7 đoạn. Nhưng với LED ma trận, phải cần hai cổng cho một ma trận LED và như vậy phải thêm một cổng nữa để điều khiển các ma trận hiển thị như mong muốn.

– Dùng vi mạch chốt dữ liệu để chốt dữ liệu cột hay hàng, dữ liệu còn lại mắc song song với các ma trận LED (hình 12.12).

Có thể có các phương án ghép nối nhiều ma trận LED như sau:

– Điều khiển mỗi ma trận LED bằng cổng logic và tranzito điều khiển hàng của ma trận LED.

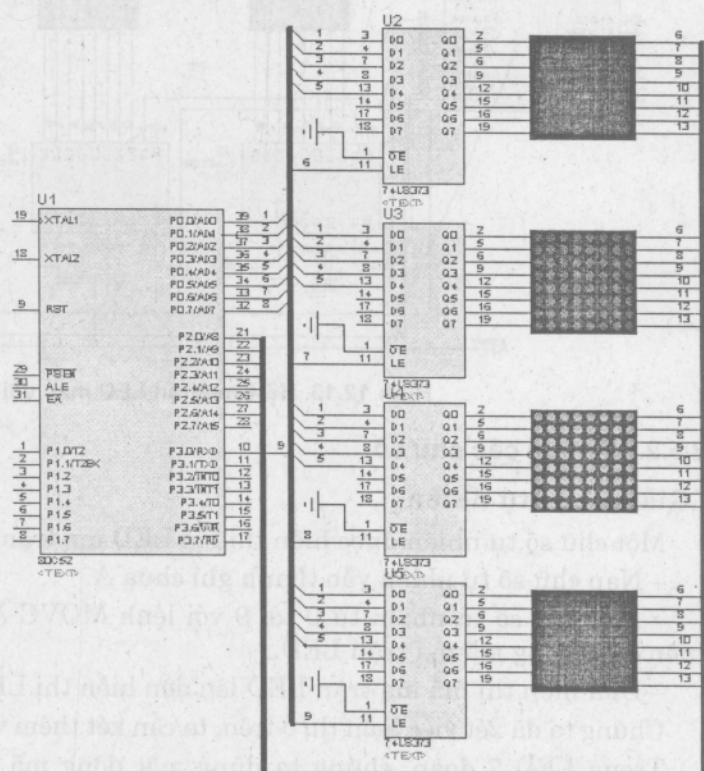
Lối vào điều khiển ghi dữ liệu vào anôt, katôt, dẫn kênh hay chốt là các bit ra của cổng VDK thứ ba (hai cổng khác nối với các đường dây hàng và đường dây cột). Với ma trận LED 5x7, ta có thể dùng các lối ra chưa sử dụng để điều khiển cho ma trận LED.

– Dùng các vi mạch tích hợp 4 hay 8 ma trận để hiển thị nhiều ma trận LED. Phương án này đắt tiền hơn nhưng gọn nhẹ, đỡ tốn công làm mạch in phức tạp.

– Dùng thanh ghi dịch vào nối tiếp, ra song song (ví dụ 74LS299) và nối tăng các thanh ghi này để điều khiển cột (hình 12.13). Phương án này giống cách nối tăng trong vi mạch tích hợp 4 hay 8 ma trận LED. Khi đó, ở chương trình sẽ dùng lệnh dịch phải (RR A) liên tiếp cho tới cột cuối cùng của ma trận LED cuối để điều khiển cột, không phân biệt cột của ma trận LED nào.

– Dùng đường dây địa chỉ để gọi ma trận LED như vi mạch tích hợp 8 ma trận LED.

Sơ đồ ma trận LED dùng chốt có đặc điểm sau:



Hình 12.12. Sơ đồ ma trận LED dùng chốt

– Các cổng nối với các vi mạch chốt tương ứng, có đường dây dữ liệu chung từ cổng P0 của VDK.

– Các hàng của ma trận LED nối với cổng P2 theo đường dây chung.

– 4 vi mạch chốt 74LS373 cho 4 ma trận LED 5x7 (chỉ dùng 5 cột của ma trận 8x8).

Dữ liệu được ghi vào chốt nhờ tín hiệu điều khiển /LE nối với ba bit P0.5, P0.6, P0.7 của cổng P0 và P3.0 của cổng P3.

Như vậy, dữ liệu về 5 cột của ma trận LED được ghi vào các chốt lần lượt nhờ các tín hiệu điều khiển chốt P0.5, P0.6, P0.7 và P3.0.

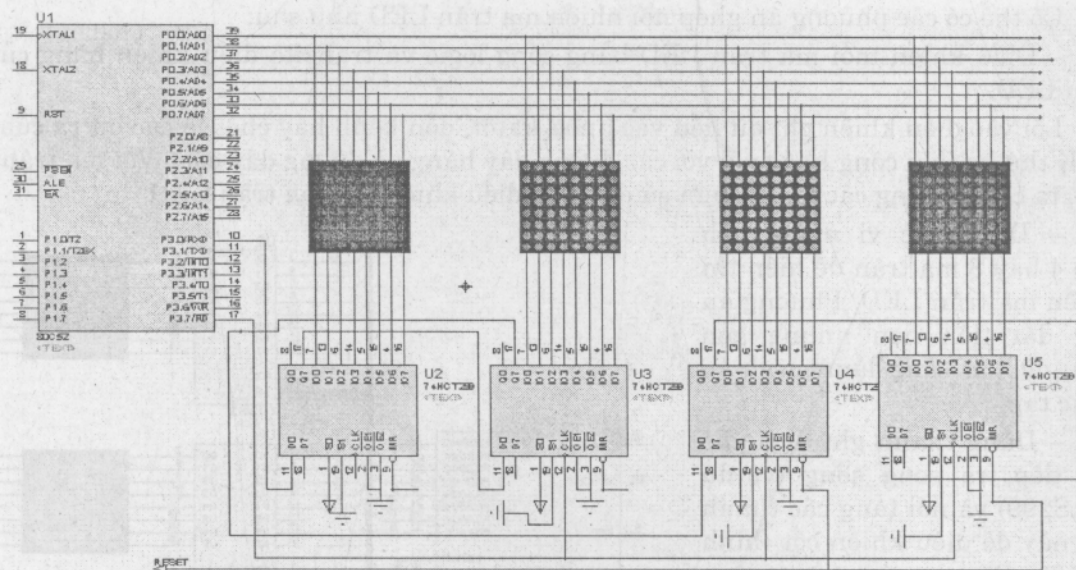
Chương trình phục vụ cho sơ đồ hình 12.12 ngoài việc đưa dữ liệu cột ở cổng P0, còn phải đưa lệnh chốt dữ liệu vào ma trận LED bằng các lệnh SETB và CLR các bit P0.5, P0.6, P0.7 và P3.0 sau thời gian trễ ngắn.

Chương trình phục vụ cho sơ đồ hình 12.13 đơn giản hơn là:

– Đưa một xung ngắn làm tín hiệu CLOCK cho các thanh ghi dịch từ chân P3.0 (dùng lệnh SETB và CLR P3.0).

– Đưa xung đầu tiên vào lối vào DI của ma trận LED thứ nhất.

Cách nối mạch đảm bảo các thanh ghi dịch phải. Có thể nối mạch để điều khiển các thanh ghi dịch trái để thực hiện các ký tự chạy từ phải sang trái.



Hình 12.13. Hệ 4 ma trận LED mắc với thanh ghi dịch

12.5.2. Hiển thị các chữ số

1. Giải mã số tự nhiên

Một chữ số tự nhiên được hiển thị lên LED ma trận theo các công đoạn sau:

- Nạp chữ số tự nhiên vào thanh ghi chứa A.
- Giải mã số tự nhiên từ 0 tới 9 với lệnh MOVC A, @A+DPTR và bảng mã các số tự nhiên dưới dạng mã ma trận LED.

– Đưa hiển thị mã ma trận LED lên đèn hiển thị LED ma trận.

Chúng ta đã xét việc hiển thị ở trên, ta cần xét thêm việc giải mã sang dạng LED ma trận.

Trong LED 7 đoạn, chúng ta dùng một dòng mã các số tự nhiên và lệnh MOVC A, @A+DPTR để giải mã từ số tự nhiên (0 tới 9) ra mã LED 7. Trong lệnh MOVC A,

@A+DPTR cần nạp số tự nhiên vào thanh ghi A, còn giá trị DPTR không đổi, chính là địa chỉ đầu tiên của bảng mã. Kết quả đọc dòng mã LED 7 đoạn đọc được vào thanh ghi A.

Với giải mã LED ma trận, mỗi số tự nhiên có 5 phần tử mã cho 5 cột nên ta đọc bảng mã các số tự nhiên từ 0 tới 9 ghi theo mã LED ma trận LED với lệnh MOVC A, @A + DPTR theo cách sau:

- Có 5 giá trị của A tăng từ 0 tới 4 cho một hàng mã của bảng mã với giá trị DPTR không đổi cho một hàng hay cho một ma trận.
- Giá trị của DPTR lên tăng 5 cho ma trận tiếp theo (bởi các lệnh MOV A, DPTR ; ADD A, #5 và MOV DPTR, A).

Ví dụ 12.9 giới thiệu lệnh gộp GMMT và chương trình giải mã LED ma trận cho các số tự nhiên ghi sẵn trong các ô nhớ từ địa chỉ 10h và các cột của mã LED ma trận của các số 1, 2, 3, 4 ghi liên tiếp trong các địa chỉ nhớ đầu 30h (cho số 1), 40h (cho số 2), 50h (cho số 3), 60h (cho số 4) và 70h (cho số 5).

Với lệnh gộp GMMT <số, địa chỉ nhớ> trên ta có thể dùng để giải mã bất kỳ số tự nhiên (0-9) nào để ghi 5 mã LED ma trận vào các dải nhớ có địa chỉ bắt đầu từ 30h, 40h, 50h, 60h v.v...

2. Hiện thị số tự nhiên

Số tự nhiên sau khi giải mã từ số tự nhiên thành mã LED ma trận sẽ được hiện thị trên đèn LED ma trận như lệnh gộp HTMT <địa chỉ nhớ> như trong ví dụ 12.10. Lệnh gộp này cũng tương tự các lệnh cho hiện thị bảng dữ liệu ở trên là:

- Lệnh gộp HTMT nhận dữ liệu đầu tiên tại địa chỉ nhớ mà lệnh gộp giải mã ma trận GMMT <số, địa chỉ nhớ> đã ghi mã sau khi giải mã.
- Các dữ liệu tiếp theo cần hiện thị lấy tại các địa chỉ tiếp theo của địa chỉ đầu tiên.
- Các mã của các ma trận tiếp theo cũng tương tự, cũng có 5 mã ở các địa chỉ liên tiếp với địa chỉ đầu tiên ghi trong lệnh gộp.

Chương trình của ví dụ 12.10 có thêm các lệnh gộp giải mã @GMMT <số, địa chỉ nhớ> trước các lệnh gộp hiện thị @HTMT <địa chỉ nhớ> để giải mã các số để hiện thị. Muốn chương trình này chạy, phải copy thêm lệnh gộp @GMMT <số, địa chỉ nhớ> vào chương trình hay vào thư viện chương trình MYLIB và kê khai \$INCLUDE MYLIB ở đầu của chương trình trên.

3. Giải mã và hiện thị các chữ số

Để đơn giản, ta có thể gộp hai lệnh gộp giải mã và hiện thị thành lệnh gộp GMHTM <số> cho một số tự nhiên tùy ý, không cần ghi đệm vào ô nhớ. Chương trình của ví dụ 12.10 giới thiệu lệnh gộp và chương trình giải mã và hiện thị (gọi tắt là hiện thị) với tên HTMT <số>.

12.5.3. Hiện thị thời gian đếm lùi ở đèn giao thông

Trong chương 11 đã trình bày cách mắc và chương trình điều khiển đèn giao thông hiện thị trên LED 7 đoạn. Chương này giới thiệu chương trình hiện thị với giải mã của ma trận LED và hiện thị thời gian đếm lùi trên các đèn ma trận LED.

; Ví dụ 12.9.GMMT: Giải mã từ 1, 2, 3, 4, 5	; Ví dụ 12.10.HTMT: Hiện thị các số 1, 2, 3, 4, 5
<pre>macro @GMMT <số tự nhiên, địa chỉ nhớ> mov b, #1 ; số tự nhiên cần giải mã mov dptr, #bangma; địa chỉ đầu bảng mã mov r1, #5 ; số cột trong một ma trận mov r0, #2 ; địa chỉ nhớ ghi mã</pre>	<pre>macro @HTMT <địa chỉ của số> mov r3, #01 ; lời điều khiển mov r2, #5 ; số lần hiện thị mov r0, #11 ; địa chỉ nhớ mã nhan1: mov a, @R0 ; đọc mã vào a</pre>

<pre> mov a, r1 ; gửi r1 vào a mul ab ; nhân 5x số tự nhiên add a, dpl ; địa chỉ ma trận mới mov dpl, a ; gửi vào dpl mov r2, #00h ; biến cột trong ma trận nhan0: mov a, r2 ; gửi biến cột r2 vào a movc a, @a+dptr ; đọc bảng mov @r0, a ; ghi vào ô nhớ inc r0 ; tăng địa chỉ nhớ inc r2 ; tăng biến cột djnz r1, nhan0 ; lặp lại cho cột khác endmac org 00h ljmp main org 100h main: mov r0, #10h ; địa chỉ nhớ 0 tới 9 mov r1, #0Ah ; số lần lặp mov a, #00h lap: mov @r0, a inc r0 inc a djnz r1, lap ; lặp 10 lần ghi 0-9 @GMMT 0, 30h ; giải mã số 0 rồi ghi ; vào các địa chỉ nhớ từ 30h @GMMT 1, 30h ; mã số 1 ghi từ 40h @GMMT 2, 40h ; mã số 2 ghi từ 50h @GMMT 3, 50h ; mã số 3 ghi từ 60h @GMMT 4, 60h ; mã số 4 ghi từ 70h sjmp \$ bangma: db 0c1h, 0bah, 0b6h, 0aeh, 0c1h ; mã số 0 db 0eeh, 0deh, 080h, 0feh, 0feh ; mã số 1 db 0b0h, 0b6h, 0b6h, 0b6h, 086h ; mã số 2 db 0b6h, 0b6h, 0b6h, 0b6h, 080h ; mã số 3 db 087h, 0f7h, 0f7h, 080h, 0f7h ; mã số 4 db 086h, 0b6h, 0b6h, 0b6h, 0b0h ; mã số 5 db 080h, 0b6h, 0b6h, 0b6h, 0b0h ; mã số 6 db 0bfh, 0b7h, 0b7h, 080h, 0f7h ; mã số 7 db 080h, 0b6h, 0b6h, 0b6h, 080h ; mã số 8 db 086h, 0b6h, 0b6h, 0b6h, 080h ; mã số 9 end </pre>	<pre> mov hang, a ; đưa ra hàng mov cot, r3 ; đưa ra cột lcall delay10ms ; chờ quan sát mov cot, #0ffh ; xoá cột mov a, r3 ; dịch điều khiển rl a ; dịch trái mov r3, a ; gửi vào a inc r0 ; tăng r0 djnz r2, nhan1 ; giảm r2, trở về nhan1 call delay10ms ; gọi trễ endmac org 00h ljmp main cot equ p1 hang equ p0 org 100h main: @GMMT 1, 30h @HTM 30h @GMMT 2, 40h @HTM 40h @GMMT 3, 50h @HTM 50h sjmp \$ delay10ms: mov r5, #250 del1: mov r6, #250h djnz r6, \$ djnz r5, del1 ret bangma: db 0c1h, 0bah, 0b6h, 0aeh, 0c1h ; mã số 0 db 0eeh, 0deh, 080h, 0feh, 0feh ; mã số 1 db 0b0h, 0b6h, 0b6h, 0b6h, 086h ; mã số 2 db 0b6h, 0b6h, 0b6h, 0b6h, 080h ; mã số 3 db 087h, 0f7h, 0f7h, 080h, 0f7h ; mã số 4 db 086h, 0b6h, 0b6h, 0b6h, 0b0h ; mã số 5 db 080h, 0b6h, 0b6h, 0b6h, 0b0h ; mã số 6 db 0bfh, 0b7h, 0b7h, 080h, 0f7h ; mã số 7 db 080h, 0b6h, 0b6h, 0b6h, 080h ; mã số 8 db 086h, 0b6h, 0b6h, 0b6h, 080h ; mã số 9 end </pre>
--	--

Ví dụ 12.11.DGT giới thiệu chương trình điều khiển đèn giao thông với đếm lùi có giải mã LED ma trận và hiển thị trên các đèn ma trận LED. Chương trình trên gồm:

- Đếm lùi thời gian thực bằng ngắt do Timer với số đếm khác nhau định sẵn trong các ô nhớ.
- Giải mã ma trận LED các số đếm lùi với lệnh gộp @GMMT <số, địa chỉ bắt đầu nhớ mã>.

- Hiển thị số đếm trên các đèn ma trận LED với lệnh gộp @HTMT <địa chỉ lấy mã>.
- Điều khiển các LED đơn to cho các màu và kích hoạt hay ngừng đếm lùi.

Như ví dụ 12.10.DHMT, chương trình 12.9.DGT này chỉ khác phần giải mã và hiển thị và dùng các lệnh gộp @GMMT và HTMT.

Các nhiệm vụ trên, ta đã xét ở các phần trước. Để tiện sử dụng, chương trình giới thiệu cả các lệnh gộp mà không gọi từ thư viện chương trình MYLIB.MAC.

; Ví dụ 12.11.DGT: Điều khiển đèn giao thông với các đèn Xanh, Đỏ, Vàng	
<pre> macro @Dem <1,2,3> mov P1, #1 ; mov Count, #12 ; l3: jbc TimeOutflag, l4 @GMMT 30h, 40h @HTMT 30h @GMMT 31h, 50h @HTMT 50h sjmp l3 endmac macro @GMMT <số tự nhiên, địa chỉ nhớ> mov b, l1 ; địa chỉ số tự nhiên ; cần giải mã mov dptr, #bangma ; địa chỉ đầu bảng mã mov r1, #5 ; số cột trong một ma trận mov r0, #2 ; địa chỉ nhớ ghi mã mov a, r1 ; gửi r1 vào a mul ab ; nhân 5x số tự nhiên add a, dpl ; địa chỉ ma trận mới mov dpl, a ; gửi vào dpl mov r2, #00h ; biến cột trong ma trận nhan0: mov a, r2 ; gửi biến cột r2 vào a movc a, @a+dptr ; đọc bng mov @r0, a ; ghi vào " nhớ inc r0 ; tăng địa chỉ nhớ inc r2 ; tăng biến cột djnz r1, nhan0 ; lặp lại cho cột khác endmac macro @HTMT < địa chỉ của số > mov r3, #01 ; lời điều khiển mov r2, #5 ; số lần hiển thị mov r0, #11 ; địa chỉ nhớ mã nhan1: mov a, @R0 ; đọc mã vào a mov hang, a ; đưa ra hàng org 100h Reset: ; ***** ; dùng Timer0 mode 1, ngắt 50000 microsec ; ***** mov tmod, #00000001b </pre>	<pre> mov cot, r3 ; đưa ra cột lcall delay10ms ; chờ quan sát mov cot, #0ffh ; xoá cột mov a, r3 ; dịch điều khiển rl a ; dịch trái mov r3, a ; gửi vào a inc r0 ; tăng r0 djnz r2, nhan1 ; gim r2, trở về nhan1 call delay10ms ; gọi trễ endmac macro @DXanh <1, 2, 3, 4, 5> Chopxanh1: mov r5, #6 l2: cpl l3 mov Count, #1 l4: jbc TimeOutflag, l5 sjmp l4 l5: djnz r5, l2 endmac org 00h mp Reset org 0bh jmp Timer0Int cot equ p3 hang equ p0 Xanh1 equ P1.0 Vang1 equ P1.1 Do1 equ P1.2 Xanh2 equ P1.3 Vang2 equ P1.4 Do2 equ P1.5 org 20h TimeOutflag equ 20h HalfSecond equ 30h Count equ 31h djnz HalfSecond, NotHalfSecond mov HalfSecond, #500/50 dec Count mov a, Count jnz Return setb TimeOutflag </pre>

<pre> mov th0, #high (65535-50000) mov tl0, #low (65535-50000) mov ie, #10000010b mov HalfSecond, #500/50 setb tr0 clr TimeOutflag Mainloop: @Dem 100001b, 20, Wait1, Chopxanh1 @DXanh 1, Loop1, Xanh1, Wait2, \$1 @Dem 100010b, 4, Wait3, \$2 \$2: @Dem 001100b, 20, Wait4, Chopxanh2 @DXanh 2, Loop2, Xanh2, Wait5, \$3 @Dem 010100b, 4, Wait6, \$4 ;***** ; Timer0 interrupt service routine ;***** Timer0Int: push acc mov th0, #high (65535-50000) mov tl0, #low (65535-50000) </pre>	<pre> NotHalfSecond: Return: pop acc reti bangma: db 0c1h, 0bah, 0b6h, 0aeh, 0c1h; mã số 0 db 0eeh, 0deh, 080h, 0feh, 0feh; mã số 1 db 0b0h, 0b6h, 0b6h, 0b6h, 086h; mã số 2 db 0b6h, 0b6h, 0b6h, 0b6h, 080h; mã số 3 db 087h, 0f7h, 0f7h, 080h, 0f7h; mã số 4 db 086h, 0b6h, 0b6h, 0b6h, 0b0h; mã số 5 db 080h, 0b6h, 0b6h, 0b6h, 0b0h; mã số 6 db 0bfh, 0b7h, 0b7h, 080h, 0f7h; mã số 7 db 080h, 0b6h, 0b6h, 0b6h, 080h; mã số 8 db 086h, 0b6h, 0b6h, 0b6h, 080h; mã số 9 delay10ms: mov r5, #250 del1: mov r6, #250h djnz r6, \$ djnz r5, del1 ret end </pre>
--	--

Chương trình dùng:

– Hai lệnh gộp là @DEM và @ChopXanh để dùng cho chương trình chính có 6 đoạn sử dụng chúng với các thông số đếm (20, 4 cho 10giây và 3giây) và nhãn khác nhau.

– Chương trình phục vụ ngắt cho Timer0 với thời gian cơ bản là 50000 microgiây

Theo trình tự của chương trình ta có:

- Đóng mạch các đèn Xanh1 và Đỏ2 trong 10giây.
- Chớp/tắt đèn Xanh1, đóng mạch Đỏ2 trong $0,3 \times 5 = 3$ giây.
- Đóng mạch các đèn Vàng1, Đỏ2 trong 2giây.
- Đóng mạch các đèn Xanh2 và Đỏ1 trong 10giây.
- Chớp/tắt đèn Xanh2, đóng mạch Đỏ2 trong $0,3 \times 5 = 3$ giây.
- Đóng mạch các đèn Vàng2, Đỏ1 trong 2giây.

12.6. HIỂN THỊ ĐỘNG TRÊN LED MA TRẬN

Ở chương 11, với các LED đơn hay LED 7 đoạn, ta đã đề cập tới việc hiển thị động của các chữ và số trên chúng. Trong chương này, ta cũng xét vấn đề hiển thị động tức nhấp nháy, tắt dần, sáng dần hay chữ chạy trên các LED ma trận theo các chiều khác nhau (trái qua phải, phải qua trái, trên xuống, dưới lên v.v...). Trong lĩnh vực quảng cáo, vấn đề hiển thị động của các đèn hiển thị đặc biệt được lưu ý vì kích thích và tạo hiệu quả cao cho người quan sát các tin đưa lên các đèn hiển thị. Với LED ma trận, chương trình thực hiện hiển thị động phức tạp hơn nhưng lại hiệu quả hơn vì đẹp hơn các LED đơn hay LED 7 đoạn.

Tại mỗi thời điểm, vi điều khiển chỉ đưa ra các hàng và một cột hoặc các cột và một

hàng của các đèn LED trong ma trận LED của đèn LED ma trận. Tùy bảng dữ liệu và giải thuật của chương trình, tùy thời gian trễ hay lặp lại, ta có thể thực hiện được một hay tổ hợp các loại hiển thị động sau:

– *Hiển thị đứng yên*: các đèn hiển thị chữ hay số đứng yên, số đèn bằng số dữ liệu cần đưa nhưng thời gian sáng và thời gian tắt rất nhỏ (cỡ 50 – 100 microsec).

– *Hiển thị chớp/tắt*: các đèn hiển thị chữ hay số đứng yên, số đèn bằng số dữ liệu cần đưa nhưng thời gian sáng và thời gian tắt lớn (cỡ sec) tới mức mắt người quan sát có thể nhận ra. Có thể thực hiện chớp/tắt bằng việc đọc bảng lần lượt có dữ liệu (tín hiệu 00H) và xoá dữ liệu (các tín hiệu 1 (hay 0FFH)).

– *Hiển thị từ phải qua trái hay từ trái qua phải*: dữ liệu của một đèn (5 cột) đưa lên lần lượt các đèn từ phải qua trái hay từ trái qua phải theo sự lùi dần hay tiến lên của dữ liệu.

– *Hiển thị sáng dần từ hai phía (đầu, cuối của bảng hiển thị)*: các giá trị điều khiển đèn sáng có giá trị 1 tiến dần vào giữa.

– *Hiển thị sáng dần từ giữa của bảng hiển thị*: các giá trị điều khiển đèn sáng có giá trị 1 tiến dần từ giữa ra hai đầu của bảng hiển thị.

– *Hiển thị sáng dần từ trên xuống dưới*: đưa hiển thị ra các cột, còn hàng cho chạy từ trên xuống dưới với thời gian trễ lớn.

12.6.1. Chương trình đọc bảng của hiển thị động

Ta sẽ viết lệnh gộp chung để đọc bảng như giới thiệu ở ví dụ 12.12.

Chương trình của ví dụ 12.12 có các phần sau:

+ *Đọc bảng*:

– Đọc bảng và đưa hiển thị bắt đầu từ địa chỉ 160 của bảng mã, ghi ở ô nhớ 10h (không đọc từ đầu bảng). LED ma trận đầu tiên ở bên trái.

– Đọc 80 cột của bảng, tại địa chỉ 80h, ghi ở ô nhớ 20h (ô nhớ ghi địa chỉ đầu tiên để đọc bảng).

+ *Xử lý địa chỉ ghi trong 2 ô nhớ 10h và 20h*: bằng cách giảm dần các giá trị ghi ban đầu bằng các lệnh DEC 10h, DEC 20h.

+ *Đọc lại bảng với địa chỉ mới*.

Do đọc bảng từ giữa và địa chỉ đầu tiên cứ giảm dần (do lệnh DEC) nên các phần tử của bảng cứ hiện theo cột tới dần cột 0. Nói khác đi, các ký tự hay số chạy tới dần cột 0 của LED ma trận đầu tiên.

12.6.2. Lệnh gộp của hiển thị động

1. Lệnh hiển thị

Từ chương trình của ví dụ 12.12 và các ví dụ khác, ta có thể viết lệnh gộp đọc bảng mã của các chương trình hiển thị động như trong thư viện ở ví dụ 12.13. Lệnh gộp trên có dạng là:

@HTHI < 1, 2, 3, 4, 5, 6, 7 >

Trong đó:

– *Các thông số 1, 5, 6*: là các nhãn đầu chương trình, đầu đoạn đọc mã cho cột và nhãn bắt đầu cho một LED kế.

– *Thông số 2*: ghi số trạng thái vào thanh ghi R4.

- Thông số 3: địa chỉ cuối, ghi vào địa chỉ nhớ 10h.
- Thông số 4: địa chỉ đầu, ghi vào địa chỉ nhớ 20h.
- Thông số 7: địa chỉ bắt đầu bảng mã.

2. Lệnh tăng/giảm địa chỉ

Hai lệnh gộp xử lý địa chỉ là dịch địa chỉ đi 80 cột, tức là các ô nhớ 10h, 20h tăng lên (cộng) hay giảm đi (trừ) đi 80 như bảng 12.16.

Bảng 12.6. Các lệnh gộp tăng hay giảm địa chỉ

MACRO @XLDPC	MACRO @XLDPT
CLR C	CLR C
MOV A, 10H	MOV A, 10H
ADDC A, #80 ; cộng vào #80	SUBB A, #80 trừ đi #80
MOV 10H, A	MOV 10H, A
CLR C	MOV A, 20H
CLR C	CLR C
MOV A, 20H	SUBB A, #80 trừ đi #80
ADDC A, #80 ; cộng vào #80	MOV 20H, A
MOV 20H, A	ENDMAC
ENDMAC	

Trong thư viện, còn có các lệnh gộp cho hiển thị màu và điều khiển LED ma trận theo tín hiệu xung.

12.6.3. Các cách thay đổi địa chỉ của hiển thị động

Vì thay đổi địa chỉ cuối (ghi ở ô nhớ địa chỉ 10h) và địa chỉ đầu (ghi ở ô nhớ địa chỉ 20h) mà việc đọc bảng thay đổi nội dung, tạo nên hiển thị động.

Ngoài hai lệnh gộp cộng hay trừ nội dung các ô nhớ 10h và 20h, còn có lệnh tăng (INC) hay giảm (DEC) các ô nhớ 10h và 20h.

12.6.4. Bảng dữ liệu

Tùy theo loại chương trình, ta có các bảng dữ liệu khác nhau:

- Với chương trình có một vòng lặp chỉ có một bảng dữ liệu. Các chương trình này chỉ thay đổi địa chỉ đọc một bảng duy nhất cho tới/lui hay chớp/tắt.

- Với chương trình có hai vòng lặp có hai bảng dữ liệu. Kết hợp thay đổi địa chỉ và bảng dữ liệu, các chương trình:

- + Chương trình 7 cho sáng dần từ hai phía.

- + Chương trình 8 cho tắt dần từ giữa.

- + Chương trình 9 cho tắt hết từ giữa.

- + Chương trình 10 cho sáng lại từ hai phía.

- Với chương trình có ba vòng lặp có ba bảng dữ liệu. Chính các bảng dữ liệu này làm cho hiển thị:

- + Chương trình 3 cho sáng dần từ dưới lên khi lần lượt dùng ba bảng mã.

- + Chương trình 5 cho sáng dần từ trên xuống khi lần lượt dùng ba bảng mã.

Chính các bảng dữ liệu trên, cùng với địa chỉ đầu (trong ô nhớ 20h) và địa chỉ cuối

(trong ô nhớ 10h) và cách thức thay đổi địa chỉ như tăng dần bởi lệnh tăng (INC), giảm dần bởi lệnh giảm (DEC), tăng đột ngột bởi lệnh cộng (ADD) và giảm đột ngột bởi lệnh trừ (SUBB) tạo nên sự hiển thị động khác nhau.

12.6.5. Các phương án hiển thị động

Qua các chương trình từ 1 tới 10, ta có nhiều cách hiển thị động lên LED ma trận như tóm tắt ở bảng 12.7.

Từ các điều trình bày trên, ta thấy:

- Đèn hiển thị LED ma trận chỉ có 10 đèn, hiển thị cho $8 \times 10 = 80$ cột.
- Bảng dữ liệu có dung lượng lớn hơn 80 cột, tùy yêu cầu của cách hiển thị động có thêm nhiều dữ liệu cho hiển thị.
- Trong mỗi vòng lặp, chỉ dùng một bảng dữ liệu. Nói khác đi, trong một vòng lặp, chỉ một dữ liệu được đưa hiển thị.
- Việc xử lý dữ liệu (+80, -80, tăng dần, giảm dần) chính là sự thay đổi dữ liệu đưa hiển thị khác với dữ liệu đã hiển thị trước hay cũng chính dữ liệu trên nhưng thứ tự đưa ra sẽ khác hay thứ tự của đèn hiển thị sẽ khác.

Tóm lại, ta hình dung các địa chỉ đầu (ghi ở địa chỉ 20h), cuối (ghi ở địa chỉ 10h) như địa chỉ hay kích thước của một cửa sổ để “nhìn” vào bảng dữ liệu có kích thước lớn hơn của sổ nhìn. Ta có thể “nhìn” động các dữ liệu bằng cách dịch chuyển cửa sổ. Tùy cách dịch cửa sổ tăng (bằng lệnh INC) hay giảm (bằng lệnh DEC) dần, tăng hay giảm đột ngột (bằng các lệnh cộng hay trừ) mà dữ liệu được “nhìn thấy” sẽ hiện lên khác nhau.

; Ví dụ 12.12: Chương trình chữ chạy tới		
ORG 0000H		DB 01111110B
LOOP_MAN: MOV R5, #3	; chữ chạy tới	DB 10000001B
LOOP1: MOV R4, #80	; tạo số trạng thái	DB 00000011B
MOV R6, #2	; tạo thời gian cho trạng thái	DB 11110101B
MOV 10H, #160	; số cột quét là 160 cột	DB 11110110B
MOV 20H, #80	; dịch cột quét là 80 cột	DB 11110101B
LOOP1_1: MOV R3, #7FH	; giá trị điều khiển cột Po	DB 00000011B
MOV R1, #0		DB 00000000B
MOV R2, #00H		DB 11111011B
MOV R0, 20H		DB 11110111B
MOV A, #7FH		DB 11101111B
LOOP1_2: MOV P1, #0FFH	; tắt hàng	DB 00000000B
MOV P0, #0FFH	; tắt cột mắc với P0	DB 10000001B
MOV P2, R2	; điều khiển LED ma trận	DB 01111110B
MOV P0, R3	; điều khiển cột P0	DB 01101110B
MOV A, R0	; đọc phần tử từ 20h	DB 01101110B
MOV DPTR, #BANG_MA_1		DB 10001101B
MOVC A, @A+DPTR	; đọc bảng mã	DB 11111111B
MOV P1, A	; đưa ra hàng mắc với P1	DB 11111111B
LCALL DELAY		DB 11111111B
INC R0		DB 11111111B
MOV A, P0		DB 11111111B
RR A		DB 00000000B
MOV R3, A		DB 11100111B
INC R1		DB 11011011B
CJNE R1, #8, LOOP1_2	; kiểm tra quét 8 cột	DB 10111101B
MOV R1, #0		DB 01111110B

<pre> INC R2 ; nhảy sang Led kế MOV A, R0 CJNE A, 10H, LOOP1_2 ; kiểm tra quét 160 cột DJNZ R6, LOOP1_1 DEC 10H DEC 20H MOV R6, #2 DJNZ R4, LOOP1_1 DELAY: MOV R7, #250 X: DJNZ R7, X RET BANG_MA_1: DB 00000000B DB 11110111B DB 11110111B DB 11110111B DB 00000000B DB 10000001B DB 01111110B DB 01111110B </pre>	<pre> DB 00000000B DB 11110111B DB 11110111B DB 11110111B DB 00000000B DB 00000011B DB 11110101B DB 11110110B DB 11110101B DB 00000011B DB 00000000B DB 11110111B DB 11110111B DB 11110111B DB 00000000B DB 00000000B DB 11110111B DB 11110111B DB 11110111B DB 00000000B DB 00000000B end </pre>
; Ví dụ 12.13: Các lệnh gộp của MYLIB.ASM	
<pre> MACRO @HTHI < 1, 2, 3, 4, 5, 6, 7 > l1: MOV R4, #12 ; tạo số trạng thái MOV R6, #1 ; tạo thời gian cho trạng thái MOV 10H, #13 ; cột quét là 80 cột MOV 20H, #14 ; địa chỉ đầu tiên l5: MOV R3, #7FH ; giá trị điều khiển cột MOV R1, #0 ; biến điều khiển cột MOV R2, #00H ; biến điều khiển LED MOV R0, 20H ; nạp số hàng MOV A, #7FH l6: MOV P1, #0FFH ; tắt hàng MOV P0, #0FFH ; tắt cột MOV P2, R2 ; chọn LED MOV P0, R3 ; điều khiển cột MOV A, R0 ; biến dòng của bảng MOV DPTR, #17 ; địa chỉ bảng mã MOVCA, @A+DPTR ; đọc bảng MOV P1, A ; đưa ra hàng LCALL DELAY ; gọi trễ INC R0 ; tăng địa chỉ dptr MOV A, P0 ; đọc P0 cho điều khiển cột RR A ; dịch phải, điều khiển cột MOV R3, A INC R1 ; tăng biến cột của ma trận CJNE R1, #8, l6 ; kiểm tra quét đủ 8 cột MOV R1, #0 ; xóa biến cột INC R2 ; nhảy sang Led kế MOV A, R0 ; nạp số hàng R0 vào A </pre>	<pre> MOV A, 10H SUBB A, #80 MOV 10H, A MOV A, 20H CLR C SUBB A, #80 MOV 20H, A ENDMAC macro @hthi1mltm <1, 2, 3, 4, 5, 6> mov r1, #00h mov dptr, #11 ; 1.bgma mov 30h, dpl mov 31h, dph l2: mov r2, #1 ; maina, lần lặp cho ma trận mov a, 30h ; xử lý địa chỉ add a, r0 mov 30h, a mov a, 31h addc a, #00h mov 31h, a l3: mov r0, #00 ; mainb, biến cột mov r3, #00000001b ; mã quét l4: mov a, r0 ; mainc movc a, @a+dptr mov hang1, a mov l5, r3 ; 5. cotd1 lcall delay mov l5, #00h ; cotd1 inc r0 ; tăng lên cho cột kế inc dptr </pre>

CJNE A, 10H, 16 ; kiểm tra quét 80 cột DJNZ R6, 15 ; lặp lại R6 = #02 lần ENDMAC MACRO @XLDPC CLR C MOV A, 10H ADDC A, #80 MOV 10H, A CLR C MOV A, 20H ADDC A, #80 MOV 20H, A ENDMAC mov hang2, a setb p2.1 ; cam xanh setb p2.4 ; tạo xung clr p2.3 ; cho phép ra call delay setb p2.3 ; lam cam mov hang2, #00h clr p2.0 ; cắt mực 1 clr p2.2 ; cắt mực 1 clr p2.4 ; xuống xung mov hang2, #00h MACRO @XLDPT CLR C	mov a, r3 ; dịch mã quét rl a mov r3, a cjne r0, #6, 14 ; lặp lại 6 lần cho 6 cột djnz r2, 13 inc r1 ; tăng dòng của bảng mov a, r1 cjne a, #16, 12 ; 6. số dòng (HEX) endmac setb p2.1 ; lam, cam mov hang1, #00h inc dptr clr a movc a, @a+dptr clr p2.0 ; cắt mực 1 clr p2.4 ; sườn xuống của xung inc r1 inc dptr cjne r1, #177, main3xd inc r0 inc r0 mov a, dpl cjne a, #000h, main1xd mov a, dph cjne a, #16h, main1xd endmac
---	---

Bảng 12.7. Các chương trình hiển thị động của LED ma trận

Bài	Hành động	Lệnh gộp	Bảng mã	Số vòng/lặp	Xử lý địa chỉ
1	Chạy tới	@HTHILoop1, 2, 160, 80, Loop11, Loop12, bma1	Bma1	1	Dec 10h
4	Chạy lui	@HTHILoop4, 80, 160, 80, Loop41, Loop42, bma1	Bma1	1	inc 10h
2	Chớp/tắt	@HTHILoop2, 2, 80, 00, Loop2-1, Loop22, bma1	Bma1	1	+80
6	Chớp/tắt	@HTHILoop6, 2, 160, 80, Loop1-1, Loop12, bma1	bma4	1	+80
7	Sáng dẫn từ giữa	@HTHILoop71, 5, 80, 00, Loop711, Loop712, bma71 @HTHILoop72, 5, 80, 00, Loop721, Loop722, bma72	Bma71 Bma72	2	+80
8	Tắt dẫn từ giữa	@HTHILoop81, 5, 160, 80, Loop811, Loop812, bma72 @HTHILoop82, 5, 240, 160, Loop81, Loop822, bma71	Bma71 Bma72	2	+80
9	Tắt hết từ giữa	@HTHILoop91, 5, 80, 00, Loop911, Loop912, bma92 @HTHILoop92, 5, 80, 00, Loop912, Loop922, bma91	Bma71 Bma72	2	-80
10	Sáng lại từ hai phía	@HTHILoop101, 5, 160, 80, Loop1011, Loop1012, bma92 HTHILoop102, 5, 240, 80, Loop1021, Loop1022, bma91	Bma92 Bma91	2	-80
3	Sáng dẫn từ dưới lên	@HTHILoop33, 10, 160, 80, Loop311, Loop312, bma33 @HTHILoop32, 10, 240, 160, Loop321, Loop322, bma32 @HTHILoop31, 5, 240, 160, Loop331, Loop332, bma31	Bma33 Bma32 Bma31	3	-80 -80 -80
5	Sáng dẫn từ dưới lên	@HTHILoop51, 10, 80, 00, Loop511, Loop512, bma31 @HTHILoop52, 10, 80, 00, Loop521, Loop522, bma32 @HTHILoop53, 5, 80, 00, Loop531, Loop532, bma33	Bma31 Bma32 Bma33	3	+80 +80 +80

CÂU HỎI, BÀI TẬP VÀ THỰC HÀNH

I. Câu hỏi

1. Cấu tạo một LED ma trận.
2. Cách hiển thị của một LED ma trận.
3. Cách thành lập một bảng dữ liệu cho một chữ số, một ký tự, một con số gồm nhiều chữ số và một chuỗi ký tự.
4. Thành lập bảng dữ liệu cho chuỗi ký tự "Trường Đại học Bách Khoa xin kính chào quý khách"
5. So sánh LED ma trận với LED 7 đoạn về cấu tạo, nguyên lý hoạt động, cách mắc và cách hiển thị.
6. Hiển thị một màu, nhiều màu.
7. Nguyên tắc của hiển thị động: chớp/tắt, chữ chạy tới/lùi, chữ chạy từ trên xuống/dưới lên, tắt dần từ hai phía, sáng lại từ hai phía, sáng lại từ giữa ra.

II. Bài tập

1. Viết chương trình hiển thị chuỗi "Trường Đại học Bách Khoa xin kính chào quý khách".
2. Viết chương trình hiển thị chuỗi "Xin kính chào xuan Dinh Hoi- 2007".
3. Viết chương trình hiển thị chuỗi "Trường Đại học Bách Khoa xin kính chào quý khách" Chớp/tắt.
4. Viết chương trình hiển thị chuỗi "Trường Đại học Bách Khoa xin kính chào quý khách" Chạy tới, chạy lui.
Viết chương trình hiển thị chuỗi "Trường Đại học Bách Khoa xin kính chào quý khách" sáng dần từ trên xuống và từ dưới lên.

III. Thực hành

1. Chạy các ví dụ của chương 12 trên chương trình PINNACLE.
2. Chạy chương trình của các bài tập 1 tới 5 trên chương trình PINNACLE.
3. Thiết kế mạch hiển thị với ma trận LED 5x7 với chương trình PROTEUS.
4. Thiết kế mạch hiển thị với ma trận LED 8x8 với chương trình PROTEUS.
5. Thiết kế mạch hiển thị với 4 ma trận LED 5x7 với chương trình PROTEUS.

CHƯƠNG 13

GHÉP NỐI VỚI MÀN HÌNH TINH THỂ LỎNG (LCD)

Mục đích của chương:

- Hiểu được cấu tạo và hoạt động của một LCD.
- Hiểu được hệ lệnh của LCD.
- Hiểu được cách lập trình cho LCD để hiển thị ký tự và số.
- Hiểu được nguyên tắc hoạt động của màn hình tinh thể lỏng cho máy vi tính và ti vi.

13.1. TỔNG QUÁT VỀ LCD

13.1.1. Yêu cầu sử dụng LCD

Hiện nay, màn hình tinh thể lỏng được dùng rộng rãi trong các thiết bị điện tử số dùng trong sinh hoạt, cũng như các thiết bị khoa học, từ dụng cụ gia đình như lò sấy thức ăn, bếp từ, ti vi màn hình phẳng tới các máy photo copy, biển quảng cáo tại các nơi công cộng.

LCD đã thay thế các thiết bị đơn giản như LED 7 đoạn, ma trận diot vì:

- LCD hiển thị được cả ký tự và chữ số dưới dạng các diot chấm sáng như của LED ma trận.
- LCD có thể soạn thảo và ghi được các thông chữ có kích thước và dạng khác nhau.
- LCD có thể hiển thị được hình ảnh đồ họa với độ phân giải cao, do đó, có thể dùng làm màn hình phẳng thay cho các ống hình điện tử (CRT) của các ti vi thế hệ cũ với độ tương phản tốt, ánh sáng nhẹ, thích hợp với mắt người quan sát.
- LCD có thể hiển thị được nhiều màu sắc đẹp gần với màu sắc tự nhiên hơn ống hình CRT.

13.1.2. Cấu tạo màn hình tinh thể lỏng

Màn hình tinh thể lỏng LCD (Liquid Crystal Display) là màn hình hiển thị có chứa nhiều tinh thể lỏng. Mỗi tinh thể có cấu tạo như hình 13.1 và cả màn LCD này có dạng hình học như hình 13.2.

1. Cấu tạo một tinh thể lỏng

Một phần tử tinh thể lỏng LCD gồm:

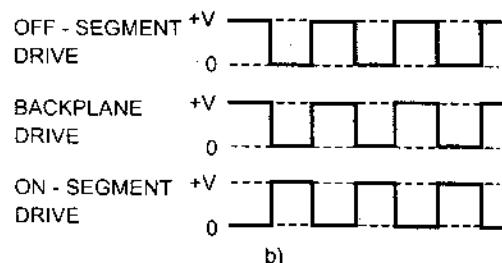
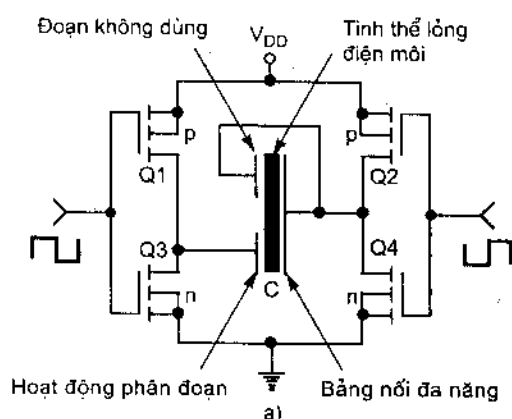
- Tinh thể lỏng điện môi (phần tử của hình 13.1) nằm kẹp giữa các tấm thủy tinh làm đế, tấm dài và hai đoạn nhỏ (không dùng và sử dụng) ở phía trước.
- Hai cặp tranzito trường Q1-Q2 và Q3-Q4 loại p-n dùng để điều khiển ghi và xóa thông tin bằng các tín hiệu xung vào đoạn sử dụng và đế.

Khi có một xung điện áp khoảng 2V-3V vào đoạn sử dụng và đế, có một hiệu điện thế 50mV giữa đoạn sử dụng và đế nên không gian giữa chúng có một điện trường, các phân tử của tinh thể được sắp xếp lại, mắt nhìn vào thấy rõ để màu đen. Tùy hình dáng điện trường có thể nhìn ra dạng ký tự màu đen khác nhau. Tín hiệu điều khiển vào đoạn sử dụng và đế phải là sóng vuông, tần số từ 30 tới 150Hz. Các tín hiệu vào đế và đoạn sử dụng phải ngược pha, còn để xóa là cùng pha (hình 13.1b).

2. Cấu tạo một màn LCD kích thước nhỏ

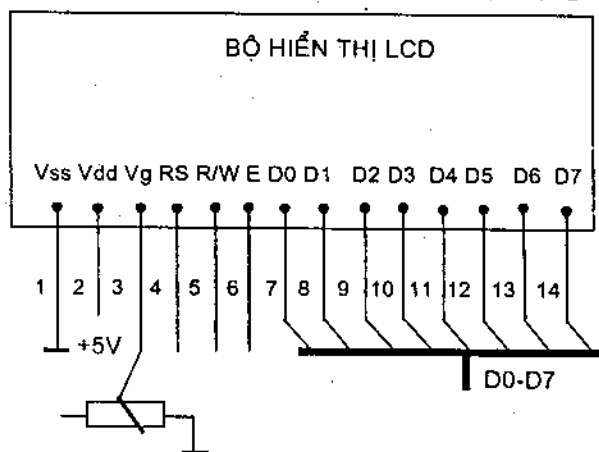
Một màn hình tinh thể lỏng LCD bất kỳ đều có dạng chữ nhật (hình 13.2), có 14 chân cắm (bảng 13.1) cho:

– 8 tín hiệu dữ liệu DB0–DB7.



Hình 13.1. Tinh thể lỏng (a)
và tín hiệu điều khiển (b)

- Hai thế nguồn nuôi V_{CC} (+5V) và V_{SS} (đất).
- Thế V_{EE} điều chỉnh độ tương phản.
- Tín hiệu lựa chọn thanh ghi RS: điều khiển thanh ghi lệnh (RS = 0) và thanh ghi dữ liệu (RS = 1).
- Tín hiệu ghi/đọc dữ liệu RW: khi ghi RW = 0 hay đọc RW = 1.
- Tín hiệu cho phép E: cho phép hoạt động.



Hình 13.2. Sơ đồ chân của LCD

13.1.3. Các loại LCD

Hiện nay, có nhiều hãng điện tử chế tạo các loại LCD như Hitachi, Sharp, Seiko, Samsung, Hanntronic, Optex, Phillip, Powertip, Matrix Orbital v.v... Thông dụng nhất là LCD của các hãng Hitachi.

Các LCD trên có khác nhau về:

– *Kích thước*: có loại kích thước nhỏ, trung bình và lớn. Loại kích thước nhỏ cho các thiết bị điện tử số dùng trong gia đình. Loại kích thước trung bình cho các thiết bị điện tử số dùng trong công nghiệp và ti vi. Loại kích thước lớn dành cho màn hình đồ họa và các biển quảng cáo.

– *Cách trao đổi tin*: song song (cho các thiết bị ở gần) và nối tiếp (cho các thiết bị ở xa).

– *Hiển thị*: chữ/số hay hình ảnh (đồ họa).

Bảng 13.1 giới thiệu một số LCD thông dụng trên thị trường.

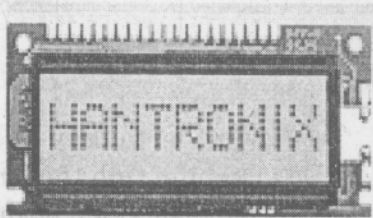
Bảng 13.1. Các loại LCD thông dụng

Hãng	Ký hiệu	Ma trận	Công dụng	Loại
Optrex	DMC	5x7, 1 hay 2 dòng	Cho thiết bị số	Chữ/số, song song
Sharp	LM	5x7, 1 hay 2 dòng	Cho thiết bị số	Chữ/số, song song
Hitachi	HD 44780	5x7, 1 hay 2 dòng	Cho thiết bị số	Chữ/số, song song
Seiko	KS0066	5x7, 1 hay 2 dòng	Cho thiết bị số	Chữ/số, song song

Hãng	Ký hiệu	Ma trận	Công dụng	Loại
Hantronix	LCM1	5x7, 1 hay 2 dòng	Cho thiết bị số	Chữ/số, song song
Matrix Orbital	LCD4021	124x64	Cho quảng cáo	Chữ/số, nối tiếp
Matrix Orbital	LCD4041	240x64	Cho quảng cáo	Chữ/số, nối tiếp
Hantronix		128x240	Cho quảng cáo	Đồ hoạ
Hantronix		240x64	Cho quảng cáo	Đồ hoạ
Hantronix		320x240	Cho quảng cáo	Đồ hoạ
Powertip	NCL	128x64	Cho quảng cáo	Đồ hoạ
Powertip	NCL	240x128	Cho quảng cáo	Đồ hoạ
Toshiba	18742	240x128	Cho quảng cáo	
Samsung	KS0108B	128x64	Cho quảng cáo	

13.1.4. Một số tính năng của LCD dùng cho kit vi điều khiển

LCD loại nhỏ dùng cho các thiết bị số thông dụng để ghép nối với các kit vi điều khiển có hình dáng như hình 13.3.



Hình 13.3. LCD dùng cho VĐK

Bảng 13.2. Các chân của LCD nối với VĐK

Chân	Ký hiệu	I/O	Công dụng
1	Vss	–	Đất (0V)
2	Vcc	–	Nguồn (+5V)
3	Vee	–	Nguồn điều khiển độ tương phản
4	RS	I	RS = 0 chọn thanh ghi lệnh/trạng thái RS = 1 chọn thanh ghi dữ liệu
5	R/W	I	RW = 1, đọc dữ liệu; RW = 0, ghi dữ liệu
6	E	I	Cho phép LCD hoạt động
7	DB0	I/O	Bit D0 của dữ liệu
8	DB1	I/O	Bit D1 của dữ liệu
9	DB2	I/O	Bit D2 của dữ liệu
10	DB3	I/O	Bit D3 của dữ liệu
11	DB4	I/O	Bit D4 của dữ liệu
12	DB5	I/O	Bit D5 của dữ liệu
13	DB6	I/O	Bit D6 của dữ liệu
14	DB7	I/O	Bit D7 của dữ liệu

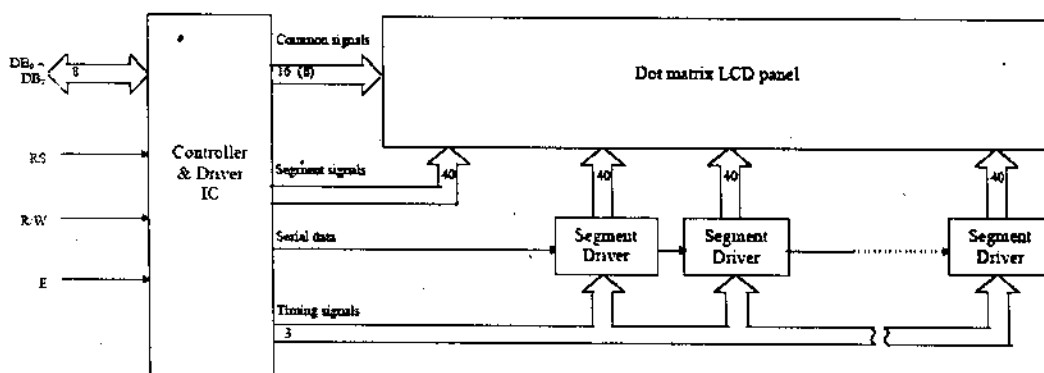
- Sau đây là một số tính năng của LCD thường mắc với VĐK:
- LCD có loại ma trận 5x8 và 5x10
 - Có thể mắc với VĐK loại 4 bit hay 8 bit.
 - Hiển thị dữ liệu trong bộ nhớ DD RAM chứa 80 ký tự 8 bit.
 - Có bộ nhớ ROM phát ký tự (CG ROM) chứa 160 ký tự với ma trận 5x7 dot.
 - Có các lệnh cho các hành động xóa hiển thị, con trỏ về đầu dòng (cursor home), bật/tắt hiển thị (display on/off), bật/tắt con trỏ (cursor on/off), nhấp nháy ký tự (blink character), dịch con trỏ (cursor shift), dịch hiển thị (display shift).
 - Có mạch xóa (reset) và dao động bên trong.

13.2. CẤU TRÚC CỦA LCD

Các LCD của các hãng khác nhau nhưng đều có một cấu trúc và tính năng thống nhất về hệ lệnh và cách lập trình.

13.2.1. Sơ đồ khối

Thực chất một màn hình tinh thể lỏng LCD có sơ đồ khối như hình 13.4.



Hình 13.4. Sơ đồ khối của LCD

Trên hình 13.4 ta thấy LCD gồm:

- Để màn hình tinh thể lỏng ma trận điểm (dot matrix LCD panel).
- Vi mạch điều khiển (controller & driver IC).
- Các bộ điều khiển các đoạn (segment driver).

Thực chất, LCD là một vi điều khiển chuyên dụng để điều khiển màn hình tinh thể lỏng LCD với các khối vi điều khiển có hệ lệnh gồm 11 lệnh với các mã lệnh dạng HEX, có bộ nhớ RAM (DD RAM) để ghi và phát dữ liệu, có bộ nhớ ROM (CG ROM) để ghi và phát các mẫu ký tự dạng mã ASCII.

Dưới đây, khi xét tới các lệnh của LCD, chúng ta sẽ thấy chúng liên quan tới các phần chức năng nào của vi mạch, trong đó có các thanh ghi lệnh và dữ liệu, bit trạng thái bận (BF– Busy Flag),

Bộ đếm địa chỉ (Address Count), bộ nhớ ROM phát các ký tự (CG ROM– Character Generator ROM), bộ nhớ RAM chứa các ký tự của người dùng (CG RAM – Character Generator RAM) và các khối khác.

13.2.2. Các khối chức năng cơ bản

1. Các thanh ghi

LCD có hai thanh ghi:

– *Thanh ghi lệnh/trạng thái (IR)*: dùng để ghi lệnh và đọc trạng thái là bit cờ bận (BF) và địa chỉ của địa chỉ nhớ DD RAM hay CG RAM. Về lệnh, có thể ghi các lệnh như xoá hiển thị, dịch con trỏ cũng như ghi địa chỉ cho DD RAM hay CG RAM.

– *Thanh ghi dữ liệu (DR)*: dùng để ghi hay đọc dữ liệu vào các địa chỉ nhớ DD RAM hay CG RAM khi địa chỉ ghi vào từ thanh ghi lệnh IR.

Sau khi ghi dữ liệu vào ô nhớ, địa chỉ tự động tăng lên 1.

Các thanh ghi thực hiện đọc hay ghi theo các tín hiệu RS và RW như ở bảng 13.3.

Bảng 13.3. Các thanh ghi và lệnh ghi/đọc

RS	R/W	Hành động
0	0	Ghi vào thanh ghi lệnh và thực hiện lệnh
0	1	Đọc trạng thái cờ bận (BF) và đếm địa chỉ (DB0–DB7)
1	0	Ghi dữ liệu vào DR cho DD RAM hay CG RAM
1	1	Đọc dữ liệu từ DR từ DD RAM hay CG RAM

2. BIT BF

Bit trạng thái này thu được từ lệnh *Đọc* và kiểm tra bit DB7 của thanh ghi lệnh IR. Bit này có giá trị:

– $BF = 0$: cho biết bên trong LCD không bận hay rồi, không thực hiện một hành động bên trong nào. Ta có thể ghi hay đọc lệnh hoặc dữ liệu cho LCD.

– $BF = 1$: cho biết bên trong LCD bận, đang thực hiện một hành động bên trong. Ta không thể ghi hay đọc lệnh hoặc dữ liệu cho LCD.

3. Thanh ghi đếm địa chỉ (Addressing Counter-AC)

Thanh ghi đếm địa chỉ này dùng để chỉ địa chỉ của các ô nhớ (DD RAM, CG RAM hay CG ROM). Nội dung của thanh ghi này có thể:

– *Ghi vào từ thanh ghi lệnh IR*: hai lệnh ghi địa chỉ cho CG RAM và D RAM là khác nhau (bảng 13.3).

– *Đọc ra từ thanh ghi lệnh IR*: các bit DB0–DB6 cùng với bit BF ở DB7 bởi lệnh *Đọc* thanh ghi lệnh (bảng 13.3).

Nội dung thanh ghi địa chỉ trên tự động tăng lên 1 sau khi *Ghi* hay *Đọc* dữ liệu trong DD RAM hay CG RAM.

4. RAM hiển thị dữ liệu (DD RAM) và cách ghi địa chỉ

Bộ nhớ RAM cho dữ liệu này có thể chứa tới 80 dữ liệu 8 bit. Vùng không dùng của RAM này có thể được VDK dùng như vùng RAM thông dụng.

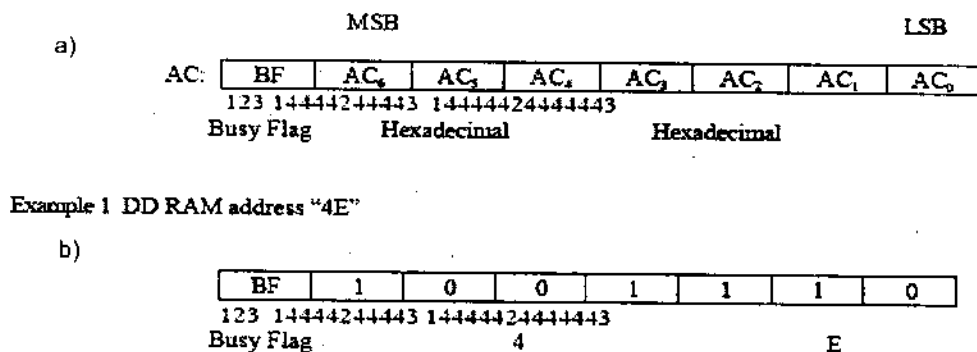
Hình 13.5 giới thiệu byte địa chỉ và bit BF đọc được khi đọc thanh ghi trạng thái.

Tùy theo cách dùng LCD với một dòng, hai dòng hay bốn dòng chữ, ta có cách ghi địa chỉ khác nhau.

LCD một dòng chữ có 80 chữ.

Địa chỉ của DD RAM một dòng chữ (lệnh có $N = 0$) có được khi đọc thanh ghi lệnh/trạng thái.

($RS = 0$, $R/W = 0$) như ở hình 13.6 với địa chỉ đầu tiên là 00H tới 4FH tức từ 00D tới 79D hay có tổng cộng 80 địa chỉ.



Hình 13.5. Byte địa chỉ (a) của LCD và ví dụ (b)

Khi đó địa chỉ hiển thị trên LCD đánh số từ 1D tới 80D (hàng trên của hình 13.6), còn địa chỉ của DD RAM từ 00 tới 4FH (hàng dưới của hình 13.6).



Hình 13.6. Các địa chỉ của LCD một dòng chữ

Sau khi ghi một ký tự hay có lệnh dịch con trỏ về bên phải, địa chỉ tự tăng lên 1. Ban đầu, con trỏ chỉ địa chỉ 00H, nếu có lệnh dịch con trỏ sang trái, địa chỉ đầu tiên là 4FH và địa chỉ cuối cùng là 4EH.

Nếu đã ghi lệnh hiển thị hai dòng (với N = 1), ta có các địa chỉ hiển thị và DD RAM như hình 13.7.

Một số LCD, như Optex còn có 4 dòng hiển thị

1	2	3	4	5		39	40	← Địa chỉ hiển thị
00	01	02	03	04		26	27	← Địa chỉ DD RAM
40	41	42	43	44		66	67	← Địa chỉ DD RAM

Hình 13.7. Địa chỉ hiển thị và địa chỉ DD RAM

5. CG ROM

CG ROM là bộ nhớ ROM chỉ đọc dữ liệu là các mẫu chữ/số và ký hiệu như bảng 13.4, được ghi khi chế tạo LCD. Các LCD của Nhật Bản còn có các mẫu chữ Nhật.

Bảng 13.4. Bảng mẫu chữ trong CG ROM

NO.7066-0A		b7-b4 b3-b0	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
	CG RAM (1)																	
0000	(2)																	
0001	(3)																	
0010	(4)																	
0011	(5)																	
0100	(6)																	
0101	(7)																	
0110	(8)																	
0111	(1)																	
1000	(2)																	
1001	(3)																	
1010	(4)																	
1011	(5)																	
1100	(6)																	
1101	(7)																	
1110	(8)																	
1111																		

Bộ nhớ CG ROM này có thể phát 80 ký hiệu 8 bit, phát 160 ký hiệu ma trận 5x7 điểm hay 32 loại 5x10 điểm từ mã ASCII 8 bit.

Nhờ bộ mẫu chữ này, LCD hiển thị được các chữ/số hay ký hiệu lên màn hình. Địa chỉ để đọc các chữ này là các mã ASCII, chính là các dữ liệu đưa vào DD RAM (theo nguyên tắc nhớ liên hợp, địa chỉ là mã ASCII của chữ, còn nội dung của chữ được đọc ra từ bộ nhớ ROM).

6. CG RAM

CG RAM là bộ nhớ RAM dùng để phát ký hiệu, do người lập trình tự viết theo mong muốn. CG RAM chứa 8 ma trận 5x7 chữ mẫu. Để hiển thị một chữ mẫu chứa trong CG RAM, một mã chữ như bảng 4 phải được ghi vào DD RAM.

13.3. HỆ LỆNH CỦA LCD

Mọi LCD đều tuân theo một hệ lệnh duy nhất, có 11 lệnh với mã số HEX từ 01H (xóa hiển thị), con trỏ trở về đầu dòng (02H), xác định chế độ cho LCD (04H tới 07H), điều khiển bật/tắt hiển thị (08H tới 0FH), dịch con trỏ và hiển thị (10H tới 1FH), xác lập chức năng (20H tới 3FH), xác lập địa chỉ CG RAM (4xH tới 4FH), xác lập địa chỉ DD RAM (80H tới FFH), đọc thanh ghi lệnh để biết bit BF và địa chỉ, ghi và đọc dữ liệu cho LCD vào thanh ghi dữ liệu.

13.3.1. Bảng lệnh

Bảng lệnh của LCD được giới thiệu ở bảng 13.5.

Bảng 13.5. Bảng lệnh của LCD

Tên lệnh	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Hành động	Thời gian
Xóa hiển thị	0	0	0	0	0	0	0	0	0	1	Xóa toàn hiển thị và xác lập địa chỉ DD RAM 0 trong đếm địa chỉ.	1,64ms
Trở về đầu	0	0	0	0	0	0	0	0	1	-	Xác lập địa chỉ DD RAM 0 nhận địa chỉ bộ đếm. Cũng trở về vị trí gốc. Nội dung DD RAM giữ không đổi.	1,64ms
Xác lập chế độ vào	0	0	0	0	0	0	0	1	1/D	S	Xác lập chiều chuyển động của con trỏ và xác định sự dịch của hiển thị. Các hành động này được thực hiện trong khi ghi hay đọc dữ liệu.	40μs
Điều khiển bật/tắt hiển thị	0	0	0	0	0	0	1	D	C	B	Xác lập ON/OFF của toàn hiển thị (D), ON/OFF con trỏ (C) và nhấp nháy ký tự ở vị trí con trỏ.	40μs
Dịch con trỏ hay hiển thị	0	0	0	0	0	1	S/C/R/L	-	-	-	Di chuyển con trỏ và dịch hiển thị không làm thay đổi nội dung DD RAM.	40μs
Xác lập chức năng	0	0	0	0	1	DL	N	F	-	-	Xác lập độ dài dữ liệu ghép nối (DL), số của dòng hiển thị (L) và phông ký tự (F).	40μs
Xác lập địa chỉ CG RAM	0	0	0	1	ADD CG RAM						Xác lập địa chỉ CG RAM. Dữ liệu CG RAM được gửi và nhận sau khi xác lập.	40μs
Xác lập địa chỉ DD RAM	0	0	1	ADD RAM							Xác lập địa chỉ DD RAM. Dữ liệu DD RAM được gửi và nhận sau khi xác lập.	40μs
Đọc cờ bận và địa chỉ	0	1	BF	AC							Đọc cờ bận (BF) chỉ hành động bên trong đang được thực hiện và đọc nội dung của bộ đếm địa chỉ.	40μs

Tên lệnh	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Hành động	Thời gian	
Ghi dữ liệu CG hay DD RAM	1	0									WRITE DATA	Ghi dữ liệu vào DD hay CG RAM.	40μs
Đọc dữ liệu CG hay DD RAM	1	1									READ DATA	Đọc dữ liệu từ DD hay CG RAM.	40μs

- Ghi chú :**
1. Thời gian thực hiện là thời gian cực đại khi tần số trung bình hay tần số dao động là 250kHz.
 2. Thời gian thực hiện thay đổi khi tần số thay đổi. Ví dụ, khi tần số trung bình hay tần số dao động là 270kHz: 40 μ s – 250/270 = 37 μ s.
 3. Các ký hiệu viết tắt trong bảng theo các giá trị như bảng 13.6.

Bảng 13.6. Giá trị các chữ của bảng 13.6

Bit	Xác lập = 0	Xác lập = 1
ID	0 = Giảm vị trí con trỏ	1 = tăng vị trí con trỏ
S	0 = không dịch hiển thị	1 = có dịch hiển thị
D	0 = tắt hiển thị	1 = bật hiển thị
C	0 = tắt con trỏ	1 = bật con trỏ
B	0 = tắt nhấp nháy con trỏ	1 = nhấp nháy con trỏ
SC	0 = dịch chuyển con trỏ	1 = không dịch con trỏ
RL	0 = dịch trái con trỏ	1 = dịch phải con trỏ
DL	0 = ghép nối 4 bit	1 = ghép nối 8 bit
N	0 = 1/8 hay 1 dòng	1 = 1/16 hay 2 dòng
F	0 = ma trận 5x7 điểm	1 = ma trận 5x10 điểm
BF	0 = có thể nhận lệnh/dữ liệu	1 = bận, hành động nội

- DD RAM : dữ liệu hiển thị (Display Data RAM)
ACC: địa chỉ của RAM máy phát ký tự
ADD : địa chỉ của RAM dữ liệu hiển thị phù hợp với địa chỉ con trỏ.
AC: bộ đếm địa chỉ (Address Counter) dùng cho DD RAM và CG RAM
CG RAM: máy phát ký tự ghi ở RAM (Character Generator)
CG ROM: máy phát ký tự ghi ở ROM (Character Generator)
Bảng 13.7 giới thiệu một số mã lệnh của một số lệnh thông dụng của LCD.

Bảng 13.7. Mã lệnh của một số lệnh thông dụng của LCD.

Mã H	Lệnh	Mã H	Lệnh
1	Xoà màn hình hiển thị	0EH	Bật hiển thị, nhấp nháy con trỏ
2	Trở về đầu dòng	0FH	Tắt hiển thị, nhấp nháy con trỏ
4	Dịch con trỏ sang trái	10H	Dịch vị trí con trỏ sang trái
6	Dịch con trỏ sang phải	14H	Dịch vị trí con trỏ sang phải
5	Dịch hiển thị sang phải	18H	Dịch toàn bộ hiển thị sang trái
7	Dịch hiển thị sang trái	1CH	Dịch toàn bộ hiển thị sang phải
8	Tắt con trỏ, tắt hiển thị	80H	Đưa con trỏ về đầu dòng thứ nhất
0AH	Bật con trỏ, bật hiển thị	C0H	Đưa con trỏ về đầu dòng thứ hai
0CH	Tắt con trỏ, bật hiển thị	38H	Hiện hai dòng, ma trận 5x7 điểm

13.3.2. Các lệnh

Từ bảng lệnh, chúng ta suy ra chi tiết các lệnh.

1. Lệnh xoá hiển thị (01H)

Lệnh này có mã số 01H và dạng như sau:

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	0	0	1

Sau khi ghi 01H vào thanh ghi lệnh, LCD bị xoá các ký tự về hiển thị. Đây là lệnh xoá mềm hay xoá bằng chương trình, tương tự lệnh xoá khi bật nguồn nuôi (xoá bằng phần cứng). Lệnh này xoá bộ nhớ của hiển thị và đưa con trỏ về đầu dòng thứ 1, thứ 2 hay thứ 4, trừ LCD L4044 đưa về cả dòng 3.

2. Lệnh đưa con trỏ trở về đầu dòng (02H)

Lệnh này có mã số 02H hay 03H và dạng như sau:

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	0	1	*

* Có giá trị bất kỳ (0 hay 1), nên lệnh có thể là 02H hay 03H.

Sau khi ghi 02H (hay 03H) vào thanh ghi lệnh, con trỏ của LCD sẽ trở về đầu dòng hiện hành (dòng 1, dòng 2 hay dòng 4) của màn LCD, trừ LCD L4044 là “xoá hiển thị” (địa chỉ 0, địa chỉ “80H”. Nội dung của DD RAM không thay đổi.

3. Xác lập chế độ vào (Entry Mode Set) (04H tới 7H)

Lệnh này được dùng để thay đổi vị trí con trỏ.

Lệnh này có dạng như sau:

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	1	I/D	S

Trong đó I/D và S tuân theo bảng 13.7, cụ thể là:

– Có giá trị 04H (I/D = 0, S = 0): con trỏ không thay đổi vị trí, giữ vị trí cũ vì I/D = 0 hay giảm địa chỉ DD RAM của khối nhớ sau khi ghi hay đọc một mã của ký tự của DD RAM hay CG RAM)

– Có giá trị 05H (I/D = 0, S = 1): dịch toàn bộ hiển thị về bên phải sau khi ghi dữ liệu vào LCD. Không có dịch dữ liệu khi đọc dữ liệu từ DD RAM.

– Có giá trị 06H (I/D = 1, S = 1): con trỏ có thay đổi vị trí về bên phải (S = 0), vì I/D = 1 hay tăng địa chỉ DD RAM của một khối nhớ sau khi ghi hay đọc một mã của ký tự của DD RAM hay CG RAM)

– Có giá trị 07H (I/D = 1, S = 1): con trỏ có thay đổi vị trí về bên trái (S = 1), vì I/D = 1 hay tăng địa chỉ DD RAM của một khối nhớ sau khi ghi hay đọc một mã của ký tự của DD RAM hay CG RAM).

4. Điều khiển bật/tắt (ON/OFF) hiển thị và con trỏ (8H tới FH)

Lệnh này có dạng như sau:

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	1	D	C	B

D: Hiển thị bật (ON) khi D = 1 và tắt (OFF) khi D = 0. Khi hiển thị tắt (D = 0), dữ liệu hiển thị trong DD RAM không thay đổi. Tin được trở lại khi D = 1.

C: Con trỏ (Cursor) được hiển thị khi C = 1 và không hiển thị khi C = 0. Nếu con trỏ biến mất, chức năng của I/D không thay đổi trong khi hiển thị dữ liệu ghi. Trong ma trận 5x7 điểm, có dòng thứ 8 đóng vai trò của con trỏ.

B: Khi B = 1, ký tự ở vị trí con trỏ bắt đầu nhấp nháy (Blinking, các điểm sáng đều tắt) trong khoảng thời gian 0,4 giây. Khi B = 0, con trỏ không nhấp nháy.

Con trỏ (C) và nhấp nháy (B) có thể xác lập đồng thời (với tần số dao động bằng 250kHz).

5. Dịch chuyển con trỏ/hiển thị (10H tới 1CH)

Lệnh này có dạng như sau:

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	1	S/C	R/L	*	*

* Không có giá trị gì.

Các chữ S/C và R/L có các giá trị như ở bảng 13.8.

Lệnh Dịch con trỏ/hiển thị di chuyển con trỏ hay dịch hiển thị không làm thay đổi nội dung DD RAM, chỉ có vị trí con trỏ và nội dung của đếm địa chỉ (AC) là liên quan. Lệnh này có ý nghĩa để chỉnh hiển thị và tìm thấy hiển thị bởi vì vị trí con trỏ hay hiển thị có thể được dịch không có việc ghi hay đọc dữ liệu. Trong trường hợp hiển thị 2 dòng, con trỏ được dịch từ khối ký tự 40 của dòng 1 sang khối ký tự 1 của dòng 2. Sự hiển thị của dòng 1 và dòng 2 được dịch đồng thời. Trong trường hợp hiển thị 4 dòng, con trỏ không di chuyển liên tục từ dòng 2 sang dòng 3. Con trỏ được dịch từ khối ký tự 40 của dòng 3 tới khối ký tự 1 của dòng 4. Các hiển thị của dòng 3 và dòng 4 được dịch đồng thời. Các mẫu chữ của dòng 2 hay dòng 4 không được dịch sang dòng 1 hay dòng 3.

Lệnh dịch con trỏ và hiển thị tuân theo các giá trị S/C và R/L như ở bảng 13.8.

Bảng 13.8. Các phép dịch con trỏ và hiển thị

S/C	R/L	Hành động
0	0	Vị trí con trỏ được dịch về bên trái (AC giảm đi 1)
0	1	Vị trí con trỏ được dịch về bên phải (AC tăng lên 1)
1	0	Toàn bộ hiển thị với con trỏ được dịch về bên trái
1	1	Toàn bộ hiển thị với con trỏ được dịch về bên phải

6. Xác lập chức năng (Function Set)

Lệnh này có dạng như sau:

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	DL	N	F	*	*

* Không có giá trị.

Lệnh này xác lập độ dài của dữ liệu (DL), số dòng hiển thị (N) và phong chữ (F).

DL: độ dài dữ liệu của bộ ghép nối.

– Khi DL = 0: dữ liệu là 4 bit (DB7 – DB4).

– Khi DL = 1: dữ liệu là 8 bit (DB0 – DB7).

Bốn bit cao được truyền trước, 4 bit thấp được truyền sau.

N : số dòng hiển thị.

F: xác lập phong chữ theo bảng 13.9.

Bảng 13.9. Các giá trị của N, F

N	F	Số dòng hiển thị	Phông chữ	Chu trình	Modul LCD
0	0	1	5x7 điểm	1/8	Các chữ của modul
1	0	2	5x7 điểm	1/16	Các chữ của modul
1	1	2	5x10 điểm	1/16	Các chữ của modul

7. Xác lập địa chỉ CG RAM

Lệnh này có dạng như sau:

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	1	A	A	A	A	A	A

Lệnh này xác định địa chỉ CG RAM với bit DB7 = 0, DB6 = 1 và 6 bit DB5–DB0 là giá trị của địa chỉ cần đưa vào. Như vậy, lệnh này có thể xác định được 64 địa chỉ khác nhau của CG RAM. Dùng địa chỉ này để ghi hay đọc ô nhớ của bộ nhớ CG RAM.

8. Xác lập địa chỉ DD RAM

Lệnh này có dạng như sau:

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	1	A	A	A	A	A	A	A

Lệnh này xác định địa chỉ DD RAM với bit DB7 = 1 và 7 bit DB6–DB0 là giá trị của địa chỉ cần đưa vào. Như vậy, lệnh này có thể xác định được 128 địa chỉ khác nhau của DD RAM. Dùng địa chỉ này để ghi hay đọc ô nhớ của bộ nhớ DD RAM.

9. Đọc Busy Flag/địa chỉ

Lệnh này có dạng như sau:

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	BF	A	A	A	A	A	A	A

Lệnh này đọc trạng thái của địa chỉ của thanh ghi lệnh hay đưa lệnh đọc thanh ghi lệnh có RS = 0 và R/W = 0.

Bit DB7 cho biết bit BF (Busy Flag) như sau:

– Nếu BF = 0: LCD không bận vì các hành động bên trong, có thể đưa lệnh, đưa dữ liệu vào LCD.

Nếu BF = 1: LCD đang bận vì các hành động bên trong, không thể đưa lệnh, đưa dữ liệu vào LCD. Người ta dùng lệnh này để kiểm tra trước khi đưa lệnh hay dữ liệu vào LCD.

10. Lệnh Ghi vào CG RAM hay DD RAM

Lệnh này có dạng như sau:

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
1	0	D	D	D	D	D	D	D	D

Lệnh này dùng để ghi 8 bit dữ liệu DDH vào thanh ghi dữ liệu với RS = 1, R/W = 0 vào bộ nhớ CG RAM hay DD RAM. Trước đó, phải dùng lệnh Xác lập địa chỉ CG RAM hay DD RAM (số 7 hay 8 ở trên).

Sau khi ghi dữ liệu, địa chỉ và dịch chuyển hiển thị được xác định bởi lệnh Xác lập chế độ vào (lệnh số 3 ở trên).

11. Lệnh Đọc CG RAM hay DD RAM

Lệnh này có dạng như sau:

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
1	1	D	D	D	D	D	D	D	D

Lệnh này dùng để đọc 8 bit dữ liệu DDH từ thanh ghi dữ liệu với RS = 1, R/W = 1 từ bộ nhớ CG RAM hay DD RAM. Trước đó, phải dùng lệnh Xác lập địa chỉ CG RAM hay DD RAM (số 7 hay 8 ở trên). Các bit dữ liệu DDH ở thanh ghi dữ liệu có được sau khi thực hiện lệnh Đọc thanh ghi dữ liệu, còn các bit DDH lúc đưa lệnh Đọc không có giá trị gì. Nói khác đi, muốn đọc dữ liệu một ô nhớ ta cần:

- Đưa địa chỉ nhớ CG RAM hay DD RAM: với lệnh số 7 hay 8 ở trên.
- Đưa lệnh Đọc thanh ghi dữ liệu: bằng lệnh 11 này bằng cách đưa RS = 1, R/W = 1.
- Đọc các bit của thanh ghi dữ liệu.

Sau khi đọc dữ liệu, địa chỉ tự động tăng lên hay giảm đi 1 đơn vị phụ thuộc vào chế độ vào, nhưng hiển thị không dịch chuyển..

13.3.3. Giảm độ thời gian của các tín hiệu Ghi, Đọc của LCD

Với lệnh Đọc hay Ghi dữ liệu với LCD, phải có thời gian thích hợp cho các tín hiệu đưa vào LCD để hành động Ghi hay Đọc ở bên trong LCD được hoàn tất. Nếu yêu cầu về thời gian này cho các tín hiệu điều khiển LCD không đáp ứng, các hành động Ghi hay Đọc không thực hiện hay chưa kết thúc. Nói khác đi, dữ liệu chưa kịp ghi vào hay đọc ra từ các ô nhớ của CG RAM hay DD RAM.

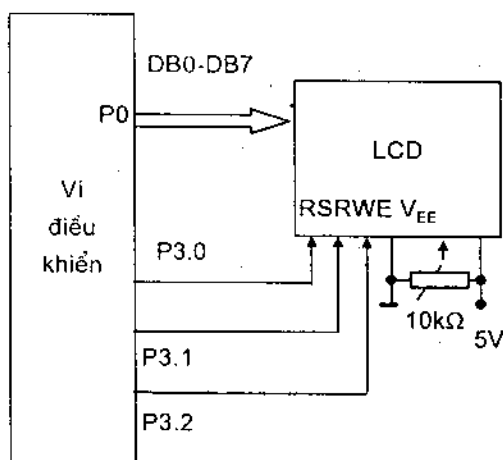
13.4. GHÉP NỐI LCD VỚI VDK HỌ MSC-51

13.4.1. Ghép nối LCD với VDK họ MSC-51

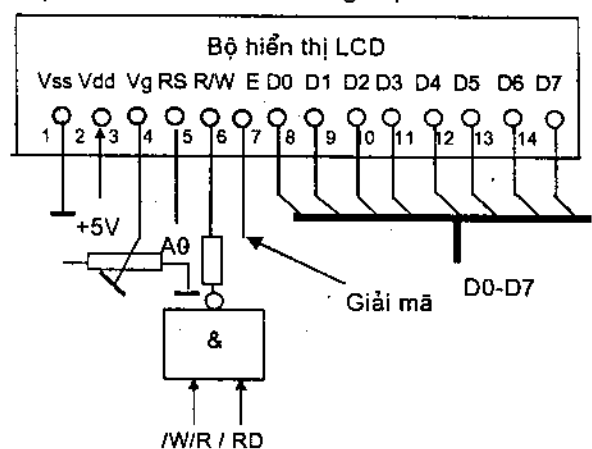
Hình 13.8 và bảng 13.10 giới thiệu cách mắc trực tiếp với vi điều khiển theo nguyên tắc sau:

- Các đường dây dữ liệu mắc với một cổng song song, ví dụ P0.
- Các đường dây RS, RW, E mắc với các lối ra của một cổng song song khác, ví dụ, P3.0, P3.1 và P3.2.

Hình 13.9 giới thiệu LCD mắc với hệ vi điều khiển có bộ giải mã cho LCD, lối vào E mắc với bộ giải mã địa chỉ chọn vi mạch. Lối vào RS mắc với đường dây địa chỉ A0 để chọn thanh ghi lệnh (A0 = 0) hay thanh ghi dữ liệu (A0 = 1). Lối vào RW mắc vào lối ra của cổng OR-NOT mà hai lối vào của nó là các tín hiệu Đọc /RD và Ghi /WR riêng biệt của VDK.



Hình 13.8. Sơ đồ mắc LCD trực tiếp với các chân của VDK điều khiển



Hình 13.9. Sơ đồ mắc LCD với hệ VDK

13.4.2. Các chương trình con và lệnh gộp cơ bản kiểu điều khiển trực tiếp từ VDK

1. Các lệnh cơ bản ghi và đọc các thanh ghi

Từ sơ đồ mắc với VĐK (hình 13.8) và giản đồ thời gian Ghi, Đọc, LCD chỉ hoạt động khi lỗi vào E = 1 và một trong 4 giá trị của các lỗi vào RS, R/W (bảng 13.11). Do đó, có 4 chương trình con cơ bản cho LCD là:

- Ghi lệnh vào thanh ghi lệnh: với RS = 0, R/W = 0, E = 1 nên P3.0 = 0, P3.1 = 0, P3.2 = 1.
 - Đọc trạng thái bit BF và địa chỉ AC: với RS = 0, R/W = 1, E = 1, nên P3.0 = 0, P3.1 = 1, P3.2 = 1.
 - Ghi dữ liệu: với RS = 1, R/W = 0, E = 1 nên P3.0 = 1, P3.1 = 0, P3.2 = 1.
 - Đọc dữ liệu: với E = 1, RS = 1, R/W = 1 nên P3.0 = 1, P3.1 = 1, P3.2 = 1.
- Từ đó, ta có bảng 13.11 cho các giá trị của P1 cho các lệnh cơ bản.

Bảng 13.10. Bảng các chân cắm của một LCD

Chân	Ký hiệu	I/O	Mô tả	Chân	Ký hiệu	I/O	Mô tả
1	Vss	–	Đất	9	DB2	I/O	Dữ liệu DB2
2	Vcc	–	Nguồn nuôi +5V	10	DB3	I/O	Dữ liệu DB3
3	V _{EE}	–	Nguồn nuôi độ tương phản	11	DB4	I/O	Dữ liệu DB4
4	RS	1	Lựa chọn thanh ghi điều khiển (RS = 0) hay dữ liệu (RS = 1)	12	DB5	I/O	Dữ liệu DB5
5	R/W	1	Lựa chọn lệnh đọc RW = 1 hay ghi RW = 0	13	DB6	I/O	Dữ liệu DB6
6	E	I/O	Cho phép	14	DB7	I/O	Dữ liệu DB7
7	DB0	I/O	Dữ liệu DB0	13	DB6	I/O	Dữ liệu DB6
8	DB1	I/O	Dữ liệu DB1	14	DB7	I/O	Dữ liệu DB7

Bảng 13.11. Các giá trị của P1 cho các lệnh cơ bản

Lệnh	RS/P3.0	R/W/P3.1	E/P3.2	P3
Ghi lệnh	0	0	1	4
Đọc trạng thái	0	1	1	6
Ghi dữ liệu	1	0	1	5
Đọc dữ liệu	1	1	1	7

Các chương trình con cơ bản này được viết dưới dạng điều khiển trực tiếp từ các bit lỗi ra của cổng song song của VĐK, ví dụ từ cổng P1 như hình 13.8.

Theo bảng 13.3, ta có 4 thanh ghi có hai địa chỉ là địa chỉ thanh ghi lệnh (LCD – CM hay và dữ liệu LCD – WR) và có 4 lệnh để ghi và đọc chúng.

2. Chương trình con

Để tiện dùng, ta viết được 4 chương trình con là ghi lệnh WR_CMGT, SEND – DATATT, RD – STATUSTT và RD – DATATT (hình 13.10) khi điều khiển LCD trực tiếp từ VĐK vào các chân tín hiệu RS, R/W và E. Khi cần dùng, ta chỉ gọi các chương trình con này bằng lệnh gọi. Các chương trình con này có thể nạp trong thư viện MYLIB.ASM và đầu mỗi chương trình chỉ cần kê khai \$INCLUDEMYLIB.ASM.

Ví dụ: 13.1 giới thiệu chương trình ghi lệnh xóa hiển thị (mã lệnh 01h) và ghi dữ liệu 57H vào LCD theo điều khiển trực tiếp.

3. Lệnh gộp và các chương trình con cơ bản kiểu điều khiển trực tiếp

Để rút ngắn cách viết chương trình, người ta viết lệnh gộp và các chương trình con điều khiển LCD bằng các tín hiệu đưa ra từ cổng song song của VĐK.

Hình 13.10 giới thiệu lệnh gộp ghi/đọc cho LCD điều khiển trực tiếp.

```

MACRO @ll <1, 2,> ; tham số 1. là tên tín hiệu, tham số 2. giá trị 0 hay 1 để xoá hay xác lập
MOV A, #12        ; gửi giá trị (0 hay 1) của tham số 2 vào A
RRC A             ; dịch phải A qua C
MOV ll, C         ; chuyển C vào tên tín hiệu
ENDMAC
RS equ P3.0       ; RS nối với P3.0
RW equ P3.1       ; RW nối với P3.1 vì nếu ghi R/W chương trình dịch không nhận
E equ P3.2        ; E nối với P3.2
DIACHI EQU 0E000H
DIACHI+1 EQU 0E001H
WR_CMDTT:
CLR RS
CLR RW           ; tham số 2. cho lệnh là 0, cho dữ liệu là 1
SETB P3.2       ; xác lập E để cho phép LCD hoạt động
CALL DELAY
CLR E           ; xoá E để cấm LCD hoạt động
RET
WR_DATATT:
CLR RS
CLR RW
SETB E          ; xác lập E để cho phép LCD hoạt động
CALL DELAY
CLR E           ; xoá E để cấm LCD hoạt động
RET
RD_CMDTT:
CLR RS
SETB RW         ; tham số 2. cho lệnh là 0, cho dữ liệu là 1
SETB E          ; xác lập E để cho phép LCD hoạt động
CALL DELAY
CLR E           ; xoá E để cấm LCD hoạt động
RET
RD_DATATT:
SETB RS
SETB RW         ; tham số 2. cho lệnh là 0, cho dữ liệu là 1
SETB E          ; xác lập E để cho phép LCD hoạt động
CALL DELAY
CLR E           ; xoá E để cấm LCD hoạt động
RET

```

Hình 13.10. Lệnh gộp ghi/đọc lệnh/dữ liệu

; Ví dụ: 13.1: Chương trình ghi/đọc trực tiếp vào LCD mắc trực tiếp

```

ORG 00h
LJMP MAIN
ORG 100H
MAIN:
MOV P3, #0ffH      ; xoá E để cấm LCD hoạt động
MOV a, #01h
MOV P0, A

```

CALL WR_CMDTG	; ghi lệnh (01h) vào thanh ghi lệnh
CALL RD_CMDTG	; đọc trạng thái từ thanh ghi trạng thái
mov a, #60h	
MOV P0, A	
CALL WR_DATATG	; ghi dữ liệu 60h vào thanh ghi dữ liệu
CALL RD_DATATG	; đọc dữ liệu từ thanh ghi dữ liệu
SJMP \$	; kết thúc chờ
\$INCLUDE MYLIB.ASM	
END	; kết thúc chương trình dịch

Ở đây dùng lệnh dịch giá trị 0 hay 1 qua bit C vào các bit P3.0, P3.1, P3.2 mà không thể nạp trực tiếp.

Từ lệnh gộp trên, ta có thể viết các chương trình con WR – CMGTG, SEND – DATATG, RD – STATUSTG và RD – DATATG (hình 13.11).

WR_CMDTG:	
@1 RS, 0	; CLR RS
@1 RW, 0	; CLR RW
@1 E, 1	; SETB P3.2, xác lập E để cho phép LCD hoạt động
CALL DELAY	
@1 E, 0	; xoá E để cấm LCD hoạt động
MOV P3, #11111011B	; 0FBH
RET	
WR_DATATG:	
@1 RS, 0	; CLR RS
@1 RW, 0	; CLR RW
@1 E, 1	; SETB E; xác lập E để cho phép LCD hoạt động
CALL DELAY	
@1 E, 0; CLR E	; xoá E để cấm LCD hoạt động
MOV P3, #11111011B	; 0FBH
RET	
RD_CMDTG:	
@1 RS, 0	; CLR RS
@1 RW, 1; SETB RW	; tham số 2. cho lệnh là 0, cho dữ liệu là 1
@1 E, 1; SETB E	; xác lập E để cho phép LCD hoạt động
CALL DELAY	
@1 E, 0; CLR E	; xoá E để cấm LCD hoạt động
MOV P3, #11111011B; 0FBH	
RET	
RD_DATATG:	
@1 RS, 1; SETB RS	
@1 RW, 1; SETB RW	; tham số 2. cho lệnh là 0, cho dữ liệu là 1
@1 E, 1; SETB E	; xác lập E để cho phép LCD hoạt động
CALL DELAY	
@1 E, 0; CLR E	; xoá E để cấm LCD hoạt động
MOV P3, #11111011B; 0FBH	
RET	

Hình 13.11. Các chương trình con điều khiển LCD trực tiếp viết theo lệnh gộp

Ví dụ 13.2 giới thiệu chương trình ghi lệnh xoá hiển thị (mã lệnh 01h), ghi dữ liệu 57H vào LCD và đọc trạng thái và dữ liệu theo điều khiển trực tiếp viết theo lệnh gộp.

So sánh với chương trình ở ví dụ 13.1, ta thấy:

- Có các hành động của chương trình con là giống nhau về ghi lệnh, ghi dữ liệu, đọc trạng thái và đọc dữ liệu.
- Bố trí khai báo \$INCLUDE MYLIB.ASM ở đầu và cuối chương trình chính. Kết quả của chương trình là như nhau, nhưng nên để cuối chương trình chính để tránh thư viện chương trình quá dài (hơn 100H) sẽ đề lên chương trình chính.
- Lệnh gộp cũng đặt trong cùng thư viện chương trình MYLIB.ASM.
- Chương trình dịch của chương trình của ví dụ 13.2 sẽ dài hơn chương trình dịch của ví dụ 13.1.

13.4.3. Lệnh gộp và các chương trình con cơ bản kiểu điều khiển theo địa chỉ

Cách điều khiển LCD này theo nguyên tắc ghi hay đọc dữ liệu vào các thanh ghi theo nguyên tắc gọi địa chỉ của chúng. Từ sơ đồ mắc LCD với VĐK như hình 13.9, ta thấy:

- *Lối vào E của LCD*: mắc với lối ra của giải mã dành cho LCD.
- *Lối vào RS của LCD*: mắc với đường dây địa chỉ A0.
- *Lối vào RW của LCD*: mắc với các tín hiệu lệnh ghi/WR và đọc /RD qua cổng OR – NO.

Như vậy:

- Thanh ghi lệnh có địa chỉ DPTR là địa chỉ cơ bản dành cho LCD với A0 = 0.
- Thanh ghi dữ liệu có địa chỉ DPTR+1 là địa chỉ cơ bản dành cho LCD +1 với A0 = 1.

Hành động Ghi vào thanh ghi lệnh hay dữ liệu chính là đưa nội dung thanh ghi chứa A ra ô nhớ ngoài có địa chỉ DPTR và ngược lại, hành động Đọc thanh ghi của LCD là đọc ô nhớ ngoài có địa chỉ DPTR vào thanh ghi chứa A.

1. Các chương trình con

Chương trình con cơ bản dành cho LCD điều khiển theo địa chỉ là WR_CMD và SEND_DATA RD_CMD và RD_DATA như hình 13.12.

WR_CMD: PUSH ACC	; lưu giữ thanh ghi chứa A
PUSH PSW	; lưu giữ thanh ghi trạng thái PSW
PUSH DPH	; lưu giữ thanh ghi địa chỉ ngoài cao DPH
PUSH DPL	; lưu giữ thanh ghi địa chỉ ngoài thấp DPL
MOV A, #LENH	; nạp dữ liệu là lệnh hay số liệu
MOV DPTR, #DIACHI	; nạp địa chỉ thanh ghi lệnh hay dữ liệu
MOVX @DPTR, A	; ghi a vào ô nhớ ngoài có địa chỉ DPTR
CALL DELAY	; gọi trễ 40 microsec
POP DPL	; hồi phục DPL
POP DPH	; hồi phục DPH
POP PSW	; hồi phục PSW
POP ACC	; hồi phục thanh ghi chứa A
RET	
SEND_DATA :	
PUSH ACC	; lưu giữ thanh ghi chứa A
PUSH PSW	; lưu giữ thanh ghi trạng thái PSW
PUSH DPH	; lưu giữ thanh ghi địa chỉ ngoài cao DPH
PUSH DPL	; lưu giữ thanh ghi địa chỉ ngoài thấp DPL
MOV A, #DULIEU	; nạp dữ liệu là số liệu
MOV DPTR, #DIACHI +1	; nạp địa chỉ thanh ghi dữ liệu
MOVX @DPTR, A	; ghi thanh ghi A vào ô nhớ ngoài địa chỉ DPTR+1

CALL DELAY	; gọi trễ 40 microsec
POP DPL	; hồi phục DPL
POP DPH	; hồi phục DPH
POP PSW	; hồi phục PSW
POP ACC	; hồi phục thanh ghi chứa A
RET	
RD_CMD: PUSH ACC	; lưu giữ thanh ghi chứa A
PUSH PSW	; lưu giữ thanh ghi trạng thái PSW
PUSH DPH	; lưu giữ thanh ghi địa chỉ ngoài cao DPH
PUSH DPL	; lưu giữ thanh ghi địa chỉ ngoài thấp DPL
MOV DPTR, #DIACHI	; nạp địa chỉ thanh ghi lệnh/trạng thái
MOVB A, @DPTR	; đọc ô nhớ ngoài có địa chỉ DPTR vào A
CALL DELAY	; gọi trễ 40 microsec
POP DPL	; hồi phục DPL
POP DPH	; hồi phục DPH
POP PSW	; hồi phục PSW
POP ACC	; hồi phục thanh ghi chứa A
RET	
RD_DATA: PUSH ACC	; lưu giữ thanh ghi chứa A
PUSH PSW	; lưu giữ thanh ghi trạng thái PSW
PUSH DPH	; lưu giữ thanh ghi địa chỉ ngoài cao DPH
PUSH DPL	; lưu giữ thanh ghi địa chỉ ngoài thấp DPL
MOV DPTR, #DIACHI+1	; nạp địa chỉ thanh ghi dữ liệu
MOVB A, @DPTR	; đọc ô nhớ ngoài có địa chỉ DPTR+1 vào A
CALL DELAY	; gọi trễ 40 microsec
POP DPL	; hồi phục DPL
POP DPH	; hồi phục DPH
POP PSW	; hồi phục PSW
POP ACC	; hồi phục thanh ghi chứa A
RET	

Hình 13.12. Các chương trình con cho LCD điều khiển theo địa chỉ

2. Các lệnh ghép cơ bản điều khiển theo địa chỉ

Lệnh ghép ghi @WR_LCD lệnh/dữ liệu và đọc RD_LCD trạng thái/dữ liệu như hình 13.13.

Với cách điều khiển theo địa chỉ, lệnh ghép cơ bản thông dụng cho một hệ VDK khi cổng ra của VDK còn dùng cho ngắt ngoài, trao đổi tín hiệu tiếp của cổng P3.

Có hai lệnh ghép là ghi lệnh/dữ liệu và đọc trạng thái/dữ liệu. Tham số #1 là địa chỉ thanh ghi của LCD, là địa chỉ cơ bản (hay địa chỉ của lối ra của bộ giải mã mắc với chân E của LCD) nếu là thanh ghi lệnh (RS = 0) hoặc địa chỉ cơ bản +1 nếu là thanh ghi trạng thái hay dữ liệu (RS = 1).

MACRO @WR_LCD < địa chỉ >	
PUSH ACC	; lưu giữ thanh ghi chứa A
PUSH PSW	; lưu giữ thanh ghi trạng thái PSW
PUSH DPH	; lưu giữ thanh ghi địa chỉ ngoài cao DPH
PUSH DPL	; lưu giữ thanh ghi địa chỉ ngoài thấp DPL
MOV DPTR, #1	; nạp tham số 1 cho địa chỉ thanh ghi lệnh hay dữ liệu
MOVB A, @DPTR	; đọc ô nhớ ngoài địa chỉ DPTR vào thanh ghi A
CALL DELAY	; gọi trễ 40 microsec

POP DPL	; hồi phục DPL
POP DPH	; hồi phục DPH
POP PSW	; hồi phục PSW
POP ACC	; hồi phục thanh ghi chứa A
ENDMAC	; kết thúc lệnh gộp
MACRO @RD_LCD < địa chỉ >	
PUSH ACC	; lưu giữ thanh ghi chứa A
PUSH PSW	; lưu giữ thanh ghi trạng thái PSW
PUSH DPH	; lưu giữ thanh ghi địa chỉ ngoài cao DPH
PUSH DPL	; lưu giữ thanh ghi địa chỉ ngoài thấp DPL
MOV DPTR, #1	; nạp tham số 1 cho địa chỉ thanh ghi lệnh hay dữ liệu
MOVB A,@DPTR	; đọc ô nhớ ngoài địa chỉ DPTR vào thanh ghi A
CALL DELAY	; gọi trễ 40 microgiây
POP DPL	; hồi phục DPL
POP DPH	; hồi phục DPH
POP PSW	; hồi phục PSW
POP ACC	; hồi phục thanh ghi chứa A
ENDMAC	; kết thúc lệnh gộp

Hình 13.13. Lệnh gộp WR_LCD và RD_LCD với các thanh ghi của LCD

Ví dụ 13.2 giới thiệu chương trình cho LCD mắc với địa chỉ cơ bản 0E000H viết dưới dạng lệnh gộp. Ví dụ 13.3 giới thiệu chương trình cho LCD mắc với địa chỉ cơ bản 0E000H viết dưới dạng chương trình con.

; Ví dụ 13.2. Chương trình có lệnh gộp	; Ví dụ 13.3. Chương trình có chương trình con
<pre> ORG 00h LJMP MAIN \$INCLUDE MYLIB.ASM MAIN: MOV A, #01 MOV P0, A @WR_LCD 0E000h ; ghi lệnh MOV A, #57H MOV P0, A @WR_LCD 0E001h ; ghi dữ liệu @RD_LCD 0E000h ; đọc trạng thái @RD_LCD 0E001h ; đọc dữ liệu SJMP \$; kết thúc chờ DELAY: MOV R6, #40 DJNZ R7, \$ RET END ; kết thúc dịch </pre>	<pre> ORG 00h LJMP MAIN MAIN: MOV A, #01 ; ghi lệnh Xóa MOV P0, A CALL WR_CMD ; ghi lệnh MOV A, #57H ; ghi dữ liệu = 57H MOV P0, A CALL SEND_DATA ; ghi dữ liệu CALL RD_CMD ; đọc trạng thái CALL RD_DATA ; đọc dữ liệu SJMP \$; kết thúc chờ DELAY: MOV R6, #40 DJNZ R7, \$ RET \$INCLUDE MYLIB.ASM END ; kết thúc dịch </pre>

13.4.4. Các chương trình con thông dụng

Từ 4 chương trình con cơ bản hay lệnh gộp cơ bản, chúng ta có thể viết được các chương trình con thường dùng cho LCD. Bảng 13.12 giới thiệu các chương trình con thông dụng cho LCD. Đây chính là các chương trình con được viết để thực hiện các lệnh của bảng lệnh 13.12 của LCD.

Bảng này chỉ có một số chương trình con thông dụng, chúng ta có thể tự viết lấy cho các chương trình con khác theo bảng lệnh 13.12.

Cách viết đơn giản nhất là dùng lệnh gộp với thông số là địa chỉ thanh ghi lệnh của LCD và mã lệnh tương ứng. Để tiện dụng, người ta đặt tên chương trình con theo tên là hành

động của lệnh như xoá màn hình, đưa con trỏ về đầu dòng v.v... và đưa mã lệnh vào trong chương trình con. Chúng ta có thể viết lấy các chương trình con này theo quy tắc trên.

LCD thông dụng đều có hai dòng, do đó, trong mỗi chương trình con, ta phải đưa hai mã lệnh cho hai dòng hay hai phần của LCD.

Chương trình con Khởi phát (lcd_ini) là dài nhất, có hai loại chương trình cho LCD 4 bit và LCD 8 bit.

Thực hiện chương trình Khởi phát cho LCD theo trình tự là khởi phát, xoá màn hình, đưa con trỏ về đầu dòng, đưa vị trí con trỏ về dòng thứ nhất hoặc thứ hai, đưa hiển thị ký tự hay số. Ngoài ra, nếu có yêu cầu hiển thị khác, ta có thể thực hiện di chuyển con trỏ tới vị trí bất kỳ, dấu con trỏ, nhấp nháy con trỏ hay nhấp nháy dòng ký tự hay số v.v...

1. Chương trình con khởi phát

a) Chương trình con Khởi phát LCD 4 bit

LCD 4 bit được khởi phát theo giải thuật như ở hình 13.4a và mã lệnh như ở bảng 13.13.

b) Chương trình con Khởi phát LCD 8 bit

LCD 8 bit được khởi phát theo giải thuật như ở hình 13.14b và mã lệnh như ở bảng 13.14.

Bảng 13.12. Các chương trình con thông dụng

Tên	Hành động	Dữ liệu	Tên trình	Hành động	Gọi tham số
Lcd_ini4	Khởi phát LCD 4 bit	Không	lcd_coff	Dấu con trỏ	Không
Lcd_ini8	Khởi phát LCD 8 bit		lcd_conb	Nhấp nháy con trỏ	
Lcd_home	Con trỏ đầu dòng	Không	lcd_pos1	Con trỏ ở dòng 1	ACC: vị trí
Lcd_clr	Xoá LCD	Không	lcd_pos2	Con trỏ ở dòng 2	ACC: vị trí
Lcd_con	Hiện con trỏ	Không	lcd_char	Hiện ký tự	ACC: mã ASCII

c) Các chương trình con thông dụng khác

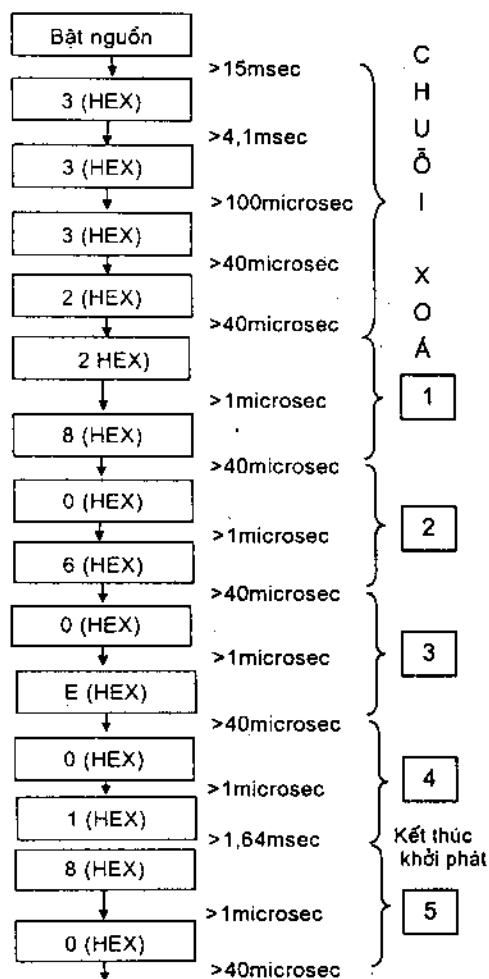
Lcd_clr: Mov a, #00h ; xoá phần 1 Call send_cmd ; gửi lệnh Mov a, #10h ; xoá phần 2 Call send_cmd ; gửi lệnh Call del_5ms ; trễ 5ms Ret	Lcd_cON: Mov a, #00h ; cho phần 1 Call send_cmd ; gửi lệnh Mov a, #0E0h ; cho phần 2 Call send_cmd ; gửi lệnh Ret
Lcd_home: Mov a, #00h ; cho phần 1 Call send_cmd ; gửi lệnh Mov a, #20h ; cho phần 2 Call send_cmd ; gửi lệnh Call del_5ms ; trễ 5ms ret	Lcd_conb: Mov a, #00h ; cho phần 1 Call send_cmd ; gửi lệnh Mov a, #0F0h ; cho phần 2 Call send_cmd ; gửi lệnh ret
Lcd_cON: Mov a, #00h ; cho phần 1 Call send_cmd ; gửi lệnh Mov a, #0E0h ; cho phần 2 Call send_cmd ; gửi lệnh Ret	Lcd_cOFF: Mov a, #00h ; cho phần 1 Call send_cmd ; gửi lệnh Mov a, #0C0h ; cho phần 2 Call send_cmd ; gửi lệnh Ret

Bảng 13.13. Các lệnh khởi phát LCD 4 bit

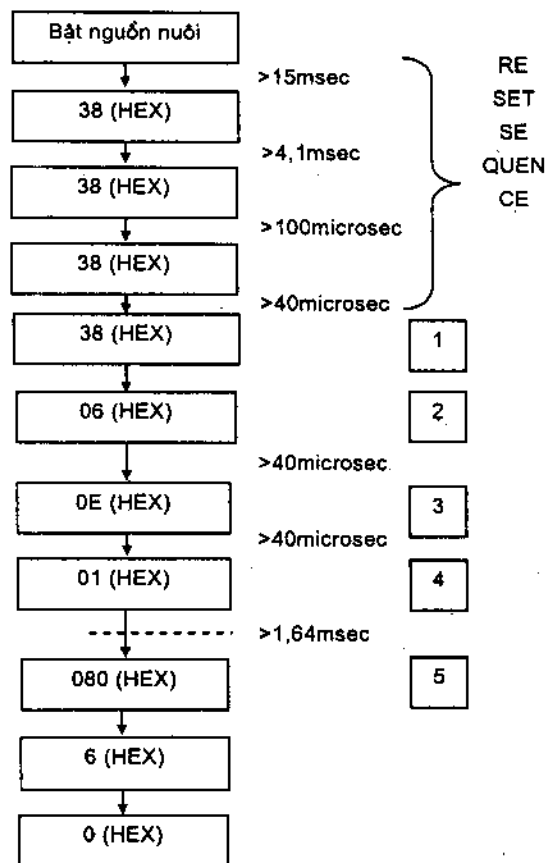
Bước	Mã lệnh	Hành động
1	2 (HEX) 8 (HEX)	Xác lập chức năng: LCD 4 bit 2 dòng, ma trận 5x7
2	0 (HEX) 6 (HEX)	Xác lập chế độ vào: tăng, không dịch con trỏ
3	0 (HEX) E (HEX)	Xác lập chế độ vào: tăng, không dịch hiển thị ON, con trỏ ON, nhấp nháy
4	0 (HEX) 1 (HEX)	Xoá màn hình
5	8 (HEX) 0 (HEX)	Xác lập địa chỉ DD RAM Con số thứ nhất

Chú ý:

- Cả hai tín hiệu RS và R/W kết thúc sẽ là “0” trong chuỗi.
- RS, R/W và Dữ liệu được chốt ở sườn xuống của E.
- Tín hiệu E phải được gửi vào sau mỗi lần truyền 4 bit dữ liệu.



Hình 13.14a. Giải thuật khởi phát LCD 4 bit



Hình 13.14b. Giải thuật khởi phát LCD 8 bit

Bảng 13.14. Các lệnh khởi phát LCD 8 bit

Bước	Mã lệnh	Hành động
1	38 (HEX)	Xác lập chức năng: 8bit, 2 dòng, 5x7
2	06 (HEX)	Xác lập chế độ vào: Tăng con trỏ lên 1, không dịch
3	0E (HEX)	ON/OFF hiển thị: Hiển thị ON Con trỏ ON Nhấp nháy OFF
4	01 (HEX)	Xoá hiển thị
5	80 (HEX)	Xác lập địa chỉ DD RAM số thứ 1

Chú ý:

- Cả hai RS, R/W kết thúc là “0” ở cuối chuỗi.
- RS, R/W và dữ liệu được chốt ở bit cuối của sườn xung xuống của tín hiệu E.
- So sánh hai giải thuật của chương trình khởi phát LCD 4 bit và 8 bit ta thấy:
 - Cả hai đều trải qua 5 bước, nhưng LCD 4 bit phức tạp hơn, mỗi bước đều có hai hành động, trong đó hành động có mã lệnh 0 (HEX) đều mở đầu các bước 2, 3 và 4, riêng bước 5 hành động 0 sẽ kết thúc bước.
 - Khởi phát theo 5 bước cho 5 hành động:
 - + *Xác lập chức năng*: 4 hay 8 bit, 2 dòng, ma trận 5x7.
 - + *Xác lập chế độ vào*: tăng lên 1 vị trí, không dịch.
 - + *Điều khiển hiển thị ON/OFF*: hiển thị ON, con trỏ ON, tắt (OFF) nhấp nháy.
 - + *Xoá màn hiển thị*.
 - + *Xác lập địa chỉ DD RAM*: vị trí đầu tiên của địa chỉ DD RAM.
 - Sau mỗi bước, phải có một thời gian trễ xác định để LCD thực hiện lệnh. Thời gian trễ cỡ microsec không cần đưa vào, nhưng thời gian cỡ msec phải đưa vào.

Ví dụ 13.4 giới thiệu chương trình khởi phát cho LCD hoạt động 8 bit, mắc ở địa chỉ E000H. Do đó, địa chỉ thanh ghi lệnh là 0E000H, thanh ghi dữ liệu là E001H. Chương trình này tuân theo giải thuật ở hình 13.14b và bảng 13.13. Trong chương trình dùng các thời gian trễ DELAY1, DELAY2 và DELAY3 cho 100msec, 10msec và 1msec và chương trình con cơ bản Ghi lệnh SEND_CMD cho LCD. Nếu lệnh con cơ bản SEND_CMD đã ghi trong thư viện MYLIB.ASM thì đầu chương trình chỉ cần khai báo \$INCLUDE MYLIB.ASM, không cần chép lại chương trình con cơ bản SEND-CMD nữa.

Ví dụ 13.4. Chương trình khởi phát LCD 8 bit mắc với địa chỉ E000H	
CALL DELAY1 ; trễ 100ms MOV A, #38H CALL SEND_CMD ; ghi lệnh CALL DELAY2 ; trễ 10ms MOV A, #38H CALL SEND_CMD ; ghi lệnh CALL DELAY3 ; trễ 1ms MOV A, #38H CALL SEND_CMD ; ghi lệnh	CALL DELAY3 POP DPL POP DPH POP PSW POP ACC RET DELAY1: MOV R6, #100 ; trễ 10ms DLAY1: MOV R7, #250 DLAY2: NOP

MOV A, #0CH ; lệnh bật hiển thị	DJNZ R7, DLAY2
CALL SEND_CMD ; ghi lệnh	DJNZ R6, DLAY1
MOV A, #01H ; lệnh Xoá hiển thị	RET
CALL SEND_CMD ; ghi lệnh	DELAY2: MOV R6, #10 ; trễ 10ms
MOV A, #06H ; lệnh con trỏ tăng	DLAY3: MOV R7, #250
CALL SEND_CMD ; ghi lệnh	DLAY4: NOP
MOV A, #80H ; lệnh Ghi địa chỉ DD RAM	NOP
CALL SEND_CMD	DJNZ R7, DLAY4
SEND_CMD:	DJNZ R6, DLAY3
PUSH ACC	RET
PUSH PSW	DLAY3: MOV R7, #250 ; trễ 1ms
PUSH DPH	DLAY5: NOP
PUSH DPL	NOP
MOV DPTR, #0E000H	DJNZ R7, DLAY5
MOVX @DPTR, A	RET

2. Chương trình con và lệnh gộp kiểm tra bit bận

Khi đọc thanh ghi trạng thái với chương trình con RD_CMDTT, RD_CMDTG (kiểu điều khiển trực tiếp) hay RD_CMD (kiểu điều khiển địa chỉ) chúng ta thu được trong thanh ghi A bit DB7 dành cho bit trạng thái bận (LCD đang hoạt động bên trong) của LCD và các bit còn lại là bộ đếm địa chỉ hay tọa độ con trỏ của LCD. Chính vì có BF = 1 này mà mỗi lệnh phải có thời gian trễ (xem bảng lệnh) để các lệnh sau đưa vào, LCD mới nhận và thực hiện. Vì các lệnh khác nhau có thời gian trễ khác nhau, nên thời gian trễ này không thể chung cho các lệnh. Nếu chọn thời gian trễ đủ lớn cho mọi lệnh, lại tốn nhiều thời gian cho LCD hoạt động. Tốt nhất, dùng chương trình kiểm tra bit, bật cho tới khi BF = 0 mới kết thúc chương trình con trước và bắt đầu thực hiện chương trình con sau.

Chương trình con kiểm tra bit BF như ở hình 13.15.

Chương trình này có các hành động sau:

– Đọc thanh ghi trạng thái (bằng một trong ba chương trình con RD_CMDTT, RD_CMDTG hay RD_CMD.

– Kiểm tra bit BF chứa trong bit Acc.7. Nếu BF = 1 trở về LOOP để đọc lại bởi lệnh JB ACC.7, LOOP cho tới khi BF = 0.

Có thể dùng chương trình con này thay cho chương trình con làm trễ thời gian DELAY sau mỗi chương trình con ghi lệnh cho LCD.

3. Chương trình con và lệnh gộp xác định vị trí con trỏ

Trước khi cho hiển thị chữ hay số lên LCD, ta phải xác định vị trí hiển thị của chúng. Có hai chương trình con và một lệnh gộp dùng để xác định vị trí hiển thị tại dòng 1 hay dòng 2 của LCD.

Hình 13.16a giới thiệu hai chương trình con trên còn hình 13.16b là lệnh gộp xác định vị trí dòng.

CHECK_BF:	; kiểm tra BF
PUSH ACC	
LOOP: CALL RD_CMD	; đọc trạng thái
JB ACC.7, LOOP	; về LOOP nếu = 1
POP ACC	
RET	

Hình 13.15. Chương trình con kiểm tra BF

Chương trình con hiển thị tại dòng 1 của LCD	Chương trình con hiển thị tại dòng 2 của LCD
Lcd_pos1: PUSH ACC ANL A, #70H ; lấy 3 bit cao = 1 ORL A, #80H ; bit DB7 = 1 CALL LCD_WR ; gọi ghi dữ liệu POP ACC SWAP A ; trao hai nibble của A ANL A, #0F0H ; giữ lại 4 bit cao CALL LCD_WR ; gọi ghi dữ liệu POP ACC RET LCD_WR: chương trình con Ghi dữ liệu.	Lcd_pos2: PUSH ACC ANL A, #70H ; lấy 3 bit cao = 1 ORL A, #0C0H ; bit DB7 = DB6 = 1 CALL LCD_WR ; gọi ghi dữ liệu POP ACC SWAP A ; trao hai nibble của A ANL A, #0F0H ; giữ lại 4 bit cao CALL LCD_WR ; gọi ghi dữ liệu POP ACC RET LCD_WR: chương trình con Ghi dữ liệu.

Hình 13.16a. Các chương trình con xác định vị trí con trỏ

Từ hai chương trình con ở trên, chúng ta có thể viết lệnh gộp chung để xác định vị trí như sau:

MACRO @LCD_POS < tham số dòng>; 80H cho dòng 1, ; 0C0H cho dòng 2 PUSH ACC ANL A, #70H ; lấy 3 bit cao = 1 ORL A, #1 ; bit DB7 = 1 (dòng 1) hay DB7 = DB6 = 1 (dòng 2) CALL LCD_WR ; gọi ghi dữ liệu POP ACC SWAP A ; trao hai nibble của A ANL A, #0F0H ; giữ lại 4 bit cao CALL LCD_WR ; gọi ghi dữ liệu POP ACC ENDMAC

Hình 13.16b. Lệnh gộp xác định vị trí con trỏ

13.4.5. Chương trình hiển thị ký tự

Ở mục trên, chúng ta mới dùng lệnh cơ bản để lập trình ghi một số lệnh điều khiển LCD. Ở đây, chúng ta lập trình cho VDK họ MCS51 để hiển thị ký tự hay chữ.

1. Chương trình con và lệnh gộp hiển thị chữ trong thanh ghi A

Để hiển thị hai chữ dưới dạng mã ASCII trong thanh ghi A, và lệnh gộp hiện một chữ như hình 13.17. Khi dùng lệnh gộp, ta phải điền tham số 1 là ký tự cần hiển thị.

Có thể dùng lệnh gộp này hiển thị một chuỗi ký tự nối tiếp bằng dùng nhiều lệnh gộp với tham số là các ký tự liên tiếp.

Chương trình con hiển thị chữ ở A ra LCD	Lệnh gộp hiển thị chữ trên LCD
Lcd_char: PUSH ACC ANL A, #0F0H ; lấy 4 bit cao ORL A, #01H ; bit DB7 = 1 CALL LCD_WR ; gọi ghi dữ liệu POP ACC SWAP A ; trao hai nibble của A ANL A, #0F0H ; giữ lại 4 bit cao OR A, #1 CALL LCD_WR ; gọi ghi dữ liệu POP ACC RET	MACRO @LCD_CHAR <1> ; 1- chữ MOV A, #1 ; chữ ADD A, #30H ; đổi mã ASCII ORL A, #01H ; bit DB7 = 1 CALL LCD_WR ; gọi ghi dữ liệu
a)	b)

Hình 13.17. Chương trình (a) và lệnh gộp (b) hiển thị một ký tự trong thanh ghi A

Để hiển thị một chữ dưới dạng mã ASCII trong thanh ghi A tại hai dòng 1 và 2 của LCD, ta có các chương trình con như hình 13.18.

Chương trình con hiển thị tại dòng 1 của LCD	Chương trình con hiển thị ở dòng 2 của LCD
Lcd_charpos1: PUSH ACC MOV A, R2 ; vị trí trong R2 CALL Lcd_pos1 ; gọi chương trình ghi vị trí 1 MOV A, R1 ; mã ASCII của chữ ở R1 CALL Lcd_char ; gọi chương trình ghi chữ POP ACC RET Lcd_pos1 ; gọi chương trình ghi vị trí. CALL Lcd_char ; gọi chương trình ghi chữ	Lcd_charpos2: PUSH ACC MOV A, R2 ; vị trí trong R2 CALL Lcd_pos2 ; gọi chương trình ghi vị trí 2 MOV A, R1 ; mã ASCII của chữ ở R1 CALL Lcd_char ; gọi chương trình ghi chữ POP ACC RET Lcd_pos2 ; chương trình con ghi lệnh. CALL Lcd_char ; gọi chương trình ghi chữ

Hình 13.18. Chương trình hiển thị một chữ trên dòng 1 hay 2 của LCD

Ví dụ 13.6 giới thiệu chương trình đưa ra hiển thị nhiều chữ dạng mã ASCII trên LCD theo lần lượt từng chữ.

Ví dụ 13.5: Chương trình hiện chữ “HELLO A”	
\$INCLUDE MYLIB>ASM LCD_CM EQU 0E000h ORG 00h LJMP MAIN MAIN: CALL INIT CALL CLR_LCD MOV A, #82H CALL SEND_CMD MOV A, #“H” CALL SEND_DATA MOV A, #“E”	CALL SEND_DATA MOV A, #“L” CALL SEND_DATA MOV A, #“L” CALL SEND_DATA MOV A, #“O” CALL SEND_DATA MOV A, #“A” CALL SEND_DATA SJMP \$; kết thúc chờ END

2. Chương trình hiển thị chuỗi chữ trong bộ nhớ RAM ngoài hay ROM trong

Chương trình ở ví dụ 13.5 đưa chuỗi chữ “HELLO A” theo từng chữ một. Có thể viết lại chương trình trên theo kiểu đưa chuỗi chữ bằng vòng lặp đọc từng chữ (bởi lệnh MOVX A, @A+DPTR của bảng chứa các chữ cần hiển thị ở bộ nhớ ngoài của hệ VDK, tới khi có dấu hiệu kết thúc chuỗi chữ, ví dụ là số “0” (mã ASCII là 30H) hay 99H. Chương trình này cũng tương tự việc đọc chuỗi chữ của bảng ghi trong bộ nhớ ROM bên trong VDK.

Ví dụ 13.6 giới thiệu chương trình hiển thị chuỗi chữ “MICROCONTROLLER” và “Hello” ở ghi ở bảng trong bộ nhớ ROM ra hai dòng trên LCD.

; Ví dụ 13.6: Chương trình đưa chuỗi chữ “MICROCONTROLLER” ra LCD	
\$INCLUDE MYLIB.ASM LCD_CM EQU 0E000H ; địa chỉ ghi lệnh LCD_WR EQU 0E001H; địa chỉ ghi dữ liệu ORG 00H LJMP MAIN MAIN: ; khởi phát LCD MOV A, #38H CALL SEND_CMD MOV A, #0EH	STRING1: DB “MICROCONTROLLER” 0 STRING2: DB “Hello” 0 LCD_PRINTSTRING: PUSH ACC MOV A, R3 PUSH ACC MOV R3, #00 LCDN: MOV A, R3

CALL SEND_CMD MOV A, #06H CALL SEND_CMD CALL LCD_CLR MOV DPTR, #STRING1 CALL LCD_PRINTSTRING CLR A MOV B, #1 @LCD_POS 1 MOV DPTR, #STRING2 CALL LCD_PRINTSTRING SJMP \$	MOVC A, @A+DPTR CJNE A, #0, LCDGO POP ACC MOV R3, A POP ACC RET LCDGO: MOV R1, A @LCD_POS 2 INC R3 INC R2 JMP LCDN END
--	---

3. Chương trình hiển thị chuỗi chữ hay đồ hoạ trong CG RAM

Chúng ta có thể tạo, ghi vào CG RAM rồi hiển thị lên màn hình của LCD. Ví dụ 13.7 giới thiệu chương trình hiển thị với ROM của VĐK và CG RAM của LCD. Đây là chương trình ví dụ của hãng Hantronix nên ký hiệu hơi khác với các ký hiệu đã dùng.

Dòng thứ nhất của LCD hiện chuỗi chữ <<< HANTRONIX<<<.

Dòng thứ hai của LCD hiện hình trái tim với 8 hàng font1 tới font8.

; Ví dụ 13.7: Chương trình với CG RAM	
EnableT0 equ 082h ; cho phép Timer0 Disable equ 000h ; cấm Timer D4100h equ 00Ch ; nạp lại timer byte cao D4100l equ 003h ; nạp lại timer = 4,1msec D100h equ 000h ; nạp lại timer D4100l equ 04Ch ; nạp timer = 100microsec Flag DATA 020h TstFlag BIT Flag.0 ; nối cổng P1 với D0- D7 ; P3.0 với Enable ; P3.1 với RS ; P3.2 với RW ORG 00H LJMP MAIN ORG 0BH LJMP Timer0 ORG 50H MAIN: CLR TR0 CLR TF0 ; khởi phát LCD CLR P3.1 CLR P3.2 SETB P3.0 MOV RTL, #D4100l MOV RTH, #D4100h MOV P1, #38H	CG RAM: MOV P1, #40H ACALL COMMAND_BYTE MOV DPTR, #CG_RAM_DATA ACALL WRITE RET FIRST_LINE: MOV P1, #80H ACALL COMMAND_BYTE MOV DPTR, FLINE_DATA ACALL WRITE RET SECOND_LINE: MOV P1, #0C0H ACALL COMMAND_BYTE MOV DPTR, SLINE_DATA ACALL WRITE RET COMMAND_BYTE: CLR P3.1 JMP BDELAY DATA_BYTE: SETB P3.1 NOP BDELAY: CLR P3.2 CLR P3.0 NOP SETB P3.0 NOP ; check busy flag MOV P1, #0FFH

ACALL COMMAND_BYTE ACALL DDELAY MOV RTL, #D100I MOV RTH, #D4100h MOV P1, #38H ACALL COMMAND_BYTE MOV P1, #0CH ACALL COMMAND_BYTE MOV P1, #01H ACALL COMMAND_BYTE ACALL CG RAM ACALL FIRST_LINE ACALL SECOND_LINE SDONE: SETB IDLE JMP SDONE ; các chương trình con WRITE_LOOP: MOV A, #0 MOVC A, @A+DPTR CJNE A, #99H, WRITE_CONT RET WRITE_CONT: MOV P1, A ACALL DATA_BYTE INC DPTR JMP WRITE_LOOP DDELAY: SETB TR0 MOV IE, EnableT0 CLR TstFlag Dloop: jnb TstFlag, Dloop RET	SETB P3.2 CLR P3.1 CLR P3.0 BLOOP: NOP MOV A, P1 ANL A, #80H CJNE A, #00H, BLOOP BWAIT: MOV A, P1 ANL A, #80H CJNE A, #00H, BLOOP CLR P3.2 RET TIMER0: SETB TstFlag CLR TR0 MOV IE, #DISABLE RETI FLINE_DATA: DB ">>> HANTRONIX <<<" DB 99H SFLINE_DATA: DB 00H, 01H, 02H, 03H, 04H, 05H, 06H, 07H, 99H CG RAM: DATA: Font1: db 0ah, 15h, 11h, 11h, 0ah, 04h, 00h, 00h Font2: db 04h, 0ah, 11h, 11h, 15h, 0ah, 00h, 00h Font3: db 04h, 0eh, 15h, 04h, 04h, 04h, 00h, 00h Font4: db 04h, 04h, 04h, 04h, 15h, 0eh, 04h, 00h Font5: db 18h, 18h, 1fh, 1fh, 1fh, 18h, 18h, 00h Font6: db 1fh, 1fh, 03h, 03h, 03h, 1fh, 1fh, 00h Font7: db 0ah, 15h, 0ah, 15h, 0ah, 15h, 0ah, 00h Font8: db 15h, 0ah, 15h, 0ah, 15h, 0ah, 15h, 00h db 99h END
--	---

13.4.6. Chương trình hiển thị số

Khác với đèn LED 7 đoạn hay ma trận diot, các số hiển thị lên LCD cũng có dạng mã ASCII như ký tự. Do đó việc hiển thị chữ số lên LCD cũng tương tự hiển thị chữ (ký tự). Dữ liệu dưới dạng mã ASCII ghi trong thanh ghi chứa A và cũng đưa ra bằng lệnh đưa thanh ghi A ra ô nhớ ngoài có địa chỉ DPTR, tức lệnh MOVX @DPTR, A, trong đó DPTR là địa chỉ nhớ ngoài dành cho thanh ghi dữ liệu của LCD mắc với hệ Vi điều khiển. .

Thanh ghi chứa A chứa được hai số BCD có nén, nên chứa được hiển thị được một hay hai chữ số dạng BCD, do đó, nó có thể đưa ra hiển thị được một hay hai số mã mã ASCII lên LCD.

Nếu cần đưa hiển thị số nhị phân trong một byte hay hai byte, ta phải biến đổi mã nhị phân ra mã ASCII để đưa ra hiển thị trên LCD.

1. Hiển thị một chữ số BCD trong thanh ghi A

Ví dụ 13.8a giới thiệu chương trình con đưa hiển thị một số BCD không nén ra LCD. Còn ví dụ 13.8b giới thiệu lệnh gộp đưa hiển thị một số BCD không nén trong thanh ghi A ra LCD.

; Ví dụ 13.8a: Chương trình con đưa 1 chữ số	; Ví dụ 13.8b: Lệnh gộp con đưa 1 chữ số
LCD_1DIGIT: MOV A, R4 ; chữ số trong R4 ANL A, #0FH ; lọc 4 bit thấp JB LZB, SHAL5 ; bit cờ LZB = 1 tới SHAL5 JZ LED5B ; LZB = 0, hiển thị dấu cách SHAL5: SETB LZB ; xác lập cờ LZB ADD A, #'0' ; biến đổi thành mã ASCII MOV R1, A JMP LCD6B ; tới LCD6B LCD5B: MOV R1, # "" LCD6B: CALL Lcd_char_pos1; hiển thị mã RET	MACRO @LCD_1DIGIT <SO> MOV A, #1 ANL A, #0FH JB LZB, SHAL5 JZ LED5B SHAL5: SETB LZB ADD A, #'0' MOV R1, A JMP LCD5B LCD6B: CALL Lcd_char_pos1 ENDMAC

Ví dụ 13.9 giới thiệu chương trình dùng lệnh gộp đưa số 25D gồm hai số BCD là 2 và 5 ra LCD. Trong chương trình dùng hai lệnh gộp, lần lượt đưa từng số: số hàng chục (2) ra trước, số hàng đơn vị (5) ra sau. Với lệnh gộp@LCD_1DIGIT <SO>, ta có thể viết chương trình đưa ra hiển thị một con số có số chữ số bất kỳ bằng cách đưa hiển thị từng chữ số của con số đó, bằng cách lặp lại từng lệnh gộp và thay tham số SO bằng chữ số của mỗi hàng (triệu, chục vạn, vạn, chục ngàn, ngàn, trăm, chục, đơn vị v.v...).

; Ví dụ 13.9: Chương trình đưa hiển thị số 25D có hai chữ số ra LCD	
MACRO @LCD_1DIGIT <SO> MOV A, #1 ANL A, #0FH JB LZB, SHAL5 JZ LED5B SHAL5: SETB LZB ADD A, #'0' MOV R1, A JMP LCD5B LCD5B: MOV R1, # "" LCD6B: @LCD_CHAR_POS 1 ENDMAC	SINCLUDE MYLIB.ASM; gọi ;LCD_CHAR_POS 1 ORG 00H @LCD_1DIGIT 2 ; hiển thị số 2 @LCD_1DIGIT 5 ; hiển thị số 5 SJMP \$ END ; Chú ý: Nếu lệnh gộp đã chứa trong thư viện MYLIB.ASM thì chương trình không cần chép lệnh gộp @LCD_1DIGIT <SO> ở cột bên trái.

2. Hiển thị hai chữ số BCD nén trong thanh ghi A

Nếu thanh ghi A 8 bit chứa được hai số BCD dạng nén, tức số hàng chục và hàng đơn vị. Ta có thể viết chương trình con và lệnh gộp đưa hiển thị cho hai số dạng BCD nén trong thanh ghi A như ví dụ 13.10. Mỗi chương trình con hay lệnh gộp này đều có hai phần 4 bit tương tự như của ví dụ 13.9. Chương trình đưa hai chữ số dùng lệnh gộp @LCD_2DIGIT <SO1, SO2> tương tự ví dụ 13.10.

; Ví dụ 13.10a: Chương trình con đưa 2 chữ số	Ví dụ 13.10b: Lệnh gộp đưa 2 chữ số
LCD_2DIGIT: MOV A, R4 ; chữ số trong R4 vào A ANL A, #0F0H ; lọc 4 bit thấp	MACRO @LCD_2DIGIT <SO1, SO2> MOV R4, #1 MOV A, R4

SWAP A ; đổi sang 4 bit thấp JB LZW, SHAL1; bit cờ LZW = 1 tới SHAL1 JZ LCD2A ; LZW = 0 SHAL1: SETB LZW; xác lập cờ LZW ADD A, #'0' ; biến đổi thành mã ASCII MOV R1, A ; gửi vào R1 JMP LCD3A ; tới LCD3A LCD2A: MOV R1, # " "; gửi khoảng trống LCD3A:CALL Lcd_char_pos1; hiển thị mã ASCII MOV A, R4 ; chữ số trong R4 vào A ANL A, #0FH ; giữ 4 bit thấp JB LZW, SHAL2; bit cờ LZW = 1 tới SHAL2 JZ LCD2B ; LZW = 0, tới LCD2B SHAL2: SETB LZW; xác lập cờ LZW ADD A, #'0' ; biến đổi thành mã ASCII MOV R1, A ; gửi vào R1 JMP LCD3B ; tới LCD3B LCD2B: MOV R1, # " "; gửi khoảng trống LCD3B: INC R2 ; tăng địa chỉ con trỏ CALL Lcd_char_pos1; hiển thị mã ASCII RET	ANL A, #0F0H SWAP A JB LZW, SHAL1 JZ LED2A SHAL1: SETB LZW ADD A, #'0' MOV R1, A JMP LCD5A LCD2A:MOV R1, # " "; gửi khoảng trống LCD3A: @LCD_CHARPOS 1 MOV R4, #12 MOV A, R4 ; chữ số trong R4 vào A ANL A, #0FH ; giữ 4 bit thấp JB LZW, SHAL2; bit cờ LZW = 1 tới JZ LCD2B ; LZW = 0, tới LCD2B SHAL2: SETB LZW; xác lập cờ LZW ADD A, #'0' ; biến đổi mã ASCII MOV R1, A ; gửi vào R1 JMP LCD3B ; tới LCD3B LCD2B: MOV R1, # " " LCD3B: INC R2 ; tăng địa chỉ con trỏ @LCD_CHARPOS 1 ENDMAC
---	---

3. Hiển thị các chữ số trong một byte nhị phân

Tương tự hiển thị các chữ số trong một byte nhị phân ra LED 7 đoạn hay ra ma trận diot, ta phải tiến hành 2 bước:

Biến đổi byte nhị phân ra các byte BCD không nén rồi đưa vào các thanh ghi R6 (số hàng trăm), R5 (số hàng chục) và R4 (số hàng đơn vị, xem các chương trước).

– Hiển thị các chữ số BCD không nén trong các thanh ghi R7, R6 và R5.

Về nguyên tắc, ta có thể dùng chương trình con hay lệnh gộp để hiển thị. Ở đây chúng ta dùng lệnh gộp cho đơn giản. Lệnh gộp @LCD_1DIGITM này không nhận tham số trực tiếp mà từ một trong các thanh ghi R5, R6 hay R7 đưa vào thanh ghi A. Ví dụ 13.11 giới thiệu chương trình biến đổi dữ liệu dạng nhị phân trong thanh ghi TH ra LCD.

Ví dụ 13.11: Chương trình hiển thị 1 byte nhị phân ra LCD	
macro @bdb_bcd mov a, #SO ; số nhị phân 1 byte mov b, #10 div ab mov r5, b ; r5 chứa số đơn vị mov b, #10 div ab mov r7, a ; r7 chứa số hàng trăm	ADD A, #'0' MOV R1, A JMP LCD5B LCD5B: MOV R1, # " " LCD6B: @LCD_CHAR_POS 1 ENDMAC ORG 00H MOV SO, TH; thanh ghi TH chứa số nhị phân

mov r6, b ; r6 chứa số hàng chục endmac MACRO @LCD_1DIGITM MOV A, #THANHGHI ANL A, #0FH JB L5B. SHAL5 JZ LED5B SHAL5: SETB L5B	@bdb_bcd MOV THANH GHI, R7 @LCD_1DIGITM ; hiển thị số hàng trăm MOV THANH GHI, R6 @LCD_1DIGITM ; hiển thị số hàng chục MOV THANH GHI, R5 @LCD_1DIGITM ; hiển thị số hàng đơn vị SJMP \$ END
---	---

4. Hiển thị các chữ số trong hai byte nhị phân

Hai byte nhị phân có thể biểu diễn được số thập phân lớn nhất là 65536D. Trong VDK, các thanh ghi của Timer/Counter là 2 byte. Số đếm của chúng đưa hiển thị ra LCD dưới dạng thập phân. Do đó, để hiển thị trên LCD, phải qua hai quá trình:

- Biến đổi số nhị phân BIN ra BCD rồi ghi vào các ô nhớ có tên chục ngàn TTHOU, ngàn THOU, trăm HUN, chục TEN và đơn vị UNIT (xem chương 11 và 12).

Đưa các số BCD TTHOU, THOU, HUN, TEN và UNIT ra LCD nhờ lệnh gộp @LCD_1DIGITM.

5. Hiển thị các chữ số có chấm ở giữa

Đôi khi cần hiển thị số thập phân hay có dấu chấm (.) giữa hàng đơn vị và hàng phần mười, ta cần hiển thị dấu chấm phân cách giữa hai chữ số trên.

Ví dụ 13.12 giới thiệu chương trình con hiển thị hai số trong thanh ghi R4 có chấm phân cách. Như vậy, thanh ghi R4 chứa 8 bit cho hai số BCD nén. Khi hiển thị, ta phải hiển thị số BCD ở 4 bit cao, hiển thị dấu chấm, rồi hiển thị tiếp số BCD trong 4 bit thấp.

: Ví dụ 13.12: Chương trình hiện một số trong hai byte nhị phân (hay 4 số HEX)	
: \$INCLUDE MYLIB.ASM ORG 001H LJMP MAIN MACRO @BINBCD : <1.NHAN, 2.LBYTE, 3.HBYTE, 4.BAC> II: CLR C MOV A, LBYTE SUBB A, #2 MOV R0, A MOV A, HBYTE SUBB A, #3 JC KT MOV HBYTE, A MOV A, R0 MOV LBYTE, A INC I4 JMP II KT: ENDMAC	BIN_BCD: PUSH ACC PUSH PSW SETB RS0 ; BANK1 CLR RS1 MOVR1, #5 BIN_0: MOV @R0, #0 INC R0 DJNZ R1, BIN_0 @BINBCD BTD_TT, 10H, 27H, TTHOU @BINBCD BTD_T, 0E8H, 03H, THOU @BINBCD BTD_HUN, 64H, 00H, HUN @BINBCD BTD_TEN, 0AH, 00H, TEN BTD_DV: MOV A, LBYTE MOV UNIT, A POP PSW MOV THANH GHI, TTHOU @LCD_1DIGITM ; hiển thị số chục ngàn MOV THANH GHI, THOU @LCD_1DIGITM ; hiển thị số hàng ngàn

LBYTE DS 1 HBYTE DS 1 TTHOU EQU 30H THOU EQU 31H HUNEQU 32H TENEQU 33H UNIT EQU 34H ORG 100H MAIN: MOV LBYTE, #0BCH; ghi số 0FCH MOV HBYTE, #0F5H ; ghi số 0FFH	MOV THANH GHI, HUN @LCD_1DIGITM; hiển thị số hàng trăm MOV THANH GHI, TEN @LCD_1DIGITM; hiển thị số hàng chục MOV THANH GHI, UNIT SJMP \$ END
--	---

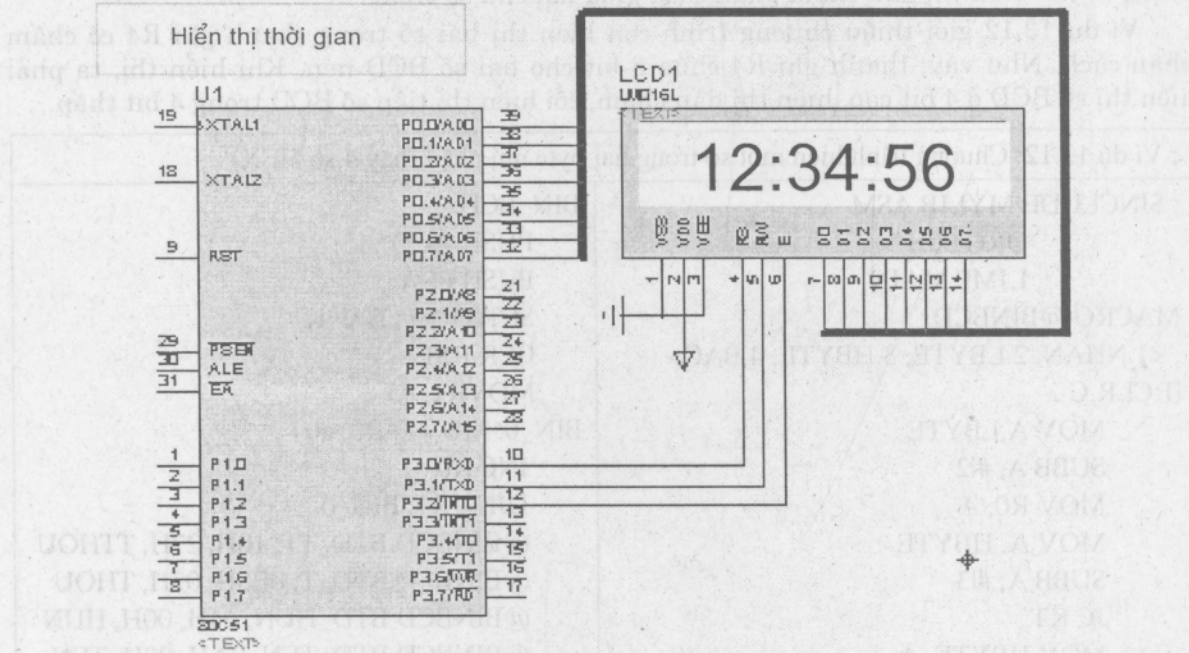
Chú ý: Nếu lệnh gộp @BINBCD đã ghi trong thư viện thì chương trình trên có thể rút ngắn hơn nhiều.

Chúng ta có thể viết chương trình con trên bằng cách dùng lệnh gộp @LCD_1DIGITM cho số thứ nhất, hiện dấu chấm rồi tới số thứ hai.

6. Hiển thị các chữ số biểu diễn thời gian

Các chữ số biểu diễn thời gian (giờ, phút, giây) có dạng hai chữ số phân cách nhau bằng dấu hai chấm (:). Do đó, sau khi hiển thị hai chữ số chỉ giờ, ta phải cho hiển thị dấu hai chấm (:) và cũng tương tự cho các chữ số chỉ phút hay giây.

Hình 13.19 giới thiệu đồng hồ có hiển thị thời gian trên LCD.



Hình 13.19. Đồng hồ thời gian có hiển thị LCD

Chương trình hiển thị thời gian trên LCD như ví dụ 13.13 với 12 giờ, 34 phút và 56 giây.

Trong đồng hồ thể thao, đơn vị là giây, có tới hàng ngàn giây và độ chính xác tới một phần trăm giây. Do đó, dòng hiển thị thời gian có dấu chấm sau đơn vị giây và phần chục và phần trăm giây. Ví dụ 13.13 giới thiệu chương trình con biểu diễn thời gian cho đồng hồ thể thao.

Ví dụ 13.13: Chương trình con hiển thị từ ngàn giây tới phần trăm giây			
HTGIAY:		CALL LCD_1DIGIT	
MOV R4, THOU	; ngàn giây vào R4	MOV R4, #"."	; dấu chấm vào R4
CALL LCD_1DIGIT		CALL LCD_1DIGIT	
MOV R4, HUN	; trăm giây vào R4	MOV R4, TENTH	; 1/10 giây vào R4
CALL LCD_1DIGIT		CALL LCD_1DIGIT	
MOV R4, TEN	; chục giây vào R4	MOV R4, HUNTH	; 1/100 giây vào R4
CALL LCD_1DIGIT		CALL LCD_1DIGIT	
MOV R4, UNIT	; giây vào R4	RET	

Với chương trình hiển thị thời gian cho hình 13.19, ta dùng:

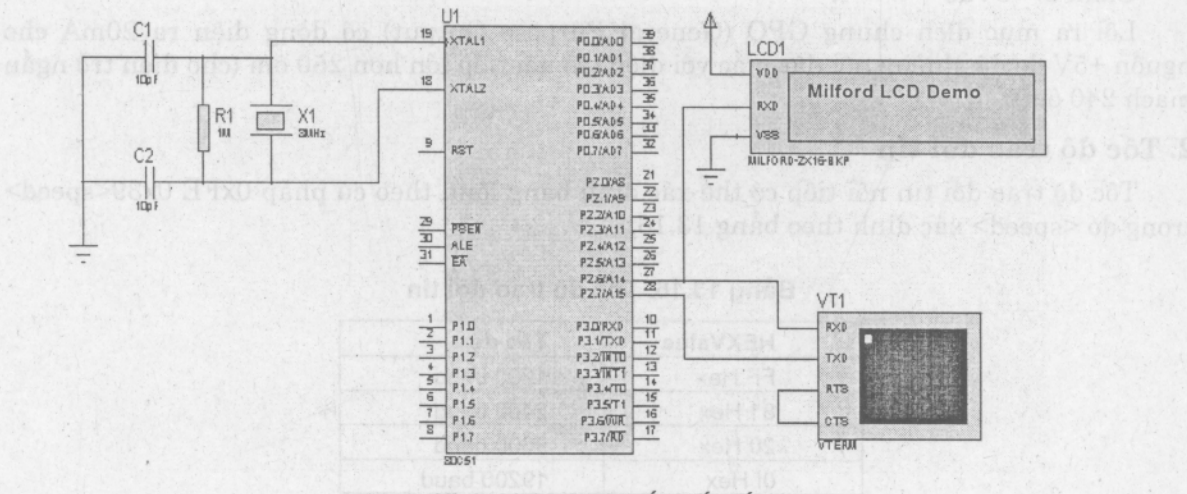
- Chương trình con hay lệnh gộp cho hai chữ số chứa trong R4 vào thanh ghi A.
- Thay dấu chấm (.) bằng dấu hai chấm (:).

Nếu đồng hồ thời gian còn có phần thập phân của giây (phần mười, phần trăm của giây), ta phải đưa hiển thị dấu chấm trước khi đưa hiển thị các số thập phân của giây.

13.5. LCD TRAO ĐỔI TIN NỐI TIẾP

Ngoài LCD mắc song song, người ta còn dùng LCD mắc nối tiếp để tiết kiệm đường dây nối giữa vi điều khiển và LCD. Hình 13.20 giới thiệu cách mắc giữa vi điều khiển và LCD mắc nối tiếp.

So với 14 đường dây nối của LCD mắc song song, LCD mắc nối tiếp này chỉ cần một dây nối nguồn nuôi Vdd và một dây truyền tin nhận nối tiếp RxD. Do đó, LCD mắc nối tiếp này thường được dùng khi VDK nói riêng hay nguồn phát tin nói chung đặt ở xa để tiết kiệm dây nối. Nhưng LCD mắc nối tiếp này phải trả giá là thời gian trao đổi tin lâu, hệ lệnh phức tạp hơn (bằng các mã ASCII).



Hình 13.20. LCD mắc nối tiếp

13.5.1. Các loại LCD nối tiếp

Hãng MATRIX ORBITAL giới thiệu trên thị trường hai loại LCD nối tiếp thông dụng là LCD4041 (160 ký tự = 40 ký tự x 4 dòng) và LCD 4021 80 ký tự = 20 ký tự x 4 dòng).

Với lệnh Đọc giá trị loại modul là ASCII “7” (hay HEX 37H, decimal 55) với cú pháp 0xFE 0x37 ta đọc được các giá trị loại modul như ở bảng 13.15.

Bảng 13.15. Các loại LCD nối tiếp

LCD0821-0X01	LCD2021-0X03	LCD1641-0X04
LCD2041-0X05	LCD4021-0X06	LCD4041-0X07
LK202-25-0X08	LK204-25-0X09	LK404-55-0X0A
VFD2021-0X0B	VFD2041-0X0C	VFD4021-0X0D
VK202-25-0X0E	VK204-25-0X0F	GLC12232-0X10
GLC12864-0X11	GLC128128-0X12	GLC24064-0X13
GLK12864-25-0X14	GLK24064-25-0X15	GLK128128-25-0X21
GLK12232-25-0X22	LK404-AT-0X31	VFD1621-0X32
LK402-12-9X33	LK162-12-0X34	LK204-25PC-0X35

13.5.2. Dây nối và tốc độ truyền nối tiếp

1. Dây nối

Tin nối tiếp giữa VDK và LCD nối tiếp trao đổi theo chuẩn RS-232 và I2C qua ổ cắm DB-9 có 9 chân cắm.

Với trao đổi tin I2C, chỉ dùng 4 chân cắm là:

Chân 1: +5V cho nguồn nuôi.

Chân 2: SCL (nhịp cho I2C).

Chân 3: SDA (dữ liệu cho I2C).

Chân 4: GND nối đất.

Với loại LCD4041-VPT có dải nguồn nuôi rộng, chân 1 cho nguồn nuôi từ +7 tới 15Vdc.

Với trao đổi tin tiếp RS-232, chân nối như sau:

Chân 2: Data out.

Chân 3: Data In.

Chân 5: GND nối đất.

Chân 9: +5Vdc

Lỗi ra mục đích chung GPO (General Purpose Output) có dòng điện ra 20mA cho nguồn +5V do đó, đường nối đất mắc với điện trở nối tiếp lớn hơn 250 ôm (cho điện trở ngắn mạch 240 ôm).

2. Tốc độ trao đổi tin

Tốc độ trao đổi tin nối tiếp có thể xác định bằng lệnh theo cú pháp 0xFE 0x39<speed> trong đó <speed> xác định theo bảng 13.16.

Bảng 13.16. Tốc độ trao đổi tin

HEXValue	Tốc độ
FF Hex	1200 baud
81 Hex	2400 baud
20 Hex	9600 baud
0f Hex	19200 baud

13.5.3. Bộ lệnh

LCD nối tiếp có các lệnh đều là các ký tự dạng mã ASCII và truyền lệnh tuân theo cú pháp:
<command prefix> <command> [<parameter><parameter>]

Ví dụ: Điều khiển Xóa hiển thị 0xFE 0x58 trong đó 58 HEX là mã ASCII của chữ X dành cho Xóa.

Bộ mã lệnh của LCD nối tiếp như ở bảng 13.17.

Trong bộ lệnh, có các nhóm lệnh sau:

- Lệnh điều khiển cuộn dòng.
- Lệnh điều khiển con trỏ.
- Lệnh điều khiển độ sáng.
- Lệnh điều khiển nhấp nháy.
- Lệnh điều khiển tạo ký tự của người dùng.
- Lệnh điều khiển đưa tin ra.
- Lệnh điều khiển di chuyển con trỏ.
- Lệnh điều khiển khởi phát đồ họa thanh ngang và đứng.
- Lệnh điều khiển tạo thanh đứng.

Một số lệnh còn yêu cầu đưa thêm thông số (xem các chú thích 1, 2, 3, 4, 5).

Bảng 13.17. Bộ mã lệnh ASCII của LCD nối tiếp

ASCII	HEX	Cú pháp	Hành động	ASCII	HEX	Cú pháp	Hành động
"C"	43H	0xFE 0x43	Tự cuộn dòng	"S"	53H	0xFE 0x53	Nhấp nháy con trỏ
"D"	44H	0xFE 0x44	Bỏ cuộn dòng	"T"	54H	0xFE 0x54	Tắt nháy con trỏ
"Q"	51H	0xFE 0x51	Cuộn lên	"N"	4EH	0xFE 0x4E <thông số>	Tạo ký tự của người dùng (2)
"R"	52H	0xFE 0x52	Bỏ cuộn lên	"V"	56H	0xFE 0x56 <số HEX>	Đưa ra chung
"B"	42H	0xFE 0x42 <số phút>	Bật sáng đèn	"W"	57H	0xFE 0x57 <số HEX>	Cấm đưa ra chung
"F"	46H	0xFE 0x46	Tắt sáng đèn	"G"	47H	0xFE 0x47 <số hàng> <số cột>	Tới vị trí (3)
"X"	58H	0xFE 0x58	Xoá hiển thị	"H"	48H	0xFE 0x48	Tới tận cùng trái
"P"	50H	0xFE 0x50 <hex value defining contrast>	Độ tương phản (1)	"h"	68H	0xFE 0x68	Khởi phát đồ họa thanh ngang
"J"	4AH	0xFE 0x4A	Bật con trỏ	"I"	7CH	0xFE 0x7C <thông số>	Tạo đồ họa thanh ngang (4)
"K"	4BH	0xFE 0x4B	Tắt con trỏ	"V"	76H	0xFE 0x76	Khởi phát đồ họa thanh đứng đầy
"L"	4CH	0xFE 0x4C	Dịch trái con trỏ	"s"	73H	0xFE 0x76	Khởi phát đồ họa đứng của thanh đầy
"M"	4DH	0xFE 0x4D	Dịch phải con trỏ	"="	3DH	0xFE 0x76 <thông số>	Tạo đồ họa thanh đứng (5)

Chú thích:

1. Lệnh điều khiển độ tương phản:
<hex value> có giá trị từ 0x00 tới 0xFF (mức tương phản).
2. Lệnh tạo ký tự của người dùng.
< character between 0x00 and 0x07>< 8 byte>.

LCD4021 cần thông số này cho tới 8 byte ký tự của người dùng như hình 13.21.

3. Lệnh di chuyển con trỏ tới vị trí.

Cú pháp 0xFE 0x47 < column number > < row number >

trong đó: <command prefix> 0xFE
 <command> 0x47
 <column number> 0x01 tới 0x28 cho hiển thị 40 cột
 <row number> 0x01 tới 0x02 cho hiển thị 2 dòng

4. Lệnh tạo đồ họa thanh ngang

Cú pháp 0xFE 0x7C <column number> <row number> <direction> <bar length>

<command prefix> 0xFE
 <command> 0x7C
 <column number> 0x01 tới 0x28 cho hiển thị 40 cột
 <row number > 0x01 tới 0x02 cho hiển thị 2 dòng
 <direction> 0 cho trái sang phải, 1 cho phải sang trái.
 <bar length> 0x00 tới 0xC8 (0 tới 200) cho hiển thị 40 cột.

MSB							LSB	
*	*	*	1	2	3	4	5	Data byte 1
*	*	*	6	7	8	9	10	Data byte 2
*	*	*	11	12	13	14	15	Data byte 3
*	*	*	16	17	18	19	20	Data byte 4
*	*	*	21	22	23	24	25	Data byte 5
*	*	*	26	27	28	29	30	Data byte 6
*	*	*	31	32	33	34	35	Data byte 7
*	*	*	36	37	38	39	40	Data byte 8

Layout pixel của các ký tự hiển thị

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25
26	27	28	29	30
31	32	33	34	35
36	37	38	39	40

Dòng con trỏ

Hình 13.21. Đồ thị tạo ký tự của người dùng

5. Lệnh tạo đồ họa thanh thẳng đứng

Cú pháp 0xFE 0x7C <column number> <bar length>

<command prefix> 0xFE
 <command > 0x3D
 <column number> 0x01 tới 0x14 cho hiển thị 20 cột
 <bar length> 0x00 tới 0x16 cho hiển thị hai dòng.

13.5.4. Đặc tính quang

Tùy loại LCD, kích thước màn hình và cỡ chữ khác nhau. Bảng 13.18 cho đặc tính quang của hai loại LCD 4021 và 4041.

Bảng 13.18. Đặc tính quang của hai loại LCD 4021 và 4041

Đặc tính quang học	LCD 4021	LCD 4041
Số ký tự	80 (20 ký tự 4 dòng)	160 (40 ký tự 4 dòng)
Dạng ma trận	5x7 với gạch dưới	5x7 với gạch dưới
Diện tích hiển thị	155,10x16,50mm (XxY)	140, 45x23,16mm (XxY)
Cỡ ký tự	3,2x5,55(XxY), không gạch dưới	2,78x4,89(XxY), không gạch dưới
Cách các chữ	3,70mm	3,53mm
Cách các dòng	5,95mm	6,09mm
Kích thước chấm	0,60x0,65mm(XxY)	0,50x0,55mm(XxY)
Cách các chấm	0,65x0,70mm(XxY)	0,57x0,62mm(XxY)
Thời gian sống	100.000giờ	100.000giờ
Màu phát sáng	xanh vàng	xanh vàng

13.5.5. Chương trình cho LCD nối tiếp

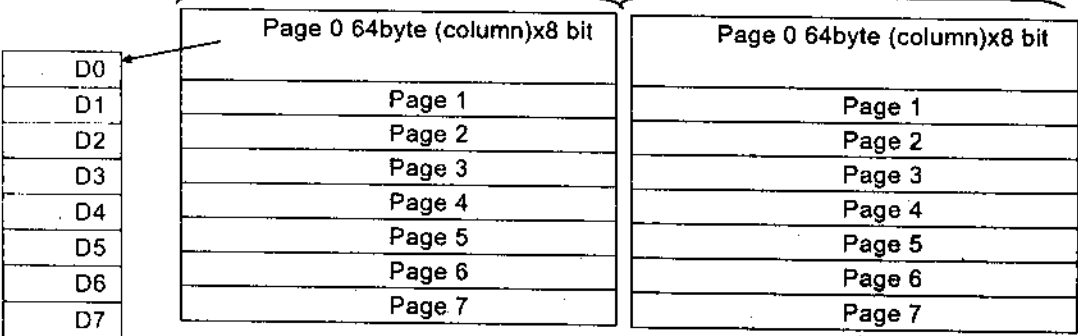
Ví dụ 13.14 là chương trình hiển thị LCD nối tiếp cho vi mạch PIC 16F877X.

<pre>; Ví dụ 13.14: Chương trình hiển thị LCD nối tiếp ở hình 13.20 ; Simple PIC 16F877X Program to drive Serial LCDs. ; Assumes 1MHz CPU clock, 2400 baud rate for the LCD. LIST p = 16F874 #include "P16F874.INC" ; Counter variables for delay cblock 0x20 char, cmd, lc1, lc2; endc ; Vector for normal start up. org 0 goto start org 4 goto inthlr ; Main program starts here: start clrw ; Clear W. movwf PORTA ; Ensure PORTA is zero ; before we enable it. movwf PORTB ; Ensure PORTB is zero ; before we enable it. ; Set up ports: bcf STATUS, RP0 ; Select Bank 0 bsf RCSTA, SPEN ; Enable USART. bsf RCSTA, CREN ; Enable Receive bsf STATUS, RP0 ; Select Bank 1 clrw ; Set W to mask for all outputs. movwf TRISA; Set TRISA register as outputs. movwf TRISB; Set TRISB register as outputs. movlw 0x19; BRG value for 2400baud 1MHz movwf SPBRG ; Write it to register. movlw 0xA4 ; CSRC/TXEN (Internal clock ; 8 bit mode, Asyncoperation, High Speed)</pre>	<pre>call putc movlw 'r' ; Value to transmit. call putc movlw 'd' ; Value to transmit. call putc movlw '' ; Value to transmit. call putc movlw 'L' ; Value to transmit. call putc movlw 'C' ; Value to transmit. call putc movlw 'D' ; Value to transmit. call putc movlw '' ; Value to transmit. call putc movlw 'D' ; Value to transmit. call putc movlw 'e' ; Value to transmit. call putc movlw 'm' ; Value to transmit. call putc movlw 'o' ; Value to transmit. call putc movlw 0xC0 ; Move cursor to row two call wrcmd movlw 0x0D ; Show the cursor call wrcmd loop: call getc movwf char sublw 0d btfsc STATUS, Z goto cls movf char, w sublw 08 btfsc STATUS, Z</pre>
---	---

movwf TXSTA; Write to TX control register. movlw 80 call delay movlw 'M' ; Value to transmit. call putc movlw 'i' ; Value to transmit. call putc movlw 'l' ; Value to transmit. call putc movlw 'f' ; Value to transmit. call putc movlw 'o' ; Value to transmit. Subroutine to sent a command wrcmd movwf cmd ; Store the command movlw 0xFE ; Write the command prefix call putc movf cmd, W ; Write the command code goto putc ; Subroutine to wait and receive a byte ; Returns character in W getc bcf STATUS, RP0 ; Select Bank 0. getc1 btfss PIR1, RCIF ; Skip if RC int flag set goto getc1 ; Try again movf RCREG, W ; Read the character bcf PIR1, RCIF ; Clear the interrupt flag return ; Subroutine to transmit a byte and wait ; W = Character putc bcf STATUS, RP0 ; Select Bank 0.	goto bspace movf char, W call putc ; Send the character straight thru goto loop cls movlw 0x01 ; Send command prefix call wrcmd goto loop bspace movlw 0x10; Sendcommand prefix call wrcmd goto loop hangclrwdt; Clear WDT in case it is enabled. goto hang movwf TXREG ; Write it! bsf STATUS, RP0 ; Select Bank 1 movf TXSTA, W ; Peek transmit status putc1 btfss TXSTA, 1 ; Skip if TXbuffer empty goto putc1 ; Try again bcf STATUS, RP0 ; Select Bank 0. return ; Delay Routine ; W = delay time delay movwf lc2 _sw2 movlw 0xFF movwf lc1 _sw3 nop decfsz lc1, f goto _sw3 decfsz lc2, f goto _sw2 return inthlr retfie END
--	---

13.6. LCD ĐỒ HOẠ

Để hiển thị hình và chữ trên các LCD kích thước lớn, người ta đã chế tạo các LCD đồ hoạ (graph). Cũng có hai loại LCD đồ hoạ là LCD song song và nối tiếp. Nhưng các chân điều khiển của LCD khác với LCD ký tự và thay đổi theo cỡ LCD đồ hoạ. LCD nối tiếp dùng thêm cả các lệnh điều khiển của bảng mã ASCII. LCD đồ hoạ nối tiếp còn mắc với loa và bàn phím để đưa ra tiếng nói và điều khiển cách thức hiển thị.



Hình 13.22. Biểu diễn dữ liệu theo trang trong LCD 128x64

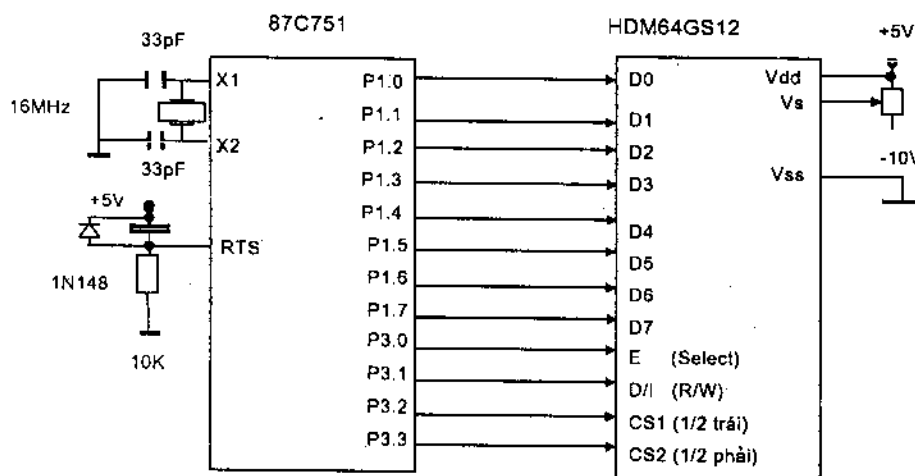
13.6.1. LCD đồ hoạ mắc song song

Ở đây, chúng ta giới thiệu ba loại LCD đồ hoạ song song là 128x64, 240x64 và 320x240 đều của hãng HANTRONIX.

1. LCD 128x64

LCD này hiển thị dữ liệu lên màn hình LCD theo hai nửa màn hình, điều khiển bởi tín hiệu CS1 (cho nửa trái) và CS2 (cho nửa phải). Mỗi nửa màn hình có 8 trang (từ 0 tới 7) như hình 13.22.

Sơ đồ mắc của LCD với vi điều khiển như hình 13.23. Tín hiệu E mắc với P3.0 để chọn LCD, còn tín hiệu D/I mắc với P3.1 để ghi dữ liệu D (P3.1 = 1) hay ghi lệnh I (P3.1 = 0). Các tín hiệu P3.2 CS1 = 1 dùng để chọn nửa trái, còn P3.2 = CS2 = 1 dùng để chọn nửa phải của màn LCD.



Hình 13.23. Sơ đồ mắc LCD HDM64GS12 với vi điều khiển 87C751

Các byte khởi phát của LCD này là:

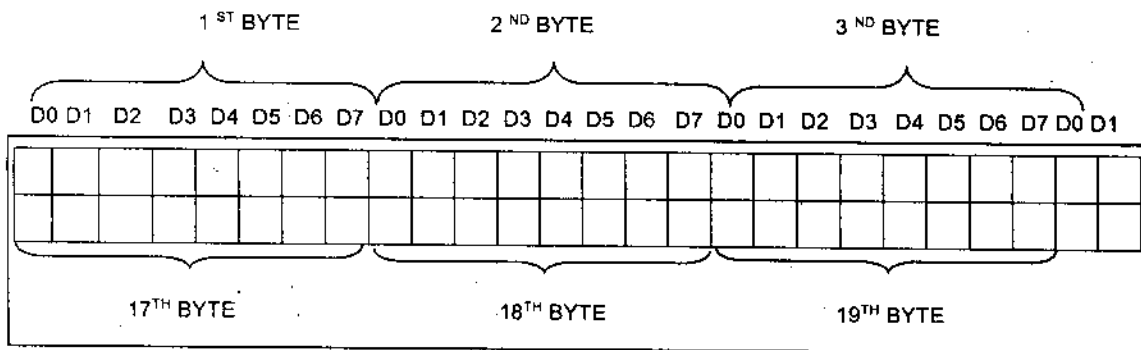
- 0C0h: dành cho dòng RAM hiển thị ở đỉnh của màn hình, đây là dòng 0.
- 040h: đây là địa chỉ cột. Nó chỉ định một trong 64 cột có thể, đây là cột 0.
- 0B8h: đây là địa chỉ trang. Nó chỉ định một trong 8 trang có thể. Đây là trang 0.
- 03fh: bật/tắt hiển thị. 03fh bật hiển thị, 03eh tắt hiển thị.

Ví dụ 13.15 giới thiệu chương trình cho LCD 128x64.

; Ví dụ 13.15: Chương trình cho LCD HD61202		
; tốc độ nhịp là 16MHz	MOV R4, 0B8H	; mẫu hiển thị
; thời gian chu trình là 0,750ms	MOV R5, #08H	; lệnh trang
; chương trình hiển thị logo của Hantronix	LOOP1:	; đếm trang
; trên LCD 128x64	MOV R0, #00H	; xác lập nửa trái
ORG 00H	MOV DPTR, #MSG11	; byte dữ liệu
LJMP START	LCALL BYTE	; gọi ghi dữ liệu
	CLR P3.1	; xác lập lệnh
	INC R4	; tăng trang
ORG 100H	MOV A, R4	
	MOV R1, A	
	LCALL WRITE	
	DJNZ R5, LOOP1	
START:		
MOV P3, #00H		; xoá P3.0
MOV R0, #00H		; xác lập nửa trái
MOV DPTR, #MSG11;		các bytes khởi phát

Sau khi khởi phát, bộ hiển thị sẵn sàng nhận dữ liệu để hiển thị. Dữ liệu có thể dưới dạng dữ liệu đồ họa bản đồ họa theo bit hay dữ liệu văn bản dạng giả ASCII (mã ASCII trừ đi 20h).

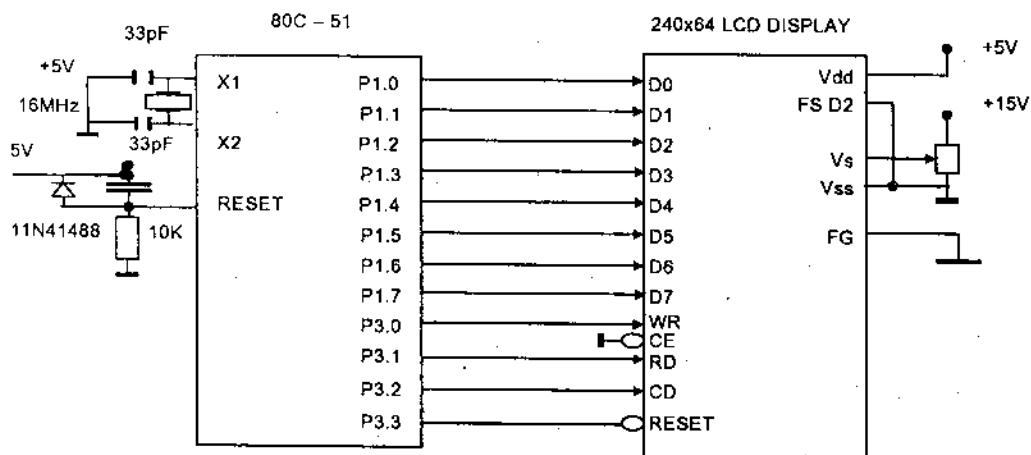
Trong chế độ bản đồ hóa theo bit, mỗi byte dữ liệu biểu diễn 8 pixel lên bộ hiển thị. Hình 13.24 giới thiệu dữ liệu dạng văn bản có thể được viết dưới dạng giả ASCII và sẽ hiển thị như là một ký tự 5x7 điểm. Dữ liệu dạng đồ họa và văn bản được viết tách biệt nhau, được người dùng chỉ định theo vùng, hay theo trang, trong bộ nhớ RAM, ở bên trong mô đun hiển thị. Người dùng có thể chọn một trong ba cách để trộn các trang văn bản và đồ họa lên bộ hiển thị hay các trang văn bản và đồ họa có thể hiển thị riêng rẽ.



Hình 13.24. Bản đồ nhớ của hiển thị 128x128

Vi điều khiển 80C - 51 được nối với vi mạch điều khiển LCD qua cổng song song trong ví dụ này. Nó cũng được nối với với đường dây dữ liệu của vi xử lý và được bản đồ hóa thành vùng nhớ dữ liệu của vi điều khiển như ở hình 13.24.

Chương trình mẫu được viết dạng hợp ngữ cho 8051 và dành cho phần cứng như hình 13.25. Nó dựa trên một hiển thị 240x64 nhưng thích hợp cho bất kỳ hiển thị đồ họa Hantronix nào với bộ điều khiển T6963C.



Hình 13.25. Sơ đồ mắc LCD 240x64 với vi điều khiển 80C - 51

Đầu tiên, chương trình Reset bộ hiển thị. Reset này phải kéo dài tối thiểu 2μS. Sau đó, nó gửi một chuỗi byte lệnh và thông số cho bộ điều khiển LCD để khởi phát nó. T6963C đòi hỏi các thông số. Nếu bất kỳ, gửi ở chế độ dữ liệu và phải đứng trước byte lệnh, nó sẽ gửi ở chế độ lệnh.

Bộ điều khiển này khởi phát với trang đồ họa ở 0000-077h. Đây là 1920 byte, nó sẽ

thích hợp với dữ liệu của toàn màn hình. Không có văn bản được hiển thị với ví dụ này nhưng khối điều khiển được khởi phát cho hành động văn bản để minh hoạ quá trình. Trang văn bản bắt đầu ngay sau trang đồ hoạ ở vị trí 0780h. Tất cả các bộ hiển thị đồ hoạ Hantronix đều có khối điều khiển T6963C ở bên trong với bộ nhớ RAM 8K.

Đường dây FS (Font Select) được xác lập về 0 trong ví dụ này sẽ thay thế ký tự 5x7 pixel trong một vùng 8x8 pixel. Điều này sẽ sinh ra 30 ký tự trên một dòng với 8 dòng đối với hiển thị ($240/8 = 30$). Số ký tự tổng cộng có thể trong chế độ này là 240. Giá trị 1 trên đường dây FS sẽ sinh ra vùng 6x8 pixel cho mỗi ký tự.

Sau khi khởi phát, khối điều khiển được xác lập ở chế độ ghi tự động và bộ hiển thị được lấp đầy với đồ hoạ bản đồ hoá theo bit.

Chương trình: Ví dụ 13.16 giới thiệu chương trình cho T6963C.

<pre> ; Ví dụ 13.16. Chương trình cho T6963C ; tốc độ nhịp của vi điều khiển là 16MHz ; phần mềm mô phỏng cho hiển thị đồ hoạ 240x64 với T6963C. ORG 00H LJMP START ORG 100H START: CLR P3.3 ; khởi phát T6963C ; reset phần cứng NOP NOP SETB P3.3 MOV DPTR, #MSGI1; bytes khởi phát LCALL MSGC ; đọc byte ; bắt đầu hiển thị đồ hoạ MOV DPTR, #MSGI2; chế độ tự động LCALL MSGC MOV DPTR, #MSG1 ; hiển thị đồ hoạ LCALL MSGD SJMP \$; các chương trình con ; MSGC gửi dữ liệu trở bởi DPTR tới mô đun ; đồ hoạ như chuỗi lệnh với hai thông số. MSGC: MOV R0, #2 MSGC2: CLR A MOV C A, @A+DPTR ; đọc byte CJNE A, 0A1H, MSGC3 ; xong? RET MSGC3: MOV R1, A LCALL WRITED ; gửi đi INC DPTR DJNZ R0, MSGC2 CLR A MOV C A, @A+DPTR ; đọc lệnh MOV R1, A LCALL WRITEC ; gửi lệnh SJMP MSGC ; lệnh tiếp ; MSGD gửi dữ liệu tới mô đun đồ hoạ. MSGD: </pre>	<pre> WRITEC: ; gửi byte trong R1 tới một ; mô đun đồ hoạ như một lệnh LCALL STATUS ; hiển thị sẵn sàng? SETB P3.2 ; C/D = 1 WRITEC1: MOV P1, R1 ; lấy dữ liệu CLR P3.0 ; chốt dữ liệu SETB P3.0 RET ; WRITED gửi byte trong R1 tới mô đun đồ ; ; hoạ như dữ liệu. LCALL STATUS ; hiển thị sẵn sàng? CLR P3.0 ; C/D = 0 SJMP WRITEC1 ; ghi dữ liệu ; STATUS kiểm tra xem hiển thị đồ hoạ đã ; sẵn sàng chưa? Nếu không, chờ sẵn sàng. STATUS: SETB P3.2 ; C/D = 1 MOV P1, #OFFH ; P1 là cổng vào MOV R3, #0BH ; che bit trạng thái STAT1: CLR P3.1 ; đọc trạng thái MOV A, P1 SETB P3.1 ANL A, R3 ; trạng thái OK? CLR C SUBB A, R3 JNZ STAT1 RET ; bảng và dữ liệu khởi phát cho 240x64 MSGI1: db 80h, 07h, 40h db 1eh, 00h, 41h db 00h, 00h, 42h db 1eh, 00h, 43h db 00h, 00h, 81h db 00h, 00h, 24h db 00h, 00h, 98h db 0a1h MSGI2: </pre>
---	--

CLR A	db 00h, 00h, 0b0h
MOVC A, @A+DPTR ; đọc byte	db 0a1h
CJNE A, #0A1H, MSGD1 ; xong?	; dữ liệu đồ hoạ theo bản đồ bit, chỉ 8 byte
RET	; đầu tiên chỉ ra ở đây. Đồ hoạ thực gồm 1920 ;
MSGD1:	byte dữ liệu nhị phân.
MOV R1, A	MSG1:
LCALL WRITED ; gửi dữ liệu	db 00H, 00H, 00H, 00H, 00H, 00H, 00H,
INC DPTR	00H db 0A1H
SJMP MSGD	END

3. LCD 320x240

Mở đầu

Do kích thước nhỏ, khối lượng nhẹ, công suất tiêu thụ thấp và dễ thao tác, các mô đun hiển thị đồ hoạ tinh thể lỏng được dùng rộng rãi trong các ứng dụng. Bộ hiển thị LCD 320x240 (1/4 VGA) là rất thông dụng trong một số môi trường tính toán khác nhau. Vì lý do này mà khối điều khiển không có mặt trong mô đun.

Có thể chọn các khối điều khiển bao gồm một vi điều khiển nhúng 8 bit với một khối điều khiển LCD, như là Epson/S-MOS SED1335 hoặc OKI MSM6255/6355. Một số vi điều khiển nhúng, như là National NS486SXF, có khối điều khiển bên trong và sẽ ghép nối trực tiếp với bộ hiển thị.

Đối với khối điều khiển nhúng dựa trên cơ sở PC như Intel 386/486EX, một chip điều khiển VGA, như là các chip và kỹ thuật F65545 hay Vadem VG-660 là tốt nhất. Nếu bộ hiển thị chạy trực tiếp từ một PC, một số bìa VGA là tương thích với hoạt động của bộ hiển thị này. Một số máy tính đơn tấm cũng tương thích với lối ra của bộ hiển thị LCD.

Ghi chú ứng dụng này sẽ liên quan tới một trong các môi trường thông dụng nhất, vi điều khiển nhúng 8 bit. Ví dụ mô tả chi tiết ở đây dựa trên cơ sở một vi điều khiển Phillip 87C751 điều khiển bởi một khối điều khiển LCD Epson/S-MOS SED1335.

Mô tả chức năng

Loại hiển thị Hantronix 320x240 có một giao diện song song công nghệ 4 bit chuẩn. Giao diện này đòi hỏi khối điều khiển liên tục làm tươi bộ hiển thị và duy trì RAM hiển thị video.

Trước khi dùng bộ hiển thị, trước tiên vi điều khiển phải được gửi một chuỗi byte khởi phát cho bộ điều khiển LCD để xác lập các thông số tùy chọn của nó và để mô tả bộ hiển thị cho bộ điều khiển.

Sau khi khởi phát, vi điều khiển ứng dụng có thể gửi dữ liệu văn bản hay đồ hoạ cho bộ điều khiển LCD, nơi nó sẽ được tạo dạng và tích lũy trong RAM hiển thị. Cùng với việc các RAM này cập nhật khối điều khiển LCD, là liên tục đọc dữ liệu từ RAM hiển thị, nối tiếp nó và gửi nó cho bộ hiển thị. Vi điều khiển ứng dụng không có truy cập trực tiếp tới RAM hiển thị và phải gửi tất cả dữ liệu và lệnh tới chip điều khiển LCD.

Sơ đồ

Vi điều khiển 87C751 được nối với chip điều khiển LCD qua cổng song song I/O trong ví dụ này. Nó cũng có thể được nối với bus dữ liệu của bộ xử lý và được bản đồ hoá thành vùng nhớ dữ liệu của bộ xử lý. Chú ý rằng có hai vi mạch nhớ 64C256-10 để ghi dữ liệu và đọc vào SED1335 qua các đường dây VD0-VD7.

Chương trình. Ví dụ 13.17 giới thiệu chương trình phục vụ cho LCD 320x240.

; Ví dụ 13.17: Chương trình phục vụ cho LCD 320x240.	
tốc độ xung nhịp 16MHz ; chu trình 0,750mS ; chương trình hiển thị 1 cây bonsai bitmap và 4 dòng văn bản trên LCD 320x240. ORG 00H LJMP START ORG 100H ; khởi phát 32241 ; trang văn bản 0000h 04afh ; trang đồ hoạ 04b0h 2a2fh START: MOV R1, #40H ; xác lập hệ LCALL COMM32 MOV DPTR, #MSG1 LCALL DATA32 MOV R1, #44H LCALL COMM32 MOV DPTR, #MSG2 LCALL DATA32 MOV R1, #5DH LCALL COMM32 MOV DPTR, #MSG3 LCALL DATA32 MOV R1, #4CH LCALL COMM32 MOV R1, #5AH LCALL COMM32 MOV DPTR, #MSG18 LCALL DATA32 MOV R1, #5BH LCALL COMM32 MOV DPTR, #MSG4 LCALL DATA32 MOV R1, #59H LCALL COMM32 MOV DPTR, #MSG5 MOV R1, #46H LCALL COMM32 MOV DPTR, #MSG10 LCALL DATA32 MOV R1, #42H LCALL COMM32 MOV DPTR, #MSG17 LCALL DATA32 SJMP \$; chương trình con ; COMM32 gửi byte trong R1 tới hiển thị ; 32241 như một lệnh COMM32:	LCALL DATA32 ; clear the text page LCALL CLRTEXT ; display bitmap MOV R1, #46H LCALL COMM32 MOV DPTR, #MSG6 LCALL DATA32 MOV R1, #42H LCALL COMM32 MOV DPTR, #MSG12 LCALL DATA32 ; display text MOV R1, #46H LCALL COMM32 MOV DPTR, #MSG7 LCALL DATA32 MOV R1, #42H LCALL COMM32 MOV DPTR, #MSG14 LCALL DATA32 MOV R1, #46H LCALL COMM32 MOV DPTR, #MSG8 LCALL DATA32 MOV R1, #42H LCALL COMM32 MOV DPTR, #MSG15 LCALL DATA32 MOV R1, #46H LCALL COMM32 MOV DPTR, #MSG9 LCALL DATA32 MOV R1, #42H LCALL COMM32 MOV DPTR, #MSG16 LCALL DATA32 MOV R1, #42H LCALL COMM32 MOV DPTR, #MSG11 LCALL DATA32 MOV R1, #46H LCALL COMM32 MOV DPTR, #MSG6 LCALL DATA32 RET ; TABLE AND DATA ; khởi phát các thông số cho 3224

SETB P3.2 COMM321: MOV A, R1 MOV P1, A CLR P3.0 CLR P3.1 SETB P3.1 SETB P3.0 RET ; WRITE32 gửi byte trong r1 tới bộ hiển ; thì 32241 như là một byte dữ liệu WRITE32: CLR P3.2 SJMP COMM321 ; DATA32 gửi thông báo trở bởi DPTR tới ; bộ hiển thị 32241 DATA32: CLR A MOVC A, @A+DPTR CJNE A, #0A1H, DATA321; RET DATA321: MOV R1, A LCALL WRITE32 INC DPTR SJMP DATA32 ; clear text RAM on 3224 CLRTEXT: MOV R1, #46H LCALL DATA32 LCALL COMM32 MOV DPTR, #MSG13 LCALL DATA32	MSG1: db 30h, 87h, 07h, 27h; xác lập hệ db 39h, 0efh, 28h, 0h, 0a1h MSG2: db 0, 0, 0efh, 0b0h ; cuộn db 04h, 0efh, 0, 0 db 0, 0, 0a1h MSG3: db 04h, 86h, 0a1h ; dạng CSR MSG4: db 01h, 0a1h ; thông số đề MSG5: db 16h, 0a1h ; bật/tắt hiển thị MSG6: db 0b0h, 04h, 0a1h ; xác lập con trỏ ; tới trang đồ họa MSG7: db 31h, 2h, 0a1h ; xác lập con trỏ, ; trang text, dòng thứ nhất MSG8: db 59h, 2, 0a1h ; dòng thứ hai MSG9: db 81h, 2, 0a1h ; dòng thứ ba MSG10: db 0a9h, 2, 0a1h ; dòng thứ tư ; 1200 khoảng trống dành cho xóa trang ; text, bảng sau không kê khai ở đây, trừ 8 ; byte đầu tiên, nhưng gồm 1200byte, tất cả chúng là 20h MSG11: db “ “, 01ah MSG18: db 0, 0a1h ; thông số HSCR ; hình cây bonsai 320x240 ; bảng sau không kê ở đây. Nó gồm 9600 ; byte, choán hết màn hình bitmap của cây ; bonsai. Ta có thể thêm một số byte trước ; byte kết thúc 01ah để thử hoặc bao gồm ; toàn bộ hình ảnh bitmap. MSG12: db 01ah MSG13: db 0, 0, 0a1h ; thông số HSCR MSG14: db ‘ HANTRONNIX’. 01ah MSG15: db 0, 0, 0a1h ; ‘Crystal Clear and’ db 01ah MSG16: db ‘Visibly Superior’, 01ah MSG17: db ‘LCD Modules’, 0a1h END
--	--

13.6.2. LCD đồ họa mắc nối tiếp

Ngoài LCD đồ họa mắc song song, người ta còn dùng LCD đồ họa mắc nối tiếp để tiết kiệm đường dây dẫn nối giữa vi điều khiển và LCD.

Ở đây chúng ta xét LCD ACS 128x64 làm ví dụ (ngang hay hàng có 128 pixel và dọc hay cột có 64 pixel).

1. Đặc điểm

Đặc điểm chung

Thiết bị đầu cuối hiển thị ACS 128x64 được thiết kế để cung cấp một giao diện người dùng RS232 giá rẻ. Một LCD ánh sáng phản xạ 128x64 để nhìn ánh sáng trực tiếp hay ở trong nhà. Ánh sáng phản xạ tự động bật khi nhận các ký tự hay các lỗi vào bàn phím tác động và sẽ tắt khi kết thúc chu kỳ của chương trình. Chúng ta có thể lựa chọn các bàn phím mắc theo ma trận hoặc không với 7 tới 12 lối vào. Có một máy phát âm tần và bộ khuếch đại để phản hồi cho người dùng.

Hành vi

- Có nhớ FLASH để dễ dàng cập nhật phần mềm.
- Chỉ có một nguồn nuôi.
- Các lối vào bàn phím Quét/Matrận và không ma trận.
- Có bộ khuếch đại âm tần 250mW cho máy phát âm tần.
- Có LED để chỉ thị hoạt động của Rx hay Tx.
- Có 30 trang đồ hoạ lập trình được, có thể trải rộng tới 94.
- Xác lập cấu hình không bị mất của người dùng cho tốc độ baud, loại bàn phím (7 phím, ma trận 3 hàng, 4 cột, gửi phím đóng hay mở), số giây vượt quá (timeout) của ánh sáng phản xạ, bật logo hiển thị, bật xác lập hiển thị, lựa chọn thủ tục (Protocol) trao đổi tin là :
 - Các lệnh SOH/ETX cho đồ hoạ.
 - Các subset ANSI để cuộn dữ liệu ký tự.
 - Công suất thấp: – 32mA với LED tắt.
 - 190mA với LED sáng.

2. Chân nối

Chân nối RS-232 DB-9P như ở bảng 13.19a và nối với loa và xoá như ở bảng 13.19b.

Bảng 13.19a. Chân nối RS-232

Chân	Tín hiệu	Chiều ở hiển thị
2	RXD	Vào
3	TXD	RA
4	DTR (tùy chọn)	RA
5	GND	Vào
9	+12V-14VDC	Vào

Bảng 13.19b. Chân nối loa và xoá

Chân	Tín hiệu
1	Gnd
2	Reset-
3	Gnd
4	N/u
5	Speaker+
6	Speaker-

Bảng 19c. Chân nối về phía PC

LCD PIN#	PC DB9 PIN#	PC DB25 PIN#	Nguồn nuôi	Tín hiệu
2	3	2	N/C	RXD
3	2	3	N/C	TXD
4 (tùy chọn)	8 (tùy chọn)	5 (tùy chọn)	N/C	CTS
5	5	7	GND	GND
9	N/C	N/C	+12V-14VDC	+12V-14VDC

CÂU HỎI, BÀI TẬP VÀ THỰC HÀNH

I. Câu hỏi

1. Nguyên tắc của hiển thị bằng đèn LCD. Cách mắc với cổng song song và nối tiếp.
2. So sánh sự hiển thị của LCD với sự hiển thị của LED 7 đoạn và ma trận diot.
3. Hệ lệnh của LCD song song và văn bản.
4. Hệ lệnh của LCD nối tiếp và đồ hoạ.

5. So sánh các chương trình đưa lệnh và dữ liệu ra hiển thị trên LCD.
6. Công dụng của bộ nhớ CG RAM trong LCD.
7. Nguyên tắc và chương trình phát ký tự trên LCD song song và văn bản.

II. Bài tập

1. Hãy viết các chương trình còn lại cho các lệnh của LCD.
2. Hãy viết lại chương trình của ví dụ 12.5 theo cách đọc chuỗi ký tự.
3. Hãy viết chương trình khởi phát, đưa ra dòng chữ "Dữ liệu truyền".
4. Hãy viết chương trình khởi phát, đưa ra một byte dữ liệu.

III. Thực hành

1. Soạn thảo và chạy thử chương trình của các ví dụ 12.1, 12.2, 12.3, 12.4, 12.5 trên PINNACLE.
2. Nạp vào VĐK và cho chạy thử chương trình của ví dụ 12.5 (theo từng ký tự và theo chuỗi ký tự trên LCD).
3. Cho chạy thử các chương trình cho LCD song song và nối tiếp trong môi trường Proteus.
4. Dịch và cho chạy thử các chương trình LCD đồ họa trong môi trường Pinnacle.
5. Tìm trong PROTEUS những bài tập mẫu về LCD, dùng LCD loại gì và phân biệt bài tập nào là LCD song song, bài tập nào là LCD nối tiếp.

CHƯƠNG 14

GHÉP NỐI VỚI BÀN PHÍM

Mục đích của chương:

- Hiểu được cấu tạo và hoạt động của bàn phím.
- Hiểu được hệ lệnh của bàn phím.

14.1. TỔNG QUÁT VỀ BÀN PHÍM

14.1.1. Yêu cầu sử dụng bàn phím trong các hệ vi điều khiển

Trong khi sử dụng hệ vi xử lý hay VDK, người dùng phải đưa lệnh điều khiển hay dữ liệu cho VDK dưới dạng một chuỗi các số 0 và 1 gọi là mã. Do đó, phải có bàn phím để thực hiện nhiệm vụ trên.

Dữ liệu của bàn phím có thể là:

- Các mức logic dạng nhị phân 0, 1 (mức TTL có mức 0 từ 0V – 0,2V, mức 1 từ 2,8V – 5V).
- Mã hay tập hợp các số nhị phân, ví dụ mã BCD (cho các số tự nhiên từ 0 tới 9), mã 16-HEX cho các số từ 0 tới F) hay mã ASCII (cho các ký tự từ A tới Z, các con số hay các ký hiệu).

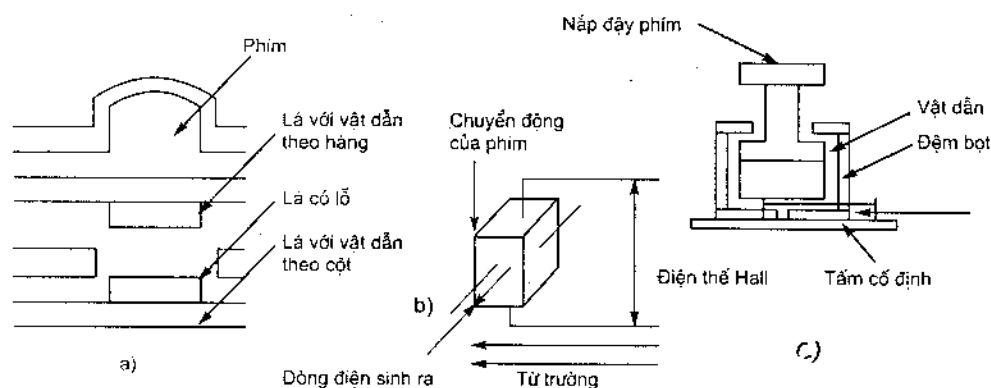
Các phím sẽ tạo mã khi phím được nhấn và VDK sẽ đọc các mã này để biết được lệnh hay dữ liệu của người điều hành.

Tuỳ yêu cầu, ta có cách bố trí và nối các phím khác nhau, do đó chương trình tạo mã cũng khác nhau.

14.1.2. Các loại phím nhấn

Phím nhấn hay chuyển mạch (keyswitch) là thiết bị ở một trong hai trạng thái cân bằng: hoặc mở (trạng thái 0 hay 1) hoặc nhấn (trạng thái 1 hay 0). Khi có ngón tay nhấn hay tác động vào phím, trạng thái của phím được thay đổi từ trạng thái này sang trạng thái khác. Người ta dựa trên tính chất này để tạo mạch điện của phím, tạo nên hai trạng thái logic (0 và 1) về điện cho phím.

Hình 14.1 giới thiệu các loại phím nhấn kiểu điện trở R (cơ khí, kiểu màng), phím nhấn điện dung chạy điện cảm L, phím nhấn hiệu ứng Hall (tạo từ trường khi nhấn phím) hay áp điện (tạo điện thế).



Hình 14.1. Các loại phím: điện trở (a), hiệu ứng Hall (b) và điện dung (c)

Nguyên tắc hoạt động chung của các loại phím nhấn trên là làm thay đổi thông số của mạch điện (điện trở R, điện dung, điện cảm L, thế điện động V từ 0 đến lớn vô cùng) tới giá trị xác định. Mạch điện tạo mã của phím sẽ tạo nên các giá trị logic (0 hay 1) khi có sự thay đổi thông số của mạch điện trên.

Nguyên tắc chung của phím nhấn là:

- Khi không nhấn phím, tín hiệu ra của phím là 0 hay 1.
- Khi có nhấn phím, tín hiệu ra của phím là 1 hay 0.

Phím nhấn kiểu chuyển mạch cơ khí (a) là đơn giản và tin cậy nhưng công kênh. Phím nhấn điện dung C được thông dụng trong các thiết bị điện tử hiện đại nhưng mạch điện tạo tín hiệu logic phức tạp, đắt tiền. Khi đặt ngón tay người dùng lên màn thiết bị, phím được nhấn và có tín hiệu truyền vào hệ vi xử lý hay vi điều khiển. Hiện nay, người ta dùng màn hình tương tác kiểu điện dung, có thể vừa hiển thị được tin ra, vừa đưa được tin vào hệ khi nhấn ngón tay lên màn hình. Tùy vị trí quy ước trước, hệ điện tử có thể nhận biết được các lệnh khác nhau mà người dùng muốn điều khiển hệ.

Trong một bàn phím có nhiều phím nhấn, chúng có thể mắc:

- *Theo kiểu song song*: mỗi phím có một mạch điện tạo mã riêng, độc lập nhau. Các phím này có thể nối trực tiếp với cổng vào/ra của VDK hay qua một thanh ghi hay mạch lập mã thập phân – nhị phân.

- *Theo kiểu ma trận*: các phím bố trí theo kiểu ma trận với các hàng và cột. Có các đường dây hàng và các đường dây cột và mỗi phím được nối với một đường dây hàng và đường dây cột. Các đường dây hàng và cột này có thể nối trực tiếp với lối ra của cổng vào/ra song song của VDK hay qua một mạch tạo mã (encoder).

Cách mắc này tiết kiệm được các bit của cổng vào/ra khi số phím nhấn là lớn (hơn 8). Ví dụ với 16 phím nhấn, chỉ cần 4 đường dây ra và 4 đường dây vào.

Về liên hệ với VDK, các đường dây ra của mỗi phím có thể:

- *Mắc trực tiếp với các lối vào của cổng song song của vi điều khiển*. Cách mắc đơn giản này còn có lợi là vi điều khiển (VDK) dễ nhận biết trực tiếp tín hiệu của những phím nào đưa vào, nhưng có nhược điểm là tốn các bit của các cổng vào/ra khi số phím nhấn là lớn (hơn 8).

- *Mắc gián tiếp với các lối vào của cổng song song của vi điều khiển*: cách mắc này có ưu điểm là đỡ tốn các bit của cổng vào/ra của VDK nhưng phải tạo mã hay nhận biết được phím nhấn thông qua mạch lập mã (phần cứng) hay chương trình (phần mềm) nhận biết phím hay đọc mã của phím nhấn.

14.1.3. Mạch điện tạo mã của phím nhấn

Mỗi phím dù cấu tạo khác nhau (chuyển mạch cơ khí, màng tiếp xúc, cảm ứng điện dung hay điện cảm, điện trở hay hiệu ứng Hall) đều có hai trạng thái:

- Trạng thái nghỉ (không được nhấn).
- Trạng thái hoạt động tức phím được nhấn.

Mạch điện phát hiện và ghi nhớ lại trạng thái phím đã để hở (chưa ấn phím lần nào) hay đã được nhấn phím một lần. Mạch điện trên tạo trạng thái 0 nhị phân khi phím chưa được ấn và trạng thái 1 nhị phân khi phím đã được ấn. Ở trạng thái nghỉ (không nhấn phím), phím có thể ở trạng thái 1 nhị phân (thường đóng) hay trạng thái 0 nhị phân (thường mở).

Mạch điện đơn giản nhất cho chuyển mạch cơ khí SW như hình 14.2, với nguồn nuôi +5V mắc nối tiếp với một điện trở R cỡ k Ω (cho dòng cỡ mA), một chuyển mạch cơ khí SW và nối đất.

Ở trạng thái nghỉ, phím SW chưa nhấn, chuyển mạch SW ở trạng thái mở, tín hiệu ra là 5V (logic 1). Ở trạng thái hoạt động, phím SW được nhấn, chuyển mạch SW ở trạng thái đóng, tín hiệu ra là 0 (logic 0).

Mỗi phím nhấn phải có một đặc trưng riêng để hệ vi điều khiển (hay vi xử lý) nhận biết được là đã ấn phím nào (thứ tự thứ mấy trong bàn phím hay có một mã hiệu riêng nào đó (mã thông dụng cho chữ và ký tự là mã điện tín 5 bit, mã EBCD 7 bit và mã ASCII 8 bit).

Với hệ vi điều khiển đơn giản không cần nhiều phím (dưới 16 phím), người ta dùng các số tự nhiên hệ 16 (hexadecimal) từ 0 tới A, B, C, D, E, F để gán cho các phím.

Các số tự nhiên trên chính là số mã của mỗi phím. Trong hệ số, để biểu diễn 16 trạng thái khác nhau cho 16 phím của bàn phím, người ta cần 4 bit nhị phân (d_0, d_1, d_2, d_3). Đọc tổ hợp của 4 bit nhị phân này, tức đọc số mã tương ứng của mỗi phím, người ta biết được phím nào (số thứ tự bao nhiêu) đã được nhấn.

Ngoài dạng mã nhị phân, vi điều khiển có thể hiểu được mã nhị-thập phân BCD (ứng với các số từ 0 tới 9 cho bàn phím có 10 phím), mã dạng 16 (ứng với các số từ 0 tới 9, A, B, C, D, E, F cho bàn phím có 16 phím), mã EBCD, mã ASCII v.v...

Việc tạo mã từ một tín hiệu của một phím nhấn có thể do:

- *Thiết bị (phần cứng)*: ấn một phím, mạch tạo mã tạo chuỗi tín hiệu xung dạng nhị phân có biên độ, độ rộng không đổi theo quy luật biểu diễn mã nào đó. Mạch tạo mã đơn giản nhất là các phím chuyển mạch nối trực tiếp với cổng vào/ra của VDK (hình 14.3).

Khi đó, các bit của cổng chính là mã của phím, ví dụ P1.0, P1.1, P1.2, v.v... Các số thứ tự của bit (0, 1, 2, v.v...) hay các số tự nhiên 0, 1, 2, v.v... chính là số mã của các phím nhấn 0, 1, 2, v.v...

- *Chương trình (phần mềm)*: nhấn một phím, vi điều khiển nhận một hay các tín hiệu của phím để tạo thành mã của phím. Có thể tạo mã hay nhận biết mã của một phím nhấn bằng cách lập trình. Có hai loại tạo mã bằng lập trình:

- Phát tín hiệu 0 ra từng hàng (hay cột), đọc tín hiệu vào từng cột (hay hàng). Từ dữ liệu phát và dữ liệu đọc, ta nhận biết được mã của phím nào nối với hàng và cột đó.

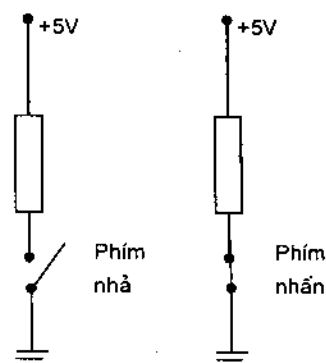
- Đọc bảng dữ liệu đã ghi nhớ sẵn trong bộ nhớ chương trình chỉ đọc (ROM–Read Only Memory) với lối vào nhị phân là địa chỉ của mã, lối ra là của bộ nhớ ROM là mã của phím. Trong trường hợp này, trước đó, ta phải ghi sẵn các mã của các phím nhấn tương ứng vào bộ nhớ ROM bằng chương trình ghi mã và thiết bị ghi chương trình (programmer) như với các vi mạch 8279, 8043, 8048.

Trong trường hợp của bàn phím điều khiển từ xa, mã (được lập ra bằng vi mạch tạo mã) được truyền đi xa (bằng hồng ngoại, siêu âm, sóng điện từ...) tới vi mạch thu để biến đổi thành mã có dạng như khi phát và truyền vào VDK. Đôi khi, để tin cậy và giữ bí mật, khi truyền mã đi xa, người ta còn điều chế hay biến điệu chuỗi tín hiệu theo biên độ, tần số hay pha của tín hiệu và sẽ được giải điều chế ở vi mạch thu rồi biến đổi thành mã như mã phát.

14.1.4. Chương trình phục vụ bàn phím

Sau khi thiết kế và lắp đặt các phím của bàn phím, chúng ta phải lập trình phục vụ bàn phím. Đó là các chương trình:

- *Chương trình tạo mã phím*: chương trình này có nhiệm vụ phát hiện nhấn phím, nhận biết phím nào được nhấn và tạo mã của phím nhấn tương ứng.



Hình 14.2. Mạch điện tạo mã của bàn phím chuyển mạch cơ khí

– *Chương trình phục vụ từng phím của bàn phím*: tùy mục đích sử dụng của từng phím của bàn phím, ta phải có chương trình để:

- + Điều khiển chương trình như các hành động của một thực đơn mong muốn.
- + Đưa dữ liệu vào cho hệ VDK.

Chương trình phục vụ từng phím này phụ thuộc vào mục đích sử dụng từng phím của bàn phím. Một thiết bị có nhiều bàn phím, chương trình này càng dài và chất lượng của chương trình này quyết định chất lượng của thiết bị.

Người ta có thể thay đổi chức năng của từng phím bằng cách thay đổi chương trình phục vụ từng phím đó. Nếu hệ điện tử có nhiều phím chưa sử dụng, ta có thể mở rộng nhiệm vụ của hệ bằng cách viết thêm chương trình phục vụ cho phím cần sử dụng.

14.2. BÀN PHÍM MẮC SONG SONG

Bàn phím gồm nhiều phím, được nối với nhau hoặc không và nối với VDK. Có hai cách mắc các phím với nhau: đó là mắc kiểu song song và mắc kiểu ma trận. Mắc song song là cách mắc đơn giản hơn, cần mạch tạo mã phím (phần cứng) và tốn số đường dây nối. Mắc kiểu ma trận không cần mạch tạo mã mà tạo mã bằng chương trình (phần mềm), tốn ít đường dây nối tới VDK.

14.2.1. Bàn phím mắc song song, nối trực tiếp với VDK

Các phím ấn tạo các tín hiệu 1 khi không ấn phím và 0 khi đã ấn phím để đưa vào cổng của vi điều khiển. Các trạng thái không ấn hay ấn phím đã có thể đưa trực tiếp vào các bit của cổng song song (hình 14.3) hay được ghi vào thanh ghi song song gồm nhiều trigơ RS (hình 14.4) rồi truyền vào VDK.

Để truyền mã của phím vào vi điều khiển, có thể đưa:

– *Song song*: nối mỗi lối ra của thanh ghi song song nối với lối vào của một bit của cổng song song. Vi điều khiển nhận biết phím nào đã nhấn bằng cách đọc bit tương ứng của cổng.

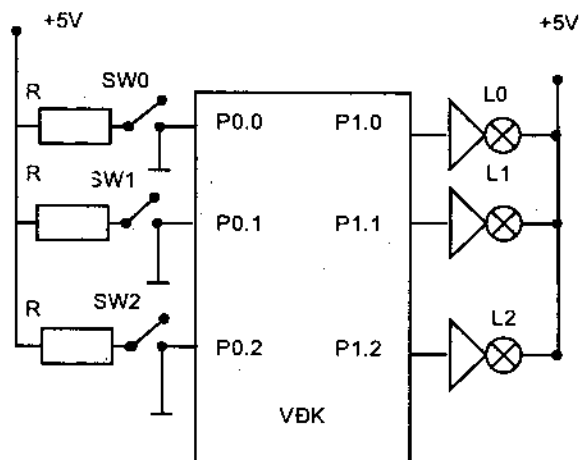
– *Nối tiếp*: thanh ghi nhận mã là thanh ghi dịch song song – nối tiếp và nối lối ra của thanh ghi dịch song song – nối tiếp vào lối vào RXD của cổng nối tiếp của vi điều khiển (chân P3.0 – chân số 10). Sau khi nhận mã ở thanh ghi chứa A (hay Acc), cần phải nhận biết mã của phím bằng các lệnh của chương trình (lệnh dịch trái RLC hay phải RRC qua bit cờ nhớ C và xem C = 0 (có ấn) hay 1 không ấn).

Bàn phím và cách tạo mã trên có ưu điểm là đơn giản và tin cậy cho một số ít phím.

Người ta thường dùng loại bàn phím này để tạo mã điều khiển hệ tự động hoá cần ít lệnh hay ít phím nhấn. Với hệ cần nhiều lệnh điều khiển hay số phím nhấn lớn hơn 8, bàn phím tạo mã kiểu tuyến tính (mã là các số tự nhiên từ 0, 1, 2, 3, 4 v.v...) này cần nhiều phần tử nhớ và cần nhiều lối vào của nhiều cổng song song. Do đó, người ta phải dùng bàn phím mắc kiểu ma trận.

Trong hình 14.3 có thể thay:

– Các phím chuyển mạch cơ khí SW0, SW1, SW2 hai vị trí (đóng, mở bằng tay), có thể thay bằng phím tự nhả do lò xo sau khi nhấn phím một thời gian (cỡ chục miligiây) cho tiện dùng nhưng lại gặp hiện tượng là phím đóng mở nhiều lần do lò xo bị rung. Do đó, ta



Hình 14.3. Bàn phím mắc song song và trực tiếp với VDK

phải khử nhiễu rung cơ khí này bằng đọc phím sau một thời gian trễ (xem chương trình phục vụ phím mắc theo ma trận) hoặc dùng trigơ RS hoặc dùng thanh ghi song song để ghi trạng thái đóng, nhả của phím.

– Các đèn LED đơn bằng các đèn LED 7 đoạn hoặc LED ma trận diot hay LCD. Khi đó, chương trình phục vụ phím phải thêm chương trình con hiển thị LED tương ứng (xem các chương 11, 12, 13).

Ví dụ 14.1 giới thiệu chương trình đọc và hiển thị phím nhấn của bàn phím như ở hình 14.3. Nếu không có một trong ba phím SW0, SW1, SW2 bị nhấn, chương trình vẫn chờ một trong ba phím SW0, SW1 hay SW2 được nhấn. Nếu có một trong ba phím SW0, SW1, SW2 bị nhấn, chương trình sẽ hiển thị một trong các LED0, LED1, LED2 tương ứng mắc với cổng ra P1.

; Ví dụ 14.1: Chương trình đọc, hiển thị phím nhấn SW0, SW1, SW2	; Ví dụ 14.2: Chương trình nhảy tới chương trình con phục vụ phím nhấn SW0
<pre> ORG 00H main: mov a, #00h mov p0, a ; xoá cổng P0 nối phím mov p2, a ; xoá cổng P2 nối LED LJMP MAIN ; trở về MAIN jnb p0.0, s1 ; kiểm tra phím SW1 mov a, #01 ; hiện 01h mov p2, a ; tại P2 CALL DELAY ; gọi trễ để quan sát LJMP KT ; trở về MAIN S1: jnb p0.1, s2 ; kiểm tra phím SW2 mov a, #02 ; hiện 02h mov p2, a ; tại P2 CALL DELAY ; gọi trễ để quan sát LJMP KT ; trở về MAIN S2: jnb p0.2, s3 ; kiểm tra phím SW3 mov a, #04 ; hiện 04h mov p2, a ; tại P2 CALL DELAY ; gọi trễ để quan sát LJMP KT ; kết thúc S3: ljmp MAIN KT: ljmp \$ DELAY: MOV R1, #10 DELAY1: MOV R2, #250 DELAY2: DJNZ R2, DELAY2 DJNZ R1, DELAY1 RET END </pre>	<pre> ; SW1, SW2 MACRO @HIEN <1. dulieu?, 2.cong?> MOV A, #1 ; gửi #1 vào A MOV P12, A ; đưa ra cổng P12 SJMP KT ; nhảy tới KT ENDMAC ORG 00H MAIN: jb p0.0, s1 ; nếu P0.0 bị nhấn, hiện @HIEN 05H. 1 ; 05h ở cổng P1 S1: jb p0.1, s2 ; nếu P0.1 bị nhấn, hiện @HIEN 0AH. 2 ; 0Ah ở cổng P2 CALL DELAY LJMP KT S2: jnb p0.2, s3 ; nếu P0.2 bị nhấn, hiện @HIEN 0F0H. 3 ; 0F0H ở cổng P3 CALL DELAY SJMP KT ; nhảy tới KT S3: LJMP MAIN ; trở về MAIN KT: ljmp \$; dừng chương trình DELAY: MOV R1, #10 DELAY1: MOV R2, #250 DELAY2: DJNZ R2, DELAY2 DJNZ R1, DELAY1 RET END </pre>

Nếu chúng ta gán cho việc nhấn các nút SWi một nhiệm vụ nào đấy, thì khi kiểm tra, thấy nút bị nhấn, chương trình sẽ nhảy tới (hay gọi) chương trình con tương ứng. Khi đó, ta đã truyền lệnh cho VDK tùy theo việc nhấn nút hay đã gọi thực đơn theo nút nhấn.

Ví dụ 14.2 chính là chương trình phục vụ cho một thực đơn dùng ba phím SW0, SW1, SW2, sẽ thực hiện các nhiệm vụ sau:

- Nhấn nút SW0 : hiển thị số 05 ra ra cổng P1.
- Nhấn nút SW1 : hiển thị số 0A (10 thập phân) ra cổng P2.
- Nhấn nút SW2 : hiển thị số F0H ra cổng P3.

Hai chương trình của hai ví dụ 14.1, 14.2 đều có một giải thuật chung là kiểm tra các

phím theo kiểu hồi vòng (nhờ lệnh `LJMP MAIN`) các bit P0.0, P0.1, P0.2, theo thứ tự ưu tiên giảm dần (từ P0.0 tới P0.2). Khi có phím nhấn, sẽ chuyển tới nhiệm vụ hiện LED đơn tương ứng (ví dụ 14.1) hay hiện các số khác nhau ở cổng khác nhau (ví dụ 14.2). Ta có thể kết hợp hai ví dụ 14.1 và 14.2 thành một chương trình duy nhất. Khi đó, nếu nhấn một phím, các LED đơn tương ứng sẽ sáng để chỉ thị phím nào bị nhấn và thực hiện nhiệm vụ tương ứng với phím bị nhấn.

Người ta dùng mạch hình 14.3 và các chương trình của các ví dụ trên để thực hiện việc điều khiển chương trình theo thực đơn từ việc nhấn phím điều khiển, mắc trực tiếp với các bit của cổng P0.

14.2.2. Bàn phím mắc song song qua thanh ghi

Bàn phím mắc song song, nối với VDK qua thanh ghi song song, gồm các trigơ RS như hình 14.4. Các lối ra /Q của các trigơ (có tín hiệu 1 khi nhấn phím SW0 tới SW3) được đưa vào các lối vào P1.0 tới P1.3. Các lối ra Q (có tín hiệu 0 khi nhấn phím SW0 tới SW3) được đưa vào /INT0 qua cổng HOẶC logic để tạo tín hiệu ngắt ngoài khi nhấn phím. Bit P3.0 đưa ra qua cổng HOẶC logic2 để cùng với tín hiệu RESET xoá các trigơ, cho các lối ra /Q về 0. Sau khi nhấn nút RESET, các lối ra Q của trigơ về 0, các lối vào P0.0 tới P0.3 có tín hiệu 0. Khi một trong các nút SW0 tới SW3 được nhấn, một trong các lối ra Q của trigơ có tín hiệu 1 và lối ra /Q có tín hiệu 0 để lối vào ngắt ngoài /INT0 có tín hiệu yêu cầu ngắt ngoài. Tín hiệu P3.0 được đưa ra để xoá trigơ sau khi phím đã được nhấn và xử lý phím nhấn.

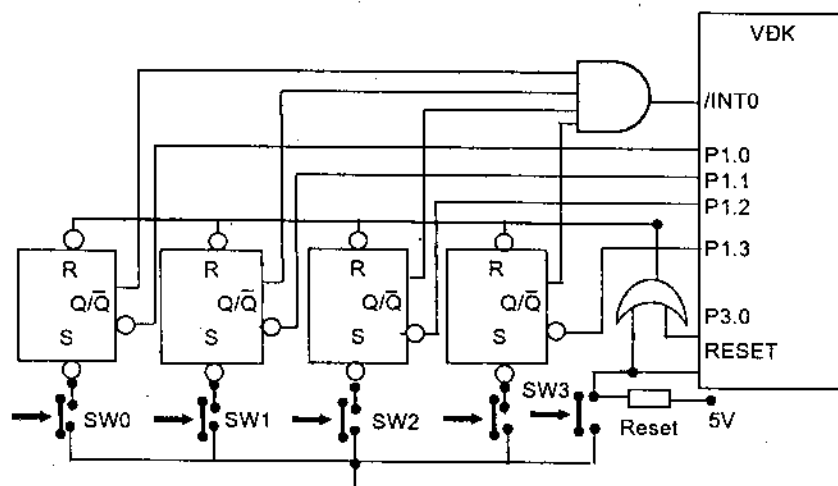
Sơ đồ ở hình 14.4 này có thể hoạt động ở chế độ ngắt chương trình khi nhấn phím. Sơ đồ này có ưu điểm là chống được hiện tượng rung của phím, nhờ được các trạng thái nhấn của nhiều phím và gây ngắt ngoài nếu có phím được nhấn.

Ví dụ 14.3 giới thiệu chương trình phục vụ ngắt bàn phím khi có phím nhấn. Khi cho chương trình chạy và chưa có nhấn phím, chương trình chờ nhấn phím bởi lệnh `SJMP $`. Khi có nhấn phím, tùy phím nhấn, chương trình sẽ nhảy tới các hành động tương ứng với phím bị nhấn.

14.2.3. Bàn phím mắc song song qua vi mạch chuyển mã 10-2

Ở các sơ đồ của hình 14.3 và 14.4, các lối vào từ các phím hay lối ra của thanh ghi đều đưa vào các lối vào VDK theo dạng thập phân. Như vậy, có N phím nhấn, phải cần N bit của cổng vào của VDK.

Nếu số phím nhấn lớn (hơn 8), cần có nhiều cổng vào nối với bàn phím. Để tiết kiệm cổng của VDK, người ta dùng vi mạch biến đổi mã 10-2 để biến đổi mã thập phân thành mã 2. Như vậy, với 4 bit của cổng vào, có thể nối với 2^N hay $2^4 = 16$ phím.



Hình 14.4. Bàn phím mắc song song qua thanh ghi

; Ví dụ 14.3: Chương trình phục vụ ngắt bàn phím có các phím SW0, SW1, SW3, SW4	
MACRO @HIEN <1. dulieu?, 2.cong?> MOV A, #1 ; gửi #1 vào A MOV P12, A ; đưa ra cổng P12 ENDMAC ORG 00H LJMP MAIN ORG 03H LJMP ISR0 ORG 100H MAIN: MOV A, #00H MOV P0, A SETB P3.0 SETB EA SETB EX0 SJMP \$ ISR0: JNB P0.0, S1 ; nếu P0.0 bị nhấn	@HIEN 05H, 1 ; hiện 05H ở cổng P1 CALL DELAY ; gọi trễ SJMP KT ; nhảy tới KT S1: JNB P0.1, S2 ; nếu P0.1 bị nhấn, @HIEN 0AH, 2 ; hiện 0AH ở P2 CALL DELAY ; gọi trễ SJMP KT ; nhảy tới KT S2: JNB P0.2, S3 ; nếu P0.2 bị nhấn, @HIEN 0FH, 3 ; hiện 0FH ở P3 CALL DELAY ; gọi trễ SJMP KT ; nhảy tới KT S3: JNB P0.3, KT ; nếu P0.2 bị nhấn, @HIEN 0FH, 1 ; hiện 0FH ở P1 CALL DELAY ; gọi trễ KT: RETI ; trở về chương trình chính DELAY: MOV R1, #2 DELAY1: MOV R2, #5 DELAY2: DJNZ R2, DELAY2 DJNZ R1, DELAY1 RET END

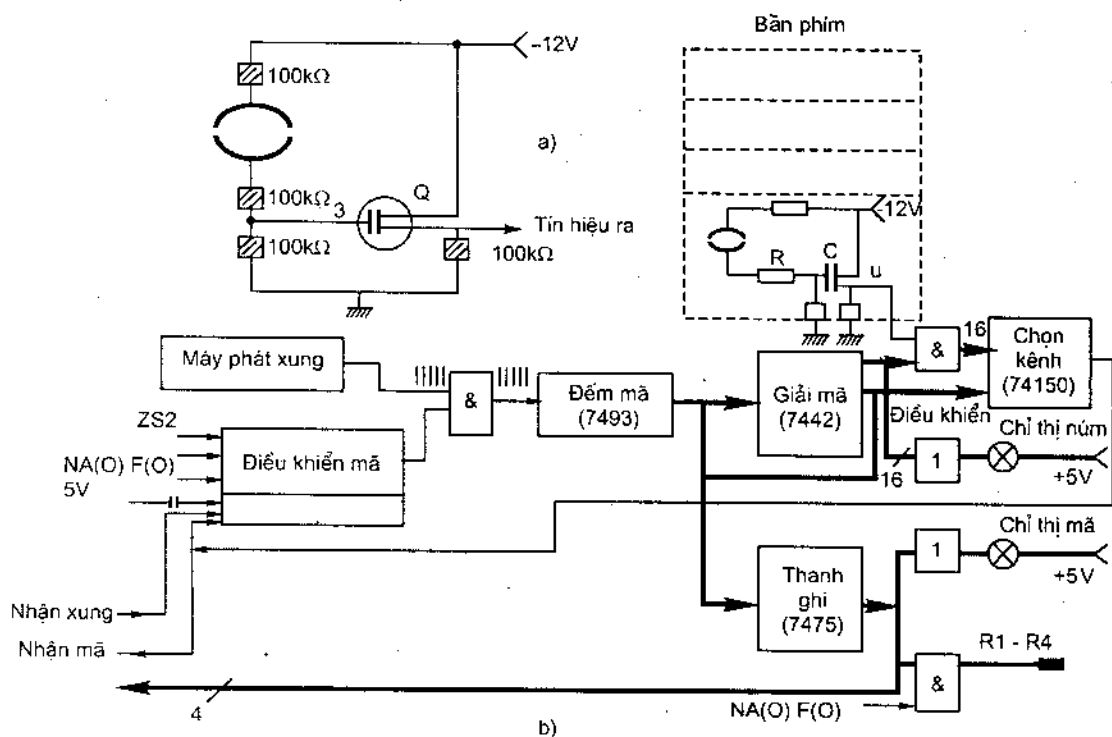
14.2.4. Bàn phím kiểu đếm xung quét

Bàn phím tạo mã dùng thanh ghi (hình 14.4) đơn giản về thiết bị nhưng tốn nhiều bit của cổng vào của VDK. Bàn phím tạo mã kiểu đếm xung quét này dùng bộ đếm để tạo mã nhị phân nên kết hợp được ưu điểm của mạch biến đổi mã 10-2.

Bảng mã tạo mã bằng phương pháp quét mã này thường được sử dụng trong các hệ điều khiển, không đòi hỏi thời gian chạy chương trình tạo mã. Hình 14.5 giới thiệu bảng mã bằng phương pháp đếm quét, có một bộ đếm xung luôn đếm số xung của một máy phát xung tuần hoàn với tần số thích hợp, cỡ kHz. Khi có nhấn phím, bộ đếm xung dừng đếm, số đếm chính là số mã. Nếu bộ đếm hệ nhị phân, số mã ở hệ nhị phân hay hệ 16.

Bàn phím tạo mã trên gồm:

- 16 phím tiếp xúc ấn bằng ngón tay người sử dụng. Khi ấn, điện trở cao của ngón tay tạo nên một tín hiệu điện thế logic 1 (lớn hơn 3V) trên điện trở tải 100kΩ.
 - Máy phát xung quét có điều khiển, dừng phát bởi xung điều khiển.
 - Bộ đếm (vi mạch 7493) để đếm số xung quét.
 - Bộ tạo xung dừng đếm để điều khiển máy phát xung đếm gồm các vi mạch giải mã 2-10 (7442), các vi mạch cổng 2 lối vào (7400), vi mạch lựa chọn hay dồn kênh (74150) và sơ đồ điều khiển máy phát xung.
 - Thanh ghi mã từ bộ đếm (7475) và các cổng hai lối vào (7400) để đưa ra ngoài, đọc mã vào đường dây R1-R4 của ngăn CAMAC (xem Kỹ thuật ghép nối máy vi tính, tập II của cùng tác giả) và chỉ thị mã của phím nhấn dạng nhị phân trên các LED đơn.
- Quá trình tạo mã của phím nhấn diễn ra như sau:
- Khi đóng nguồn điện nuôi 5V, bộ điều khiển mã cho chuỗi xung của máy phát xung vào bộ đếm mã.



Hình 14.5. Phím (a) và sơ đồ tạo mã (b) kiểu đếm xung

– Khi nhấn một phím, tùy theo thời điểm nhấn phím và số thứ tự phím, có một xung vào bộ chọn kênh, đưa ra một xung vào bộ điều khiển để cấm xung từ máy phát vào bộ đếm.

– Bộ đếm dừng tại thời điểm nhấn phím. Thanh ghi ghi lại số đếm hay số thứ tự hoặc số mã của phím nhấn.

– Số mã được chỉ thị trên đèn LED hay vào đường dây đọc dữ liệu R1– R4 của đường dây dãn CAMAC để truyền vào máy vi tính qua khối điều khiển dãn đặt trong dãn CAMAC.

Bảng tạo mã này đơn giản, hoạt động nhanh, tin cậy. Chỉ cần lệnh đọc và truyền dữ liệu vào máy vi tính nối với khối điều khiển dãn CAMAC. Bảng tạo mã này đã được thiết kế, chế tạo (tác giả đã tham gia) và đã sử dụng để điều khiển máy gia tốc Synchrotron tại viện Nghiên cứu Liên hợp hạt nhân Đúp Na (Liên Xô).

Có thể thay đổi bảng phím tạo mã trên để nối với VDK bằng cách nối trực tiếp các lối ra của thanh ghi 7475 với một cửa vào của VDK.

14.3. BÀN PHÍM MẮC KIỂU MA TRẬN

Bàn phím mắc song song (trực tiếp, qua thanh ghi, qua vi mạch biến đổi mã) ở trên có ưu điểm về tạo mã phím ngắn gọn, chương trình phục vụ phím đơn giản nhưng mạch (phần cứng) phức tạp và tốn các bit của cổng vào/ra song song mắc với bàn phím. Người ta đã đề nghị mắc bàn phím theo kiểu ma trận để tiết kiệm đường dây nối, không cần mạch tạo mã, nhưng chương trình phát hiện và nhận biết số hiệu phím phức tạp hơn.

Có thể nối bàn phím nối theo ma trận trực tiếp với cổng vào/ra của VDK hoặc qua vi mạch tạo mã có sẵn.

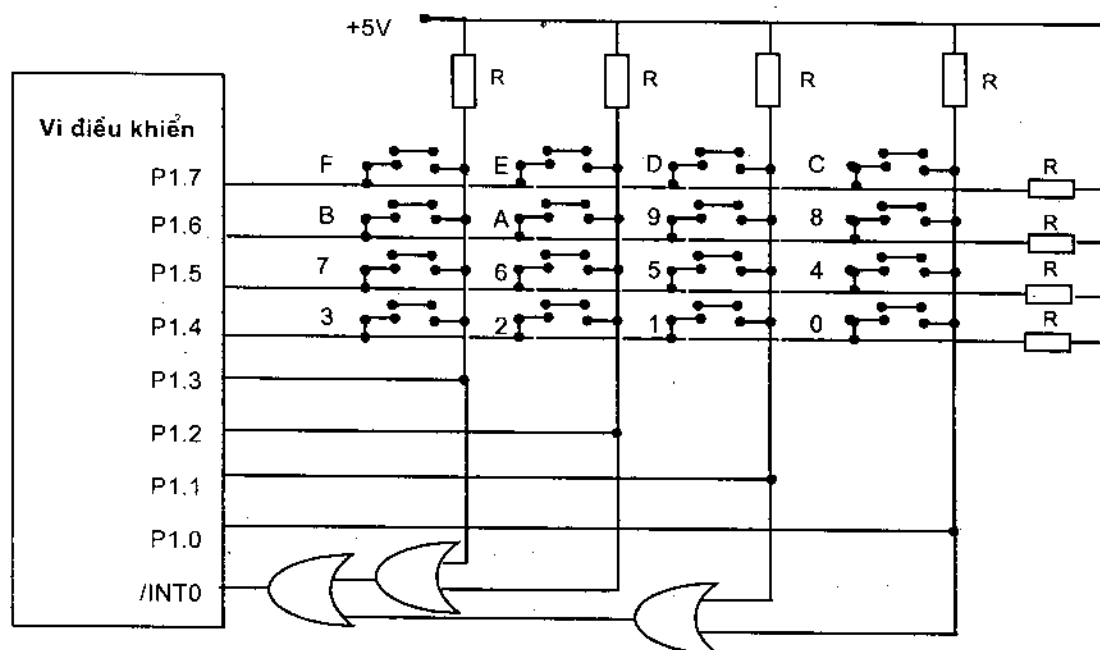
14.3.1. Bàn phím 4x4 nối trực tiếp với VDK

1. Cách nối các phím kiểu ma trận

Với số phím nhấn lớn hơn 8, người ta bố trí và nối phím theo kiểu ma trận theo hàng

và cột. Với 16 phím nhấn, có 4 đường dây hàng và 4 đường dây cột theo ma trận 4x4 (4 hàng, 4 cột). Các đường dây hàng và đường dây cột được nối với nguồn +5V qua điện trở R cỡ kΩ.

Các phím nhấn được nối giữa một đường dây hàng và một đường dây cột. Nói khác đi, các phím được đặt ở các chỗ giao nhau giữa hàng và cột và là cầu nối giữa chúng với nhau khi nhấn phím.



Hình 14.6. Bàn phím kiểu ma trận 4x4

2. Cách tạo mã của phím nhấn

Vì cổng song song của vi điều khiển, ví dụ P1 như hình vẽ, đều có mức logic 1 (do nối với nguồn 5V qua điện trở treo ở bên trong) nên phải đưa tín hiệu logic 0 ra từng cột rồi đọc tín hiệu vào theo từng hàng hoặc ngược lại.

Khi :

– Không nhấn một phím nối với cột đã đưa tín hiệu logic 0: ví dụ P1.0, thì tín hiệu đọc vào ở các hàng, ví dụ, P1.4, P1.5, P1.6, P1.7 đều bằng logic 1.

– Khi có một phím nhấn nối với cột đã đưa tín hiệu logic 0: ví dụ, phím nối với hàng P1.6, điện thế của hàng trở nên bằng logic 0 vì phím nhấn đã nối tắt giữa đường dây cột (có logic 0) và hàng.

Như vậy, nhờ đưa ra tín hiệu logic 0 ở một cột thứ P1.i, khi ấn phím ở hàng thứ P1.j, ta đọc được tín hiệu 0 của phím thứ P1.ij của ma trận bàn phím. Biết chỉ cột thứ i và hàng thứ j, ta suy được mã Aij của phím nối với cột thứ i và hàng thứ j.

3. Cách bố trí phím và cách tính mã của phím

Trên hình 14.6, ta quy ước là :

– Hàng cuối cùng nối với đường dây hàng P1.4 có các phím số 0 nối với P1.0 của cột 0, phím số 1 nối với P1.1 của cột 1, phím số 2 nối với P1.2 của cột 2 và phím số 3 nối với P1.3 của cột 3 kể từ bên phải, theo cột tăng dần (nối với P1.0) từ ngoài vào trong (cột nối với P1.3).

– Tương tự, hàng nối với đường dây hàng P1.5 có các phím số 4, 5, 6 và 7.

– Tương tự, hàng nối với đường dây hàng P1.6 có các phím số 8, 9, A, B (số dạng HEX).

- Tương tự, hàng nối với đường dây hàng P1.7 có các phím số C, D, E, F.
Ta có các mã của 16 phím bố trí theo ma trận 4 cột và 4 hàng của hình 14.6 tăng dần theo hàng, còn cột tăng 4 như ở bảng 14.1.

Bảng 14.1. Bố trí số hiệu của các phím theo hàng

Hàng \ Cột	3 P1.3	2 P1.2	1 P1.1	0 P1.0
3 – P1.7	F	E	D	C
2 – P1.6	B	A	9	8
1 – P1.5	7	6	5	4
0 – P1.4	3	2	1	0

- Từ bảng 14.1, ta suy ra công thức tính mã hay số thứ tự của phím có số cột và số hàng là:
Số hiệu phím = số hiệu cột + 4x số hiệu hàng.
Từ cách nối trên và bảng 14.1, ta suy ra bảng oosh của các phím từ 0 tới FH như ở bảng 14.2.

Bảng 14.2. Bảng mã của các phím từ 0 tới 0FH theo cách mắc ở hình 14.6

Phím số	Hàng				Cột				Nhận xét	Quét
	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0		
00H	1	1	1	0	1	1	1	0	Quét cột 0 với giá trị = 0	Hàng 0 = 0
01H	1	1	0	1	1	1	1	0		Hàng 1 = 0
02H	1	0	1	1	1	1	1	0		Hàng 2 = 0
03H	0	1	1	1	1	1	1	0		Hàng 3 = 0
04H	1	1	1	0	1	1	0	1	Quét cột 1 với giá trị = 0	Hàng 0 = 0
05H	1	1	0	1	1	1	0	1		Hàng 1 = 0
06H	1	0	1	1	1	1	0	1		Hàng 2 = 0
07H	0	1	1	1	1	1	0	1		Hàng 3 = 0
08H	1	1	1	0	1	0	1	1	Quét cột 2 với giá trị = 0	Hàng 0 = 0
09H	1	1	0	1	1	0	1	1		Hàng 1 = 0
0AH	1	0	1	1	1	0	1	1		Hàng 2 = 0
0BH	0	1	1	1	1	0	1	1		Hàng 3 = 0
0CH	1	1	1	0	0	1	1	1	Quét cột 3 với giá trị = 0	Hàng 0 = 0
0DH	1	1	0	1	0	1	1	1		Hàng 1 = 0
0EH	1	0	1	1	0	1	1	1		Hàng 2 = 0
0FH	0	1	1	1	0	1	1	1		Hàng 3 = 0

- Có thể có cách mắc ngược lại là hoán vị hàng cho cột và cột cho hàng. Khi đó, ta có các bảng tăng dần theo cột, hàng tăng 4 (bảng 14.3), nhưng mã của các phím không đổi với công thức tính số hiệu của phím là:

Bảng 14.3. Bố trí số hiệu của các phím theo cột

Hàng \ Cột	3 P1.3	2 P1.2	1 P1.1	0 P1.0
3 – P1.7	F	B	7	3
2 – P1.6	E	A	6	2
1 – P1.5	D	9	5	1
0 – P1.4	C	8	4	0

- Từ bảng 14.3, ta suy ra công thức tính mã hay số thứ tự của phím có số cột và số hàng là:
Số hiệu phím = số hiệu hàng + 4x số hiệu cột.

Trong thực tế, khi sử dụng bàn phím 16 phím mắc theo ma trận 4x4, người ta dùng:

- 10 phím có số hiệu từ 0 tới 9 để gán cho các số tự nhiên từ 0 tới 9.
- Các phím với số hiệu từ 0AH tới 0FH làm các phím điều khiển để điều khiển chương trình (như các phím SW0 tới SW3 ở hình 14.3. Các phím điều khiển này, để tiện dùng, ta gán cho chúng các tên như Enter, Right, Left v.v...
- Để tiện dùng, ta có thể gán cho các phím của các số tự nhiên theo trật tự khác, ví dụ, 0, 1, 2 và 3, 4, 5, 6, 7, 8, 9 kể từ trái sang phải theo thói quen (không theo trật tự từ phải sang trái như bảng 14.1. Khi đó, ta phải có chương trình để đọc bảng dữ liệu hoặc chương trình đổi mã của phím (xem ở dưới).

4. Chương trình tạo hay đọc mã phím nhấn

Với cách mắc các phím theo kiểu ma trận với hàng và cột nối trực tiếp với các bit của cổng vào hay ra của VDK này, ta chỉ cần các điện trở R nối với nguồn nuôi +5V để tạo các mức logic 1 khi đường dây ra có mức logic 1 và các mức logic 0 khi đường dây ra có mức logic 0. Việc tạo mã cho từng phím (bảng 14.2) được thực hiện bởi chương trình (phần mềm).

Khi một phím được nhấn, chương trình cần phát hiện phím được nhấn đó có nhà không? phím đó thuộc hàng nào và cột nào? Từ đó suy được mã của phím đã mắc giữa hàng và cột đó bằng công thức tính số hiệu mã ở trên hoặc đọc bảng mã đã ghi sẵn trong bộ nhớ chương trình (ROM) bằng lệnh đọc bảng của chương trình.

Có hai loại chương trình phục vụ bàn phím:

- *Chương trình trực tiếp*: chương trình chạy, chờ nhấn phím để đọc mã rồi đưa hiển thị. Chương trình này đơn giản, nhưng tốn thời gian chờ nhấn phím của VDK.

- *Chương trình chờ ngắt do bàn phím*: sau khi khởi phát chương trình phục vụ ngắt do bàn phím, chương trình có thể chuyển sang thực hiện nhiệm vụ khác, không phải chờ nhấn phím. Khi có phím được nhấn, có một ngắt xảy ra, có thể do:

+ Luôn kiểm tra cờ ngắt của bàn phím (do xác định bit cờ cho bàn phím) bằng chương trình.

+ Ngắt ngoài /INT0 (thường dùng ngắt này vì có mức ưu tiên cao nhất) hay /INT1 do đã nối mạch xử lý ngắt vào lối vào /INT0 (hình 14.6).

Giải thuật và chương trình đọc mã phím nhấn của vi điều khiển mắc theo hình 14.7.

Chương trình theo trình tự sau:

- Đưa mức logic 0 ra cột (hay hàng) 0 của ma trận tức chân P1.0 của vi điều khiển.

- Chờ ấn một phím bất kỳ trên các hàng của cột.

- Đọc đồng thời các hàng (hay cột) vào nibble cao tức các chân P1.4, P1.5, P1.6, P1.7 của vi điều khiển. Nếu có phím nào của cột được nhấn, giá trị tương ứng của hàng đó có giá trị 0 (giá trị của cột đưa ra) và chương trình chuyển tới xử lý cột (hay hàng) để xem hàng (hay cột) nào được nhấn theo thứ tự tăng dần (ví dụ cột 0 có 0, 1, 2, 3).

- Nếu hàng không có phím nhấn, giá trị đọc vào vẫn là giá trị cũ đưa ra trên cột của cổng và chương trình tới đưa lệnh có giá trị 0 cho cột tiếp theo (do dịch trái giá trị cũ).

- Việc tìm hàng nào trong một tín hiệu cột đưa ra 0 theo lệnh RRC A (dịch phải thanh ghi A qua bit C), nếu có bit = 0 của phím bị nhấn, chương trình chuyển tới tính số hiệu (hay mã) của phím.

- Phần tính mã phím theo công thức $4 \times \text{số hiệu của hàng} + \text{số hiệu của cột}$, tùy theo giá trị của cột tính từ gốc 0 tới 3 (cho các cột P1.0 tới P1.3) và hàng từ 0 tới 3 (cho các hàng từ P1.4 tới P1.7).

Chú ý:

- Cổng P1 nối với các cột (D0-D3) và các hàng (D7-D4), cổng P0 chỉ thị mã phím. Có thể đổi cách mắc cột (D7-D4) và hàng (D0-D3) khi đó số mã sẽ thay đổi (tùy bố trí).

- Chương trình trên có thể mở rộng cho ma trận nhiều cột và nhiều hàng hơn. Khi đó, phải thay giá trị R5, R6 và số lần của lệnh INC B bằng các giá trị cột và hàng tương ứng.

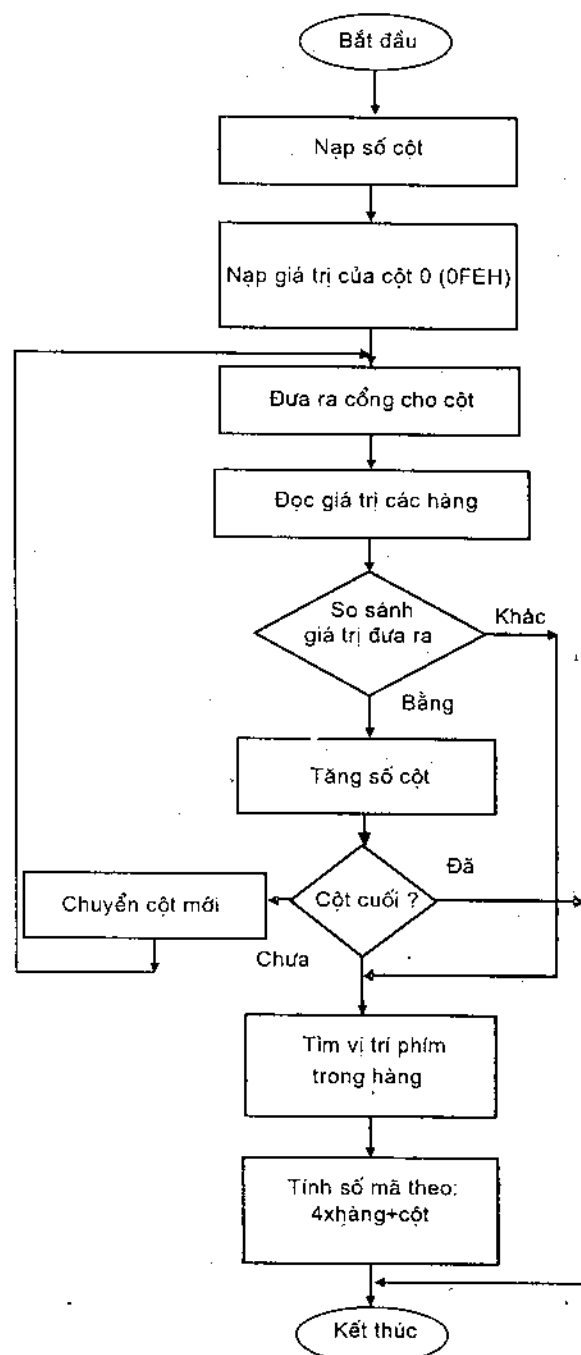
- Chương trình trên đọc, ghi vào ô nhớ địa chỉ 30h và hiển thị số hiệu mã ra cổng P3

dạng thập phân. Muốn hiển thị ra LED 7 đoạn, ma trận diot hay LCD, ta phải chèn thêm chương trình con biến đổi mã sang dạng tương ứng.

– Chương trình con DELAY có giá trị trễ phải lớn hơn 20msec để chống rung và đủ quan sát mã.

.. Có thể bố trí lại thứ tự phím hay thay đổi số mã theo ý thích người sử dụng bàn phím.

– Có thể quy ước lấy một phím có vai trò <Enter> của bàn phím thông dụng để kết thúc quá trình nhập lệnh điều khiển hoặc dữ liệu.



Hình 14.7. Giải thuật chương trình ví dụ 14.4

; Ví dụ 14.4: Chương trình đọc mã phím nhấn tính theo công thức

```

Flag equ 20h      ; cờ bàn phím khi nhấn
ORG 00H
MOV A, #0FEH      ; đưa 0
MOV R6, #4        ; nạp số cột
COT: MOV P0, A     ; đưa 0 ra cổng P1
CALL DELAY
MOV R7, A         ; lưu giá trị cũ
MOV A, P0         ; đọc hàng
ANL A, #0F0H      ; che giá trị cột
CJNE A, #0F0H, MA ; so sánh
MOV A, R7         ; nạp lại
RL A              ; chuyển cột mới
INC B             ; tăng lên 4 cho hàng sau
INC B
INC B
INC B
DJNZ R6, COT      ; lặp lại cột mới
CLR Flag          ; xoá cờ ngắt
SJMP KT          ; nhảy kết thúc
MA: MOV R5, #04H  ; nạp số hàng
MOV R0, #00H
SETB C            ; xác lập C = 1
MOV R7, A         ; lưu giá trị hàng
SWAP A
LAP: RRC A        ; dịch phải qua C, tìm phím nhấn
JNC XONG          ; C = 0 có phím nhấn
INC R0            ; tăng số phím trong cột
DJNZ R5, LAP      ; lặp lại 4 cột
XONG: SETB Flag   ; xác lập cờ bàn phím
MOV A, R0         ; gửi số phím trong cột vào A
ADD A, B          ; tính mã theo công thức
MOV P3, A         ; đưa ra hiển thị ở cổng P3
CALL DELAY
KT: MOV 30H, A    ; lưu mã vào ô nhớ 30H
SJMP $
DELAY: MOV R1, #10
DELAY1: MOV R2, #250
DELAY2: DJNZ R2, DELAY2
        DJNZ R1, DELAY1
        RET
        END
  
```

5. Chương trình đọc mã phím theo bảng dữ liệu trong nhớ ROM

Chương trình đọc mã phím nhấn tính theo công thức của ví dụ 14.4 có nhược điểm là bảng mã cố định với công thức tính mã dạng thập phân cố định như bảng 14.1 hay 14.3 ở trên. Do đó, khi hiển thị ta phải chèn chương trình con biến đổi mã cho đèn hiển thị tương ứng hoặc khi bố trí các phím khác đi, ta phải lập trình để chuyển đổi mã.

Nhược điểm trên được khắc phục bằng chương trình đọc mã phím theo bảng dữ liệu trong bộ nhớ ROM của VDK.

Ví dụ 14.5 giới thiệu chương trình đọc mã phím nhấn theo bảng dữ liệu có trong chương trình hay trong bộ nhớ ROM.

Dữ liệu trên của bảng được bố trí theo các phím của máy tính bấm tay đơn giản. Dạng mã của các phím là mã ASCII (các số và ký hiệu trong dấu ") và hiển thị ra LED đơn nối với cổng P2.

Chương trình trên có những đặc điểm sau:

- Giải thuật tìm phím nhấn cũng tương tự của ví dụ 14.4 đọc bảng mã theo công thức, nhưng ở đây, đọc phím nhấn theo số lần quét ở R0 (cả hàng và cột). Có thể thay R0 bằng DPTR là địa chỉ mã phím.

- Chương trình vẫn dùng cho bàn phím bố trí kiểu khác, chỉ cần thay bảng dữ liệu tương ứng với cách bố trí phím và mã cần dùng. Nếu cần hiển thị ra LED 7 đoạn, diot ma trận hay LCD, ta chỉ cần thay dữ liệu của bảng bằng các mã tương ứng, còn bố trí bảng tùy yêu cầu với mã tương ứng.

- Có thể mở rộng chương trình cho bàn phím có nhiều hàng và nhiều cột hơn.

- Bit cờ của bàn phím sẽ được dùng cho chương trình phục vụ ngắt bàn phím sau này.

Ví dụ 14.6 giới thiệu chương trình đọc mã theo từng hàng trong bảng dữ liệu với giải thuật như hình 14.8. Chương trình này có giải thuật đơn giản hơn chương trình đọc bảng theo toàn bảng (tăng thanh ghi R0 theo cột và theo hàng).

Chương trình ở ví dụ 14.6 được viết lại dưới dạng lệnh gộp như ví dụ 14.7.

Chương trình của ví dụ 14.7 được hợp dịch và liên kết ra các câu lệnh như chương trình của ví dụ 14.6. So với chương trình đọc cả bảng dữ liệu (ví dụ 14.5) thì hai chương trình này đơn giản về giải thuật và chương trình của ví dụ 14.7 với lệnh gộp ngắn gọn hơn nhiều vì không cần lặp lại các lệnh giống nhau.

Có thể viết chương trình 14.6 với các lệnh gọi các chương trình con, nhưng vẫn dài hơn chương trình với lệnh gộp, vì cần các lệnh gán thông số trước khi gọi các chương trình con. Chương trình dịch của chương trình gọi chương trình con này có văn bản ngắn hơn chương trình dịch của lệnh gộp vì không phải chép lại các lệnh lặp lại giống nhau, nhưng lại tốn thời gian thực hiện vì phải tiến hành thủ tục gọi chương trình con và trở về chương trình chính khi kết thúc chương trình con.

: Ví dụ 14.5: Chương trình đọc mã phím nhấn theo bảng dữ liệu trong ROM

ORG 00H LJMP MAIN Flag equ 20h ; cờ bàn phím khi nhấn ORG 100H MAIN: MOV R0, #00H ; số lần quét, thứ tự phím MOV R1, #4 ; số cột MOV R3, #4 ; số hàng MOV R2, #0FFH ; mã cho cột MOV R4, #00H ; lưu hàng MOV R7, #00H ; lưu mã ASCII của phím COT: MOV A, R2 RLC A ; dịch trái cho quét cột MOV R2, A ; lưu giá trị quét cột MOV P1, R2 ; đưa ra cột NOP MOV A, P1 ; đọc vào các hàng MOV R4, A ; lưu giá trị đọc các hàng XRL A, R2 ; so với giá trị đưa ra JNZ HAVEKEY ; nếu khác, có phím nhấn INC R0 ; tăng thứ tự phím cho hàng INC R0 ; tăng thứ tự phím cho hàng INC R0 ; tăng thứ tự phím cho hàng INC R0 ; tăng thứ tự phím cho hàng DJNZ R3, COT ; lặp lại cột mới LJMP KT HAVEKEY: ; tìm mã phím được nhấn SETB Flag ; xác lập cờ khi có phím nhấn sjmp \$; Data tables for returned row bits, keycodes: db '7', '8', '9', '/' DB '4', '5', '6', '*' DB '1', '2', '3', '-' DB 'C', '0', '=', '+' RET END	MOV A, R4 ; nạp giá trị đọc hàng vào A SWAP A ; trao sang 4 bit thấp ANL A, #0FH ; chọn 4 bit thấp MOV R4, A ; lưu 4 bit thấp trong R4 NEXTBIT: RRC A ; dịch phải A qua C JNC HANG ; nhảy nếu có bit nhấn phím INC R0 ; tăng thứ tự phím nhấn JMP NEXTBIT ; trở về, chưa có phím nhấn HANG: ; tìm phím nhấn trong hàng MOV A, R4 ANL A, #0FH ; lọc 4 bit cuối MOV R4, A ; lưu lại trong R4 NEXTBIT1: RRC A ; dịch phải tìm bit nhấn JNC XONG ; tới XONG nếu bit = 0 DJNZ R3, NEXTBIT1; trở về nếu chưa gặp XONG: MOV DPTR, #KEYCODES MOV A, R0 ; lưu thứ tự trong A MOVC A, @A+DPTR ; đọc mã phím JMP KT ; tới kết thúc KT0: CLR A KT: MOV R7, A ; lưu mã phím trong R7 MOV P2, A ; đưa ra hiển thị trên P2 SJMP \$; kết thúc chờ KEYCODES: ; bảng mã xếp theo ma trận phím DB '7', '8', '9', '/' DB '4', '5', '6', '*' DB '1', '2', '3', '-' DB 'C', '0', '=', '+' END
---	---

: Ví dụ 14.6: Chương trình đọc theo từng hàng của bảng mã

ORG 00H ; bắt đầu (RESET) K1: MOV P1, #0FFH ; MOV P1, #0F0H ; nối đất các cột D0-D3 MOV A, P1 ; đọc vào P1 ANL A, #0F0H ; giữ các hàng D4 - D7 CJNE A, #0F0H, K1 ; không nhấn, về K1 K2: ACALL DELAY1 ; chờ trễ cỡ 20ms MOV A, P1 ; đọc vào các hàng CJNE A, #0F0H, OVER ; có nhấn, tới OVER SJMP K2 ; không nhấn phím, về K2 OVER: ACALL DELAY ; chờ nhả phím MOV A, P1 ; đọc vào các hàng ANL A, #0F0H ; bỏ qua các cột CJNE A, #0F0H, OVER1 ; có nhấn, tới OVER1 SJMP K2 ; không nhấn phím, về K2 OVER1: MOV P1, #0FEH ; đưa 0 ra cột D0 MOV A, P1 ; đọc các hàng nối với cột D0 ANL A, #0F0H ; giữ lại các hàng CJNE A, #0F0H, ROW0 ; có nhấn, tới ROW0 MOV P1, #0FDH ; đưa 0 ra cột D1 MOV A, P1 ; đọc các hàng nối cột D1 ANL A, # 0F0H ; giữ các hàng nối cột D1 CJNE A, #0F0H, ROW1 ; có nhấn, tới ROW1 MOV P1, #0BH ; đưa 0 ra cột D2 MOV A, P1 ; đọc các hàng nối cột D2 ANL A, #0F0H ; giữ lại các hàng nối cột D2 CJNE A, #0F0H, ROW2 ; có nhấn, tới ROW2 MOV P1, #0F7 ; đưa 0 ra cột D3 MOV A, P1 ; đọc các hàng nối cột D3	ANL A, #0F0H ; giữ lại các hàng CJNE A, # 0F0H, ROW3 ; có nhấn, tới ROW3 LJMP K2 ; trở về K2 ROW0: MOV DPTR, #COD0 ; địa chỉ hàng CDE0 SJMP FIND ; nhảy tới tìm mã ROW1: MOV DPTR, #COD1 ; địa chỉ hàng CDE1 SJMP FIND ; nhảy tới tìm mã ROW2: MOV DPTR, #COD2 ; hàng CDE2 SJMP FIND ; nhảy tới tìm mã ROW3: MOV DPTR, #COD3 ; hàng CDE3 SJMP FIND ; nhảy tới tìm mã FIND: RRC A ; dịch phải qua bit C JNC MATCH ; nếu C = 0 tới đọc mã INC DPTR SJMP FIND ; trở về tìm mã MATCH: ; đọc hàng trong bảng CLR A MOVC A, @A+DPTR ; đọc bảng MOV P0, A ; đưa hiển thị ra cổng P0 LJMP K1 ; trở về K1 ; Bảng mã theo các hàng phím ORG 200H COD0: DB '0', '1', '2', '3' COD1: DB '4', '5', '6', '7' COD2: DB '8', '9', 'A', 'B' COD3: DB 'C', 'D', 'E', 'F' END
---	---

Chú ý:

– Chương trình trên có cả các lệnh kiểm tra phím luôn bị nhấn (không nhả) và chống phím rung bởi trễ 20ms.

– Có thể đổi bảng mã tương ứng với đèn hiển thị mắc với cổng P0.

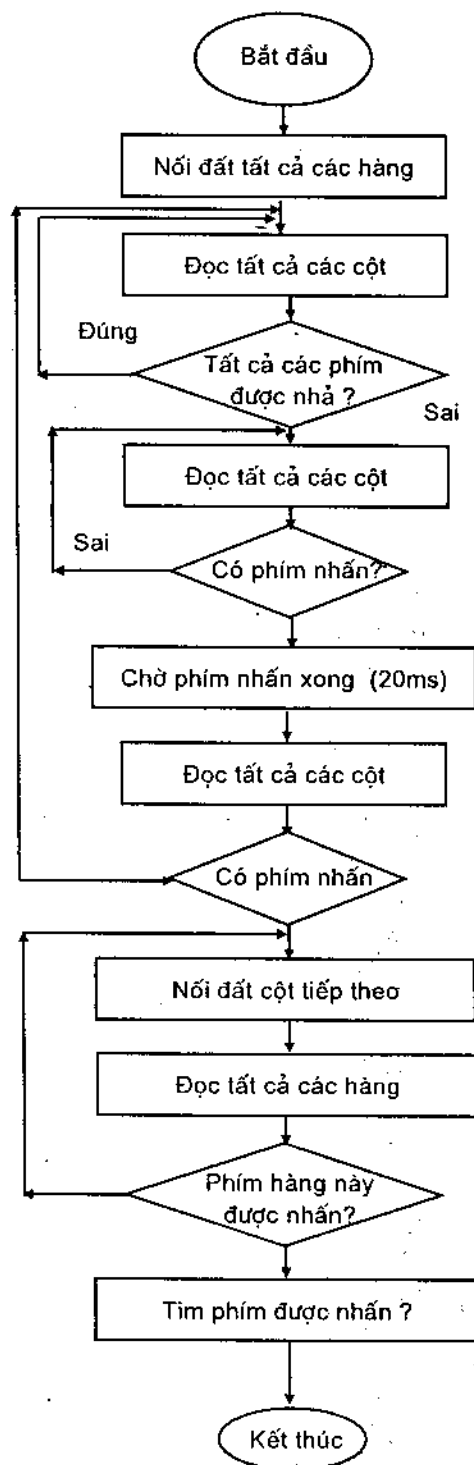
– Có thể mở rộng bảng mã nếu có nhiều cột và nhiều hàng hơn.

Trong chương trình mô phỏng PROTEUS, mục Thí nghiệm về PROTEUS, có hình vẽ và chương trình phục vụ bàn phím. Người đọc có thể tham khảo cách viết khác về cách phát hiện và tạo mã. Chương trình này dài hơn các chương trình đã giới thiệu về bàn phím.

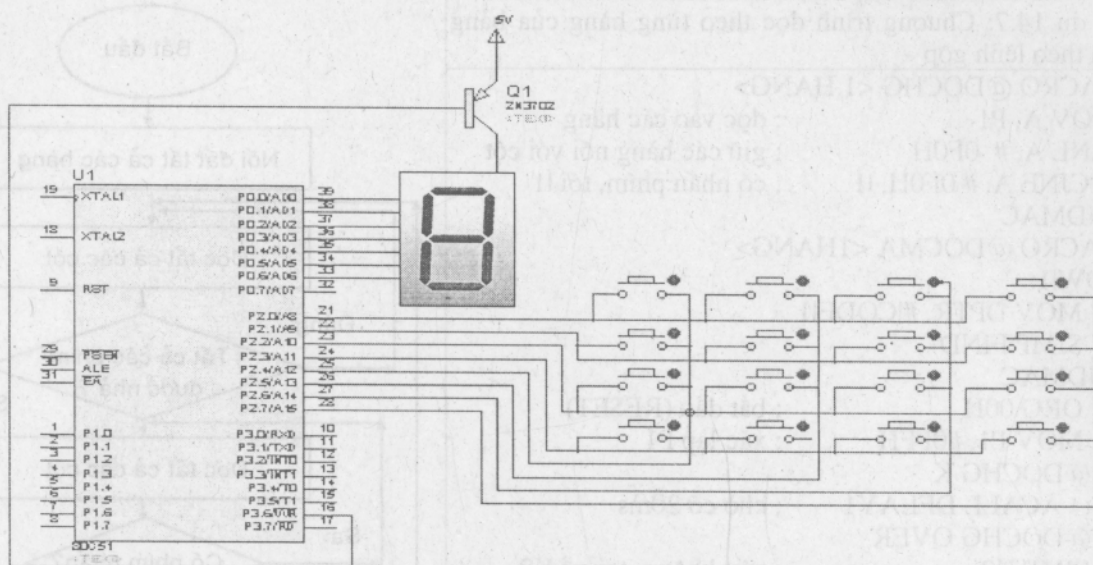
Hình 14.9 giới thiệu sơ đồ bàn phím có chỉ thị chỉ số phím nhấn dùng LED 7 đoạn. Ví dụ 14.8 giới thiệu chương trình phục vụ bàn phím trên với cách tính mã theo công thức và giải mã phím cho LED 7 đoạn theo kiểu đọc bảng dữ liệu. Nếu cần thiết kế và sử dụng bàn phím, chúng ta nên dùng sơ đồ và chương trình này. Có thể dùng ma trận LED hay LCD để hiển thị, ta chỉ cần thay đoạn chương trình hiển thị dùng ma trận LED hay LCD đã xét ở các chương trước.

Ví dụ 14.7: Chương trình đọc theo từng hàng của bảng mã theo lệnh gộp

```
MACRO @DOCHG <1.HANG>
MOV A, P1          ; đọc vào các hàng
ANL A, # 0F0H      ; giữ các hàng nối với cột
CJNE A, # 0F0H, I1 ; có nhấn phím, tới I1
ENDMAC
MACRO @DOCMA <1.HANG>
ROWI1:
MOV DPTR, #CODEI1
SJMP FIND
ENDMAC
ORG 00H           ; bắt đầu (RESET)
K: MOV P1, #0FFH  ; xác lập P1
@DOCHG K
K2: ACALL DELAY1  ; chờ cỡ 20ms
@DOCHG OVER
SJMP K2           ; nếu không, trở về K2
OVER: ACALL DELAY ; chờ trễ nhà phím
@DOCHG OVER1
SJMP K2          ; trở về K2 nếu không nhấn
OVER1:
MOV P1, #1111110B ; đưa cột D0 = 0
@DOCHG, ROW0      ; nhấn phím, tới ROW0
MOV P1, #1111101B ; đưa ra cột D1 = 0
@DOCHG, ROW1      ; nhấn phím, tới ROW1
MOV P1, #11111011B ; đưa ra cột D2 = 0
@DOCHG, ROW2      ; nhấn phím, tới ROW2
MOV P1, #11110111B ; đưa ra cột D3
@DOCHG, ROW3      ; nhấn phím, tới ROW3
LJMP K2           ; trở về K2
@DOCMA 0, CODE0
@DOCMA 1, CODE1
@DOCMA 2, CODE2
@DOCMA 3, CODE3
FIND: RRC A       ; dịch phải qua bit C
JNC MATCH        ; nếu C = 0 tới đọc mã
INC DPTR
SJMP FIND        ; trở về tìm mã
MATCH:           ; đọc hàng trong bảng
CLR A
MOVC A, @A+DPTR
MOV P0, A
LJMP K
; bảng mã theo các hàng phím
ORG 200H
CODE0: DB '0', '1', '2', '3'
CODE1: DB '4', '5', '6', '7'
CODE2: DB '8', '9', 'A', 'B'
CODE3: DB 'C', 'D', 'E', 'F'
END
```



Hình 14.8. Giải thuật đọc mã theo bảng



Hình 14.9. Sơ đồ bàn phím 4x4 có hiển thị bằng LED 7 đoạn

; Ví dụ 14.8: Chương trình phục vụ bàn phím có hiển thị LED 7 đoạn.

```

ORG 00H
LJMP START
ORH 30H
START: MOV DPTR, #BGMA
MAIN: CALL SCANKEY
MOV A, R7
MOVC A, @A+DPTR
MOV P0, A
JMP MAIN
; *****
; KEY VALUE = (R7)
; *****
SCANKEY: MOV R2, #0FFH
CLR C
MOV R3, #4
NEWCOLUMN: MOV A, R2
RLC A
ORL A, #0F0H
MOV R2, A
MOV P2, A
MOV A, P2
MOV R6, A
XRL A, R2
JNZ HAVEKEY
DJNZ R3, NEWCOLUMN
JMP SCANKEY
HAVEKEY: CALL DECODE
RET
; *****
; INPUT: R6
; OUTPUT: R7
; *****
DECODE: MOV A, R6
FINDROWPOSITION:
SWAP A
ANL A, #0FH
MOV R7, #0H
NEXTBIT: RRC A
JNC FINDCOLUMPOSITION
INC R7
JMP NEXTBIT
FINDCOLUMPOSITION:
MOV A, R6
ANL A, #0FH
MOV R5, #00H
NEXTBIT1: RRC A
JNC RETURN
INC R5
JMP NEXTBIT1
RETURN: MOV A, R5
RL A
RL A
ADD A, R7
MOV R7, A
RET
ORG 300H
BGMA:
DB 0C0H, 0F9H, 0A4H, 0B0H; 0-3
DB 099H, 092H, 082H, 0F8H; 4-7
DB 080H, 090H, 088H, 083H; 8-0BH
DB 0A7H, 0A1H, 086H, 08EH; 0CH-0FH
END

```

14.3.2. Bàn phím với vi mạch tạo mã 74922 và 74923

Bàn phím nối trực tiếp với vi điều khiển có ưu điểm là đơn giản về phần cứng, thậm chí không cần các điện trở R nối với nguồn +5V (treo các cổng vào/ra) vì đã có mạch ở bên trong VĐK. Nhưng các mạch bàn phím trên cần chương trình tạo mã (các ví dụ 14.4, 14.5, 14.6, 14.7) nên mất nhiều thời gian của VĐK. Người ta đã khắc phục nhược điểm trên bằng các vi mạch tạo mã khi nhấn phím.

Các vi mạch này có đặc điểm chung là:

- Các phím được mắc theo ma trận (4x4, 4x5, 8x8, và lớn hơn).
- Có bộ đếm quét (phần cứng) tương tự cách tạo mã như ở mục 14.3.
- Khi có phím nhấn, bộ đếm mã dừng đếm, và nội dung đếm chính là mã của phím nhấn.
- Đưa tín hiệu yêu cầu đọc mã để yêu cầu VĐK đọc mã của phím.

Hãng Intel đã chế tạo các loại vi mạch tạo mã sau:

- Vi mạch mật độ thấp 74922 (16 phím) và 74923 (20 phím) cho các hệ VĐK.
- Vi mạch mật độ trung bình có cả hiển thị 8279 với ma trận 8x8 cho các hệ vi xử lý.
- Vi mạch mật độ lớn 8242, 8248 cho bàn phím của máy vi tính PC.

Hai loại vi mạch này có sẵn VĐK bên trong để thực hiện chương trình điều khiển tạo mã.

1. Cấu tạo vi mạch

Vi mạch có sơ đồ chân như hình 14.10a và đường dây như hình 14.10b.

Vi mạch 74922 mắc với bàn phím có 16 phím theo ma trận 4x4, còn vi mạch 74923 mắc với bàn phím có 20 phím theo ma trận 4x5.

Bàn phím được mắc với các đường dây hàng Y1-Y5 và đường dây cột X1-X5.

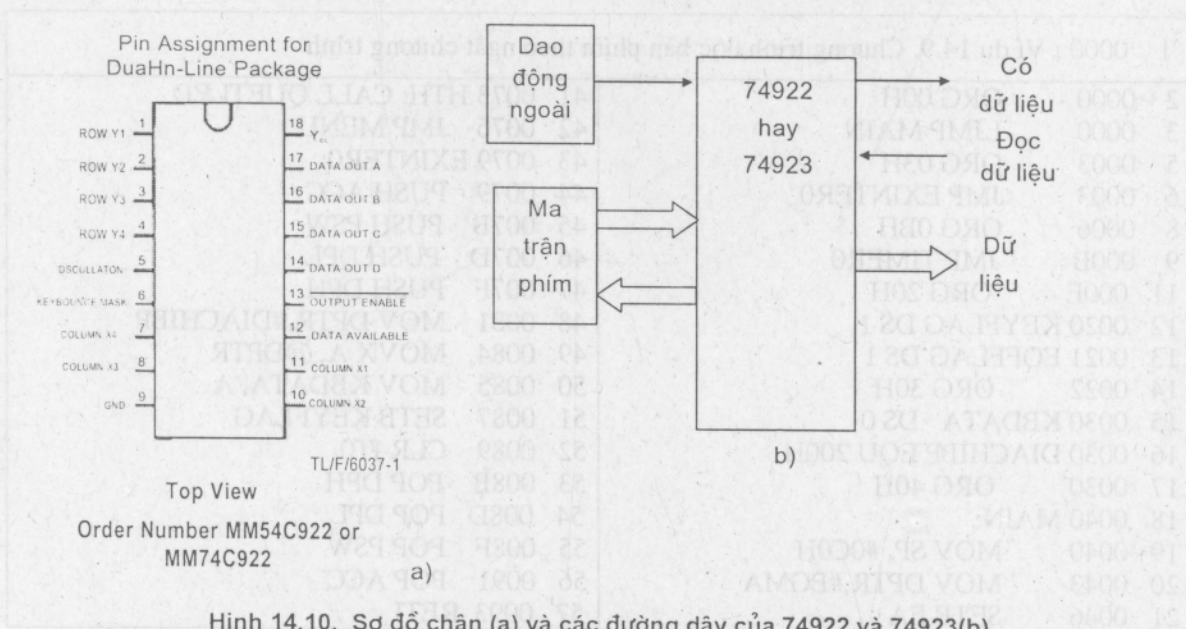
Lối vào dao động ngoài (dùng máy phát ngoài có tần số mong muốn) có thể mắc xuống đất để dùng mạch dao động trong (có tần số cố định).

Tụ C của vi mạch được dùng để chống rung của phím khi nhấn hay nhả phím.

Lối ra Data available (dữ liệu có giá trị) dùng báo mạch ngoài là có phím nhấn và có dữ liệu).

Lối vào Tristate control (điều khiển ba trạng thái) hay /Output enable dùng đọc dữ liệu ra.

Các lối ra (Output) A, B, C, D và E đưa 4 hay 5 bit dữ liệu ra dạng nhị phân.



Hình 14.10. Sơ đồ chân (a) và các đường dây của 74922 và 74923(b)

Khi có xung nhịp ngoài của hệ thống thì:

- Ghi vào trigơ D khi có xung nhịp ngoài để đưa tín hiệu yêu cầu đọc dữ liệu theo kiểu ngắt chương trình. Trigơ ghi yêu cầu ngắt này được xoá khi có xung điều khiển đọc dữ liệu hay:

- Tạo tín hiệu điều khiển đưa dữ liệu ra khi có xung nhịp ngoài và đưa yêu cầu đọc dữ liệu.

Khi không có xung nhịp ngoài của hệ thống thì:

- Đọc ngay dữ liệu ra khi có Data available.
- Đọc dữ liệu ra khi có bit D4 báo bàn phím mở rộng 32 phím để Yêu cầu và Data available.

2. Chương trình đọc mã phím

Ở các ví dụ trên (ví dụ 14.1 tới ví dụ 14.8) chương trình phục vụ bàn phím theo kiểu VDK luôn chạy chương trình phục vụ bàn phím, không chạy chương trình nào khác. Như vậy, VDK luôn chờ nhấn phím và không tiến hành một hành động nào khác, như vậy, VDK được dùng kém hiệu suất.

Trong hệ VDK, VDK còn chạy chương trình chính khác, do đó, chương trình phục vụ bàn phím là một chương trình phục vụ ngắt ngoài, thường cho ngắt ngoài /INT0 với mức ưu tiên cao nhất. Do đó, về phương diện mạch, phải nối tín hiệu ngắt bàn phím hay Data available ở các sơ đồ trên vào lối vào P3.2 dành cho /INT0 (chân số 12 của 8051). Nếu không nối mạch như trên, có thể hội vòng ngắt bằng cách kiểm tra tín hiệu trên nối với bit lối vào bất kỳ.

Chương trình phục vụ bàn phím theo ngắt chương trình phải có hai phần:

- *Khởi phát cho phép ngắt chương trình:* ghi các bit cho phép ngắt toàn bộ EA và cho phép ngắt ngoài /INT0.

- Chương trình đọc dữ liệu hay mã của phím nhấn và xác lập cờ bàn phím Keyflag.

Ví dụ 14.9 giới thiệu chương trình phục vụ bàn phím mắc với vi mạch 74922 của đề tài “ Nghiên cứu ứng dụng linh kiện lập trình, linh kiện chất lượng cao vào chế tạo một vài thiết bị hạt nhân điển hình” của Viện Khoa học và Kỹ thuật Hạt nhân Hà Nội với giải thuật như hình 14.11.

1 0000 ; Ví dụ 14.9. Chương trình đọc bàn phím theo ngắt chương trình		
2	0000	ORG 00H
3	0000	LJMP MAIN
5	0003	ORG 03H
6	0003	JMP EXINTER0
8	0006	ORG 0BH
9	000B	JMP TIMERO
11	000E	ORG 20H
12	0020	KEYFLAG DS 1
13	0021	EOFFLAG DS 1
14	0022	ORG 30H
15	0030	KBDATA DS 0
16	0030	DIACHIBP EQU 200H
17	0030	ORG 40H
18	0040	MAIN:
19	0040	MOV SP, #0C0H
20	0043	MOV DPTR, #BGMA
21	0046	SETB EA
41	0073	HTH: CALL QUETLED
42	0076	JMP MENU
43	0079	EXINTER0:
44	0079	PUSH ACC
45	007B	PUSH PSW
46	007D	PUSH DPL
47	007F	PUSH DPH
48	0081	MOV DPTR, #DIACHIBP
49	0084	MOVX A, @DPTR
50	0085	MOV KBDATA, A
51	0087	SETB KEYFLAG
52	0089	CLR ITO
53	008B	POP DPH
54	008D	POP DPL
55	008F	POP PSW
56	0091	POP ACC
57	0093	RET

22 0048	SETB EX0	58 0094	TIMER0:
23 004A	SETB ET0	59 0094	PUSH ACC
24 004C	MOV TMOD, #01H	60 0096	PUSH PSW
25 004F	MOV TH0, #HIGH(-10000)	61 0098	MOV TH0, #HIGH(-10000)
26 0052	MOV TL0, #LOW(-10000)	62 009B	MOV TL0, #LOW(-10000)
27 0055	SETB TR0	63 009E	LCALL QUETLED
28 0057	MENU: JNB KEYFLAG, HTH	64 00A1	POP PSW
29 005A	CLR KEYFLAG	65 00A3	POP ACC
30 005C	MOV A, KBDATA	66 00A5	RET
31 005E	XRL A, #0AH	67 00A6	QUETLED:
32 0060	JZ HTH	68 00A6	MOV A, KBDATA
33 0062	MOV A, KBDATA	69 00A8	MOVC A, @A+DPTR
34 0064	XRL A, #0BH	70 00A9	MOV P2, A
35 0066	JZ HTH	71 00AB	RET
36 0068	MOV A, KBDATA	72 00AC	KT: SJMP \$
37 006A	XRL A, #0FH	73 00AE	BGMA:
38 006C	JZ KT	74 00AE	DB 0C0H, 0F9H, 0A4H, 0B0H
39 006E	CLR KEYFLAG	75 00B2	DB 099H, 092H, 082H, 0D8H
40 0070	JMP MENU	76 00B6	DB 080H, 090H, 88H, 83H
		77 00BA	DB 0C6H, 0A1H, 86H, 08EH
		78 00BE	END ; HẾT

Chương trình trên có hai ngắt là ngắt ngoài EXINT0 khi có nhấn phím và ngắt thời gian sau 10 000microgiây để hiển thị phím nhấn theo ngắt thời gian 10000microgiây. Chương trình có một thực đơn sau:

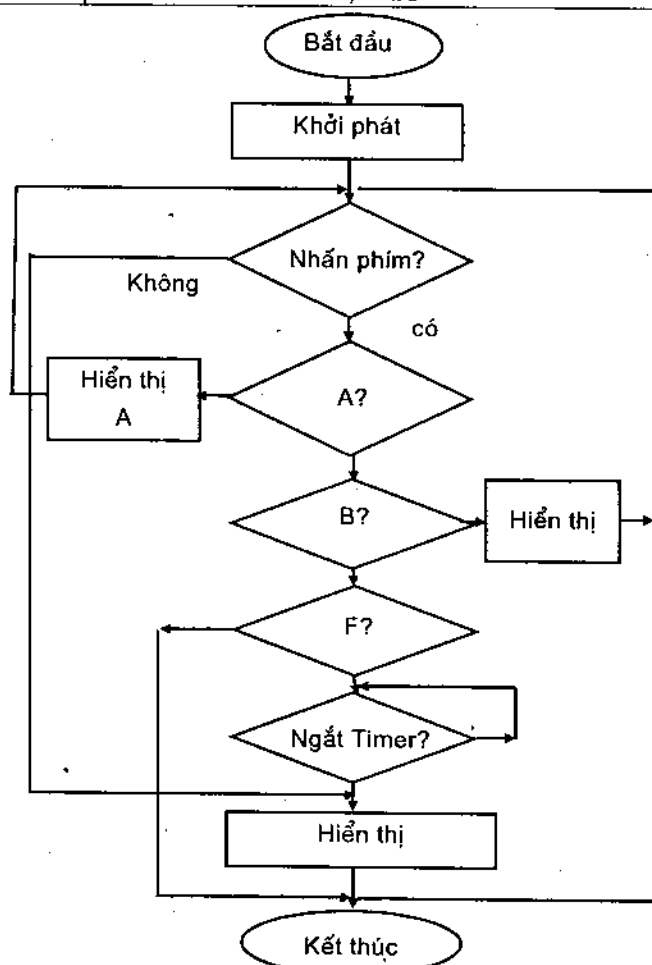
- *Ấn phím ký hiệu A*: sẽ hiển thị mã 7 đoạn của 0AH khi có ngắt thời gian sau 10 000 microgiây ở đèn mắc vào cổng P2.

- *Ấn phím ký hiệu B*: sẽ hiển thị mã 7 đoạn của 0BH khi có ngắt thời gian sau 10 000 microgiây ở đèn mắc vào cổng P2.

- *Ấn phím ký hiệu F*: sẽ thoát khỏi thực đơn.

- *Ấn phím khác với các phím trên*: sẽ hiển thị trên LED 7 đoạn khi có ngắt thời gian sau 10 000 microgiây.

Chú ý: Chương trình trên có thể bỏ ngắt ngoài cho bàn phím nhưng thêm vào chương trình con quét bàn phím trước chương trình con QUETLED để quét bàn phím theo ngắt thời gian.



Hình 14.11. Giải thuật của ví dụ 14.9

3. Chương trình biến đổi mã phím

Ở trên, nếu ta mắc các phím theo hàng (hay cột) theo thứ tự 0, 1, 2, 3, v.v... như hình 14.12a. Nếu trong thực tế, để tiện dùng, ta bố trí các phím khác đi theo trật tự mong muốn, ví dụ như hình 14.12b. Khi đó, ta phải có chương trình biến đổi mã phím như ví dụ 14.10.

0	1	2	3
4	5	6	7
8	9	A	X

a)

1	4	7	A
2	5	8	0
3	6	9	X

b)

Hình 14.12. Cách bố trí bàn phím cũ (a) và mới (b)

; Ví dụ 14.10: Chương trình biến đổi mã phím	
ORG 00H LJMP 40H MACRO @KB0 <1.PHIM, 2.DATA > MOV A, KB_DATA XRL A, #1 JZ DATA2 ENDMAC MACRO @KB1 <1..DATA, 2.PHIM > DATA11: MOV KB_DATA, #2 JMP ENDCV ENDMAC KB_DATA EQU 30H ORG 40H @KB0 0, 1 @KB0 1, 4 @KB0 2, 7 @KB0 3, A @KB0 4, 2	@KB0 5, 5 @KB0 6, 8 @KB0 7, 0 @KB0 8, 3 @KB0 9, 6 @KB0 0AH, 9 ENDCV: SJMP \$ @KB1 0, 0 @KB1 1, 1 @KB1 2, 2 @KB1 3, 3 @KB1 4, 4 @KB1 5, 5 @KB1 6, 6 @KB1 7, 7 @KB1 8, 8 @KB1 9, 9 @KB1 A, 0AH END

Ví dụ trên dùng hai lệnh gộp:

- @KB0 <1.PHÍM, 2.DATA > để đọc mã phím tại địa chỉ cho mã phím KB_DATA, so sánh để nhảy tới nhãn gán giá trị của mã mới.
- @KB1 <1.DATA, 2.PHIM > để gán giá trị mới cho phím.

Nếu hợp dịch chương trình và xem tệp .LST, ta thấy được toàn văn bản của chương trình (sau khi đã biến đổi các lệnh gộp theo hàng ngang thành hàng dọc).

14.4. BÀN PHÍM CHO HỆ VI XỬ LÝ VÀ MÁY VI TÍNH

Hệ vi xử lý và máy vi tính yêu cầu bàn phím có nhiều phím (32, 64 và 101). Ngoài các phím cho các ký tự chữ thường và chữ hoa (chữ A tới Z), các chữ số tự nhiên (0 tới 9) còn có

thêm các phím điều khiển. Để điều khiển bàn phím, người ta đã chế tạo các vi điều khiển để điều khiển bàn phím.

Vi mạch AY5-2376 của hãng General Instrument thực hiện nhiệm vụ tạo mã và truyền nối tiếp vào vi xử lý. Nó chính là một vi điều khiển chuyên dùng, hoạt động như một vi xử lý vào-ra. Sau khi tạo mã, nó đưa xung STB để yêu cầu vi xử lý trung tâm (chính) biết để yêu cầu nhận dữ liệu.

Vi mạch Intel 8279, 8276 có số phím nhỏ hơn 64 phím, thường dùng cho các hệ vi xử lý, làm cả nhiệm vụ tạo mã cho bàn phím và hiển thị ra đèn LED 7 đoạn.

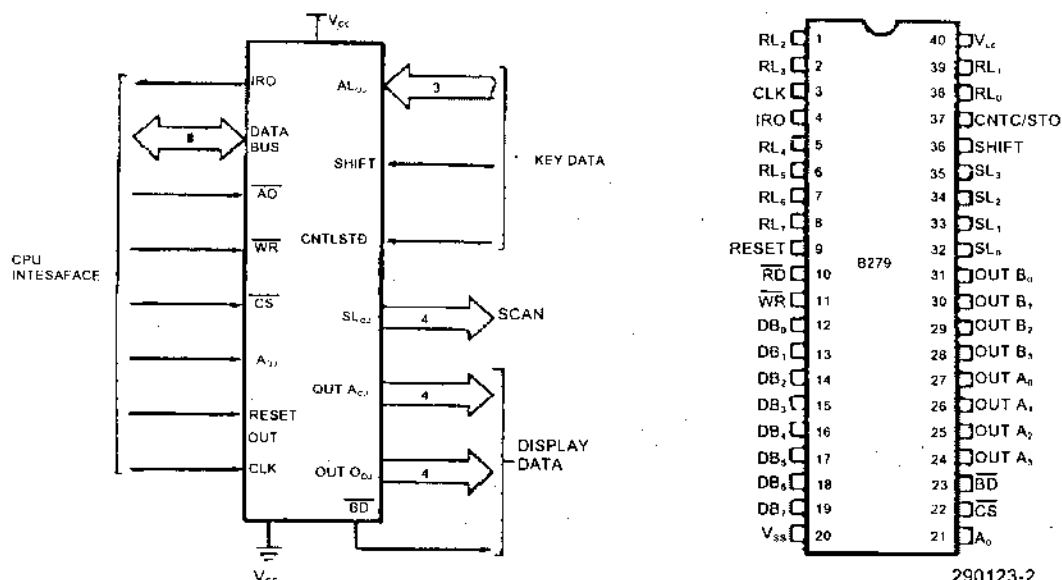
Vi mạch 8042 và 8048 là hai VDK có trong bàn phím máy vi tính, tạo mã cho bàn phím của máy vi tính PC, có tới 101 phím.

14.4.1. Ghép nối bàn phím với vi mạch 8279

1. Cấu tạo

Vi mạch có sơ đồ chân như hình 14.13. Vi mạch này có thể mắc với bàn phím và đèn hiển thị LED 7 đoạn. Trong chương 11, chúng ta đã xét các kiến thức của 8279 và mắc nó với đèn hiển thị, ở đây, chỉ xét tới việc mắc với bàn phím.

Vi mạch có sơ đồ khối như hình 14.14 và sơ đồ mắc với bàn phím và đèn hiển thị LED 7 đoạn như hình 14.15. Vi mạch này có thể mắc tối đa 8x8 phím.



Hình 14.13. Sơ đồ chân của 8279

2. Tạo và đọc mã bàn phím

Bàn phím được mắc với các đường dây:

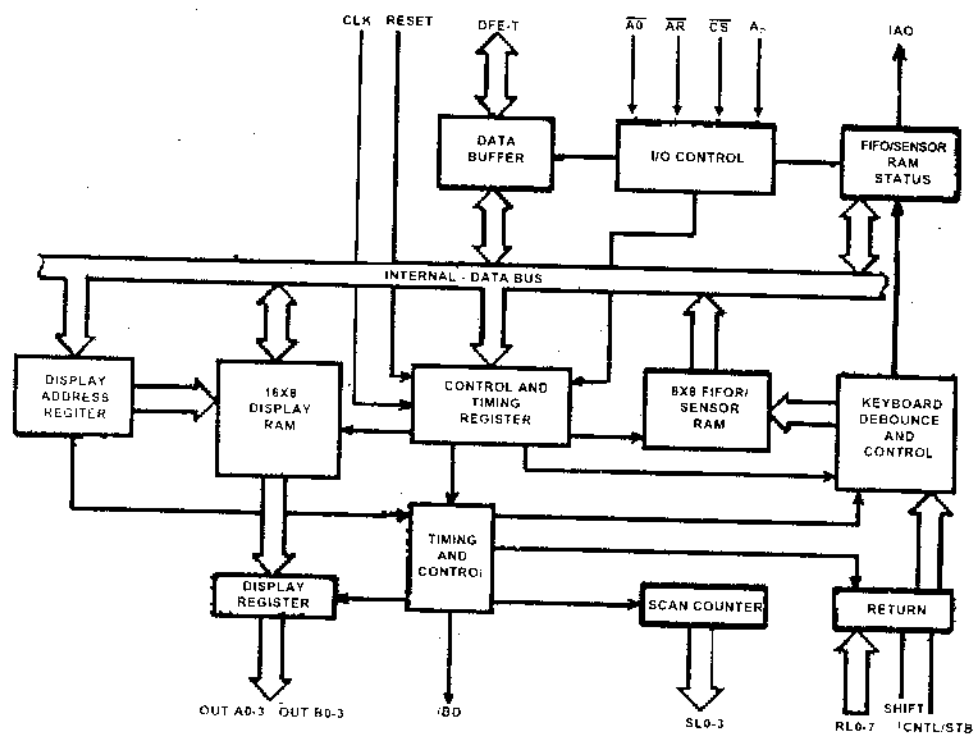
- Vào/ra RL0-RL7 cho các đường dây hàng và cột của bàn phím. Hình 14.16 có 8 hàng và 3 cột (qua giải mã 3-8 cho $8 \times 3 = 24$ phím. Có thể mở rộng tới 8×8 phím nếu nối hết các đường dây cột.

- Tín hiệu vào SHIFT của phím SHIFT để tạo chữ hoa và chữ thường.

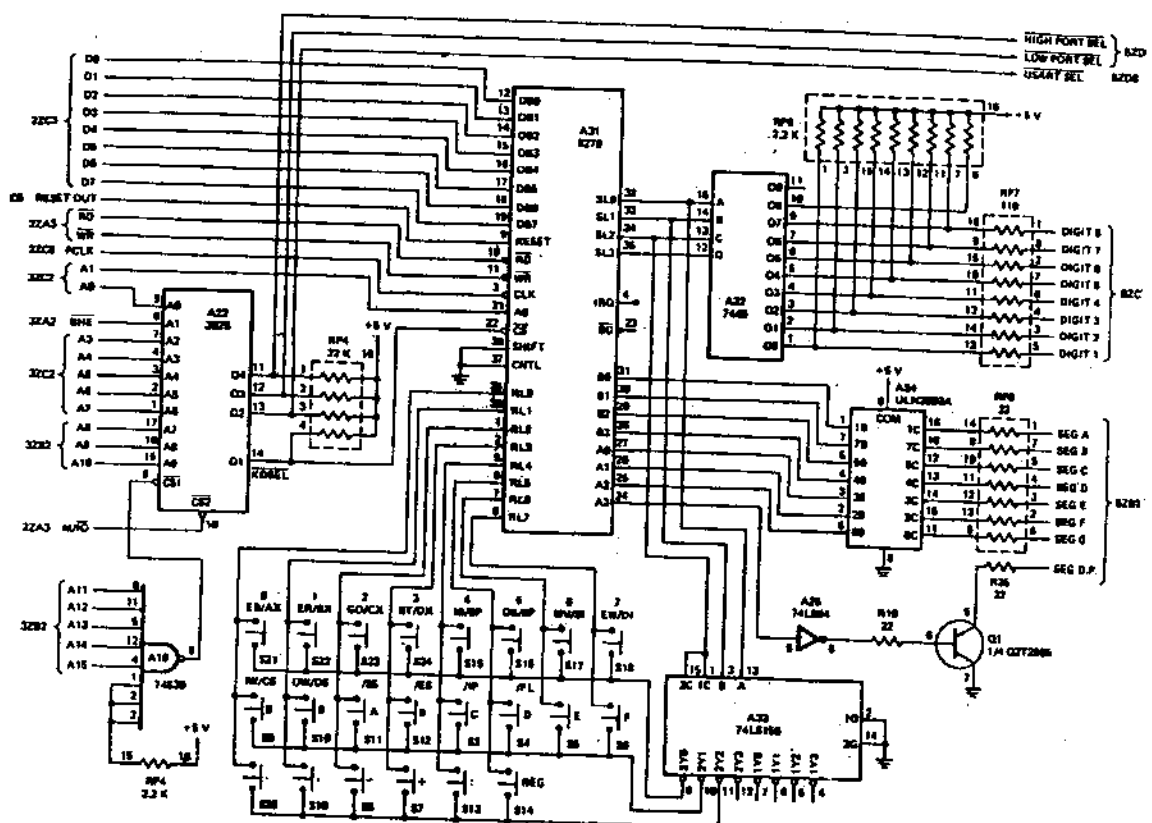
- Tín hiệu CNTL-STB điều khiển ghi dữ liệu của bàn phím.

Chế độ quét bàn phím được xác lập bởi lời Xác lập bàn phím/hiển thị bởi ba bit KKK cho 8 chế độ quét bàn phím (xem chương 11). Mã của phím được đọc vào bởi 8 bit RL0-RL7.

Mã phím được tạo và đọc vào bộ nhớ RAM với các lệnh sau:



Hình 14.14. Sơ đồ khối của 8279



Hình 14.15. Sơ đồ mắc bàn phím và hiển thị của 8279

- Ghi lời điều khiển đọc RAM FIFO là 20H.
- Đọc dữ liệu của RAM FIFO.

Ví dụ 14.11 giới thiệu chương trình khởi phát, đọc mã và hiển thị của 8279 có địa chỉ FFEAH cho thanh ghi dữ liệu.

Vì mạch này thường được dùng cho hệ vi điều khiển, hệ vi xử lý có số phím bằng hay nhỏ hơn 64 phím.

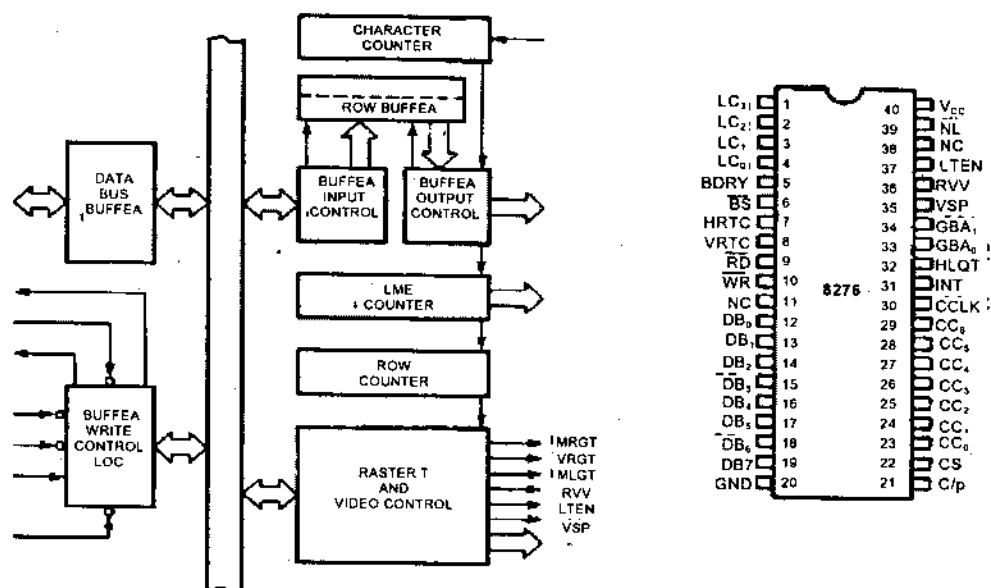
14.4.2. Ghép nối bàn phím với vi mạch 8276

Vì mạch 8276 được chế tạo để ghép nối bàn phím và ống hình CRT. Vì mạch có sơ đồ khối và sơ đồ chân như hình 14.16.

1	0000	; Ví dụ 14.11: Chương trình khởi phát, đọc mã và hiển thị của 8279	
2	0000	ORG 00H	
3	0000	LJMP MAIN	
4	0003	KB_DATA DS 1	
5	0004	MAIN:	; khởi phát
6	0004	MOV DPTR,#0FFEAH	; ghi địa chỉ thanh ghi điều khiển
7	0007	MOV A, 00H	; chế độ vào bên trái, quét mã
8	0009	MOVX @DPTR, A	; ghi lời điều khiển
9	000A	MOV A, #00111000b	; lời clock chia cho 24
10	000C	MOVX @DPTR, A	; đưa ra
11	000D		; đọc mã phím
12	000D	MOV A, 01000000B	; lời điều khiển đọc RAM FIFO
13	000F	MOV DPTR,#0FFEAH	; ghi địa chỉ thanh ghi điều khiển
14	0012	MOVX @DPTR,A	; đưa lời điều khiển ra 8279
15	0013	MOV DPTR,#0FFE8H	; ghi địa chỉ thanh ghi dữ liệu
16	0016	MOVX A, @DPTR	; đọc mã phím vào
17	0017	MOV KB_DATA, A	; gửi vào ô nhớ KB_DATA
18	0019		; đưa ra hiển thị
19	0019	MOV A, #10010000B	; ghi vào RAM hiển thị, vị trí thứ nhất, tự tăng
20	001B	MOV DPTR,#0FFEAH	; địa chỉ thanh ghi điều khiển
21	001E	MOVX @DPTR,A	; ghi lệnh ra
22	001F	MOV DPTR, #0FFE8H	; địa chỉ thanh ghi dữ liệu
23	0022	MOV A, #KB_DATA	; đọc dữ liệu vào A
24	0024	MOVX @DPTR, A	; đưa dữ liệu ra hiển thị
25	0025	SJMP \$	; kết thúc chờ

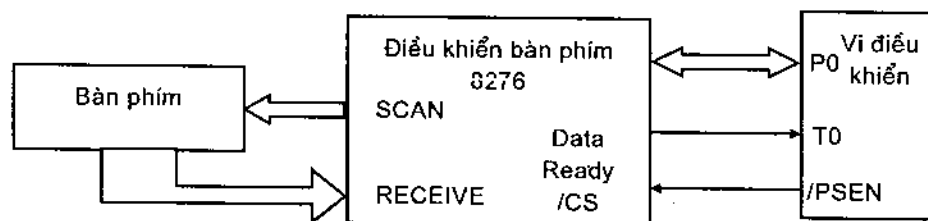
Tương tự vi mạch 8279, vi mạch này cũng được ghép nối với bàn phím qua:

- 8 đường dây hàng đi vào VDK.
- 8 đường dây cột đi ra từ vi mạch giải mã 74156 mà 3 lối vào của mạch này nối với 3 đường dây ra của VDK.



Hình 14.16. Sơ đồ khối và sơ đồ chân của 8276

Ghép nối giữa vi mạch 8276 với bàn phím và VDK như hình 14.17.



Hình 14.17. Sơ đồ mắc giữa 8276 với bàn phím và VDK

Vi mạch 8276 trao đổi tin song song với VDK theo kiểu ngắt thời gian ngoài INT0 và khi có dữ liệu về mã bàn phím, có tín hiệu yêu cầu đưa vào lối vào T0. VDK đọc dữ liệu vào bằng cách đưa ra tín hiệu /PSEN vào lối vào /CS của 8276.

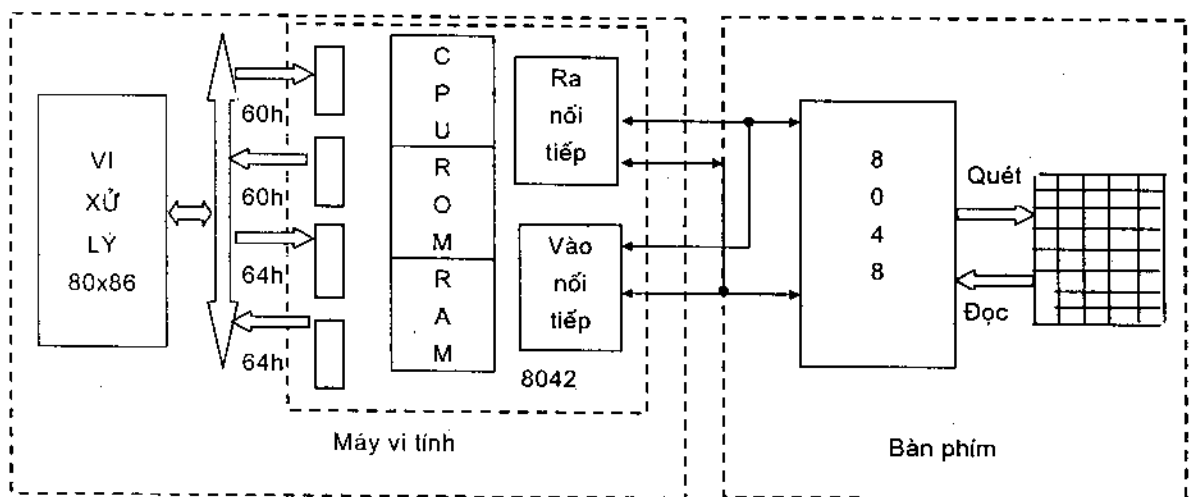
Chương trình phục vụ bàn phím của 8276 cũng tương tự vi mạch 8279. Sau khi khởi phát, khi đưa ra các tín hiệu quét SCAN và đọc các tín hiệu nhận RECEIVE, khi có phím nhấn, dữ liệu mã của phím được tạo thành, có tín hiệu DATA READY đưa vào lối vào T0 để ngắt chương trình. VDK đưa tín hiệu /PSEN để đọc ra dữ liệu mã của phím.

14.4.3. Ghép nối bàn phím PC AT với VDK

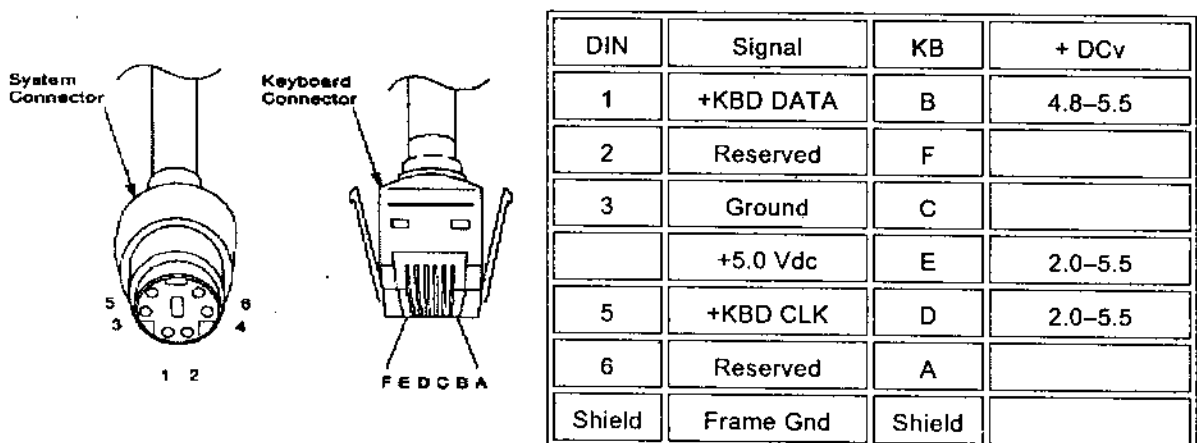
1. Ghép nối bàn phím trong máy tính PC AT

Bàn phím của máy vi tính hiện nay có 101 phím, được mắc với VDK 8048 để tạo mã và đưa tin vào bộ đệm bàn phím (có VDK 8042) nằm trong máy vi tính như hình 14.18. Giữa bàn phím và bộ đệm trong máy vi tính có trao đổi tin qua đường dây và ổ cắm nối tiếp với các tín hiệu như hình 14.19. VDK 8048 dùng để tạo mã phím và trao đổi tin nối tiếp với bộ đệm bàn phím trong máy vi tính. Trong phần này, chúng ta không xét chi tiết việc tạo mã quét lúc nhấn và nhả phím mà chỉ quan tâm tới việc ghép nối và trao đổi tin giữa bàn phím PC với VDK, cụ thể 8042 để ghép nối và trao đổi tin với một VDK khác.

Bàn phím của PC gồm các phím loại ma trận điện dung (keyboard capacitive matrix) được vi điều khiển điều khiển 8048 mắc qua 12 đường dây MD00 tới MD11 (nối với các chân P10 tới P27). Các đường dây SENSE A tới SENSE H nối với bộ khuếch đại (Sense amplifier) rồi đưa vào vi điều khiển qua các đường dây SA closed, Select 0 tới Select 2. Tín hiệu Reset dùng để xóa bộ khuếch đại. Các tín hiệu +Serial Data dùng để truyền tin nối tiếp từ bàn phím vào máy vi tính, còn tín hiệu Request/Clock dùng để truyền yêu cầu trao đổi tin tin (-Request) hoặc tín hiệu nhịp (Clock) tới máy vi tính.



Hình 14.18. Sơ đồ mắc vi xử lý 80x86 với bàn phím AT



Hình 14.19. Sơ đồ chân cắm và các tín hiệu của bàn phím PC

Bàn phím PC này có nhiệm vụ:

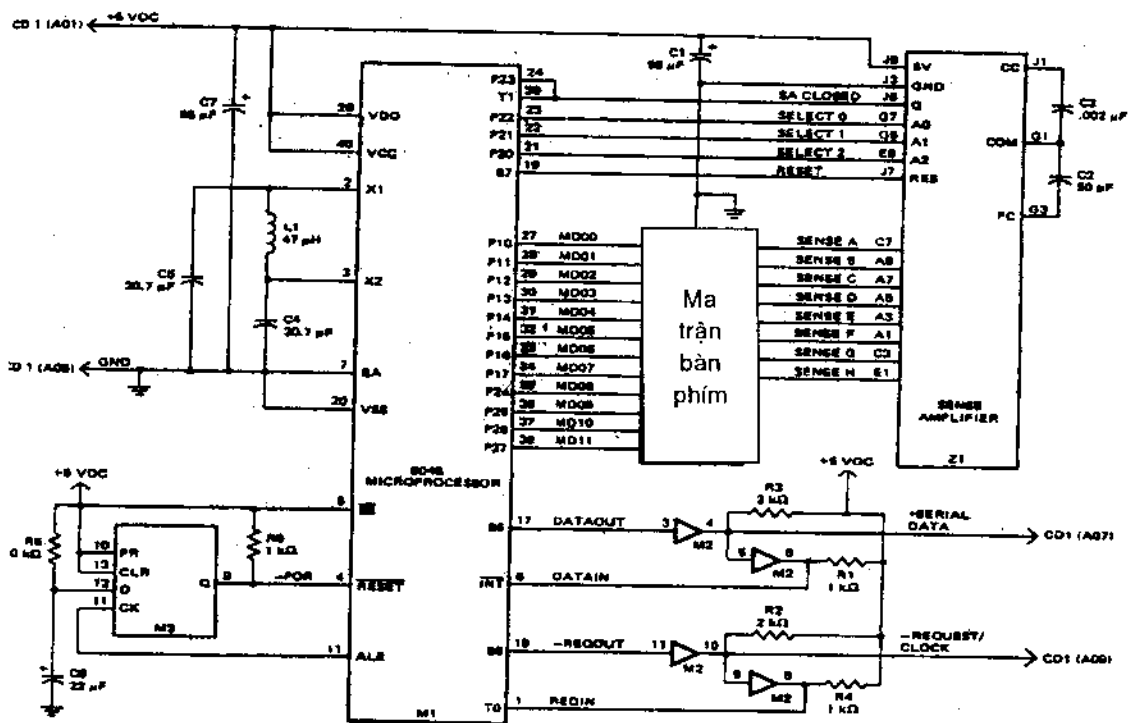
- Tạo mã quét (theo thứ tự xếp đặt) và tới 256 mã ASCII cho các phím.
- Ghi nhớ tạm vào bộ nhớ đệm.
- Đưa tín hiệu cờ (flag) để máy vi tính đọc dữ liệu vào.
- Truyền tin nối tiếp vào máy vi tính.

Do đó, cũng như hệ vi xử lý, vi mạch điều khiển này là một vi điều khiển hay vi xử lý vào-ra.

Trong máy vi tính cá nhân PC-IBM, người ta đã dùng vi mạch Intel 8242 hay 8248.

Chương trình trong ROM của vi điều khiển 8048 điều khiển việc kiểm tra các mạch điện tử của bàn phím khi nó vừa được nối với nguồn nuôi và thực hiện quét bàn phím theo ba công đoạn (bàn phím tốt hay xấu, phát hiện phím bị nhấn và chống rung, tạo mã phím nhấn).

Sơ đồ mắc của 8048 như hình 14.20.



Hình 14.20. Ghép nối bàn phím với VĐK 8048 trong bàn phím PC

Việc ghép nối và các tin trao đổi giữa bàn phím và vi xử lý của máy tính PC như giới thiệu trên hình 14.21.

Khi có phím của bàn phím bị nhấn, VDK 8042 sẽ xử lý (chống rung, tạo mã quét) rồi gửi mã quét tới VDK 8048 của máy tính để xử lý. VDK này sẽ gây ra ngắt INT9H cho máy vi tính. Máy vi tính sẽ điều khiển VDK này đọc mã ASCII có trong bộ nhớ ROM (địa chỉ là mã quét) rồi gửi vào bộ đệm bàn phím chờ vi xử lý của máy tính đọc vào. Tín trao đổi giữa Vi xử lý và VDK 8048 thông qua cổng có địa chỉ 60H và 61H (xem bàn phím của máy vi tính PC).

2. Ghép nối bàn phím PC với VĐK

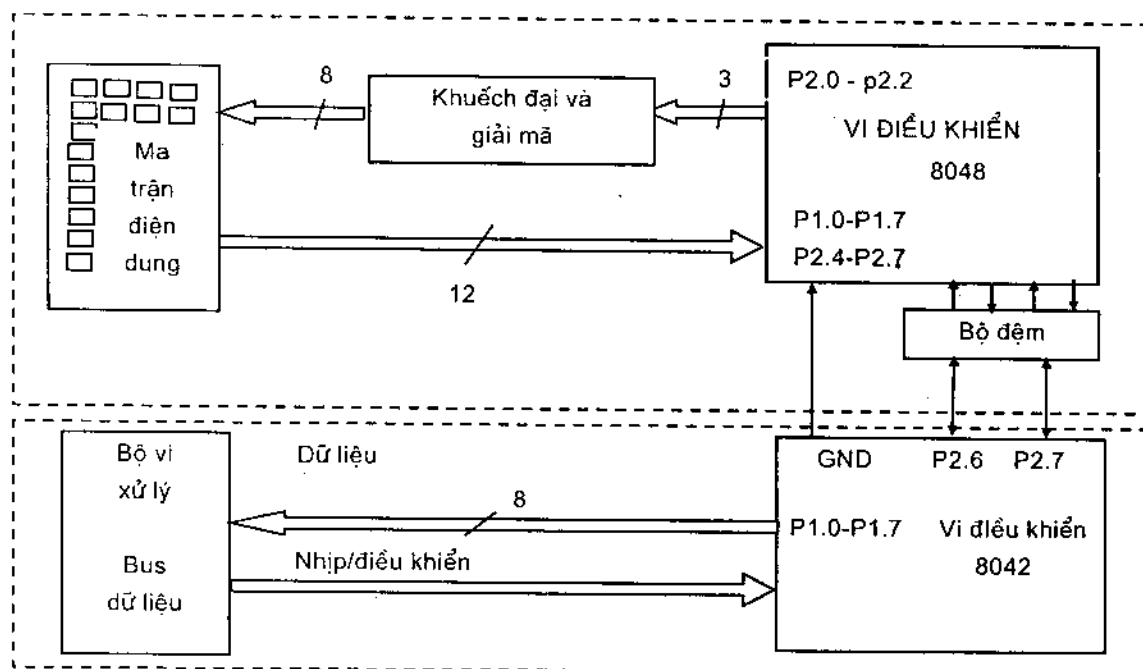
Muốn ghép nối bàn phím PC với VĐK, ta quan tâm tới các tín hiệu và thủ tục trao đổi tin nối tiếp giữa hai vi điều khiển 8048 và 8042, để từ đó có thể ghép nối và viết chương trình phục vụ.

VDK muốn ghép nối với bàn phím PC phải tuân theo ổ cắm tròn hay dẹt như hình 14.19 và trao đổi tin nối tiếp, hai chiều, trong đó :

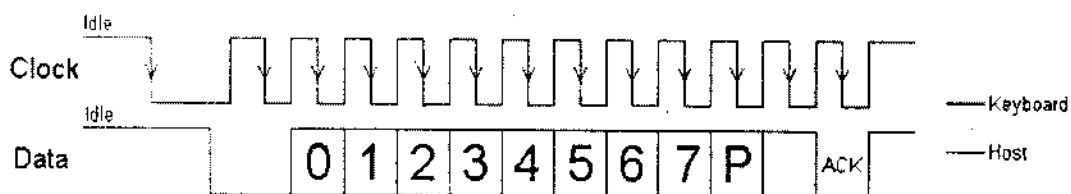
- Chân số 1 (hay B): dành cho dữ liệu nối tiếp hai chiều.
- Chân số 5 (hay D): dành trao đổi tín hiệu nhịp.

Tín hiệu nhịp (CLK) và tín hiệu nối tiếp có dạng như hình 14.22 cho tín trao đổi giữa máy tính và bàn phím (a) và giữa bàn phím và máy tính (b).

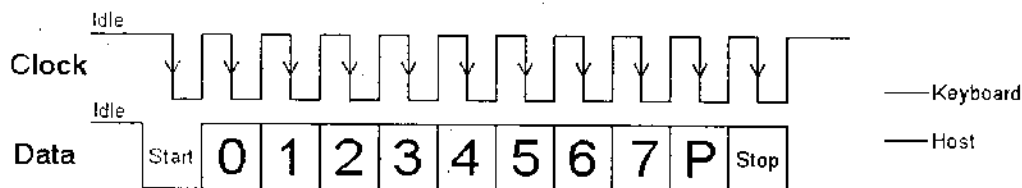
Hình 14.22 giới thiệu sơ đồ và các tín hiệu trao đổi tín KBD Clock (1) và KBD DATA (2) với bàn phím của máy tính PC AT của VĐK HC7051A. Ngoài hai tín hiệu trên, VĐK còn nối trực tiếp với các đèn chỉ thị Num Lock, Caps Lock và Scroll Lock của bàn phím. VĐK trên còn có chân RESET và truyền dữ liệu nối tiếp ra Tx/D.



Hình 14.21. Tín trao đổi giữa bàn phím và vi xử lý trong PC



a) Dạng tín hiệu nhịp và tín giữa VDK - bàn phím - VDK

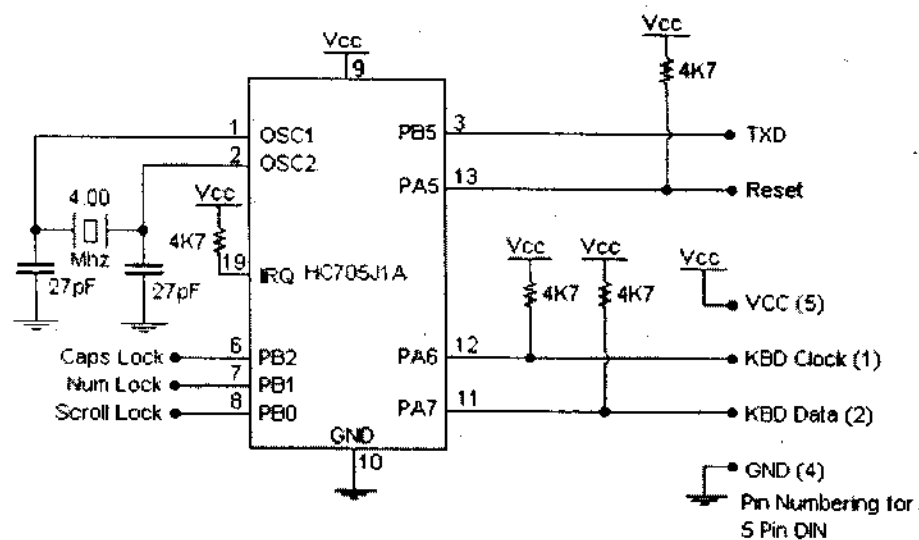


b) Dạng tín hiệu nhịp và tín giữa VDK - bàn phím

Hình 14.22. Dạng tín hiệu nhịp và tín giữa VDK - bàn phím

Chương trình giữ đường dây KBD Clock ở mức thấp. Chương trình giữ vòng lặp khi đường dây rỗng (Idle). Nếu KBD clock xuống thấp, vòng lặp bị đứt và chân KBD Data được đọc. Điều trên xảy ra khi bit Start xuống mức thấp. Nếu không, chúng ta phân nhánh tới bắt đầu chương trình con nhận dữ liệu và lặp lại như đoạn sau.

Hình 14.23 giới thiệu sơ đồ mắc nối giữa bàn phím và VDK HC 705J1A và các chương trình con phục vụ bàn phím.



Hình 14.23. Sơ đồ và các tín hiệu trao đổi giữa VDK HC705J1A và bàn phím

14.5. BÀN PHÍM TRONG THIẾT BỊ ĐIỀU KHIỂN TỪ XA

Nhiều thiết bị điện tử dân dụng (tivi, quạt, điều hoà nhiệt độ, chuông điện v.v...) cũng như thiết bị khoa học, hay truyền thông (xe tự hành, người máy v.v...) đều sử dụng cách điều khiển từ xa với thiết bị điều khiển có phím nhấn. Phương tiện truyền tín hiệu điều khiển từ bộ phận phát tín hiệu điều khiển tới bộ phận thu tín hiệu điều khiển có thể là ánh sáng hồng ngoại, siêu âm, sóng vô tuyến v.v... trong đó tia hồng ngoại là thông dụng nhất.

Nguyên tắc điều khiển từ xa là:

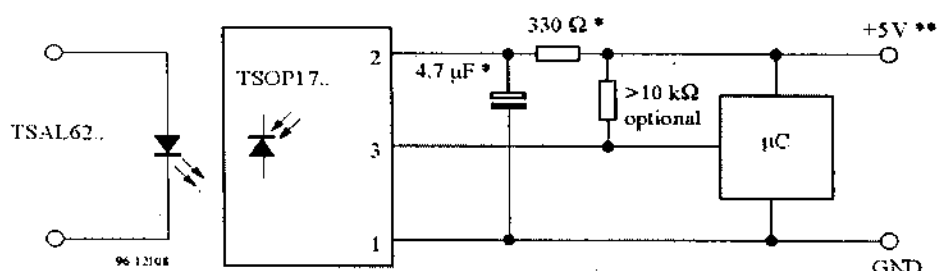
- Bộ điều khiển phát các tín hiệu điều khiển (số hay tương tự) đã mã hoá ra không khí khi nhấn các phím điều khiển khác nhau.

- Bộ phận nhận tín hiệu điều khiển có bộ thu nhận các tín hiệu tới, giải mã và đưa điều khiển các hành động khác nhau tùy tín hiệu điều khiển phát đi.

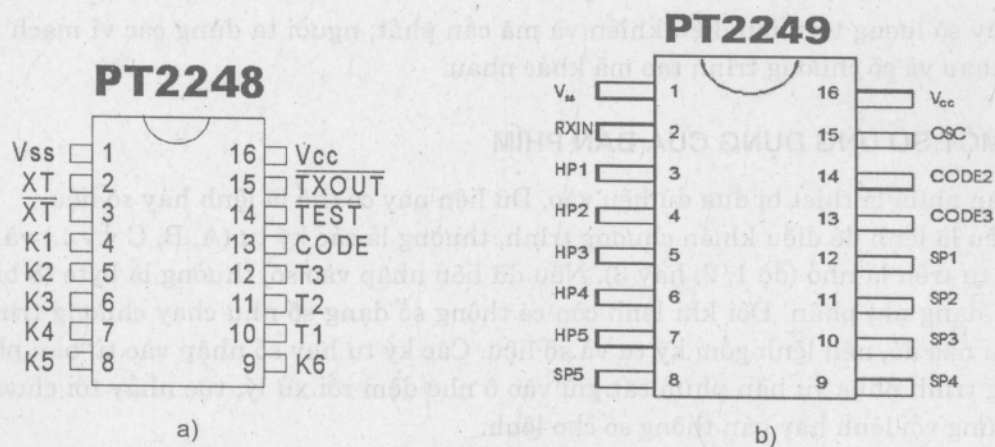
Để chống nhiễu, chống suy giảm và giữ bí mật, tín hiệu điều khiển phát và thu còn điều biến (modulation) hay giải điều biến (demodulation) theo biên độ, tần số hay pha của sóng mang cao tần.

Sơ đồ tổng quát một mạch phát và thu của một mạch điều khiển từ xa dùng sóng hồng ngoại như hình 14.24. Trong mạch, dùng hai vi mạch phát PT2248 và thu PT2249 với các chân ở hình 14.25.

Ở đây, chúng ta chỉ thảo luận về bàn phím, mạch tạo mã của tín hiệu phát và thu tín số dưới dạng sóng hồng ngoại. Với sóng cao tần cũng như siêu âm, nguyên tắc cũng tương tự với ánh sáng hồng ngoại.



Hình 14.24. Nguyên tắc phát và thu sóng hồng ngoại



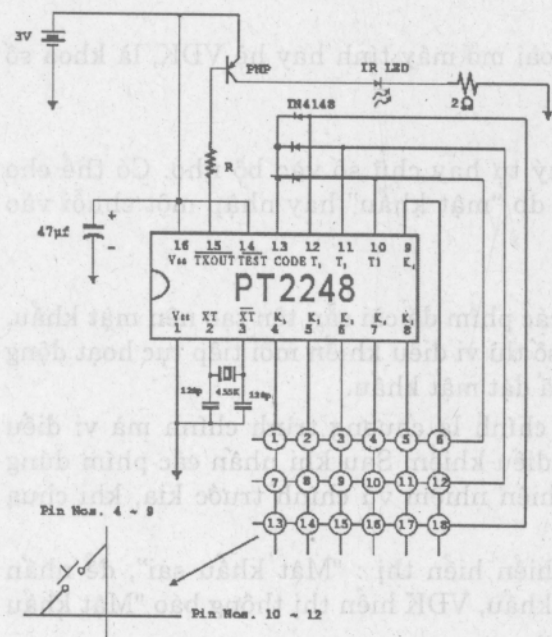
Hình 14.25. Sơ đồ chân của vi mạch phát PT2248 (a) và thu PT 2249 (b)

1. Mạch tạo mã phát

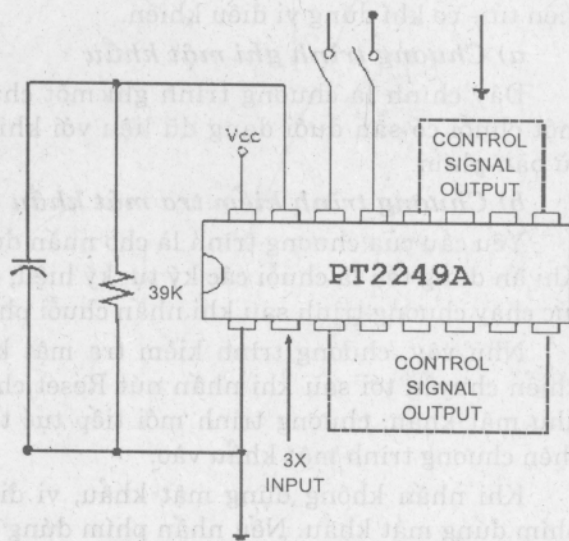
Mạch tạo mã phát để điều khiển từ xa do phát chùm tia hồng ngoại ở đèn LED như ở hình 14.26. Các phím nhấn số 1 tới 18 được nối thành ma trận và với chân của vi mạch phát PT 2248 như hình vẽ. Khi nhấn một phím, mã nối tiếp được tạo ra và phát trên đèn LED. Tùy số thứ tự phím, ta có mã khác nhau.

2. Mạch thu mã phát

Mạch thu mã cũng trên đèn LED và giải mã để tạo ra các tín hiệu khác nhau ở lối ra mạch thu trên cơ sở vi mạch thu PT 2249 như hình 14.27. Mã của mạch thu này sẽ giống mã phát và được giải mã thành tín hiệu điều khiển đưa ra ở lối ra của mạch thu mã. Giữa mạch phát và thu phải có sự kết hợp để tạo mã. Người ta phải đọc mã quét của từng phím và lập trình để tạo mã phát.



Hình 14.26. Mạch phát sóng hồng ngoại qua đèn LED với bàn phím và vi mạch PT 2248



Hình 14.27. Mạch thu sóng hồng ngoại qua đèn LED với vi mạch PT 2249A

Tùy số lượng tín hiệu điều khiển và mã cần phát, người ta dùng các vi mạch thu phát khác nhau và có chương trình tạo mã khác nhau.

14.6. MỘT SỐ ỨNG DỤNG CỦA BÀN PHÍM

Bàn phím là thiết bị đưa dữ liệu vào. Dữ liệu này có thể là lệnh hay số liệu.

Nếu là lệnh để điều khiển chương trình, thường là các ký tự (A, B, C v.v...) và số lượng các ký tự trên là nhỏ (độ 1, 2, hay 3). Nếu dữ liệu nhập vào số, thường là byte (8 bit hay lời 16 bit) dạng nhị phân. Đôi khi lệnh còn có thông số dạng số như chạy chương trình từ địa chỉ đầu nào đó, nên lệnh gồm ký tự và số liệu. Các ký tự hay số nhập vào từ bàn phím được chương trình phục vụ bàn phím cất giữ vào ô nhớ đệm rồi xử lý, tức nhảy tới chương trình tương ứng với lệnh hay gán thông số cho lệnh.

1. Ghi và mở mật khẩu bằng nhấn phím

Mật khẩu là một chuỗi ký tự và chữ số mà người điều hành tự đặt để chương trình ghi vào bộ nhớ của máy vi tính hay VDK và sau đó kiểm tra bằng cách nhấn phím.

Trong máy vi tính, chúng ta đã cài mật khẩu hay từ khoá (keyword) để khoá máy và chỉ mở được nó khi gõ phím đúng mật khẩu đã đặt. Chương trình viết mật khẩu đã có sẵn, người điều hành máy chỉ cần đặt tên các phím cần gõ vào khi cần mở máy. Nếu gõ không đúng các phím đã đặt, máy không mở và chờ gõ tiếp cho đúng hoặc tắt máy, không dùng nữa.

Với vi điều khiển, người điều hành vi điều khiển phải:

- Viết chương trình ghi mật khẩu tức một chuỗi ký tự và chữ số tự đặt thành mật khẩu.
- Viết chương trình kiểm tra mật khẩu để khi nhập mật khẩu tức gõ phím đúng các phím (ký tự, số, ký hiệu) đã đặt chương trình kiểm tra mật khẩu này mới kết thúc để chạy chương trình chính (MAIN).

Một ứng dụng cụ thể nữa của mật khẩu, ngoài mở máy tính hay hệ VDK, là khoá số điện tử – cơ khí dùng vi điều khiển.

a) Chương trình ghi mật khẩu

Đây chính là chương trình ghi một chuỗi ký tự hay chữ số vào bộ nhớ. Có thể cho một chuỗi có sẵn dưới dạng dữ liệu với khi báo db "mật khẩu" hay nhập một chuỗi vào từ bàn phím.

b) Chương trình kiểm tra mật khẩu

Yêu cầu của chương trình là chờ nhấn đúng các phím đã cài sẵn tên tạo nên mật khẩu. Khi ấn đúng tên là chuỗi các ký tự, ký hiệu, chữ số thì vi điều khiển mới tiếp tục hoạt động tức chạy chương trình sau khi nhấn chuỗi phím đã đặt mật khẩu.

Như vậy, chương trình kiểm tra mật khẩu chính là chương trình chính mà vi điều khiển chuyển tới sau khi nhấn nút Reset cho vi điều khiển. Sau khi nhấn các phím đúng như mật khẩu, chương trình mới tiếp tục thực hiện nhiệm vụ chính trước kia, khi chưa chèn chương trình mật khẩu vào.

Khi nhấn không đúng mật khẩu, vi điều khiển hiển thị : "Mật khẩu sai", để nhấn phím đúng mật khẩu. Nếu nhấn phím đúng mật khẩu, VDK hiển thị thông báo "Mật khẩu đúng" và kết thúc chương trình kiểm tra mật khẩu.

Chú ý:

- Chương trình trên đưa hiển thị dạng mã ASCII mắc với cổng P0. Nếu muốn hiển thị

dạng mã 7 đoạn hay ma trận diot, phải chèn thêm chương trình con giải mã từ ASCII sang dạng mã tương ứng.

– Trong thực tế, người ta kiểm tra mật khẩu hai lần. Muốn vậy, phải lặp lại đoạn chương trình kiểm tra mật khẩu một lần nữa.

– Nếu chương trình trên dùng cho khoá điện tử – số, phải thêm các đoạn lệnh để thực hiện các hành động sau:

+ Khoá: ở đầu chương trình, có các lệnh để đưa tín hiệu điện cho nam châm điện từ hút chốt của khoá.

+ Mở khoá: ở cuối chương trình, có các lệnh để ngắt tín hiệu điện cho nam châm điện từ nhà chốt của khoá.

2. Điều khiển chương trình theo thực đơn

Nếu ký tự nhập vào là lệnh, ta cần thực đơn để điều khiển chương trình. Nếu thực đơn chỉ có một ký tự, chương trình thực đơn sẽ kiểm tra các ký tự này, nếu đúng, sẽ nhảy tới chương trình có nhãn tương ứng. Chương trình thực đơn của ví dụ cho phím nối trực tiếp với các bit của cổng VĐK.

Thực đơn dạng chữ đơn

Thực đơn kiểu ký tự đơn này là các chữ A, B, v.v... ứng với các hành động khác nhau của chương trình. Cuối cùng, có một ký tự để thoát khỏi thực đơn. Ví dụ 14.12 giới thiệu thực đơn với các ký tự sau: A – hiển thị số 01h ra cổng P1.

B – hiển thị số 02h ra cổng P2.

F – thoát

; Ví dụ 14.12: Thực đơn kiểu ký tự đơn	
org 00h	xrl a, #0Fh
ljmp main	jnz main
org 20h	kt: sjmp \$
keyflag equ 20h	bp: jnb keyflag, bp
org 30h	clr keyflag
kb_data equ 30h	mov a, kb_data
org 40h	ret
main: call bp	HIEN1: mov a, #01
xrl a, #0ah	mov P1, a
jz HIEN1	jmp kt
call bp	HIEN2: mov a, #02
xrl a, #0bh	mov P2, a
jz HIEN2	sjmp kt
call bp	end

Thực đơn kiểu ký tự đơn bị hạn chế số hành động vì số ký tự có hạn, người ta dùng kết hợp tên thực đơn là ký tự và số.

Nếu thực đơn gồm ký tự và một chữ số, ví dụ F1, F2 và F9 thoát khỏi thực đơn. Ta phải nhấn và kiểm tra phím hai lần (ví dụ 14.13).

– Nhấn phím F.

– Nhấn chữ số 1, 2 hay 9 để so sánh.

Ví dụ 14.13 giới thiệu chương trình điều khiển thực đơn sau:

– F1: hiển thị số 1 ra cổng P0.

- F2: hiển thị số 2 ra cổng P2.
- F9: thoát khỏi thực đơn.

3. Nhập dữ liệu từ bàn phím

Trong chương trình, đôi khi cần dữ liệu nhập vào từ bàn phím, ví dụ, xác lập thời gian hay số đếm cho chương đếm ở tập một. Do đó, cần chương trình nhận các số đếm từ bàn phím, dạng ASCII hay BCD. Chương trình cần phải đổi sang dạng nhị phân để tích vào hai byte nhớ là HBYTE và LBYTE. Trong chương trình này dùng các phép biến đổi mã như đã xét ở chương về hiển thị LED đơn và LED 7 đoạn.

Nếu cần so sánh với số đếm tức thời của bộ đếm, ta cần chương trình so sánh trong chương TIMER của tập I.

Chương trình nhập dữ liệu từ bàn phím được giới thiệu ở ví dụ 14.14.

; Ví dụ 14.13: Thực đơn kiểu ký tự và số	
<pre> org 00h ljmp main org 20h keyflag equ 20h org 30h kb_data equ 30h org 40h main: call bp ; chờ ký tự từ bàn phím xrl a, #0Fh ; kiểm tra ký tự F jnz main ; về main nếu không đúng main1: call bp ; chờ ký tự từ bàn phím xrl a, #01h ; kiểm tra số 01h jz HIEN1 ; tới HIEN1 nếu là F1 call bp ; chờ ký tự từ bàn phím xrl a, #02h jz HIEN2 ; tới HIEN2 nếu là F2 </pre>	<pre> call bp ; chờ ký tự từ bàn phím xrl a, #09h ; kết thúc nếu là F9 jnz main1 ; về main 1 nếu khác F9 kt: sjmp \$; kết thúc chờ bp: jnb keyflag, bp ; chờ cờ bàn phím clr keyflag ; xoá cờ bàn phím mov a, kb_data ; đọc dữ liệu từ bàn phím ret HIEN1: mov a, #01 ; hiện 01 ra cổng P1 mov P1, a jmp kt HIEN2: mov a, #02 ; hiện 02 ra cổng P2 mov P2, a sjmp kt ; kết thúc chờ end </pre>

; Ví dụ 14.14: Chương trình nhập dữ liệu từ bàn phím	
<pre> macro @docso <l> jnb keyflag, \$ clr keyflag mov a, kb_data anl a, #0FH mov ll, a endmac macro @nhan2 <lbyte> mov a, ll rlc a mov ll, a endmac org 00h ljmp main org 20h keyflag ds 1 </pre>	<pre> mov HBYTE, #00h main: jnb keyflag, main clr keyflag mov a, kb_data anl a, #0FH xrl a, #0Ch jnz main @docso TTHOU @docso THOU @docso HUN @docso TEN @docso UNIT mov r5, #5 mov r0, #TTHOU BCD0: call mul10 clr c </pre>

<pre> org 30h kb_data equ 30h hbyte ds 1 lbyte ds 1 TTHOU equ 35h THOU equ 36h HUN equ 37h TEN equ 38h UNIT equ 39h org 100h mov r0, #00h mov r1, #00h mov LBYTE, #00h mov r1, a mov a, HBYTE mov r2, a clr c @nhan2 LBYTE @nhan2 HBYTE clr c @nhan2 LBYTE @nhan2 HBYTE mov a, LBYTE add a, r1 mov LBYTE,a mov a, HBYTE </pre>	<pre> mov a, LBYTE add a, @R0 mov LBYTE, a jc BCD1 BCD2: inc R0 djnz R5, BCD0 jmp KT BCD1: mov a, HBYTE inc a mov HBYTE,a jmp BCD2 mul10: mov a, LBYTE addc a, r2 mov HBYTE, a clr c @nhan2 LBYTE @nhan2 HBYTE ret KT: pop psw pop acc mov a, LBYTE mov P0, a mov a, HBYTE mov P1, a sjmp \$ end </pre>
--	---

4. Điều khiển chương trình theo lệnh và địa chỉ

Ngoài thực đơn, người ta có thể điều khiển chương trình bằng lệnh (dạng mã ASCII) và địa chỉ nhập vào từ bàn phím.

Khi đó, lệnh phải được quy ước là một hay hai ký tự cố định và số nhập vào chính là địa chỉ đầu tiên của lệnh đầu tiên của chương trình.

Nhiệm vụ của chương trình là:

– *Nhận và kiểm tra lệnh:* Nếu đúng như một trong những lệnh của chương trình thực đơn hay giám sát (monitor) thì chương trình mới chuyển tới nhận dữ liệu là địa chỉ. Nếu không đúng, chương trình sẽ trở về nhận lệnh.

– *Nhận và kiểm tra địa chỉ đầu tiên của chương trình:* Chương trình nhận và kiểm tra địa chỉ đúng của chương trình. Nếu không đúng, lại trở về nhận địa chỉ. Nếu địa chỉ đúng, địa chỉ này nhảy tới lệnh có địa chỉ nhận vào để thực hiện lệnh. Ví dụ một số lệnh sau:

– Lệnh GO XXXX: Thực hiện chương trình tại địa chỉ XXXX, lệnh này tương tự lệnh JMP XXXX.

– Lệnh D XXXX YYYY: Đưa ra màn hình các ô nhớ có địa chỉ từ XXXX tới địa chỉ YYYY.

Chúng ta sẽ thảo luận chi tiết hơn về các lệnh trên ở chương 18, về chương trình giám sát cho một hệ VDK.

CÂU HỎI, BÀI TẬP VÀ THỰC HÀNH

I. Câu hỏi

1. Tác dụng của bàn phím trong hệ vi tính và vi điều khiển?
2. Các loại bàn phím.
3. Các phương pháp tạo mã và đọc mã.
4. Chương trình phục vụ bàn phím là gì? Cách viết cho vi điều khiển họ MCS-51.
5. Thực đơn là gì? Cách viết chương trình cho thực đơn chính, thực đơn con cho một chương trình với một chữ đơn, chữ và số.
6. Vai trò của bàn phím trong việc gọi (hay điều khiển) chương trình.

II. Bài tập

1. Thiết kế và viết chương trình cho một sơ đồ tạo mã dùng thanh ghi có 8 phím để đưa vào một cổng của vi điều khiển.
2. Thiết kế và viết chương trình cho một bàn phím có 16 phím, đưa tin ra theo từng hàng và đọc phím nhấn theo các cột.
3. Thiết kế và viết chương trình cho một bàn phím 8 bit theo kiểu quét.
4. Viết chương trình cho bàn phím 16 phím kiểu ma trận và tạo mã dùng bảng nhớ ROM.
5. Viết chương trình thực đơn có 4 nhiệm vụ là "1, 2, 3, 4".
6. Viết chương trình sử dụng và ghi lại mật khẩu với các từ khoá A, B, C, D, E, F.
7. Viết chương trình cho khoá điện tử-cơ với mật khẩu là tổ hợp 5 chữ số "0, 1, 2, 3, 4".

III. Thực hành

1. Dịch và cho chạy thử các ví dụ có trong chương.
2. Dịch và cho chạy thử các chương trình đã viết ở các bài tập trên.
3. Xem trong chương trình mô phỏng PROTEUS có những loại bàn phím nào.
5. Xem trong chương trình mô phỏng PROTEUS có những chương trình phục vụ bàn phím nào.
6. Dùng bàn phím đã thiết kế trong PROTEUS (có sẵn hay tự thiết kế) chạy mô phỏng các chương trình phục vụ bàn phím (các ví dụ trong chương hay chương trình tự viết ở phần bài tập).

CHƯƠNG 15

GHÉP NỐI VỚI BỘ NHỚ NGOÀI CỦA VI ĐIỀU KHIỂN 8051/8052

Mục đích của chương:

- Hiểu rõ yêu cầu và biện pháp mở rộng bộ nhớ ngoài của một hệ VDK.
- Nắm vững các loại nhớ của một hệ VDK.
- Cách ghép nối một và nhiều vi mạch nhớ trong một hệ VDK.
- Hiểu rõ cách viết chương trình phục vụ cho các loại nhớ trong một hệ VDK.

15.1. TỔNG QUÁT VỀ BỘ NHỚ NGOÀI CỦA VI ĐIỀU KHIỂN 8051/8052

15.1.1. Yêu cầu mắc bộ nhớ ngoài của vi điều khiển

Các vi điều khiển (VDK) 8051/8052 nối riêng và các VDK nối chung đều có bộ nhớ trong với dung lượng hạn chế (RAM trong có 128 byte, PROM trong có 1 kByte, 2 kByte, 4 kByte, 8 kByte v.v...). Do đó, với ứng dụng cần nhiều dung lượng nhớ, người ta phải mắc thêm vào VDK các bộ nhớ ngoài RAM và PROM cho dữ liệu và chương trình.

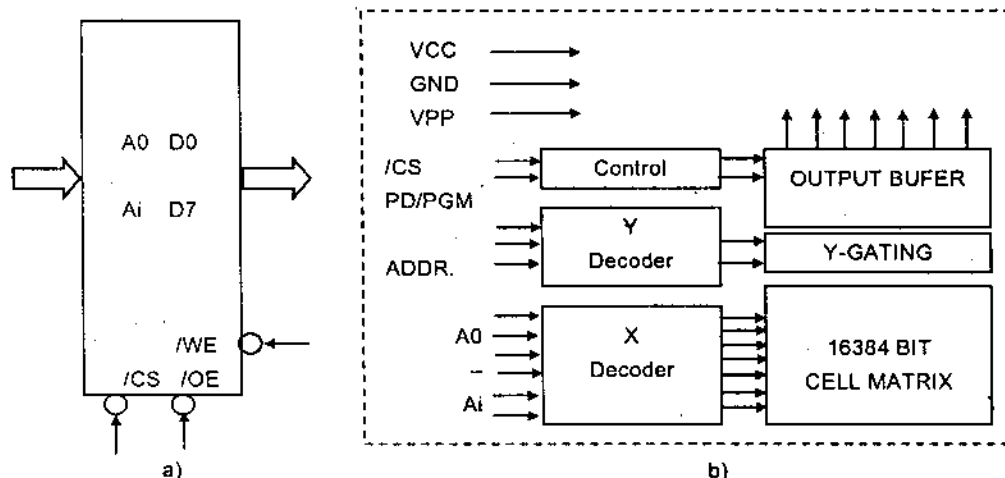
15.1.2. Đặc điểm chung của vi mạch nhớ ngoài

1. Đường dây mắc

Mỗi vi mạch nhớ đều có:

- Các đường dây địa chỉ A0–Ai: được dùng để gọi hay truy cập tới từng phần tử nhớ.
- Các đường dây dữ liệu D0–D7: dùng để đưa dữ liệu vào hay dữ liệu ra.
- Các đường dây điều khiển: dùng để chọn vi mạch /CS (Chip Select) hay /CE (Chip Enable), cho phép dữ liệu ra /OE (Output Enable) và cho phép ghi (/WE–Write Enable).

Một vi mạch (chip) nhớ có sơ đồ chân như hình 15.1a và sơ đồ khối như hình 15.1b.



Hình 15.1. Sơ đồ tín hiệu (a) và sơ đồ khối (b) của chip nhớ

2. Các thông số của một vi mạch nhớ

Mỗi chip nhớ đều có ba thông số cơ bản là dung lượng nhớ, tổ chức bộ nhớ và tốc độ ghi/đọc.

– **Dung lượng nhớ:** Số lượng bit mà các chip nhớ bán dẫn có thể dùng để lưu trữ (ghi) dữ liệu, được gọi là *dung lượng nhớ*. Đơn vị của dung lượng nhớ này là bit tức số phần tử nhớ nhỏ nhất và byte (8 bit). Bội số của bit là kbit (kilô bit = 1.024 bit) và Mbit (mega = 1.000.000 bit).

Dung lượng nhớ này liên quan tới cách tổ chức bộ nhớ theo ma trận và số đường dây địa chỉ. Mỗi vị trí nhớ có thể.

– **Tổ chức bộ nhớ:** Bộ nhớ được tổ chức theo vùng (location) hay ô nhớ bên trong chip vi mạch (IC). Mỗi vùng nhớ có thể có 1 bit, 4 bit, 8 bit hay 16 bit. Số bit của mỗi vùng nhớ bằng số đường dây dữ liệu của IC nhớ. Thường mỗi vùng có 8 bit dữ liệu vì có 8 đường dây dữ liệu D0–D7. Số vùng trong một IC nhớ do số đường dây địa chỉ A0–A7 quyết định. Nếu mỗi IC nhớ có các đường dây địa chỉ A0–Ai, tức có i đường dây địa chỉ, nên số vùng nhớ là 2^i . Ví dụ, một IC nhớ có 8 đường dây dữ liệu D0–D7 và 15 đường dây địa chỉ A0–A15, sẽ có $2^{15+1} = 2^{16} = 64\text{kbyte}$ dữ liệu có 8 bit D0–D7.

Tóm lại:

- Mỗi IC nhớ chứa 2^x vùng nhớ, trong đó x là số địa chỉ của IC nhớ.
- Mỗi vùng nhớ chứa y bit, trong đó y là số chẵn dữ liệu của IC.
- Tổng cộng IC nhớ có $2^x \times y$ bit nhớ.

Bảng 15.1 giới thiệu số vùng nhớ theo đơn vị kbit cho các chip nhớ dùng cho VDK.

Bảng 15.1. Bảng vùng nhớ và đường dây địa chỉ cho chip nhớ dùng trong VDK

x	A0–Ai	2^x
10	A0–A10	1K
11	A0–A11	2K
12	A0–A12	4K
13	A0–A13	8K
14	A0–A14	16K
15	A0–A15	32K
16	A0–A16	64K

– **Tốc độ ghi và đọc ô nhớ:** Tùy từng loại linh kiện chế tạo, thời gian ghi và đọc hay tốc độ ghi và đọc khác nhau (từ vài ns tới trăm ns). Thời gian ghi hay đọc còn được gọi là thời gian truy cập (access) nhớ, là thời gian bắt đầu từ lúc đưa địa chỉ, đưa các tín hiệu điều khiển ghi hay đọc tới thời điểm đưa ra tín hiệu dữ liệu.

3. Các loại vi mạch nhớ

Các vi mạch nhớ được chia thành hai loại: nhớ chương trình, chỉ đọc PROM (Programme Read Only Memory) và nhớ truy cập ngẫu nhiên RAM (Random Acces Memory). Các vi mạch nhớ trên được tổ chức theo ma trận các phần tử nhớ (trigơ nhớ bán dẫn (cho SRAM– nhớ tĩnh Static), mạch nạp và phóng điện trên tụ điện C (cho DRAM– nhớ động Dynamic), tranzito ở mức điện áp logic 1 hay 0 (cho PROM).

a) Nhớ PROM

Là loại nhớ chỉ đọc dữ liệu và dữ liệu không bị mất khi mất nguồn nuôi. Khi ghi, người ta phải dùng điện thế cao (25V hay 12,5V) để tạo tín hiệu 1 cho bit nhớ (để đốt cầu chì hay

tạo một lớp mây điện tử gắn cực nguồn S của nhớ bán dẫn CMOS). Với nhớ bán dẫn, người ta có thể xoá được tín hiệu 1 đã ghi. Có thể xoá bằng tia cực tím của ánh sáng mặt trời trong khoảng thời gian lớn (cỡ nhiều giờ) hay xoá bằng điện áp ngược trong khoảng thời gian nhỏ (cỡ giây) nên gọi là nhớ Flash (chớp sáng). Từ đó, có các vi mạch nhớ PROM sau:

- PROM (Programmable ROM – nhớ ROM ghi được chương trình) hay OTP ROM.

Loại nhớ ROM (Read Only Memory) này người dùng có thể ghi tin logic 1 bằng cách “đốt” hay làm nóng chảy cầu chì hoặc cực của tranzito bằng điện thế cao (25V hay 12,5V). Do đó, loại PROM này còn có tên OTP (One Time Programmable – lập trình một lần) hay ROM “đốt” và chỉ ghi được chương trình một lần duy nhất, không thể xoá được.

- EPROM (Erasable Programmable ROM – nhớ ROM xoá được).

Loại nhớ ROM này có thể ghi và xoá tin hàng ngàn lần. Việc xoá tin (kéo dài cỡ 20 phút) bằng tia cực tím (của ánh sáng mặt trời hay đèn tia tím) thông qua một cửa sổ ở trên lưng IC (thường được che bằng một dải băng dính) nên còn có tên là UV-PROM (Ultra Violet PROM). Việc ghi tin của UV-PROM này theo trình tự sau:

- + *Xoá chương trình*: Lấy ra khỏi chân cắm, đặt dưới ánh sáng UV cỡ 15 – 20 phút.

- + *Ghi chương trình*: Đặt vào giá trên bộ ghi chương trình (programmer), với điện thế 12,5V đặt lên lối vào Vpp của IC nhớ, sẽ xuất hiện đám mây điện tử gắn cực nguồn của tranzito khi ghi tin logic 1 và không có đám mây điện tử khi ghi tin logic 0.

- EEPROM (Electrically Erasable Programmable ROM – ROM ghi chương trình xoá được bằng điện). Loại nhớ này có ưu điểm hơn nhớ EPROM về xoá nhanh bằng điện. Hơn nữa, có thể chọn vùng nhớ để xoá, nhưng ưu điểm cơ bản của loại nhớ này là có thể xoá ngay trên bo (board) mạch với điện thế 12,5V, có khi chỉ 5 – 7V, không cần máy nạp chương trình (programmer).

- Nhớ Flash: Từ những năm 1990, nhớ Flag ROM trở nên thông dụng vì nhiều lý do:

- + Quá trình xoá toàn bộ nội dung nhớ chỉ trong một số giây, nên gọi là nháy mắt (flash) hay chớp nhoáng.

- + Phương pháp xoá bằng điện nên đôi khi còn gọi là Flash EEPROM hay Flash ROM.

Sự khác nhau giữa EEPROM và Flash ROM là khi nội dung nhớ của Flash được xoá thì toàn bộ vi mạch nhớ được xoá, trong khi EEPROM có thể xoá từng vùng hay từng byte mong muốn. Một số Flash ROM gần đây có thể chia thành nhiều khối và xoá được lần lượt từng khối. Trong tương lai, Flash ROM này có thể thay thế ổ đĩa cứng vì thời gian truy cập cỡ 100ns trong khi ổ cứng là chục mili giây. Hiện nay, số lần ghi/xoá là 5000.000 lần cho Flash và EEPROM, còn UV-EPROM là 2000.

- ROM mật nạ: ROM mật nạ là loại ROM mà nội dung của nó được ghi chương trình bởi người chế tạo IC. Nói khác đi, nó không phải là ROM ghi bởi người dùng. Vì quá trình chế tạo là tốn kém nên ROM mật nạ này được dùng khi cần dung lượng lớn và chắc chắn rằng nội dung nhớ không thay đổi. Ưu điểm cơ bản của ROM mật nạ là giá rẻ hơn các loại ROM khác nhưng khi có lỗi, cả IC nhớ bị loại bỏ.

- Ngoài các loại nhớ ROM trên, hiện nay còn nhớ EEPROM mắc nối tiếp hai đường dây I²C (xem trong phần mềm mô phỏng Proteus).

b) Nhớ RAM

Ngoài nhớ PROM chỉ đọc để nhớ chương trình, hệ VDK hay vi xử lý còn có nhớ RAM (đọc và ghi) để nhớ dữ liệu. Nhớ RAM này còn chia ra nhớ RAM tĩnh (SRAM – Static RAM), nhớ RAM động (DRAM – Dynamic RAM) và nhớ RAM không mất nội dung NVRAM (Non Volatile RAM – nhớ RAM không bay hơi).

- SRAM: mỗi phần tử hay bit nhớ RAM này là một trigơ, gồm 6 hay 4 tranzito mắc nối

tăng và ghép nối colectơ-bazơ. Hình 15.2 giới thiệu sơ đồ chân và sơ đồ khối của IC nhớ RAM 6116 2Kx8 bit. IC nhớ RAM này có dung lượng 2Kx8 bit, nên có:

- + 12 đường dây địa chỉ A0–A11
- + 8 đường dây dữ liệu D0–D7.
- + WE (Write Enable): Cho phép ghi dữ liệu.
- + OE (Output Enable): Cho phép đọc dữ liệu.
- + CS (Chip Select): Chọn vi mạch.

– DRAM: Nhớ RAM động (dynamic) này có cấu tạo là các tụ điện C dùng để ghi tín logic 1 (có điện tích $Q_0 = CV_0$) và logic 0 (không điện tích $Q_0 = CV_0 = 0$) với các mạch điện có tranzito để tích (khi ghi), phóng điện (khi xóa) và đọc tín. Do đó nhớ DRAM này có đặc điểm sau:

- + Kích thước nhỏ nên có thể có dung lượng nhớ lớn (cỡ MB), dùng trong hệ vi xử lý.
- + Điện tích trên tụ điện bị rò nên phải có mạch điện làm tươi (refresh) bởi các xung điện nhịp của máy phát đưa vào theo các hàng (RAS) và các cột (CAS) của ma trận nhớ bởi một vi mạch điều khiển nhớ (ví dụ 8203). Do đó, trong hệ VDK, người ta không dùng loại DRAM này.

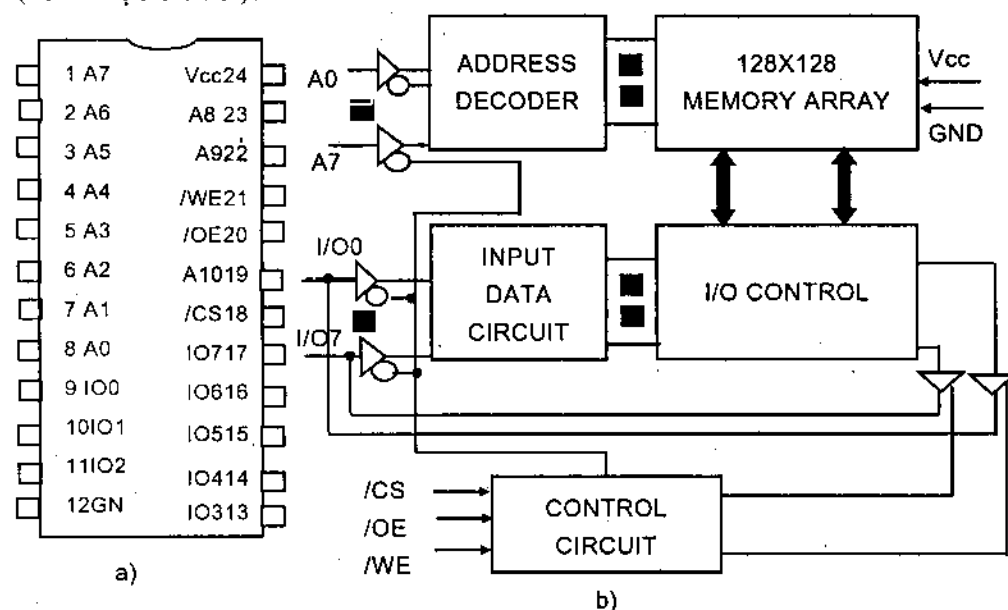
– NV–SRAM: loại nhớ SRAM này không bay hơi (NV–Non Volatile) khi ngắt nguồn điện nuôi vi mạch, vì một trong hai lý do sau:

- + Có nguồn nuôi bằng pin lithium và mạch điều khiển đóng mạch điện khi phát hiện điện thế nguồn nuôi bị suy giảm dưới mức cho phép.

- + Có thêm phần tử nhớ EEPROM mắc song song với SRAM. Do đó, khi ghi, cả hai phần tử nhớ trên được ghi. Khi mất nguồn nuôi, phần tử nhớ để EEPROM vẫn đọc được tín đã ghi.

Vì mạch điện phức tạp, giá đắt, nên NV–SRAM này ít được dùng trong các hệ VDK.

Hiện nay, trong VDK tiên tiến AT89C51RD2/ED2 có cả nhớ Flash EEPROM và NVRAM (xem mục ở dưới).



Hình 15.2. Sơ đồ chân (a) và sơ đồ khối (b) của RAM 6116

15.2. GHÉP NỐI VI MẠCH NHỚ VỚI VI ĐIỀU KHIỂN

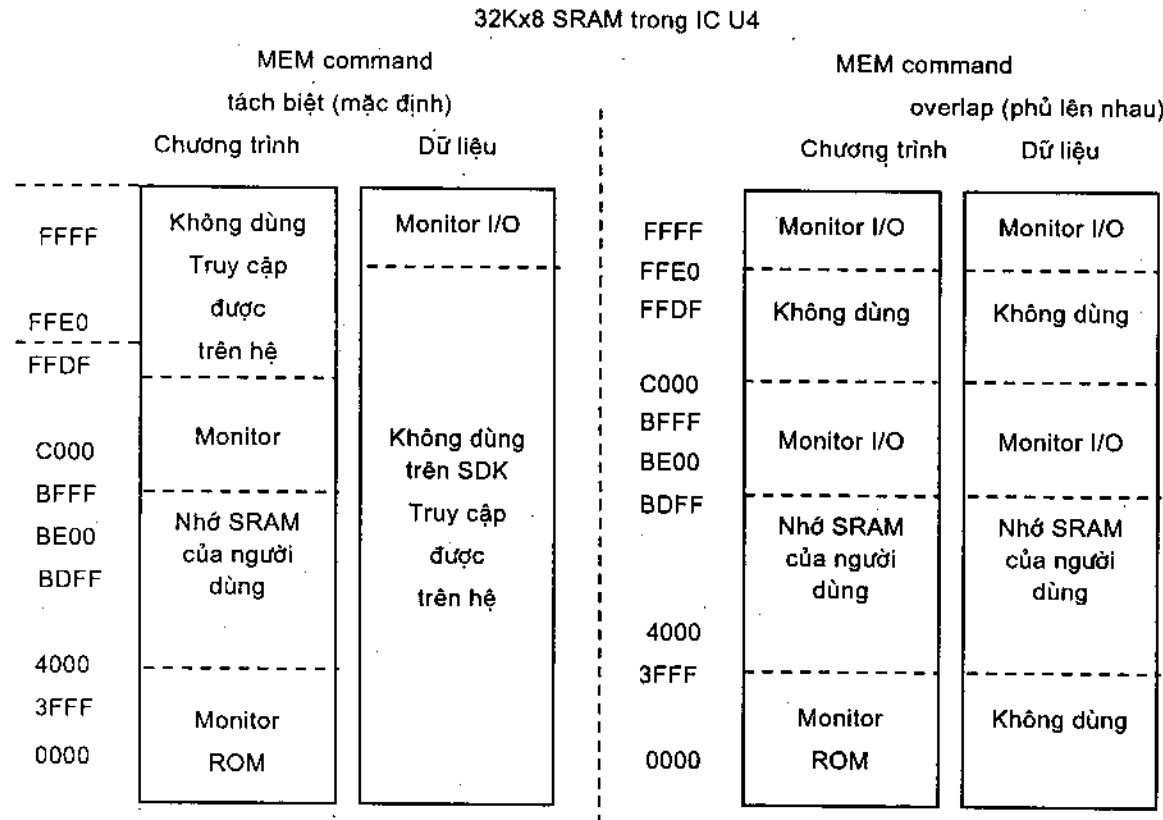
15.2.1. Nguyên tắc sử dụng nhớ ngoài của hệ VDK

Vi điều khiển mắc với các vi mạch nhớ ngoài để ghi chương trình và dữ liệu khi lỗi vào /EA/Vpp nối với đất (còn nối với nguồn nuôi +5V khi dùng các bộ nhớ trong).

Chương trình giám sát (monitor), tiện ích (utility) và dữ liệu (bảng, hằng số) cố định được ghi vào nhớ PROM từ máy vi tính. Việc ghi này nhờ bộ ghi chương trình (programmer) hay có thể ghi thẳng vào bộ nhớ flash trên bo mạch của VDK. Sau đó chương trình sẽ chạy theo sự điều khiển của người điều hành (theo nhấn nút RESET hay nhấn phím từ bàn phím theo thực đơn). Khi ngắt nguồn điện nuôi bo mạch chứa hệ VDK, các chương trình trên các vi mạch nhớ PROM này vẫn còn lưu trong IC nhớ và có thể chạy lại chương trình khi cấp nguồn nuôi và điều khiển bởi người điều hành.

Chương trình và dữ liệu tạm thời của người dùng cũng được ghi từ máy vi tính PC sang bộ nhớ RAM ngoài để chạy chương trình và ghi hay đọc dữ liệu. Khi tắt nguồn điện nuôi bo mạch của hệ VDK, chương trình và dữ liệu này sẽ mất đi.

Về nguyên tắc, các IC nhớ PROM và RAM có thể có đồng thời trên bo mạch và mỗi loại có thể chiếm tối đa là 64k ô nhớ (từ địa chỉ 0000H tới FFFFH) hay nói khác đi, các vùng địa chỉ của hai loại IC nhớ trên có thể chùng hay phủ (overlay) lên nhau vì chúng được đọc bởi các tín hiệu khác nhau (/PSEN cho PROM, /READ cho RAM). Do đó, bản đồ các vùng nhớ của một hệ VDK có thể như hình 15.3.



Hình 15.3. Bản đồ nhớ của SDK của HTE

Trong thực tế, người ta dùng các loại nhớ như sau:

- Nhớ PROM dùng để ghi chương trình cố định như chương trình giám sát (MONITOR), tương tự hệ điều hành của máy vi tính.
- Nhớ PROM dùng để ghi dữ liệu cố định như các bảng dữ liệu, các hằng số v.v...
- Nhớ RAM dùng để ghi chương trình tiện ích (utility) của người dùng và các dữ liệu tạm thời. Chương trình của người dùng này được soạn thảo, dịch và liên kết trên máy vi tính PC rồi được tải sang vi mạch nhớ RAM để chạy chương trình. Khi loại chương trình đã chạy tốt và cần dùng sau này, người dùng sẽ ghi vào nhớ PROM bởi máy nạp.

Người ta có thể kết hợp dùng cả hai loại IC nhớ và phân chia địa chỉ cho hai loại nhớ PROM và RAM với bản đồ địa chỉ nhớ. Hình 15.3 giới thiệu bản đồ địa chỉ các vùng nhớ của 8031 SDK (Software Development Kit) của hãng HTE (Hi Tech Equipment Corporation). Trước khi ghép nối các vi mạch nhớ ngoài cho hệ VDK, người thiết kế phải phân chia các vùng nhớ cho các vi mạch nhớ theo bản đồ các vùng nhớ tương tự hình 15.3 ở trên. Người ta có thể dùng các chuyển mạch để thay đổi các địa chỉ của các vùng nhớ của các IC nhớ. Do đó có hai phương án phân chia vùng nhớ (nhớ PROM và SRAM có địa chỉ tách biệt hay phủ lên nhau) tương ứng với hai bản đồ nhớ.

15.2.2. Nguyên tắc ghép nối IC nhớ ngoài với VDK

Khi dùng các IC nhớ ngoài (PROM, RAM), các lối vào địa chỉ (A0–A_i), dữ liệu (D0–D7) và điều khiển (/CS, /OE, /WE) của IC nhớ phải được nối với các đường dây tương ứng của VDK.

- Đường dây địa chỉ của IC nhớ được nối như sau:

+ *Các đường dây địa chỉ thấp A0–A7*: được nối với vi mạch chốt Latch (ví dụ 74245, 74LS373) như ở hình 15.4. Vi mạch chốt này dùng để ghi nhớ địa chỉ A0–A7 đưa ra cổng P0 tại chu trình trước cùng với tín hiệu điều khiển ghi địa chỉ ALE (Address Line Enable).

+ *Các đường dây địa chỉ cao A8–A15*: được nối với cổng P2 đưa ra địa chỉ cao tại chu trình tiếp theo.

- Đường dây dữ liệu D0–D7 của IC nhớ được nối với cổng P0 để đưa ra hay đọc vào VDK sau khi đã đưa ra địa chỉ thấp và cao A0–A15.

- Các lối vào của PROM được nối như sau:

+ *Lối vào / CE*: được nối với lối ra của bộ giải mã chọn địa chỉ cho nhớ PROM đó.

+ *Lối vào / OE*: được nối với lối ra /PSEN để đọc chương trình ra từ IC PROM.

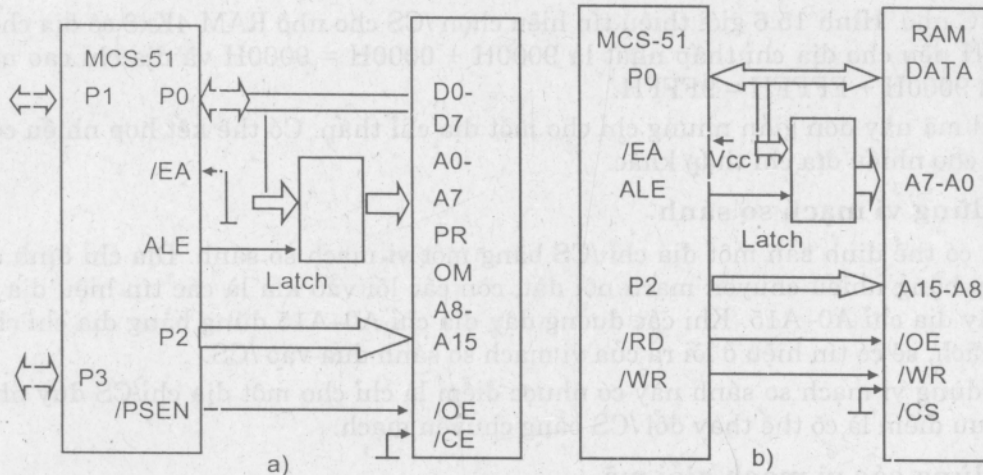
- Các lối vào của RAM được nối như sau:

+ *Lối vào / CE hay / CS*: được nối với lối ra của bộ giải mã chọn địa chỉ cho RAM đó.

+ *Lối vào / WE*: được nối với lối ra /WR của VDK (chân P3,6) để ghi dữ liệu cho RAM.

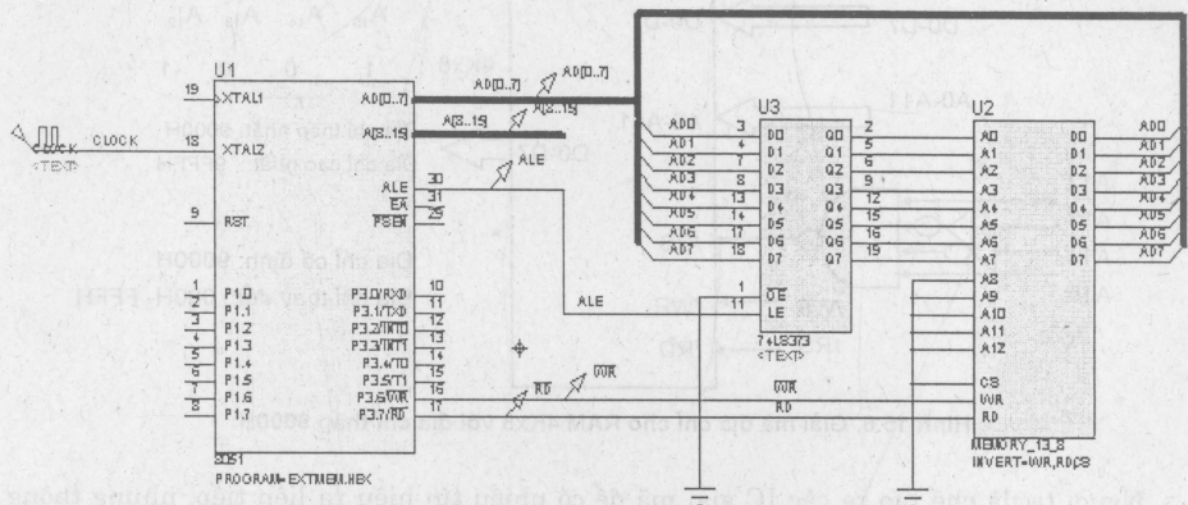
+ *Lối vào / OE*: được nối với lối ra /RD của VDK (chân P3,7) để đọc dữ liệu ra từ IC RAM.

Bộ giải mã sẽ đưa ra các tín hiệu chọn vi mạch (/CE–chip Select) cho các vi mạch nhớ (xem ở dưới) và có khi còn kết hợp cả tín hiệu chọn địa chỉ và điều khiển ghi hay đọc IC nhớ để chọn địa chỉ tương ứng cho đúng. Trong hình 15.4, để đơn giản, các lối vào /CE và /CS đều nối đất vì chưa có bộ giải mã địa chỉ để chọn địa chỉ vùng nhớ cho IC nhớ.



Hình 15.4. Cách mắc vi mạch nhớ ngoài: EPROM (a) và RAM (b)

Hình 15.5 giới thiệu mạch ghép nối VĐK 8051 với nhớ RAM 256 Byte.



Hình 15.5. Cách mắc nhớ ngoài Ram vẽ trong PROTEUS

15.2.3. Giải mã địa chỉ chọn IC nhớ

VĐK hay vi xử lý điều khiển các thiết bị ngoài (qua các cổng vào/ra và các IC nhớ thông qua các tín hiệu chọn địa chỉ /CS và ghi (/WR) hay đọc (/RD) dữ liệu. Để có các tín hiệu /CS này, phải có bộ giải mã địa chỉ. Với các IC nhớ, tín hiệu /CS (hay /CE trong PROM) chính là địa chỉ bắt đầu hay cố định của vùng nhớ dành cho IC nhớ đó. Các địa chỉ thấp A0-Ai của vi mạch nhớ là các địa chỉ để chọn các bit nhớ trong IC nhớ.

Có ba cách giải mã địa chỉ là dùng các vi mạch cổng logic, vi mạch nhớ chuyên dùng và vi mạch giải mã kiểu PLD.

1. Giải mã dùng các vi mạch cổng logic

Các vi mạch cổng logic chỉ chọn một vùng địa chỉ hay một địa chỉ thấp nhất hay địa chỉ

Cách giải mã này đơn giản nhưng chỉ cho một địa chỉ thấp. Có thể kết hợp nhiều cổng logic khác để cho nhiều địa chỉ thấp khác.

Người ta có thể định sẵn một địa chỉ /CS bằng một vi mạch so sánh. Địa chỉ định sẵn được thiết lập bằng nhiều chuyển mạch nối đất, còn các lối vào kia là các tín hiệu địa chỉ của đường dây địa chỉ A0–A15. Khi các đường dây địa chỉ A0–A15 đúng bằng địa chỉ chọn bởi chuyển mạch, sẽ có tín hiệu ở lối ra của vi mạch so sánh đưa vào /CS.

3. Giải mã dùng các vi mạch giải mã

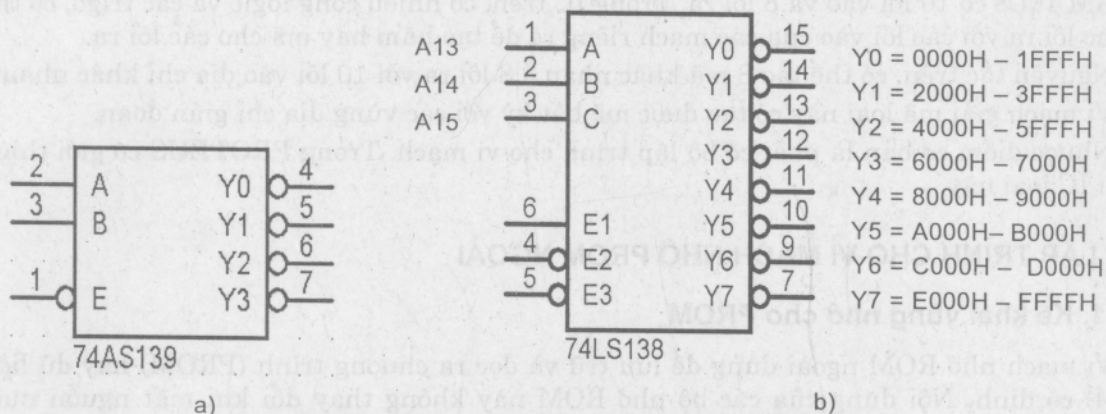
The diagram illustrates the 8255 PPI chip configuration. The data bus is D0-D7, and the address bus is A0-A11. The chip is configured with a 4Kx8 memory map. The address range is from 9000H to 9FFFH, with a fixed address of 9000H. The address range is also shown as 000H-FFFFH.

- Các lối ra (thường có mức logic 0) để điều khiển chọn vùng nhớ liên tiếp cho các IC

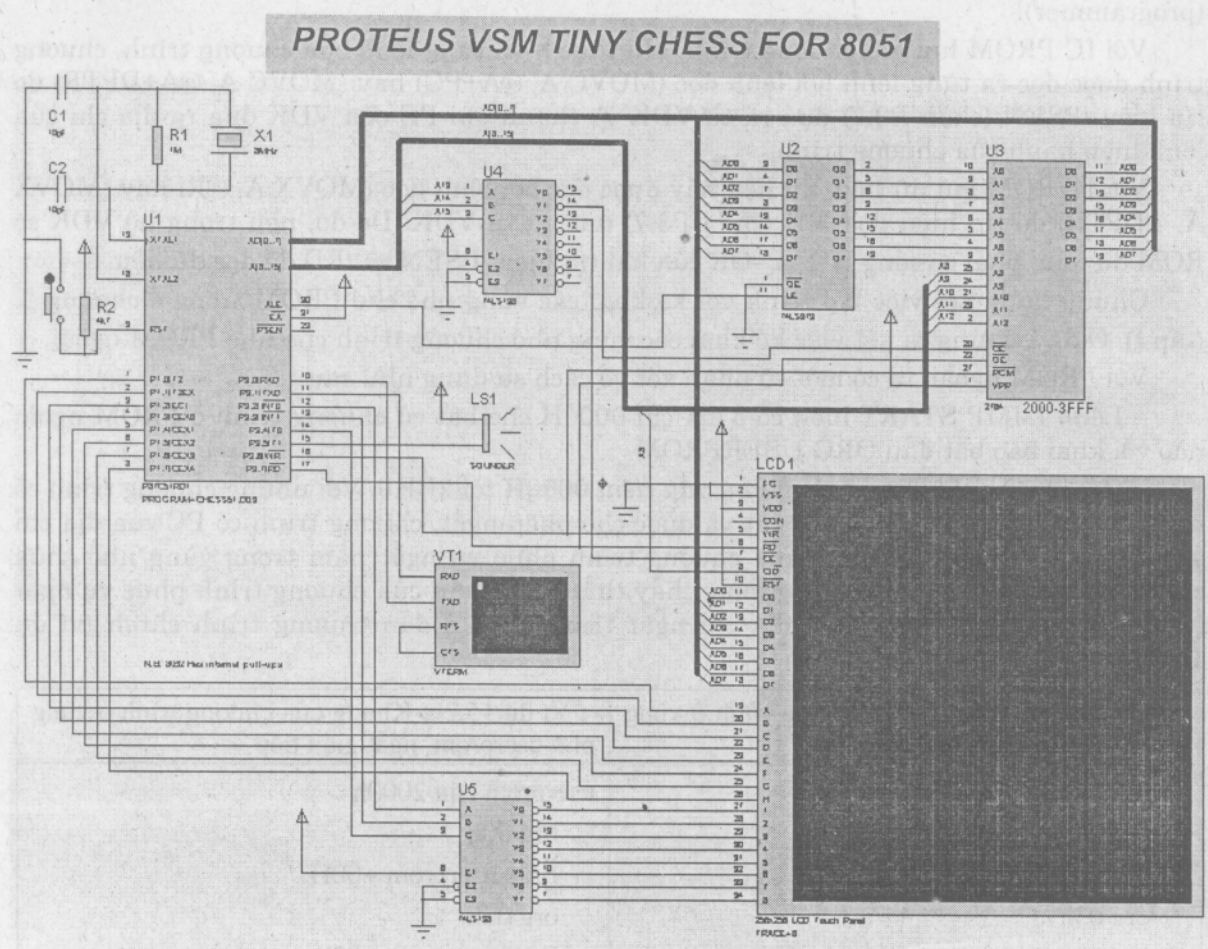
nhớ hay cổng vào/ra của các thiết bị ngoài ghép nối. Các lối ra này sẽ nối với các lối vào /CS của các IC nhớ hay các cổng vào/ra của các thiết bị ngoài.

Hình 15.7 giới thiệu mạch giải mã 74138 mắc cho các lối ra Y0–Y7 để mắc với nhiều IC nhớ và các thanh ghi của các thiết bị vào ra với 8 dải địa chỉ thấp khác nhau.

Hình 15.8 giới thiệu hệ VĐK bàn cờ với 8051 trong PROTEUS, trong đó dùng IC giải mã 74138 để điều khiển IC nhớ ngoài và màn hình LCD đồ họa.



Hình 15.7. Vi mạch giải mã 74139 (a) và 74138 (b)



Hình 15.8. Hệ VĐK bàn cờ dùng 74138 để giải mã địa chỉ nhớ

Ví dụ 15.1 giới thiệu mạch giải mã với 74138 khi các lối vào A, B, C là A11, A12, A13 và A14 = 0, A15 = 0.

4. Giải mã dùng các vi mạch PLD

Các vi mạch PLD (Programmable Logic Device – logic lập trình) là vi mạch AM 16L8 có thể lập trình làm mạch giải mã.

AM 16L8 có 10 lối vào và 8 lối ra. Trong IC trên, có nhiều cổng logic và các trigơ, có thể nối các lối ra với các lối vào của các mạch riêng rẽ để tạo hàm hay mã cho các lối ra.

Nguyên tắc trên, có thể tạo 8 mã khác nhau ở 8 lối ra với 10 lối vào địa chỉ khác nhau.

Vi mạch giải mã loại này có tạo được mã bất kỳ với các vùng địa chỉ gián đoạn.

Nhược điểm cơ bản là phải có bộ lập trình cho vi mạch. Trong PROTEUS có giới thiệu nhiều IC loại này.

15.3. LẬP TRÌNH CHO VI MẠCH NHỚ PROM NGOÀI

15.3.1. Kê khai vùng nhớ cho PROM

Vi mạch nhớ ROM ngoài dùng để lưu trữ và đọc ra chương trình (PROM) hay dữ liệu (ROM) cố định. Nội dung của các bộ nhớ ROM này không thay đổi khi mất nguồn nuôi VDK. Các vi mạch ROM này đều được ghi từ máy vi tính PC sang bởi bộ ghi chương trình (programmer).

Với IC PROM lưu giữ chương trình, khi thực hiện từng lệnh của chương trình, chương trình được đọc ra từng lệnh bởi lệnh đọc (MOVC A, @A+PC) hay (MOVC A, @A+DPTR) do tín hiệu /PSEN (chân P3.7) đưa ra từ VDK và thanh ghi PC của VDK đưa ra địa chỉ của lệnh hiện hành của chương trình.

Với IC ROM giữ dữ liệu, dữ liệu này được đọc bởi lệnh đọc (MOVB A, @Ri hay (MOVX A, @DPTR) do tín hiệu đọc /RD (chân P3.7) đưa ra từ VDK. Do đó, nếu trong hệ VDK có ROM dữ liệu, phải có cổng HOẶC-OR của hai tín hiệu /PSEN và /RD để đọc dữ liệu.

Chúng ta đã xét việc lập trình với kê khai các vùng nhớ cho PROM trong ở chương 2. (tập I). Ở đây, chúng ta xét việc kê khai các vùng nhớ chương trình cho nhớ PROM ngoài.

Với PROM ngoài, ta có một số nhận xét về cách sử dụng như sau:

– Lệnh LJMP START luôn có ở địa chỉ 0000H cho bất cứ chương trình ở PROM ngoài nào với khai báo bắt đầu ORG USERPROM.

– Các vectơ ngắt luôn ở vùng nhớ đầu tiên: 0000H tới 2BH0. Với những chương trình có ngắt, khi có nguyên nhân gây ngắt và được cho phép ngắt, chương trình có PC của địa chỉ vectơ ngắt, rồi nhảy tới nhãn của chương trình phục vụ ngắt nằm trong vùng nhớ chứa chương trình chính. Có thể dùng lệnh nhảy thẳng tới nhãn của chương trình phục vụ ngắt (ví dụ 15.2a). Hoặc nhảy qua địa chỉ ngắt trung gian ở đầu chương trình chính (ví dụ 15.2.b).

; Ví dụ 15.2a: Khung của chương trình ở vùng nhớ userprom, ngắt trực tiếp <pre>USERPROM. EQU 2000h org 00h ljmp start org 03h ljmp int_0_routine org 0Bh</pre>	; Ví dụ 15.2b: Khung của chương trình ở vùng nhớ userprom, ngắt gián tiếp <pre>userprom equ 2000h org 00h ljmp userprom +00H org 03h ljmp userprom+ 03h org 13h</pre>
---	--

<pre> ljmp timer_0_routine org 13h ljmp int_1_routine org 1bh ljmp timer_1_routine ORG USERPROM start: mov ie, #85h mov a, #01 mov p0, a sjmp \$ int_0_routine: mov a, #055h mov p1, a reti timer_0_routine: reti int_1_routine: mov a, #0aah mov p1, a reti timer_1_routine: reti end </pre>	<pre> ljmp userprom + 13h org userprom ljmp start org userprom +03h ljmp int_0_routine org userprom +13h ljmp int_1_routine org userprom start: mov ie, #85h mov a, #01 mov p0, a sjmp \$ int_0_routine: mov a, #055h mov p1, a reti int_1_routine: mov a, #0aah mov p1, a reti end </pre>
---	--

- Các chương trình con liên quan tới thiết bị nằm ở các vùng nhớ riêng dành cho chúng. Trong ví dụ 15.1 chúng ta đã dùng vi mạch giải mã 74138 để giải mã các địa chỉ A13, A14 và A15 cho các vùng nhớ 8 kB. Đây chính là bản đồ nhớ của VDK có 16 đường dây địa chỉ A0–A15 với dung lượng nhớ tổng cộng 64kB. Trong hệ VDK với phân bố nhớ trên, mỗi vùng cho một loại thiết bị như:

- Lỗi ra Y0 (địa chỉ 0000H– 1FFFH): dành cho (hay nối với) nhớ ROM 27C64 để ghi chương trình (đọc bằng tín hiệu /PSEN) và nhớ RAM 6264 ghi dữ liệu (đọc bằng tín hiệu /RD).

- Lỗi ra Y1 (địa chỉ 2000H– 3FFFH): dành cho (hay nối với) nhớ ROM 27C64 để ghi chương trình hay dữ liệu dạng hằng số như các bảng.

- Lỗi ra Y2 (địa chỉ 4000H– 5FFFH): dành cho (hay nối với) chương trình của hiển thị.

- Lỗi ra Y3 (địa chỉ 6000H– 7FFFH): dành cho (hay nối với) chương trình của động cơ bước.

- Lỗi ra Y4 (địa chỉ 8000H– 9FFFH): dành cho (hay nối với) cổng song song mở rộng PPI.

- Lỗi ra Y5 (địa chỉ A000H– 9FFFH): để trống dành cho (hay nối với) thiết bị ghép nối khác.

Lỗi ra Y6 (địa chỉ C000H– 9FFFH): để trống dành cho (hay nối với) thiết bị ghép nối khác.

- Lỗi ra Y7 (địa chỉ E000H– 9FFFH): để trống dành cho (hay nối với) thiết bị ghép nối khác như LCD (địa chỉ E000H– E003H) và bàn phím (địa chỉ F000H).

Với cách nối các vi mạch nhớ và các thiết bị như trên, chương trình của ví dụ 15.3 tại địa chỉ gốc 0000H và chương trình con tại các vùng nhớ của nhớ PROM theo giải mã ở trên.

Ví dụ 15.3 giới thiệu chương trình có chương trình con ở nhiều vùng nhớ khác nhau. Trước mỗi chương trình con, ta phải kê khai vùng nhớ. Có thể dùng cách nhảy gián tiếp, ta gán các nhãn giả của chương trình con, trong đó có lệnh nhảy tới nhãn thật của chương trình con nằm ở các vùng nhớ khác nhau (tương tự vectơ ngắt giả).

– *Chương trình dùng con trỏ địa chỉ DPTR*: địa chỉ này là địa chỉ ô nhớ chương trình, có giá trị từ 0000H tới 65535. Nếu có thiết bị ghép nối, giá trị DPTR cũng chính là các địa chỉ của cổng vào/ra của thiết bị. Do đó, cần xem cách ghép nối để gán giá trị đúng khi lập trình cho PROM.

15.3.2. Ghi và đọc bảng dữ liệu trong ROM

Trong bộ nhớ ROM dữ liệu, các dữ liệu thường được tạo ra bằng chương trình và ghi vào ROM bằng bộ ghi chương trình. Các dữ liệu này được ghi thành bảng, chuỗi ký tự, hình vẽ và được nhập vào từ bàn phím.

Các dữ liệu trong ROM được đọc ra bằng lệnh `MOVC A, @A+DPTR` hay `MOVC A, @A+PC` để đưa ra cổng của VDK. Các cổng của VDK này có thể nối ghép với các LED đơn, LED ma trận, LCD trong các biển quảng cáo hay máy vẽ đồ thị, dao động ký số dùng VDK.

Ví dụ 15.4 giới thiệu chương trình tạo bảng dữ liệu cho một chuỗi ký tự "đây là hình sin" và các dữ liệu của hình sin.

; Ví dụ 15.3: Khung của chương trình ; có chương trình con ở vùng nhớ khác	
<pre>org 00h ljmp start org 100h start: mov ie, #85h mov a, #01 mov p0, a CALL CONVERT_HEX CALL CONVERT_REGISTER CALL DISPLAY CALL GET_KEY CALL DELAY_N sjmp \$ ORG 4100H CONVERT_HEX: MOV A, #1 MOV P0, A RET ORG 8100H CONVERT_REGISTER: MOV A, #2 MOV P0, A RET ORG 6100H</pre>	<pre>DISPLAY: MOV A, #3 MOV P0, A RET ORG 0A100H GET_KEY: MOV A, #4 MOV P0, A RET ORG 0E100H DELAY_N: MOV A, #5 MOV P0, A RET END</pre>

: Ví dụ 15.4: Chương trình tạo và đọc bảng dữ liệu trong ROM	
<pre> org 00h ljmp start org 100h start: mov dptr, #bang1 loop1: mov a, #00h movc a, @a+dptr jz kt1 mov P0, a inc dpl jmp loop1 </pre>	<pre> kt1: mov dptr, #bang2 loop2: mov a, #00h movc a, @a+dptr jz kt2 mov P0, a inc dpl jmp loop2 kt2: sjmp \$ bang1: db 'hình sin', 00h bang2: db 128, 192, 238, 255, 238, 192 db 128, 64, 17, 0, 17, 64, 128 end </pre>

15.3.3. Đọc dữ liệu trong ROM rồi ghi vào RAM trong

Dữ liệu đã ghi vào ROM ngoài. Có thể đọc trực tiếp rồi đưa ra cổng nối với thiết bị ngoài như ví dụ 15.4 ở trên. Đọc và đưa ra theo cách trên tốn nhiều thời gian hơn là đưa trực tiếp từ RAM trong của VDK. Do đó, trong một số ứng dụng yêu cầu thời gian đưa ra ngắn, người ta đọc từ ROM rồi ghi vào các ô nhớ của RAM trong. Từ RAM trong, người ta đưa ra cổng nối với thiết bị ngoài. Ví dụ 15.5 giới thiệu chương trình tạo bảng và ghi vào RAM trong của VDK.

: Ví dụ 15.5: Chương trình chuyển từ nhớ ROM ngoài vào RAM trong	
<pre> org 00h ljmp start org 8100h start: mov dptr, #bang1 mov r1, #00h mov r0, #50h loop1: mov a, #00h movc a, @a+dptr jz kt1 mov @r0, a inc dpl inc r0 inc r1 jmp loop1 kt1: mov dptr, #bang2 loop2: mov a, #00h movc a, @a+dptr jz kt2 mov @r0, a mov r0, #50h inc dpl </pre>	<pre> inc r0 inc r1 jmp loop2 kt2: loop3: mov a, @r0 mov P0, a call delay inc r0 djnz r1, loop3 sjmp \$ bang1: db '##y l# h#nh sin', 00h bang2: db 128, 192, 238, 255, 238, 192 db 128, 64, 17, 0, 17, 64, 128 delay: mov r7, #250 del: mov r6, 200 del1: mov r5, #3 djnz r5, \$ djnz r6, del1 djnz r7, del ret end </pre>

15.4. LẬP TRÌNH CHO VI MẠCH NHỚ RAM NGOÀI

Các vi mạch nhớ RAM ngoài dùng để ghi nhớ dữ liệu tạm thời trong khi chạy chương trình và sẽ mất đi khi tắt nguồn nuôi. Các dữ liệu tạm thời này có thể từ máy vi tính PC tải sang (chương trình, dữ liệu), từ thiết bị ngoài ghép nối với VDK (ADC, bàn phím v.v...). Với RAM ngoài, có thể:

– Ghi (bởi tín hiệu /WR – chân P3.6 của 8051/8052) do lệnh MOVX @Ri, A với i = 0, 1 cho các địa chỉ thấp (A0–A7) của RAM ngoài hay MOVX @DPTR, A cho các địa chỉ (thấp, cao A0–A15) của RAM ngoài.

– Đọc (bởi tín hiệu /RD – chân P3.7 của 8051/8052) do lệnh MOVX A, @Ri với i = 0, 1 cho các địa chỉ thấp của RAM ngoài hay MOVX A, @DPTR cho các địa chỉ (thấp, cao A0–A15) của RAM ngoài.

Việc ghi hay đọc RAM ngoài đều dùng lệnh MOVX giữa thanh ghi chứa A với ô nhớ ngoài với các thanh ghi địa chỉ (R0, R1) hay con trỏ địa chỉ DPTR.

Một số chương trình thông dụng cho bộ nhớ RAM ngoài như xoá vùng nhớ ghi/đọc vùng nhớ, trao đổi dữ liệu cho thiết bị ngoài như máy vi tính PC, ADC, hiển thị, bàn phím v.v...

15.4.1. Xoá vùng nhớ

Lập trình để xoá vùng nhớ tức ghi giá trị #00H vào ô nhớ. Ví dụ 15.6 giới thiệu chương trình xoá 100 byte nhớ từ địa chỉ 1250H.

15.4.2. Ghi vào vùng nhớ

Ví dụ 15.7 giới thiệu chương trình ghi vào vùng nhớ các giá trị tăng dần từ 0 vào các địa chỉ từ 1250H tới 1290H.

Ví dụ 15.6: Chương trình xoá vùng nhớ	Ví dụ 15.7: Chương trình ghi vùng nhớ
<pre>ORG 00H LJMP START ORG 100H START: PUSH ACC PUSH DPH PUSH DPL MOV A, #00H MOV R0, #100 MOV DPTR, #1200 LAP: MOVX @DPTR, A INC DPL DJNZ R0, LAP POP DPL POP DPH POP ACC SJMP \$ END</pre>	<pre>ORG 00H LJMP START ORG 100H START: PUSH ACC PUSH DPH PUSH DPL MOV R0, #00H MOV DPTR, #1250H LAP: MOV A, R0 MOVX @DPTR, A INC DPL INC R0 MOV A, DPL CJNE A, #90H, LAP POP DPL POP DPH POP ACC SJMP \$ END</pre>

Chương trình trên chỉ có DPL thay đổi từ 50H lên 90H nên chỉ so sánh DPL với 90H. Nếu địa chỉ nhớ DPTR có cả DPH và DPL thay đổi, ta phải so sánh DPH trước, rồi so sánh DPL sau.

15.4.3. Đọc dữ liệu vào rồi ghi vào vùng nhớ đệm

Dữ liệu vào ở đây có thể từ máy vi tính PC, từ vi mạch ADC, từ bàn phím v.v... nên dạng dữ liệu là khác nhau. Có thể ghi nhớ nguyên dạng dữ liệu vào, nhưng dạng BCD hay HEX tốn nhiều ô nhớ, ta phải biến đổi 2 byte dạng HEX thành 1 byte nhị phân để tiết kiệm địa chỉ nhớ (xem biến đổi mã ở tập I hay chương 14 về bàn phím).

Ví dụ 15.8 giới thiệu chương trình đọc và ghi dữ liệu từ ADC có địa chỉ ADC0.

; Ví dụ 15.8: Chương trình đọc dữ liệu từ ADC và ghi vào ô nhớ đệm HBYTE, LBYTE

```

ORG 00H
LJMP START
ORG 30H
READ_ADC EQU 0F10H
LBYTE DS 0
HBYTE DS 0
ORG 100H
START: MOV DPTR, #READ_ADC      ; gán địa chỉ ADC cho DPTR
      MOVX A, @DPTR             ; đọc dữ liệu vào A
      MOV LBYTE, A              ; gửi dữ liệu từ A vào LBYTE
      MOV DPTR, #READ_ADC      ; gán địa chỉ ADC cho DPTR
      MOVX A, @DPTR             ; đọc dữ liệu vào A
      MOV HBYTE, A              ; gửi dữ liệu từ A vào HBYTE
      SJMP $
      END

```

15.5. NHỚ EEPROM NỐI TIẾP

15.5.1. Nhớ EEPROM nối tiếp trao đổi tin theo hai đường dây của I²C

Ở trên, ta đã xét các IC nhớ chương trình xoá được bằng điện (EEPROM) với cách đưa địa chỉ A0–A1 và dữ liệu D0–D7 song song. Ở đây, chúng ta xét các IC nhớ EEPROM đưa địa chỉ và dữ liệu nối tiếp và để tiết kiệm đường dây tín hiệu.

1. Đặc điểm chung của EEPROM nối tiếp

Các IC nhớ nối tiếp này có các đặc điểm chung như sau:

- Có hai đường dây trao đổi tin nối tiếp:
- + Đường dây SDC: dùng để VĐK đưa tín hiệu nhịp CLK.
- + Đường dây SDA: dùng để trao đổi dữ liệu (vào và ra) giữa VĐK và IC nhớ.

Do đó, các IC nhớ loại này còn được gọi là IC nhớ I²C với ý nghĩa là trao đổi tin nối tiếp theo 2 dây dẫn tín hiệu (xem chương 17 Mở rộng cổng nối tiếp...).

- Trao đổi tin nối tiếp theo thủ tục I²C:

Muốn trao đổi tin loại này, VDK phải nối ghép với cổng I²C và tuân theo thủ tục trao đổi tin loại này.

– Có thể đưa thêm đường dây địa chỉ ngoài vào: Các lối vào A0, A1, A2 có thể đưa vào các địa chỉ chọn vùng nhớ cho IC. Với IC của hãng THOMPSON có lối vào E0, E1, E2 dùng để cho phép (Enable) xâm nhập các vùng nhớ khác nhau.

– Có thể có thêm đường dây bảo vệ dữ liệu WP (Write Protect) hay /WC (Write Control): Để đưa bit bảo vệ bộ nhớ, chống ghi đè lên dữ liệu đã ghi.

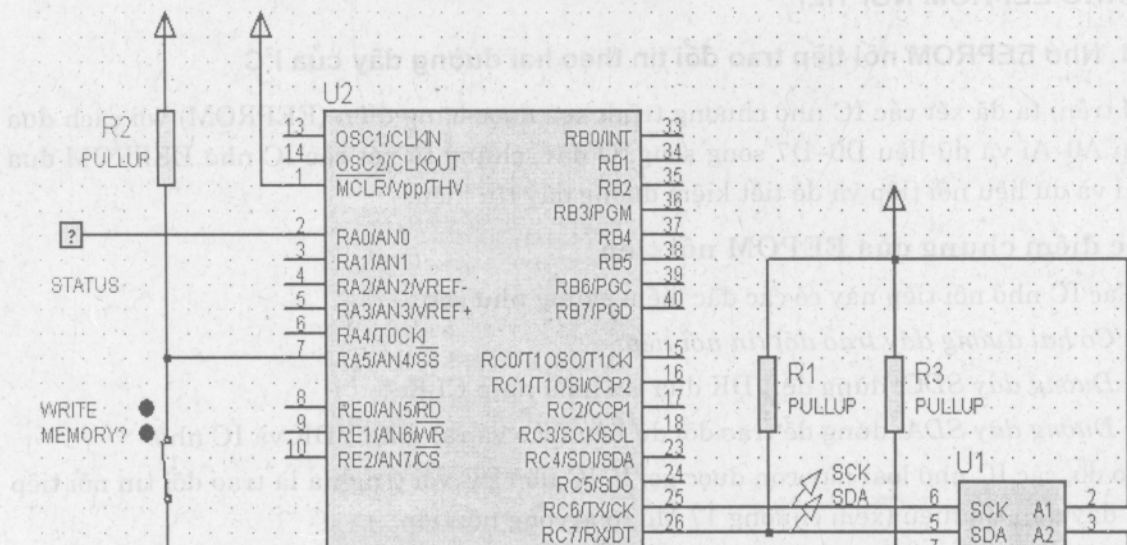
– Có rất ít chân tín hiệu, kích thước nhỏ, dung lượng nhớ lớn: Với thủ tục trao đổi tin I²C và không có địa chỉ và dữ liệu song song, các IC nhớ này chỉ có 4, 6, 8 chân cắm nhưng dung lượng nhớ có thể từ 128 bit tới mega bit với kích thước rất nhỏ (nhỏ hơn IC loại 74xx cỡ mật độ trung bình) vì các IC, I²C này có mật độ rất lớn (VLI) nhờ kỹ thuật tiên tiến trong chế tạo vi mạch bán dẫn. Trong chương trình mô phỏng PROTEUS giới thiệu một số IC nhớ loại trên.

2. Sơ đồ chân, sơ đồ khối và sơ đồ ghép nối VDK

Loại chân đơn giản nhất chỉ có 5 chân, trong đó có chân không nối (NC). Các IC khác có thêm các chân địa chỉ (từ A0 tới A3) hay chân cho phép thâm nhập vùng nhớ E1 tới E3.

Một IC nhớ nối tiếp có các khối sau:

- Điều khiển Logic Start/Stop..
- Logic điều khiển nhớ Memory Control Logic.
- Thanh ghi địa chỉ tổ và bộ so sánh.
- Đếm địa chỉ lời.
- Giải mã địa chỉ nhớ XDEC, YDEC.
- Ma trận nhớ EEPROM Array.
- Khối điều khiển quét ngang và dọc cho ma trận HV Generator.
- Khối điều khiển ghi/đọc và khuếch đại chiều Sense AMP R/W Control.



Hình 15.9. Sơ đồ ghép nối VDK với IC nhớ nối tiếp

Vi mạch nhớ được ghép nối với VDK theo nguyên tắc sau:

- Các lối vào SCK và SDA được ghép nối trực tiếp với hai bit bất kỳ của một cổng ra nào đó của VDK. Với sơ đồ của hình 15.9, hai chân SCK và SDA nối với hai lối ra TxCK và TxDT của VDK.

Lối vào /WP nối với chuyển mạch để nối với +5 V khi trao đổi tin, với 0 V khi cấm ghi hay bảo vệ tin trong khối nhớ.

Các lối vào A1, A2, A3 được nối với chân của VDK, nối với nguồn Vcc hay đất Vss hoặc SDA.

Hình 15.9 giới thiệu sơ đồ VDK PIC 16F877 mắc với một IC nhớ 24C04A.

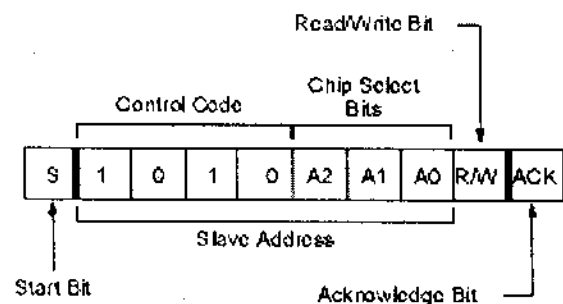
3. Thủ tục và dạng tin trao đổi

Trao đổi tin với IC nối tiếp này theo thủ tục trao đổi tin nối tiếp I²C (xem chương 17) trong đó:

- Đường dây SCK đưa tín hiệu nhịp của tốc độ trao đổi tin nối tiếp (thường với tốc độ 9600 baud).

- Đường dây SDA đưa dữ liệu với khung tin 8 bit và các bit Start/Stop của trao đổi tin không đồng bộ.

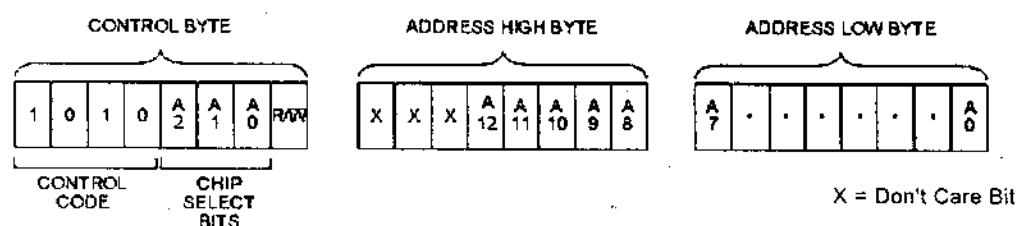
- Thứ tự trao đổi tin từ VDK là byte điều khiển (hình 15.10), địa chỉ tổ (IC nhớ) (hình 15.11), một byte dữ liệu (hình 15.12), nhiều byte dữ liệu (hình 15.13).



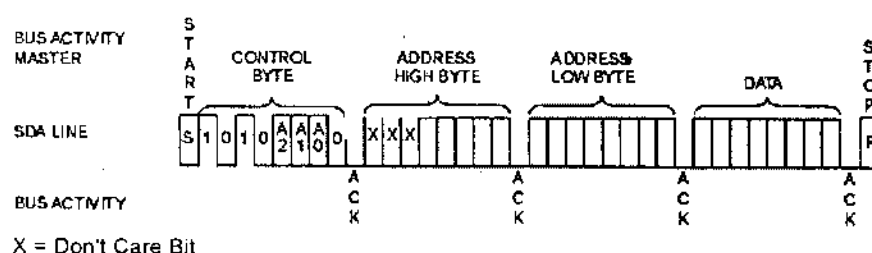
Hình 15.10. Byte tin về điều khiển

Byte điều khiển được VDK (thiết bị chủ phát trước tiên cho các thiết bị tớ nhận, trong đó:

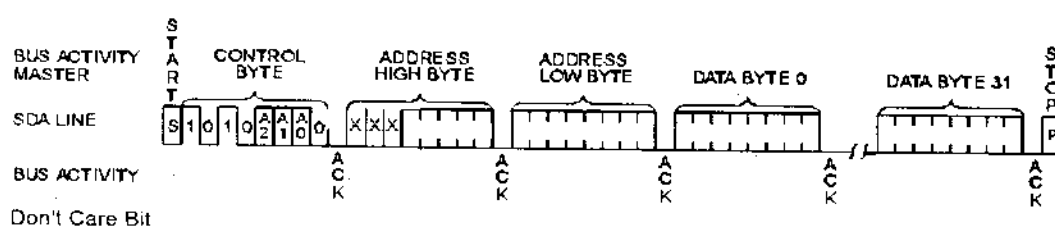
- + Bit S là bit Start của khung tin nối tiếp.
- + Bit 1 010 chỉ lệnh điều khiển ghi địa chỉ cho thiết bị tớ.
- + Các bit A2A1A0 chỉ chọn các chip nhớ khác nhau nếu có.
- + Bit R/W điều khiển (ghi W = 1) hay đọc (R = 0).
- + Bit ACK (xác nhận) dùng để thiết bị tớ (IC nhớ) trả lời đã nhận lệnh (ACK = 1).



Hình 15.11. Byte tin về địa chỉ



Hình 15.12. Byte tin về dữ liệu ghi



Hình 15.13. Chuỗi byte dữ liệu ghi

4. Lập trình trao đổi tin

Tin trao đổi nối tiếp ra hay vào VDK trên một chân nối SDA là các bit 0 hay 1 nối tiếp có dạng như các hình 15.11 tới 15.13.

Do đó, lập trình cho VDK phải tuân theo các khung tin này và viết dưới dạng chương trình con hay lệnh gộp (macro).

Để tìm hiểu về viết chương trình cho các IC nhớ nối tiếp, ta có thể tham khảo các chương trình sau:

- Chương trình PICI2I2C.ASM trao đổi tin trên I²C trong thư mục Labcenter Electronic/PROTEUS/Sample/PICI2CI2CE618EEPROM viết cho VDK PIC của Microchip.
- Chương trình FL51XINC.ASM trao đổi tin trên I²C trong thư mục Labcenter Elelectronix/PROTEUS/Sample/PICI2CI2CE618EEPROM viết cho VDK PIC của Microchip.
- Chương trình I2CMEM1.ASM trao đổi tin trên I²C trong thư mục Labcenter Electronic/PROTEUS/Sample/PIC Serial Memory Example viết cho VDK PIC của Microchip.
- Chương trình viết cho lệnh gộp với chương trình dịch MC48 của VDK 8748 và EEPROM 8542.

15.5.2. Nhớ EEPROM nối tiếp trao đổi tin theo một đường dây

Vì mạch nhớ X84641 của hãng Xicor là một vi mạch nhớ có thể trao đổi tin vào/ra, theo một đường dây nối trực tiếp với một bit cổng song song của VDK.

Hình 15.14 giới thiệu sơ đồ chân, sơ đồ khối và chân cắm của X 84641.

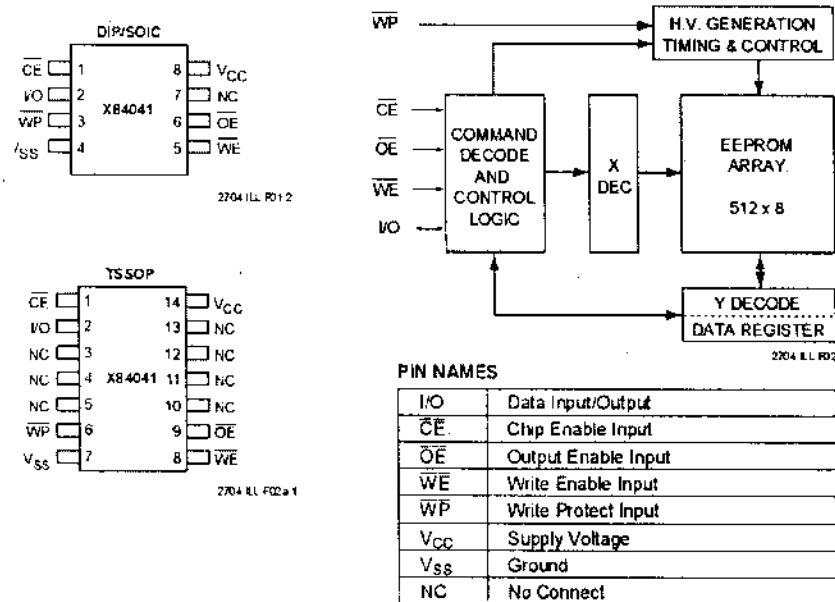
IC X 84641 hay 84041 có thể có 8 hay 14 chân như hình 15.14.

Từ bảng các chân cắm, ta thấy:

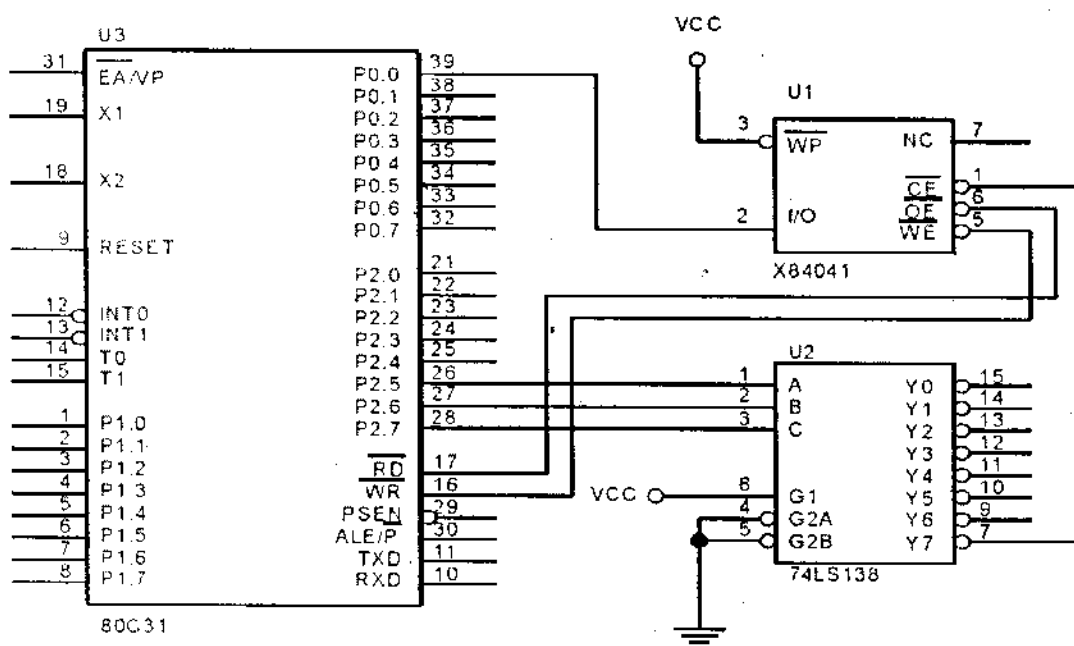
- Chỉ ghi tin nối tiếp từ đường dây I/O vào IC khi đã chọn vi mạch (/CE = 0), cho phép ghi (/WE = 0) và bảo vệ ghi ở logic 1 (/WP = 1).
- Chỉ đọc tin nối tiếp từ IC ra đường dây I/O khi đã chọn vi mạch (/CE = 0), cho phép đưa ra (/OE = 0).

Sơ đồ khối của IC trên tương tự sơ đồ khối của IC 24AA của hãng Microchip, gồm các khối giải mã lệnh và logic điều khiển, giải mã địa chỉ theo hàng và cột cho ma trận nhớ, ma trận nhớ và khối phát và điều khiển ngang và dọc cho ma trận.

Hình 15.15 giới thiệu sơ đồ ghép nối VDK 8051 với IC nhớ trên với một đường dây dữ liệu I/O vào bit P0.0



Hình 15.14. Sơ đồ chân, sơ đồ khối và chân cắm của X84641 hay X84041



Hình 15.15. Sơ đồ mắc X84041 với VDK 80C31

15.6. NHỚ NV-RAM

Các vi mạch nhớ NV-RAM hay NORAM là các vi mạch nhớ không mất nội dung khi mất nguồn điện nuôi (không bay hơi- NV-No Volatile) và trao đổi tin nối tiếp.

Có hai loại NV-RAM:

- Loại nhớ RAM và có nguồn pin ngay trong IC nhớ.

– Loại có đồng thời nhớ RAM và ROM.
 Chương trình trao đổi tin cho sơ đồ hình 15.15 có thể tìm thấy trong AN107 của hãng XICOR.

15.6.1. Nhớ NV-RAM nối tiếp không có nguồn nuôi

1. Đặc điểm chung

Nhớ NV-RAM là loại nhớ không mất nội dung nhớ (Non Volatile – không bay hơi) khi mất nguồn điện nuôi. Về đặc điểm này, nhớ NV-RAM giống nhớ ROM nhưng khác là ghi và đọc bởi điện áp nuôi bình thường (3V–5V), nhưng khác là không cần nạp như PROM.

Có đặc điểm trên là vì:

– Có thể có sẵn nguồn nuôi bằng pin Lithilium đặt trong IC và có mạch kiểm tra nguồn nuôi ngoài, khi điện thế nuôi ngoài sụt tới một mức nào đó, mạch sẽ điều khiển nối nguồn nuôi bên trong.

– Có thể có hai bộ nhớ RAM và ROM chồng lên nhau, cùng một địa chỉ để ghi vào RAM rồi ghi lại (SAVE) vào ROM. Khi đọc dữ liệu ra, sẽ đọc từ ROM (RECOVER) để ghi vào RAM rồi đọc ra đường dây dữ liệu ngoài. Vì mạch loại này cần có thêm một số lệnh (xem bảng lệnh ở hình 15.16).

Có hai loại NV-RAM:

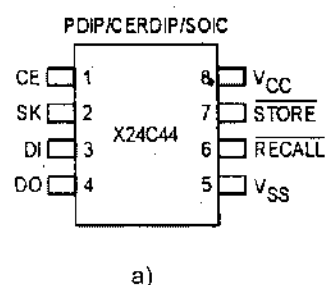
– NV-RAM với các đường dây dữ liệu song song như các vi mạch loại M48 như M48Z08, M48Z18, M48Z02, M48S12, M48Z512, v.v...

– NV-RAM nối tiếp như các vi mạch loại X24C, ta sẽ xét vi mạch X24C44.

2. Các sơ đồ và hoạt động

Sơ đồ chân của X24C44 có các chân (hình 15.16) như sau:

- Chân chọn vi mạch (/CE).
- Chân đưa tín hiệu nhịp (SCK).
- Hai chân đưa dữ liệu vào (DI) và ra (DO).
- Hai chân đưa lệnh tích lũy (/STORE) và gọi lại dữ liệu (/RECALL).
- Hai chân nguồn nuôi +5V (Vcc) và đất 0V (Vss).



Symbol	Description
CE	Chip Enable
SK	Serial Clock
DI	Serial Data In
DO	Serial Data Out
RECALL	Recall Input
STORE	Store Input
V _{cc}	+5V
V _{ss}	Ground

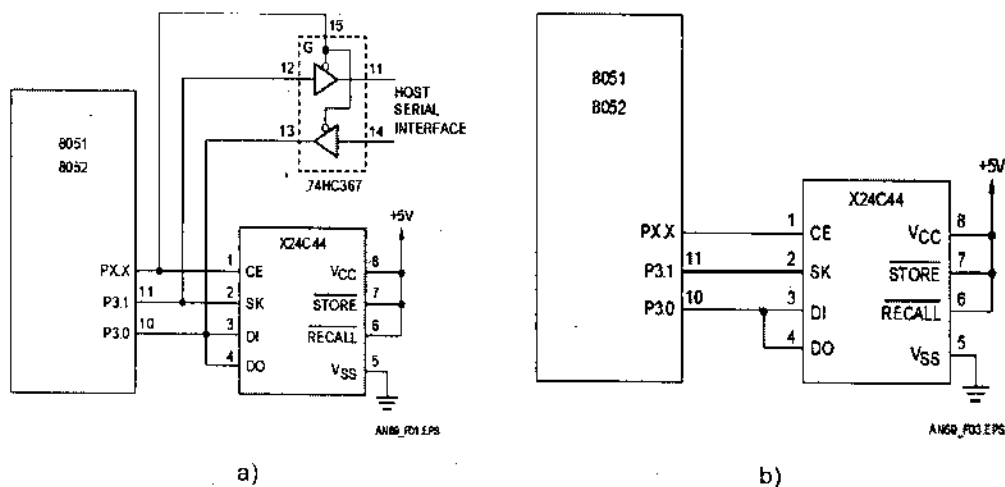
b)

Hình 15.16. Sơ đồ khối của X24C44

Sơ đồ có các khối sau:

- Thanh ghi lệnh.
- Giải mã lệnh.
- Giải mã hàng và cột.

- Ma trận nhớ SRAM tĩnh.
 - Bộ đếm 4 bit.
 - Logic điều khiển để thực hiện hai lệnh lưu trữ (STORE) và gọi lại (RECALL).
- Hình 15.17 giới thiệu cách ghép nối vi mạch nhớ X24C44 với VDK 8051/8052.



Hình 15.17. Sơ đồ nối X24C44 với VDK theo hai (a) và một dây tín hiệu (b)

Có hai cách nối ghép:

- *Nối theo hai đường dây dữ liệu:* Đường dây DI dữ liệu vào DI và đường dây dữ liệu ra DO (hình 15.17a). Trên hình vẽ, còn có đường dây ba trạng thái với vi mạch ba trạng thái và điều khiển vào hay ra bởi tín hiệu điều khiển.
- *Nối theo một đường dây dữ liệu vào/ra* (hình 15.17b). Loại nối ghép này tiết kiệm được dây dẫn tín hiệu vì là dây dẫn hai chiều.

3. Chương trình

Các chương trình con đọc (READ), ghi (WRITE), lưu trữ (SAVE) và gọi lại (RECOVER) sẽ thực hiện lệnh STORE và RECALL như ở ví dụ 15.9.

Ví dụ 15.9: Các chương trình con Ghi, Đọc, lưu giữ và gọi lại cho X24C44	
<pre> ORG 00H LJMP START R_DATA EQU 87H RECALL EQU 85H WRITE EQU 84H W_DATA EQU 83H SLEEP EQU 82H STORE EQU 81H MON51 EQU 00BCH LENTGTH EQU 32H DIN BIT P1.2 ENABLE BIT P1.1 CLK BIT P1.0 ORG 60H NVRAM DS LENGTH ORG 8000H START: </pre>	<pre> SETB ENABLE ACALL W_BYTE CLR ENABLE ACALL KETTHUC RET ; RECOVER 8051 RAM LOCATION 60H-7FH FROM X24C44 RECOVER: MOV R0, #NVRAM CLR ENABLE MOV A, #RECALL SETB ENABLE ACALL W_BYTE CLR ENABLE MOV R7, #0 AGAIN2: MOV A, R7 RL A RL A </pre>

WX2444: ACALL SAVE RX2444: ACALL RECOVER LJMP MON51 : SAVE 8031 RAM LOCATION 60H - 7FH : IN X24C44 NVRAM SAVE: MOV R0, #NVRAM CLR ENABLE MOV A, #RECALL SETB ENABLE ACALL W_BYTE CLR ENABLE MOV A, #WRITE SETB ENABLE ACALL W_BYTE CLR ENABLE MOV R7, #0 AGAIN1: MOV A, R7 RL A RL A RL A ORL A, #W_DATA SETB ENABLE ACALL W_BYTE MOV R5, #2 LOOP1: MOV A, @R0 INC R0 ACALL W_BYTE DJNZ R5, LOOP1 CLR ENABLE INC R7 CJNE R7, #16, AGAIN MOV A, #STORE	RL A ORL A, #R_DATA SETB ENABLE ACALL W_BYTE MOV R5, #2 LOOP2: ACALL R_BYTE MOV @R0, A INC R0 DJNZ R5, LOOP2 CLR ENABLE INC R7 CJNE R7, #16, AGAIN CALL KETTHUC RET KETTHUC: MOV A, #SLEEP SETB ENABLE ACALL W_BYTE CLR ENABLE RET W_BYTE: MOV R6, #8 AGAIN3: MOV C, DIN SETB CLOCK CLR CLOCK DJNZ R6, AGAIN3 RET R_BYTE: MOV R6, #8 AGAIN4: MOV DIN, C RLC A SETB CLOCK CLR CLOCK DJNZ R6, AGAIN4 RET END
---	--

15.6.2. Nhớ NV-RAM song song có nguồn nuôi

1. Đặc điểm chung

Loại nhớ NV-RAM này có cấu trúc tương tự NV-RAM nối tiếp đã trình bày ở trên. Chỉ có điều khác là các địa chỉ nhớ A_i-A₀ đưa vào song song. Do đó quá trình ghi/đọc tin đơn giản hơn, thời gian nhanh hơn, nhưng vi mạch có thể tích lớn hơn.

Cũng có hai loại NV-RAM song song là loại không có nguồn nuôi hay không có pin Lithilium ở bên trong (có thêm EEPROM) và loại có pin. Ở đây chúng ta chỉ xét loại có pin ở bên trong, để suy ra loại NV-RAM nối tiếp có pin ở bên trong.

Có rất nhiều hãng chế tạo loại nhớ NV-RAM song song này, trong đó có Xicor, Cata, Dallas Maxim, Texass v.v...

Chúng ta xét loại M48 của hãng Motorola.

NV = RAM này có nhiều loại dung lượng nhớ khác nhau như 8Kx8 bit, 16Kx8 bit, 32Kx8 bit, 128Kx8 bit, 512Kx8 bit.

Tính năng cơ bản của NV = RAM M48z02/12 như bảng 15.2:

Bảng 15.2. Các tín hiệu của M48z02/12

AO-A10	Address Inputs
DOK07	Data Inputs I Outputs
	Chip Enable
	Output Enable
W	WRITE Enable
Vcc	Supply Voltage
USS	Ground

2. Các sơ đồ

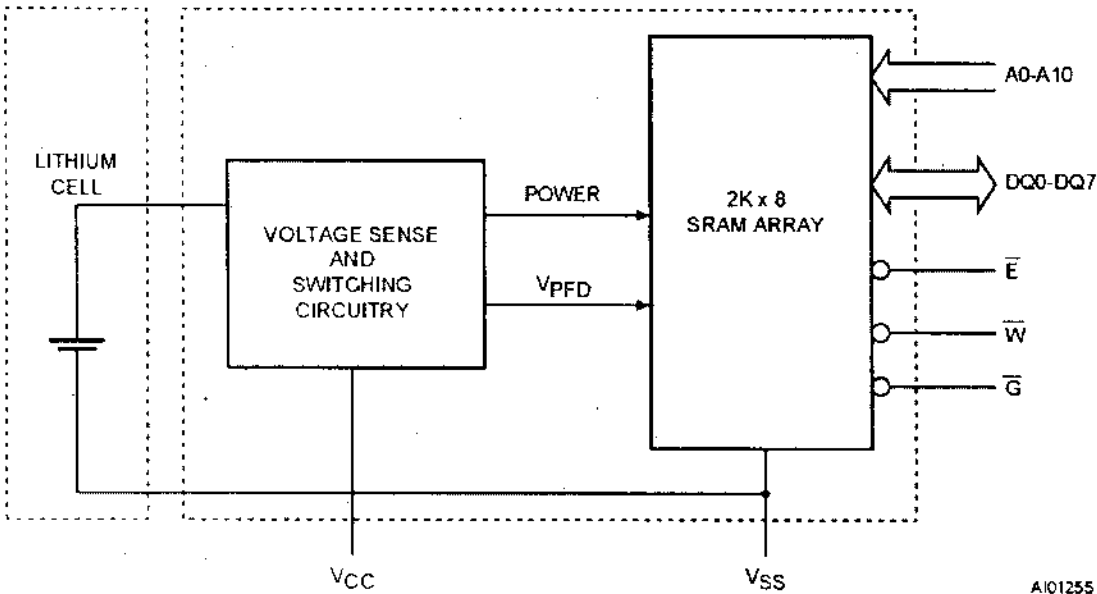
Các tín hiệu của M48z02/12 như ở bảng 15.2.

Các tín hiệu như sau:

- A0-A10: địa chỉ song song.
- D00-D07: các dữ liệu ra hay vào.
- /E: chọn chip.
- /G: cho phép dữ liệu ra.
- /W: cho phép ghi dữ liệu vào.
- Vcc: nguồn nuôi +5V.
- Vss: đất.

Sơ đồ khối của M48z02/12 như ở hình 15.18, gồm các khối sau:

- Nguồn pin Lithilium.
- Phát hiện thế nuôi ngoài thấp và điều khiển đóng mạch nguồn pin.
- Ma trận nhớ SRAM.



Hình 15.18. Sơ đồ khối của NV-RAM M48z02/12

15.7. BỘ NHỚ CỦA VDK AT89C51RD2/ED2

VDK AT89C51RD2/ED2 của hãng ATMEL nhưng chế tạo theo hệ lệnh của họ VDK MCS-51 và có các bộ nhớ tiên tiến như nhớ Flash và NV-RAM.

15.7.1. Bộ nhớ XRAM mở rộng

VDK AT89C51RD2/ED2 cung cấp thêm không gian nhớ RAM trên chip để tăng sự quản lý thông số dữ liệu và ngôn ngữ bậc cao. RAM này có thể lên tới 1792 byte, được chia thành 4 vùng nhớ:

- 128 byte thấp (địa chỉ 00H tới 7FH), ghi địa chỉ trực tiếp và gián tiếp.
- 128 byte cao hơn (địa chỉ 80H tới FFH), chỉ ghi địa chỉ gián tiếp.
- Các thanh ghi nhớ đặc biệt SFR (địa chỉ 80H tới FFH), chỉ ghi địa chỉ trực tiếp.

Các byte, truy cập gián tiếp bằng lệnh MOVX và được xoá với bit EXTRAM trong thanh ghi AUXR.

15.7.2. Bộ nhớ dữ liệu EEPROM

Chỉ có chip AT89C51ED2 có tính chất này. Bộ nhớ này trên chip 2kB được đặt tại các địa chỉ từ 0000H tới 07FFH của không gian nhớ XRAM/ERRAM và được lựa chọn bằng thiết lập các bit điều khiển trong thanh ghi EECON. Được truy cập bởi lệnh MOX.

15.7.3. Bộ nhớ FLASH/EEPROM

Bộ nhớ này được lập trình ngay trên mạch với điện áp nuôi thấp Vcc (5V). Bộ nhớ này có 64kB nhớ chương trình, được tổ chức tách biệt thành 512 trang (mỗi trang 128 byte). Bộ nhớ này được lập trình ngay trên hệ thống với trao đổi tin nối tiếp hoặc song song. Mạch nạp (bootloader) nối tiếp mặc định cho phép lập trình ngay trên hệ thống.

Bộ nhớ này có ba bit khoá LB0, LB1 và LB2 để tăng khả năng bảo mật cho chương trình.

Thủ tục ISP (In-System Programming- thủ tục lập trình trên hệ thống) như sau:

- Dữ liệu 8 ký tự, không kiểm tra tính chẵn/lẻ, 2 bit dừng, tốc độ baud tự động bởi bộ nạp bootloader.
- Khung tin theo dạng HEX của Intel (1 byte đánh dấu, 1 byte Recien, 2 byte độ lệch, 1 byte loại bản ghi, n byte dữ liệu hay thông tin, 1 byte kiểm tra tổng).

CÂU HỎI, BÀI TẬP VÀ THỰC HÀNH

I. Câu hỏi

1. Kể các đặc điểm chính của các loại vi mạch nhớ cơ bản.
2. Nêu các ứng dụng của các loại nhớ PROM, RAM.
3. So sánh hai loại nhớ ghi/đọc dữ liệu song song và nối tiếp.
4. So sánh hai loại nhớ RAM, ROM và NV-RAM, nêu các ứng dụng của chúng.
5. Cách ghép nối các IC nhớ với VDK 8051/8052.
6. Các loại giải mã địa chỉ, các loại vi mạch giải mã địa chỉ, vai trò của vi mạch giải mã địa chỉ trong hệ VDK.

II. Bài tập

1. Quan sát và liệt kê cách ghép nối bộ nhớ ngoài và cách giải mã trong các sơ đồ sau có trong PROTEUS:

- C51 Calculator.
- 8051 Memory.
- AVR external Memory.
- PIC I2CE518EEPROM.

2. Quan sát và liệt kê các IC nhớ và địa chỉ giải mã của chúng trong sơ đồ EVB.

3. Quan sát và liệt kê các IC nhớ nối tiếp trao đổi tin theo ba dây (SPI), hai dây (I2C) và một dây.

4. Thiết kế một bản mạch sử dụng các bộ nhớ khác nhau (PROM, RAM, nối tiếp và NV-RAM).

III. Thực hành

1. Cho chạy mô phỏng các sơ đồ có nhớ trong PROTEUS.

2. Nghiên cứu cấu trúc và vẽ giải thuật của các chương trình trên, nếu có thể.

CHƯƠNG

16

GHÉP NỐI MỞ RỘNG CỔNG SONG SONG CHO VI ĐIỀU KHIỂN

Mục đích của chương:

- Hiểu rõ yêu cầu ghép nối cổng song song mở rộng cho VDK.
- Hiểu rõ cách ghép nối cổng song song mở rộng cho VDK.
- Hiểu rõ lập trình cho cổng song song mở rộng của VDK.

16.1. YÊU CẦU VÀ CÁCH MỞ RỘNG CỔNG CHO VI ĐIỀU KHIỂN

16.1.1. Yêu cầu mở rộng cổng cho vi điều khiển (VDK)

Vi điều khiển loại thông dụng 40 chân của họ MCS-51 chỉ có 4 cổng song song (trong đó cổng P3 phải dành cho cổng nối tiếp, các tín hiệu ngắt ngoài /INT0, /INT1, hai lối vào đếm T0, T1, cũng như hai lối ra đưa lệnh ghi /WR và đọc /RD) và TxD và RxD của cổng nối tiếp.

Nếu VDK trên được dùng trong một hệ VDK mở rộng, số cổng song song trên chỉ đủ cho bộ nhớ ngoài (cần hai cổng P0 và P2 để đưa địa chỉ nhớ A15-A0, và một cổng cho bàn phím (cổng P1). Như vậy, hệ VDK trên không còn cổng để trao đổi tin với các thiết bị ngoài song song khác như các bộ hiển thị (LED 7 đoạn, LCD), động cơ bước, động cơ một chiều v.v... Do đó cần phải ghép nối mở rộng cổng song song cho VDK để VDK có thể mắc nối với các thiết bị ngoài vừa kể trên, đồng thời có thể mắc thêm các thiết bị ngoài trao đổi tin song song khác.

Vấn đề mở rộng cổng song song này càng cấp thiết cho các VDK dùng trong công nghiệp có ít chân cắm, nên có ít lối ra song song như các VDK PIC của hãng MICROCHIP (18, 28 chân) hay AT89C1051(1KB PROM) và AT89C2051 (2KB PROM) của hãng ATMEL.

16.1.2. Các phương án mở rộng cổng song song

Để mở rộng cổng song song, tức có nhiều cổng song song để mắc VDK với nhiều thiết bị ngoài như khối nhớ (cần 2 cổng địa chỉ và 1 cổng dữ liệu dẫn kênh trên P0, P2), bàn phím (cần một cổng dữ liệu), LED 7 đoạn (cần 1 cổng dữ liệu), màn hình thể lỏng LCD (cần 2 cổng cho dữ liệu và điều khiển) và điều khiển số cho động cơ bước, rơ le điện tử (cần 1 cổng dữ liệu).

Để mở rộng cổng song song ta có thể chọn một trong các phương án sau:

– *Dùng vi mạch không chương trình hoá:* như các thanh ghi. Phương án này rẻ tiền hơn nhưng phức tạp về lập trình điều khiển.

– *Dùng vi mạch cổng song song 8255 chương trình hoá:* vi mạch này có một cổng vào 8 bit (D0-D7) nối với VDK, nhưng có 3 cổng ra hay vào 8 bit (PA, PB, PC). Vi mạch này sử dụng đơn giản, có thể lập trình hoá cho chiều vào hay ra của dữ liệu (cho cả 3 cổng PA, PB và PC ở chế độ 0), cho vào/ra thuận nghịch (cổng PA ở chế độ 2) và ở chế độ hội thoại (cho

cổng PA, PB ở chế độ 1 và dùng cổng PC làm tín hiệu hội thoại). Hơn nữa, vi mạch này còn có chế độ đưa lệnh cho từng bit trên một đường dây dữ liệu của cổng PC.

Do đó, vi mạch 8255 này thường được dùng làm cổng mở rộng cho VDK, một đường dây dữ liệu của một cổng song song của VDK nối với đường dây dữ liệu D0–D7 của 8255 có thể mở rộng thành 3 cổng song song, mỗi cổng có 8 bit, tổng cộng thành 24 bit.

Khi ghép nối với thiết bị ngoài, đôi khi cần các dòng tải lớn hay mức điện áp khác với mức TTL (5V và 0V), ta cần mắc thêm các vi mạch tăng công suất và chuyển mức.

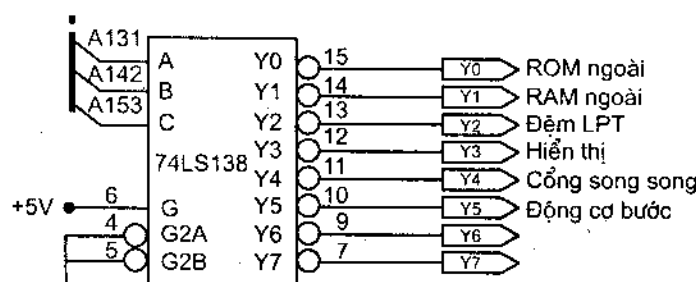
16.1.3. Giải mã địa chỉ cho cổng song song

Cổng song song cũng như vi mạch nhớ, cần tín hiệu chọn vi mạch (CS—chip select) để cho phép hoạt động. Ở phần giải mã địa chỉ cho vi mạch nhớ, chúng ta đã đề cập đến cách giải mã để chọn địa chỉ đầu tiên của vùng nhớ của thiết bị, trong đó có cổng song song. Nhưng chúng ta cần để ý tới những đặc điểm khác nhau của vi mạch nhớ và cổng song song là:

- Vi mạch nhớ có một vùng địa chỉ liên tục từ địa chỉ thấp nhất hay địa chỉ gốc (với $A_0 - A_i = 0$) tới địa chỉ cao nhất ($A_0 - A_i = 1$) và địa chỉ cơ sở là địa chỉ vùng nhớ.
- Cổng song song có một địa chỉ duy nhất hoặc một dải hẹp giá trị (3, 4) địa chỉ và địa chỉ thấp nhất (địa chỉ cơ sở) là địa chỉ gốc của vùng nhớ dành cho chúng. Ví dụ, cổng dùng vi mạch thanh ghi đếm đường dây 74LS245 chỉ có một địa chỉ, trong khi 82C55A có ba địa chỉ liên tiếp kể từ địa chỉ cơ sở (hay gốc) cho ba cổng PA, PB và PC.
- Trong một hệ vi xử lý hay vi điều khiển, mỗi thiết bị chỉ có một địa chỉ duy nhất gán cho nó, tránh sự trùng lặp, nhất là giữa cổng và bộ nhớ có nhiều địa chỉ từ thấp tới cao. Do đó bộ giải mã địa chỉ phải có mặt của tất cả các đường dây địa chỉ của hệ vi xử lý hay VDK. Nếu khác đi, sẽ có sự trùng lặp về địa chỉ thấp giữa cổng hay thiết bị khác và bộ nhớ.

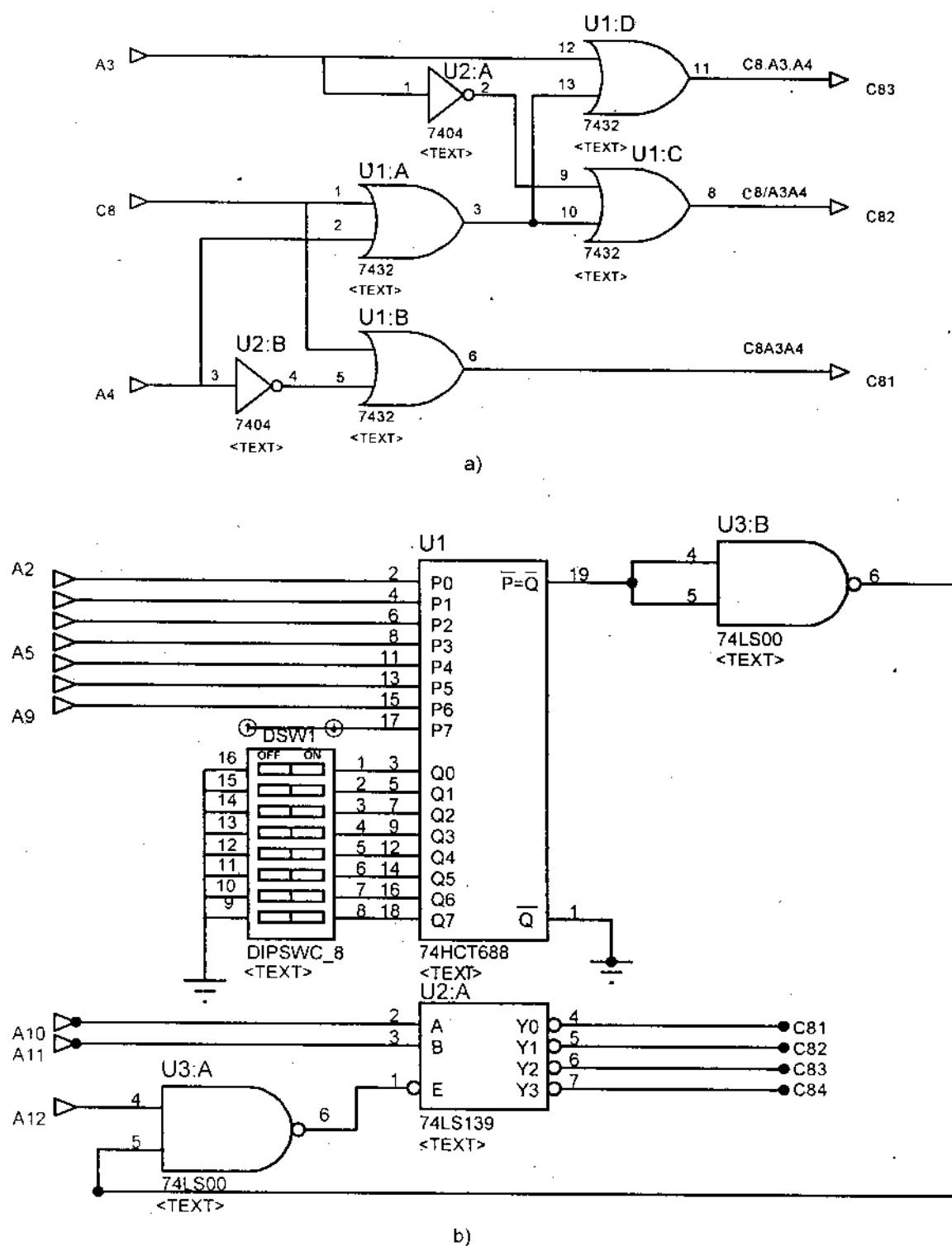
Do đó, cổng song song nói riêng, cổng các thanh ghi của các thiết bị ghép nối nói chung cần có sự giải mã tiếp theo cho các địa chỉ cụ thể trong vùng nhớ dành cho chúng. Với khối nhớ, việc giải mã tiếp theo cho từng ô nhớ được tiến hành ở bên trong vi mạch nhớ. Ví dụ, cổng song song LPT của máy vi tính có nhiều địa chỉ là 378H (LPT1), 3F8 (LPT2) v.v...

Hình 16.1 giới thiệu bộ giải mã các dải địa chỉ cho các vùng nhớ, đếm LPT, hiển thị, cổng song song của một hệ VDK điển hình.



Hình 16.1. Sơ đồ giải mã cho hệ VDK

Hình 16.2 giới thiệu hai cách giải mã các dải địa chỉ cho các vùng nhớ của các thanh ghi đếm hay các cổng của thiết bị ngoài.



Hình 16.2. Giải mã vùng địa chỉ con kiểu mạch logic (a) và so sánh với chuyển mạch (b)

Hình 16.2a cho các vùng địa chỉ cố định sau:

– CS1 = A3/A4

– CS2 = /A3.A4

– CS3 = A3.A4

Hình 16.2b kiểu so sánh có các đặc điểm:

– Có tất cả các địa chỉ từ A2 tới A12.

– CS là lối ra của vùng nhớ lớn.

– Các địa chỉ từ A2 tới A9 có thể cho giá trị Ai hay /Ai phụ thuộc chuyển mạch tương ứng với Ai đó không hay được nối với đất.

16.2. MỞ RỘNG CỔNG SONG SONG DÙNG THANH GHI ĐỆM

16.2.1. Mở rộng cổng ra song song dùng vi mạch chốt dữ liệu

Các thanh ghi chốt dữ liệu vào của cổng mở rộng và bộ giải mã dùng để điều khiển chọn một trong các thanh ghi (đưa vào lối vào /CS (chọn chip)).

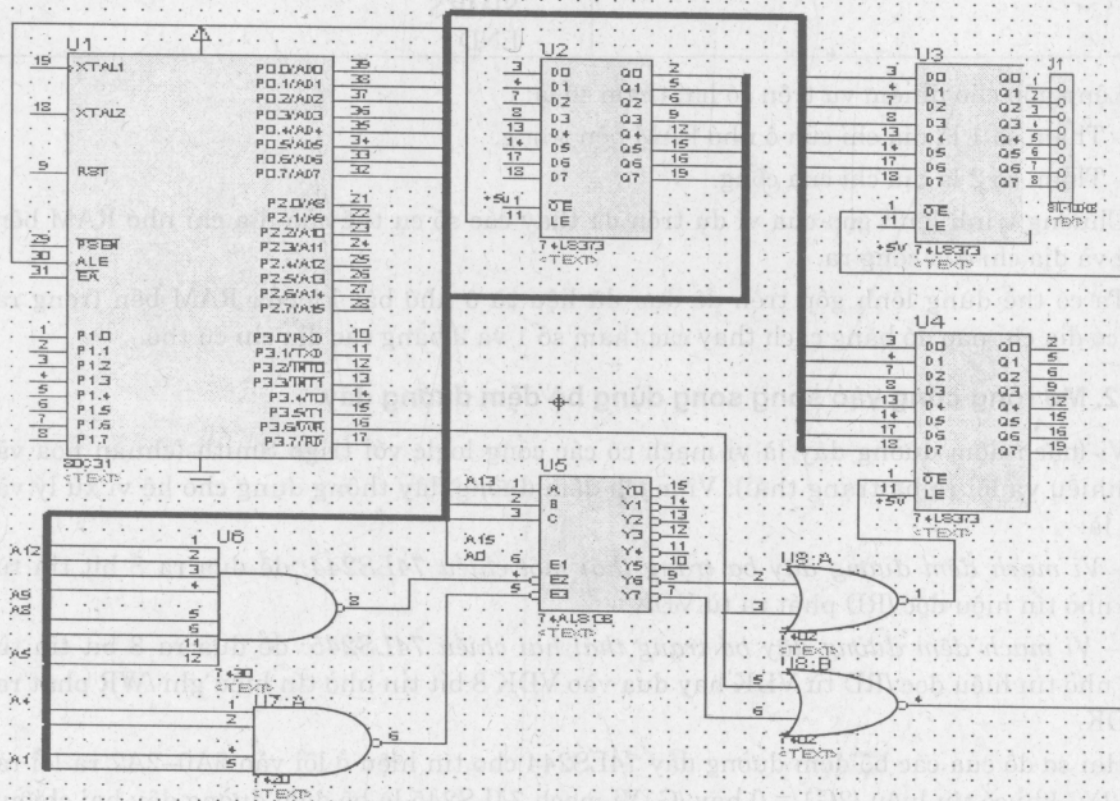
Có thể chọn một trong các thanh ghi có 8 lối vào và lối ra có thể có 3 trạng thái hay không.

Vì mạch đếm đường dây thông dụng cho hệ vi xử lý và VDK là:

– Vì mạch thanh ghi chốt 74LS373: gồm 8 trigơ D, ghi 8 bit tin nhờ mức của xung nhịp CLK, có lối ra ba trạng thái và đọc dữ liệu ra khi /OE = 0.

– Vì mạch thanh ghi chốt 74LS374: gồm 8 trigơ D, ghi 8 bit tin nhờ sườn xuống của xung nhịp CLK, có lối ra ba trạng thái và đọc dữ liệu ra khi /OE = 0.

Ngoài các vi mạch thông dụng trên, còn có các vi mạch tương tự như 74573, 74574 v.v...



Hình 16.3. Sơ đồ mắc mở rộng hai cổng ra dùng 74LS373

Hai sơ đồ của 74373 và 74374 chỉ khác nhau là 74373 ghi dữ liệu vào bằng mức LE dương, còn 74374 kích thích bằng sườn lên của xung CP. Cả hai vi mạch đọc dữ liệu ra bằng tín hiệu /OE = 0 và không ảnh hưởng tới nội dung của các trigơ D. Hai vi mạch chốt dữ liệu này được dùng để chốt các dữ liệu vào D0–D7 và đọc ra khi có tín hiệu /OE.

Sơ đồ mở rộng đường dây dùng thanh ghi đệm và bộ giải mã cho hai cổng ra của VĐK dùng vi mạch 74LS373 như hình 16.3. Sơ đồ giải mã trên dùng vi mạch 74138 với ba lối vào A13, A14, A15 để cho địa chỉ cổng:

+ U5A là $B + C = A14 + A15 + A0 = 11000001B = 0C001H$.

+ U5B là $A + B + C = A15 + A14 + A13 + A0 = 11100001B = 0E001H$.

Ví dụ 16.1 giới thiệu chương trình viết bằng lệnh và bảng lệnh gộp để đưa dữ liệu từ các ô nhớ có địa chỉ 30H và 31H ra cổng ra 1 và 2 như hình 16.3.

: Ví dụ 16.1a: Chương trình đưa nội dung ô nhớ viết bằng lệnh	: Ví dụ 16.1b: Chương trình đưa nội dung ô nhớ viết bằng lệnh gộp
<pre> ORG 00H MOV A, 30H MOV DPTR, #0C001H MOVX @DPTR, A MOV A, 31H MOV DPTR, #0E001H MOVX @DPTR, A SJMP \$ END </pre>	<pre> ORG 00H LJMP START MACRO @GHICONG <1, 2> MOV A, 11 MOV DPTR, #12 MOVX @DPTR, A ENDMAC START: @GHICONG 30H, 0C001H @GHICONG 31H, 0E001H SJMP \$ END </pre>

Lệnh gộp cho nhiệm vụ trên có hai tham số là:

- Tham số 1 là địa chỉ của ô nhớ RAM bên trong.
- Tham số 2 là địa chỉ của cổng.

Chương trình lệnh gộp của ví dụ trên đã thay các số cụ thể của địa chỉ nhớ RAM bên trong và địa chỉ của cổng ra.

Ta có thể dùng lệnh gộp trên để đưa dữ liệu từ ô nhớ bất kỳ của RAM bên trong ra cổng có địa chỉ nào đó bằng cách thay các tham số 1 và 2 bằng các địa chỉ cụ thể.

16.2.2. Mở rộng cổng vào song song dùng bộ đệm đường dây

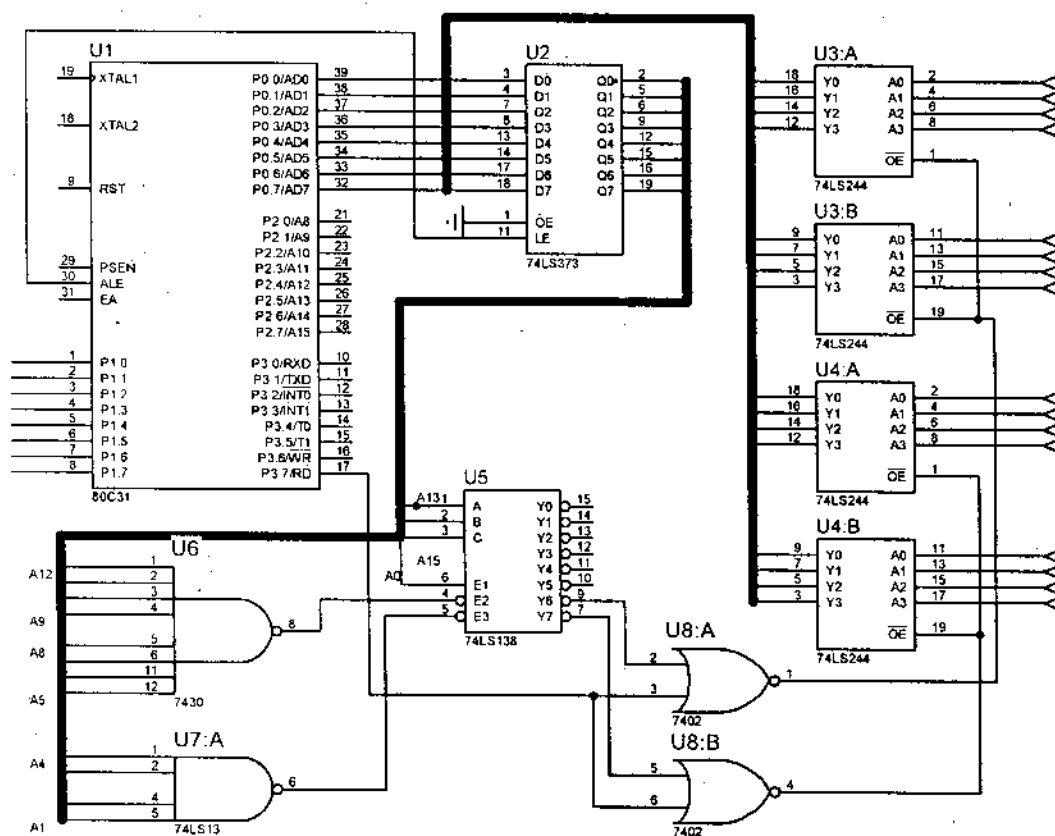
Vi mạch đệm đường dây là vi mạch có các cổng logic với trigơ Smith (chuẩn hoá và khử nhiễu và lối ra ba trạng thái). Vi mạch đệm đường dây thông dụng cho hệ vi xử lý và VĐK là:

- Vi mạch đệm đường dây ba trạng thái một chiều 74LS244: để đưa ra 8 bit tin từ VĐK nhờ tín hiệu đọc /RD phát ra từ VĐK.

- Vi mạch đệm đường dây ba trạng thái hai chiều 74LS245: để đưa ra 8 bit tin từ VĐK nhờ tín hiệu đọc /RD từ VĐK hay đưa vào VĐK 8 bit tin nhờ tín hiệu ghi /WR phát ra từ VĐK.

Hai sơ đồ của các bộ đệm đường dây 74LS244 cho tín hiệu ở lối vào 2A0–2A7 ra lối ra 2Y0–2Y7 khi có tín hiệu /2G1 = 0 hay /G. Vi mạch 74LS245 là bộ đệm đường dây hai chiều, điều khiển đưa dữ liệu ra khi /G = 0 và DIR = 0 hoặc đưa dữ liệu vào khi /G và DIR = 1.

Hai vi mạch chốt dữ liệu này được dùng để đưa các dữ liệu vào từ thiết bị ngoài (đã ghi sẵn trên các thanh ghi đệm) khi có tín hiệu điều khiển /1G, /2G hay /G và DIR.



Hình 16.4. Sơ đồ mở rộng cổng vào của 74LS244 và 74LS245

Các sơ đồ trên, đã thay thế các vi mạch thanh ghi chốt 74LS373 hay 74LS374 vì:

– Khi đưa tin vào từ các thiết bị ngoài, các thiết bị ngoài này đã ghi dữ liệu vào các thanh ghi đệm và chỉ chờ vi xử lý hay VĐK nhận tin vào, không cần bước ghi đệm vào thanh ghi chốt.

Ví dụ 16.2 giới thiệu chương trình viết bằng lệnh và bằng lệnh gộp để đưa dữ liệu từ cổng vào 1 và 2 vào các ô nhớ có địa chỉ 30H và 31H ra như hình 16.4.

Lệnh gộp cho nhiệm vụ trên có hai tham số là:

- Tham số 1 là địa chỉ của cổng.
- Tham số 2 là địa chỉ của ô nhớ RAM bên trong.

Chương trình lệnh gộp của ví dụ trên đã thay các số cụ thể của địa chỉ cổng vào cho thiết bị ngoài và địa chỉ nhớ RAM bên trong.

Ta có thể dùng lệnh gộp trên để đưa dữ liệu vào từ thiết bị ngoài vào địa chỉ nhớ RAM bên trong bằng cách thay các tham số 1 và 2 bằng các địa chỉ cụ thể.

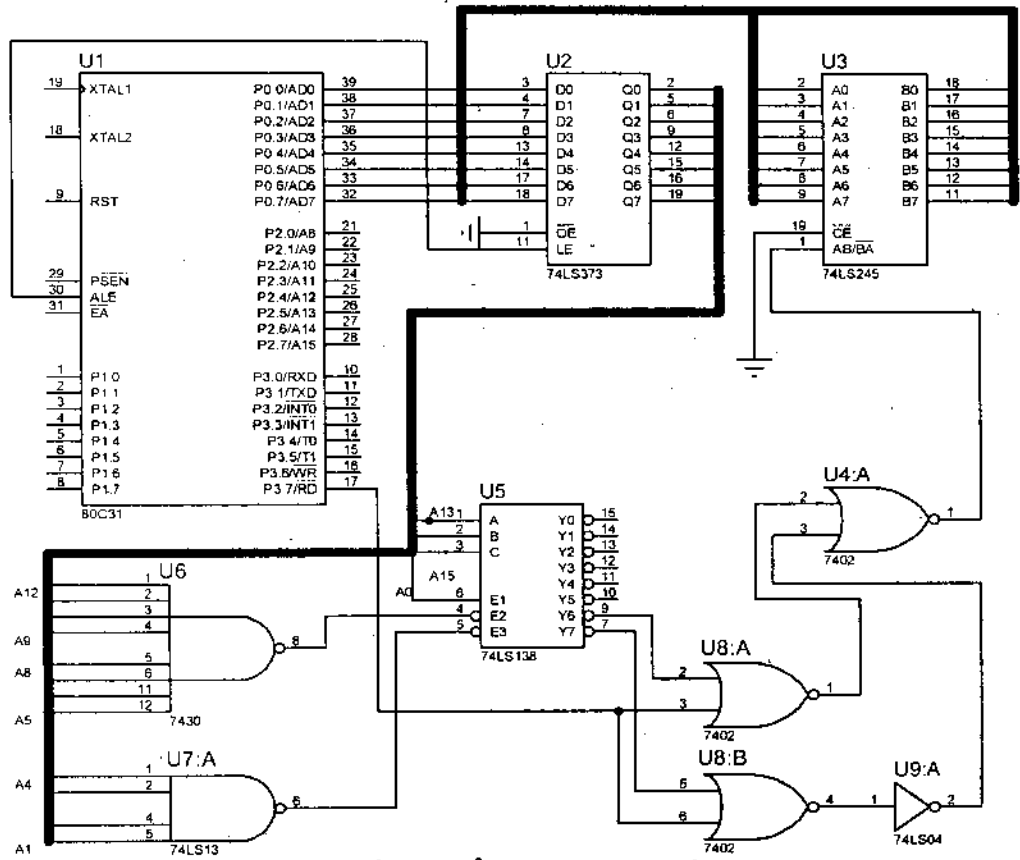
; Ví dụ 16.2a: Chương trình đưa nội dung ô nhớ viết bằng lệnh	; Ví dụ 16.2b: Chương trình đưa nội dung ô nhớ viết bằng lệnh gộp
<pre> ORG 00H MOV DPTR, #0C001H MOVX @DPTR, A MOV A, 30H MOV DPTR, #0E001H MOVX @DPTR, A MOV A, 31H SJMP \$ END </pre>	<pre> ORG 00H LJMP START MACRO @DOCCONG <1, 2> MOV DPTR, #1 MOVX @DPTR, A MOV A, 12 ENDMAC START: @DOCCONG 0C001H, 30H @DOCCONG 0E001H, 31H SJMP \$ END </pre>

16.2.3. Mở rộng cổng vào song song dùng bộ đệm đường dây

Bộ đệm đường dây 74LS245 có lối vào điều khiển DIR có thể dùng để điều khiển dữ liệu vào – ra hai chiều khi:

- DIR = 0 đưa dữ liệu vào.
- DIR = 1 đưa dữ liệu ra.

Hình 16.5 giới thiệu sơ đồ mở rộng cổng vào – ra hai chiều dùng 74LS245.



Hình 16.5. Sơ đồ mở rộng cổng vào - ra hai chiều dùng 74LS245

Chương trình đưa dữ liệu cho cổng vào/ra này cũng tương tự chương trình cho cổng ra và vào (các ví dụ 16.2).

16.2.4. Mở rộng cổng vào/ra song song dùng thanh ghi dịch nối tiếp – song song

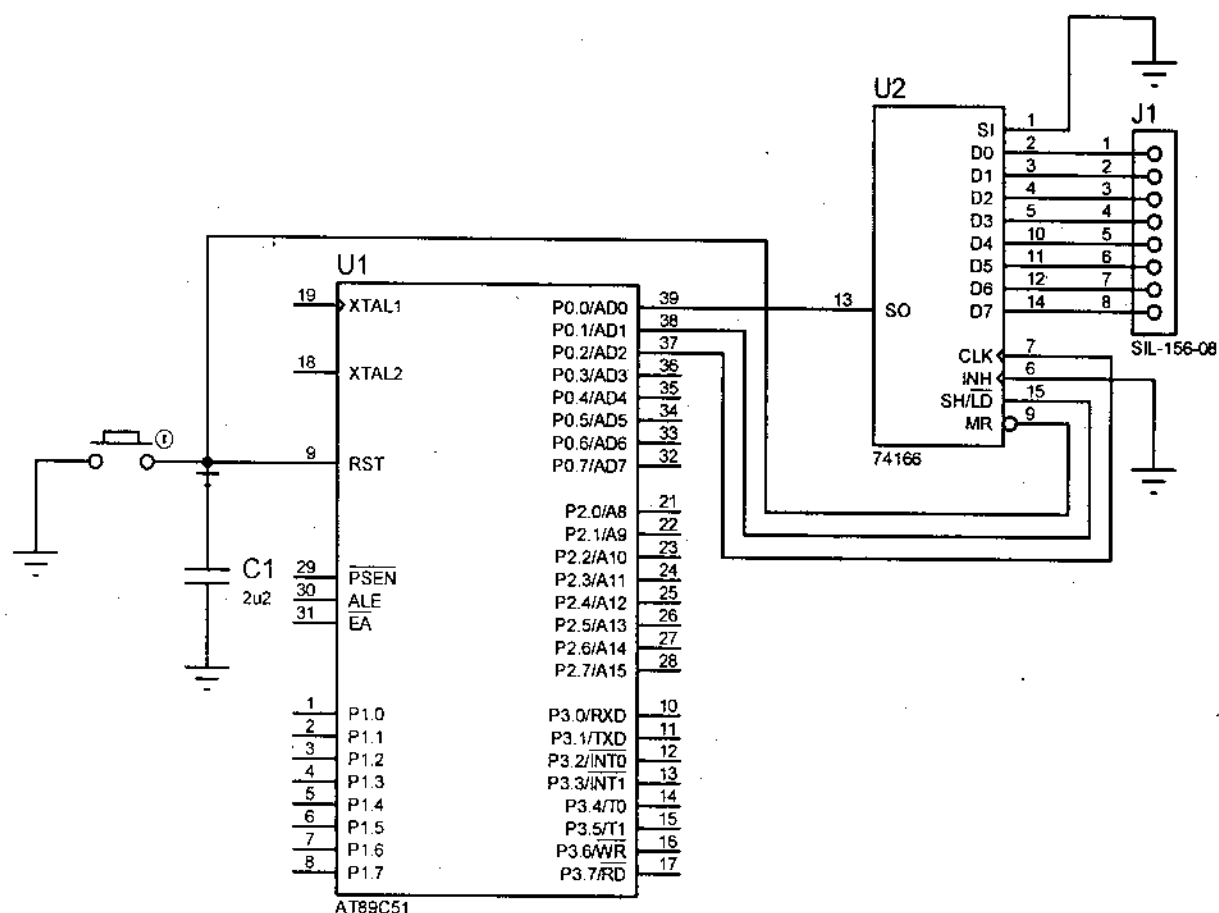
Thanh ghi dịch vào song song ra nối tiếp nạp dữ liệu song song, sau đó sẽ dịch nối tiếp để cho ra nối tiếp từng bit SO vào một bit của cổng song song của VDK. Đọc bit của cổng song song này, nạp vào bit Acc.0 của thanh ghi chứa A. Như vậy, sau 8 lần dịch và đọc, trong thanh ghi A có 8 bit của cổng vào song song.

Nhiều vi mạch vào song song ra nối tiếp đã được chế tạo, trong đó có 74LS166 được dùng để mở rộng cổng vào song song.

Các chân của 74LS166 như sau:

- *CLR* : xoá các bit về 0.
- *A–H*: các chân dữ liệu vào song song.
- *CLOCK* và *CLOCK INHIBIT*: nhịp và cấm nhịp để ghi hay cấm ghi dữ liệu vào song song.
- *SER* : lối vào tín hiệu nối tiếp.

Sơ đồ ghép nối với VDK làm cổng vào song song như hình 16.6.



Hình 16.6. Sơ đồ mở rộng cổng vào dùng 74LS166

Sơ đồ hình 16.6 cần lưu ý sau:

- Lối vào SER nối đất để bit dịch đầu tiên là 0. Sau khi dịch đủ 8 bit cho một byte, lối ra SO = 0 (nối với P0.0 để kết thúc chương trình đưa dữ liệu vào).
- Chân P0.1 đưa điều khiển nạp song song khi /LD = 0, nên giá trị ban đầu P0.1 = 1. Sau đó trở về 1 để dịch bit trong 74LS166.
- Tín hiệu nhịp CLK nối với bit P0.2 để dịch bit.
- Tín hiệu xoá CLEAR nối với chân RESET để xoá nội dung thanh ghi về 0 khi khởi phát.
- Tín hiệu CLK INHIBIT nối đất để cấm xung nhịp do nhiễu.

Như vậy, chỉ cần 3 bit của cổng song song, ta có thể mở rộng thành cổng song song 8 bit khi dùng thanh ghi dịch song song nối tiếp.

Chương trình cho cổng vào dùng 74LS166 như ở ví dụ 16.3.

; Ví dụ 16.3: Chương trình cho cổng vào dùng 74LS166					
ORG 00H			SETB C	; C = 1	
MOV R0, #7H	; cho 8 bit		MOV P0.2, C	; đưa CLK = 1	
CLR C	; C = 0		MOV C, P0.0	; đọc P0.0 vào C	
MOV P0.1, C	; P0.1 cho /LD để nạp		MOV ACC.0, C	; đưa C vào Acc.0	
SETB C	; C = 1		RLC A	; dịch trái A	
MOV P0.1, C	; SH = 1		DJNZ R0 LAP	; lặp lại 8 lần	
LAP: CLR C			SJMP \$	; kết thúc chờ	
MOV P0.2, C	; đưa CLK = 0		END		

Chú ý:

- Chương trình trên nạp song song 8 bit của byte dữ liệu với các bit A (D0) tới H (D7).
- Chương trình trên dịch bit Qh cao nhất vào S0 trước tiên và sau đó lần lượt các bit D6 tới D0.
- Lệnh RL A dịch các bit vào trước thành các bit cao.

Tương tự phương pháp trên, ta có thể dùng thanh ghi dịch vào nối tiếp, ra song song để mở rộng cổng ra song song.

Nhiều vi mạch vào song song ra nối tiếp đã được chế tạo, trong đó có MC 14094 được dùng để mở rộng cổng ra song song.

Các chân của MC14096 như sau:

- DATA: đưa dữ liệu vào nối tiếp.
- STROBE: điều khiển ghi dữ liệu nối tiếp.
- CLOCK: nhịp cho dịch bit.
- OUTPUT ENABLE: đưa dữ liệu ra song song trên Q1-Q8.

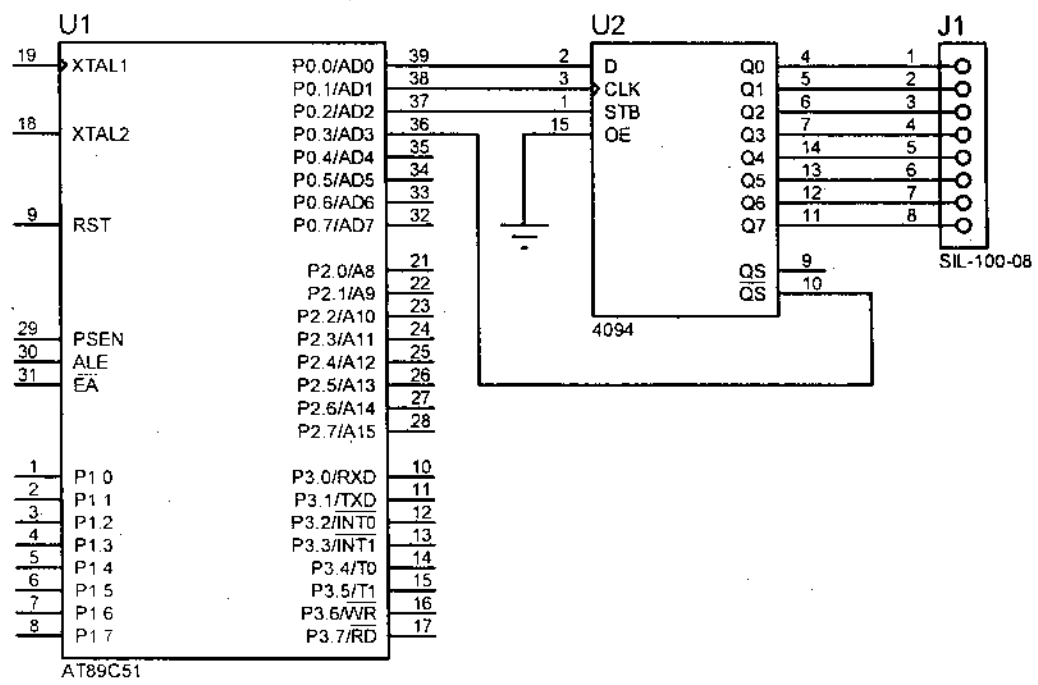
Qs và Q's: tín hiệu thứ 8 được dịch ra, để báo hiệu đã dịch xong.

Ta cần lưu ý là có hai trạng thái ghi dịch dữ liệu DATA = 0 và 1 là:

CLK	OUTPUT ENABLE	STRONE	DATA	Q1	Qn
Tăng lên 1	1	1	0	0	Qn-1
Tăng lên 1	1	1	1	1	Qn-1

Sơ đồ cổng ra dùng MC14094 như ở hình 16.7.

: Ví dụ 16.4: Chương trình đưa dữ liệu ra cổng song song ở hình 16.7	
DỮ LIỆU EQU 30H; dữ liệu ở 30H MOV R1, #00H MOV R0, #7H; cho 8 bit MOV B, 30H LAP: CLRC; C = 0 MOV A, #00H MOV P0, A MOV A, B RRC A MOV B, A MOV P0.0, C; P0.0 đưa ra DATA MOV ACC.7, C ANL A, #80H	ADD A, R1 RR A MOV R1, A SETB C; xác lập C MOV P0.1, C; CLK = 1 MOV P0.2, C DJNZ R0, LAP MOV A, R1 ADD A, #80H MOV R1, A SJMP \$ END



Hình 16.7. Sơ đồ cổng ra dùng MC 14094

Ví dụ 16.4 giới thiệu chương trình cho cổng ra dùng MC14094. Chương trình cho cổng mở rộng đưa ra dữ liệu như ví dụ 16.4.

Chú ý:

- Chương trình trên đưa dữ liệu từ ô nhớ địa chỉ 30H của RAM trong ra cổng mở rộng dùng thanh ghi dịch nối tiếp – song song MC14094 và đồng thời đưa vào R1 để theo dõi dữ liệu dịch.
- Có ba lệnh MOV A, R1; ADD A, #80H và MOV R1, A để bù lại giá trị của R1 vì trước khi kết thúc (bởi SJMP \$), thanh ghi đã bị dịch sang phải một bit.

16.2.5. Mở rộng cổng vào/ra song song dùng vi mạch PCF 8574

Người ta đã chế tạo ra vi mạch PCF 8574 để ghép nối mở rộng cổng song song hai chiều (vào/ra) cho hệ VDK và hệ vi xử lý.

- Vì mạch PCF8574 có những đặc điểm sau:
- Sơ đồ khối có thanh ghi dịch và cổng song song đưa tín vào hay ra hai chiều.
 - Trao đổi tín nối tiếp theo I2C (xem chương 15 về vi mạch nhớ nối tiếp) nên chỉ cần 2 bit của cổng song song của VDK để trao đổi tín.
 - Các lối ra hay vào song song nối với thiết bị ngoài đưa tín ra hay đưa tín vào (hai chiều).
 - Các lối vào địa chỉ A0–A2 dùng để điều khiển vùng nhớ bên trong. Với cổng đơn giản, các lối vào này nối với đất.
- Chương trình trao đổi tín tương tự chương trình trao đổi tín theo I2C cho bộ nhớ nối tiếp.

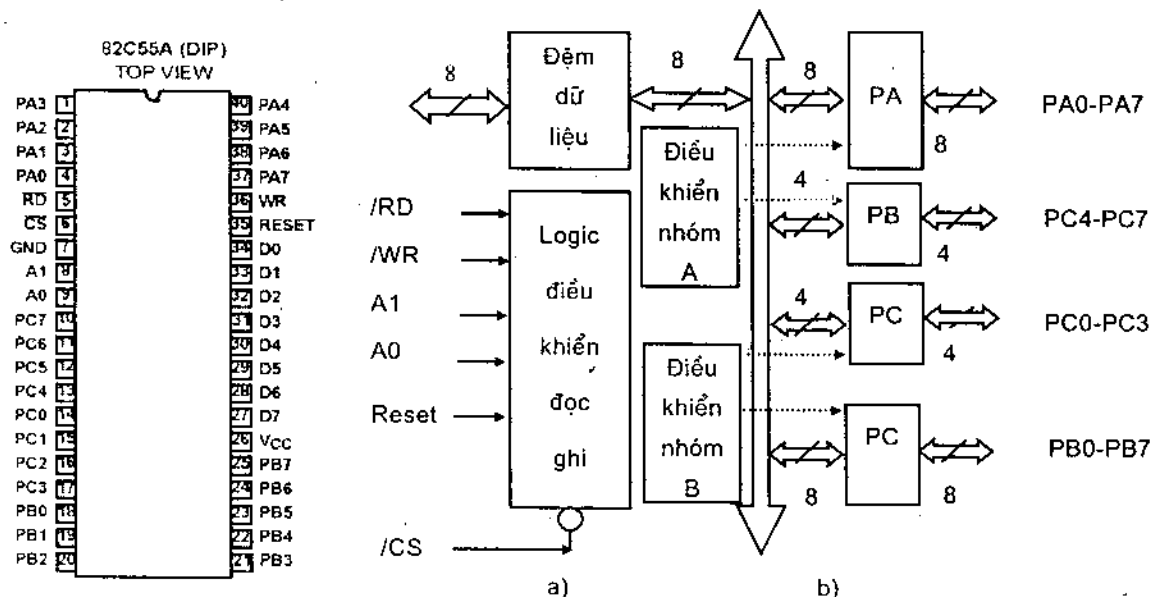
16.3. MỞ RỘNG CỔNG SONG SONG DÙNG VI MẠCH 8255A

16.3.1. Đặc điểm của vi mạch 8255A

1. Sơ đồ chân và sơ đồ khối

Vi mạch 8255A là một mạch tích hợp mật độ cao, có 3 thanh ghi đệm dữ liệu và một thanh ghi điều khiển CWR. Chế độ hoạt động của vi mạch và chiều vào/ra của các thanh ghi PA, PB, PC có thể ghi bằng chương trình (chương trình hoá – Programmable).

Hình 16.8 giới thiệu sơ đồ chân và sơ đồ khối của 8255A.



Hình 16.8. Sơ đồ chân và sơ đồ khối của 8255A

Bảng 16.1 giới thiệu tên các chân của 82C55.

Từ sơ đồ khối, ta thấy vi mạch 8255A có:

- *Khối điều khiển* : gồm khối điều khiển logic đọc/ghi, các khối điều khiển nhóm A và B để điều khiển chế độ vào/ra (cho các cổng PA, PB, PC), giải mã địa chỉ cho các thanh ghi (nhận các tín hiệu địa chỉ A0, A1), điều khiển ghi, đọc (nhận các tín hiệu /WR, /RD) và điều khiển phát các tín hiệu hội thoại, cho phép ngắt INTE, yêu cầu ngắt INTR và trạng thái trên PC.

Bảng 16.1. Tên các chân của 82C55

Loại	Ký hiệu	Chân số	Công dụng
Nguồn nuôi	Vss	26	Nuôi vi mạch
	GND	7	Đất
Điều khiển	RESET	35	Xoá về 0
	/CS	6	Chọn vi mạch
	A0	9	Địa chỉ thanh ghi
	A1	8	Địa chỉ thanh ghi
	/RD	5	Lệnh Đọc
	/WR	36	Lệnh Ghi
Cổng vào/ra	PA0-PA7	4, 3, 2, 1, 40, 39, 38, 37	Cổng vào/ra PA
	PB0-PB7	18-25	Cổng vào/ra PB
	PC0-PC7	14, 15, 16, 17, 13, 12, 11, 10	Cổng vào/ra PC
Dữ liệu	D0-D7	34-27	Đệm dữ liệu

– Thanh ghi đệm dữ liệu vào/ra: là thanh ghi đệm dữ liệu 8 bit hai chiều (vào hay ra), nối với cổng vào/ra của VDK hay Vi xử lý để trao đổi tin giữa VDK (hay VXL) với các cổng PA, PB, PC.

– Thanh ghi điều khiển CWR: dùng để điều khiển chế độ hoạt động và chiều vào/ra của các cổng PA, PB, PC. Các bit của thanh ghi này như trên hình 16.9a.

– Các cổng vào/ra PA, PB, PC: các cổng 8 bit này có chiều vào, ra hay cả vào và ra thuận nghịch (chỉ có ở cổng PA ở chế độ M = 2) tùy theo lời lệnh điều khiển ghi ở thanh ghi điều khiển.

Tín hiệu Reset nối với Reset của VDK để xoá các thanh ghi.

16.3.2. Các thanh ghi

Ngoài bộ đệm dữ liệu không có địa chỉ dùng để trao đổi tin giữa các thanh ghi điều khiển, các thanh ghi đệm dữ liệu PA, PB, PC, vi mạch 8255A có 4 thanh ghi dữ liệu vào hay ra PA, PB, PC và điều khiển CWR. Các thanh ghi này có địa chỉ cơ sở (gốc) là địa chỉ đưa vào lối vào /CS và có địa chỉ lệch (offset) là 0H, 01H, 02H và 03H. Ở chế độ trao đổi tin có hội thoại (bắt tay), thanh ghi PC đóng vai trò của thanh ghi trạng thái. Các địa chỉ của các thanh ghi và các lệnh đọc ghi như ở bảng 16.2.

Bảng 16.2. Các địa chỉ và các lệnh đọc, ghi của 8255A

A1	A0	/CS	/WR	/RD	Lệnh của VDK	Chiều truyền dữ liệu
0	0	0	1	0	Đọc thanh ghi PA	D7-D0 <— PA
0	1	0	1	0	Đọc thanh ghi PB	D7-D0 <— PB
1	1	0	1	0	Đọc thanh ghi PC	D7-D0 <— PC
1	0	0	1	0	Đọc thanh ghi điều khiển	D7-D0 <— điều khiển
x	x	1	x	x	(không dùng)	(không dùng)
0	0	0	0	1	Ghi thanh ghi PA	D7 - D0 —> PA
0	1	0	0	1	Ghi thanh ghi PB	D7 - D0 —> PB
1	0	0	0	1	Ghi thanh ghi PC	D7 - D0 —> PC
1	1	0	0	1	Ghi thanh ghi điều khiển	D7 - D0 —> điều khiển

1. Thanh ghi điều khiển

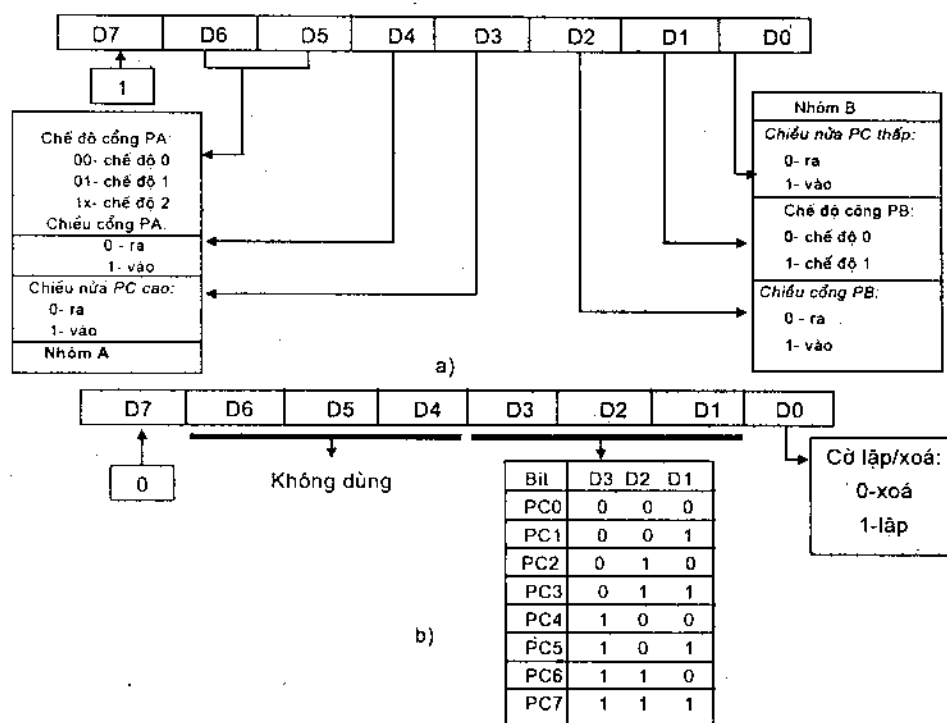
Thanh ghi điều khiển có địa chỉ là địa chỉ cơ sở (đưa vào /CS) + 03H.

Thanh ghi này dùng để ghi lời điều khiển chế độ (bit D7 = 1) có các bit như hình 16.9a.

Thanh ghi này khi bit D7 = 0, dùng để ghi lời điều khiển xác lập/xoá các bit của cổng PC như hình 16.9b. Thanh ghi điều khiển này quyết định chế độ, chiều vào/ra của các thanh ghi đệm dữ liệu PA, PB và PC.

2. Thanh ghi trạng thái

Chỉ ở chế độ M = 1, 2, chế độ trao đổi tin có sự hội thoại giữa 8255A với thiết bị ngoài, khi đó, thanh ghi PC (PC thấp cho nhóm B có PB và PC cao cho nhóm A có PA) làm các tín hiệu hội thoại trạng thái của các cổng PA và PB.



Hình 16.9. Lời điều khiển chế độ (a) và lập/xoá các bit của cổng PC (b)

Lời trạng thái ở các chế độ M = 1, 2 có dạng như hình 16.10.

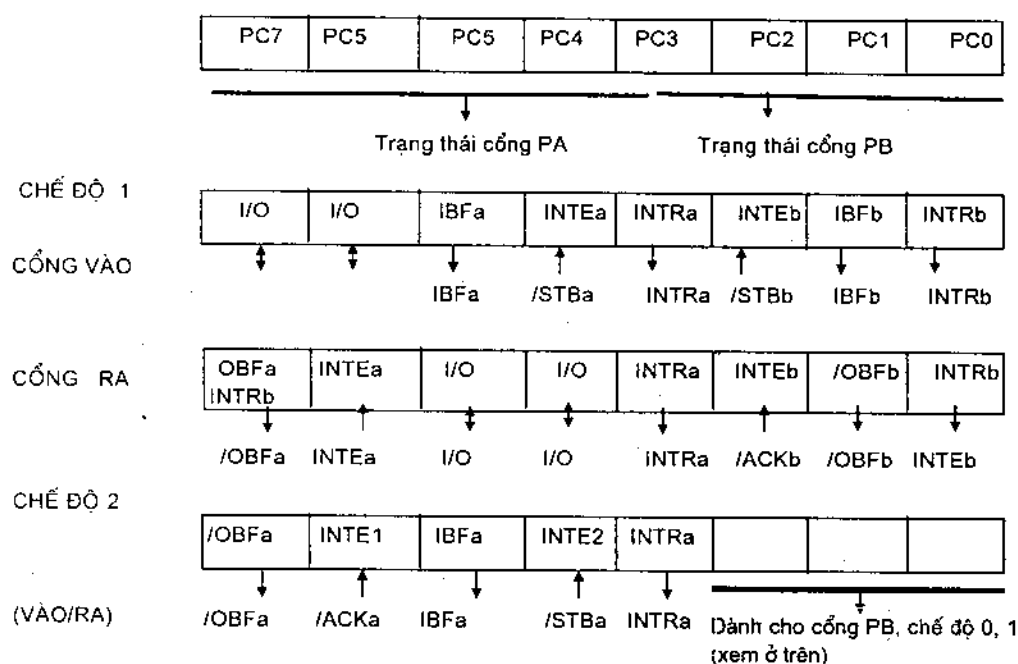
Tuỳ theo cổng là cổng vào hay cổng ra, ta có các bit trạng thái tương ứng khác nhau. Nhưng các bit yêu cầu ngắt chương trình INTRa, INTRb luôn cố định ở PC3, PC0 và có chiều đi ra để nối vào các chân /INT0, /INT1 của VĐK (hay INTR của vi xử lý).

Các bit trạng thái cho phép ngắt INTEa, INTEb, INTE1, INTE2 ở bên trong vi mạch, do các trigơ cho phép ngắt trên các bit tương ứng của PC, khi ghi lời điều khiển có D7 = 0 ghi vào thanh ghi điều khiển CWR (hình 16.10).

Các tín hiệu IBFa, IBFb cho biết dữ liệu của bộ đệm vào của 8255A đã đầy (do thiết bị ngoài ghi vào bởi tín hiệu STA, STB).

Các tín hiệu /OBFa, /OBFb cho biết dữ liệu của bộ đệm ra của 8255A đã đầy (VĐK đã ghi ra).

Người ta đọc các bit trạng thái của thanh ghi PC này để biết trạng thái của vi mạch, khi kiểm tra trạng thái ở chế độ trao đổi hội thoại theo kiểm tra trạng thái (hỏi vòng trạng thái), hay ngắt chương trình. Ở chế độ ngắt chương trình, phải nối các tín hiệu Yêu cầu ngắt chương trình INTRa, INTRb với một trong các lỗi vào ngắt (/INT0, /INT1) của VĐK (hay INTR của vi xử lý 8086).



Hình 16.10. Thanh ghi trạng thái của các cổng PA, PB ở các chế độ hội thoại M = 1, 2

3. Thanh ghi dữ liệu PA, PB, PC

Ở chế độ 0, các thanh ghi PA, PB, PC dùng để tích dữ liệu đệm vào hay đưa ra cho thiết bị ngoài. Ở chế độ 1 và 2, chỉ có hai thanh ghi PA, PB dùng để tích dữ liệu đệm vào hay đưa ra cho thiết bị ngoài còn PC là thanh ghi trạng thái. Ở chế độ 2, thanh ghi PA vừa là đệm dữ liệu vào hay ra (thuận nghịch hay hai chiều).

16.3.3. Các chế độ

Tuỳ theo giá trị các bit thấp (từ D6 tới D0) của thanh ghi điều khiển, vi mạch 8255A có các chế độ sau:

- *Chế độ M = 0:* ở chế độ này, các cổng PA, PB và PC hoạt động độc lập nhau, đều đưa tín vào hoặc đưa tín 8 bit ra không có hội thoại (handshake – bắt tay).
- *Chế độ M = 1:* ở chế độ này, cổng PA và PB trao đổi tín có hội thoại với VĐK và thiết bị ngoài và dùng các bit của cổng PC làm tín hiệu hội thoại.
- *Chế độ M = 2:* ở chế độ này, cổng PA trao đổi tín theo hai chiều thuận nghịch (vào/ra) và theo chế độ hội thoại.

Tóm lại, ta có bảng tổng kết về các cổng PA, PB, PC, các tín hiệu cho các chế độ như bảng 16.3.

Bảng 16.3. Các tín hiệu của các cổng PA, PB và PC ở các chế độ

PORT PA	CHẾ ĐỘ 0		CHẾ ĐỘ 1		CHẾ ĐỘ 2	CHẾ ĐỘ BIT
PORT PB	VÀO	RA	VÀO	RA	I/O	Không dùng
PORT PC	VÀO	RA	VÀO	RA	Không	Không dùng
0	VÀO	RA	INTRb	INTRb	I/O	1-set,0-reset Xác lập bit
1			IBFB	/OBFB	I/O	
2			/STBb	/ACKb	I/O	
3			INTRa	INTRa	I/O	
4			/STBa	I/O	/STBa	Không dùng
5			IBFa	I/O	IBFa	
6			I/O	/ACKa	/ACKa	
7			I/O	/OBFa	/OBFa	

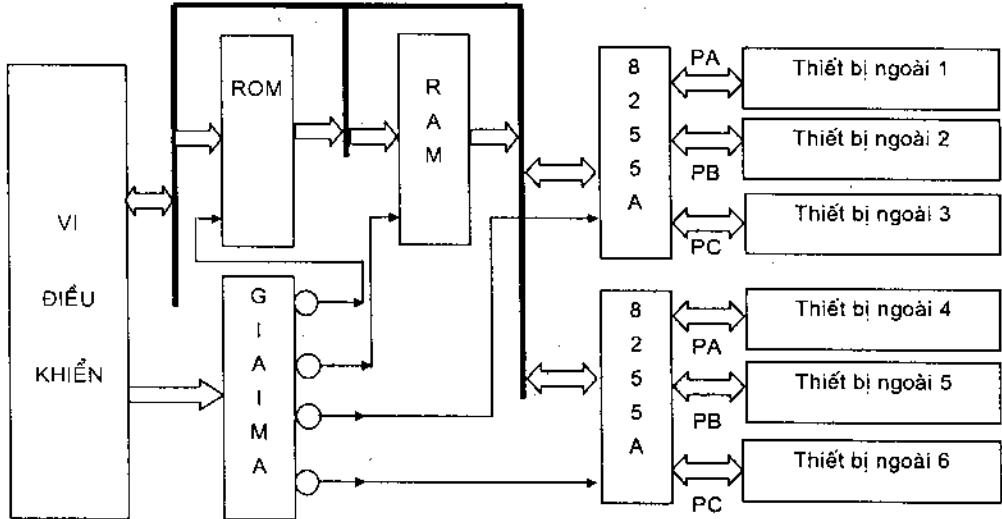
16.3.4. Một số lưu ý khi dùng 82C55 trong hệ VDK khi mắc với thiết bị ngoài

1. Mạch ghép nối

Khi mắc 82C55 trong hệ VDK, ta cần lưu ý những điểm sau:

- 82C55 mắc với cổng dữ liệu P0 (dồn kênh cùng với địa chỉ thấp A0–A7) hoặc P1, còn cổng P2 dành cho các địa chỉ cao A8–A15.
- Lối vào /CS của 82C55 nối với lối ra /CS của mạch giải mã. Mạch giải mã này phải có địa chỉ vùng hẹp /CS dành cho TBN. Ví dụ, hình 16.12 cho tín hiệu giải mã /CS có giá trị 8000H.
- Các lối vào địa chỉ A0, A1 nối với đường dây ra của địa chỉ A0, A1 trên vi mạch chốt địa chỉ (thường dùng 74LS373 hay 74LS374).
- Các lối vào /WR, /RD nối với các lối ra tương ứng của VDK.
- Vi mạch 8255A nối với TBN qua đường dây dữ liệu của một trong các cổng PA, PB và PC. Nếu TBN có hai đường dây dữ liệu vào, ra riêng rẽ, cần nối với hai cổng PA, PB hoặc PC.

Vi mạch cổng song song 82C55 nối giữa VDK và TBN và tin trao đổi theo sơ đồ khối như hình 16.11. Như vậy, một vi mạch 82C55 nối với một cổng của VDK có thể mở rộng thành ba cổng song song PA, PB, PC và trao đổi với ba thanh ghi đệm dữ liệu của thiết bị ngoài.

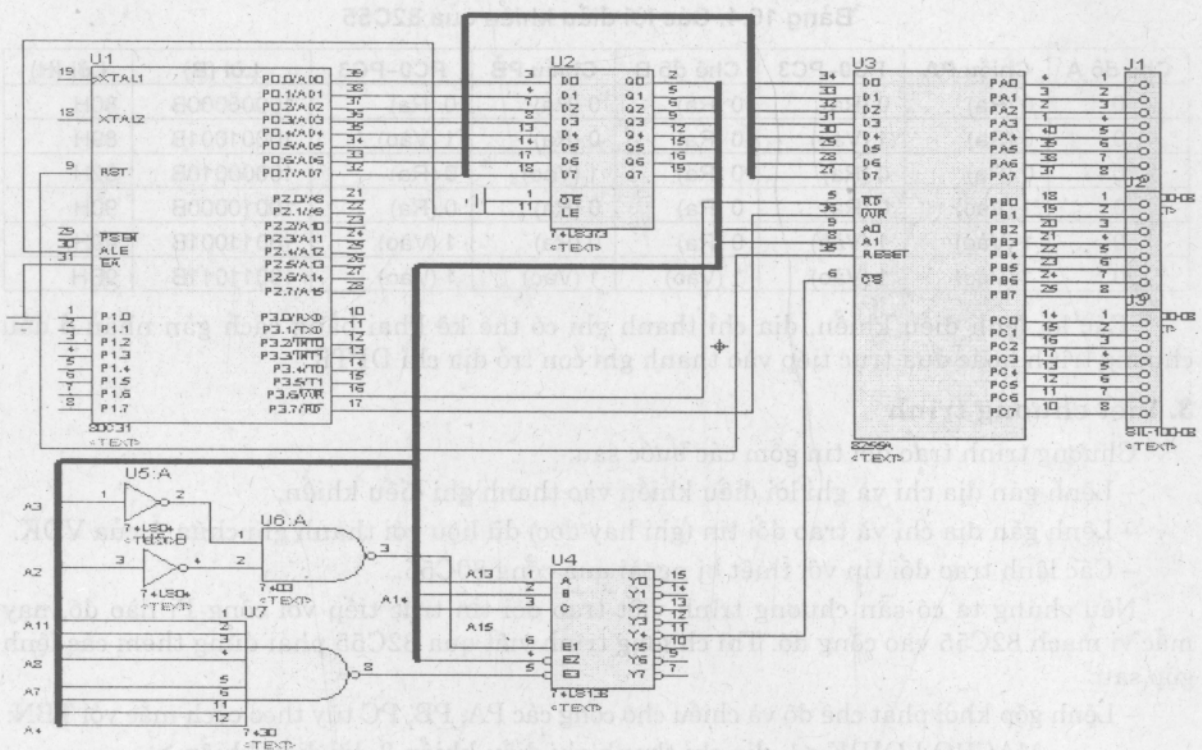


Hình 16.11. Cách mắc 8255A với VDK và thiết bị ngoài

Nếu có nhiều TBN cần ghép nối với VDK mà số cổng của một vi mạch 8255A không đủ, ta có thể mắc song song nhiều vi mạch 8255A (vì cổng ra của VDK có điện trở treo mức tín hiệu lên 1) nhưng các vi mạch 8255A trên có lối vào /CS nối với các lối ra giải mã khác nhau. Chỉ khi có lệnh đọc hay ghi có đúng địa chỉ dành cho TBN, vi mạch 8255A tương ứng mới hoạt động.

Chiều trao đổi tin vào hay ra của cổng (PA, PB hay PC) phải tương ứng với nhiệm vụ đưa tin vào hay ra, do lời điều khiển ghi vào thanh ghi điều khiển quyết định.

Sơ đồ nguyên lý của hệ trao đổi tin giữa VDK với TBN qua một vi mạch 82C55 như hình 16.12.



Hình 16.12. Ghép nối cổng 82C55 với VDK và thiết bị ngoài

Các cổng PA, PB và PC được nối với các ổ cắm 8 chân, dùng để nối với các lối vào của TBN.

2. Tính các địa chỉ và lời điều khiển

Trước khi viết chương trình, ta phải:

– *Xác định địa chỉ các cổng PA, PB, PC và thanh ghi điều khiển:* trên cơ sở địa chỉ cơ bản (do tín hiệu giải mã /CS quyết định) như ví dụ 16.5.

Ví dụ 16.5 là chương trình ghi tin ra cổng PA, PC và đọc tin vào cổng PB của vi mạch 8255A có địa chỉ cơ bản là 8000H (của giải mã nối vào /CS).

Lời điều khiển cho chế độ M = 0, cổng PA đưa tin ra, cổng PB đưa tin vào là 10000010B = 82H.

Địa chỉ thanh ghi điều khiển là: $8000 + 03H = 8003H$.
 Địa chỉ thanh ghi ra PA là: $8000H + 00H = 8000H$.
 Địa chỉ thanh ghi vào PB là: $8000H + 01H = 80001H$.
 Thanh ghi PC (địa chỉ là $8002H$) chưa sử dụng,
 – *Xác định lời điều khiển để ghi vào thanh ghi điều khiển:* lời điều khiển này luôn có bit $D7 = 1$, chế độ và chiều trao đổi tin tương ứng với TBN.
 Bảng 16.3 giới thiệu các lời điều khiển cho các chế độ khác nhau và chiều của cổng khác nhau.

Bảng 16.4. Các lời điều khiển của 82C55

Chế độ A	Chiều PA	PC0-PC3	Chế độ B	Chiều PB	PC0-PC3	Lời (B)	Lời (H)
0	0 (Ra)	0 (Ra)	0 (Ra)	0 (Ra)	0 (Ra)	10000000B	80H
0	0 (Ra)	1 (Vào)	0 (Ra)	0 (Ra)	1 (Vào)	10001001B	89H
0	0 (Ra)	0 (Ra)	0 (Ra)	1 (Vào)	0 (Ra)	10000010B	82H
0	1 (Vào)	0 (Ra)	0 (Ra)	0 (Ra)	0 (Ra)	10010000B	90H
0	1 (Vào)	1 (Vào)	0 (Ra)	0 (Ra)	1 (Vào)	10011001B	99H
0	1 (Vào)	1 (Vào)	1 (Vào)	1 (Vào)	1 (Vào)	10011011B	9BH

Các lời lệnh điều khiển, địa chỉ thanh ghi có thể kê khai bằng cách gán nhãn ở đầu chương trình hoặc đưa trực tiếp vào thanh ghi con trỏ địa chỉ DPTR.

3. Viết chương trình

Chương trình trao đổi tin gồm các bước sau:

- Lệnh gán địa chỉ và ghi lời điều khiển vào thanh ghi điều khiển.
- Lệnh gán địa chỉ và trao đổi tin (ghi hay đọc) dữ liệu với thanh ghi chứa A của VĐK.
- Các lệnh trao đổi tin với thiết bị ngoài qua cổng 80C55.

Nếu chúng ta có sẵn chương trình viết trao đổi tin trực tiếp với cổng Pi nào đó, nay mắc vi mạch 82C55 vào cổng đó. Thì chương trình viết qua 82C55 phải dùng thêm các lệnh gộp sau:

- Lệnh gộp khởi phát chế độ và chiều cho cổng các PA, PB, PC tùy theo cách mắc với TBN:
 MACRO LOIDK <1. địa chỉ thanh ghi điều khiển 2. lời điều khiển >
 MOVDPTR, #|1 ; gán địa chỉ thanh ghi điều khiển
 MOV A, #|2 ; gán dữ liệu vào A
 MOVX@ DPTR, A; ghi lời lệnh điều khiển, lời này được tính cho mạch cụ thể.
 ENDMAC
- Lệnh gộp đưa dữ liệu từ A ra Pi (MOV Pi, A) được thay bởi các lệnh:
 MACRODUADL <1. địa chỉ cổng, 2. dữ liệu>
 MOV DPTR, #|1 ; gán địa chỉ PA
 MOV A, #|2 ; gửi dữ liệu vào A
 MOVX@ DPTR, A; ghi lời lệnh điều khiển, lời này được tính cho mạch cụ thể.
 ENDMAC
- Lệnh gộp đọc dữ liệu từ cổng Pi vào A:
 MACRO @DOC DL
 MOV DPTR, #|1 ; gán địa chỉ PB
 MOVXA, @DPTR ; đọc dữ liệu của PB vào A
 ENDMAC

Dưới đây, chúng ta sẽ xét các chương trình cụ thể cho các mạch hiển thị, bàn phím v.v...

;Ví dụ 16.5: Chương trình gửi dữ liệu ra PA và đọc vào từ PB				
	ORG 100H			
LOIDK	EQU 82h			; kê khai lời lệnh điều khiển
PA	EQU 8000H			; kê khai địa chỉ PA
PB	EQU 8001H			; kê khai địa chỉ PB
PC	EQU 8003H			; kê khai địa chỉ thanh ghi điều khiển
DATAA	DS 0			
	MOVDPTR, #DK			; gán địa chỉ thanh ghi điều khiển
	MOV @DPTR, #LOIDK			; ghi lời lệnh điều khiển
	MOV DPTR, #PA			; gán địa chỉ PA
	MOV A, #DATAA			; gửi dữ liệu DATAA vào A
	MOVX @DPTR, A			; ghi dữ liệu ra PA
	MOV DPTR, #PB			; gán địa chỉ PB
	MOVXA, @DPTR			; đọc dữ liệu của PB vào A
	SJMP \$			; kết thúc chờ
	END			

16.3.5. Ghép nối trao đổi tin của 8255A ở chế độ cơ bản M = 0

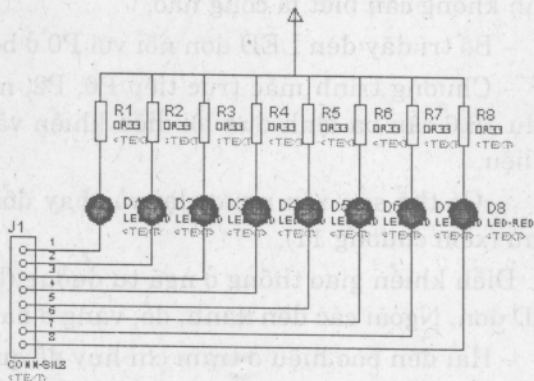
Đa số trường hợp dùng 8255A ở chế độ M = 0, chế độ trao đổi tin trực tiếp, không có hội thoại. Chế độ trao đổi tin này nhanh, mạch nối đơn giản, đỡ tốn thời gian kiểm tra và chờ các tín hiệu hội thoại, nhưng trao đổi tin không tin cậy.

Tại các chương 11, 12, 13, 14 chúng ta đã viết chương trình cho các hệ TBN như LED đơn, LED 7 đoạn, LED ma trận, LCD và bàn phím. Các TBN này mắc trực tiếp với cổng của VĐK.

1. Chương trình cho mạch hiển thị trên các LED đơn

Ví dụ 16.6 giới thiệu hai chương trình hiển thị chớp, tắt và dịch chuyển cho các LED đơn (hình 16.13) mắc trực tiếp với hai cổng P1, P2 và mắc qua PA, PB của 82C55 có địa chỉ cơ bản là 8000H.

; Ví dụ 16.6: Chương trình LED đơn sáng dần từ phải → trái, mắc với cổng PA, PB				
	org 00h			
	ljmp main			
LOIDK	EQU 80h			; lời lệnh điều khiển
PA	EQU 8000H			; địa chỉ PA
PB	EQU 8001H			; địa chỉ PB
PC	EQU 8003H			; địa chỉ thanh ghi điều khiển
MACRO@LOIDK <1. địa chỉ, 2. lời điều khiển>				
	MOV DPTR, #!			; gán địa chỉ điều khiển
	MOV A, #!2			; gán dữ liệu vào A
	MOVX @DPTR, A;			ghi lời điều khiển
ENDMAC				
MACRO @DUADL <1cổng, 2. dữ liệu>				
	MOV DPTR, #!1			; gán địa chỉ PA
	MOV A, #!2			; gửi dữ liệu vào A
	MOVX @DPTR, A;			ghi dữ liệu.



Hình 16.13. Tám LED đơn


```

ENDMAC
    Org 4000H
main:
    @LOIDK 8003H, 80H
    @DUADL8001H. 00H
    @DUADL8000H. 00H
    acall delay
    @DUADL8001H. 0FFH
    @DUADL8000H. 0FFH
    mov a, #0
nhan1: setb c                ; xác lập C
        rrc a              ; dịch phải qua C, C->D7
        mov dptr, #8001H   ; địa chỉ PB
        movx @dptr, a      ; ghi ra cổng PB
        acall delay        ; gọi trễ
        cjne a, #0ffh, nhan1 ; trở về nhan1
nhan2: setb c
        rrc a
        mov dptr, #8000H   ; địa chỉ PA
        movx @dptr, a      ; ghi ra cổng PA
        acall delay
        cjne a, #0ffh, nhan2
        sjmp main
delay: mov r5, #20
del:   mov r6, #0ffh
dl1:   mov r7, #0ffh
        djnz r7, $
        djnz r6, dl1
        djnz r5, del
        ret
        end

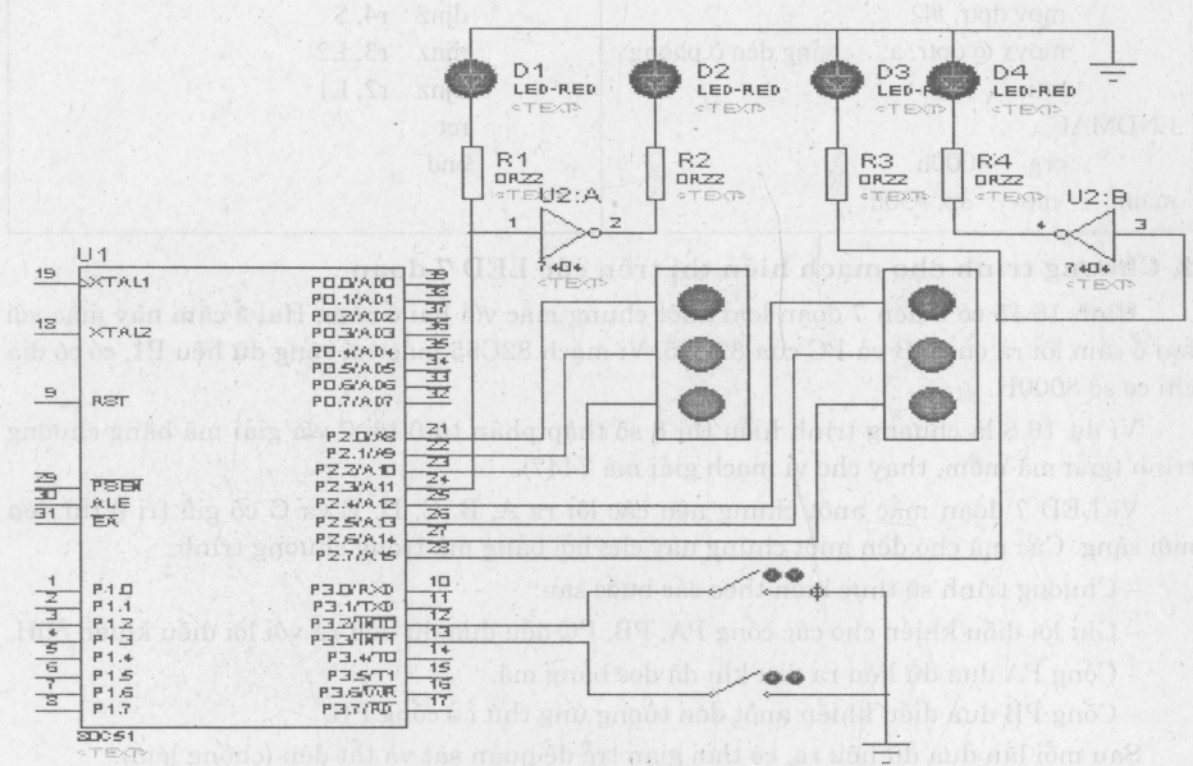
```

Chú ý:

- Mạch cho chương trình ví dụ 16.6 mắc với một cổng bất kỳ của VDK, trong chương trình không cần biết là cổng nào.
- Bố trí dây đèn LED đơn nối với P0 ở bên trái, nối với P2 ở bên phải.
- Chương trình mắc trực tiếp P0, P2, ngắn hơn chương trình mắc qua 82C55 vì trong ví dụ 16.6 cần ba lệnh đưa lời điều khiển và cần hai lệnh (gán địa chỉ, đưa dữ liệu ra) đưa dữ liệu.
- Có thể sắp xếp ngược lại và thay đổi lệnh dịch trái (RLC A) ta có chiều sáng khác nhau (xem chương 11).
- Điều khiển giao thông ở ngã tư đường (hình 16.14) là một ví dụ cụ thể của hiển thị đèn LED đơn. Ngoài các đèn xanh, đỏ, vàng ở hai trục đường, còn có:
 - Hai đèn báo hiệu ở trạm chỉ huy để quan sát.
 - Hai phím điều khiển cố định để điều khiển đèn sáng theo thời gian tùy ý của người điều hành. Hai phím này tạo hai mức ngắt ngoài cho chương trình và được nối với hai lối vào /INT0 và /INT1.

– Các lối ra điều khiển các đèn được nối với một ổ cắm 8 chân, nối với lối ra của một cổng PA, PB hay PC.

Chương trình cho mạch điều khiển giao thông như ở ví dụ 16.7. Hình vẽ và chương trình này có trong tệp PROTEUS. Thí nghiệm PROTEUS được tác giả viết lại dưới dạng lệnh gộp (macro lệnh) cho ngắn gọn, người đọc có thể hợp dịch và so sánh với chương trình nguyên mẫu.



Hình 16.14. Mạch điều khiển giao thông

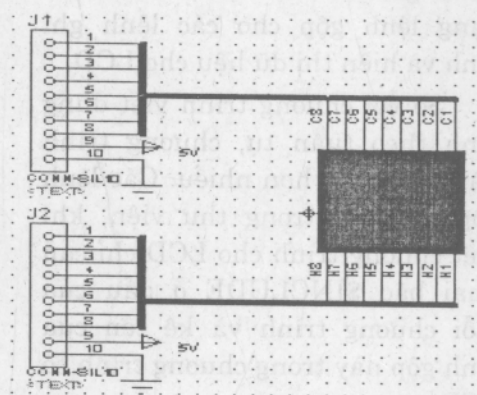
; Ví dụ 16.7. Chương trình điều khiển đèn giao thông (hình 16.14)	
<pre> org 00h ljmp main org 03h ljmp ngatut1 org 13h ljmp ngatut2 MACRO @SANG <1, 2, 3, ; 4. > mov r1, #1.solan lapl2: mov dptr, #13 ; nhãn, địa chỉ cổng mov a, #4 ; dữ liệu movx @dptr, a ; sáng đèn lcall delay </pre>	<pre> mov ie, #10000101b clr it0 clr it1 lap: @SANG 2, 1, 8002H,41H @SANG 15, 2, 8002H,91H @SANG 5, 3, 8002H,11H @SANG 4, 4, 8002H,21H @SANG 2, 5, 8002H,41H @SANG 15, 6, 8002H,4AH @SANG 5, 7, 8002H,42H @SANG 4, 8, 8002H,44H ljmp lap </pre>

; Ví dụ 16.8. Chương trình hiển thị đèn LED 7 đoạn mắc qua cổng PA, PB của 82C55

<pre> org 0000h PA equ 8000H PB equ 8001H main0: mov r0, #8h ; số lần lặp bằng số đèn mov r1, #01h ; điều khiển LED sáng mov dptr, #bangma ; địa chỉ bảng mã main: clr a movc a, @a+dptr ; đọc mã vào a mov dptr, #PA ; địa chỉ cổng PA movx @dptr, a ; đưa dữ liệu ra PA mov dptr, #PB ; địa chỉ cổng PB mov a, r1 ; gán lời điều khiển movx @dptr, a </pre>	<pre> call delay inc dptr mov a, r1 ; gán lời điều khiển rlc a ; điều khiển đèn sau mov r1, a djnz r0, main ; lặp lại 8 lần jmp main0 delay: mov r0, #255 djnz r0, \$ ret bangma: db 03fh, 06h, 04bh, 04fh, 066h db 05bh, 7dh, 07h, 07fh, 06fh end </pre>
--	---

3. Chương trình cho mạch hiển thị trên các LED ma trận

Ta đã xét ma trận diot ở chương 12 với các hàng và cột của ma trận mắc trực tiếp các cổng của VĐK. Hình 16.16 giới thiệu mạch có một ma trận diot mắc qua hai ổ cắm để cắm vào hai cổng ra bất kỳ của 82C55. Chương trình ở ví dụ 16.9 để sáng lên chữ H với 5 cột và 7 hàng, đưa trực tiếp vào các hàng và cột. Có thể thay đổi các giá trị của A cho cột và B cho hàng để có các ký tự khác nhau.



Hình 16.16. Mạch ma trận LED

; Ví dụ 16.9: Chương trình LED ma trận

<pre> ORG 00H ljmp LOOP ORG 4030H LOOP: MOV A, #00110011B MOV B, #00000000B CALL MTX MOV A, #00001100B MOV B, #11110011B CALL MTX LJMP LOOP MTX: MOV DPTR, #8000h MOV R1, #1 </pre>	<pre> LOOP1: MOV R2, #5 LOOP2: MOV R3, #255 LOOP3: MOVX @DPTR, A MOV A, #00000000B MOVX @DPTR, A MOV DPTR, #8001h MOV A, B MOVX @DPTR, A DJNZ R3, LOOP3 DJNZ R2, LOOP2 DJNZ R1, LOOP1 RET END </pre>
--	--

Chú ý:

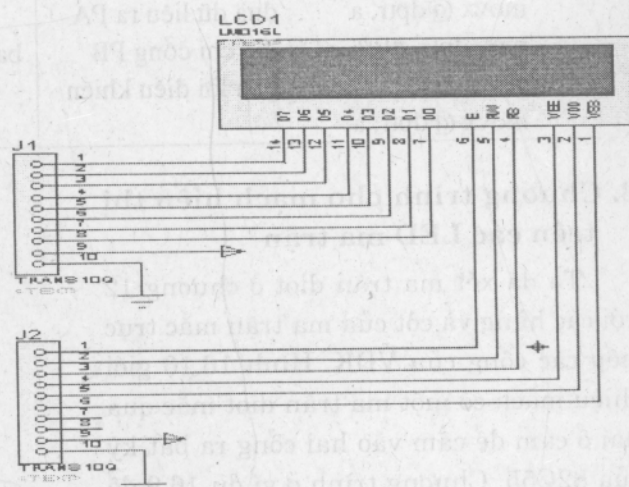
- Giá trị của A là giá trị các cột của ma trận LED, đưa ra cổng PA (thay cho P1 nếu mắc trực tiếp).
- Giá trị của B là giá trị các hàng của ma trận LED, đưa ra cổng PB (thay cho P2 nếu mắc trực tiếp).
- Đèn LED chỉ sáng nếu PA = 1 và PB = 0.
- Có hai cặp giá trị PA, PB cho hai cột thẳng đứng sáng và cặp PA, PB cho một vệt ngang sáng.
- Kết quả ma trận sáng lên chữ H.

4. Chương trình cho mạch hiển thị trên LCD

Mạch LCD dùng để nối với hai cổng của 82C55 có dạng như hình 16.17.

Ví dụ 16.10 giới thiệu chương trình ghi chữ “NO” ra LCD dưới dạng lệnh gộp cho các lệnh ghi lệnh và hiển thị dữ liệu cho LCD.

Sơ với chương trình viết dùng lệnh theo tuần tự, chương trình này ngắn gọn hơn nhiều. Các lệnh gộp này đặt trong thư viện, khi viết chương trình cho LCD chỉ cần khai báo \$INCLUDE ở đầu của mỗi chương trình và kê tên của lệnh gộp này trong chương trình có LCD.



Hình 16.17. Mạch LCD

; Ví dụ 16.10: Chương trình ghi chữ “NO” ra LCD dùng các lệnh gộp

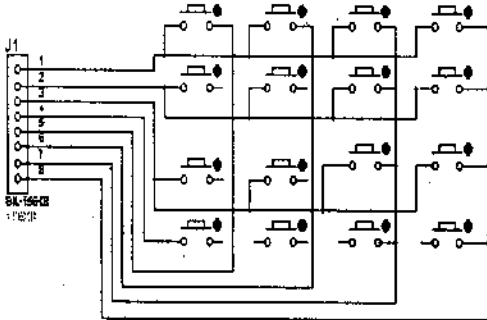
<pre> ORG 00H LJMP MAIN PB equ 8001H PC equ 8002H DCDK equ 80003H Org 4000H MACRO @LENH <1,2> ; ghi lệnh MOV A, #1 ; lệnh @GHI PB,PC,100H,00H ; ghi lệnh ra ACALL DELAY ; gọi trễ ENDMAC MACRO @GHI <1,2,3,4> ; ghi lệnh dữ liệu MOV DPTR, #1 ; địa chỉ cổng dữ liệu MOVX @DPTR, A ; ghi ra MOV DPTR, #2 ; địa chỉ cổng lệnh </pre>	<pre> MAIN: MOV A, #80H MOV DPTR, #DCDK MOVX @DPTR, A @LENH 38H @LENH 0EH @LENH 01H @LENH 06H MOV A, "N" @GHI PB, PC, 101H, 01H MOV A, "O" @GHI PB, PC, 101H, 01H DELAY: MOV R7, #200 DELO: MOV R6, # 200 </pre>
---	---

MOV A, #13 ; các tín hiệu lệnh MOVX @DPTR, A ; đưa lệnh ghi ra NOP NOP MOV A, #4 ; MOVX @DPTR, A ; chốt lệnh ghi ra ENDMAC	DEL1: MOV R5, # 50H DJNZ R5, \$ DJNZ R6, DEL1 DJNZ R7, DEL0 RET END
--	--

5. Chương trình cho mạch nối bàn phím

Mạch nối các phím của bàn phím như hình 16.18. Mạch này có 4 lối ra cho hàng và 4 lối vào cho cột. Do đó nên cắm vào ổ cắm của lối ra của cổng PC và lời điều khiển cho 4 lối ra PC thấp là cổng ra và 4 lối vào ở PC cao là cổng vào. Lời điều khiển chế độ là:

10001000B = 88H.
Chương trình phục vụ bàn phím như ví dụ 16.11.



Hình 16.18. Mạch nối các phím của bàn phím

16.3.6. Ghép nối trao đổi tin ở chế độ xác lập/xoá bit của cổng PC

1. Chế độ BSR

Khi ghi lời điều khiển chế độ với bit D7 của lời điều khiển chế độ bằng 0 vào thanh ghi điều khiển (địa chỉ = địa chỉ cơ sở +3), vi mạch 82C55 chuyển sang chế độ lập = 1 hay xoá (= 0) các bit ở cổng PC như hình 16.9. Như vậy, ta có bảng 16.5 với các giá trị khác nhau của lời điều khiển cho các tín hiệu từ PC0 tới PC5. Ta có thể suy ra cho các PCi khác.

Tương tự các cổng P0 tới P3 của VDK họ MSC-51, cổng PC của 82C55 cũng có thể xác lập và xoá bit khi ghi lời điều khiển ở chế độ BSR. Do đó, bất kỳ ứng dụng nào của các cổng của VDK có xác lập và xoá bit, khi chuyển qua dùng 82C55, ta phải dùng cổng PC.

Có thể dùng cổng PC ở chế độ BSR để điều khiển rơ-le, động cơ bước, máy phát xung, điều khiển động cơ một chiều theo kiểu điều chế độ rộng xung.

; Ví dụ 16.11: Chương trình đọc mã phím nhấn tính theo công thức	
ORG 00H LJMP MAIN Flag EQU 20h ; cờ bàn phím khi nhấn DCDK EQU 8003H PC EQU 8002H ORG 4000H MAIN: MOV DPTR, #DCDK MOV A, #88H MOVX @DPTR, A MOV A, #0FEH ; đưa D0 = 0 MOV R6, #4 ; nạp số cột COT: MOV DPTR, #PC MOV @DPTR, A; đưa 0 ra cổng PCt	DJNZ R6, COT ; lặp lại cột mới CLR Flag ; xoá cờ ngắt SJMP KT ; nhảy kết thúc MA: MOV R5, #04H ; nạp số hàng MOV R0, #00H SETB C ; xác lập C = 1 MOV R7, A ; lưu giá trị hàng SWAP A LAP: RRC A ; dịch phải qua C, tìm phím nhấn JNC XONG ; C = 0 có phím nhấn INC R0 ; tăng số phím trong cột DJNZ R5, LAP ; lặp lại 4 cột XONG: SETB Flag ; xác lập cờ bàn phím

CALL DELAY MOV R7, A ; lưu giá trị cũ MOV A, @DPTR; đọc Cột ANL A, # 0F0H ; che giá trị cột CJNE A, #0F0H, MA; so sánh MOV A, R7 ; nạp lại RL A ; chuyển cột mới INC B ; tăng lên 4 cho hàng sau INC B INC B INC B	MOV A, R0 ; gửi số phím trong cột vào A ADD A, B ; tính mã theo công thức MOV P3, A ; đưa ra hiển thị ở cổng P3 CALL DELAY KT: MOV 30H, A ; lưu mã vào ô nhớ 30H SJMP \$ DELAY: MOV R1, #10 DELAY1: MOV R2, #250 DELAY2: DJNZ R2, DELAY2 DJNZ R1, DELAY1 RET END
--	---

Bảng 16.5. Bảng các lời điều khiển ở chế độ lập/xoá bit (BSR)

PC	Giá trị	d7	d6	d5	d4	d3	d2	d1	d0	Giá trị HEX
PC0	0	0	0	0	0	0	0	0	0	00H
	1	0	0	0	0	0	0	0	1	01H
PC1	0	0	0	0	0	0	0	1	0	02H
	1	0	0	0	0	0	0	1	1	03H
PC2	0	0	0	0	0	0	1	0	0	04H
	1	0	0	0	0	0	1	0	1	05H
PC3	0	0	0	0	0	0	1	1	0	06H
	1	0	0	0	0	0	1	1	1	07H
PC4	0	0	0	0	0	1	0	0	0	08H
	1	0	0	0	0	1	0	0	1	09H
PC5	0	0	0	0	0	1	0	1	0	0AH
	1	0	0	0	0	1	0	1	1	0BH

2. Xử lý theo bit ở chân các cổng vào/ra

Các cổng vào/ra của VDK có đặc tính là có khả năng xử lý theo bit (xác lập, xoá, kiểm tra hay chờ xác lập hay xoá v.v...). Ví dụ như $Clr\ P_{i,j}$, $Setb\ P_{i,j}$ hay $jnb\ P_{i,j}$ của bit thứ j ($j = 0$ tới 7) của cổng thứ i ($i = 0$ tới 3).

Nếu bây giờ không mắc trực tiếp thiết bị ngoài vào cổng P_i nữa, các lệnh trên lệnh phải chuyển đổi ra sao ?

Rất may mắn là thanh ghi chứa A của VDK có khả năng xử lý theo bit. Khi đó, ta phải chuyển các lệnh xử lý theo bit trên vào thanh ghi chứa A.

Với các lệnh lập/xoá bit của các cổng PA, PB, ta thực hiện như sau:

- Đọc các bit của cổng vào thanh ghi A.
- Xử lý theo bit trên các bit của A.
- Chuyển bit của A cho bit của cổng PA, PB hay PC của 82C55. Riêng các lệnh xác lập và xoá theo bit có thể thực hiện trên các bit của cổng PC.

Với lệnh chờ xác lập ($JB\ P_i, \$$) hay xoá bit ($JNB\ P_i, \$$) của cổng P_i nay chuyển cho các cổng PA, PB hay PC, ta thực hiện như sau:

- Đọc cổng PA, PB, hay PC vào A.
 - Chờ xác lập (JB ACCi, \$) hay chờ xoá bit (JNB ACCi, \$) của A.
- Bảng 16.6 giới thiệu một số lệnh xử lý bit khi ghép nối từ cổng P0 sang PA.

Bảng 16.6. Một số lệnh xử lý bit từ cổng Pi

Lệnh cho P0	Lệnh cho PA	Lệnh cho Pi	Lệnh cho PB
CLR P0.3	MOV DPTR, #PA CLR ACC.3 MOVX @DPTR, A	JNB P1.3, \$	MOVDPTR, #PB MOV A, @DPTR JNB ACC.3, \$
SETBP0.7	MOV DPTR, #PA SETB ACC.7 MOVX @DPTR, A	JNB P3.5, \$	MOVDPTR, #PB MOV A, @DPTR JNB ACC.5, \$

Chú ý rằng, trong xử lý bit này, tên cổng của VDK mắc trực tiếp không có trong chương trình mới.

3. Điều khiển rơ le qua cổng PC ở chế độ BSR

Rơ le là thiết bị dùng để đóng mạch điện có nhiều tiếp điểm. Rơ le có thể có loại một tiếp điểm (chuyển mạch), hai hay nhiều tiếp điểm. Rơ le có hai trạng thái thường đóng và thường mở. Điều khiển rơ le tức cung cấp cho cuộn dây điện từ một thế hay một dòng điện nhất định để có thể hút được cần tiếp xúc, khiến hai điểm thường hở đóng lại để tiếp xúc với nhau, cho một dòng điện (thường điện áp hay cường độ dòng điện lớn chạy qua).

Rơ le dùng trong hệ điều khiển dùng VDK thường nuôi với điện thế mức TTL và dòng điều khiển bé (cỡ chục mA). Người ta hay dùng vi mạch ULN2803 là những cổng logic công suất (dùng các mảng tranzito mắc kiểu darlington, cho dòng đi qua tới 500mA), có các diot chặn dòng cảm ứng của cuộn dây của rơ le. Mạch điều khiển 4 rơ le mắc qua vi mạch tăng dòng ULN2803. Mạch có nguồn nuôi các cuộn dây của rơ le có thể lên tới 50V. Mạch tiếp điểm của rơ le có thể là dòng một chiều, xoay chiều, có điện áp tới 220V và dòng tải lớn (cỡ chục Ampe).

Sơ đồ trên được cắm với lối ra của cổng PC của 82C55 ở chế độ BSR. Khi PCi được xác lập, có dòng vào cuộn dây, tiếp điểm thường mở đóng lại. Khi PCi bị xoá, dòng điều khiển ngắt, tiếp điểm lại mở ra.

Có thể điều khiển các rơ le trên bằng đưa các tín hiệu ra các cổng PA, PB hay PC ở chế độ M = 0, nhưng điều khiển rơ le ở chế độ BSR này tiết kiệm được các bit của cổng PC.

4. Điều khiển động cơ bước qua cổng PC ở chế độ BSR

a) Cấu tạo và hoạt động

Động cơ bước là động cơ quay từng bước với góc quay α dưới tác động của xung điện, tác động lên các cuộn dây tạo từ trường quay của stato. Động cơ bước có các cực là các nam châm điện từ với các cuộn dây và roto là một nam châm vĩnh cửu. Trường hợp đơn giản nhất là động cơ bước có 4 cực với 4 cuộn dây có các đầu nối A, B, C, D và hai đầu nối chung ở giữa (nối với đất hay nguồn nuôi). Các đầu cuộn dây A, B, C, D được nối với các bit chân ra của cổng song song của VDK qua bộ khuếch đại dòng điện bằng linh kiện rời (điện trở,

tranzito, diot) hay vi mạch chuyên dùng (như vi mạch 2803 dành cho động cơ bước công suất nhỏ).

Tùy trạng thái logic của các tín hiệu ở các đầu dây và độ kéo dài của xung, động cơ bước có thể quay theo chiều thuận (theo chiều kim đồng hồ) hay trái (ngược chiều kim đồng hồ) với các bước góc α khác nhau và số bước trên một vòng 360° khác nhau (bảng 16.9).

Mỗi bước của động cơ bước quay được một góc :

$$\alpha = 360^\circ / N_r \cdot N_p$$

với N_r — số răng của rôto

N_p — số pha của stato.

Với động cơ bước có 4 cuộn dây, điều khiển bằng xung đơn cực (theo mức logic 0, 1), số bước quay K trong một vòng 360° hay một chu kỳ cho là:

Với α : góc của bước, tính bằng độ (cho theo catalog).

n : hệ số (= 1. điều khiển cả bước, = 2. điều khiển nửa bước).

Vận tốc trung bình của động cơ bước trong thời gian t là:

với f là tần số dịch bước của xung điều khiển.

b) Điều khiển động cơ bước qua cổng PC ở chế độ BSR

Nếu đầu dây chung nối với nguồn nuôi 5V, các lối vào A, B, C, D (nối trực tiếp với các bit của cổng ra của VĐK) có các xung điện với điện thế 0V của logic 0.

Nếu đầu dây chung nối với đất, các lối vào A, B, C, D (nối với các bit của cổng ra của VĐK) có các xung với biên độ điện thế 5V của logic 1.

Tùy số cặp cực (Bắc – Nam) của stator hay số cặp cuộn dây, động cơ bước được điều khiển khác nhau và góc quay khác nhau.

Với động cơ một pha, các tín hiệu vào điều khiển như bảng 16.8 và góc của bước như bảng 16.7 và chu kỳ quay một vòng là 4 bước.

Bảng 16.7. Góc bước của động cơ bước

Góc của bước	Số bước/vòng
0,72	500
1,8	200
2,0	180
2,5	144
5,0	72
7,5	48
15	24

Bảng 16.8. Bảng điều khiển cho động cơ một pha 4 bước

Bước	Đầu dây A	Đầu dây B	Đầu dây C	Đầu dây D	Lời điều khiển
1	1 (H)	0 (L)	1 (H)	1 (H)	0Ah
2	0 (L)	1 (H)	1 (H)	0 (L)	06h
3	0 (L)	1 (H)	0 (L)	1 (H)	5h
4	1 (H)	0 (L)	0 (L)	1 (H)	9h

Với động cơ hai pha, các tín hiệu vào điều khiển như bảng 16.9 và chu kỳ quay là 8 bước.

Bảng 16.9. Bảng điều khiển cho động cơ hai pha 8 bước

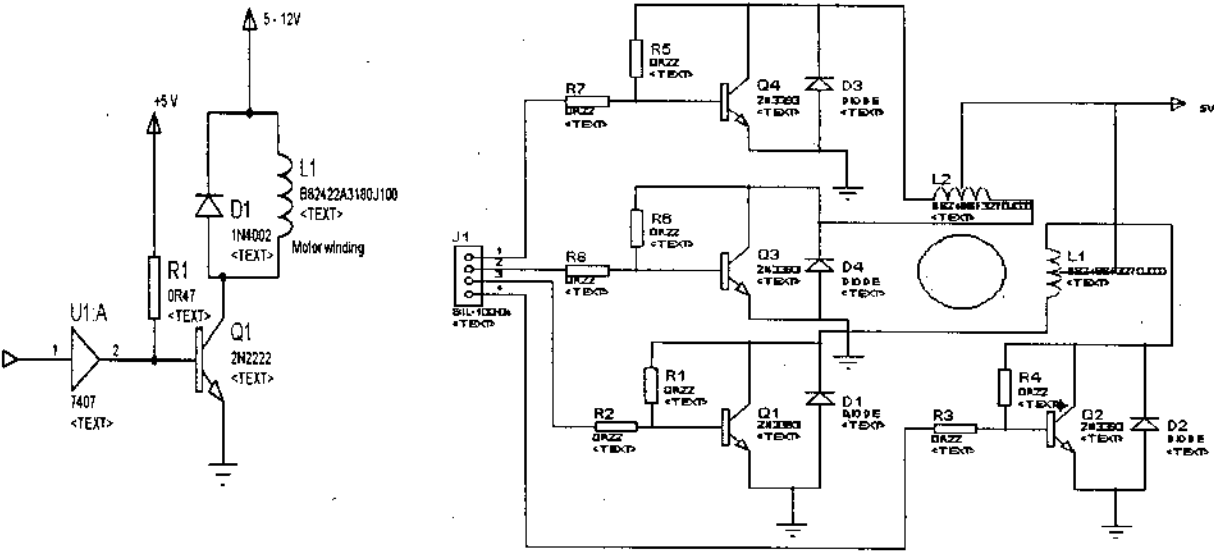
Bước	Đầu dây A	Đầu dây B	Đầu dây C	Đầu dây D	Lời điều khiển
1	1	0	1	0	0Ah
2	0	0	0	1	01h
3	0	1	1	0	06h
4	1	0	0	0	08h
5	0	1	0	1	05h
6	0	0	1	0	02h
7	1	0	0	1	09h
8	0	1	0	0	04h

Qua hai bảng trên, ta thấy bảng 8 bước có các bước chính như động cơ 4 bước và chèn thêm các bước phụ là 08h, 01h, 04h và 02h. Động cơ 8 bước này có ưu điểm là kéo được tải lớn hơn và có bước nhỏ và chính xác hơn.

Nếu bố trí các đầu dây khác đi, ta có bảng giá trị khác và đơn giản là bảng có các giá trị 8, 4, 2, 1.

c) Mạch điều khiển động cơ bước

Mạch điều khiển động cơ bước đơn giản nhất như hình 16.19.



Hình 16.19. Sơ đồ điều khiển động cơ bước đơn giản

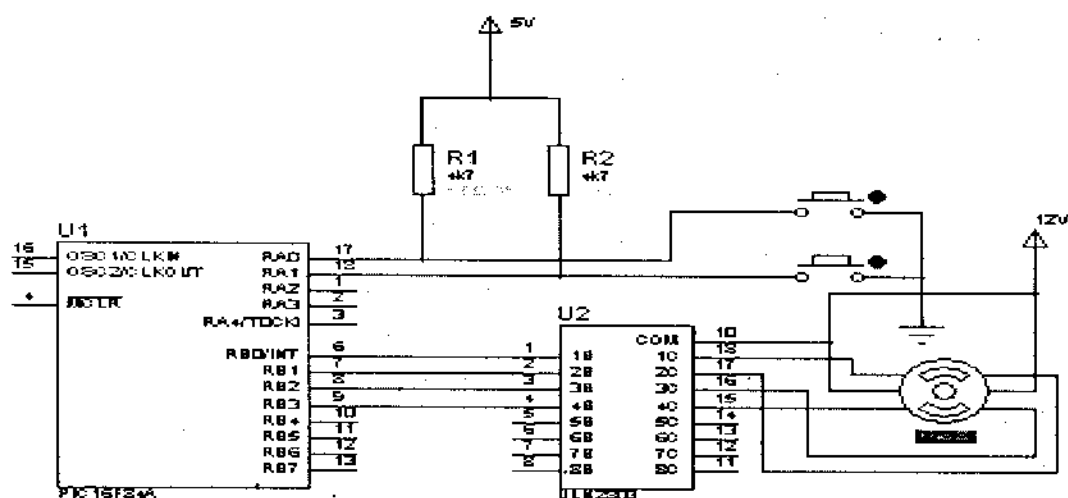
Ổ cắm 4 chân của mạch trên có thể cắm trực tiếp vào 4 chân của cổng ra của VDK hay vào 4 lối ra của các cổng PA, PB, PC ở chế độ M = 0 hoặc 4 chân của PC thấp (PC0–PC3) hoặc PC cao (PC4–PC7) của cổng PC của 82C55 chế độ BSR.

Sở dĩ có điện trở 1K, tranzito 2N222 và diot 1N 4001 trên vì:

- Cần hạn chế dòng điện vào cuộn dây của động cơ bước.
- Cần khuếch đại dòng để nam châm điện từ của cuộn dây đủ lớn để hút roto quay.

– Cần dập tắt dòng điện cảm ứng của cuộn dây khi ngắt mạch điện.

Trong mạch đơn giản trên, ta có thể thay các tranzito công suất và diot bằng vi mạch 2803 (hình 16.20).



Hình 16.20. Sơ đồ mạch điều khiển động cơ bước đơn cực với 2803

Nếu mắc động cơ bước qua vi mạch 2803 thì:

– 4 lối vào của 2803 nối với 8 bit D7–D0 của một cổng Pi.

4 lối ra của 2803 nối với 4 lối vào A, B, C, D và đầu chung của động cơ bước với nguồn điện nuôi. Các đầu A, B, C, D tương ứng với 4 bit thấp Pi.0, Pi.1, Pi.2, Pi.3 hay 4 bit cao Pi.4, Pi.5, Pi.6, Pi.7.

Hiện nay, người ta đã chế tạo các vi mạch chuyên dùng để điều khiển động cơ bước như SAA1027 hay cặp L297/L298 của hãng Thompson cho động cơ bước lưỡng cực.

d) Chương trình điều khiển động cơ bước với PC ở chế độ BSR

Chương trình điều khiển động cơ bước có thể có các loại chương trình sau:

– *Điều khiển động cơ bước mắc trực tiếp với cổng của VDK hay qua vi mạch 82C55:* Riêng với 82C55 có thể ở chế độ vào/ra thông dụng (M = 0) hay xác lập/xóa bit (BSR).

– *Điều khiển động cơ bước chạy tuần tự theo các trạng thái:* các trạng thái này được lập sẵn cho các đầu dây A, B, C, D (theo các bảng 16.8 và 16.9). Các trạng thái này có thể đưa trực tiếp theo lệnh trong đó có giá trị ban đầu và lệnh dịch thanh ghi A), đọc từ các ô nhớ RAM ghi trạng thái, từ bảng dữ liệu trong bộ nhớ PROM.

– *Điều khiển động cơ bước chạy tự động dừng, đảo chiều quay theo phím điều khiển:* nhấn phím điều khiển có thể thay đổi trạng thái chuyển động của động cơ bước.

– *Điều khiển động cơ bước chạy tự động dừng, đảo chiều quay:* với lệnh đếm và so sánh số bước, so sánh thời gian đặt sẵn, ta có thể điều khiển động cơ bước tự dừng lại, tự đảo chiều quay v.v...

Điều khiển động cơ bước chạy tự động theo trạng thái môi trường ngoài: với các tín hiệu ngoài đưa vào cổng vào qua cảm biến, ta có thể lập trình cho động cơ bước tự động chạy, dừng, đảo chiều quay v.v... Đây là trường hợp của chiếc xe tự hành thông minh của người máy (ROBOT).

Các chương trình trên có thể đồng thời hiện số bước, số vòng quay hay thời gian chạy, chiều quay v.v...

Các ví dụ của các chương trình điều khiển động cơ bước sau đây sẽ chọn cách mắc các sơ đồ điều khiển qua 82C55 và dùng cổng PC thấp ở chế độ BSR.

Ví dụ 16.12 giới thiệu lệnh gộp và chương trình điều khiển động cơ bước quay thuận và quay ngược loại đơn giản, có 4 pha với lời điều khiển quay thuận đưa vào các cuộn dây A, B, C và D.

; Ví dụ 16.12: Điều khiển động cơ bước theo 4 pha và dùng bảng dữ liệu điều khiển	
<pre> ORG 00H LJMP MAIN STSTOP EQU P3.2 THNG EQU P3.3 DCDK EQU 8003H ORG 4030H MAIN: MOV P3, #0FFH ; xác lập cổng lên 1 LAP: MOV R0, #00H JB STSTOP, \$ JNB STSTOP, KT JNB THNG, NGUOC THUAN: MOV DPTR, #STEP1 MOV A, R0 MOVC A, @A+DPTR MOV DPTR, #DCDK MOVX @DPTR, A CALL DELAY INC R0 MOV A, R0 CJNE A, #8, THUAN JMP LAP NGUOC: MOV DPTR, #STEP2 MOV A, R0 MOVC A, @A+DPTR MOV DPTR, #DCDK MOVX @DPTR, A CALL DELAY INC R0 MOV A, R0 CJNE A, #8, NGUOC JMP LAP KT: SJMP \$ DELAY: MOV R7, #5 DEL1: MOV R6, #200 DEL2: MOV R5, #200 DJNZ R5, \$ DJNZ R6, DEL2 DJNZ R7, DEL1 RET STEP1: DB 01H, 00H, 03H, 02H, 05H, 04H, 07H, 06H ; chạy thuận STEP2: DB 07H, 06H, 05H, 04H, 03H, 02H, 01H, 00H ; chạy ngược ; DB 88H, 88H, 22H, 11H ; chạy thuận ; DB 11H, 22H, 44H, 88H ; chạy ngược END </pre>	<pre> MOVC A, @A+DPTR MOV DPTR, #DCDK MOVX @DPTR, A CALL DELAY INC R0 MOV A, R0 CJNE A, #8, NGUOC JMP LAP KT: SJMP \$ DELAY: MOV R7, #5 DEL1: MOV R6, #200 DEL2: MOV R5, #200 DJNZ R5, \$ DJNZ R6, DEL2 DJNZ R7, DEL1 RET STEP1: DB 01H, 00H, 03H, 02H, 05H, 04H, 07H, 06H ; chạy thuận STEP2: DB 07H, 06H, 05H, 04H, 03H, 02H, 01H, 00H ; chạy ngược ; DB 88H, 88H, 22H, 11H ; chạy thuận ; DB 11H, 22H, 44H, 88H ; chạy ngược END delay: mov r5, #20H ; trê del1: ; lcall hthi mov r6, #250H del2: mov r7, #250H djnz r7, \$ djnz r6, del2 djnz r5, del1 ret end </pre>

Chú ý:

– Trong chương trình dùng hai phím P3.2 cho Start (nhấn phím) và Stop (nhả phím) và P3.3 cho quay thuận (không nhấn phím) và quay ngược (có nhấn phím) đưa vào hai lối vào ngắt ngoài nhưng không cho phép ngắt chương trình. Có thể dùng phím bất kỳ trên cổng PC, nhưng phải đọc vào thanh ghi chứa A rồi kiểm tra theo bit.

– Bảng dữ liệu trên cho PCi có một giá trị đơn lên 1, sau thời gian trễ DELAY lại xoá về 0. Nếu cần điều khiển hai dòng điện có trong động cơ, ta phải cho lần lượt hai PCi xác lập, sau đó xoá về 0 sau thời gian DELAY.

– Ta phải điều chỉnh độ kéo dài của xung điều khiển bằng thời gian trễ DELAY sao cho thích hợp vì nếu quá nhỏ, động cơ chưa kịp quay.

– Có thể đưa các chương trình con về hiển thị và giải mã ở trước hay trong DELAY để hiển thị số bước đã chạy của động cơ.

– Nếu động cơ có bố trí cuộn dây khác (đơn cực, lưỡng cực), ta chỉ cần thay dữ liệu của bảng dữ liệu.

5. Điều khiển phát xung qua cổng PC ở chế độ BSR

Trong chương 7 của tập I, chúng ta đã xét các bit của cổng ra của VDK còn sử dụng để phát các tín hiệu xung. Khi đó, các cổng ra đưa các tín hiệu ra 1 hay 0 với thời khoảng trễ DELAY cố định hay khác nhau. Như vậy, biên độ xung luôn là logic 1 (cỡ gần 5V) và logic 0 (cỡ gần 0V).

Ở một chân ra của VDK nếu ta cho mức tín hiệu trong thời gian DELAY, ta có xung đơn hay thay đổi tuần hoàn theo thời gian, ta có một xung vuông với chu kỳ là $2 \times \text{DELAY}$. Nếu cho thời gian trễ Delay1 và độ kéo dài Delay2 khác nhau, ta có một chuỗi xung hình chữ nhật có thời gian lập lại hay chu kỳ $t = \text{Delay1} + \text{Delay2}$ và tần số $N = 1/t$.

Nếu ở nhiều chân ra của VDK, ta cho mức tín hiệu thay đổi tuần hoàn theo thời gian, thì ta có nhiều xung vuông với chu kỳ là thời gian trễ DELAY1. Nếu các xung ra ở các chân có độ trễ tương đối với nhau là DELAY2, ta có chuỗi xung ra có cùng độ rộng là DELAY1 nhưng trễ nhau là DELAY2. Chuỗi xung trên (ví dụ 16.12) được sử dụng trong máy phân tích logic nhiều kênh. Với các chân ra và lập trình với độ trễ DELAYi khác nhau, ta có chuỗi xung ra mong muốn như:

- Chuỗi xung hình chữ nhật (có DELAY1 khác DELAY2).
- Chuỗi xung có số xung xác định (cần đếm và ấn định số xung phát ra).
- Chuỗi xung ra có độ rộng tăng dần (cần đếm và tăng dần thời gian trễ DELAY1 theo số xung n).

Chuỗi xung ra có độ rộng không đổi và trễ tăng dần (DELAY1 không đổi và cần đếm và tăng dần thời gian trễ DELAY2 theo số xung n) được dùng trong phép ghi nhớ dạng xung.

Với xung vuông có độ kéo dài bằng thời gian nghỉ, tần số N và độ kéo dài thay đổi theo công thức:

$$2 \times N \times (t_0) = 1 \text{ với } t_0 \text{ là độ kéo dài hay thời gian trễ giữa hai xung phát.}$$

Ta có thể phát được các âm thanh với tần số Ni tương ứng với các nốt nhạc (do, re, mi, fa, sol, la, si) cho các bát độ khác nhau và có thể phát được một bản nhạc mong muốn. Ví dụ nốt la có tần số 440Hz ứng với thời gian trễ là 1,13miligiây.

: Ví dụ 16.13: Fchx, chuỗi xung trễ liên tiếp	
<pre> macro @BSRPC <1.bitPCi ?, 2 lời? 3.tré? > mov DPTR, #1; địa chỉ th. ghi điều khiển mov a, #2 ; lời điều khiển movx @DPTR, A ; ghi lời chế độ BSR mov r7, #13 del: call delay1s djnz r7, del endmac ORG 00H lap: nop @BSRPC 8003h, 01h, 1 ; phát 1giây tại PC0 @BSRPC 8003h, 00h, 1 ; xoá 1giây tại PC0 @BSRPC 8003h, 03h, 2 ; phát 1giây tại PC1 @BSRPC 8003h, 02h, 1 ; xoá 1giây tại PC1 </pre>	<pre> @BSRPC 8003h, 05h, 3 ; phát 3giây tại PC2 @BSRPC 8003h, 04h, 2 ; xoá 2giây tại PC2 @BSRPC 8003h, 07h, 5 ; phát 5giây tại PC3 @BSRPC 8003h, 06h, 4 ; xoá 4giây tại PC3 ljmp lap ; lặp lại delay1s: mov r1, #2 ; trể 250 x 200 x 20 = 1 giây loop: mov r2, #20 loop1: mov r3, #20 loop2: djnz r3, loop2 djnz r2, loop1 djnz r1, loop ret end ; kết thúc dịch </pre>

6. Điều khiển động cơ một chiều theo kiểu số bằng PCi

a) Động cơ một chiều

Động cơ một chiều (DC – Direct Current – dòng một chiều) là động cơ được điều khiển bằng dòng điện một chiều. Động cơ một chiều là động cơ có các cuộn dây của cuộn rôto có dòng điện một chiều chạy qua, quay trong một từ trường cố định của một nam châm vĩnh cửu làm stato. Để đảm bảo chiều quay liên tục trong từ trường, dòng điện trong rôto phải luôn đổi chiều theo chiều của từ trường bằng các cổ góp điện hình vành khuyên. Trong một động cơ một chiều thực tế, có nhiều cuộn dây quấn xung quanh rôto, nối với nhiều thanh tiếp xúc đối xứng nhau, đặt xung quanh cổ góp. Tiếp xúc với thanh tiếp xúc này là hai chổi than, nối với hai cực dương và âm của một nguồn điện một chiều cố định.

b) Điều khiển động cơ một chiều

Trong tự động hoá, dùng động cơ một chiều công suất nhỏ, để việc điều khiển đơn giản. Có thể điều khiển động cơ một chiều này bằng một trong hai cách là dùng dòng điện một chiều hay chuỗi xung số.

– *Dùng dòng điện một chiều:* VĐK hay vi xử lý của máy vi tính, chỉ đưa ra các tín hiệu số rời rạc, do đó, phải có bộ biến đổi số – tương tự (DAC – Digital analog Converter) mắc giữa VĐK và động cơ một chiều.

– *Dùng chuỗi xung số:* chuỗi xung số đưa ra từ một bit của VĐK hay vi xử lý, tùy độ rộng (độ kéo dài xung) và độ rộng (thời gian nghỉ) của xung, điện thế ra trung bình trên một tụ điện C có điện dung lớn mắc song song với bit đưa chuỗi xung ra và đất, ta có điện thế trung bình khác nhau (hình 16.21). Từ đó, ta có hai cách điều khiển động cơ một chiều khác nhau:

+ *Điều khiển không đảo chiều quay:* động cơ chỉ nối với một bit của VĐK. Lối ra của bit này nối với cực dương của cuộn dây của động cơ, còn cực âm nối đất. Dòng điện trung bình của lối ra trên tụ điện C sẽ nuôi cuộn dây rôto của động cơ. Do đó, thường nối một bit của VĐK với động cơ qua một mạch (hay vi mạch) khuếch đại dòng (hay công suất).

+ *Điều khiển có đảo chiều quay:* động cơ phải nối với hai bit P0, P1 của VĐK. Hai bit

P0, P1 trên phải có giá trị 1, 0 khác nhau, để đảm bảo dòng điện một chiều chạy từ $P_i = 1$ sang $P_i + 1 = 0$.

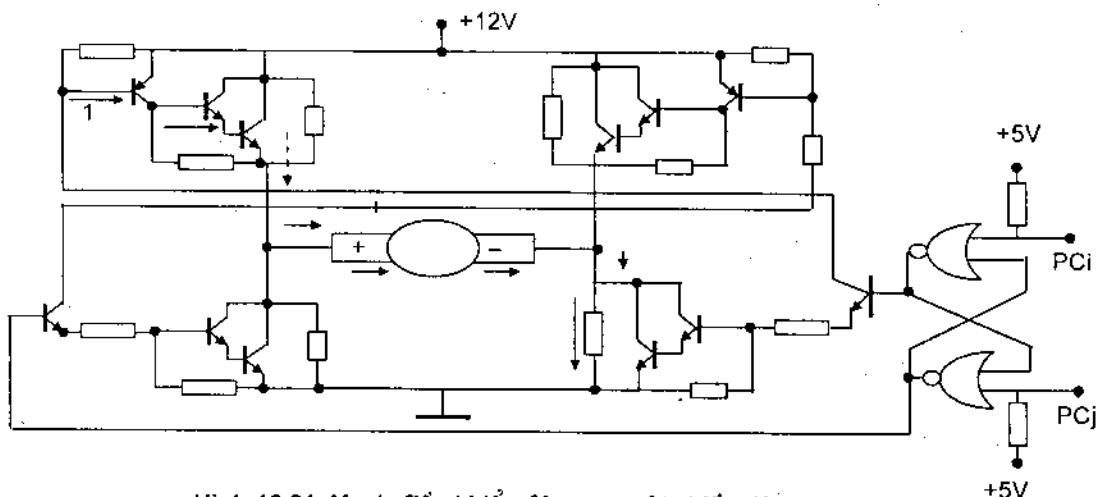
Với cách trên, ta có thể đảo chiều quay của động cơ bằng cách cho cặp giá trị P_i khác nhau (1,0 hay 0,1). Để đảm bảo tin cậy, người ta nối hai bit P0, P1 với một trigơ (flip flop) tạo bởi hai cổng logic hoặc với tín hiệu vào tích cực là 0 (như ở hình 16.21).

Tuỳ giá trị $PC_i = 1$ hay $= 0$ và $PC_j = 0$ hay 1 , luôn luôn có dòng điện đi qua động cơ nhưng ngược chiều nhau. Nói khác đi, ta có thể điều khiển chiều quay của động cơ bằng cách đưa ra $PC_i = 1$ (có dòng điện theo hình vẽ) hay 0 và $PC_j = 0$ hay 1 .

Tuỳ độ rộng xung t_0 hay thời gian nghỉ t_{ngh} hay tần số xung khác nhau, ta có tốc độ động cơ khác nhau.

Nếu trang bị cho một chiếc xe tự chạy có hai động cơ nối với hai bánh trước, hai bánh sau được kéo, đẩy theo, ta có thể điều khiển hai động cơ :

- Cả hai động cơ quay thuận theo chiều tiến: xe chạy tiến về phía trước.
- Cả hai động cơ quay ngược theo chiều lùi : xe chạy lùi về phía sau.
- Động cơ trái tiến, động cơ phải dừng: xe rẽ về phía bên phải.
- Động cơ trái dừng, động cơ phải tiến: xe rẽ về phía bên trái.



Hình 16.21. Mạch điều khiển động cơ một chiều dùng PC_i, PC_j

Tương tự, ta có thể cho xe lùi, rẽ trái, lùi, rẽ phải.

Ví dụ 16.14 giới thiệu lệnh gộp và chương trình điều khiển động cơ một chiều cho xe tự hành. Tương tự động cơ bước, ta điều khiển động cơ một chiều này theo bảng số cho các trường hợp xe tiến, lùi, rẽ trái, rẽ phải v.v...

Xe có thể:

- Không chạy: khi không có phím nào được nhấn.
- Chuẩn bị chạy: nhấn phím P3.2 ($P3.2 = 1$).
- Chạy thẳng, theo chiều tiến: khi phím nối với bit PC.0 được nhấn và cả hai động cơ cùng quay theo chiều thuận.
- Chạy thẳng, theo chiều lùi: khi phím nối với bit PC1 được nhấn và cả hai động cơ cùng quay theo chiều ngược.
- Rẽ trái theo chiều tiến: khi phím PC2 được nhấn và động cơ bên trái ngừng quay, còn động cơ bên phải quay theo chiều thuận.

– *Rẽ phải theo chiều tiến*: khi phím PC3 được nhấn khi động cơ bên phải ngừng quay còn động cơ bên trái quay theo chiều thuận.

– *Rẽ trái theo chiều lùi*: khi phím PC4 được nhấn, động cơ bên phải ngừng quay còn động cơ bên trái quay theo chiều ngược.

– *Rẽ phải theo chiều lùi*: khi phím PC5 được nhấn và động cơ bên phải ngừng quay còn động cơ bên trái quay theo chiều ngược.

Các phím trên được bố trí thành một bàn phím, đặt ở bên ngoài xe và nối với xe bởi 4 dây nối.

Ví dụ 16.14: Chương trình xe tự chạy có điều khiển bằng bàn phím	
<pre> MACRO @chay < 1. bang?, 2. bit điều khiến?> chay: mov dptr, #bang!2 ; 1. thuận, 2. ngược mov r1, #8h ; động c X jnb start, kt ; chờ nhấn để kết thúc lap1: mov a, #00h movc a, @a+dptr mov P0, a lcall delay inc dptr inc r0 jnb l1, menu djnz r1, lap1 ljmp chay kt: nop ENDMAC org 100h main: mov p3, #00h mov P1, #00h clr start menu0: jnb start, \$ mov c, PC0 jc ctien mov c, PC1 jc clui mov c, PC2 jc ctrai mov c, PC3 jc ctp hai mov c, PC4 jc tiep1 mov c, PC5 </pre>	<pre> jc tiep2 sjmp menu0 ctien: ; chạy tiến @chay P1.0, 1 clui: ; chạy lùi @chay P1.1, 2 ctrai: ; chạy tiến rẽ trái @chay P1.2, 3 menu: ljmp menu0 tiep1: ljmp cltrai tiep2: ljmp clphai ctphai: ; chạy tiến rẽ phải @chay P1.3, 4 cltrai: ; chạy lùi rẽ trái @chay P1.4, 5 clphai: ; chạy lùi rẽ phải @chay P1.5, 6 sjmp \$ delay: MOV R5, #20H ; trễ DELAY2 DEL1: MOV R6, #25H DEL2: MOV R7, #20H DJNZ R7, \$ DJNZ R6, DEL2 DJNZ R5, DEL1 CLR C RET bang 1: db 99h, 88h, 0aah, 0eeh, 66h, 77h, 55h, 11h bang 2: db 11h, 55h, 77h, 66h, 0eeh, 0aah, 88h, 99h bang 3: db 09h, 08h, 0ah, 0eh, 06h, 07h, 05h, 01h bang 4: db 90h, 80h, 0a0h, 0e0h, 60h, 70h, 50h, 10h bang 5: db 01h, 05h, 07h, 06h, 0eh, 0ah, 08h, 09h bang 6: db 10h, 50h, 70h, 60h, 0e0h, 0a0h, 80h, 90h end </pre>

Quá trình ngừng chạy động cơ khi:

- Các phím chế độ nhả (= 0), chương trình trở về chương trình chọn thực đơn.
- Phím P3.2 nhả (STOP), chương trình ngừng chạy.

Trong chương trình cần một số lệnh nhảy liên tiếp vì lệnh JC chỉ thực hiện trong khoảng –128 hay +128 địa chỉ ô nhớ.

16.4. TRAO ĐỔI TIN CỦA 82C55 Ở CHẾ ĐỘ NGẮT CHƯƠNG TRÌNH VÀ HỘI THOẠI (M = 1, M = 2)

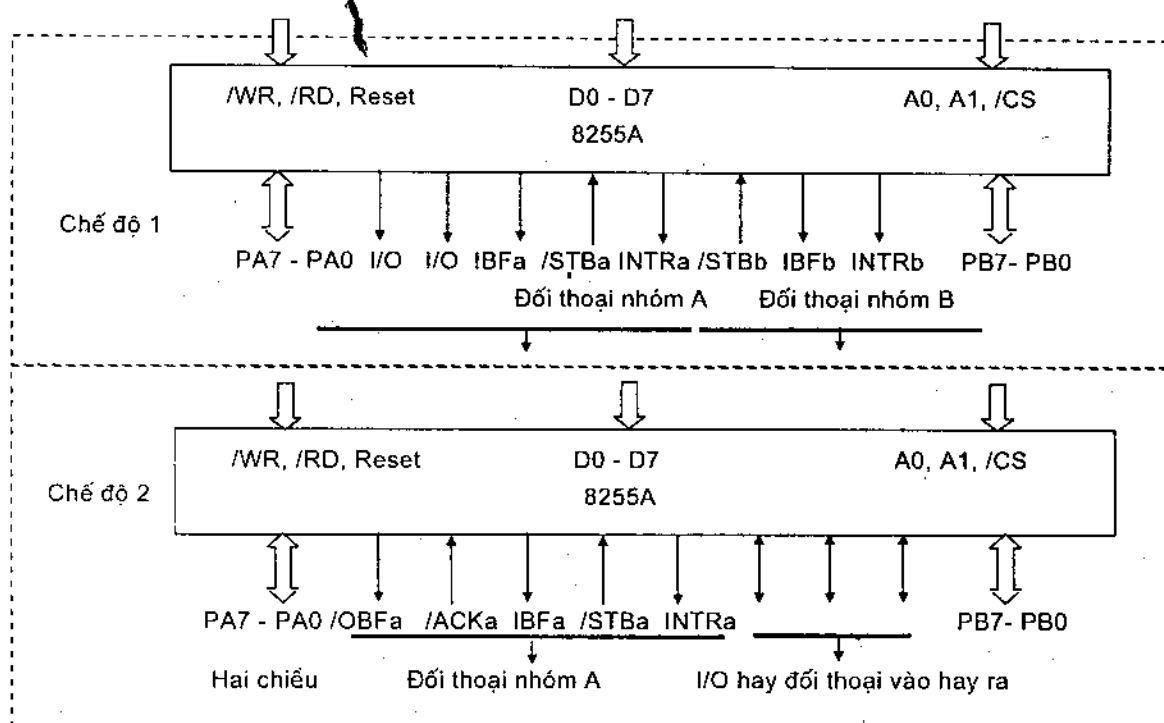
Tin trao đổi giữa VDK và thiết bị ngoài qua cổng song song 8255A ở chế độ cơ bản (M = 0) có nhược điểm là không tin cậy vì lúc trao đổi tin, VDK không biết có yêu cầu trao đổi tin hay không.

Để khắc phục nhược điểm trên, người ta sử dụng chế độ hội thoại (M = 1, 2). Trong chế độ này, tin trao đổi là tin cậy (không bị mất tin) theo cơ chế Yêu cầu (Hỏi) và Đáp ứng yêu cầu trao đổi tin (Đáp) hay theo chế độ ngắt chương trình.

16.4.1. Chế độ trao đổi tin ở chế độ hội thoại của 82C55

Tùy theo giá trị các bit thấp (từ D6 tới D0) của lời điều khiển (địa chỉ bằng địa chỉ cơ sở +3), vi mạch 8255A có các chế độ hội thoại (handshake – bắt tay) sau:

– *Chế độ M = 1*: cổng PA và PB trao đổi tin có hội thoại với VDK và thiết bị ngoài và dùng các bit của cổng PC làm tín hiệu hội thoại. PC thấp (PC0–PC3) cho cổng PB và PC cao (PC3–PC7) cho cổng PA* (hình 16.22).



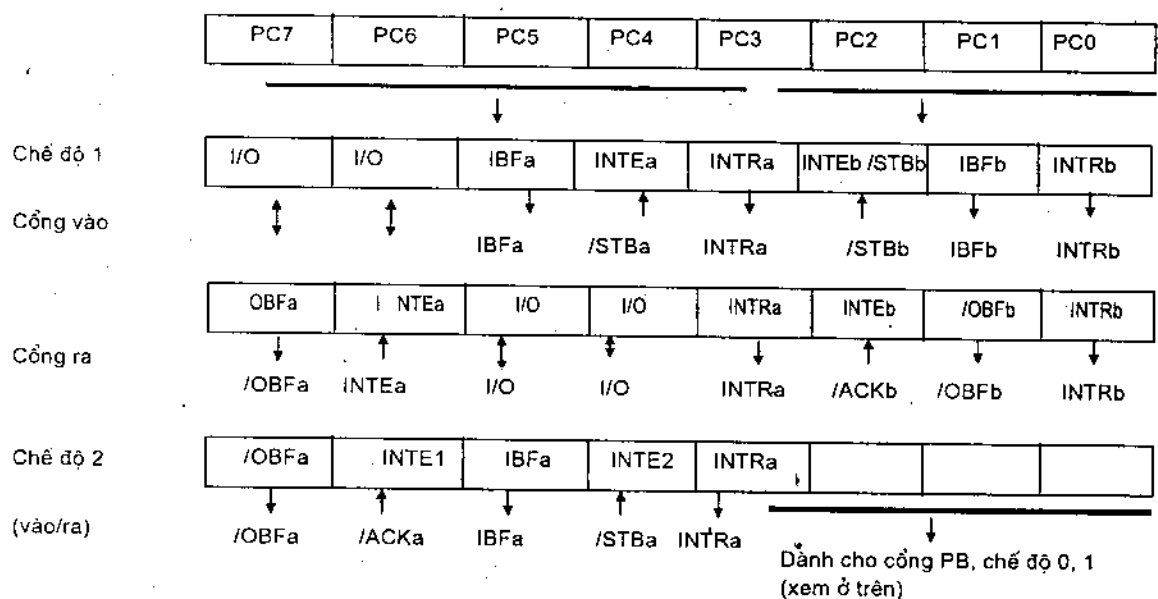
Hình 16.22. Các tín hiệu của các chế độ hội thoại

– *Chế độ M = 2*: cổng PA trao đổi tin theo chế độ hội thoại, hai chiều thuận nghịch (vào/ra) và PB có thể ở chế độ M = 0 hay 1.

Cổng PC ở chế độ hội thoại (M = 1, 2) được ghi lời trạng thái với các tín hiệu cụ thể như hình 16.23. Ta cần chú ý rằng:

– Ở chế độ M = 1: các bit PC6, PC7 không được dùng để hội thoại mà là các bit vào hay ra thông thường.

– Ở chế độ M = 2: các bit PC_i được sử dụng hết.



Hình 16.23. Thanh ghi trạng thái của các cổng PA, PB ở các chế độ hội thoại M = 1, 2

Tùy theo cổng là cổng vào hay cổng ra, ta có các bit trạng thái tương ứng khác nhau. Nhưng các bit Yêu cầu ngắt chương trình INTRa, INTRb luôn cố định ở PC3, PC0 và có chiều đi ra để nối vào các chân /INT0, /INT1 của VDK (hay INTR của vi xử lý).

Các bit trạng thái cho phép ngắt INTEa, INTEb, INTE1, INTE2 ở bên trong vi mạch, do các trigơ cho phép ngắt trên các bit tương ứng của PC khi ghi lời điều khiển có D7 = 0 vào thanh ghi điều khiển CWR.

Các tín hiệu IBFa, IBFb cho biết dữ liệu của bộ đệm vào của 8255A đã đầy (do thiết bị ngoài ghi vào bởi tín hiệu STA, STB).

Các tín hiệu /OBFa, /OBFb cho biết dữ liệu của bộ đệm ra của 8255A đã đầy (VDK đã ghi ra).

Người ta đọc các bit trạng thái của thanh ghi PC này để biết trạng thái của vi mạch, khi kiểm tra trạng thái ở chế độ trao đổi hội thoại theo kiểm tra trạng thái (hỏi vòng trạng thái), hay ngắt chương trình.

16.4.2. Trao đổi tin theo chế độ ngắt chương trình

Ở chế độ ngắt chương trình này, phải nối các tín hiệu Yêu cầu ngắt chương trình INTRa, INTRb với một trong các lối vào ngắt ngoài (/INT0, /INT1) của VDK (hay INTR của vi xử lý 8086).

Ở chế độ M = 1, có hai mức ngắt do cổng PA (INTRa nối với /INT0) và do cổng PB (INTRb nối với /INT1).

Ở chế độ M = 2, có hai mức ngắt đều do cổng PA, INTRa nối với /INT0 cho chiều đưa tin ra và vào.

Trước khi cho ngắt chương trình do cổng PA hay PB, ta phải khởi phát cổng bằng những lệnh sau:

- Ghi chế độ M = 1 (cho cổng PA hay PB) hay M = 2 (cho chỉ cổng PA).

– Ghi cho phép ngắt, tức các bit INTEa hay INTE1 (PC5) và INTE2 (bit PC4) cho cổng PA và INTEb (bit PC0) vào thanh ghi PC.

Sau đó chương trình có thể thực hiện nhiệm vụ khác. Khi có Yêu cầu trao đổi tin từ thiết bị ngoài, tức thiết bị ngoài đưa các tín hiệu /ACKa, /ACKb vào các bit PC6, PC2, 82C55 tự động phát các tín hiệu Yêu cầu ngắt chương trình INTRa, INTRb cho VDK (hay INTR của vi xử lý). VDK làm thủ tục ngắt chương trình rồi trao đổi tin với thiết bị ngoài qua các cổng PA, PB tương ứng với tín hiệu Yêu cầu ngắt chương trình đã gửi trước đó.

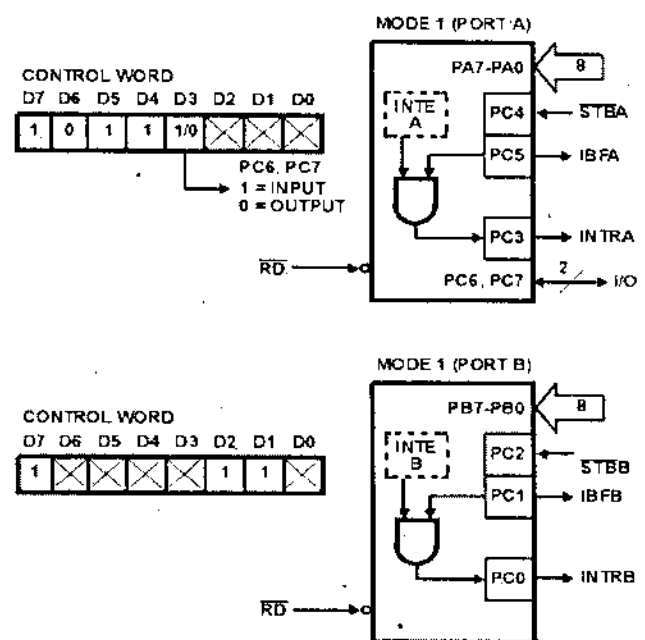
16.4.3. Trao đổi tin theo kiểm tra trạng thái

1. Đưa tin vào ở chế độ hội thoại M = 1

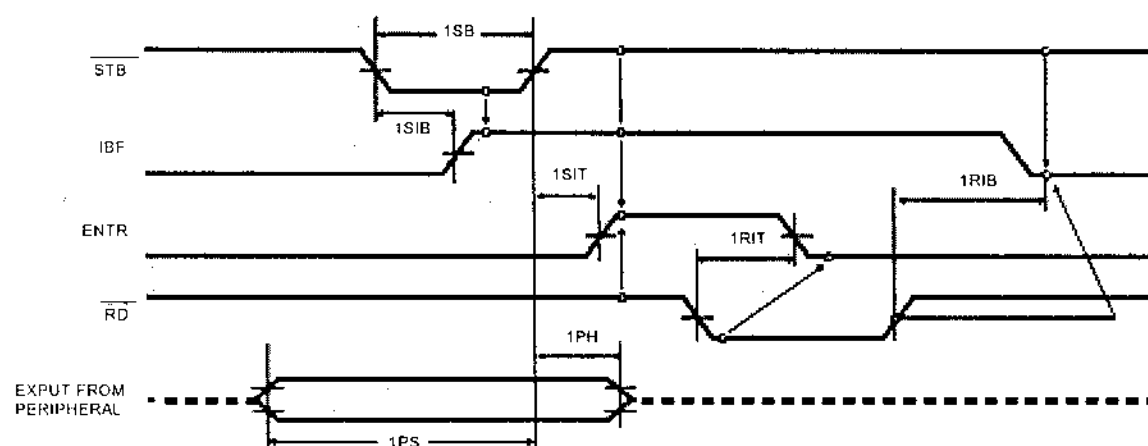
Ở chế độ này, có lời điều khiển và các tín hiệu hội thoại như hình 16.24.

Hình 16.25, giới thiệu giản đồ các tín hiệu theo thời gian ở chế độ đưa tin vào.

Thiết bị ngoài (TBN) đưa dữ liệu vào theo các cổng PA, PB và tín hiệu STBa hay STBb để chốt dữ liệu cho các cổng PA, PB. Đồng thời, PC5 xác lập lên 1 đưa ra TBN để báo ngừng đưa dữ liệu. Nếu trước đó INTEa, INTEb đã xác lập lên 1 để cho phép ngắt chương trình, hoặc VDK kiểm tra trạng thái ở bit PC1 hay PC5, VDK đưa lệnh đọc (/RD) đưa tin vào thanh ghi chứa A. Sau khi đọc tin về dữ liệu, bit IBFa, IBFb được xóa về 0, để yêu cầu TBN đưa tin vào tiếp.



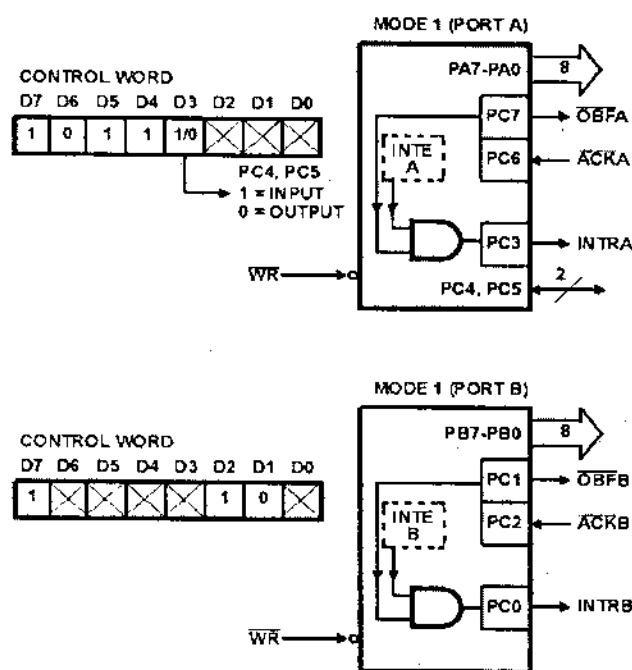
Hình 16.24. Đưa tin vào ở chế độ hội thoại M = 1



Hình 16.25. Giản đồ các tín hiệu khi TBN đưa tin vào ở chế độ hội thoại M = 1

2. Đưa tin ra ở chế độ hội thoại M = 1

Ở chế độ này, có lời điều khiển và các tín hiệu hội thoại như hình 16.26a.

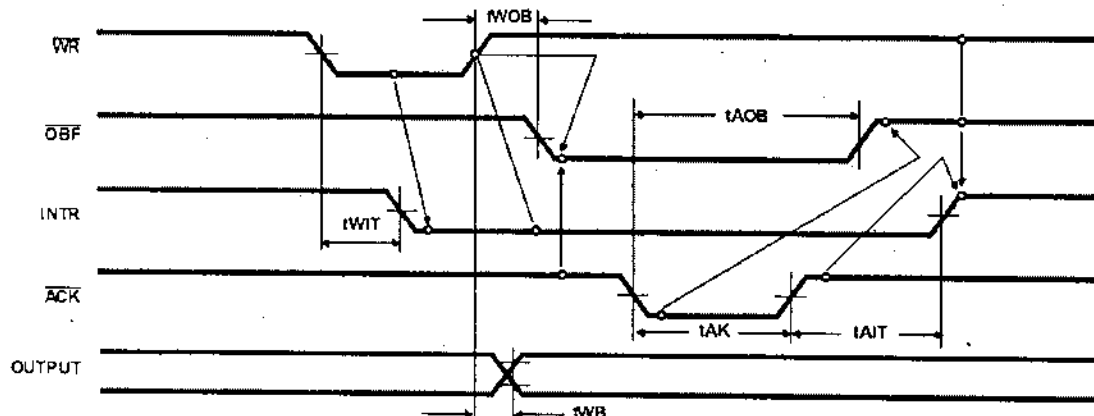


Hình 16.26a. Đưa tin ra ở chế độ hội thoại M = 1

Hình 16.26b giới thiệu giản đồ các tín hiệu theo thời gian ở chế độ đưa tin ra.

VĐK đọc tin vào thanh ghi A từ các thanh ghi cổng PA hay PB bởi lệnh ghi /WR. Khi ghi xong, xuất hiện tín hiệu /OBF báo cho TBN biết là bộ đệm ra đã rỗng (dữ liệu ra đã sẵn sàng, yêu cầu TBN đọc tin ra (có thể dùng tín hiệu /OBF để ghi tin cho TBN). Khi TBN ghi tin xong, phát ra tín hiệu trả lời /ACK là đã nhận tin xong. Nếu VĐK đã ghi INTEa, INTEb, tín hiệu Yêu cầu ngắt chương trình INTRa, INTRb được phát ra để yêu cầu VĐK đọc tin vào. VĐK có thể đọc và kiểm tra trạng thái /ACK để tiếp tục đưa tin ra.

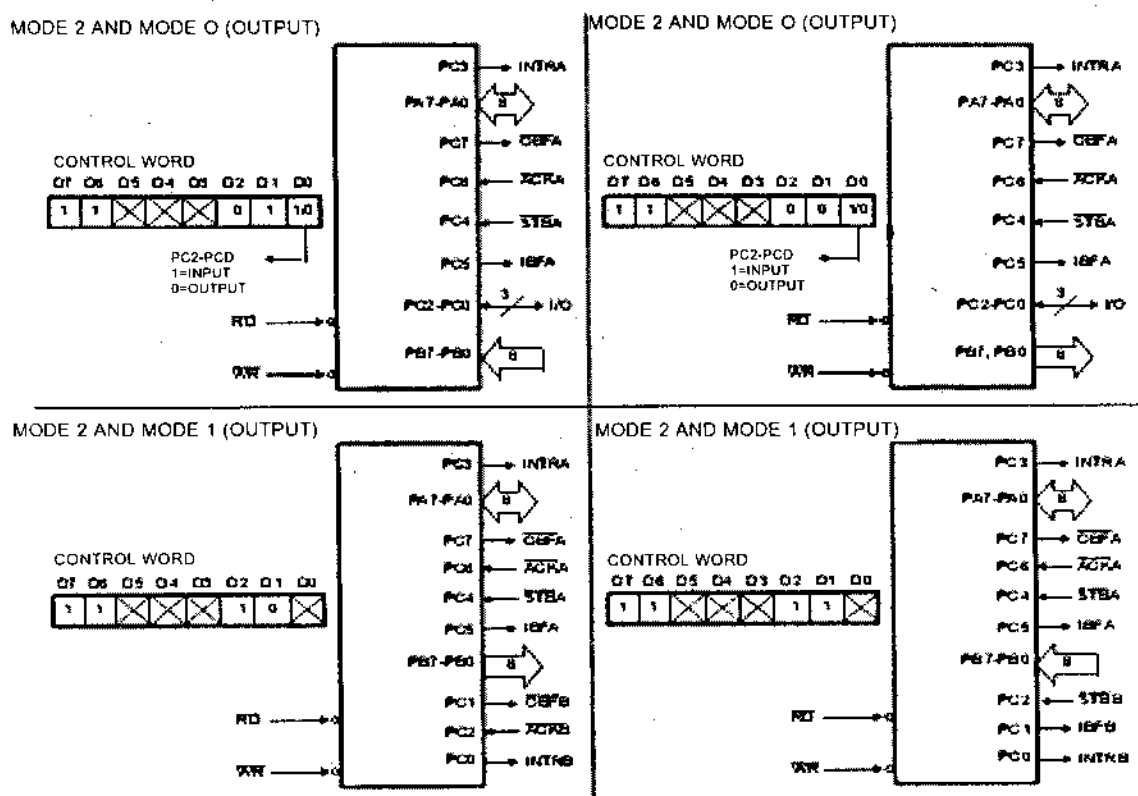
VĐK đọc trạng thái (bit PC6 cho PA hay PC2 cho PB xem có bằng 1 không, để đưa tin ra PA hay PB cho quá trình trao đổi tin tiếp sau.



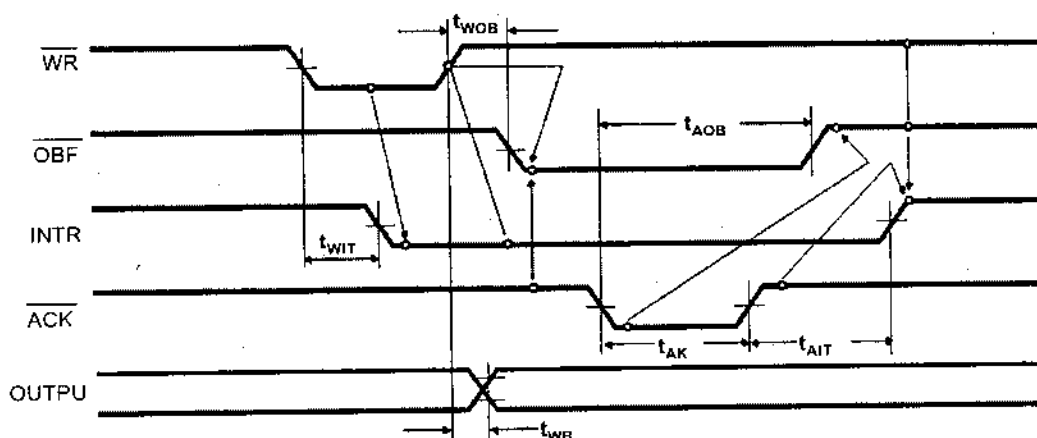
Hình 16.26b. Giản đồ các tín hiệu đưa tin ra ở chế độ hội thoại M = 1

16.4.4. Trao đổi tin theo kiểm tra trạng thái ở chế độ M = 2

- Ở chế độ M = 2, tương tự chế độ M = 1 nhưng:
 - Chỉ cho cổng PA.
 - Tin trao đổi là hai chiều (thuận nghịch trên một đường dây PA. Các tín hiệu hội thoại như hình 16.27 và giản đồ như hình 16.28).



Hình 16.27. Trao đổi tin ở chế độ M = 2



Hình 16.28. Giản đồ các tín hiệu đưa tin ra ở chế độ M = 2

16.5. TRAO ĐỔI TIN SONG SONG CỦA VDK QUA 82C55

VDK cũng cần trao đổi tin song song với máy vi tính, với máy in và bộ nhớ EEPROM trong máy nạp chương trình (PROGRAMMER) qua vi mạch 82C55.

16.5.1. Trao đổi tin với máy in

Ổ cắm của máy in có các tín hiệu như ở bảng 16.10.

Bảng 16.10. Các chân và địa chỉ truy cập ổ cắm của máy in LPT

Chân	Tên gọi	Truy cập	Hướng dữ liệu
1	STROBE	ĐCCS = 2, BIT 0	Lối vào/ra, đảo
2	D0	ĐCCS, BIT 0	Lối ra
3	D1	ĐCCS, BIT 1	Lối ra
4	D2	ĐCCS, BIT 2	Lối ra
5	D3	ĐCCS, BIT 3	Lối ra
6	D4	ĐCCS, BIT 4	Lối ra
7	D5	ĐCCS, BIT 5	Lối ra
8	D6	ĐCCS, BIT 6	Lối ra
9	D7	ĐCCS, BIT 7	Lối ra
10	ACK	ĐCCS + 1, BIT 6	Lối vào
11	BUSY	ĐCCS + 1, BIT 7	Lối vào, đảo
12	PE	ĐCCS + 1, BIT 5	Lối vào
13	SLCT	ĐCCS + 1, BIT 4	Lối vào
14	AF	ĐCCS + 2, BIT 1	Lối vào/ra, đảo
15	ERROR	ĐCCS + 1, BIT 3	Lối vào
16	INIT	ĐCCS + 2, BIT 3	Lối vào/ra
17	SLCTIN	ĐCCS + 2, BIT 3	Lối vào/ra, đảo
18 – 25	GND		

Quan sát bảng tín hiệu trên, ta thấy:

· Thanh ghi dữ liệu của máy in (địa chỉ cơ sở): 8 đường dữ liệu D0–D7 (chân 2–9 của ổ cắm DB25).

– Thanh ghi trạng thái (đưa ra từ máy in) với các tín hiệu sau:

- + /ERROR: máy in báo lỗi, chân 15.
- + SLCT: máy in báo đã được chọn, chân 13
- + PE: máy in báo hết giấy, chân 12.
- + /ACK: máy in báo đã nhận xong dữ liệu, chân 10.
- + /BUSY: máy in báo bận, chưa nhận dữ liệu, chân 11.

– Thanh ghi điều khiển với các tín hiệu sau:

+ /STRB (D0 – STROB) : để chốt dữ liệu, chân 1.

+ /AF (Auto Feed)–D1: điều khiển quay cuộn giấy, chân 14.

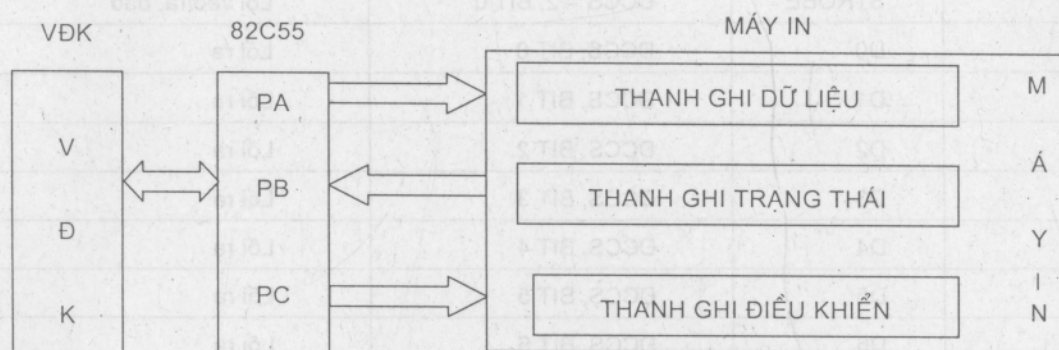
+ INIT (khởi phát)–D2: điều khiển máy in đóng nguồn điện nuôi, chân 16.

+ /SLCTIN (lựa chọn vào) –D3: điều khiển chọn máy in, chân 17.

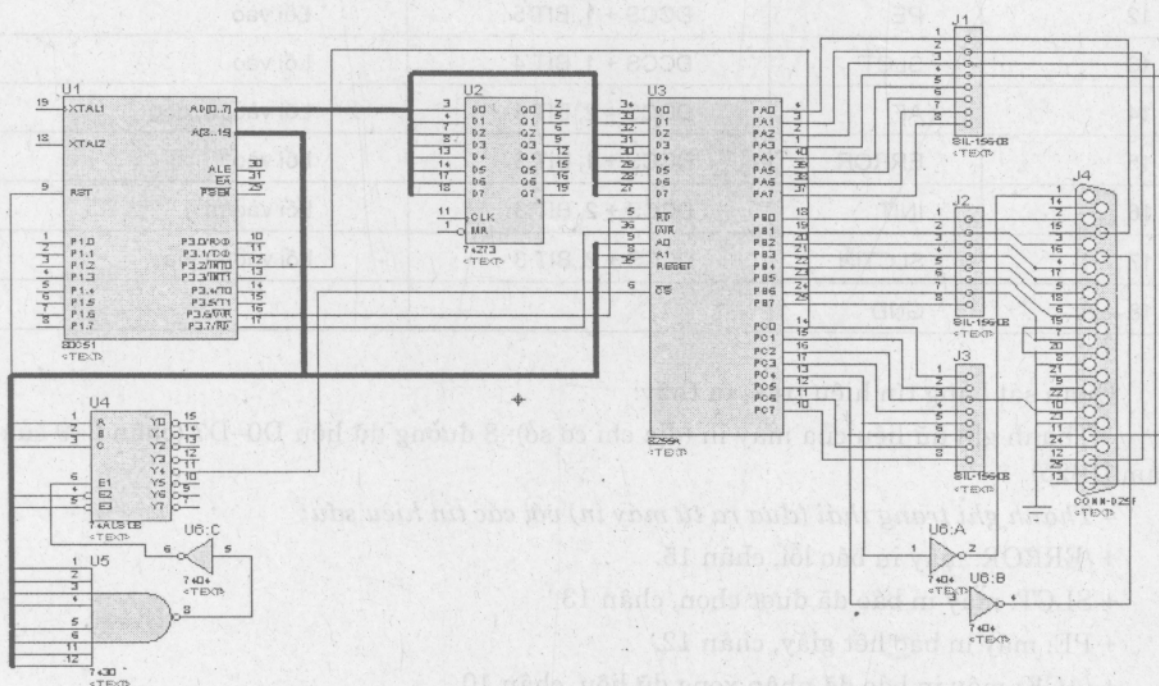
+ IRQ: D4, cho phép ngắt.

Sơ đồ khối của hệ điều khiển máy in của VĐK như hình 16.29 và sơ đồ nguyên lý như hình 16.30.

Giải thuật của chương trình điều khiển máy in như hình 16.31.



Hình 16.29. Sơ đồ khối của khối điều khiển máy in cho VĐK

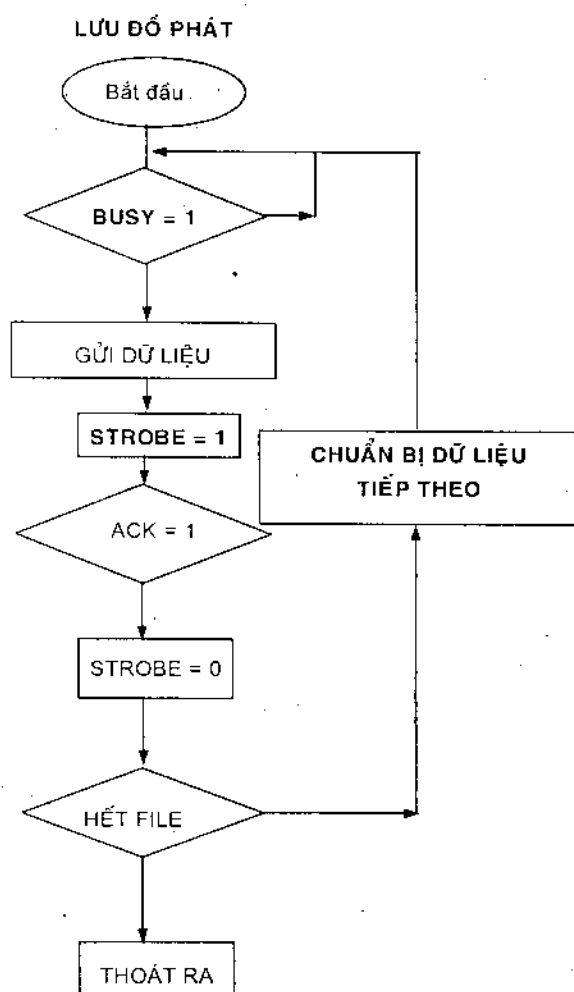


Hình 16.30. Sơ đồ nguyên lý của khối điều khiển máy in cho VĐK

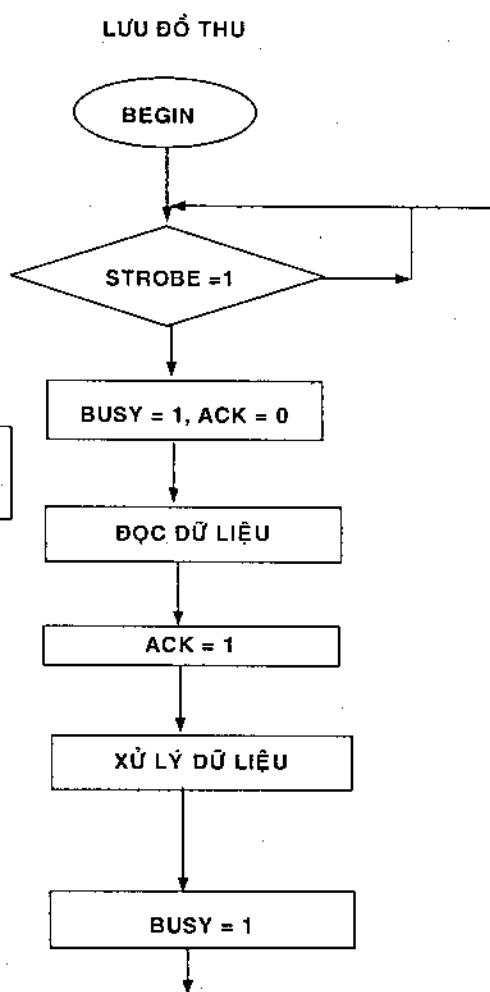
16.5.2. Trao đổi tin với máy nạp chương trình

Trong máy nạp chương trình song song, dùng hai vi mạch 82C55 để trao đổi tin. Dùng hai vi mạch 82C55 thì giải thuật của chương trình như hình 16.31. Còn giải thuật của chương trình xoá EPROM như hình 16.33.

So với máy nạp chương trình trao đổi tin nối tiếp (chương 9 tập I) thì máy nạp này có tốc độ nhanh hơn vì trao đổi tin song song và có hiển thị nội dung từng ô nhớ.



Hình 16.31a. Lưu đồ của phần phát dữ liệu



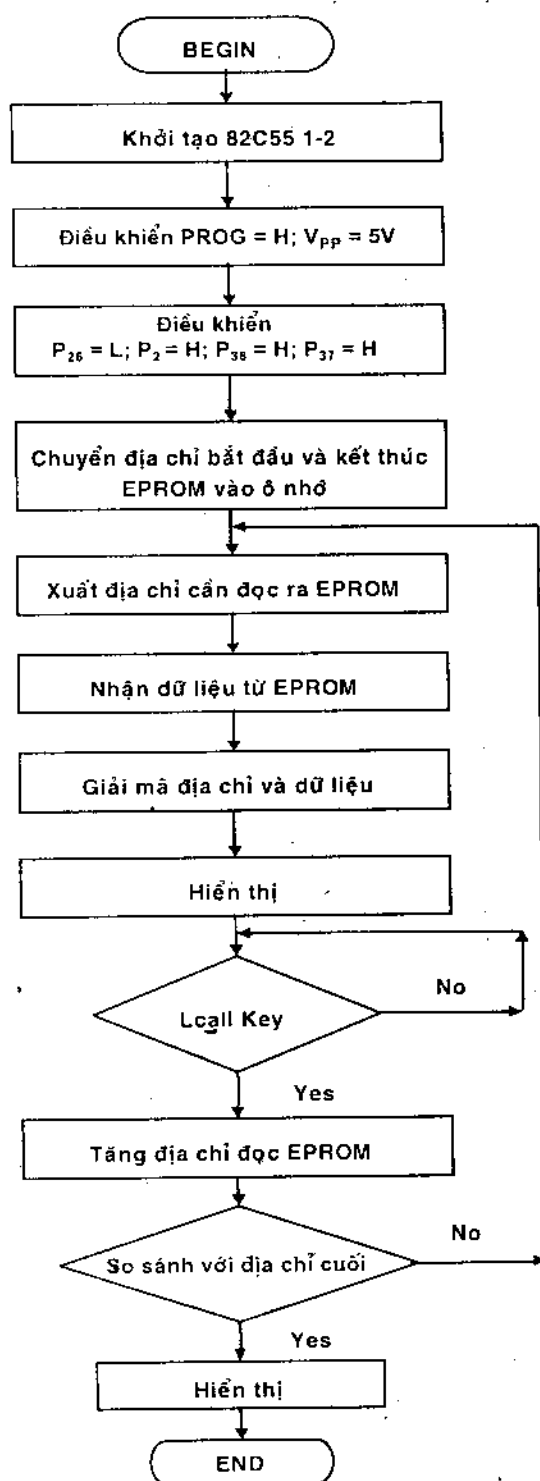
Hình 16.31b. Lưu đồ thu dữ liệu

16.5.3. Trao đổi tin với máy vi tính PC qua cổng máy in LPT

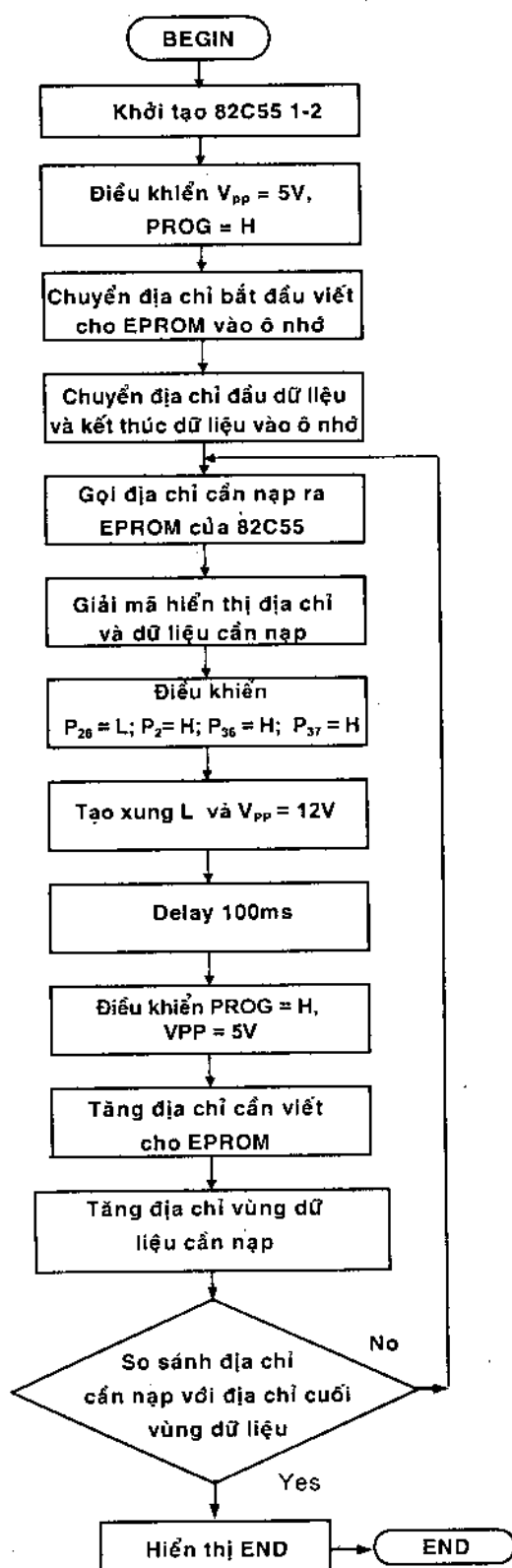
Ngoài trao đổi tin nối tiếp với máy vi tính PC qua cổng nối tiếp RS232, VDK còn có thể trao đổi tin song song với máy vi tính PC qua cổng máy in LPT. Vì đưa tin vào PC chỉ có 5 đường dây, trong khi cần đưa vào 8 bit tin. Do đó phải dùng giải pháp dồn kênh dùng vi mạch 74LS257 như sơ đồ trên hình 16.34 mắc giữa VDK 8051/52 và PC qua cổng LPT.

Giải thuật và chương trình trao đổi tin với PC của VDK cũng tương tự giải thuật và chương trình trao đổi tin với máy in (xem ở trên). Máy vi tính cũng phải có chương trình trao đổi tin tương ứng với hội thoại của chương trình trao đổi tin của VDK.

LƯU ĐỒ CHƯƠNG TRÌNH ĐỌC MÃ DỮ LIỆU



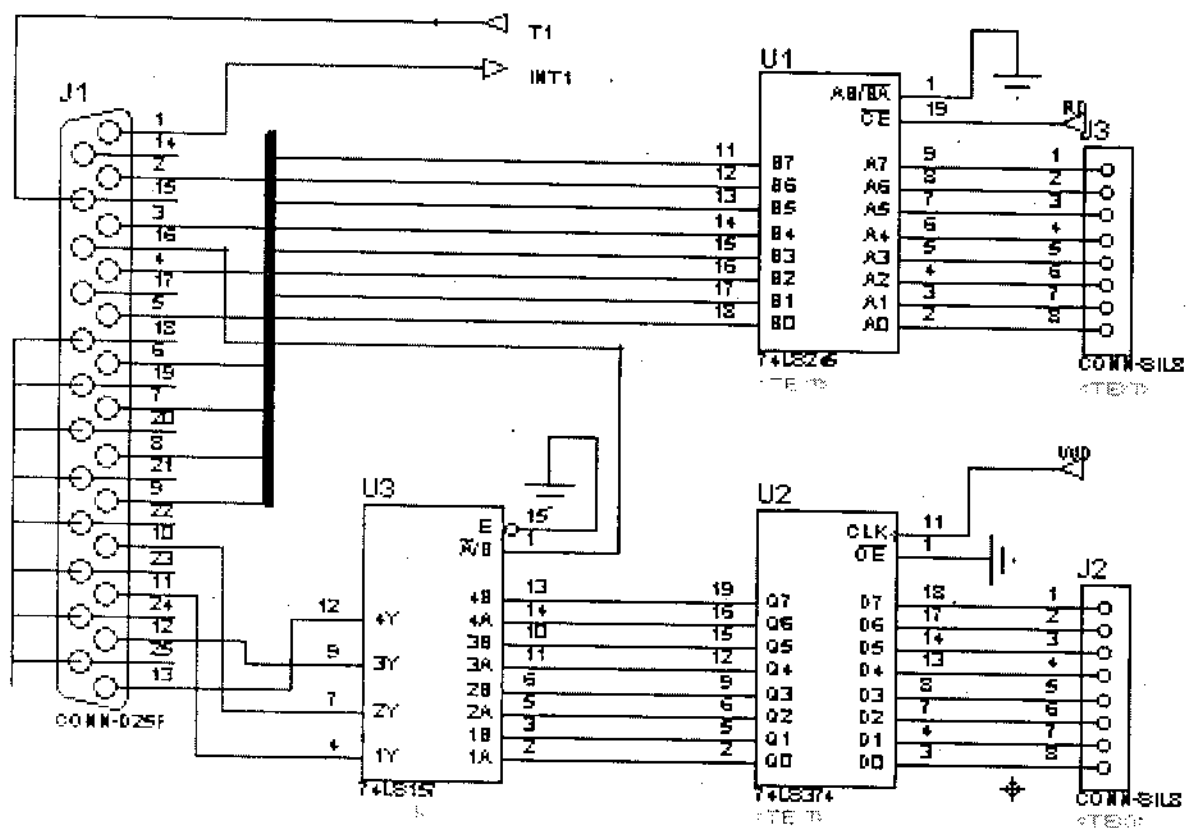
LƯU ĐỒ CHƯƠNG TRÌNH NẠP EPROM



Hình 16.32. Giải thuật của chương trình đọc và nạp EPROM



Hình 16.33. Lưu đồ chương trình xoá EPROM



Hình 16.34. Mạch trao đổi tin với máy vi tính PC

16.6. TRAO ĐỔI TIN SONG SONG CỦA VĐK QUA 82C55 VỚI THIẾT BỊ NGOÀI CÔNG SUẤT

Thiết bị ngoài công suất là các thiết bị yêu cầu dòng điện vào lớn (tối chục ampe), điện thế cao (tối trăm von).

Do đó, cần phải:

- Ghép nối với TBN qua bộ ghép nối quang, ví dụ, như vi mạch PC-807 và PD-827.
- Ghép nối với TBN qua mạch khuếch đại công suất với các tranzito công suất với thể nuôi lớn và dòng tải lớn (bảng 16.11).

Bảng 16.11. Bảng một số tranzito công suất

Loại	Thế cực đại	Dòng cực đại
2N3904	50V	200mA
ULN 2803	50V	500mA
TIP110	60V	2,0A
IRF 130	100V	14A
GT15H101		15A
TMG300J	600V	300A

Với động cơ bước công suất lớn, người ta phải dùng các vi mạch chế tạo sẵn như UCN5804B.

CÂU HỎI, BÀI TẬP VÀ THỰC HÀNH

I. Câu hỏi

1. Trình bày về các đặc điểm của các bộ đệm đường dây thay cho cổng song song. Phân biệt bộ đệm đường dây và thanh ghi.
2. Trình bày về các đặc điểm của các thanh ghi có thể dùng làm cổng song song.
3. Các thanh ghi trong một thiết bị ghép nối thiết bị ngoài.
4. Đặc điểm của vi mạch 82C55 so với các loại thanh ghi khác.
5. Các chế độ và các ứng dụng của 82C55 trong mỗi chế độ.
6. Cách ghép nối các thiết bị ngoài với các đường dây của một hệ vi điều khiển.
7. Vai trò của bộ giải mã trong một hệ VĐK. Các loại mạch giải mã dùng trong một hệ VĐK.
8. Các mạch ghép nối hiển thị (LED đơn, LED ma trận, LCD), bàn phím, động cơ bước, động cơ một chiều trong hệ VĐK dùng cổng mở rộng 82C55.
9. Cách ghép nối và trao đổi tin giữa một VĐK với máy in, bộ nạp chương trình và máy vi tính PC. So sánh sự giống và khác nhau giữa chúng.
10. Các cách ghép nối hệ VĐK với các thiết bị ngoài công suất cao.

II. Bài tập

1. Tra cứu và nghiên cứu các tài liệu về các thanh ghi đệm đường dây, thanh ghi đệm được dùng trong chương này, theo thư mục GOOLE.COM VN/ALL DATA SHEET trên mạng.
2. Tra cứu và nghiên cứu các tài liệu về 82C55 theo thư mục trên.
3. Dùng PROTEUS, thiết kế một sơ đồ trao đổi tin số đưa dữ liệu ra với các thanh ghi dữ liệu, trạng thái và điều khiển dùng các thanh ghi rời với địa chỉ cơ sở là 8000H.
4. Dùng PROTEUS, thiết kế một sơ đồ trao đổi tin số nhận dữ liệu vào với các thanh ghi dữ liệu, trạng thái và điều khiển dùng các thanh ghi rời với địa chỉ cơ sở là 8000H.
5. Dùng PROTEUS, thiết kế một sơ đồ trao đổi tin số thông dụng để đưa dữ liệu vào và ra với các thanh ghi dữ liệu, trạng thái và điều khiển dùng các thanh ghi rời với địa chỉ cơ sở là 8000H.
6. Dùng PROTEUS, thiết kế một sơ đồ trao đổi tin số đưa dữ liệu ra với các thanh ghi dữ liệu, trạng thái và điều khiển dùng vi mạch 82C55 với địa chỉ cơ sở là 8000H.
7. Tiến hành bài tập 6 với địa chỉ cơ sở là 0A000H.
8. Dùng PROTEUS, thiết kế một sơ đồ điều khiển động cơ bước đơn cực với công suất vừa (dùng tranzito công suất).
9. Dùng PROTEUS, thiết kế một sơ đồ điều khiển máy in dùng VĐK 89C51.

10. Dùng PROTEUS, thiết kế một sơ đồ trao đổi tin giữa VDK 89C51 với máy vi tính PC qua cổng máy in LPT2.

III. Thực hành

1. Dùng chương trình Pinnacle, soạn thảo, dịch và liên kết, cho chạy thử (từng bước và cả chương trình) và chạy trên kit có sẵn các ví dụ có trong chương.
2. Dùng chương trình Pinnacle, soạn thảo, dịch và liên kết, cho chạy thử (từng bước và cả chương trình) và chạy trên kit có sẵn các chương trình phục vụ trao đổi tin giữa hệ VDK 8051 với máy in.
3. Dùng chương trình Pinnacle, soạn thảo, dịch và liên kết, cho chạy thử (từng bước và cả chương trình) và chạy trên kit có sẵn các chương trình phục vụ trao đổi tin giữa hệ VDK 8051 với máy vi tính PC.
4. Cho chạy mô phỏng các chương trình có trong mục PROTEUS/Chạy mô phỏng với các chương trình sẵn có và viết mới trong PINNACLE.
5. Nghiên cứu và vẽ giải thuật một số chương trình có sẵn trong thư mục trên.

CHƯƠNG 17

GHÉP NỐI MỞ RỘNG CỔNG NỐI TIẾP CHO VI ĐIỀU KHIỂN

Mục đích của chương:

- Hiểu rõ yêu cầu ghép nối cổng nối tiếp mở rộng cho VDK.
- Hiểu rõ các loại cổng nối tiếp cho VDK như UART, I2C, USB, CAN và Profile BUS.
- Hiểu rõ thiết kế, lập trình và chạy mô phỏng cho một số cổng nối tiếp cho VDK.

Cũng như máy vi tính PC, VDK trao đổi tin theo dạng song song và nối tiếp với các thiết bị ngoài, trong đó có máy vi tính PC.

Ở chương 9 và 10 của tập I, chúng ta đã nghiên cứu trao đổi tin qua cổng nối tiếp không có ngắt và có ngắt cho VDK 8051/8052.

Ở chương này, chúng ta nghiên cứu cổng nối tiếp trao đổi theo UART cải tiến, SPI, I2C của VDK AT89C51RD2/ED2 thuộc họ MCS-51, cũng như trao đổi tin nối tiếp khác là ONE WAY (một đường hay một dây) và CAN.

17.1. TỔNG QUÁT VỀ ĐẶC ĐIỂM TRAO ĐỔI TIN NỐI TIẾP CỦA MÁY VI TÍNH PC

Máy vi tính PC trao đổi tin không đồng bộ UART theo các chuẩn RS (RS232C, RS 423, RS 422 và RS485) và đồng bộ theo các chuẩn SDLC và HDLC cũng như USB và IEEE1394 (FIRE WAY) (xem chương 5 và 10 của Kỹ thuật ghép nối máy vi tính tập một và tập hai của cùng tác giả).

17.1.1. Trao đổi tin nối tiếp theo chuẩn RS

1. Trao đổi tin nối tiếp theo chuẩn RS-232C

Trong máy vi tính PC-IBM, các vi mạch Intel 8251, 8250 hay 16450 đảm bảo trao đổi tin nối tiếp UART (Universal Asynchrone Receiver and Transmitter-truyền và nhận không đồng bộ) khi không có và có modem.

Những đặc điểm chính của RS232C (hay V24) như sau:

– Tin truyền theo các đường dây TxD và nhận RxD theo 8 bit của theo mã ASCII với 1 bit chẵn/lẻ P trong một khung có 1 bit Start (logic 0) và 1, 1 1/2 hay 2 bit Stop (logic 1). UART.

– Các mức tín hiệu là từ -3V tới -24V, +3V tới +24V cho chuẩn V.24 hay -3V tới -12V và +3V tới +12V cho RS232C, nên phải có vi mạch ghép nối để chuyển sang mức TTL (0V tới +5V) với vi mạch MC1489 (RS-232C sang TTL) và MC1488 (TTL sang RS - 232C) hay vi mạch MAX232, MAX233.

– Chân cắm của cổng nối tiếp và các tín hiệu (tín hiệu truyền TxD, tín hiệu nhận RxD, đất và các tín hiệu hội thoại) với modem.

– Khi không có modem, có sự nối tắt các tín hiệu DTR-DSR, RTS-CTS (hình 10-7 trang 62 của Kỹ thuật ghép nối máy vi tính tập hai).

– Tốc độ truyền (baud) có thể thay đổi từ 1200, 2400, 4800, 9600 hay 19200 baud.

Trao đổi tin theo RS232C theo ba đường dây (TxD, RxD và đất) tin cậy nhưng chậm, không truyền được xa (dưới 15m) vì nhiều và cần các vi mạch chuyển mức tín

hiệu TTL, (của PC hay thiết bị số) sang tín hiệu RS232C. Do đó, trong hệ trao đổi tin nhanh và xa, người ta dùng các chuẩn RS khác.

2. Các chuẩn RS khác

Các chuẩn RS khác khắc phục được nhược điểm của RS232C với các đặc tính như bảng 17.1.

Trong máy vi tính PC không có phần cứng phục vụ trao đổi tin theo các chuẩn này, mà phải lắp thêm vi mạch hoặc bìa (board) hay card, ví dụ card của hãng Quatech cho ghép nối trao đổi tin theo hai chuẩn RS 422 hay RS 485.

Bảng 17.1. Các đặc tính của các chuẩn RS

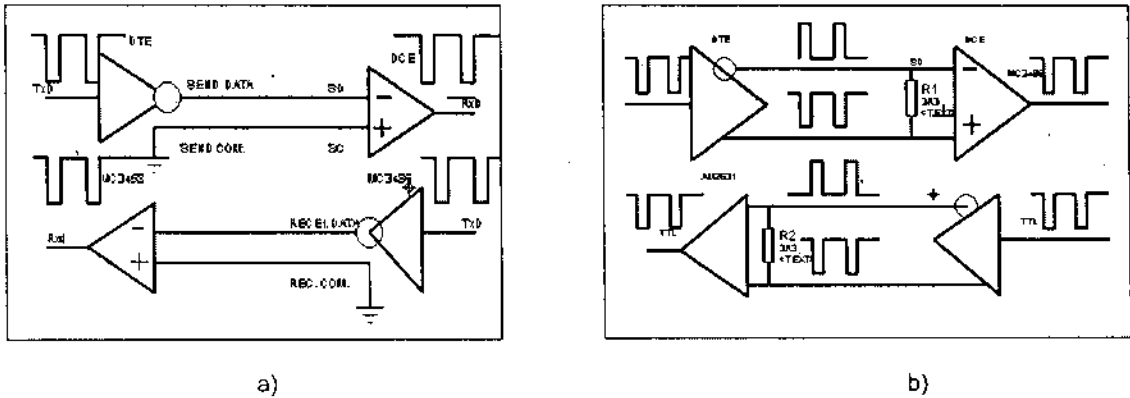
Các đặc tính		RS232	RS423	RS422	RS485
Chế độ hoạt động		Mắc không cân bằng	Mắc không cân bằng	Mắc cân bằng (vi phân)	Mắc cân bằng (vi phân)
Tổng số đường dây hay driver trên mạng (một driver tích cực ở một thời điểm cho mạng RS485)		1 DRIVER 1 RECVR	1 DRIVER 10 RECVR	1 DRIVER 10 RECVR	32 DRIVER 32 RECVR
Khoảng cách cực đại		50 FT.	4000 FT.	4000 FT.	4000 FT.
Tốc độ tối đa (40ft. – 4000ft. for RS422/RS485)		20kb/s	100kb/s	10Mb/s– 100kb/s	10Mb/s– 100kb/s
Xung ra cực đại của driver		+/-25V	+/-6V	-0,25V đến +6V	-7V đến +12V
Mức tín hiệu ra cực tiểu của driver (Loaded Min.)	Có tải	+/-5V đến +/-15V	+/-3,6V	+/-2,0V	+/-1,5V
Mức tín hiệu ra cực đại của driver (Unloaded Max)	Không tải	+/-25V	+/-6V	+/-6V	+/-6V
Điện trở tải của driver (Ohms)		3k đến 7k	>=450	100	54
Dòng max. của driver khi trở kháng cao	Bật nguồn nuôi	N/A	N/A	N/A	+/-100µA
Dòng max. của driver khi trở kháng cao	Tắt nguồn nuôi	+/-6mA @ +/-2V	+/-100µA	+/-100µA	+/-100µA
Tốc độ biến đổi (Max.)		30V/µs	Adjustable	N/A	N/A
Dải điện thế tín hiệu vào nhận		+/-15V	+/-12V	-10V đến +10V	-7V đến +12V
Độ nhạy vào nhận		+/-3V	+/-200mV	+/-200mV	+/-200mV
Điện trở vào nhận (1 tải chuẩn cho RS485)		3k đến 7k	4k min	4k min	>=12k

Hình 17.1a và 17.1b giới thiệu hai cách mắc của hai mạch trao đổi tin không cân bằng và cân bằng (hay vi phân) của các chuẩn RS khác với các vi mạch ghép nối dùng các vi mạch khuếch đại thuật toán.

Trong cách mắc không cân bằng, đường dây thứ hai của tín hiệu được nối đất của nguồn phát. Do đó có nhiễu trên đường truyền. Có thể khử được một phần của nhiễu này nếu dùng dây xoắn hay cáp bọc kim, nhưng không loại trừ được triệt để nên không thể

truyền tín hiệu đi xa. Tốc độ truyền tin trên các đường dây không cân bằng này còn chậm vì sự thay đổi tín hiệu logic 1 và 0 quá lớn (chênh lệch 48V hay 24V).

Trong cách mắc cân bằng, đường dây thứ hai của tín hiệu được mắc với mạch đảo của bộ khuếch đại vi phân, do đó có thể khử được nhiễu ở hai đường dây truyền, hạ sự chênh lệch giữa hai tín hiệu xuống còn $\pm 6V$ nên có thể tăng tốc độ truyền như ở bảng 17.2. Đặc biệt chuẩn RS485, có thể mắc 32 đầu nhận với một đầu phát tín hiệu truyền. Người ta dùng đặc điểm này của RS485 để truyền tin nối tiếp trên mạng, giữa một máy tính PC với nhiều máy PC khác và các thiết bị ngoài khác mắc với PC.



Hình 17.1. Sơ đồ mắc không cân bằng (a) và cân bằng hay vi phân (b) của các chuẩn RS

17.1.2. Trao đổi tin nối tiếp theo chuẩn USB

Các máy tính Pentium V hiện đại đều có trang bị các cổng USB (đặt trước hay sau máy) để trao đổi tin nối tiếp theo gói, tốc độ cao với máy vi tính khác và các thiết bị ngoài (chuột, máy in, bộ nhớ ngoài dung lượng lớn tới GBYTE, máy ghi âm MP3, video v.v...) theo các phiên bản USB 1.0 và 2.0. Phiên bản USB 1.0 (1996) trao đổi tin theo hai tốc độ trung bình và thấp như bảng 17.2.

Bảng 17.2. Đặc tính của USB 1.0

	12 Mbps	1.5 Mbps
Cable	STP	UTP
Max. Cable length	5 meter	3 meter
Connector	A-Series or B-Series	
Max. amount of HUBs	5	
Max. amount of units	127	

Phiên bản USB 2.0 trao đổi tin theo ba tốc độ cực lớn (400Mbps), trung bình (12Mbps) và thấp (1,5Mbps).

Cáp nối của USB là cáp bọc kim, có 4 dây nối và nối với các đầu nối như ở bảng 17.3 với màu sắc khác nhau.

Bảng 17.3. Các tín hiệu và chân cắm của cáp USB

Signal	Color	Pin
+Data	Green	3
-data	White	2
VCC	Red	1
GND	Black	4

Như vậy, tin được truyền đồng thời trên hai đường dây +data (+D) và -data (-D) như RS285 và mức logic 1 vì phân được định nghĩa là $(D+) - (D-) > 200\text{mV}$ và mức logic 0 là $(D+) - (D-) < -200\text{mV}$. Đường dây dùng mã NRZI (Non Return Zero – không trở về mức 0: với giá trị 1 giữ nguyên mức điện áp truyền, giá trị 0 dẫn đến thay đổi mức điện áp truyền).

Khác với RS485 là đường dây USB còn truyền cả nguồn nuôi 5V và đất cho thiết bị ngoài.

Độ trễ tối đa từ đầu phát tới đầu thu là 70ns, cho phép truyền tối đa trên một đường dây cáp tốc độ cao dài 30m có 5 hub.

Vì máy vi tính chủ (điều khiển phát hay nhận tin) chỉ có hai đầu USB nên muốn nối nhiều thiết bị ngoài truyền tin theo USB, phải nối với các bộ nối mạch hub. Mỗi hub này cho phép nối với các thiết bị ngoài.

Ở các đài phát, các tín hiệu D+ và D- đều có điện trở tải $15\text{k}\Omega$ nối đất. Ở các đầu thu tín hiệu, có điện trở $1,5\text{k}\Omega$ nối với nguồn 3,3V như sau:

- + Truyền tốc độ cao: đường dây D+ nối với $1,5\text{k}\Omega$, đường dây D- để trống.

- + Truyền tốc độ thấp: đường dây D- nối với $1,5\text{k}\Omega$, đường dây D+ để trống.

Tin truyền theo các gói tin, trong các khung truyền 1ms. Như vậy:

- Với tốc độ thấp (1,5Mbps), mỗi bit tin có chiều dài là 666,7ns.

- Với tốc độ cao (12Mbps), mỗi bit tin có chiều dài là 83,33ns.

- Với tốc độ cực cao (480Mbps), mỗi bit tin có chiều dài là 125µs.

Tin truyền trên USB theo các gói tin, truyền bit thấp trước, với các vùng (field) sau:

- Sync: các gói tin phải bắt đầu bằng vùng đồng bộ (Sync) này. Vùng Sync có 8 bit, dùng để đồng bộ nhịp của bộ phát và thu. Hai bit cuối chỉ bắt đầu vùng PID.

- PID: PID dùng để nhận dạng cho gói ID, cho biết loại tin gì được gửi đi. PID có 4 bit cho 16 loại tin (bảng 17.4), theo 4 nhóm TOKEN (hỏi), Data (dữ liệu), Handshake (hội thoại hay bắt tay) và Special (riêng).

- ADDR: địa chỉ có 7 bit cho 127 địa chỉ thiết bị ngoài khác nhau vì địa chỉ 0 không có nghĩa.

- ENDP: vùng Endpoint (điểm cuối – địa chỉ nhớ dành cho thiết bị ngoài) có 4 bit, cho phép tối đa 16 endpoint. Với tốc độ thấp, chỉ có 2 bit cho tối đa 4 endpoint.

- RC: Cyclic Redundancy Checks – kiểm tra số dư theo vòng để phát hiện lỗi truyền (tương tự bit chẵn/lẻ P trong RS232C). Các gói Token có 5 bit CRC, trong khi gói data có 16 bit CRC.

- EOP: End of Packet – kết thúc gói. Chỉ thị bởi Single Ended Zero (SEO), gần bằng 2 thời gian của 2 theo sau bởi J thời gian 1 bit.

Từ các vùng trên, ta có các gói tin như hình 17.2.

Trong đó có các gói tin sau:

- Gói tin Token: có ba loại là:

- + In: thông báo cho thiết bị USB là máy chủ muốn đọc tin.

- + Out: thông báo cho thiết bị USB là máy chủ muốn gửi tin.

- + Setup: dùng để bắt đầu một cuộc truyền tin.

- Gói tin Data: truyền từ 0 tới 1023 byte, có hai loại là:

- + Data0

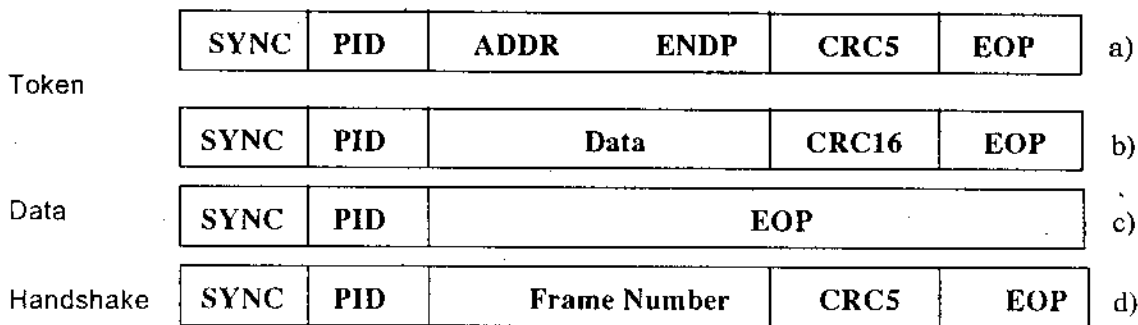
- + Data1

- Gói tin Handshake: có ba loại với vùng PID là:

- + **ACK**: thông báo là gói tin đã nhận không có lỗi.
- + **NAK**: thông báo là hiện thời thiết bị không thể gửi tin hay nhận tin. Khi tương tác ngắt, còn dùng để thông báo là máy chủ không có tin gửi đi.
- + **STALL**: thiết bị nhận thấy là nó đang ở trạng thái cần máy chủ can thiệp.
- *Gói tin Start of Frame Packet*: gói tin SOF này có khung 11 bit được gửi sau mỗi 1ms + 500ns

Bảng 17.4. Các giá trị của PID về các gói tin

Nhóm	Giá trị PID	Nhận dạng gói
Token	0001	OUT Token
	1001	IN Token
	0101	SOF Token
	1101	SETUP Token
Data	0011	DATA0
	1011	DATA1
	0111	DATA2
	1111	MDATA
Handshake	0010	ACK Handshake
	1010	NAK Handshake
	1110	STALL Handshake
	0110	NYET (No Response Yet)
Special	1100	PREamble
	1100	ERR
	1000	Split
	0100	Ping



Hình 17.2. Dạng các gói tin của USB

- USB định nghĩa có 4 cách trao đổi tin là:
- *Truyền tin theo điều khiển*: tin truyền do máy chủ khởi xướng.
 - *Truyền tin theo ngắt*: tin truyền do thiết bị ngoài yêu cầu máy chủ.
 - *Truyền đẳng thời hay đồng bộ cách biệt*: tin truyền theo thời gian xác định (cho video, máy nghe nhạc).

· *Truyền theo khối* (bulk): truyền hai chiều giữa máy chủ và thiết bị ngoài USB.

Trong máy vi tính đời mới đã có sẵn hai cổng trao đổi tin theo USB do chipset 400 của Intel đảm nhận. Nếu không, có thể dùng vi điều khiển tạo bộ ghép nối USB như ezUSB-52 dùng VDK 8052 (xem ở dưới).

Từ hệ điều hành Windows 95 cho PC đã có phần mềm phục vụ trao đổi tin theo USB và các thiết bị USB có khả năng cắm nóng (hot plugging) tức cắm hay loại bỏ trong khi máy vi tính PC đang chạy, không cần tắt nguồn nuôi. Hệ điều hành sẽ tự nhận biết thiết bị USB vừa cắm, gán địa chỉ và khởi động nó.

17.1.3. Trao đổi tin nối tiếp theo chuẩn IEEE1394

Chuẩn 1394–1995 hay FireWire ra đời (1995) bổ sung cho chuẩn USB để trao đổi tin ở tốc độ cực lớn, từ 100Mbps (S100), 200bps (S200) và 400bps (S400). Gần đây, tốc độ đã nâng lên tới 3,2 Gbps, khi dùng cáp quang.

Đầu nối của 1394 có 4 tiếp điểm, chia hai hàng tương ứng với 3 cặp dây cáp (nguồn nuôi–đất, tin truyền, tin nhận).

Đặc điểm chính của 1394 như sau:

- Có khả năng cắm nóng như USB.

Tốc độ từ thấp tới cao, có thể thay chuẩn SCSI cho các thiết bị nhanh như đĩa từ, hình ảnh động v.v...

- Không phân biệt thiết bị chủ hay tớ.

- Trao đổi tin theo gói (mỗi gói 125 microsec với địa chỉ 64 bit (10bit mã số của nút trong mạng, 6 bit mã số của nút – node) và 48 địa chỉ bộ nhớ). Cấu trúc địa chỉ này có thể quản lý được 1023 mạng, 63 điểm nối hay 63 thiết bị ngoài và 281 Terabyte nhớ.

- Trao đổi tin theo đẳng thời (định thời gian sẵn cho từng kênh) và không đồng bộ.

- Từ Windows 98 đã có phần mềm hỗ trợ trao đổi tin nối tiếp cao tốc theo 1394.

Phiên bản 1394.3 có thể truyền dữ liệu ngang cấp (peer to peer) cho máy quét ảnh, máy quay phim số đưa tin trực tiếp ra máy in, không cần qua máy tính chủ.

17.2. TRAO ĐỔI TIN NỐI TIẾP CỦA VDK MCS-51

17.2.1. Đặc điểm của trao đổi tin nối tiếp của VDK MCS-51

Ở chương 9 và 10 của tập một đã nghiên cứu cổng nối tiếp không có ngắt và có ngắt cho VDK 8051/8052. Ở đây chỉ tóm tắt các đặc điểm chính như sau:

- Mức của tín hiệu vẫn là mức TTL để tương thích với VDK. Do đó, nếu trao đổi tin nối tiếp với máy vi tính PC, phải mắc các lối vào TxD và RxD của VDK qua vi mạch MAX 232 hay mạch ghim điện áp 0V và 5V qua một diot.

- Khung tin trao đổi hoàn toàn theo chuẩn RS232C (mã ASCII với khung tin có 1 bit Start, một bit Stop).

- Có ba chế độ trao đổi tin: điều khiển bởi thanh ghi SCON với chế độ 1 (không bit chặn/lẻ), chế độ 2 (9 bit vì thêm bit chặn/lẻ) và chế độ 3 (thêm bit thứ 9 TB8 hay RB8).

- Có chế độ trao đổi tin giữa nhiều VDK: khi bit SM2 = 1 của SCON, có một VDK chủ và nhiều VDK tớ. Trong chế độ này, có truyền cả địa chỉ của VDK tớ (xem ở dưới).

17.2.2. Mở rộng cổng trao đổi tin nối tiếp của VDK MCS-51

Trong thực tế, đôi khi cần trao đổi tin nối tiếp của một VDK với nhiều thiết bị trao

– Dùng các lối ra của cổng song song: mỗi lối vào hay lối ra của cổng song song này là một đường truyền (RxC, TxD) dữ liệu nối tiếp. Khi đó, phải lập trình để đưa tin ra hay nhận tin vào nối tiếp, tương tự trường hợp mở rộng cổng song song.

2. Chương trình trao đổi tin nối tiếp

Chương trình phát (qua chân TxD) và nhận tin nối tiếp (qua chân RxD) như ở ví dụ 17.1.

Đặc điểm của chương trình của ví dụ 17.1 như sau:

VĐK tự phát các ký tự mã ASCII tăng dần từ 30H (số 0 thập phân) qua chân TxD sau mỗi thời gian trễ Delay 1sec dùng Timer0 ở chế độ 1.

- Tốc độ baud trao đổi tin nối tiếp do Timer1 ở chế độ 2 là 9600 baud với tần số nhịp 11,059MHz

VĐK thu các ký tự phát trên chân RxD khi nhận tín từ TxD và gây ra ngắt UARTInt do cờ RI = 1. Tín nhận vào từ bộ đệm SBUF chuyển sang thanh ghi A, đảo và đưa ra hiển thị ở cổng P1.

Như vậy, chương trình sẽ phát các mã ASCII tăng dần (ghi vào ô nhớ đệm Buffer có địa chỉ 30H), có thể quan sát tín phát ở TxD bởi thực đơn VIEW/UART của Pinnacle và quan sát tín hiệu nhận ở RxD trên cổng P1.

Nếu chạy mô phỏng trong chương trình mô phỏng Pinnacle, ta quan sát được tín hiệu nhận trên cổng P1 khi cho phép nhận, tức kích cho cờ nhận RI = 1.

3. Chạy mô phỏng chương trình

Nếu chạy mô phỏng với sơ đồ trên hình 17.3, ta mới quan sát được tín hiệu nhận trong thanh ghi A hay cổng P1. Khi đó, các đèn LED sẽ chỉ thị các mã ASCII đã nhận qua RxD do phát đi từ chân TxD. Chạy mô phỏng chương trình ở ví dụ 17.1 theo trình tự sau:

- Vào chương trình mô phỏng PROTEUS.
- Vào sơ đồ của hình 17.3.
- Nhập chương trình dạng .HEX của ví dụ 17.1.
- Chạy chương trình bằng nút nhấn tam giác của thanh công cụ số 4.
- Quan sát các tín hiệu trên các LED mắc với cổng P1.

: Ví dụ 17.1: Truyền nhận nối tiếp một VĐK	
Buffer equ 30h org 00h jmp Start org 023h jmp UARTInt Start: ; dùng Timer0 Mode 1, Timer1 (for UART) mode 2 mov tmod, #00100001b ; baud rate 9600 gbit/sec @11,059MHz anl pcon, #01111111b mov th1, #0fdh setb tr1 ; serial mode 1 \$ receive enable mov scon, #01010000b ; enable UART interrupt mov ic, #10010000b	push acc setb rs0 clr rs1 jbc ri, ReceiveInt clr ti jmp Return ReceiveInt: mov a, sbuf cpl a mov p1, a Return: pop acc pop psw reti ; Delay Time = (R5)*50ms (fx = 12MHz) Delay: mov th0, #high (65536-50000) mov tl0, #low (65536-50000)

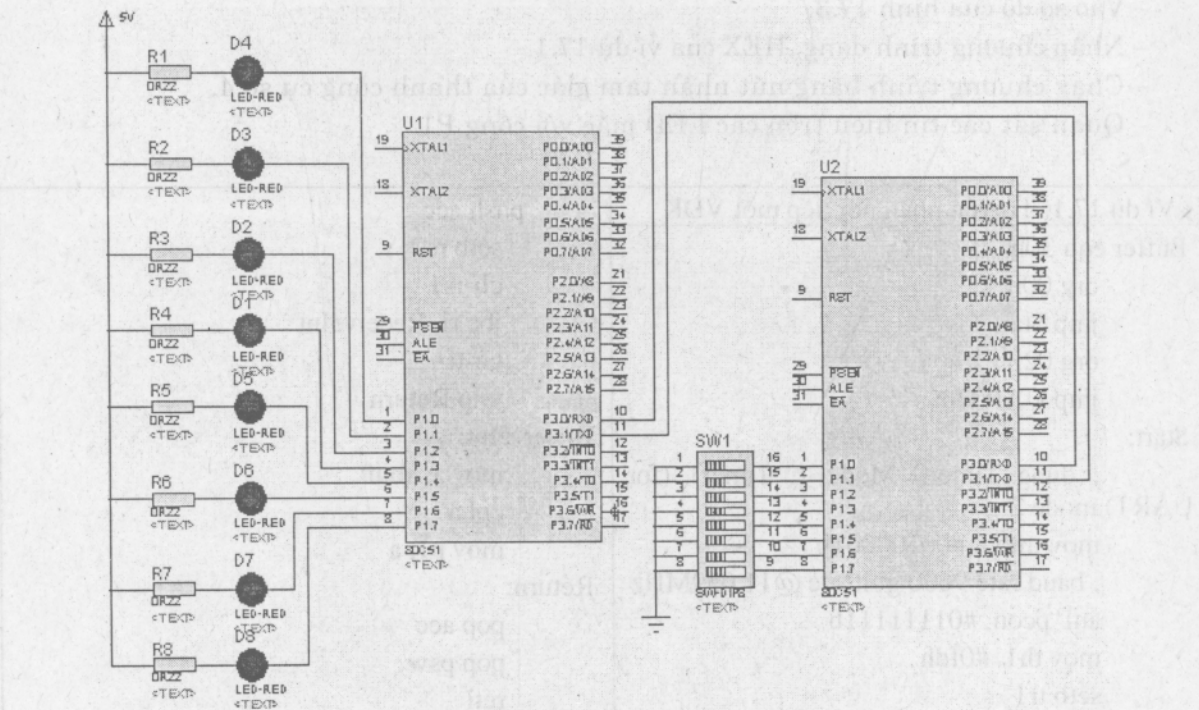
<pre> mov Buffer, #30h Loop: mov sbuf, Buffer mov r5, #20 ; delay 1 sec call Delay inc Buffer jmp Loop UARTInt: push psw </pre>	<pre> setb tr0 ; start Timer Wait: jbc tf0, TimeOut jmp Wait TimeOut: djnz r5, Delay clr tr0; stop Timer ret end </pre>
--	---

17.2.4. Thiết kế, lập trình và chạy mô phỏng trao đổi tin nối tiếp giữa hai VĐK

1. Thiết kế mạch

Mạch trao đổi tin nối tiếp giữa hai VĐK được thiết kế trong PROTEUS như hình 17.4. Đặc điểm của mạch như sau:

- VĐK ở phía trên có cổng P1 được nối với các LED để chỉ thị tin nhận từ chân RxD (giống hình 17.3).
- VĐK ở phía dưới có cổng P1 nối với các chuyển mạch để tạo các tín hiệu vào logic 0 (chuyển mạch nối đất) hay logic 1 (chuyển mạch ngắt nối đất) để đưa ra phát nối tiếp ở chân TxD.
- Chân TxD của VĐK phía dưới nối với chân RxD của VĐK phía trên.
- Chân TxD của VĐK phía trên nối với chân RxD của VĐK phía dưới.

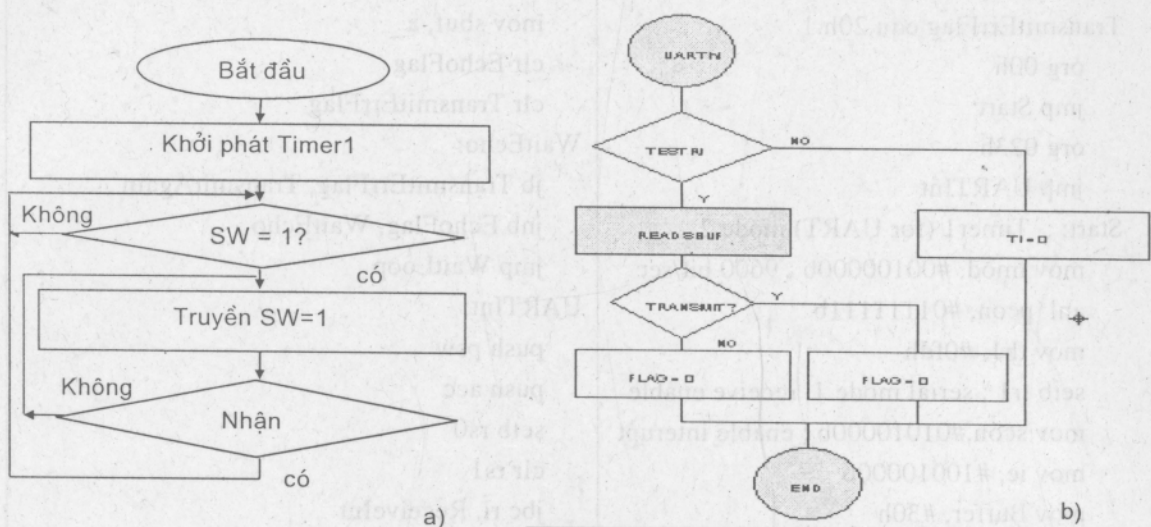


Hình 17.4. Sơ đồ trao đổi tin nối tiếp hai VĐK

Sơ đồ trên hình 17.4 đảm bảo tin từ hệ chuyển mạch mắc với cổng P1 của VĐK phía dưới đưa ra chỉ thị trên các LED mắc với cổng P1 của VĐK phía trên, sau khi đã truyền nối tiếp qua các chân TxD dưới lên RxD trên.

2. Chương trình truyền tin từ VDK phía dưới

Giải thuật của chương trình phục vụ truyền tin ở VDK phía dưới cho VDK phía trên (ví dụ 17.2) ở hình 17.5.



Hình 17.5. Giải thuật của chương trình truyền tin (a) và ngắt (b) của ví dụ 17.2.

Chương trình của ví dụ 17.2 nạp vào VDK phía dưới. Chương trình chính của ví dụ 17.2 này có giải thuật (a) và có các đặc điểm sau:

- Khởi phát Timer1 ở chế độ 2 để tạo tốc độ 9600 baud với tần số nhịp 11,059MHz.
- Cho phép ngắt nhận nối tiếp ở chế độ 1 với RI = 1 và xóa ô nhớ đệm Buffer có địa chỉ 30H.
- Chờ nhấn phím SW của bộ chuyển mạch, đọc cổng P1 vào A, so sánh với ô nhớ đệm, nếu có giá trị cũ, trở về chờ nhấn phím.
- Khi có phím nhấn SW = 1 (A khác Buffer), truyền A vào SBuf, xóa cờ EchoFlag và cờ TransmitErrFlag.
- Chờ báo truyền xong khi EchoFlag = 1. Nếu cờ truyền sai TransmitErrFlag = 1, trở về truyền lại.
- Trở về chờ nhấn phím.

Chương trình phục vụ ngắt UARTInt của ví dụ 17.2 với giải thuật (hình 17.5b) có đặc điểm sau:

- Xác lập dải thanh ghi R0–R7 ở dải 1.
- Chờ cờ ngắt nhận RI = 1 và xóa RI, TI để kết thúc chương trình ngắt.
- Khi có RI = 1, nhận tin từ SBUF vào A để so sánh với ô nhớ đệm Buffer, nếu khác, xác lập cờ truyền sai TransmitErrFlag = 1. Nếu A = Buffer (truyền đúng), xác lập cờ truyền EchoFlag = 1.
- Kết thúc chương trình phục vụ ngắt.

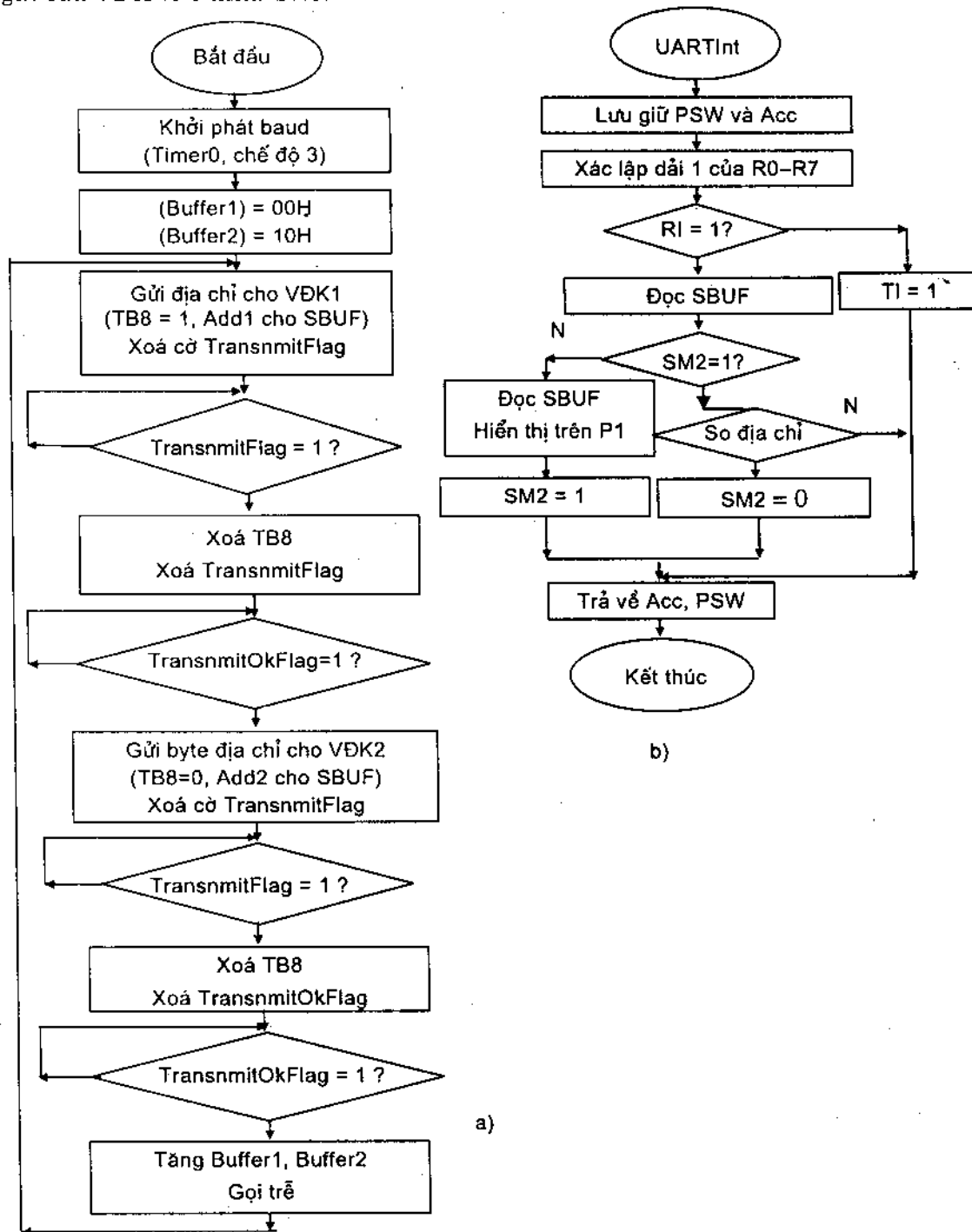
3. Chương trình nhận tin từ VDK phía trên

Chương trình nhận tin này được nạp vào VDK phía trên và tương tự chương trình nhận tin ở ví dụ 17.1. Với chương trình chính xác lập chế độ cho Timer1, chế độ 2 cho tốc độ baud, còn chương trình phục vụ ngắt tương tự chương trình phục vụ ngắt của ví dụ 17.1 là chờ RI = 1, đọc SBUF và đưa hiển thị ra cổng P1.

: Ví dụ 17.2: Truyền nối tiếp ở VDK dưới <pre> Buffer equ 30h EchoFlag equ 20h.0 TransmitErrFlag equ 20h.1 org 00h jmp Start org 023h jmp UARTInt Start: ; Timer1 (for UART) mode 2 mov tmod, #00100000b ; 9600 bit/sec anl pcon, #01111111b mov th1, #0fdh setb tr1 ; serial mode 1, receive enable mov scon, #01010000b ; enable interrupt mov ie, #10010000b mov Buffer, #30h WaitLoop: mov a, pl cjne a, Buffer, SWChange jmp WaitLoop SWChange: mov Buffer, a TransmitAgain: mov sbuf, a clr EchoFlag clr TransmitErrFlag WaitEcho: jb TransmitErrFlag, TransmitAgain jnb EchoFlag, WaitEcho jmp WaitLoop </pre>	<pre> SWChange: mov Buffer, a TransmitAgain: mov sbuf, a clr EchoFlag clr TransmitErrFlag WaitEcho: jb TransmitErrFlag, TransmitAgain jnb EchoFlag, WaitEcho jmp WaitLoop UARTInt: push psw push acc setb rs0 clr rs1 jbc ri, ReceiveInt clr ti jmp Return ReceiveInt: mov a, sbuf cjne a, Buffer, DataError setb EchoFlag sjmp Return DataError: setb TransmitErrFlag Return: pop acc pop psw reti end </pre>
: Ví dụ 17.3: Nhận tin nối tiếp ở VDK trên <pre> org 00h jmp Start org 023h jmp UARTInt Start: ; dùng Timer1 (for UART) mode 2 mov tmod, #00100000b ; 9600 bit/sec anl pcon, #01111111b mov th1, #0fdh setb tr1 ; serial mode 1 (receive enable) mov scon, #01010000b mov ie, #10010000b ; enable interrupt Loop: jmp Loop </pre>	<pre> UARTInt: push psw push acc setb rs0 clr rs1 jbc ri, ReceiveInt clr ti jmp Return ReceiveInt: mov a, sbuf mov pl, a mov sbuf, a Return: pop acc pop psw reti end </pre>

2. Chương trình truyền tin của VDK chủ

Giải thuật của chương trình của VDK chủ (ví dụ 17.4) và của chương trình phục vụ ngắt của VDK tớ ở hình 17.8.



Hình 17.8. Giải thuật của VDK chủ (a) và chương trình ngắt nhận tin của VDK tớ (b)

Chương trình chính của VDK chủ ở ví dụ 17.4 có những đặc điểm sau:

- Có các ô nhớ đệm Buffer1 và Buffer2 cho dữ liệu của các VDK tớ.

- Có cờ truyền xong TransmitOkFlag.
 - Có khai báo các địa chỉ của các VDK tớ.
 - Timer0 dành cho tốc độ 9600 baud ở chế độ 2.
 - Chế độ trao đổi tin nối tiếp ở chế độ 3 (trao đổi nhiều VDK).
- Vòng lặp để truyền dữ liệu với các bước sau:
- Xác lập TB8 = 1, xoá cờ truyền TransmitOkFlag, gửi byte địa chỉ Addr1 rồi chờ cờ truyền TransmitOkFlag xác lập.
 - Xác lập TB8 = 1, xoá cờ truyền TransmitOkFlag, gửi byte địa chỉ Addr2 rồi chờ cờ truyền TransmitOkFlag xác lập.
 - Tăng giá trị của các ô nhớ đệm Buffer1, Buffer2.
 - Gọi trễ 1 sec.
 - Trở về vòng lặp để truyền dữ liệu mới.
- Chương trình phục vụ ngắt nhận của VDK chủ ở ví dụ 17.4 có những đặc điểm sau:
- Giống chương trình phục vụ ngắt của ví dụ 17.3 nhưng không có nhận tin (chỉ có nhận ReceiveInt).
 - Có thêm lệnh xác lập cờ TransmitOkFlag.

<pre> ; Ví dụ 17.4: VDK chủ truyền nối tiếp cho nhiều VDK tớ Buffer1 equ 30h Buffer2 equ 31h TransmitOkFlag equ 20h.0 Addr1 equ 1 Addr2 equ 2 org 00h jmp Start org 023h jmp UARTInt Start: ; Timer0 mode 1, Timer1 mode 2 mov tmod, #00100001b; 9600 bit/sec anl pcon, #01111111b mov th1, #0fdh setb tr1 mov scon, #11010000b; serial mode 3 movie, #10010000b mov Buffer1, #0 mov Buffer2, #10H Loop: setb TB8 ; send byte Addr clr TransmitOkFlag mov SBUF, #Addr1; gửi cho VDK tớ 1 WaitTransmit1: jnb TransmitOkFlag, WaitTransmit1 clr TB8 clr TransmitOkFlag mov SBUF, Buffer1 jnb TransmitOkFlag, \$; chờ cờ = 1 setb TB8 ; gửi byte địa chỉ </pre>	<pre> clr TransmitOkFlag mov SBUF, #Addr2; gửi cho VDK tớ 2 WaitTransmit2: jnb TransmitOkFlag, WaitTransmit2 clr TB8 clr TransmitOkFlag mov SBUF, Buffer2 jnb TransmitOkFlag, \$; chờ cờ = 1 inc Buffer1 inc Buffer2 mov r5, #20 call Delay jmp Loop ; truyền dữ liệu tiếp theo UARTInt: push psw push acc setb rs0 clr rs1 jbc ri, ReceiveInt clr ti setb TransmitOkFlag jmp Return ReceiveInt: Return: pop acc pop psw reti ; Delay Time = (R5)*50ms @fxTAL = 12MHz Delay: ; như ví dụ 17.1 end </pre>
---	---

3. Chương trình nhận tin của VDK tớ

Mỗi một VDK tớ có một chương trình riêng với chương trình phục vụ ngắt để nhận tin có giải thuật như hình 17.8b. Các chương trình của các VDK tớ này đều có cấu trúc giống nhau, chỉ có điều khác là khai báo địa chỉ tương ứng cho VDK đó là khác nhau, để khi VDK chủ gửi tin trên đường dây TxD (nối với các chân RxD của các VDK tớ), chỉ VDK tớ (ở chế độ 3) nào có địa chỉ Addr tương ứng mới nhận tin.

Ví dụ 17.5 giới thiệu chương trình của VDK tớ 1 với khai báo Addr equ 1. Với chương trình nạp cho VDK tớ 2, ta chỉ cần đổi khai báo là Addr equ 2.

Chương trình chính của ví dụ 17.5 cũng giống chương trình chính của VDK chủ (chế độ Timer là 3).

Chương trình phục vụ ngắt của VDK tớ này có giải thuật như hình 17.8b và có những đặc điểm sau:

- Phần đầu có các lệnh giống chương trình phục vụ ngắt nhận tin ở các chương trình trước.
- Phần nhận tin theo ngắt chương trình có các bước sau:
 - + Đọc SBUF vào A rồi đưa vào Buffer
 - + Chờ xác lập SM2
 - Nếu SM2 = 1, chuyển tới ghi địa chỉ. Nếu thanh ghi A chứa #Address sẽ xoá SM2. Nếu là dữ liệu sẽ kết thúc.
 - Nếu SM2 = 0, hiển thị dữ liệu nhận ra cổng P1 và xác lập SM2 = 1 rồi kết thúc chương trình ngắt.

; Ví dụ 17.5: Nhận nối tiếp cho VDK tớ	
Addr equ 1 Buffer equ 30h org 00h jmp Start org 023h jmp UARTInt Start: ; dùng Timer1 (for UART) mode 2 mov tmod, #00100000b ; baud rate = 9600 bit/sec, 11,059MHz anl pcon, #01111111b mov th1, #0fdh setb tr1 ; serial mode 3 @ receive enable mov scon, #11010000b mov ie, #10010000b ; enable UART interrupt setb SM2 Loop: jmp Loop UARTInt: push psw	push acc setb rs0 clr rs1 jbc ri, ReceiveInt clr ti jmp Return ReceiveInt: mov a, sbuf mov Buffer, a jb SM2, IsAddrByte mov P1, a setb SM2 sjmp Return IsAddrByte: cjne a, #Addr, Return clr SM2 ; sẵn sàng nhận byte dữ liệu Return: pop acc pop psw reti end

4. Chạy mô phỏng chương trình

Muốn chạy mô phỏng các chương trình truyền của VDK chủ (ví dụ 17.4) cho các VDK tổ (ví dụ 17.5) ta phải:

- Vào chương trình mô phỏng PROTEUS.
- Vào sơ đồ của hình 17.7.
- Nhập chương trình dạng.HEX của ví dụ 17.4 vào VDK chủ.
- Nhập chương trình dạng.HEX của ví dụ 17.5 vào các VDK tổ tương ứng.
- Chạy chương trình bằng nút nhấn tam giác của thanh công cụ số 4.
- Quan sát các tín hiệu trên các LED mắc với cổng P1 để biết các VDK tổ nhận tín.
- Có thể, một VDK tổ nào đó khai báo sai địa chỉ hay không nạp chương trình dạng.HEX, VDK tổ đó sẽ không có hiển thị trên các LED mắc với cổng P1 vì VDK tổ đó không nhận tín.

17.3. TRAO ĐỔI TIN NỐI TIẾP CỦA UART CỦA VDK AT89C51RD2/ED2

Hãng ATMEL đã chế tạo VDK AT89C51RD2/ED2 với công nghệ CMOS, tương thích với 8052 của họ MCS-51 với nhiều tính năng cải tiến, trong đó có trao đổi tin nối tiếp UART và SPI.

17.3.1. Những đặc điểm cơ bản của AT89C51RD2/ED2

AT89C51RD2/ED2 tương thích với 8052 và có các đặc tính cải tiến mới.

Những đặc điểm chính của AT89C51RD2/ED2 là:

- AT89C51RD2/ED2 tương thích với 8052 về các đặc điểm sau:
 - + Hệ lệnh tương thích với hệ lệnh của 8051.
 - + Có 6 cổng I/O 8 bit cho vi mạch phiên bản 64 chân hay 68 chân.
 - + Có 4 cổng I/O 8 bit cho vi mạch phiên bản 44 chân.
 - + Có 3 Timer/Counter và một watchdog.
 - + Có 256 byte nhớ RAM bên trong.
 - + Có 9 nguồn ngắt chương trình với 4 mức ưu tiên.
- Những tính năng mới là:
 - + Có bộ giám sát nguồn nuôi tích hợp để theo dõi nguồn nuôi bên trong.
 - + Dùng nguồn nuôi chuẩn 5V để ghi chương trình trong hệ (ISP-In System Programming).
 - + Có ROM BOOT chứa các chương trình con cho lập trình nhớ Flash mức thấp.
 - + Có tốc độ cao với tần số nhịp 40MHz (Vcc 2,7V tới 5,5V) hay 60MHz (Vcc 4,5V tới 5,5V).
 - + Bộ nhớ có các đặc tính sau:
 - Có 64Kbyte nhớ Flash cho chương trình/dữ liệu trên chip.
 - Có RAM mở rộng (XRAM) 1792 byte, chọn kích thước bởi chương trình.
 - Có 2048byte nhớ EEPROM cho dữ liệu (chỉ AT89C51ED2 với các chu trình ghi 100k).
 - + Có hai con trỏ dữ liệu.
 - + Có giao diện ngắt bàn phím ở cổng P1.
 - + Có trao đổi tin nối tiếp UART với phát hiện sai khung tin, nhận dạng địa chỉ và thay đổi tốc độ truyền baud.

- + Có trao đổi tin nối tiếp theo SPI (Serial Pherpherial Interface).
- + Có một ma trận đếm chương trình hoá (PCA) 16 bit, có thể điều chế độ rộng xung (PWM).
- + Có bộ chia trước xung nhịp 8 (Clock Prescaler) 8 bit.

17.3.2. Trao đổi tin nối tiếp UART của AT89C51RD2/ED2

Cổng I/O nối tiếp của AT89C51RD2/ED2 là tương thích với cổng I/O nối tiếp của 80C52. Nó có hai chế độ trao đổi tin không đồng bộ (UART) và đồng bộ (SPI).

Trao đổi tin UART có ba chế độ song công 1, 2 và 3 (hai chiều truyền nhận đồng thời). UART này khác với UART của 80C52 ở ba điểm:

- Phát hiện lỗi sai khung truyền hay không nhận được bit STOP ở mỗi khung tin.
- Nhận dạng địa chỉ tự động.
- Thay đổi tốc độ baud.

1. Phát hiện lỗi khung truyền

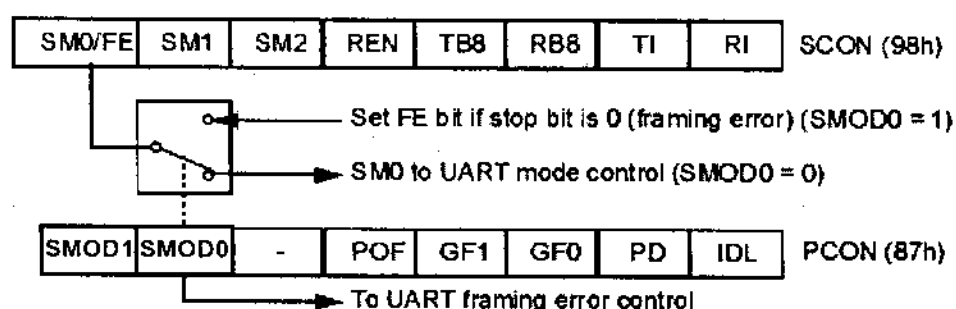
Đặc điểm này có ở cả ba chế độ 1, 2 và 3. Để có thể phát hiện lỗi sai khung truyền, ta phải:

– Xóa bit D6 (SMOD0 của thanh ghi PCON (87H) về 0 để điều khiển lỗi sai khung FE cho UART. Khi đó bit D7 (SM0/FE) của thanh ghi SCON (98H) đóng chuyển mạch vào SM0 để UART điều khiển chế độ phát hiện lỗi sai khung truyền như hình 17.9.

– Khi có lỗi khung truyền tức không có bit Stop sau byte tin do lỗi truyền khung (vì nhiễu trên đường dây hay tới đồng thời của hai CPU của hai VDK), ta phải xác lập bit D6 (SMOD0 của thanh ghi PCON (87H) lên 1 để nhận giá trị của lỗi sai khung truyền vào FE của SCON. Như vậy, bit FE này chính là cờ lỗi sai khung truyền.

Khi có lỗi sai khung truyền thì:

- Rì tăng bit Stop của bit dữ liệu sau cùng của khung tin từ mức thấp lên mức cao.
- Muốn phát hiện lỗi sai khung truyền, ta phải đọc bit FE = 1 của TCON.



Hình 17.9. Các thanh ghi SCON và PCON điều khiển lỗi sai khung truyền

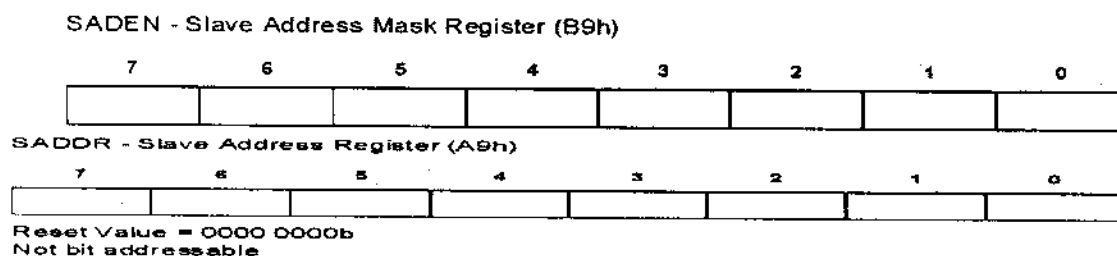
2. Nhận biết địa chỉ một cách tự động

Trong hệ trao đổi tin có nhiều VDK, vì có dây nối TxD của VDK với các lối vào RxD của các VDK tổ (hình 17.7), các VDK ở chế độ trao đổi tin M = 3. Mỗi VDK tổ phải nhận biết được lời địa chỉ mà VDK chủ dành cho mình để nhận tin về dữ liệu. Chương trình của ví dụ 17.5 đã phải nhận biết địa chỉ bằng lệnh (phần mềm).

Để tự động nhận biết địa chỉ cho VDK tổ, VDK AT89C51RD2/ED2 đã đưa vào hai thanh ghi (phần cứng) (hình 17.10):

– Thanh ghi địa chỉ thiết bị tớ SADDR (A9H) để ghi địa chỉ dành cho VDK của thiết bị tớ.

– Thanh ghi mặt nạ hay cho phép địa chỉ thiết bị tớ SADEN (B9H) để ghi các bit cho phép địa chỉ (= 1) hay cấm (= 0) địa chỉ dành cho VDK của thiết bị tớ hay tức 1 của SADEN cho phép giữ nguyên giá trị của SADDR, còn 0 của SADEN cho phép SADDR có giá trị bất kỳ.



Hình 17.10. Các thanh ghi phục vụ nhận biết địa chỉ tự động

Với giá trị khác nhau của hai thanh ghi này ghi ở các VDK khác nhau, ta có thể nhận biết được địa chỉ của VDK qua các ví dụ sau đây:

– *Địa chỉ đã cho*: lấy AND các giá trị của các bit của hai thanh ghi, còn khi SADEN = 0, bit địa chỉ đã cho là bất kỳ X.

SADDR bằng 0101 0110B

SADEN bằng 1111 1100B

Địa chỉ cho là 0101 01XXB

Áp dụng cho các VDK khác nhau, ta có:

VDK tớ A:

SADDR bằng 1111 0001B

SADEN bằng 1111 1010B

Địa chỉ cho là 1111 0X0XB

– *VDK tớ B:*

SADDR bằng 1111 0011B

SADEN bằng 1111 1001B

Địa chỉ cho là 1111 0XX1B

– *Địa chỉ quảng bá* (địa chỉ cho mọi thiết bị tớ hay VDK chủ truyền tin cho mọi VDK tớ có địa chỉ quảng bá): lấy OR các giá trị của các bit hai thanh ghi, còn khi SADEN = 0, bit địa chỉ đã cho là bất kỳ X.

VDK tớ A:

SADDR bằng 1111 0001B

SADEN bằng 1111 1010B

Địa chỉ cho là 1111 1X11B

VDK tớ B:

SADDR bằng 1111 0011B

SADEN bằng 1111 1001B

Địa chỉ cho là 1111 0X11B

Khi nhấn nút RESET của VDK, các giá trị của hai thanh ghi này bị xoá về 0, nghĩa là chưa có cơ chế nhận biết địa chỉ. Nhận biết địa chỉ chỉ có tác dụng sau khi đã ghi các giá trị của các thanh ghi trên bằng lệnh của chương trình.

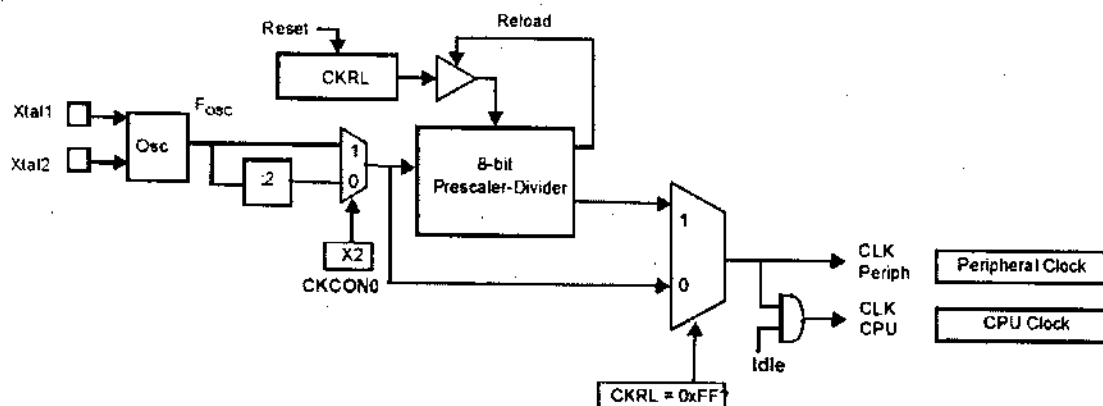
3. Thay đổi và lưu giữ tốc độ truyền

Khác với VDK 8052, AT89C51RD2/ED2 này có thể thay đổi và lưu giữ tốc độ

truyền nối tiếp UART bằng bộ chia thang trước (Prescaler Divider) và bộ chia bên trong của UART.

a) Thay đổi tần số nhịp bằng bộ chia thang trước (Prescaler Divider)

Với bộ chia thang trước này (hình 17.11), ta có thể thay đổi được tần số của xung nhịp cho VDK và cho thiết bị ngoài với các công thức như hình 17.12.



Hình 17.11. Bộ chia thang trước (prescaler) của xung nhịp

- A hardware RESET puts the prescaler divider in the following state:
 - CKRL = FFh: $F_{CLK\ CPU} = F_{CLK\ PERIPH} = F_{OSC}/2$ (Standard C51 feature)
- Any value between FFh down to 00h can be written by software into CKRL register in order to divide frequency of the selected oscillator:
 - CKRL = 00h: minimum frequency
 $F_{CLK\ CPU} = F_{CLK\ PERIPH} = F_{OSC}/1020$ (Standard Mode)
 $F_{CLK\ CPU} = F_{CLK\ PERIPH} = F_{OSC}/510$ (X2 Mode)
 - CKRL = FFh: maximum frequency
 $F_{CLK\ CPU} = F_{CLK\ PERIPH} = F_{OSC}/2$ (Standard Mode)
 $F_{CLK\ CPU} = F_{CLK\ PERIPH} = F_{OSC}$ (X2 Mode)

$F_{CLK\ CPU}$ and $F_{CLK\ PERIPH}$

In X2 Mode, for CKRL <> 0xFF:

$$F_{CPU} = F_{CLK\ PERIPH} = \frac{F_{OSC}}{2 \times (255 - CKRL)}$$

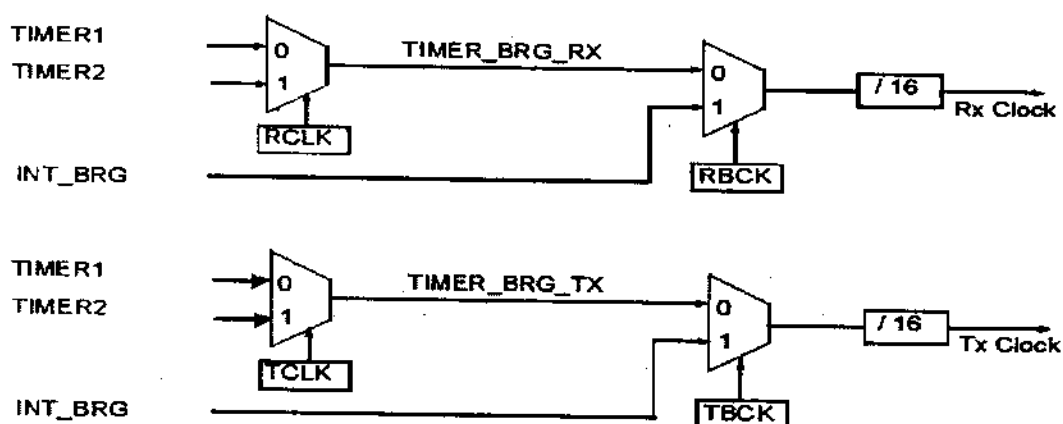
In X1 Mode, for CKRL <> 0xFF then:

$$F_{CPU} = F_{CLK\ PERIPH} = \frac{F_{OSC}}{4 \times (255 - CKRL)}$$

Hình 17.12. Các công thức tần số xung nhịp ở lối ra của bộ chia thang trước

b) Thay đổi tần số nhịp bằng máy phát nhịp bên trong BRG

Với lựa chọn các tần số nhịp RCLK và RBCK của các thanh ghi điều khiển T2CON, BDRCON, ta có thể thay đổi tần số xung nhịp RxClock và TxClock như hình 17.13.

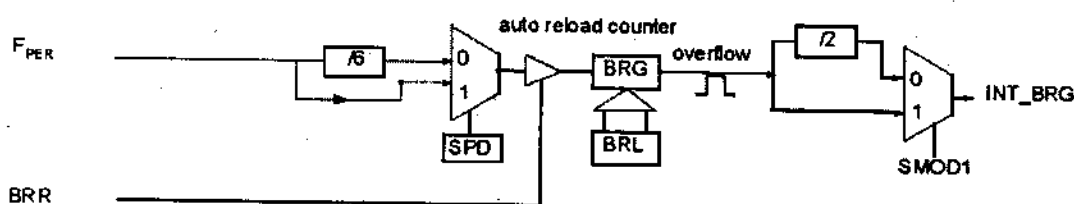


Hình 17.13. Lựa chọn tốc độ baud theo RBCK và TBCK

Với thanh ghi nạp lại tốc độ baud BRL (9Ah) dùng cho máy phát tốc độ baud bên trong (hình 17.14) ta có thể thay đổi tốc độ baud của UART bằng tín hiệu BRR là tín hiệu tràn của bộ thay đổi tốc độ baud bên trong BRG với công thức sau:

$$\text{Baud_Rate} = \frac{2^{\text{SMOD1}} \cdot F_{\text{PER}}}{6^{(x-\text{SPD})} \cdot 32 \cdot (256 - \text{BRL})}$$

$$\text{BRL} = 256 - \frac{2^{\text{SMOD1}} \cdot F_{\text{PER}}}{6^{(x-\text{SPD})} \cdot 32 \cdot \text{Baud_Rate}}$$



Hình 17.14. Phát tốc độ baud bên trong BRG

17.4. TRAO ĐỔI TIN NỐI TIẾP SPI CỦA AT89C51RD2/ED2

Ngoài trao đổi tin nối tiếp theo UART, VDK AT89C51RD2/ED2 cũng như một số VDK khác còn trao đổi tin theo SPI (Serial Peripheral Interface – giao diện thiết bị ngoài nối tiếp).

17.4.1. Đặc điểm chính của trao đổi tin nối tiếp SPI

SPI cho phép trao đổi tin nối tiếp đồng bộ song công (truyền và nhận đồng thời) giữa CPU và các thiết bị ngoài, bao gồm cả VDK khác.

SPI có thể:

- Truyền đồng bộ hai chiều trên ba đường dây (hình 17.15).
- Có hành động của thiết bị chủ và thiết bị tớ.
- Có 8 tốc độ của xung nhịp có thể định bằng chương trình.
- Nhịp nối tiếp với cực tính và pha theo chương trình.
- Có cờ lỗi của chế độ chủ với khả năng ngắt CPU.
- Bảo vệ cờ xung đột khi ghi dữ liệu.

Mỗi thiết bị trao đổi tin theo SPI có 4 đường dây nối:

– MOSI (Master Output Slave Input–ra từ chủ, vào tớ): dùng để

VDK chủ truyền tin cho thiết bị/linh kiện tớ nhận tin.

MISO (Master Input Slave

– Output–ra từ tớ vào chủ): dùng để VDK chủ nhận tin từ thiết bị/linh kiện tớ.

– SCK (SPI Serial Clock– nhịp nối tiếp SPI): dùng để truyền tín hiệu nhịp, để đồng bộ sự chuyển dữ liệu vào và ra của thiết bị ngoài trên các đường dây MOSI và MISO của chúng.

– /SS– Slave Select– lựa chọn thiết bị tớ: dùng để chọn một trong các thiết bị tớ.

Như vậy, chỉ cần 4 tín hiệu, trao đổi tin SPI có thể trao đổi tin giữa một VDK chủ và nhiều thiết bị tớ, được điều khiển bởi đường dây /SS đi ra từ VDK chủ. Có bao nhiêu đường dây /SS, sẽ có thể trao đổi tin với bấy nhiêu thiết bị ngoài.

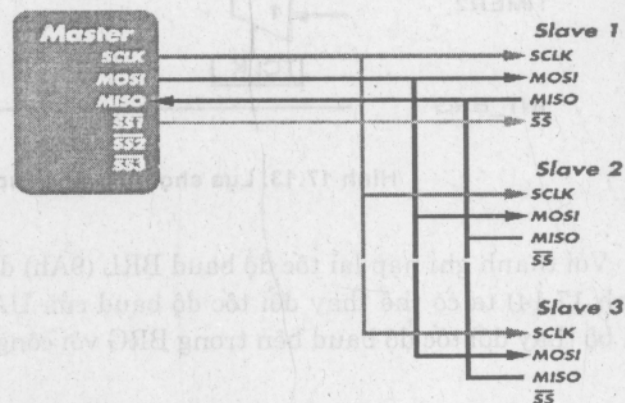
17.4.2. Cấu trúc khối trao đổi tin theo SPI của VDK AT89C51RD2/ED2

Để đảm bảo trao đổi tin theo SPI, VDK AT89C51RD2/ED2 có khối điều khiển như hình 17.16.

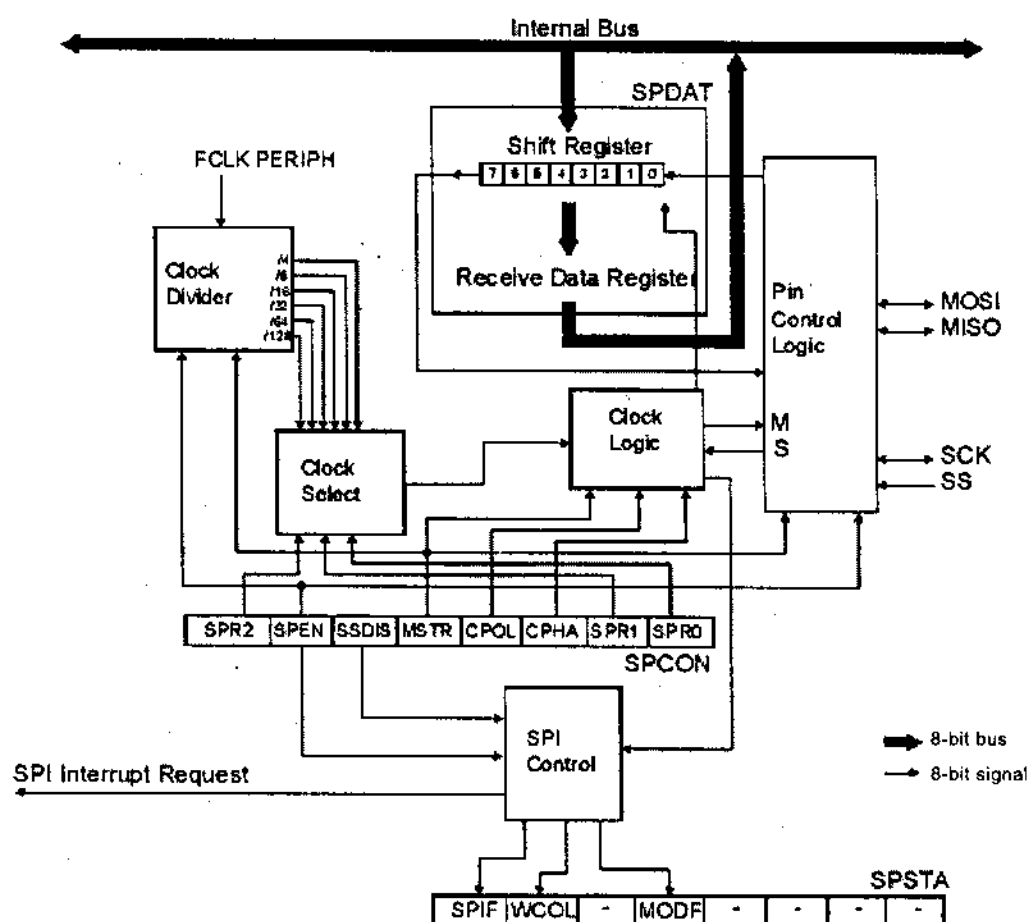
Khối trên có hai chế độ : chế độ chủ cho VDK chủ và chế độ tớ cho thiết bị tớ.

Khối có các thanh ghi sau:

- Thanh ghi điều khiển SPI (SPCON).
 - Thanh ghi trạng thái (SPSTA).
 - Thanh ghi dữ liệu SPI (SPDATA) với các bit R0–R7.
- Ngoài ra, khối còn có ngắt.



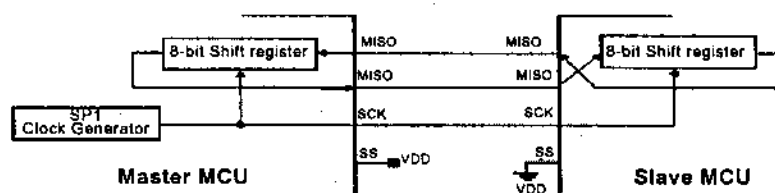
Hình 17.15. Sơ đồ nối dây giữa các VDK của SPI



Hình 17.16. Sơ đồ khối điều khiển SPI của VĐK AT89C51RD2/ED2

17.4.3. Trao đổi tin theo SPI của VĐK AT89C51RD2/ED2

Sơ đồ mắc để trao đổi tin theo SPI giữa VĐK chủ và thiết bị tớ như hình 17.17.



Hình 17.17. Sơ đồ mắc giữa VĐK chủ và thiết bị ngoài trao đổi tin theo SPI

Bằng chương trình, có thể chọn một trong bốn tổ hợp của pha và cực tính của xung nhịp nối tiếp (SCK) bằng cách dùng hai bit trong SPCON là CPOL (cực tính) và CPHA (pha).

Trước khi trao đổi tin phải xác định chế độ của các VĐK cũng như thiết bị ngoài và

thiết bị chủ hay nhận bởi tín hiệu /SS. Tín hiệu nhịp SCK được dẫn trên đường dây SCK. Tín truyền được dẫn theo đường dây MOSI còn tín nhận được dẫn theo đường dây MISO.

17.4.4. Chương trình trao đổi tin theo SPI của VDK ATmega8535(L)

Hãng ATMEL đã chế tạo VDK ATmega8535 tương tự VDK SA8535 của hãng SIEMEN hay AT90S8535 của hãng AVR. VDK này có cả trao đổi tin theo SPI và I2C.

Các ví dụ 17.6 giới thiệu các chương trình con khởi phát, truyền và nhận cho SPI chủ và SPI tớ dưới dạng mã Assembly. Chúng ta có thể dùng để đổi sang các lệnh Assembly của VDK tương ứng.

; Ví dụ 17.6: Chương trình truyền (hợp ngữ)	Chương trình con nhận (hợp ngữ)
SPI_Masterinit: ; Set MOSI, SCK là ra, các t/hiệu khác là vào ldi r17, (1<<DD_MOSI)(1<<DD_SCK) out DDR_SPI, r17 ; Enable SPI, Master, set clock rate fck/16 ldi r17, (1<<_SPE) (1<<MSTR)(1<<SPRO) out SPCR, r17 ret ; SPI_MasterTransmit: ; Start transmission of data (r16) out SPDR, r6 Wait_Transmit: ; Wait for transmission complet sbis SPSR, SPIF rjmp Wait_Transmit ret	SPI_slaveinit: ; Set MISO là ra, các t/hiệu khác là vào ldi r17, (1<<DD_MISO) out DDR_SPI, r17 ; Enable SPI, Master, set clock rate fck/16 ldi r17, (1<<SPE) out SPCR, r17 ret ; SPI_SlaveReceive: ; Wait for reception complet sbis SPSR, SPIF rjmp SPI_SlaveReceive in r16, SPDR ret

Trao đổi tin theo Microwire chính là theo SPI nhưng với gói 16 bit.

17.5. TRAO ĐỔI TIN THEO USB VÀ I2C

Hai đường dây chuẩn USB và I2C có đặc điểm chung là chỉ trao đổi tin theo hai đường dây trong khi SPI theo 4 đường dây. Riêng cáp truyền tin của USB có 4 đường nhưng hai đường dành cho nguồn nuôi +5V, đất.

17.5.1. Trao đổi tin theo USB

Chúng ta đã xét đặc điểm trao đổi tin nối tiếp theo USB cho máy vi tính PC ở đầu chương này với chipset có sẵn trong PC. Với VDK, muốn trao đổi tin nối tiếp theo USB, ta cần một vi mạch ghép nối với đường dây USB. Có nhiều hãng điện tử chế tạo các vi mạch ghép nối này. Hãng Cypress (mua công nghệ của Intel) đã chế tạo vi mạch để ghép nối VDK thông dụng với đường dây USB, có hai loại vi mạch sau:

- CY7C63: dành cho USB tốc độ thấp.

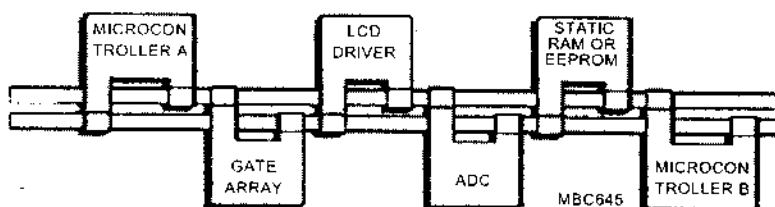
- CY7C64: dành cho USB tốc độ cao.

Hãng National Semiconductor đã chế tạo USBN9602 cho giao tiếp song song và USB cũng như Microwire.

Hãng Netchip đã chế tạo vi mạch NET2888 dùng cho VDK ghép nối với đường dây USB.

17.5.2. Đặc điểm chung của trao đổi tin theo I2C

Trao đổi tin I2C (IIC—Inter-Integrated Circuit Interface— giao diện mạch tích hợp) chỉ có hai đường dây, một đường dây dữ liệu SDA hai chiều thuận nghịch và đường dây tín hiệu xung nhịp SCL dùng để nối các vi mạch tích hợp và các VDK (hình 17.18).



Hình 17.18. Mạch trao đổi tin theo I2C của hai VDK

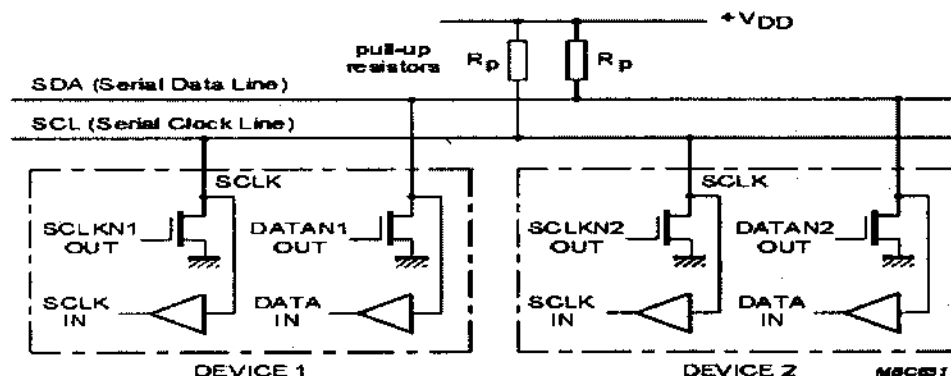
I2C đã qua hai phiên bản, phiên bản 1.0—1992 và phiên bản 2.0—1998.

I2C dùng với các thuật ngữ sau:

- *Thiết bị truyền*: thiết bị gửi dữ liệu cho đường dây (BUS).
- *Thiết bị nhận*: thiết bị nhận dữ liệu từ đường dây.
- *Thiết bị chủ*: thiết bị khởi phát cuộc truyền, phát tín hiệu nhịp (Clock) và kết thúc truyền tin.
- *Thiết bị tớ*: thiết bị được thiết bị chủ gọi địa chỉ.
- *Thiết bị nhiều chủ*: có thể nhiều thiết bị chủ đồng thời trên đường dây và không làm mất tin.
- *Sự trọng tài (Arbitration)*: thủ tục để đảm bảo rằng chỉ một thiết bị chủ phát tin.
- *Sự đồng bộ*: thủ tục để đồng bộ các tín hiệu nhịp của hai hay nhiều thiết bị.

1. Đường dây

Đường dây của I2C chỉ có hai dây dẫn hai tín hiệu SDA và SCL. Cả hai đường dây trên là hai chiều, nối với nguồn nuôi +5V qua điện trở kéo lên R_p như hình 17.19.



Connection of Standard- and Fast-mode devices to the I2C-bus.

Hình 17.19. Nối các thiết bị chuẩn và chế độ nhanh với I2C

2. Truyền dữ liệu

Trong cuộc truyền đầy đủ với hai byte dữ liệu trên, sau điều kiện ban đầu START là địa chỉ (7 bit), lệnh đọc/ghi (R/W), xác nhận (ACK), dữ liệu (8 bit), xác nhận, dữ liệu, xác nhận và kết thúc bởi điều kiện STOP.

Địa chỉ với định nghĩa các bit trong byte thứ nhất như ở bảng 17.5.

Có các loại chế độ truyền:

- Truyền nhanh hay chuẩn (F/S) : tốc độ tới 400 kbit/s.
- Truyền tốc độ cao (Hs): tốc độ tới 3,4 Mbit/s.

17.5.3. Trao đổi tin theo I2C của VDK ATmega8535

Hãng ATMEL đã chế tạo VDK ATmega8535 thuộc loại AVR có giao diện nối tiếp hai dây (TWI—Two—wire Serial Interface) để trao đổi tin theo I2C cho 128 thiết bị ngoài (địa chỉ 8 bit) như bảng 17.5.

Bảng 17.5. Các địa chỉ tớ

SLAVE ADDRESS	R/W BIT	DECRPTION
0000 000	0	General call address
0000 000	1	START byte ⁽¹⁾
0000 001	X	CBUS address ⁽²⁾
0000 010	X	Reserved for different bus forma ⁽³⁾
0000 011	X	Reserved for future purposes
0000 1XX	X	Hs—mode master code
1111 1XX	X	Reserved for future purposes
1111 0XX	X	10—bit stave addressing

1. Đặc điểm chung

TWI của ATmega8535 có các đặc điểm chính như sau:

- Là giao diện truyền thông đơn giản, mạnh và mềm dẻo chỉ cần hai dây SDA và SCK.
- Hỗ trợ cả hai hành động chủ và tớ.
- Thiết bị có thể hoạt động như bộ phát hay bộ nhận tin.
- Có 7 bit địa chỉ cho phép gọi tới 128 địa chỉ thiết bị tớ.
- Có hỗ trợ hành động nhiều chủ.
- Tốc độ truyền tin tới 400 kHz.
- Các địa chỉ của thiết bị tớ hoàn toàn gọi nhờ chương trình.

2. Cấu tạo TWI

TWI có hai đường dây SDA và SCK và 4 khối chính: Khối giao diện đường dây (Bus Interface Unit), khối máy phát tốc độ baud (Bit Rate Generator), khối chọn địa chỉ (Address Match Unit) và khối điều khiển (Control Unit).

Các đường dây SDA và SCK đều có các điều khiển tốc độ cao (slew—rate) và lọc nhiễu đột biến (Spike Filter).

Các thanh ghi của TWI là:

– *Thanh ghi điều khiển (TWCR)*: dùng để điều khiển bởi các bit TWINT (D7, cờ ngắt chương trình), TWEA (xác nhận có thể), TWST (phát Start), TWSTO (phát Stop), TWWC (cờ xung đột ghi), TWEN (cho phép TWI hoạt động), dự trữ (D1) và TWIE (cho phép TW ngắt chương trình).

– *Thanh ghi trạng thái (TWSR)*: TWS7.TWS3 ghi trạng thái của đường dây và logic TWI,

TWPS1 và TWPS0 (D0) dành ghi giá trị chia trước (Prescaler) là 1, 4, 16 hay 64 cho xung nhịp.

– *Thanh ghi địa chỉ (TWAR)*: ghi địa chỉ thiết bị tớ.

– *Thanh ghi dịch địa chỉ/dữ liệu (TWDR)*: ghi địa chỉ hay dữ liệu đệm.

– *Thanh ghi tốc độ bit (TWBR)*: ghi thừa số chia tốc độ baud theo công thức sau:

$$\text{Tần số SCK} = \text{Tần số CPU} / 16 + 2(\text{TWBR}).4.\text{TWPS}$$

Trong đó : – TWBR là giá trị của thanh ghi tốc độ bit TWBR.

– TWPS là giá trị các bit chia trước trong thanh ghi trạng thái TWSR.

3. Sử dụng TWI

TWI của VDK ATmega8535 được dùng để điều khiển trao đổi tin theo I2C của VDK ATmega8535.

Việc điều khiển trao đổi tin này theo 7 hành động như hình 17.20 trong đó các hành động ứng dụng hay phần mềm (1, 3, 5, 7) và các hành động phần cứng (số 2, 4, 6). Ô trống là chỉ thị bit TWINT xác lập hay yêu cầu ngắt chương trình.

Các bước hành động (1–7) như sau:

1. *Truyền START cho BUS* : bằng cách ghi bit TWSTA (D5) vào thanh ghi TWCR.

2. *Xác lập cờ ngắt TWINT*: chỉ rằng bit START đã gửi lên BUS, thanh ghi trạng thái TWSR ghi mã trạng thái.

3. *Kiểm tra trạng thái*: đọc TWSR xem START đã gửi chưa? Chương trình ứng dụng nạp SLA+W vào thanh ghi đệm dữ liệu TWDR và nạp tín hiệu điều khiển thích hợp vào TWCR, đảm bảo rằng TWINT lên 1 và TWSTA = 0. Khi thiết bị tớ nhận dữ liệu về địa chỉ, sẽ gửi xác nhận A cho chủ.

4. *Xác lập cờ ngắt TWINT*: mã trạng thái chỉ rằng SLA+W đã gửi, ACK đã nhận.

5. *Kiểm tra trạng thái* : xem SLA+W đã gửi và ACK đã nhận chưa ? Chương trình ứng dụng nạp dữ liệu vào TWDR và nạp tín hiệu điều khiển thích hợp vào TWCR, đảm bảo rằng TWINT đã ghi lên 1.

6. *Xác lập cờ ngắt TWINT*: mã trạng thái chỉ rằng dữ liệu đã gửi và nhận ACK.

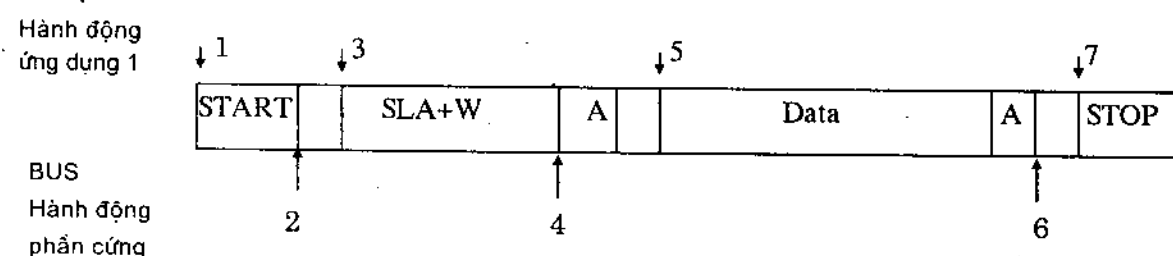
7. *Kiểm tra trạng thái*: xem dữ liệu đã gửi và đã nhận ACK. Chương trình áp dụng ghi bit TWSO vào TWCR để gửi STOP lên BUS khi biết chắc là TWINT = 1.

Qua các bước trên có thể tóm tắt quá trình như sau:

– *Khi TWI kết thúc một hành động và chờ phản ứng*: cờ TWINT sẽ xác lập lên 1. Đường dây SCL xuống thấp cho tới khi TWINT được xóa về 0.

– *Khi cờ TWINT xác lập*: người dùng phải cập nhật tất cả các thanh ghi TWI với giá trị đã có cho chu trình sau của BUS. Ví dụ, TWDR phải được nạp với giá trị đã truyền trong chu trình BUS sau. Sau khi tất cả các thanh ghi được cập nhật và các phần mềm khác đã hoàn thành, TWCR được ghi. Khi ghi TWCR, bit TWINT sẽ được xác lập.

– Khi ghi 1 vào TWINT: sẽ xoá cờ. Rồi TWI sẽ bắt đầu thực hiện hành động mà TWCR xác định.



Hình 17.20. Các bước (1–7) hành động của TWI

4. Các chế độ truyền tin

Trên đường dây I2C có mắc nhiều VDK và các thiết bị/linh kiện khác. Giữa chúng có thể có trao đổi tin với tư cách là thiết bị chủ phát hay truyền tin và thiết bị tớ nhận tin. Tùy thiết bị chủ hay tớ khởi phát hay yêu cầu trao đổi tin, ta có 4 trường hợp trao đổi tin chính như hình 17.21.

Cả 4 trường hợp, thiết bị chủ luôn phát tín hiệu SCK cho đường dây SCK.

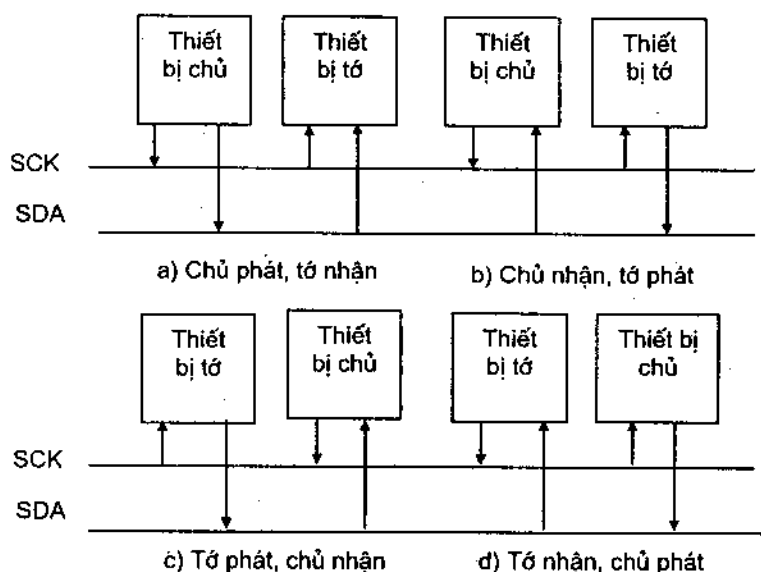
Khi thiết bị chủ chủ động khởi xướng trao đổi tin, có hai trường hợp :

- Chủ phát, tớ nhận.
- Chủ nhận, tớ phát.

Khi thiết bị tớ chủ động khởi xướng trao đổi tin, có hai trường hợp :

- Tớ phát, chủ nhận.
- Tớ nhận, chủ phát.

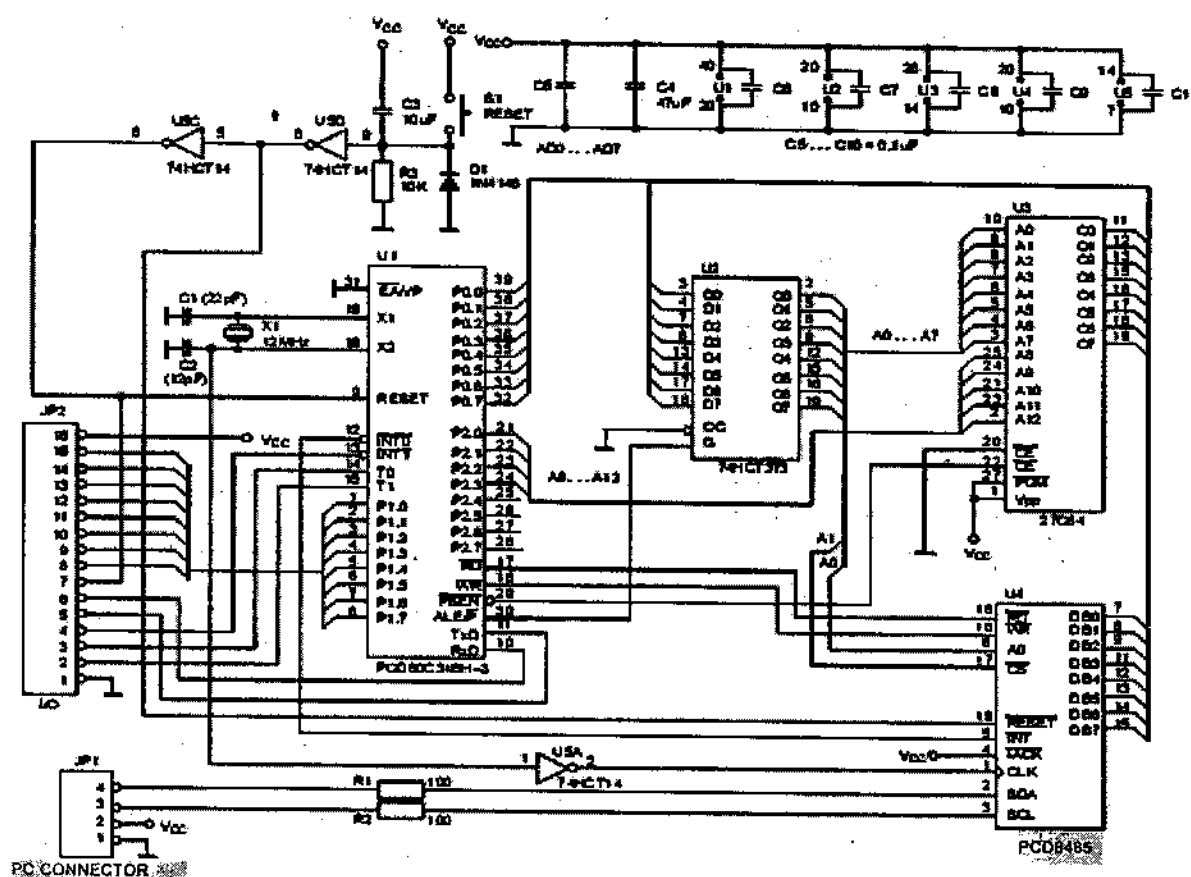
Khi trong mạch có nhiều thiết bị chủ cùng yêu cầu truyền tin cho cùng một thiết bị tớ hay các thiết bị tớ khác nhau, phải có sự giám sát hay trọng tài điều khiển lần lượt từng thiết bị chủ trao đổi tin theo thứ tự từng thiết bị chủ một.



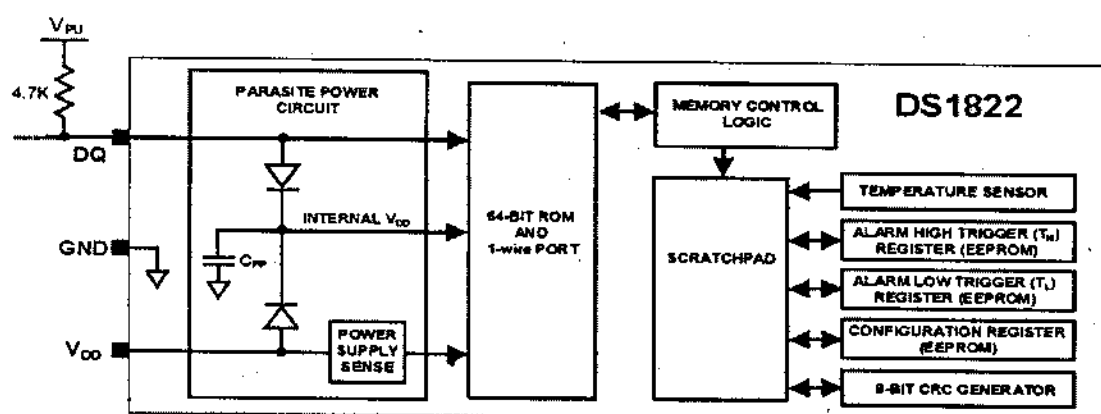
Hình 17.21. Các trường hợp trao đổi tin của I2C

17.5.4. Trao đổi tin theo I2C của VDK không có TWI

Với VDK không có sẵn khối TWI như VDK ATmega8535, ta phải mắc một vi mạch ghép nối để biến đổi từ đường dây song song hay nối tiếp theo UART sang BUS I2C. Hãng Phillip đã chế tạo vi mạch PCD8584 (trong tài liệu AN425 trên mạng) để ghép nối và biến đổi sang I2C như hình 17.22. Người ta có thể dùng hệ trên để ghép nối VDK với bàn phím của máy vi tính PC thông dụng qua vi mạch 87C751 (tài liệu AN434 trên mạng) để nối đường dây I2C với bàn phím.

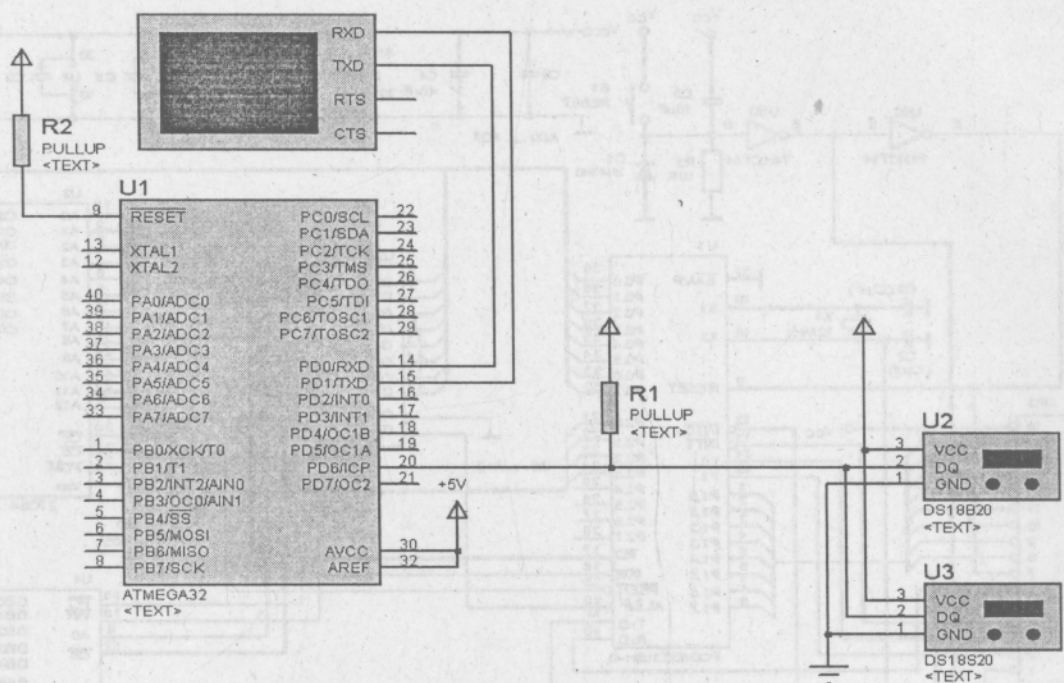


Hình 17.22. Sơ đồ mắc VDK thường với BUS I2C qua PCD8485



Hình 17.23. Sơ đồ khối của máy đo nhiệt độ DS1822

Hình 17.24 giới thiệu mạch trao đổi tin một dây với VDK ATMEGA32 với thiết bị ngoài DS18B20.



Hình 17.24. Mạch trao đổi tin một dây

Bạn đọc có thể thấy các chương trình cho các mạch trao đổi tin trong PROTEUS/SAMPLES.

17.6. TRAO ĐỔI TIN NỐI TIẾP THEO MẠNG CỦA VĐK

Cũng như các máy vi tính PC, các VĐK có thể trao đổi tin nối tiếp theo mạng gồm nhiều VĐK. Chúng ta đã gặp vấn đề trao đổi tin trong hệ nhiều VĐK trong:

- Hệ trao đổi tin nối tiếp nhiều VĐK ở chế độ M = 3.
- Hệ trao đổi tin nối tiếp nhiều VĐK trong hệ đường dây SPI.
- Hệ trao đổi tin nối tiếp nhiều VĐK trong hệ đường dây I2C.

Chúng ta xét hai trường hợp mới là:

- Trao đổi tin nối tiếp theo mạng CAN.
- Trao đổi tin nối tiếp theo mạng với VĐK có phục vụ trao đổi tin theo mạng như VĐK 83C152.

Tính năng chủ yếu của VĐK 83C152 như sau:

- Tương thích với cấu trúc của VĐK 8051.
- Có 8K ROM mặt nạ.
- Có cổng trao đổi tin nối tiếp nhiều thủ tục (SDLC/HDLC, CSMA/CD và SDLC/HDLC và các thủ tục khác) với tốc độ 2,048 tới 2,4Mbps.
- Trao đổi tin song công và đơn công.
- Tần số nhịp 16,5MHz.
- Gọi địa chỉ cho 64kB nhớ chương trình và dữ liệu.
- Có hai khối DMA với các tín hiệu yêu cầu HOLD/HLDA.
- Có 11 mức ngắt.

CÂU HỎI, BÀI TẬP VÀ THỰC HÀNH

I. CÂU HỎI

1. Nêu ưu và nhược điểm của trao đổi tin theo RS232C.
2. So sánh các chuẩn trao đổi tin nối tiếp RS.
3. So sánh trao đổi tin nối tiếp theo chuẩn RS của VĐK AT89C51RD2/ED2 với của VĐK 8051.
4. Trao đổi tin SPI của VĐK AT89C51RD2/ED2.
5. Trao đổi tin I2C VĐK ATmega8535.
6. Trao đổi tin theo USB của VĐK.
7. Trao đổi tin theo CAN với các VĐK và vi mạch ghép nối.
8. Trao đổi tin theo mạng của VĐK 83C152.

II. BÀI TẬP

1. Viết chương trình trao đổi tin nối tiếp theo RS232C của VĐK 8051.
2. Viết chương trình trao đổi tin nối tiếp theo RS232C của VĐK 8051 ở chế độ ngắt chương trình.
3. Viết chương trình trao đổi tin nối tiếp theo SPI cho VĐK 8051 với tham khảo chương trình cho ATmega8535.
4. Viết lại các chương trình ở các ví dụ trong chương theo lệnh gộp.

III. THỰC HÀNH

1. Dịch và chạy mô phỏng các ví dụ 17.1 tới 17.5 trong Pinnacle.
2. Chạy mô phỏng các ví dụ 17.1 tới 17.5 trong Proteus.

CHƯƠNG 18

GHÉP NỐI VỚI HỆ VI ĐIỀU KHIỂN

Mục đích của chương:

- Giới thiệu một số hệ vi điều khiển (VĐK) trong giảng dạy, đo lường, điều khiển, xử lý tin và truyền thông.
- Giới thiệu chương trình phục vụ cho hệ VĐK trong đó có chương trình giám sát.
- Giới thiệu một số VĐK loại khác với 8051/8052 dùng trong các hệ ứng dụng VĐK.

Trong các chương 1–10 của tập một, chúng ta đã nghiên cứu về cấu trúc, hệ lệnh, cách lập trình và các hành động (định thời, đếm, trao đổi tin song song và nối tiếp và ngắt chương trình) của VĐK thuộc họ MCS–51.

Trong các chương 11–17 của tập II này, chúng ta đã nghiên cứu cách ghép nối VĐK với các thiết bị ngoài (LED, ma trận diot, LCD, bàn phím) và mở rộng bộ nhớ ngoài, các cổng song song và nối tiếp của một hệ VĐK.

Trong chương 18 này, chúng ta sẽ nghiên cứu một số hệ VĐK trong các lĩnh vực học tập, đo lường, điều khiển, xử lý và truyền tin. Chương này sẽ ứng dụng các kiến thức của các chương trước (1–17) để xét một số ứng dụng cơ bản của VĐK trong đời sống, trong khoa học và kỹ thuật.

Trong quá trình trình bày, chúng ta sẽ làm quen với một số VĐK khác của họ MCS–51 và các họ khác, đồng thời cũng làm quen với một số ngôn ngữ và công cụ phần mềm khác trong khi xét về phần mềm phục vụ hệ.

18.1. HỆ VĐK TRONG THỰC HÀNH VĐK

Với mục đích giảng dạy và học tập, người ta đã nghiên cứu và chế tạo các giá thực nghiệm (KIT) hay dạng tấm (board) để thực hành về VĐK tại các trường đại học và cao đẳng dưới dạng các đề tài nghiên cứu, các luận văn (thạc sỹ, kỹ sư). Các công ty điện tử ở trong nước và nước ngoài cũng thiết kế, chế tạo và bán cho các kit trên.

Mục tiêu của các kit này là:

- Về phần cứng phải tường minh, đơn giản, giá rẻ và hấp dẫn người dùng.
- Về nội dung, phải học tập được các vấn đề cơ bản của VĐK về cấu trúc, lập trình, ghép nối và ứng dụng. Người ta đã xây dựng nên các bài thí nghiệm trên các kit này.

Tùy nội dung cần thực hành, người ta có thể chế tạo ra các kit khác nhau với thành phần cấu tạo khác nhau. Việc phân loại kit dựa theo tiêu chí là không (loại I) hay có bộ nhớ ngoài (loại II). Có thể có loại kit loại III trên cơ sở kit loại II để học tập, nghiên cứu và phát triển phần mềm cho các ứng dụng VĐK trong đời sống và sản xuất thuộc các lĩnh vực đo lường – điều khiển, xử lý và truyền thông.

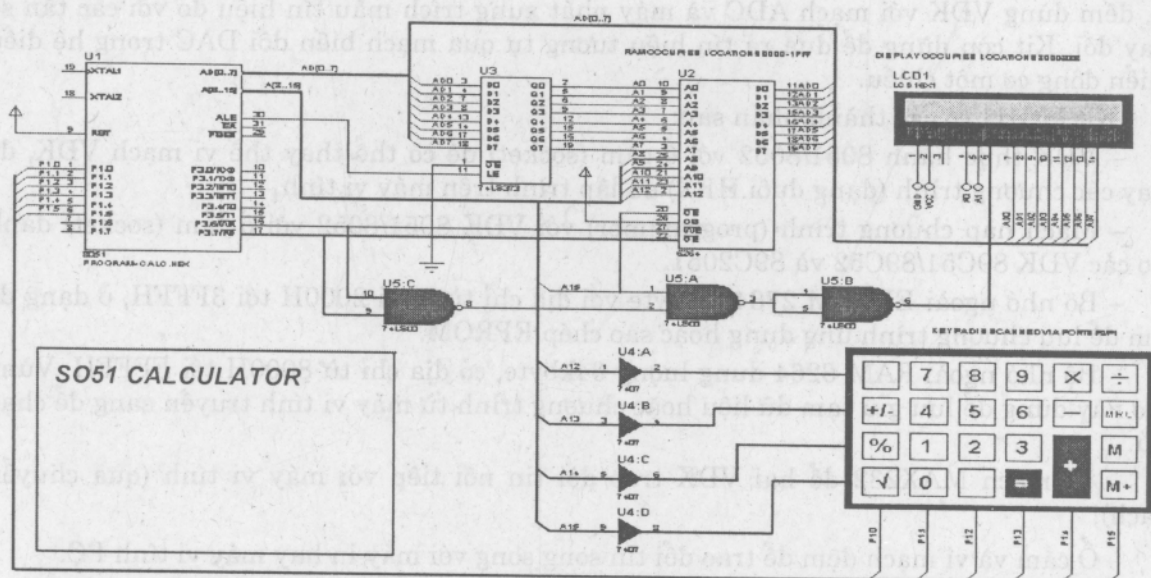
18.1.1. Các loại kit do các trường chế tạo

1. Kit của trường Đại học Sư phạm Kỹ thuật TpHCM

a) Cấu tạo kit

Kit loại I có các thành phần sau:

- VDK thuộc họ MCS-51 với ổ cắm (socket) để dễ thay thế vi mạch.
- Vi mạch đếm 74245 dữ liệu hai chiều điều khiển bởi 4 chuyển mạch (switch) nhỏ, với chiều vào (logic 1) hay ra (logic 1).
- Đi kèm với kit trên, có 6 mạch rời bên ngoài là 32 led đơn, 8 led 7 đoạn và bàn phím, ma trận LED, ADC 0809 và DAC để nghiên cứu 6 ứng dụng tương ứng.
- Kit loại II (hệ thống II) của trường Đại học sư phạm kỹ thuật TpHCM có thêm các thành phần sau:
 - Vi mạch nhớ ngoài1 EPROM 2746, 8kbyte với địa chỉ từ 0H tới 1FFFFH để lưu chương trình hệ thống. Nếu dùng VDK có nhớ Flash (như 89Sxx) sẽ dùng bộ nhớ nội bên trong, nên không cần vi mạch này.
 - Vi mạch nhớ ngoài2 EPROM 2746, 8kbyte với địa chỉ từ 0 tới 2000H tới 3FFFFH, ở dạng để cắm để lưu chương trình ứng dụng hoặc sao chép EPROM.
 - Bộ nhớ RAM 62256 dung lượng 32kbyte, có địa chỉ từ 8000H tới FFFFH. Vùng nhớ này dùng để lưu giữ tạm dữ liệu hoặc chương trình từ máy vi tính truyền sang để chạy thử.
 - Bàn phím có 24 phím, dùng IC 8279 để quản lý bàn phím với địa chỉ nhớ 4020H (cho dữ liệu) và 4021H (cho lời điều khiển).
 - Ba vi mạch 8255 dành để mở rộng cổng song song, mạch nạp bộ nhớ.
 - LCD trên kit.
 - Mạch nạp bộ nhớ EPROM với hai ổ cắm 40 chân (để nạp chương trình cho VDK) và 28 chân (để nạp chương trình cho bộ nhớ EPROM từ 2732 tới 27512).
- Hình 18.1 giới thiệu hệ VDK có mắc bộ nhớ ngoài, bàn phím, LCD trong mạch Calculator để giới thiệu một phần sơ đồ nguyên lý của kit loại II.



Hình 18.1. Sơ đồ nguyên lý của hệ có bộ nhớ ngoài của Calculator

b) Phần mềm phục vụ kit

VDK chỉ thực hiện được chương trình có đuôi .HEX đã viết từ máy vi tính.

Do đó, chương trình với ngôn ngữ Assembly phải được soạn thảo, hợp dịch, liên kết ra tệp .HEX và chạy thử bằng phương pháp mô phỏng. Sau đó chương trình đuôi .HEX phải được ghi vào bộ nhớ chương trình PROM bên trong VDK thực hành qua bộ ghi chương trình (programmer) bên ngoài hay ở trên kit.

Trường Đại học Sư phạm Kỹ thuật TpHCM đã tạo ra phần mềm SPKT_C để:

- Soạn thảo, dịch và liên kết chương trình từ dạng Assembly sang dạng .HEX có chỉ thị lỗi.
- Điều khiển đọc, nạp và xoá chương trình có chỉ thị lỗi trên các VDK 89C51/89C52 cũng như 89C2051 trên hai ổ cắm bằng trao đổi tín nối tiếp giữa máy tính PC và VDK trên mạch nạp.

Giao diện của SPKT_C trên màn hình máy vi tính có biểu tượng cho hành động biên dịch (compile) VDK 8951/52 và ghi chương trình vào các VDK 89C2051 và 89C2051 lắp trên các ổ cắm của bộ nạp trên kit.

c) Các bài thực hành

Với kit loại I và các khối thiết bị đặt bên ngoài, có thể thực hiện 6 ứng dụng với LED đơn (5 bài), với LED7 đoạn và bàn phím (9 bài), với LED ma trận (3 bài), với vi mạch ADC và DAC (4 bài), với động cơ bước công suất nhỏ (3 bài). Mỗi bài có các dạng khác nhau của chương trình.

Với kit loại II và các khối thiết bị đặt bên ngoài, có thể thực hiện được 4 bài với LCD và LED đơn, LED ma trận thông qua cổng mở rộng với vi mạch 8255A.

Với hai kit và phần mềm trên, người dùng có thể thực hành được các kiến thức cơ bản, cách ghép nối và ứng dụng của VDK 89C51/89C52 và 89C2051.

2. Kit của trường Đại học Sư phạm Kỹ thuật Nam Định

a) Cấu tạo kit

Trường Đại học Sư phạm Kỹ thuật Nam Định đã chế tạo kit MGH1 như là kit loại II và thêm các vi mạch ADC, DAC có trên kit. Như vậy, kit này có thể dùng để nghiên cứu về đo, đếm dùng VDK với mạch ADC và máy phát xung trích mẫu tín hiệu đo với các tần số thay đổi. Kit còn dùng để đưa ra tín hiệu tương tự qua mạch biến đổi DAC trong hệ điều khiển động cơ một chiều.

Kit MGH1 có các thành phần sau:

- VDK thực hành 8051/8052 với ổ cắm (socket) để có thể thay thế vi mạch VDK, để chạy các chương trình (dạng đuôi.HEX) đã lập trình trên máy vi tính.
- Mạch nạp chương trình (programmer) với VDK 8051/8052 với ổ cắm (socket) dành cho các VDK 89C51/89C52 và 89C2051.
- Bộ nhớ ngoài EPROM 2764, 8kbyte với địa chỉ từ 0 tới 2000H tới 3FFFH, ở dạng để cắm để lưu chương trình ứng dụng hoặc sao chép EPROM.
- Bộ nhớ ngoài RAM 6264 dung lượng 64kbyte, có địa chỉ từ 8000H tới FFFFH. Vùng nhớ này dùng để lưu giữ tạm dữ liệu hoặc chương trình từ máy vi tính truyền sang để chạy thử.
- Vi mạch MAX232 để hai VDK trao đổi tín nối tiếp với máy vi tính (qua chuyển mạch).
- Ổ cắm và vi mạch đệm để trao đổi tín song song với máy in hay máy vi tính PC.
- Bàn phím có 16 phím, nối trực tiếp với cổng song song mở rộng.
- Vi mạch 8255A dùng để mở rộng cổng song song.
- Mạch dao động thạch anh, nút nhấn RESET và chuyển mạch dùng trao đổi tín nối tiếp giữa máy vi tính với một trong hai VDK.
- 8 đèn LED đơn để hiển thị dạng nhị phân.

- Màn hình tinh thể lỏng LCD 7 đoạn để hiển thị số và chữ.
- Các vi mạch ADC và DAC để trao đổi tin với thiết bị ngoài tương tự.
- Máy phát xung vuông nhiều tần số để kiểm tra bộ đếm và trích mẫu cho tín hiệu vào tương tự.
- 8 chuyển mạch nhỏ để tạo các tín hiệu vào logic 0 hay 1.
- Các nguồn nuôi $\pm 12V$, $\pm 5V$ có ổn áp và chống nhiễu.
- Các ổ cắm nối các chân của các cổng song song P0, P1 và P2 để có thể ghép nối với thiết bị ngoài khác như động cơ bước v.v...

Như vậy, kit này thuộc loại II (có nhớ ngoài), có các thiết bị ngoài cơ bản (bàn phím, LED đơn, LCD) ở trên kit. Kit này còn có các vi mạch ADC, DAC và máy phát để giao tiếp với thiết bị ngoài tương tự. Nhược điểm cơ bản của kit này là không có LED7 đoạn để chỉ thị trực giác các số tự nhiên (LED đơn chỉ thị dạng nhị phân).

b) Phần mềm phục vụ kit

Để soạn thảo, hợp dịch, liên kết và chạy mô phỏng chương trình, kit trên dùng hai phần mềm có sẵn là Pinnacle và Ptoteus (có thể dùng chương trình hợp dịch loại khác). Sau hợp dịch và liên kết, ta có chương trình dạng đuôi.HEX để chạy mô phỏng hay chạy thật trên VDK sau khi nạp vào trên máy nạp. Với chuyển mạch cho mạch trao đổi tin nối tiếp, ta có thể nạp vào bộ nhớ RAM ngoài của VDK và chạy ngay chương trình.

Cần phải viết các chương trình phục vụ cho hệ với cấu tạo trên.

- Các chương trình con phục vụ các thiết bị ngoài như LED7 đoạn, LED ma trận, LCD, bàn phím, chương trình ghi và đọc khối nhớ RAM, chương trình trao đổi tin qua cổng song song, chương trình trao đổi tin qua cổng nối tiếp, chương trình điều khiển mô tơ bước, chương trình cho ADC và DAC. Các chương trình trên đã có trong các ví dụ ở các chương 11 tới 16 của tập II này).

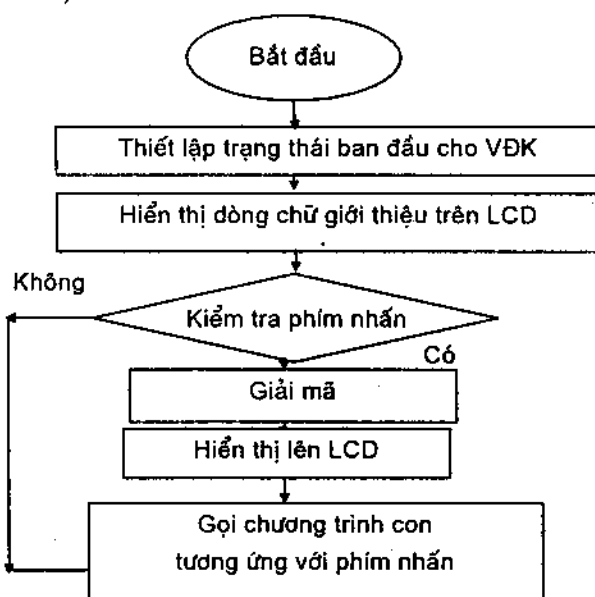
- Chương trình quản lý cả hệ hay giám sát (monitor) với thực đơn chính, các thực đơn con (xem chương bàn phím). Chúng ta sẽ đề cập đến chương trình quản lý này ở máy tính đơn tấm (SBC-Single Board Computer).

VDK nạp chương trình của kit phải có chương trình ghi bên trong để nhận tin nối tiếp từ PC rồi ghi vào VDK thực hành trên kit.

Với máy vi tính PC, điều khiển trao đổi tệp chương trình dạng .HEX bởi chương trình Term.exe.

VDK thực hành loại 89CXX phải tháo ra, đặt vào ổ cắm của bộ nạp chương trình.

VDK thực hành loại 89SXX với nhớ Flash, có thể nạp chương trình ngay trên ổ cắm của hệ (không phải tháo ra để lắp vào ổ cắm của bộ nạp).



Hình 18.2. Giải thuật của chương trình giám sát cho MGH1

Với bài tập trao đổi tin nối tiếp với máy vi tính PC, cần có chương trình nhận tin nối tiếp từ PC sang VDK và chương trình truyền tin nối tiếp của PC ra cổng COM2 nối với VDK.

Người thực hành trên kit có thể chưa cần nghiên cứu hai chương trình cho bộ nạp và trao đổi tin nối tiếp qua cổng COM2 của PC vì ngoài nội dung của thực hành về VDK.

Chương trình giám sát được viết để quản lý các chương trình con cho các bài thực hành.

Giải thuật của chương trình giám sát ở hình 18.2 chính là thực đơn để gọi một trong các bài thực hành hay các chương trình con phục vụ cho từng bài. Trên hình 18.3 việc gọi chương trình con tương ứng với phím nhấn chính là một chuỗi kiểm tra mã phím để gọi (hay nhảy) tới chương trình con tương ứng.

18.1.2. Hệ thiết bị thực tập của VIELINA

Phân viện nghiên cứu điện tử-tin học-tự động hoá Tp HCM (Vielina HCMC) đã chế tạo "thiết bị thực tập về vi xử lý (thực chất là vi điều khiển) MPE-300" với các khối thiết bị ngoại vi là MPE-301 tới MPE-307.

1. Cấu trúc hệ

Thiết bị có cấu trúc tương tự kit MGH1 nhưng có những điểm khác biệt sau:

- Có 4 đèn hiển thị LED7 đoạn.
- Có 4 rơ le điện tử, công suất nhỏ.
- Không có LCD và máy phát xung.
- Có các khối ADC (MPE-304), DAC (MPR-305), mô tơ bước (MPE-306) và ở bên ngoài.

2. Phần mềm phục vụ

Phần mềm phục vụ MPE có hai loại để sử dụng trong môi trường MSDOS hay WINDOWS.

Tài liệu viết sẵn 6 hàm chức năng đặc biệt (chương trình con) là:

- CONVERT_HEX: đổi mã HEX 00 nhớ 120 ra mã BCD, địa chỉ gọi là 2100H.
- DISPLAY : hiển thị mã cho LED7 đoạn, địa chỉ gọi là 2103H.
- CONVERT_REGISTER: đổi mã HEX ở hai ô nhớ (120 và 121) ra mã LED7 đoạn, địa chỉ gọi là 2106H.
- GET_KEY: kiểm tra phím nhấn, địa chỉ gọi là 2109H.
- DELAY_N: trễ tính theo N ms, địa chỉ gọi là 210CH.
- CONVERT_LED: đổi từ mã HEX ra mã cho LED7 đoạn, địa chỉ gọi là 2100H.

Các chương trình con trên sẽ được người dùng đưa vào chương trình của mình khi cần dùng hoặc tham khảo để viết chương trình.

Tài liệu về kit có các chương trình MPE300.ASM (kiểm tra toàn bộ kit), MPE304.ASM (cho khối ADC), MPE305.ASM (cho khối DAC), MPE306.ASM (cho motor bước), EP64_2.EXE (nạp chương trình vào VDK) và RAM2.EXE (tải chương trình dạng.BIN sang nhớ RAM trên kit).

Chương trình RAM2.EXE chạy trong MSDOS, dùng để tải chương trình (dạng.BIN) của người dùng từ PC sang bộ nhớ RAM kit với thực đơn trên màn hình với ba thao tác :

- Nhấn phím R của bàn phím của PC để truyền sang RAM.
- Nhấn phím F của bàn phím của PC để nhập file, sau khi đã nhập tên file của chương trình cần truyền vào mục Tên FILE (.bin): ở trên màn hình.
- Nhấn phím T của bàn phím của PC để thoát khỏi chương trình RAM2.EXE.

18.2. HỆ PHÁT TRIỂN PHẦN MỀM VÀ CHƯƠNG TRÌNH GIÁM SÁT

Hệ phát triển phần mềm là một hệ VDK được chế tạo để người dùng thực tập và viết các chương trình ứng dụng cho VDK. Tùy theo nội dung chương trình giám sát, ta gọi hệ này là một kit phát triển (Evaluation Board) hay máy vi tính đơn tấm (SBC– Single Board Computer), có thể nối với bàn phím của máy vi tính PC và truyền mã phím (ASCII) vào kit qua cổng nối tiếp. Có hệ còn dùng màn hình CRT của PC thay cho LCD. Hệ có VDK, các bộ nhớ ngoài dung lượng lớn (tối 64kbyte), có trao đổi tin nối tiếp theo cổng COM của PC.

Tùy số lệnh điều khiển và nhiệm vụ, người ta chia thành kit phát triển phần mềm.

18.2.1. Kit phát triển (EB)

1. Cấu tạo

Một EB là một kit thông dụng với một VDK, có thể tải (Download) và chạy (Go) chương trình của người dùng, ví dụ EB Mini51.

- Optional LCD BackLighting Control via port pin
- 10 pin connector for 5 x 5 matrix keyboard
- 14 (or 12) unused port pins are brought out on a connector
- On-board regulator to generate +5VDC (requires 9 Volts DC)
- Accompanying disk contains ISP programming software and working example programs written in C to access the serial port, LCD and keyboard
- I2C Compatible 64 KBytes EEPROM (24C512)
- I2C Compatible Real Time Clock (DS1307) with Lithium Battery
- (Only in Mini51–A831) on-chip 4KBytes EEPROM.
- (Only in Mini51–A831) 8 Analog Inputs (12 bit), 2 analog outputs (12 bit), temperature sensor

- LCD và bàn phím nối bên ngoài.

EB này thích hợp để kiểm tra nhiều chương trình của người dùng với dung lượng nhớ khác nhau (64K/96K/16K/62K).

Trên EB có:

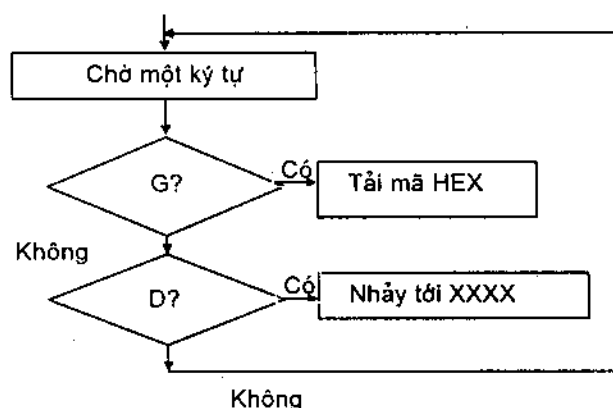
- 8 Analog inputs
- 8 bit resolution
- Analog input range 0 to +5V
- 2 Analog outputs with 8 bit resolution
- Analog output range 0 to +5V
- Serial input/output via I2C-bus
- Board size: 52mm x 32mm
- ADC/DAC driver routines (in C) available

2. Chương trình giám sát MINMON

Hệ MINI51 có chương trình giám sát MINMON (trên mạng) hai lệnh điều khiển:

- Lệnh D: dùng để tải (Download) file mã HEX (Download) vào hệ.
- Lệnh G: dùng để chạy (Go) chương trình.

MINMON có giải thuật như hình 18.3.



Hình 18.3. Giải thuật của thực đơn của MINMON

MINMON có chương trình chính và chương trình điều khiển “D” và “G” như dưới đây.

```

; * MINMON – The Minimal 8051 Monitor Program *
; * Rigel Corporation, of Gainesville, Florida *
stack equ 2fh ; bottom of stack
; – stack starts at 30h –
errorf equ 0 ; bit 0 is error status
;=====
; 8032 hardware vectors
;=====
org 00h ; power up and reset vector
ljmp start
org 03h ; interrupt 0 vector
ljmp start
org 0bh ; timer 0 interrupt vector
ljmp start
org 13h ; interrupt 1 vector
ljmp start
org 1bh ; timer 1 interrupt vector
ljmp start
org 23h ; serial port interrupt vector
ljmp start
org 2bh ; 8052 extra interrupt vector
ljmp start
;=====
; begin main program
;=====
org 100h
start:

```



```

clr .ea ; disable interrupts
lcall init
; initialize hardware
lcall print ; print welcome message
db 0ah, 0dh, "Welcome to 6.115!", 0ah, 0dh, "MINMON> ", 0h
monloop:
mov sp, #stack ; reinitialize stack pointer
clr ea; disable all interrupts
clr errorf ; clear the error flag
lcall print ; print prompt
db 0dh, 0ah, "*", 0h
clr ri; flush the serial input buffer
lcall getcmd ; read the single-letter command
mov r2, a ; put the command number in R2
ljmp nway ; branch to a monitor routine
endloop: ; come here after command has finished
sjmp monloop ; loop forever in monitor loop
=====
init:
mov tmod, #20h; set timer 1 for auto reload -- mode 2
mov tcon, #41h; run counter 1 and set edge trig ints
mov th1, #0fdh ; set 9600 baud with xtal = 11.059mHz
mov scon, #50h; set serial control reg for 8 bit data
; and mode 1
ret
=====
; monitor jump table ; bang nhay
=====
jumtab:
dw badcmd; command '@' 00
dw badcmd; command 'a' 01
dw badcmd; command 'b' 02
dw badcmd; command 'c' 03
dw downld; command 'd' 04 used
dw badcmd; command 'e' 05
dw badcmd; command 'f' 06
dw goaddr; command 'g' 07 used
dw badcmd; command 'h' 08
dw badcmd; command 'i' 09
dw badcmd; command 'j' 0a
dw badcmd; command 'k' 0b
dw badcmd; command 'l' 0c
dw badcmd; command 'm' 0d

```

```

dw badcmd; command 'n' 0e
dw badcmd; command 'o' 0f
dw badcmd; command 'p' 10
dw badcmd; command 'q' 11
dw badcmd; command 'r' 12
dw badcmd; command 's' 13
dw badcmd; command 't' 14
dw badcmd; command 'u' 15
dw badcmd; command 'v' 16
dw badcmd; command 'w' 17
dw badcmd; command 'x' 18
dw badcmd; command 'y' 19
dw badcmd; command 'z' 1a

```

```

=====
; command goaddr 'g'
; this routine branches to the 4 hex digit address which follows
=====

```

```

goaddr:
    lcall getbyt ; get address high byte
    mov r7, a ; save in R7
    lcall prthex
    lcall getbyt ; get address low byte
    push acc; push lsb of jump address
    lcall prthex
    lcall crlf
    mov a, r7 ; recall address high byte
    push acc; push msb of jump address
    ret ; do jump by doing a ret

```

```

=====
; command downld 'd'

```

```

downld:
    lcall crlf
    mov a, #'>' ; acknowledge by a '>'
    lcall sndchr
dl:
    lcall getchr ; read in ':'
    cjne a, #':', dl
    lcall getbytx ; get hex length byte
    jz enddl ; if length = 0 then return
    mov r0, a ; save length in r0
    lcall getbytx ; get msb of address
    setb acc.7 ; make sure it is in RAM
    mov dph, a ; save in dph

```

```

lcall getbytx ; get lsb of address
mov dpl, a ; save in dpl
lcall getbytx ; read in special purpose byte (ignore)
dloop:
lcall getbytx ; read in data byte
movx @dptr, a ; save in ext mem
inc dptr; bump mem pointer
djnz r0, dloop; repeat for all data bytes in record
lcall getbytx ; read in checksum
mov a, # '.'
lcall sndchr ; handshake '.'
sjmp dl ; read in next record
enddl:
lcall getbytx ; read in remainder of the
lcall getbytx ; termination record
lcall getbytx
lcall getbytx
mov a, # '.'
lcall sndchr ; handshake '.'
ljmp endloop ; return
getbytx:
lcall getbyt
jb errorf, gb_err
ret
gb_err:
ljmp badpar

```

Các chương trình con hỗ trợ của MINMON là badcmd (in lệnh sai), getbyt (nhận byte), getcmd (nhận lệnh điều khiển), nway (xử lý vào bảng ngắt).

Các chương trình con thông dụng của MINMON là sendchr (gửi ký tự), getchr (nhận ký tự) print (in ký tự), crlf (xuống dòng ở đầu dòng).

18.2.2. Máy tính đơn tấm SBC51

Máy tính đơn tấm (SBC) có một bộ nhớ chương trình và dữ liệu dung lượng lớn, để người dùng có thể kiểm tra các áp dụng lớn và phức tạp trên đó.

Nói chung SBC có nhiều các thiết bị vào/ra như bàn phím, bộ hiển thị, cổng song song, cổng nối tiếp tương tự như một máy tính để bàn, nhưng có thể là các thiết bị kích thước nhỏ như LCD.

1. Cấu tạo SBC51

SBC51 là một SBC có VDK họ MCS-51, có 4 mẫu VDK như sau:

- SBC51-A uses AT89C52 @ 11,059MHz (tốc độ chuẩn)
- SBC51-P uses P89C51RD2 @ 11,059MHz (tốc độ gấp đôi)
- SBC51-P9 uses P89C669 @ 11,059MHz (tốc độ gấp đôi)
- SBC51-U uses DS89C450 @ 11,059MHz (12xtốc độ)

- On-chip serial port available as RS232 port with DB9 connector.
- Second serial port (available only with SBC51-P9 and SBC51-U version) available as TTL or RS232.
- 64K ISP flash program memory (96KB in SBC51-P9).
- On-chip data XRAM 768 bytes (only with double speed option) / 1024 bytes (only with 12X speed option)
- On-board 124KB data memory, accessible as 2 banks of 62KB (in SBC51-A, SBC51-P, SBC51-U).
- SBC51-P9 has on-board 128KBytes continuous data memory (possible due to MX architecture).
- On-board PS/2 keyboard interface connector.
- On-board parallel printer port with DB25 female connector.
- 64KB I2C compatible EEPROM for non-volatile data storage.
- I2C compatible RTC with suitable Battery for clock/calendar.
- (Optional) Text LCD module – 16 char X 1 row or 16 char x 2 rows.
- Unused chip-select signals available for external I/O expansion.
- Suitable power supply included (requires 230VAC).
- Data bus and other necessary control signals available on connector for expansion.
- Unused (or optionally used) port pins available on connector.
- Board size: 135 mm X 115 mm.
- Optional add-on board (SS-AI12) provides 8 bipolar analog inputs (12 bit resolution).

2. Chương trình giám sát MON51

Chương trình giám sát MON51 có một chương trình chính MAIN và các chương trình con GETPAR (nhận thông số cho lệnh điều khiển), I/O (vào/ra), CONVERT (biến đổi mã), LOAD (nạp file), DUMPT (đưa ô nhớ ra hiển thị) SFR (đọc và ghi các thanh ghi SFR), IS (kiểm tra byte), SET (ghi vào ô nhớ) và MACRO (các lệnh nhảy viết bằng MACRO lệnh).

Chương trình chính có giải thuật như hình 18.4.

Lưu đồ này mô tả hoạt động của modul MAIN, đây là modul chính điều khiển hoạt động của toàn bộ chương trình. Chương trình hoạt động theo các bước sau:

- 1- Khai báo các biến, dành chỗ các vị trí nhớ, khởi động các thanh ghi
- 2- Khởi tạo con trỏ stack, gửi dấu nhắc ra VDT, chuẩn bị nhận lệnh
- 3- Nhận lệnh và giải mã lệnh
- 4- Thực hiện lệnh
- 5- Quay về bước 2

Chương trình có 4 lệnh điều khiển có các chữ sau chữ lệnh là các thông số của lệnh. Cụ thể là:

– Lệnh *D* hiển thị các ô nhớ với cú pháp:

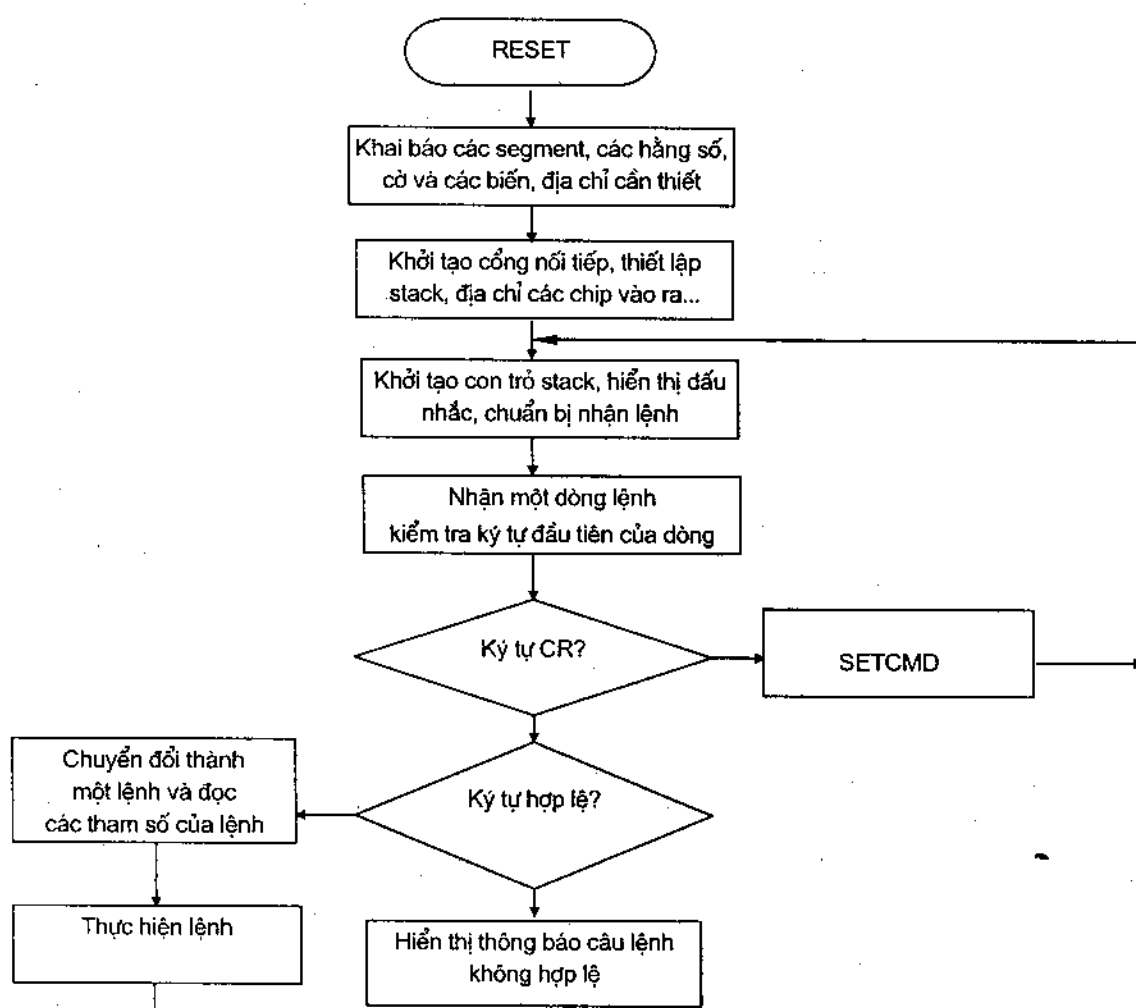
D <m><start><end><CR> trong đó:

+ *m* là bit chọn không gian nhớ là B (định địa chỉ bit), C (nhớ mã lệnh), I (nhớ dữ liệu bên trong), R (các thanh ghi SFR), X (bộ nhớ ngoài).

+ <start>: địa chỉ đầu.

+ <end>: địa chỉ cuối.

+ <CR> : phím Enter của bàn phím.



Hình 18.4. Lưu đồ của chương trình chính

- Lệnh *SET* ghi vào ô nhớ với cú pháp sau:
S <m><address><CR><option> với option có thể là <value> (giá trị là con số), <SP> (phím cách space), - (khảo sát vị trí trước đó) và <Q> (thoát và trở về dấu nhắc của MON51).
- Lệnh *GO* chạy chương trình với cú pháp:
GO <address><CR> trong đó <address> là địa chỉ đầu của chương trình.
- Lệnh *LO* nạp file với cú pháp:
LO <CR>

18.2.3. Máy tính đơn tấm SBC 8031SDK

1. Cấu trúc và sử dụng

Hãng HTE (HiTech Equipment Corporation) đã thiết kế và chế tạo một bo mạch có tên là máy tính đơn tấm SBC 8031SDK với :

- VDK 8031 không có nhớ ROM trong mà dùng nhớ ngoài 6264 có dung lượng 8kbyte.

- Vì mạch nhớ EPROM 2764/27128 dùng để nhớ chương trình giám sát (monitor) cho SBC.
- Nhớ RAM ngoài 6264/62256 dùng để nhớ chương trình của người dùng, được dịch sang đuôi.HEX để chạy trên VDK.
- Các vi mạch nhớ được chốt địa chỉ thấp (A0-A7) bởi vi mạch đệm 74HC573 qua cổng P0 và địa chỉ cao (A8.A15) được đưa trực tiếp qua cổng P2. Hệ có nhiều chuyển mạch dạng cầu nối để chọn các vùng nhớ.
- Cổng nối tiếp để nối với cổng COM2 của máy tính PC qua cáp dài bằng và ổ cắm DB-25.
- Có nhiều cầu nối để thay đổi tình thế thạch anh, dung lượng nhớ RAM, EPROM, chế độ chương trình (ngoài, trong) v.v...

Một số tính năng chính của SBC trên là:

- Hành động như máy tính đơn tấm đọc lập cũng như một ICE (In Circuit Emulator) 8032 đơn giản ở chế độ nhớ ngoài.
- Chương trình giám sát trong ROM dùng để phát triển chương trình.
- Nối với cổng COM của PC hay thiết bị đầu cuối CRT (màn hình).
- Chương trình được nạp và gỡ rối qua một đường dây nối tiếp trên.
- Chương trình của người dùng được nạp vào RAM tĩnh.
- Dịch hợp ngữ và cả ngôn ngữ bậc cao.

Do đó, người sử dụng dùng SBC trên để phát triển chương trình với chu trình có các bước sau:

- Trên PC, soạn thảo chương trình (tệp nguồn) khi dùng một bộ soạn thảo văn bản mã ASCII.
- Dịch chương trình nguồn sang dạng đối tượng (object) có đuôi.HEX bằng chương trình dịch Assembler hay compiler.
- Nạp tệp.HEX sang bộ nhớ chương trình SDK với lệnh LOAD (nạp).
- Chạy thử chương trình với các điểm dừng hay từng bước để tìm lỗi.
- Soạn thảo lại tệp nguồn để sửa lỗi và trở về bước 2 ở trên cho tới khi hết lỗi.

2. Chương trình giám sát (Monitor)

Chương trình giám sát này cũng tương tự MINON, MON51 nhưng có nhiều lệnh điều khiển hơn với các nhóm như sau:

- *Nhóm điều khiển với tổ hợp phím:*

- + <BS> : phím khoảng trống (Back space) dùng cho soạn thảo dòng để sửa lỗi.
- + <CR>: Carriage Return dùng để kết thúc chuỗi dòng điều khiển.
- + : phím Delet dùng cho dòng soạn thảo để sửa lỗi.
- + ^C (Control C): hủy lệnh và trở về chế độ điều khiển.
- + ^S (Control S): dừng cuộn màn hình.
- + ^Q (Control Q): khởi phát cuộn màn hình.
- + ^X (Control X): xoá toàn bộ dòng vào và bắt đầu chương trình nhập vào.

- *Nhóm lệnh dịch chương trình:*

- + ? : Hiện thị danh sách lệnh.
- + ASM [start_address] : hợp dịch thuật nhớ 8051 và hexadecimal vào bộ nhớ từ địa chỉ bắt đầu [start_address].
- + DASM [start_address], [end_address] : dịch ngược từ địa chỉ tới địa chỉ thành thuật nhớ và địa chỉ và dữ liệu dạng hexadecimal từ địa chỉ bắt đầu [start_address] tới địa chỉ cuối [end_address].

- *Nhóm chạy chương trình:*
 - + EXEC [start_address]: chuyển hoàn toàn điều khiển cho chương trình của người dùng.
 - + GO [start_address] [, break point]: bắt đầu thực hiện chương trình ở địa chỉ đầu tới điểm dừng.
 - + HELP: cho danh sách lệnh.
 - + S [number _ of _ step]: chạy chương trình từng bước.
 - + TRACE: theo dõi chạy chương trình bắt đầu ở địa chỉ đầu. Hiển thị nội dung các thanh ghi chính và mã lệnh của chương trình.
 - + VIEW: xem ngắt trong khi chạy chương trình.bật hay tắt.
 - + INTR [interrupt_number]: thay đổi ngắt gỡ rối được dùng bởi monitor cho ngắt chỉ định.
 - + LOAD [offst_address]: tải một file dạng HEX Intel từ cổng nối tiếp vào vùng nhớ RAM ngoài.
 - + PC [neww_value]: hiển thị và cho phép thay đổi giá trị của đếm chương trình (PC),
 - + PS: chuyển giữa theo dõi gọi chương trình con và bỏ qua nó ở chế độ từng bước.
 - + MEM: các hành động chuyển mạch U4 giữa mã tách biệt và chồng lên nhau và truy cập ngoài.
 - + HEX address, offset: cộng offset vào địa chỉ rồi hiển thị ra dạng HEX. Cũng trừ offset vào địa chỉ rồi hiển thị ra dạng HEX.
- *Nhóm lệnh thao tác vùng nhớ:*
 - + Hiển thị nhớ dữ liệu của mã lệnh (DC [start_address],[end_address]), nhớ nội (DI [start_address],[end_address]), thanh ghi (DR), nhớ ngoài (DX[start_address], [end_address]).
 - + Lấp đầy bộ nhớ cho mã lệnh của người dùng (FC), cho nhớ nội (FI), cho nhớ ngoài (FX).
 - + Di chuyển byte trong bộ nhớ dữ liệu lệnh (MC start_address, end_address, new_address), trong nhớ ngoài (MX start_address, end_address, new_address).
 - + Hiển thị và cho phép thay đổi một loạt nhớ có địa chỉ theo bit (SB bit_address), từ mã dữ liệu lệnh (SC address), từ nhớ nội (SI address), từ nhớ ngoài (SX address).

18.2.4. Chương trình chính có thực đơn truyền nối tiếp

Các thực đơn của các chương trình giám sát ở trên với lưu đồ ở hình 18.4 có các lệnh điều khiển được gõ vào từ bàn phím của máy vi tính PC rồi truyền nối tiếp sang kit VDK hay SBC. Khác với các chương trình giám sát ở trên có lệnh điều khiển theo các ký tự (mã ASCII), chương trình chính của chương trình giám sát có thể là các chữ số "1", "2", "3", "4", "5" (dạng mã ASCII là 30H, 31H, 32H, 33H, 34H và 35H).

\$MOD 824 FLAG BIT 00H FIRSTGO BIT 01H LED EQU P3.4 DSEG ORG 060H COUNT: DS 1 DACOUT: DS 1 DPH1: DS 1 DPL1: DS 1 CSEG	JB FLAG, \$ JMP SKIP2 SKIPI: CLR FIRSTGO MOV DPTR, #TITLE CALL SENDSTRING MOV DPTR, #LINKS1 CALL SENDSTRING MOV DPTR, #LINKS2 CALL SENDSTRING SKIP2: MOV DPTR, #MENU
---	---

<pre> ORG 00H JMP MAIN ORG 03H CLR FLAG RETI ORG 60H ; MAIN: MOV SP, #80H ; cấu hình UART MOV RCAP2H, #0FFH MOV RCAP2L, #-5 MOV TH2, #0FFH MOV TL2, #-5 MOV SCON, #52H MOV T2CON, #34H SETB IT0 SETB EX0 SETB EA SETB FLAG MOV A, #01H SETB TI MOV DPTR, #GOMENU CALL SENDSTRING SETB FIRSTGO BLINK: CPL LED CALL DELAY JB FLAG, BLINK SETB LED MAINMENU: JB FIRSTGO, SKIP1 MOV DPTR, #RETMENU </pre>	<pre> CALL SENDSTRING SETB FLAG CALL SENDSTRING JNB RI, \$ MOV A, SBUF CLR RI CJNE A, #"1", \$+6 JMP TEST1 CJNE A, #"2", \$+6 JMP TEST2 CJNE A, #"3", \$+6 JMP TEST3 CJNE A, #"4", \$+6 JMP TEST4 CJNE A, #"5", \$+6 JMP TEST5 JMP MAINMENU TEST1: JMP MENU TEST2: JMP MENU TEST1: JMP MENU TEST3: JMP MENU TEST4: JMP MENU TEST5: JMP MENU ; SUBROUTINE END </pre>
--	--

18.3. HỆ NGHIÊN CỨU VÀ ỨNG DỤNG VỚI ADC VÀ DAC

Hệ VDK này có vi mạch ADC và DAC dùng để ứng dụng và nghiên cứu khoa học.

Một số kit loại II ở trên, ví dụ kit MGH1 cũng đã đưa vào các vi mạch ADC và DAC độ phân giải nhỏ (8 hay 12 bit), để học tập hay nghiên cứu ở mức độ chính xác chưa cao.

Để nghiên cứu và ứng dụng đòi hỏi độ chính xác cao, người ta dùng các ADC và DAC có 16 bit, 24 bit.

Các khối ADC và DAC có thể có sẵn trong vi mạch VDK chuyên dùng hơn như PIC 16F87/88(10bit), SAB 80C515A (8bit), SAB 80C8535 (10bit) của hãng Siemen, ADU-12 (12 bit), ADU-16 (16 bit), ADU-24 (24 bit) của Microchip, AT8535, ATCX051, ATmega8535 (10bit) của ATMEL và một số VDK khác.

Trong các kit nghiên cứu và ứng dụng, người ta dùng các vi mạch ADC rời bên ngoài với số bit cao hơn (12 bit) như ADC MAX1290, DAC MAX5007 của hãng Motorola.

Dưới đây xin giới thiệu hai kit loại trên.

18.3.1. Kit 52

Kit này có tên là kit vào ra tương tự và số thông dụng cho VDK 8052 với ký hiệu Kit52.

Kit gồm:

- VDK 8052.- vi mạch ADC Max186 với độ phân giải 12 bit, có 8 lối vào tương tự.
- Các vi mạch L297 và L298 để điều khiển động cơ bước.
- Một cổng ra số 8 bit công suất lớn dùng để điều khiển động cơ bước.
- Một cổng trao đổi tin nối tiếp.
- Một lối ra tương tự (từ DAC) dùng để điều khiển thiết bị ngoài tương tự như động cơ một chiều, máy vẽ phổ v.v...

Về phần mềm phục vụ kit, có một số chương trình ngôn ngữ C với dạng:

Void Kit52_Init (void);

WORD Kit52_RdMax186 (cho byte điều khiển).

Void Kit52_WrMaxx541 (cho lời dữ liệu)

Void Kit52_Stepper (điều khiển động cơ bước).

Người dùng có thể dùng các lệnh trên để viết chương trình cho các ứng dụng về đo lường trong hệ đo 8 kênh, điều khiển thiết bị ngoài tương tự (động cơ một chiều và động cơ bước).

18.3.2. Kit IAEA-52

Kit này do Cơ quan năng lượng nguyên tử quốc tế chế tạo (nên có tên IAEA-International Atom Energy Agency) trên cơ sở VDK 8052. Kit này còn có tên là kit Vào/ra số/tương tự và SCA/MCA (Single Channel Amplitude/Multi Channel Amplitude – phân tích biên độ đơn kênh và đa kênh) thông dụng. Ngoài phân tích năng lượng (biên độ của bức xạ hạt nhân), kit còn phân tích biên độ của một tín hiệu điện tương tự bất kỳ, điều khiển động cơ bước, điều khiển thiết bị ngoài số, điều khiển thiết bị ngoài tương tự.

1. Cấu tạo

Kit có các khối sau:

- Vi điều khiển 8052.
- LCD với ghép nối.
- Vi mạch ADC MAX 1290 (8 kênh, 12 bit) cho ADC trung bình, phân tích chậm.
- Vi mạch ADC AD9224 (12 bit, 40MHz) cho ADC nhanh, phân tích chính xác.
- Hai SCA (phân tích đơn kênh) với các tín hiệu ngưỡng trên và dưới.
- Hai SCA (phân tích đơn kênh) với các tín hiệu TTL
- Khối điều khiển động cơ bước.
- Khối ghép nối bàn phím PS2 của máy vi tính.

Đặc điểm chính của kit IAEA như sau:

- VDK 8052 cho cổng USB, RS232, 32KB RAM và 8KB EEPROM nối tiếp.
- ADC MAX 1290 có 8 kênh, 12 bit, tần số xung trích mẫu 10kHz.
- DAC MAX5307 có 12 bit, tần số xung biến đổi 10kHz.
- Lối ra số, 16 bit, mức TTL.
- Lối vào số, 16 bit, mức TTL.
- Lối vào/ra số hai chiều, 16 bit, mức TTL.
- Ghép nối hiển thị LCD 8 bit thông dụng.
- Ghép nối bàn phím PS2.

- Các lối vào SCA với mức cao và thấp có thể điều khiển bằng chương trình.
- Các lối vào đếm cho SCA mức TTL.
- Điều khiển động cơ bước với vi mạch L297 và driver với vi mạch L298.
- Tùy chọn ADC nhanh (40MHz), một kênh với vi mạch AD9224 cho các áp dụng MCA.

2. Phần mềm

Kit IAEA_52 có thể lập trình với ngôn ngữ Assembly và C, nhưng ngôn ngữ C thích hợp hơn và có sẵn nhiều chương trình con. Muốn sử dụng chúng, chỉ cần khai báo ở đầu chương trình lệnh:

```
#include <EzUsb52.h> cho hoạt động của cổng USB.
```

```
#include <IAEA52.h> cho hoạt động của IAEA52.
```

Trong tài liệu có chương trình mẫu cho việc thu thập và biểu diễn kết quả đo phổ năng lượng bức xạ của SCA và MCA trên màn hình với phần mềm thiết bị ảo Labview.

18.4. HỆ ĐẾM, ĐO VÀ PHÂN TÍCH

VĐK đã được ứng dụng nhiều trong lĩnh vực khoa học và đời sống. Dưới đây, ta giới thiệu một số ứng dụng điển hình trong lĩnh vực đo lường, điều khiển, tích lũy tin và đời sống. Về cấu tạo phần cứng, các hệ đều có các loại VDK khác nhau tùy yêu cầu sử dụng, chỉ khác nhau về mạch đầu vào, mạch đầu ra và cách điều khiển hệ hoạt động cũng như phần mềm phục vụ. Với khuôn khổ của tài liệu, chúng ta chỉ nêu những nét chính, những gợi ý cho người đọc hình dung được cấu tạo, chương trình phục vụ. Ai cần tìm hiểu thêm hay nghiên cứu thiết kế có thể tìm đọc các tài liệu gốc.

18.4.1. Hệ đếm

VĐK có bộ đếm (Counter) nên dùng để đếm số xung điện đi vào chân T0 hoặc T1 hay T2 (cho 8052). Xung điện này có thể đưa ra từ máy phát xung (đo hay đếm tần số), từ mạch thu bức xạ (hạt nhân, tia X, hồng ngoại) hay từ mạch thu và xử lý để đếm người, đếm sản phẩm qua băng truyền, đếm số bức xạ truyền qua một vật thể để suy tính năng vật lý của vật như đo độ tro của than, độ ẩm của môi trường v.v...

1. Máy đo liều lượng bức xạ hạt nhân

Ủy ban An toàn hạt nhân quốc gia Việt Nam đã chế tạo máy đo độ bức xạ hạt nhân bằng phương pháp đếm số hạt bức xạ trung bình trong đơn vị thời gian. Máy gồm có VDK, đầu dò bức xạ khác nhau tùy loại bức xạ (tia X, alpha, beta, gamma, nơ tron), LCD, một số phím chức năng. Máy đếm số xung vào đầu dò (detector) trong một số thời gian (định sẵn bằng phím nhấn) rồi chia cho thời gian để tính liều lượng trung bình bức xạ phát ra trong một đơn vị thời gian (giây). Bằng phím nhấn và chương trình định sẵn, có thể đếm số bức xạ trong thời gian định sẵn hoặc đếm thời gian theo lượng bức xạ định sẵn hoặc đếm số bức xạ tổng cộng (máy đo liều) mà một người đã nhận trong nhiều ngày v.v....

2. Máy đo bằng phương pháp đếm

Viện kỹ thuật hạt nhân Hà Nội đã nghiên cứu, thiết kế và chế tạo (đề tài cấp bộ) máy đo độ tro của than bằng phương pháp đếm bức xạ gamma khi truyền qua than trên băng truyền. Với máy đo độ ẩm của mặt đường nhựa, sẽ đếm số hạt bức xạ khi tán xạ trên mặt đường.

Cấu trúc của máy gồm:

- Nguồn bức xạ gamma (Cs^{137}).
- Đầu dò bức xạ tia gamma.

- Nguồn nuôi đầu dò bức xạ (cao áp, cỡ 400V cho ống đếm Geiger Muller).
- Bộ thu thập và tạo dạng xung đếm (tiền khuếch đại, khuếch đại, tạo dạng)
- VDK 8052 hoạt động ở chế độ đếm.
- Các bộ nhớ ngoài EPROM, RAM với bộ giải mã (vi mạch 74138).
- Vi mạch điều khiển cổng nối tiếp MAX232.
- Bàn phím điều khiển bởi vi mạch 74C923.
- LCD.

Máy đếm trên có thể trao đổi tin nối tiếp với máy vi tính PC để nạp chương trình từ PC sang nhớ EPROM (nếu dùng vi mạch loại 80SXX hay AT90C52) và dữ liệu từ bộ nhớ RAM sang PC. Khi đi đo ngoài hiện trường, dùng nguồn nuôi là pin. Có thể dùng NVRAM để lưu giữ dữ liệu khi tắt nguồn nuôi.

Máy hoạt động với thực đơn với hai chức năng đếm và trao đổi tin nối tiếp. Mỗi chức năng trên lại có thực đơn nhỏ cho các nhiệm vụ chi tiết hơn. Hình 18.15 giới thiệu thực đơn chính của máy đếm với:

- *Chữ A*: chuyển tới hành động đếm.
- *Chữ B*: chuyển tới trao đổi tin nối tiếp.

Thực đơn con của Set_counter có 6 nhiệm vụ, trong đó có lệnh thoát trở về thực đơn chính (F), là:

- *Chữ A*: xác lập số đếm ban đầu.
- *Chữ B*: bắt đầu đếm.
- *Chữ C*: xoá bộ đếm.
- *Chữ D*: đếm nhiều lần.
- *Chữ E*: xem nội dung số đếm.
- *Chữ F*: thoát trở về thực đơn chính.

Thực đơn con của Set_serial có hai nhiệm vụ nên tương tự thực đơn chính (trừ lệnh F) là:

- *Chữ A*: gửi dữ liệu vào PC.
- *Chữ B*: nhận dữ liệu từ PC.
- *Chữ F*: thoát trở về thực đơn chính.

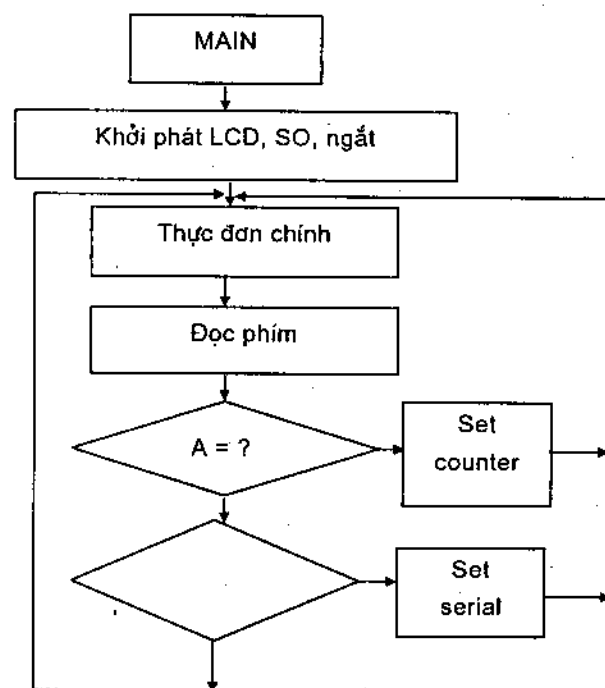
Trước khi đo với máy đếm trên, phải đo các mẫu chuẩn và vẽ đường biểu diễn thực nghiệm. Dữ liệu đo gửi về máy vi tính PC để tính độ trôi theo đường cong chuẩn thực nghiệm với các phép tính phức tạp.

Đồng hồ số cũng là một ứng dụng của hệ đếm dùng VDK. Nếu cần đếm thời gian chính xác, cần có mạch RTD của Dallas có pin bên trong để giữ nhịp cho VDK đếm thời gian.

18.4.2. Hệ đo biên độ tín hiệu điện có VDK

1. Cấu tạo

Muốn đo một đại lượng vật lý (nhiệt độ, độ ẩm, lực v.v...) ta phải có:



Hình 18.5. Thực đơn chính của máy đếm

- + Mạch biến đổi ADC với các khối sau:
 - Biến đổi đại lượng vật lý trên thành tín hiệu điện bằng bộ cảm biến (sensor), bộ phát hiện (detector) hay ống đếm các hạt bức xạ (hồng ngoại, tia X, gamma, beta, anpha v.v...).
 - Chuẩn biên độ tín hiệu xung có biên độ A tỷ lệ với độ lớn của đại lượng vật lý trong dải 0–5V (mức TTL) và thời gian kéo dài ngắn bằng các mạch điện tử.
 - Phát hiện đỉnh của biên độ cực đại của tín hiệu (detect pik) bằng mạch điện dùng diot.
 - Trích mẫu và giữ biên độ cực đại của tín hiệu bằng mạch S/H (Sample/ Hold) bởi một tín hiệu xung strobe (đốt cửa).
 - Biến đổi biên độ cực đại của tín hiệu trên thành số đếm N tỷ lệ với biên độ S bởi mạch ADC (biến đổi biên độ A thành số D).
 - Đưa số đếm biên độ D vào cổng song song của VDK để ghi nhớ trong bộ nhớ RAM.
- + Mạch hiển thị số đếm N trên các LED7 đoạn hay bằng chữ số trên LCD.
- + Mạch trao đổi tin nối tiếp với máy vi tính PC.

Sơ đồ khối một mạch đo một đại lượng vật lý như ở hình 18.6.

Với các đại lượng vật lý khác nhau, ta phải dùng các cảm biến thích hợp khác nhau.

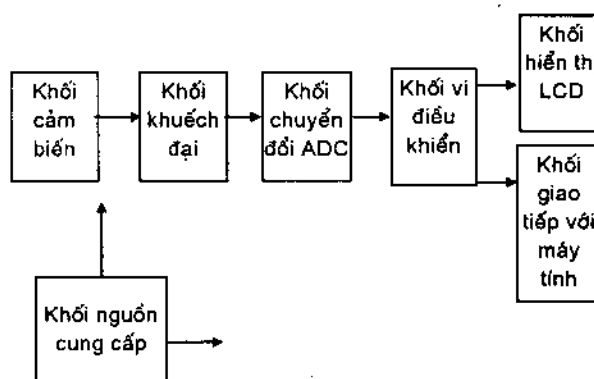
Tùy yêu cầu của phép đo (độ chính xác và thời gian phân giải), ta phải chọn khối ADC có thông số thích hợp (số bit, tần số trích mẫu). Một số VDK có sẵn ADC ở bên trong, ta phải chọn VDK có thông số về ADC thích hợp.

Ta có thể dùng kit cho nghiên cứu và ứng dụng Kit52 hay IAEA_52 ở trên.

Dữ liệu của từng phép đo có thể hiển thị trên LED7 đoạn hay LCD.

Dữ liệu của nhiều phép đo có thể được vẽ thành dạng phổ phân bố biên độ (hàm $N = f(A_j)$) cho máy phân tích đơn kênh (một biên độ) SCA và cho máy phân tích nhiều kênh (nhiều biên độ) MCA.

Chúng ta có thể dùng các kit trên thay cho hai khối ADC và khối vi điều khiển ở hình 18.6 trong các hệ đo phổ biên độ.



Hình 18.6. Sơ đồ khối của hệ đo dùng VDK

2. Phần mềm phục vụ

Trong phần mềm phục vụ hệ đo dùng ADC, ta phải có:

- Chương trình khởi phát VDK
- Chương trình điều khiển chọn cảm biến hay lối vào của ADC nhiều lối vào.
- Chương trình điều khiển ADC với phát tín hiệu start và nhận tín hiệu kết thúc biến đổi (EOC–End Of Convert).
- Chương trình tích lũy dữ liệu vào vùng nhớ dành cho phép đo.
- Chương trình xử lý kết quả như lọc nhiễu, vẽ đồ thị phân bố các biên độ đo được theo thời gian hay phổ phân bố biên độ.
- Nếu VDK ghép nối với máy vi tính PC, cần có chương trình trao đổi tin với PC (nhận lệnh từ bàn phím theo thực đơn, gửi dữ liệu về PC, hiển thị dữ liệu dạng số hay dạng phổ lên màn hình của máy vi tính, chương trình xử lý phổ (trừ nhiễu, điều khiển con trỏ

ghi vị trí hay kênh đầu và cuối, lấy diện tích phổ đã đánh dấu kênh đầu và kênh cuối, đo đỉnh phổ, tính năng lượng của đỉnh phổ theo phổ chuẩn đưa vào PC làm mẫu v.v...

18.4.3. Hệ đo thời gian dùng VDK

VDK còn dùng để đo thời gian khi cho tín hiệu bắt đầu Start cho lối vào /INT0,1 lên cao và tín hiệu Stop cho lối vào /INT0,1 xuống thấp (xem mục 8.6 trang 157 của tập một).

Cũng có thể đo nhiều khoảng thời gian (xem mục 8.6 của tập một) và vẽ phổ phân bố thời gian tương tự như phổ biên độ. Người ta dùng phổ thời gian này để biết được phổ năng lượng theo thời gian bay ($N = f(t)$ hay $N = f(E_i)$) của các hạt neutron trong vật lý hạt nhân.

18.4.4. Hệ đo nhiều thông số dùng VDK

Ngoài đo biên độ A và thời gian T ở trên, còn có các phép đo sau:

- Đo đồng thời nhiều biên độ trong khoảng thời gian nào đó hay phép ghi nhớ dạng xung trong các dao động ký kiểu số.

- Sự phân bố số đếm theo không gian với nhiều detector bức xạ trong việc theo dõi sự phân bố theo không gian (góc) của tia vũ trụ, tia bức xạ.

- Đo đồng thời phổ biên độ, phổ thời gian và phổ phân bố theo không gian trên được gọi là phép đo nhiều thông số hay nhiều chiều (xem Kỹ thuật ghép nối máy vi tính tập hai của cùng tác giả).

18.5. HỆ ĐIỀU KHIỂN

Ngoài đo lường các đại lượng vật lý, VDK còn dùng để điều khiển thiết bị ngoài.

18.5.1. Cấu tạo chung một hệ điều khiển dùng VDK

Hệ điều khiển dùng VDK là hệ dùng VDK đưa tín hiệu ra các thiết bị ngoài là các cơ cấu chấp hành như rơle, van điện từ, động cơ bước, động cơ một chiều (servo), động cơ ba pha v.v...

Tùy thiết bị ngoài là thiết bị loại số hay tương tự, ta có điều khiển số hay tương tự.

Ngoài mạch điều khiển, hệ điều khiển còn có mạch đo thông số phản ánh tác động của điều khiển. Người ta dùng thông số phản hồi này để quyết định lối ra của mạch điều khiển nên mạch trên có tên là mạch điều khiển theo vòng kín (có phản hồi). Mạch điều khiển vòng kín này dùng để duy trì thông số ra không đổi nên có tên là mạch điều chỉnh hay tự động duy trì, ví dụ, điều chỉnh để duy trì tốc độ của động cơ.

Do đó, mạch điều khiển dùng VDK có cấu trúc như hình 18.7, gồm:

- VDK có các tín hiệu vào để điều khiển cường bức và cổng trao đổi tín nối tiếp nối với cổng RS232C của máy vi tính PC.

- Lối ra số để điều khiển các cơ cấu chấp hành hay thiết bị ngoài số.

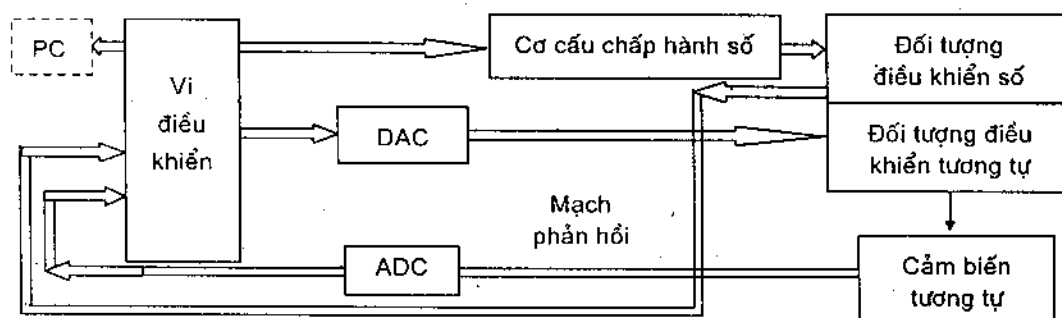
- Lối ra số đưa qua mạch DAC để điều khiển cơ cấu chấp hành hay thiết bị ngoài tương tự.

- Tín hiệu đo thông số của đối tượng điều khiển được đưa về lối vào của VDK. Nếu tín hiệu này là tương tự, phải biến đổi thành dạng số bởi mạch ADC.

Trong VDK có thể có mạch ADC (PIC, SBA8535, ATmega8535 v.v...) nên hệ điều khiển trên chỉ cần mạch DAC. Nếu VDK có cả hai mạch ADC và DAC như ADUC816, ADUC824, ta không cần dùng các mạch ADC, DAC ngoài.

Vì trong hệ điều khiển cần VĐK, mạch DAC và ADC nên có thể dùng Kit52 và Kit IAEA52 ở trên để tạo nên hệ điều khiển với VĐK.

Nếu cơ cấu chấp hành cần điện áp hoạt động lớn hơn 5V và tiêu thụ dòng lớn, lối vào của cơ cấu chấp hành phải có mạch driver (khuếch đại thế, khuếch đại dòng) cho phù hợp.



Hình 18.7. Sơ đồ khối của hệ điều khiển (số tương tự) có phản hồi

18.5.2. Hệ điều khiển số với động cơ bước

Động cơ bước là một thiết bị số, là động cơ chạy từng bước một, theo một góc quay xác định tùy cấu tạo dưới tác động của xung điện đưa vào hai cực của các cuộn dây tạo từ trường quay của stator. Đơn giản nhất là động cơ bước có 4 cuộn dây với 4 đầu nối A, B, C, D. Rotor của động cơ bước là một nam châm vĩnh cửu.

Động cơ bước rất hay dùng trong các thiết bị điều khiển tự động. Mạch điều khiển mô tơ bước (với VĐK PIC 16F84A và driver ULN2003A) cùng chương trình có trong tệp Proteus/Sample/Motor Example/Pic Stepper. Chúng ta có thể chạy mô phỏng mô tơ bước với chương trình trên.

Trong máy khoan, ta dùng ba mô tơ bước để điều khiển mũi khoan dịch chuyển theo không gian ba chiều.

18.5.3. Hệ điều khiển tương tự với động cơ một chiều (servo)

Động cơ một chiều là động cơ chạy liên tục nhiều vòng khi cho dòng một chiều vào cuộn dây kích thích từ thông.

Hai cách điều khiển động cơ servo là điều khiển theo điều chế độ rộng xung điện kích thích và điều khiển bằng điện thế một chiều.

1. Điều khiển theo điều chế độ rộng xung điện (PWM-Pulse Wide Modulation)

Khi cho chuỗi xung điện với tần số khác nhau, ta có biên độ trung bình của chuỗi xung trên khác nhau. Nếu đặt chuỗi xung trên vào cuộn dây kích thích từ trường quay của servo, ta có tốc độ của động cơ khác nhau. Mạch và chương trình điều khiển servo theo PWM có trong tệp Proteus/ Sample/Motor Example/HC11PWM. VĐK điều khiển servo này là MC68HC11A8. Tốc độ của servo được hiển thị trên 4 LED7 đoạn. Chúng ta có thể chạy mô phỏng servo với chương trình trên để quan sát.

2. Điều khiển theo điện thế một chiều

Một tín hiệu số đưa ra từ VĐK qua mạch DAC sẽ biến thành tín hiệu tương tự. Tín hiệu tương tự trên đưa vào cuộn dây kích thích từ trường sẽ làm servo quay.

Mạch và chương trình điều khiển servo qua DAC có trong tệp Proteus/ Sample/ Motor Example/

Servo. Ở đây, tín hiệu một chiều được lấy trên một chiết áp có dòng điện đi qua, có thể thay bằng mạch VDK và DAC có lối ra dùng để đưa vào lối vào cuộn dây kích thích từ. Chúng ta có thể chạy mô phỏng servo với chương trình trên để quan sát.

3. Bộ điều chỉnh PID với điều khiển tốc độ động cơ một chiều

Một hệ tự động được mô hình hoá như hình 18.8, trong đó bộ điều chỉnh làm nhiệm vụ điều chỉnh hệ khi có tín hiệu vào $x(t)$ cho tín hiệu ra $y(t)$ và tín hiệu vào bộ điều chỉnh $e(t) = x(t) - z(t)$ cho tín hiệu ra $u(t)$.

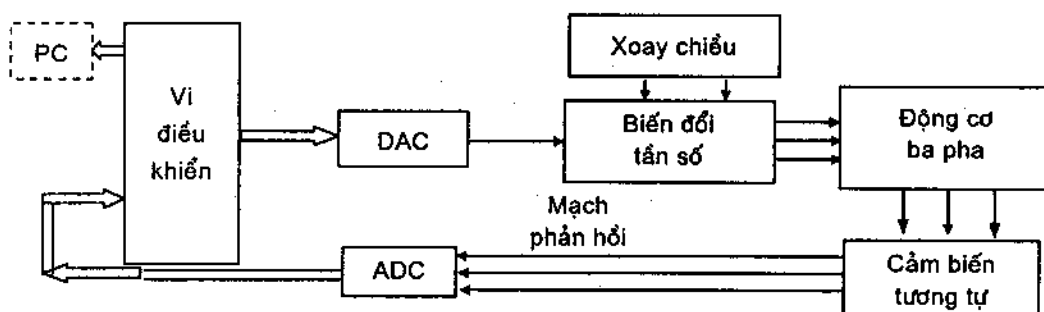
Nghiên cứu hệ tự động bằng cách nghiên cứu bộ điều chỉnh có hàm truyền đạt $G_1(s)$ sao cho thích hợp để hàm ra $y(t)$ là không đổi khi có hàm vào $x(t)$.

Tính giá trị của bộ điều chỉnh PID cho giá trị $U(t)$ là tỷ lệ, tích phân, vi phân của tín hiệu vào $e(t)$ có thể dùng một trong ba cách sau:

- Dùng MATLAB
- Dùng mô đun mềm PID trong Step 7
- Tự nghiên cứu chế tạo mô đun PID số để tính toán hệ số điều chỉnh cho bộ điều chỉnh.

Mô đun PID số có các thành phần sau:

- Một VDK 89S8252 có cổng nối tiếp với PC qua vi mạch MAX232
- Vi mạch nhớ ngoài EPROM 2764 và nhớ RAM 6264.
- Một mạch giải mã 74138.
- Hai cổng song song mở rộng 8255A.
- Một vi mạch ADC0809
- Một vi mạch DAC 0808.
- Một LCD để hiển thị chữ.
- Bốn đèn LED 7 đoạn để hiển thị số dạng thập phân.
- Bàn phím được điều khiển bởi VDK AT2005, với 16 phím được dùng như sau:
 - + Các số tự nhiên từ 0 tới 9
 - + Exit : thoát về thực đơn chọn Mode (chế độ)
 - + Kit : để không cho phép kết nối với máy tính PC.
 - + PC: kết nối với PC.
 - + About: thông tin thêm về tác giả kit.



Hình 18.8. Sơ đồ khối của hệ điều khiển (số tương tự) có phản hồi

– Một động cơ một chiều có mạch đo tốc độ bằng đĩa quang.

Chương trình phục vụ kit PID được viết bằng ngôn ngữ C, bao gồm:

_ Chương trình điều khiển với thực đơn chọn chế độ như sau:

+ MODE 1: hiển thị tốc độ đặt của động cơ (bằng chiết áp).

+ MODE 2: nhập thông số cho bộ PID.

+ MODE 3: nhập thông số cho đối tượng điều khiển.

+ MODE 4: nhập tần số lấy mẫu của hệ thống.

+ MODE 5: nhập dạng tín hiệu phản hồi (1. tương tự, 2. số).

Chương trình thực hiện luật điều khiển PID.

– Chương trình điều khiển bàn phím, hiển thị LCD, hiển thị LED7 đoạn và trao đổi tin nối tiếp với PC.

Kit đã dùng tính các hệ số cho PID và đo đặc tính quá độ hệ điều khiển tốc độ động cơ một chiều thấy phù hợp với kết quả của phần mềm Mathlab.

18.5.4. Hệ điều khiển động cơ ba pha lồng sóc

Động cơ xoay chiều ba pha lồng sóc được điều khiển với biến tần theo sơ đồ ở hình 18.8. Biến tần có chứa VDK.

Trong hệ cân bằng tải ở nhà máy xi măng có dùng động cơ trên với cảm biến đo tốc độ gắn với ròng rọc, cảm biến đo khối lượng gắn trên giá mà tải đè lên. Hệ cân bằng tải trên được lập trình với PLC Step7 của SIEMEN. Đang có đề nghị thay PLC đắt tiền bằng một PLC được xây dựng trên cơ sở VDK 8751(theo tài liệu của Hàn Quốc) với các cổng vào, các cổng ra bằng ghép nối quang, vi mạch đệm công suất ULN2803 và rơ le.

18.5.5. Hệ điều khiển trong người máy (ROBOT)

Robot là thành tựu mới nhất của tự động hóa, ứng dụng các thành quả của cơ khí, của đo lường hiện đại với cảm biến khác nhau (nhìn, xúc giác, nghe v.v...), của điều khiển với cơ cấu chấp hành tiên tiến, của tin (phần cứng và phần mềm) với các VDK tiên tiến, hệ BUS công nghiệp (Profibus, FielBUS và CAN), của cơ khí chính xác và kích thước cực nhỏ (tới nanô mét). Có thể nói, trong ROBOT tập trung tất cả các thành tựu mới nhất của khoa học, trong đó dùng nhiều VDK theo một mạng trao đổi tin. Robot đã được dùng trong nhiều lĩnh vực khoa học và đời sống (máy lau nhà, chó canh cửa, đồ chơi trẻ con v.v...).

18.6. HỆ TÍCH LŨY, XỬ LÝ VÀ TRUYỀN TIN VỚI VDK

Vì điều khiển còn được ứng dụng trong các hệ tích lũy, xử lý và truyền tin.

1. Hệ tích lũy tin

Trong hệ tích lũy tin (Data logger), VDK có thể:

– Điều khiển bộ dồn kênh tương tự (có biên độ của tín hiệu ra bằng biên độ tín hiệu vào) để một trong các lối vào của các cảm biến đưa tín hiệu vào một ADC duy nhất của hệ đo nhiều cảm biến (tiết kiệm ADC).

– Điều khiển một trong các lối vào ADC nhiều lối vào nối với các cảm biến để đưa tín hiệu vào ADC.

– Điều khiển một trong nhiều VDK tổ hoạt động ở chế độ $M = 3$ của trao đổi tin nối tiếp (chương 17).

Về chương trình phục vụ cho hệ tích lũy tín tương tự chương trình giám sát, có chương trình chính và các chương trình con.

Chương trình chính quản lý hệ chương trình theo thời gian thực định sẵn bởi người dùng. Là một vòng lặp liên tục kiểm tra các điều kiện, thực hiện điều kiện được thỏa mãn và gọi các chương trình con tương ứng.

Các chương trình con có:

- Chương trình định thời gian thực, tạo tín hiệu đồng bộ mọi hoạt động của hệ thống, thống nhất thời gian giữa người và máy, người dùng có thể nhập thời gian định sẵn từ bàn phím.

- Chương trình chọn kênh dữ liệu cần trích mẫu và phát tín hiệu Start cho ADC.

- Chương trình tổ chức bộ nhớ, ghi và đọc tín của mỗi kênh vào vùng nhớ dành riêng cho chúng.

- Chương trình phục vụ ngắt theo thời gian, chương trình tiến hành một số thao tác quan trọng, làm tín hiệu cho bộ thời gian.

- Chương trình phục vụ ngắt do ADC kết thúc quá trình biến đổi khi có tín hiệu EOC của ADC. Dữ liệu của ADC sẽ được ghi vào vùng nhớ tương ứng với kênh được chọn.

Trong Proteus, có sơ đồ và chương trình tích lũy tín cho ổ đĩa cứng HDD khi vào thư mục Labcenter/Protuse/ Sample/Data logger.

Hệ ghi và phát lại tiếng nói (ghi và phát âm) là một hệ tích lũy tín đơn kênh nhưng với dung lượng tín lớn, tần số xung trích mẫu (Sampling) lớn cho tiếng nói trung thực và với thời gian thực. Người ta đã chế tạo các vi mạch dành riêng cho ghi và phát âm, trong đó có VDK với tần số xung nhịp lớn, thời gian ghi và phát âm lớn.

Trong vi mạch ghi/phát âm có VDK tốc độ nhanh, ADC, DAC với 12 bit trở lên và tuổi thọ cao (ghi/phát nhiều lần) và giữ tín được lâu.

Vi mạch đơn chip ISD2560/75/90/120 có thể ghi và phát tiếng nói trong 60, 75, 90 và 120 giây. Vi mạch này có thể giữ được tín trong 100 năm và ghi và xóa tới 1000.000 lần.

Vi mạch đơn chip ISD4000 có thể ghi và phát âm từ 2 tới 16 phút. Có nhớ RAM NVRAM (không mất nội dung khi tắt nguồn nuôi), dùng VDK có ghép nối đường dây SPI và Microware.

Vi mạch này cũng lưu giữ tín được 100 năm và có 100K chu trình ghi/đọc

Để trộn tiếng nói, hãng TEXAS Instrument đã chế tạo VDK MSP430 với ADC 10 bit, cảm biến nhiệt độ, hai DAC 12bit, 27 lệnh chính và 7 chế độ địa chỉ.

2. Hệ xử lý tín

Hệ xử lý tín bằng phương pháp số (DSP- Digital Signal Procesing) phải thực hiện các phép tính nhanh, chính xác với các con số lớn nhiều bit.

Hãng Motorola đã chế tạo các VDK chuyên dùng cho xử lý tín hiệu như các họ DSP-56000, DSP-56800, DSP-56100, DSP563000, DSP96000 và MSC 8101.

Một vi mạch DSP có các thành phần là bộ lọc tần thấp, bộ biến đổi ADC, bộ xử lý tín hiệu số DSP, bộ biến đổi DAC và bộ lọc là phẳng.

VDK DSP-56000/56001 có một số đặc điểm chính khác biệt sau:

- Có thêm đơn vị máy phát địa chỉ AGU.

- Bộ mô phỏng trên chip máy vi tính PC nối với chip, có thể xâm nhập từng thanh ghi của chip.

- Mạch tạo xung đồng bộ và vòng khóa pha PLL (Phase Locked Loop).

- Đơn vị số học ALU có 4 thanh ghi đầu vào (24bit), 2 thanh ghi tích lũy A, B (56 bit với 48 bit cơ bản, 8 bit mở rộng), 2 mạch dịch/hạn chế dữ liệu, 1 bộ nhân tích lũy song song.

- Đơn vị số học và nhân tích lũy MAC có: bộ nhân 24x24bit, bộ lấy tổng 56 bit, ghi dịch 56 bit, thiết bị logic, thiết bị làm tròn và lấy tỷ lệ và lệnh có thể có 3 toán hạng.
- Cổng chỉ có 3 với cổng A (24bit), B- giao diện chủ và C- trao đổi tin nối tiếp đồng bộ và dị bộ.

Hiện có phần mềm Measurement & Automation Explorer, Matlab & Simulink và các phần mềm khác được dùng để xử lý tin.

3. Hệ trao đổi tin

Tin sau khi tích lũy, xử lý sẽ gửi đi tới các trạm thu tin như VDK chủ, máy vi tính trung gian hay máy vi tính chủ quản lý trung tâm đo, máy vi tính quản lý mạng qua các đường dây nối giữa chúng (GPIB, CAMAC, RS232, RS485, I2C, oneway, Profibus và CAN).

Hệ trao đổi tin có cấu hình từ đơn giản (một số VDK nối với nhau dưới sự điều khiển của VDK chủ) tới mạng CAN, mạng LAN, mạng WAN, Mạng Internet. Mỗi mạng có giao thức riêng và có phần mềm quản lý và phục vụ để trao đổi tin nhanh, tin cậy và tránh xung đột đường dây. Tin có thể gửi đi xa không cần dây nối (không dây) nhờ sóng hồng ngoại, sóng vô tuyến v.v...

CÂU HỎI, BÀI TẬP VÀ THỰC HÀNH

I. CÂU HỎI

1. Kể các loại kit thực hành VDK, so sánh giữa chúng.
2. Kể các loại hệ VDK trong đo lường và điều khiển.
3. Vai trò của thực đơn và chương trình giám sát (monitor).

II. BÀI TẬP

1. Dùng Proteus, thiết kế các hệ điều khiển bàn phím và hiển thị trên LED đơn, led 7 đoạn, LCD.
2. Dùng Proteus, thiết kế các hệ điều khiển động cơ bước, động cơ servo có bàn phím và hiển thị.

III. THỰC HÀNH

1. Cho chạy mô phỏng trên Proteus một số hệ đã giới thiệu (calculator, động cơ bước, động cơ servo, Data logger).
2. Dịch và chạy mô phỏng (trên Pinnacle và Proteus) các ví dụ của chương.

TÀI LIỆU THAM KHẢO

1. Ngô Diên Tập, 1999. *Vi xử lý trong đo lường và điều khiển*. NXB Khoa học và Kỹ thuật. Hà Nội.
2. Ngô Diên Tập, 2003. *Kỹ thuật vi điều khiển với AVR*. NXB Khoa học và Kỹ thuật. Hà Nội.
3. Tống Văn On, 2001. *Họ vi điều khiển 8051*. NXB Lao động và Xã hội. Hà Nội.
4. Tống Văn On và Hoàng Đức Hải, 2006. *Thiết kế hệ thống với họ 8051*. NXB Phương Đông. Tp. Hồ Chí Minh.
5. Nguyễn Tăng Cường và Phan Quốc Thắng, 2004. *Cấu trúc và lập trình họ vi điều khiển 8051*. NXB Khoa học và Kỹ thuật. Hà Nội.
6. Lê Văn Doanh và Phạm Khắc Chương, 1998. *Kỹ thuật vi điều khiển*. NXB Khoa học và Kỹ thuật. Hà Nội.
7. Nguyễn Đình Phú. *Vi điều khiển 8051*. Tài liệu Đại học tại chức, trường Đại học Sư phạm Kỹ thuật Tp. Hồ Chí Minh.
8. Nguyễn Mạnh Giang, 2007. *Cấu trúc, lập trình, ghép nối và ứng dụng vi điều khiển*. Tập một. *Cấu trúc và lập trình VĐK 8051/8052*. NXB Giáo dục, Hà Nội.
9. Bùi Quý Lục, 2005. *Hệ thống điều khiển số trong công nghiệp*. NXB Khoa học và Kỹ thuật. Hà Nội.
10. Đào Văn Hiệp, 2003. *Kỹ thuật Robot*. NXB Khoa học và Kỹ thuật. Hà Nội.
11. Nguyễn Phùng Quang, 2006. *MATLAB & SIMULINK dành cho kỹ sư điều khiển tự động*. NXB Khoa học và Kỹ thuật. Hà Nội.
12. Nguyễn Phùng Quang, 1996. *Điều khiển truyền động điện xoay chiều ba pha*. NXB Giáo dục. Hà Nội.
13. Nguyễn Ngọc Cẩn, 1995. *Kỹ thuật điều khiển tự động*. Trường ĐHBK Tp. Hồ Chí Minh.

MỤC LỤC

Lời nói đầu

Chương 11: GHÉP NỐI VÀ ỨNG DỤNG ĐÈN LED	5
11.1. Đèn hiển thị LED rời	6
11.2. Hiển thị trên một đèn LED 7 đoạn	13
11.3. Thiết kế, lập trình hiển thị hệ nhiều đèn LED	18
11.4. Hiển thị một chuỗi số hay ký tự với mạch giải mã	28
11.5. Cách ghép nối LED 7 đoạn tiết kiệm cổng của vi điều khiển	34
11.6. Hiển thị chữ và số trong các biển quảng cáo	38
Câu hỏi, bài tập và thực hành	40
Chương 12. GHÉP NỐI VÀ ỨNG DỤNG ĐÈN LED MA TRẬN	41
12.1. Cấu tạo, hoạt động, cách mắc và các loại mạch của ma trận LED	41
12.2. Thiết kế, lập trình và chạy mô phỏng hiển thị trên một ma trận LED	44
12.3. Điều khiển hiển thị theo màu sắc và theo xung tín hiệu	51
12.4. Ma trận LED tích hợp và thông minh	55
12.5. Hệ hiển thị trên nhiều ma trận LED	59
12.6. Hiển thị động trên LED ma trận	64
Câu hỏi, bài tập và thực hành	70
Chương 13. GHÉP NỐI VỚI MÀN HÌNH TINH THỂ LỎNG (LCD)	71
13.1. Tổng quát về LCD	71
13.2. Cấu trúc của LCD	73
13.3. Hệ lệnh của LCD	77
13.4. Ghép nối LCD với VĐK họ MSC-51	82
13.5. LCD trao đổi tin nối tiếp	101
13.6. LCD đồ họa	106
Câu hỏi, bài tập và thực hành	114
Chương 14. GHÉP NỐI VỚI BÀN PHÍM	116
14.1. Tổng quát về bàn phím	116
14.2. Bàn phím mắc song song	119
14.3. Bàn phím mắc kiểu ma trận	123
14.4. Bàn phím cho hệ vi xử lý và máy vi tính	136
14.5. Bàn phím trong thiết bị điều khiển từ xa	144
14.6. Một số ứng dụng của bàn phím	146
Câu hỏi, bài tập và thực hành	150

<i>Chương 15. GHÉP NỐI VỚI BỘ NHỚ NGOÀI CỦA VI ĐIỀU KHIỂN 8051/8052</i>	151
15.1. Tổng quát về bộ nhớ ngoài của vi điều khiển 8051/8052	151
15.2. Ghép nối vi mạch nhớ với vi điều khiển.....	155
15.3. Lập trình cho vi mạch nhớ PROM ngoài.....	160
15.4. Lập trình cho vi mạch nhớ RAM ngoài	162
15.5. Nhớ EEPROM nối tiếp	165
15.6. Nhớ NV-RAM.....	169
15.7. Bộ nhớ của VDK AT89C51RD2/ED2.....	174
Câu hỏi, bài tập và thực hành.....	174
 <i>Chương 16. GHÉP NỐI MỞ RỘNG CỔNG SONG SONG CHO VI ĐIỀU KHIỂN</i>	176
16.1. Yêu cầu và cách mở rộng cổng cho vi điều khiển.....	176
16.2. Mở rộng cổng song song dùng thanh ghi đệm.....	179
16.3. Mở rộng cổng song song dùng vi mạch 8255A	186
16.4. Trao đổi tin của 82C55 ở chế độ ngắt chương trình và hội thoại ($M = 1$, $M = 2$)	210
16.5. Trao đổi tin song song của VDK qua 82C55	215
16.6. Trao đổi tin song song của VDK qua 82C55 với thiết bị ngoài công suất.....	219
Câu hỏi, bài tập và thực hành.....	220
 <i>Chương 17. GHÉP NỐI MỞ RỘNG CỔNG NỐI TIẾP CHO VI ĐIỀU KHIỂN</i>	222
17.1. Tổng quát về đặc điểm của trao đổi tin nối tiếp của máy vi tính PC	222
17.2. Trao đổi tin nối tiếp của VDK MCS-51	227
17.3. Trao đổi tin nối tiếp của UART của VDK AT89C51RD2/ED2.....	237
17.4. Trao đổi tin nối tiếp SPI của AT89C51RD2/ED2.....	241
17.5. Trao đổi tin theo USB và I2C.....	244
17.7. Trao đổi tin nối tiếp theo mạng của VDK	250
Câu hỏi, bài tập và thực hành.....	251
 <i>Chương 18. GHÉP NỐI VỚI HỆ VI ĐIỀU KHIỂN</i>	252
18.1. Hệ VDK trong thực hành VDK	252
18.2. Hệ phát triển phần mềm và chương trình giám sát	257
18.3. Hệ nghiên cứu và ứng dụng với ADC và DAC	266
18.4. Hệ đếm, đo và phân tích.....	268
18.5. Hệ điều khiển	271
18.6. Hệ tích lũy, xử lý và truyền tin với VDK	274
Câu hỏi, bài tập và thực hành.....	276
Tài liệu tham khảo	277

Chịu trách nhiệm xuất bản:

Chủ tịch HĐQT kiêm Tổng Giám đốc NGÔ TRẦN ÁI
Phó Tổng Giám đốc kiêm Tổng biên tập NGUYỄN QUÝ THAO

Tổ chức bản thảo và chịu trách nhiệm nội dung:

Chủ tịch HĐQT kiêm Giám đốc CTCP Sách ĐH-DN
TRẦN NHẬT TÂN

Biên tập và sửa bản in:

DƯƠNG VĂN BẰNG

Trình bày bìa:

HOÀNG MẠNH DỨA

Chế bản:

ĐINH XUÂN DŨNG

CẤU TRÚC – LẬP TRÌNH – GHÉP NỐI VÀ ỨNG DỤNG VI ĐIỀU KHIỂN

TẬP HAI: GHÉP NỐI VÀ ỨNG DỤNG VDK 8051/8052

Mã số: 7B661M7 - DAI

In 1.000 cuốn (QĐ 83), khổ 19 x 27 cm, tại Xí nghiệp in Hà Tây.

Địa chỉ: 15 Quang Trung, TP. Hà Đông

Số ĐKKH xuất bản: 17 - 2007/CXB/72 - 2217/GD.

In xong và nộp lưu chiểu tháng 10 năm 2007.



CÔNG TY CỔ PHẦN SÁCH ĐẠI HỌC - DẠY NGHỀ

HEVOBCO

25 HÀN THUYỀN - HÀ NỘI

Website : www.hevobco.com.vn



VƯƠNG MIỆN KIM CƯƠNG
CHẤT LƯỢNG QUỐC TẾ

TÌM ĐỌC SÁCH THAM KHẢO KỸ THUẬT CỦA NHÀ XUẤT BẢN GIÁO DỤC

- | | |
|---|--|
| 1. Cơ sở kỹ thuật Laser | Trần Đức Hân – Nguyễn Minh Hiền |
| 2. Kỹ thuật điện tử | Đỗ Xuân Thụ |
| 3. Bài tập kỹ thuật điện tử | Đỗ Xuân Thụ – Nguyễn Viết Nguyên |
| 4. Kỹ thuật điện tử số | Đặng Văn Chuyết |
| 5. Kỹ thuật audio – video | Nguyễn Thanh Trà – Thái Vĩnh Hiền |
| 6. Kỹ thuật mạch điện tử phi tuyến | Phạm Minh Việt – Trần Công Nhượng |
| 7. Hệ thống viễn thông (Hai tập) | Phạm Minh Việt – Thái Hồng Nhị |
| 8. Cơ sở kỹ thuật điện tử số | Vũ Đức Thọ |
| 9. Kỹ thuật truyền tin số và truyền dữ liệu | Phạm Minh Việt – Thái Hồng Nhị |
| 10. Thiết bị đầu cuối thông tin | Vũ Đức Thọ |
| 11. Kỹ thuật điện tử ứng dụng | Nguyễn Vũ Sơn |
| 12. Vô tuyến điện tử | Ngọc Văn An – Đặng Hùng
Nguyễn Đăng Lâm – Lê Xuân Thê
Đỗ Xuân Kiên |
| 13. Tính toán mạng thông tin di động số | Vũ Đức Thọ |
| 14. Dụng cụ bán dẫn và vi mạch | Lê Xuân Thê |
| 15. Trường điện từ | Lâm Hồng Thạch – Nguyễn Khuyến |
| 16. Thông tin số | Nguyễn Viết Kính – Trịnh Anh Vũ |
| 17. Hệ thống thông tin công nghiệp | Phạm Thượng Hàn (CB) |
| 18. Cấu trúc, lập trình, ghép nối và ứng dụng vi điều khiển (Hai tập) | Nguyễn Mạnh Giang |

Bạn đọc có thể mua tại các Công ty Sách - Thiết bị trường học ở các địa phương hoặc các Cửa hàng của Nhà xuất bản Giáo dục :

Tại Hà Nội : 25 Hàn Thuyên ; 187B Giảng Võ ; 232 Tây Sơn ; 23 Tràng Tiền ;

Tại Đà Nẵng : Số 15 Nguyễn Chí Thanh ; Số 62 Nguyễn Chí Thanh ;

Tại Thành phố Hồ Chí Minh : Cửa hàng 451B - 453, Hai Bà Trưng, Quận 3 ;

240 Trần Bình Trọng – Quận 5.

Tại Thành phố Cần Thơ : Số 5/5, đường 30/4 ;

Website : www.nxbgd.com.vn



8 934980 763193



Giá: 39.500 đ