

TRƯỜNG ĐẠI HỌC LẠC HỒNG  
KHOA CƠ ĐIỆN  
--☞☞☞--

## BÁO CÁO NGHIÊN CỨU KHOA HỌC

ĐỀ TÀI:

***ĐIỀU KHIỂN THIẾT BỊ BẰNG GIỌNG NÓI  
TRUYỀN TỪ XA***



GVHD: Ths Nguyễn Vũ Quỳnh  
SVTH: Phạm Ngọc Đăng Khoa  
Lớp: 05 CĐT1

### LỜI CẢM ƠN

Sau hơn một năm tìm hiểu và thực hiện thì đề tài: “ĐIỀU KHIỂN THIẾT BỊ BẰNG GIỌNG NÓI TRUYỀN TỪ XA” đã thu được những thành công bước đầu trong cuộc sống và trong điều khiển các thiết bị tự động hóa. Trong thời gian đầu thử nghiệm đề tài đã hoạt động một cách tương đối ổn định và nhận được sự đánh giá cao về khả năng sáng tạo, cũng như cách áp dụng khoa học kỹ thuật tiên tiến vào phục vụ nhu cầu điều khiển trong công nghiệp và trong cuộc sống của con người.

Trong quá trình thực hiện, đề tài nhận được sự hướng dẫn và giúp đỡ của thầy thạc sĩ Nguyễn Vũ Quỳnh, câu lạc bộ Tự Động Hóa, và tất cả các thầy cô khoa Cơ Điện trường đại học LẠC HỒNG. Thành công của đề tài cũng là lời cảm ơn đến các cá nhân và câu lạc bộ đã giúp đỡ, hướng dẫn em trong suốt quá trình thiết kế và thi công hệ thống.

Vì là lần đầu tiên khai thác một lĩnh vực còn khá mới mẻ, nên mặc dù em đã bỏ ra rất nhiều tâm huyết, thời gian, và công sức, nhưng các chắc sẽ không tránh khỏi những thiếu sót, những hạn chế khi áp dụng vào thực tiễn cuộc sống hiện nay. Hy vọng rằng những vấn đề còn hạn chế trong đề tài sẽ nhận được nhiều ý kiến đóng góp chân thành của các cá nhân, tổ chức trong trường đại học LẠC HỒNG và các bạn đọc gần xa.

# MỤC LỤC

<b>PHẦN A</b>	<b><i>LÝ THUYẾT</i></b>	<b>Số trang</b>
<b><u>CHƯƠNG 1:</u></b>	<b>GIỚI THIỆU CHI TIẾT BỘ PHẬN XỬ LÝ GIỌNG NÓI</b>	<b>12</b>
1.1	Giới thiệu nguyên lý IC HM2007	12
1.2	Giới thiệu IC nhớ SRAM 6264	17
<b><u>CHƯƠNG 2:</u></b>	<b>LÝ THUYẾT MẠCH ĐIỀU KHIỂN TỪ XA</b>	<b>19</b>
2.1	Chi tiết về chip AVR Atmega8.	19
2.2	Cấu trúc ngắt của Atmega8.	29
2.3	Các bộ phận ngoại vi khác.	34
2.4	Hệ thống xung clock và lập trình bộ nhớ on – chip.	38
<b><u>CHƯƠNG 3:</u></b>	<b>NGÔN NGỮ C CHO AVR</b>	<b>39</b>
3.1	Khái niệm.	39
3.2	Tóm tắt cấu trúc điều khiển.	45
3.3	Chẳng hợp ngữ vào trong chương trình C	49
3.4	Tổ chức bộ nhớ SRAM	50
3.5	Phần mềm lập trình cho bộ điều khiển từ xa AVR Atmega8	51
3.6	Phương pháp và phần mềm nạp cho Atmega8	54
<b>PHẦN B:</b>	<b><i>THIẾT KẾ - THI CÔNG</i></b>	
<b><u>CHƯƠNG 4:</u></b>	<b>THIẾT KẾ MẠCH ĐIỀU KHIỂN BẰNG GIỌNG NÓI</b>	<b>64</b>
4.1	Sơ đồ nguyên lý mạch điều khiển tín hiệu giọng nói.	64
4.2	Các board mạch IC HM2007 đã thực hiện thử nghiệm.	67
<b><u>CHƯƠNG 5:</u></b>	<b>THIẾT KẾ MẠCH ĐIỀU KHIỂN TỪ XA</b>	<b>69</b>
5.1	Sơ đồ nguyên lý mạch điều khiển từ xa.	69
5.2	Sơ đồ thiết kế mạch in và thi công.	70
5.3	Hình ảnh thực tế bộ Atmega8 của thiết bị.	70
<b><u>CHƯƠNG 6:</u></b>	<b>THIẾT KẾ CÁC MODUL NGÕ RA CỦA SẢN PHẨM</b>	<b>71</b>
6.1	Mục đích thiết kế các modul ngõ ra.	71
6.2	Hình ảnh thực tế thiết kế và board mạch ngõ ra.	71
<b><u>CHƯƠNG 7:</u></b>	<b>THIẾT KẾ MẪU VỎ HỘP BÊN NGOÀI CHO THIẾT BỊ</b>	<b>74</b>
7.1	Ý tưởng thiết kế.	74
7.2	Sản phẩm hoàn chỉnh trên phần mềm.	75

**PHẦN C: SẢN PHẨM**

- Hệ thống điều khiển robot sử dụng modul 24VDC. 77
- Bộ điều khiển thiết bị 220VAC bằng giọng nói truyền từ xa. 78
- Khả năng ứng dụng, thành quả bước đầu của đề tài. 79

**KẾT LUẬN – KIẾN NGHỊ**

- Kết luận. 80
- Những khó khăn trong quá trình thực hiện đề tài. 80
- Ưu điểm, khuyết điểm cần cải tiến của thiết bị. 82
- Kiến nghị. 82

**DANH MỤC TÀI LIỆU THAM KHẢO 83****PHỤ LỤC**

- Hình ảnh cải tiến board mạch chủ của thiết bị.
- Chương trình chính lập trình cho bộ điều khiển từ xa.

**DANH MỤC HÌNH ẢNH**

<b>Số thứ tự</b>	<b>Chú thích hình ảnh</b>	<b>Số trang</b>
1	Hình A. Tổng quan hệ thống điều khiển	9
2	Hình 1.1 Tổng quan IC HM 2007	12
3	Hình 1.2 Sơ đồ chân các loại IC HM 2007	13
4	Hình 1.3 Bàn phím ma trận	15
5	Hình 1.4 Bản vẽ mạch hiển thị	15
6	Hình 1.5 Sơ đồ khối SRAM 6264	17
7	Hình 1.6 Cấu tạo bên trong SRAM 6264	18
8	Hình 2.1 Hình ảnh các loại AVR	20
9	Hình 2.2 Sơ đồ khối cấu trúc vi điều khiển AVR	20
10	Hình 2.3 Tổng quan chế độ hoạt động Boot loader	21
11	Hình 2.4 Bản đồ bộ nhớ ATmega8	23
12	Hình 2.5 Sơ đồ bộ định thời 1	25
13	Hình 2.6 Sơ đồ ngõ ra khối	27
14	Hình 2.7 Sơ đồ khối bộ định thời 0	27
15	Hình 2.8 Sơ đồ khối bộ định thời 2	28
16	Hình 2.9 Bảng vector ngắt của Atmega8	30,31
17	Hình 2.10 Các ngắt lồng nhau	31
18	Hình 2.11 Bảng điều khiển kiểu bắt mẫu ngắt	32
19	Hình 2.12 Sơ đồ giản lược của bộ so sánh tương tự	34
20	Hình 2.13 Sơ đồ khối đơn giản bộ ADC	35
21	Hình 2.14 Sơ đồ ngõ vào vi sai	36
22	Hình 2.15 Sơ đồ khối bộ USART	37
23	Hình 2.16 Sơ đồ hệ thống xung clock cho Atmega8	38
24	Hình 3.1 Chương trình lập trình Atmega8	51
25	Hình 3.2 Giao diện lập trình của phần mềm CodeVision	51
26	Hình 3.3 Cách tạo một project trên CodeVision	52
27	Hình 3.4 Các bước thực hiện	52
28	Hình 3.5 Các bước thực hiện	52
29	Hình 3.6 Cách chọn loại AVR	53
30	Hình 3.7 Các bước thực hiện	53
31	Hình 4.1 Sơ đồ nguyên lý mạch xử lý giọng nói	64
32	Hình 4.2 Sơ đồ nguyên lý IC HM 2007 trong Capture	65
33	Hình 4.3 Board HM 2007 (lần 1)	66
34	Hình 4.4 Board HM 2007 (lần 2)	66
35	Hình 4.5 Board 1 lớp thiết kế thử nghiệm	67
36	Hình 4.6 Board mạch 2 lớp thực tế	67
37	Hình 4.7 Board cho sản phẩm hoàn chỉnh	68
38	Hình 5.1 Sơ đồ mạch Atmega8 trên Capture	69
39	Hình 5.2 Sơ đồ mạch in Atmega8 trên layout	70
40	Hình 5.3 Mạch thực tế	70
41	Hình 5.4 Bộ thu (phát) từ xa của thiết bị	70
42	Hình 6.1 Bản thiết kế 1 modul ngõ ra 220VAC	71

43	Hình 6.2 Sơ đồ mạch in modul 220VAC với 6 ngõ ra	72
44	Hình 6.3 Sơ đồ mạch in modul 220VAC với 4 ngõ ra	72
45	Hình 6.4 Modul ngõ ra 24VDC thực tế	73
46	Hình 6.5 Board 2 lớp của Modul 220VAC (với 6 ngõ ra)	73
47	Hình 7.1 Thiết kế cơ khí khung vỏ mạch điều khiển (NX5)	74
48	Hình 7.2 Thiết kế cơ khí modul mạch động lực (NX5)	75
49	Hình 7.3 Sản phẩm hoàn chỉnh trên thiết kế	75
50	Hình B. Điều khiển robot bằng giọng nói	77
51	Hình C. Bộ điều khiển giọng nói và modul 220VAC	78

## **DANH MỤC CÁC TỪ VIẾT TẮT**

CMOS: Complementary Metal-Oxide-Semiconductor (một thuật ngữ chỉ một loại công nghệ dùng chế tạo vi mạch tích hợp)

MPS: Material Product System (Modul sản xuất linh hoạt)

NX5: Phần mềm thiết kế cơ khí Unifraphic

ISR : Interrupt Service Routine (trình phục vụ ngắt)

INT : Interrupt (trình phục vụ ngắt)

RF : Radio Frequency (một dạng sóng tuyền trên AVR)

PWM: Pulse Width Modulation (kênh điều chế độ rộng xung)

TTL : Transistor–transistor logic (thuật ngữ chỉ công nghệ chế tạo vi mạch)

USART: Universal Synchronous and Asynchronous serial Receiver and Transmitter  
(bộ truyền dữ liệu nối tiếp)

## LỜI MỞ ĐẦU

### 1. GIỚI THIỆU

Khoảng thời gian từ năm 2001 đến nay được xem là thời gian các công ty, doanh nghiệp trong và ngoài nước áp dụng nhiều tiến bộ khoa học kỹ thuật vào các ngành công nghiệp chủ chốt của Việt Nam, các dây chuyền công nghệ mới lần lượt ra đời nhằm đơn giản hóa quá trình sản xuất, máy móc hiện đại đã bắt đầu làm việc thay thế con người trong nhiều lĩnh vực sản xuất.

Bên cạnh đó các thành tựu khoa học công nghệ tiên tiến cũng đang được ứng dụng phục vụ cho cuộc sống của con người chúng ta. Hàng loạt các sản phẩm tự động hóa tiên tiến được được phát minh và bán rộng rãi trên thị trường như: robot hút bụi trên sàn phẳng do Nhật sản xuất, máy giặt đa năng, máy rửa chén tự động, thiết bị giám sát nhà qua internet...

Đối với nước ngoài thì việc điều khiển bằng giọng nói đã được nghiên cứu và chế tạo để ứng dụng vào đời sống và sản xuất cũng chỉ mới ra đời trong vài năm trở lại đây. Như ở Mỹ đã được ứng dụng để điều khiển robotcam trong y khoa. Riêng ở nước ta lĩnh vực này còn khá mới mẻ. Do đó chúng ta cần có sự đầu tư để nghiên cứu theo kịp công nghệ mới này để phục vụ trực tiếp cho công việc giảng dạy tại trường nhằm giúp sinh viên hiểu rõ hơn về lý thuyết, tạo điều kiện cho sinh viên có những ý tưởng mới trên những nền tảng đã có sẵn.

Thấy được khả năng phát triển và nhu cầu tìm hiểu về điều khiển bằng giọng nói của chính bản thân và của những người yêu thích mong muốn được sử dụng dịch vụ này, tôi đã bắt tay vào thực hiện nghiên cứu đề tài: “Điều khiển thiết bị bằng giọng nói truyền từ xa”

### 2. TẦM QUAN TRỌNG

Ở Việt Nam việc ứng dụng công nghệ tiên tiến trên thế giới còn chậm phát triển, quá trình đưa công nghệ mới vào phục vụ đời sống, sản xuất gặp nhiều khó khăn. Tận dụng những ic đã nhập sẵn và ic chuyên dụng do nước ngoài sản xuất để thiết kế thành sản phẩm cụ thể là một nhu cầu cần thiết cho việc giảng dạy trong trường học, trong cuộc sống và từ đó phát triển cao hơn để ứng dụng trong các lĩnh vực điều khiển phức tạp[1]. Đề tài: **“ĐIỀU KHIỂN THIẾT BỊ BẰNG GIỌNG NÓI TRUYỀN TỪ XA”** được tìm hiểu và thực hiện nhằm đưa con người tiến gần hơn tới công nghệ, và mở ra một hướng đi mới cho việc nghiên cứu. Điều quan trọng hơn hết là các vấn đề liên quan tới đề tài, nguyên lý hoạt động của mạch xử lý giọng nói, mạch truyền từ xa sử dụng chip AVR Atmega 8, các modul ngõ ra tích hợp, và cách lập trình hệ thống sẽ được giới thiệu trong đề tài này. Nó sẽ là nguồn thông tin hữu ích cho những ai muốn tìm hiểu và phát triển trong lĩnh vực này, nhằm mở ra một hướng đi mới cho công nghệ điều khiển tự động hóa.



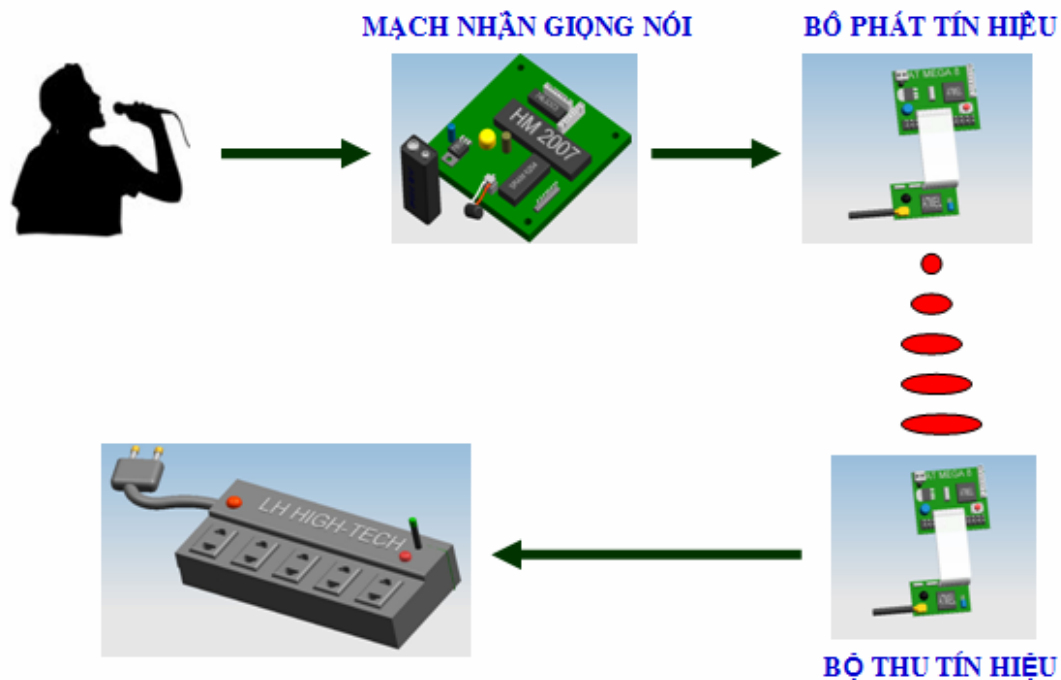
### 3. MỤC ĐÍCH NGHIÊN CỨU

Đề tài: “ĐIỀU KHIỂN THIẾT BỊ BẰNG GIỌNG NÓI TRUYỀN TỪ XA” được thực hiện nhằm tạo ra một hệ thống biết tuân theo mệnh lệnh giọng nói của con người chúng ta. Đề tài không dừng lại ở mức tìm hiểu lý thuyết hay hoàn thiện mạch sử dụng ic HM 2007 như một số sinh viên các trường đại học khác đã tìm hiểu trong thời gian trước. Sản phẩm của đề tài trước hết có thể được ứng dụng vào điều khiển các thiết bị tự động hóa như: tay máy công nghiệp, robot tự hành, xy lanh, cảm biến... với modul ngõ ra 24VDC. Đề tài còn được thiết kế mở rộng thêm modul ngõ ra 220VAC để điều khiển các thiết bị điện dân dụng phục vụ cuộc sống như đèn, quạt, máy tính.....

Đặc biệt đề tài được tích hợp công nghệ điều khiển từ xa sử dụng tín hiệu truyền trên sóng RF (Radio Frequency) đã mở ra một hướng phát triển mới cho đề tài. Con người chỉ cần ngồi tại một vị trí cách thiết bị vài trăm mét và điều khiển theo những yêu cầu mà họ mong muốn. Với bộ điều khiển chỉ sử dụng điện áp từ 5VDC - 9VDC nên tránh cho người điều khiển tiếp xúc trực tiếp với các nguồn điện áp cao. Do đó một hướng phát triển mạnh trong cuộc sống của đề tài là thiết lập hệ thống điều khiển giọng nói trong các trường mầm non, tiểu học và phòng trẻ em.

### 4. GIỚI THIỆU TỔNG QUAN

#### TỔNG QUAN HỆ THỐNG ĐIỀU KHIỂN



Hình 1. Tổng quan hệ thống điều khiển

Đề tài “ Điều khiển thiết bị bằng giọng nói từ xa” bao gồm bốn giai đoạn chính:

+ Thiết kế và thi công mạch nhận dạng và xử lý giọng nói sử dụng IC chuyên dụng HM 2007. Đây là một giai đoạn mang tính kiên trì và sáng tạo trong quá trình thiết kế để cho hệ thống hoạt động ổn định. Vì mục tiêu đạt đến của đề tài là thực hiện một sản phẩm hoàn chỉnh, có thể sử dụng ngay trên thị trường nên yếu tố mỹ quan và chất lượng được đặt lên hàng đầu.

+ Hoàn thành kết nối thêm thiết bị điều khiển từ xa, giao tiếp giữa bộ phận điều khiển và các modul chấp hành. Hệ thống sẽ được truyền từ xa bằng cách lập trình giao tiếp, đưa tín hiệu từ bộ phát đến bộ thu thông qua ngôn ngữ C. Chip vi xử lý dán ATMEGA8 sẽ được sử dụng chủ đạo trong hệ thống truyền từ xa. Đây là một giai đoạn quan trọng và mang tính thiết yếu của đề tài. Sóng RF sẽ bảo đảm việc truyền và nhận dữ liệu một cách đơn giản hơn các loại thiết bị sử dụng giao tiếp qua internet.

+ Thiết kế các modul ngõ ra nhận tín hiệu từ bộ phát, tín hiệu sẽ được kích bởi điện 5VDC và đưa ra các thiết bị sử dụng điện 5VDC, 24VDC và 220VAC. Như vậy hệ thống sẽ bao gồm 3 loại modul ngõ ra để phục vụ mọi nhu cầu điều khiển của các thiết bị tự động hóa đang có trên thị trường. Ở giai đoạn này Modul ngõ ra sử dụng điện 220VAC được xem là có ứng dụng thân thiện nhất với cuộc sống con người, modul này sẽ giúp con người có thể điều khiển các thiết bị điện trong nhà, hay ở công sở.

+ Thiết kế bản vẽ cơ khí, và gia công vỏ hộp cho toàn bộ thiết bị, giai đoạn cuối cùng này đi thiên về khả năng sáng tạo mẫu mã, thiết kế sản phẩm bắt mắt cho người tiêu dùng. Đòi hỏi người thực hiện đề tài cần có kiến thức về cơ khí, có khả năng vẽ trên các phần mềm 3D như Auto CAD, NX5, Catia.... Theo xu thế công nghệ hiện nay, phần mềm vẽ Unigrafc (NX5) đang là một phần mềm mạnh trong thiết kế mẫu mã, được nhiều công ty lớn như SYM, Pepsico, Sanko Mod... sử dụng thiết kế mẫu mã các loại xe máy, mẫu chai nước giải khát, và mẫu điện thoại di động. Do là một sinh viên ngành Cơ Điện Tử em đã ứng dụng phần mềm NX5 vào thiết kế mẫu mã cho thiết bị một cách hoàn chỉnh.

*Bốn giai đoạn để hoàn thành đề tài, mỗi giai đoạn có một khó khăn riêng, đề tài được lên ý tưởng thiết kế từ cuối năm 2008, và thực hiện tới đầu tháng 11/2009 mới đem lại những thành quả bước đầu của sản phẩm. Ở giai đoạn đầu, mạch điều khiển thiết bị bằng giọng nói sau khi hoàn thành đã nhận được nhiều đơn đặt hàng của các cá nhân, câu lạc bộ tự động hóa, họ là những người quan tâm đến đến khả năng nhận giọng nói của IC HM2007, đây là thành công bước đầu, của đề tài. Các Modul mạch điều khiển bằng giọng nói nhận được nhiều sự quan tâm của những sinh viên điện – điện tử tại các trường đại học kỹ thuật lớn ở Việt Nam, bây giờ sinh viên có thể mua thiết bị và hoàn thành các ý tưởng liên quan tới “xử lý giọng nói” một cách dễ dàng, với giá cả thấp hơn rất nhiều so với các sản phẩm liên quan chỉ bán ở thị trường Mỹ.*

**PHẦN A:**

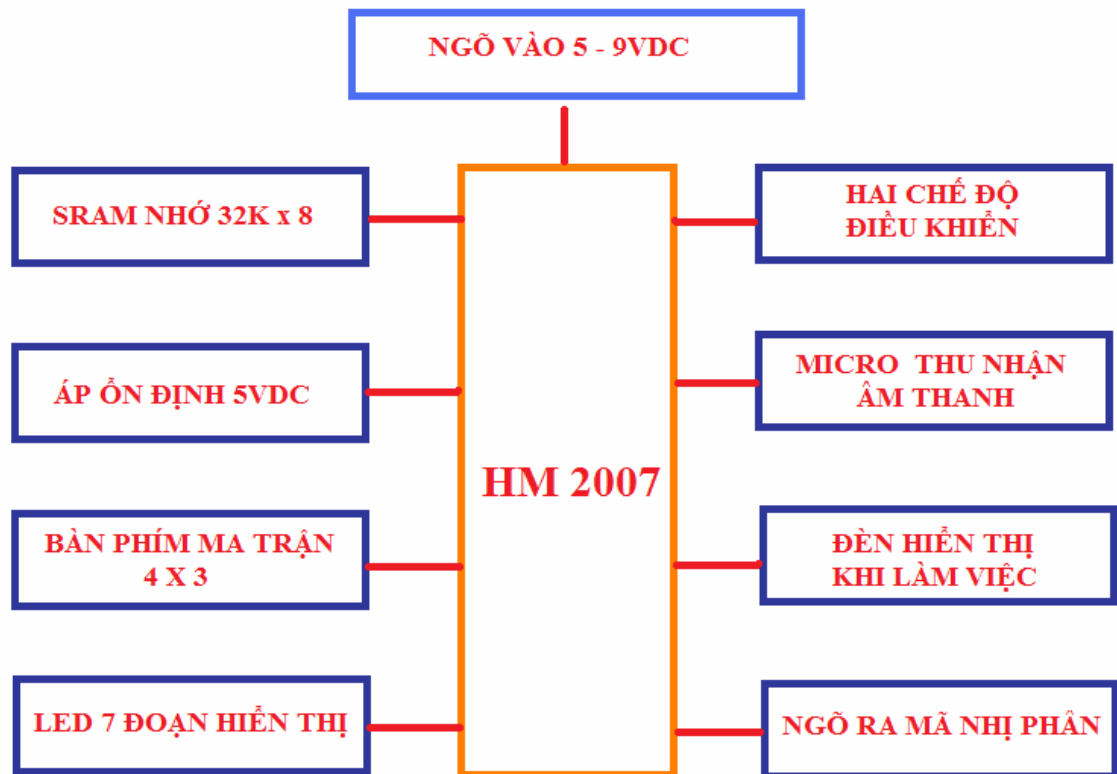
**LÝ THUYẾT**

## CHƯƠNG 1

### GIỚI THIỆU CHI TIẾT BỘ PHẬN XỬ LÝ GIỌNG NÓI

#### 1.1 Giới thiệu nguyên lý IC HM2007 [2]

IC HM 2007 là một thiết bị đơn chip CMOS, xử lý giọng nói dưới dạng mạch LSI điều chế tín hiệu tương tự, điều chế phổ âm, nhận lệnh và điều khiển chức năng các hệ thống. Theo tiêu chuẩn, thì ic HM2007 có thể nhận tới 40 lệnh, việc truyền và nhận lệnh được thực hiện bằng micro đưa tín hiệu vào, cùng một bàn phím, một ic nhớ SRAM và nhiều bộ phận khác. Từ đây tín hiệu được xử lý và xây dựng thành một hệ thống thông minh trong việc nhận diện giọng nói.



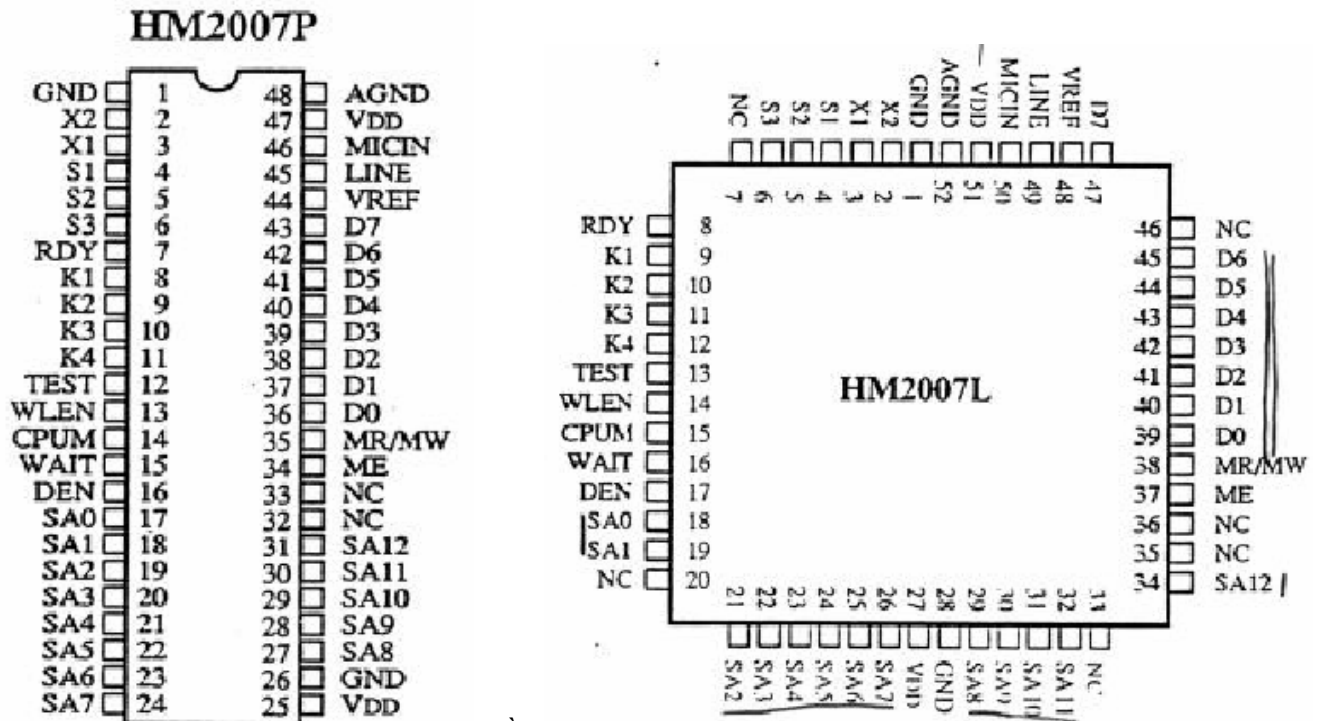
Hình 1.1 Tổng quan IC HM 2007 [2]

##### 1.1.1 Đặt tính

- Thiết bị đơn chip nhận biết âm thanh dạng CMOS LSI
- Tiếng nói được nhận vào hệ thống theo một chuẩn riêng biệt.
- IC nhớ SRAM có thể được kết nối trực tiếp.
- Một chip HM 2007 có thể nhận được 40 từ.
- Thời gian tối đa mỗi từ phù hợp mà ic có thể xử lý là 1.92 giây.
- Kết cấu phức tạp.
- Một micro đi kèm thiết bị.
- Có hai chế độ sử dụng: chế độ động thường, và chế độ CPU.

- Thời gian đáp ứng: chưa tới 300ms
- Nguồn cấp 5VDC
- Bao gồm hai loại: loại thường 48 chân, và loại dán 52 chân.

### 1.1.2 Sơ đồ chân 2 loại IC HM2007



Hình 1.2 Sơ đồ chân các loại IC HM2007

### 1.1.3 Chức năng các chân của IC HM 2007 (loại 48 chân)

Tên chân	Số chân	Chức năng
WAIT	15	Tín hiệu điều khiển ngõ vào, hoạt động ở mức thấp. Khi chân này ở mức thấp thì ic HM2007 ở chế độ nghỉ, không chấp nhận bất cứ âm thanh nào đưa vào xử lý Khi chân Wait ở mức cao thì ta có thể bắt đầu huấn luyện ic nhận biết giọng nói thu vào.
DEN	16	Khả năng nhận tín hiệu Khi tín hiệu được đưa vào hoàn tất, chip sẽ bắt đầu xử lý và đưa vào các chân D0 – D7, dữ liệu sẽ được xử lý bởi ic chốt 74LS373.
SA0 , SA1	17 , 18	Bus địa chỉ cho bộ nhớ ngoài
SA2 – SA7 SA8 – SA12	19 – 24 27 - 31	Bus này được dùng như một đường địa chỉ cho bộ nhớ ngoài khi chân Me hoạt động.
V <sub>DD</sub>	25, 47	Chân cấp nguồn (5VDC)
GND	26	Chân nối nguồn âm
NC	32,33	Không kết nối

ME	34	Chân điều khiển bộ nhớ, chân này sẽ gửi tín hiệu sang SRAM và được lưu lại để thực hiện lệnh. (Chân này nối trực tiếp với chân CE của SRAM)
MR/MW	35	Chân thiết lập và phản hồi tín hiệu đến bộ nhớ
D0 – D6	36 - 42	Đường dữ liệu cho bộ nhớ ngoài
D7	43	Được dùng như bus I/O của bộ nhớ khi chân ME tích cực, đây là tín hiệu ngõ vào cho bộ chốt dữ liệu khi chân DEN hoạt động.
Vref	44	Điện áp cấp cho bộ biến đổi tương tự sang số

Line	45	Chân kiểm tra
Micin	46	Chân nối với micro. Được hoạt động kèm theo tụ và điện trở.
AGND	48	Mất tương tự
GND	1	Cấp nguồn âm
X2,X1	2,3	Chân nối với thạch anh 3.58M
S1, S2, S3	4,5,6	Chân nối với bàn phím ở chế độ thường, và là chân đọc ghi dữ liệu ở chế độ xử lý.
RDY	7	Thông báo tín hiệu giọng nói ngõ vào. Khi HM 2007 sẵn sàng nhận âm thì sẽ có một tín hiệu mức thấp gửi đi. Nếu ic không nhận thì gửi tín hiệu mức cao.
K1- K4	8 - 11	Chân nối với bàn phím

#### 1.1.4 Chức năng làm việc

Có hai chế độ hoạt động trong IC HM2007

##### 1.1.4.1 Chế độ thường

Ở chế độ này ic được kết nối với một bàn phím, một SRAM, và các thiết bị ngoại vi để thiết lập một môi trường làm việc bằng giọng nói. SRAM có thể dùng loại dung lượng 8K.

##### *Mở nguồn*

HM2007 hoạt động khi có nguồn cấp vào, khi chân WAIT ở mức thấp thì ic bắt đầu kiểm tra bộ nhớ. Khi chân WAIT ở mức cao, HM2007 sẽ bỏ qua việc kiểm tra bộ nhớ, sẽ bắt đầu xử lý tín hiệu nhận giọng nói.

### ***Thu tín hiệu***

Khi chân WAIT nhận mức cao thì chân RDY được đưa xuống mức thấp và HM2007 sẵn sàng nhận âm vào để kiểm tra giọng nói. Khi có tín hiệu giọng nói đưa vào, chân RDY sẽ lên mức cao và HM2007 bắt đầu làm việc. Đó là lệnh mà người điều khiển cài đặt cho bộ nhớ, kết quả sẽ được hiển thị trên 2 led 7 đoạn. Tín hiệu được xử lý và đưa đến Bus ngõ ra tín hiệu. Tín hiệu xuất ra dưới dạng mã nhị phân. Khi chân WLEN được đưa lên mức cao, độ dài của từ là 1,92s, và nếu chân WLEN ở mức thấp, thì độ dài từ đưa vào là 0,92s.

Khi chân WAIT ở mức thấp, âm ngõ vào sẽ không được nhận cho tới khi chân WAIT trở lại trạng thái mức cao.

### ***Cách sử dụng thiết bị***

- Khi ta muốn xóa các dữ liệu đã được nạp trước đó thì từ bàn phím ma trận ta nhập số 99 rồi nhấn vào nút CLR. Mọi dữ liệu về giọng nói lúc này sẽ bị xóa và ta phải cài đặt lại khi muốn tiếp tục sử dụng, việc cài đặt này khá đơn giản, chỉ cần một vài thao tác là chúng ta có thể cài đặt giọng nói vào một cách dễ dàng.
- Để cài tín hiệu giọng nói, trên bàn phím ta nhấn các giá trị mặc định từ 00-99 rồi nhấn vào nút TRAIN, lúc đó HM 2007 sẽ bắt đầu xử lý để đưa tín hiệu vào. Lúc bắt đầu cài từ, nếu chân WAIT ở mức cao, HM2007 sẽ gửi tín hiệu mức thấp ra chân RDY để báo rằng HM2007 sẵn sàng nhận âm vào. Nếu chân WAIT mức thấp, âm sẽ không được nhận cho tới khi chân WAIT báo mức cao.

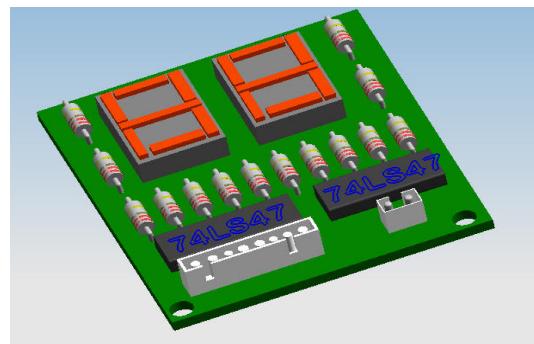
### **Chi tiết các thiết bị đi kèm board mạch chính.**

\* Key board phím ma trận và mạch hiển thị

( Kích thước : 5.7cm x 5.7cm )



Hình 1.3 Bàn phím ma trận [6]



Hình 1.4 Bản vẽ mạch hiển thị

Hai ic 74LS47 sẽ giải mã tín hiệu xuất ra từ HM2007 và hiển thị trạng thái huấn luyện, khi cấp nguồn HM2007 kiểm tra SRAM (Ram tĩnh). Nếu Ram kiểm tra xong, mạch hiển thị "00" trên 2 Led bảy đoạn. Trạng thái "00" báo hiệu mạch đã sẵn sàng và chờ lệnh.

***Một ví dụ chi tiết về cách huấn luyện IC HM2007 [6]***

Cấp nguồn, màn hình hiển thị "00" trên 2 Led bảy đoạn, và Led trên main board sáng  
==> sẵn sàng chờ lệnh.

Nhấn phím 1 ==> màn hình hiển thị "01" và Led tắt. Sau đó nhấn TRN(Training - huấn luyện), và Led sáng trở lại.

Nói vào 'microphone' một từ hoặc 1 cụm từ có độ dài < 0.96 s.

Ví dụ: Đi quét nhà : ngay lập tức màn hình hiển thị "55" (ý nó báo: từ quá dài)  
Làm lại: “Quét nhà” thì ngay lập tức Led trên main board nhấp nháy (chấp nhận), từ  
“Quét nhà” bây giờ được lập trình là "01". Mỗi khi nghe đúng từ này, màn hình sẽ  
hiển thị đúng mã số "01".

Tiếp tục huấn luyện các từ mới: Nhấn "02" rồi nhấn TRN để huấn luyện từ thứ 2. cứ  
thế, cứ thế. Tối đa huấn luyện được 40 từ.

***Cách xóa các từ đã huấn luyện***

Muốn xóa từng từ riêng lẻ trong bộ nhớ ==> nhập số của từ muốn xóa và nhấn CLR  
(clear - xóa).

Muốn xóa tất cả các từ trong bộ nhớ ==> nhập "99" rồi nhấn CLR.

**1.1.4.2 Chế độ điều khiển bởi Vi xử lí ngoài (chế độ CPU)**

Chế độ này bao gồm các chức năng: nhận dạng âm, cài âm vào, báo kết quả, nhận  
và cấp dữ liệu. K –bus được dùng như một dữ liệu nhị phân giữa bộ điều khiển ngoài  
và HM2007. Các chân từ S1 đến S3 xem như là chân điều khiển đọc ghi dữ liệu.

Có ba thanh ghi trong HM2007, một thanh ghi bộ đệm ngõ vào, một thanh ghi  
trạng thái và một thanh ghi bộ đệm ngõ ra. Đầu tiên là thanh ghi chỉ ghi và cuối cùng  
là thanh ghi chỉ đọc. Nếu chân S1 mức cao, dữ liệu đọc từ K-BUS sẽ lấy từ thanh ghi  
bộ đệm ngõ ra. Nếu S1 mức thấp, dữ liệu K – Bus sẽ lấy từ thanh ghi trạng thái. S2 và  
S3 là tín hiệu để điều khiển đọc ghi. Đó là lúc đang đọc, bộ điều khiển ngoài có thể  
lấy dữ liệu từ K – Bus. Chú ý S2 và S3 không thể đồng thời là mức cao và trạng thái  
của S1 sẽ không được nhận trong quá trình ghi. [4]

***Nguyên lí hoạt động ở chế độ CPU***

Khi có nguồn cấp thì HM2007 bắt đầu hoạt động giống như ở chế độ thường và  
sau đó thanh ghi trạng thái sẽ có giá trị 10 để chờ lệnh. Sau khi nhận lệnh Recog, thì  
ic sẽ bắt đầu cho xử lý nhận biết âm. Thiết bị ngoài có thể hiển thị trạng thái của  
HM2007. Khi trạng thái hoạt động chuyển sang 01, và chân WAIT mức thấp,  
HM2007 sẽ trở về trạng thái hoạt động 10 và sau đó sẵn sàng nhận lệnh mới. Khi  
trạng thái hoạt động chuyển sang 01 và chân WAIT mức cao, đó là lúc sẵn sàng nhận  
âm vào và sau đó xử lý nhận biết âm. Khi trạng thái hoạt động trở về 01 một lần nữa,  
thì sau đó việc xử lý nhận biết hoàn tất HM 2007 đang đợi lệnh khác.



Sau khi nhận biết âm. Kết quả nhận biết được đưa vào trong bộ đệm, thiết bị ngoài có thể gửi lệnh RESULT dạng cơ số 10 để lấy kết quả nhận biết. Khi dữ liệu đã được đọc trạng thái hoạt động sẽ trở về 10 và đợi lệnh kế. Sau khi lệnh RESULT được gửi, việc đọc được thực hiện bởi CPU và kết quả được gửi bởi HM2007.

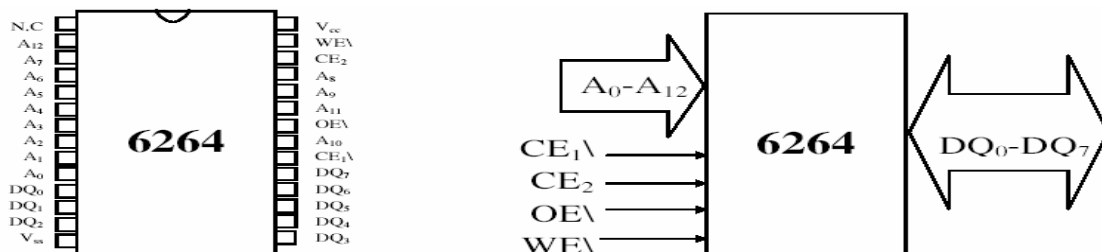
Khi HM2007 nhận mã lệnh TRAIN, ic sẽ cần thêm thông tin để biết vị trí khung được cài đặt. Từ đầu tiên là bốn bit thấp của giá trị vị trí, từ thứ hai là hai bit cao của giá trị vị trí của từ.

Nếu một số vị trí khung từ được chấp nhận và chân WAIT là mức cao, HM2007 bắt đầu xử lý cài đặt khung từ. Nếu chân WAIT mức thấp HM2007 sẽ bỏ qua việc xử lý cài đặt. Sau khi xử lý cài đặt, trạng thái hoạt động sẽ trở về 10 và đợi lệnh kế tiếp. Khi HM2007 nhận mã lệnh UPLOAD, chip cần hai từ để biết vị trí khung từ nơi chưa chứa data. Từ đầu tiên là 4 bit thấp và từ thứ hai là 2 bit cao.

Khi lệnh RESET được đưa vào HM2007 chip sẽ xóa tất cả nhưng khung nhớ trong bộ nhớ.

## 1.2 Giới thiệu IC nhớ SRAM 6264 [4]

IC UM6264 là một ic xử lý nhanh, có dung lượng 65536 bit với 8Kbyte dữ liệu, áp cấp 5VDC, thời gian đáp ứng truy cập khoảng 150ns. Ngõ vào ra dữ liệu được dùng chung, các ngõ ra này tương thích họ TTL. Công suất tiêu tán ở trạng thái chờ rất thấp chỉ khoảng 0.1mW so với khi hoạt động bình thường là 200mW.



CHẾ ĐỘ	WE\	CE <sub>1</sub> \	CE <sub>2</sub>	OE\	NGÕ RA
KHÔNG NHẬN	X	H	X	X	Hi-Z
	X	X	L	X	Hi-Z
NGÕ RA	H	L	H	H	Hi-Z
ĐỌC	H	L	H	L	D <sub>out</sub>
VIẾT	L	L	H	H	D <sub>in</sub>

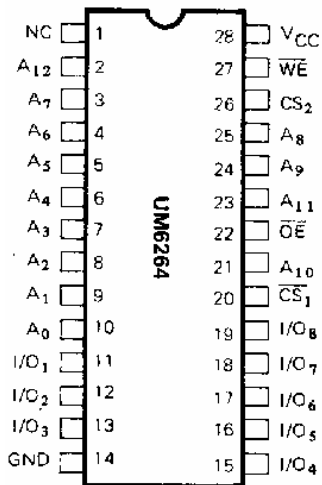
Hình 1.5 Sơ đồ khối SRAM 6264 [2]

IC UM6264 bao gồm các chân:

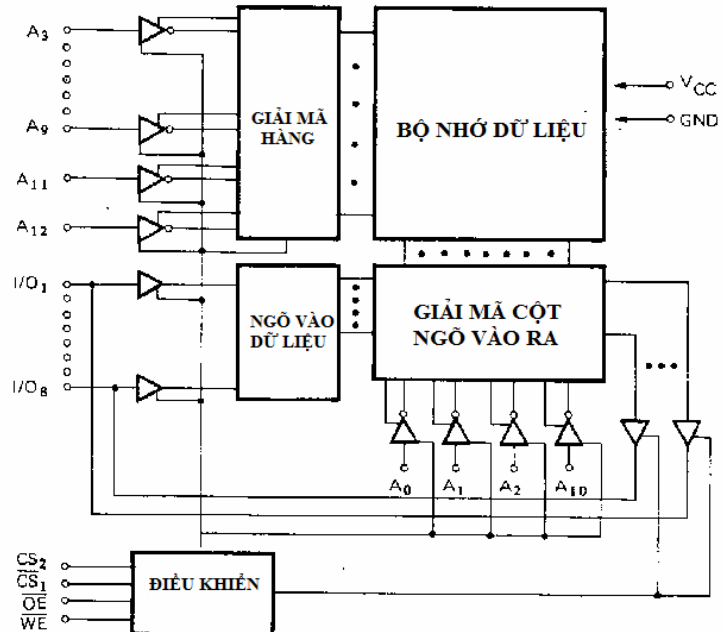
- Các chân nguồn VCC/GND

- Các chân dữ liệu D0 – D7
- Các chân địa chỉ A0 – A12
- Các chân điều khiển là WE, OE, CS1, CS2

SƠ ĐỒ CHÂN



SƠ ĐỒ KHỐI



Hình 1.6 Cấu tạo bên trong SRAM 6264

IC UM6264 là thiết bị lưu trữ dữ liệu quan trọng trong hệ thống, đây là loại chip được sử dụng trong PLC để làm bộ nhớ. Với hệ thống này IC 6264 phải được chọn lựa loại tương thích thì mạch mới hoạt động ổn định lâu dài. IC này có một khuyết điểm là khi chúng ta ngưng cấp nguồn thì dữ liệu tự động sẽ xóa, do đó ta cần có một nguồn pin 3V cấp vào chân Back up dữ liệu của hệ thống.

## CHƯƠNG II

### GIỚI THIỆU LÝ THUYẾT VỀ MẠCH ĐIỀU KHIỂN TỪ XA CỦA THIẾT BỊ

#### 2.1 Chi tiết về chip AVR Atmega8 [1]

##### 2.1.1 *Tổng quan*

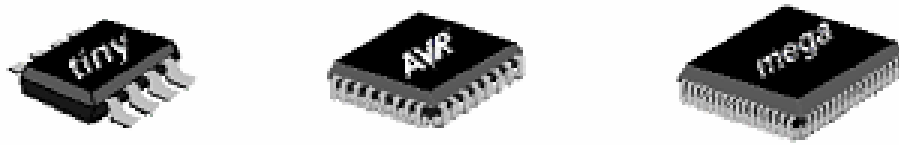
##### Những tính năng chính của Atmega8

- Rom: 8Kbyte
- Sram: 4Kbytes
- EEPROM: 4Kbytes
- 64 thanh ghi I/O
- 160 thanh ghi vào ra mở rộng
- 32 thanh ghi đa mục đích
- 2 bộ định thời 8 bit (0,2)
- 2 bộ định thời 16 bit (1,3)
- Bộ định thời watchdog
- Bộ dao động nội RC tần số 1MHz, 2MHz, 4MHz, 8MHz
- ADC 8 kênh với độ phân giải 10 bit (Ổ dòng Xmega lên tới 12 bit)
- 2 kênh PWM 8 bit
- 6 kênh PWM có thể lập trình thay đổi độ phân giải từ 2 tới 16 bit
- Bộ so sánh tương tự có thể lựa chọn ngõ vào
- Hai khối USART lập trình được
- Khối truyền nhận nối tiếp SPI
- Khối giao tiếp nối tiếp hai dây TWI
- Hỗ trợ Boot loader
- 6 chế độ tiết kiệm năng lượng
- Lựa chọn tần số hoạt động bằng phần mềm
- Đóng gói 64 chân kiểu TQFP
- Tần số tối đa 16MHz
- Điện thế: 4,5V – 5,5V

.....

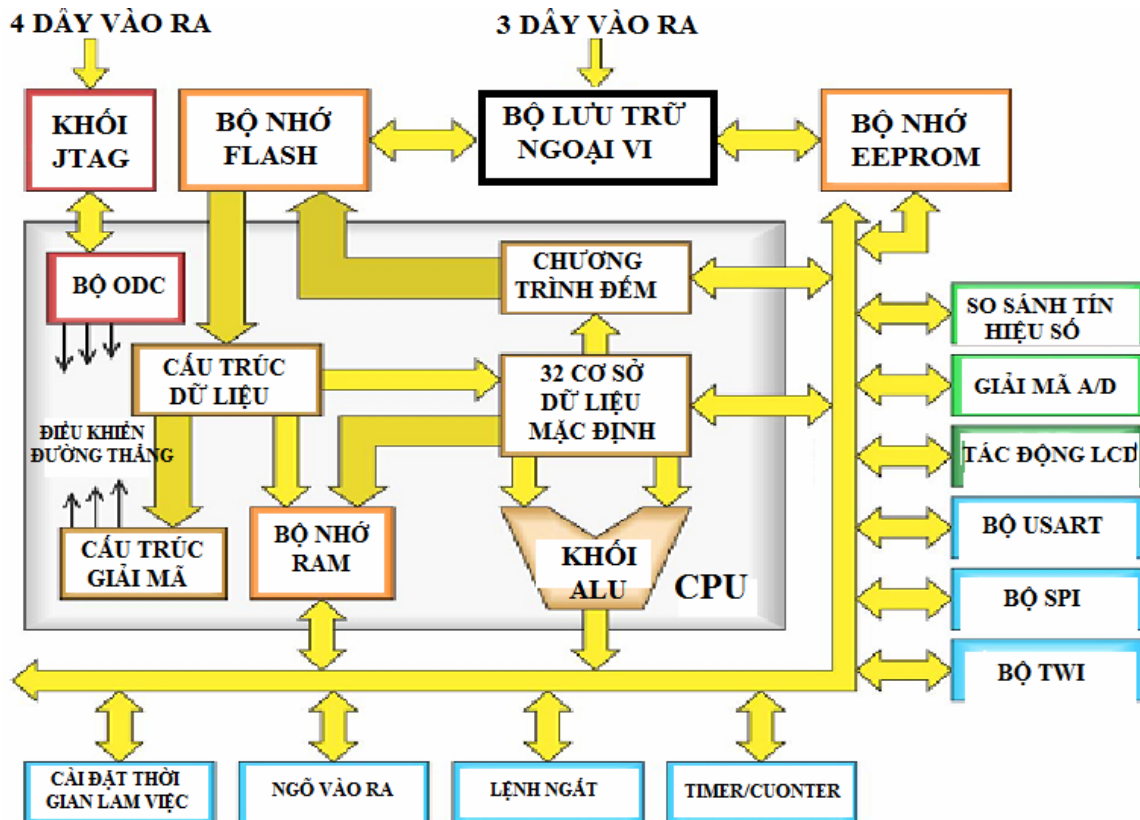
Vì điều khiển AVR do hãng Atmel (Hoa kỳ) sản xuất được giới thiệu lần đầu tiên vào năm 1996, AVR có rất nhiều dòng khác nhau bao gồm dòng Tiny AVR ( như AT tiny 13, AT tiny 22...) có kích thước bộ nhớ nhỏ, ít bộ phận ngoại vi, rồi đến dòng AVR (chẳng hạn AT90S8535, AT90S8515.....) có kích thước bộ nhớ vào loại trung bình và mạnh hơn là dòng MEGA ( như AT mega 32, At mega 128.....) với bộ nhớ có kích thước vài Kbyte đến vài trăm Kb cùng với các bộ ngoại vi đa dạng được tích hợp trên chip, cũng có dòng tích hợp cả bộ LCD trên chip (dòng LCD AVR). Tốc độ của dòng Mega cũng cao hơn so với các dòng khác. Sự khác nhau cơ bản giữa các dòng chính là cấu trúc ngoại vi, còn nhân thì vẫn như nhau. Đặt biệt năm 2008. ATMEL lại tiếp tục cho ra đời dòng AVR mới là XmegaAVR, với những tính năng

mạnh mẽ chưa từng có ở các dòng AVR trước đó. Có thể nói XmegaAVR là dòng MCU 8 bit mạnh nhất hiện nay. [3]



Hình 2.1 Hình ảnh các loại AVR [2]

Cấu trúc cơ bản của vi điều khiển AVR như sau



Hình 2.2 Sơ đồ khối cấu trúc vi điều khiển AVR [4]

## 2.1.2 Cấu trúc bộ nhớ và cổng vào ra

### 2.1.2.1 Cấu trúc bộ nhớ

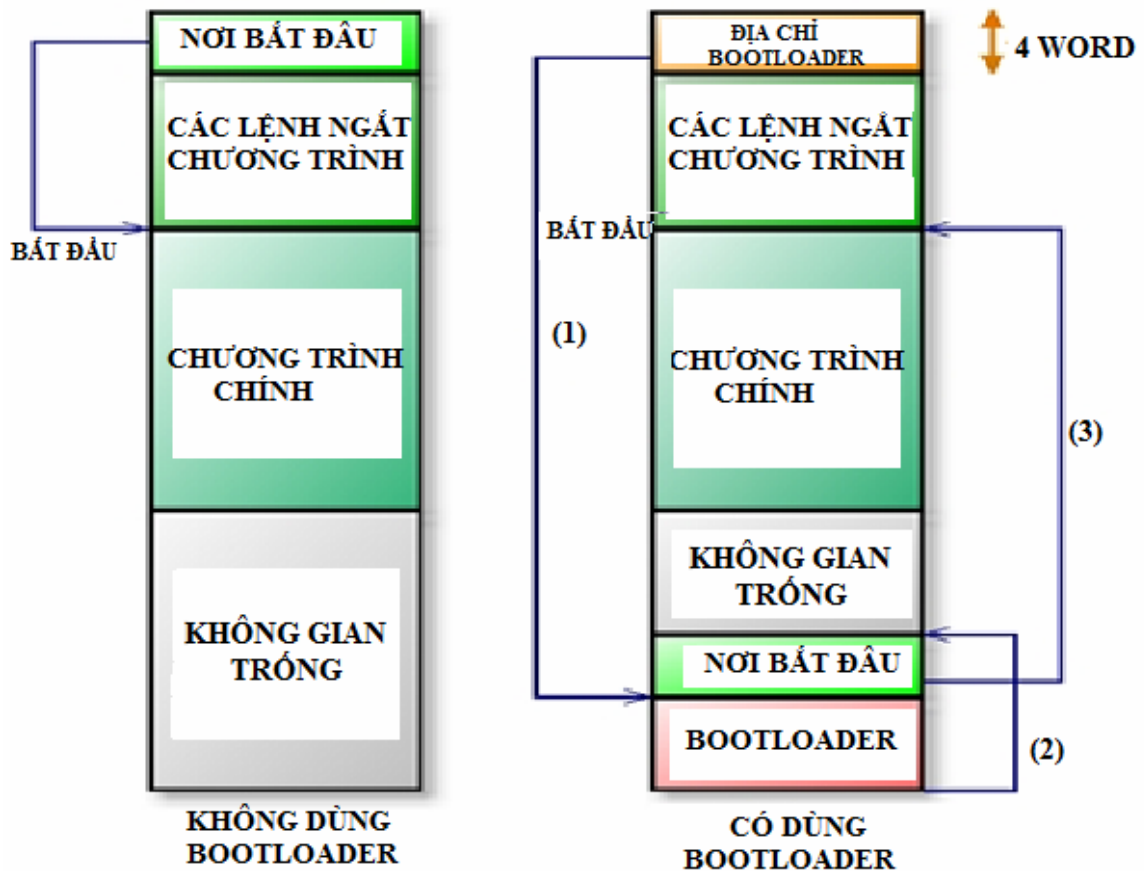
Bộ nhớ vi điều khiển AVR có cấu trúc Harvard là cấu trúc có đường Bus riêng cho bộ nhớ chương trình và bộ nhớ dữ liệu. Bộ nhớ AVR được chia làm 2 phần chính: Bộ nhớ chương trình (program memory) và bộ nhớ dữ liệu (Data memory)

- Bộ nhớ chương trình:

Bộ nhớ chương trình của AVR là bộ nhớ Flash có dung lượng 128K bytes. Bộ nhớ chương trình có độ rộng Bus là 16 bit. Những địa chỉ đầu tiên của bộ nhớ chương trình được dùng trong bảng vectơ ngắt. Đối với Atmega 8 bộ nhớ chương trình có thể chia làm 2 phần: phần boot loader (Boot loader program section) và phần ứng dụng (Application program section).

Phần boot loader chứa chương trình boot loader. Chương trình boot loader là một phần mềm nhỏ nạp trong vi điều khiển và được chạy lúc khởi động. Phần mềm này có thể tải vào trong vi điều khiển chương trình của người sử dụng và sau đó thực thi chương trình này. Mỗi khi reset vi điều khiển CPU sẽ nhảy tới thực thi chương trình boot loader trước, chương trình boot loader sẽ dò xem có chương trình nào cần nạp vào vi điều khiển hay không, nếu có chương trình cần nạp, boot loader sẽ nạp chương trình vào vùng nhớ ứng dụng (Application program section), rồi thực thi chương trình này. Ngược lại, boot loader sẽ chuyển tới chương trình ứng dụng có sẵn trong vùng nhớ ứng dụng để thực thi chương trình này.

Phần ứng dụng (Application program section) là vùng nhớ chứa chương trình ứng dụng của người dùng. Kích thước của phần boot loader và phần ứng dụng có thể tùy chọn.



Hình 2.3 Tổng quan chế độ hoạt động boot loader

Hình trên thể hiện cấu trúc bộ nhớ chương trình có sử dụng boot loader và không sử dụng boot loader, khi sử dụng phần boot loader ta thấy 4 word đầu tiên thay vì chỉ thị cho CPU chuyển tới chương trình ứng dụng của người dùng (là chương

trình có nhãn Start) thì chỉ thị CPU nhảy tới phần chương trình boot loader để thực hiện trước rồi mới quay trở lại thực hiện chương trình ứng dụng.

- **Bộ nhớ dữ liệu:** Bộ nhớ dữ liệu của AVR được chia làm hai phần chính là bộ nhớ SRAM và bộ nhớ EEPROM. Tuy cùng là bộ nhớ dữ liệu nhưng hai bộ nhớ này lại tách biệt nhau và được đánh địa chỉ riêng

- **Bộ nhớ SRAM:** có dung lượng 4K bytes, bộ nhớ SRAM có hai chế độ hoạt động là chế độ thông thường và chế độ tương thích với AT mega 8 muốn thiết lập bộ nhớ SRAM hoạt động theo chế độ nào ta sử dụng bit cầu chì M103C.

- **Bộ nhớ EEPROM:** Đây là bộ nhớ dữ liệu có thể ghi xóa ngay trong lúc vi điều khiển đang hoạt động và không bị mất dữ liệu khi nguồn cung cấp bị mất. Với vi điều khiển AT mega8, bộ nhớ EEPROM có kích thước là 4K byte. EEPROM được xem như là một bộ nhớ vào ra được đánh địa chỉ độc lập với SRAM. Để điều khiển vào ra dữ liệu với EEPROM ta sử dụng ba thanh ghi:

#### + Thanh ghi EEAR (EEARL):

Bit	15	14	13	12	11	10	9	8	
	-	-	-	-	EEAR11	EEAR10	EEAR9	EEAR8	EEARH
	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEARL
	7	6	5	4	3	2	1	0	
<b>ĐỌC / VIẾT</b>	R	R	R	R	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
<b>GIÁ TRỊ BẮT ĐẦU</b>	0	0	0	0	X	X	X	X	
	X	X	X	X	X	X	X	X	

Đây là thanh ghi 16 bit lưu giữ địa chỉ các ô nhớ của EEPROM, thanh ghi EEAR được kết hợp từ 2 thanh ghi 8 bit là EEARH và thanh ghi EEARL. Vì bộ nhớ EEPROM của Atmega8 có dung lượng 4Kbyte = 4069 byte =  $2^{12}$  byte nên ta chỉ cần 12 bit của thanh ghi EEAR, 4 bit từ 15-12 được dự trữ ta nên ghi 0 vào các bit dự trữ này.

#### + Thanh ghi EEDR

BIT	7	6	5	4	3	2	1	0	
	MSB							LSB	EEDR
<b>ĐỌC / VIẾT</b>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
<b>GIÁ TRỊ BẮT ĐẦU</b>	0	0	0	0	0	0	0	0	

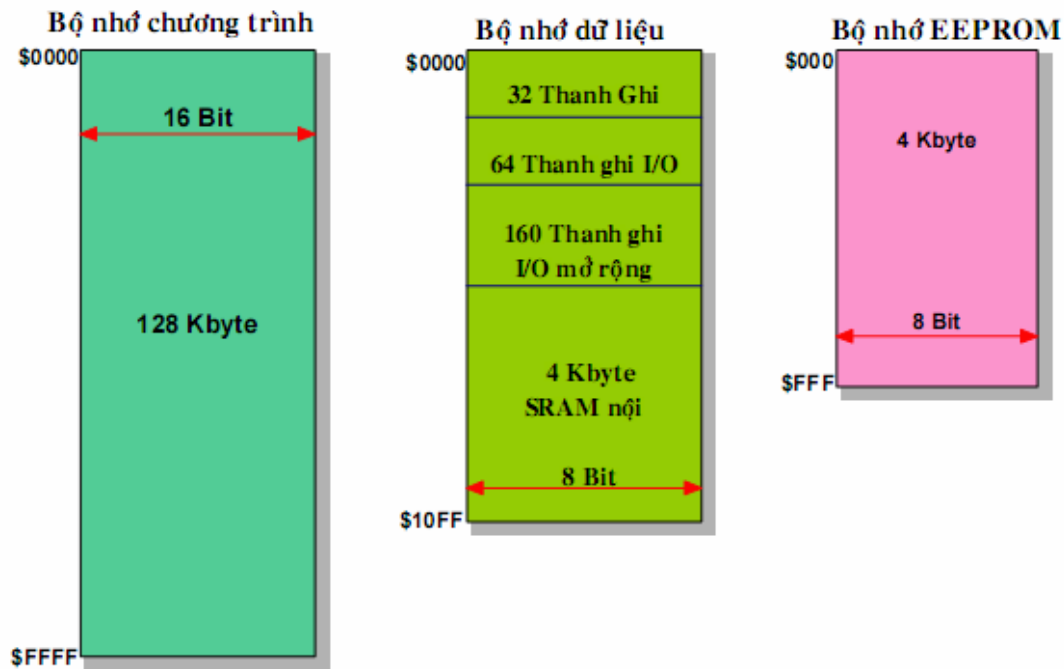
Đây là thanh ghi dữ liệu của EEPROM, là nơi chứa dữ liệu ta định ghi vào hay lấy ra từ EEPROM.

#### + Thanh ghi EECR

BIT	7	6	5	4	3	2	1	0	
	-	-	-	-	EERIE	EEMWE	EEWE	EERE	EECR
ĐỌC/ VIẾT	R	R	R	R	R/W	R/W	R/W	R/W	
GIÁ TRỊ BẮT ĐẦU	0	0	0	0	0	0	X	0	

Đây là thanh ghi điều khiển EEPROM, ta chỉ sử dụng 4 bit đầu của thanh ghi này, bốn bit cuối là dự trữ, ta nên ghi 0 vào các bit dự trữ

### Tóm tắt bản đồ bộ nhớ bên trong atmega8



Bản đồ bộ nhớ ATmega128

Hình 2.4 [1]

## 2.1.3 Cổng vào ra

### 2.1.3.1 Giới thiệu

Cổng vào ra là một trong số các phương tiện để vi điều khiển giao tiếp với các thiết bị ngoại vi. AT mega8 có tất cả các cổng vào ra 8 bit là: PortA, PortB, PortC, PortD. Các cổng vào ra của AVR là cổng vào hai chiều có thể định hướng, tức có thể chọn hướng của cổng là hướng vào (input) hay hướng ra (output). Tất cả các cổng vào ra của AVR đều có chức năng Đọc – Chỉnh sửa – Ghi (Read – Modify – Write) khi sử dụng chúng như là các cổng vào ra số thông thường. Điều này có nghĩa là khi ta thay đổi hướng một chân nào đó thì nó không làm ảnh hưởng tới hướng của các chân khác. Tất cả các chân của các Port đều có điện trở kéo lên (pull-up) riêng, ta có thể cho phép hay không cho phép điện trở kéo lên này hoạt động.

Điện trở kéo lên là một điện trở được dùng khi thiết kế các mạch điện tử logic. Nó có một đầu được nối với nguồn điện áp dương (VCC – Vdd) và đầu còn lại được nối với tín hiệu lỗi vào/ra của một mạch logic chức năng.

### 2.1.3.2 Cách hoạt động

Khi khảo sát các cổng như là các cổng vào ra số thông thường thì tính chất của các cổng (PortA, PortB, ...) là tương tự nhau, nên ta chỉ cần khảo sát một cổng nào đó trong số 7 cổng của vi điều khiển là đủ.

Mỗi một cổng vào ra của vi điều khiển được liên kết với ba thanh ghi: PORTx, DDRx, PINx. ( x thay thế cho A,B,...). Ba thanh ghi này sẽ được phối hợp với nhau để điều khiển hoạt động của cổng, chẳng hạn thiết lập cổng thành lỗi vào có sử dụng điện trở kéo lên... Sau đây là nguyên lý chi tiết vai trò của ba thanh ghi trên:

- **Thanh ghi DDRx.**

Đây là thanh ghi 8 bit (có thể đọc ghi) có khả năng điều khiển hướng của cổng (là lỗi vào hay lỗi ra). Khi một bit của thanh ghi này được set lên 1 thì chân tương ứng với nó được cấu hình thành ngõ ra. Ngược lại, nếu bit của thanh ghi DDRx là 0 thì chân tương ứng với nó được thiết lập thành ngõ vào.

Ví dụ: Khi ta set tất cả 8 bit của thanh ghi DDRA đều là 1, thì 8 chân tương ứng của PortA, là PA1, PA2, .... PA7 (tương ứng với các chân của vi điều khiển) được thiết lập thành ngõ ra.

BIT	7	6	5	4	3	2	1	0	
	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	DDRA
ĐỌC/VIẾT	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
GIÁ TRỊ BẮT ĐẦU	0	0	0	0	0	0	0	0	

*Thanh ghi DDRA*

- **Thanh ghi PORTx**

PORTx là thanh ghi 8 bit có thể đọc ghi. Đây là thanh ghi dữ liệu của PORTx. Nếu thanh ghi DDRx thiết lập cổng là lỗi ra, khi đó giá trị của thanh ghi PORTx cũng là giá trị của các chân tương ứng của PORTx, nói cách khác, khi ta ghi một giá trị logic lên 1 bit của thanh ghi này thì chân tương ứng với bit đó cũng có cùng mức logic. Khi thanh ghi DDRx thiết lập cổng thành lỗi vào thì thanh ghi PORTx đóng vai trò như một thanh ghi điều khiển cổng.

Cụ thể, nếu một bit của thanh ghi này được ghi thành 1 thì điện trở treo ở chân tương ứng với nó sẽ được kích hoạt, ngược lại nếu bit được ghi thành 0 thì điện trở treo ở chân tương ứng sẽ không được kích hoạt, cổng ở trạng thái cao trở (HI-Z).

BIT	7	6	5	4	3	2	1	0	
	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	PORTA
ĐỌC/VIẾT	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
GIÁ TRỊ BẮT ĐẦU	0	0	0	0	0	0	0	0	

*Thanh ghi PORTA*



- **Thanh ghi PINx**

PINx không phải là một thanh ghi thật sự, đây là địa chỉ trong bộ nhớ I/O kết nối trực tiếp với các chân của cổng. Khi ta đọc PORTx tức ta đọc dữ liệu được chốt trong PORTx, còn khi đọc PINx thì giá trị logic hiện thời ở chân của cổng tương ứng được đọc. Vì thế đối với thanh ghi PINx ta có thể đọc mà không thể ghi.

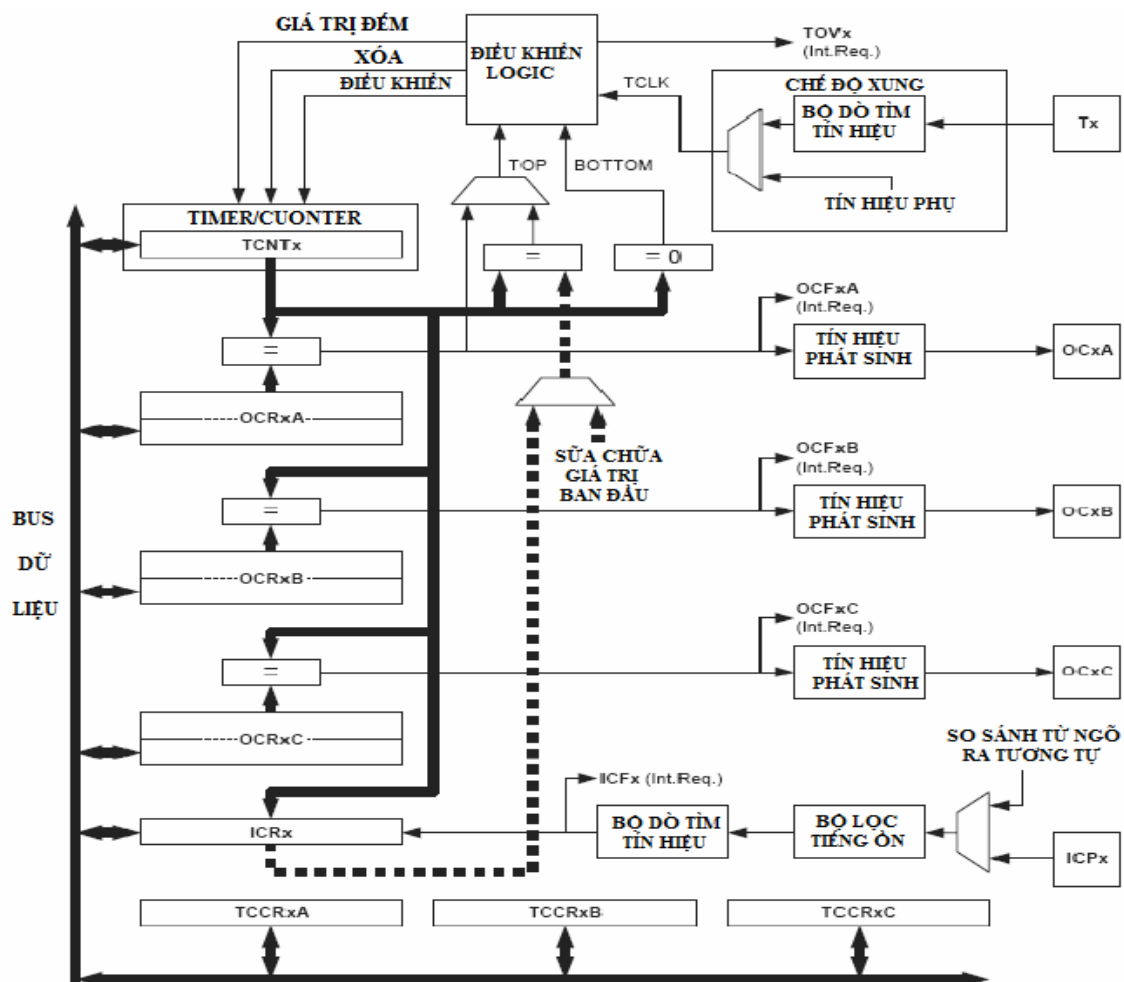
BIT	7	6	5	4	3	2	1	0	
	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	PINA
ĐỌC/VIẾT	R	R	R	R	R	R	R	R	
GIÁ TRỊ BẮT ĐẦU	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

*Thanh ghi PINA*

### 2.1.4 Bộ định thời của AT mega8 [1]

AT mega8 có 4 bộ định thời, bộ định thời 1 và 3 là bộ định thời 16 bit, bộ định thời 0 và 2 là bộ định thời 8 bit. Sau đây là mô tả chi tiết của 4 bộ định thời.

#### 2.1.4.1 Bộ định thời 1



Hình 2.5 Sơ đồ bộ định thời 1

Bộ định thời 1 và 3 là bộ định thời 16 bit, bộ định thời 1 sử dụng 13 thanh ghi liên quan, còn bộ định thời 3 sử dụng 11 thanh ghi liên quan với nhiều chế độ thực thi khác nhau. Vì bộ định thời 1 và 3 hoạt động giống nhau nên bài viết này em chỉ trình bày về bộ định thời 1.

Các định nghĩa sau sẽ được sử dụng trong bộ định thời 1:

**BOTTOM:** Bộ đếm đạt đến giá trị BOTTOM khi nó có giá trị 0000h.

**MAX:** Bộ đếm có giá trị MAX khi nó bằng FFFFh.

**TOP:** Bộ đếm đạt giá trị TOP khi nó bằng với giá trị cao nhất trong chuỗi đếm, giá trị cao nhất trong chuỗi đếm không nhất thiết là FFFFh mà có thể là bất cứ giá trị nào được qui định trong thanh ghi OCRnX (X=A,B,C) hay ICRn, tùy theo chế độ thực thi.

Bộ định thời 1 bao gồm các thanh ghi:

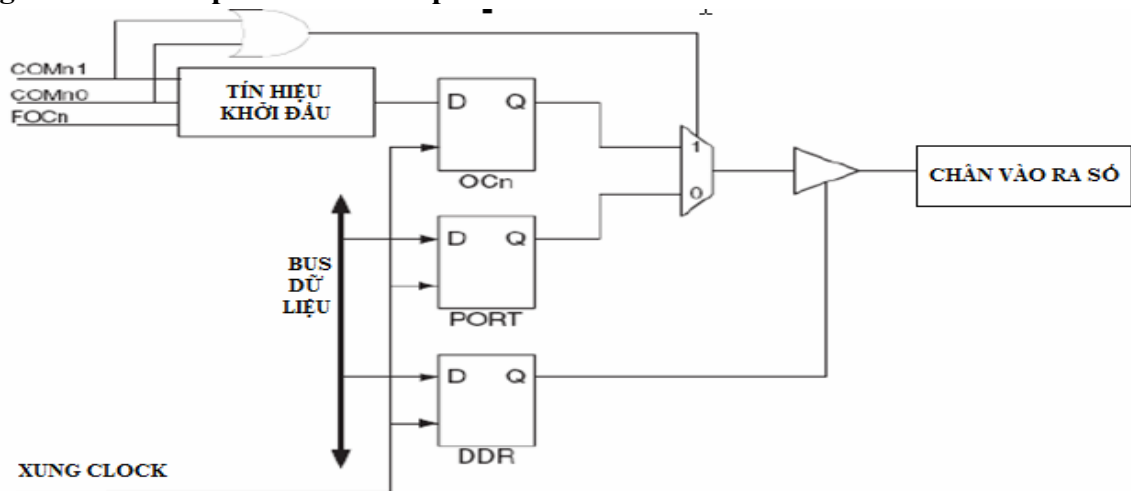
- Thanh ghi TCCR1A (Timer/Cuonter 1 Control Register)
- Thanh ghi TCCR1B
- Thanh ghi TCCR1C
- Thanh ghi Timer/Counter – TCNT1H and TCNT1L
- Thanh ghi Output Compare Register 1A – OCR1AH and OCR1AL
- Thanh ghi Output Compare Register 1B – OCR1BH and OCR1BL
- Thanh ghi Output Compare Register 1C – OCR1CH and OCR1CL
- Thanh ghi Input Capture Register 1 – ICR1H and ICR1L
- Thanh ghi Timer/Cuonter Interrupt Mask Register – TIMSK
- Thanh ghi Extended Timer/Cuonter Interrupt Mask Register – ETIMSK
- Thanh ghi Timer/Cuonter Interrupt Flag Register – TIFR
- Thanh ghi Extended Timer/Cuonter Interrupt Flag Register – ETIFR
- Thanh ghi Special Function IO Register – SFIOR

#### 2.1.4.2 Bộ định thời 3

Do bộ định thời 3 có cấu trúc giống bộ định thời 1, nên ở đây chỉ trình bày các thanh ghi có liên quan tới bộ định thời 3.

- Thanh ghi TCCR3A
- Thanh ghi TCCR3B
- Thanh ghi TCCR3C
- Thanh ghi Timer/Cuonter1 – TCNT3H and TCNT3L
- Thanh ghi Output Compare Register 3A – OCR3AH and OCR3AL
- Thanh ghi Output Compare Register 3B – OCR3BH and OCR3BL
- Thanh ghi Output Compare Register 3C – OCR3CH and OCR3CL
- Thanh ghi Extended Timer/Cuonter Interrupt Mask Register – ETIMSK
- Thanh ghi Extended Timer/Cuonter Interrupt Flag Register – ETIFR
- Thanh ghi Special Funtion IO Register – SFIOR

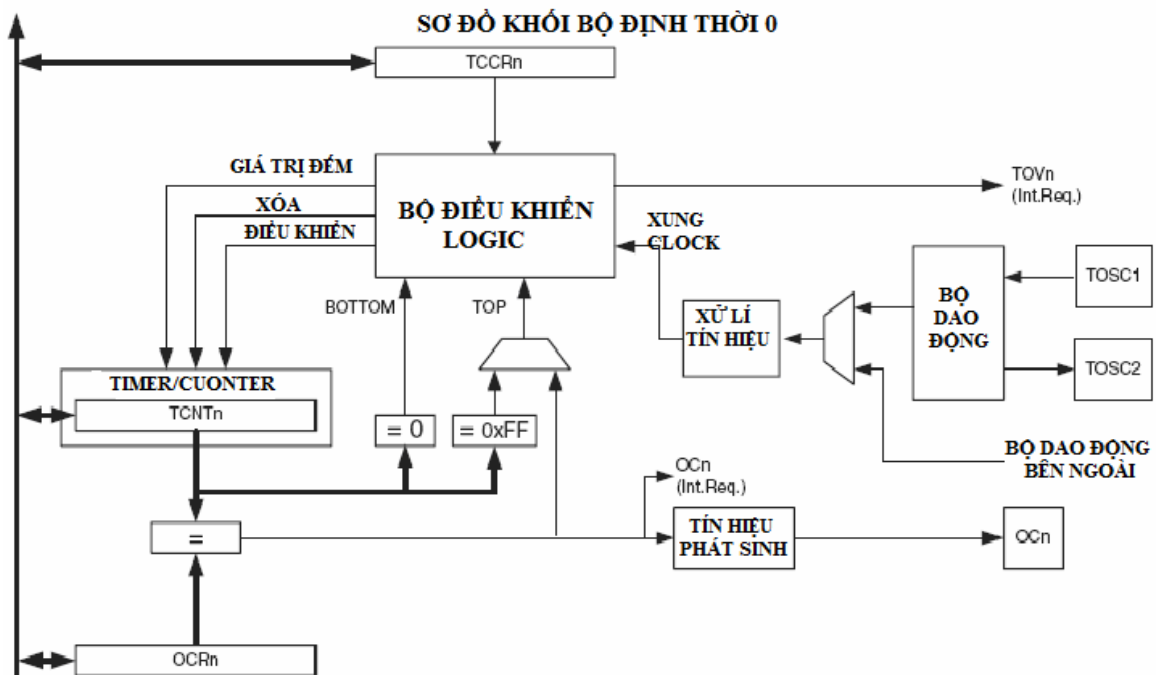
### Ngõ ra khỏi Compare Match Output Unit



### Hình 2.6 Sơ đồ ngõ ra khối

Nhìn hình ta thấy Pin OcnX (chân hạn pin 15 của IC tương ứng với OC1A), là ngõ ra của khối Compare Match Output Unit, có thể được nối với 3 thanh ghi là OCnX, PortX và DDRX. Thanh ghi nào được nối với Ocn là phụ thuộc vào các bit COMn1:0 (tức là tùy theo chế độ hoạt động của bộ định thời). Nếu ta thiết lập bộ định thời hoạt động ở chế độ thường ( tức không sử dụng chức năng so sánh khớp) thì chân Ocn trở thành chân vào ra số thông thường. Ngõ ra khối Compare Match Output Unit của bộ định thời 1 cũng giống như bộ định thời 3.

### 2.1.4.3 Bộ định thời 0



Hình 2.7 Sơ đồ khối bộ định thời 0

Bộ định thời 0 là bộ định thời 8 bit, bộ định thời 0 liên quan tới 7 thanh ghi với nhiều chế độ thực thi khác nhau.

### Các định nghĩa

*Các định nghĩa sau sẽ được sử dụng cho bộ định thời 0 và 2*

**BOTTOM:** Bộ đếm đạt tới giá trị BOTTOM khi nó có giá trị 00h.

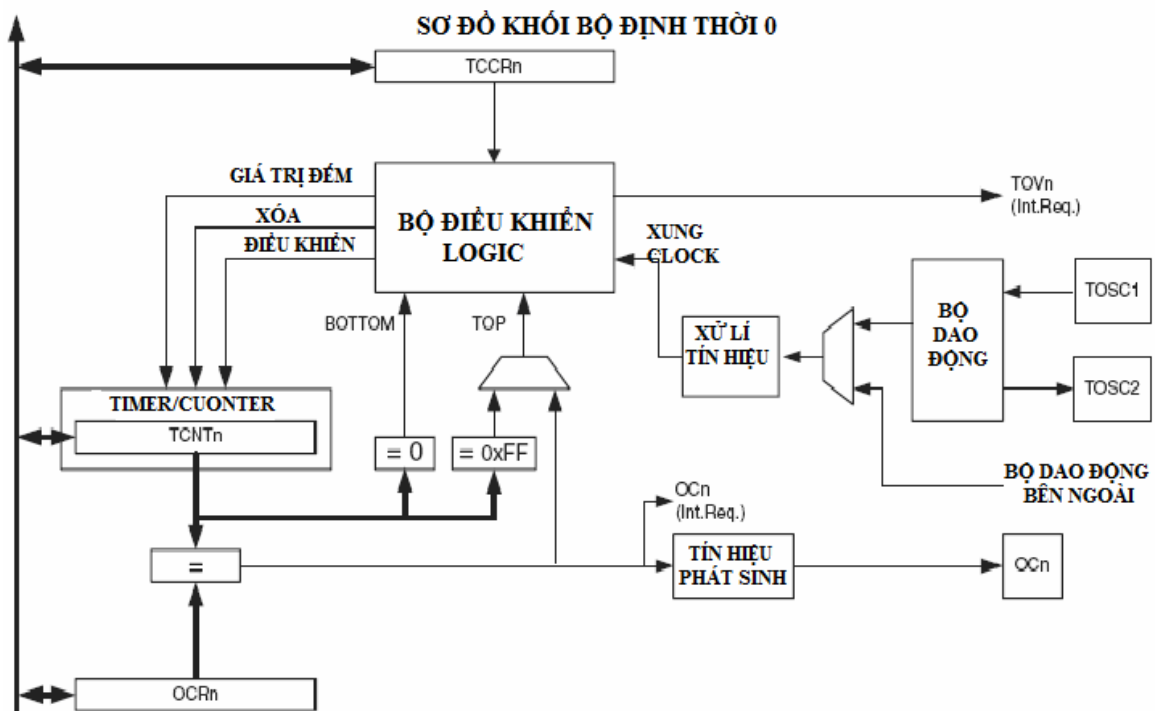
**MAX:** Bộ đếm đạt tới giá trị Max khi nó bằng FFh.

**TOP:** Bộ đếm đạt giá trị TOP khi nó bằng với giá trị cao nhất trong chuỗi đếm, giá trị cao nhất trong chuỗi đếm không nhất thiết là FFh mà có thể là bất kỳ giá trị nào được qui định trong thanh ghi OCRn (n=0,2), tùy theo chế độ thực thi.

Bộ định thời 0 có vài đặc điểm chính như: Bộ đếm đơn kênh, xóa bộ định thời khi có sự kiện so sánh khớp (compare match) và tự nạp lại, có thể đếm từ bộ giao động 32 KHz bên ngoài, chế độ PWM hiệu chỉnh pha... Các thanh ghi trong bộ định thời 0 bao gồm:

- Thanh ghi Timer/Cuonter Control Register – TCCR0
- Thanh ghi Timer/Cuonter Register - TCNT0
- Thanh ghi Output Compare Register – OCR0
- Thanh ghi Timer/Cuonter Intertupt Mask Register – TIMSK
- Thanh ghi Timer/Cuonter Interrupt Flag Register – TIFR
- Thanh ghi Special Function IO Register – SFIOR
- Thanh ghi Asynchronous Status Register – ASSR

#### 2.1.4.4 Bộ định thời 2



Hình 2.8 Sơ đồ khối bộ định thời 2

Bộ định thời 2 là bộ định thời 8 bit, bộ định thời 2 liên quan tới 5 thanh ghi với nhiều chế độ thực thi khác nhau. Thuộc tính chính của bộ định thời 2 gồm: Bộ đếm đơn kênh, xóa bộ định thời khi có sự kiện “so sánh khớp” và tự động nạp lại, PWM hiệu chỉnh pha, đếm sự kiện bên ngoài.

#### *CÁC THANH GHI BỘ ĐỊNH THỜI 2*

- Thanh ghi Timer/Cuonter Cuontrol Register – TCCR2
- Thanh ghi Timer/Cuonter Register – TCNT2
- Thanh ghi Output Compare Register – OCR2
- Thanh ghi Timer/Cuonter Interrupt Mask Register – TIMSK
- Thanh ghi Timer/Cuonter Interrupt Flag Register – TIFR

## **2.2 Cấu trúc ngắt của ATmega8**

### **2.2.1 Khái niệm về ngắt**

Ngắt là một sự kiện bên trong hay bên ngoài làm ngắt bộ vi điều khiển để báo cho nó biết rằng thiết bị cần dịch vụ của nó.

Một bộ vi điều khiển có thể phục vụ một vài thiết bị, có hai cách để thực hiện điều này đó là sử dụng các ngắt (interrupt) và thăm dò (polling). Trong phương pháp sử dụng các ngắt thì mỗi khi có một thiết bị bất kỳ cần đến dịch vụ của nó thì nó báo cho bộ vi điều khiển bằng cách gửi một tín hiệu ngắt. Khi nhận được tín hiệu ngắt thì bộ vi điều khiển ngắt tất cả những gì nó đang thực hiện để chuyển sang phục vụ thiết bị. Chương trình đi cùng với ngắt được gọi là dịch vụ ngắt ISR (Interrupt Service Routine) hay còn gọi là trình quản lý ngắt (Interrupt handler). Còn trong phương pháp thăm dò thì bộ vi điều khiển hiển thị liên tục tình trạng của một thiết bị đã cho và điều kiện thỏa mãn thì nó phục vụ thiết bị. Sau đó chuyển sang hiển thị trạng thái của thiết bị kế tiếp cho đến khi tất cả đều được phục vụ.

Mặc dù phương pháp thăm dò có thể hiển thị tình trạng của một vài thiết bị và phục vụ mỗi thiết bị khi các điều kiện nhất định được thỏa mãn nhưng nó không tận dụng hết công dụng của bộ vi điều khiển. Điểm mạnh của phương pháp ngắt là bộ vi điều khiển có thể phục vụ rất nhiều thiết bị (tất nhiên là không tại cùng một thời điểm). Mỗi thiết bị có thể nhận được sự chú ý của bộ vi điều khiển dựa trên mức ưu tiên cho các thiết bị vì nó kiểm tra tất cả các thiết bị theo kiểu xoay vòng. Quan trọng hơn là trong phương pháp ngắt thì bộ vi điều khiển cũng có thể che hoặc làm lơ một yêu cầu dịch vụ của thiết bị. Điều này lại một lần nữa không thể thực hiện được trong phương pháp thăm dò. Lý do quan trọng nhất mà phương pháp ngắt được ưa chuộng nhất là vì phương pháp thăm dò làm hao phí thời gian của bộ vi điều khiển bằng cách hỏi dò từng thiết bị kể cả khi chúng không cần đến dịch vụ.

### **2.2.2 Trình phục vụ ngắt của bảng Vector ngắt.**

Đối với mỗi ngắt thì phải có một trình phục vụ ngắt ISR (Interrupt Service Routine) hay trình quản lý ngắt (Interrupt handler). Khi một ngắt được gọi thì bộ vi điều khiển phục vụ ngắt. Khi một ngắt được gọi thì bộ vi điều khiển chạy trình phục vụ ngắt. Đối với mỗi ngắt thì có một vị trí cố định trong bộ nhớ để giữ lại địa chỉ ISR của nó. Nhóm các vị trí nhớ được dành riêng để gửi các địa chỉ của các ISR được gọi là bảng véc tơ ngắt.

Khi kích hoạt một ngắt thì bộ vi điều khiển đi qua các bước sau:

- Vi điều khiển kết thúc lệnh đang thực hiện và lưu địa chỉ của lệnh kế tiếp (PC) vào ngăn xếp.
- Nó nhảy đến một vị trí cố định trong bộ nhớ được gọi là bảng véc tơ ngắt nơi lưu giữ địa chỉ của một trình phục vụ ngắt.
- Bộ vi điều khiển nhận địa chỉ ISR từ bảng véc tơ ngắt và nhảy tới đó. Nó bắt đầu thực hiện trình phục vụ ngắt cho đến lệnh cuối cùng của ISR là RETI (trở về từ ngắt).
- Khi thực hiện lệnh RETI bộ vi điều khiển quay trở về nơi nó đã bị ngắt. Trước hết nó nhận địa chỉ của bộ đếm chương trình PC từ ngăn xếp bằng cách kéo hai byte trên đỉnh của ngăn xếp vào PC. Sau đó bắt đầu thực hiện các lệnh từ địa chỉ đó.

### 2.2.3 Bảng Vector ngắt của ATmega8

Đây là bảng véc tơ ngắt của Atmega8, cùng với địa chỉ của nó trong bộ nhớ chương trình.

Số vector	Địa chỉ	Nguồn (điểm gốc)	Ý nghĩa
1	\$0000	RESET	Reset AVR
2	\$0002	INT0	Ngắt ngoài 0
3	\$0004	INT1	Ngắt ngoài 1
4	\$0006	INT2	Ngắt ngoài 2
5	\$0008	INT3	Ngắt ngoài 3
6	\$000A	INT4	Ngắt ngoài 4
7	\$000C	INT5	Ngắt ngoài 5
8	\$000E	INT6	Ngắt ngoài 6
9	\$0010	INT7	Ngắt ngoài 7
10	\$0012	TIMER2 COMP	So sánh Timer/Cuonter 2
11	\$0014	TIMER2 OVF	Báo tràn Timer/Cuonter 2
12	\$0016	TIMER1 COMPA	Sử dụng Timer/Cuonter 1
13	\$0018	TIMER1 COMPA	So sánh Timer/Cuonter1 (A)
14	\$001A	TIMER1 COMPB	So sánh Timer/Cuonter1 (B)
15	\$001C	TIMER1 OVF	Báo tràn Timer/Cuonter 1
16	\$001E	TIMER0 COMP	So sánh Timer/Cuonter0
17	\$0020	TIMER0 OVF	Báo tràn Timer/Cuonter0
18	\$0022	SPI.STC	Khởi truyền nhận nối tiếp
19	\$0024	USART0. RX	Bộ truyền dữ liệu nối tiếp 0 RX
20	\$0026	USART0.UDRE	Bộ dữ liệu trống USART0
21	\$0028	USART0.TX	Bộ truyền dữ liệu nối tiếp TX
22	\$002A	ADC	Bộ chuyển đổi ADC
23	\$002C	EE READY	Bộ nhớ EEPROM
24	\$002E	ANALOG COMP	So sánh tín hiệu tương tự
25	\$0030	TIMER1 COMPC	So sánh Timer/Cuonter1 (C)
26	\$0032	TIMER3 CAPT	Sử dụng Timer/Cuonter3
27	\$0034	TIMER3 COMPA	So sánh Timer/Cuonter3 (A)
28	\$0036	TIMER3 COMPB	So sánh Timer/Cuonter3 (B)

29	\$0038	TIMER3 COMPC	So sánh Timer/Cuonter3 (C)
30	\$003A	TIMER3 OVF	Báo tràn Timer 3
31	\$003C	USART1.RX	Bộ truyền dữ liệu nối tiếp 1 RX
32	\$003E	USART1.UDRE	Bộ dữ liệu rỗng USART1
33	\$0040	USART1.TX	Bộ truyền dữ liệu nối tiếp 1 TX
34	\$0042	TWI	Hai giá trị bên ngoài
35	\$0044	SPM READY	Bộ nhớ chương trình

Hình 2.9 Bảng vector ngắt của Atmega8 [3]

#### 2.2.4 Thứ tự ưu tiên ngắt

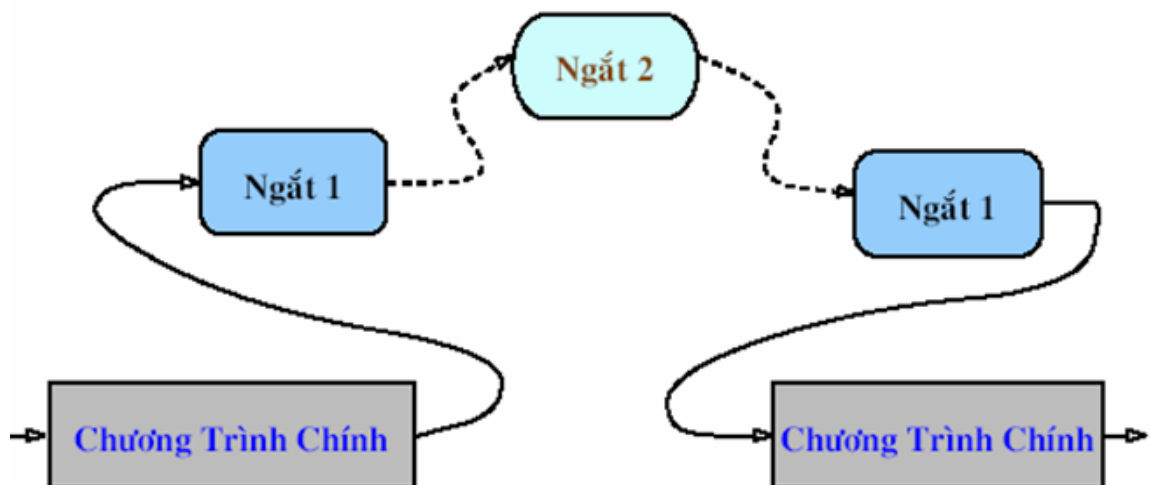
Không như vi điều khiển họ 8051, ở đó thứ tự ưu tiên của các ngắt có thể thay đổi được bằng cách lập trình. Với vi điều khiển AVR thứ tự ưu tiên các ngắt là không thể thay đổi và theo qui tắc: “Một véc tơ ngắt có địa chỉ thấp hơn trong bộ nhớ chương trình có mức độ ưu tiên cao hơn”. Chấn hạn ngắt ngoài 0 (INT0) có mức độ ưu tiên cao hơn ngắt ngoài 1 (INT1).

Để cho phép một ngắt người dùng cần cho phép ngắt toàn cục (set bit 1 trong thanh SREG) và các bit điều khiển tương ứng.

Khi một ngắt xảy ra và đang được phục vụ thì bit I trong thanh ghi SREG bị xóa, như thế khi có một ngắt khác xảy ra thì nó sẽ không được phục vụ, do đó để cho phép các ngắt trong một ISR (interrupt service routine) khác đang thực thi, thì trong chương trình ISR phải có lệnh SEI để set lại bit I trong SREG.

#### 2.2.5 Ngắt trong ngắt.

Khi AVR đang thực hiện một trình phục vụ ngắt thuộc một ngắt nào đó thì lại có một ngắt khác được kích hoạt. Trong những trường hợp như vậy thì một ngắt có mức ưu tiên cao hơn có thể ngắt một ngắt có mức ưu tiên thấp hơn. Lúc này ISR của ngắt có mức ưu tiên cao hơn sẽ được thực thi. Khi thực hiện xong ISR của ngắt có mức ưu tiên cao hơn thì nó mới quay lại phục vụ tiếp ISR của ngắt có mức ưu tiên thấp hơn trước khi trở về chương trình chính. Đây gọi là ngắt trong ngắt.



Hình 2.10 Các ngắt lồng nhau

Chú ý:

Giả định là khi một ISR nào đó đang thực thi thì xảy ra một yêu cầu ngắt từ một ISR khác có mức ưu tiên thấp hơn thì ISR có mức ưu tiên thấp hơn không được phục vụ, nhưng nó sẽ không bị bỏ qua luôn mà ở trạng thái chờ. Nghĩa là ngay sau khi ISR có mức ưu tiên cao hơn thực thi xong thì đến lượt ISR có mức ưu tiên thấp hơn sẽ được phục vụ.

**2.2.6 Các ngắt ngoài [3]**

AT mega8 có 8 ngắt ngoài từ INT0 đến INT7 (ở đây chưa kể tới ngắt Reset). Tám ngắt này tương ứng với 8 chân của MCU là INT0, INT1.....INT7. Để ý là ngay cả khi các chân INT0, INT1,...INT7 của MCU được cấu hình như là chân lối ra, thì các ngắt ngoài vẫn có tác dụng nếu được cho phép.

Các ngắt ngoài có thể bắt mẫu theo kiểu cạnh lên (Rising), cạnh xuống (Falling) hay mức thấp (Low level). Điều này được qui định trong hai thanh ghi EICRA và EICRB. Dưới đây là mô tả chi tiết 2 thanh ghi EICRA và EICRB và các thanh ghi liên quan tới ngắt ngắt ngoài.

- Thanh ghi External Interrupt Control Register A – EICRA

BIT	7	6	5	4	3	2	1	0	
	ISC31	ISC30	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00	EICRA
ĐỌC/ VIẾT	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
GIÁ TRỊ BẮT ĐẦU	0	0	0	0	0	0	0	0	

Tám bit của thanh ghi EICRA sẽ điều khiển kiểu bắt mẫu cho 4 ngắt INT3, INT2, INT1, INT0. Qui định cụ thể được thể hiện trong bảng sau:

ISCn1	ISCn0	Kiểu bắt mẫu
0	0	Mức thấp sẽ tạo yêu cầu ngắt
0	1	Dự trữ
1	0	Cạnh xuống (Falling) sẽ tạo yêu cầu ngắt
1	1	Cạnh lên (Rising) sẽ tạo yêu cầu ngắt

Hình 2.11 Bảng điều khiển kiểu bắt mẫu ngắt

- Thanh ghi External Interrupt Control Register B – EICRB



BIT	7	6	5	4	3	2	1	0	
	ISC71	ISC70	ISC61	ISC60	ISC51	ISC50	ISC41	ISC40	EICRB
ĐỌC/VIẾT	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
GIÁ TRỊ BẮT ĐẦU	0	0	0	0	0	0	0	0	

Tám bit của thanh ghi EICRA sẽ điều khiển kiểu bắt mẫu cho 4 ngắt INT7, INT6, INT5, INT4. Qui định cụ thể được thể hiện trong bảng sau:

ISCn1	ISCn0	Kiểu bắt mẫu
0	0	Mức sẽ tạo yêu cầu ngắt
0	1	Bất cứ sự thay đổi mức logic nào ở chân INTn sẽ tạo ra một yêu cầu ngắt.
1	0	Cạnh xuống (Falling) giữa hai mẫu sẽ tạo yêu cầu ngắt.
1	1	Cạnh lên (Rising) giữa hai mẫu sẽ tạo yêu cầu ngắt.

Với  $n = 7, 6, 5, 4, \dots$

- Thanh ghi External Interrupt Mask Register - EIMSK

BIT	7	6	5	4	3	2	1	0	
	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0	EIMSK
ĐỌC/VIẾT	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
GIÁ TRỊ BẮT ĐẦU	0	0	0	0	0	0	0	0	

- Thanh ghi External Interrupt Flag Register – EIFR

BIT	7	6	5	4	3	2	1	0	
	INTF7	INTF6	INTF5	INTF4	INTF3	INTF2	INTF1	INTF0	EIFR
ĐỌC/VIẾT	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
GIÁ TRỊ BẮT ĐẦU	0	0	0	0	0	0	0	0	

- Thanh ghi MCU Control Register – MCUCR

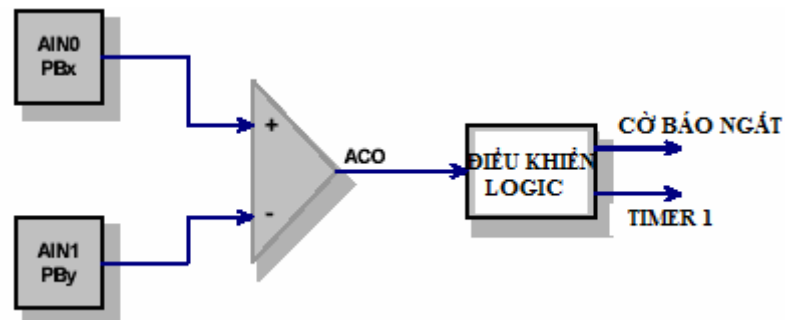
BIT	7	6	5	4	3	2	1	0	
	SRE	SRW10	SE	SM1	SM0	SM2	IVSEL	IVCE	MCUCR
ĐỌC/VIẾT	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
GIÁ TRỊ BẮT ĐẦU	0	0	0	0	0	0	0	0	

## 2.3 Các bộ phận ngoại vi khác [3] [4]

Ngoài các bộ phận ngoại vi đã được giới thiệu ở trên như: Bộ định thời, các cổng vào ra, EEPROM .... Vi điều khiển AT mega8 có nhiều bộ phận ngoại vi khác, các bộ ngoại vi này rất tiện lợi trong các ứng dụng điều khiển (bộ PWM) xử lý số liệu (bộ ADC, bộ so sánh Analog), giao tiếp (bộ USART, SPI, I2C).... Việc tích hợp các bộ ngoại vi này vào trong chip giúp cho các thiết kế trở nên thuận tiện hơn, kích thước bo mạch cũng gọn gàng hơn.

### 2.3.1 Bộ so sánh tương tự

Sơ đồ đơn giản của bộ so sánh tương tự (Analog Comparator). Như hình bên dưới bộ so sánh có hai ngõ vào tương tự là AIN0 và AIN1 và một ngõ ra số ACO. Nguyên tắc hoạt động của bộ so sánh tương tự là: Khi ngõ vào AIN0 có điện thế cao hơn ngõ vào AIN1 thì ngõ ra ACO sẽ ở mức cao (tương ứng với mức logic 1), ngược lại khi ngõ vào AIN0 có điện thế thấp hơn ngõ vào AIN1 thì ngõ ra ACO sẽ ở mức thấp (tương ứng với mức logic 0). Thường thì trong hai ngõ vào, có một ngõ vào có điện thế được giữ cố định dùng để làm điện thế tham chiếu ( $V_{ref}$ ), điện thế ngõ còn lại có thể thay đổi để có thể tham chiếu với ngõ vào  $V_{ref}$ . Trạng thái của ngõ ra ACO của bộ so sánh có thể được dùng để tạo ra một ngắt, kết nối với bộ định thời 1 để sử dụng chức năng input capture của bộ định thời này.



Hình 2.12 Sơ đồ giản lược của bộ so sánh tương tự

Ở đây có sự khác biệt về chi tiết ở bộ so sánh tương tự đối với các dòng AVR khác nhau, chẳng hạn bộ so sánh tương tự của AT90S8535 hơi khác với bộ so sánh tương tự ở AT mega8, tuy nhiên cấu trúc cơ bản thì vẫn như nhau. Ta thấy hai ngõ vào AIN0 và AIN1 tương ứng với hai chân PBx và PBy ( $x=2, y=3$ , đối với AT90S8535). Ở Atmega8 ta có nhiều lựa chọn ngõ vào hơn, các thanh ghi chú trong bộ nhớ sẽ giúp ta thiết lập các lựa chọn này.

### 2.3.2 Bộ biến đổi ADC

#### 2.3.2.1 Giới thiệu bộ ADC của Atmega 8

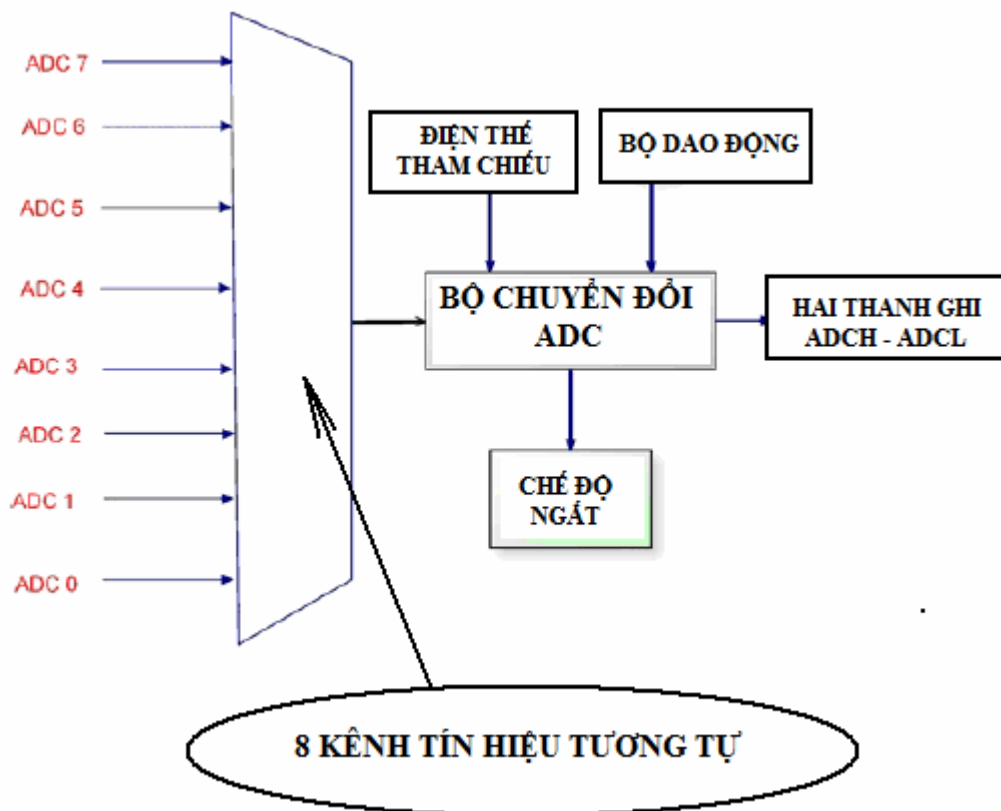
Bộ biến đổi ADC có chức năng biến đổi tín hiệu tương tự (analog signal) có giá trị thay đổi trong một dải biết trước thành tín hiệu số (digital signal). Bộ ADC của Atmega8 có độ phân giải 10 bit, sai số tuyệt đối là 2LSB, dải tín hiệu ngõ vào từ 0V-

Vcc, tín hiệu ngõ vào có nhiều lựa chọn như: có 8 ngõ vào đa hợp đơn hướng (Multiplexed Single Ended), 7 ngõ vào vi sai (Differential Input),... Bộ ADC của ATmega8 là loại ADC xấp xỉ liên tiếp với hai chế độ có thể lựa chọn là chuyển đổi liên tục (Free Running) và chuyển đổi từng bước (Single Conversion).

- Chuyển đổi liên tục: là chế độ mà sau khi khởi động thì bộ ADC thực hiện chuyển đổi liên tục không ngừng.

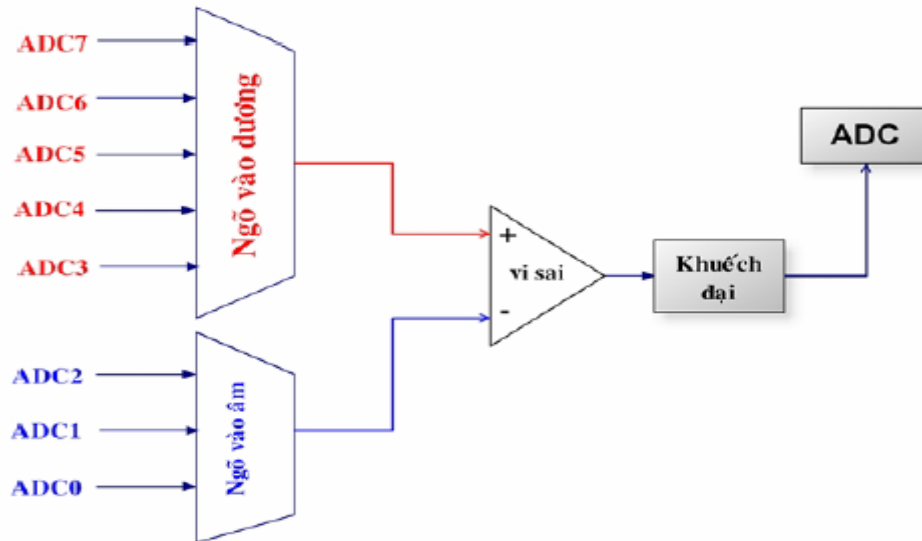
- Chuyển đổi từng bước: là bộ ADC sau khi hoàn thành một chuyển đổi thì sẽ ngừng, một chuyển đổi tiếp theo chỉ được bắt đầu khi phần mềm có yêu cầu chuyển đổi tiếp.

Sơ đồ khối đơn giản của một bộ ADC được thể hiện như sau:



Hình 2.13 Sơ đồ khối đơn giản bộ ADC [5]

Nguyên tắc hoạt động của khối ADC: Tín hiệu tương tự đưa vào các ngõ ADC0: 7 được lấy mẫu và biến đổi thành tín hiệu số tương ứng. Tín hiệu số được lưu hành trong hai thanh ghi ADCH và ADCL. Một ngắt có thể được tạo ra khi hoàn thành một chu trình biến đổi ADC. Bộ ADC của Atmega8 phức tạp hơn nhiều, tuy nhiên cơ sở vẫn dựa vào nguyên tắc trên.



Hình 2.14 Sơ đồ ngõ vào vi sai [5]

Ví dụ: Đoạn chương trình nhỏ sau cho phép bộ ADC hoạt động ở chế độ biến đổi từng bước, ngõ vào là chân ADC3, không dùng ngắt.

```
ldi r16,3 ;
out ADMUX, r16 // Chọn ngõ vào ADC 3, điện thế tham chiếu Vref

ldi r16, 0b10000101
out ADCSRA, r16 // không dùng ngắt, hệ số chia clock là 32, chạy từng
bước

sbi ADCSRA, ADSC // khởi động bộ ADC

Wait:
Sbis ADCSRA, ADIF // đợi ADC hoàn thành
rjmp Wait

in r16, ADCL // lưu kết quả ADC
in r17, ADCH
```

### 2.3.3 Bộ truyền dữ liệu nối tiếp USART

#### 2.3.3.1 Tóm lược về USART

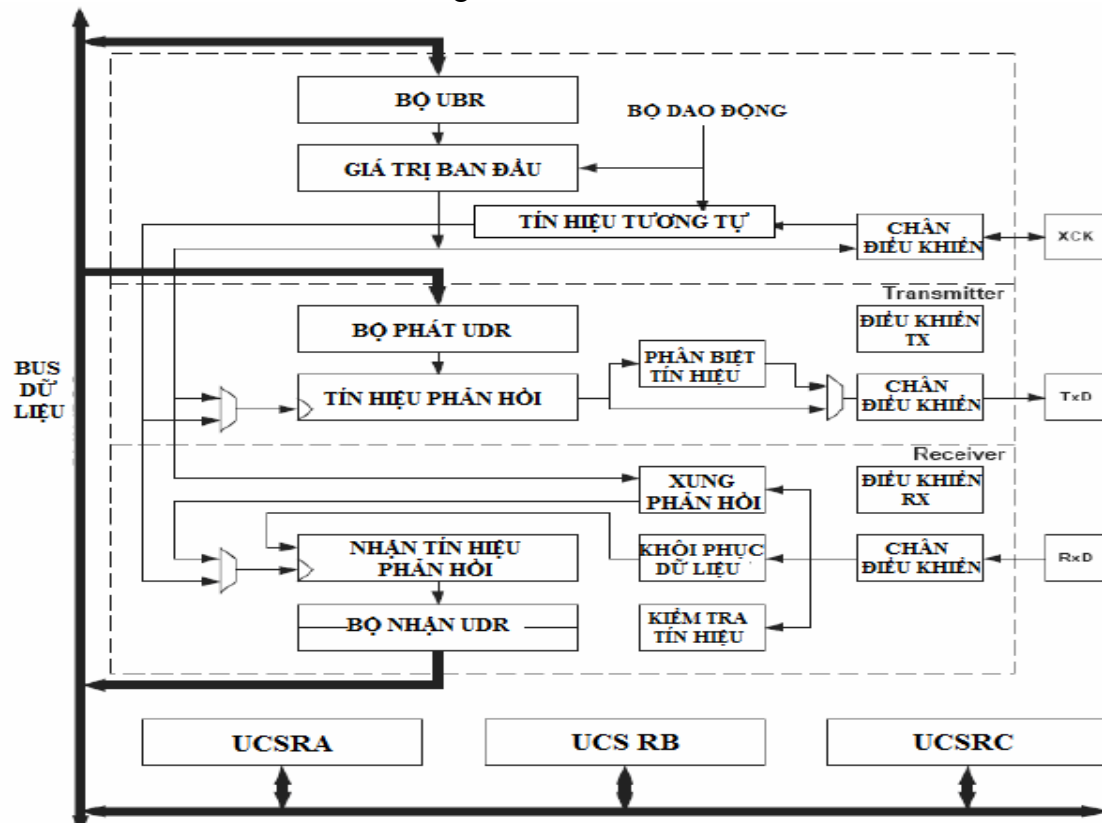
USART (Universal Synchronous and Asynchronous serial Receiver and Transmitter): Bộ điều khiển đồng bộ và bất đồng bộ, đây là khối chức năng dùng cho việc truyền thông giữa vi điều khiển với các thiết bị khác. Trong vấn đề truyền dữ liệu số, có thể phân chia cách thức truyền dữ liệu ra hai chế độ cơ bản là: Chế độ nhận đồng bộ (Synchronous) và chế độ truyền nhận bất đồng bộ (Asynchronous). Ngoài ra, nếu góc độ phần cứng thì có thể phân chia theo cách khác đó là: Truyền nhận dữ liệu theo kiểu nối tiếp (serial) và song song (parallel).

Truyền đồng bộ: là kiểu truyền dữ liệu trong đó bộ truyền (Transmitter) và bộ nhận (Receiver) sử dụng một xung đồng hồ (clock). Do đó, hoạt động truyền và nhận giữ liệu ra đồng thời.

Truyền bất đồng bộ: Là kiểu truyền dữ liệu trong đó mỗi bộ truyền và bộ nhận có bộ dao động xung clock riêng, tốc độ xung clock ở hai khối này có thể khác nhau, nhưng thường không quá 10%. Do đó không dùng chung xung clock, nên để đồng bộ quá trình truyền và nhận dữ liệu, người ta phải truyền các bit đồng bộ (Start, Stop....) đi kèm với các bit dữ liệu.

#### Giới thiệu bộ USART của Atmega 8

AT mega8 có hai bộ USART là USART0 và USART1. Hai bộ USART này là độc lập nhau, điều này có nghĩa là hai khối USART0 và USART1 có thể hoạt động cùng một lúc. Bên dưới là sơ đồ khối đơn giản của khối USART.

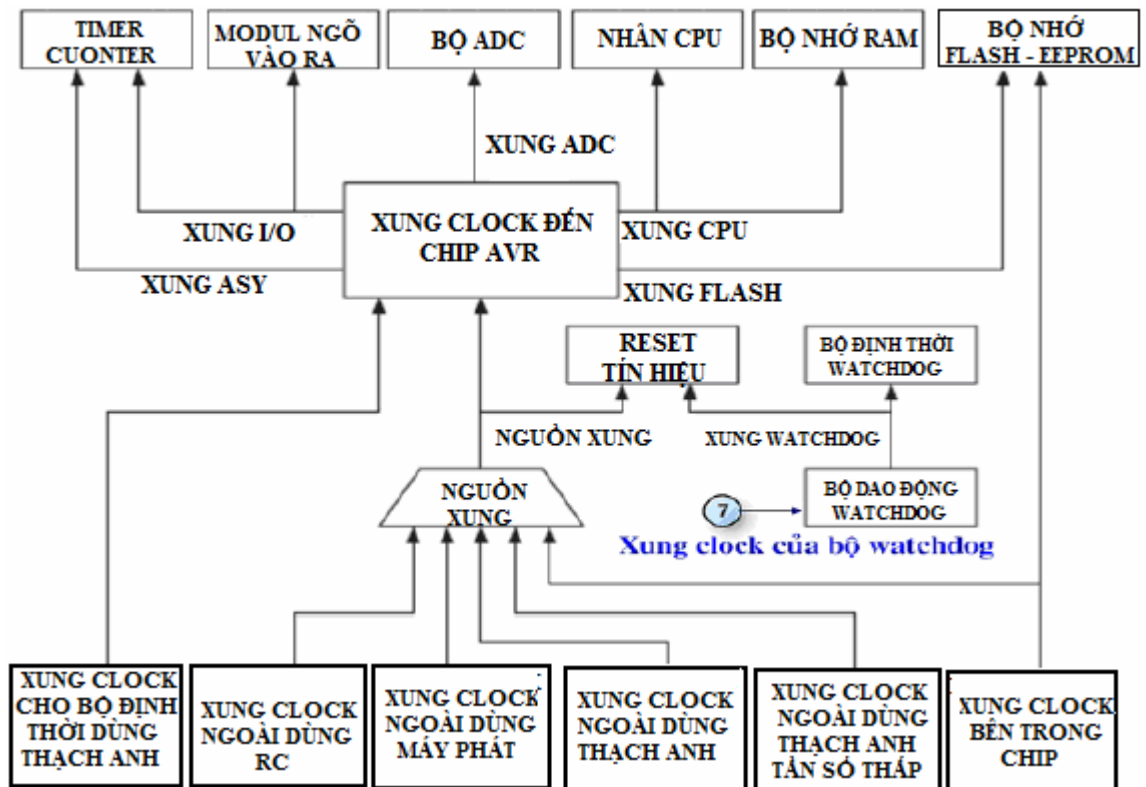


Hình 2.15 Sơ đồ khối bộ USART

## 2.4 Hệ thống xung CLOCK và lập trình bộ nhớ ON – CHIP

### 2.4.1 Hệ thống xung CLOCK

Hệ thống xung clock Atmega8 được chia thành nhiều khối khác nhau, mỗi khối (modul) sẽ cung cấp xung clock cho các khối ngoại vi ứng dụng tương ứng.



Hình 2.16 Sơ đồ hệ thống xung clock cho Atmega8

## CHƯƠNG III

### NGÔN NGỮ C CHO AVR

#### 3.1 Khái niệm

##### 3.1.1 Các chú thích và tiền xử lý (PreProcessor) [1]

- Các chú thích.

Thông thường bắt đầu một chương trình là các chú thích về project cách chú thích phải bắt đầu bằng dấu // hay /\* *các chú thích* \*/ và được trình biên dịch bỏ qua khi biên dịch, chẳng hạn:

```
/******  
// comments placed in there  
// File: demo.c  
// Au thor: Pham Ngoc Dang Khoa  
// Date: 2007
```

- Các tiền xử lý.

#include: Dùng để chèn các file cần thiết vào project, các file này nên để trong thư mục inc của trình biên dịch CodeVisionAVR.

Ví dụ:

#include <mega8.h> cho phép sử dụng các thanh ghi của Atmega8. Tức báo cho trình biên dịch biết chúng ta đang sử dụng vi điều khiển Atmega8. Đây sẽ là dòng code đầu tiên trong chương trình C.

#define: Dùng định nghĩa một giá trị nào đó bằng các kí tự.

Ví dụ:

```
#define max 0xff
```

Định nghĩa max có giá trị là 0xff. Chú ý không có dấu chấm phẩy (;) ở cuối câu vì define chỉ là một macro chứ không phải là một lệnh. Macro cũng có thể có tham số.

Ví dụ:

```
#define SUM(a,b) a+b  
Main( )  
{  
// các lệnh khác  
Int I = SUM(2,3)  
// các lệnh khác
```

};

Thì **i** sẽ được gán thành  $i = 2+3 = 5$

### • Các kiểu dữ liệu (Data Types)

Ngoài các kiểu dữ liệu của C, CodeVisionAVR còn có kiểu dữ liệu **bit** là kiểu dữ liệu 1 bit, nên giải giá trị chỉ có 0 và 1. Kiểu bit chỉ hỗ trợ đối với khai báo biến toàn cục là chính. Với biến bit cục bộ, trình biên dịch chỉ cho khai báo tối đa 8 biến **bit**.

Ví dụ:

Bit a; //a là biến kiểu bit

Các kiểu khác được cho trong bảng dưới.

Kiểu dữ liệu	Kích cỡ (bit)	Giới hạn
Bit	1	0,1
Char	8	-128 đến 127
Unsigned char	8	0 đến 225
Signed char	8	-128 đến 127
Int	16	-32768 đến 32767
Short int	16	-32768 đến 32767
Unsigned int	16	0 đến 65535
Signed int	16	-32768 đến 32767
Long int	32	-2147483648 đến 2147483647
Unsigned long int	32	0 đến 4294967295
Signed long int	32	-2147483648 đến 2147483647
Float	32	$\pm 1.175e38$ đến $\pm 3.402e38$
double	32	$\pm 1.175e38$ đến $\pm 3.402e38$

### • Hằng

- Các hằng số được đặt trong bộ nhớ FLASH, chứ không đặt trong RAM.
- Không được khai báo hằng trong chương trình con.
- Giá trị 100 được hiểu là số thập phân (decimal), 0b101 để chỉ giá trị nhị phân (binary) và 0xff để chỉ giá trị thập lục (hexadecimal)

Ví dụ:

Const char a = 128; // hằng số a có kiểu char và có giá trị là 128.



### • Biến

- Biến gồm có biến toàn cục (global) là biến mà hàm nào cũng có thể truy xuất, và biến cục bộ (local) là biến mà chỉ có thể truy xuất trong hàm mà nó được khai báo.
- Biến toàn cục, nếu không có giá trị khởi tạo sẽ được mặc định là 0. Biến cục bộ, nếu không có giá trị khởi tạo sẽ có giá trị không biết trước.
- Biến toàn cục được lưu trữ trong các thanh ghi Rn, nếu dùng hết các thanh ghi thì sẽ chuyển sang lưu trữ trong vùng SRAM. Để ngăn cản các biến toàn cục được lưu vào các thanh ghi Rn, dù các thanh ghi này vẫn còn tự do, ta dùng từ khóa **volatile**.
- Biến toàn cục nếu không lưu trong các thanh ghi đa chức năng thì được lưu trữ trong bộ nhớ SRAM, còn biến cục bộ, nếu không lưu trong các thanh ghi đa chức năng, thì được lưu trữ trong vùng **data STACK**. Khi chương trình trả về giá trị cuối cùng cho hàm thì các biến cục bộ được lưu trữ trong stack sẽ bị khóa. Để biến cục bộ không bị xóa khi thoát khỏi hàm ta dùng từ khóa **static**.
- Biến **bit** toàn cục được cấp phát ở các thanh ghi R2 tới R14 của vi điều khiển, các bit được cấp phát từ R2 tới R14 theo thứ tự khai báo, nhắc lại là Atmega8 có 32 thanh ghi đa chức năng R0 đến R31.
- Trong chương trình C, nơi bắt đầu thực thi chương trình là điểm bắt đầu của hàm Main. Thực tế, khi biên dịch sang hợp ngữ (assembly), điểm bắt đầu của chương trình vẫn là vị trí **vector reset** (địa chỉ 0000h). Trước khi chạy tới vị trí chương trình main, chương trình hợp ngữ sẽ thực hiện khởi tạo các biến toàn cục,... Do đó, khi chạy vào hàm main, các biến toàn cục, mà thực chất là các ô nhớ (byte hay word), đã có giá trị khởi tạo sẵn. Với các biến cục bộ, trình hợp ngữ không khởi tạo trước giá trị.

- Ví dụ: khai báo biến cục bộ như sau:

```
Main ( )
{ unsigned char test = 9;
  Test+=1;
}
```

Sẽ dịch sang hợp ngữ là

```
LDI Rn, 0x09 ;// n tùy theo dòng chip và chương trình
SUBI Rn, 0xFF ;// trình ta viết, R17 chặn hạn
```

Như vậy, với biến cục bộ, khi nào sử dụng thì mới khởi tạo.

#### Ví dụ 1:

```
/* khai báo biến toàn cục */
char a;
int b;
/* có thể khởi tạo giá trị */
Long c = 0b1111;
/* chương trình con */
Int increment (void)
```

```

{
/* khai báo biến static */
Static int n ;
Return n++ ;
}
/* chương trình chính */
Void main (void) {
/* khai báo biến cục bộ */
Char d;
Int e;
/* có thể khởi tạo giá trị */
Long f = 16;
d = increment () ;
/* d = 1 */
e = increment () ;
/* e = 2, vì khi thoát khỏi hàm increment thì giá trị của biến static n vẫn
không bị xóa */

```

Ví dụ 2:

```

bit bit_mot ; // bit 0 của thanh ghi R2 được cấp cho biến bit_mot
bit bit_hai ; // bit 1 của thanh ghi R2 được cấp cho biến bit_hai

```

Đề ý là các biến kiểu bit trên là biến toàn cục, đối với biến bit cục bộ, trình biên dịch sẽ cất trong thanh ghi R15. Các thanh ghi R2 tới R14 cũng có thể được cấp phát cho biến thanh ghi (register variable), tùy vào các tùy chọn khi cấu hình cho trình biên dịch.

Biến volatile:

- Đề tương thích với các thiết bị ngoại vi khi ghép nối với vi điều khiển, chặn hạn bộ ADC, ghép nối với RTC.... Người ta dùng các biến volatile.  
*Biến Volatile là biến mà giá trị của nó không được thay đổi bởi chương trình, nhưng có thể được thay đổi bởi phần cứng.*

### • Chuyển đổi kiểu dữ liệu

Trong một biểu thức toán học, các toán hạng có thể có kiểu dữ liệu khác nhau, khi đó trình biên dịch sẽ tự động chuyển tất cả các toán hạng về cùng một kiểu duy nhất. Thứ tự ưu tiên chuyển đổi là:

Char -> unsigned char -> int -> unsigned int -> long -> unsigned long -> float

Ví dụ 1.

```
int a ;  
long c, b;  
c = a*b ; // a sẽ được tự động chuyển thành long
```

Ví dụ 2.

Phép nhân sau đây cho kết quả sai:

```
int a, b = 30000;  
long c ;  
c = a*b ;
```

Phép toán trên sẽ nhân a với b trước, với tích thu được là int bị tràn, rồi mới chuyển tích thu được sang long, rồi gán tích bị tràn này cho c. Để không bị tràn, ta sửa lại biểu thức trên như sau:

```
int a,b = 30000;  
long c ;  
c = (long) a*b ;
```

Lúc này a,b được chuyển thành long trước khi nhân, nên tích sẽ là long không bị tràn, rồi gán kết quả cho c.

### 3.1.2 Mảng (Array)

Mảng là một dãy các biến xếp liên tục nhau. Kí hiệu [ ] dùng để khai báo mảng. Mảng khai báo ngoài hàm gọi là mảng toàn cục (global array), mảng khai báo trong hàm gọi là mảng cục bộ (local array).

Ví dụ:

```
int global_array [4] = {1,2,3,4}  
// mảng có 4 phần tử (dạng nguyên) có khởi tạo giá trị ban đầu.  
global_array [0] = 9 ;  
// ghi giá trị 9 vào phần tử đầu tiên của mảng  
int multidim_array [2] [3] = {{1,2,3},{4,5,6}}  
// mảng đa chiều có khởi tạo giá trị ban đầu.
```

### 3.1.3 Hàm (Function)

- Hàm là đoạn chương trình thực hiện trọn vẹn một công việc nhất định.
- Hàm chia cắt việc lớn bằng nhiều việc nhỏ. Nó giúp cho chương trình sáng sủa, dễ sửa, nhất là đối với các chương trình lớn.
- Chương trình phục vụ ngắt (ISR) cũng có thể xem là một hàm, nhưng không có tham số truyền vào mà cũng không có tham số trả về.
- Giá trị trả về của hàm được lưu trong các thanh ghi R30, R31, R22, R23.

- **Con trỏ (Pointer)**

Những biến lưu trữ địa chỉ của một biến khác gọi là con trỏ (pointer). Có hai toán tử liên quan tới con trỏ là: **&** và **\***.

**&**: là toán tử lấy địa chỉ, có nghĩa là “địa chỉ của”.

**\***: là toán tử tham chiếu, có nghĩa là “Giá trị được trỏ bởi”.

Để sử dụng con trỏ ta phải khai báo nó. Kiểu khai báo như sau:

```
Type * pointer_name
```

Ví dụ:

```
Int *con_tro ;
```

*Đề ý là dấu sao (\*) mà chúng ta đặt khi khai báo một con trỏ chỉ có nghĩa rằng: Đó là một con trỏ và hoàn toàn không liên quan đến toán tử tham chiếu \* mà chúng ta đã nói ở trên. Đó đơn giản chỉ là hai tác vụ khác nhau được biểu diễn bởi cùng một dấu.*

Khi một biến con trỏ được khai báo, nó chưa chứa đựng giá trị nào cả, giống như các kiểu biến khác. Để gán địa chỉ cho con trỏ chúng ta cần phải gán giá trị cho con trỏ đó (tức khởi tạo con trỏ).

Ví dụ:

```
Int number;
int *con_tro;// khai báo biến con trỏ là một con trỏ nguyên
con_tro = &number ;// biến con_tro tới biến number
```

Sau khi khởi tạo, ta có thể sử dụng con trỏ bình thường trong các biểu thức.

Ví dụ:

```
int value1 = 5 ;
int value2 = 15 ;
int * mypointer;
mypointer = &value1; // con trỏ mypointer trỏ tới biến value1
*mypointer = 10; // giá trị của biến value1 = 10
mypointer = &value2; // con trỏ mypointer trỏ tới biến value2
*mypointer = 20; // giá trị của biến value2 = 20
```

### 3.1.3 Truy xuất các thanh ghi vào ra (accessing the i/o registers)

Việc truy xuất các thanh ghi I/O của AVR khá đơn giản, tất cả các thanh ghi I/O của AVR đã được khai báo trong file **io.h**. (hoặc file header cho từng chip cụ thể,

**mega8.h**) vào chương trình là có thể sử dụng các thanh ghi này. Chú ý là việc truy xuất bit trong các thanh ghi có địa chỉ 5Fh trở lên trong vùng nhớ SRAM là không thể thực hiện được.

Ví dụ:

```
include<io.h>
char temp ;
temp = PIND; // đọc giá trị ở cổng D vào biến temp
TCCR0 = 0x4F; // ghi giá trị 4Fh vào thanh ghi TCCR0
DDRD = 0x0c; // set bit 2 và 3 của thanh ghi DDRD
```

### 3.2 Tóm tắt cấu trúc điều khiển [1] [6]

#### 3.2.1 Cấu trúc điều kiện: **if** và **else**.

```
if (condition 1)
{
    Khối lệnh 1
}
else if (condition 2)
{
    Khối lệnh 2
}
else
{
    Khối lệnh khác
}
```

Ví dụ.

```
if (input == KEY_1) PORTD = 0x01;
else if (input == KEY_2) PORTD = 0x02;
else if (input == KEY_3) PORTD = 0x03;
else
    PORTD = 0x00
```

#### 3.2.2 Vòng lặp **While** và **do – While**

```
while (expression) statement ; // (1)
do statement while (condition); // (2)
```

Chức năng của (1) đơn giản chỉ là lặp lại **statement** khi điều kiện **expression** còn thỏa mãn.

Chức năng của (2) hoàn toàn giống vòng lặp **while** chỉ trừ một điều là điều kiện điều khiển vòng lặp được tính toán sau khi **statement** được thực hiện, vì

vậy `statement` sẽ được thực hiện ít nhất một lần ngay cả khi `condition` không bao giờ được thỏa mãn.

Ví dụ:

```
int i ;
while (I < 128)
{
    PORD = I;
    i = i*2 ;
}
```

Để có thể lặp vô hạn, ta dùng cấu trúc:

```
While (1)
{
    Statement
}
```

### 3.2.3 Vòng lặp for

`for (initialization; condition; increase) statement;`

Chức năng chính của nó là lặp lại `statement` chừng nào `condition` còn mang giá trị đúng như trong vòng lặp `while`. Nhưng thêm vào đó, `for` cung cấp chỗ dành cho lệnh khởi tạo và lệnh tăng. Vì vậy vòng lặp này được thiết kế đặt biệt lặp lại một hành động với một số lần nhất định.

- **Initialization** được thực hiện. Nói chung nó đặt một giá trị ban đầu cho biến điều khiển. Lệnh này được thực hiện chỉ một lần.
- **Condition** được kiểm tra, nếu nó là đúng vòng lặp tiếp tục còn nếu không vòng lặp kết thúc và `statement` được bỏ qua.
- **Statement** được thực hiện. Nó có thể có một lệnh đơn hoặc là một khối lệnh được bao trong một cặp ngoặc nhọn.
- Cuối cùng, **increase** được thực hiện để tăng biến điều khiển và vòng lặp quay trở lại kiểm tra.

Ví dụ:

```
For (int i = 1; I <= 128; i = i*2)
{
    PORD = I ;
}
```

Cấu trúc sau sẽ lặp vô hạn giống như cấu trúc `while (1)`

```
for (;)
{
    // Statement
}
```

### 3.2.4 Lệnh rẽ nhánh break và continue [1]

- Sử dụng break chúng ta có thể thoát khỏi vòng lặp ngay cả khi điều kiện để nó kết thúc chưa được thỏa mãn. Lệnh này có thể được dùng để kết thúc một vòng lặp không xác định hay buộc nó phải kết thúc giữa chừng thay vì kết thúc một cách bình thường.
- Lệnh continue làm cho chương trình bỏ qua phần còn lại của vòng lặp và nhảy sang lần lặp tiếp theo.

#### Ví dụ 1:

```
int n;
for (n=10; n>0; n--)
{
    PORTD = n ;
    if (n== 7)
    {
        break;
    }
}
```

Chương trình trên sẽ cho PORTD = 10, 9, 8, 7.

Chú ý, nếu sửa lại đoạn code trên như sau:

```
int n;
for (n=10; n >0; n--)
{
    if (n== 7)
    {
        break;
    }
    PORTD = n ;
}
```

Thì PORTD = 10, 9, 8.

#### Ví dụ 2.

```
For (int n =10; n>0; n--)
{
    if (n==5) continue;
    PORTD = n ;
```

Kết quả là PORTD = 10, 9, 8, 7, 6, 4, 3, 2, 1.

Chú ý, nếu sửa lại đoạn code trên như sau:

```
For (int n = 10; n>0; n--)
{
    PORTD = n
    if (n == 5) continue;
}
```

Thì PORTD = 10, 9, 8, 7, 6, 5, 4, 3, 2, 1

**Lệnh nhảy goto**

Lệnh goto cho phép nhảy vô điều kiện với bất kì điểm nào trong chương trình.

Ví dụ:

```
int n = 10;
loop :
PORTD = n ;
n-- ;
if (n>0) goto loop;
```

PORTD = 10, 9, 8, 7, 6, 5, 4, 3, 2, 1.

Loop là nhãn của chương trình, giống cách viết trong hợp ngữ.

Đề ý, lệnh n--, lệnh này sẽ giảm n đi 1. Ta có thể viết gọn hai câu lệnh:

```
PORTD = n ;
n-- ;
```

thành: PORTD = n--; lệnh này được hiểu là thực hiện phép gán trước rồi mới giảm n đi 1. Nếu sửa lại thành PORTD = --n ; thì sẽ giảm n đi 1 rồi mới thực hiện phép gán.

Tức tương đương với:

```
n-- ;
PORTD = n ;
```

Lúc này PORTD = 9, 8, 7, 6, 5, 4, 3, 2, 1.

Trường hợp ++n và n++ cũng hiểu tương tự, với dấu + chỉ sự tăng lên.

**3.2.5 Cấu trúc lựa chọn Switch**

```
Switch (expression) {
case constant1 :
block of instructions 1
break;
case constant2 :
block of instructions 2
break;
.....
.....
.....
default
default block of instructions
}
```



Switch hoạt động theo cách sau: switch tính biểu thức và kiểm tra xem nó có bằng constant1 hay không, nếu đúng thì nó thực hiện block of instructions 1 cho đến khi tìm thấy từ khóa break, sau đó nhảy đến phần cuối của cấu trúc lựa chọn switch. Còn nếu không, switch sẽ kiểm tra xem biểu thức có bằng constant 2 hay không. Nếu đúng nó sẽ thực hiện block of instructions 2 cho đến khi tìm thấy từ khóa break. Cuối cùng, nếu giá trị biểu thức không bằng bất kỳ hằng nào được chỉ định ở trên thì chương trình sẽ thực hiện các lệnh trong phần default nếu nó tồn tại vì phần này không bắt buộc phải có.

Có sự tương tự giữa lệnh Switch và cấu trúc if – else

```
Switch (x) {
    case 1:
        PORTD = 0x01 ;
        break;
    case 2:
        PORTD = 0x02;
        break;
    default:
        PORTD = 0x00;
}
```

Tương đương với:

```
If (x == 1)
{
    PORTD = 0x01;
}
Else if (x == 2)
{
    PORTD = 0x02;
}
else
{
    PORTD = 0x00;
}
```

### **3.3 Chèn hợp ngữ vào trong chương trình C**

Để có thể viết hợp ngữ trong chương trình C, ta dùng chỉ thị **#asm** và **#endasm**. Các thanh ghi R0, R1, R22 R23, R24, R25, R26, R27, R30, R31 có thể sử dụng trong đoạn chương trình hợp ngữ.

Ví dụ:

```
#asm
Sei // cho phép ngắt toàn cục
```

```
#endasm
```

Nếu chỉ viết trên một dòng thì có thể viết gọn là:

```
#asm (“sei”)
```

### **3.4 Tổ chức bộ nhớ SRAM [1]**

Trình biên dịch phân chia và quản lí bộ nhớ SRAM của AVR như sau: để truy xuất trực tiếp tới một địa chỉ nào đó trong các vùng nhớ của AVR ta dùng cách sau, cách này thích hợp khi ta muốn quản lí một khối nhớ cho một chức năng nào đó:

#### **Truy xuất bộ nhớ RAM**

Unsigned char \*Pointer;

Pointer= (unsigned char \*) 0x90h ; // truy xuất vào địa chỉ 0x90h của **SRAM**

#### **Truy xuất bộ nhớ Flash**

Flash unsigned char \*Pointer;

Pointer= (flash unsigned char \*) 0x90h ; //truy xuất vào địa chỉ 0x90h của **flash**

#### **Truy xuất bộ nhớ Eeprom**

Eeprom unsigned char \*Pointer;

Pointer = (eeprom unsigned char \*) 0x90h; truy xuất vào địa chỉ 0x90h của **eeprom**

### 3.5 Phần mềm lập trình cho bộ điều khiển từ xa AVR Atmega8

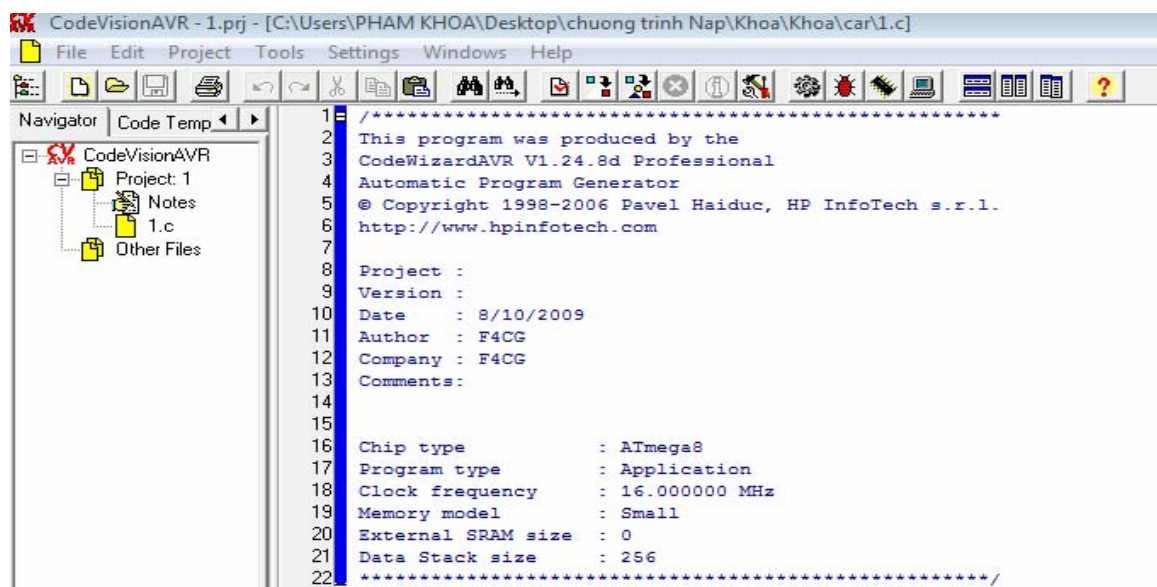
*Giới thiệu phần mềm CODEVISIONAVR [3]*



Hình 3.1 Chương trình lập trình ATmega8

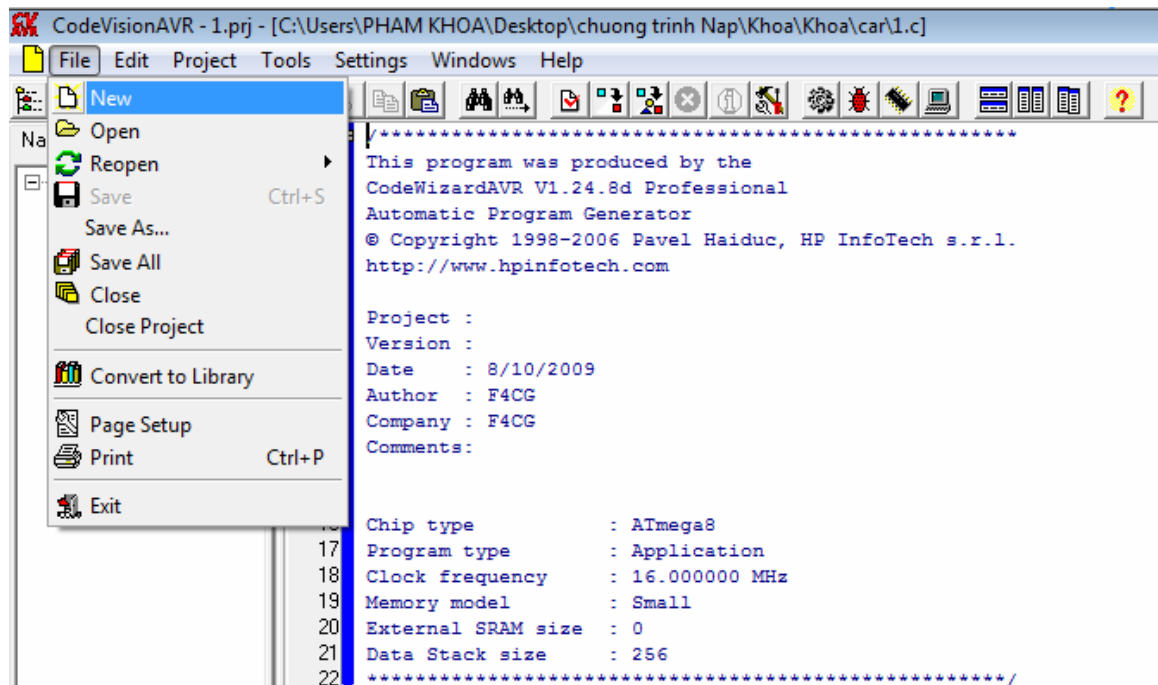
CodeVisionAVR là phần mềm chuyên dụng để lập trình chip AVR, ngôn ngữ lập trình C hay Asm và một số ngôn ngữ thông dụng khác đều có thể chạy trên nền Code Vision AVR. Trong nghiên cứu khoa học này em sử dụng chương trình C để tiến hành lập trình trên chip AVR để giao tiếp và truyền tín hiệu từ xa thông qua sóng RF (một dạng tín hiệu truyền của Atmega 8). [3]

#### **Giới thiệu sơ lược về phần mềm Codevision**



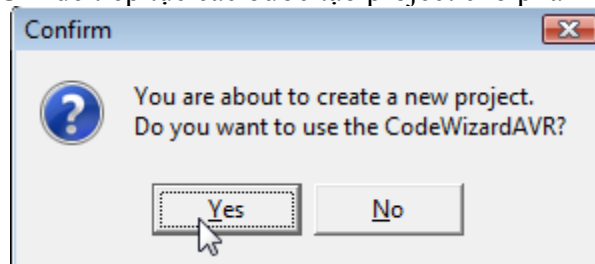
Hình 3.2 Giao diện lập trình của phần mềm CodeVisionAVR

Tạo một chương trình mới **FILE >> NEW**



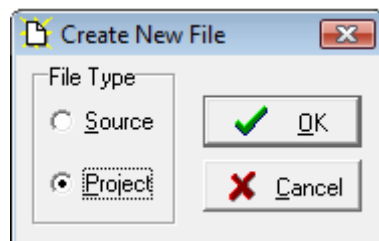
Hình 3.3 Các tạo một project trên CodeVision

Nhấn **OK** để tiếp tục các bước tạo project cho phần mềm:



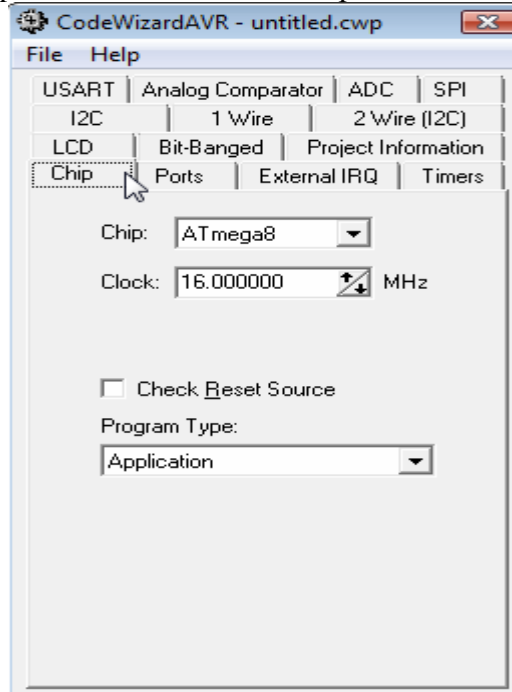
Hình 3.4 Các bước thực hiện

Check vào nút **Project >>> OK**



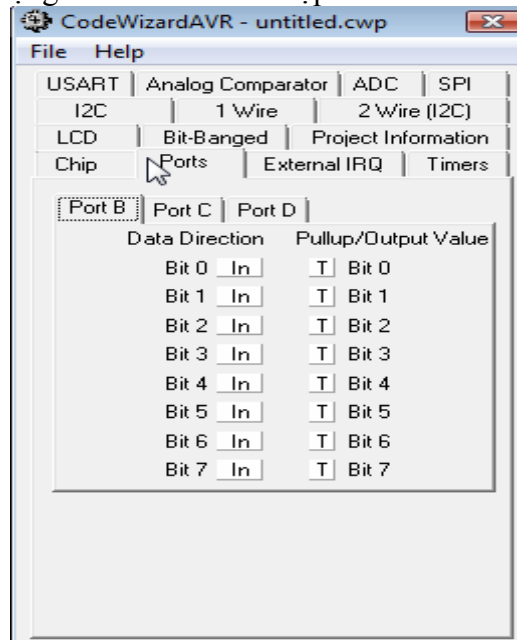
Hình 3.5 Các bước thực hiện

Chọn thẻ Chip để chọn loại AVR cần lập trình và tốc độ xung Clock



Hình 3.6 Cách chọn loại AVR

Gán các trạng thái cho Port cần lập trình



Hình 3.7 Các bước thực hiện

Sau các bước này là ta có thể tiến hành lập trình trên CodeVision AVR

### 3.6 Phương pháp và phần mềm nạp cho ATmega8 [4]

Phần mềm AVR Pro được sử dụng trong việc nạp dữ liệu cho bộ thu và phát của đề tài: “Điều khiển thiết bị bằng giọng nói từ xa”. Đây là phần mềm được tích hợp cho cổng Com 9 chân, và cổng USB của mạch nạp tương ứng. Trong quá trình thực hiện đề tài thì em dùng mạch nạp cổng USB để đưa dữ liệu vào chip Atmega 8.

#### 3.6.1 Tính năng mạch nạp [6]

Mạch nạp USB AVR sử dụng tốt nhất trên nền hệ điều hành Windows XP Professional và Vista.

- Nạp được hầu hết các dòng AVR và một số chip 89S của Atmel.
- Hỗ trợ đầy đủ các tác vụ nạp chip thông thường như: Ghi/xóa/đọc nội dung trong chip, kiểm tra lỗi sau khi nạp.
- Hỗ trợ khóa chip và lập trình fuse bit .
- Header nạp ISP chuẩn ICE 5x2 như dòng KIT thí nghiệm STK của ATMEL giúp kết nối thuận tiện.
- Tốc độ nạp cao, sử dụng được với hầu hết các trình biên dịch: Code Vision, AVR Studio...
- Cực kì đơn giản trong kết nối, cài đặt và sử dụng.

#### 3.6.2 Cách cài đặt Driver và nạp chương trình cho ATmega8 [5]

Để quá trình cài đặt driver cho mạch nạp AVR diễn ra suôn sẻ cần lưu ý: *không nên kết nối mạch nạp với KIT chứa chip cần nạp trước khi hoàn tất quá trình cài đặt driver.*

**Trình tự cài đặt như sau:**

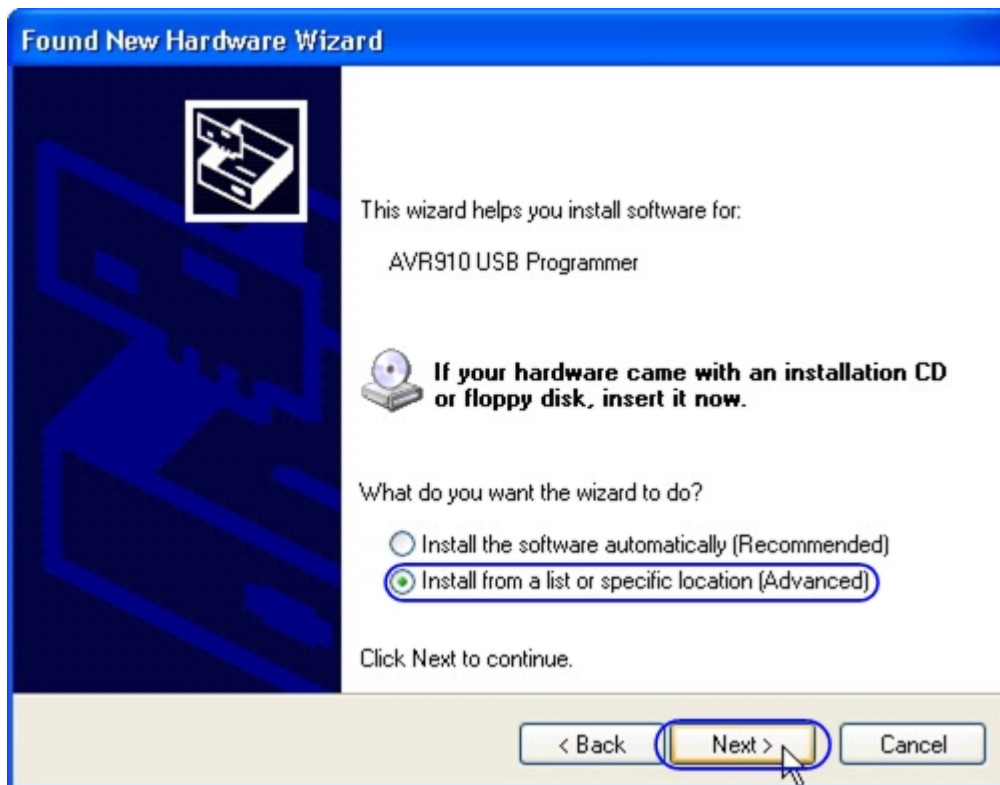
- Đưa đĩa CD kèm theo sản phẩm vào ổ CD.
- Cắm mạch nạp AVR vào cổng USB trên PC.

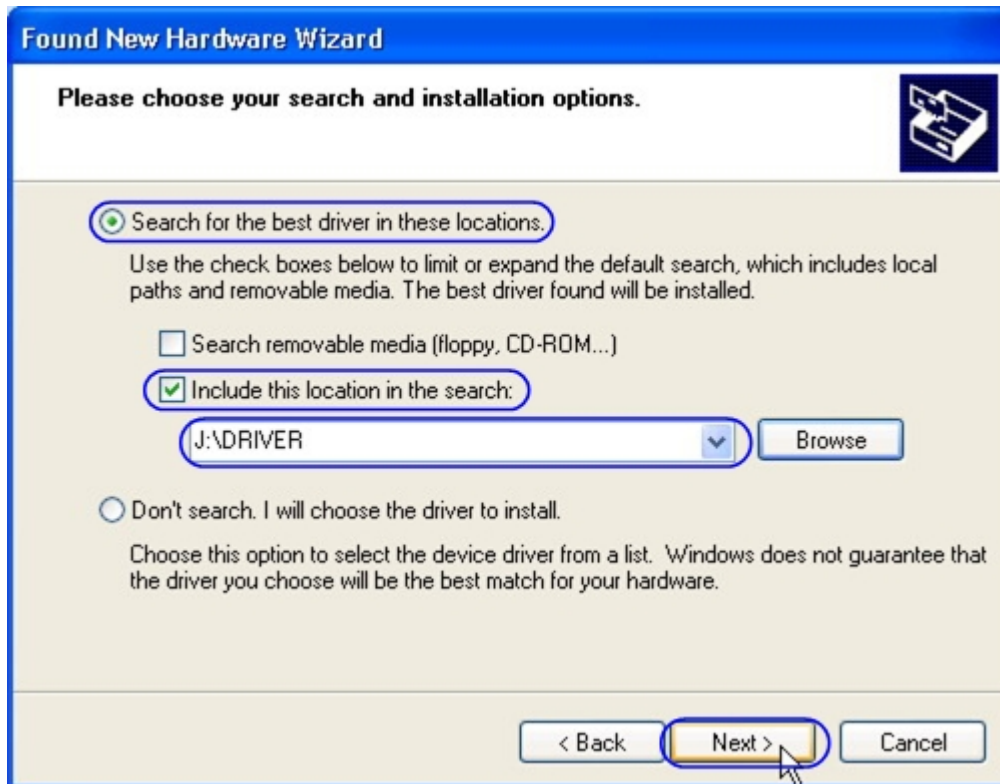
Trên màn hình hiện ra thông báo tìm thấy phần cứng mới dưới góc phải và hộp thoại yêu cầu người dùng chỉ ra driver tương ứng cho phần cứng mới này.



Chỉ định cụ thể đường dẫn đến thư mục chứa driver nằm trên đĩa CD theo các bước như hình dưới đây:

( Đường dẫn mặc định là [Tên ổ CD]\[Driver]\ )





- Trong quá trình cài đặt, nếu hộp thoại dưới đây xuất hiện, chọn “Continue anyway”...



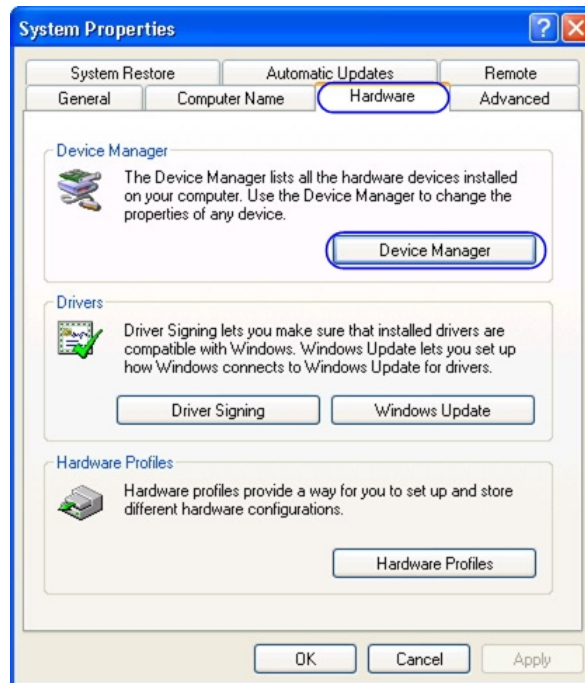


\* Hộp thoại sau đây thông báo quá trình cài đặt thành công.

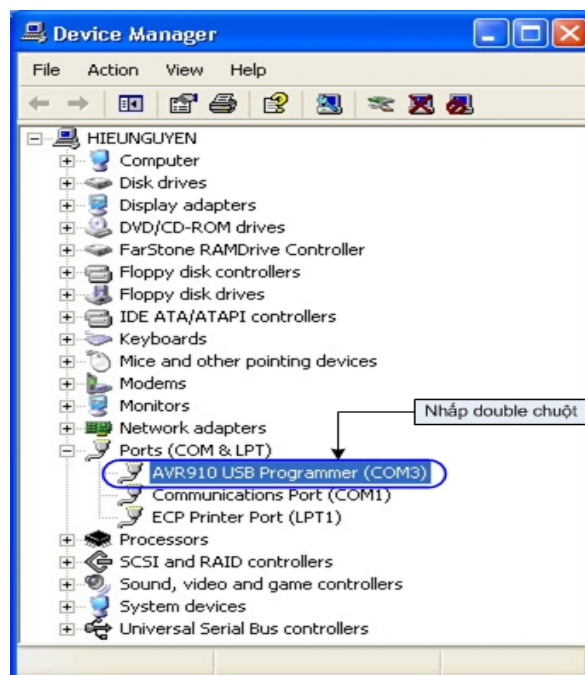


Sau khi cài đặt xong driver, thiết bị sẽ được PC xác nhận dưới hình thức cổng COM ảo, do đó cần tinh chỉnh thông số cổng COM này tối ưu nhất.

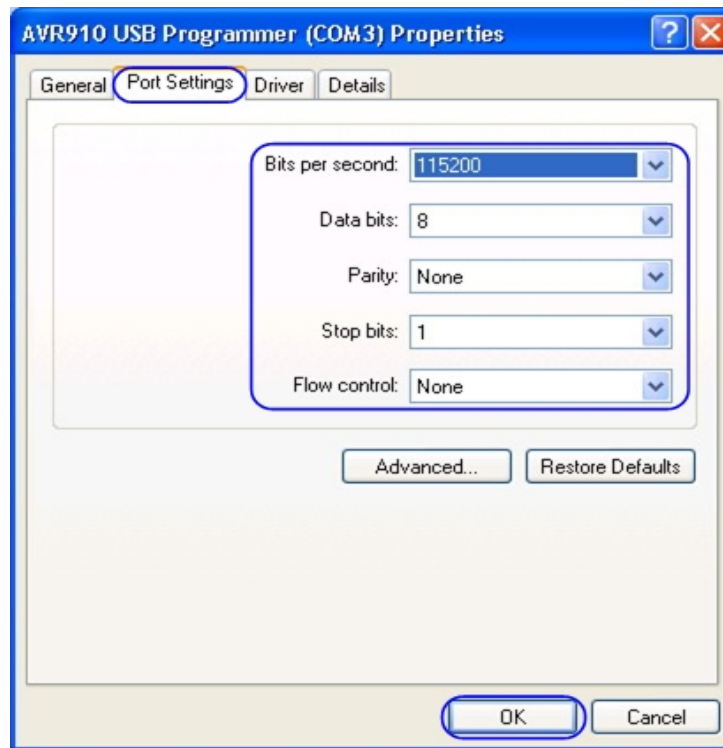
- Vào Start → Control Panel → System...



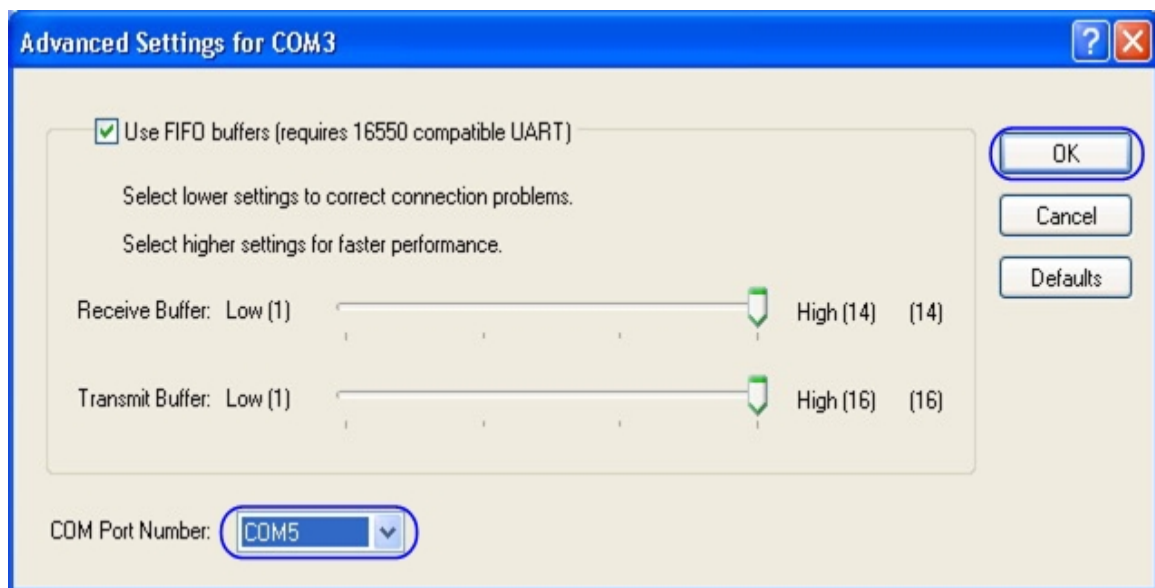
- Nhấp chọn như trên để vào chương trình quản lý thiết bị trên PC (Device Manager)... [1]



- Hiệu chỉnh các thông số như hình bên dưới:



- Một số trường hợp mạch nạp vẫn chưa hoạt động, nhấp chọn Advanced.. trên hình trên...

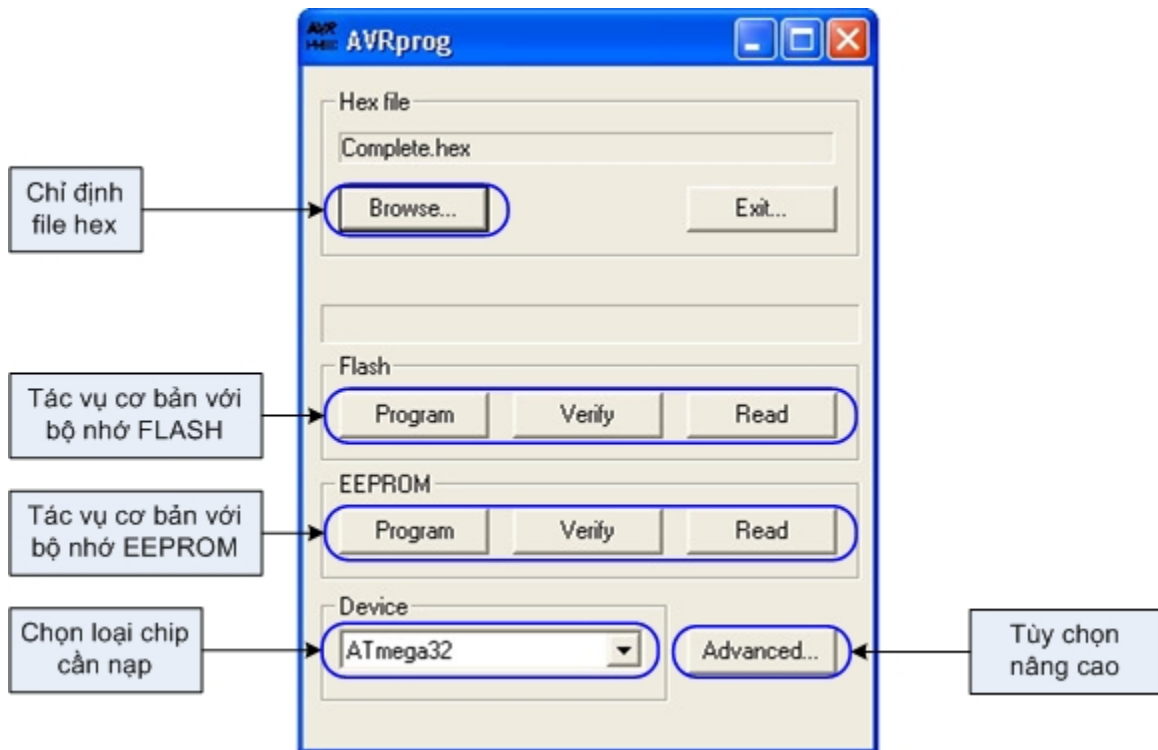


- Chọn COM Port Number khác, lưu ý nếu trong danh sách các cổng COM, cổng nào đang bị chiếm dụng (in use) thì không được chọn cổng COM này.  
→ Như vậy là hoàn tất bước cài đặt driver cho mạch nạp USB AVR 910.

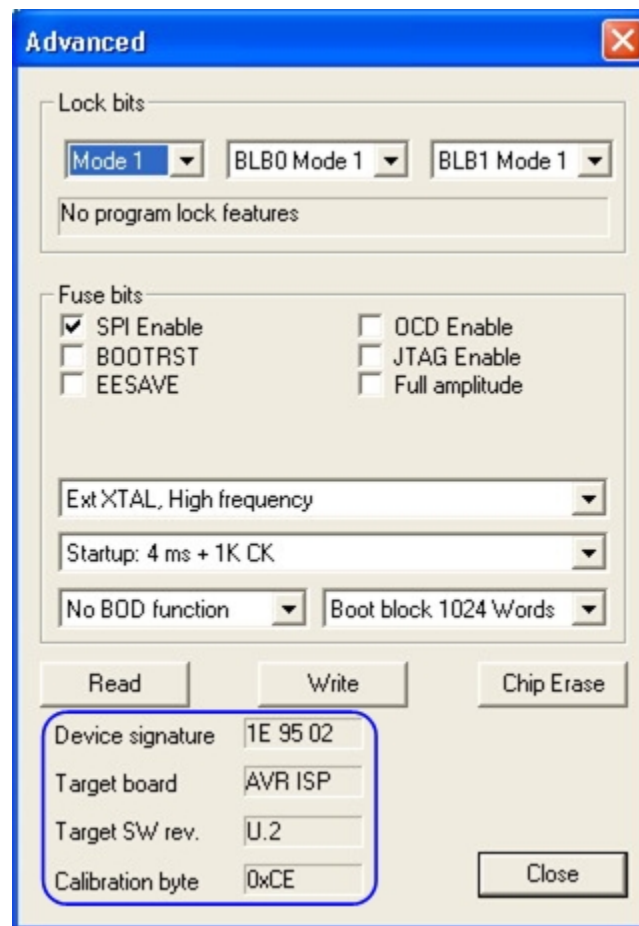
### 3.6.3 Phần mềm nạp

Các phần mềm nạp tương thích bao gồm AVR Studio và Code Vision. Chương trình AVRProg tuy giao diện đơn giản nhưng có ưu điểm là nạp rất nhanh, còn CodeVision thì ngược lại.

Sau đây sẽ hướng dẫn **Các bước nạp chip sử dụng chương trình AVR Prog trong AVR Studio:**



- Cắm mạch nạp vào cổng USB.
- Khởi động chương trình AVR Prog.
- Cắm cable nạp giữa mạch nạp với KIT chứa chip cần nạp tương ứng.
- Chọn loại chip tương ứng trên tab Device, nhấp chọn Advanced để test xem mạch nạp nhận ra chip chưa...Nếu không xuất hiện bất kì thông báo lỗi nào mà hiện ngay lên cửa sổ Advanced, trong khung đánh dấu như bên dưới hiển thị rõ ràng thông số của chip (không có dấu “?”) có nghĩa là mạch nạp hoạt động tốt và đã nhận ra chip.



→ Như vậy là có thể bắt đầu sử dụng mạch để nạp chip.

*Các mode, tùy chọn trong cửa sổ Advanced cần tìm hiểu kỹ trong datasheet của nhà sản xuất để có sự thiết đặt đúng.*

### 3.6.4 Cách khắc phục lỗi khi nạp

**Chương trình nạp không nhận ra phần cứng mạch nạp tương thích:**

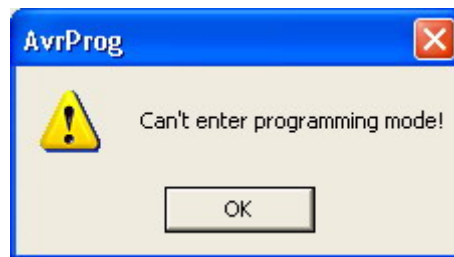
- Khi khởi động chương trình nạp xuất hiện thông báo lỗi như sau



- Các bước khắc phục:

- + Tắt chương trình nạp và rút mạch nạp ra khỏi PC. Lắp lại thao tác trong “Các bước nạp chip sử dụng chương trình AVR Prog trong AVR Studio” lần nữa.
- + Nếu chưa được, kiểm tra driver đã cài đặt chưa. Nếu đã cài đặt thành công, vào chương trình quản lý thiết bị, xem cổng USB (COM) đang sử dụng cho mạch nạp có bị chiếm dụng không và thử đổi sang cổng USB (COM) khác và tiến hành thử lại.

**Chương trình nhận ra phần cứng tương ứng và đã vào được chương trình nạp, nhưng khi nhấp Advanced thì không nhận ra chip, thay vào đó là bảng thông báo:**



- Các bước khắc phục:

- Hãy rút mạch nạp ra khỏi máy tính, tắt chương trình nạp, refresh và lắp lại thao tác trong “Các bước nạp chip sử dụng chương trình AVR Prog trong AVR Studio” lần nữa.
- Nếu không được, kiểm tra chuẩn kết nối phần cứng trên mạch chứa chip cần nạp.
- Khi cắm mạch nạp vào cổng USB khác trên máy, PC yêu cầu cài lại driver cho thiết bị.
- Thông thường, việc cài driver chỉ có tác dụng đối với mạch nạp cắm ứng với một cổng USB nhất định trên PC. Vì vậy, khi bạn cắm mạch nạp trên cổng USB khác trên máy, bạn vẫn phải cài driver để sử dụng mạch nạp trên cổng USB này.

## **PHẦN B**

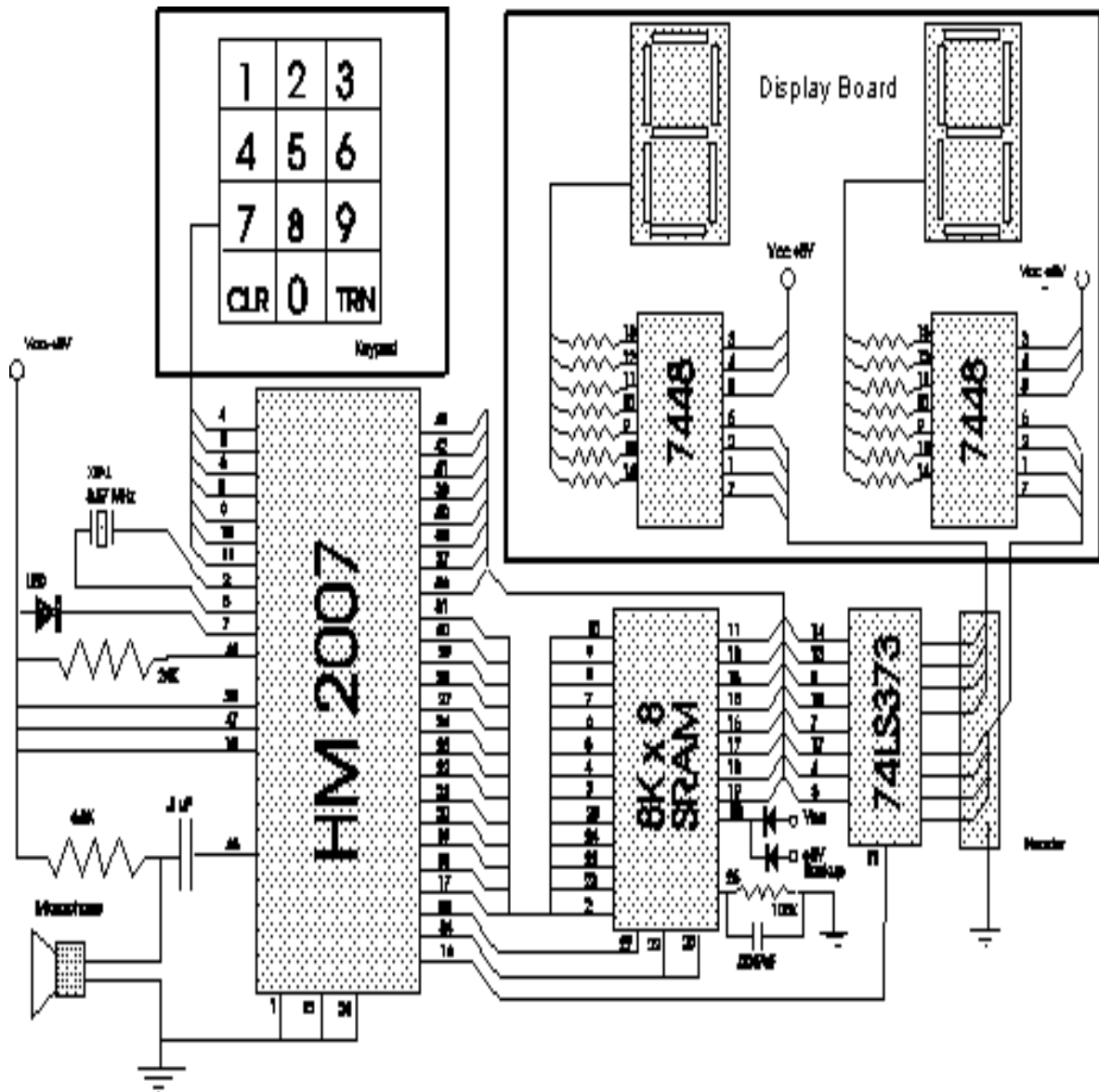
# **THIẾT KẾ THI CÔNG**

## CHƯƠNG 4

### THIẾT KẾ VÀ THI CÔNG MẠCH XỬ LÝ GIỌNG NÓI

#### 4.1 Sơ đồ nguyên lý mạch điều khiển tín hiệu giọng nói

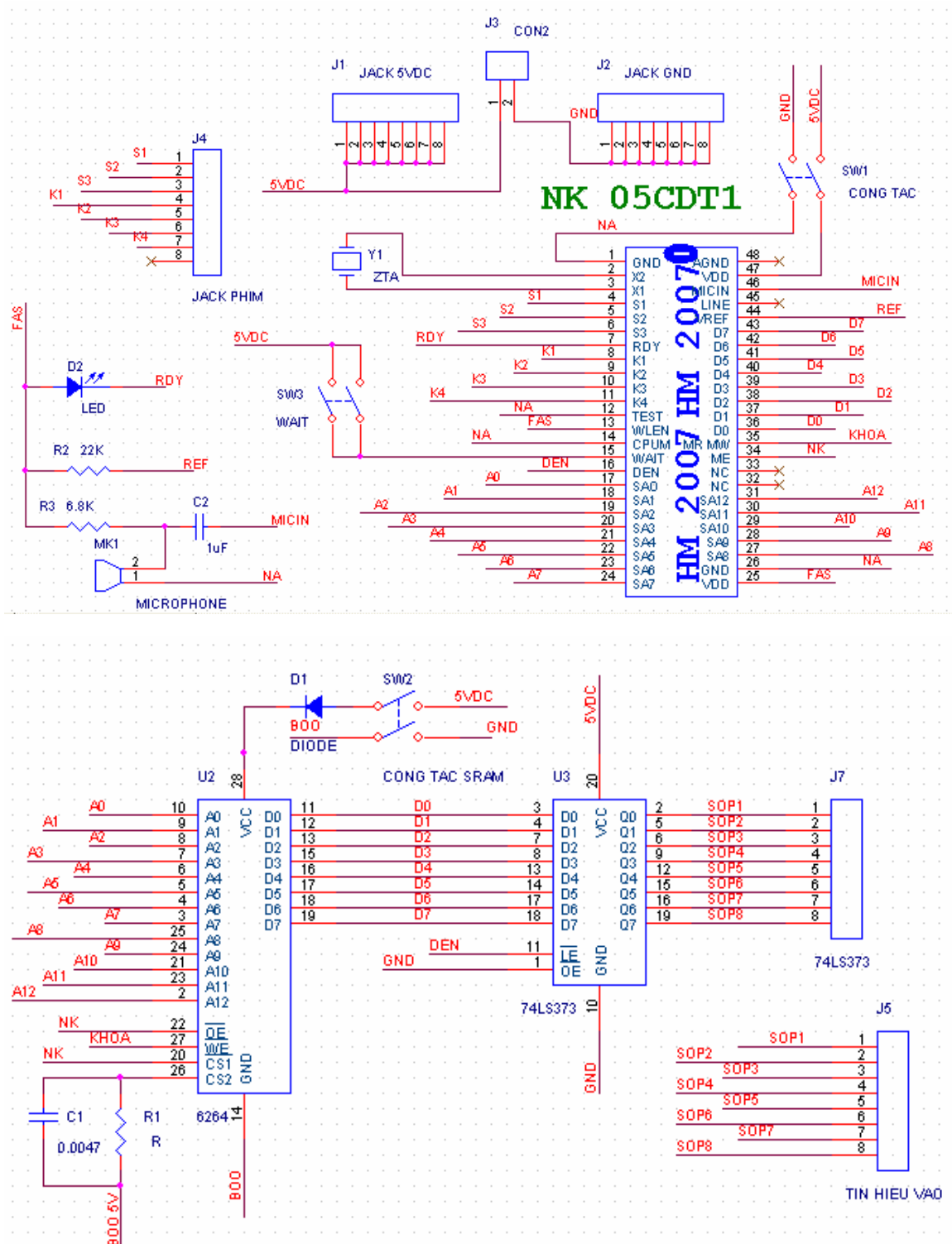
##### 4.1.1 Sơ đồ nguyên lý



Hình 4.1 Sơ đồ nguyên lý mạch xử lý giọng nói [2]



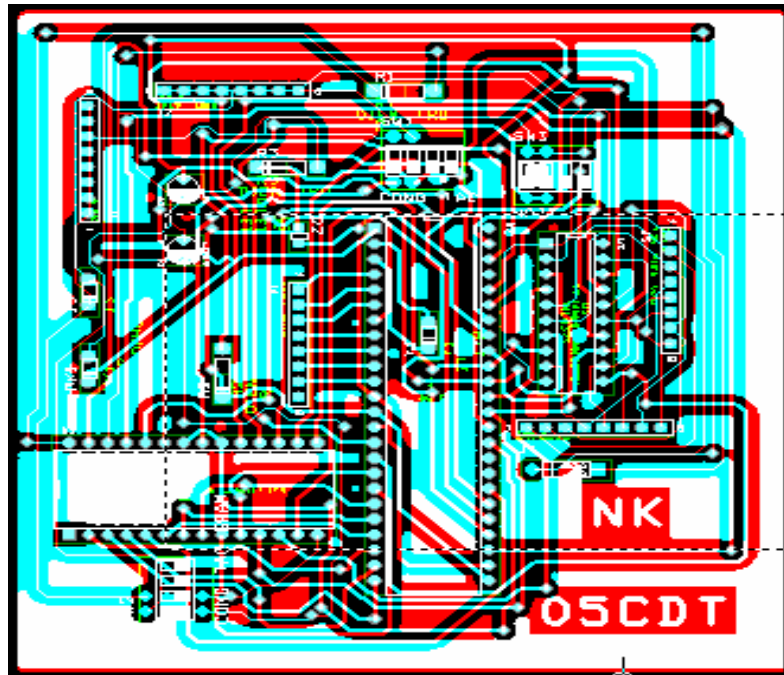
## BẢN THIẾT KẾ NGUYÊN LÝ TRÊN ORCAD



Hình 4.2 Mạch nguyên lý dùng ic HM2007 trong Capture (phần mềm Orcad)

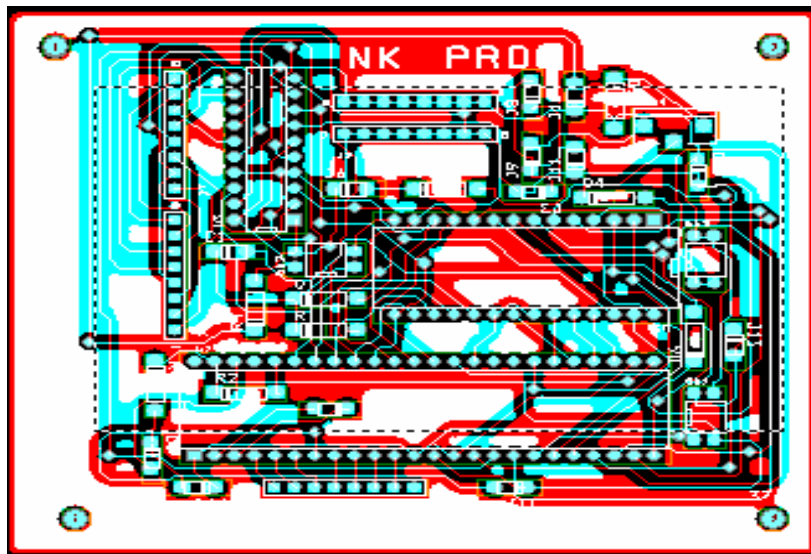
**4.1.2 Sơ đồ mạch in 2 lớp**

Board mạch 2 lớp thiết kế lần 1



Hình 4.3 Board HM 2007 (lần 1)

Board mạch 2 lớp thiết kế lần hai (sản phẩm cho thiết bị hoàn chỉnh)



Hình 4.4 Board HM2007 lần 2

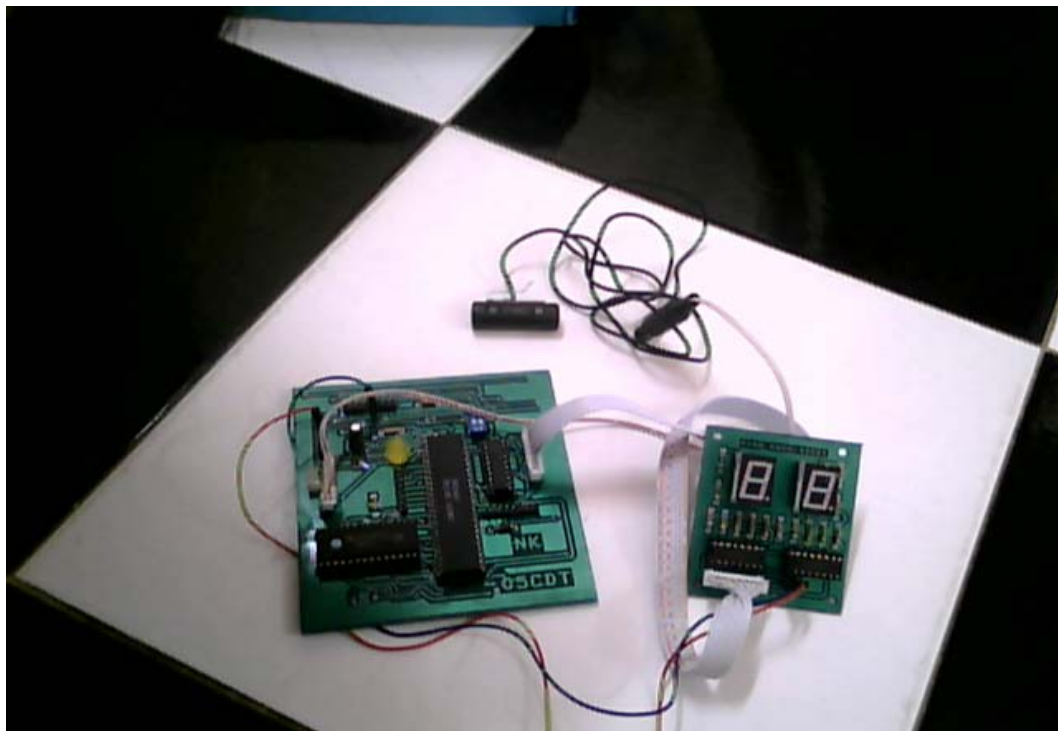
## **4.2 Các board mạch IC HM2007 đã thực hiện thử nghiệm**

Board một lớp thực hiện thử nghiệm lần đầu tiên



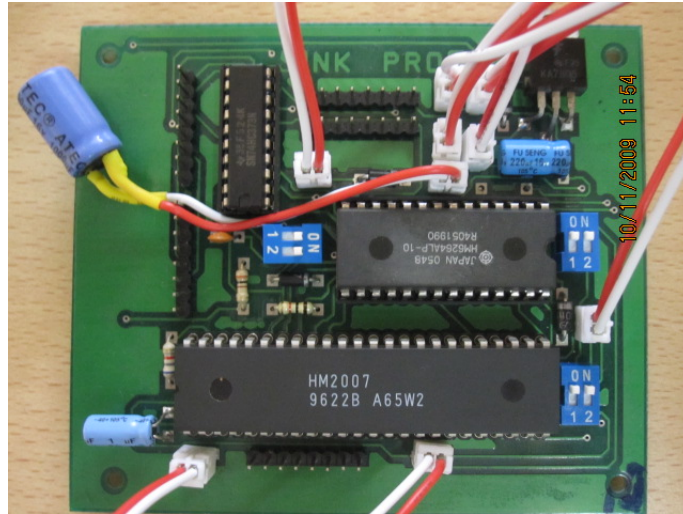
Hình 4.5: Board 1 lớp thiết kế thử nghiệm

Board hai lớp thực hiện lần đầu tiên



Hình 4.6: Board mạch 2 lớp thực tế

Board mạch in 2 lớp thiết kế cải tiến lần 2.



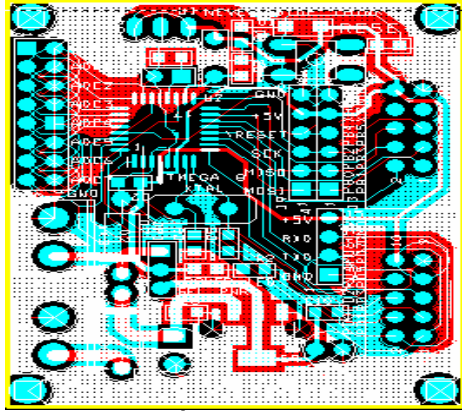
Hình 4.7 Board cho sản phẩm hoàn chỉnh



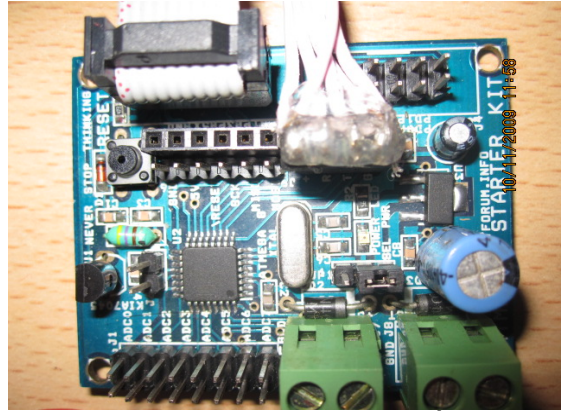


## **5.2 Sơ đồ thiết kế mạch in và thi công**

Mạch đơn giản gồm nguồn vào 5V cấp cho vi xử lý và đặt biệt trung tâm là một chip ATmega 8 với các jack ngõ ra điều khiển các thiết bị.

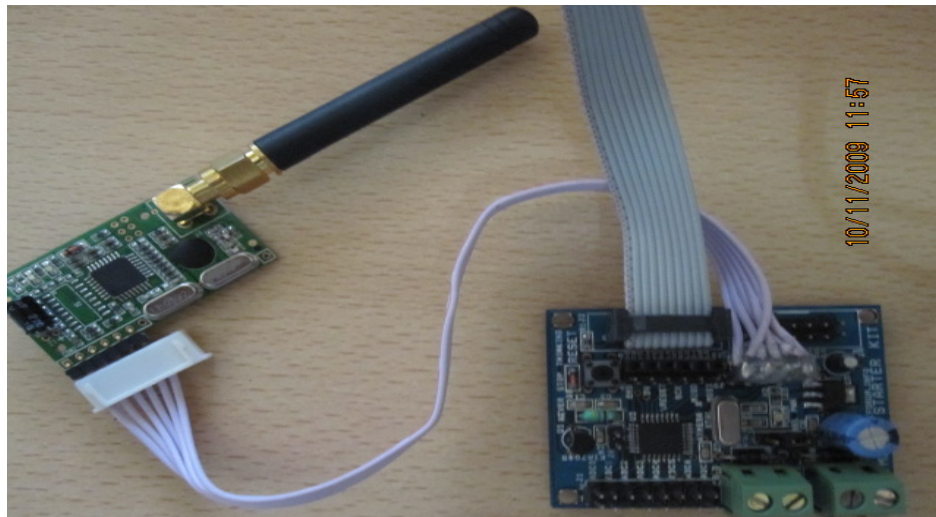


Hình 5.2 Sơ đồ mạch in trên layout



Hình 5.3 Mạch thực tế

## **5.3 Hình ảnh thực tế bộ Atmega8 của thiết bị**



Hình 5.4 Bộ thu (phát) từ xa của thiết bị

Do mạch chọn sử dụng là ic dán, nên khi thiết kế bản vẽ chi tiết và nguyên lý em cũng mạnh dạng sử dụng các linh kiện dán như (led, ic ổn áp,...) để giảm kích thước mạch và tăng tính thẩm mỹ cho hệ thống điều khiển.

Một hệ thống thu (hoặc phát) sử dụng 2 chip AVR Atmega8 xử lý dữ liệu và truyền đi, do đó toàn hệ thống sử dụng đồng loạt 4 chip Atmega8 cho cả bộ phận nhận và thu tín hiệu. Đây là bước cải tiến mới cho việc điều khiển thiết bị ở khoảng cách từ xa, đem lại sự tiện lợi, an toàn cho người sử dụng.

## CHƯƠNG 6

### QUÁ TRÌNH THIẾT KẾ CÁC MODUL NGÕ RA CỦA SẢN PHẨM

#### 6.1 Mục đích thiết kế các Modul ngõ ra.

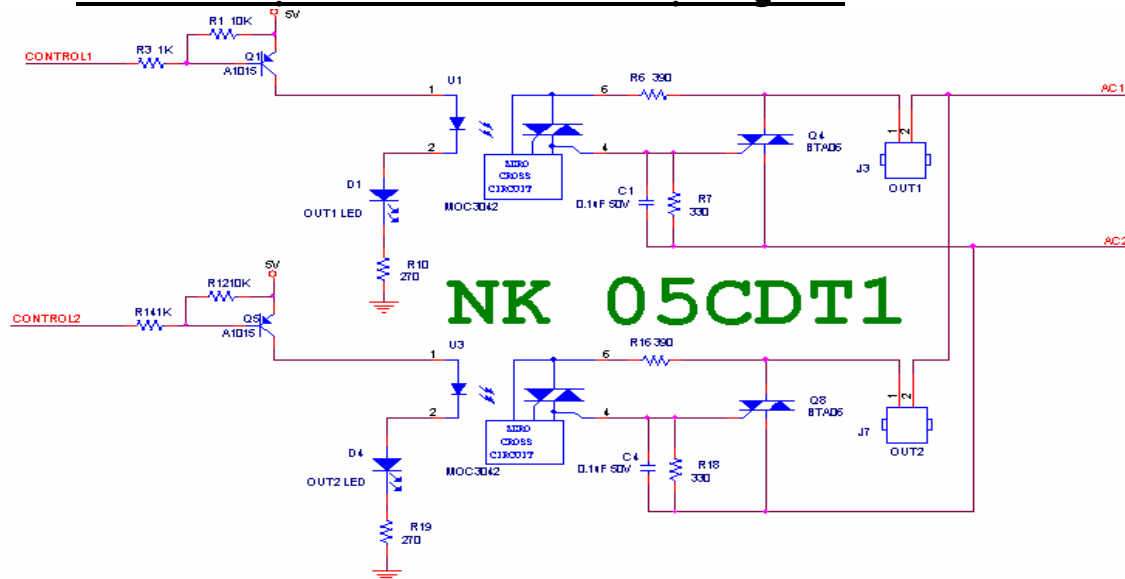
Vì tín hiệu sau khi xử lý và đưa ra dưới dạng các số nhị phân 0 và 1 sử dụng điện áp 5VDC. Do đó để điều khiển được các thiết bị có mức điện áp cao hơn ta phải thiết kế các mạch động lực cho thiết bị (các modul ngõ ra). Nhận thấy ở nước ta thì các thiết bị phần lớn sử dụng các mức điện áp sau:

- 5VDC: dành cho các IC và vi điều khiển.
- 24VDC: dành cho các cảm biến, van khí nén, động cơ robot...
- 220VAC: Cho các thiết bị điện dân dụng, sử dụng trong cuộc sống con người như, đèn, quạt, bếp điện ....

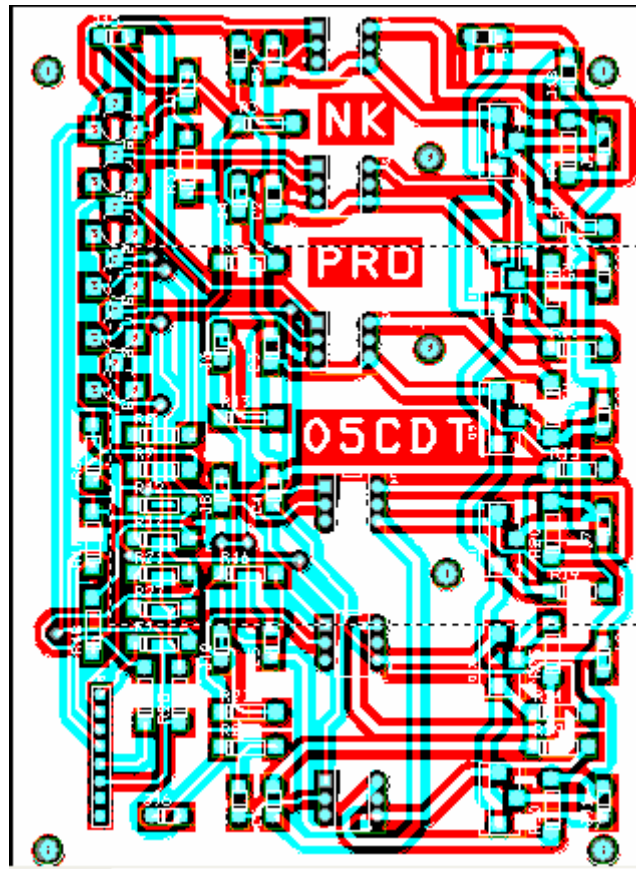
Qua các khảo sát trên thì em đã mạnh dạn thiết kế mạch nguyên lý và board 2 lớp cho các modul ngõ ra của thiết bị. Modul ngõ ra 5VDC và 24 VDC đã được thành công và đưa vào điều khiển các robocon hoạt động ổn định tại xưởng robocon trường đại học Lạc Hồng, các tay gấp phân loại sản phẩm, các board thực hành khí nén đều được chạy ổn định bằng tín hiệu giọng nói.

Đặt biệt đề tài chú trọng phát triển mạnh việc điều khiển các thiết bị trong nhà sử dụng mức điện áp 220VAC, do đó Modul 220VDC có thể nói là quan trọng nhất của thiết bị, Modul được thiết kế gồm 1 bộ điều khiển từ xa và mạch động lực, cùng với thiết kế cơ khí vỏ hộp mỹ quan đã làm tăng khả năng ứng dụng và dần dần đưa sản phẩm vào phục vụ cuộc sống.

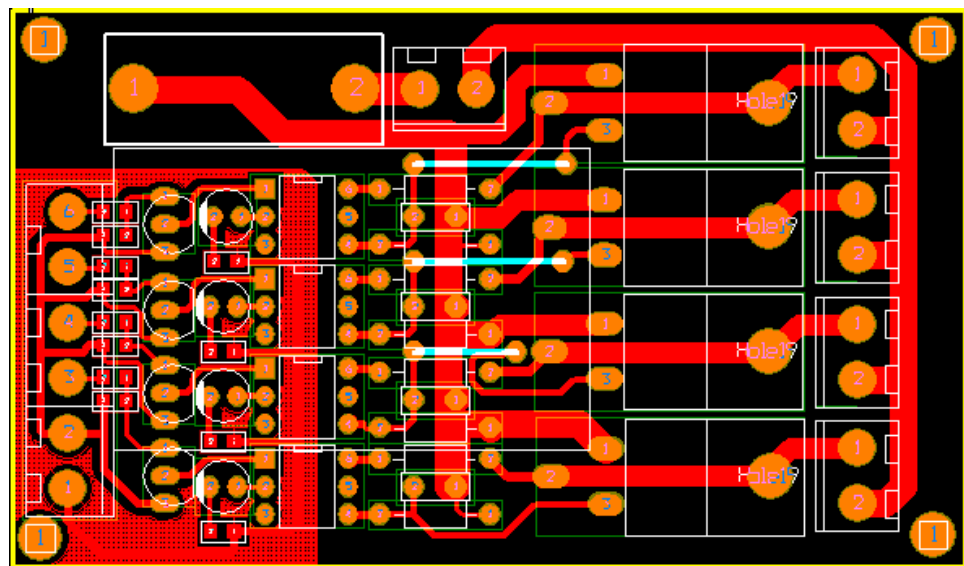
#### 6.2 Hình ảnh thực tế thiết kế và board mạch ngõ ra.



Hình 6.1 Bản thiết kế 1 modul ngõ ra 220VAC trên Capture

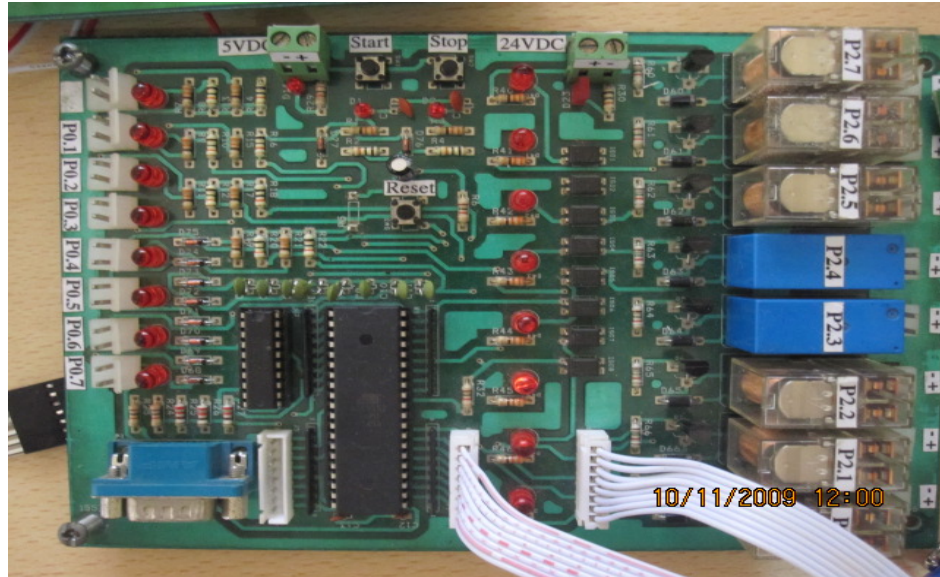


Hình 6.2 Sơ đồ mạch in modul 220VAC với 6 ngõ ra



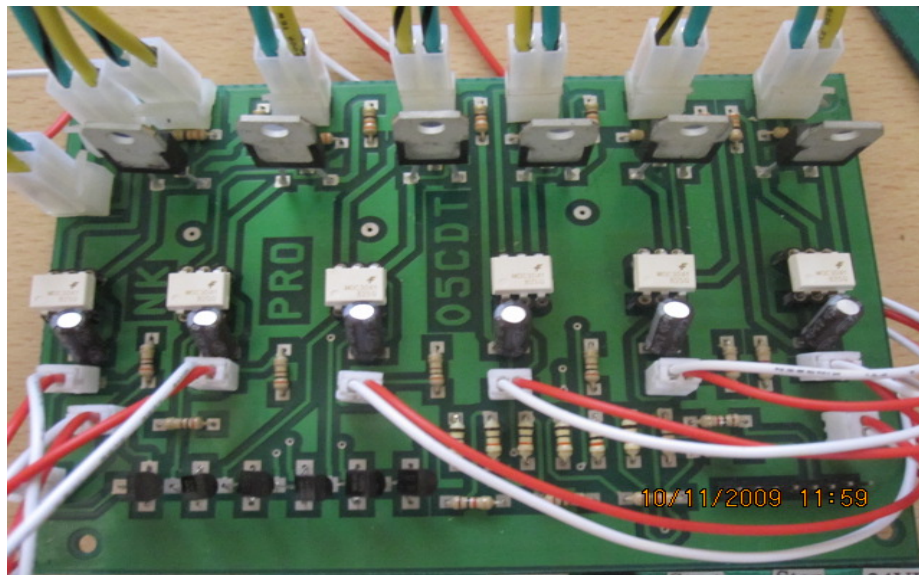
Hình 6.3 Sơ đồ mạch in thiết kế với 4 ngõ ra 220VAC





Hình 6.4 Modul ngõ ra 24VDC thực tế

Modul sử dụng điện 24VDC có tích hợp thêm các chức năng của ATME89C51 để có thể dễ dàng điều khiển thiết bị, hay có thể lấy thẳng tín hiệu nhận từ mạch phát điều khiển mà không thông qua 89C51.



Hình 6.5 Board 2 lớp của Modul 220VAC (với 6 ngõ ra)

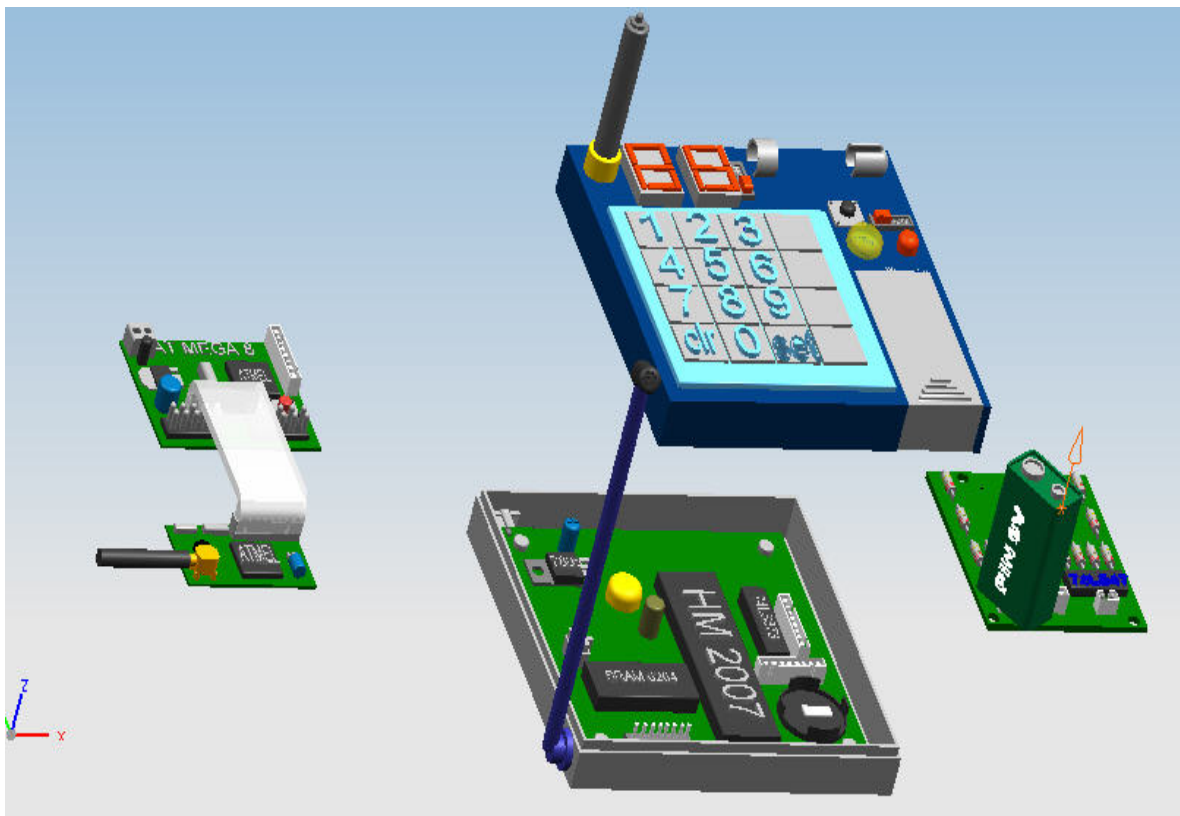
## CHƯƠNG 7

### THIẾT KẾ MẪU VỎ HỘP BÊN NGOÀI CHO THIẾT BỊ

#### 7.1 Ý tưởng thiết kế

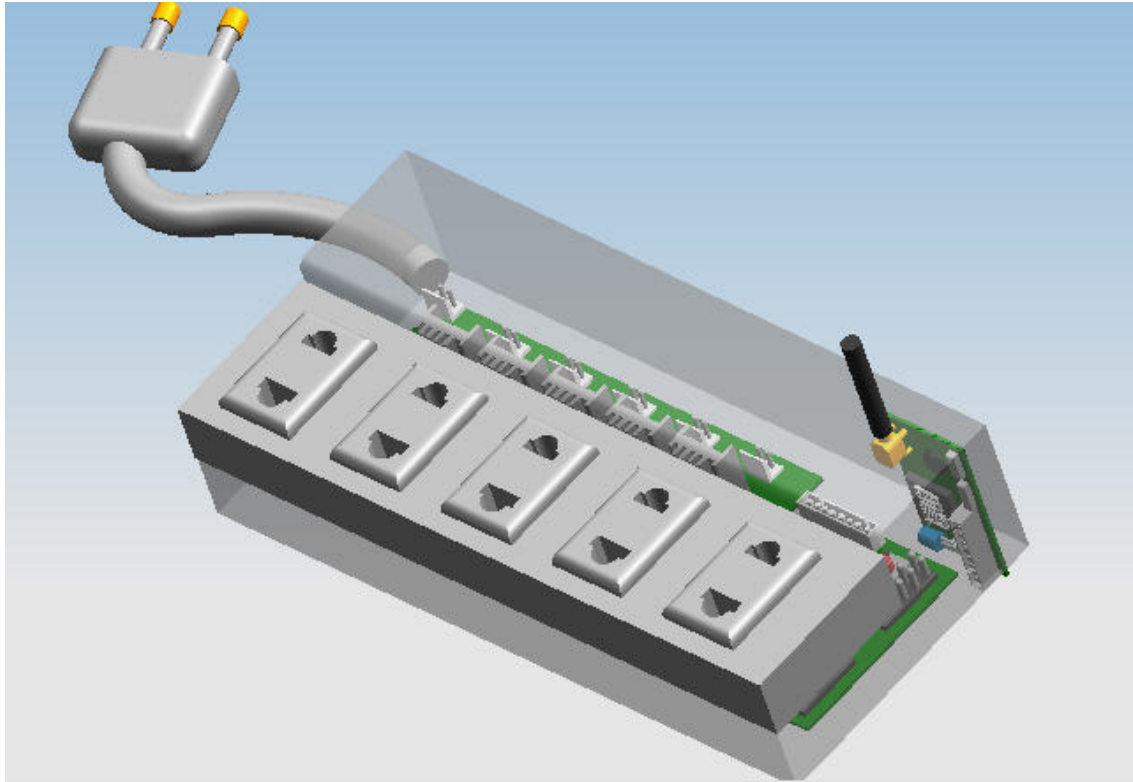
Nhận thấy đây là một đề tài có khả năng phát triển và ứng dụng cao, nên khi thực hiện đề tài em đã dành thời gian lên bản vẽ và thiết kế vỏ hộp cho thiết bị một cách hoàn chỉnh để có thể nhanh chóng đưa sản phẩm vào sử dụng thực tế. Phần vỏ ngoài sẽ bao gồm 2 thiết bị:

- Bộ vỏ của mạch điều khiển có kích thước 12 x 15 x 6 (cm), bên trong sẽ chứa đựng board xử lý giọng nói HM2007, mạch phát ATmega 8, bàn phím, pin được thiết kế như hình bên dưới. Bộ vỏ sẽ được thiết kế trên chất liệu gỗ, đảm bảo mỹ quan cho thiết bị thu phát.



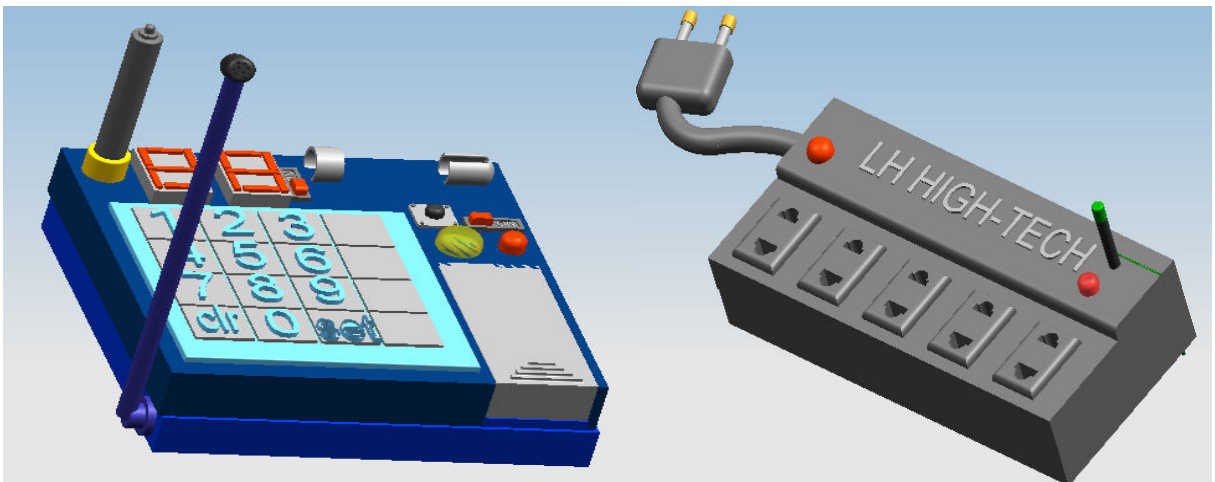
Hình 7.1 Thiết kế cơ khí khung vỏ của mạch điều khiển trên phần mềm Unigrafc

- Bộ vỏ của mạch động lực cũng được thiết kế bằng gỗ nhằm tăng tính mỹ quan và sang trọng của thiết bị kết hợp với các loại ổ cắm có sẵn trên thị trường để hoàn thiện sản phẩm. Khi thực hiện đề tài em chỉ thiết kế khung vỏ cho modul 220VAC, vì với các modul 5VDC hay 24VDC ta không cần thiết phải thiết kế khung vì khi sử dụng các modul này người ta thường gắn trực tiếp trên thiết bị sử dụng.



Hình 7.2 Thiết kế cơ khí Modul mạch động lực 220VAC trên phần mềm Unigrafic

## 7.2 Sản phẩm hoàn chỉnh trên phần mềm



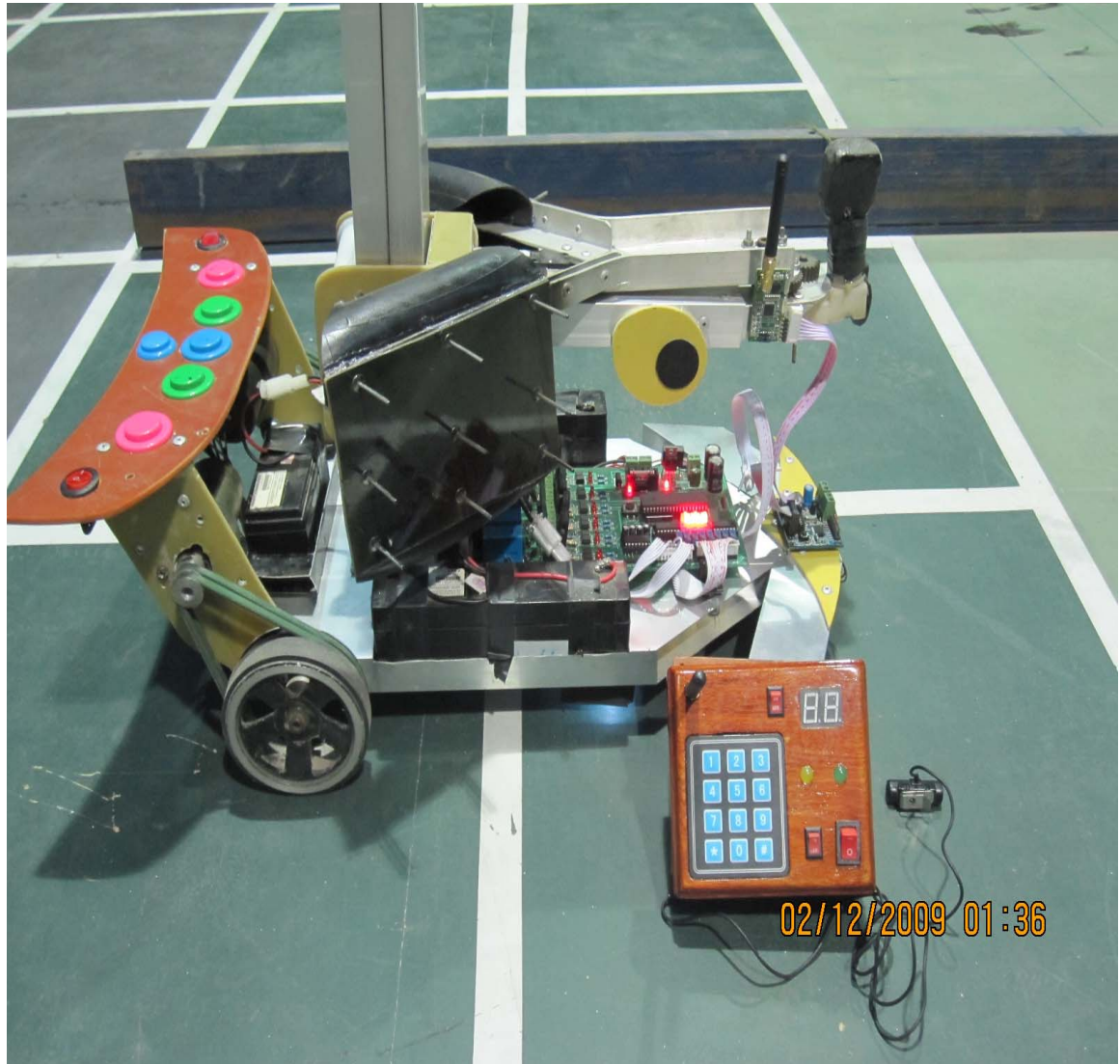
Hình 7.3 Sản phẩm hoàn chỉnh trên thiết kế.

## PHẦN C

# SẢN PHẨM



- **HỆ THỐNG ĐIỀU KHIỂN ROBOT SỬ DỤNG MODUL 24VDC**



Hình B. Điều khiển robot bằng giọng nói

Với 40 tín hiệu giọng nói có thể cài đặt trong thiết bị. Do đó robot sẽ được điều khiển một cách linh hoạt và đảm bảo thực hiện được mọi chức năng như người điều khiển mong muốn. Với công nghệ **“điều khiển thiết bị bằng giọng nói truyền từ xa”** người điều khiển có thể ra lệnh thực thi nhiệm vụ đối với robot trong phạm vi bán kính 200m, do đó có thể bao quát được quá trình hoạt động. Với kết cấu đơn giản, gọn nhẹ, truyền từ xa không cần dây kết nối đã mở ra cho đề tài nhiều hướng phát triển mạnh. Đặc biệt là các ứng dụng điều khiển các thiết bị và dây chuyền tự động hóa hiện đại.

- **BỘ ĐIỀU KHIỂN THIẾT BỊ 220VAC BẰNG GIỌNG NÓI TỪ XA**



Hình C. Bộ điều khiển giọng nói và modul 220VAC

- **KHẢ NĂNG ỨNG DỤNG – THÀNH QUẢ BƯỚC ĐẦU CỦA ĐỀ TÀI**

Không chỉ dừng lại ở việc truyền tải tín hiệu đi xa đề tài còn được thiết kế thêm các Modul ngõ ra thích ứng với nhiều thiết bị điều khiển tự động. Giờ đây chúng ta chỉ cần kết nối là có thể dùng giọng nói điều khiển chương trình của một con chip vi xử lý hoạt động ở điện áp 5VDC thực thi chương trình với modul ngõ ra 5VDC, có thể điều khiển các motor, relay, van khí nén thủy lực.... sử dụng 24VDC bằng giọng nói ở một nơi cách nó vài trăm mét với modul ngõ ra 24VDC. Và một ngày một mỗi sau khi đi làm về, bạn bước vào căn nhà của mình, toàn bộ các thiết bị điện 220VAC đều làm việc theo những gì bạn nói sẽ làm cho bạn cảm thấy thoải mái khi về nhà. Chỉ với một bộ điều khiển kích thước 8x12 cm trong tay, bạn có thể điều khiển rất nhiều ứng dụng của cuộc sống. Một vài ứng dụng trong giai đoạn đầu hoàn thành của thiết bị.

- + Dùng giọng nói ngồi từ xa điều khiển hệ thống tay gấp phân loại sản phẩm, thiết bị điều khiển là các xy lanh khí nén, các van đảo chiều một và hai cuộn coil, và các cảm biến công nghiệp. Hệ thống đã hoạt động ổn định như ta điều khiển thiết bị trên PLC Omron CPU 21.

- + Điều khiển hệ thống MPS trường đại học LẠC HỒNG

- + Hệ thống được kết nối từ bộ phận phát đến bộ phận thu tín hiệu và điều khiển robot tại xưởng robocon hoạt động chạy và thực thi các nhiệm vụ theo yêu cầu giọng nói của người điều khiển.

- + Với ngõ ra 220VAC, hệ thống dễ dàng điều khiển các thiết bị dân dụng trong cuộc sống gia đình như quạt, đèn, nồi em bé, đóng mở cửa....

- + Chế tạo các thiết bị điện an toàn (người dùng không tiếp xúc gần với điện thế cao) nhưng vẫn sử dụng và ra lệnh đóng ngắt nguồn điện một cách dễ dàng.

- + Hệ thống có thể dùng trong điều khiển các robot tự hành ở những vùng nguy hiểm, mà con người không thể đến được.

## KẾT LUẬN – KIẾN NGHỊ

### • Kết luận

Sau hơn một năm thực hiện thì đề tài “điều khiển thiết bị bằng giọng nói truyền từ xa” đã thu được rất nhiều thành công. Hệ thống tuy không phải lần đầu tiên được tìm hiểu và nghiên cứu ở Việt Nam, nhưng thành công của mạch nhận dạng và xử lý giọng nói trong đề tài này đã vượt xa các nghiên cứu khoa học của sinh viên các trường đại học khác ở độ ổn định và khả năng xử lý cũng như tính mỹ quan của thiết bị.

Đề tài không chỉ dừng lại ở mức nhận diện giọng nói, sau đó xuất tín hiệu ngõ ra, việc thiết kế, lập trình ứng dụng được các bộ thu phát từ xa sử dụng chip AVR Atmega8 là một thành công khá lớn của đề tài. Như vậy một người điều khiển có thể ngồi trong phòng và điều khiển các thiết bị mà mình mong muốn mà không phải đi bật công tắc, hay cần mạng internet phức tạp. Khả năng thu phát từ xa có thể lên tới 300m ở những nơi điều kiện lý tưởng (không có vật che khuất), đối với môi trường làm việc ở công sở và đời sống thì khoảng cách thu và nhận tín hiệu giữa bộ phận điều khiển và các modul ngõ ra là trong phạm vi bán kính 200m. Điều này giúp đề tài có nhiều điểm mạnh mà các thiết bị điện dân dụng khác không thực hiện được:

- Người điều khiển chỉ ngồi và ra lệnh với bộ điều khiển sử dụng điện áp 5VDC, tránh cho người tiêu dùng tiếp xúc trực tiếp với lưới điện 220VAC, có thể gây nguy hiểm cho con người khi tiếp cận với ổ cắm rò rỉ điện.
- Giảm được số lượng lớn dây nối khi các thiết bị ở khoảng cách xa.
- Một thiết bị điều khiển có thể điều khiển một lúc nhiều modul ngõ ra cùng lúc trong phạm vi 40 lệnh “giọng nói” được thu vào. Đây là nét đặc biệt của đề tài, trên thực tế tính toán thì, mỗi modul ngõ ra thiết kế 6 Jack cắm điện 220VAC, do đó chỉ cần 1 bộ điều khiển sử dụng chip HM 2007 là có thể điều khiển tương ứng 6 modul ngõ ra bằng sóng RF truyền từ xa.

### • Những khó khăn trong quá trình thực hiện đề tài

Dùng giọng nói điều khiển thiết bị còn là vấn đề khá mới mẻ ở Việt Nam và có rất ít thông tin nói về việc cấu thành thiết bị, do đó khi tiến hành thực hiện đề tài bản thân em đã trải qua rất nhiều khó khăn để có thể hoàn thành thiết bị.

- IC HM2007 là một IC chỉ sản xuất và được bán tại một số cửa hàng tại Mỹ, việc tìm và đặt hàng với số lượng lớn đã là một khó khăn của đề tài. Hơn 4 tháng, tìm hiểu qua internet và các mối quan hệ bạn bè thì em mới có được sản phẩm HM2007. Đây là một giai đoạn khá cần thiết để thực hiện đề tài, việc này đã giúp đề tài được tiến hành tốt ở giai đoạn đầu và dần đi sâu



vào các hệ thống quan trọng của sản phẩm. HM2007 là sản phẩm mà rất nhiều sinh viên và dân điện tử mong muốn có được để thực hiện các ý tưởng táo bạo của mình. Việc đưa được hm 2007 về Việt Nam có thể nói là một thành công bước đầu của đề tài nói riêng, và phục vụ nhu cầu của những người tiêu dùng Việt Nam nói chung.

- Các tài liệu về IC HM2007 còn mang tính giới hạn, hầu hết các chức năng phải do mình tìm tòi và thử nghiệm để có được một độ chính xác cao. Ngay cả datasheet của IC HM2007 down trên mạng cũng chỉ là bản nháp được đánh máy và sửa chữa bằng viết tay bởi người sản xuất ra nó. Sơ đồ nguyên lý của IC cũng chỉ là sơ đồ khối của các thiết bị, không có một sơ đồ kết nối cụ thể do đó rất khó thiết kế board thành công ngay lần đầu tiên. Bản thân em khi thực hiện đề tài cũng đã mất hơn 2 tháng để làm board và thiết kế thành board 2 lớp hoàn chỉnh phục vụ nhu cầu của đề tài.

- Tuy nhiên, thiết kế vào hoàn thành board xử lý giọng nói chỉ là thành công bước đầu của đề tài. Đây chỉ là một trong bốn phần cần hoàn thiện của đề tài “ điều khiển thiết bị bằng giọng nói từ xa”. Đây là một đề tài lớn mang tính công nghệ, đòi hỏi người thực hiện thông thạo và có kiến thức ở tất cả các lĩnh vực:

- + Khả năng sử dụng thành thạo máy tính và mạng internet.
- + Khả năng giao tiếp và linh hoạt trong cuộc sống.
- + Khả năng thiết kế board mạch 2 lớp.
- + Khả năng sử dụng phần mềm, và lập trình trên ngôn ngữ C.
- + Khả năng thiết kế bản vẽ cơ khí phần vỏ hộp
- + Tính sáng tạo và khả năng phát triển tư duy để tạo ra sản phẩm.

Trên thực tế, đây là một đề tài đã được rất nhiều người hướng đến thực hiện, nhưng họ chỉ dừng lại ở mức tìm hiểu lý thuyết hoặc hoàn thành board điều khiển nhận dạng giọng nói để điều khiển các thiết bị đơn giản qua cáp tín hiệu. Ở đây đề tài được phát triển thêm bộ điều khiển từ xa, giao tiếp trên một sóng truyền riêng, và các modul ngõ ra được thiết kế để dẫn hoàn thiện sản phẩm, để phục vụ cho nhiều ứng dụng trong cuộc sống.

- **Ưu điểm, khuyết điểm cần cải tiến của thiết bị**

- + **Ưu điểm:**

- Hệ thống có thể điều khiển bằng giọng nói hoặc **điều khiển bằng tay như một remote** điều khiển từ xa
- Sản phẩm được thiết kế nhỏ gọn, có thể mang theo và sử dụng ở mọi nơi.
- Thiết bị điều khiển sử dụng pin 9V, có bán phổ biến trên thị trường. Với một pin 9V sản phẩm có thể sử dụng và hoạt động liên tục được trên 2 tiếng. Và thời gian chờ lên tới 8 tiếng. Hoặc có thể kết nối nguồn trực tiếp với Adapter.
- Chỉ cần một thiết bị điều khiển 40 lệnh, có thể lập trình và điều khiển 6 modul ngỏ ra điều khiển thiết bị.
- Bộ điều khiển gọn nhẹ, thiết kế để mọi người có thể sử dụng một cách dễ dàng. (chỉ cần thời gian 5 phút là người sử dụng có thể đưa các lệnh bằng giọng nói của mình vào điều khiển một cách dễ dàng).
- Toàn bộ thiết bị chạy trên mạch in 2 lớp nên khả năng hoạt động ổn định cao.
- Có thể sử dụng micro không dây để tiện lợi cho việc di chuyển của người dùng.
- Thiết kế vỏ với vân gỗ mang lại vẻ sang trọng cho thiết bị, đồng thời tăng khả năng cách điện của các thiết bị với nhau.
- Đây là một sản phẩm mới, hoàn toàn chưa bán trên thị trường Việt Nam.

- + **Khuyết điểm:**

- Đề tài được hoàn thành và đã được phát triển thành một sản phẩm hoàn chỉnh, nên việc đưa sản phẩm ra thị trường là một vấn đề thiết yếu cần thực hiện, tuy nhiên với khả năng của một sinh viên năm cuối thì việc bán sản phẩm chỉ dừng lại ở việc bán lẻ các Modul (đặt biệt là bộ modul điều khiển bằng giọng nói)
- Ở mạch điều khiển bằng giọng nói IC nhớ SRAM 6264 có thể bị mất dữ liệu khi mất nguồn 5V cấp vào (hết pin). Tuy nhiên, việc nhập lại dữ liệu cho SRAM 6264 chỉ mất của người sử dụng khoản 5 phút để thiết lập lại tín hiệu giọng nói của mình.
- Tín hiệu giọng nói đưa vào ở khoản cách gần, và phạm vi hẹp.

- **Kiến nghị**

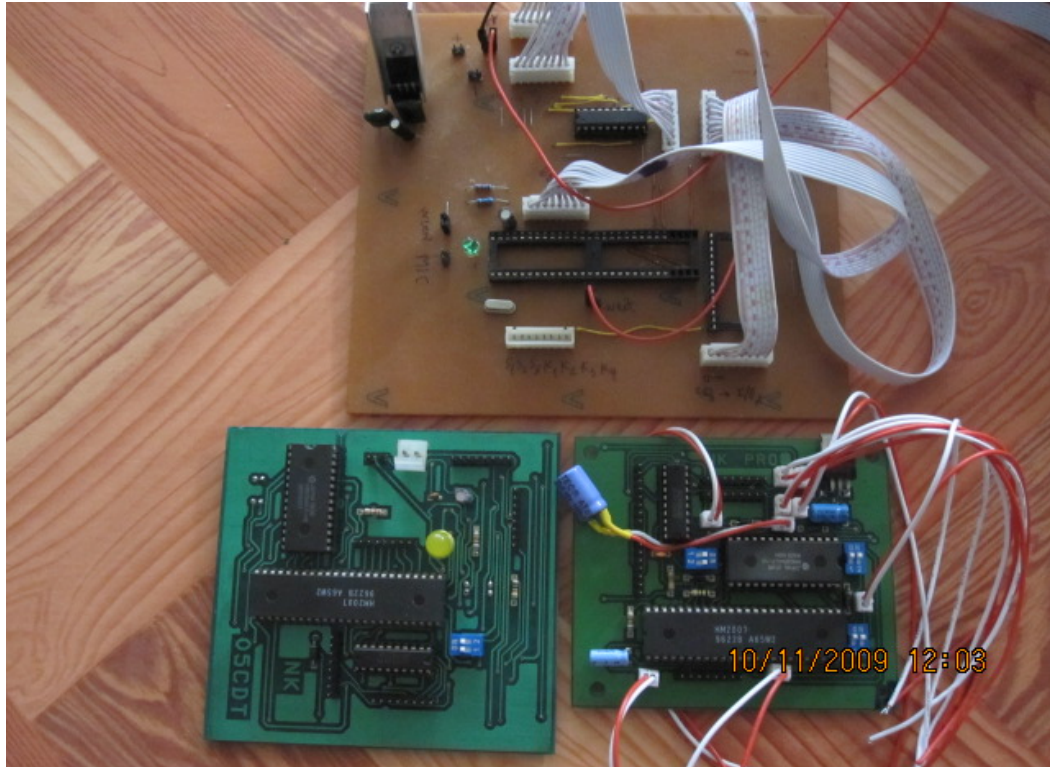
Đề tài “điều khiển thiết bị bằng giọng nói truyền từ xa” sau khi hoàn thành vào tháng 11 năm 2009 đã mở ra một hướng điều khiển thiết bị mới cho cuộc sống cũng như trong sản xuất công nghiệp. Tuy nhiên để đưa được vào phục vụ nhu cầu cuộc sống đề tài cần rất nhiều sự cải tiến về mặt kỹ thuật cũng như sự giúp đỡ trên nhiều phương diện từ trường đại học LẠC HỒNG và các đơn vị có trách nhiệm.

## DANH MỤC TÀI LIỆU THAM KHẢO

- [1] Tài liệu chip AVR và lập trình C của tác giả Lê Trung Thắng.
- [2] Tài liệu về IC HM2007 của sinh viên Nguyễn Minh Trí thực hiện năm 1998
- [3] Phần mềm lập trình CodeVisionAVR Help được phát triển bởi Pavel Haiduc and HP InfoTech.
- [4] Tài liệu tham khảo viết báo cáo được dịch từ datasheet tiếng Anh của thiết bị (HM 2007, Sram 6264, ATmega 8,.....) trên trang:  
[www.alldatasheet.com](http://www.alldatasheet.com)  
Các từ khóa : “HM2007”, “6264”, “Atmega8”, “ 74LS373”, “74LS47”, “Moc3402”, “BTA06”, “A1015”.
- [5] [www.google.com](http://www.google.com) , [www.wikipedia.org](http://www.wikipedia.org)  
Từ khóa: “ điều khiển bằng giọng nói”, “ Voice command”
- [6] Các trang web tham khảo tài liệu.  
[www.diendandientu.com](http://www.diendandientu.com)  
[www.dientuvienthong.net](http://www.dientuvienthong.net)  
[www.dientuvietnam.net](http://www.dientuvietnam.net)  
[www.5giay.vn](http://www.5giay.vn)

## PHỤ LỤC

- **Hình ảnh cải tiến board mạch chủ của thiết bị.**



Board mạch sản phẩm được thiết kế lại nhiều lần để tăng tính năng sử dụng và giảm kích thước của sản phẩm “Điều khiển thiết bị bằng giọng nói truyền từ xa”

- **Chương trình chính lập trình cho bộ điều khiển từ xa**

### Chương trình chính mạch phát tín hiệu:

```
While (1)
{
if (PINB.0==1 && PINB.1==0 && PINB.2==0 && PINB.3==0) //
    putchar ( 'a' );
if (PINB.0==0 && PINB.1==1 && PINB.2==0 && PINB.3==0) //
    putchar ( 'b' );
if (PINB.0==1 && PINB.1==1 && PINB.2==0 && PINB.3==0) //
    putchar ( 'c' );
if (PINB.0==0 && PINB.1==0 && PINB.2==1 && PINB.3==0) //
    putchar ( 'd' );
if (PINB.0==1 && PINB.1==0 && PINB.2==1 && PINB.3==0) //
    putchar ( 'e' );
```

```

if (PINB.0==0 && PINB.1==1 && PINB.2==1 && PINB.3==0) //
    putchar ( 'f');
if (PINB.0==1 && PINB.1==1 && PINB.2==1 && PINB.3==0) //
    putchar ( 'g');
if (PINB.0==0 && PINB.1==0 && PINB.2==0 && PINB.3==1) //
    putchar ( 'h');
if (PINB.0==1 && PINB.1==0 && PINB.2==0 && PINB.3==1) //
    putchar ( 'i');
};

```

### **Chương trình chính mạch thu tín hiệu**

```

While (1)
{
    Char ;
    h = getchar ( ) ;
    switch 9 (h)
    {
        case 'a'
            PORTB = 0 x FE;
            break;
        case 'b'
            PORTB = 0 x FD;
            break;
        case 'c'
            PORTB = 0 x FB;
            break;
        case 'd'
            PORTB = 0 x F7;
            break;
        case 'e'
            PORTB = 0 x EF;
            break;
        case 'f'
            PORTB = 0 x DF;
            break;
        case 'g'
            PORTB = 0 x FF;
            break;
        case 'h'
            PORTB = 0 x FF;
            break;
        case 'i'
            PORTB = 0 x FF;
            break;
    };
};

```

