TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

TRƯỜNG TRUNG CẤP KINH TẾ - KỸ THUẬT SỐ 2

GIÁO TRÌNH

MÔ ĐUN: ĐIỀU KHIỂN HỆ THỐNG CƠ ĐIỆN TỬ SỬ DỤNG VI ĐIỀU KHIỂN

NGÀNH, NGHỀ: CƠ – ĐIỆN TỬ

TRÌNH ĐỘ: TRUNG CẤP

(Ban hành kèm theo Quyết định số: /QĐ-TC2 ngày …. tháng ..... năm……

của Hiệu trưởng Trường Trung cấp Kinh tế - Kỹ thuật số 2)

Đồng Nai, năm 2022

(Lưu hành nội bộ)

TUYÊN BỐ BẢN QUYỀN

LỜI GIỚI THIỆU

Công nghệ thông tin đang được ứng dụng rộng rãi trong nhiều lĩnh vực khoa học công nghệ và cuộc sống thường nhật. Bên cạnh khối lượng phần mềm hệ thống và ứng dụng đồ sộ, công nghệ phần cứng cũng phát triển vô cùng nhanh chóng. Có thể nói các hệ thống máy tính được cải thiện trong những khoảng thời gian rất ngắn, càng ngày càng nhanh hơn, mạnh hơn và hiện đại hơn.

Những kiến thức cơ bản về phần cứng của các hệ thống máy tính luôn là đòi hỏi cấp thiết trong quá trình điều khiển, giám sát các hệ thống công nghệ và kỹ thuật hiện đại.

Giáo trình ***Điều khiển hệ thống cơ điện tử sử dụng Vi điều khiển*** này được viết trên cơ sở những bài giảng theo sát đề cương môn học đã được thực hiện tại Khoa Cơ – Điện tử trực thuộc Trường Trung cấp Kinh tế - Kỹ thuật số 2 từ khi thành lập đến nay, và luôn luôn được sửa chữa, bổ sung để đáp ứng nhu cầu kiến thức của học sinh, sinh viên học tập tại Khoa.

Giáo trình được lưu hành nội bộ gồm bảy bài như sau:

Bài mở đầu: Sơ lược về lịch sử và hướng phát triển của vi điều khiển

Bài 1: Cấu trúc họ vi điều khiển 89C51

Bài 2: Tập lệnh 89C51

Bài 3: Bộ định thời

Bài 4: Cổng nối tiếp

Bài 5: Tổ chức ngắt

Bài 6: Phần mềm mô phỏng – Lập trình tổng hợp

Giáo trình với các bài tập ứng dụng theo các nội dung lý thuyết ở các bài cụ thể giúp người học vận dụng ngay vào thực tế điều khiển.

Trong bài 6 tác giả đã giới thiệu hệ thống các bài tập thực hành tổng hợp với những ứng dụng trong thực tiễn đời sống và sản xuất với mô tả đầy đủ về yêu cầu công nghệ cũng như các địa chỉ vào – ra giúp người học có thể dễ dàng thực hành và vận hành điều khiển và sáng tạo trong lập trình..

Tuy đã dành nhiều thời gian, nhưng chắc chắn còn nhiều thiếu sót, rất mong các bạn đọc giả góp ý để tài liệu này hoàn chỉnh hơn.

Tôi chân thành cảm ơn Ban Giám hiệu nhà trường, Phòng Đào tạo và Khoa Công nghệ thông tin đã tạo điều kiện để tôi được tham gia biên soạn giáo trình

môn học này. Tôi cũng ghi nhận và cảm ơn Hội đồng nghiệm thu giáo trình của nhà trường, cảm ơn các cán bộ kỹ thuật tại doanh nghiệp, bạn bè, đồng nghiệp đã đóng góp các ý kiến thiết thực để tôi làm cơ sở hoàn chỉnh hơn giáo trình đưa vào giảng dạy thực tế tại trường.

Đồng Nai, ngày 22 tháng 7 năm 2022

Tham gia biên soạn:

Chủ biên: ThS. Trần Thị Bích Hạnh

MỤC LỤC

GIÁO TRÌNH MÔ ĐUN

1. Tên mô đun: Điều khiển hệ thống cơ điện tử sử dụng vi điều khiển

2. Mã số mô đun: MĐ30

3. Vị trí, tính chất của mô đun:

3.1. Vị trí: Trước khi học mô đun này học sinh phải hoàn thành các mô đun: An toàn lao động, Điện kỹ thuật, Đo lường điện điện tử, Điện tử công nghiệp, Kỹ thuật số, Kỹ thuật cảm biến.

3.2. Tính chất: Là mô đun bắt buộc trong chương trình đào tạo nghề Cơ điện tử.

4. Mục tiêu mô đun:

4.1. Kiến thức:

- Trình bày được cấu trúc họ vi điều khiển 89c51.

- Đọc sơ đồ và phân tích nguyên lý hoạt động của mạch dùng vi điều khiển.

4.2. Kỹ năng:

- Vận hành được các thiết bị và dây chuyền sản xuất dùng vi điều khiển

- Đo thử, kiểm tra mạch điều khiển, phán đoán nguyên nhân gây hư hỏng, thay thế mới và tương đương linh kiện hư hỏng.

- Viết được chương trình ứng dụng dùng vi điều khiển và mô phỏng mạch chạy đúng theo yêu cầu trên phần mêm chuyên dùng.

- Lập trình và kết nối giao tiếp được với thiết bị ngoại vi: led đơn, Led 7 đoạn, Led ma trận, động cơ…

4.3. Năng lực tự chủ và trách nhiêm:

- Tích cực, chủ động và sáng tạo trong học tập.

- Vệ sinh xưởng thực tập ngăn nắp, sạch sẽ, có tinh thần làm việc nhóm, có tác phong công nghiệp

5. Nội dung chương trình:

5.1. Chương trình khung:

STT	MÔN HỌC	Số tín chỉ	Thời gian học tập (giờ)			
			Tổng giờ	Lý thuyết	Thực hành/ thực tập/ thí nghiệm/ bài tập/ thảo luận	Thi/ Kiểm tra/ BC
I.	**Các môn chung**	**12**	**255**	**94**	**148**	**13**
MH01	Anh văn căn bản	4	90	30	56	4
MH02	Chính trị	2	30	15	13	2

MH03	Pháp luật	1	15	9	5	1
MH04	Giáo dục thể chất	1	30	4	24	2
MH05	Giáo dục quốc phòng - An ninh	2	45	21	21	3
MH06	Tin học	2	45	15	29	1
II.	**Các môn học, mô đun cơ sở**	**17**	**270**	**135**	**118**	**17**
MH07	Vật liệu công nghiệp	2	30	15	13	2
MH08	An toàn lao động	2	30	15	13	2
MH09	Điện kỹ thuật	3	45	30	13	2
MĐ10	Đo lường điện	2	30	15	13	2
MĐ11	Khí cụ điện	2	45	15	27	3
MH12	Vẽ kỹ thuật cơ khí	2	30	15	13	2
MH13	Vẽ kỹ thuật điện	2	30	15	13	2
MH14	Dung sai và đo lường kỹ thuật	2	30	15	13	2
III	**Các môn học, mô đun chuyên môn nghề**	**57**	**1440**	**347**	**1012**	**81**
MĐ15	Máy điện	3	60	30	27	3
MĐ16	Điện tử công nghiệp	6	150	30	114	6
MĐ17	Kỹ thuật số	3	75	30	42	3
MH18	Tiếng Anh chuyên nghề Cơ điện tử	3	60	28	28	4
MĐ19	Trang bị điện	6	150	30	114	6
MĐ20	Kỹ thuật cảm biến	2	30	15	13	**2**
MH21	AUTOCAD	2	45	15	28	2
MĐ22	Tháo lắp các cụm máy công cụ	2	45	15	28	2
MĐ23	Điều khiển khí nén	3	75	15	57	3
MĐ24	Điều khiển thuỷ lực	3	75	15	57	3
MĐ25	Vận hành máy công cụ	2	45	15	28	2
MĐ26	Vận hành máy CNC	2	45	15	28	2
MĐ27	PLC căn bản	3	75	18	51	6
MĐ28	Điều khiển hệ thống cơ điện tử sử dụng PLC	6	150	30	112	8
MĐ29	Thực tập tốt nghiệp	6	240	16	200	24
MĐ30	Điều khiển hệ thống cơ điện tử sử dụng vi điều khiển	5	120	30	85	5
	Tổng cộng	**86**	**1965**	**576**	**1278**	**111**

5.2. Chương trình chi tiết của mô đun:

Số TT	**Tên các bài trong mô đun**	**Thời gian**			
		Tổng số	**Lý thuyết**	Thực hành, thí nghiệm, thảo luận, bài tập	**Kiểm tra***

1	Bài mở đầu: Sơ lược về lịch sử và hướng phát triển của vi điều khiển 1. Lịch sử phát triển 2. Khái niệm vi điều khiển 3. Lĩnh vực và ứng dụng và hướng phát triển	04	4	0	
2	Bài 1: Cấu trúc họ vi điều khiển 89C51 1. Tổng quan 2. Sơ đồ chân 3. Cấu trúc Port I/O 4. Tổ chức bộ nhớ 5. Các thanh ghi chức năng đặc biệt. 6. Bộ nhớ ngoài	08	4	4	
3	Bài 2: Tập lệnh 89c51 1. Các khái niệm cơ bản 2. Các cách định địa chỉ 3. Các nhóm lệnh 4. Luyện tập	20	5	14	1
4	Bài 3: Bộ định thời 1. Giới thiệu về bộ định thời của VĐK 2. Thanh ghi SFR của timer 3. Các chế độ làm việc 4. Khởi tạo và truy xuất thanh ghi Timer 7. Luyện tập	16	4	11	1
5	Bài 4: Cổng nối tiếp 1. Khái quát chung 2. Khởi tạo và truy suất thanh ghi PORT nối tiếp 3. Luyện tập	16	3	12	1
6	Bài 5: Tổ chức ngắt 1. Tổ chức ngắt 2. Xử lý ngắt 3. Thiết kế chương trình dùng ngắt 4. Luyện tập	28	4	23	1
7	Bài 6: Phần mềm mô phỏng – lập trình tổng hợp 1. Phần mềm mô phỏng Protues 2. Thiết kế mạch điều khiển	28	6	20	2

	3. Các bài tập mở rộng				
	Cộng	**120**	**30**	**84**	**6**

* Ghi chú: *Thời gian kiểm tra được tích hợp giữa lý thuyết với thực hành được tính vào giờ thực hành*

6. Điều kiện thực hiện mô đun:

6.1. phòng học chuyên môn hóa, nhà xưởng*: học tại xưởng thực hành vi điều khiển – có trang bị các máy tính cài sẵn phần mềm mô phỏng.*

6.2. Trang thiết bị máy móc:

- Máy chiếu, máy tính cá nhân
- Các phần tử modul chính cho thực hành vi điều khiển:

+ Bộ nguồn DC 5v, 12v, 24 V
+ Bộ nhập các tín hiệu vào
- Kit thực hành vi điều khiển và mô hình kèm theo

6.3.Học liệu, dụng cụ, nguyên vật liệu:

- Giáo trình vi điều khiển.
- Tài liệu hướng dẫn sử dụng họ 89c51.
- Sơ đồ các bài tập ứng dụng trên kit thực hành.
- Phần mềm mô phỏng.
- Linh kiện:

Vi mạch 89c51, thạch anh, điện trở, tụ, rờ le, đông cơ, Led các loại theo bài
Mạch in, dây nối chì hàn…..

- Bộ dây điện, Testboard
- Mỏ hàn, kềm cắt, kềm nhọn…
- Đồng hồ VOM
- Dụng cụ tháo, ráp vi mạch

7. Phương pháp và nội dung đánh giá:

7.1. Nội dung:

7.1.1. Về kiến thức:

Được đánh giá bằng hình thức kiểm tra viết, trắc nghiệm theo các nội dung: trình bày cấu tạo, đặc điểm,ứng dụng của các loại Vi điều khiển được học

7.1.2. Về kỹ năng:

Đánh gía kỹ năng thực hành theo những nội dung sau:

Mỗi học viên, hoặc mỗi nhóm học viên thực hiện công việc sau đây theo yêu cầu của giáo viên:

- Lắp ráp được các mạch ứng dụng từng phần do giáo viên đề ra.
- Thực hiện viết các chương trình theo yêu cầu cho trúoc

Tiêu chí đánh giá theo các nội dung:

- Độ chính xác của công việc
- Tính thẩm mỹ của mạch điện
- Độ an toàn trên mạch điện
- Thời gian thực hiện công việc
- Độ chính xác theo yêu cầu kỹ thuật

7.1.3. Năng lực tự chủ và trách nhiệm:

Đánh giá phong cách học tập thể hiện ở: Tỉ mỉ, cẩn thận, chính xác.

- Có ý thức tự giác, tính kỷ luật cao, tinh thần trách nhiệm trong công việc.
- Có tinh thần hợp tác giúp đỡ lẫn nhau. Được đánh giá qua quá trình học tập.

7.2. Phương pháp:

Áp dụng hình thức kiểm tra tích hợp giữa lý thuyết với thực hành

7.2.1. Cách đánh giá

- Áp dụng quy chế đào tạo Trung cấp hệ chính quy ban hành kèm theo Thông tư số 09/2017/TT-LĐTBXH, ngày 13/3/2017 của Bộ trưởng Bộ Lao động – Thương binh và Xã hội.

- Hướng dẫn thực hiện quy chế đào tạo áp dụng tại Trường Trung cấp Kinh tế - Kỹ thuật số 2 như sau:

Điểm đánh giá	**Trọng số**
+ Điểm kiểm tra thường xuyên (Hệ số 1)	40%
+ Điểm kiểm tra định kỳ (Hệ số 2)	
+ Điểm thi kết thúc môn học	60%

7.2.2. Phương pháp đánh giá

Phương pháp đánh giá	**Phương pháp tổ chức**	**Hình thức kiểm tra**	**Chuẩn đầu ra đánh giá**	**Số cột**	**Thời điểm kiểm tra**
Thường xuyên	Viết/ Hỏi đáp tại lớp	Tự luận/ Trắc nghiệm/ Vấn đáp	A1, A2, A3, B1, B2, B3, C1, C2	2	Sau 08 giờ.
Định kỳ	Viết/ Thực hành theo nhóm	Tự luận/ Thực hành	A4, B4, C3	5	Sau 20 giờ
Kết thúc môn học	Thực hành cá nhân	Tự luận và thực hành	A1, A2, A3, A4, A5, B1, B2, B3, B4, B5, C1, C2, C3,	1	Sau 140 giờ

7.2.3. Cách tính điểm

- Điểm đánh giá thành phần và điểm thi kết thúc môn học được chấm theo thang điểm 10 (từ 0 đến 10), làm tròn đến một chữ số thập phân.

- Điểm môn học là tổng điểm của tất cả điểm đánh giá thành phần của môn học nhân với trọng số tương ứng. Điểm môn học theo thang điểm 10 làm tròn đến một chữ số thập phân, sau đó được quy đổi sang điểm chữ và điểm số theo thang điểm 4 theo quy định của Bộ Lao động Thương binh và Xã hội về đào tạo theo tín chỉ.

8. Hướng dẫn thực hiện mô đun:

8.1. Phạm vi áp dụng chương trình:

- Chương trình mô đun được sử dụng để giảng dạy cho trình độ trunng cấp nghề.
- Chương trình có thể dùng để dạy học sinh ngắn hạn (sơ cấp nghề) có trình độ văn hóa trên lớp 12 và đã qua đào tạo điện tử trung cấp có nhu cầu chuyển đổi nghề.

8.2. Hướng dẫn một số điểm chính về phương pháp giảng dạy, học tập mô đun:

Nội dung được biên soạn theo phương pháp tích hợp do đó cần lưu ý một số điểm chính sau

- Đối với giáo viên:

+ Trước khi giảng dạy cần phải căn cứ vào nội dung của từng bài học chuẩn bị đầy đủ các điều kiện thực hiện bài học để đảm bảo chất lượng giảng dạy.

+ Vật liệu, dụng cụ và trang thiết bị phải được chuẩn bị đầy đủ trước khi thực hiện bài giảng

+ Các bài thí nghiệm cần thực hiện theo thứ tự của hệ thống, có thể thêm vào các bài tập ứng dụng.

+Trong phần thực hành, giáo viên cần phải ôn lại các kiến thức có liên quan và trình bày kỹ lưỡng các bước tiến hành. Sau mỗi bài tập phải thu lại các báo cáo để đánh giá trình độ hiểu biết của học sinh. Giảng dạy các bài thực hành cần có mô hình, vật thật, các sơ đồ nguyên lý, sơ đồ nối dây.

+ Trong các bài thực tập, Giáo viên giảng dạy chọn ra các bài thực hành tương ứng với phần lý thuyết đã dạy.

+ Tăng cường sử dụng thiết bị, đồ dùng dạy học, trình diễn mẫu để tăng hiệu quả dạy học.

+Thực hiện giảng dạy ở nơi thực tập hoặc xưởng thực hành.

+ Hệ thống nguồn điện cung cấp cần được phân biệt và kiểm tra chính xác trước khi cho học sinh thực tập.

- Đối với người học:

+ Học sinh cần được chia thành các nhóm nhỏ từ 1 đến 4 học sinh, để thực hiện nội dung thực hành.

+Trước khi học mô đun này học sinh phải hoàn thành: An toàn lao động Điện kỹ thuật, Đo lường điện điện tử, Điện tử công nghiệp.

+ Thực hiện theo hướng đẫn của giáo viên, tích cực, chủ động và sáng tạo trong học tập.

+Chú ý an toàn điện cho người và thiết bị trong quá trình thực hành mạch.

8.3. Những trọng tâm chương trình cần chú ý:

- Về phân bổ thời gian: Căn cứ vào thực tế của nơi đào tạo giáo viên hướng dẫn có thể thay đổi thời lượng, của từng nội dung, nhưng vẫn phải đảm bảo số giờ qui định trong chương trình.
- Về nội dung chi tiết trong chương trình: Căn cứ vào thực tế trang bị của nhà trường hoặc nhu cầu đào tạo tại địa phương, nhà trường có thể thay thế các họ PLD tương thích với nhu cầu đào tạo và thiết bị hiện có, nhưng vẫn phải đảm bảo mục tiêu của mô đun.
- Cần giới thiệu các sản phẩm, mô hình thực tế để học sinh có thể tham gia bài giảng và ghi nhớ sâu hơn.
- Cần chú ý các biện pháp an toàn về điện. Chống va đập, rơi rớt các thiết bị, thường xuyên theo dõi học sinh trong học tập, thực hành.

8.4. Tài liệu tham khảo:

- Microprocessor and IC families - Walter H. Buchbaum. Sc.D
- Mikrocompute Lehrbuch - HPI Fachbuchreihen Pflaum Verlag Munchen
- 8951 Development Boad, Rev 5 - Paul Stoffregen
- The 8951 microcontroller - I. Scott Makenzie
- Họ vi điều khiển - Tống văn On - Đại học Bách khoa TP.HCM

Bài mở đầu

Sơ lược về lịch sử và hướng phát triển của vi điều khiển.

❖ GIỚI THIỆU BÀI MỞ ĐẦU

Bài mở đầu là bài giới thiệu sơ lược về lịch sử và hướng phát triển của vi điều khiển với những đặc điểm và phạm vi sử dụng của chúng để người học có được kiến thức nền tảng và dễ dàng tiếp cận nội dung mô đun ở những bài tiếp theo.

❖ MỤC TIÊU CỦA BÀI MỞ ĐẦU

- Trình bày được cấu trúc chung của vi điều khiển
- Phát biểu được các ứng dụng của vi điều khiển và hướng phát triển của vi điều khiển
- Tích cực, chủ động và sáng tạo trong học tập.

❖ PHƯƠNG PHÁP GIẢNG DẠY VÀ HỌC TẬP BÀI MỞ ĐẦU

- *Đối với người dạy: sử dụng phương pháp giảng giảng dạy tích cực (diễn giảng, vấn đáp, dạy học theo vấn đề); yêu cầu người học thực hiện câu hỏi thảo luận của bài (cá nhân hoặc nhóm).*
- *Đối với người học: chủ động đọc giáo trình trước buổi học; hoàn thành đầy đủ câu hỏi thảo luận theo cá nhân hoặc nhóm và nộp lại cho người dạy đúng thời gian quy định.*

❖ ĐIỀU KIỆN THỰC HIỆN BÀI MỞ ĐẦU

- ***Phòng học chuyên môn hóa/nhà xưởng:*** học tại xưởng thực hành vi điều khiển – có trang bị các máy tính cài sẵn phần mềm mô phỏng.
- ***Trang thiết bị máy móc:*** Máy chiếu và các mô hình, thiết bị dạy học khác
- ***Học liệu, dụng cụ, nguyên vật liệu:*** Chương trình mô đun, giáo trình, tài liệu tham khảo, giáo án, phim ảnh, và các KIT thực hành vi điều khiển.
- ***Các điều kiện khác:*** Không có

❖ KIỂM TRA VÀ ĐÁNH GIÁ BÀI MỞ ĐẦU

- **Nội dung:**

✓ *Kiểm tra và đánh giá tất cả nội dung về kiến thức, kỹ năng đã nêu trong mục tiêu của bài*

✓ *Năng lực tự chủ và trách nhiệm: Trong quá trình học tập, người học cần:*

+ Nghiên cứu bài trước khi đến lớp

+ Chuẩn bị đầy đủ tài liệu học tập.

+ *Tham gia đầy đủ thời lượng môn học.*

+ *Nghiêm túc trong quá trình học tập.*

- **Phương pháp:**

✓ ***Điểm kiểm tra thường xuyên:*** *1 điểm kiểm tra (hình thức - vấn đáp)*

✓ ***Kiểm tra định kỳ lý thuyết:*** *không có*

NỘU DUNG BÀI MỞ ĐẦU

1. Tóm tắt lịch sử ra đời và phát triển của các dòng vi xử lý và vi điều khiển

1.1 Vi xử lý

Vi xử lý được chế tạo từ các tranzito tích hợp trên một vi mạch tích hợp đơn. Xuất hiện lần đầu tiên vào những năm đầu của thập kỷ 70 của thế kỷ 20. Sử dụng mã BCD trên nền 4 bit. Các vi xử lý 4bit và 8 bit được sử dụng trong các thiết bị đầu cuối, máy in, các hệ thống tự động...Đến giữa những năm 1970 thì lần đầu tiên các vi xử lý 8 bit với 16 bit địa chỉ được sử dụng như máy tính đa mục đích. Các hãng sản xuất vi xử lý đầu tiên ở thời điểm này là Intel, Texas Instruments và Garrett AiResearch với ba dòng chip tương ứng: Intel 4004, TMS 1000 và Central Air Data Computer. Đây là những vi xử lý 4 bit. Sau sự ra đời của các vi xử lý 4 bit thì các hãng cho ra đời các dòng 8 bit, 12 bit, 16 bit, 32 bit, 64 bit. Intel 8008 là vi xử lý 8 bit đầu tiên trên thế giới được sản xuất năm 1972. Tiếp sau thành công của 8008 là các phiên bản như 8080 (1974), Zilog Z80 (1976). Các vi xử lý của Motorola 6800 được phát hành tháng 8 năm 1974 và MOS technology ra đời năm 1975. Intersil 6100 là vi xử lý 12 bit, từ khi được sản xuất bởi công ty Harris nó được biết đến với tên HM-6100 được sử dụng trong quân đội suốt thập niên 1980. Vi xử lý 16 bit đầu tiên được giới thiệu bởi hãng National Semiconductor IMP-16 vào năm 1973 đây là vi xử lý đa chip. Đến năm 1975 hãng này giới thiệp vi xử lý đơn chip đầu tiên. Hãng Texas Instruments ra đời vi xử lý 16 bit đơn chip TI-990 sử dụng như một máy tính mini. Intel cũng cho ra đời dòng vi xử lý 16 bit lấy tên 8086. Vi xử lý 16 bit chỉ xuất hiện trên thị trường một thời gian ngắn thì dòng 32 bit đã bắt đầu xuất hiện. MC6800 là vi xử lý 32 bit đầu tiên của hãng Motorola, họ 68k có 32 bit thanh ghi nhưng sử dụng đường dẫn dữ liệu 16 bit bên trong và 16 bit dữ liệu bên ngoài để giảm số lượng pin, hỗ trợ 24 bit địa chỉ. Motorola thường được biết đến như vi xử lý 16 bit mặc dù nó có cấu trúc 32 bit. Vi xử lý 32 bit đầy đủ đầu tiên là AT&T Bell Labs BELLMAC-32A với mẫu đầu tiên vào năm 1980 và sản xuất năm 1982. Vi xử lý 32 bit đầu tiên của Intel là dòng iAPX 432 được giới thiệu năm 1981 nhưng không thu được thành công. Vi xử lý ARM đầu tiên ra đời năm 1985 với thiết kế RISC viết tắt của reduced instruction set computer máy tính có tập lệnh rút gọn, các vi xử lý ARM được sử dụng chủ yếu trong các điện thoại di động. Vi xử lý 64 bit được thiết kế cho các máy tính cá nhân. Nó được thiết kế vào đầu những năm 1990 đến đầu những năm 2000 chứng kiến vi xử lý 64 bit nhằm vào thị trường máy tính. Vi xử lý AMD 64 bit tương thích ngược với x86, x86-64 còn gọi là AMD64 trong tháng 9 năm 2003, tiếp sau thành công của Intel64. Kỷ nguyên của máy tính 64 bit đã bắt đầu.

1.2 Vi điều khiển

Vi điều khiển là một máy tính được tích hợp trên một chíp, nó thường được sử dụng để điều khiển các thiết bị điện tử. Vi điều khiển thực chất gồm một vi xử lý có hiệu suất đủ cao và giá thành thấp (so với các vi xử lý đa năng dùng trong máy tính) kết hợp với các thiết bị ngoại vi như các bộ nhớ, các mô đun vào/ra, các mô đun biến đổi từ số sang tương tự và từ tương tự sang số, mô đun điều chế độ rộng xung

(PWM)... Vi điều khiển thường được dùng để xây dựng hệ thống nhúng. Nó xuất hiện nhiều trong các dụng cụ điện tử, thiết bị điện, máy giặt, lò vi sóng, điện thoại, dây truyền tự động... Hầu hết các loại vi điều khiển hiện nay có cấu trúc Harvard là loại cấu trúc mà bộ nhớ chương trình và bộ nhớ dữ liệu được phân biệt riêng. Cấu trúc của một vi điều khiển gồm CPU, bộ nhớ chương trình (thường là bộ nhớ ROM hoặc bộ nhớ Flash), bộ nhớ dữ liệu (RAM), các bộ định thời, các cổng vào/ra để giao tiếp với các thiết bị bên ngoài, tất cả các khối này được tích hợp trên một vi mạch. Các loại vi điều khiển trên thị trường hiện nay:

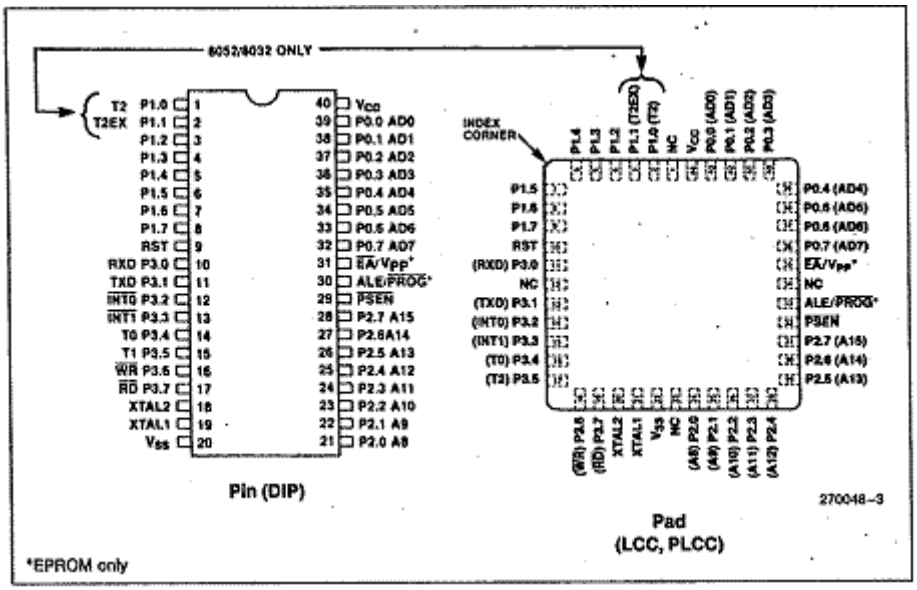
- Freescale 68HC11 (8-bit)
- Intel 8051
- STMicroelectronics STM8S (8-bit), ST10 (16-bit) và STM32 (32-bit)
- Atmel AVR (8-bit), AVR32 (32-bit), và AT91SAM (32-bit)
- Freescale ColdFire (32-bit) và S08 (8-bit)
- Hitachi H8 (8-bit), Hitachi SuperH (32-bit)
- MIPS (32-bit PIC32)
- PIC (8-bit PIC16, PIC18, 16-bit dsPIC33 / PIC24)
- PowerPC ISE
- PSoC (Programmable System-on-Chip)
- Texas Instruments Microcontrollers MSP430 (16-bit), C2000 (32-bit), và Stellaris (32-bit)
- Toshiba TLCS-870 (8-bit/16-bit)
- Zilog eZ8 (16-bit), eZ80 (8-bit)
- Philips Semiconductors LPC2000, LPC900, LPC700

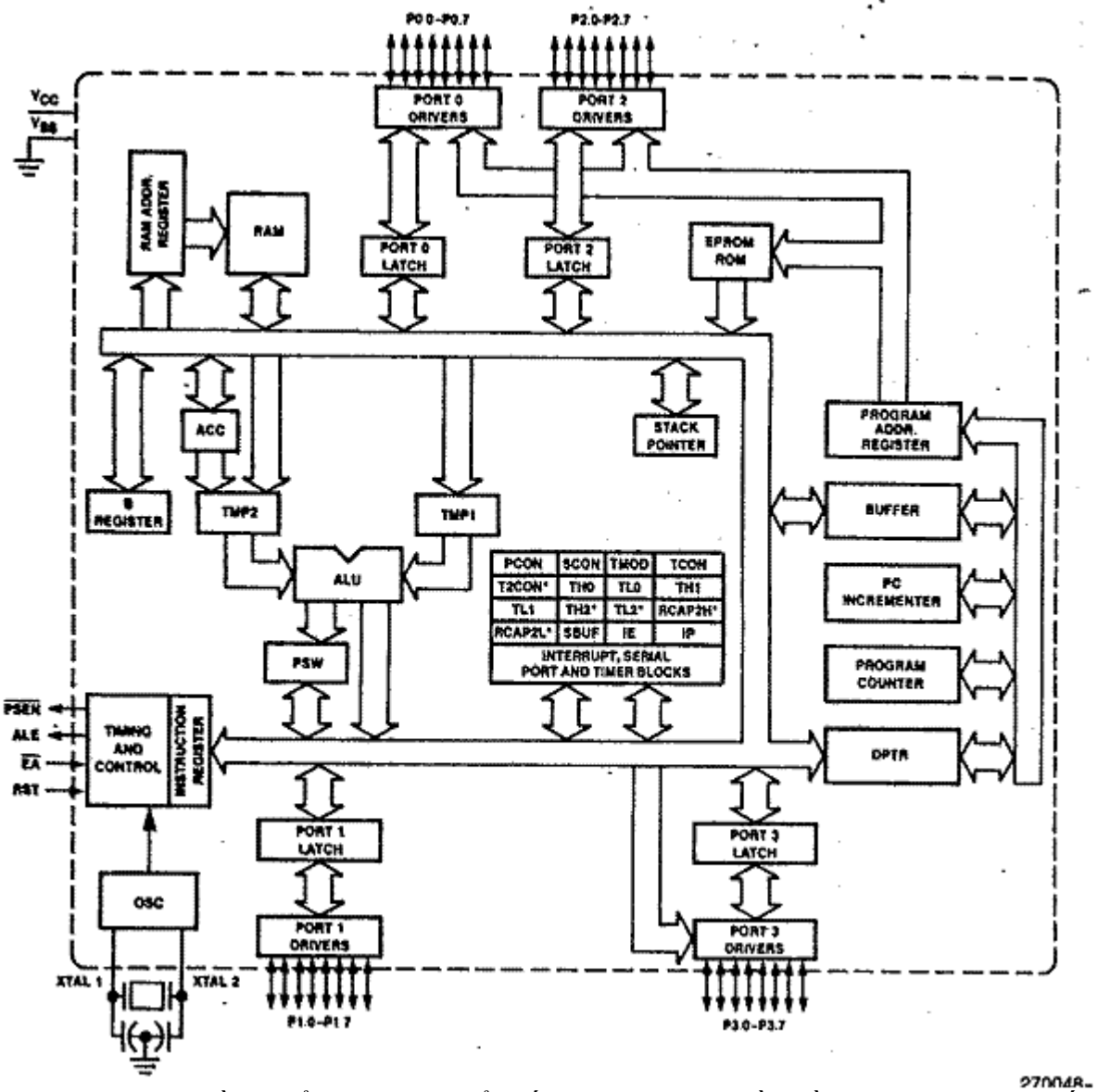
2. Khái niệm về vi điều kiển

Có thể nói việc sử dụng các loại vi điều khiển và vi xử lý trong các thiết bị điện tử tự động ở Việt Nam rất đa dạng, phong phú tùy vào yêu cầu kỹ thuật và giá thành sản phẩm.

Đối với các thiết bị như các máy ATM, máy giặt thường sử dụng vi điều khiển 8051, các bộ điều khiển trong robot công nghiệp, trong hệ thống ô tô thường sử dụng PIC, AVR, PsoC, còn trong điện thoại sử dụng các chip ARM…

2.1 Vi điều khiển 8051





Intel 8051 - là vi điều khiển đơn tinh thể kiến trúc Harvard, lần đầu tiên được sản xuất bởi Intel năm 1980, để dùng trong các hệ thống nhúng. Trong những năm 1980 và đầu

những năm 1990 đã rất nổi tiếng. Tuy nhiên hiện tại đã cũ và được thay thế bằng các thiết bị hiện đại hơn, với các lõi phối hợp 8051, được sản xuất bởi hơn 20 nhà sản xuất độc lập như Atmel, Maxim IC (công ty con của Dallas Semiconductor), NXP Semiconductors (Philips Semiconductor trước đây), Winbond, Silicon Laboratories, Texas Instruments và Cypress Semiconductor. Tên gọi chính thức của họ vi điều khiển Intel 8051 - MCS 51. Những vi điều khiển Intel 8051 được sản xuất với việc dùng công nghệ MOSFET, những những bản sau, chứa kí hiệu "C" trong tên, như 80C51, dùng công nghệ CMOS và yêu cầu công suất thấp, hơn những cái MOSFET trước (điều này cho phép trang bị cho các thiết bị với nguồn là pin). Các thông số kỹ thuật: 8 bit ALU, 8 bit thanh ghi. 8 bit dữ liệu bus 16 bit địa chỉ bus vì vậy không gian bộ nhớ tối đa cho ROM và RAM lên tới 64 kb Bộ nhớ dữ liệu SRAM 128 bytes Bộ nhớ chương trình ROM 4 kb. 32 chân vào/ra đa hướng. Giao tiếp nối tiếp UART. Hai bộ timer/counter 16 bit. Hai ngắt ngoài. Sơ đồ chân của 8051: Sơ đồ khối điều khiển: Lập trình cho 8051: Các nhà sản xuất 8051 đều hỗ trợ ngôn ngữ lập trình Assembler tuy nhiên ngôn ngữ này thường ít được dùng cho những ứng dụng lớn do tính phù hợp của nó, vì vậy trong các ứng dụng thực tế hay sử dụng ngôn ngữ C. Ngoài ra còn một số ngôn ngữ khác được phát triển cho 8051 như Pascal, Basic, Forth.

2.2 Vi điều khiển AVR

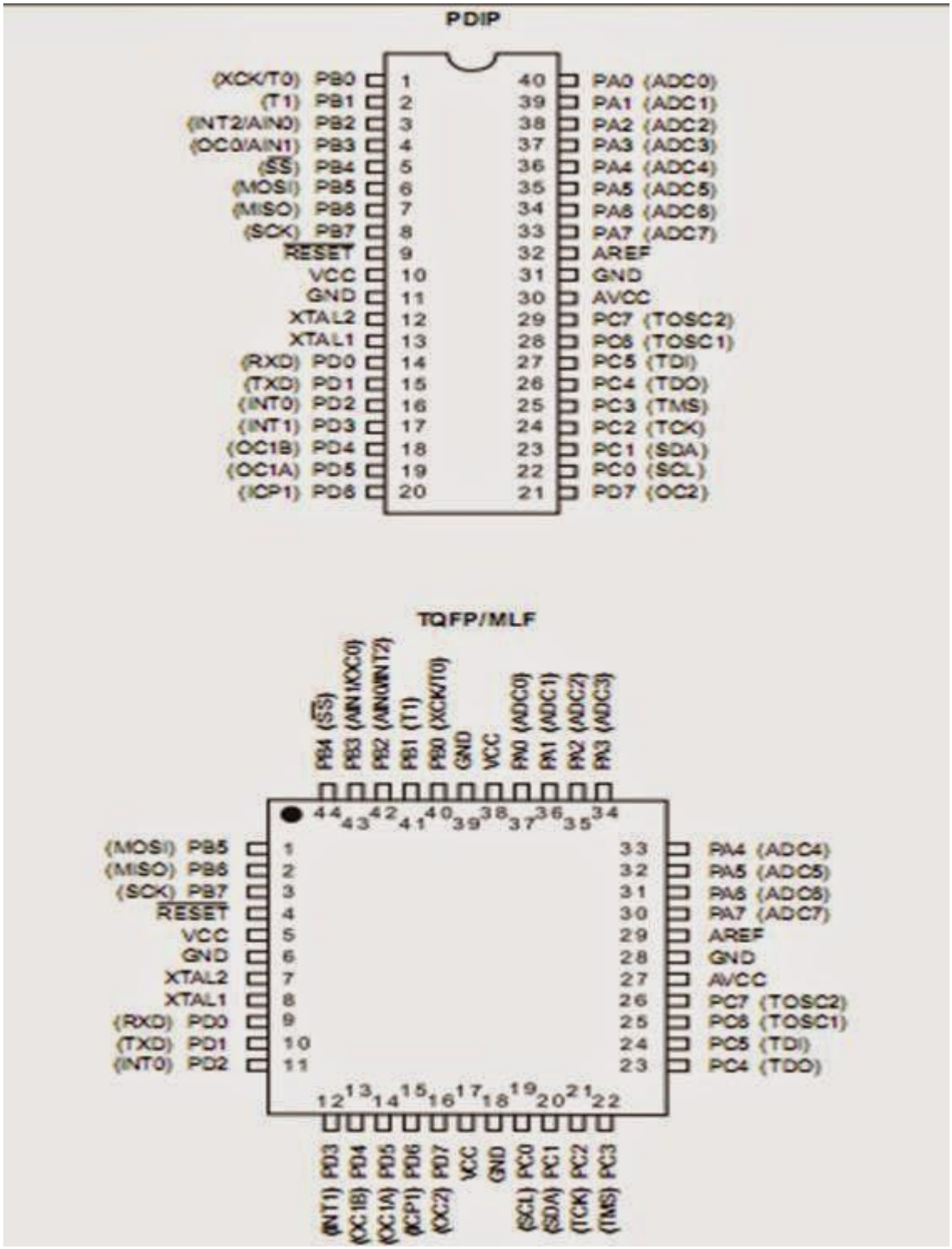
Là dòng vi điều khiển do hãng Atmel sản xuất có nhiều loại AVR như:

- 32-bit AVR UC3.
- 8/16-bit AVR XMEGA.
- 8-bit mega AVR.
- 8-bit tiny AVR.

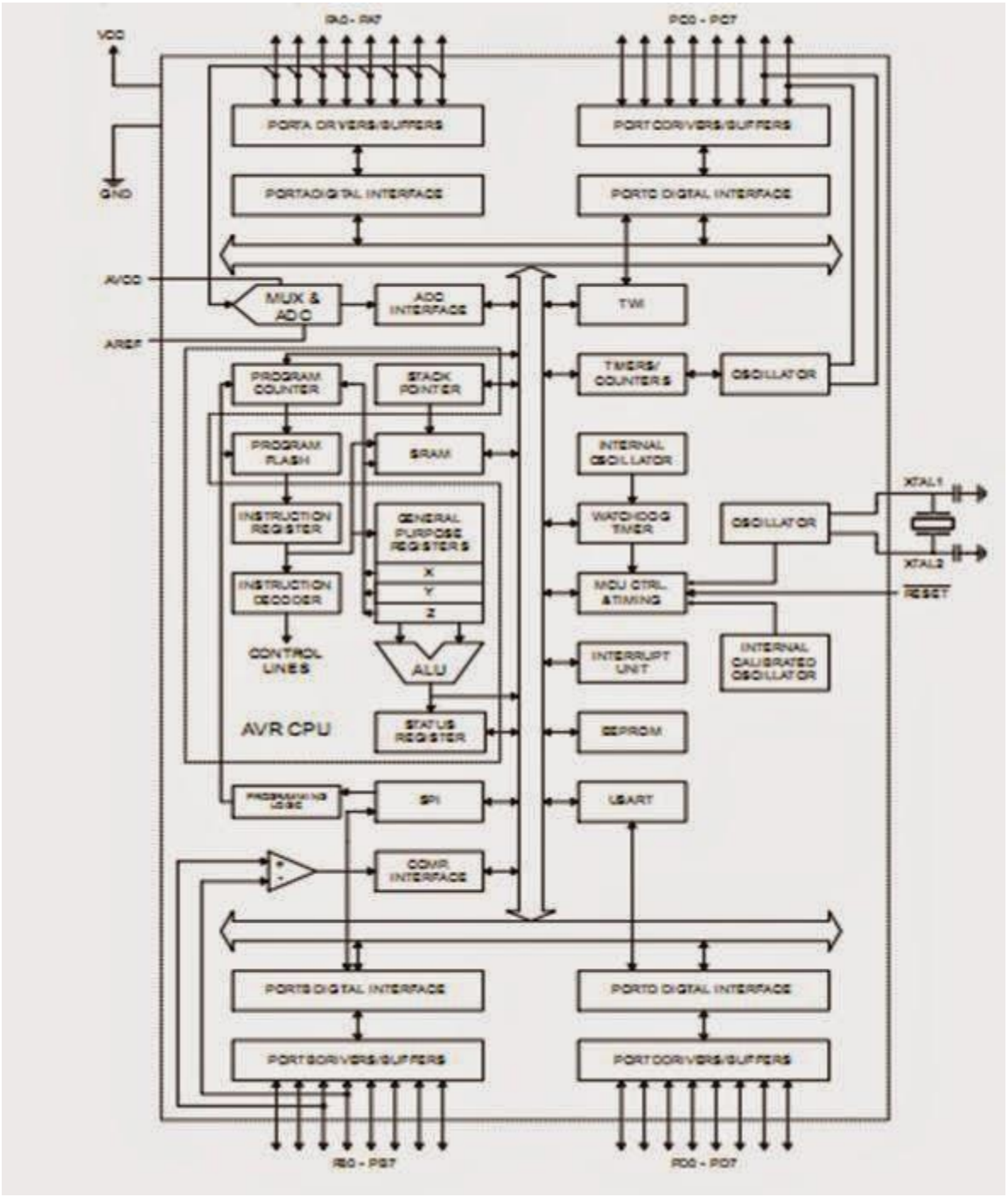
Vi điều khiển Atmega 16: Là vi điều khiển 8 bit với tiêu thụ điện năng thấp dựa trên kiến trúc RISC (Reduced Instruction Set Computer). Vào ra Analog – digital và ngược lại. Với công nghệ này cho phép các lệnh thực thi chỉ trong một chu kì xung nhịp, vì thế tốc độ xử lý dữ liệu có thể đạt đến 1 triệu lệnh trên giây ở tần số 1Mhz. Vi điều khiển này cho phép người thiết kế có thể tối ưu hoá chế độ tiêu thụ năng lượng mà vẫn đảm bảo tốc độ xử lý. Lõi AVR có tập lệnh phong phú với số lượng với 32 thanh ghi làm việc chung với nhau. Tất cả 32 thanh ghi đều được nối trực tiếp với ALU (Arithmetic Logic Unit), cho phép 2 thanh ghi truy cập độc lập trong một chỉ lệnh đơn trong một chu kỳ xung nhịp. Kiến trúc đạt được có tốc độ xử lý nhanh gấp 10 lần vi điều khiển dạng CISC (Complex Instruction Set Computer) thông thường. Atmega 16 được hỗ trợ đầy đủ phần mềm và công cụ phát triển hệ thống bao gồm: Trình dịch Assembly như AVR studio của Atmel, Trình dịch C như win AVR, CodeVisionAVR C, ICCAVR. C - CMPPILER của GNU… Trình dịch C đã được nhiều người dùng và đánh giá tương đối mạnh, dễ tiếp cận đối với những người bắt đầu tìm hiểu AVR, đó là trình dịch CodeVisionAVR C. Phần mềm này hỗ trợ nhiều ứng dụng và có nhiều hàm có sẵn nên việc lập trình tốt hơn. - Bộ nhớ: Flash 16KB

EEPROM 512 Byte SRAM 1KB. - Ngoại vi: Hai timer 8 bit Một timer 16 bit Bộ counter với tần số riêng Bốn bộ điều chế độ rộng xung PWM. Tám kênh ADC 10 bit. USART. Giao tiếp SPI, Giao diện I2C. Watchdog timer. Bộ so sánh tương tự trên chip. - Tính năng: Tập lệnh gồm 131 lệnh, hầu hết thực hiện trong một chu kỳ máy. Xử lý 16 triệu lệnh ở tần số 16 MHZ. 32 chân vào/ra có thể lập trình được. Sáu chế độ sleep . 40 pin kiểu PDIP, 44 pin kiểu TQFP và kiểu QFL/MLF. 32 thanh ghi 8 bit đa dụng. Ngắt trong và ngắt ngoài. Điện áp hoạt động từ 2,7-5,5V cho Atmega 16A.

-Sơ đồ chân



Sơ đồ khối điều khiển:



2.3 Vi điều khiển PIC

PIC là một họ vi điều khiển RISC được sản xuất bởi công ty Microchip Technology. Dòng PIC đầu tiên là PIC1650 được phát triển bởi Microelectronics Division thuộc General Instrument . PIC bắt nguồn là chữ viết tắt của "Programmable Intelligent Computer" (Máy tính khả trình thông minh). Là vi điều khiển với kiến trúc RISC thực thi một lệnh với một chu kỳ máy (bằng bốn chu kỳ của bộ dao động). Ngày nay có nhiều dòng PIC được sản xuất với hàng loạt các mô đun ngoại vi tích hợp sẵn như ADC, PWM, USART, SPI…với bộ nhớ chương trình từ 512 word đến 32 Kword.

Các họ vi điều khiển PIC: - Họ 8 bit: PIC 10/ PIC 12/ PIC 16/ PIC 18 - Họ 16 bit: PIC 24F/ PIC 24H/ dsPIC 30/ dsPIC 33 - Họ 32 bit: PIC 32. Một vài đặc tính:

- Chân vào/ra I/O có thể lập trình được.
- Flash và ROM có thể tuỳ chọn từ 256 byte đến 512 Kbyte
- Bộ dao động bên trong.
- 8/16/32 bit Timers.
- Bộ nhớ EEPROM nội
- Chuẩn giao tiếp nối tiếp đồng bộ và không đồng bộ USART
- MSSP Peripheral cho giao tiếp I2C và SPI
- Các chế độ so sánh, bắt giữ và điều chế độ rộng xung PWM.
- Bộ so sánh điện áp.
- Bộ chuyển đổi ADC (tần số có thể lên tới 1 MHz).
- Hộ trợ các giao thức USB, CAN, Ethernet.
- Mô đun điều khiển động cơ, mô đun đọc encoder.
- Hộ trợ bộ nhớ ngoài.
- DSP những tính năng xử lý tín hiệu số (dsPIC)

Lập trình cho PIC: Hãng Microchip cung cấp môi trường lập trình MPLAB nó bao gồm phần mềm mô phỏng, trình dịch ASM, liên kết và gỡ rối. Ngoài ra hãng này cũng bán trình biên dịch C cho các dòng PIC18 và dsPIC tích hợp trong MPLAB. Ngoài ra còn một số công ty khác cung cấp trình biên dịch C, PASCAL, BASIC cho PIC đó có thể là phần mềm thương mại hoặc phần mềm mã nguồn mở.

2.4 Vi điều khiển ARM

Cấu trúc ARM (viết tắt từ tên gốc là Acorn RISC Machine) là một loại cấu trúc vi xử lý 32-bit kiểu RISC được sử dụng rộng rãi trong các thiết kế nhúng. Được phát triển lần đầu trong một dự án của công ty máy tính Acorn. Do có đặc điểm tiết kiệm năng lượng, các bộ CPU ARM chiếm ưu thế trong các sản phẩm điện tử di động, mà với các sản phẩm này việc tiêu tán công suất thấp là một mục tiêu thiết kế quan trọng hàng đầu. Ngày nay, hơn 75% CPU nhúng 32-bit là thuộc họ ARM, điều này khiến ARM trở thành cấu trúc 32-bit được sản xuất nhiều nhất trên thế giới. CPU ARM được tìm thấy khắp nơi trong các sản phẩm thương mại điện tử, từ thiết bị cầm tay (PDA, điện thoại di động, máy đa phương tiện, máy trò chơi cầm tay, và máy tính cầm tay) cho đến các thiết bị ngoại vi máy tính (ổ đĩa cứng, bộ định tuyến để bàn.). Một nhánh nổi tiếng của họ ARM là các vi xử lý Xscale của Intel. Giới thiệu về vi điều khiển LPC2148: Là dòng vi điều khiển ARM được sản xuất bởi hãng Philips. Tính năng:

- Vi điều khiển 16/32-bit ARM7TDMI-S
- 40k RAM tĩnh (8k +32k), 512k flash
- Tích hợp USB 2.0
- Hộ trợ hai bộ ADC 10 bit
- Một bộ DAC 10 bit
- 2 bộ timer 32 bit, 6 ngõ điều chế độ rộng xung
- Đồng hồ thời gian thực hỗ trợ tần số 32kHz
- Khả năng thiết lập chế độ ưu tiên và định địa chỉ cho ngắt
- 45 chân GPIO vào ra đa dụng
- 9 chân ngắt ngoài (tích cực cạnh hoặc tích cực mức)
- CPU clock đạt tối đa 60MHz thông qua bộ PLL lập trình được
- Xung PLCK hoạt động độc lập.

On-chip Flash Memory: LPC 2148 có 512K bộ nhớ Flash có thể được dùng để lưu trữ code và dữ liệu. Trong khi thực thi ứng dụng, vẫn có thể xóa hoặc lập trình Flash thông qua IAP (In Application Programming). Khi đó trình loader trên chip được sử dụng, bộ nhớ trống còn lại là 500K. Bộ nhớ Flash có thể ghi xóa được ít nhất 100000 lần, lưu trữ dữ liệu đến 20 năm. On-chip Static RAM: LPC 2148 có 32K RAM tĩnh, có thể được truy xuất theo đơn vị byte, half word & word. Bộ điều khiển SRAM sử dụng phương thức write-back buffer để ngăn chặn tình trạng treo CPU khi có thao tác ghi. Bộ đệm luôn giữ dữ liệu cuối cùng từ chương trình gửi tới bộ nhớ. Dữ liệu chỉ được ghi vào SRAM khi có 1 thao tác ghi khác từ chương trình. Lập trình cho ARM: Ngôn ngữ lập trình chính cho ARM hiện nay là ngôn ngữ C. Các trình biên dịch cho ARM thường được dùng:

- Keil ARM.
- IAR.
- HTPICC for ARM.
- ImageCraft ICCV7 for ARM

3. Lĩnh vực và ứng dụng và hướng phát triển

Các dòng vi điều khiển từ đơn giản đến phức tạp rất đa dạng tùy vào yêu cầu kỹ thuật và giá thành của sản phẩm mà chọn loại vi điều khiển thích hợp cho việc thiết kế và lập trình sản phẩm.

3.1. Ưu điểm của vi điều khiển

Những ưu điểm chính của vi điều khiển là:

a) Vi điều khiển hoạt động như một máy vi tính không có bất kỳ bộ phận kỹ thuật số nào.

b) Tích hợp cao hơn bên trong vi điều khiển làm giảm chi phí và kích thước của hệ thống.

c) Việc sử dụng vi điều khiển rất đơn giản, dễ khắc phục sự cố và bảo trì hệ thống.

d) Hầu hết các chân được lập trình bởi người dùng để thực hiện các chức năng khác nhau.

e) Dễ dàng kết nối thêm các cổng RAM, ROM, I / O.

f) Cần ít thời gian để thực hiện các hoạt động.

3.2. Nhược điểm của vi điều khiển

a) Vi điều khiển có kiến trúc phức tạp hơn so với vi xử lý.

b) Chỉ thực hiện đồng thời một số lệnh thực thi giới hạn.

c) Chủ yếu được sử dụng trong các thiết bị vi mô.

d) Không thể trực tiếp giao tiếp các thiết bị công suất cao.

3.3. Ứng dụng và hướng phát triển của vo điều khiển

Chúng ta có thể tìm thấy vi điều khiển trong tất cả các loại thiết bị điện tử hiện nay. Bất kỳ thiết bị nào liên quan đến đo lường, lưu trữ, điều khiển, tính toán hoặc hiển thị thông tin đều phải có chip vi điều khiển bên trong. Ứng dụng lớn nhất của vi điều khiển là trong ngành công nghiệp ô tô (vi điều khiển được sử dụng rộng rãi để kiểm soát động cơ và điều khiển công suất trong ô tô). Bạn cũng có thể tìm thấy vi điều khiển bên trong bàn phím, chuột, modem, máy in và các thiết bị ngoại vi khác. Trong thiết bị thử nghiệm, vi điều khiển giúp bạn dễ dàng thêm các tính năng như khả năng lưu trữ số đo, tạo và lưu trữ các thói quen của người dùng và hiển thị thông báo cũng như dạng sóng. Sản phẩm tiêu dùng sử dụng bộ vi điều khiển bao gồm máy quay kỹ thuật số, đầu phát quang, màn hình LCD / LED…

Bài 1: Cấu trúc họ vi điều khiển 89C51

❖ GIỚI THIỆU BÀI 1

Bài 1 là bài giới thiệu tổng quan về cấu trúc phần cứng của VĐK 89C51 và tổ chức bộ nhớ của vi điều khiển. Trên cơ sở đó người học có được kiến thức về cấu trúc và các địa chỉ vùng nhớ của VĐK làm tiền đề cho việc lập trình điều khiển ở các nội dung tiếp theo.

❖ MỤC TIÊU CỦA BÀI 1

- Mô tả được cấu trúc họ vi điều khiển.
- Thực hiện truy xuất bộ nhớ dữ liệu, bộ nhớ chương trình đúng qui trình kỹ thuật
- Thực hiện đúng kỹ thuật phương pháp mở rộng bộ nhớ ngoài.
- Trình bày được nguyên lý hoạt động của mạch reset
- Tích cực, chủ động và sáng tạo trong học tập.

❖ PHƯƠNG PHÁP GIẢNG DẠY VÀ HỌC TẬP BÀI 1

- *Đối với người dạy: sử dụng phương pháp giảng giảng dạy tích cực (diễn giảng, vấn đáp, dạy học theo vấn đề); yêu cầu người học thực hiện câu hỏi thảo luận của bài (cá nhân hoặc nhóm).*
- *Đối với người học: chủ động đọc giáo trình trước buổi học; hoàn thành đầy đủ câu hỏi thảo luận theo cá nhân hoặc nhóm và nộp lại cho người dạy đúng thời gian quy định.*

❖ ĐIỀU KIỆN THỰC HIỆN BÀI 1

- ***Phòng học chuyên môn hóa/nhà xưởng:*** học tại xưởng thực hành vi điều khiển – có trang bị các máy tính cài sẵn phần mềm mô phỏng.
- ***Trang thiết bị máy móc:*** Máy chiếu và các mô hình, thiết bị dạy học khác
- ***Học liệu, dụng cụ, nguyên vật liệu:*** Chương trình mô đun, giáo trình, tài liệu tham khảo, giáo án, phim ảnh, và các KIT thực hành vi điều khiển.
- ***Các điều kiện khác:*** Không có

❖ KIỂM TRA VÀ ĐÁNH GIÁ BÀI 1

- **Nội dung:**

✓ *Kiểm tra và đánh giá tất cả nội dung về kiến thức, kỹ năng đã nêu trong mục tiêu của bài*

✓ *Năng lực tự chủ và trách nhiệm: Trong quá trình học tập, người học cần:*

+ Nghiên cứu bài trước khi đến lớp

+ *Chuẩn bị đầy đủ tài liệu học tập.*

+ *Tham gia đầy đủ thời lượng môn học.*

+ *Nghiêm túc trong quá trình học tập.*

- **Phương pháp:**

✓ ***Điểm kiểm tra thường xuyên:*** *1 điểm kiểm tra (hình thức - vấn đáp)*

✓ ***Kiểm tra định kỳ lý thuyết:*** *không có*

NỘU DUNG BÀI 1

1.1. Tổng quan:

Hiện nay, các bộ vi điều khiển 8 bit đứng đầu là họ 8051 có số lương lớn nhất các nhà cung cấp đa dạng (nhiều nguồn). Nhà cung cấp có nghĩa là nhà sản xuất bên cạnh nhà sáng chế của bộ vi điều khiển. Trong trường hợp 8051 thì nhà sáng chế của nó là Intel, nhưng hiện nay có rất nhiều hãng sản xuất nó (cũng như trước kia đã sản xuất).

Các hãng này bao gồm: Intel, Atmel, Philips/signe-tics, AMD, Siemens, Matra và Dallas, Semicndictior.

Bảng địa chỉ của một số hãng sản xuất các thành viên của họ 8051.

Hãng	Địa chỉ Website
Intel	www.intel.com/design/mcs51
Antel	www.atmel.com
Plips/ Signetis	www.semiconductors.philips.com
Siemens	www.sci.siemens.com
Dallas Semiconductor	www.dalsemi.com

8051 là một bộ xử lý 8 bit có nghĩa là CPU chỉ có thể làm việc với 8 bit dữ liệu tại một thời điểm. Dữ liệu lớn hơn 8 bit được chia ra thành các dữ liệu 8 bit để cho xử lý. 8051 có tất cả 4 cổng vào - ra I/O mỗi cổng rộng 8 bit. Các nhà sản xuất đã cho xuất xưởng chỉ với 4K byte ROM trên chíp.

Bảng các đặc tính của 8051 đầu tiên.

Đặc tính	**Số lượng**
ROM trên chíp	4K byte
RAM	128 byte
Bộ định thời	2
Các chân vào - ra	32
Cổng nối tiếp	1
Nguồn ngắt	6

Trong máy tính, con chíp Intel hay ADM là 1 bộ vi xử lí, nó không có RAM, ROM,cổng IO và các thiết bị ngoại vi on Chip. Còn vi điều khiển chứa 1 bộ vi xử lí và RAM,ROM, cổng IO, và có thể có các thiết bị ngoại vi.

1.2. Sơ đồ chân vi điều khiển 8051:

Là IC đóng vỏ dạng DIP có 40 chân, mỗi chân có một kí hiệu tên và có các chức năng như sau:

* Chân 40: nối với nguồn nuôI +5V.

* Chân 20: nối với đất(Mass, GND).

* Chân 29 (PSEN)(program store enable) là tín hiệu điều khiển xuất ra của 8051, nó cho phép chọn bộ nhớ ngoài và được nối chung với chân của OE (Outout Enable) của EPROM ngoài để cho phép đọc các byte của chương trình. Các xung tín hiệu PSEN hạ thấp trong suốt thời gian thi hành lệnh. Những mã nhị phân của chương trình được đọc từ EPROM đi qua bus dữ liệu và được chốt vào thanh ghi lệnh của 8051 bởi mã lệnh.(chú ý việc đọc ở đây là đọc các lệnh (khác với đọc dữ liệu), khi đó VXL chỉ đọc các bit opcode của lệnh và đưa chúng vào hàng đợi lệnh thông qua các Bus địa chỉ và dữ liệu)

* Chân 30 (ALE : Adress Latch Enable) là tín hiệu điều khiển xuất ra của 8051, nó cho phép phân kênh bus địa chỉ và bus dữ liệu của Port 0.

Chân 31 (EA : Eternal Acess) được đưa xuống thấp cho phép chọn bộ nhớ mã ngoàI đối với 8051.

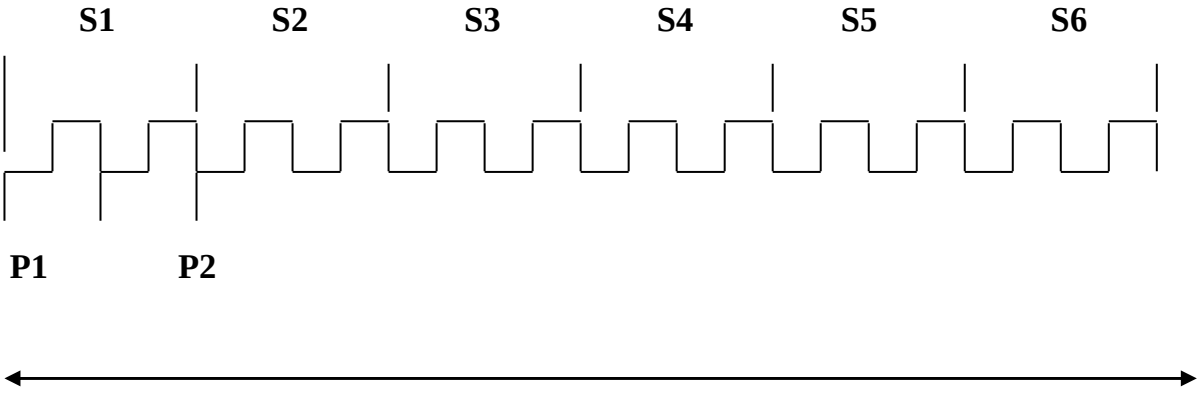
Đối với 8051 thì: EA = 5V: Chọn ROM nội. EA = 0V: Chọn ROM ngoại.

* Chân 18, 19 nối với thạch anh tạo thành mạch tạo dao động cho VĐK

Tần số thạch anh thường dùng trong các ứng dụng là : 11.0592Mhz(giao tiếp với cổng com máy tính) và 12Mhz

Tần số tối đa 24Mhz. Tần số càng lớn VĐK xử lí càng nhanh.

Dao dộng cua thạch anh

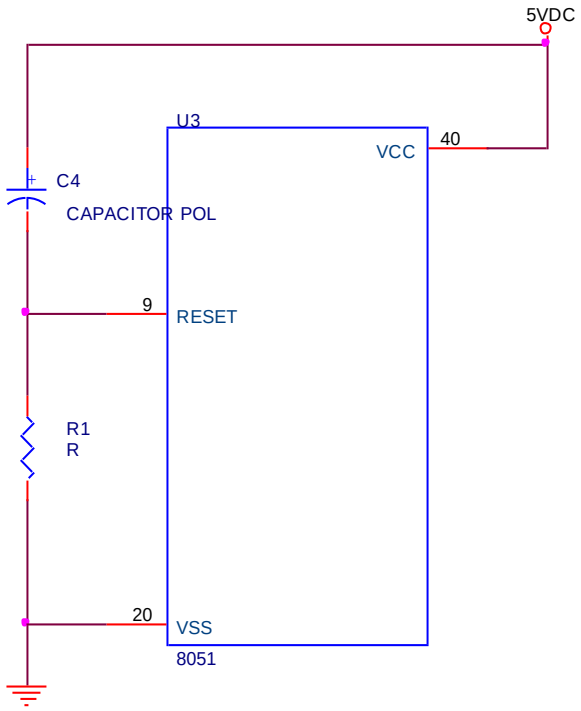


1 chu ki máy

Một chu kì máy = 12 dao động của thạch anh tần số thạch anh là 12 Mhz có nghĩa là tần số làm việc của chip là 1Mhz <-> chu kì là 1uS.Lệnh lập trình cho vi điều khiển có lệnh vi điều khiển mất 1 chu kì máy mới thực hiện xong, có lệnh nhiều hơn một chu kì máy. Cụ thể khi lập trính sẽ biết lệnh đó bao nhiêu chu kì máy.

* Chân 9 được mắc với 1 mạch ngoài tạo thành mạch reset. Khi reset VĐK hoạt động lại từ đầu. (Ram bị xóa, các thanh ghi bị xóa)

Mạch RESET



* 32 chân còn lại chia làm 4 cổng vào ra

1.3. Cấu trúc port I/O

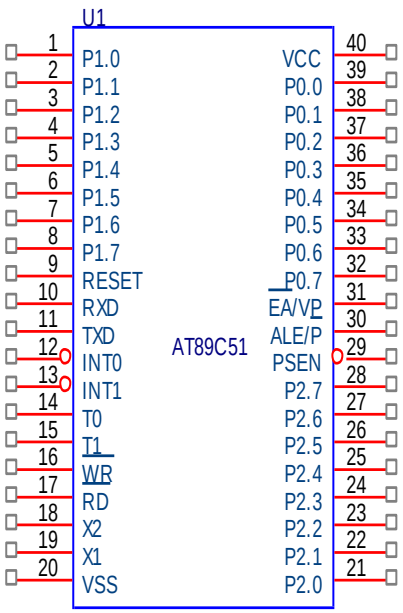
Với 4 cổng vào ra (Port I/O) tức là có thể dùng chân đó để đọc mức logic (0;1 tương ứng với 0V ; 5V)vào hay xuất mức logic ra(0;1)

P0 từ chân 39 → 32 tương ứng là các chân P0_0 → P0_7

P1 từ chân 1 → 8 tương ứng là các chân P1_0 → P1_7

P2 từ chân 21 →28 tương ứng là các chân P2_0 → P2_7

P3 từ chân 10 → 17 tương ứng là các chân P3_0 → P3_7



P0	P1	P2	P3	Port's Bit
P0.0	P1.0	P2.0	P3.0	D0
P0.1	P1.1	P2.1	P3.1	D1
P0.2	P1.2	P2.2	P3.2	D2
P0.3	P1.3	P2.3	P3.3	D3
P0.4	P1.4	P2.4	P3.4	D4
P0.5	P1.5	P2.5	P3.5	D5
P0.6	P1.6	P2.6	P3.6	D6
P0.7	P1.7	P2.7	P3.7	D7

Riêng cổng 3 có 2 chức năng ở mối chân như trên hình vẽ:

P3.0 – RxD : chân nhận dữ liệu nối tiếp khi giao tiếp RS232(Cổng COM).

P3.1 _ TxD : phân truyền dữ liệu nối tiếp khi giao tiếp RS232.

P3.2 _ INTO : interrupt 0 , ngắt ngoài 0.

P3.3 _ INT1: interrupt 1, ngắt ngoài 1.

P3.4 _T0 : Timer0 , đầu vào timer0.

P3.5_T1 : Timer1, đầu vào timer 1.

P3.6_ WR: Write, điều khiển ghi dứ liệu.

P3.7 _RD: Read , điều khiển đọc dữ liệu.

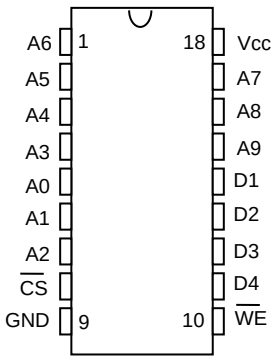
1.4. Tổ chức bộ nhớ:

1.4.1. Tổ chức bộ nhớ vật lý

Tổ chức bộ nhớ cho một hệ Vi xử lý (máy vi tính) phụ thuộc không chỉ vào một hệ Vi xử lý cụ thể, mà còn phụ thuộc vào cách bố trí thuận lợi bên trong hệ thống. Trước hết, hãy làm quen với các khái niệm chip nhớ và từ nhớ để phân tích vấn đề tổ chức vật lý một bộ nhớ, sau đó mở rộng khái niệm tổ chức theo quan điểm của người lập trình (tổ chức logic).

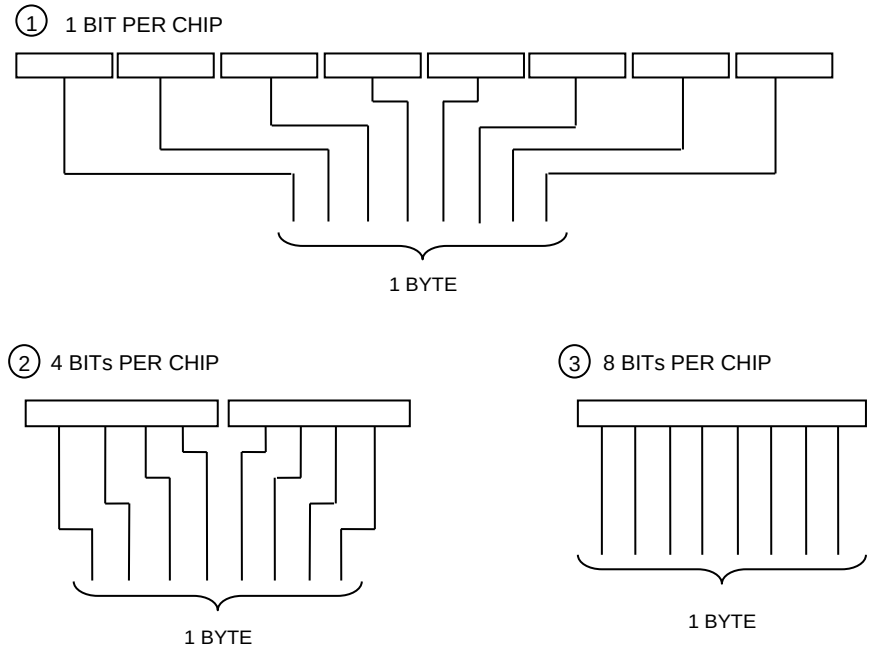
Các chip nhớ được sản xuất dưới nhiều kích cỡ khác nhau, phụ thuộc vào công nghệ chế tạo. Chip nhớ là một vi mạch cụ thể, được bố trí các chân cơ bản như Hình III.8 Các chân của một chip nhớ thông thường gồm các lối vào của BUS địa chỉ, lối dữ liệu, các chân điều khiển chọn chip, ghi/đọc và các chân nguồn.

A0 ÷ A9 Các chân địa chỉ
D1 ÷ D4 Các chân dữ liệu
CS Chân chọn chip
WE Điều khiển Ghi/Đọc
Vcc Chân nguồn nuôi +5V
GND Chân nối đất



Hình III.8 Sơ đồ nối chân một vi mạch nhớ RAM 1Kx4

Tuỳ theo từng chip, số lượng chân địa chỉ và số lượng chân dữ liệu có thể khác nhau phụ thuộc vào độ dài *từ nhớ của chip* và *dung lượng* của chip nhớ. Độ dài từ nhớ của chip nhớ có thể là 1bit, 4 bits hoặc 8 bits, trong khi số chân địa chỉ có thể từ 10 trở lên tuỳ thuộc vào dung lượng của chip nhớ. Trong trường hợp độ dài từ nhớ của chip là 1 bit, ta cần phải ghép song song 8 chip để tạo thành 1 byte, ghép song song 16 chip để tạo một từ word – 2 bytes).



Hình III.8 Tạo từ nhớ 8 bit từ các chíp nhớ có độ dài từ nhớ nhỏ hơn

1.4.2. Thiết kế vỉ nhớ cho hệ Vi xử lý

Thiết kế vỉ nhớ là một việc rất quan trọng và rất cần thiết trong việc xây dựng một hệ Vi xử lý. Các vỉ nhớ được thiết kế thông thường là EPROM, các loại vỉ nhớ RAM, từ các chip nhớ có sẵn. Thông thường, các chíp nhớ được chọn là những chip thông dụng trên thị trường, có các thông số kỹ thuật chủ yếu sau:

a. *Dung lượng nhớ của chip nhớ* tính theo đơn vị Kbyte
b. *Độ dài từ nhớ của chíp nhớ* tính theo số bits
c. *Một số thông số kỹ thuật khác như thời gian truy xuất, công suất tiêu tán của chip v.v…*Những thông số này không có ảnh hưởng lớn đến quá trình thiết kế và xây dựng vỉ nhớ.

Các thông số được cho trước trong việc thiết kế một vỉ nhớ bao gồm:

a. *Loại chip nhớ* ví dụ dùng EPROM 2764 (8Kx8) hay RAM TMS 2064 (8Kx8) v.v…
b. *Dung lượng của vỉ nhớ* là dung lượng vỉ nhớ phải có, ví dụ 64KB, 128KB v.v…
c. *Địa chỉ đầu của vùng nhớ* ví dụ vỉ nhớ có địa chỉ đầu là A0000H chẳng hạn.

Ví dụ minh hoạ: Dùng EPROM 2764 (8Kx8bit) xây dựng vỉ nhớ có dung lượng 32KB, địa chỉ đầu là 22000H.

Giải: Dựa trên yêu cầu của đề ra, phải thực hiện các bước sau:

1. Xác định số chip nhớ cần thiết để tạo từ nhớ cơ bản (độ dài 8 bits), có thể tính theo công thức:

$n = \frac{8}{k}$ trong đó n là số chip cần để tạo được từ nhớ cơ bản

k là độ dài từ nhớ của chip nhớ

Tín hiệu chọn vỏ $\overline{CS}$ của các chip này được nối chung với nhau, các chip này được coi như một chip liên thông, các bit dữ liệu sẽ được định vị theo thứ tự từ $D_7 \div D_0$ tương ứng với các bit từ $D_7 \div D_0$ của BUS dữ liệu.

2. Xác định số chip nhớ, hoặc số chip liên thông để tạo được dung lượng nhớ theo yêu cầu. Trong trường hợp cụ thể của đề ra, cần 4 chip để tạo được dung lượng nhớ 32KB. Tính theo công thức:

$M = \frac{Q}{D}$ trong đó Q là dung lượng của vỉ nhớ

D là dung lượng của chip nhớ hoặc dung lượng của chip liên thông

M là số chip nhớ hoặc số chip liên thông cần thiết

3. Xác định số dây địa chỉ cơ sở (tức là số dây địa chỉ thấp được nối trực tiếp vào chip nhớ hoặc chip liên thông): Số dây địa chỉ m phụ thuộc vào dung lượng nhớ của chip nhớ hoặc chip liên thông theo biểu thức sau:

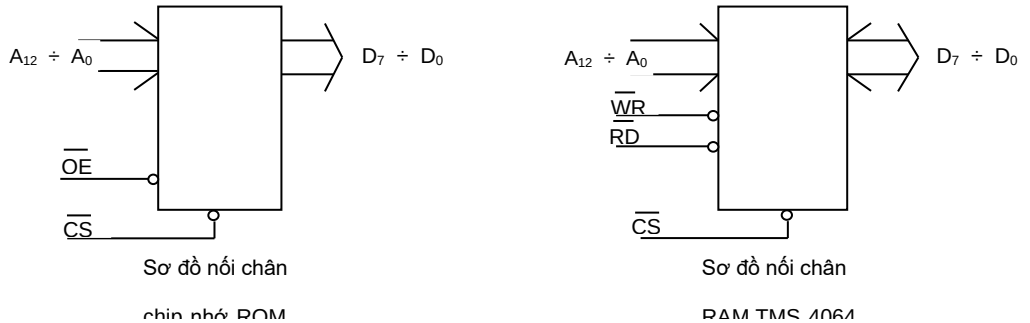
$2^m = D$ trong đó D là dung lượng của chip nhớ

m là số dây địa chỉ cơ sở

4. Từ số chip hoặc số chip liên thông, xác định số dây địa chỉ cần thiết để tạo các dây chọn chip riêng biệt. Tính theo công thức:

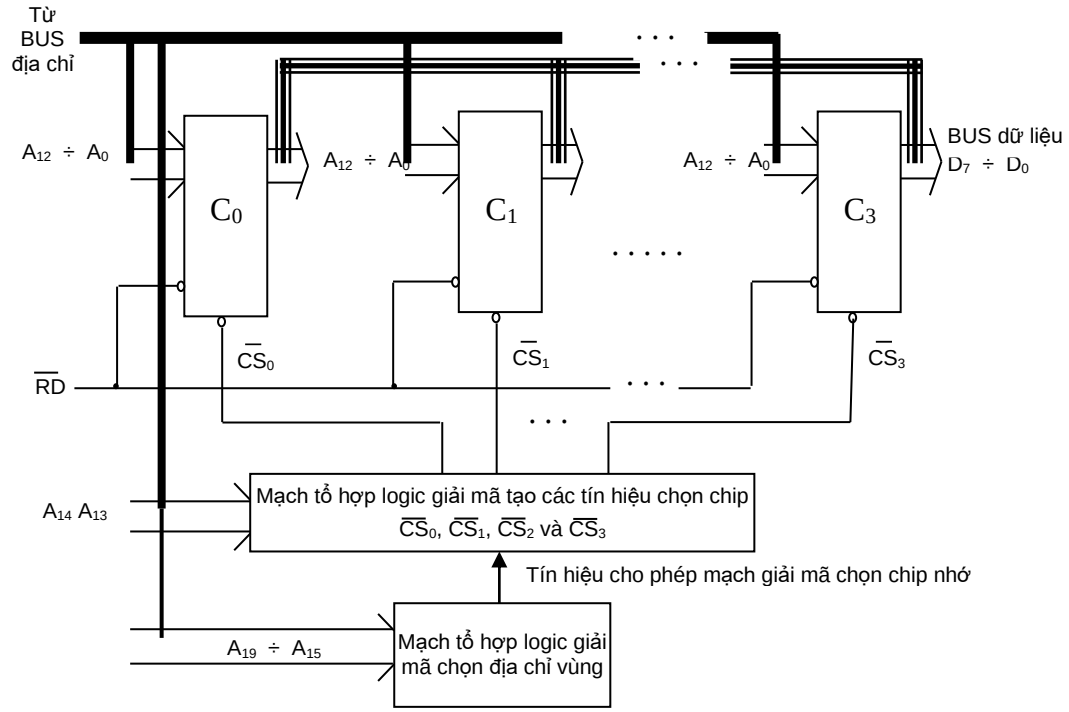
$2^i = M$ trong đó i là số dây địa chỉ cần để giải mã *xác định các tín hiệu chọn chip cho các chip nhớ hoặc chip liên thông*. M là số lượng chip hoặc số lượng chip liên thông. Xây dựng mạch tổ hợp tạo các tín hiệu chọn chip CS_i .

5. Các dây địa chỉ còn lại được sử dụng để tạo tín hiệu xác định vùng nhớ của vỉ nhớ trong không gian nhớ (được gán cho vỉ nhớ theo địa chỉ đầu của vỉ nhớ theo yêu cầu).



Hình II.9 Sơ đồ nối chân chip nhớ ROM và chip nhớ RAM

Sơ đồ khối vỉ nhớ như sau, các mạch tổ hợp logic xây dựng theo kiến thức học được ở môn học *Kỹ thật điện tử số.*



Hình II. 10 Sơ đồ khối vỉ nhớ 32KB từ các chip ROM 2764

1.5. Ram nội và các thanh ghi chức năng đặc biệt SFR của 8051:

F0	**F7**	**F6**	**F5**	**F4**	**F3**	**F2**	**F1**	**F0**
E0	**E7**	**E6**	**E5**	**E4**	**E3**	**E2**	**E1**	**E0**
D0	**D7**	**D6**	**6D**	**6C**	**6B**	**6A**	**69**	**68**
B8	**-**	**-**	**-**	**BC**	**BB**	**BA**	**B9**	**B8**
B0	**B7**	**B6**	**B5**	**B4**	**B3**	**B2**	**B1**	**B0**
A8	**AF**	**AE**	**AD**	**AC**	**AB**	**AA**	**A9**	**A8**
A0	**A7**	**A6**	**A5**	**A4**	**A3**	**A2**	**A1**	**A0**
99	**Không định địa chỉ từng bít**							
98	**9F**	**9E**	**9D**	**9C**	**9B**	**9A**	**99**	**98**

90	**97**	**96**	**95**	**94**	**93**	**92**	**91**	**90**
8D	**Không định địa chỉ từng bít**							
8C	**Không định địa chỉ từng bít**							
8B	**Không định địa chỉ từng bít**							
8A	**Không định địa chỉ từng bít**							
89	**Không định địa chỉ từng bít**							
88	**8F**	**8E**	**8D**	**8C**	**8B**	**8A**	**89**	**88**
87	**Không định địa chỉ từng bít**							
83	**Không định địa chỉ từng bít**							
82	**Không định địa chỉ từng bít**							
81	**Không định địa chỉ từng bít**							
80	**87**	**86**	**85**	**84**	**83**	**82**	**81**	**80**
THANH GHI CHƯC NĂNG ĐẶC BIÊT								

Các thanh ghi SFR có địa chỉ nằm giữa 80H và FFH các địa chỉ này ở trên 80H, vì các địa chỉ từ 00 đến 7FH là địa chỉ của bộ nhớ RAM bên trong 8051. Không phải tất cả mọi địa chỉ từ 80H đến FFH đều do SFR sử dụng, nhưng vị trí ngăn nhớ từ 80H đến FFH chưa dùng là để dữ trữ và lập trình viên 8051 cũng không được sử dụng.

Bảng chức năng của thanh ghi chức năng đặc biệt SFR

SFR định địa chỉ từng bit (những thanh ghi cần nhớ đối với khi lập trình cơ bản C)

1.6. Bộ nhớ ngoài của họ 8051:

Có hai bộ vi điều khiển thành viên khác của họ 8051 là 8052 và 8031.

Bộ vi điều khiển 8052: 8052 có tất cả các đặc tính chuẩn của 8051 ngoài ra nó có thêm 128 byte RAM và một bộ định thời nữa. Hay nói cách khác là 8052 có 256 byte RAM và 3 bộ định thời. Nó cũng có 8K byte ROM. Trên chíp thay vì 4K byte như 8051.

Bảng so sánh các đặc tính của các thành viên họ 8051.

Đặc tính	8051	8052
ROM trên chíp	4K byte	8K byte
RAM	128 byte	256 byte
Bộ định thời	2	3
Chân vào – ra	32	32
Cổng nối tiếp	1	1
Nguồn ngắt	6	8

Do vậy tất cả mọi chương trình viết cho 8051 đều chạy trên 8052 nhưng điều ngược lại là không đúng. Đặc biệt: Một nhà sản xuất chính của họ 8051 khác nữa là Philips Corporation. Hãng này có một dải lựa chọn rộng lớn cho các bộ vi điều khiển họ 8051. Nhiều sản phẩm của hãng đã có kèm theo các đặc tính như các bộ chuyển đổi ADC, DAC, chân PWM, cổng I/0 mở rộng.

Bài 2: Tập lệnh 89C51

❖ GIỚI THIỆU BÀI 2

Bài 2 là bài giới thiệu về tập lệnh trong họ vi điều khiển 89C51. Trên cơ sở đó người học vận dụng vào các bài tập cụ thể để soạn thảo được một chương trình điều khiển với những ứng dụng thực tiễn.

❖ MỤC TIÊU CỦA BÀI 2

- Phân biệt được các kiểu định địa chỉ và dữ liệu
- Trình bày được đặc tính và công dụng của từng lệnh trong 89c51
- Xác định được độ lớn và thời gian thực hiện chương trình
- Kết hợp được các lệnh riêng lẻ để viết được một chương trình đúng theo yêu cầu cho trước
- Tích cực, chủ động và sáng tạo trong học tập.

❖ PHƯƠNG PHÁP GIẢNG DẠY VÀ HỌC TẬP BÀI 2

- *Đối với người dạy: sử dụng phương pháp giảng giảng dạy tích cực (diễn giảng, vấn đáp, dạy học theo vấn đề); yêu cầu người học thực hiện câu hỏi thảo luận của bài (cá nhân hoặc nhóm).*
- *Đối với người học: chủ động đọc giáo trình trước buổi học; hoàn thành đầy đủ câu hỏi thảo luận theo cá nhân hoặc nhóm và nộp lại cho người dạy đúng thời gian quy định.*

❖ ĐIỀU KIỆN THỰC HIỆN BÀI 2

- ***Phòng học chuyên môn hóa/nhà xưởng:*** học tại xưởng thực hành vi điều khiển – có trang bị các máy tính cài sẵn phần mềm mô phỏng.
- ***Trang thiết bị máy móc:*** Máy chiếu và các mô hình, thiết bị dạy học khác
- ***Học liệu, dụng cụ, nguyên vật liệu:*** Chương trình mô đun, giáo trình, tài liệu tham khảo, giáo án, phim ảnh, và các KIT thực hành vi điều khiển.
- ***Các điều kiện khác:*** Không có

❖ KIỂM TRA VÀ ĐÁNH GIÁ BÀI 2

- **Nội dung:**

✓ *Kiểm tra và đánh giá tất cả nội dung về kiến thức, kỹ năng đã nêu trong mục tiêu của bài*

✓ *Năng lực tự chủ và trách nhiệm: Trong quá trình học tập, người học cần:*

+ Nghiên cứu bài trước khi đến lớp

+ Chuẩn bị đầy đủ tài liệu học tập.

+ *Tham gia đầy đủ thời lượng môn học.*

+ *Nghiêm túc trong quá trình học tập.*

- **Phương pháp:**

✓ ***Điểm kiểm tra thường xuyên:*** *1 điểm kiểm tra (hình thức - vấn đáp)*

✓ ***Kiểm tra định kỳ lý thuyết:*** *1 điểm kiểm tra (hình thức – tự luận)*

NỘU DUNG BÀI 2

2.1. Các khái niệm cơ bản:

2.1.1. Khái niệm về lệnh trong vi điều khiển

Bộ VĐK có tập lệnh được tối ưu hoá để ứng dụng trong các hệ thống điều khiển, đo lường 8 bit. Để tăng khả năng truy xuất RAM nội trên các dữ liệu nhỏ, các kiểu định địa chỉ đặc biệt đã được áp dụng. Ngoài ra tập lệnh của VĐK còn hỗ trợ các biến 1 bit, cho phép quản lý bit trực tiếp trong các hệ logic và điều khiển bit có yêu cầu xử lý bit. Do họ VĐK AT89/80C51 có các mã lệnh 8 bit, nên số lệnh có thể lên đến 256 lệnh (thực tế có 255 lệnh, còn 1 lệnh chưa được định nghĩa). Trong đó có 139 lệnh 1 byte, 92 lệnh 2 byte và 24 lệnh 3 byte. Mỗi lệnh đều được đặc trưng bởi mã lệnh (mã máy), mã gợi nhớ, số byte của lệnh và số chu kỳ máy cần để thực thi lệnh. Các lệnh của AT89/80C51 được chia thành 5 nhóm lệnh:

- Nhóm lệnh di chuyển dữ liệu.
- Nhóm lệnh số học.
- Nhóm lệnh logic.
- Nhóm lệnh rẽ nhánh chương trình.
- Nhóm lệnh điều khiển biến logic.

2.1.2. Các ký hiệu dùng trong mô tả lệnh:

Ký hiệu	***ý nghĩa***
<-	Được thay thế bởi…
( )	Nội dung của…
(( ))	Dữ liệu được trỏ bởi…
Rrr	1 trong 8 thanh ghi (R0-R7) của các băng thanh ghi
Ddddddd	Các bit dữ liệu
Aaaaaaaa	Các bit địa chỉ
Bbbbbbbb	địa chỉ của 1 bit
I	Định địa chỉ gián tiếp thông qua R0 hoặc R1
Eeeeeeee	Địa chỉ tương đối 8 bit

2.2. Các cách định địa chỉ.:

+ Rn: Thanh ghi R0-R7 của băng thanh ghi hiện hành đang được chọn để định địa chỉ thanh ghi.

+ Direct: Địa chỉ 8 bit của ô nhớ dữ liệu nội trú, nó có thể là ô nhớ trong RAM nội hoặc SFR. (00h-FFh)

+ @Ri: Ô nhớ 8 bit của RAM nội được định địa chỉ gián tiếp thông qua thanh ghi R0 hoặc R1.

+ Source (Src): toán hạng nguồn, có thể là Rn hoặc direct hoặc @Ri.

+ Dest: Toán hạng đích, có thể là Rn hoặc direct hoặc @Ri.

+ #Data: Hằng số 8 bit chứa trong lệnh.

+ #Data16: Hằng số 16 bit chứa trong lệnh.

+ Bit: Bit được định địa chỉ trực tiếp trong RAM nội trú hoặc SFR.

+ Rel: Offset 8 bit có dấu (từ -128 đến +127). Nó được lệnh SJMP và các lệnh nhảy có điều kiện sử dụng.

+ Addr11: địa chỉ 11 bit của bộ nhớ chương trình , được lệnh ACALL và AJMP sử dụng.

+ Addr16: địa chỉ 16 bit của 64Kb bộ nhớ chương trình, được lệnh LCALL và LJMP sử dụng.

2.3. Các nhóm lệnh

2.3.1. Nhóm lệnh di chuyển dữ liệu

2.3.1.1. Lệnh MOV dạng Byte:

Cú pháp câu lệnh: **MOV <dest-byte>, <src-byte>**

Chức năng: Sao chép nội dung của toán hạng nguồn vào toán hạng đích, nội dung của toán hạng nguồn không thay đổi. Lệnh này không làm ảnh hưởng tới các cờ và các thanh ghi khác.

Câu lệnh	***Số byte***	***Số chu kỳ***	***Mã lệnh***	***Hoạt động***
MOV A, Rn	1	1	11101rrr	(A)<-(Rn)
MOV A, direct	2	1	11100101 aaaaaaaa	(A)<-(direct)
MOV A, @Ri	1	1	1110111i	(A)<-((Ri))
MOV A, #data	2	1	01110100 dddddddd	(A)<-#data
MOV Rn, A	1	1	11111rrr	(Rn)<-(A))
MOV Rn, direct	2	2	10101rrr aaaaaaaa	(Rn)<-(direct)
MOV Rn, #data	2	1	01111rrr dddddddd	(Rn)<-#data
MOV direct, A	2	1	11110101 aaaaaaaa	(direct)<-(A)
MOV direct, Rn	2	2	10001rrr aaaaaaaa	(direct)<-(Rn)
MOV direct, direct	3	2	10000101 aaaaaaaa aaaaaaaa	(direct)<-(direct)
MOV direct, @Ri	2	2	1000011i aaaaaaaa	(direct)<-((Ri))
MOV direct, #data	3	2	01110101 aaaaaaaa dddddddd	(direct)<-#data
MOV @Ri, A	1	1	1111011i	((Ri))<-(A)
MOV @Ri, direct	2	2	1010011i	((Ri))<-(direct)
MOV @Ri, #data	2	1	0111011i dddddddd	((Ri))<-#data

2.3.1.2 Lệnh MOV dạng Bit:

Cú pháp câu lệnh: **MOV <dest-bit>, <scr-bit>**

Chức năng: Chuyển bit dữ liệu ở dạng sao chép toán hạng nguồn vào toán hạng đích. Một trong 2 toán hạng phải là cờ nhớ (C), toán hạng còn lại sẽ là bit bất kỳ được định địa chỉ trực tiếp. Lệnh không làm ảnh hưởng tới các thanh ghi khác hoặc các cờ khác.

Câu lệnh	***Số byte***	***Số chu kỳ***	***Mã lệnh***	***Hoạt động***
MOV C, bit	2	1	10100010 bbbbbbbb	(C)<-(bit)
MOV bit, C	2	2	10010010 bbbbbbbb	(bit)<-(C)

2.3.1.3. Lệnh MOV dạng Word:

Cú pháp câu lệnh: **MOV DPTR, #data16**

Chức năng: Giá trị 16 bit ở toán hạng thứ 2 trực tiếp trong câu lệnh được nạp vào thanh ghi DPTR. Hằng số 16 bit này được đặt ở byte 2 và byte 3 của lệnh. Byte 2 là byte cao được nạp cho thanh ghi DPH, byte 3 là byte thấp được nạp vào thanh ghi DPL. Lệnh này không ảnh hưởng tới các cờ.

Câu lệnh	***Số byte***	***Số chu kỳ***	***Mã lệnh***	***Hoạt động***
MOV DPTR,#data16	3	2	10010000 dddddddd dddddddd	(C)<-(bit)

2.3.1.4. Lệnh chuyển byte mã lệnh:

Cú pháp câu lệnh: **MOVC A, @A + <thanh ghi cơ sở>**

Chức năng: Nạp cho thanh ghi tích luỹ byte mã lệnh từ bộ nhớ chương trình. Địa chỉ của byte được tìm nạp trong bộ nhớ là tổng nội dung của thanh ghi A 8 bit với nội dung của thanh ghi cơ sở 16 bit (có thể là DPTR hoặc PC - thanh ghi đếm chương trình). Trong trường hợp sau, PC được tăng để trỏ đến địa chỉ của lệnh tiếp theo ((PC)<-(PC+1)) trước khi được công với nội dung của thanh ghi A, còn thanh ghi DPTR không bị thay đổi. Lệnh không ảnh hưởng tới các cờ.

Câu lệnh	***Số byte***	***Số chu kỳ***	***Mã lệnh***	***Hoạt động***
MOVC A,@A+DPTR	1	2	10010011	(A)<-((A)+(DPTR))
MOVC A,@A+PC	1	2	10000011	(A)<-((A)+(PC))

2.3.1.5. Lệnh chuyển dữ liệu ra ngoài:

Cú pháp câu lệnh: **MOVX <dest-byte>, <src-byte>**

Chức năng: Chuyển dữ liệu giữa thanh ghi tích luỹ với bộ nhớ ngoài. Các lệnh này được chia làm 2 loại, một loại cung cấp địa chỉ 8 bit và 1 loại cung cấp địa chỉ 16 bit.

Nếu dữ liệu được chuyển là 8 bit, nội dung của R0 hoặc R1 trong băng thanh ghi hiện hành sẽ cung cấp địa chỉ 8 bit đa hợp với dữ liệu trên P0. 8 bit địa chỉ này đủ để mã hoá cho các cổng I/O mở rộng bên ngoài chip hoặc cho 1 dãy RAM kích thước tương đối nhỏ. Với các dãy RAM có kích thước lớn hơn một chút, một vài chân của cổng bất kỳ nào đó có thể được sử dụng để tạo ra các bit địa chỉ cao. Các chân này nên được điều khiển bởi 1 lệnh xuất đặt trước lệnh MOVX.

Nếu dữ liệu được chuyển là 16 bit, thì DPTR tạo ra địa chỉ 16 bit. P2 xuất ra 8 bit địa chỉ cao (nội dung của DPH), còn P0 xuất ra 8 bit địa chỉ thấp đa hợp với dữ liệu.Thanh ghi chức năng đặc biệt P2 duy trì nội dung trước đó trong khi các bộ đệm xuất của P2 đang phát các nội dung của DPH. Dạng này nhanh hơn và hiệu quả hơn khi truy xuất nhiều dãy dữ liệu rất lớn (lên đến 64 Kb) do ta không cần thêm lệnh để thiết lập các cổng khác.

Câu lệnh	***Số byte***	***Số chu kỳ***	***Mã lệnh***	***Hoạt động***
MOVX A, @Ri	1	2	11100011	(A)<-((Ri))
MOVX @Ri, A	1	2	11110011	((Ri))<(A)
MOVX A, @DPTR	1	2	11100000	(A)<-((DPTR))
MOVX @DPTR, A	1	2	11110000	((DPTR))<-(A)

2.3.1.6. Lệnh chuyển số liệu vào ngăn xếp:

Cú pháp câu lệnh: **PUSH direct**

Chức năng: Chuyển số liệu có trong câu lệnh vào ngăn xếp. Trước tiên, con trỏ ngăn xếp (SP) được tăng lên 1, sau đó số liệu sẽ được chuyển vào đỉnh của ngăn xếp mà địa chỉ đỉnh này được trỏ bởi SP. Ngăn xếp nằm ở RAM nội trú.

Câu lệnh	***Số byte***	***Số chu kỳ***	***Mã lệnh***	***Hoạt động***
PUSH direct	2	2	11000000 aaaaaaaa	(SP)<-(SP+1) ((SP))<-(direct)

2.3.1.7. Lệnh chuyển số liệu ra khỏi ngăn xếp:

Cú pháp câu lệnh: **POP direct**

Chức năng: Chuyển nội dung của ngăn xếp ở RAM trong, có địa chỉ được SP trỏ tới đến nơi có địa chỉ trực tiếp trong câu lệnh. Sau đó, con trỏ ngăn xếp (SP) được giảm đi 1. Lệnh không ảnh hưởng tới các cờ.

Câu lệnh	***Số byte***	***Số chu kỳ***	***Mã lệnh***	***Hoạt động***

POP direct	2	2	11010000 aaaaaaaa	(direct)<-((SP)) (SP)<-(SP-1)

2.3.1.8.Hoán chuyển dữ liệu:

Cú pháp câu lệnh: **XCH A, <byte>**

Chức năng: Hoán chuyển nội dung giữa thanh ghi A với thanh ghi hoặc bộ nhớ có địa chỉ chứa trong toán hạng thứ 2 của câu lệnh. Toán hạng thứ 2 có thể được định địa chỉ kiểu thanh ghi, thanh ghi trực tiếp hoặc thanh ghi gián tiếp.

Câu lệnh	***Số byte***	***Số chu kỳ***	***Mã lệnh***	***Hoạt động***
XCH A, Rn	1	1	11001rrr	(A)<-->(Rn)
XCH A, direct	2	1	11000101 aaaaaaaa	(A) <-->(direct)
XCH A, @Ri	1	1	1100011i	(A) <-->((Ri))

2.3.1.9.Hoán chuyển 4 bit thấp:

Cú pháp câu lệnh: **XCHD A,@Ri**

Chức năng: Hoán chuyển 4 bit thấp nội dung trong thanh ghi A với ô nhớ của RAM bên trong, có địa chỉ được định gián tiếp qua thanh ghi được chỉ ra trong lệnh. Lệnh này không ảnh hưởng tới trạng thái các cờ và nửa cao của các thanh ghi trong lệnh.

Câu lệnh	***Số byte***	***Số chu kỳ***	***Mã lệnh***	***Hoạt động***
XCHD A, @Ri	1	1	1101011i	(A3-A0)<-->((Ri3-Ri0))

2.3.2. Nhóm lệnh tính toán số học.

2.3.2.1. Lệnh thực hiện phép cộng.

Cú pháp của câu lệnh: **ADD A, <scr-byte>**

Chức năng: Cộng giá trị 1 byte ở địa chỉ được chỉ ra ở câu lệnh với nội dung trong thanh ghi tích luỹ, kết quả được lưu vào thanh ghi tích luỹ. Nếu có nhớ từ bit số 7 hoặc bit số 3 thì cờ nhớ hoặc cờ nhớ phụ được thiết lập, ngược lại các cờ nêu trên được xoá. Khi cộng 2 số nguyên không dấu mà bị tràn thì cờ nhớ cũng được thiết lập để cho ta biết phép toán bị tràn. Trường hợp thực hiện lệnh ADD mà có nhớ từ bit số 6 nhưng không có nhớ từ bit số 7, hoặc có nhớ từ bit số 7 nhưng không có nhớ từ bit số 6 thì cờ tràn sẽ được thiết lập, ngược lại thì OV bị xoá. Khi cộng 2 số nguyên có dấu mà tổng là 1 số âm thì OV được thiết lập.

Câu lệnh	*Số byte*	*Số chu kỳ*	*Mã lệnh*	*Hoạt động*
ADD A, Rn	1	1	00101rrr	(A)<- (A) + (Rn)
ADD A, direct	2	1	00100101 aaaaaaaa	(A)<- (A) + (direct)
ADD A, @Ri	1	1	0010011i	(A)<- (A) + ((Ri))
ADD A, #data	2	1	00100100 dddddddd	(A)<- (A) + #data

2.3.2.2. Lệnh cộng có nhớ.

Cú pháp của câu lệnh: **ADDC A, <scr-byte>**

Chức năng: Cộng đồng thời nội dung của 1 byte ở địa chỉ được chỉ ra trong câu lệnh với nội dung chứa trong thanh ghi tích luỹ và cờ nhớ. Nếu có nhớ từ bit số 7 hoặc số 3 thì cờ nhớ hoặc cờ nhớ phụ được thiết lập bằng 1, ngược lại các cờ nêu trên bị xoá. Khi cộng các số nguyên không dấu mà bị tràn thì cờ nhớ cũng được thiết lập. Trường hợp thực hiện lệnh ADDC mà có nhớ từ bit số 6 nhưng không nhớ từ bit số 7, hoặc có nhớ từ bit só 7 nhưng không nhớ từ bit số 6 thì cờ tràn sẽ được thiết lập, ngược lại cờ này bị xoá. Khi cộng các số nguyên có dấu mà tổng là 1 số âm thì OV được thiết lập.

Câu lệnh	*Số byte*	*Số chu kỳ*	*Mã lệnh*	*Hoạt động*
ADDC A, Rn	1	1	00110rrr	(A)<- (A) + (C) + (Rn)
ADDC A, direct	2	1	00110101 aaaaaaaa	(A)<- (A) + (C) + (direct)
ADDC A, @Ri	1	1	0011011i	(A)<- (A) + (C) + ((Ri))
ADDC A, #data	2	1	00110100 dddddddd	(A)<- (A) + (C) + #data

2.3.2.3. Lệnh trừ có mượn.

Cú pháp của câu lệnh: **SUBB A, <scr-byte>**

Chức năng: Trừ thanh ghi tích luỹ cho toán hạng thứ 2 và cờ nhớ, kết quả được lưu vào thanh ghi tích luỹ. Cờ nhớ được đặt bằng 1 nếu có số mượn được cần đến cho bit số 7, ngược lại thì cờ nhớ bị xoá. Cờ nhớ phụ được thiết lập nếu có nhớ cho bit 3. Trường hợp thực hiện lệnh SUBB mà có số mượn được cần đến cho bit 7(không phải cho bit 6), hoặc cho bit 6 (không phải cho bit 7) thì cờ tràn sẽ được thiết lập, ngược lại thì OV bị xoá. Khi trừ các số nguyên có dấu mà kết quả là 1 số âm thì OV được thiết lập.

Câu lệnh	*Số byte*	*Số chu kỳ*	*Mã lệnh*	*Hoạt động*
SUBB A, Rn	1	1	10011rrr	(A)<- (A) - (C) - (Rn)
SUBB A, direct	2	1	10010101	(A)<- (A) - (C) - (direct)

			aaaaaaaa	
SUBB A, @Ri	1	1	1001011i	(A)<- (A) - (C) - ((Ri))
SUBB A, #data	2	1	10010100 dddddddd	(A)<- (A) - (C) - #data

2.3.2.4. Lệnh tăng lên 1 đơn vị.

Cú pháp của câu lệnh: **INC <byte>**

Chức năng: Tăng giá trị của byte trong câu lệnh lên 1 đơn vị. Nếu giá trị ban đầu của byte là 0FFh, thì sau khi thực hiện lệnh INC nội dung của byte sẽ là 00h. Lệnh này không làm ảnh hưởng tới trạng thái các cờ.

Câu lệnh	***Số byte***	***Số chu kỳ***	***Mã lệnh***	***Hoạt động***
INC A	1	1	00000100	(A)<- (A) + 1
INC Rn	1	1	00001rrr	(Rn)<- (Rn) + 1
INC direct	2	1	00000101 aaaaaaaa	(direct)<- (direct) + 1
INC @Ri	1	1	0000011i	((Ri))<- ((Ri)) + 1

2.3.2.5. Lệnh giảm 1 đơn vị.

Cú pháp của câu lệnh: **DEC <byte>**

Chức năng: Giảm giá trị của byte trong câu lệnh xuống 1 đơn vị. Nếu giá trị ban đầu của byte là 00h, thì sau khi thực hiện lệnh DEC nội dung của byte sẽ là 0FFh. Lệnh này không làm ảnh hưởng tới trạng thái các cờ.

Câu lệnh	***Số byte***	***Số chu kỳ***	***Mã lệnh***	***Hoạt động***
DEC A	1	1	00010100	(A)<- (A) – 1
DEC Rn	1	1	00011rrr	(Rn)<- (Rn) – 1
DEC direct	2	1	00010101 aaaaaaaa	(direct)<- (direct) – 1
DEC @Ri	1	1	0001011i	((Ri))<- ((Ri)) – 1

2.3.2.6. Lệnh tăng con trỏ dữ liệu.

Cú pháp của câu lệnh: **INC DPTR**

Chức năng: Tăng con trỏ dữ liệu lên 1 đơn vị. Khi byte thấp của con trỏ dữ liệu bị tràn, thì byte cao của con trỏ dữ liệu tăng lên 1 đơn vị. Lệnh này không ảnh hưởng tới trạng thái các cờ.

Câu lệnh	***Số byte***	***Số chu kỳ***	***Mã lệnh***	***Hoạt động***
INC DPTR	1	2	10100011	(DPTR)<- (DPTR) + 1

2.3.2.7. Lệnh thực hiện phép nhân.

Cú pháp của câu lệnh: **MUL AB**

Chức năng: Nhân các số nguyên không dấu 8 bit trong thanh ghi tích luỹ với thanh ghi B. Byte thấp của kết quả 16 bit được lưu trong thanh ghi tích luỹ, còn byte cao được lưu trong thanh ghi B. Nếu kết quả lớn hơn 0FFh thì cờ tràn được thiết lập, cờ nhớ luôn bị xoá.

Câu lệnh	*Số byte*	*Số chu kỳ*	*Mã lệnh*	*Hoạt động*
MUL AB	1	4	10100100	(B)<- byte cao của (A)x(B) (A)<- byte thấp của (A)x(B)

2.3.2.8. Lệnh thực hiện phép chia.

Cú pháp của câu lệnh: **DIV AB**

Chức năng: Chia số nguyên không dấu 8 bit trong thanh ghi tích luỹ cho số nguyên không dấu 8 bit trong thanh ghi B. Thương số được lưu trong thanh ghi tích luỹ, còn số dư được lưu trong thanh ghi B. Cờ tràn và cờ nhớ bị xoá.

Câu lệnh	*Số byte*	*Số chu kỳ*	*Mã lệnh*	*Hoạt động*
DIV AB	1	4	10000100	(A)<- thương của (A)/(B) (B)<- số dư của (A)/(B)

2.3.2.9. Hiệu chỉnh số thập phân.

Cú pháp của câu lệnh: **DA A**

Chức năng: Hiệu chỉnh thập phân nội dung 8 bit trong thanh ghi A sau khi thực hiện phép cộng.

Nếu 4 bit thấp trong thanh ghi A có giá trị lớn hơn 9 hoặc cờ nhớ phụ được thiết lập thì phải cộng thêm 6 vào thanh ghi A để cho chữ số thập phân được chính xác. Phép cộng này sẽ đặt cờ nhớ nếu số nhớ từ 4 bit thấp chuyển đến tất cả 4 bit cao, ngược lại phép toán không xoá cờ nhớ.

Nếu 4 bit cao trong thanh ghi A có giá trị lớn hơn 9 hoặc cờ nhớ (CF) được thiết lập, thì cũng phải cộng thêm 6 vào thanh ghi A.

Câu lệnh	*Số byte*	*Số chu kỳ*	*Mã lệnh*
DA A	1	1	11010100

Hoạt động:

- Nếu [[(A3-A0)>9]hoặc[(AC)=1]] thì (A3-A0)<-(A3-A0) + 6

- Nếu [[(A7-A4)>9]hoặc[(C)=1]] thì (A7-A4)<-(A7-A4) + 6

2.3.3. Nhóm lệnh tính toán logic.

2.3.3.1. Lệnh AND cho các biến 1 byte.

Cú pháp câu lệnh: **ANL <dest-byte>, <src-byte>**

Chức năng: Thực hiện phép toán logic AND theo mức bit giữa các biến dài 1 byte đã cho, kết quả được lưu vào toán hạng đích. Toán hạng nguồn cho phép 6 chế độ địa chỉ hoá. Khi toán hạng đích là thanh ghi tích luỹ thì toán hạng nguồn có thể là thanh ghi, trực tiếp, thanh ghi-gián tiếp hoặc tức thời. Khi toán hạng đích là địa chỉ trực tiếp thì toán hạng nguồn có thể là thanh ghi tích luỹ hoặc dữ liệu tức thời. Lệnh này không làm ảnh hưởng tới trạng thái các cờ.

Câu lệnh	***Số byte***	***Số chu kỳ***	***Mã lệnh***	***Hoạt động***
ANL A, Rn	1	1	01011rrr	(A)<-(A) AND (Rn)
ANL A, direct	2	1	01010101 aaaaaaaa	(A)<-(A) AND (dir.)
ANL A, @Ri	1	1	0101011i	(A)<- (A) AND ((Ri))
ANL A, #data	2	1	01010100 dddddddd	(A)<- (A) AND #data
ANL direct, A	2	1	01010010 aaaaaaaa	(dir.)<-(dir.)AND (A)
ANL direct, #data	3	2	01010011 aaaaaaaa dddddddd	(dir.)<(dir.)AND#data

2.3.3.2. Lệnh AND cho các biến 1 bit

Cú pháp câu lệnh: **ANL C, <src-bit>**

Chức năng: Thực hiện phép tính logic AND cho các biến mức bit. Nếu giá trị logic của toán hạng nguồn bằng 0, thì cờ nhớ bị xoá. Dấu "/" đứng trước 1 toán hạng cho biết bit nguồn được lấy bù trước khi thực hiện AND với cờ nhớ nhưng ***giá trị của bit nguồn không bị thay đổi bởi thao tác lấy bù***. Lệnh này không làm ảnh hưởng tới trạng thái các cờ khác. Toán hạng nguồn chỉ được sử dụng kiểu định địa chỉ trực tiếp.

Câu lệnh	***Số byte***	***Số chu kỳ***	***Mã lệnh***	***Hoạt động***
ANL C, bit	2	2	10000010 bbbbbbbbb	(C)<-(C) AND (bit)
ANL C, /bit	2	2	10110000 bbbbbbbbb	(C)<-(C) AND NOT (bit)

2.3.3.3. Lệnh OR cho các biến 1 byte

Cú pháp câu lệnh: **ORL <dest-byte>, <src-byte>**

Chức năng: Thực hiện phép toán logic OR theo mức bit giữa các biến dài 1 byte đã cho, kết quả được lưu vào toán hạng đích. Toán hạng nguồn cho phép 6 chế độ địa chỉ hoá. Khi toán hạng đích là thanh ghi tích luỹ thì toán hạng nguồn có thể là thanh ghi, trực tiếp, thanh ghi-gián tiếp hoặc tức thời. Khi toán hạng đích là địa chỉ trực tiếp thì toán hạng nguồn có thể là thanh ghi tích luỹ hoặc dữ liệu tức thời. Lệnh này không làm ảnh hưởng tới trạng thái các cờ.

Câu lệnh	*Số byte*	*Số chu kỳ*	*Mã lệnh*	*Hoạt động*
ORL A, Rn	1	1	01001rrr	(A)<-(A) OR (Rn)
ORL A, direct	2	1	01000101 aaaaaaaa	(A)<-(A) OR (dir.)
ORL A, @Ri	1	1	0100011i	(A)<- (A) OR ((Ri))
ORL A, #data	2	1	01000100 dddddddd	(A)<- (A) OR #data
ORL direct, A	2	1	01000010 aaaaaaaa	(dir.)<-(dir.) OR (A)
ORL direct, #data	3	2	01000011 aaaaaaaa dddddddd	(dir.)<(dir.) OR #data

2.3.3.4. Lệnh OR cho các biến 1 bit

Cú pháp câu lệnh: **ORL C, <src-bit>**

Chức năng: Thực hiện phép tính logic OR cho các biến mức bit. Nếu giá trị logic của toán hạng nguồn bằng 1, thì cờ nhớ được thiết lập. Dấu "/" đứng trước 1 toán hạng cho biết bit nguồn được lấy bù trước khi thực hiện OR với cờ nhớ nhưng ***giá trị của bit nguồn không bị thay đổi bởi thao tác lấy bù***. Lệnh này không làm ảnh hưởng tới trạng thái các cờ khác.

Câu lệnh	*Số byte*	*Số chu kỳ*	*Mã lệnh*	*Hoạt động*
ORL C, bit	2	2	01110010 bbbbbbbbb	(C)<-(C) OR (bit)
ORL C, /bit	2	2	10100000 bbbbbbbbb	(C)<-(C) OR NOT (bit)

2.3.3.5. Lệnh X-OR cho các biến 1 byte

Cú pháp câu lệnh: **XRL <dest-byte>, <src-byte>**

Chức năng: Thực hiện phép toán logic X-OR theo mức bit giữa các biến dài 1 byte đã cho, kết quả được lưu vào toán hạng đích. Toán hạng nguồn cho phép 6 chế độ địa chỉ hoá. Khi toán hạng đích là thanh ghi tích luỹ thì toán hạng nguồn có thể là thanh ghi, trực tiếp, thanh ghi-gián tiếp hoặc tức thời. Khi toán hạng đích là địa chỉ trực tiếp thì toán hạng nguồn có thể là thanh ghi tích luỹ hoặc dữ liệu tức thời. Lệnh này không làm ảnh hưởng tới trạng thái các cờ.

Câu lệnh	*Số byte*	*Số chu kỳ*	*Mã lệnh*	*Hoạt động*
XRL A, Rn	1	1	01101rrr	(A)<-(A) XOR (Rn)
XRL A, direct	2	1	01100101 aaaaaaaa	(A)<-(A) XOR (dir.)
XRL A, @Ri	1	1	0110011i	(A)<- (A) XOR ((Ri))

XRL A, #data	2	1	01100100 dddddddd	(A)<- (A) XOR #data
XRL direct, A	2	1	01100010 aaaaaaaa	(dir.)<-(dir.)XOR (A)
XRL direct, #data	3	2	01100011 aaaaaaaa dddddddd	(dir.)<(dir.) XOR #data

2.3.3.6. Lệnh dịch trái thanh ghi A

Cú pháp câu lệnh: **RL A**

Chức năng: 8 bit trong thanh ghi A được dịch trái 1 bit. Bit 7 được quay đến vị trí của bit 0. Các cờ không bị ảnh hưởng.

Câu lệnh	*Số byte*	*Số chu kỳ*	*Mã lệnh*	*Hoạt động*
RL A	1	1	00100011	(An+1) <- (An), với n = 0…6 (A0) <- (A7)

2.3.3.7. Lệnh dịch trái thanh ghi A cùng với cờ nhớ

Cú pháp câu lệnh: **RLC A**

Chức năng: 8 bit trong thanh ghi A và cờ nhớ cùng được dịch trái 1 bit. Bit 7 được chuyển đến cờ nhớ và trạng thái ban đầu của cờ nhớ được đưa về bit 0. Các cờ khác không bị ảnh hưởng.

Câu lệnh	*Số byte*	*Số chu kỳ*	*Mã lệnh*	*Hoạt động*
RLC A	1	1	00110011	(An+1) <- (An), với n = 0…6 (A0) <- (C) (C) <- (A7)

2.3.3.8. Lệnh dịch phải thanh ghi A.

Cú pháp câu lệnh: **RR A**

Chức năng: 8 bit trong thanh ghi A được dịch sang phải 1 bit. Bit 0 được quay đến vị trí của bit 7. Các cờ không bị ảnh hưởng.

Câu lệnh	*Số byte*	*Số chu kỳ*	*Mã lệnh*	*Hoạt động*
RR A	1	1	00000011	(An) <- (An+1), với n = 0…6 (A7) <- (A0)

2.3.3.9. Lệnh dịch phải thanh ghi A cùng với cờ nhớ

Cú pháp câu lệnh: **RRC A**

Chức năng: 8 bit trong thanh ghi A và cờ nhớ cùng được dịch phải 1 bit. Bit 0 được chuyển đến cờ nhớ và trạng thái ban đầu của cờ nhớ được đưa về bit 7. Các cờ khác không bị ảnh hưởng.

Câu lệnh	*Số byte*	*Số chu kỳ*	*Mã lệnh*	*Hoạt động*
RRC A	1	1	00010011	(An) <- (An+1), với n = 0…6 (A7) <- (C) (C) <- (A0)

2.3.3.10. Lệnh tráo đổi nội dung hai nửa byte của A

Cú pháp câu lệnh: **SWAP A**

Chức năng: Tráo đổi nội dung 2 nửa thấp và cao (mỗi nửa 4 bit) của thanh ghi A (các bit từ 0 đến 3 và các bit từ 4 đến 7). Thao tác này còn được hiểu là quay thanh ghi A 4 bit. Các cờ không bị ảnh hưởng.

Câu lệnh	*Số byte*	*Số chu kỳ*	*Mã lệnh*	*Hoạt động*
SWAP A	1	1	11000100	(A3-A0) <- (A7-A4)

2.3.4. Nhóm lệnh rẽ nhánh chương trình.

2.134.1. Lệnh gọi tuyệt đối.

Cú pháp câu lệnh: **ACALL addr11**

Chức năng: Gọi không điều kiện một chương trình con đặt tại địa chỉ được chỉ ra trong câu lệnh. Lệnh này tăng bộ đếm chương trình thêm 2 đơn vị để PC chứa địa chỉ của lệnh kế lệnh ACALL, sau đó cất nội dung 16 bit của PC vào ngăn xếp (byte thấp cất trước) và tăng con trỏ ngăn xếp lên 2 đơn vị. Địa chỉ đích sẽ được hình thành bằng cách ghép 5 bit cao của thanh ghi PC (sau khi được tăng), 3 bit cao của byte mã lệnh và byte thứ 2 của lệnh. Do đó chương trình con được gọi phải nằm trong đoạn 2 Kbyte của bộ nhớ chương trình chí it phải chứa lệnh đầu tiên của chương trình con này. Lệnh không làm ảnh hưởng tới các cờ.

Câu lệnh	*Số byte*	*Số chu kỳ*	*Mã lệnh*	*Hoạt động*
ACALL addr11	2	2	aaa10001 aaaaaaaa	(PC) <- (PC) + 2 (SP) <- (SP) + 1 ((SP)) <- (PC7-PC0) (SP) <- (SP) + 1 ((SP)) <- (PC15-PC8) (PC10-PC0) <- (page address)

2.3.4.2. Lệnh gọi dài.

Cú pháp câu lệnh: **LCALL addr16**

Chức năng: Gọi một chương trình con đặt tại địa chỉ được chỉ ra trong câu lệnh. Lệnh này tăng bộ đếm chương trình thêm 3 đơn vị để PC chứa địa chỉ của lệnh kế lệnh LCALL, sau đó cất nội dung 16 bit của PC vào ngăn xếp (byte thấp cất trước)

và tăng con trỏ ngăn xếp lên 2 đơn vị. Tiếp theo nó sẽ chuyển byte thứ 2 và byte thứ 3 trong câu lệnh LCALL vào byte cao và byte thấp của PC. Việc thực thi chương trình tiếp tục với lệnh ở địa chỉ này. Như vậy chương trình con có thể bắt đầu bằng bất cứ nơi nào trong không gian bộ nhớ chương trình 64 Kbyte. Lệnh không làm ảnh hưởng tới các cờ.

Câu lệnh	*Số byte*	*Số chu kỳ*	*Mã lệnh*	*Hoạt động*
LCALL addr16	3	2	00010010 aaaaaaaa aaaaaaaa	(PC) <- (PC) + 3 (SP) <- (SP) + 1 ((SP)) <- (PC7-PC0) (SP) <- (SP) + 1 ((SP)) <- (PC15-PC8) (PC) <- addr15-addr0

2.3.4.3. Lệnh quay trở lại từ chương trình con.

Cú pháp câu lệnh: **RET**

Chức năng: Trở về từ chương trình con. Lệnh này được thực hiện sau khi thực hiện xong lệnh ACALL hoặc LCALL. RET lấy lại byte cao và byte thấp của PC từ ngăn xếp, giảm SP đI 2 đơn vị. Chương trình tiếp tục được thực hiện với lệnh có địa chỉ ở trong PC. Các cờ không bị ảnh hưởng.

Câu lệnh	*Số byte*	*Số chu kỳ*	*Mã lệnh*	*Hoạt động*
RET	1	2	00100010	(PC15-PC8) <- ((SP)) (SP) <- (SP) - 1 (PC7-PC0) <- ((SP)) (SP) <- (SP) – 1

2.3.4.4. Lệnh quay trở lại từ ngắt.

Cú pháp câu lệnh: **RETI**

Chức năng: Trở về từ chương trình con. RETI lấy lại byte cao và byte thấp của PC từ ngăn xếp, phục hồi logic ngắt để có thể nhận các ngắt khác có cùng mức ưu tiên ngắt với ngắt được xử lý, sau đó giảm SP đi 2 đơn vị. Chương trình tiếp tục được thực hiện với lệnh trước khi xử lý ngắt với địa chỉ ở trong PC. Các cờ không bị ảnh hưởng.

Câu lệnh	*Số byte*	*Số chu kỳ*	*Mã lệnh*	*Hoạt động*
RETI	1	2	00110010	(PC15-PC8) <- ((SP)) (SP) <- (SP) - 1 (PC7-PC0) <- ((SP)) (SP) <- (SP) – 1

2.3.4.5. Lệnh nhảy gián tiếp.

Cú pháp câu lệnh: **JMP @A+DPTR**

Chức năng: Cộng giá trị không dấu 8 bit của thanh ghi A với con trỏ dữ liệu 16 bit và nạp kết quả vào bộ đếm chương trình, kết quả này chính là địa chỉ để nạp lệnh kế tiếp. Việc cộng 16 bit được thực hiện: Số nhớ từ 8 bit thấp được truyền đến tất cả các bit cao. Cả 2, thanh ghi A và DPTR đều không bị thay đổi. Lệnh này không ảnh hưởng tới trạng thái các cờ.

Câu lệnh	*Số byte*	*Số chu kỳ*	*Mã lệnh*	*Hoạt động*
JMP @A+DPTR	1	2	01110011	(PC)<-(A)+(DPTR)

2.3.4.6. Lệnh nhảy nếu 1 bit được thiết lập.

Cú pháp câu lệnh: **JB bit, rel**

Chức năng: Nếu bit đã cho có giá trị bằng 1 thì nó nhảy tới địa chỉ đã xác định trong câu lệnh, ngược lại nó sẽ tiếp tục thực hiện lệnh tiếp theo. Địa chỉ đích được tính bằng cách cộng thêm độ lệch có dấu (tương đối) trong byte thứ 3 của lệnh với nội dung trong PC (sau khi được tăng đến địa chỉ của byte đầu tiên của lệnh kế tiếp). Bit được kiểm tra không bị thay đổi, lệnh không ảnh hưởng tới các cờ.

Câu lệnh	*Số byte*	*Số chu kỳ*	*Mã lệnh*	*Hoạt động*
JB bit, rel	3	2	00100000 bbbbbbbbb eeeeeeee	(PC)<-(PC)+3 Nếu (bit)=1 thì: (PC)<-(PC) + rel

2.3.4.7. Lệnh nhảy nếu 1 bit không được thiết lập.

Cú pháp câu lệnh: **JNB bit, rel**

Chức năng: Nếu bit đã cho có giá trị bằng 0 thì nó nhảy tới địa chỉ đã xác định trong câu lệnh, ngược lại nó sẽ tiếp tục thực hiện lệnh tiếp theo. Địa chỉ đích được tính bằng cách cộng thêm độ lệch có dấu (tương đối) trong byte thứ 3 của lệnh với nội dung trong PC (sau khi được tăng đến địa chỉ của byte đầu tiên của lệnh kế tiếp). Bit được kiểm tra không bị thay đổi, lệnh không ảnh hưởng tới các cờ.

Câu lệnh	*Số byte*	*Số chu kỳ*	*Mã lệnh*	*Hoạt động*
JNB bit, rel	3	2	00110000 bbbbbbbbb eeeeeeee	(PC)<-(PC)+3 Nếu (bit)=0 thì: (PC)<-(PC) + rel

2.3.4.8. Lệnh nhảy nếu 1 bit được thiết lập và xoá bit đó.

Cú pháp câu lệnh: **JBC bit, rel**

Chức năng: Nếu bit đã cho có giá trị bằng 1 thì nó nhảy tới địa chỉ đã xác định trong câu lệnh và xoá bit này, ngược lại nó sẽ tiếp tục thực hiện lệnh tiếp theo. Địa chỉ đích được tính bằng cách cộng thêm độ lệch có dấu (tương đối) trong byte thứ 3 của lệnh với nội dung trong PC (sau khi được tăng đến địa chỉ của byte đầu tiên của lệnh kế tiếp). Lệnh không ảnh hưởng tới các cờ.

Câu lệnh	***Số byte***	***Số chu kỳ***	***Mã lệnh***	***Hoạt động***
JBC bit, rel	3	2	00010000 bbbbbbbb eeeeeeee	(PC)<-(PC)+3 Nếu (bit)=1 thì: (bit)<- 0 (PC)<- (PC) + rel

2.3.4.9. Lệnh nhảy nếu cờ nhớ được thiết lập.

Cú pháp câu lệnh: **JC rel**

Chức năng: Nếu cờ CF có giá trị bằng 1 thì nó nhảy tới địa chỉ đã xác định trong câu lệnh, ngược lại nó sẽ tiếp tục thực hiện lệnh tiếp theo. Địa chỉ đích được tính bằng cách cộng thêm độ lệch có dấu (tương đối) trong byte thứ 2 của lệnh với nội dung trong PC (sau khi được tăng bởi 2). Lệnh không ảnh hưởng tới các cờ.

Câu lệnh	***Số byte***	***Số chu kỳ***	***Mã lệnh***	***Hoạt động***
JC rel	2	2	01000000 eeeeeeee	(PC)<-(PC)+2 Nếu (C)=1 thì: (PC)<- (PC) + rel

2.3.4.10. Lệnh nhảy nếu cờ nhớ không được thiết lập.

Cú pháp câu lệnh: **JNC rel**

Chức năng: Nếu cờ CF có giá trị bằng 0 thì nó nhảy tới địa chỉ đã xác định trong câu lệnh, ngược lại nó sẽ tiếp tục thực hiện lệnh tiếp theo. Địa chỉ đích được tính bằng cách cộng thêm độ lệch có dấu (tương đối) trong byte thứ 2 của lệnh với nội dung trong PC (sau khi được tăng bởi 2). Lệnh không ảnh hưởng tới các cờ.

Câu lệnh	***Số byte***	***Số chu kỳ***	***Mã lệnh***	***Hoạt động***
JNC rel	2	2	01010000 eeeeeeee	(PC)<-(PC)+2 Nếu (C)=0 thì: (PC)<- (PC) + rel

2.3.4.11. Lệnh nhảy nếu thanh ghi A bằng 0.

Cú pháp câu lệnh: JZ rel

Chức năng: Nếu tất cả các bit của thanh ghi A có giá trị bằng 0 thì nó nhảy tới địa chỉ đã xác định trong câu lệnh, ngược lại nó sẽ tiếp tục thực hiện lệnh tiếp theo. Địa chỉ đích được tính bằng cách cộng thêm độ lệch có dấu (tương đối) trong byte thứ 2 của lệnh với nội dung trong PC (sau khi được tăng bởi 2). Lệnh không ảnh hưởng tới các cờ. Nội dung thanh ghi A không bị thay đổi.

Câu lệnh	***Số byte***	***Số chu kỳ***	***Mã lệnh***	***Hoạt động***
JZ rel	2	2	01100000 eeeeeeee	(PC)<-(PC)+2 Nếu (A)=0 thì: (PC)<- (PC) + rel

2.3.4.12. Lệnh nhảy nếu thanh ghi A khác 0.

Cú pháp câu lệnh: **JNZ rel**

Chức năng: Nếu có 1 hoặc nhiều bit của thanh ghi A có giá trị bằng 1 thì nó nhảy tới địa chỉ đã xác định trong câu lệnh, ngược lại nó sẽ tiếp tục thực hiện lệnh tiếp theo. Địa chỉ đích được tính bằng cách cộng thêm độ lệch có dấu (tương đối) trong byte thứ 2 của lệnh với nội dung trong PC (sau khi được tăng bởi 2). Lệnh không ảnh hưởng tới các cờ. Nội dung thanh ghi A không bị thay đổi.

Câu lệnh	***Số byte***	***Số chu kỳ***	***Mã lệnh***	***Hoạt động***
JNZ rel	2	2	01110000 eeeeeeee	(PC)<-(PC)+2 Nếu (A) < > 0 thì: (PC)<- (PC) + rel

2.3.4.13. Lệnh nhảy khi so sánh 2 toán hạng.

Cú pháp câu lệnh: **CJNE <dest-byte>, <src-byte>, rel**

Chức năng: So sánh giá trị của 2 toán hạng đầu tiên, nếu 2 toán hạng không bằng nhau thì chương trình được rẽ nhánh. Địa chỉ đích rẽ nhánh được tính bằng cách cộng độ lệch tương đối (có dấu) trong byte sau cùng của lệnh với nội dung của PC (sau khi nội dung của PC được tăng đến địa chỉ bắt đầu của lệnh kế tiếp CJNZ). Cờ nhớ (CF) sẽ được thiết lập nếu như giá trị nguyên không dấu của toán hạng đích nhỏ hơn giá trị nguyên không dấu của toán hạng nguồn, ngược lại thì cờ này bị xoá. Lệnh này không làm thay đổi giá trị của các toán hạng

Câu lệnh	***Số byte***	***Số chu kỳ***	***Mã lệnh***	***Hoạt động***
CJNE A, direct, rel	3	2	10110101 aaaaaaaa eeeeeeee	(PC)<-(PC)+3 Nếu (A) < > (dir.) thì: (PC)<- (PC) + offset

				Nếu (A) < (dir.) thì: (C) <- 1, ngược lại: (C) <- 0
CJNE A, #data, rel	3	2	10110100 dddddddd eeeeeeee	(PC)<-(PC)+3 Nếu (A) < > #data thì: (PC)<- (PC) + offset Nếu (A) < #data thì: (C) <- 1, ngược lại: (C) <- 0
CJNE Rn, #data, rel	3	2	10111rrr dddddddd eeeeeeee	(PC)<-(PC)+3 Nếu (Rn)< >#data thì: (PC)<- (PC) + offset Nếu (Rn) < #data thì: (C) <- 1, ngược lại: (C) <- 0
CJNE @Ri, #data, rel	3	2	1011011i dddddddd eeeeeeee	(PC)<-(PC)+3 Nếu ((Ri))< >#data thì: (PC)<- (PC) + offset Nếu ((Ri)) < #data thì: (C) <- 1, ngược lại: (C) <- 0

2.3.4.14. Lệnh giảm và nhảy.

Cú pháp câu lệnh: **DJNZ <byte>, <rel-address>**

Chức năng: Giảm ô nhớ đi 1 và nhảy tới địa chỉ cho bởi toán hạng thứ 2 nếu như kết quả khác 0. Nếu kết quả ban đầu là 00h thì nó chuyển qua 0FFh. Địa chỉ đích được tính bằng cách cộng thêm độ lệch có dấu trong byte lệnh cuối cùng với nội dung của PC (sau khi tăng PC tới byte đầu tiên của lệnh tiếp theo). Ngăn nhớ được giảm giá trị có thể là 1 thanh ghi hoặc 1 byte địa chỉ trực tiếp. Lệnh này không ảnh hưởng tới trạng thái các cờ.

Câu lệnh	***Số byte***	***Số chu kỳ***	***Mã lệnh***	***Hoạt động***
DJNZ Rn, rel	2	2	11011rrr eeeeeeee	(PC)<-(PC)+2 (Rn)<- (Rn) - 1 Nếu (Rn) < > 0 thì: (PC) <- (PC) + rel
DJNZ Direct, rel	3	2	11010101 aaaaaaaa eeeeeeee	(PC)<-(PC)+2 (dir.)<- (dir.) - 1 Nếu (dir.) < > 0 thì: (PC) <- (PC) + rel

2.3.4.15. Lệnh tạm ngừng hoạt động.

Cú pháp câu lệnh: **NOP**

Chức năng: Tạm ngừng hoạt động khi có lệnh này và chương trình sẽ tiếp tục được thực hiện ở lệnh tiếp theo. Lệnh này không ảnh hưởng tới trạng thái các thanh ghi và các cờ.

Câu lệnh	*Số byte*	*Số chu kỳ*	*Mã lệnh*	*Hoạt động*
NOP	1	1	00000000	(PC)<-(PC)+2

2.3.5. Nhóm lệnh điều khiển biến logic.

2.3.5.1. Lệnh xoá bit

Cú pháp câu lệnh: **CLR bit**

Chức năng: Xoá bit được chỉ ra trong câu lệnh về 0. Lệnh này có thể thao tác trên cờ nhớ, hoặc trên 1 bit bất kỳ được định địa chỉ trực tiếp. Lệnh không làm ảnh hưởng tới trạng thái các cờ.

Câu lệnh	*Số byte*	*Số chu kỳ*	*Mã lệnh*	*Hoạt động*
CLR C	1	1	11000011	(C) <- 0
CLR bit	2	1	11000010 bbbbbbbb	(bit) <- 0

2.3.5.2. Lệnh xoá thanh ghi tích luỹ

Cú pháp câu lệnh: **CLR A**

Chức năng: Xoá tất cả các bit của thanh ghi tích luỹ về 0. Các cờ không bị ảnh hưởng.

Câu lệnh	*Số byte*	*Số chu kỳ*	*Mã lệnh*	*Hoạt động*
CLR A	1	1	11100100	(A) <- 0

2.3.5.3. Lệnh thiết lập bit

Cú pháp câu lệnh: **SETB bit**

Chức năng: Thiết lập bit được chỉ ra trong câu lệnh lên mức logic 1. Lệnh này có thể thao tác trên cờ nhớ, hoặc trên 1 bit bất kỳ được định địa chỉ trực tiếp. Lệnh không làm ảnh hưởng tới trạng thái các cờ.

Câu lệnh	*Số byte*	*Số chu kỳ*	*Mã lệnh*	*Hoạt động*
SETB C	1	1	11010011	(C) <- 1
SETB bit	2	1	11010010 bbbbbbbb	(bit) <- 1

2.3.5.4. Lệnh lấy bù của bit

Cú pháp câu lệnh: **CPL <bit>**

Chức năng: Lấy bù bit được chỉ ra trong câu lệnh. Một bit có giá trị 1 được đổi thành 0 và ngược lại. Lệnh này có thể thao tác trên cờ nhớ, hoặc trên 1 bit bất kỳ được định địa chỉ trực tiếp. Lệnh không làm ảnh hưởng tới trạng thái các cờ.

Câu lệnh	***Số byte***	***Số chu kỳ***	***Mã lệnh***	***Hoạt động***
CPL C	1	1	10110011	(C) <- NOT (C)
CPL bit	2	1	10110010 bbbbbbbb	(bit) <- NOT (bit)

2.3.5.5. Lệnh lấy bù của thanh ghi tích luỹ

Cú pháp câu lệnh: **CPL A**

Chức năng: Lấy bù tất cả các bit của thanh ghi A. Lệnh không làm ảnh hưởng tới trạng thái các cờ.

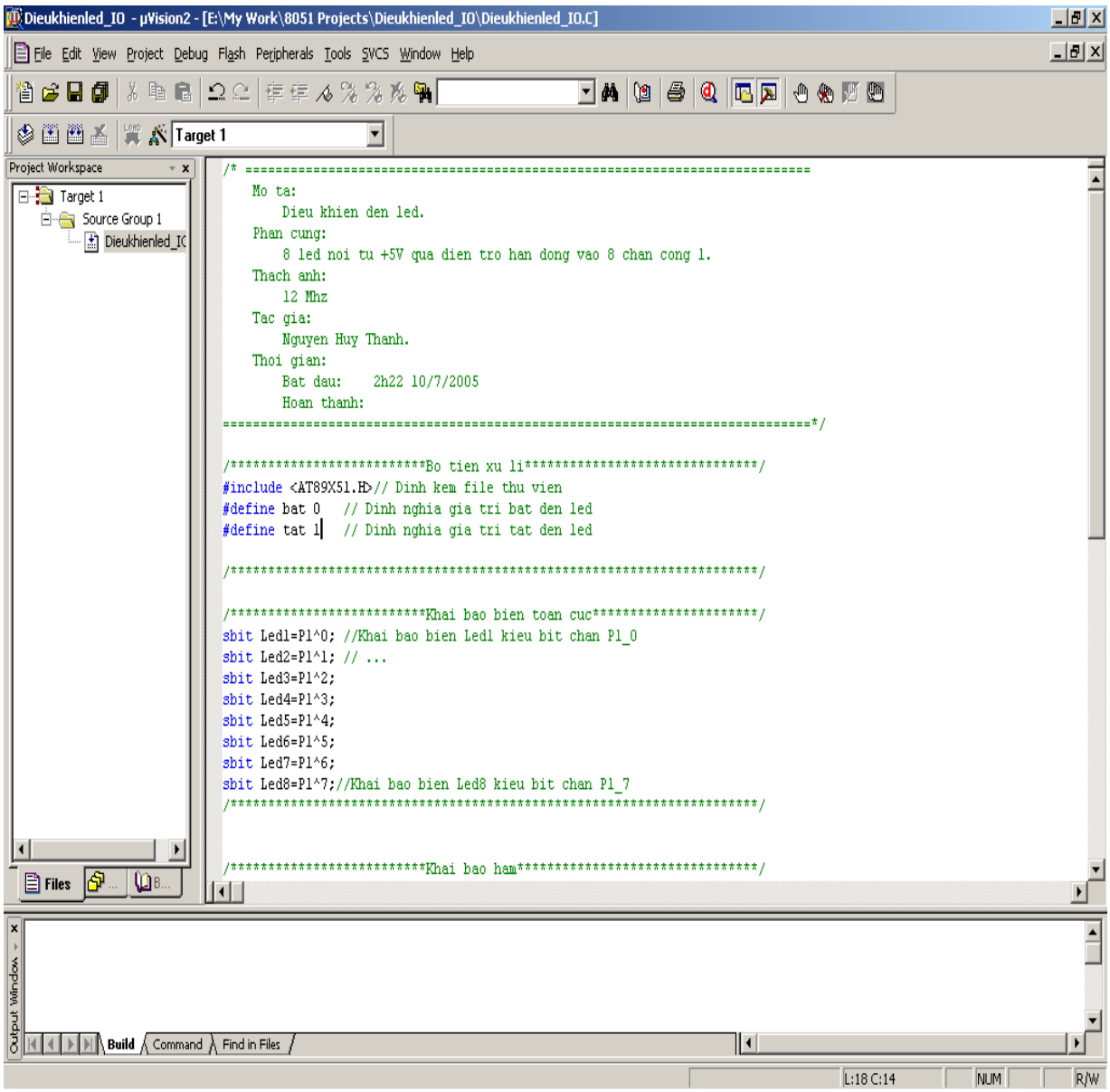
Câu lệnh	***Số byte***	***Số chu kỳ***	***Mã lệnh***	***Hoạt động***
CPL A	1	1	11110100	(A) <- NOT (A)

2.4. Một số bài tập luyện tập:

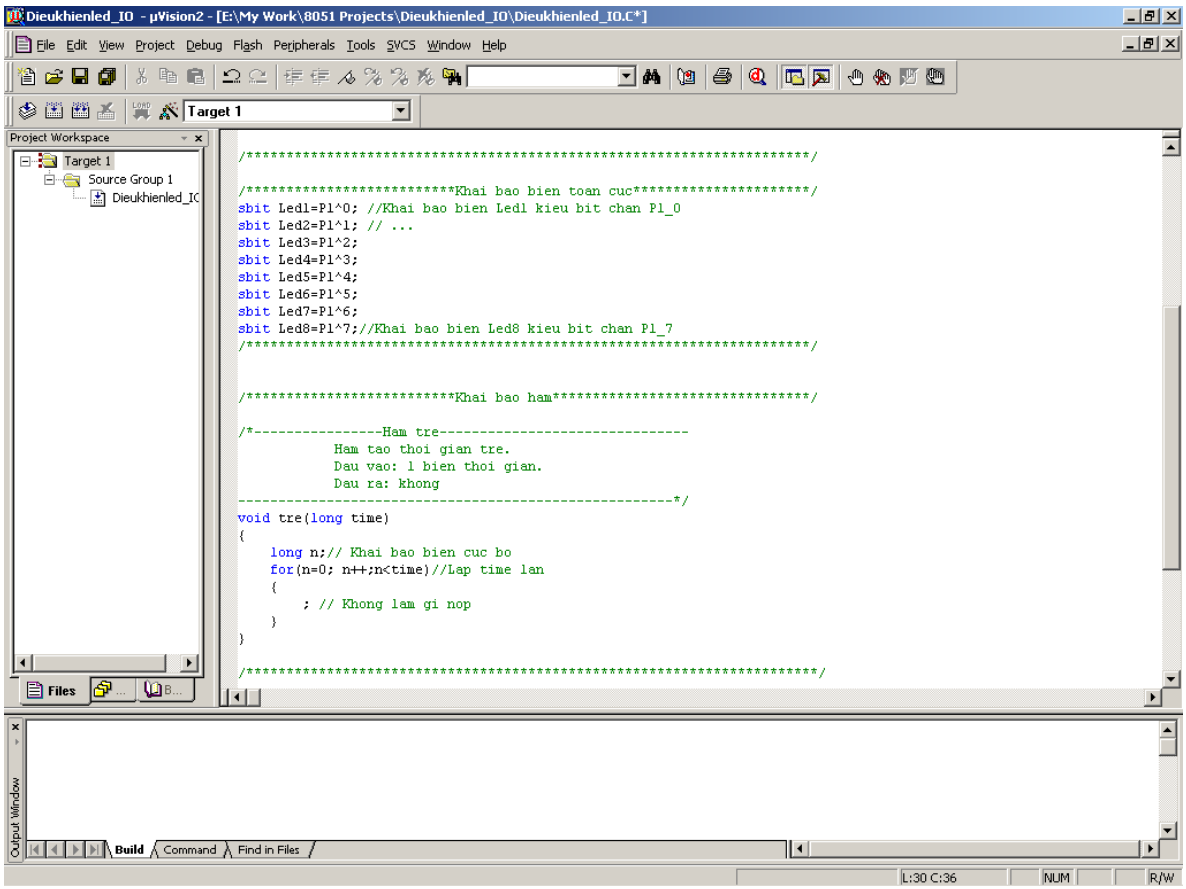
2.4.1. Trình tự thực hiện một mạch điều khiển

Bài tập mẫu: Viết 1 mạch điều khiển đèn led đơn theo mẫu. Khi viết xong 1 dòng lệnh nên giải thích dòng lệnh đó làm gì. Việc thực hiện như sau:

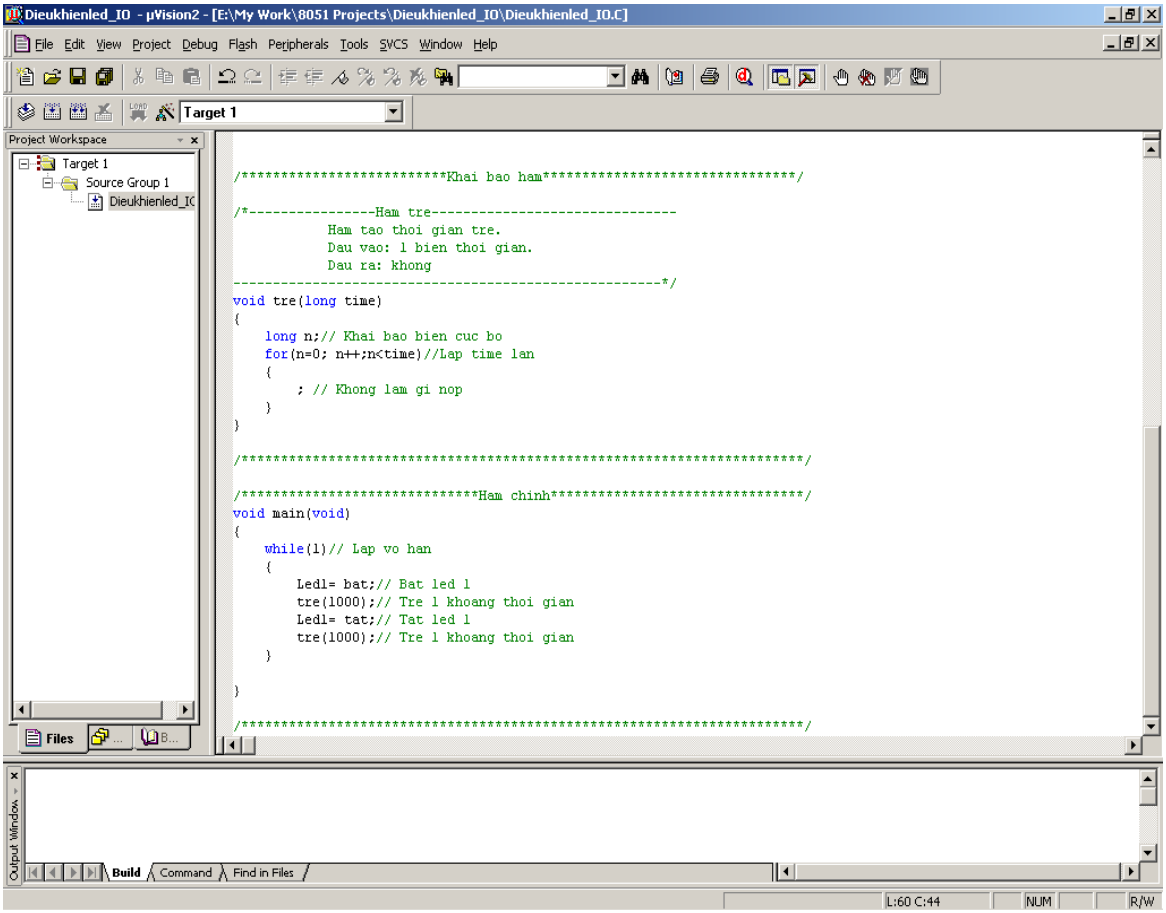
1. Soạn thảo chương trình



Ta nên chia chương trình như màn hình trên. Với 1 file nhỏ thì nó hơi rườm rà. Nhưng với 1 file lớn khoảng 1000 dòng code thì nó lại rất rõ ràng. Nên tạo 1 file mẫu rồi nhớ vào 1 file text để ở đâu đó mỗi lần dùng chỉ việc copy qua chứ không nên mỗi lần tạo một file như vậy lại phản tác dụng. Phía trên là phần bộ tiền xử lí và khai báo biến. Tiếp theo là viết hàm trễ.



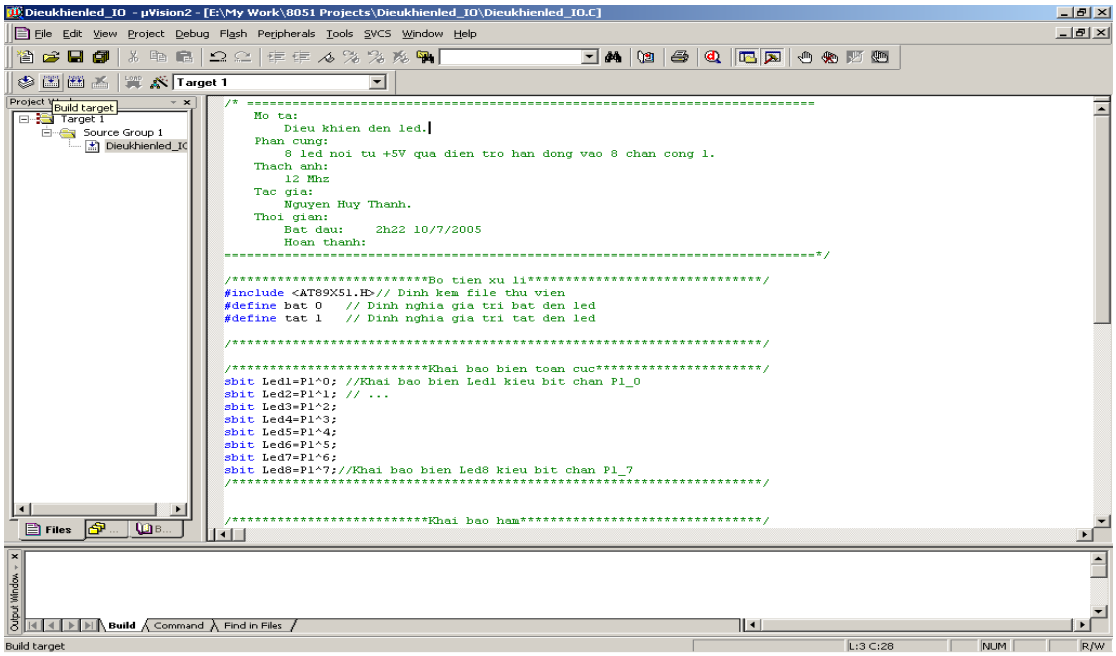
Tiếp theo là viết hàm main. Như sau:



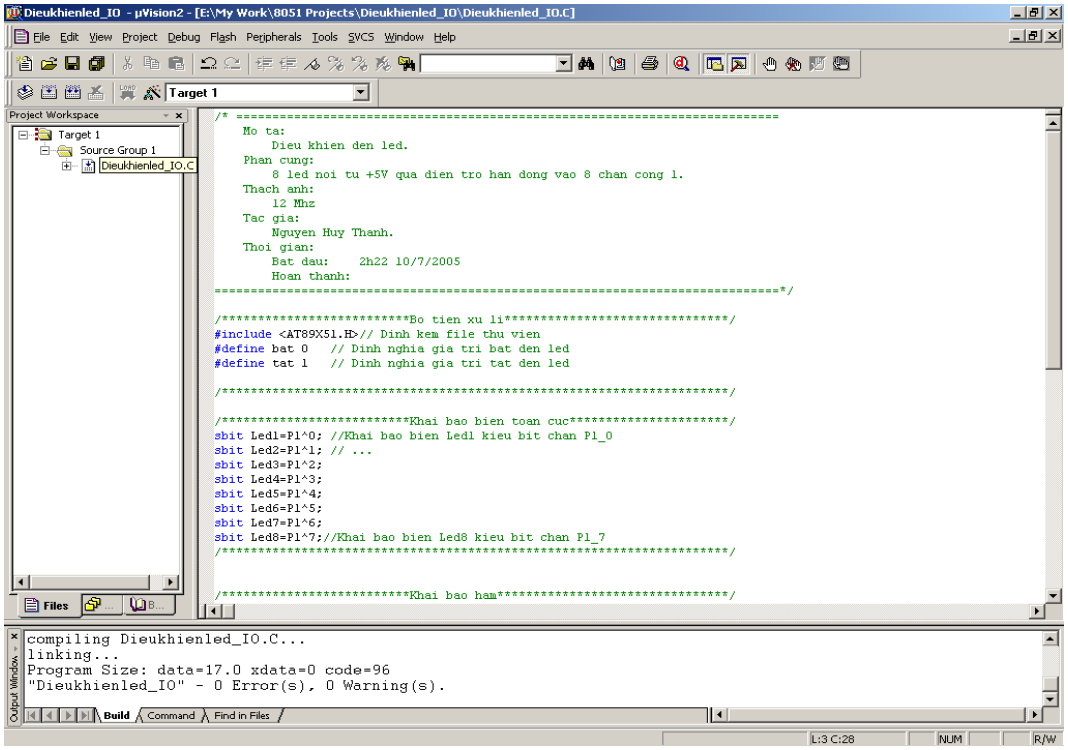
Nhấn tổ hợp phím Ctrl+S. Hoặc chọn File → Save để nhớ file vừa soạn thảo.

2. Dịch chương trình:

Soạn thảo song nhấn tổ hợp phím Ctrl +S để lưu. Sau đó biên dịch chương trình bằng cách ấn phím F7 hoặc chọn Build target là biểu tượng ngay trên cửa sổ workspace như trên hình:



Được cửa sổ như sau:



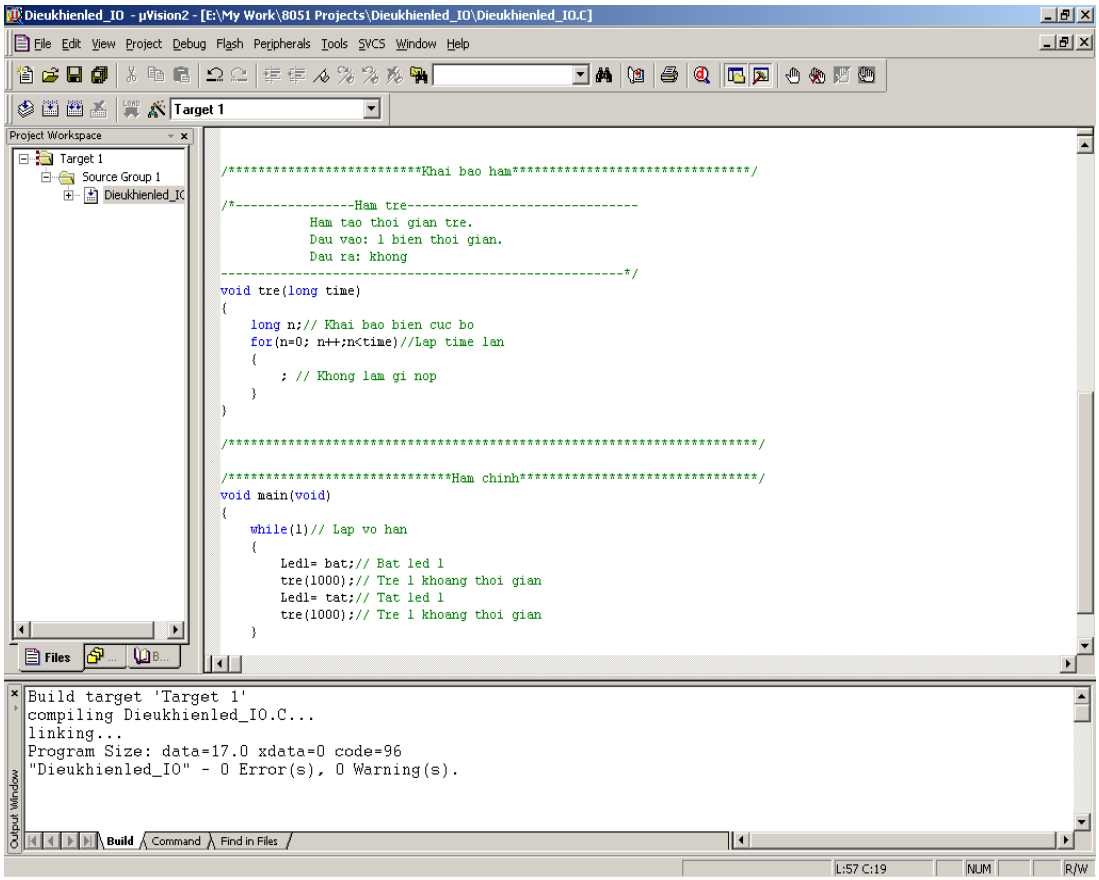
Trong cửa sổ Output Window ngay phía trên dòng chữ này có các dòng chữ Compiling …

Linking…

Program Size: data =17.0 code =96

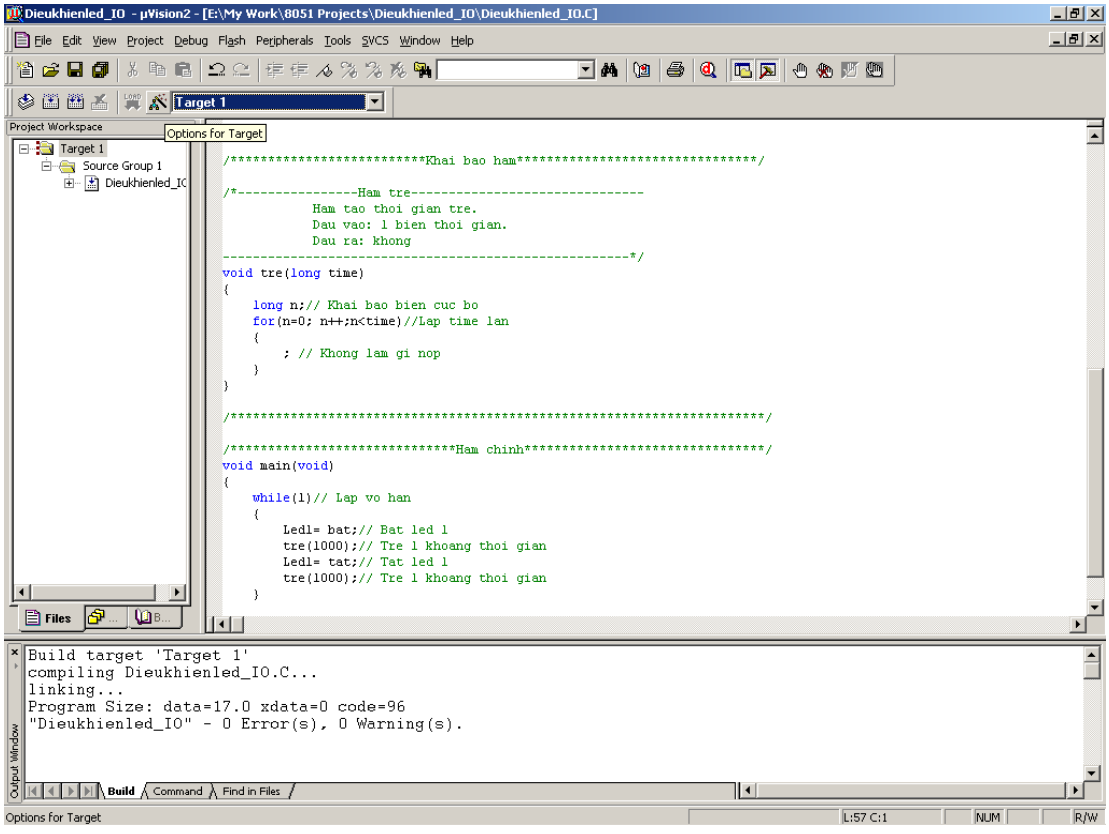
… 0 error , 0 Warning .

Nếu không có các dòng trên là mạch đã lỗi, khi đó, ta kiểm tra xem soạn thảo đúng chưa. Sau khi sửa lỗi thì biên dịch lại. Sau khi dịch lại được hình sau:

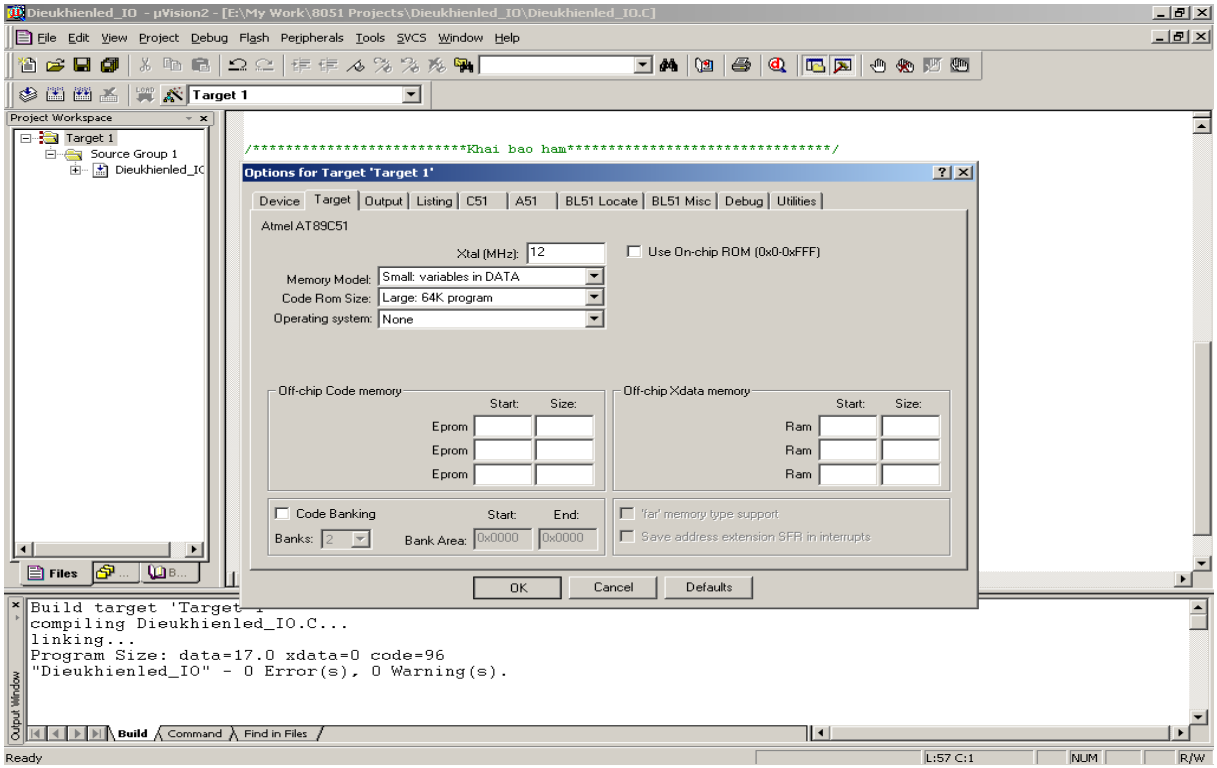


3. Chạy mô phỏng và sửa lỗi.

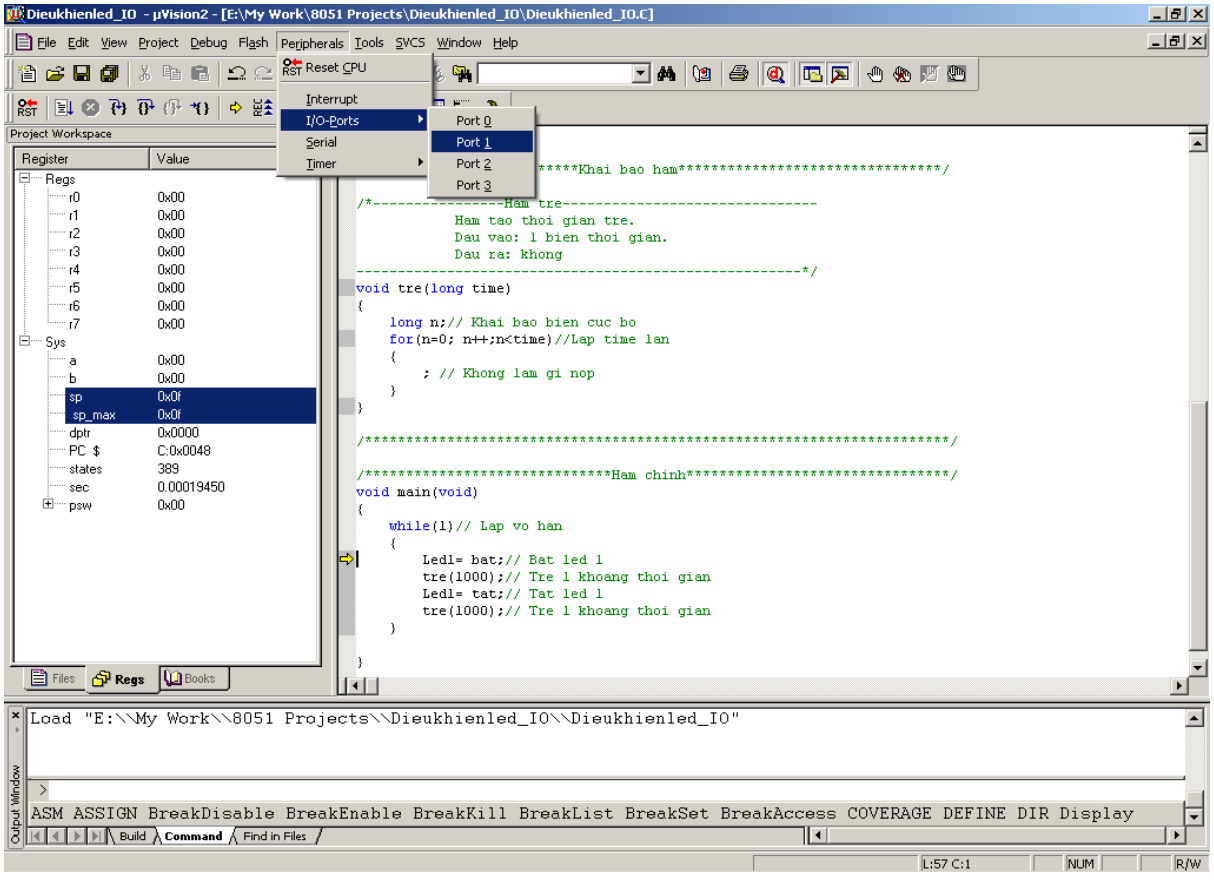
Trước khi mô phỏng chúng ta khởi tạo như sau. Vào Option for target 1.



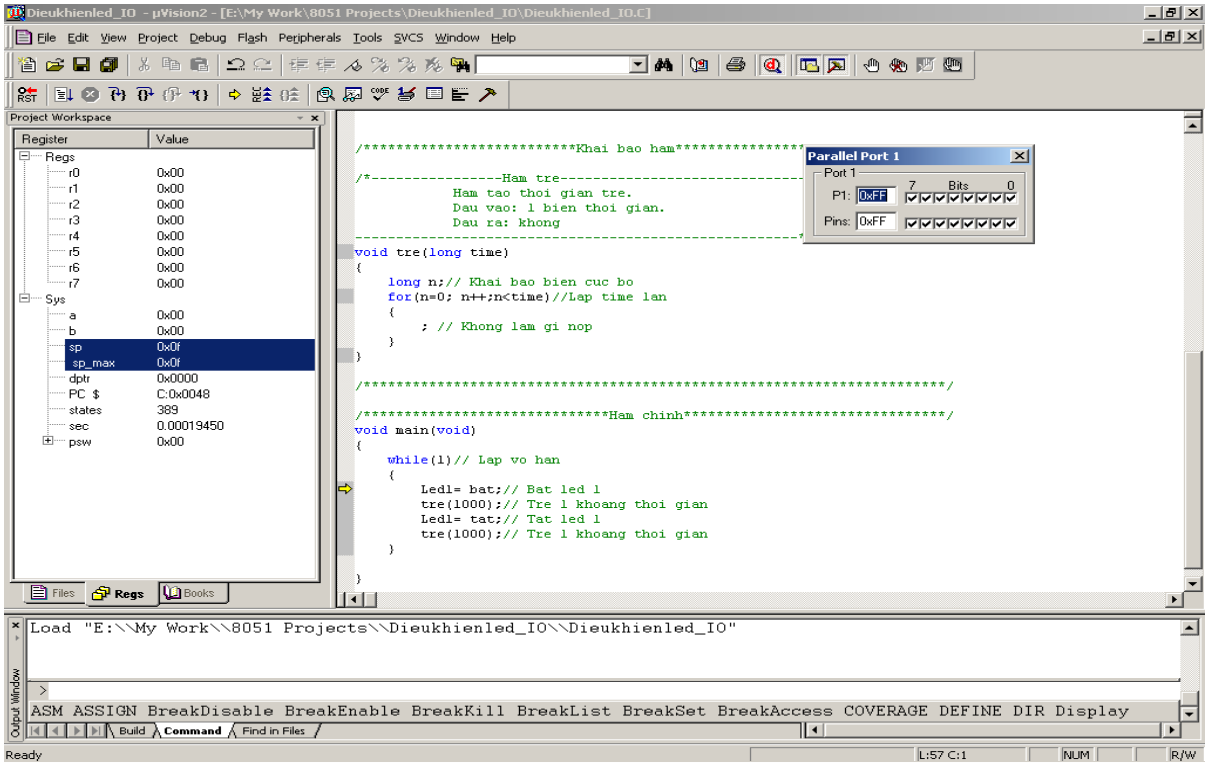
Được bảng sau. Nhập tần số thạch anh là 12 Mhz đúng với tần số thạch anh.



Chọn OK. Để mô phỏng nhấn tổ hợp phím Ctrl + F5. Hoặc nhấn vào biểu tượng có chữ D mầu đỏ trong menu view trên thanh công cụ. Được cửa sổ sau:



Trong menu Peripherals (các thiết bị ngoại vi) chọn IO port, Port 1. Được như sau:



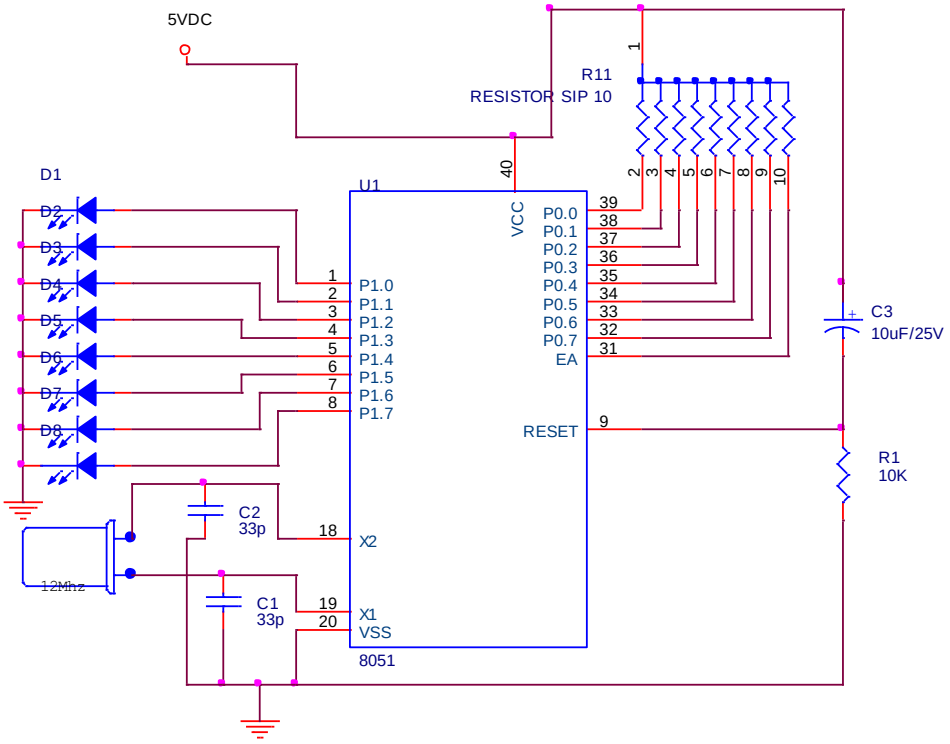
Xuất hiện 1 cửa sổ nhỏ Parallel Port 1 đó là cửa sổ mô phỏng của AT89C51. Dấu lựa chọn tương đương chân ở mức cao (5V), dấu không lựa chọn tương đương chân ở mức thấp (0V). Trong menu peripherals còn các ngoại vi khác như timer , interrupt, serial.

Để chạy chương trình click chuột phải vào màn hình soạn thảo. Rồi ấn F11. Mỗi lần ấn sẽ chạy 1 lệnh. Khi mô phỏng nếu hàm delay lâu quá 1000 lần lặp. Nhấn tổ hợp phím Ctrl + F11 để bỏ qua hàm. Hoặc nhấn F10 để chạy từng dòng lệnh.

Để thoát khỏi mô phỏng nhấn tổ hợp phím Ctrl+F5 hoặc nhấn vào biểu tượng mô phỏng.

2.4.2. Một số bài tập áp dụng.

- Xét mạch điều khiển led như hình vẽ



2.4.2.1. Ráp mạch

* Bước 1: Ráp mạch dao động:

* Bước 2: Ráp mạch reset.

* Bước 3: Ráp trở băng.

Nếu không có trở băng có thể thay trở băng 10 chân bằng 9 con trở thường vì trở băng 10 chân chính là 9 con trở đấu chung 1 đầu.

* Bước 4: Ráp led:

* Bước 5: Đấu 1 dây nhỏ từ chân 40 lên nguồn 5V.

2.4.2.2. Lập trình:

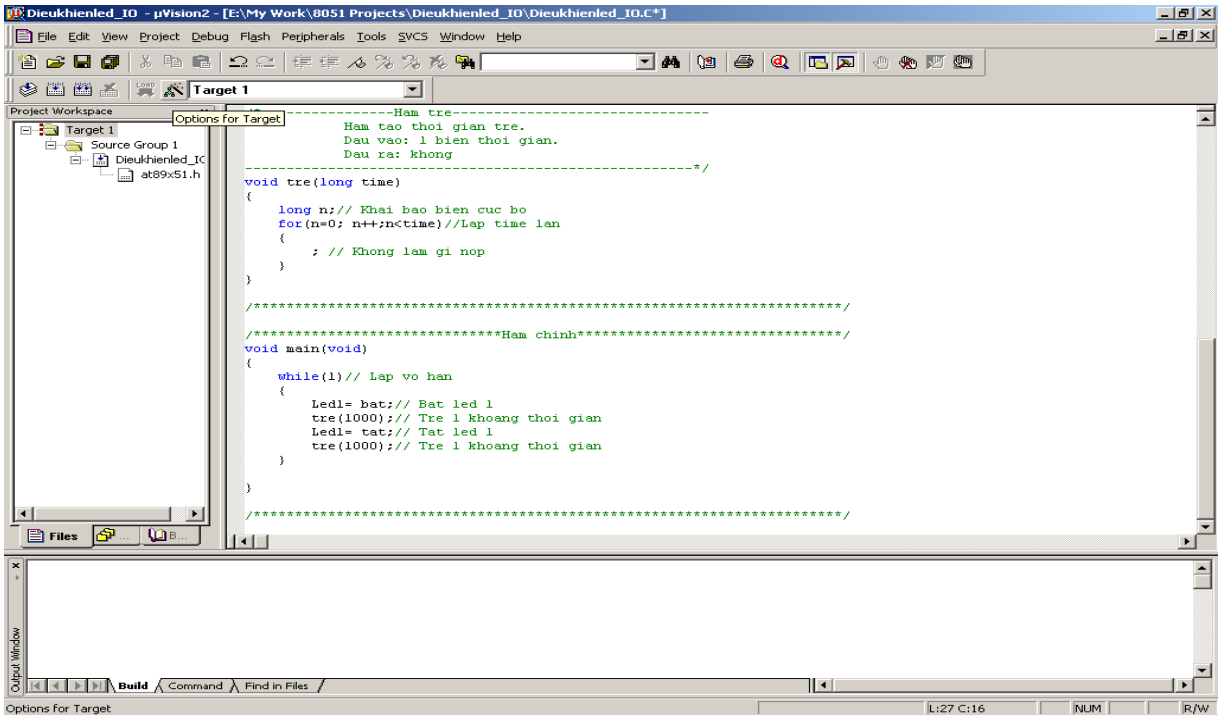
Code như sau:

```
/* ================================================
        Mo ta:
            Dieu khien den led.
        Phan cung:
            8 led noi tu +5V qua dien tro han dong vao 8 chan cong 1.
    Thach anh: 12 Mhz
================================================*/
```

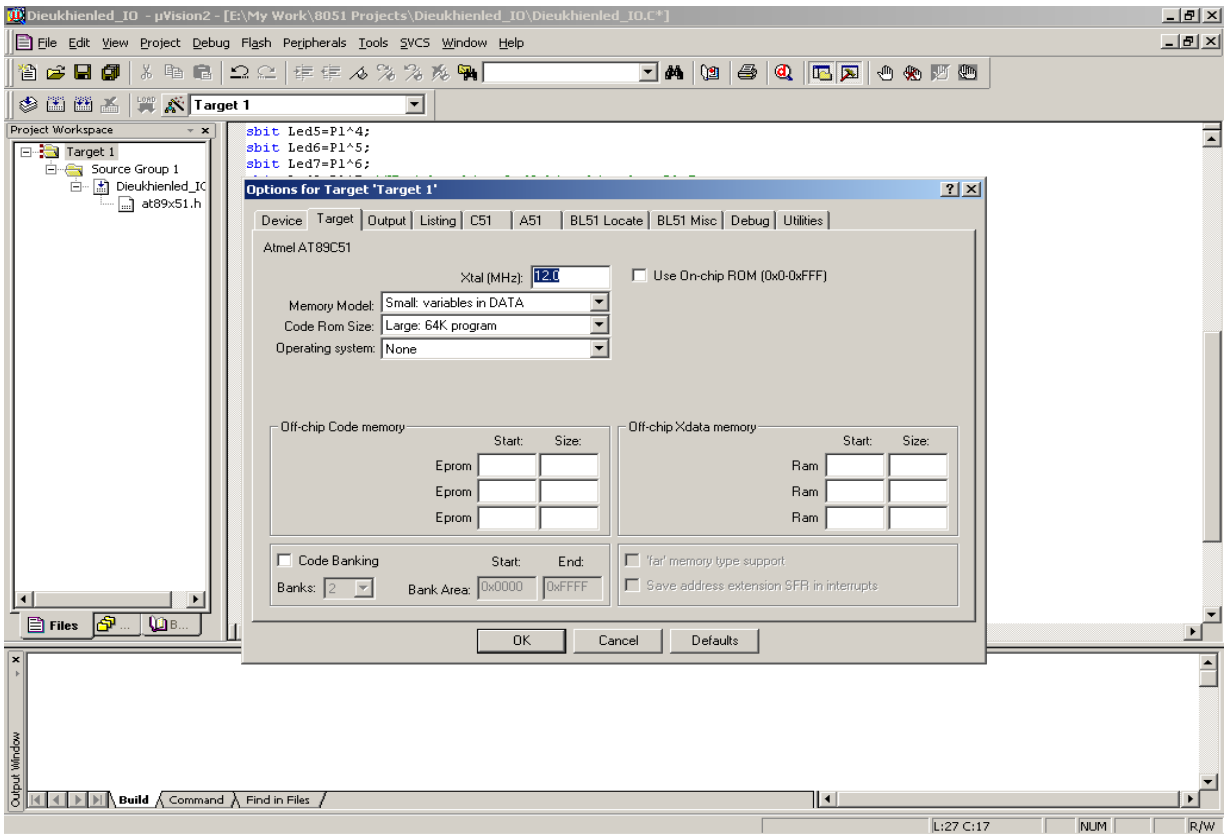
```
/*******************Bo tien xu li*********************************/
#include <AT89X51.H>// Dinh kem file thu vien
#define bat 1 // Dinh nghia gia tri bat den led
#define tat 0// Dinh nghia gia tri tat den led
/*************************************************************/
/*******************Khai bao bien toan cuc***********************/
sbit Led1=P1^0; //Khai bao bien Led1 kieu bit chan P1_0
sbit Led2=P1^1; // ...
sbit Led3=P1^2;
sbit Led4=P1^3;
sbit Led5=P1^4;
sbit Led6=P1^5;
sbit Led7=P1^6;
sbit Led8=P1^7;//Khai bao bien Led8 kieu bit chan P1_7
/*************************************************************/
/*******************Khai bao ham*****************************/
/*----------------Ham tre------------------------------
                        Ham tao thoi gian tre.
                        Dau vao: 1 bien thoi gian.
                        Dau ra:         khong
-------------------------------------------------------*/
void tre(long time)
{
        long n;// Khai bao bien cuc bo
        for(n=0; n<time; n++)//Lap time lan
        {
                ; // Khong lam gi nop
        }
}

/*************************************************************/
```

```
/*******************Ham chinh*************************/
void main(void)
{
	while(1)// Lap vo han
	{
		Led1= bat;// Bat led 1
		tre(1000);// Tre 1 khoang thoi gian
		Led1= tat;// Tat led 1
		tre(1000);// Tre 1 khoang thoi gian
	}
}
/************************************************************/
```

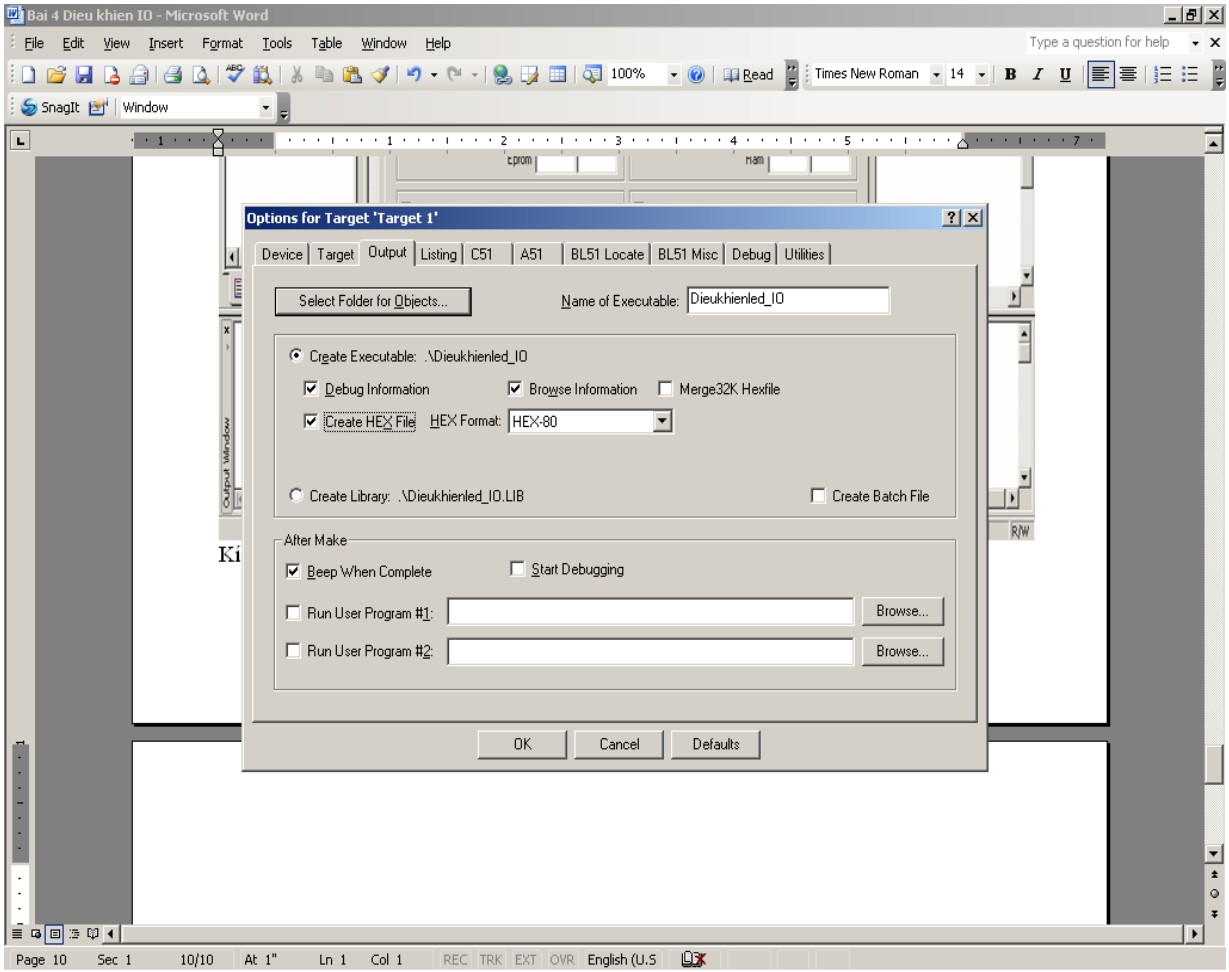
* Để có thể nạp chương trình vào chíp thì fải tạo ra file .hex. Để tạo ra file .hex làm như sau . Vào Option for target chỗ chỉnh tần số thạch anh.



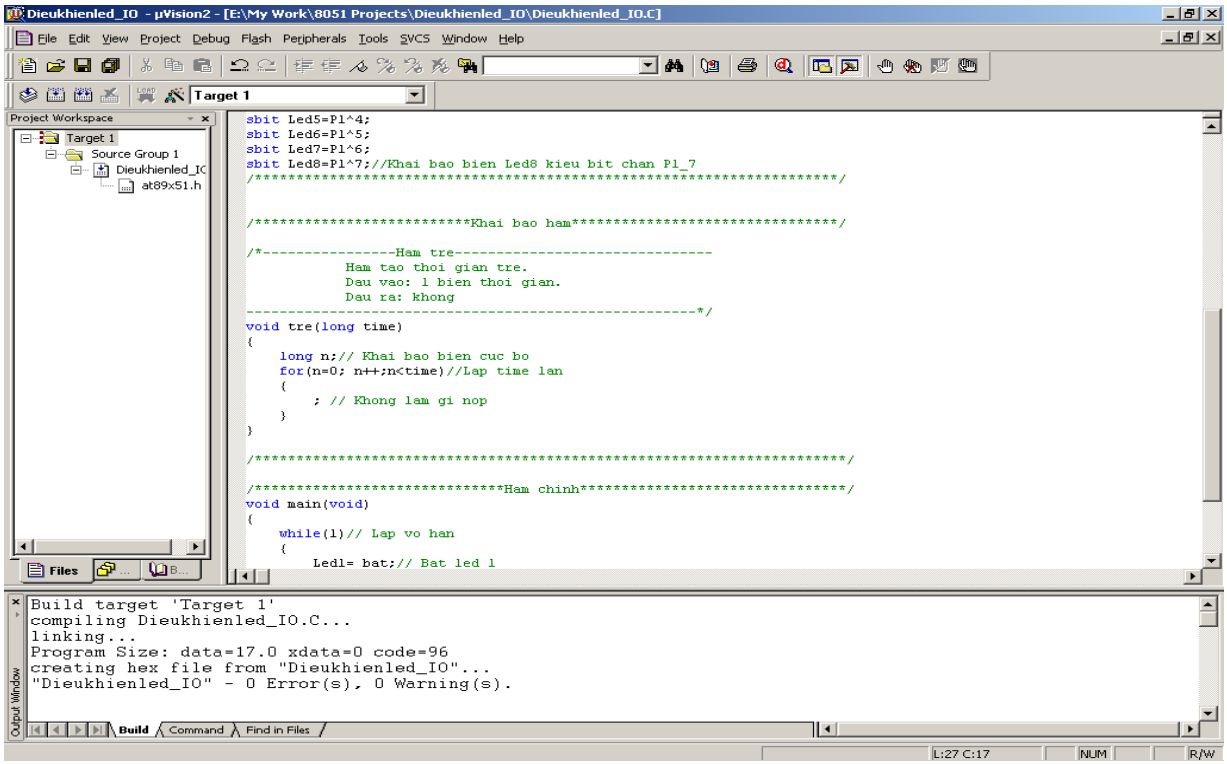
Được hình sau:



Kích vào tab Output. Được hình sau:



Lựa chọn vào check: Create Hex File. Nhấn OK. Nhấn fím F7 để biên dịch lại. Khi đó dưới cửa sổ output window được chữ Creating hex file…



2.4.3. Nạp chương trình vào vi điều khiển:

Nối đầu cổng COM vào cổng COM máy tính.

Nối nguồn vào mạch nạp.

Gắn vi điều khiển vào socket 40 chân.

Mở phần mềm EZDL4. Thấy có chữ identifing target chip …. Nháy. Trên EZDL4 sẽ thấy chữ AT89C51 hoặc AT89C52 tùy thuộc vào loại chíp đã lựa chọn.

Click vào “Send”. Chọn đường dẫn đến thư mục lưu project chọn file cần nạp. Nhấn OK. Chờ mạch báo chữ Complete thì lấy vi điều khiển ra gắn vào mạch. Chạy thử trên mạch:

Bài 3: Bộ định thời

❖ GIỚI THIỆU BÀI 3

Bài 3 là bài giới thiệu về bộ định thời trong họ vi điều khiển 89C51. Trên cơ sở đó người học vận dụng vào các bài tập cụ thể để soạn thảo được một chương trình điều khiển với những ứng dụng thực tiễn.

❖ MỤC TIÊU CỦA BÀI 3

- Trình bày cấu tạo và các chế độ làm việc của bộ định thời 89c51 theo nội dung đã học
- Thực hiện khởi tạo bộ nhớ đúng yêu cầu kỹ thuật
- Thực hiện đọc bộ định thời trong khi hoạt động đúng yêu cầu kỹ thuật
- Thực hiện lập trình điều khiển dùng bộ định thời đúng yêu cầu kỹ thuật
- Tích cực, chủ động và sáng tạo trong học tập.

❖ PHƯƠNG PHÁP GIẢNG DẠY VÀ HỌC TẬP BÀI 3

- *Đối với người dạy: sử dụng phương pháp giảng giảng dạy tích cực (diễn giảng, vấn đáp, dạy học theo vấn đề); yêu cầu người học thực hiện câu hỏi thảo luận của bài (cá nhân hoặc nhóm).*
- *Đối với người học: chủ động đọc giáo trình trước buổi học; hoàn thành đầy đủ câu hỏi thảo luận theo cá nhân hoặc nhóm và nộp lại cho người dạy đúng thời gian quy định.*

❖ ĐIỀU KIỆN THỰC HIỆN BÀI 3

- ***Phòng học chuyên môn hóa/nhà xưởng:*** học tại xưởng thực hành vi điều khiển – có trang bị các máy tính cài sẵn phần mềm mô phỏng.
- ***Trang thiết bị máy móc:*** Máy chiếu và các mô hình, thiết bị dạy học khác
- ***Học liệu, dụng cụ, nguyên vật liệu:*** Chương trình mô đun, giáo trình, tài liệu tham khảo, giáo án, phim ảnh, và các KIT thực hành vi điều khiển.
- ***Các điều kiện khác:*** Không có

❖ KIỂM TRA VÀ ĐÁNH GIÁ BÀI 3

- **Nội dung:**

✓ *Kiểm tra và đánh giá tất cả nội dung về kiến thức, kỹ năng đã nêu trong mục tiêu của bài*

✓ *Năng lực tự chủ và trách nhiệm: Trong quá trình học tập, người học cần:*

+ Nghiên cứu bài trước khi đến lớp

+ Chuẩn bị đầy đủ tài liệu học tập.

+ *Tham gia đầy đủ thời lượng môn học.*

+ *Nghiêm túc trong quá trình học tập.*

- **Phương pháp:**

✓ ***Điểm kiểm tra thường xuyên:*** *Không có*

✓ ***Kiểm tra định kỳ thực hành:*** *1 điểm kiểm tra (hình thức – Thực hành)*

NỘU DUNG BÀI 3

3.1. Giới thiệu về bộ định thời của vi điều khiển

3.1.1. Các biến được định nghĩa trong time.h

Dưới đây liệt kê một số kiểu biến được định nghĩa trong time.h:

Biến	**Mô tả**
size_t	Đây là kiểu nguyên không dấu và là kết quả của từ khóa **sizeof**
clock_t	Đây là một kiểu thích hợp để lưu trữ Processor time (thời gian của bộ vi xử lý).
time_t is	Đây là một kiểu thích hợp để lưu trữ Calendar time.
struct tm	Đây là một cấu trúc được sử dụng để giữ date và time.

Cấu trúc tm có định nghĩa như sau:

```
struct tm {
  int tm_sec;         /* biểu diễn giây, từ 0 tới 59  */
  int tm_min;         /* biểu diễn phút, từ 0 tới 59  */
  int tm_hour;        /* biểu diễn giờ, từ 0 tới 23   */
  int tm_mday;        /* biểu diễn ngày của tháng, từ 1 tới 31  */
  int tm_mon;         /* biểu diễn tháng, từ 0 tới 11      */
  int tm_year;        /* biểu diễn năm, bắt đầu từ 1900    */
  int tm_wday;        /* ngày trong tuần, từ 0 tới 6       */
  int tm_yday;        /* ngày trong năm, từ 0 tới 365      */
  int tm_isdst;       /* biểu diễn Daylight Saving Time    */
};
```

3.1.2. Các Macro được định nghĩa trong time.h

Bảng dưới liệt kê một số Macro được định nghĩa trong time.h:

- **NULL:** Macro này là giá trị của một hằng con trỏ null
- **CLOCKS_PER_SEC:** Macro này biểu diễn tốc độ đồng hồ mỗi giây (Processor Clock per Second).

3.2. Thanh ghi SFR của timer

Thanh ghi/ Bit	Ký hiệu	Chức năng
TMOD		Chọn model cho bộ định thời 1
7	GATE	Bít điều khiển cổng. Khi được set lên 1, bộ định thời chỉ hoạt động trong khi INT1 ở mức cao

6		C/T	Bít chọn chức năng đếm hoặc định thời:				
			1= đếm sự kiện				
			0= định thời trong một khoảng thời gian				
5		M1	Bit chọn chế độ thứ nhất				
4		M0	Bit chọn chế độ thứ 2				
			M1	M0	Chế độ	Chức năng	
			0	0	0	Chế độ định thời 13 bit	
			0	1	1	Chế độ định thời 16 bit	
			1	0	2	Chế độ tự động nạp lại 8 bit	
			1	1	3	Chế độ định thời chia xẻ	
3		GATE	Bit điều khiển cổng cho bộ định thời 0				
2		C/T	Bit chọn chức năng đếm / định thời cho bộ định thời 0				
1		M1	Bit chọn chế độ thứ nhất cho bộ định thời 0				
0		M0	Bit chọn chế độ thứ 2 cho bộ định thời 0				
TF1	TR1	TF1	TR0	IE1	IT1	IE0	IT0
Thanh ghi / Bit		Ký hiệu	Chức năng				
TCON			Điều khiển bộ đinh thời				
TCON.7		TF1	Cờ tràn của bộ định thời 1. Cờ này được set bởi phần cứng khi có tràn, được xoá bởi phần mềm, hoặc bởi phần cứng khi bộ vi xử lý trỏ đến trình phục vụ ngắt				
TCON.6		TR1	Bit điều khiển hoạt động của bộ định thời 1. Bit này được set hoặc xoá bởi phần mềm để điều khiển bộ định thời hoạt động hay ngưng				
TCON.5		TF0	Cừ tràn của bộ định thời 0				
TCON.4		TR0	Bit điều khiển hoạt động của bộ định thời 0				
TCON.3		IE1	Cừ ngắt bên ngoài 1 (kích khởi cạnh). Cờ này được set bởi phần cứng khi có cạnh âm (cuống) xuất hiện trên chân INT1, được xoá bởi phần mềm, hoặc phần cứng khi CPU trỏ đến trình phục vụ ngắt				
TCON.2		IT1	Cờ ngắt bên ngoài 1 (kích khởi cạnh hoặc mức). Cờ này được set hoặc xoá bởi phần mềm khi xảy ra cạnh âm hoặc mức thấp tại chân ngắt ngoài				
TCON.1		IE0	Cờ ngắt bên ngoài 0 (kích khởi cạnh)				
TCON.0		IT0	Cờ ngắt bên ngoài 0 (kích khởi cạnh hoặc mức)				
EA		ET2	ES	ET1	EX1	EX0	ET0
IE			Điều khiển các nguồn ngắt (0: không cho phép; 1: cho phép)				
IE.7		EA	Cho phép/ không cho phép toàn cục				
IE.6		---	Không sử dụng				
IE.5		ET2	Cho phép ngắt do bộ định thời 2				

IE.4	ES	Cho phép ngắt do port nối tiếp
IE.3	ET1	Cho phép ngắt cho bộ định thời 1
IE.2	EX1	Cho phép ngắt từ bên ngoài (ngắt ngoài 1)
IE.1	EX0	Cho phép ngắt từ bên ngoài (ngắt ngoài 0)
IE.0	ET0	Cho phép ngắt do bộ định thời 0

3.3. Các chế độ làm việc trong time.h

Sau đây là một số hàm được định nghĩa trong time.h:

STT	Hàm & Mô tả
1	Hàm char *asctime(const struct tm *timeptr) Trả về một con trỏ tới một chuỗi biểu diễn ngày và thời gian của cấu trúc timeptr
2	Hàm clock_t clock(void) Trả về tốc độ đồng hồ (processor clock) được sử dụng từ lúc bắt đầu một trình triển khai (thường là lúc bắt đầu chương trình)
3	Hàm char *ctime(const time_t *timer) Trả về một chuỗi biểu diễn localtime dựa trên tham số timer
4	Hàm double difftime(time_t time1, time_t time2) Trả về số giây khác nhau giữa time1 và time2 (tức là time1 – time2).
5	Hàm struct tm *gmtime(const time_t *timer) Giá trị của timer được chia thành cấu trúc tm và được biểu diễn trong UTC, hoặc GMT
6	Hàm struct tm *localtime(const time_t *timer) Giá trị của timer được chia thành cấu trúc tm và được biểu diễn trong Local Timezone
7	Hàm time_t mktime(struct tm *timeptr) Chuyển đổi cấu trúc được trỏ tới bởi timeptr vào trong một giá trị time_t theo Local Timezone
8	Hàm size_t strftime(char *str, size_t maxsize, const char *format, const struct tm *timeptr) Định dạng thời gian được biểu diễn trong cấu trúc timeptr theo các qui tắc định dạng được định nghĩa trong format và được lưu trữ vào trong str
9	Hàm time_t time(time_t *timer) Ước lượng Calendar time hiện tại và mã hóa nó vào trong định dạng time_t

3.4. Khởi tạo và truy xuất thanh ghi Timer

3.4.1. Hàm asctime() trong C

Hàm **char *asctime(const struct tm *timeptr)** Trả về một con trỏ tới một chuỗi biểu diễn ngày và thời gian của cấu trúc **struct timeptr**.

Khai báo hàm asctime() trong C:

Dưới đây là phần khai báo cho asctime() trong C:

```
char *asctime(const struct tm *timeptr)
```

Tham số:

Tham số **timeptr** là một con trỏ tới cấu trúc **tm** mà chứa một Calendar time được chia nhỏ thành các thành phần như sau:

```
struct tm {
  int tm_sec;         /* biểu diễn giây, từ 0 tới 59 */
  int tm_min;         /* biểu diễn phút, từ 0 tới 59 */
  int tm_hour;        /* biểu diễn giờ, từ 0 tới 23  */
  int tm_mday;        /* biểu diễn ngày của tháng, từ 1 tới 31 */
  int tm_mon;         /* biểu diễn tháng, từ 0 tới 11   */
  int tm_year;        /* biểu diễn năm, bắt đầu từ 1900 */
  int tm_wday;        /* ngày trong tuần, từ 0 tới 6    */
  int tm_yday;        /* ngày trong năm, từ 0 tới 365   */
  int tm_isdst;       /* biểu diễn Daylight Saving Time */
};
```

Trả về giá trị:

Hàm này trả về một chuỗi chứa thông tin date và time trong một định dạng con người có thể đọc **Www Mmm dd hh:mm:ss**: ở đây **Www** là ngày trong tuần, **Mmm** là các ký tự chỉ tháng, **dd** là ngày của tháng, **hh:mm:ss** là thời gian và **yyyy** là năm.

Ví dụ:

Chương trình C sau minh họa cách sử dụng của asctime() trong C:

```
#include <stdio.h>
#include <string.h>
#include <time.h>

int main()
{
```

```
  struct tm t;
  /* Quantrimang.com */
  t.tm_sec    = 15;
  t.tm_min    = 16;
  t.tm_hour   = 6;
  t.tm_mday   = 18;
  t.tm_mon    = 6;
  t.tm_year   = 118;
  t.tm_wday   = 5;

  puts(asctime(&t));
  return(0);
}
```

Biên dịch và chạy chương trình C trên sẽ cho kết quả:

3.4.1. Hàm clock() trong C

Hàm **clock_t clock(void)** Trả về số tích tắc đồng hồ đã trôi qua từ khi chương trình được chạy. Để lấy số giây được sử dụng bởi CPU, bạn sẽ cần chia cho CLOCKS_PER_SEC.

Trên hệ điều hành 32 bit thì CLOCKS_PER_SEC bằng 1000000, hàm này sẽ trả về cùng giá trị xấp xỉ mỗi 72 phút.

Khai báo hàm clock() trong C:

Dưới đây là phần khai báo cho clock() trong C:

```
clock_t clock(void)
```

Tham số:

- Hàm này không nhận tham số nào.

Trả về giá trị:

Hàm này trả về số tích tắc đồng hồ đã trôi qua từ khi chương trình được chạy. Nếu thất bại, hàm trả về -1.

Ví dụ:

Chương trình C sau minh họa cách sử dụng của clock() trong C:

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

```
int main()
{
  clock_t start_t, end_t, total_t;
  int i;
  start_t = clock();
  printf("Bat dau chuong trinh, start_t = %ld\n", start_t);

  printf("Quet qua mot vong lap lon, start_t = %ld\n", start_t);
  for(i=0; i< 10000000; i++)
  {
  }
  end_t = clock();
  printf("Ket thuc vong lap, end_t = %ld\n", end_t);

  total_t = (double)(end_t - start_t) / CLOCKS_PER_SEC;
  printf("Tong thoi gian su dung boi CPU: %f\n", total_t  );
  printf("Thoat chuong trinh...\n");

  return(0);
}
```

Chạy code trên ta nhận được kết quả như sau:

```
Bat dau chuong trinh, start_t = 884
Quet qua mot vong lap lon, start_t = 884
Ket thuc vong lap, end_t = 32819
Tong thoi gian su dung boi CPU: 0.000000
Thoat chuong trinh...
```

3.4.3. Hàm ctime() trong C

Hàm **char *ctime(const time_t *timer)** trả về một chuỗi biểu diễn localtime dựa trên tham số timer

Chuỗi trả về có định dạng sau: **Www Mmm dd hh:mm:ss** trong đó **Www** là ngày trong tuần, **Mmm** là các ký tự chỉ tháng, **dd** là ngày của tháng, **hh:mm:ss** là thời gian và **yyyy** là năm.

Khai báo hàm ctime() trong C:

Dưới đây là phần khai báo cho ctime() trong C:

```
char *ctime(const time_t *timer)
```

Tham số:

- **timer** -- Đây là con trỏ tới một đối tượng time_t mà chứa một Calendar time.

Trả về giá trị:

Hàm này trả về một chuỗi chứa thông tin date và time trong một định dạng con người có thể đọc.

Ví dụ:

Chương trình C sau minh họa cách sử dụng của ctime() trong C:

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

int main ()
{
  time_t curtime;

  time(&curtime);

  printf("Thời gian hiện tại = %s", ctime(&curtime));

  return(0);
}
```

Biên dịch và chạy chương trình C trên sẽ cho kết quả:

```
Thời gian hiện tại = Mon Oct 15 04:49:13 2018
```

3.4.4. Hàm difftime() trong C

Hàm **double difftime(time_t time1, time_t time2)** trả về số giây khác nhau giữa **time1**và **time2**, ví dụ như là **(time1 - time2)**. Hai time được xác định trong Calendar time, biểu diễn thời gian đã trôi qua từ Epoch (00:00:00 1/1/19700 theo UTC).

Khai báo hàm difftime() trong C:

Dưới đây là phần khai báo cho difftime() trong C:

```
double difftime(time_t time1, time_t time2)
```

Tham số:

- **time1** -- Đây là đối tượng time_t cho thời gian kết thúc.
- **time2** -- Đây là đối tượng time_t cho thời gian bắt đầu.

Trả về giá trị:

Hàm này trả về số giây khác nhau giữa hai thời gian (time2 – time1) dưới dạng một giá trị double.

Ví dụ:

Chương trình C sau minh họa cách sử dụng của difftime() trong C:

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

int main ()
{
  time_t start_t, end_t;
  double diff_t;

  printf("Bắt đầu chương trình...\n");
  time(&start_t);

  time(&end_t);
  diff_t = difftime(end_t, start_t);

  printf("Thời gian thực thi = %f\n", diff_t);
  printf("Thoát chương trình...\n");

  return(0);
}
```

Biên dịch và chạy chương trình C trên sẽ cho kết quả:

```
Bắt đầu chương trình...
Thời gian thực thi = 0.000000
Thoát chương trình...
```

3.4.5. Hàm gmtime() trong C

Hàm **struct tm *gmtime(const time_t *timer)** sử dụng giá trị được trỏ tới bởi timer để điền vào một cấu trúc tm với các giá trị mà biểu diễn thời gian tương ứng, được diễn đạt trong UTC hoặc GMT.

Khai báo hàm gmtime() trong C:

Dưới đây là phần khai báo cho gmtime() trong C:

```
struct tm *gmtime(const time_t *timer)
```

Tham số:

- **timeptr** -- Đây là con trỏ trỏ tới một giá trị time_t biểu diễn một Calendar time.

Trả về giá trị:

Hàm này trả về con trỏ tới cấu trúc tm với thông tin thời gian được điền vào trong. Dưới đây là chi tiết về cấu trúc timeptr.

```
struct tm {
  int tm_sec;         /* biểu diễn giây, từ 0 tới 59 */
  int tm_min;         /* biểu diễn phút, từ 0 tới 59 */
  int tm_hour;        /* biểu diễn giờ, từ 0 tới 23  */
  int tm_mday;        /* biểu diễn ngày của tháng, từ 1 tới 31 */
  int tm_mon;         /* biểu diễn tháng, từ 0 tới 11  */
  int tm_year;        /* biểu diễn năm, bắt đầu từ 1900 */
  int tm_wday;        /* ngày trong tuần, từ 0 tới 6 */
  int tm_yday;        /* ngày trong năm, từ 0 tới 365 */
  int tm_isdst;       /* biểu diễn Daylight Saving Time */
};
```

Ví dụ:

Chương trình C sau minh họa cách sử dụng của gmtime() trong C, bạn có thể tham khảo thêm danh sách múi giờ tại đây:

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

#define BST (+1)
#define CCT (+8)

int main ()
```

```
{

  time_t rawtime;
  struct tm *info;

  time(&rawtime);
  /* Get GMT time */
  info = gmtime(&rawtime );

  printf("Thời gian hiện tại:\n");
  printf("Tại London: %2d:%02d\n", (info->tm_hour+BST)%24, info->tm_min);
  printf("Tại Trung Quốc: %2d:%02d\n", (info->tm_hour+CCT)%24, info->tm_min);

  return(0);
}
```

Biên dịch và chạy chương trình C trên sẽ cho kết quả:

```
Thời gian hiện tại:
Tại London: 5:51
Tại Trung Quốc: 12:51
```

3.4.6. Hàm localtime() trong C

Hàm **struct tm *localtime(const time_t *timer)** sử dụng time được trỏ tới bởi timer để điền một cấu trúc tm với các giá trị mà biểu diễn Local time tương ứng. Giá trị của timer được chia vào trong cấu trúc tm và được diễn đạt trong Local Timezone.

Khai báo hàm localtime() trong C:

Dưới đây là phần khai báo cho localtime() trong C:

```
struct tm *localtime(const time_t *timer)
```

Tham số:

- **timer** -- Là con trỏ trỏ tới giá trị time_t biểu diễn một calendar time.

Trả về giá trị:

Hàm này trả về con trỏ tới cấu trúc tm với thông tin thời gian được điền vào trong. Dưới đây là chi tiết về cấu trúc timeptr.

```
struct tm {
  int tm_sec;         /* biểu diễn giây, từ 0 tới 59 */
```

```
  int tm_min;        /* biểu diễn phút, từ 0 tới 59 */
  int tm_hour;       /* biểu diễn giờ, từ 0 tới 23 */
  int tm_mday;       /* biểu diễn ngày của tháng, từ 1 tới 31 */
  int tm_mon;        /* biểu diễn tháng, từ 0 tới 11  */
  int tm_year;       /* biểu diễn năm, bắt đầu từ 1900 */
  int tm_wday;       /* ngày trong tuần, từ 0 tới 6 */
  int tm_yday;       /* ngày trong năm, từ 0 tới 365 */
  int tm_isdst;      /* biểu diễn Daylight Saving Time */
};
```

Ví dụ:

Chương trình C sau minh họa cách sử dụng của localtime() trong C:

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

int main ()
{
  time_t rawtime;
  struct tm *info;
  char buffer[80];

  time( &rawtime );

  info = localtime( &rawtime );
  printf("Local time và Local date hiện tại là: \n%s", asctime(info));

  return(0);
}
```

Biên dịch và chạy chương trình C trên sẽ cho kết quả:

```
Local time và Local date hiện tại là:
Mon Oct 15 11:56:31 2018
```

3.4.7. Hàm mktime() trong C

Hàm **time_t mktime(struct tm *timeptr)** chuyển đổi cấu trúc được trỏ tới bởi timeptr vào trong một giá trị time_t theo Local Timezone.

Khai báo hàm mktime() trong C:

Dưới đây là phần khai báo cho mktime() trong C:

```
time_t mktime(struct tm *timeptr)
```

Tham số:

- **timeptr** -- là con trỏ trỏ tới giá trị time_t biểu diễn một calendar time, mà được chia nhỏ thành các thành phần với cấu trúc:

```
struct tm {
  int tm_sec;         /* biểu diễn giây, từ 0 tới 59 */
  int tm_min;         /* biểu diễn phút, từ 0 tới 59  */
  int tm_hour;        /* biểu diễn giờ, từ 0 tới 23   */
  int tm_mday;        /* biểu diễn ngày của tháng, từ 1 tới 31 */
  int tm_mon;         /* biểu diễn tháng, từ 0 tới 11 */
  int tm_year;        /* biểu diễn năm, bắt đầu từ 1900 */
  int tm_wday;        /* ngày trong tuần, từ 0 tới 6  */
  int tm_yday;        /* ngày trong năm, từ 0 tới 365 */
  int tm_isdst;       /* biểu diễn Daylight Saving Time */
};
```

Trả về giá trị:

Hàm này trả về giá trị time_t tương ứng với tham số calendar time đã truyền. Nếu có lỗi, hàm này trả về giá trị -1.

Ví dụ:

Chương trình C sau minh họa cách sử dụng của mktime() trong C:

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

int main ()
{
  int ret;
  struct tm info;
```

```
  char buffer[80];

  info.tm_year = 2016 - 1900;
  info.tm_mon = 7 - 1;
  info.tm_mday = 4;
  info.tm_hour = 0;
  info.tm_min = 0;
  info.tm_sec = 1;
  info.tm_isdst = -1;

  ret = mktime(&info);
  if( ret == -1 )
  {
    printf("Error: không thể lấy time bằng cách sử dụng mktime\n");
  }
  else
  {
    strftime(buffer, sizeof(buffer), "%c", &info );
    printf(buffer);
  }

  return(0);
}
```

Biên dịch và chạy chương trình C trên sẽ cho kết quả:

```
Mon Jul 4 00:00:01 2016
```

3.4.8. Hàm strftime() trong C

Hàm **size_t strftime(char *str, size_t maxsize, const char *format, const struct tm *timeptr)** định dạng thời gian được biểu diễn trong cấu trúc timeptr theo các qui tắc định dạng được định nghĩa trong format và được lưu trữ vào trong str.

Khai báo hàm strftime() trong C:

Dưới đây là phần khai báo cho strftime() trong C:

```
size_t strftime(char *str, size_t maxsize, const char *format, const struct tm *timeptr)
```

Tham số:

- **str** -- Đây là con trỏ tới mảng đích, nơi mà chuỗi kết quả được sao chép.
- **maxsize** -- Đây là số ký tự tối đa để được sao chép tới str.
- **format** -- Đây là chuỗi chứa bất cứ tổ hợp nào của các ký tự thông thường và các format specifier đặc biệt. Những format specifier này được thay thế bởi hàm này bởi các giá trị tương ứng để biểu diễn thời gian được xác định trong tm. Các format specifier là:

Specifier	Thay thế cho	Ví dụ
%a	Tên ngày trong tuần viết tắt	Sun
%A	Tên ngày trong tuần đầy đủ	Sunday
%b	Tên tháng viết tắt	Mar
%B	Tên tháng đầy đủ	March
%c	Biểu diễn date và time	Sun Aug 19 02:56:02 2012
%d	Ngày trong tháng (01-31)	19
%H	Giờ, trong định dạng 24h (00-23)	14
%I	Giờ, trong định dạng 12h (01-12)	05
%j	Ngày trong năm (001-366)	231
%m	Tháng, dưới dạng biểu diễn số (01-12)	08
%M	Phút (00-59)	55
%p	AM hoặc PM	PM
%S	Giây (00-61)	02
%U	Số tuần, có ngày Chủ nhật đầu tiên là ngày đầu tiên của tuần (00-53)	33
%w	Ngày trong tuần dưới dạng biểu diễn số (0-6)	4
%W	Số tuần, có ngày thứ Hai đầu tiên là ngày đầu tiên của tuần (00-53)	34
%x	Biểu diễn date	08/19/12
%X	Biểu diễn time	02:50:06
%y	Năm, biểu diễn dưới dạng hai số cuối (00-99)	01
%Y	Năm	2012
%Z	Tên Timezone	CDT
%%	Ký hiệu %	%

- **timeptr** -- là con trỏ trỏ tới giá trị time_t biểu diễn một calendar time, mà được chia nhỏ thành các thành phần với cấu trúc:

```
struct tm {
   int tm_sec;         /* biểu diễn giây, từ 0 tới 59 */
   int tm_min;         /* biểu diễn phút, từ 0 tới 59 */
   int tm_hour;        /* biểu diễn giờ, từ 0 tới 23 */
   int tm_mday;        /* biểu diễn ngày của tháng, từ 1 tới 31 */
   int tm_mon;         /* biểu diễn tháng, từ 0 tới 11 */
   int tm_year;        /* biểu diễn năm, bắt đầu từ 1900 */
   int tm_wday;        /* ngày trong tuần, từ 0 tới 6 */
   int tm_yday;        /* ngày trong năm, từ 0 tới 365 */
   int tm_isdst;       /* biểu diễn Daylight Saving Time */
};
```

Trả về giá trị:

Nếu chuỗi kết quả có kích cỡ nhỏ hơn kích cỡ các ký tự (bao gồm ký tự null kết thúc), thì toàn bộ ký tự được sao chép tới str (không bao gồm ký tự null kết thúc) được trả về. Nếu không, hàm trả về 0.

Ví dụ:

Chương trình C sau minh họa cách sử dụng của strftime() trong C:

```
#include <stdio.h>
#include <time.h>
int main () {
   time_t rawtime;
   struct tm *info;
   char buffer[80];
   time( &rawtime );
   info = localtime( &rawtime );
   strftime(buffer,80,"%x - %I:%M%p", info);
   printf("Date & time đã định dạng theo hàm strftime là: \n|%s|\n", buffer );
   return(0);
}
```

Biên dịch và chạy chương trình C trên sẽ cho kết quả:

```
Date & time đã định dạng theo hàm strftime là:
|10/15/18 - 04:58AM|
```

3.4.9. Hàm time() trong C

Hàm **time_t time(time_t *seconds)** trả về thời gian từ Epoch (00:00:00 1/1/1970 theo UTC), được ước lượng bằng giây. Nếu tham số **seconds** không là NULL, thì giá trị trả về cũng được lưu trữ trong biến **seconds**.

Khai báo hàm time() trong C:

Dưới đây là phần khai báo cho time() trong C:

```
time_t time(time_t *t)
```

Tham số:

- **seconds** -- Đây là con trỏ tới một đối tượng của kiểu time_t, nơi giá trị seconds sẽ được lưu trữ.

Trả về giá trị:

Hàm trả về Calendar time hiện tại dưới dạng một đối tượng time_t.

Ví dụ:

Chương trình C sau minh họa cách sử dụng của time() trong C:

```
#include <stdio.h>
#include <time.h>
int main ()
{
  time_t seconds;
  seconds = time(NULL);
  printf("Số giờ (h) bắt đầu từ 1/1/1970 = %ld giờ\n", seconds/3600);
  return(0);
}
```

Biên dịch và chạy chương trình C trên sẽ cho kết quả:

```
Số giờ (h) bắt đầu từ 1/1/1970 = 427660 giờ
```

- string.h trong C
- stdlib.h trong C
- stddef.h trong C
- Trắc nghiệm về lập trình C P4
- Bộ câu hỏi trắc nghiệm về lập trình có giải P6
- stdio.h trong C

Bài 4: Cổng nối tiếp

❖ GIỚI THIỆU BÀI 4

Bài 4 là bài giới thiệu về cổng nối tiếp và cách thức truy xuất chúng trong họ vi điều khiển 89C51. Trên cơ sở đó người học vận dụng vào các bài tập cụ thể để soạn thảo được một chương trình điều khiển với những ứng dụng thực tiễn.

❖ MỤC TIÊU CỦA BÀI 4

- Trình bày cấu tạo và các chế độ làm việc của cổng truyền thông nối tiếp theo nội dung đã học
- Thực hiện cổng truyền thông nối tiếp đúng yêu cầu kỹ thuật.
- Thực hiện thu phát dữ liệu nối tiếp bằng 89c51 đạt yêu cầu kỹ thuật
- Tích cực, chủ động và sáng tạo trong học tập.

❖ PHƯƠNG PHÁP GIẢNG DẠY VÀ HỌC TẬP BÀI 4

- *Đối với người dạy: sử dụng phương pháp giảng giảng dạy tích cực (diễn giảng, vấn đáp, dạy học theo vấn đề); yêu cầu người học thực hiện câu hỏi thảo luận của bài (cá nhân hoặc nhóm).*
- *Đối với người học: chủ động đọc giáo trình trước buổi học; hoàn thành đầy đủ câu hỏi thảo luận theo cá nhân hoặc nhóm và nộp lại cho người dạy đúng thời gian quy định.*

❖ ĐIỀU KIỆN THỰC HIỆN BÀI 4

- ***Phòng học chuyên môn hóa/nhà xưởng:*** học tại xưởng thực hành vi điều khiển – có trang bị các máy tính cài sẵn phần mềm mô phỏng.
- ***Trang thiết bị máy móc:*** Máy chiếu và các mô hình, thiết bị dạy học khác
- ***Học liệu, dụng cụ, nguyên vật liệu:*** Chương trình mô đun, giáo trình, tài liệu tham khảo, giáo án, phim ảnh, và các KIT thực hành vi điều khiển.
- ***Các điều kiện khác:*** Không có

❖ KIỂM TRA VÀ ĐÁNH GIÁ BÀI 4

- **Nội dung:**

✓ *Kiểm tra và đánh giá tất cả nội dung về kiến thức, kỹ năng đã nêu trong mục tiêu của bài*

✓ *Năng lực tự chủ và trách nhiệm: Trong quá trình học tập, người học cần:*

+ Nghiên cứu bài trước khi đến lớp

+ Chuẩn bị đầy đủ tài liệu học tập.

+ Tham gia đầy đủ thời lượng môn học.

+ Nghiêm túc trong quá trình học tập.

- **Phương pháp:**

✓ ***Điểm kiểm tra thường xuyên:*** *Không có*

✓ ***Kiểm tra định kỳ thực hành:*** *1 điểm kiểm tra (hình thức – Thực hành)*

NỘU DUNG BÀI 4

4.1. Khái quát chung

Giao tiếp nối tiếp có nghĩa là truyền dữ liệu từng chút một tại một thời điểm, khi truyền thông song song, số lượng bit có thể được truyền tại một thời điểm phụ thuộc vào số lượng dòng dữ liệu có sẵn để liên lạc.

Hai phương thức giao tiếp nối tiếp là

- Truyền thông đồng bộ: Chuyển dữ liệu hàng loạt trong cấu trúc khung tại một thời điểm
- Truyền thông không đồng bộ: Truyền dữ liệu byte trong cấu trúc khung tại một thời điểm

8051 đã được xây dựng trong UART với RXD (nhận dữ liệu nối tiếp pin) và TXD (dữ liệu nối tiếp truyền pin) trên PORT3.0 và PORT3.1 tương ứng.

Giao tiếp không đồng bộ

Giao tiếp nối tiếp không đồng bộ được sử dụng rộng rãi để truyền byte theo định hướng.

4.1.2. Cấu trúc khung trong giao tiếp không đồng bộ:

- **START bit:** Đó là một chút mà bắt đầu giao tiếp nối tiếp và nó luôn luôn thấp.
- **Gói bit dữ liệu** : Các bit dữ liệu có thể là gói 5 đến 9 bit. Thông thường, chúng tôi sử dụng gói dữ liệu 8 bit, luôn được gửi sau bit START.
- **STOP bit** : Đây là một hoặc hai bit. Nó được gửi sau khi gói dữ liệu bit để cho biết kết thúc khung. Stop bit luôn logic cao.

Trong khung giao tiếp nối tiếp không đồng bộ, bit START đầu tiên được theo sau bởi byte dữ liệu và ở bit STOP cuối cùng, tạo thành một khung 10 bit. Đôi khi bit cuối cùng cũng được sử dụng như bit chẵn lẻ.

8051 Cấu trúc khung nối tiếp

Tốc độ truyền dữ liệu

4.1.2. Tiêu chuẩn giao diện

Tốc độ truyền dữ liệu được đo bằng bit trên giây (bps). Trong hệ nhị phân, nó cũng được gọi là tốc độ truyền (số lần thay đổi tín hiệu mỗi giây). Tốc độ truyền chuẩn được hỗ trợ là 1200, 2400, 4800, 19200, 38400, 57600 và 115200. Thông thường hầu hết thời gian 9600 bps được sử dụng khi tốc độ không phải là vấn đề lớn.

- 8051 giao tiếp nối tiếp có mức điện áp TTL là 0 v cho logic 0 và 5v cho logic 1.
- Trong máy tính và hầu hết các thiết bị cũ cho giao tiếp nối tiếp, giao thức RS232 với đầu nối DB9 được sử dụng. Giao tiếp nối tiếp RS232 có các mức điện áp khác nhau hơn 8051 giao tiếp nối tiếp. tức là +3 v đến +25 v cho logic zero và -3 v đến -25 v cho logic 1.

- Vì vậy, để giao tiếp với giao thức RS232, chúng ta cần phải sử dụng chuyển đổi mức điện áp như MAX232 IC.
- Mặc dù có 9 chân trong đầu nối DB9, chúng tôi không cần phải sử dụng tất cả các chân. Chỉ cần kết nối 2 Tx (Truyền), Rx thứ 3 (Nhận) và pin GND thứ 5.

4.2. Khởi tạo và truy xuất thanh ghi PORT nối tiếp

Lập trình UART 8051

4.2.1. Baud Rate tính toán:

- Để đáp ứng các tốc độ truyền chuẩn thường tinh thể với 11,0592 MHz được sử dụng.
- Như chúng ta đã biết, 8051 chia tần số pha lê cho 12 để có tần số chu trình máy là 921,6 kHz.
- Khối UART nội bộ 8051 chia tần số chu trình máy này là 32, cho tần số 28800 Hz được UART sử dụng.
- Để đạt được tốc độ truyền 9600, tần số 28800 Hz một lần nữa sẽ được chia cho 3.
- Điều này đạt được bằng cách sử dụng Timer1 ở chế độ-2 (chế độ tải lại tự động) bằng cách đặt 253 trong TH1 (8-bit reg.)
- Vì vậy, 28800 Hz sẽ được chia cho 3 như bộ đếm thời gian sẽ tràn sau mỗi 3 chu kỳ .
- chúng ta có thể đạt được tốc độ truyền khác nhau bằng cách đặt hệ số phân chia vào thanh ghi TH1.

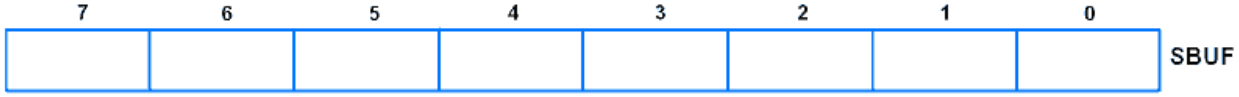
Hệ số phân chia để đạt được tốc độ truyền khác nhau

Tốc độ truyền	TH1 (Hex)
9600	FD
4800	FA
2400	F4
1200	E8

4.2.2. Đăng ký giao tiếp nối tiếp

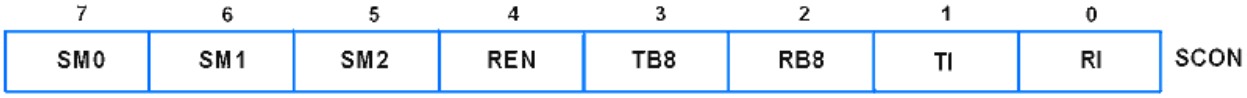
SBUF: Đăng ký bộ đệm nối tiếp

Đây là thanh ghi dữ liệu liên lạc nối tiếp được sử dụng để truyền hoặc nhận dữ liệu qua nó.



SCON: Đăng ký kiểm soát nối tiếp

Thanh ghi điều khiển nối tiếp SCON được sử dụng để cài đặt chế độ hoạt động truyền thông nối tiếp. Nó cũng được sử dụng để điều khiển các hoạt động truyền và nhận.



Bit 7: 6 - SM0: SM1 : Trình chỉ định chế độ nối tiếp

Chế độ	SM0	SM1	Chế độ
0	0	0	1/12 của chế độ đăng ký thay đổi tần số dao động cố định tốc độ truyền
1	0	1	UART 8 bit với bộ đếm thời gian 1 được xác định tốc độ truyền
2	1	0	UART 9 bit với 1/32 tốc độ truyền cố định của Osc
3	1	1	UART 9 bit với bộ đếm thời gian 1 được xác định tốc độ truyền

Thông thường chế độ-1 (SM0 = 0, SM1 = 1) được sử dụng với 8 bit dữ liệu, 1 bit bắt đầu và 1 bit dừng.

Bit 5 - SM2: cho Giao tiếp đa xử lý

Bit này cho phép tính năng liên lạc đa bộ xử lý ở chế độ 2 & 3.

Bit 4 - REN: Nhận Kích hoạt

1 = Nhận cho phép

0 = Nhận vô hiệu hóa

Bit 3 - TB8: Truyền bit thứ 9

Đây là bit thứ 9 được truyền ở chế độ 2 & 3 (chế độ 9 bit)

Bit 2 - RB8: Nhận bit thứ 9

Đây là lần thứ 9 nhận bit ở chế độ 2 & 3 (chế độ 9 bit), ở chế độ 1, nếu SM2 = 0 thì RB8 giữ bit dừng nhận được

Bit 1 - TI: Truyền cờ ngắt

Bit này cho biết quá trình truyền hoàn tất và được thiết lập sau khi truyền byte từ bộ đệm. Thông thường TI (Transmit Interrupt Flag) được đặt bằng phần cứng ở cuối bit 8 ở chế độ 0 và ở đầu bit dừng ở các chế độ khác.

Bit 0 - RI: Nhận cờ ngắt

Bit này cho biết việc tiếp nhận hoàn tất và được thiết lập sau khi nhận được byte hoàn chỉnh trong bộ đệm. Thông thường RI (Nhận cờ ngắt) được đặt bằng phần cứng ở chế độ nhận ở cuối bit 8 ở chế độ 0 và tại bit dừng nhận thời gian ở các chế độ khác.

4.2.3. Các bước lập trình

1. Định cấu hình hẹn giờ 1 ở chế độ tự động tải lại.
2. Tải TH1 với giá trị theo tốc độ truyền yêu cầu, ví dụ: đối với tải 9600 tốc độ truyền 0xFD. (-3 trong số thập phân)
3. Tải SCON với chế độ nối tiếp và bit điều khiển. ví dụ cho chế độ 1 và cho phép nhận, tải 0x50.
4. Bắt đầu timer1 bằng cách đặt bit TR1 thành 1.
5. Tải dữ liệu truyền trong thanh ghi SBUF.
6. Chờ cho đến khi dữ liệu được tải hoàn toàn được truyền đi bằng cách bỏ phiếu TI flag.
7. Khi cờ TI được thiết lập, hãy xóa nó và lặp lại từ bước 5 để truyền thêm dữ liệu.

4.3. Luyện tập

Bài tập 1:

Hãy chương trình 8051 (ở đây AT89C51) để gửi dữ liệu ký tự “kiểm tra” serially ở tốc độ 9600 baud ở chế độ 1

Chương trình truyền dữ liệu nối tiếp

```
#include <reg51.h>  /* Include x51 header file */
void UART_Init()
{
 TMOD = 0x20;  /* Timer 1, 8-bit auto reload mode */
 TH1 = 0xFD;  /* Load value for 9600 baud rate */
 SCON = 0x50;  /* Mode 1, reception enable */
 TR1 = 1;  /* Start timer 1 */
}
void Transmit_data(char tx_data)
{
 SBUF = tx_data;  /* Load char in SBUF register */
 while (TI==0);  /* Wait until stop bit transmit */
 TI = 0;   /* Clear TI flag */
}
```

```
void String(char *str)
{
 int i;
 for(i=0;str[i]!=0;i++) /* Send each char of string till the NULL */
 {
  Transmit_data(str[i]); /* Call transmit data function */
 }
}

void main()
{
 UART_Init();  /* UART initialize function */
 String("test");  /* Transmit 'test' */
 while(1);
}
```

Bài tập 2: Ngắt nối tiếp 8051

8051 UART có ngắt nối tiếp. Bất cứ khi nào dữ liệu được truyền hoặc nhận, cờ ngắt nối tiếp TI và RI được kích hoạt tương ứng.

Ngắt nối tiếp 8051 có địa chỉ vectơ (0023H), nơi nó có thể nhảy để phục vụ ISR (thường xuyên dịch vụ ngắt) nếu ngắt toàn cầu và nối tiếp được bật.

Chúng ta hãy xem cách thức làm gián đoạn nối tiếp sẽ được sử dụng trong lập trình truyền thông nối tiếp.

Các bước lập trình

1. Đặt hẹn giờ 1 ở chế độ tải lại tự động.
2. Tải TH1 với giá trị theo tốc độ truyền yêu cầu, ví dụ: đối với tải 9600 tốc độ truyền 0xFD.
3. Tải SCON với chế độ nối tiếp và bit điều khiển. ví dụ cho chế độ 1 và cho phép tải tiếp nhận 0x50.
4. Bắt đầu timer1 bằng cách đặt bit TR1 thành 1.
5. Bật bit ngắt toàn cục và nối tiếp, tức là EA = 1 và ES = 1.

6. Bây giờ bất cứ khi nào dữ liệu được nhận hoặc truyền, cờ ngắt sẽ được thiết lập và bộ điều khiển sẽ nhảy tới ISR nối tiếp.
7. Lưu ý rằng cờ TI / RI phải được xóa bằng phần mềm trong ISR.

Lưu ý: Đối với ngắt truyền và tiếp nhận, cùng một địa chỉ vectơ ngắt được gán, vì vậy khi bộ điều khiển nhảy tới ISR, chúng ta phải kiểm tra xem nó có bị ngắt Tx hay Rx gián đoạn bởi trạng thái bit TI và RI.

Bài tập áp dụng: Hãy chương trình 8051 (ở đây AT89C51) để nhận dữ liệu ký tự serially ở tốc độ 9600 baud ở chế độ 1 và gửi dữ liệu nhận được trên cổng 1 bằng cách sử dụng ngắt nối tiếp.

- Trong ví dụ này, sau khi nhận được byte, cờ RI được đặt sẽ tạo ra ngắt nối tiếp.
- Sau khi ngắt, 8051 sẽ nhảy tới thường trình ISR nối tiếp.
- Trong thói quen ISR, chúng tôi sẽ gửi dữ liệu đến cổng 1.

Chương trình gián đoạn nối tiếp

```
#include <reg51.h>  /* Include x51 header file */
void Ext_int_Init()
{ EA  = 1;  /* Enable global interrupt */
 ES = 1;    /* Enable serial interrupt */   }
void UART_Init()
{ TMOD = 0x20;  /* Timer 1, 8-bit auto reload mode */
 TH1 = 0xFD;  /* Load value for 9600 baud rate */
 SCON = 0x50;  /* Mode 1, reception enable */
 TR1 = 1;  /* Start timer 1 */
void Serial_ISR() interrupt 4
{ P1 = SBUF;  /* Give received data on port 1 */
 RI = 0;   /* Clear RI flag */}

void main()
{ P1 = 0x00;  /* Make P1 output */
 Ext_int_Init();   /* Call Ext. interrupt initialize */
 UART_Init();
 while(1);
}
```

Bài 5: Tổ chức ngắt

❖ GIỚI THIỆU BÀI 5

Bài 5 là bài giới thiệu về tổ chức ngắt trong họ vi điều khiển 89C51. Hướng dẫn về tầm quan trọn của tổ chức ngắt trong điều khiển hệ thống cơ điện tử và phương thức truy xuất tổ chức ngắt. Trên cơ sở đó người học vận dụng vào các bài tập cụ thể để soạn thảo được một chương trình điều khiển với những ứng dụng thực tiễn.

❖ MỤC TIÊU CỦA BÀI 5

- Trình bày tác dung thực tế của một hệ thống được điều khiển bằng tín hiệu ngắt theo nội dung đã học
- Thực hiện tổ chức ngắt và cơ chế thực hiện chương trình phục vụ ngắt của 89c51 đúng yêu cầu kỹ thuật.
- Thực hiện tổ chức ngắt đạt yêu cầu kỹ thuật
- Tích cực, chủ động và sáng tạo trong học tập.

❖ PHƯƠNG PHÁP GIẢNG DẠY VÀ HỌC TẬP BÀI 5

- *Đối với người dạy: sử dụng phương pháp giảng giảng dạy tích cực (diễn giảng, vấn đáp, dạy học theo vấn đề); yêu cầu người học thực hiện câu hỏi thảo luận của bài (cá nhân hoặc nhóm).*
- *Đối với người học: chủ động đọc giáo trình trước buổi học; hoàn thành đầy đủ câu hỏi thảo luận theo cá nhân hoặc nhóm và nộp lại cho người dạy đúng thời gian quy định.*

❖ ĐIỀU KIỆN THỰC HIỆN BÀI 5

- ***Phòng học chuyên môn hóa/nhà xưởng:*** học tại xưởng thực hành vi điều khiển – có trang bị các máy tính cài sẵn phần mềm mô phỏng.
- ***Trang thiết bị máy móc:*** Máy chiếu và các mô hình, thiết bị dạy học khác
- ***Học liệu, dụng cụ, nguyên vật liệu:*** Chương trình mô đun, giáo trình, tài liệu tham khảo, giáo án, phim ảnh, và các KIT thực hành vi điều khiển.
- ***Các điều kiện khác:*** Không có

❖ KIỂM TRA VÀ ĐÁNH GIÁ BÀI 5

- **Nội dung:**

✓ *Kiểm tra và đánh giá tất cả nội dung về kiến thức, kỹ năng đã nêu trong mục tiêu của bài*

✓ *Năng lực tự chủ và trách nhiệm: Trong quá trình học tập, người học cần:*

+ Nghiên cứu bài trước khi đến lớp

+ Chuẩn bị đầy đủ tài liệu học tập.

+ *Tham gia đầy đủ thời lượng môn học.*

+ *Nghiêm túc trong quá trình học tập.*

- **Phương pháp:**

✓ ***Điểm kiểm tra thường xuyên:*** *Không có*

✓ ***Kiểm tra định kỳ thực hành:*** *1 điểm kiểm tra (hình thức – Thực hành)*

NỘU DUNG BÀI 5

5.1. Tổ chức ngắt

Theo các nhà sản xuất thí vi điều khiển 8051 có 6 ngắt được phân bố như sau:
- RESET: Khi chân RESET được kích hoạt từ 8051, bộ đếm chương trình nhảy về địa chỉ 0000H.
- 2 ngắt dành cho các bộ định thời: 1 cho Timer0 và 1 cho Timer1. Địa chỉ tương ứng của các ngắt này là 000BH và 001BH.
- 2 ngắt dành cho các ngắt phần cứng bên ngoài: chân 12 (P3.2) và 13 (P3.3) của cổng P3 là các ngắt phần cứng bên ngoài INT0 và INT1 tương ứng. Địa chỉ tương ứng của các ngắt ngoài này là 0003H và 0013H.
- Truyền thông nối tiếp: có 1 ngắt chung cho cả nhận và truyền dữ liệu nối tiếp. Địa chỉ của ngắt này trong bảng vector ngắt là 0023H.

Cỏc nguồn ngắt:

Ngắt do	**Cờ**	**Địa chỉ vector**
Reset hệ thống	**RST**	**0000H**
Ngắt ngoài 0	**IE0**	**0003H**
Bộ định thời 0	**TF0**	**000BH**
Ngắt ngoài 1	**IE1**	**0013H**
Bộ định thời 1	**TF1**	**001BH**
Port nối tiếp	**RI hoặc TI**	**0023H**
Bộ định thời 2	**TF2 hoặc EXF2**	**002BH**

Một chương trình chính không có ngắt thì chạy liên tục, còn chương trình có ngắt thì cứ khi nào điều kiện ngắt được đảm bào thì con trỏ sẽ nhảy sang hàm ngắt thực hiện xong hàm ngắt lại quay về đúng chỗ cũ thực hiện tiếp chương trình chính. Ngắt dùng cho mục đích đa nhiệm.

5.2. Xử lý ngắt.

5.2.1. Quy trình khi thực hiện một ngắt

Khi kích hoạt một ngắt bộ vi điều khiển thực hiện các bước sau:

- Vi điều khiển sẽ hoàn thành nốt lệnh đang thực hiện và lưu địa chỉ của lệnh kế tiếp vào ngăn xếp.
- Nó cũng lưu tình trạng hiện tại của tất cả các ngắt.
- Nó nhảy đến một vị trí cố định trong bộ nhớ được gọi là bảng vector ngắt, nơi lưu giữ địa chỉ của một trình phục vụ ngắt.
- Bộ vi điều khiển nhận địa chỉ ISR từ bảng vector ngắt và nhảy tới đó. Nó bắt đầu thực hiện trình phục vụ ngắt cho đến lệnh cuối cùng của ISR và trở về chương trình chính từ ngắt.

5.2.2. Các bước cho phép và cấm ngắt

Khi bật lại nguồn thì tất cả mọi ngắt đều bị cấm (bị che), có nghĩa là không có ngắt nào được bộ vi điều khiển đáp ứng trừ khi chúng được kích hoạt.

Các ngắt phải được kích hoạt bằng phần mềm để bộ vi điều khiển đáp ứng chúng. Có một thanh ghi được gọi là thanh ghi cho phép ngắt IE (Interrupt Enable) – ở địa chỉ A8H chịu trách nhiệm về việc cho phép và cấm các ngắt.

Bảng sau trình bày chi tiết về thanh ghi IE

Bit	Tên	Địa chỉ	Chức năng
7	**EA**	AFh	Cho phép/cấm hoạt động của cả thanh ghi
6	-	AEh	Chưa sử dụng
5	-	ADh	Chưa sử dụng
4	**ES**	ACh	Cho phép ngắt cổng truyền thông nối tiếp
3	**ET1**	ABh	Cho phép ngắt Timer 1
2	**EX1**	AAh	Cho phép ngắt ngoài 1
1	**ET0**	A9h	Cho phép ngắt Timer 0
0	**EX0**	A8h	Cho phép ngắt ngoài 0

Để cho phép một ngắt ta phải thực hiện các bước sau:

- Nếu EA = 0 thì không có ngắt nào được đáp ứng cho dù bit tương ứng của nó trong IE có giá trị cao. Bit D7 - EA của thanh ghi IE phải được bật lên cao để cho phép các bit còn lại của thanh ghi hoạt động được.

- Nếu EA = 1 thì tất cả mọi ngắt đều được phép và sẽ được đáp ứng nếu các bit tương ứng của chúng trong IE có mức cao.

Ví dụ 1: Hãy lập trình cho 8051:

a) cho phép ngắt nối tiếp, ngắt Timer0 và ngắt phần cứng ngoài 1 (EX1)

b) cấm ngắt Timer0

c) sau đó trình bày cách cấm tất cả mọi ngắt chỉ bằng một lệnh duy nhất.

Lời giải:

```
a) Mov IE, #96h;     //1001 0110: lệnh này tương đương với 4 lệnh phía dưới
Hoặc
        SetB EA;     //Cho phép sử dụng ngắt
        SetB ES;     //Cho phép ngắt cổng nối tiếp
         SetB ET0;   //Cho phép ngắt timer0
        SetB EX1;    //Cho phép ngắt ngoài 1
b) Mov ET0,#00h;          //Cấm ngắt timer0
c) Mov EA,#00h;           //Cấm tất cả các ngắt
```

5.3. Thiết kế chương trình dùng ngắt

Trong các chương trước ta đã biết cách sử dụng các bộ định thời Timer0 và Timer1 bằng phương pháp thăm dò. Trong phần này ta sẽ sử dụng các ngắt để lập trình cho các bộ định thời của 8051.

5.3.1 Cờ quay về 0 của bộ định thời và ngắt

Chúng ta đã biết rằng cờ bộ định thời TF được bật lên cao khi bộ định thời đạt giá trị cực đại và quay về 0 (Roll - over). Trong các phần trước, chúng ta cũng chỉ ra cách kiểm tra cờ TF bằng một vòng lặp. Trong khi thăm dò cờ TF thì ta phải đợi cho đến khi cờ TF được bật lên. Vấn đề với phương pháp này là bộ vi điều khiển bị trói buộc trong khi chờ cờ TF được bật và không thể làm được bất kỳ việc gì khác.

Sử dụng các ngắt sẽ giải quyết được vấn đề này và tránh được sự trói buộc bộ vi điều khiển. Nếu bộ ngắt định thời trong thanh ghi IE được phép thì mỗi khi nó quay trở về 0 bộ vi điều khiển sẽ bị ngắt, bất chấp nó đang thực hiện việc gì và nhảy tới bảng vector ngắt để phục vụ ISR. Bằng cách này thì bộ vi điều khiển có thể làm những công việc khác cho đến khi nó được thông báo rằng bộ định thời đã quay về 0. Xem hình 3 và ví dụ 1.

5.3.2. Các bước lập trình ngắt định thời

Để hiểu trình tự lập trình ngắt định thời, ta xem các ví dụ sau:

Ví dụ 1: Hãy viết chương trình nhận liên tục dữ liệu 8 Bit ở cổng P0 và gửi nó đến cổng P1 trong khi nó cùng lúc tạo ra một sóng vuông chu kỳ 200ms trên chân P2.1.

Hãy sử dụng bộ Timer0 để tạo ra sóng vuông, tần số của 8051 là XTAL = 11.0592MHz.

Lời giải:

Bước 1: tìm giá trị cần nạp cho thanh ghi timer và xác định chế độ hoạt động của timer

Theo yêu cầu Chu kỳ 200ms, vậy nửa chu kỳ là 100ms. Ta có: 100ms/1,085ms=92. (Đối với mạch hoạt động với tần số thạch anh 11.0592 Mhz) Suy ra giá trị cần nạp cho timer0 là: -92 <=> A4H. Ta sử dụng timer0 chế độ 2, 8 bit tự nạp lại.

Chương trình viết bằng Assembly

```
Org 0000h Ljmp Main ;     Nhảy đến chương trình chính, tránh dùng không gian bộ
                          nhớ ; của ngắt
Org 000Bh ;               Bảng vectơ ngắt của timer0
Cpl P2.1 ;                Đảo mức cho chân P2.1
Reti ;                    Kết thúc chương trình con, về chương trình chính
Org 0030h Main:
Mov TMOD,#02h ;           Chọn timer 0, chế độ 2 – tự nạp lai
Mov TH0,#-92 ;            Nạp giá trị cho thanh ghi TH0
Mov TL0,#-92 ;            Nạp giá trị ban đầu cho thanh ghi TL0
Mov P0,#0FFh ;            Thiết lập chân P0 làm cổng vào
Mov IE,#82h ; IE = 1000.0010 –  cho phép ngắt toàn cục và timer 0
SetB TR0 ;                Khởi động timer 0 Back:
Mov A,P0 ; Lấy giá trị cổng vào P0
Mov P1,A ; Xuất giá trị ra lại cổng 1
Sjmp Back ; Lặp lại quá trình
End ; Không cần xóa cờ TF0, 8051 tự động xóa.
```

Lưu ý: Các xung được tạo ra ở các ví dụ trên không thật sự chính xác, vì chưa tính đến hao phí của các lệnh. Hãy để ý những điểm dưới đây của chương trình trong ví dụ 1:

1. Trình ứng dụng không được sử dụng vùng không gian bộ nhớ của ngắt, do đó có lệnh Ljmp Main
2. Trình phục ngắt có không gian bộ nhớ nhỏ
3. Để cho phép ngắt chúng ta phải thực hiện lệnh cho phép ngắt bộ Timer0 với lệnh Mov IE, #82h; trong chương trình chính main.
4. Trong khi dữ liệu ở cổng P0 được nhận vào và chuyển liên tục sang cổng P1 thì mỗi khi bộ Timer0 trở về 0, cờ TF0 được bật lên và bộ vi điều khiển thoát ra khỏi vòng lặp BACK và nhảy đến địa chỉ 000BH để thực hiện trình phục vụ ISR của bộ Timer0.
5. Trong trình phục vụ ngắt ISR của Timer0 ta thấy rằng không cần đến lệnh xóa cờ TF0 của timer0 trước lệnh RETI. Lý do này là vì 8051 đã tự xoá cờ TF0 ngay khi nhảy đến ISR.

5.4. Luyện tập

Bài tập 1: Ngắt theo mức

Giả sử chân INT1 được nối đến công tắc bình thường ở mức cao. Mỗi khi nó ấn xuống thấp phải bật một đèn LED ở chân P1.3 (bình thường Led tắt), khi nó được bật lên nó phải sáng vài giây. Chừng nào công tắc được giữ ở trạng thái thấp đèn LED phải sáng liên tục

Bài tập 2: Ngắt theo sườn

Ngắt theo sườn là ngắt sẽ xảy ra khi có một sườn âm xuất hiện trên các chân ngắt của vi điều khiển. Điều này làm cho ngắt theo sườn khắc phục được nhược điểm của ngắt theo mức như ta đã thấy ở trên.

Để kích hoạt chế độ ngắt theo sườn thì chúng ta phải viết chương trình cài đặt cho các bit của thanh ghi TCON:

Bài 6: Phần mềm mô phỏng – Lập trình tổng hợp

❖ GIỚI THIỆU BÀI 6

Bài 6 là bài hướng dẫn sử dụng phần mềm mô phỏng các mạch đã được lập trình trước khi giao tiếp với vi điều khiển, từ đó có phương án xử lý lỗi trong quá trình thực hiện mạch điều khiển. Trên cơ sở đó người học vận dụng vào các bài tập cụ thể để soạn thảo được một chương trình điều khiển với những ứng dụng thực tiễn.

❖ MỤC TIÊU CỦA BÀI 6

- Trình bày được cấu trúc chung của chương trình hợp ngữ theo nội dung đã học.
- Thực hiện viêt chương trình tổ chức lớn bằng cách phân chia thành các mô đun chương trình đúng qui trình kỹ thuật.
- Viết được chương trình điều khiển theo yêu cầu
- Dùng Protues mô phỏng được hoạt động của mạch vi điều khiển chạy đúng theo yêu cầu.
- Tích cực, chủ động và sáng tạo trong học tập.

❖ PHƯƠNG PHÁP GIẢNG DẠY VÀ HỌC TẬP BÀI 6

- *Đối với người dạy: sử dụng phương pháp giảng giảng dạy tích cực (diễn giảng, vấn đáp, dạy học theo vấn đề); yêu cầu người học thực hiện câu hỏi thảo luận của bài (cá nhân hoặc nhóm).*
- *Đối với người học: chủ động đọc giáo trình trước buổi học; hoàn thành đầy đủ câu hỏi thảo luận theo cá nhân hoặc nhóm và nộp lại cho người dạy đúng thời gian quy định.*

❖ ĐIỀU KIỆN THỰC HIỆN BÀI 6

- ***Phòng học chuyên môn hóa/nhà xưởng:*** học tại xưởng thực hành vi điều khiển – có trang bị các máy tính cài sẵn phần mềm mô phỏng.
- ***Trang thiết bị máy móc:*** Máy chiếu và các mô hình, thiết bị dạy học khác
- ***Học liệu, dụng cụ, nguyên vật liệu:*** Chương trình mô đun, giáo trình, tài liệu tham khảo, giáo án, phim ảnh, và các KIT thực hành vi điều khiển.
- ***Các điều kiện khác:*** Không có

❖ KIỂM TRA VÀ ĐÁNH GIÁ BÀI 6

- **Nội dung:**

✓ *Kiểm tra và đánh giá tất cả nội dung về kiến thức, kỹ năng đã nêu trong mục tiêu của bài*

✓ *Năng lực tự chủ và trách nhiệm: Trong quá trình học tập, người học cần:*

+ *Nghiên cứu bài trước khi đến lớp*

+ *Chuẩn bị đầy đủ tài liệu học tập.*

+ *Tham gia đầy đủ thời lượng môn học.*

+ *Nghiêm túc trong quá trình học tập.*

- **Phương pháp:**

✓ ***Điểm kiểm tra thường xuyên:*** *1 điểm kiểm tra (hình thức – tự luận)*

✓ ***Kiểm tra định kỳ thực hành:*** *1 điểm kiểm tra (hình thức – Thực hành)*

NỘU DUNG BÀI 6

6.1 Phần mềm mô phỏng Proteus

6.1.1. Giới thiệu phần mềm Proteus

Proteus là phần mềm mô phỏng vật lý các mạch điện tử, hay gọi là giả lập linh kiện trên máy tính, giúp chúng ta có thể dễ dàng thao tác và xử lý trực tiếp mà không cần phải nối dây hoặc cần các dụng cụ chuyên dụng để thực hành. Phần mềm gồm 2 chương trình chính:

- ISIS cho phép vẽ sơ đồ nguyên lý và mô phỏng mạch
- ARES dùng để vẽ mạch in.

6.1.2. Hướng dẫn sử dụng Proteus để vẽ sơ đồ nguyên lý (Schematic)

Bước 1: Khởi động chương trình Proteus Professional

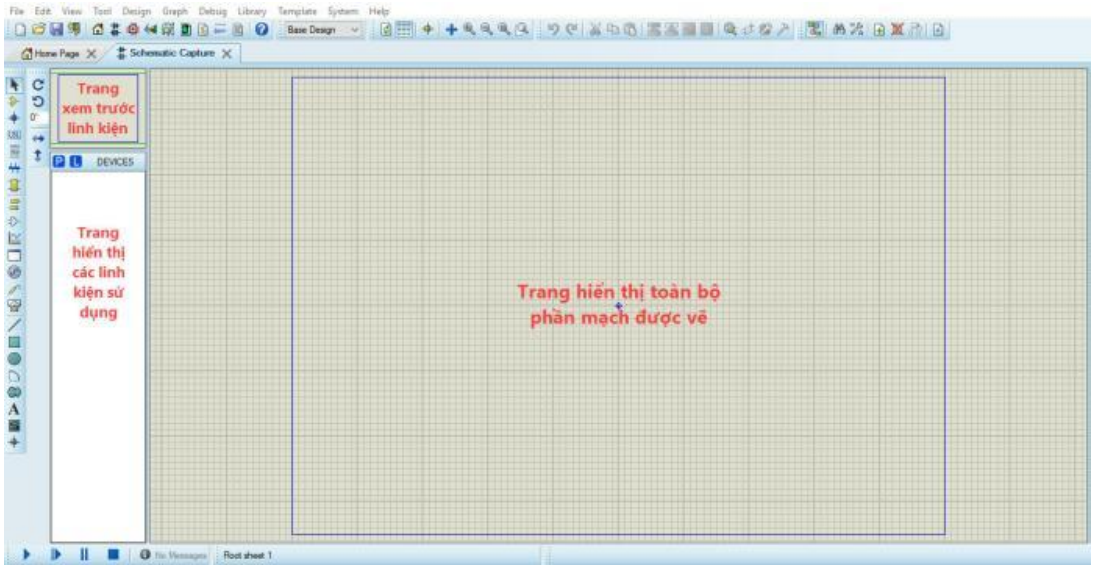
Chạy chương trình Proteus Professional bằng cách nhấp vào biểu tượng ISIS Professional trên desktop hoặc chọn Windows >> Programs >> Proteus Professional >> ISIS Professional.

Sau khi phần mềm khởi động xong thì bạn sẽ thấy phần giao diện của nó như sau:

Bước 2: Mở chương trình ISIS Professional

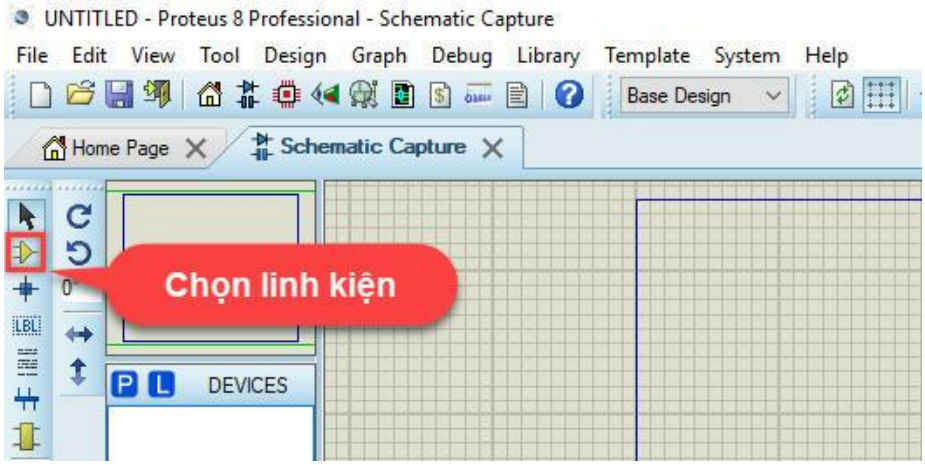
Nhấp vào biểu tượng Schematic Capture trên thanh công cụ của giao diện Proteus để mở chương trình con ISIS Professional.

Sau khi chương trình ISIS được mở ra, một vùng làm việc với các nút giao diện để thiết kế mạch sẽ xuất hiện như hình bên dưới. Các bạn lưu ý trên vùng làm việc của ISIS có một khung vuông màu xanh, khi vẽ mạch thì bạn phải đảm bảo toàn bộ phần mạch bạn vẽ phải nằm trong khung vuông này.

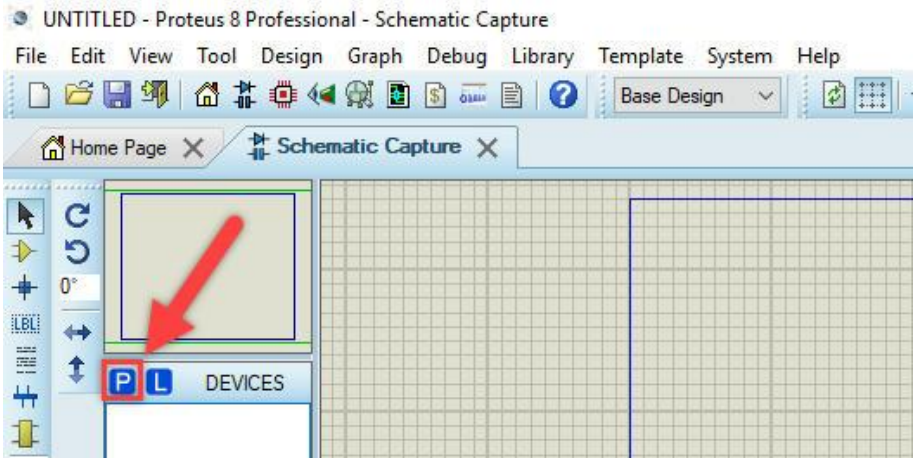


Bước 3: Lấy tất cả các linh kiện sử dụng từ thư viện của Proteus

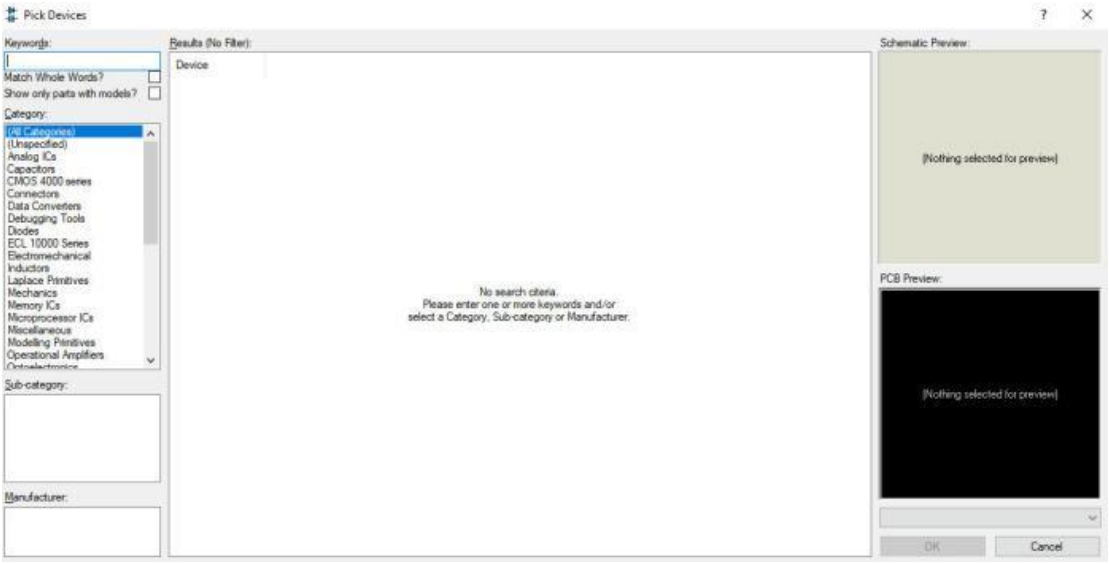
Để chọn mở linh kiện của Proteus, đầu tiên bạn nhấp vào nút Component Mode.



Tiếp theo bạn nhấp vào chữ P để mở thư viện.



Khi thư viện được mở ra, một cửa sổ sẽ xuất hiện như sau:



Trong đó:

Keywords: tìm kiếm linh kiện

Category và Sub-category: chứa các thư viện linh kiện trong chương trình Proteus

Results: hiển thị các linh kiện khi được chọn trong thư viện

Schematic Review: hiển thị hình dạng của linh kiện

PCB Preview: hiển thị sơ đồ chân PCB của linh kiện

Trong cửa sổ chọn linh kiện này bạn gõ tên linh kiện cần tìm vào ô Keywords. Ví dụ, bạn tìm IC 555, hãy gõ 555 vào ô Keywords thì IC 555 và tất cả các linh kiện liên quan đến 555 sẽ xuất hiện tự động ở phần Results. Bạn double click vào IC này để chọn nó. Những linh kiện đã được chọn sẽ xuất hiện ở trong ô Devices.

Bạn thực hiện tương tự và lấy thêm các linh kiện: điện trở, tụ hóa, tụ thường, led đơn, nguồn pin.

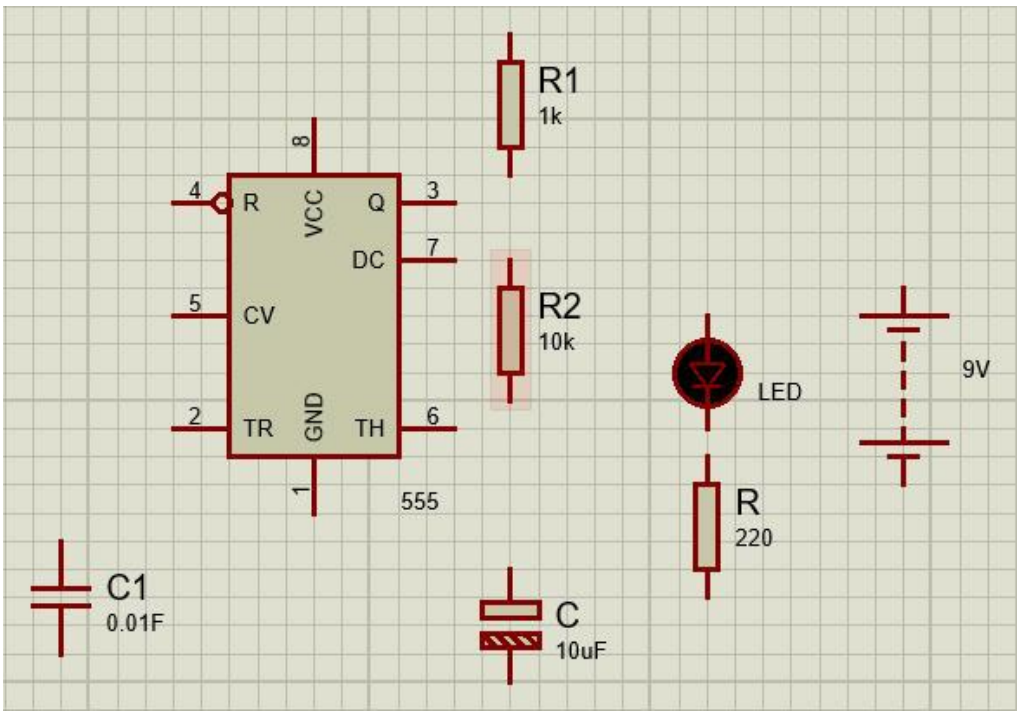
Sau khi đã lấy đầy đủ các linh kiện từ thư viện, bạn nhấp vào nút OK để đóng cửa sổ thư viện trở về màn hình thiết kế.

Lưu ý: Các linh kiện được chọn phải có sơ đồ chân PCB còn nếu không bạn phải tạo sơ đồ chân linh kiện khi chuyển sang phần thiết kế mạch in.

Bước 4: Đưa linh kiện ra ngoài màn hình thiết kế

Nhấp chuột vào linh kiện cần lấy trong ô Devices, sau đó di chuyển con trỏ ra ngoài màn hình thiết kế nơi cần đặt linh kiện và click chuột thì linh kiện sẽ được đặt tại đó.

Bạn di chuyển hết linh kiên ra ngoài màn hình thiết kế như hình sau:



Di chuyển linh kiện

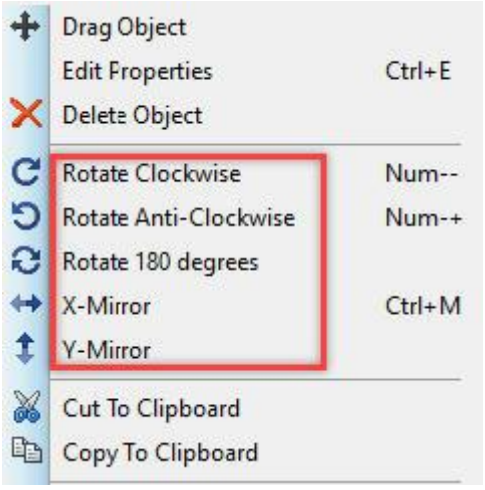
Để di chuyển linh kiện từ vị trí này đến vị trị khác, bạn thao tác như sau:

Nhấp và giữ trái chuột vào linh kiện cần di chuyển, sau đó rê chuột đến vị trí mới và thả chuột ra. Bạn cũng có thể dùng lệnh Block Move trên thanh công cụ di chuyển linh kiện.

Xoay linh kiện

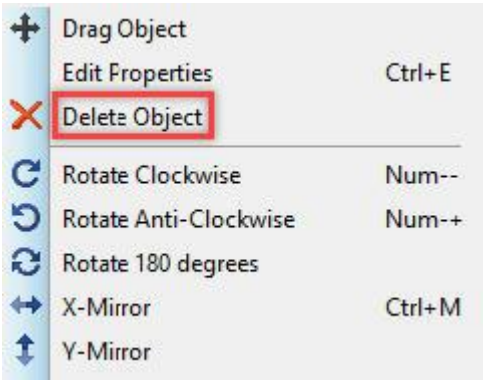
Để xoay các linh kiện bạn thao tác như sau:

Đặt con trỏ lên linh kiện cần xoay sau đó bấm phải chuột, bạn chọn các lệnh xoay (rotate) theo chiều kim đồng hồ, ngược chiều kim đồng hồ, xoay 180^0. Bạn có thể lật (mirror) linh kiện theo chiều ngang hay chiều dọc cũng từ cửa sổ tắt này. Bạn cũng có thể dùng công cụ Block Rotate trên thanh công cụ để xoay linh kiện.



Xóa linh kiện

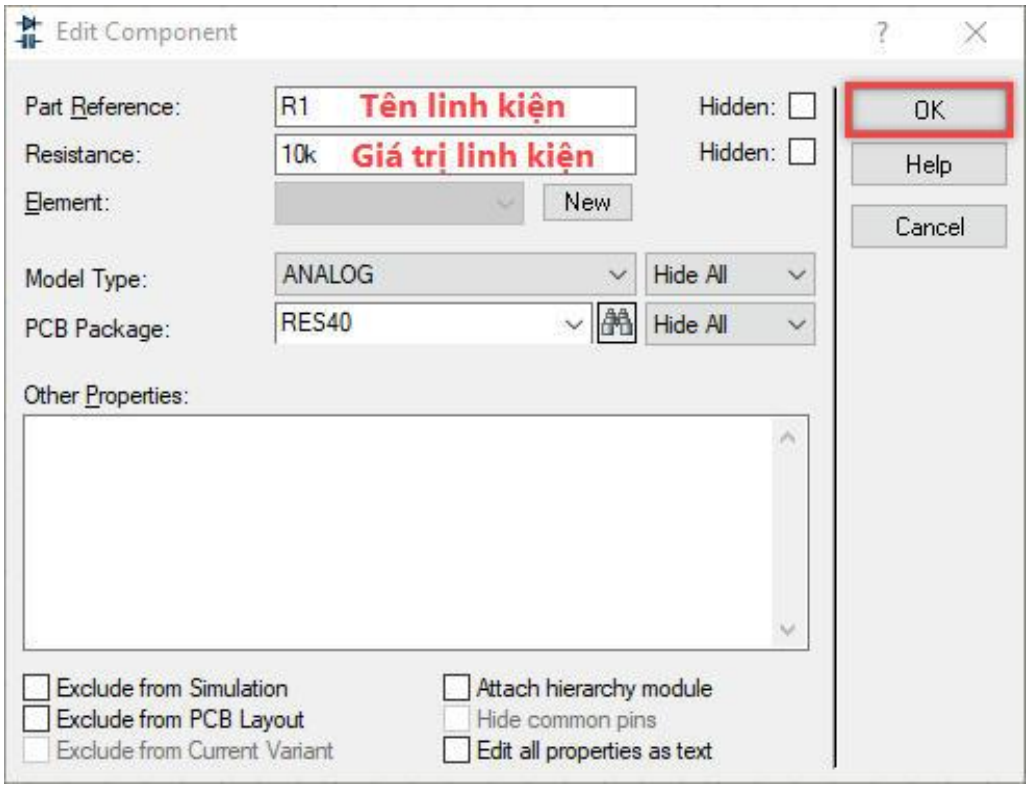
Bạn để con trỏ lên linh kiện cần xóa rồi bấm phải chuột sau đó bạn chọn lệnh Delete Object từ shortcut menu. Bạn cũng có thể dùng phím Delete để xóa linh kiện hoặc dùng công cụ Block Delete trên thành công cụ để xóa linh kiện.



Bước 5: Thay đổi thông số kỹ thuật của linh kiện

Để vẽ mạch một cách nhanh chóng chúng ta không nhất thiết phải lấy linh kiện có các thông số chính xác, nhất là trong mạch có nhiều linh kiện giống nhau nhưng khác thông số kỹ thuật. Nếu lấy từng linh kiện đúng với các thông số yêu cầu thì sẽ mất rất nhiều thời gian và đôi khi trong thư viện không có linh kiện với thông số mình cần tìm. Vì vậy, ta cần phải thay đổi các thông số kỹ thuật cho linh kiện.

Ví dụ: Sau khi đặt điện trở ra ngoài màn hình thiết kế, bạn double click vào linh kiện này, một cửa sổ sẽ hiện ra bạn tiến hành thay đổi tên và giá trị của điện trở vào 2 ô Part Reference và Resistance tương ứng. Cuối cùng bạn nhấp chọn OK để hoàn tất việc chỉnh sửa.



Bước 6: Bố trí, sắp xếp lại linh kiện cho hợp lý

Bạn dùng các lệnh di chuyển linh kiện, lật linh kiện,…như đã trình bày ở trên để bố trí, sắp xếp lại các linh kiện trong mạch sao cho thật hợp lý trước khi tiến hành bước tiếp theo. Mục đích của việc làm này là làm cho sơ đồ mạch được rõ ràng khi quá trình thiết kế mạch được hoàn tất.

Bước 7: Nối dây

Sau khi lấy và sắp xếp các linh kiện theo mong muốn, bạn tiến hành nối các chân linh kiện cho mạch. Bạn tiến hành như sau:

Đặt con trỏ trên chân linh kiện cần nối dây cho đến khi ô vuông màu đỏ xuất hiện sau đó bạn click chuột vào chân linh kiện và chế độ nối dây được bắt đầu. Bạn rê chuốt đến chân linh kiện cần nối khác và click chuột một lần nữa để kết thúc quá trình nối dây. Bạn thao tác tương tự như vậy cho đến khi hoàn thành sơ đồ mạch.

Để xóa đường nối dây sai, bạn nhấp phải chuột trên đường dây nối và chọn Delete Wire hoặc double click phải trên đường dây nối.

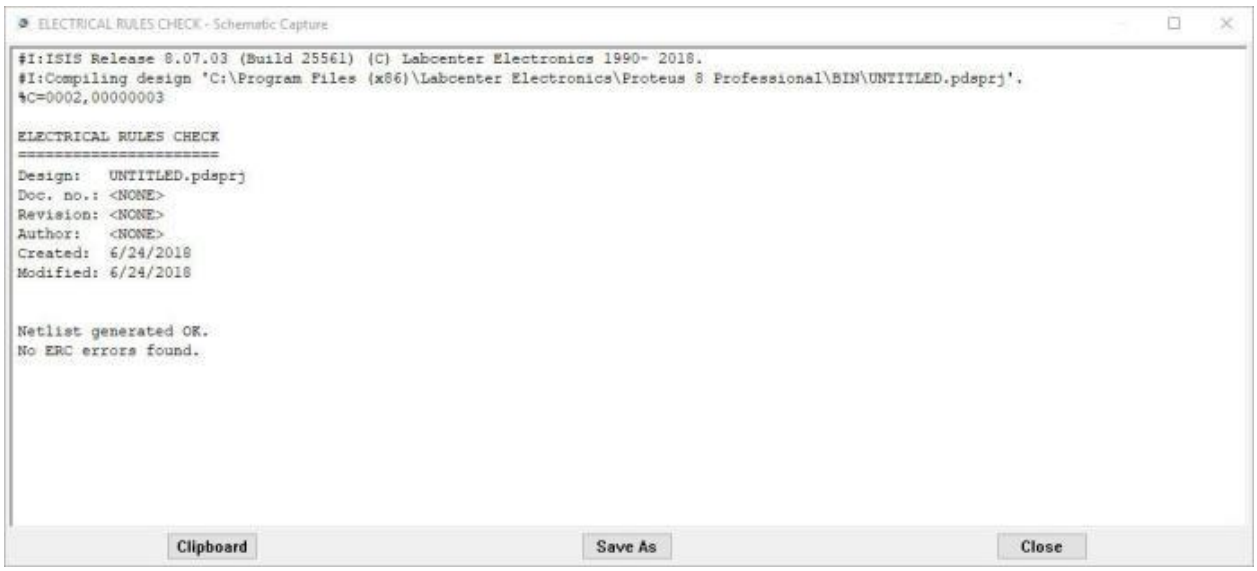
Bước 8: Kiểm tra sơ đồ mạch nguyên lý

Kiểm tra sơ đồ mạch sau khi hoàn thành xong mạch thiết kế là rất quan trong, nó giúp bạn tìm được những lỗi mà trong quá trình thiết kế bạn chưa phát hiện ra được.

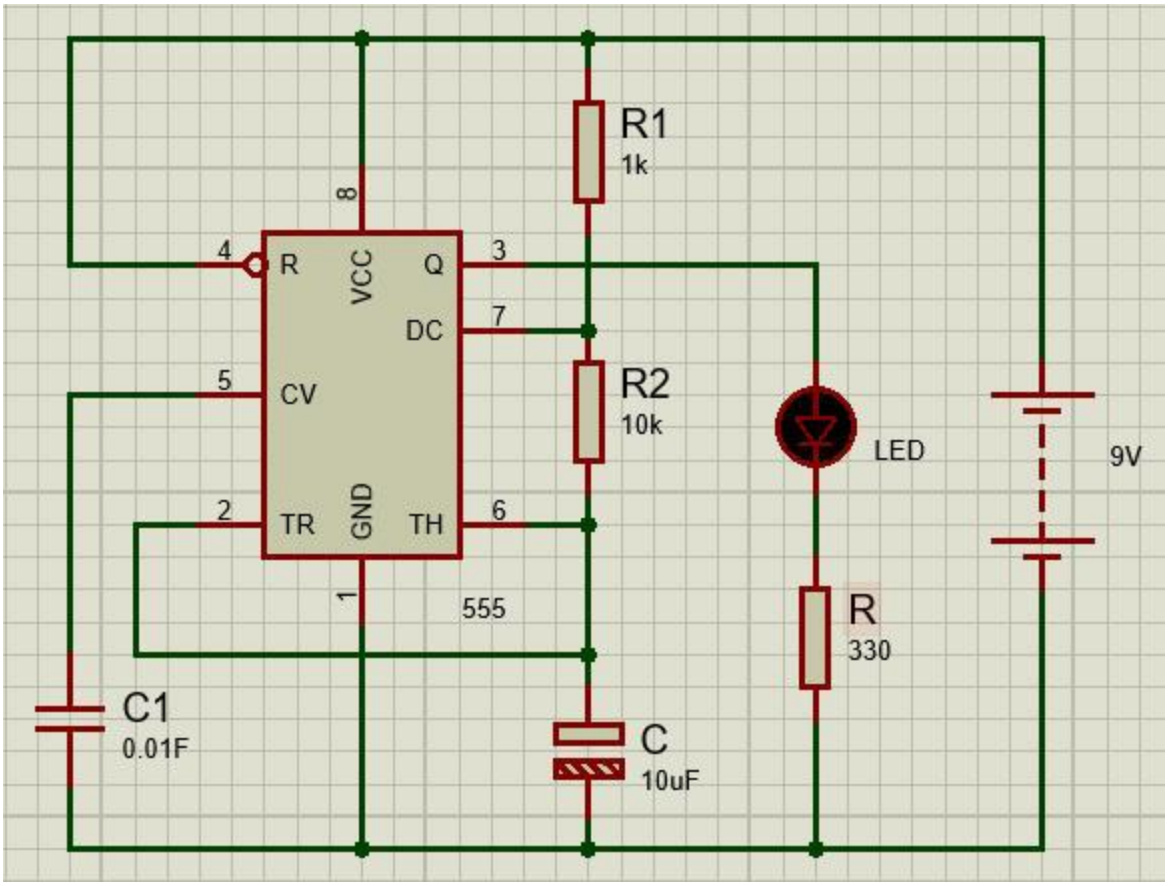
Để kiểm tra lỗi ta thao tác như sau:

Trên thanh công cụ, bạn chọn Tool >> Electrical Rule Check

Nếu có thông lỗi bạn tìm cách khắc phục cho đến khi không còn lỗi và nhận được dòng thông báo (No ERC errors found) như hình dưới đây nhé.



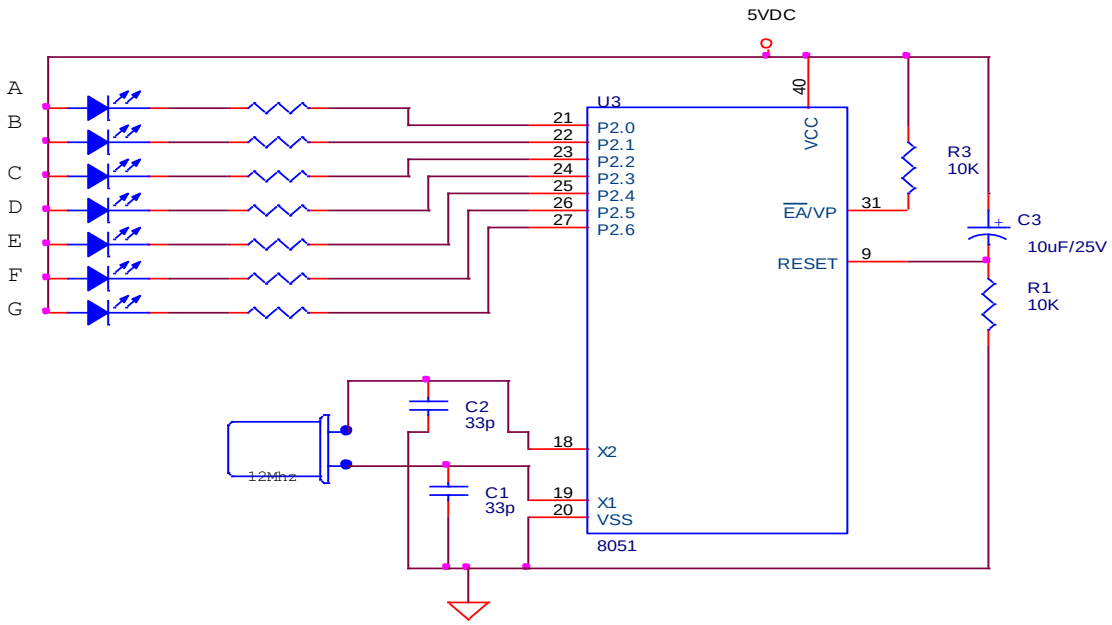
Sau khi kiểm tra và hiệu chỉnh sơ đồ mạch như mong muốn bạn nhớ lưu lại. Mạch dạo động đa hài phi ổn dùng IC 555 được vẽ bằng chương trình ISIS của Proteus như sau:



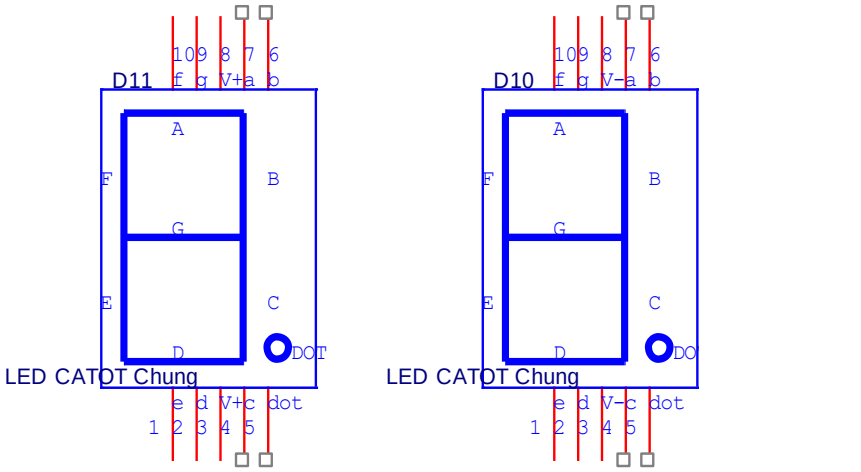
6.2. Thiết kế mạch điều khiển

6.2.1. Giao tiếp điều khiển led 7 đoạn

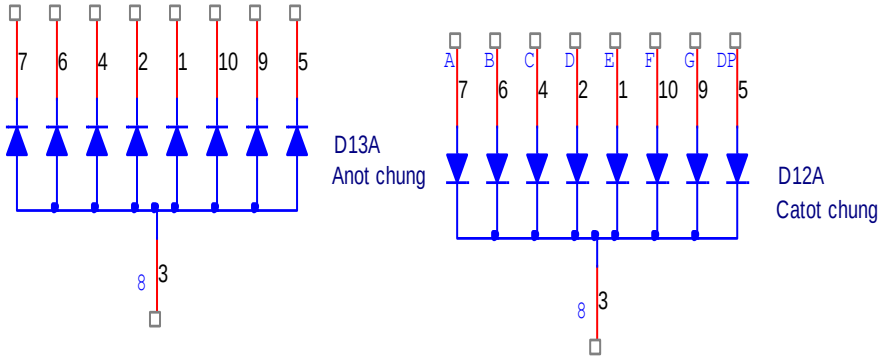
a/ Sơ đồ mạch điện:



Có hai loại led 7 đoạn: Anốt chung và Catốt chung. Hình dưới là sơ đồ chân của hai loại led.



Cấu tạo như sau: Chỉ là 8 con led đấu chung 1 đầu: Anốt hoặc Catốt.



b/ Nguyên lí hoạt động:

Khi đấu nguồn vào mạch tất cả các chân của các cổng IO của VĐK là 5V (Nếu cổng 0 không lắp điện trở treo thì sẽ là 0V). Nhìn sơ đồ mạch không có chênh lệch điện áp nên không có đèn nào sáng. Muốn sáng thanh nào chỉ việc đưa ra điện áp 0V ở chân vi điều khiển nối với thanh đó.

	Thanh hiện	Thanh tắt	Giá trị (P2)
Để hiện thị số 1:	B,C	các thanh còn lại	1111 1001
Để hiện thị số 2:	A,B,D,E,G	các thanh còn lại	1010 0100
….			
Để hiện thị số 8:	Tất cả các thanh	không thanh nào	1000 0000
			gfe dcba

Bít thứ 8 P2.7 không dùng.

Ngoài ra led 7 thanh còn có thể hiện thị 1 số chữ

Đê hiển thị chữ B: Giống sô 8

Hiển thị chữ A: A,B,C,E,F,G D 1000 1000

c/ Soạn thảo chương trình:

Ví dụ:

Hàm hiển thị số 1:

```
void so1(void)
{ tat();
P2=0xF5;}
```

Mô phỏng cho hiển thị cổng P2 lên. Để dấu lựa chọn ở các đèn tắt (1) , bỏ dấu lựa chọn ở các đèn cần bật (0). Rồi đọc giá trị hexa.

d/ Nạp chương trình:

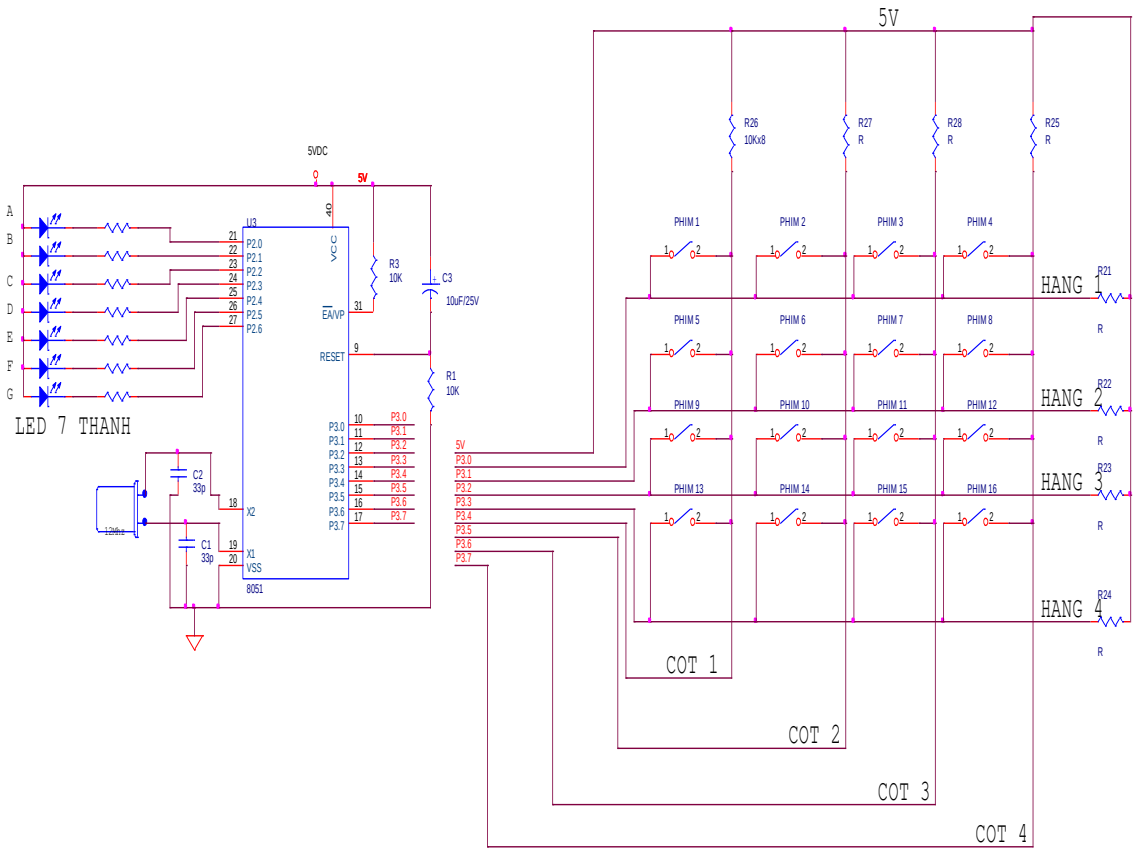
6.2.2. Đọc bàn phím:

Đọc ma trận phím.

Nhiệm vụ:

Quét bàn phím 16 phím bấm (4x4), xem phím nào được bấm, các phím được đánh số từ 0 đến 15 rồi hiển thị giá trị ra led 7 thanh.

a/ Sơ đồ mạch điện:



b/ Nguyên lí quét phím:

- Có 2 cách quét phím theo cột và theo hàng, ta chọn cách quét theo hàng, quét theo cột có thể làm tương tự.

- Bước 1: Ta đưa chân P3.0 nối với Hàng 1 xuống 0V. Kiểm tra giá trị logic của các chân P3.4,P3.5,P3.6,P3.7. Nếu phím 1 được bấm thì Cột 1_ P3.4 sẽ có giá trị bằng 0. Nếu phím 2 được bấm thì Cột 2_ P3.5 sẽ có giá trị bằng 0. Nếu phím 3 được bấm thì

Cột 3_ P3.6 sẽ có giá trị bằng 0. Nếu phím 4 được bấm thì Cột 4_ P3.7 sẽ có giá trị bằng 0. Ta căn cứ vào đó để xác định xem phím nào được bấm.

- Bước 2: Đưa chân P3.1 nối với Hàng 2 xuống 0V. Kiểm tra giá trị logic của các chân P3.4,P3.5,P3.6,P3.7. Nếu phím 5 được bấm thì Cột 1_ P3.4 sẽ có giá trị bằng 0. Nếu phím 6 được bấm thì Cột 2_ P3.5 sẽ có giá trị bằng 0. Nếu phím 7 được bấm thì Cột 3_ P3.6 sẽ có giá trị bằng 0. Nếu phím 8 được bấm thì Cột 4_ P3.7 sẽ có giá trị bằng 0. Ta căn cứ vào đó để xác định xem phím nào được bấm.

- Bước 3: Đưa chân P3.2 nối với Hàng 3 xuống 0V. Kiểm tra giá trị logic của các chân P3.4,P3.5,P3.6,P3.7. Nếu phím 9 được bấm thì Cột 1_ P3.4 sẽ có giá trị bằng 0. Nếu phím 10 được bấm thì Cột 2_ P3.5 sẽ có giá trị bằng 0. Nếu phím 11 được bấm thì Cột 3_ P3.6 sẽ có giá trị bằng 0. Nếu phím 12 được bấm thì Cột 4_ P3.7 sẽ có giá trị bằng 0. Ta căn cứ vào đó để xác định xem phím nào được bấm.

- Bước 4: Ta đưa chân P3.3 nối với Hàng 1 xuống 0V. Kiểm tra giá trị logic của các chân P3.4,P3.5,P3.6,P3.7. Nếu phím 13 được bấm thì Cột 1_ P3.4 sẽ có giá trị bằng 0. Nếu phím 14 được bấm thì Cột 2_ P3.5 sẽ có giá trị bằng 0. Nếu phím 15 được bấm thì Cột 3_ P3.6 sẽ có giá trị bằng 0. Nếu phím 16 được bấm thì Cột 4_ P3.7 sẽ có giá trị bằng 0. Ta căn cứ vào đó để xác định xem phím nào được bấm.

Dùng câu lệnh if để kiểm tra.

c/ Soạn thảo chương trình:

- Tạo 1 project mới, copy phần hiển thị các sô 0…9 các chữ A…Y của bài trước. Rồi bổ sung các hàm sau. Hàm hiện thị phím ấn.

```
void phim_duoc_an(unsigned char phim)
{
    switch(phim)// Tuy vao so lan
        {case 0: { so0(); break; }// Neu so lan =0 hien so 0 thoat khoi switch
        case 1: { so1(); break; }// Neu so lan =1 hien so 1 thoat khoi switch
        case 2: { so2(); break; }// ....
        case 3: { so3(); break; }
        case 4: { so4(); break; }
        case 5: { so5(); break; }
        case 6: { so6(); break; }
        case 7: { so7(); break; }
        case 8: { so8(); break; }
        case 9: { so9(); break; }// Neu so lan =9 hien so 9 thoat khoi switch
        }}
```

Hàm quét phím:

/*Khai bao 1 mang 4 phan tu nhu sau: quetphim[4]={P0=0xFE,0xFD,0xFB,0xF7}

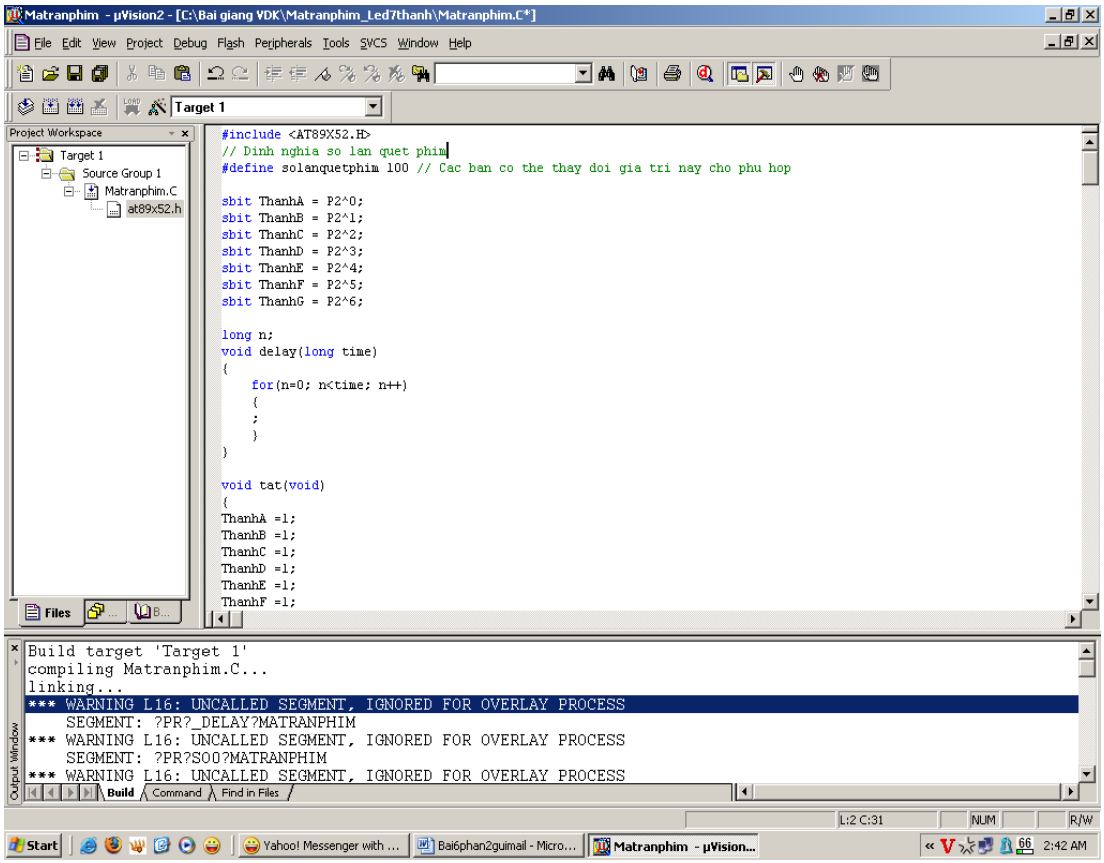
De dua 0 ra lan luot cac hang phim, khi do neu nut nao duoc an thi chan vi dieu khien se xuong 0.Chu y fai kiem tra phim khoang 100 lan.*/

```
unsigned char quetphim[4]={0xFE,0xFD,0xFB,0xF7};
// Dinh nghia so lan quet phim
#define solanquetphim 100 // Cac ban co the thay doi gia tri nay cho phu hop
unsigned char quetbanphim(void)
{
unsigned char giatribanphim;// Bien de luu gia tri phim an tu 0 den 15 ma hoa 16 phim
unsigned char x,y;
        //Quet 4 hang phim
        for(x=0; x<4;x++)
        {
                P3=quetphim[x];// Dua lan luot cac hang xuong 0
                for(y=0;y<solanquetphim;y++)// Kiem tra solanquetphim lan
                        {if(P3_4==0) giatribanphim=0+4*x;// Gia tri phim tuong ung
                        if(P3_5==0) giatribanphim=1+4*x;// Tuy thuoc vao hang x
                        if(P3_6==0) giatribanphim=2+4*x;// La may ma gia tri cua
                        if(P3_7==0) giatribanphim=3+4*x;// gia tri ban phim tuong ung.
                        }       }
        return(giatribanphim);
}
```

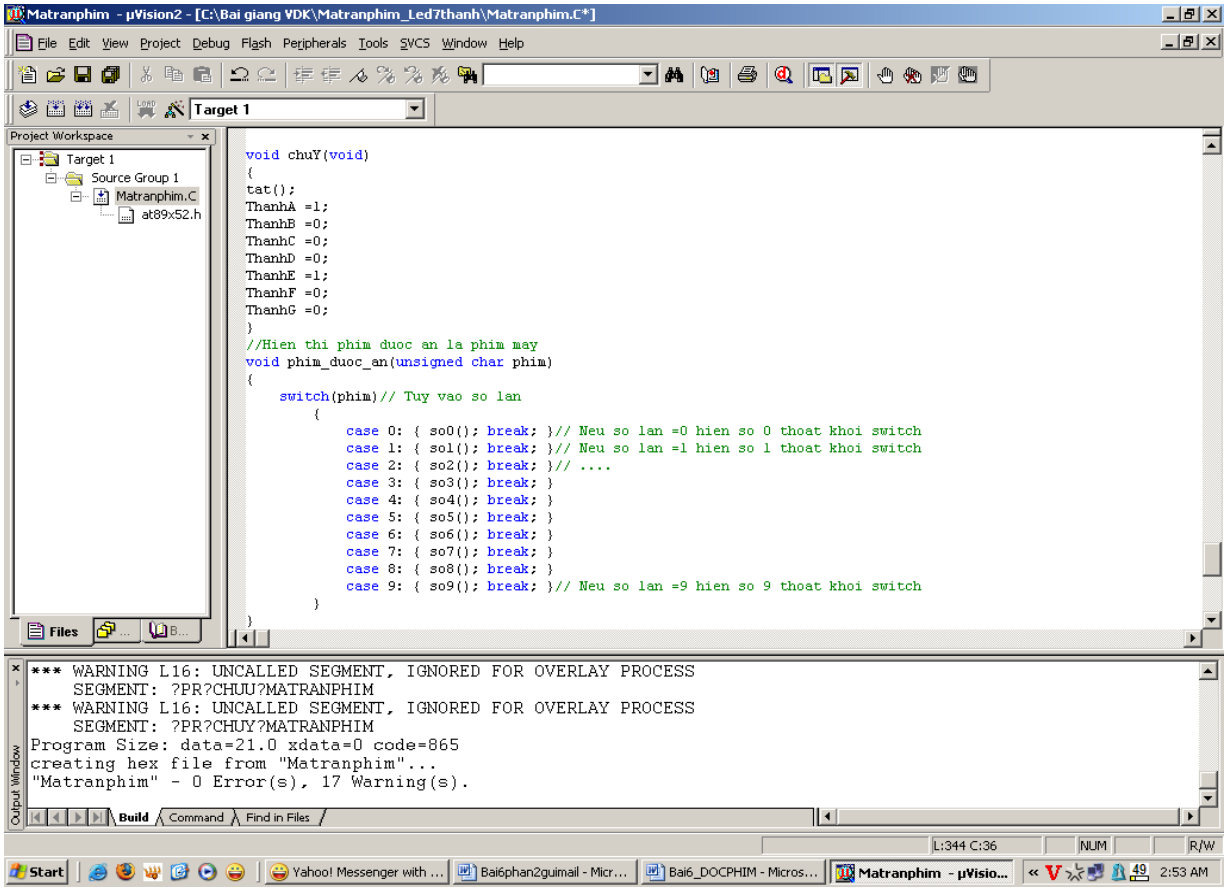
Hàm Main.

```
void main(void)
{
unsigned char i;
        while(1)
        {
                i=quetbanphim();
                phim_duoc_an(i);
        }}
```

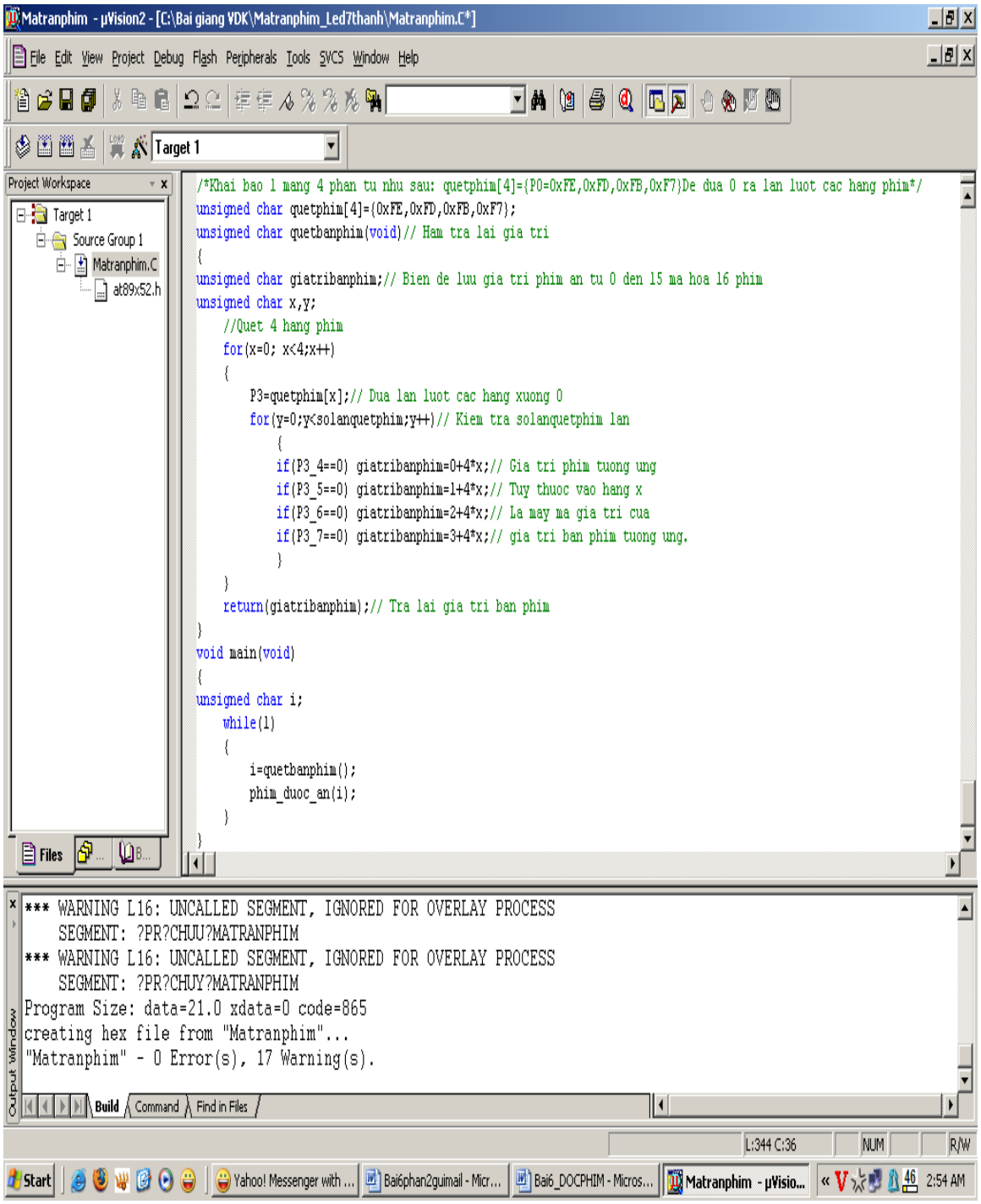
Thêm câu lệnh #define vào đầu chương trình:



Viết hàm phím được ấn:



Viết hàm quét bàn phím và hàm main.



6.2.3. Điều khiển LCD 16x2

Nhiệm vụ: Điều khiển hiển thị LCD 16x2 dòng chữ “www.EmbestDKS.com” chạy trên màn hình LCD.

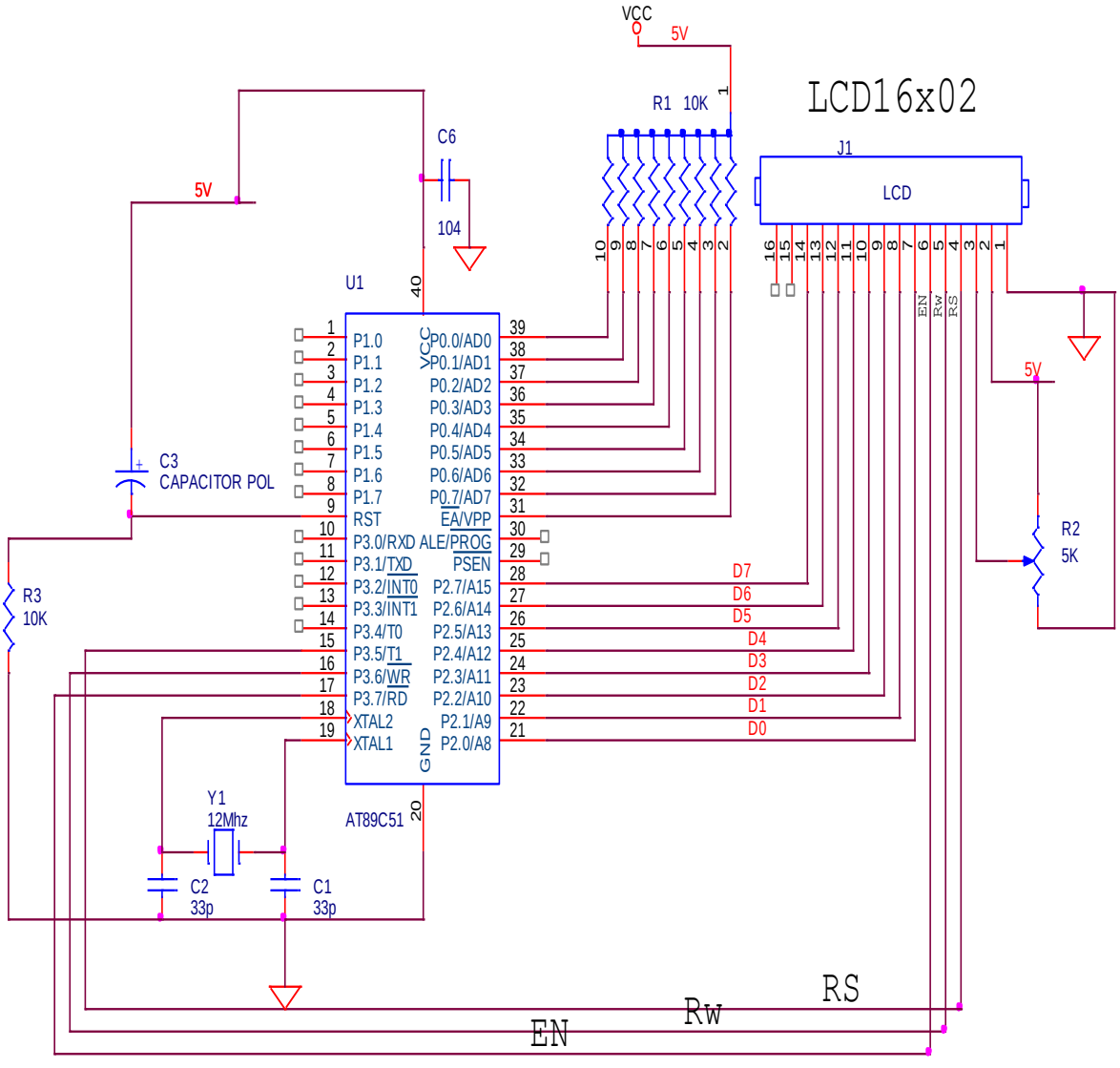
Có 16 chân như sau:

Chân	Ký hiệu	I/O	Mô tả
1	V_{SS}	-	Đất
2	V_{CC}	-	Dương nguồn 5v
3	V_{EE}	-	Cấp nguồn điều khiển phản

4	RS	I	RS = 0 chọn thanh ghi lệnh. RS = 1 chọn thanh dữ liệu
5	R/W	I	R/W = 1 đọc dữ liệu. R/W = 0 ghi
6	E	I/O	Cho phép
7	DB0	I/O	Các bít dữ liệu
8	DB1	I/O	Các bít dữ liệu
9	DB2	I/O	Các bít dữ liệu
10	DB3	I/O	Các bít dữ liệu
11	DB4	I/O	Các bít dữ liệu
12	DB5	I/O	Các bít dữ liệu
13	DB6	I/O	Các bít dữ liệu
14	DB7	I/O	Các bít dữ liệu

Chân 15 và chân 16 là anốt và katốt của 1 led dùng để sáng LCD trong bóng tối. Chúng ta không sử dụng. Nếu muốn dùng thì nối chân A qua 1 điện trở từ 1K đến 5K lên dương 5V, chân K xuống mass đèn sẽ sáng.

a/ Sơ đồ mạch điện:



b/ Nguyên lí hoạt động của LCD:

- Chân V_{CC}, V_{SS} và V_{EE}: Các chân V_{CC}, V_{SS} và V_{EE}: Cấp dương nguồn - 5v và đất tương ứng thì V_{EE} được dùng để điều khiển độ tương phản của LCD.

- Chân chọn thanh ghi RS (Register Select): Có hai thanh ghi trong LCD, chân RS (Register Select) được dùng để chọn thanh ghi như sau: Nếu RS = 0 thì thanh ghi mà lệnh được chọn để cho phép người dùng gửi một lệnh chẳng hạn như xoá màn hình, đưa con trỏ về đầu dòng v.v… Nếu RS = 1 thì thanh ghi dữ liệu được chọn cho phép người dùng gửi dữ liệu cần hiển thị trên LCD.

- Chân đọc/ ghi (R/W): Đầu vào đọc/ ghi cho phép người dùng ghi thông tin lên LCD khi R/W = 0 hoặc đọc thông tin từ nó khi R/W = 1.

- Chân cho phép E (Enable): Chân cho phép E được sử dụng bởi LCD để chốt dữ liệu của nó. Khi dữ liệu được cấp đến chân dữ liệu thì một xung mức cao xuống thấp phải được áp đến chân này để LCD chốt dữ liệu trên các chân dữ liêu. Xung này phải rộng tối thiểu là 450ns.

- Chân D0 - D7: Đây là 8 chân dữ liệu 8 bít, được dùng để gửi thông tin lên LCD hoặc đọc nội dung của các thanh ghi trong LCD. Để hiển thị các chữ cái và các con số, chúng ta gửi các mã ASCII của các chữ cái từ A đến Z, a đến f và các con số từ 0 - 9 đến các chân này khi bật RS = 1.

Cũng có các mã lệnh mà có thể được gửi đến LCD để xoá màn hình hoặc đưa con trỏ về đầu dòng hoặc nhấp nháy con trỏ.

- Chú ý: Chúng ta cũng sử dụng RS = 0 để kiểm tra bít cờ bận để xem LCD có sẵn sàng nhân thông tin. Cờ bận là bít D7 và có thể được đọc khi R/W = 1 và RS = 0 như sau:

Nếu R/W = 1, RS = 0 khi D7 = 1 (cờ bận 1) thì LCD bận bởi các công việc bên trong và sẽ không nhận bất kỳ thông tin mới nào. Khi D7 = 0 thì LCD sẵn sàng nhận thông tin mới. Lưu ý chúng ta nên kiểm tra cờ bận trước khi ghi bất kỳ dữ liệu nào lên LCD.

- Sau đây là bảng mã lệnh của LCD:

Mã (Hex)	Lệnh đến thanh ghi của LCD
1	Xoá màn hình hiển thị
2	Trở về đầu dòng
4	Giảm con trỏ (dịch con trỏ sang trái)
6	Tăng con trỏ (dịch con trỏ sang phải)
5	Dịch hiển thị sang phải
7	Dịch hiển thị sang trái
8	Tắt con trỏ, tắt hiển thị
A	Tắt hiển thị, bật con trỏ
C	Bật hiển thị, tắt con trỏ

E	Bật hiển thị, nhấp nháy con trỏ
F	Tắt con trỏ, nhấp nháy con trỏ
10	Dịch vị trí con trỏ sang trái
14	Dịch vị trí con trỏ sang phải
18	Dịch toàn bộ hiển thị sang trái
1C	Dịch toàn bộ hiển thị sang phải
80	Ép con trỏ về đầu dòng thứ nhất
C0	Ép con trỏ về đầu dòng thứ hai
38	Hai dòng và ma trận 5 × 7

- Điều khiển LCD qua các bước sau:

Bước 0 : Chuẩn bị phần cứng. Xoay biến trở 5 K điều chỉnh độ tương phản của LCD. Xoay cho đến khi các ô vuông (các điểm ảnh) của LCD hiện lên thì xoay ngược biến trở lại 1 chút.

Bước 1 : Khởi tạo cho LCD.

Bước 2 : Gán các giá trị cho các bit điều khiển các chân RS,RW,EN cho phù hợp với các chế độ: Hiển thị kí tự lên LCD hay Thực hiện 1 lệnh của LCD.

Bước 3: Xuất byte dữ liệu ra cổng điều khiển 8 bit dữ liệu của LCD.

Bước 4: Kiểm tra cờ bận xem LCD sẵn sàng nhận dữ liệu mới chưa.

Bước 5: Quay vòng lại bước 1.

c/ Lập trình:

- Để có thể lập trình cho LCD ta thêm vào thư viện string.h của trình biên dịch bằng câu lệnh:

#include <string.h>

- Khai báo các chân của LCD gắn với các cổng:

```
/*
RS chon thanh ghi
        =0 ghi lenh;            =1 ghi du lieu
RW doc ghi
        =0 ghi;                 =1 doc
E cho fep chot du lieu
        xung cao xuong thap toi thieu 450 ns.
Bit co ban D7
        khi RS=0 RW=1 neu D7=1 LCD ban;          D7=0 LCD san sang.
*/
```

```
sfr LCDdata = 0xA0;// Cong 2, 8 bit du lieu  P0 co dia chi 0x80, P1 0x90 , P2 0xA0
sbit BF = 0xA7; // Co ban bit 7
sbit RS = P3^5;
sbit RW = P3^4;
sbit EN = P3^3;
```

- Viết 1 số hàm điều khiển LCD như sau:

* Hàm kiểm tra LCD có bận hay không:

```
void wait(void)
{               long n = 0;
                EN=1;// Dua chan cho fep len cao
                RS=0;// Chon thanh ghi lenh
                RW=1;//       Doc tu LCD
                LCDdata=0xFF;// Gia tri 0xFF
                while(BF){n++; if(n>100) break;}// Kiem tra co ban
                // Neu ban dem n den 100 roi thoat khoi while
                EN=0;// Dua xung cao xuong thap de chot
                RW=0;// Doc tu LCD
}
```

* Hàm điều khiển LCD thực hiện 1 lệnh:

```
void LCDcontrol(unsigned char x)
{               EN=1;// Dua chan cho fep len cao
                RS=0;// Chon thanh ghi lenh
                RW=0;// Ghi len LCD
                LCDdata=x;// Gia tri x
                EN=0;// Xung cao xuong thap
                wait();// Doi LCD san sang
}
```

Hàm có 1 biến đầu vào là các giá trị trong bảng mã lệnh của LCD.

* Hàm khởi tạo LCD:

```
void LCDinit(void)
{               LCDcontrol(0x30);//Che do 8 bit.
```

```
                LCDcontrol(0x30);
                LCDcontrol(0x30);
                LCDcontrol(0x38);// 2 dong va ma tran 5x7
                LCDcontrol(0x0C);// Bat con tro
                LCDcontrol(0x06);// Tang con tro xang fai
                LCDcontrol(0x01);// Xoa man hinh
}
```

* Hàm lệnh cho LCD hiển thị 1 kí tự :

```
void LCDwrite(unsigned char c)
{               EN=1;// Cho fep muc cao
                RS=1;// Ghi du lieu
                RW=0;// Ghi len LCD
                LCDdata=c;// Gia tri C
                EN=0;// Xung cao xuong thap
                wait();// Cho
}
```

Hàm có 1 biến đầu vào là mã của kí tự trong bảng ASCII.

* Hàm lệnh cho LCD hiển thị 1 xâu kí tự (dòng chữ):

```
void LCDputs(unsigned char *s,unsigned char row)
{
unsigned char len;
if(row==1) LCDcontrol(0x80);// Ep con tro ve dau dong 1
else LCDcontrol(0xC0);// Ep con tro ve dau dong 2
                len=strlen(s);// Lay do dai bien duoc tro boi con tro
                while(len!=0)// Khi do dai van con
                {
                   LCDwrite(*s);// Ghi ra LCD gia tri duoc tro boi con tro
                        s++;// Tang con tro
                        len--;// Tru do dai
                }
}
```

Hàm có hai biến đầu vào là: xâu kí tự cần hiển thị và dòng cần hiển thị xâu đó (1 hoặc 2).

*s là con trỏ, trỏ tới biến s

* Hàm hiển thị 1 số integer:

```
void LCDwritei(int d)
{
unsigned char i,j,k,l;
i=d%10;// Chia lay phan du, duoc chu so hang don vi
d=d/10;// Chia lay phan nguyen, duoc nhung chu so da bo hang don vi
j=d%10;// Duoc chu so hang chuc
d=d/10;// Nhung chu so da bo hang don vi va hang chuc
k=d%10;// Duoc hang tram
l=d/10;// Duoc hang nghin
LCDwrite(48+l);// Hien thi ki tu trong bang ascii
LCDwrite(48+k);// Trong bang ascii so 0 co co so thu tu la 48
LCDwrite(48+j);
LCDwrite(48+i);
}
```

Hàm có 1 biến đầu vào là số int lớn đến hàng nghìn cần hiển thị.

* Hàm trễ:

```
void delay(long time)
{
    long n;
    for(n=0;n<time;n++) ;
}
```

* Hàm main:

```
void main(void)
{
char x;
    LCDinit();
    LCDputs("8052 MCU",1);
    delay(30000);
```

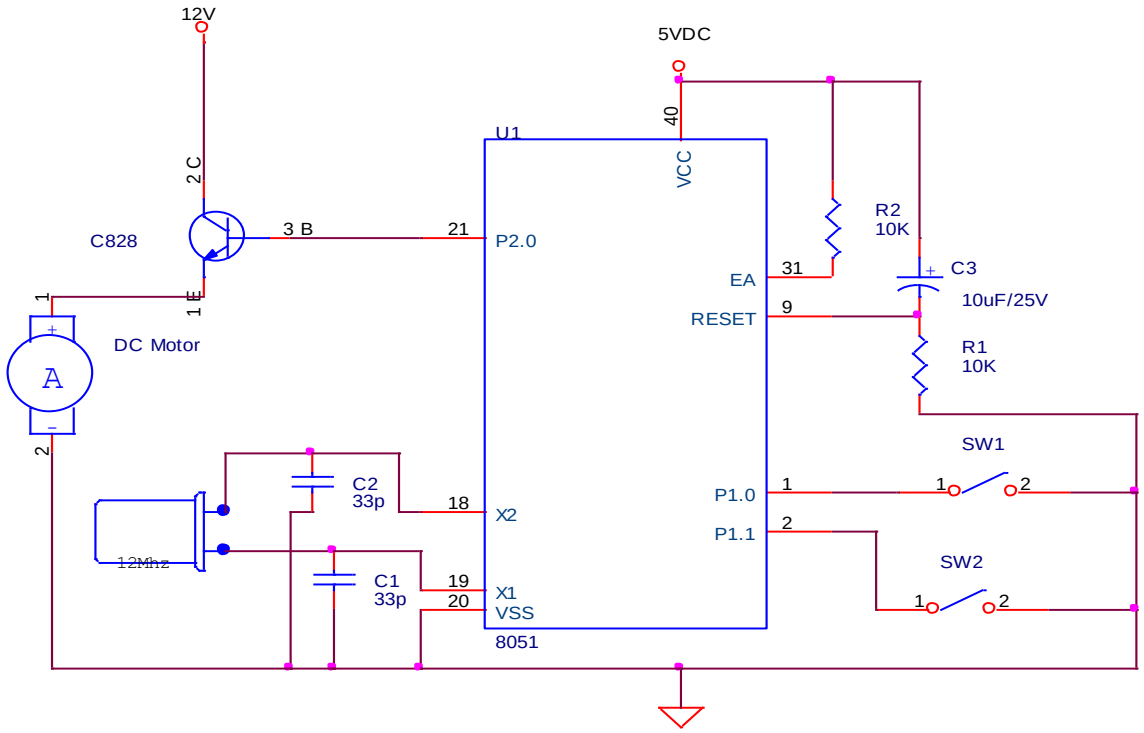
```
while(1)
{       for(x=0;x<16;x++)// Dich 16 lan.
{       LCDputs("8052 MCU",1);
LCDcontrol(0x18);// Dich hien thi sang trai.
delay(5000);// Tre
}       }       }
```

6.2.4. Điều chế độ rộng xung – Điều khiển tốc độ động cơ

Nhiệm vụ:

Tạo ra xung có độ rộng thay đổi, 10 cấp, tần số 1Khz, để điều khiển tốc độ động cơ (10 cấp tốc độ).

a/ Sơ đồ mạch điện:



b/ Soạn thảo chương trình:

- Cách tạo xung có độ rộng thay đổi bằng VĐK.

+ Cách 1: Việc điều khiển nhấp nháy 1 con led, đó là tạo ra 1 xung ở 1 chân của vi điều khiển, nhưng xung đó có độ rộng cố định, tần số lớn, ta có thể điều chỉnh lại hàm delay để tần số của nó đúng 1 Khz. Tuy nhiên vì là dùng hàm delay nên trong thời gian có xung lên 1 (5V) và thời gian không có xung (0V) vi điều khiển không làm gì cả, hơn nữa tạo xung bằng việc delay mà cần 2 bộ phát xung ở 2 kênh, có cùng tần số mà khác độ rộng xung thì trở nên rất khó khăn. Cho nên chúng ta dùng bộ định thời Timer của vi điều khiển trong trường hợp này rất tiện.

+ Cách 2: Dùng ngắt Timer của bộ vi điều khiển.

Ngắt do	Cờ	Địa chỉ vector
Reset hệ thống	RST	0000H
Ngắt ngoài 0	IE0	0003H
Bộ định thời 0	TF0	000BH
Ngắt ngoài 1	IE1	0013H
Bộ định thời 1	TF1	001BH
Port nối tiếp	RI hoặc TI	0023H
Bộ định thời 2	TF2 hoặc EXF2	002BH
Ngắt do	Cờ	Địa chỉ vector
Reset hệ thống	RST	0000H
Ngắt ngoài 0	IE0	0003H
Bộ định thời 0	TF0	000BH
Ngắt ngoài 1	IE1	0013H
Bộ định thời 1	TF1	001BH
Port nối tiếp	RI hoặc TI	0023H
Bộ định thời 2	TF2 hoặc EXF2	002BH

Riêng ngắt Reset không tính, bắt đầu đếm từ 0 và từ ngắt ngoài 0.

- Để sử dụng ngắt ta phải làm các công việc sau:

1) Khởi tạo ngắt: dùng ngắt nào thì cho phép ngắt đó hoạt động bằng cách gán giá trị tương ứng cho thanh ghi cho phép ngắt IE (Interrupt Enable):

EA	ET2	ES	ET1	EX1	EX0	ET0
IE		Điều khiển các nguồn ngắt (0: không cho phép; 1: cho phép)				
IE.7	EA	Cho phép/ không cho phép toàn cục				
IE.6	---	Không sử dụng				
IE.5	ET2	Cho phép ngắt do bộ định thời 2				
IE.4	ES	Cho phép ngắt do port nối tiếp				
IE.3	ET1	Cho phép ngắt cho bộ định thời 1				
IE.2	EX1	Cho phép ngắt từ bên ngoài (ngắt ngoài 1)				
IE.1	EX0	Cho phép ngắt từ bên ngoài (ngắt ngoài 0)				
IE.0	ET0	Cho phép ngắt do bộ định thời 0				
EA	ET2	ES	ET1	EX1	EX0	ET0
IE		Điều khiển các nguồn ngắt (0: không cho phép; 1: cho phép)				
IE.7	EA	Cho phép/ không cho phép toàn cục				
IE.6	---	Không sử dụng				
IE.5	ET2	Cho phép ngắt do bộ định thời 2				
IE.4	ES	Cho phép ngắt do port nối tiếp				

IE.3	ET1	Cho phép ngắt cho bộ định thời 1
IE.2	EX1	Cho phép ngắt từ bên ngoài (ngắt ngoài 1)
IE.1	EX0	Cho phép ngắt từ bên ngoài (ngắt ngoài 0)
IE.0	ET0	Cho phép ngắt do bộ định thời 0

IE là thanh ghi có thể xử lí từng bít.

2) Cấu hình cho ngắt: Trong 1 ngắt nó lại có nhiều chế độ ví dụ: với ngắt timer. Ta phải cấu hình cho nó chạy ở chế độ nào, chế độ timer hay counter, chế độ 16 bit, hay 8 bit,… bằng cách gán các giá trị tương ứng cho thanh ghi TMOD (Timer MODe).

TMOD		Chọn model cho bộ định thời 1			
7	GATE	Bít điều khiển cổng. Khi được set lên 1, bộ định thời chỉ hoạt động trong khi INT1 ở mức cao			
6	C/T	Bít chọn chức năng đếm hoặc định thời:			
		1= đếm sự kiện			
		0= định thời trong một khoảng thời gian			
5	M1	Bit chọn chế độ thứ nhất			
4	M0	Bit chọn chế độ thứ 2			
		M1	M0	Chế độ	Chức năng
		0	0	0	Chế độ định thời 13 bit
		0	1	1	Chế độ định thời 16 bit
		1	0	2	Chế độ tự động nạp lại 8 bit
		1	1	3	Chế độ định thời chia xẻ
3	GATE	Bit điều khiển cổng cho bộ định thời 0			
2	C/T	Bit chọn chức năng đếm / định thời cho bộ định thời 0			
1	M1	Bit chọn chế độ thứ nhất cho bộ định thời 0			
0	M0	Bit chọn chế độ thứ 2 cho bộ định thời 0			
TMOD		Chọn model cho bộ định thời 1			
7	GATE	Bít điều khiển cổng. Khi được set lên 1, bộ định thời chỉ hoạt động trong khi INT1 ở mức cao			
6	C/T	Bít chọn chức năng đếm hoặc định thời:			
		1= đếm sự kiện			
		0= định thời trong một khoảng thời gian			
5	M1	Bit chọn chế độ thứ nhất			
4	M0	Bit chọn chế độ thứ 2			
		M1	M0	Chế độ	Chức năng
		0	0	0	Chế độ định thời 13 bit
		0	1	1	Chế độ định thời 16 bit
		1	0	2	Chế độ tự động nạp lại 8 bit
TMOD		Chọn model cho bộ định thời 1			
7	GATE	Bít điều khiển cổng. Khi được set lên 1, bộ định			

		thời chỉ hoạt động trong khi INT1 ở mức cao			
6	C/T	Bít chọn chức năng đếm hoặc định thời:			
		1= đếm sự kiện			
		0= định thời trong một khoảng thời gian			
5	M1	Bit chọn chế độ thứ nhất			
4	M0	Bit chọn chế độ thứ 2			
		M1	M0	Chế độ	Chức năng
		0	0	0	Chế độ định thời 13 bit
		0	1	1	Chế độ định thời 16 bit
		1	0	2	Chế độ tự động nạp lại 8 bit
		1	1	3	Chế độ định thời chia xẻ
3	GATE	Bit điều khiển cổng cho bộ định thời 0			
2	C/T	Bit chọn chức năng đếm / định thời cho bộ định thời 0			
1	M1	Bit chọn chế độ thứ nhất cho bộ định thời 0			
0	M0	Bit chọn chế độ thứ 2 cho bộ định thời 0			

3) Bắt đầu chương trình có ngắt:

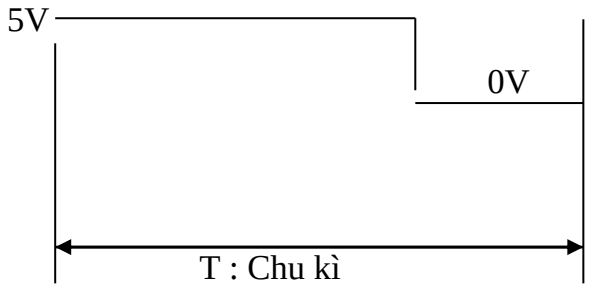
- Trước khi bắt đầu cho chạy chương trình ta phải cho phép ngắt toàn cục được xảy ra bằng cách gán EA (Enable All interrupt) bằng 1, thì ngắt mới xảy ra.

- Thường thì ngay vào đầu chương trình (hàm main) trước vòng while (1) chúng ta đặt công việc khởi tạo, cấu hình và cho phép kiểm tra ngắt. Ví dụ với bộ định thởi timer ta gán các giá trị phù hợp cho thanh ghi TCON(Timer CONtrol).

TCON		Điều khiển bộ đinh thời
TCON.7	TF1	Cờ tràn của bộ định thời 1. Cờ này được set bởi phần cứng khi có tràn, được xoá bởi phần mềm, hoặc bởi phần cứng khi bộ vi xử lý trỏ đến trình phục vụ ngắt
TCON.6	TR1	Bit điều khiển hoạt động của bộ định thời 1. Bit này được set hoặc xoá bởi phần mềm để điều khiển bộ định thời hoạt động hay ngưng
TCON.5	TF0	Cừ tràn của bộ định thời 0
TCON.4	TR0	Bit điều khiển hoạt động của bộ định thời 0
TCON.3	IE1	Cừ ngắt bên ngoài 1 (kích khởi cạnh). Cờ này được set bởi phần cứng khi có cạnh âm (cuống) xuất hiện trên chân INT1, được xoá bởi phần mềm, hoặc phần cứng khi CPU trỏ đến trình phục vụ ngắt
TCON.2	IT1	Cờ ngắt bên ngoài 1 (kích khởi cạnh hoặc mức). Cờ này được set hoặc xoá bởi phần mềm khi xảy ra cạnh âm hoặc mức thấp tại chân ngắt ngoài
TCON.1	IE0	Cờ ngắt bên ngoài 0 (kích khởi cạnh)
TCON.0	IT0	Cờ ngắt bên ngoài 0 (kích khởi cạnh hoặc mức)
TCON		Điều khiển bộ đinh thời
TCON.7	TF1	Cờ tràn của bộ định thời 1. Cờ này được set bởi phần cứng khi có tràn, được xoá bởi phần mềm, hoặc bởi

		phần cứng khi bộ vi xử lý trỏ đến trình phục vụ ngắt
TCON.6	TR1	Bit điều khiển hoạt động của bộ định thời 1. Bit này được set hoặc xoá bởi phần mềm để điều khiển bộ định thời hoạt động hay ngưng
TCON.5	TF0	Cừ tràn của bộ định thời 0
TCON.4	TR0	Bit điều khiển hoạt động của bộ định thời 0
TCON.3	IE1	Cừ ngắt bên ngoài 1 (kích khởi cạnh). Cờ này được set bởi phần cứng khi có cạnh âm (cuống) xuất hiện trên chân INT1, được xoá bởi phần mềm, hoặc phần cứng khi CPU trỏ đến trình phục vụ ngắt
TCON.2	IT1	Cờ ngắt bên ngoài 1 (kích khởi cạnh hoặc mức). Cờ này được set hoặc xoá bởi phần mềm khi xảy ra cạnh âm hoặc mức thấp tại chân ngắt ngoài
TCON.1	IE0	Cờ ngắt bên ngoài 0 (kích khởi cạnh)
TCON.0	IT0	Cờ ngắt bên ngoài 0 (kích khởi cạnh hoặc mức)

Với yêu cầu của bài. Tạo xung tần số 1Khz → Chu kì = $1/10^3$ = 0,001 giây= 1 mili giây=1000 uS= 1000 chu kì máy. Với 10 cấp tốc độ, tức là bạn phải tạo ra được xung 10%, 20%, 30%, 40%, …, 90%, 100%. 1 xung như sau:



1000 miro giây.

Khoảng thời gian xung kéo dài 5V là T1. Xung 10% tức là T1/ T= 10%=1/10. Xung 20% T2/T=2/10…PWM (Thay đổi độ rộng xung)

c/ Nguyên lí hoạt động:

- Xung PWM: Đưa ra mở transitor, xung với độ rộng lớn hơn transitor sẽ mở lâu hơn động cơ sẽ quay nhanh hơn. Không có xung động cơ sẽ không quay, có xung 100% động cơ sẽ quay max.Tuy nhiên xung phải lớn hơn mức nào đó thì mới đủ khởi động cho động cơ. Để có thể thay đổi 10 cấp tốc độ với chu kì 1000uS, ta khởi tạo cho ngắt timer: 100 uS ngắt 1 lần. Trong hàm ngắt kiểm tra xem ta cần cấp xung bao nhiêu % thì ta sẽ gán giá trị cho nó. Cụ thể như sau:

* Hàm khởi tạo ngắt.

Dùng ngắt timer 0, 100 uS ngắt 1 lần, dùng chế độ 2 8 bit tự động nạp lại của timer (vì mình chỉ cần đếm đến 100). TL0 nạp bằng 156. Đối với chế độ 2 khi tràn bộ đếm TL0 sẽ quay vòng giá trị bằng 0, nhưng sau đó nó lại được nạp giá trị lưu trong TH0 (giá trị nạp lại), do đó ta chỉ cần gán giá trị cho TL0 và TH0 trong hàm khởi tạo, còn ở các chế độ khác 16 bit, 2 timer counter 8 bit, khi tràn bộ đếm TL0 không được nạp lại mà ta phải tự gán lại giá trị cho nó trong hàm ngắt.

void khoitaotimer0(void)// Ham khoi tao

{ EA=0;// Cam ngat toan cuc

TMOD=0x02;// Timer 0 che do 2 8 bit auto reload

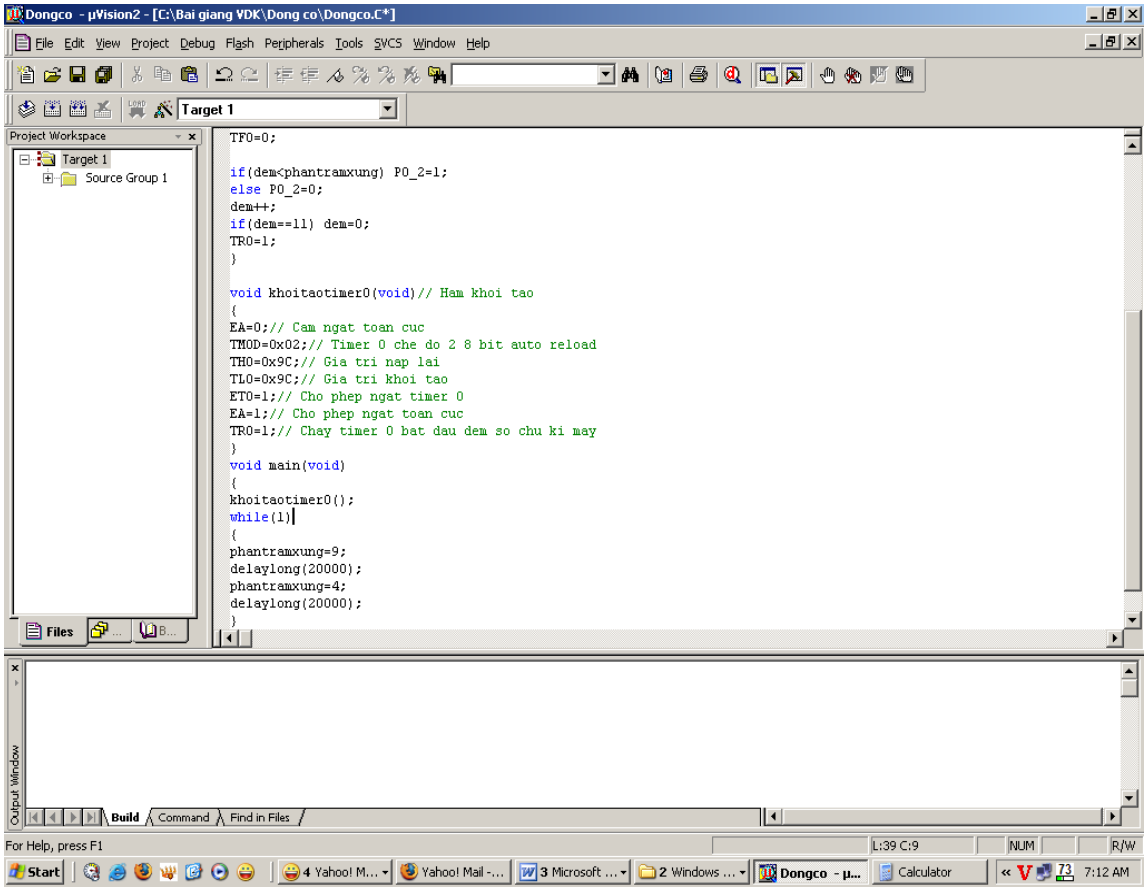
TH0=0x9B;// Gia tri nap lai 155 doi ra so hex

TL0=0x9B;// Gia tri khoi tao 155 doi ra so hex

ET0=1;// Cho phep ngat timer 0

EA=1;// Cho phep ngat toan cuc

TR0=1;// Chay timer 0 bat dau dem so chu ki may}



* Hàm ngắt:

unsigned char dem=0;// Khai bao bien dem de dem tu 1 den 10

unsigned char phantramxung;// Bien chua phan tram xung(0...10)

void timer0(void) interrupt 1 //Ngat timer 0

{ TR0=0;// Dung chay timer 0

TF0=0;// Xoa co, o che do co tu duoc xoa,che do khac can toi cu viet vao day

dem++;

if(dem<phantramxung) P2_0=1;// Neu bien dem < phan tram xung thi dua gia tri 1 ra chan, xung 5V

else P2_0=0;// Neu dem = phan tram xung

if(dem==10) dem=0;// Neu dem du 10 thi gan lai bang 0 de bat dau chu ki moi

TR0=1;// Cho chay timer }

Để có thể thay đổi độ rộng xung thì ta lưu độ rộng xung vào 1 biến, vì hàm ngắt không cho truyền biến vào ta khai báo biến đó là biến toàn cục để có thể gán giá trị ở mọi hàm.

100 uS ngắt 1 lần để xác định đủ chu kì 1000 uS ta cần đếm từ 1 đến 10 ta khai báo biến đếm.

void timer0(void) interrupt 1 //Ngat timer 0

{ TR0=0;// Dung chay timer 0

TF0=0;// Xoa co, o che do co tu duoc xoa,che do khac can toi cu viet vao day

TH0=0xAB;

TL0=0xAB;

….

TR0=1;// Cho chay timer}

Cấu trúc hàm ngắt timer nào cũng phải theo, do chế độ 2 tự động nạp lại nên kô cần gán giá trị cho TH0 và TL0.

Về biến dem sẽ đếm từ 1 đến 10 nếu bằng 10 kết thúc 1 chu kì 10*100 =1000 uS, ta gán lại nó bằng 0 để sang chu kì mới.

if(dem<phantramxung) P2_0=1;// Neu bien dem < phan tram xung thi dua gia tri 1 ra chan, xung 5V

else P2_0=0;// Neu dem = phan tram xung

Câu lệnh này kiểm tra nếu đếm nhỏ hơn phantramxung thì sẽ đưa ra cổng giá trị 1, bằng hoặc lớn hơn sẽ đưa ra giá trị 0. Khi vào chương trình chính ta chỉ việc thay đổi giá trị biến phantramxung thì độ rộng xung sẽ thay đổi.

* Hàm main:

void main(void)

{ khoitaotimer0();

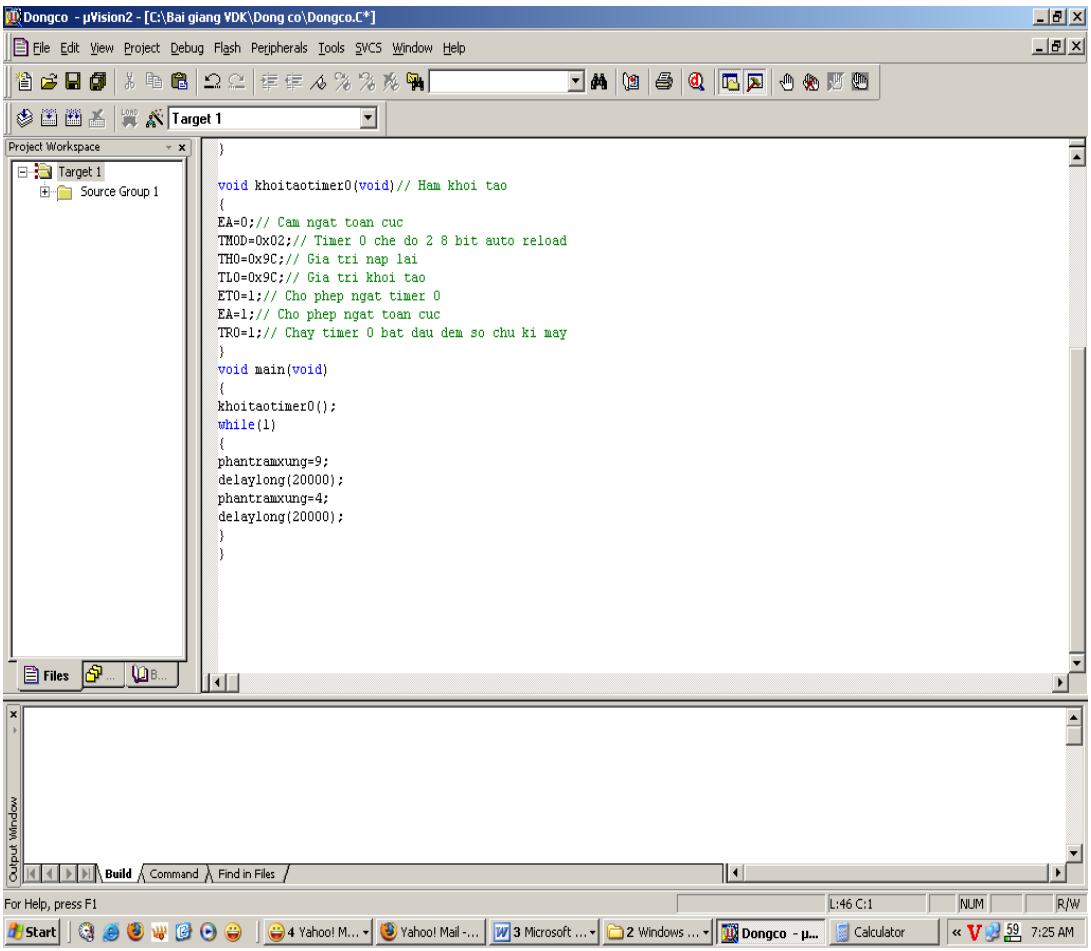
while(1)

{ phantramxung=9;

delaylong(20000);

phantramxung=4;

delaylong(20000); }}



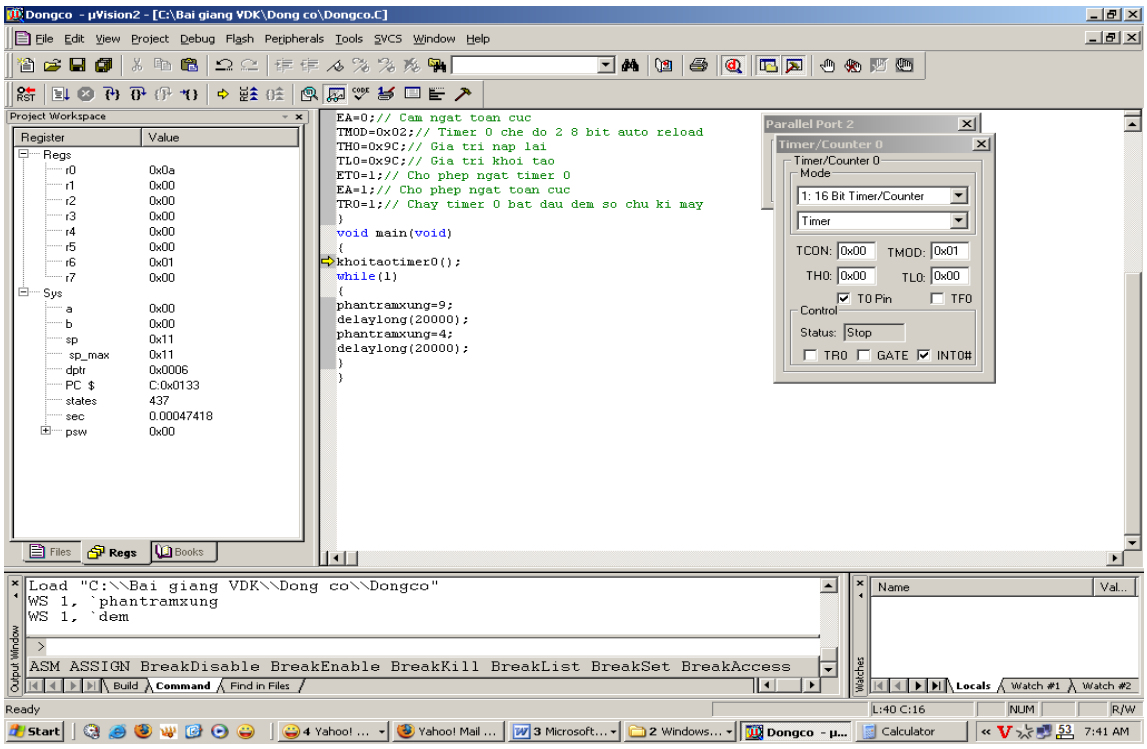
Giả sử khi các bạn gán phantramxung=4; Thì cứ mỗi 100uS ngắt xảy ra 1 lần, và kiểm tra biến đếm. Lần đầu dem=1 <4 nên giá trị P2_0 = 1 mức cao, lần thứ 2 , 200 uS, dem =2<4 P2_0 = 1 mức cao, lần thứ 3, 300uS, dem=3<4, P2_0=1 mức cao, lần thứ 4, 400uS, dem =4 <4 sai, P2_0=0, bắt đầu xuống mức thấp, có xung từ cao xuông thấp, dem = 5<4 sai , P2_0=0 mức thấp, …, dem =10 <4 sai P2_0 mức thấp đủ 1000 uS , 400uS cao, 600uS thấp quay vòng dem=0, ngắt lần thứ 11, dem=1 < 4 , P2_0=1 mức cao, có xung thấp lên cao….

Để PWM 2 chân P2_0 và P3_5, các bạn khai báo thêm 1 biến phantramxung2 và đưa thêm dòng lệnh sau vào hàm ngắt.

if(dem<phantramxung) P3_5=1;// Neu bien dem < phan tram xung thi dua gia tri 1 ra chan, xung 5V

else P3_5=0;// Neu dem = phan tram xung

Chú ý: Thực ra 1 chu kí như ta vừa làm không chính xác 100% là 1Khz, vì ta chưa tính đến độ dài của hàm ngắt,mỗi lần ngắt 100uS, 10 lần là 1000uS đã đủ, còn thời gian thực hiện hàm ngắt nữa, như vậy là chu kì của ta lớn hơn 1000uS, tần số sẽ <1Khz, nhưng thực sự sai số đó không đáng kể. Nếu các bạn muốn chính xác tôi cũng chiều lòng các bạn. Các bạn chạy debug, để thạch anh đúng 12Mhz, quan sat dòng sec xem hàm ngắt diễn ra trong bao nhiêu chu kì máy, khi nạp giá trị cho TL0 và TH0 ta lấy 155 trừ đi giá trị đó được gía trị a gán vào, như vậy a+thời gian thực hiện hàm ngắt đúng đủ 100uS.



Chú ý: Chỉ được chạy với động cơ công suất nhỏ, nếu động cơ công suất lớn phải có mạch điều khiển riêng.

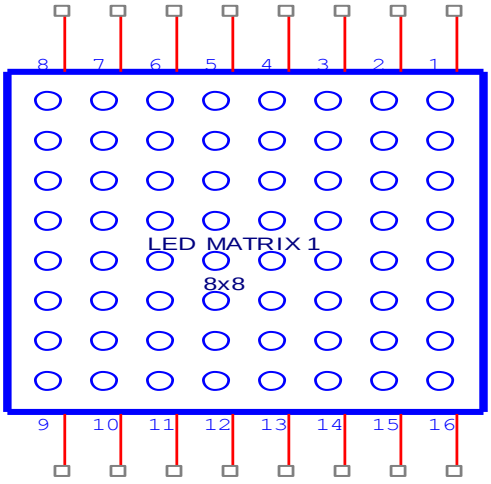
6.2.5. Điều khển Led ma trận

Nhiệm vụ:

Điều khiển Led ma trận 8x8. Hiện thị dòng chữ chạy “MTC”

Chuẩn bị:

Led ma trận 8x8

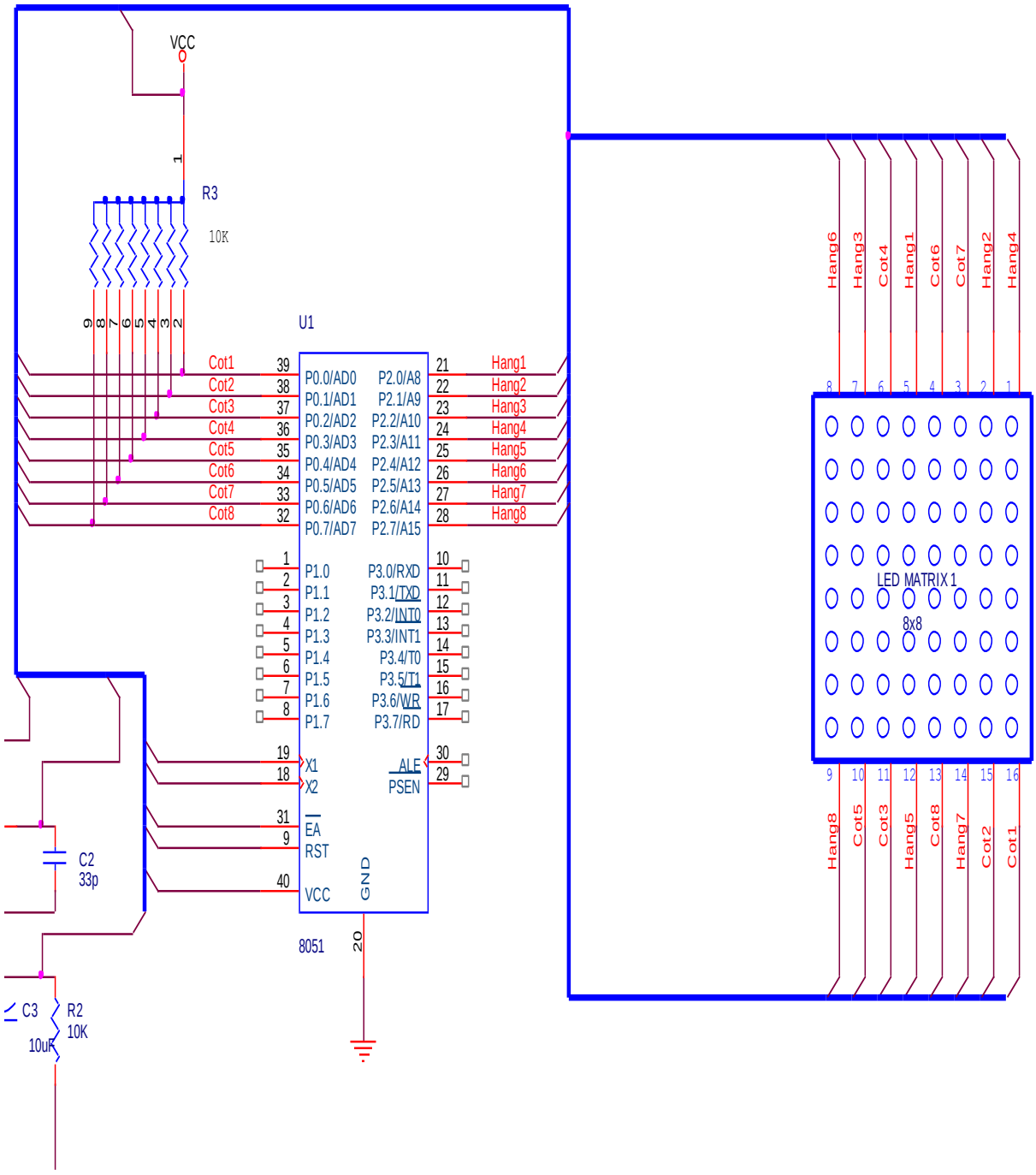


Sơ đồ chân led ma trận 8x8:

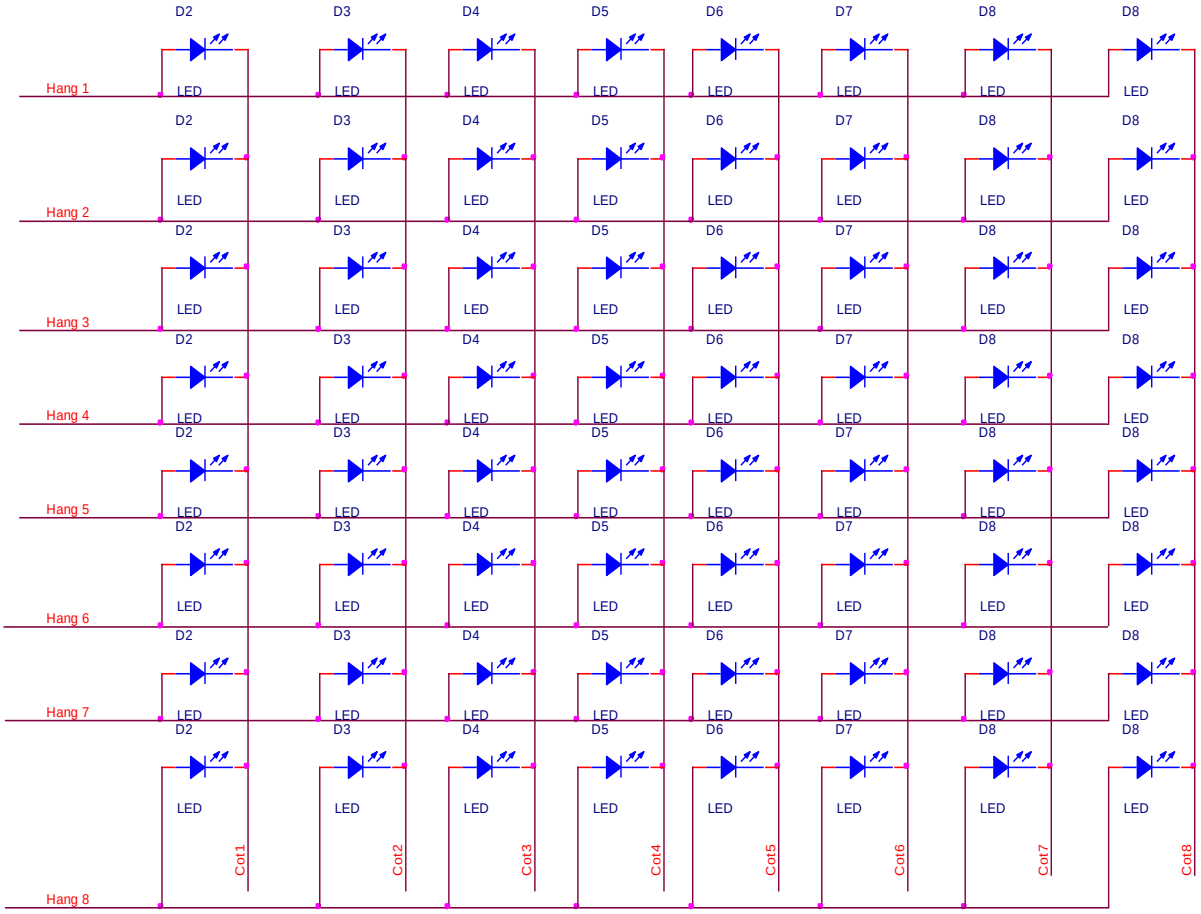
Chân	Cột	Hàng	Chân	Cột	Hàng
1		4	9		8
2		2	10	5	
3	7		11	3	

4	6		12		5
5		1	13	8	
6	4		14		7
7		3	15	2	
8		6	16	6	

a/ Sơ đồ mạch điện:



b/ Nguyên lí hoạt động:



Muốn cho led sáng, cấp điện dương 5V vào hàng, 0V vào cột, dòng 10mA đến 15 mA.

Ví dụ: muốn đèn led ở vị trí 5x4 sáng, ta đưa điện áp cột 4(P0_3) xuống 0V, điện áp hàng 5(P2_5) lên 5V.

Hiển thị chữ: thống kê các điểm sáng thành chữ rồi cho các hàng cột điện áp tương ứng. Có thể dùng công cụ debug để lấy giá trị cổng tương ứng với các led sáng.

Giống như quét bàn phím, đưa điện áp 0V ra từng cột nối với cổng 0. Như vậy sẽ có 8 giá trị: 0xFE, 0xFD, 0xFB, 0xF7, 0xEF, 0xDF, 0xBF, 0x7F phảI đưa vào 1 mảng 8 phần tử, rồi sau đó đưa vào 1 vòng for tăng dần 1 biến để tăng phần tử mảng cot[8].

Với mỗi lần 1 chân cổng 0 xuống 0V ta dùng cổng 2 đưa ra 1 giá trị 8 bít để điều khiển trong 1 cột nhứng đèn nào sáng. Ví dụ muốn hàng 1 và hàng 3 sáng thì hàng 1 và 3 có giá trị 5V còn các hàng khác 0V, ta được giá trị 8 bít sau: 0x05 (1010 000).

Tại mỗi thời điểm chỉ có một số đèn trên 1 cột sáng, nhưng do ta quét 8 cột với tần số nhanh, vì mắt có hiện tượng lưu ảnh nên ta thấy trong 1 thời điểm ta thấy toàn bộ kí tự. Với 8 cột lần lượt bằng 0V ta phảI đưa ra tương ứng 8 giá trị 8 bit ra cổng 2, do đó ta fảI lưu 8 giá trị đó vào 1 mảng 8 kí tự_ kytu1[8], ta sẽ viết các ký tự trên 7 cột. Để mỗi kí tự sẽ cách nhau 1 cột không sáng. Ta khai báo mảng kytu1[9] có 9 phần tử và phần tử đầu tiên có giá trị đẩy ra cổng 2 là 0x00 để tắt toàn bộ cột đó.

Quá trình điều khiển hiển thị như sau:

Cột 1, hàng 1, cột 2 hàng 2, …, cột 8 , hàng 8.

Để làm chữ chạy:

Thêm 1 biến vào để điều khiển thứ tự hiển thị hàng.

Hiện 1 chữ trên led như trên đã đưa ra:

Cột 1, hàng 1, cột 2 hàng 2, …, cột 8 , hàng 8.

Muốn chữ đó dịch chuyển sang tráI ta hiển thị như sau:

Cột 1, hàng 2, cột 2 hàng 3, …, cột 7, hàng 8,cột 8 , hàng 1 ký tự sau.

Cột 1, hàng 3, cột 2 hàng 4, …, cột 7 hàng 1 ký tự sau,cột 8 , hàng 2 ký tự sau.

6.2.6. Bài tập mở rộng

Theo các chủ đề thực tiễn với ý tưởng sáng tạo của giáo viên và học sinh.

Chủ đề 1: Điều khiển hệ thống tín hiệu, cảnh báo, bảo vệ.

Chủ đề 2: Điều khiển cơ cấu kiểm tra, giám sát, bảo mật.

Chủ đề 3: Điều khiển cơ cấu nâng hạ.

Chủ đề 4: Điều khiển hệ thống quảng cáo, bảng điện tử

Chủ đề 5: Điều khiển cơ cấu truyền chuyển động cơ khí.

TÀI LIỆU THAM KHẢO

[1]. I. Scott Mackenzie, The 8051 Microcontroller

[2]. INTEL, The MCS*51 Microcontroller Family User's Manuel, 1994.

[3]. ATMEL, The AT89 Family of Microcontrollers, 2003.

[4]. SIEMENS, Microcomputer Components – SAB80C515 8 bit Single-chip Microcontroller Family, 1995.

[5]. Walter H. Buchbaum. Sc.D, Microprocessor and IC families.

[6]. HPI Fachbuchreihen Pflaum Verlag Munchen, Mikrocompute Lehrbuch.

[7]. Rev 5 - Paul Stoffregen, 8951 Development Boad,

[8]. Văn Thế Minh, Kỹ thuật Vi xử lý, NXB GD - 1997.

[9]. Đỗ Xuân Tiến, Kỹ thuật VXL & lập trình ASSEMBLY cho hệ VXL, NXB KH&KT - 2001.

[10]. Tống văn On, Họ vi điều khiển, Đại học Bách khoa TP.HCM