**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

**KHOA CÔNG NGHỆ ĐIỆN TỬ VÀ TRUYỀN THÔNG**



**BÀI GIẢNG LẬP TRÌNH NHÚNG CƠ BẢN**

**( Tài liệu lưu hành nội bộ)**

**Thái Nguyên, Năm 2021**

**Thái nguyên, năm 2014**

* **Thái Nguyên, năm 2013**

**MỤC LỤC**

[**MỤC LỤC** 2](#_Toc61613483)

[CHƯƠNG 1: TỔNG QUAN VỀ HỆ THỐNG NHÚNG 5](#_Toc61613484)

[1.1. Khái niệm hệ thống nhúng 5](#_Toc61613485)

[1.2. Lịch sử phát triển 5](#_Toc61613486)

[1.3. Xu hướng phát triển trong tương lai 6](#_Toc61613487)

[1.4. Đặc điểm của hệ thống nhúng 7](#_Toc61613488)

[1.4.1. Giao diện 8](#_Toc61613489)

[1.4.2. Kiến trúc CPU 8](#_Toc61613490)

[1.4.3. Thiết bị ngoại vi 9](#_Toc61613491)

[1.4.4. Công cụ phát triển 9](#_Toc61613492)

[1.4.5. Độ tin cậy 10](#_Toc61613493)

[1.5. Các thành phần cơ bản trong kiến trúc phần cứng hệ thống nhúng 11](#_Toc61613494)

[1.5.1. Đơn vị xử lý trung tâm CPU 12](#_Toc61613495)

[1.5.2. Bus địa chỉ, dữ liệu và điều khiển 14](#_Toc61613496)

[1.5.3. Bộ nhớ 16](#_Toc61613500)

[1.6. Một số nên phần cứng nhúng thông dụng 16](#_Toc61613501)

[1.6.1. Vi điều khiển họ 8051 16](#_Toc61613502)

[1.6.2. Vi điều khiển PIC 18](#_Toc61613503)

[1.6.3. Nền tảng phần cứng STM32 20](#_Toc61613504)

[1.6.4. Nền tảng phần cứng KL46 20](#_Toc61613505)

[1.6.5. Nền tảng phần cứng Arduino 21](#_Toc61613506)

[CHƯƠNG 2: HỌ VI ĐIỀU KHIỂN NHÚNG CƠ BẢN 8051 24](#_Toc61613508)

[2.1. Tổng quan 24](#_Toc61613509)

[2.2. Sơ đồ khối và chức năng các khối của chip 8051 25](#_Toc61613510)

[2.3. Sơ đồ chân và chức năng các chân của chip 8051 26](#_Toc61613511)

[2.3.1. Port 0 26](#_Toc61613512)

[2.3.2. Port 1 27](#_Toc61613513)

[2.3.3. Port 2 27](#_Toc61613514)

[2.3.4. Port 3 28](#_Toc61613515)

[2.3.5. Các chân vào ra khác 28](#_Toc61613519)

[2.4. Các thanh ghi cơ bản 30](#_Toc61613520)

[2.5. Công cụ phát triển lập trình nhúng 43](#_Toc61613521)

[2.5.1. Trình biên dịch 43](#_Toc61613522)

[2.5.2. Mạch nạp code vào Chip 44](#_Toc61613523)

[2.5.3. Phần mềm mô phỏng 44](#_Toc61613524)

[2.6. Cấu hình tối thiểu để 8051 hoạt động được 46](#_Toc61613525)

[2.7. Cấu trúc chương trình lập trình cho 8051 47](#_Toc61613526)

[2.8. Kiểu dữ liệu và toán tử dùng trong C cho 8051 48](#_Toc61613527)

[2.9. Các câu lệnh rẽ nhánh và kiểm tra thường dùng 50](#_Toc61613528)

[2.10. Các phương pháp lập trình nhúng với họ 8051 53](#_Toc61613529)

[2.11. Quy tắc lập trình 56](#_Toc61613530)

[CHƯƠNG 3: LẬP TRÌNH ĐIỀU KHIỂN NHÚNG 60](#_Toc61613531)

[3.1. Lập trình điều khiển LED đơn 60](#_Toc61613532)

[3.2. Lập trình điều khiển LED 7 thanh 66](#_Toc61613533)

[3.3. Lập trình điều khiển LCD 73](#_Toc61613534)

[3.3.1. Cấu tạo của LCD 73](#_Toc61613535)

[3.3.2. Lập trình điều khiển LCD 77](#_Toc61613537)

[3.4. Lập trình xử lý ngắt 83](#_Toc61613538)

[3.4.1. Ngắt của bộ định thời 83](#_Toc61613539)

[3.4.2. Lập trình với các ngắt ngoài của vi điều khiển 86](#_Toc61613540)

[3.5. Lập trình Timer 92](#_Toc61613541)

[3.5.1. Cơ chế tạo trễ của timer và cách tính toán giá trị nạp vào timer 92](#_Toc61613542)

[3.6. Lập trình truyền thông nối tiếp UART 95](#_Toc61613543)

[CHƯƠNG 4: THIẾT KẾ HỆ THỐNG NHÚNG 100](#_Toc61613544)

[4.1. Thiết kế hệ thống hiện thị 8 ký tự dùng vi điều khiển AT89C51 100](#_Toc61613545)

[4.1.1. Yêu cầu thiết kế 100](#_Toc61613546)

[4.1.2. Thiết kế sơ đồ nguyên lý 100](#_Toc61613547)

[4.1.3. Xây dựng chương trình điều khiển 101](#_Toc61613548)

[4.2. Thiết kế hệ thống giao tiếp bàn phím 4x4 dùng AT89C51 102](#_Toc61613549)

[4.2.1. Yêu cầu hệ thống 102](#_Toc61613550)

[4.2.2. Thiết kế sơ đồ nguyên lý 102](#_Toc61613551)

[4.2.3. Xây dựng chương trình điều khiển 103](#_Toc61613552)

[4.2.4. Đánh giá kết quả 107](#_Toc61613553)

[4.3. Thiết kế hệ thống hiện thị trên LED ma trận 8x8 107](#_Toc61613554)

[4.3.1. Yêu cầu 107](#_Toc61613555)

[2.6.2. Thiết kế sơ đồ mạch nguyên lý 107](#_Toc61613556)

[4.3.2. Xây dựng chương trình điều khiển 108](#_Toc61613557)

[4.3.3. Đánh giá kết quả 110](#_Toc61613558)

[4.4. Thiết kế đồng hồ thời gian thực hiện thị thời gian 111](#_Toc61613559)

[4.4.1. Yêu cầu thiết kế 111](#_Toc61613560)

[4.4.2. Thiết kế sơ đồ mạch nguyên lý 111](#_Toc61613561)

[4.4.3. Xây dựng chương trình điều khiển 112](#_Toc61613562)

[4.4.4. Đánh giá kết quả 117](#_Toc61613563)

[4.5. Thiết kế hệ thống điều khiển và xác định tốc độ động cơ 118](#_Toc61613564)

[4.5.1. Yêu cầu hệ thống 118](#_Toc61613565)

[4.5.2. Thiết kế sơ đồ mạch nguyên lý 118](#_Toc61613566)

[5.5.3. Xây dựng chương trình điều khiển 119](#_Toc61613567)

[5.5.4. Đánh giá kết quả 124](#_Toc61613568)

# **CHƯƠNG 1: TỔNG QUAN VỀ HỆ THỐNG NHÚNG**

## **1.1. Khái niệm hệ thống nhúng**

Hệ thống nhúng (Embedded system) là một thuật ngữ để chỉ một hệ thống có khả năng tự trị được nhúng vào trong một môi trường hay một hệ thống mẹ. Đó là các hệ thống tích hợp cả phần cứng và phần phềm để thực hiện một hoặc một nhóm chức năng chuyên biệt cụ thể.

Hệ thống nhúng (HTN) thường được thiết kế để thực hiện một chức năng chuyên biệt nào đó. Khác với các máy tính đa chức năng, chẳng hạn như máy tính cá nhân, một hệ thống nhúng chỉ thực hiện một hoặc một vài chức năng nhất định, thường đi kèm với những yêu cầu cụ thể và bao gồm một số thiết bị máy móc và phần cứng chuyên dụng mà ta không tìm thấy trong một máy tính đa năng nói chung. Vì hệ thống chỉ được xâydựng cho một số nhiệm vụ nhất định nên các nhà thiết kế có thể tối ưu hóa nó nhằm giảm thiểu kích thước và chi phí sản xuất. Các hệ thống nhúng thường được sản xuất hàng loạt với số lượng lớn. HTN rất đa dạng, phong phú về chủng loại. Đó có thể là những thiết bị cầm tay nhỏ gọn như đồng hồ kĩ thuật số và máy chơi nhạc MP3, hoặc những sản phẩm lớn như đèn giao thông, bộ kiểm soát trong nhà máy hoặc hệ thống kiểm soát các máy năng lượng hạt nhân.

## **1.2. Lịch sử phát triển**

Hệ thống nhúng đầu tiên là Apollo Guidance Computer (Máy tính dẫn đường Apollo) được phát triển bởi Charles Stark Draper tại phòng thí nghiệm của trường đại học MITnăm 1960. Hệ thống nhúng được sản xuất hàng loạt đầu tiên là máy hướng dẫn cho tên lửa quân sự vào năm 1961. Nó là máy hướng dẫn Autonetics D-17, được xây dựng sử dụng những bóng bán dẫn và một đĩa cứng để duy trì bộ nhớ. Khi Minuteman II được đưa vào sản xuất năm 1996, AutoneticsD-17 đã được thay thế với một máy tính mới sử dụng mạch tích hợp. Tính năng thiết kế chủ yếu của máy tính Minuteman II là nó đưa ra thuật toán có thể lập trình lại sau đó để làm cho tên lửa chính xác hơn, và máy tính có thể kiểm tra tên lửa, giảm trọng lượng của cáp điện và đầu nối điện.

Từ những ứng dụng đầu tiên vào những năm 1960, các hệ thống nhúng phát triển mạnh mẽ về khả năng xử lý. Bộ vi xử lý đầu tiên hướng đến người tiêu dùng là Intel 4004, được phát minh phục vụ máy tính điện tử và những hệ thống nhỏ khác. Tuy nhiên nó vẫn cần các chip nhớ ngoài và những hỗ trợ khác. Vào những năm cuối 1970, những bộ xử lý 8 bit đã được sản xuất, nhưng nhìn chung chúng vẫn cần đến những chip nhớ bên ngoài.

Vào giữa thập niên 80, kỹ thuật mạch tích hợp đã đạt trình độ cao dẫn đến nhiều thành phần có thể đưa vào một chip xử lý. Các bộ vi xử lý được gọi là các vi điều khiển và được chấp nhận rộng rãi. Với giá cả thấp, các vi điều khiển đã trở nên rất hấp dẫn để xây dựng các hệ thống chuyên dụng. Đã có một sự bùng nổ về số lượng các hệ thống nhúng trong tất cả các lĩnh vực thị trường và số các nhà đầu tư sản xuất theo hướng này. Ví dụ, rất nhiều chip xử lý đặc biệt xuất hiện với nhiều giao diện lập trình hơn là kiểu song song truyền thống để kết nối các vi xử lý. Vào cuối những năm 80, các hệ thống nhúng đã trở nên phổ biến trong hầu hết các thiết bị điện tử và khuynh hướng này vẫn còn tiếp tục cho đến nay.

## **1.3. Xu hướng phát triển trong tương lai**

Sau máy tính lớn (mainframe), PC và Internet thì hệ thống nhúng đang là làn sóng đổi mới thứ 3 trong công nghệ thông tin và truyền thông.

Xu hướng phát triển của các hệ thống nhúng hiện nay là:

* Phần mềm ngày càng chiếm tỷ trọng cao và đã trở thành một thành phần cấu tạo nên thiết bị bình đẳng như các phần cơ khí, linh kiện điện tử, linh kiện quang học…
* Các hệ nhúng ngày càng phức tạp hơn đáp ứng các yêu cầu khắt khe về thời gian thực, tiêu ít năng lượng và hoạt động tin cậy ổn định hơn.
* Các hệ nhúng ngày càng có độ mềm dẻo cao đáp ứng các yêu cầu nhanh chóng đưa sản phẩm ra thương trường, có khả năng bảo trì từ xa, có tính cá nhân cao.
* Các hệ nhúng ngày càng có khả năng hội thoại cao, có khả năng kết nối mạng và hội thoại với người sử dụng.
* Các hệ nhúng ngày càng có tính thích nghi, tự tổ chức cao có khả năng tái cấu hình như một thực thể, một tác nhân.
* Các hệ nhúng ngày càng có khả năng tiếp nhận năng lượng từ nhiều nguồn khác nhau (ánh sáng, rung động, điện từ trường, sinh học….) để tạo nên các hệ thống tự tiếp nhận năng lượng trong quá trình hoạt động.

## **1.4. Đặc điểm của hệ thống nhúng**

Hệ thống nhúng thường có một số đặc điểm chung như sau:

- Các hệ thống nhúng được thiết kế để thực hiện một số nhiệm vụ chuyên dụng  
chứ không phải đóng vai trò là các hệ thống máy tính đa chức năng. Một số hệ  
thống đòi hỏi ràng buộc về tính hoạt động thời gian thực để đảm bảo độ an toàn và tính ứng dụng. Một số hệ thống không đòi hỏi hoặc ràng buộc chặt chẽ, cho phép đơn giản hóa hệ thống phần cứng để giảm thiểu chi phí sản xuất.

- Một hệ thống nhúng thường không phải là một khối riêng biệt mà là một hệ thống phức tạp nằm trong thiết bị mà nó điều khiển.

- Phần mềm được viết cho các hệ thống nhúng được gọi là firmware và được lưu trữ trong các chip bộ nhớ chỉ đọc (ROM - Read Only Memory) hoặc bộ nhớ flash chứ không phải là trong một ổ đĩa. Phần mềm thường chạy với số tài nguyên phần cứng hạn chế: không có bàn phím, màn hình hoặc có nhưng với kích thước nhỏ, bộ nhớ hạn chế.

Sau đây, chúng ta sẽ đi sâu, xem xét cụ thể đặc điểm của các thành phần của hệ thống nhúng:

### **1.4.1. Giao diện**

Các hệ thống nhúng có thể không có giao diện (đối với những hệ thống đơn nhiệm) hoặc có đầy đủ giao diện giao tiếp với người dùng tương tự như các hệ điều hành trong các thiết bị để bàn. Đối với các hệ thống đơn giản, thiết bị nhúng sử dụng nút bấm, đèn LED và hiển thị chữ cỡ nhỏ hoặc chỉ hiển thị số, thường đi kèm với một hệ thống menu đơn giản.

Còn trong một hệ thống phức tạp hơn, một màn hình đồ họa, cảm ứng hoặc có các nút bấm ở lề màn hình cho phép thực hiện các thao tác phức tạp mà tối thiểu hóa được khoảng không gian cần sử dụng. Ý nghĩa của các nút bấm có thể thay đổi theo màn hình và các lựa chọn. Các hệ thống nhúng thường có một màn hình với một nút bấm dạng cần điểu khiển (joystick button). Sự phát triển mạnh mẽ của mạng toàn cầu đã mang đến cho những nhà thiết kế hệ nhúng một lựa chọn mới là sử dụng một giao diện web thông qua việc kết nối mạng. Điều này có thể giúp tránh được chi phí cho những màn hình phức tạp nhưng đồng thời vẫn cung cấp khả năng hiển thị và nhập liệu phức tạp khi cần đến, thông qua một máy tính khác. Điều này là hết sức hữu dụng đối với các thiết bị điều khiển từ xa, cài đặt vĩnh viễn. Ví dụ, các router là các thiết bị đã ứng dụng tiện ích này.

### **1.4.2. Kiến trúc CPU**

Các bộ xử lý trong hệ thống nhúng có thể được chia thành hai loại: Vi xử lý và vi điều khiển. Các vi điều khiển thường có các thiết bị ngoại vi được tích hợp trên chip nhằm giảm kích thước của hệ thống. Có rất nhiều loại kiến trúc CPU được sử dụng trong thiết kế hệ nhúng như ARM, MIPS, Coldfire/68k, PowerPC, x86, PIC, 8051, Atmel AVR…

Điều này trái ngược với các loại máy tính để bàn, thường bị hạn chế với một vài kiến trúc máy tính nhất định. Các hệ thống nhúng có kích thước nhỏ và được thiết kế để hoạt động trong môi trường công nghiệp thường lựa chọn PC/104 và PC/104++ làm nền tảng. Những hệ thống này thường sử dụng DOS, Linux hoặc các hệ điều hành nhúng thời gian thực như QNX hay VxWorks. Còn các hệ thống nhúng có kích thước rất lớn thường sử dụng một cấu hình thông dụng là hệ thống on chip (System on a chip – SoC), một bảng mạch tích hợp cho một ứng dụng cụ thể (An Application Specific Integrated Circuit – ASIC). Sau đó nhân CPU thêm vào như một phần của thiết kế chip. Một chiến lược tương tự là sử dụng FPGA (field-programmable gate array) và lập trình cho nó với những thành phần nguyên lý thiết kế bao gồm cả CPU.

### **1.4.3. Thiết bị ngoại vi**

Hệ thống nhúng giao tiếp với bên ngoài thông qua các thiết bị ngoại vi, ví dụ như:

* Serial Communication Interfaces (SCI): RS-232, RS-422, RS-485.
* Universal Serial Bus (USB).
* Networks: Controller Area Network, LonWorks.
* Bộ định thời: PLL(s), Capture/Compare và Time Processing Units.
* Discrete IO: General Purpose Input/Output (GPIO)

### **1.4.4. Công cụ phát triển**

Tương tự như các sản phẩm phần mềm khác, phần mềm hệ thống nhúng cũng được phát triển nhờ việc sử dụng các trình biên dịch (compilers), chương trình dịch hợp ngữ (assembler) hoặc các công cụ gỡ rối (debuggers). Tuy nhiên, các nhà thiết kế hệ thống nhúng có thể sử dụng một số công cụ chuyên dụng như:

* Bộ gỡ rối mạch hoặc các chương trình mô phỏng (emulator).
* Tiện ích để thêm các giá trị checksum hoặc CRC vào chương trình, giúp hệ thống nhúng có thể kiểm tra tính hợp lệ của chương trình đó.
* Đối với các hệ thống xử lý tín hiệu số, người phát triển hệ thống có thể sử dụng phần mềm workbench như MathCad hoặc Mathematica để mô phỏng các phép toán.
* Các trình biên dịch và trình liên kết (linker) chuyên dụng được sử dụng để tối ưu hóa một thiết bị phần cứng.
* Một hệ thống nhúng có thể có ngôn ngữ lập trình và công cụ thiết kế riêng của nó hoặc sử dụng và cải tiến từ một ngôn ngữ đã có sẵn.

Các công cụ phần mềm có thể được tạo ra bởi các công ty phần mềm chuyên dụng về hệ thống nhúng hoặc chuyển đổi từ các công cụ phát triển phần mềm GNU. Đôi khi, các công cụ phát triển dành cho máy tính cá nhân cũng được sử dụng nếu bộ xử lý của hệ thống nhúng đó gần giống với bộ xử lý của một máy PC thông dụng.

### **1.4.5. Độ tin cậy**

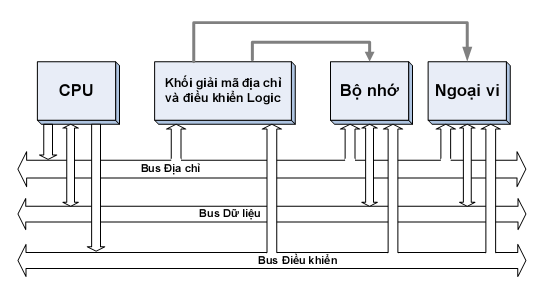
Các hệ thống nhúng thường nằm trong các cỗ máy được kỳ vọng là sẽ chạy hàng năm trời liên tục mà không bị lỗi hoặc có thể khôi phục hệ thống khi gặp lỗi. Vì thế, các phần mềm hệ thống nhúng được phát triển và kiểm thử một cách cẩn thận hơn là phần mềm cho máy tính cá nhân. Ngoài ra, các thiết bị rời không đáng tin cậy như ổ đĩa, công tắc hoặc nút bấm thường bị hạn chế sử dụng. Việc khôi phục hệ thống khi gặp lỗi có thể được thực hiện bằng cách sử dụng các kỹ thuật như watchdog timer – nếu phần mềm không đều đặn nhận được các tín hiệu watchdog định kì thì hệ thống sẽ bị khởi động lại.

Một số vấn đề cụ thể về độ tin cậy như:

* Hệ thống không thể ngừng để sửa chữa một cách an toàn, ví dụ như ở các hệ thống không gian, hệ thống dây cáp dưới đáy biển, các đèn hiệu dẫn đường… Giải pháp đưa ra là chuyển sang sử dụng các hệ thống con dự trữ hoặc các phần mềm cung cấp một phần chức năng.
* Hệ thống phải được chạy liên tục vì tính an toàn, ví dụ như các thiết bị dẫn đường máy bay, thiết bị kiểm soát độ an toàn trong các nhà máy hóa chất… Giải pháp đưa ra là lựa chọn backup hệ thống.
* Nếu hệ thống ngừng hoạt động sẽ gây tổn thất rất nhiều tiền của ví dụ như các dịch vụ buôn bán tự động, hệ thống chuyển tiền, hệ thống kiểm soát trong các nhà máy.

## **1.5. Các thành phần cơ bản trong kiến trúc phần cứng hệ thống nhúng**

Thông thường đi kèm với các hệ thống nhúng có rất nhiều những yêu cầu, thiết bị và bộ phận. Trong phần này chúng ta sẽ tìm hiểu một số bộ phận chính và một số yêu cầu chính của các hệ thống nhúng.

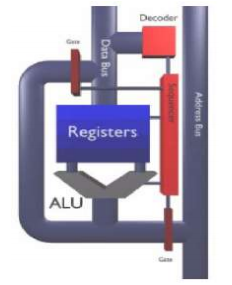


*Hình 1.1. Kiến trúc phần cứng nhúng*

Các thành phần phần cứng bao gồm:

* Đơn vị xử lý trung tâm CPU.
* Khối giải mã địa chỉ và điều khiển Logic
* Bus địa chỉ, dữ liệu và điều khiển.
* Bộ nhớ.
* Ngoại vi.
* Giao diện.

### **1.5.1. Đơn vị xử lý trung tâm CPU**



*Hình 1.2. Khối xử lý trung tâm*

Người ta vẫn biết tới phần lõi xử lý của các bộ vi xử lý (VXL) là đơn vị xử lý trung tâm CPU (Central Processing Unit) đóng vai trò như bộ não chịu trách nhiệm thực thi các phép tính và thực hiện các lệnh. Phần chính của CPU đảm nhận chức năng này là đơn vị logic toán học (ALU - Arthimetic Logic Unit). Ngoài ra để hỗ trợ hoạt động cho ALU còn thêm một số thành phần khác như bộ giải mã (decoder), bộ tuần tự (Sequencer) và các thanh ghi.

Bộ giải mã chuyển đổi (thông dịch) các lệnh lưu trữ ở trong bộ mã chương trình thành các mã mà ALU có thể hiểu được và thực thi. Bộ tuần tự có nhiệm vụ quản lý dòng dữ liệu trao đổi qua bus dữ liệu của VXL. Các thanh ghi được sử dụng để CPU lưu trữ tạm thời các dữ liệu chính cho việc thực thi các lệnh và chúng có thể thay đổi nội dung trong quá trình hoạt động của ALU. Hầu hết các thanh ghi của VXL đều là các bộ nhớ được tham chiếu (mapped) và hội nhập với khu vực bộ nhớ và có thể được sử dụng như bất kỳ khu vực nhớ khác.

Các thanh ghi có chức năng lưu trữ trạng thái của CPU. Nếu các nội dung của bộ nhớ VXL và các nội dung của các thanh ghi tại một thời điểm nào đó được lữu giữ đầy đủ thì hoàn toàn có thể tạm dừng thực hiện phần chương trình hiện tại trong một khoảng thời gian bất kỳ và có thể trở lại trạng thái của CPU trước đó. Thực tế số lượng các thanh ghi và tên gọi của chúng cũng khác nhau trong các họ VXL/VĐK và thường do chính các nhà chế tạo qui định, nhưng về cơ bản chúng đều có chung các chức năng như đã nêu. Khi thứ tự byte trong bộ nhớ đã được xác định thì người thiết kế phần cứng phải thực hiện một số quyết định xem CPU sẽ lưu dữ liệu đó như thế nào.

Một số thanh ghi với chức năng điển hình thường được sử dụng trong các kiến trúc CPU như sau:

Thanh ghi con trỏ ngăn xếp (stack pointer): Thanh ghi này lưu giữ địa chỉ tiếp theo của ngăn xếp. Theo nguyên lý giá trị địa chỉ chứa trong thanh ghi con trỏ ngăn xếp sẽ giảm nếu dữ liệu được lưu thêm vào ngăn xếp và sẽ tăng khi dữ liệu được lấy ra khỏi ngăn xếp.

Thanh ghi chỉ số (index register): Thanh ghi chỉ số được sử dụng để lưu địa chỉ khi mode địa chỉ được sử dụng. Nó còn được biết tới với tên gọi là thanh ghi con trỏ hay thanh ghi lựa chọn tệp (Microchip).

Thanh ghi địa chỉ lệnh/Bộ đếm chương trình (Program Counter): Một trong những thanh ghi quan trọng nhất của CPU là thanh ghi bộ đếm chương trình. Thanh ghi bộ đếm chương trình lưu địa chỉ lệnh tiếp theo của chương trình sẽ được CPU xử lý. Mỗi khi lệnh được trỏ tới và được CPU xử lý thì nội dung giá trị của thanh ghi bộ đếm chương trình sẽ tăng lên một. Chương trình sẽ kết thúc khi thanh ghi PC có giá trị bằng địa chỉ cuối cùng của chương trình nằm trong bộ nhớ chương trình.

Thanh ghi tích lũy (Accumulator): Thanh ghi tích lũy là một thanh ghi giao tiếp trực tiếp với ALU, được sử dụng để lưu giữ các toán tử hoặc kết quả của một phép toán trong quá trình hoạt động của ALU.

### **1.5.2. Bus địa chỉ, dữ liệu và điều khiển**

### **Bus địa chỉ**

Bus địa chỉ là các đường dẫn tín hiệu logic một chiều để truyền địa chỉ tham chiếu tới các khu vực bộ nhớ và chỉ ra dữ liệu được lưu giữ ở đâu trong không gian bộ nhớ. Trong qúa trình hoạt động CPU sẽ điều khiển bus địa chỉ để truyền dữ liệu giữa các khu vực bộ nhớ và CPU. Các địa chỉ thông thường tham chiếu tới các khu vực bộ nhớ hoặc các khu vực vào ra, hoặc ngoại vi. Dữ liệu được lưu ở các khu vực đó thường là 8bit (1 byte), 16bit, hoặc 32bit tùy thuộc vào cấu trúc từng loại vi xử lý/vi điều khiển. Hầu hết các vi điều khiển thường đánh địa chỉ dữ liệu theo khối 8bit. Các loại vi xử lý 8bit, 16bit và 32bit nói chung cũng đều có thể làm việc trao đổi với kiểu dữ liệu 8bit và 16bit.

Chúng ta vẫn thường được biết tới khái niệm địa chỉ truy nhập trực tiếp, đó là khả năng CPU có thể tham chiếu và truy nhập tới trong một chu kỳ bus. Nếu vi xử lý có N bit địa chỉ tức là nó có thể đánh địa chỉ được 2N khu vực mà CPU có thể tham chiếu trực tiếp tới. Qui ước các khu vực được đánh địa chỉ bắt đầu từ địa chỉ 0 và tăng dần đến 2N-1.

Hiện nay các vi xử lý và vi điều khiển nói chung chủ yếu vẫn sử dụng phổ biến các bus dữ liệu có độ rộng là 16, 20, 24, hoặc 32bit. Nếu đánh địa chỉ theo byte thì một vi xử lý 16bit có thể đánh địa chỉ được 2^16 khu vực bộ nhớ tức là 65,536 byte = 64Kbyte. Tuy nhiên có một số khu vực bộ nhớ mà CPU không thể truy nhập trực tiếp tới tức là phải sử dụng nhiều nhịp bus để truy nhập, thông thường phải kết hợp với việc điều khiển phần mềm. Kỹ thuật này chủ yếu được sử dụng để mở rộng bộ nhớ và thường được biết tới với khái niệm đánh địa chỉ trang nhớ khi nhu cầu đánh địa chỉ khu vực nhớ vượt quá phạm vi có thể đánh địa chỉ truy nhập trực tiếp.

Ví dụ: CPU 80286 có 24 bit địa chỉ sẽ cho phép đánh địa chỉ trực tiếp cho 2^24 byte nhớ. CPU 80386 và các loại vi xử lý mạnh hơn có không gian địa chỉ 32 bit sẽ có thể đánh được tới 2^32 byte địa chỉ trực tiếp.

### **Bus dữ liệu**

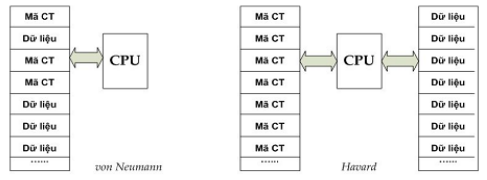
Bus dữ liệu là các kênh truyền tải thông tin theo hai chiều giữa CPU và bộ nhớ hoặc các thiết bị ngoại vi vào ra. Bus dữ liệu được điều khiển bởi CPU để đọc hoặc viết các dữ liệu hoặc mã lệnh thực thi trong quá trình hoạt động của CPU. Độ rộng của bus dữ liệu nói chung sẽ xác định được lượng dữ liệu có thể truyền và trao đổi trên bus. Tốc độ truyền hay trao đổi dữ liệu thường được tính theo đơn vị là [byte/s]. Số lượng đường bit dữ liệu sẽ cho phép xác định được số lượng bit có thể lưu trữ trong mỗi khu vực tham chiếu trực tiếp. Nếu một bus dữ liệu có khả năng thực hiện một lần truyền trong 1 μs, thì bus dữ liệu 8bit sẽ có băng thông là 1Mbyte/s, bus 16 bit sẽ có băng thông là 2Mbyte/s và bus 32bit sẽ có băng thông là 4Mbyte/s. Trong trường hợp bus dữ liệu 8bit với chu kỳ bus là T=1μs (tức là sẽ truyền được 1byte/1chu kỳ) thì sẽ truyền được 1 Mbyte trong 1s hay 2Mbyte trong 2s.

### **Bus điều khiển**

Bus điều khiển phục vụ truyền tải các thông tin dữ liệu để điều khiển hoạt động của hệ thống. Thông thường các dữ liệu điều khiển bao gồm các tín hiệu chu kỳ để đồng bộ các nhịp chuyển động và hoạt động của hệ thống. Bus điều khiển thường được điều khiển bởi CPU để đồng bộ hóa nhịp hoạt động và dữ liệu trao đổi trên các bus. Trong trường hợp vi xử lý sử dụng dồn kênh bus dữ liệu và bus địa chỉ tức là một phần hoặc toàn bộ bus dữ liệu sẽ được sử dụng chung chia sẻ với bus địa chỉ thì cần một tín hiệu điều khiển để phân nhịp truy nhập cho phép chốt lưu trữ thông tin địa chỉ mỗi khi bắt đầu một chu kỳ truyền.

### **1.5.3. Bộ nhớ**

Kiến trúc bộ nhớ được chia ra làm hai loại chính và được áp dụng rộng rãi trong hầu hết các Chip xử lý nhúng hiện nay là kiến trúc bộ nhớ von Neumann và Havard. Trong kiến trúc von Neumann không phân biệt vùng chứa dữ liệu và mã chương trình. Cả chương trình và dữ liệu đều được truy nhập theo cùng một đường. Điều này cho phép đưa dữ liệu vào vùng mã chương trình ROM, và cũng có thể lưu mã chương trình vào vùng dữ liệu RAM và thực hiện từ đó.

Kiến trúc Havard tách/phân biệt vùng lưu mã chương trình và dữ liệu. 

*Hình 1.3. Kiến trúc bộ nhớ von Neumann và Havard*

Kiến trúc Havard tách/phân biệt vùng lưu mã chương trình và dữ liệu. Mã chương trình chỉ có thể được lưu và thực hiện trong vùng chứa ROM và dữ liệu cũng chỉ có thể lưu và trao đổi trong vùng RAM. Hầu hết các vi xử lý nhúng ngày nay sử dụng kiến trúc bộ nhớ Havard hoặc kiến trúc Havard mở rộng (tức là bộ nhớ chương trình và dữ liệu tách biệt nhưng vẫn cho phép khả năng hạn chế để lấy dữ liệu ra từ vùng mã chương trình). Trong kiến trúc bộ nhớ Havard mở rộng thường sử dụng một số lượng nhỏ các con trỏ để lấy dữ liệu từ vùng mã chương trình theo cách nhúng vào trong các lệnh tức thời. Một số Chip vi điều khiển nhúng tiêu biểu hiện nay sử dụng cấu trúc Havard là 8031, PIC, Atmel AVR90S.

## **1.6. Một số nên phần cứng nhúng thông dụng**

### **1.6.1. Vi điều khiển họ 8051**

Thiết kế hệ điều khiển vi điều khiển với họ 8051 là việc thiết kế các ứng dụng trên chip AT89C51 đây là dòng chip thông dụng hiện nay với giá thành hợp lý phù hợp với nhu cầu phát triển các hệ thống không đòi hỏi độ phức tạp quá cao. Dựa trên dòng vi điều khiển này nhiều kỹ sư thiết kế đã phát triển được rất nhiều ứng dụng với độ phức tạp khá cao.

Họ vi điều khiển họ 8051 là IC đóng vỏ dạng DIP có 40 chân, mỗi chân có một ký hiệu tên và các chức năng như sau:

- Chân số 40 nối với nguồn nuôi +5V

- Chân số 20 nối với đất ( Mass, GND)

- Chân số 29 ( PSEN) (program store enable) là tín hiệu điều khiển xuất ra của 8051, có cho phép chọn bộ nhớ ngoài và được nối chung với chân OE (Outout Enable) của EPROM ngoài để cho phép đọc các byte của chương trình. Các xung tín hiệu PSEN hạ thấp trong suốt thời gian thi hành lệnh. Những mã nhị phân của chương trình được đọc từ EPROM đi qua bus dữ liệu và được chốt vào thanh ghi lệnh của 8051 bởi mã lệnh ( chú ý đọc ở đây là đọc các lệnh khác với đọc dữ liệu).

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

- Chân số 31 (EA: Eternal Acess) được đưa xuống thấp cho phép chọn bộ nhớ ngoài đối với 8051. Khi EA=5V chọn ROM nội, EA=0V chọn ROM ngoại.

Số chân còn lại của 8051 gồm 32 chân được chia làm 4 cổng vào ra

- Cổng P0 từ chân 39 32 tương ứng là chân P0\_0 P0\_7

- Cổng P1 từ chân 1 8 tương ứng là chân P1\_0 P1\_7

- Cổng P2 từ chân 21 28 tương ứng là chân P2\_0 P2\_7

- Cổng P3 từ chân 10 17 tương ứng là chân P3\_0 P3\_7

CPU( CPU centralprocessing unit) bao gồm:

- Thanh ghi tích lũy A

- Thanh ghi tích lũy phụ B

- Đơn vị logic học (ALU)

- Thanh ghi từ trạng thái chương trình

- Bốn băng thanh ghi

- Con trỏ ngăn xếp

- Bộ nhớ chương trình( ROM) gồm 8Kbyte Flash. Bộ nhớ dữ liệu( RAM) gồm 256 byte. Bộ UART, có chức năng truyền nhận nối tiếp ba bộ Timer/Counter 16 bit thực hiện chức năng định thời và đếm sự kiện. Khối điều khiển ngắt với 2 nguồn ngắt ngoài và 4 nguồn ngắt trong. Bộ lập trình ( ghi chương trình lên Flash ROM) cho phép người sử dụng có thể nạp các chương trình cho chíp mà không cần các bộ nạp chuyên dụng. Bộ chia tần số với hệ số chia là 12 và 4 cổng xuất nhập với 32 chân.

### **1.6.2. Vi điều khiển PIC**

Hiện nay trên thị trường có rất nhiều họ vi điều khiển như 8051, Motorola, AVR, ARM…Ngoài họ vi điều khiển 8051 được trình bày rất chi tiết ở phần trên trong môn học này chọn thêm họ vi điều khiển PIC để mở rộng vốn kiến thức và phát triển các ứng dụng trên họ vi điều khiển này vì các nguyên nhân sau:

- Họ vi điều khiển này có thể tìm mua dễ dàng tại Việt Nam

- Giá thành cho một con vi điều khiển không quá cao

- Là một sự bổ sung kiến thức rất tốt về kiến thức cũng như về ứng dụng cho họ vi điều khiển mang tính truyền thống là họ vi điều khiển 8051.

- Số lượng người sử dụng họ vi điều khiển PIC hiện nay tại Việt Nam cũng như trên toàn thế giới, họ vi điều khiển này được sử dụng khá rộng rãi. Điều này tạo thuận lợi trong quá trình tìm hiểu nghiên cứu và phát triển các ứng dụng.

- Sự hỗ trợ của nhà sản xuất về trình biên dịch, các công cụ lập trình, nạp chương trình đơn giản.

- Các tính năng đa dạng của vi điều khiển PIC, vá các tính năng không ngừng được phát triển.

Đây là vi điều khiển thuộc họ PIC16Fxxx với tập lệnh gồm 35 lệnh có độ dài 14 bit. Mỗi lệnh đều được thực thi trong một chu kỳ xung clock. Tốc độ hoạt động tối đa cho phép là 20 MHz với một chu kì lệnh 200ns. Bộ nhớ chương trình 8Kx14 bit, bộ nhớ dữ liệu 368x8 byte RAM và bộ nhớ dữ liệu EEPROM với dung lượng 256x8 byte. Số PORT I/O là 5 với 33 pin I/O.

Cac đặc tính ngoại vi bao gồm các chức năng sau:

- Time 0: bộ đếm 8 bit với bộ chia tần 8 bit

- Time 1: bộ đếm 16 bit với bộ chia tần số, có thể được thực hiện chức năng đếm dựa vào xung clock ngoại vi ngay khi vi điều khiển hoạt động ở chế độ sleep.

- Time 2: Bộ đếm 8 bit với bộ chia tần số, bộ postcaler

- Hai bộ Capture/so sánh/ điều chế độ rộng xung

- Chuẩn giao tiếp nối tiếp SSP ( Synchoronous Serial Port) , SPI và I2C.

- Chuẩn giao tiếp nối tiếp UART với 9 bit địa chỉ

- Có 8 kênh chuyển đổi ADC 10 bit, hai bộ so sánh

Bên cạnh đó là một vài đặc tính khác của vi điều khiển như:

- Bộ nhớ flash có khả năng ghi xóa 100.000 lần

- Bộ nhớ EEPROM với khả năng ghi xóa được 1.000.000 lần

- Dữ liệu EPROM có thể lưu trữ được trên 40 năm

### **1.6.3. Nền tảng phần cứng STM32**

STM32 là chíp vi điều khiển 32 bit lõi Arm Cortex với cấu hình mạnh mẽ cho dù với **phân khúc thấp nhất** STM32F0x cũng có thể hoạt động lên tới 48Mhz, 64kB Flash, 16kB RAM, 8 bộ Timer 16 bit, 1 bộ Timer 32 bit, 10 bộ ADC 12 bit, 8 bộ USART, 2 bộ SPI, 2 bộ I2C. Giá của dòng vi điều khiển này ở trên thị trường hiện nay rơi vào khoảng 30k đến 50k (STM32F030K6T6). Chíp này hoạt động lên đến 48MHz, có 32kB Flash, 4kB RAM, 5 bộ Timer 16 bit, 10 bộ ADC 12 bit, 1 bộ USART, 1 bộ I2C, 1 bộ SPI. Đây là mức giá rẻ nhất so với các dòng vi điều khiển có cấu hình tương đương.

### **1.6.4. Nền tảng phần cứng KL46**

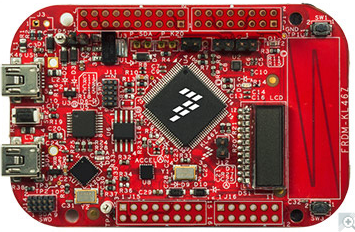
Kit FRDM-KL46Z là một kit ARM® Cortex™-M0+ giá rẻ do công ty Freescale sản xuất. Kit có kích thước khá nhỏ, nhỏ hơn bao thuốc lá. Nhưng trên kit tích hợp khá nhiều tính năng hay.

- chip ARM MKL46Z256VLL4MCU, core clock 48MHz, bộ nhớ flash 256kB, 32kB sram, tích hợp bộ điều khiển segment LCD, USB otg.  
- Cảm ứng điện dung (Capacitive touch slider), trên chip ARM đã tích hợp controller cho touch sensing.

- Cảm biến Accelerometer MMA8451Q

- Cảm biến từ trường magnetometer MAG3110

- Linh hoạt trong lựa chọn nguồn, có thể sử dụng 5V từ cổng USB máy tính, hoặc từ pin 3V3.



*Hình 1.4. Hình ảnh KIT KL46Z*

- Form factor compatible with Arduino ™ R3 pin layout

- Tích hợp bộ debug Open SDA trên kit, có thể sử dụng Open SDA để debug và giao tiếp RS232

- cảm biến ánh sáng

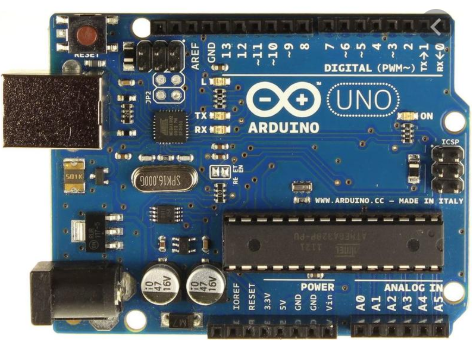
- I/O (2 leds, 2 buttons)

- LCD segment 4 digits Một số ý tưởng ứng dụng được trên kit mà ko cần phải thêm bất kỳ ngoại vi nào :

- ứng dụng la bàn số sử dụng cảm biến từ trường  
- tạo ra 1 usb device cho riêng mình sử dụng usb controller có sẵn trên chip, MKL46Z256VLL4MCU hỗ trợ USB low speed vào USB full speed.  
- thiết kế một máy nghe nhạc sử dụng PWM hoặc DAC  
- ngoài ra có thể sử dụng expand Header kết nối với các board mạch khác  
- có thể chạy một số hệ điều hành thời gian thực FreeRTOS, MQxlite.

### **1.6.5. Nền tảng phần cứng Arduino**

Arduino đã và đang được sử dụng rất rộng rãi trên thế giới, và ngày càng chứng tỏ được sức mạnh của chúng thông qua vô số ứng dụng độc đáo của người dùng trong cộng đồng nguồn mở (open-source). Tuy nhiên tại Việt Nam Arduino vẫn còn chưa được biết đến nhiều.



Hình 1.5: Hình ảnh kit vi điều khiểnt Arduino

Arduino cơ bản là một nền tảng tạo mẫu mở về điện tử (open-source electronics prototyping platform) được tạo thành từ phần cứng lẫn phần mềm. Về mặt kĩ thuật, có thể coi Arduino là một bộ điều khiển logic có thể lập trình được. Đơn giản hơn, Arduino là một thiết bị có thể tương tác với thế giới bên ngoài thông qua các cảm biến và hành vi được lập trình sẵn. Với thiết bị này, việc lắp ráp và điều khiển các thiết bị điện tử sẽ dễ dàng hơn bao giờ hết. Một điều không hề dễ dàng cho những ai đam mê công nghệ và điều khiển mà không có nhiều thời gian để tìm hiểu sâu hơn về kĩ thuật lập trình và cơ điện tử.

Hiện tại có rất nhiều loại vi điều khiển và đa số được lập trình bằng ngôn ngữ C/C++ hoặc Assembly nên rất khó khăn cho những người có ít kiến thức sâu về điện tử và lập trình. Nó là trở ngại cho mọi người muốn tạo riêng cho mình một món đồ mang tính công nghệ. Do vậy, đó là lí do Arduino được phát triển nhằm đơn giản hóa việc thiết kế, lắp ráp linh kiện điện tử cũng như lập trình trên vi xử lí và mọi người có thể tiếp cận dễ dàng hơn với thiết bị điện tử mà không cần nhiều về kiến thức điện tử và thời gian. Sau đây là những thế mạnh của Arduino so với các nền tảng vi điều khiển khác:

* Chạy trên đa nền tảng: Việc lập trình Arduino có thể thực hiện trên các hệ điều hành khác nhau như Windows, Mac Os, Linux trên Desktop, Android trên di động.

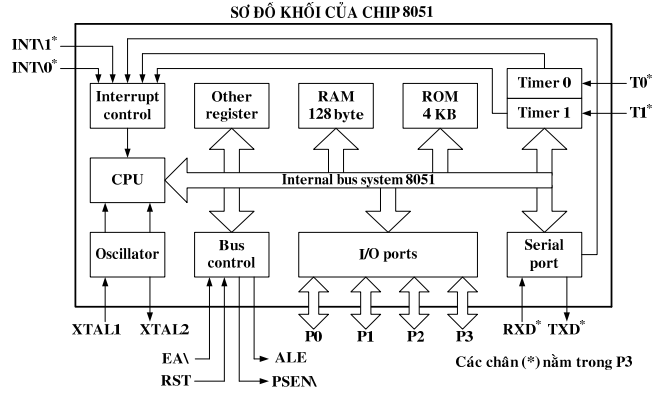
CHƯƠNG 2: HỌ VI ĐIỀU KHIỂN NHÚNG CƠ BẢN 8051

## **2.1. Tổng quan**

Vào năm 1981. Hãng Intel giới thiệu một số bộ vi điều khiển được gọi là 8051. Bộ vi điều khiển này có 128 byte RAM, 4K byte ROM trên chíp, hai bộ định thời, một cổng nối tiếp và 4 cổng (đều rộng 8 bit) vào ra tất cả được đặt trên một chíp. Lúc ấy nó được coi là một “hệ thống trên chíp”. 8051 là một bộ xử lý 8 bit có nghĩa là CPU chỉ có thể làm việc với 8 bit dữ liệu tại một thời điểm. Dữ liệu lớn hơn 8 bit được chia ra thành các dữ liệu 8 bit để cho xử lý. 8051 có tất cả 4 cổng vào - ra I/O mỗi cổng rộng 8 bit. Mặc dù 8051 có thể có một ROM trên chíp cực đại là 64 K byte, nhưng các nhà sản xuất lúc đó đã cho xuất xưởng chỉ với 4K byte ROM trên chíp.

8051 đã trở nên phổ biến sau khi Intel cho phép các nhà sản xuất khác sản xuất và bán bất kỳ dạng biến thế nào của 8051 mà họ thích với điều kiện họ phải để mã lại tương thích với 8051. Điều này dẫn đến sự ra đời nhiều phiên bản của 8051 với các tốc độ khác nhau và dung lượng ROM trên chíp khác nhau được bán bởi hơn nửa các nhà sản xuất.  Điều này quan trọng là mặc dù có nhiều biến thể khác nhau của 8051 về tốc độ và dung lương nhớ ROM trên chíp, nhưng tất cả chúng đều tương thích với 8051 ban đầu về các lệnh. Điều này có nghĩa là nếu ta viết chương trình của mình cho một phiên bản nào đó thì nó cũng sẽ chạy với mọi phiên bản bất kỳ khác mà không phân biệt nó từ hãng sản xuất nào.

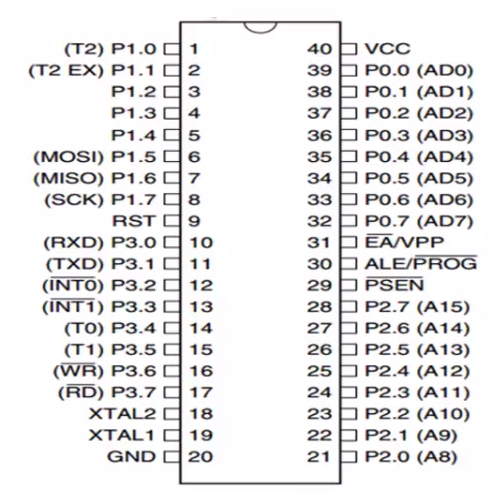
## **2.2. Sơ đồ khối và chức năng các khối của chip 8051**



*Hình 2.1. Sơ đồ khối chức năng chíp 8051*

- CPU (Central Processing Unit): đơn vị xử lý trung tâm → tính toán và điều khiển quá trình hoạt động của hệ thống.  
- OSC (Oscillator): Mạch dao động → tạo tín hiệu xung clock cung cấp cho các khối trong chip hoạt động.  
- Interrupt control: điều khiển ngắt → nhận tín hiệu ngắt từ bên ngoài (INT0\, INT1\), từ bộ định thời (Timer 0, Timer 1) và từ cổng nối tiếp (Serial port), lần lượt đưa các tín hiệu ngắt này ñến CPU ñể xử lý.  
- Other registers: Các thanh ghi khác → lưu trữ dữ liệu của các port xuất/nhập, trạng thái làm việc của các khối trong chip trong suốt quá trình hoạt động của hệ thống.  
- RAM (Random Access Memory): Bộ nhớ dữ liệu trong chip → lưu trữ các dữ liệu.  
- ROM (Read Only Memory): Bộ nhớ chương trình trong chip → lưu trữ chương trình hoạt động của chip.  
- I/O ports (In/Out ports): Các port xuất/nhập → điều khiển việc xuất nhập dữ liệu dưới dạng song song giữa trong và ngoài chip thông qua các port P0, P1, P2, P3.  
- Serial port: Port nối tiếp → điều khiển việc xuất nhập dữ liệu dưới dạng nối tiếp giữa trong và ngoài chip thông qua các chân TxD, RxD.  
- Timer 0, Timer 1: Bộ định thời 0, 1 → dùng để định thời gian hoặc đếm sự kiện (đếm xung) thông qua các chân T0, T1.  
- Bus control: điều khiển bus → điều khiển hoạt động của hệ thống bus và việc di chuyển thông tin trên hệ thống bus.  
- Bus system: Hệ thống bus → liên kết các khối trong chip lại với nhau

## **2.3. Sơ đồ chân và chức năng các chân của chip 8051**



*Hình 2.2. Sơ đồ chân chíp 8051*

### **2.3.1. Port 0**

- Port 0 (P0.0 – P0.7) có số chân từ 32 – 39.  
- Port 0 có hai chức năng:  
- Port xuất nhập dữ liệu (P0.0 - P0.7) không sử dụng bộ nhớ ngoài.  
- Bus địa chỉ byte thấp và bus dữ liệu đa hợp (AD0 – AD7) có sử dụng bộ nhớ  
ngoài.  
Lưu ý: Khi Port 0 đóng vai trò là port xuất nhập dữ liệu thì phải sử dụng các điện trở kéo lên bên ngoài.  
- Ở chế độ mặc định (khi reset) thì các chân Port 0 (P0.0 - P0.7) được cấu hình là port xuất dữ liệu. Muốn các chân Port 0 làm port nhập dữ liệu thì cần phải lập trình lại, bằng cách ghi mức logic cao (mức 1) đến tất cả các bit của port trước khi bắt đầu nhập dữ liệu từ port (vấn đề này ñược trình bày ở phần kế tiếp).

### **2.3.2. Port 1**

- Port 1 (P1.0 – P1.7) có số chân từ 1 – 8.

- Port 1 có một chức năng:

- Port xuất nhập dữ liệu (P1.0 – P1.7) sử dụng hoặc không sử dụng bộ nhớ  
ngoài.  
- Ở chế độ mặc định (khi reset) thì các chân Port 1 (P1.0 – P1.7) được cấu hình là port xuất dữ liệu. Muốn các chân Port 1 làm port nhập dữ liệu thì cần phải lập trình lại, bằng cách ghi mức logic cao (mức 1) đến tất cả các bit của port trước khi bắt ñầu nhập dữ liệu từ port (vấn đề này được trình bày ở phần kế tiếp).

### **2.3.3. Port 2**

- Port 2 (P2.0 – P2.7) có số chân từ 21 – 28.  
- Port 2 có hai chức năng:  
- Port xuất nhập dữ liệu (P2.0 – P2.7) không sử dụng bộ nhớ ngoài.  
- Bus địa chỉ byte cao (A8 – A15) có sử dụng bộ nhớ ngoài.  
- Ở chế độ mặc định (khi reset) thì các chân Port 2 (P2.0 – P2.7) được cấu hình là port xuất dữ liệu. Muốn các chân Port 2 làm port nhập dữ liệu thì cần phải lập trình lại, bằng cách ghi mức logic cao (mức 1) đến tất cả các bit của port trước khi bắt ñầu nhập dữ liệu từ port (vấn đề này được trình bày ở phần kế tiếp).

### **2.3.4. Port 3**

- Port 3 (P3.0 – P3.7) có số chân từ 10 – 17.  
- Port 3 có hai chức năng:  
- Port xuất nhập dữ liệu (P3.0 – P3.7) không sử dụng bộ nhớ ngoài hoặc các  
chức năng đặc biệt.  
- Các tín hiệu điều khiển có sử dụng bộ nhớ ngoài hoặc các chức năng đặc biệt.  
- Ở chế độ mặc định (khi reset) thì các chân Port 3 (P3.0 – P3.7) được cấu hình là port xuất  
dữ liệu. Muốn các chân Port 3 làm port nhập dữ liệu thì cần phải lập trình lại, bằng cách ghi mức logic cao (mức 1) đến tất cả các bit của port trước khi bắt đầu nhập dữ liệu từ port (vấn đề này được trình bày ở phần kế tiếp)

* **Chức năng của các chân Port 3**

|  |  |  |  |
| --- | --- | --- | --- |
| Bít | Tên | Địa chỉ | Chức năng |
| P3.0 | RxD | B0H | Chân nhận dữ liệu của port nối tiếp |
| P3.1 | TxD | B1H | Chân phát dữ liệu của port nối tiếp |
| P3.2 | INT0\ | B2H | Ngõ vào ngắt ngoài 0 |
| P3.3 | INT1\ | B3H | Ngõ vào ngắt ngoài 1 |
| P3.4 | T0 | B4H | Ngõ vào của bộ định thời/đếm 0 |
| P3.5 | T1 | B5H | Ngõ vào của bộ định thời/đếm 1 |
| P3.6 | WR\ | B6H | Điều khiển ghi vào RAM ngoài |
| P3.7 | RD\ | B7H | Điều khiển đọc từ RAM ngoài |

### **2.3.5. Các chân vào ra khác**

**Chân PSEN\:**  
- PSEN (Program Store Enable): cho phép bộ nhớ chương trình, chân số 29.  
- Chức năng:  
• Là tín hiệu cho phép truy xuất (đọc) bộ nhớ chương trình (ROM) ngoài.  
• Là tín hiệu xuất, tích cực mức thấp.  
PSEN\ = 0 → trong thời gian CPU tìm - nạp lệnh từ ROM ngoài.  
PSEN\ = 1 → CPU sử dụng ROM trong (không sử dụng ROM ngoài).  
- Khi sử dụng bộ nhớ chương trình bên ngoài, chân PSEN\ thường được nối với chân OE\ của ROM ngoài ñể cho phép CPU đọc mã lệnh từ ROM ngoài.  
**Chân ALE:**  
- ALE (Address Latch Enable): cho phép chốt địa chỉ, chân số 30.  
- Chức năng:  
• Là tín hiệu cho phép chốt địa chỉ để thực hiện việc giải đa hợp cho bus địa chỉ  
byte thấp và bus dữ liệu đa hợp (AD0 – AD7).  
• Là tín hiệu xuất, tích cực mức cao.  
ALE = 0 → trong thời gian bus AD0 - AD7 đóng vai trò là bus D0 - D7.  
ALE = 1 → trong thời gian bus AD0 - AD7 đóng vai trò là bus A0 - A7.  
- Khi lập trình cho ROM trong chip thì chân ALE đóng vai trò là ngõ vào của xung lập trình  
f(ALE) (MHz): tần số xung tại chân ALE.  
f(OSC) (MHz): tần số dao động trên chip (tần số thạch anh).  
- Khi lệnh lấy dữ liệu từ RAM ngoài được thực hiện thì một xung ALE bị bỏ qua  
**Chân EA\:**  
- EA (External Access): truy xuất ngoài, chân số 31.  
- Chức năng:  
• Là tín hiệu cho phép truy xuất (sử dụng) bộ nhớ chương trình (ROM) ngoài

Là tín hiệu nhập, tích cực mức thấp.  
EA\ = 0 Chip 8051 sử dụng chương trình của ROM ngoài.  
EA\ = 1 Chip 8051 sử dụng chương trình của ROM trong.

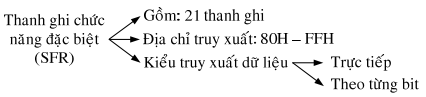
Lưu ý: Chân EA\ phải được nối lên Vcc (nếu sử dụng chương trình của ROM trong) hoặc nối xuống GND (nếu sử dụng chương trình của ROM ngoài), không bao giờ được phép bỏ trống chân này

**Chân XTAL1, XTAL2:**  
- XTAL (Crystal): tinh thể thạch anh, chân số 18-19.  
- Chức năng:  
Dùng ñể nối với thạch anh hoặc mạch dao động tạo xung clock bên ngoài, cung  
cấp tín hiệu xung clock cho chip hoạt động.  
- XTAL1 ngõ vào mạch tạo xung clock trong chip.  
- XTAL2 ngõ ra mạch tạo xung clock trong chip.

Chân RST:  
- RST (Reset): thiết lập lại, chân số 9.  
 Chức năng:  
- Là tín hiệu cho phép thiết lặp (đặt) lại trạng thái ban đầu cho hệ thống. Là tín hiệu nhập, tích cực mức cao.  
RST = 0 → Chip 8051 hoạt động bình thường.  
RST = 1 → Chip 8051 được thiết lặp lại trạng thái ban đầu

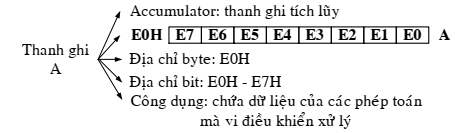
## **2.4. Các thanh ghi cơ bản**

* **Thanh ghi chức năng đặc biệt (SFR):**

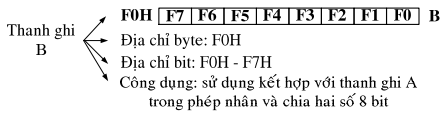


Không được phép đọc hay ghi dữ liệu vào các địa chỉ SFR mà nó chưa được đăng ký (nghĩa là các địa chỉ SFR chưa được đặt tên). Vì việc đọc hay ghi dữ liệu vào các nơi này có thể làm phát sinh những hoạt động không mong muốn và đó có thể là nguyên nhân làm cho chương trình của ta không tương thích với các phiên bản sau của chip MCS-51 (có thể ở các phiên bản đó các địa chỉ SFR này được sử dụng cho một vài mục đích khác).

* **Thanh ghi A / Thanh ghi tích lũy ACC**



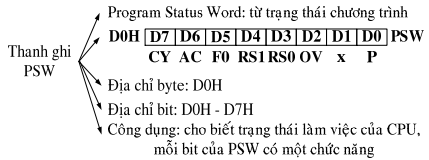
* **Thanh ghi B**



- Phép nhân 2 số 8 bit không dấukết quả là số 16 bit.  
Byte cao chứa vào thanh ghi B.  
Byte thấp chứa vào thanh ghi A.  
- Phép chia 2 số 8 bit thương số và số dư là số 8 bit.

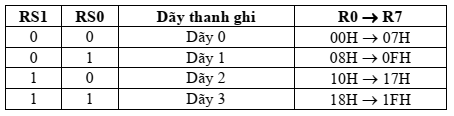
Thương số chứa vào thanh ghi A.  
Số dư chứa vào thanh ghi B

* **Thanh ghi từ PSW**



- Cờ CY (Carry Flag): cờ nhớ báo có nhớ/mượn tại bit 7.  
CY = 0: nếu không có nhớ từ bit 7 hoặc không có mượn cho bit 7.  
CY = 1: nếu có nhớ từ bit 7 hoặc có mượn cho bit 7.  
- Cờ AC (Auxiliary Carry): cờ nhớ phụ báo có nhớ/mượn tại bit 3.  
• AC = 0: nếu không có nhớ từ bit 3 hoặc không có mượn cho bit 3.  
• AC = 1: nếu có nhớ từ bit 3 hoặc có mượn cho bit 3.  
- Cờ F0 (Flag 0): cờ zero → có nhiều mục đích dành cho các ứng dụng khác nhau của người lập trình (dự trữ cho các phiên bản chip trong tương lai)

- Bit RS0, RS1 (Register Select): bit chọn dãy thanh ghicho phép xác định dãy thanh ghi tích cực (hay dãy thanh ghi mà các thanh ghi có tên là R0-R7)

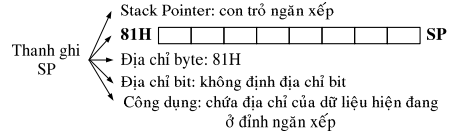


- Cờ OV (Overflow): cờ tràn báo kết quả tính toán của phép toán số học (phép toán có dấu) có nằm trong khoảng từ -128 ñến +127 hay không.  
OV = 0: nếu -128 <kết quả <+127.  
OV = 1: nếu kết quả < -128 hoặc kết quả > +127. Nói cách khác là: đối với phép cộng thì OV=1 nếu có nhớ từ bit 7 nhưng không có nhớ từ bit 6 hoặc nếu có nhớ từ bit 6 nhưng không có nhớ từ bit 7. đối với phép trừ thì OV=1 nếu có mượn cho bit 7 nhưng không có mượn cho bit 6 hoặc nếu có mượn bit 6 nhưng không có mượn bit 7.

- Cờ P (Parity): cờ chẵn lẻ → báo số chữ số 1 trong thanh ghi A là số chẵn hay số lẻ (trong  
chip 8051 sử dụng chế ñộ parity chẵn).  
P = 0: nếu số chữ số 1 trong thanh ghi A là số chẵn (parity chẵn).

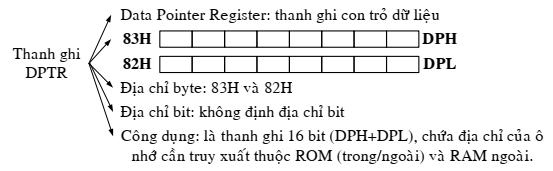
P = 1: nếu số chữ số 1 trong thanh ghi A là số lẻ (parity chẵn).

* **Thanh ghi SP**

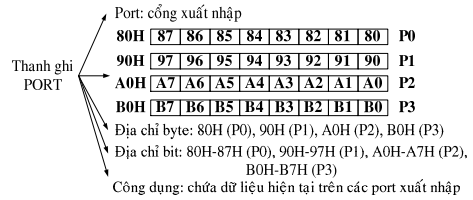


- Ngăn xếp là vùng nhớ dùng ñể lưu trữ tạm thời các dữ liệu.  
- đối với chip 8051 thì vùng nhớ được dùng để làm ngăn xếp được giữ trong RAM nội.  
- để sử dụng ngăn xếp thì ta phải khởi động thanh ghi SP (nghĩa là nạp giá trị cho thanh ghi SP) vùng nhớ của ngăn xếp có địa chỉ bắt đầu: (SP)+1 và địa chỉ kết thúc: 7FH.  
- Nếu không khởi động SP vùng nhớ của ngăn xếp có địa chỉ bắt đầu: 08H và địa chỉ kết thúc: 7FH (chế độ mặc định)

* **Thanh ghi DPTR**

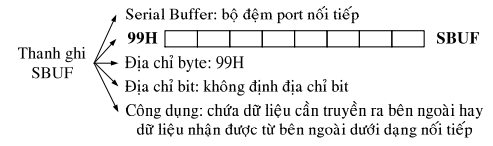


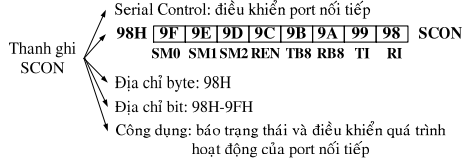
* **Thanh ghi port xuất nhập**



Trong trường hợp phần cứng có sử dụng ROM hoặc RAM bên ngoài thì ta không thể sử dụng Port 0 và Port 2 để xuất nhập dữ liệu. Vì khi đó chip 8051 sẽ sử dụng hai port này để xác định địa chỉ và dữ liệu cho bộ nhớ ngoài. Khi đó, ta chỉ có thể sử dụng Port 1 và Port 3 để xuất nhập dữ liệu. Ở chế độ mặc định (khi reset) thì tất cả các chân của các port (P0 – P3) được cấu hình là port xuất dữ liệu. Muốn các chân port của chip 8015 làm port nhập dữ liệu thì ta cần phải được lập trình lại, bằng cách ghi mức logic cao (mức 1) đến tất cả các bit (các chân) của port trước khi bắt đầu nhập dữ liệu từ port.

* **Thanh ghi port nối tiếp**



Đếm dữ liệu nối tiếp gồm 2 thanh ghi riêng biệt, một thanh ghi đệm phát và một thanh ghi đệm thu. Khi dữ liệu được chuyển tới SBUF, nó sẽ đi vào bộ đệm phát và được giữ ở đấy để chế biến thành dạng truyền tin nối tiếp. Khi dữ liệu được truyền đi từ SBUF, nó sẽ đi ra từ bộ đệm thu.  




SCON là thanh ghi trạng thái và điều khiển cổng nối tiếp. Nó không chứa các bít chọn chế độ, mà còn chứa bít dữ liệu thứ 9 dành cho việc truyền và nhận tin (TB8 và RB8) và chứa bít ngắt cổng nối tiếp.

- SM0, SM1: Là các bít cho phép chọn chế độ cổng nối tiếp.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SM0 | SM1 | Mode | Đặc điểm | Tốc độ Baud |
| 0 | 0 | 0 | Thanh ghi dịch | F/12 |
| 0 | 1 | 1 | 8 bít UART | Có thể thay đổi ( được đặt bởi bộ Timer) |
| 1 | 0 | 2 | 9 bít UART | F/64 hoặc F/32 |
| 1 | 1 | 3 | 9 bít UART | Có thể thay đổi ( được đặt bởi bộ Timer) |

- SM2: Cho phép truyền tin đa xử lý, thể hiện ở Mode 2 và 3, ở chế độ 2 hoặc 3, nếu đặt SM2=1 thì RI sẽ không được kích hoạt nếu bít dữ liệu thứ 9 (RB8) nhận được giá trị bằng 0. Ở Mode 1 nếu SM2=1 thì RI sẽ không được kích hoạt nếu bít dừng có hiệu lực đã không được nhận. ở chế độ 0, SM2 nên bằng 0.

- REN: Cho phép nhận nối tiếp. Được đặt hoặc xóa bởi phần mềm để cho phép hoặc không cho phép nhận.

- TB8: Là bít dữ liệu thứ 9 mà sẽ được truyền ở Mode 2 và 3. Được đặt hoặc xóa bởi phần mềm

- RB 8: Là bít dữ liệu thứ 9 mà sẽ được nhận ở Mode 2 và 3. Ở Mode 1, nếu SM2=0 thì RB8 là bít dừng đã được nhận ở Mode 0, RB 8 không được sử dụng.

- TI: Cờ Ngắt truyền được đặt bởi phần cứng tại cuối thời điểm của bít thứ 8 trong Mode 0 hoặc đầu thời điểm của bít dừng trong các Mode khác. Ở bất kỳ quá trình nào nối tiếp cũng phải được xóa bằng phần mềm.

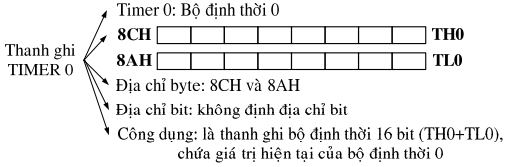
- RI: Cờ Ngắt nhận. Được đặt bởi phần cứng tại cuối thời điểm của bít thứ 8 trong Mode 0 hoặc giữa thời điểm của bít dừng trong Mode khác. Ở bất kỳ quá trình nhận nối tiếp nào nó cũng phải được xóa bằng phần mềm.

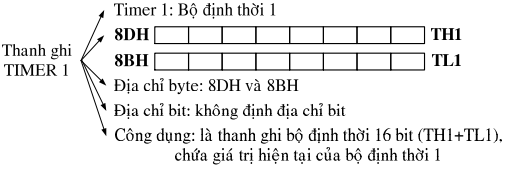
* **Thanh ghi định thời / Timer/Counter**

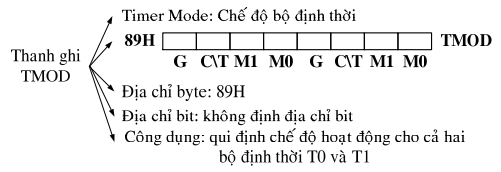
Trong họ 8051 có 2 thanh ghi Timer/Counter dài 16 bít đó là Timer 0 và Timer 1. Một số dòng vi điều khiển khác như AT89C52 còn có thêm Timer 2. Cả 3 bộ Timer này đều có thể được điều khiển để thực hiện chức năng định thời gian hay bộ đếm thông qua thanh ghi TMOD.

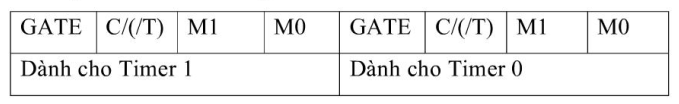
- Khi thanh ghi Timer/Counter làm việc ở kiểu Timer, thì sau mỗi chu kỳ máy nội dung trong thanh ghi tăng thêm một đơn vị. Vì vậy thanh ghi này đếm số chu kỳ máy. Một chu kỳ máy có 12 chu kỳ dao động, do đó tốc độ đếm của thanh ghi là 1/12 tần số dao động.

- Khi thanh ghi Timer/Counter làm việc ở kiểu Counter xung nhịp bên ngoài được đưa vào để đếm ở T0 hoặc T1. Nội dung của thanh ghi được tăng lên khi có sự chuyển trạng thái từ 1 về 0 tại chân đầu vào ngoài T0 hoặc T1.









- GATE: Khi GATE=1 và TRx=1, bộ TIMER/COUNTERx hoạt động chỉ khi chân INTx ở mức cao. Khi GATE=0 bộ TIMER/COUNTERx sẽ hoạt động chỉ khi TRx=1.

- C/(/T): Bít này cho phép chọn chức năng là Timer hay Counter. Bít này bằng 0 thì thực hiện chức năng Timer. Bít này bằng 1 thì thực hiện chức năng Counter

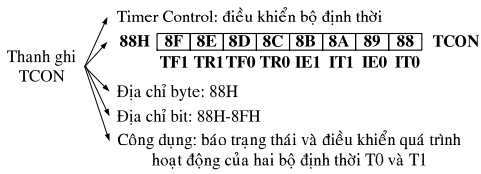
- M0, M1: Bít chọn Mode để xác định trạng thái và kiểu Timer/Counter:

M1=0, M0=0: Chọn kiểu bộ Timer 13 bít. Trong đó THx dài 8 bít, TLx dài 5 bít.

M1=0, M0=1: Chọn kiểu bộ Timer 16 bít. THx và TLx dài 16 bít được ghép tầng.

M1=1,M0=0: Các thanh ghi tự động nạp lại mỗi khi tràn. Khi bộ Timer bị tràn, THx dài 8 bít được giữ nguyên giá trị, còn giá trị nạp lại được đưa vào TLx.

M1=1, M0=1: Kiểu phân chia bộ Timer TL0 là 1 bộ Timer/Counter 8 bit được điều khiển bằng các bit điều khiển Timer 1.



- Thanh ghi TCON là thanh ghi điều khiển bộ Timer/Counter



- TF1: Cờ tràn Timer 1 được đặt bởi phần cứng khi bộ Timer 1 tràn. Được xóa bởi phần cứng khi bộ vi xử lý hướng tới chương trình con phục vụ ngắt.

- TR1: Bít điều khiển bộ Timer 1 hoạt động. Được đặt/xóa bởi phần mềm để điều khiển bộ Timer 1 ON/OFF

- TF0: Cờ tràn Timer 0 được đặt/xóa bởi phần mềm để điều khiển bộ Timer 0 ON/OFF

- IE1: Cờ ngắt ngoài 1 được đặt bởi phần cứng khi sườn xung của ngắt ngoài 1 được phát hiện. Được xóa bởi phần cứng khi ngắt được xử lý.

- IT1: Bít điều khiển ngắt 1 để tạo ra ngắt ngoài. Được đặt/xóa bởi phần mềm

- IE0: cờ ngắt ngoài 0 được đặt bởi phần cứng khi sườn xung của ngắt ngoài 0 được phát hiện. Được xóa bởi phần cứng khi ngắt được xử lý.

- IT0: Bít điều khiển ngắt 0 để tạo ra ngắt ngoài. Được đặt/xóa bởi phần mềm

- Trong AT89C51 và AT89C52 đều có bộ Timer 0 và Timer 1. Chức năng Timer hay Counter được chọn lựa bởi bít điều khiển C/T trong thanh ghi TMOD. Hai bộ Timer/Counter này có 4 chế độ hoạt động được lựa chọn bởi cặp bít ( M0, M1) trong thanh ghi TMOD. Chế độ 0,1 và 2 giống nhau cho chức năng Timer/Counter nhưng chế độ 3 thì khác. Bốn chế độ hoạt động được mô tả như sau:

**+ Chế độ 0:** Cả 2 bộ Timer 0 và 1 ở chế độ 0 có cấu hình như trong hình nhừ một thanh ghi 13 bít, bao gồm 8 bít của thanh ghi THx và 5 bít thấp của thanh ghi TLx. 3 bít cáo của thanh ghi TLx không xác định chắc chắn nên được làm ngơ. Khi thanh ghi được xóa về 0 , thì cờ ngắt thời gian TFx được thiết lập. Bộ Timer/Counter hoạt động khi bít điều khiển TRx được thiết lập (TRx=1) và hoặc Gate trong TMOD bằng 0 hoặc /INTx=1. Nếu đặt Gate=1 thì cho phép điều khiển Timer/Counter bằng đường vào ngoài /INTx để dễ dàng xác định độ rộng xung.

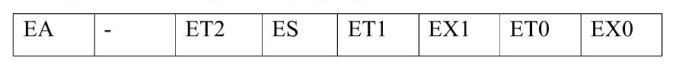
- Khi hoạt động ở chức năng thời gian thì bít C/(/T)= 0, do vậy xung nhịp từ bộ dao động nội qua bộ chia tần cho ra tần số f=f/12 để đưa vào để đếm trong thanh ghi Timer/Counter. Khi hoạt động ở chức năng bộ đếm thì bít C/(/T)= 1 khi đó xung nhịp ngoài đưa vào sẽ được đếm.

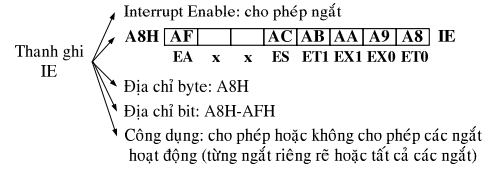
**+ Chế độ 1:** Hoạt động tương tự như chế độ 0 chỉ khác là thanh ghi Timer/Counter được sử dụng cả 16 bít. Xung nhịp được dùng kết hợp với thanh ghi thời gian byte thấp và byte cao ( TH1 và TL1). Khi xung clock được nhận, bộ Timer sẽ đếm tăng lên 0000h, 0001h, 0002h. Khi hiện tượng tràn xẩy ra cờ tràn sẽ chuyển FFFFh về 0000h và bộ Timer tiếp tục đếm. Cờ tràn của Timer1 là bít TF1 ở thanh ghi TCON nó được đọc hoặc ghi bởi phần mềm.

**+ Chế độ 2:** Chế độ này của thanh ghi Timer cũng hoạt động tương tự như 2 trế độ trên nhưng nó được tổ chức như bộ đếm 8 bít ( TL1) với chế độ tự động nạp lại khi xẩy ra hiện tượng tràn ở TL1, không chỉ thiết lập bít TF1 mà còn tự động nạp lại cho TL1 bằng nội dung của TH1 đã được thiết lập bởi phần mềm. Quá trình nạp lại cho phép nội dung của TH1 không bị thay đổi. Chế độ 2 của Timer/Counter 0 cũng tương tự như Time/Counter 1

**+ Chế độ 3:** Ở chế độ này chức năng Timer/Counter 0 và chức năng Timer/Counter 1 khắc nhau. Bộ Timer 1 ở chế độ 3 chỉ chứa chức năng đếm của nó, kết quả giống khi đặt TR1=0. Bộ timer 0 ở chế độ 3 thiết lập TH0,TL0 như là 2 bộ đếm riêng biệt. Bộ đếm TL0 được điều khiển bởi các bít: C/(/T), GATE, TR0, /INT0 và khi đếm tràn nó thiết lập cờ ngắt TF0. Bộ đếm TH0 chỉ được điều khiển bởi bít TR1 và khi đếm tràn nó thiết lập cờ ngắt TF1. Vậy TH0 điều khiển ngắt Timer/Counter 1.

**Thanh ghi ngắt IE**





- EA: Nếu EA=0 không cho phép ngắt nào hoạt động. Nếu EA=1 mỗi nguồn ngắt riêng biệt được phép hoặc không được phép hoạt động bằng cách đặt hoặc xóa bít Anable của nó.

- ET2: Bít cho phép hoặc không cho phép ngắt bộ Timer 2

- ES: Bít cho phép hoặc không cho phép ngắt cổng nối tiếp ( SPI và UART)

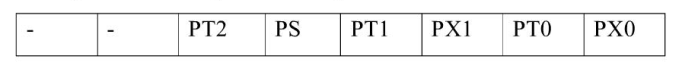
- ET1: Bít cho phép hoặc không cho phép ngắt tràn bộ Timer 1

- EX1: Bít cho phép hoặc không cho phép ngắt ngoài 1

- ET0: Bít cho phép hoặc không cho phép ngắt tràn bộ Timer 0

- EX0: Bít cho phép hoặc không cho phép ngắt ngoài 0

* **Thanh ghi ưu tiên ngắt IP**



- PT2: Xác định mức ưu tiên của Timer 2

- PS: Định nghĩa mức ưu tiên của ngắt cổng nối tiếp

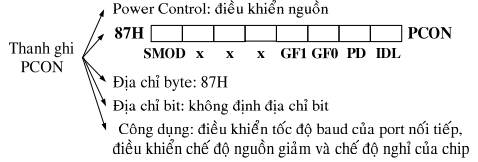
- PT1: Định nghĩa mức ưu tiên ngắt của Timer 1

- PX1: Định nghĩa mức ưu tiên của ngắt ngoài 1

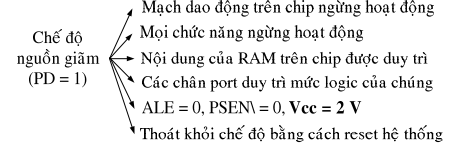
- PT0: Định nghĩa mức ưu tiên của ngắt Timer 0

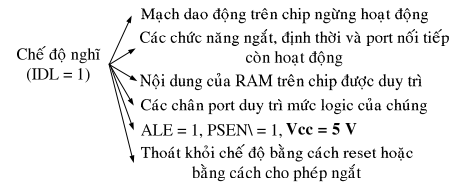
- PX0: Định nghĩa mức ưu tiên của ngắt ngoài 0

* **Thanh ghi điều khiển nguồn**



- Bit SMOD (Serial Mode)cho phép tăng gấp đôi tốc độ truyền dữ liệu nối tiếp (tốc độ baud) khi SMOD = 1.  
- Bit GF1, GF0 (General Function) cho phép người lập trình dùng với mục đích riêng (dự trữ cho các phiên bản chip trong tương lai).  
- Bit PD (Power Down) dùng để qui định chế độ nguồn giãm.  
- Bit IDL (Idle)dùng để qui định chế độ nghĩ.

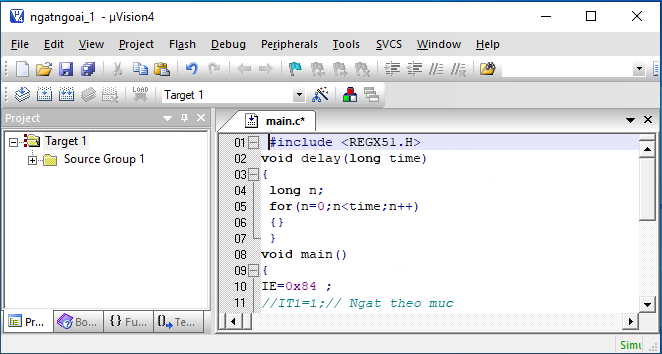




## **2.5. Công cụ phát triển lập trình nhúng**

### **2.5.1. Trình biên dịch**

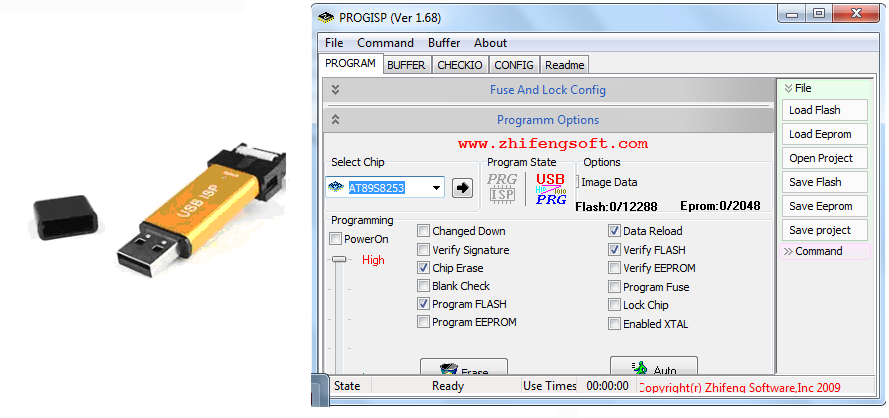
Trình biên dịch là phần mềm mà trên đó sẽ được viết các chương trình điều khiển để nạp xuống cho Vi điều khiển. Ta không thể viết chương trình cho vi điều khiển trên word, excel ... mà mỗi một loại vi điều khiển khác nhau lại phải dùng một phần mềm chuyên dụng riêng để viết chương trình cho nó. Đối với dòng vi điều khiển 8051 thì cũng có một vài trình biên dịch khác nhau, nhưng cơ bản nhất, phổ biến nhất là trình biên dịch Keil C. Từ phần mềm này ta có thể viết chương trình bằng cả 2 loại ngôn ngữ là C hoặc ASM (Trong suốt các phần tiếp theo, tôi sẽ hướng dẫn các viết chương trình dùng ngôn ngữ C trên trình biên dịch keil C để nạp chương trình cho vi điều khiển). Vi điều khiển không thể hiểu được các ngôn ngữ mà chúng ta viết chương trình, nó chỉ hiểu được các mã máy (mà do nhà sản xuất tạo ra), phần mềm này có chức năng "phiên dịch" các dòng lệnh mà ta viết bằng C hoặc ASM sang ngôn ngữ mã máy (tạo ra một file .hex) từ đó ta sẽ dùng mạch nạp để nạp file .hex này cho vi điều khiển.



*Hình 2.3. Giao diện trình biên dịch Keil C*

### **2.5.2. Mạch nạp code vào Chip**

Mạch nạp USB ISP, giá rẻ, nạp nhanh, nạp được cho cả họ AVR, không cần cài driver. Phần mềm nạp Progisp 1.68 chạy được trên nền win 7, win 10.

****

*Hinh 2.4. Hình ảnh mạch nạp và phần mềm nạp chíp 8051*

### **2.5.3. Phần mềm mô phỏng**

**Proteus** là phần mềm cho phép mô phỏng hoạt động của mạch điện tử bao gồm phần thiết kế mạch và viết chương trình điều khiển cho các họ vi điều khiển như MCS-51, PIC, AVR, …

Proteus là phần mềm mô phỏng mạch điện tử của Lancenter Electronics, mô phỏng cho hầu hết các linh kiện điện tử thông dụng, đặc biệt hỗ trợ cho cả các MCU như PIC, 8051, AVR, Motorola.

Phần mềm bao gồm 2 chương trình: ISIS cho phép mô phỏng mạch và ARES dùng để vẽ mạch in. Proteus là công cụ mô phỏng cho các loại Vi Điều Khiển khá tốt, nó hỗ trợ các dòng VĐK PIC, 8051, PIC, dsPIC, AVR, HC11, MSP430, ARM7/LPC2000 ... các giao tiếp I2C, SPI, CAN, USB, Ethenet,... ngoài ra còn mô phỏng các mạch số, mạch tương tự một cách hiệu quả. Proteus là bộ công cụ chuyên về mô phỏng mạch điện tử.

 ISIS đã được nghiên cứu và phát triển trong hơn 12 năm và có hơn 12000 người dùng trên khắp thế giới. Sức mạnh của nó là có thể mô phỏng hoạt động của các hệ vi điều khiển mà không cần thêm phần mềm phụ trợ nào. Sau đó, phần mềm ISIS có thể xuất file sang ARES hoặc các phần mềm vẽ mạch in khác.

 Trong lĩnh vực giáo dục, ISIS có ưu điểm là hình ảnh mạch điện đẹp, cho phép ta tùy chọn đường nét, màu sắc mạch điện, cũng như thiết kế theo các mạch mẫu (templates)

Những khả năng khác của ISIS là:

• Tự động sắp xếp đường mạch và vẽ điểm giao đường mạch.

• Chọn đối tượng và thiết lập thông số cho đối tượng dễ dàng

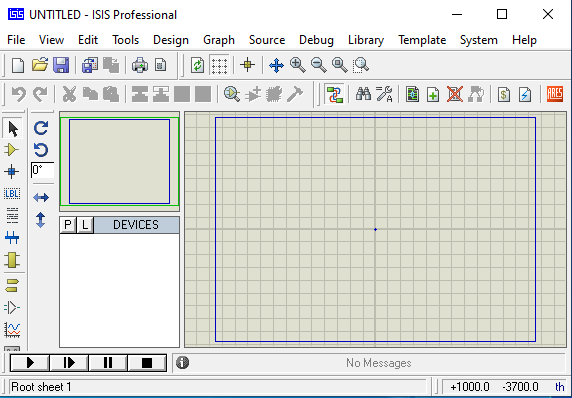
• Xuất file thống kê linh kiện cho mạch

• Xuất ra file Netlist tương thích với các chương trình làm mạch in thông dụng.

• Đối với người thiết kế mạch chuyên nghiệp, ISIS tích hợp nhiều công cụ giúp cho việc quản lý mạch điện lớn, mạch điện có thể lên đến hàng ngàn linh kiện.

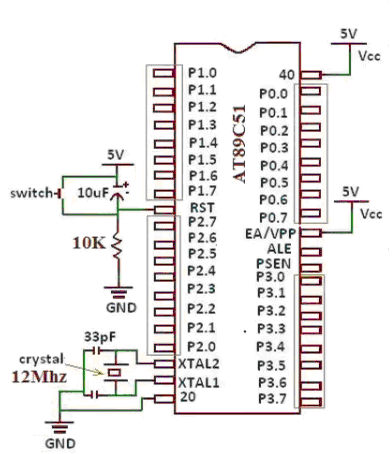
• Thiết kế theo cấu trúc (hierachical design)

• Khả năng tự động đánh số linh kiện



*Hình 2. 5. Giao diện phần mềm mô phỏng Proteus*

## **2.6. Cấu hình tối thiểu để 8051 hoạt động được**



*Hình 2.6. Cấu hình tối thiểu để vi điều khiển họ 8051 hoạt động*

- Chân 20: cấp GND

- Chân 40: cấp nguồn VCC 5V

- Chân 9: Nối với mạch reset

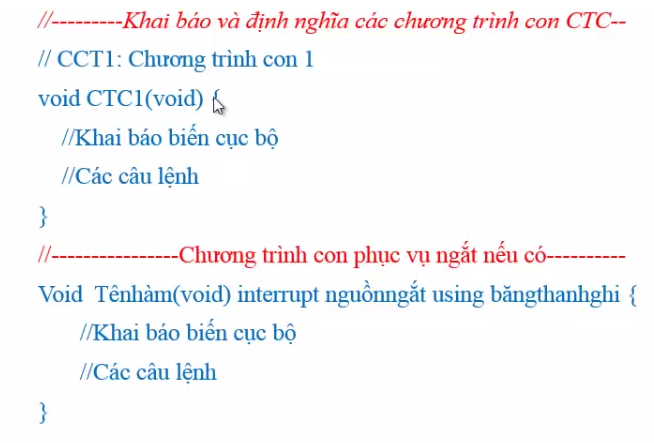
- Chân 18,19: Nối với mạch tạo dao động

- Chân số 31: Nối với VCC để chạy chương trình lưu trong bộ nhớ trong

- Nếu cổng P0 muốn làm cổng vào thì cần nối với trở treo 10K kéo lên VCC

## **2.7. Cấu trúc chương trình lập trình cho 8051**

****

****

## **2.8. Kiểu dữ liệu và toán tử dùng trong C cho 8051**

* **Các loại biến dùng trong C cho 8051**

|  |  |  |  |
| --- | --- | --- | --- |
| **Dạng biến** | **Số bit** | **Số byte** | **Miền giá trị** |
| bit | 1 |  | 0,1 |
| char | 8 | 1 | -128 +127 |
| unsigned char | 8 | 1 | 0 255 |
| int | 16 | 2 | -32,768 +32,767 |
| Unsigned int | 16 | 2 | 0 65,535 |
| log | 32 | 4 | -2,147,483,648+2,147,483,648 |
| Unsigned long | 32 | 4 | 0 4,294,697,295 |

* **Các phép toán số học**

|  |  |
| --- | --- |
| **Tên phép toán** | **Ký hiệu** |
| Phép gán | = |
| Phép cộng | + |
| Phép trừ | - |
| Phép nhân | \* |
| Phép chia lấy phần nguyên | / |
| Phép chia lấy dư | % |
| Phép so sánh bằng | = = |

* **Các phép toán logic**

|  |  |
| --- | --- |
| **Chức năng** | **Phép toán** |
| Phép AND | && |
| Phép OR | || |
| Phép NOT | ! |

* **Các phép toán so sánh**

|  |  |  |
| --- | --- | --- |
| **Phép toán** | **ý nghĩa** | **Ví dụ** |
| **>** | So sánh lớn hơn | a>b  4>5 sẽ trả ra giá trị 0 |
| **>=** | So sánh lớn hơn hoặc bằng | a>=b  6>=2 sẽ trả ra giá trị 1 |
| **<** | So sánh nhỏ hơn | a<b  6<7 sẽ trả ra giá trị 1 |
| **<=** | So sánh nhỏ hơn hoặc bằng | a<=b  8<=5 sẽ trả ra giá trị 0 |

* **Các phép toán thao tác bit**

|  |  |  |
| --- | --- | --- |
| **Phép toán** | **Ý nghĩa** | **Ví dụ** |
| **&** | Phép và (AND) | Bit\_1 & Bit\_2 |
| **|** | Phép hoặc (OR) | Bit\_1 | Bit\_2 |
| **!** | Phép đảo (NOT) | !Bit\_1 |
| **^** | Phép hoặc loại trừ (XOR) | Bit\_1 ^ Bit\_2 |
| **<<** | Dịch trái | a<<3 |
| **>>** | Dịch phải | a>>4 |
| **~** | Lấy bù theo bit | ~a |

## **2.9. Các câu lệnh rẽ nhánh và kiểm tra thường dùng**

* Câu lệnh **if**

|  |  |
| --- | --- |
| **Cấu trúc lệnh** | **Ví dụ** |
| *if ( điều kiện)*  *{*  *// Các\_câu\_lệnh\_xử\_lý*  *}* | *if (giay==60)*  *{*  *phut=phut+1;*  *giay=0;*  *}* |

* Câu lệnh **if/else**

|  |  |
| --- | --- |
| **Cấu trúc lệnh** | **Ví dụ** |
| *if ( điều kiện)*  *{*  *// khôi lệnh 1*  *}*  *Else*  *{*  *// khối lệnh 2*  *}* | *if (nut==bam)*  *{*  *P2\_0=0;*  *}*  *else*  *{*  *P2\_0=1;*  *}* |

* Câu lệnh rẽ nhánh **Switch/case**

|  |  |
| --- | --- |
| **Cấu trúc lệnh** | **Ví dụ** |
| Switch (Biến )  {  case giá\_trị :  // Khối\_lệnh\_1  break;  case giá\_trị\_2 :  // khối\_lệnh\_2;  break;  ...  case default :  // Khối\_lệnh\_default  break;  } | *Switch(count)*  *{*  *case 0:*  *P2=0;*  *break;*  *case 1:*  *P2=1;*  *break;*  *case default:*  *P2=2;*  *break;*  *}* |

* Câu lệnh vòng lặp xác định **for**

|  |  |
| --- | --- |
| **Cấu trúc lệnh** | **Ví dụ** |
| *For( i=0; i<n;i++)*  *{*  *// Các câu lệnh xử lý*  *}* | *For(i=0;i<1000;i++)*  *{*  *count++;*  *P2=P2+1;*  *}* |

* Câu lệnh vòng lặp xác định **while**

|  |  |
| --- | --- |
| **Cấu trúc lệnh** | **Ví dụ** |
| *while (Điều\_kiện)*  *{*  *// Các câu lệnh xử lý*  *}* | *while (i<1000)*  *{*  *count++;*  *P2=P2+1;*  *}* |

## **2.10. Các phương pháp lập trình nhúng với họ 8051**

Lập trình nhúng với họ vi điều khiển 8051 có nhiều phương pháp khai báo biến khác nhau sau đây là một số phương pháp khai báo có thể sử dụng khi xấy dựng chương trình nhúng với vi điều khiển họ 8051.

* **Kiểu lập trình điều khiển từng chân trên cổng vào ra**

#include <REGX51.H>

#define bat 1

#define tat 0

sbit LED1=P2^0;

sbit LED2=P2^1;

void delay(long t){

while(t--);

}

*void main()*

{

while(1)

{

LED1=tat; delay(5000);// tat tu trai sang phai

LED2=tat; delay(5000);

}}

* **Kiểu lập trình xuất trực tiếp giá trị vào cổng**

#include <REGX51.H>

void delay(long t){

while(t--);

}

void main()

{

while(1)

{

P2=0x01; delay(5000);// sang tu trai sang phai

P2=0x03; delay(5000);

P2=0x07; delay(5000);

P2=0x0f; delay(5000);

* **Kiểu lập trình sử dụng vòng lặp**

#include <REGX51.H>

int i;a;

void delay(long t){

while(t--);

}

*void sangthuan()*

{

a=0x01;

for(i=0; i<=3;i++)

{

P2=a;

delay(5000);

a=a<<1;

a=a|0x01;

}}

*void main()*

{

while(1)

{

sangthuan();

P2=0x00; delay(5000);

sangnghich();

}}

* **Kiểu lập trình tác động trức tiếp vào chân vi điều khiển**

#include <REGX51.H>

void delay(long t){

while(t--);

}

void main()

{

while(1)

{

P2\_0=1; delay(5000);// bat tu trai sang phai

P2\_1=1;; delay(5000);

P2\_2=1; delay(5000);

P2\_3=1; delay(5000);

}}

## **2.11. Quy tắc lập trình**

|  |  |
| --- | --- |
| **Quy tắc** | **Ví dụ** |
| Quy tắc 1: Chương trình nên được tách thành nhiều chương trình con.  - Mỗi chương trình con thực hiện một công việc và càng độ lập với nhau càng tốt.  - Điều này giúp cho chương trình dễ kiểm soát dễ sửa và dễ đọc |  |

|  |  |  |
| --- | --- | --- |
| **Quy tắc** | **Không nên viết** | **Hãy viết là** |
| **Quy tắc 2:** Hạn chế sử dụng biến toàn cục  - Nên sử dụng các tham số khi truyền thông tin cho các chương trình con.  - Tránh sử dụng các biến toàn cục để truyền thông tin giữa các chương trình con vì như vậy sẽ làm mất tính độc lập giữa các chương trình con và rất khó khăn khi kiểm soát giá trị của chúng khi chương trình con thi hành. | # include <REGX51.H>  Int i;  void delay() {  for (i=0;i<=300;i++)  }  void nhay Led(){  P2=0x00; delay();  P2=0xff; delay();  }  void main() {  for( i=0;i<5;i++)  nhay Led();  } | # include <REGX51.H>  void delay() {  int i;  for (i=0;i<=300;i++)  }  void nhay Led( int lan){  int i;  for(i=0;i<lan;i++){  P2=0x00; delay();  P2=0xff; delay();  }}  void main() {  nhay Led(5);  } |

|  |  |  |
| --- | --- | --- |
| **Quy tắc** | **Không nên viết** | **Hãy viết là** |
| **Quy tắc 3:** Mỗi câu lênh nên được đặt riêng trên một dòng.  - Chương trình dễ đọc  - Dễ quan sát, tìm lỗi | # include <REGX51.H>  Int i;  void delay() {  for (i=0;i<=300;i++)  }  void main() {  while(1) {  P2=0x00 ; delay();  P2=0xff ; delay();  }} | # include <REGX51.H>  Int i;  void delay() {  for (i=0;i<=300;i++)  }  void main() {  while(1) {  P2=0x00 ;  delay();  P2=0xff ;  delay();  }} |
| **Quy tắc** | Không nên viết | Hãy viết là |
| **Quy tắc 4:** Câu lệnh phải thể hiện đúng cấu trúc ngôn ngữ.  - Sử dụng thụt đầu dòng hợp lý.  - Các câu lênh nằm giữa cặp dấu {} được viết thụt vào một khoảng tab.  - Các lệnh ngang cấp thì phải thụt vào như nhau. | # include <REGX51.H>  Int i;  void delay() {  for (i=0;i<=300;i++)  }  void main() {  while(1) {  P2=0x00 ; delay();  P2=0xff ; delay();  }} | # include <REGX51.H>  int i;  void delay()  {  for (i=0;i<=300;i++)  }  void main() {  while(1) {  P2=0x00 ;  delay();  P2=0xff ;  delay();}} |

|  |  |  |
| --- | --- | --- |
| **Quy tắc** | **Không nên viết** | **Hãy viết là** |
| **Quy tắc 5:** Đặt tên  - Tên biến, tên hàm nên đặt sao cho gợi nhớ chức năng.  - Từ đầu tiên nên viết thường tên từ tiếp theo nên viết hoa cho dễ phân tách từng từ.  - Tên hằng số thì viết hoa | Int Count;  void HienThi(int SoHienThị) | Int count;  void hienThi(int soHienThi) |

CHƯƠNG 3: LẬP TRÌNH ĐIỀU KHIỂN NHÚNG

## **3.1. Lập trình điều khiển LED đơn**

Lập trình điều khiển LED đơn là bài toán cơ bản nhất đối với người học lập trình. Giúp sinh viên làm quen với phương pháp lập trình nhúng, cấu trúc một chương trình lập trình nhúng. Giúp sinh viên hiểu sâu hơn về điều khiển vào ra trên các cổng của vi điều khiển.

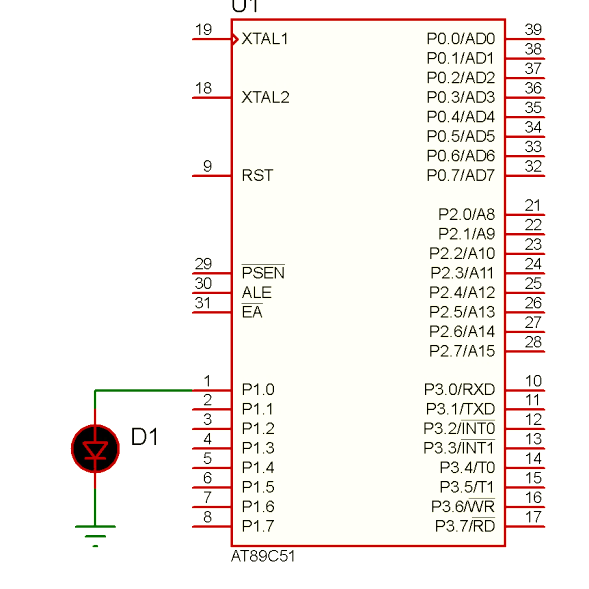
* **Cấu tạo LED đơn**

LED là cụm từ viết tắt của từ Tiếng Anh (Light Emitting Diode). Nó có nghĩa là các đi ốt phát quang. Và các đi ốt này được chứa trong con chip bán dẫn. Hoạt động của những con chip này sẽ được tác động khi có dòng điện chạy qua lấp đầy chỗ trống và sinh ra hiện tượng bức xạ ánh sáng.



*Hình 3.1. Hình ảnh LED đơn*

**Bài toán 1:** Ghép nối LED đơn với chân P1.0 của vi điều khiển, viết chương trình điều khiển LED nhấp nháy với thời gian trễ 25000



*Hình 3.2. Sơ đồ mạch nguyên lý*

+ Chương trình điều khiển :

/\*==================Bo tien xu li===================\*/

#include<AT89x51.h> // Dinh kem file thu vien

#define bat 1 // Dinh nghia gia tri bat den Led

#define tat 0 // Dinh nghia gia tri tat den Led

/\*==================khai bao bien==================\*/

sbit Led = P1^0; // Khai bao bien Led kieu bit chan P1.0

/\*================= Khai bao hàm==================\*/

/\*------------------------------ham tre -------------------------------------\*/

void delay(long time)

{

while(time--);

}

/\*--------------------------------ham chinh--------------------------------\*/

void main(void)

{

while(1)

{

Led = bat; // bat Led

delay(25000); // tre 1s

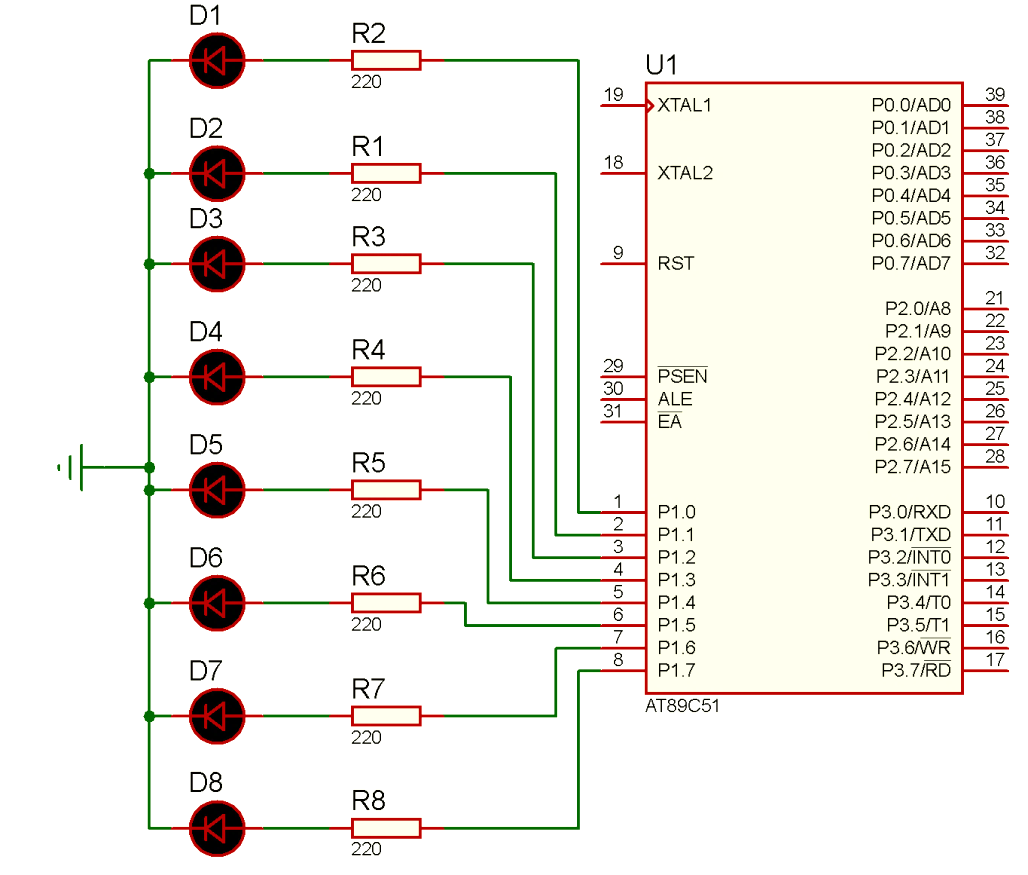
Led = tat; // tat Led

delay(25000); //tre 1s

}

}

**Bài toán 2:** Ghép nối 8 LED đơn với chân các chân của cổng P1 của vi điều khiển, viết chương trình điều khiển LED nhấp nháy với thời gian trễ là 25000.

****

*Hình 3.3. Sơ đồ mạch nguyên lý*

+ Chương trình điều khiển 1:

*#include <REGX51.H>*

*#define bat 1*

*#define tat 0*

*sbit led1=P1^0;*

*sbit led2=P1^1;*

*sbit led3=P1^2;*

*sbit led4=P1^3;*

*sbit led5=P1^4;*

*sbit led6=P1^5;*

*sbit led7=P1^6;*

*sbit led8=P1^7;*

*void delay()*

*{*

*long i;*

*for (i=0;i<25000;i++)*

*{}}*

*void main()*

*{*

*while(1)*

*{*

*led1=bat;delay();*

*led1=tat;delay();*

*led2=bat;delay();*

*led2=tat;delay();*

}}

+ Chương trình điều khiển 2:

*#include <REGX51.H>*

*int i,j,a,b;*

*void delay()*

*{*

*long i;*

*for (i=0;i<10000;i++)*

*{}}*

*// Sang trai tung led*

*void sangtrai()*

*{*

*a=0x01;*

*for(i=0;i<=7;i++)*

*{*

*P1=a;delay();*

*a=a<<1;*

*}}*

*// Sang trai giu trang thai*

*void sangtraigiutrangthai()*

*{*

*a=0x01;*

*for(i=0;i<=7;i++)*

*{*

*P1=a;delay();*

*a=a<<3;*

*a=a|0x01;*

*}}*

*// Sang trai chay*

*void sangphai()*

*{*

*a=0x80;*

*for(i=0;i<8;i++)*

*{*

*P1=a; delay();*

*a=a>>1;*

*} }*

*void sangphaigiu()*

*{*

*a=0x80;*

*for(i=0;i<8;i++)*

*{*

*P1=a; delay();*

*a=a>>1;*

*a=a|0x80;*

*} }*

*void giuara()*

*{*

*a=0x10;*

*b=0x08;*

*for(i=0,j=0;i<4,j<4;i++,j++)*

*{*

*P1=a+b; delay();*

*a=a<<1;*

*b=b>>1;*

*}}*

*void giuaragiu()*

*{*

*a=0x10;*

*b=0x08;*

*for(i=0,j=0;i<4,j<4;i++,j++)*

*{*

*P1=a+b;delay();*

*a=a<<1;*

*b=b>>1;*

*a=a|0x10;*

*b=b|0x08 ;*

*}}*

*void haiben()*

*{*

*a=0x80;*

*b=0x01;*

*for(i=0,j=0;i<4,j<4;i++,j++)*

*{*

*P1=a+b;delay();*

*a=a>>1;*

*b=b<<1;*

*}*

*}*

*void haibengiu()*

*{*

*a=0x80;*

*b=0x01;*

*for (i=0,j=0;i<4,j<4;i++,j++)*

*{*

*P1=a+b; delay();*

*a=a>>1;*

*a=a|0x80;*

*b=b<<1;*

*b=b|0x01;*

*}*

*}*

*void main()*

*{*

*while(1)*

*{*

*sangtrai();*

*sangtraigiutrangthai();*

*sangphai() ;*

*sangphaigiu();*

*giuara();*

*giuaragiu();*

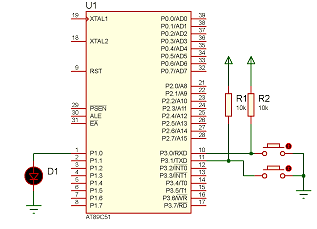
*haiben();*

*haibengiu();*

*}*

*}*

**Bài toán 3:** Một đèn Led được nối với chân P1.0 của vi điều khiển. Một công tắc START nối với chân P3.0 và công tắc STOP nối với chân P3.1 của vi điều khiển. hãy viết chương trình điều khiển để khi bật công tắc START thì Led sang, khi bật công tắc STOP thì Led tắt.



*Hình 3.3. Sơ đồ mạch nguyên lý*

+ Chương trình điều khiển:

\*/==================Bo tien xu li===================\*/

#include<AT89x51.h> // Dinh kem file thu vien

#define bat 1 // Dinh nghia gia tri bat den Led

#define tat 0 // Dinh nghia gia tri tat den Led

/\*==================khai bao bien==================\*/

sbit Led = P1^0; // Khai bao bien Led kieu bit chan P1.0

sbit STOP = P3^0; // cong tac STOP de tat Led

sbit START = P3^1; // cong tac START de bat Led

/\*=================== ham chinh==================\*/

void main(void)

{

Led = tat; //ban dau tat Led

while(1){

if((START==0)&&(STOP==1)) {

Led = bat;

}

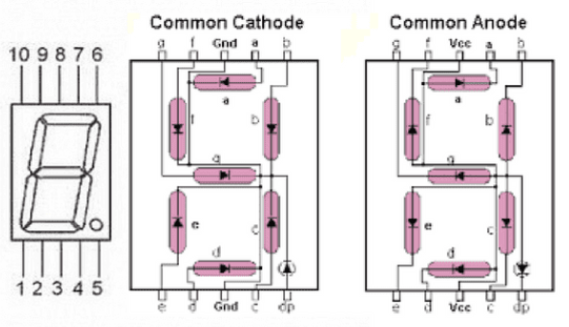
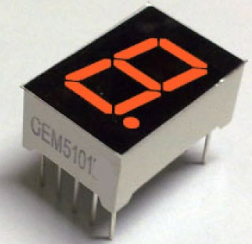
if((START==1)&&(STOP==0)){

Led = tat;

}}}

## **3.2. Lập trình điều khiển LED 7 thanh**

* **LED 7 thanh**



*Hình 3.4. Hình ảnh và cấu tạo LED 7 thanh*

* **Bảng trạng thái LED 7 thanh Anode chung**

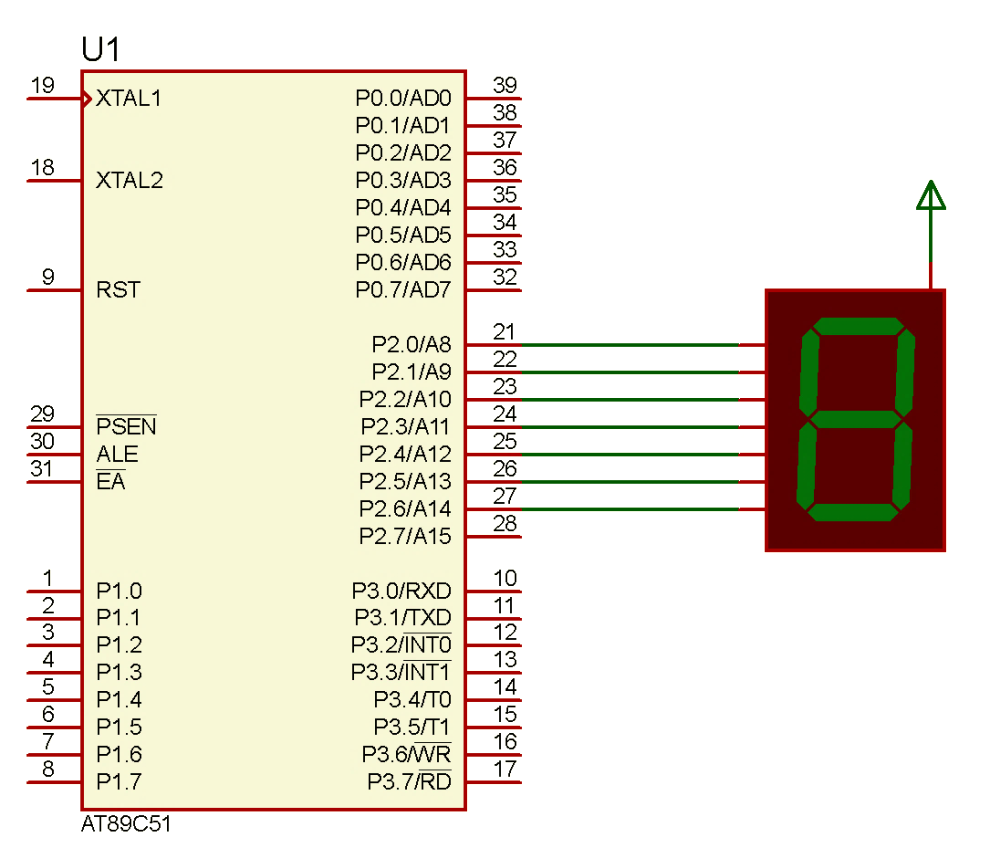
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| STT | Giá trị nhị phân | g | f | e | d | c | b | a | Hexa |
| 0 | 0100000 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 40 |
| 1 | 01111001 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 79 |
| 2 | 00100100 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 24 |
| 3 | 00110000 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 30 |
| 4 | 00011001 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 19 |
| 5 | 00010010 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 12 |
| 6 | 00000010 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 02 |
| 7 | 01111000 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 78 |
| 8 | 00000000 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 00 |
| 9 | 00010000 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 10 |

* **Bảng trạng thái LED 7 thanh Cathode chung**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| STT | Giá trị nhị phân | g | f | e | d | c | b | a | Hexa |
| 0 | 00111111 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 3F |
| 1 | 00000110 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 06 |
| 2 | 01011011 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 5B |
| 3 | 01001111 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 4F |
| 4 | 01100110 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 66 |
| 5 | 01101101 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 6D |
| 6 | 01111101 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |  |
| 7 | 00000111 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |  |
| 8 | 11111111 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |  |
| 9 | 01101111 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |  |

**Bài toán 1:** LED 7 thanh anode chung có a,b,c,d,e,f được nối với các chân trên cổng P2 của vi điều khiển.Điều khiển LED 7 thanh đếm từ 0 đến 9 sau đó đếm 9 về 0.

+ Sơ đồ mạch

****

*Hình 3.5. Sơ đồ mạch nguyên lý*

+ Chương trình điều khiển:

#include <REGX51.H>

long i,j, dem;

long so[10]={0x40,0x79,0x24,0x30,0x19,0x12,0x02,0x78,0x00,0x10} ;

void delay()

{

for(i=0;i<10000;i++)

{

}}

void main()

{

while(1)

{

// Dem tien

for (dem=0;dem<10;dem++)

{

P2=so[dem];delay();

}

// Dem lui

for (dem=9;dem>=0;dem--)

{

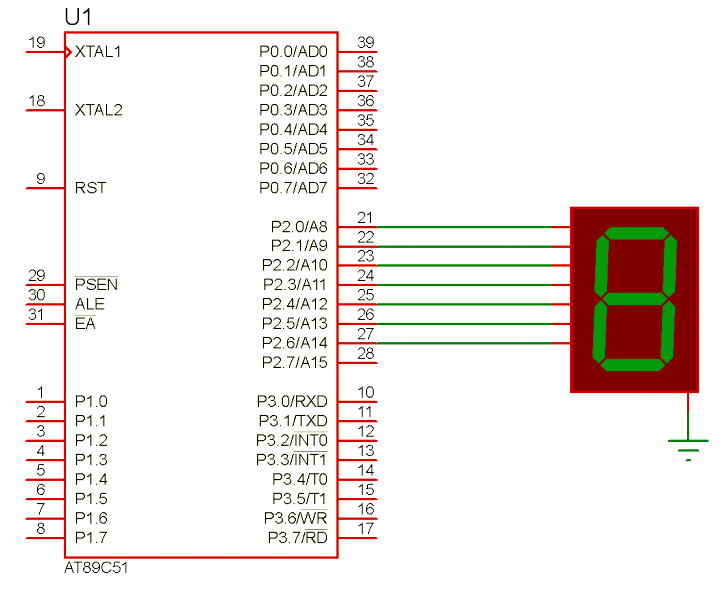
P2=so[dem];delay();

}

} }

**Bài toán 2:** LED 7 thanh Cathode chung có a,b,c,d,e,f được nối với các chân trên cổng P2 của vi điều khiển.Điều khiển LED 7 thanh đếm từ 0 đến 9 sau đó đếm 9 về 0.

+ Sơ đồ mạch

****

*Hình 3.6. Sơ đồ mạch nguyên lý*

+ Chương trình điều khiển:

#include <REGX51.H>

long so[]={0x3F,0x06,0x5B,0x4F,0x66,0x6D,0xFD,0x07,0xFF,0x6F};

long dem;

void delay()

{

long i;

for (i=0;i<10000;i++)

{;}

}

void main()

{

while(1)

{

// hien thi tién

for(dem=0;dem<10;dem++)

{

P2=so[dem];delay();

}

// Hien thi lui

for(dem=9;dem>=0;dem--)

{

P2=so[dem];delay();

}

// dem so chan

for(dem=0;dem<10;dem++)

{

if(dem%2==0)

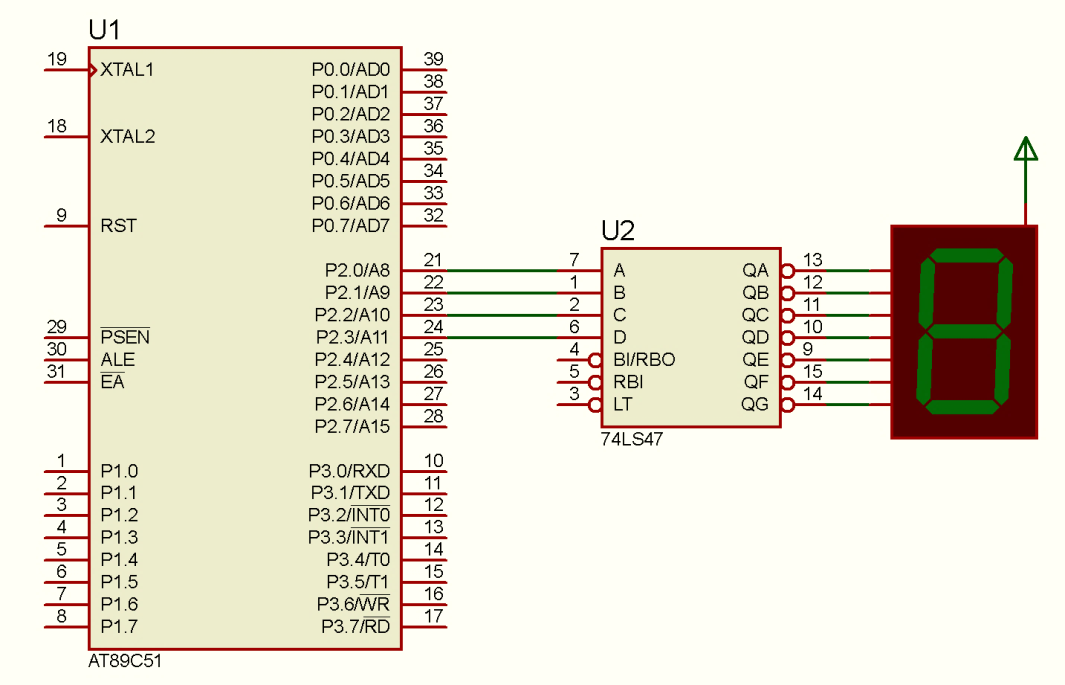
{

P2=so[dem];delay();

}}

**Bài toán 3:** LED 7 thanh Anode chung có a,b,c,d,e,f được nối với IC giải mã 74LS47. Chân A, B,C,D trên IC 74LS47 nối với cổng P2 của vi điều khiển.Điều khiển LED 7 thanh đếm từ 0 đến 9 sau đó đếm 9 về 0.

**+ So đồ mạch**

****

*Hình 3.7. Sơ đồ mạch nguyên lý*

**+ Chương trình điều khiển**

#include <REGX51.H>

long so[]={0x00,0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08,0x09};

int dem;

void delay()

{

long i;

for(i=0;i<10000;i++)

{

}

}

void main()

{

while(1)

{

//dem tien

for(dem=0;dem<10;dem++)

{

P2=so[dem];delay();

}

// Dem lui

for(dem=9;dem>=0;dem--)

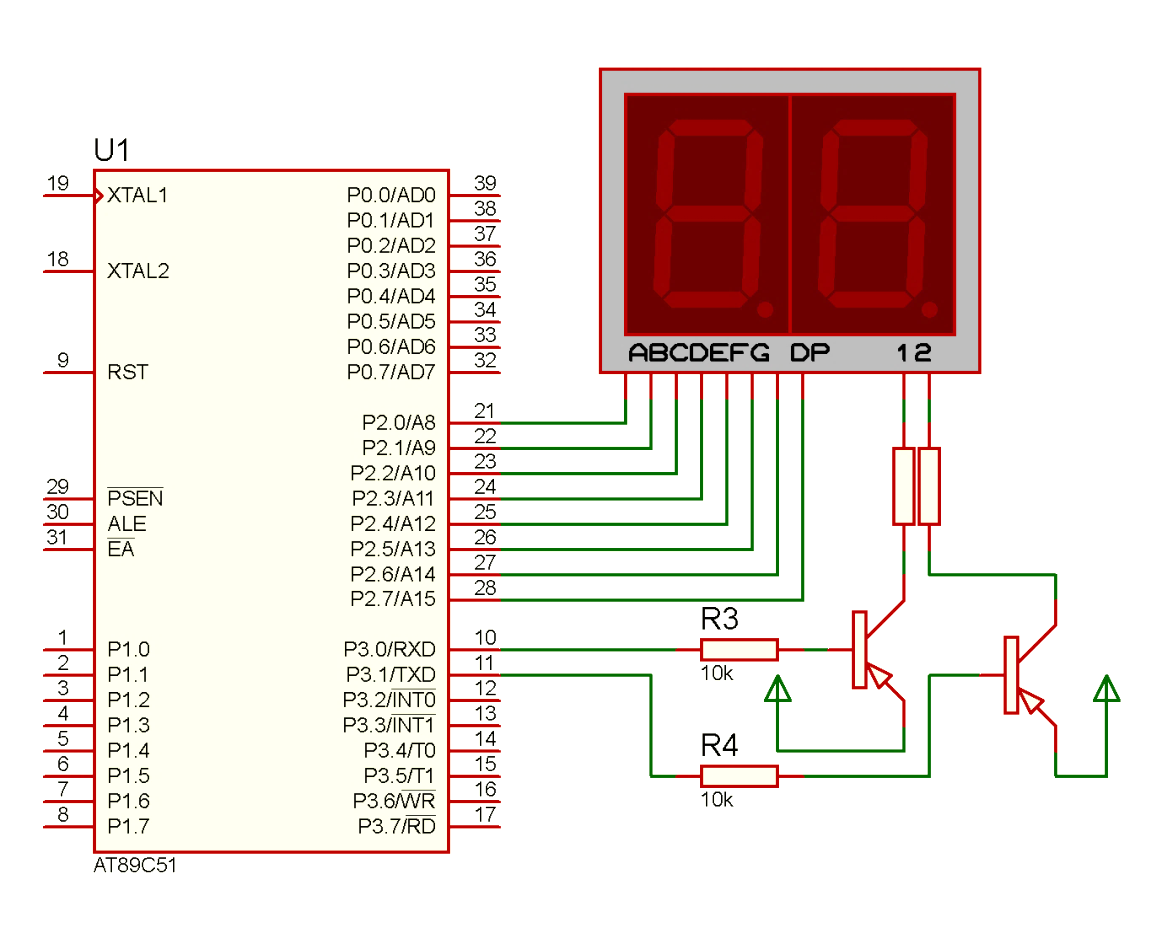
{

P2=so[dem];delay();

}

**Bài toán 4:** LED 7 thanh đôi các chân a,b,c,d,e,f được nối với các chân cổng P2 của vi điều khiển.Điều khiển LED 7 thanh đếm từ 00 đến 99 sau đó đếm 99 về 00.

**+ Sơ đồ mạch**

****

*Hình 3.8. Sơ đồ mạch nguyên lý*

**+ Chương trình điều khiển**

**#**include <REGX51.H>

char so[]={ 0x40,0x79,0x24,0x30,0x19,0x12,0x02,0x78,0x00,0x10};

#define led1 P3\_0

#define led2 P3\_1

#define sang 0;

#define tat 1;

int dem;

unsigned char chuc, donvi;

char i;

void delay(int t)

{

while(t--);

}

void demtien()

{

for(dem=0;dem<=99;dem++){

chuc=dem/10;

donvi=dem%10;

for(i=0;i<10;i++){

led1=sang;

P2= so[chuc];

delay(1000);

led1=tat;

led2=sang;

P2=so[donvi];

delay(1000);

led2=tat;

}}}

void demlui()

{

for(dem=99;dem>=0;dem--)

{

chuc=dem/10;

donvi=dem%10;

for(i=0;i<10;i++)

{

led1=sang;

P2=so[chuc];

delay(1000);

led1=tat;

led2=sang;

P2=so[donvi];

delay(1000);

led2=tat;

}}}

void main()

{

led1=led2=tat;

while(1)

{

demtien();

demlui();

}}

## **3.3. Lập trình điều khiển LCD**

* + 1. **Cấu tạo của LCD**

Ngày nay, thiết bị hiển thị [**LCD 1602**](http://www.suachualaptop24h.com/Linh-kien-laptop.html) (Liquid Crystal Display) được sử dụng trong rất nhiều các ứng dụng của vi điều khiển. LCD 1602 có rất nhiều ưu điểm so với các dạng hiển thị khác như: khả năng hiển thị kí tự đa dạng (chữ, số, kí tự đồ họa); dễ dàng đưa vào mạch ứng dụng theo nhiều giao thức giao tiếp khác nhau, tiêu tốn rất ít tài nguyên hệ thống, giá thành rẻ.

Thông số kĩ thuật  của sản phẩm LCD 1602:

- Điện áp MAX : 7V

- Điện áp MIN : - 0,3V

- Hoạt động ổn định : 2.7-5.5V

- Điện áp ra mức cao : > 2.4

- Điện áp ra mức thấp : <0.4V

- Dòng điện cấp nguồn : 350uA - 600uA

- Nhiệt độ hoạt động : - 30 - 75 độ C

 Số chân, tên của LCD 16X2 và các chức năng tương ứng được hiển thị trong bảng bên dưới:

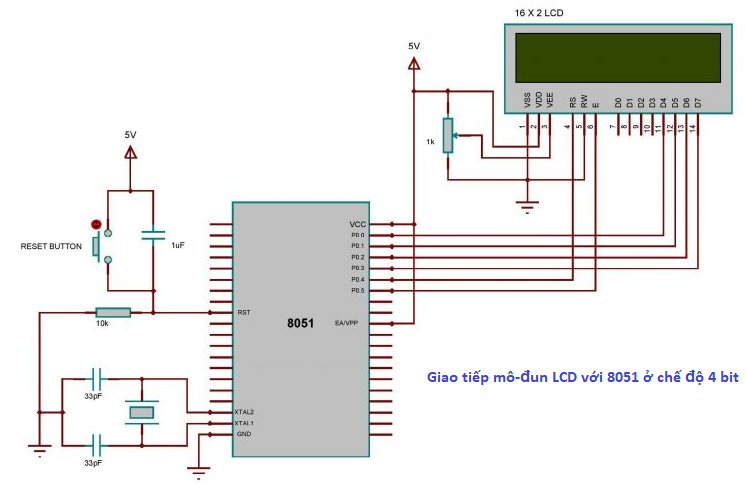
|  |  |  |
| --- | --- | --- |
| **Số Chân** | **Tên chân** | **Chức năng** |
| 1 | Vss | Chân này phải được nối GND |
| 2 | Vcc | Chân nối nguồn cấp (5V) |
| 3 | Vee | Chỉnh độ tương phản |
| 4 | RS | Chọn thanh ghi |
| 5 | R/W | Đọc hoặc ghi |
| 6 | E | Cho phép mô-đun |
| 7 | DB0 | Chận dữ liệu |
| 8 | DB1 | Chân dữ liệu |
| 9 | DB2 | Chân dữ liệu |
| 10 | DB3 | Chân dữ liệu |
| 11 | DB4 | Chân dữ liệu |
| 12 | DB5 | Chân dữ liệu |
| 13 | DB6 | Chân dữ liệu |
| 14 | DB7 | Chân dữ liệu |
| 15 | LED+ | Anode của led |
| 16 | LED- | Cathode của led |

Chân Vee là để điều chỉnh độ tương phản của màn hình LCD và độ tương phản có thể được điều chỉnh bằng cách thay đổi điện áp ở chân này. Thực hiện bằng cách nối một đầu của biến trở với Vcc (5V), đầu kia với GND và nối chân giữa của biến trở với chân Vee. Xem sơ đồ mạch để hiểu rõ hơn. LCD có hai thanh ghi tích hợp là thanh ghi dữ liệu và thanh ghi lệnh. Thanh ghi dữ liệu để đặt dữ liệu sẽ được hiển thị và thanh ghi lệnh để đặt các lệnh. LCD 16 × 2 có một bộ lệnh mỗi lệnh để thực hiện một công việc cụ thể với màn hình. Nếu mức logic cao đưa vào chân RS thì sẽ chọn thanh ghi dữ liệu và mức logic thấp ở chân RS thì sẽ chọn thanh ghi lệnh. Nếu chúng ta đặt chân RS lên cao và đặt dữ liệu vào dòng dữ liệu 8 bit (DB0 đến DB7) thì LCD sẽ nhận ra đó là dữ liệu sẽ được hiển thị. Nếu chúng ta làm cho chân RS ở mức thấp và đặt dữ liệu trên đường dữ liệu thì LCD sẽ nhận ra đó là một lệnh.

Chân R / W có nghĩa là để chọn giữa chế độ đọc và ghi. Mức cao ở chân này cho phép chế độ đọc và mức thấp ở chân này cho phép chế độ ghi.

Chân E là để kích hoạt LCD chuyển đổi từ cao xuống thấp ở chân này sẽ cho phép hoạt động. DB0 đến DB7 là các chân dữ liệu. Dữ liệu được hiển thị và lệnh được đặt trên các chân này.

LED+ là cực dương của đèn LED phía sau và chân này phải được kết nối với Vcc thông qua một điện trở giới hạn dòng thích hợp. LED- là cực âm của đèn LED phía sau và chân này phải được nối đất.



*Hình 3.9. Sơ đồ kết nối chuẩn để LCD hoạt động*

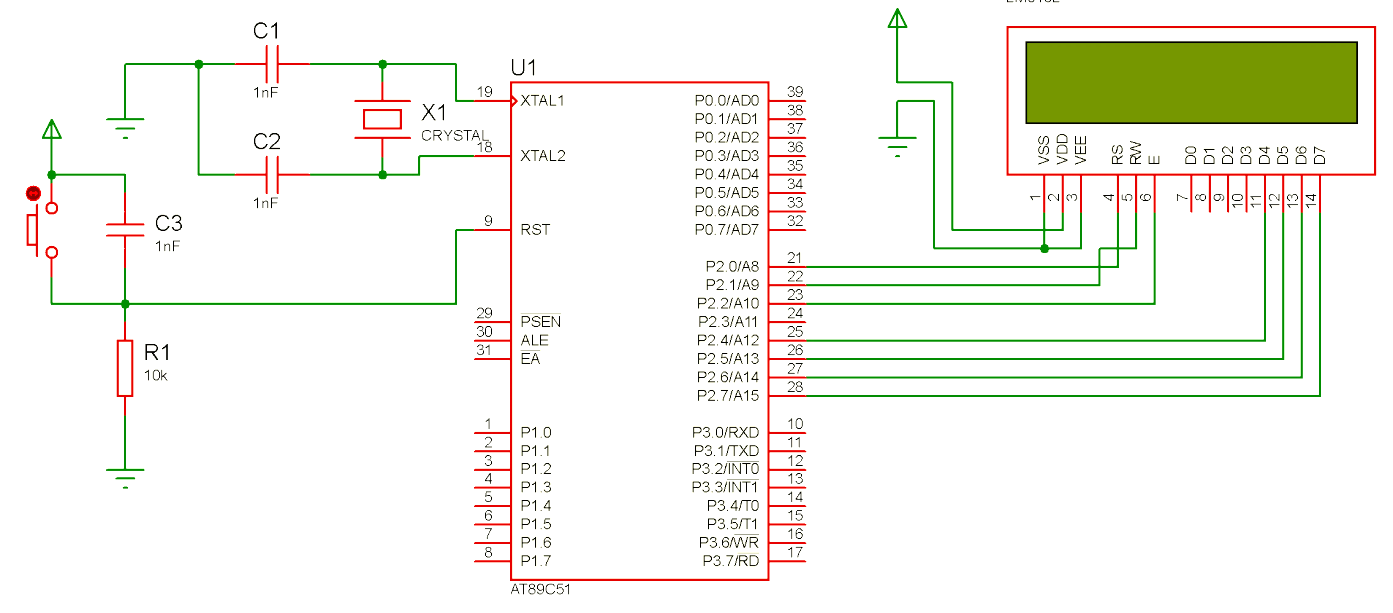
**+ Các lệnh cho mô-đun LCD 16x2**

|  |  |
| --- | --- |
| **Lệnh** | **Chức năng** |
| 0F | LCD bật, con trỏ bật, con trỏ nhấp nháy bật |
| 01 | Xoá toàn màn hình |
| 02 | Quay về màn hình chính |
| 04 | Giảm con trỏ |
| 06 | Tăng con trỏ |
| 0E | Màn hình bật, con trỏ nhấp nháy tắt |
| 80 | Bắt con trỏ trở về vị trí đầu tiên của hàng 1 |
| C0 | Bắt con trỏ trở về vị trí đầu tiên của hàng 2 |
| 38 | Sử dụng 2 hàng và ma trận 5x7 |
| 83 | Con trỏ hàng 1 vị trí 3 |
| 3C | Kích hoạt dòng 2 |
| 08 | Tắt màn hình hiển thị và con trỏ |
| C1 | Nhảy đến dòng 2 vị trí 1 |
| OC | Bật màn hình hiển thị, tắt con trỏ |
| C2 | Nhảy đến hàng 2, vị trí 2 |

### **3.3.2. Lập trình điều khiển LCD**

**Bài toán 1:** Cho LCD 16x2 được nối với cổng P2 của vi điều khiển. Lập trình điều khiển LCD hiện thị các ký tự bất kỳ lên màn hình. Yêu cầu giao tiếp chế độ 4 bit.

+ Sơ đồ mạch nguyên lý

****

*Hình 3.10. Sơ đồ mạch nguyên lý*

**+ Chương trình điều khiển:**

#include <REGX51.H>

/\*Khai bao chan giao tiep\*/

#define LCD\_RS P2\_0

#define LCD\_RW P2\_1

#define LCD\_EN P2\_2

#define LCD\_D4 P2\_4

#define LCD\_D5 P2\_5

#define LCD\_D6 P2\_6

#define LCD\_D7 P2\_7

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void delay\_us(unsigned int t){

unsigned int i;

for(i=0;i<t;i++);

}

void delay\_ms(unsigned int t){

unsigned int i,j;

for(i=0;i<t;i++)

for(j=0;j<125;j++);

}

/\*Ctr giao tiep LCD 16x2 4bit\*/

void LCD\_Enable(void){

LCD\_EN =1;

delay\_us(3);

LCD\_EN=0;

delay\_us(50);

}

//Ham Gui 4 Bit Du Lieu Ra LCD

void LCD\_Send4Bit(unsigned char Data){

LCD\_D4=Data & 0x01;

LCD\_D5=(Data>>1)&1;

LCD\_D6=(Data>>2)&1;

LCD\_D7=(Data>>3)&1;

}

// Ham Gui 1 Lenh Cho LCD

void LCD\_SendCommand(unsigned char command){

LCD\_Send4Bit(command >>4);/\* Gui 4 bit cao \*/

LCD\_Enable();

LCD\_Send4Bit(command); /\* Gui 4 bit thap\*/

LCD\_Enable();

}

void LCD\_Clear(){// Ham Xoa Man Hinh LCD

LCD\_SendCommand(0x01);

delay\_us(10);

}

// Ham Khoi Tao LCD

void LCD\_Init(){

LCD\_Send4Bit(0x00);

delay\_ms(20);

LCD\_RS=0;

LCD\_RW=0;

LCD\_Send4Bit(0x03);

LCD\_Enable();

delay\_ms(5);

LCD\_Enable();

delay\_us(100);

LCD\_Enable();

LCD\_Send4Bit(0x02);

LCD\_Enable();

LCD\_SendCommand( 0x28 ); // giao thuc 4 bit, hien thi 2 hang, ki tu 5x8

LCD\_SendCommand( 0x0c); // cho phep hien thi man hinh

LCD\_SendCommand( 0x06 ); // tang ID, khong dich khung hinh

LCD\_SendCommand(0x01); // xoa toan bo khung hinh

}

void LCD\_Gotoxy(unsigned char x, unsigned char y){

unsigned char address;

if(!y)address=(0x80+x);

else address=(0xc0+x);

delay\_us(1000);

LCD\_SendCommand(address);

delay\_us(50);

}

void LCD\_PutChar(unsigned char Data){//Ham Gui 1 Ki Tu

LCD\_RS=1;

LCD\_SendCommand(Data);

LCD\_RS=0 ;

}

void LCD\_Puts (char \*s){//Ham gui 1 chuoi ky tu

while (\*s){

LCD\_PutChar(\*s);

s++;

}

}

/\*Chuong trinh chinh\*/

void main(){

LCD\_Init();//Khoi tao LCD

delay\_ms(1000);

LCD\_Puts(" KY THUAT MAY TINH");//Gui chuoi len LCD

delay\_ms(1000);

LCD\_Clear();//Xoa man hinh

LCD\_Gotoxy(0,0);//Tro toi vi tri

LCD\_Puts(" KTMT K18 ");

delay\_ms(2000);

LCD\_Gotoxy(0,1);//Tro toi vi tri

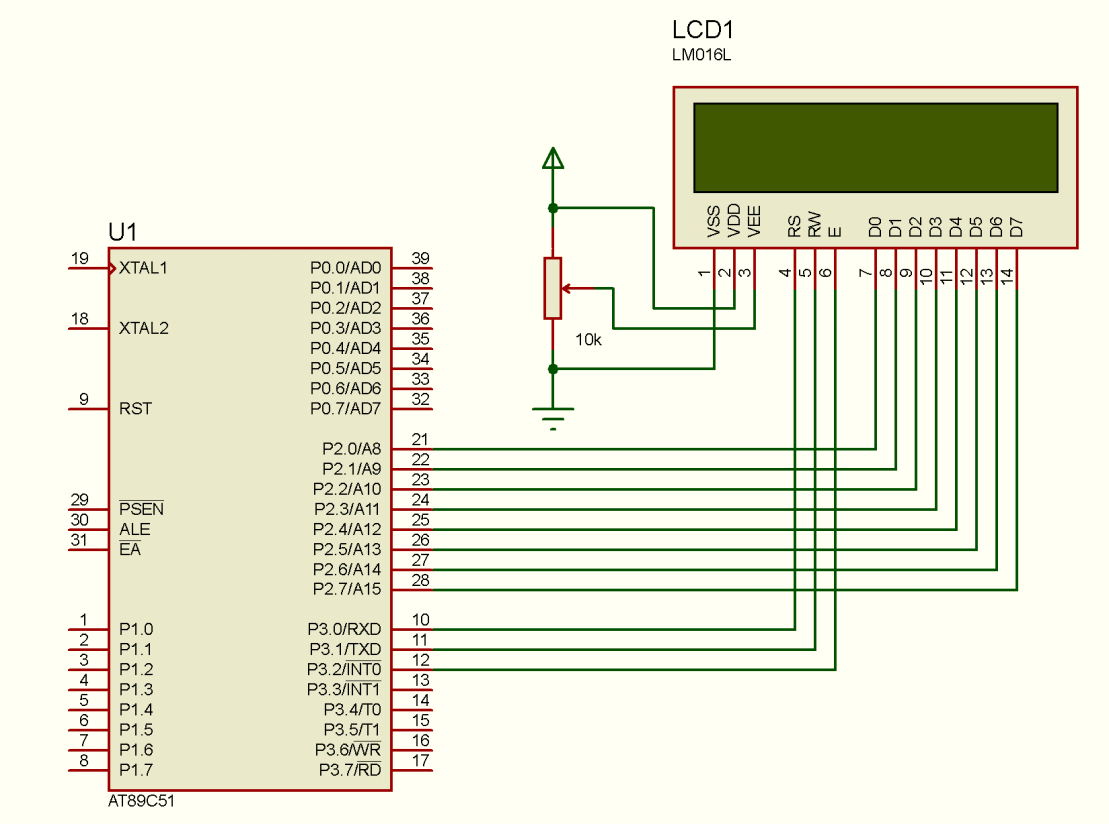
LCD\_Puts(" ICTU");

while(1);//ko lam gi ca

}

**Bài toán 2:** Cho LCD 16x2 được nối với cổng P2 của vi điều khiển. Lập trình điều khiển LCD hiện thị các ký tự bất kỳ lên màn hình. Yêu cầu giao tiếp chế độ 8 bit.

**+ Sơ đồ mạch nguyên lý**

****

*Hình 3.11. Sơ đồ mạch nguyên lý*

**+ Chương trình điều khiển**

#include <REGX51.H>

#include <string.h>

sbit RS=P3^0;

sbit RW=P3^1;

sbit EN=P3^2;

sbit BF=P2^7;

void delay(int t)

{

int a,b;

for(a=0; a<t; a++){

for(b=0;b<125;b++);

}}

// Ham kiem tra ban

void ban()

{

P2=0xFF;

RS=0; // Lenh

RW=1; // Doc gia tri tu LCD

delay(1); // Tao tre 1ms

while(BF){ // Kiem tra chan BF (D7)

EN=0;

delay(1);

EN=1;

}}

// Ham gui di mot lenh

void lenh(unsigned char x)

{

// Cac co ham kiem tra ban o day

ban();

P2=x; // Ma cua lenh can truyen

RS=0; // chon truyen lenh

RW=0; // 8051 ghi xuong LCD

delay(1);

EN=1;

delay(1);

EN=0;

}

// Ham gui di mot ky tu xuong LCD

void kitu(unsigned char x)

{

ban();

P2=x; // Ma cua lenh can truyen

RS=1; // chon truyen lenh

RW=0; // 8051 ghi xuong LCD

delay(1);

EN=1;

delay(1);

EN=0;

}

// Ham gui mot ky tu xuong LCD

void chuoi(unsigned char \*s) // s la chuoi can truyen

{

char i,l;

l=strlen(s); // Lay do dai chuoi ky tu

for(i=0;i<l;i++) // gui tung ky tu trong chuoi

{

kitu(s[i]);

delay(50);

}}

// Ham cai dat cho LCD

void khoitao()

{

lenh(0x38); // LCD hai dong ma tran diem 5x7

lenh(0x0c); // Bat man hinh, tat con tro

lenh(0x01); // xoa man hinh

lenh(0x80); // Dua con tro ve dau dong thu nhat

}

void main()

{

khoitao(); // Khoi tao cai dat

chuoi("XIN CHAO CAC BAN");

delay(500);

lenh(0xc0); // dua con tro ve dau dong 2

chuoi(" KTMT\_ICTU ");

while(1); // vong lap vo han

}

## **3.4. Lập trình xử lý ngắt**

### **3.4.1. Ngắt của bộ định thời**

* Nguyên tắc sử dụng bộ định thời ở chế độ ngắt:

+ Khai báo ngắt của bộ định thời muốn sử dụng

+ Nguồn gây ngắt.

+ Chương trình chính và chương trình phục vụ ngắt.

- Với ngắt định thời việc khai báo nguồn ngắt bao gồm việc khai báo ngắt toàn cục là EA = 1 và khai báo ngắt cho từng bộ định thời muốn dùng ET0 = 1(timer0) và ET1 = 1(timer1)

- Nguồn báo ngắt định thời(với cả 2 chế độ) đều là khi bộ đếm tràn (khi cờ TFx = 1), khi cờ TFx = 1 thì vi điều khiển kết thúc công việc hiện tại ở chương trình chính và chuyển vào chương trình phục vụ ngắt căn cứ theo địa chỉ của chương trình phục vụ ngắt. Khi thực hiện xong chương trình phục vụ ngắt vi điều khiển quay trở về chương trình tại nơi bị gián đoạn khi trước.

Tóm lại ngắt định thời là việc ta sử dụng bộ định thời tạo trễ một khoảng thời gian được tính toán trước, sau khoảng thời gian này 1 công việc ta mong muốn sẽ được thực hiện ở chương trình phục vụ ngắt. Công việc đó có thể coi là độc lập với công việc ở chương trình chính

- Các công việc lập trình sử dung ngắt định thời :

+ Tính toán thời gian trễ mong muốn và công việc muốn thực hiện sau khoảng thời gian đó.

+ Khai báo ngắt định thời trong chương trình chính (EA=1, ETX=1).

+ Lựa chon bộ định thời và chế độ của nó trong thanh ghi TMOD

+ Khởi động bộ định thời.

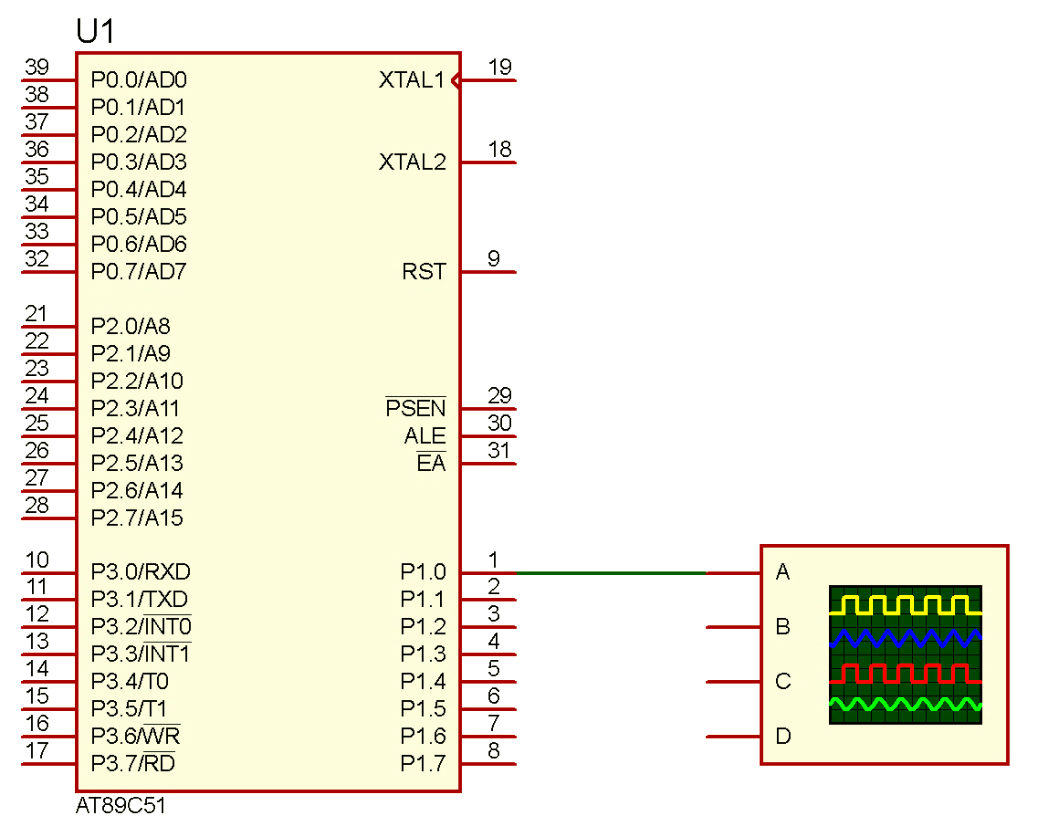
+ Xây dựng chương trình chính và các chương trình con cần thiết.

+ Xây dựng chương trình phục vụ ngắt thực hiện 1 công việc mong muốn (xác định rõ địa chỉ ngắt theo thứ tự ngắt trong bảng vector ngắt)

**Chú ý:** trong chương trình phục vụ ngắt phải xoá cờ báo tràn TFx và nạp lại giá trị cho bộ định thời với chế độ 1 còn chế độ 2 thì không cần.

**Bài toán 1:** sử dụng bộ định thời với ngắt tạo một xung vuông với tần số 1kHZ đối xứng (50% mức 1 và 50% mức 0) trên chân P1.0.

**+ Sơ đồ mạch nguyên lý**

****

*Hình 3.12. Sơ đồ mạch nguyên lý*

Ta có : f = 1kHZ → T = 1/1000s = 1000us → TON = TOFF = T/2 = 500us. Như vậy chu kì xung là đối xứng nên hai phần này bộ đếm đếm giống nhau(tạo trễ khoảng thời gian bằng nhau). nửa chu kì đầu bộ đếm tạo trễ 500us cho mức 0, sau khi bộ đếm đếm đến giá trị đỉnh của thanh ghi chứa là FFFF thì cờ báo TFx = 1 gây ngắt, vi điều khiển phải chuyển vào chương trình phục vụ ngắt và sẽ lật trạng thái xung ra. Trong nửa chu kì còn lại cũng như vậy và cứ như thế.

- Do thời gian trễ là 500us do đó ta chọn chế độ 16 bit, sử dụng kết quả từ ví dụ trên để nạp giá trị ban đầu cho TH và TL.

**+ Chương trình điều khiển**

/\*================khai bao thu vien===============\*/

#include<AT89x51.h>

/\*=============khai bao bien=====================\*/

sbit xung = P1^0;

/\*==============khai bao ham ==================\*/

/\*--------------------------chuong trinh phuc vu ngat timer0 tao xung--------------------\*/

void timer0\_int(void) interrupt 1

{

TF0=0; // xoa co bao tran timer0

xung = !xung; // tao muc 1 ra chan P1.0

TH0 = 0xfe; // (65536-500)↔hex =0fe0c (thach anh 12MHz)

TL0 = 0x0c;

}

/\*=============chuong trinh chinh=============\*/

void main(void)

{

EA = 1; // cho phep ngat toan cuc

ET0 = 1; // cho phep ngat T0

TMOD = 0x01; // timer0 che do 1

xung = 0; //gia tri ban dau cua xung muon tao

TH0 = 0xfe; // (65536-500)↔hex =0fe0c (thach anh 12MHz)

TL0 = 0x0c;

TR0 = 1; // khoi dong timer

while(1) PCON=1; //khong lam gi ca de doi ngat.

}

### **3.4.2. Lập trình với các ngắt ngoài của vi điều khiển**

- Bộ vi điều khiển 8051có hai ngắt ngoài là INT0 và INT1 với 2 chân tác động đầu vào tương ứng là P3.2 và P3.3.

- Ngắt ngoài là ngắt của vi điều khiển mà tác động ngắt ở đây chính là các tác động bên ngoài của vi điều khiển tác động vào vi điều khiển thong qua các chân ngắt đưới dạng một tín hiệu điện áp dạng xung.

- Tác động ngắt ngoài có hai dạng là tác động theo dạng mức và dạng sườn.

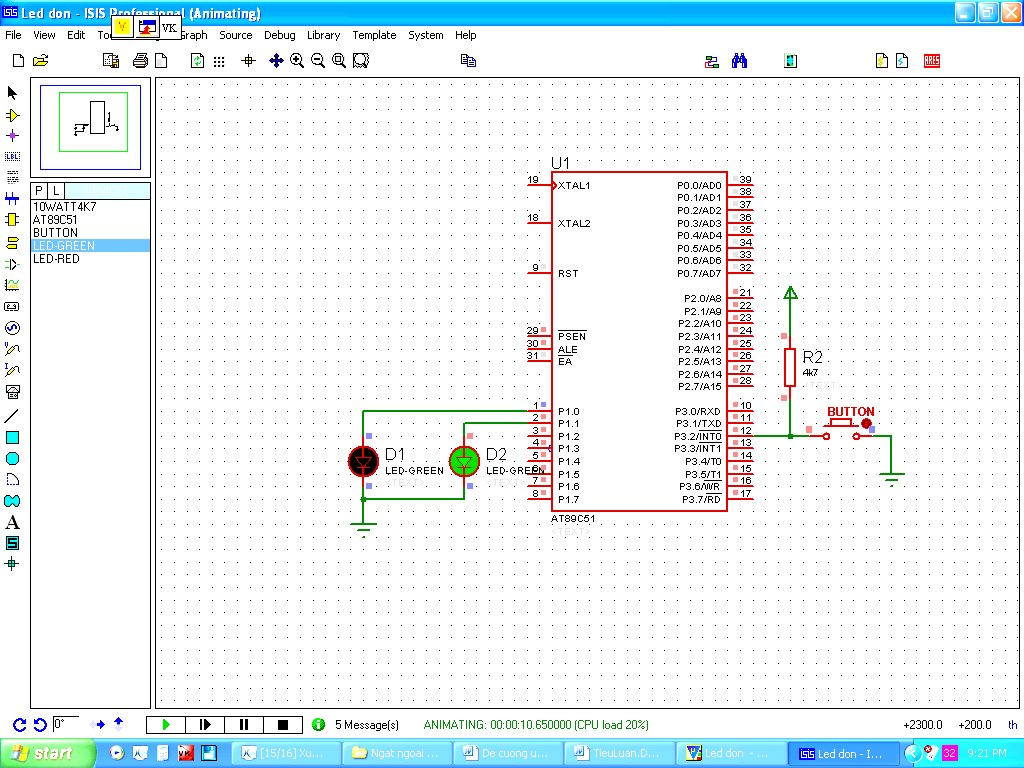
+ kích hoạt theo mức: ở chế độ này các chân INT0 và INT1 bình thường ở mức cao giống như các chân khác của vi điều khiển, khi có tín hiệu mức thấp cấp tới thì tín hiệu này kích hoạt ngắt. lưu ý là trước khi thực hiện lệnh cuối cùng của chương trình phục vụ ngắt thì mức thấp tại các chân ngắt phải được chuyển lên mức cao, nếu không sẽ lại gây ra một ngắt ngay lập tức.

+ Kích hoạt theo sườn: bình thường các chân ngắt của vi điều khiển ở mức cao, khi có tín hiệu tác động vào chúng có dạng sườn xuống thì sẽ tác động ngắt.

- Để sử dụng chế độ ngắt này thì phải tác động vào thanh ghi TCON cụ thể là TCON.1=1 hoặc IT0=1 thì cho phép ngắt ngoài 0 kích hoạt sườn, còn TCON.2=1 hoặc IT1=1 thì cho phép ngắt ngoài 1 kích hoạt sườn.

**Bài toán 1:** Sử dụng ngắt ngoài của vi điều khiển với nhiệm vụ là: bình thường vi điều khiển bật một LED đơn tại chân P1.0 và khi có ngắt ngoài thì tắt LED ở P1.0 và bật Led ở chân P1.1 sau 1s thì bật lại LED P1.0 và tắt LED P1.1.

+ Sơ đồ mạch nguyên lý

**

*Hình 3.13. Sơ đồ mạch nguyên lý*

+ Chương trình điều khiển

/\*==================Bo tien xu li============\*/

#include<AT89x51.h> // Dinh kem file thu vien

#define bat 1 // Dinh nghia gia tri bat den Led

#define tat 0 // Dinh nghia gia tri tat den Led

/\*==================khai bao bien==================\*/

sbit Led1 = P1^0; // Khai bao bi?n Led ki?u bit chan P1.0

sbit Led2 = P1^1; // Khai bao bi?n Led ki?u bit chan P1.0

/\*================= Khai bao hàm==================\*/

/\*------------------------------ham tre -------------------------------------\*/

void delay(long time)

{

while(time--);

}

void INT\_0(void) interrupt 0 // chuong trinh phuc vu ngat

{

Led1=tat;

Led2=bat;

delay(25000);

Led1=bat;

Led2=tat;

}

/\*--------------------------------ham chinh--------------------------------\*/

void main(void)

{

EA=1; // cho phep ngat toan cuc

EX0=1; // cho phep ngat ngoai 0

IT0=1; //ngat kich phat suon

Led1=bat;

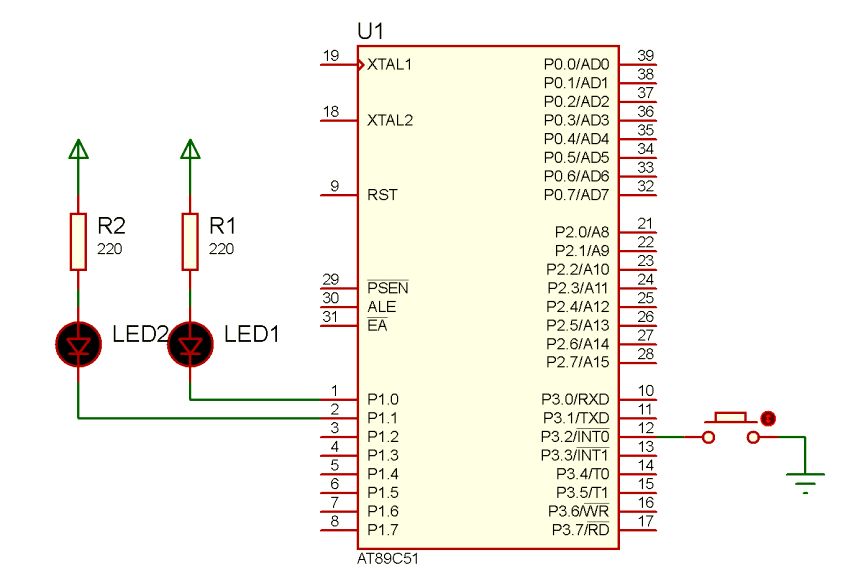
Led2=tat;

while(1);

}

**Bài toán 2:** Sử dụng ngắt ngoài INT0 theo mức để nháy LED1 trên chân P1\_0 (5 lần) khi công tắc tại chân P3\_2 được nhấn. Chương trình chính nháy liên tục LED2 trên chân P1\_1.

+ Sơ đồ mạch nguyên lý



*Hình 3.14. Sơ đồ mạch nguyên lý*

+ Chương trình điều khiển

#include <REGX51.H>

void delay(long time)

{

long n;

for(n=0;n<time;n++)

{}

}

void khoi\_tao()

{

P3\_2=1;

EA=1;// cho phep ngat toan cuc

IT0=1; // bit dieu khien ngat ngoai

EX0=1;// cho phep ngat ngoai 0

}

void ngat() interrupt 0

{

int i;

for(i=0;i<5;i++)

{

P1\_0=~P1\_0;

delay(1000);

}

}

void main()

{

P1\_0=1;

P1\_1=1;

khoi\_tao();

while(1)

{

P1\_1=~P1\_1;

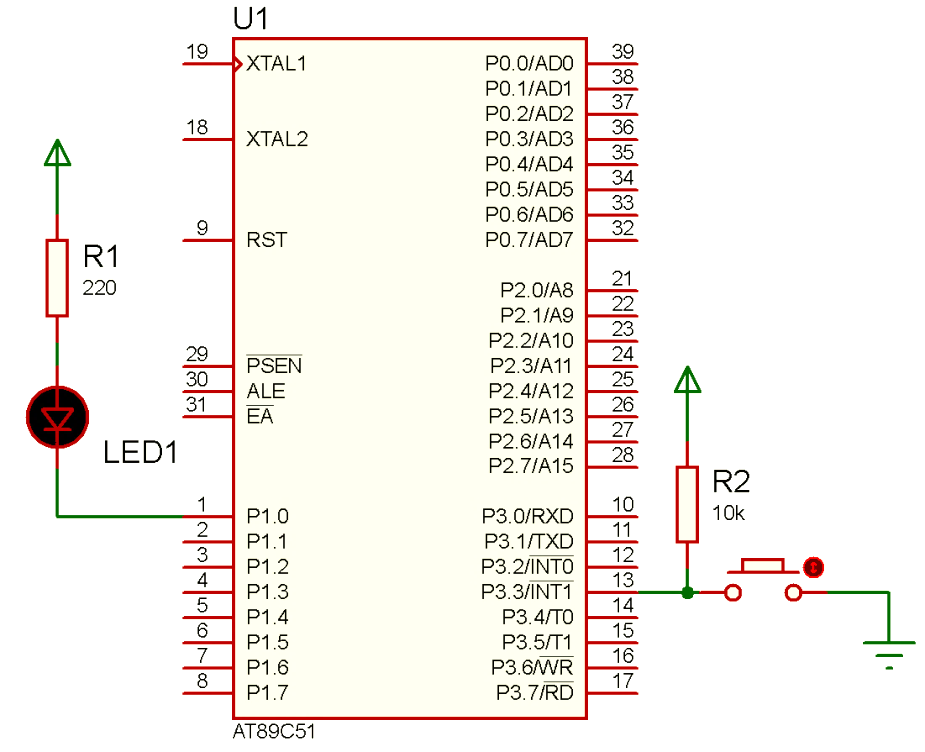
delay(1000);

}

}

**Bài toán 2:** Chân ngắt ngoài INT1 được nối với nút nhấn bình thường ở mức cao. Khi nút nhấn xuống mức thấp thì LED1 nối với chân P1\_0 bật sáng trong một khoảng thời gian. Nếu công tắc được giữ ở mức thấp LED1 sáng liên tục. ( Ngắt theo mức)

+ Sơ đồ mạch nguyên lý



*Hình 3.15. Sơ đồ mạch nguyên lý*

**+ Chương trình điều khiển**

#include <REGX51.H>

void delay(long time)

{

long n;

for(n=0;n<time;n++)

{} }

void main()

{

IE=0x84 ;

//IT1=1;// Ngat theo muc

while(1)

{}}

void nutan(void) interrupt 2

{

int a=50000;

P1\_3=0;

while(a--)

{ }

P1\_3=1;

}

## **3.5. Lập trình Timer**

### **3.5.1. Cơ chế tạo trễ của timer và cách tính toán giá trị nạp vào timer**

* ***Chế độ 1:***

- Ơ chế độ 1 đó là bộ định thời 16 bit, do đó các giá trị trong khoảng từ 0000 đến FFFF có thể sử dụng để nạp cho TH và TL của bộ định thời.

- Sau khi TH, TL được nạp giá trị ban đầu 16 bit thì bộ định thời phải được khởi động với lệnh TR0 = 1 với timer0 và TR1 = 1 với timer1, khi này bộ định thời bắt đầu đếm tăng theo xung clock từ giá trị ban đầu cho tới giá trị đỉnh là FFFF. Khi đó bộ định thời sẽ quay vong từ FFFF về 0000 và bật cờ báo tràn TF0 với timer0 và TF1 với timer1(ban dầu TFx=0) khi đó cần phải có thao tác để xoá cờ để lần sau còn biết được khi nào bộ đếm tràn.

- Khi bộ đếm tràn thì TH và Tl của bộ định thời sẽ mang giá trị 0 do đó phải có thao tác nạp lại giá trị ban đầu cho chúng và xoá cờ TF để bộ đếm được lặp lại.

- Như vậy khoảng thời gian mà bộ định thời tạo trễ chính là khoangre thời gian nó đếm tăng từ giá trị ban đầu được nạp cho tới giá trị đỉnh FFFF.

- Tính toán giá trị nạp vào thanh ghi chứa của bộ định thời từ thời gian muốn tạo trễ(t): gọi N = 65536 – t/chu ki máy. Sau khi chuyển sang mã HEX được 1 số có 4 chữ số có dạng: x1x2y1y2. khi đó giá trị nạp cho thanh ghi chứa như sau: TH = x1x2, TL = y1y2.

Ví dụ: muốn tạo trễ 500us thì:

Giả sử chu kì máy là 1us: có N = 65536 – 500/1 = 65036. chuyển sang mã HEX là FE0C, từ đó: TH = 0xFE, TL = 0x0C.

- Các bước lập trình cho bộ định thời để tạo trễ ở chế độ 1:

+B1: chọn chế độ 1 cho bộ định thời cần dung, từ đó xác định giá trị nạp cho thanh ghi TMOD.

+ B2: tính toán giá trị ban đầu cần nạp cho TH và TL từ thời gian trễ mong muốn.

+ B3: khởi động bộ định thời.

+ B4: kiểm tra trạng thái bật của cờ TF.

+ B5:dung bộ định thời.

+ B6: xoá cờ TF cho vòng lặp kế tiếp.

+ B7: quay trở về B2 để nạp lại giá trị cho TH và TL.

* ***Chế độ 2:***

- Ở chế độ này bộ định thời là 8 bit, do vậy chỉ cho phép các giá trị từ 00 đến FF được nạp vào thanh ghi TH của bộ định thời. sau khi nạp giá trị 8 bit thì vi điều khiển sẽ sao nội dung của TH sang TL và bộ định thời được khởi động bằng lệnh TRx=1.

- Sau khi được khởi động thì bộ định thời bắt đầu đếm tăng trong thanh ghi TL từ giá trị ban đầu đến giá trị đỉnh FF. và khi quay vòng từ FF về 00 thì cờ TFx được bật lên 1, khi này thanh ghi TL mang giá trị 0 nhưng TL sẽ ngay lập tức được tự động nạp lại với giá trị ban đầu được lưu trong thanh ghi TH.

- Như vậy trong chế độ này mỗi khi bộ đếm tràn thì thanh ghi chứa sẽ được vi điều khiển nạp lại giá trị ban đầu, do đó chế độ này được gọi là chế độ tự động nạp lại.

- Tính giá trị nạp vào thanh ghi chứa của bộ định thời từ thời gian trễ (t):

Gọi N = 256 – t/chu kì máy. Sau khi chuyển sang mã HEX được 1 số, ta nạp số này vào thanh ghi TH và cả TL(nạp vào TL giá trị đếm lần 1 và nạp vào TH cho những lần sau để khi bộ đếm tràn thì phần cứng của vi điều khiển sẽ tự động sao TH sang TL).

Ví dụ: tạo trễ 50usthì là:

Giả sử chu kì máy là 1us: có N = 256 – 50/1 = 205, chuyển sang mã HEX : 0CE

Từ đó: TH = 0xCE và TL = 0xCE.

- Các bước lập trình cho bộ định thời để tạo trễ ở chế độ 2.

+ B1: nạp giá trị cho thanh ghi thiết lập chế độ TMOD với timer muốn sử dụng.

+ B2: nạp vào TH giá trị đém ban đầu.

+B3: khởi động bộ định thời.

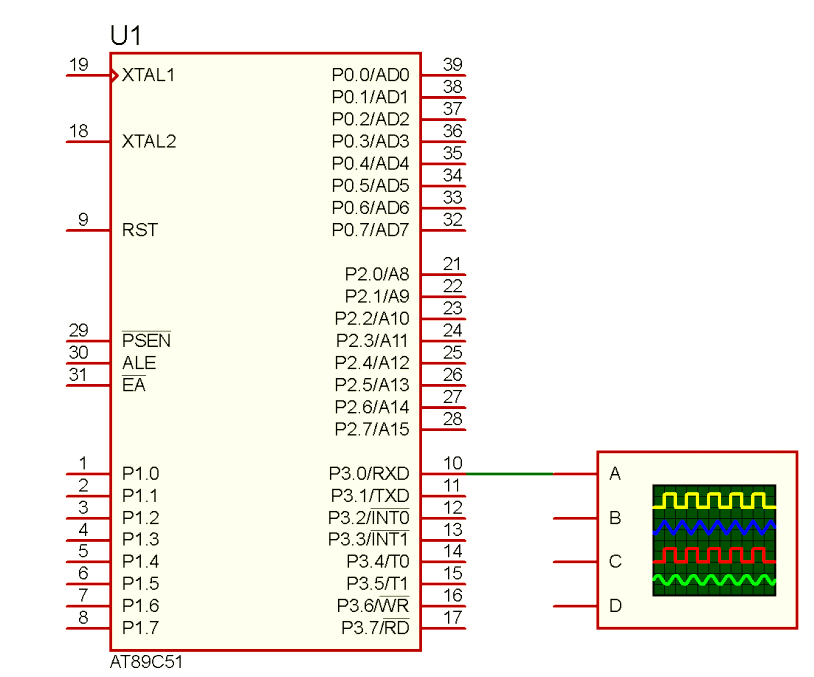
+ B4: kiểm tra cờ báo tràn TFx.

+ B5: xoá cờ tràn TFx, quay về B4.

**Lưu ý:** 2 chế độ khác của bộ định thời là chế độ 0 (chế độ bộ định thời 13 bit, bộ định thời/bộ đếm 8 bit, định tỷ lệ trước 5 bit ) và chế độ 3 (chế độ bộ định thời chia tách).

**Bài toán 1:** Lập trình timer 0 chế độ 1 tạo thời gian trễ 5ms sử dụng tần số thạch anh 11.0592MHz hiển thị quả lên máy hiện sóng

**+ Sơ đồ mạch nguyên lý**



*Hình 3.16. Sơ đồ mạch nguyên lý*

**+ Chương trình điều khiển**

#include <REGX51.H>

void delay()

{

TMOD=0x01;

TH0=0xFC;

TL0=0x18;

TR0=1 ;

while(!TF0);

TF0=0;

TR0=0;

}

void main()

{

P3\_0=0;

while(1)

{

P3\_0=~P3\_0;

delay();

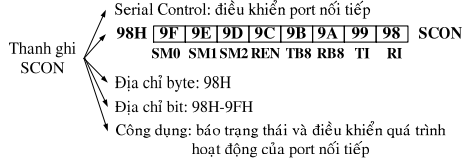
}

}

## **3.6. Lập trình truyền thông nối tiếp UART**

Giao tiếp UART là chuẩn giao tiếp nối tiếp được sử dụng phổ biến trong các dòng vi điều khiển. Mặc dù có tốc độ truyền không cao, nhưng ngược lại giao tiếp UART có phần cứng gọn nhẹ, dễ thiết kế và giá thành thấp. Giao tiếp UART được sử dụng nhiều trong các ứng dụng như giao tiếp máy tinh, giao tiếp module SIM, module GPS.

#### **Thanh ghi cấu hình cổng nối tiếp SCON.**

     Đây là thanh ghi 8 bit, chứa các bit sử dụng để cài đặt chế độ của giao tiếp UART và các bit cờ báo trạng thái truyền và nhận.  




SCON là thanh ghi trạng thái và điều khiển cổng nối tiếp. Nó không chứa các bít chọn chế độ, mà còn chứa bít dữ liệu thứ 9 dành cho việc truyền và nhận tin (TB8 và RB8) và chứa bít ngắt cổng nối tiếp.

- SM0, SM1: Là các bít cho phép chọn chế độ cổng nối tiếp.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SM0 | SM1 | Mode | Đặc điểm | Tốc độ Baud |
| 0 | 0 | 0 | Thanh ghi dịch | F/12 |
| 0 | 1 | 1 | 8 bít UART | Có thể thay đổi ( được đặt bởi bộ Timer) |
| 1 | 0 | 2 | 9 bít UART | F/64 hoặc F/32 |
| 1 | 1 | 3 | 9 bít UART | Có thể thay đổi ( được đặt bởi bộ Timer) |

#### **Thanh ghi sử dụng cài đặt tốc độ Baudrate: TMOD, TH1**

Như các bạn thấy trong Bảng 1.2, tùy vào chế độ hoạt động mà tốc độ baudrate có cách tính khác nhau. Với chế độ được sử dụng thông dụng nhất, chế độ 1, 8 bit dữ liệu, 1 bit start, 1 bit stop. Tốc độ baudrate được xác định qua timer 1.

     Trước hết để sử dụng timer 1 để xác định tốc độ baudrate chúng ta cần cài đặt timer 1 ở chế độ 8 bit tự nạp lại. Do đó, cần gán giá trị 0x20 cho thanh ghi TMOD (TMOD = 0x20).  Giá trị gán vào thanh ghi TH1 để xác định tốc độ baudrate được tính như sau.  TH1 =256 - (F(crystal)/12/32/baudrate).    (Với F(crystal) tính bằng đơn vị Hz).

     Ví dụ: Muốn truyền UART với tốc độ baudrate là 9600 ta cần gán cho TH1 giá trị sau.

     TH1 =256 - (F(crystal)/12/32/9600).

     Nếu mạch sử dụng thạch anh 11.0592 MHz = 11059200 Hz, ta sẽ có:

     TH1 = 256 - (11059200/12/32/9600) = 253.

     Như vậy gán cho thanh ghi TH1 giá trị 253 (TH1 = 253), là ta đã cài đặt giao tiếp UART với tốc độ baudrate 9600.

#### **Bit nhân đôi tốc độ baudrate**

     Trong 8051 có một bit được sử dụng để nhân đôi tốc độ truyền của giao tiếp UART. Đó là bit SMOD (bit thứ 7) trong thanh ghi PCON. Khi set bit SMOD lên 1, tốc độ truyền của giao tiếp UART sẽ được nhân đôi.

     Trở lại với ví dụ ở mục 2 phía trên, nếu như chúng ta muốn sử dụng tốc độ baudrate 19200 thì giá trị gán cho TH1 phải là bao nhiêu? Mạch vẫn sử dụng thạch anh 11.0592 MHz.

     Theo công thức tính đã cho ở trên ta sẽ có:

     TH1 = 256 - (11059200/12/32/19200) = 254.5

     Nhưng số cần gán cho TH1 phải là một số nguyên, nếu chúng ta gán 254 hoặc 255 thì vẫn không đúng tốc độ baudrate 19200.

     Vậy để có được tốc độ baudrate 19200, chúng ta cần sử dụng bit nhân đôi tốc độ SMOD. Gán giá trị 253 cho thanh ghi TH1 (để có baudrate 9600), và set bit SMOD lên 1 để nhân đôi tốc độ lên 19200.

#### **Thanh ghi truyền/nhận dữ liệu: SBUF.**

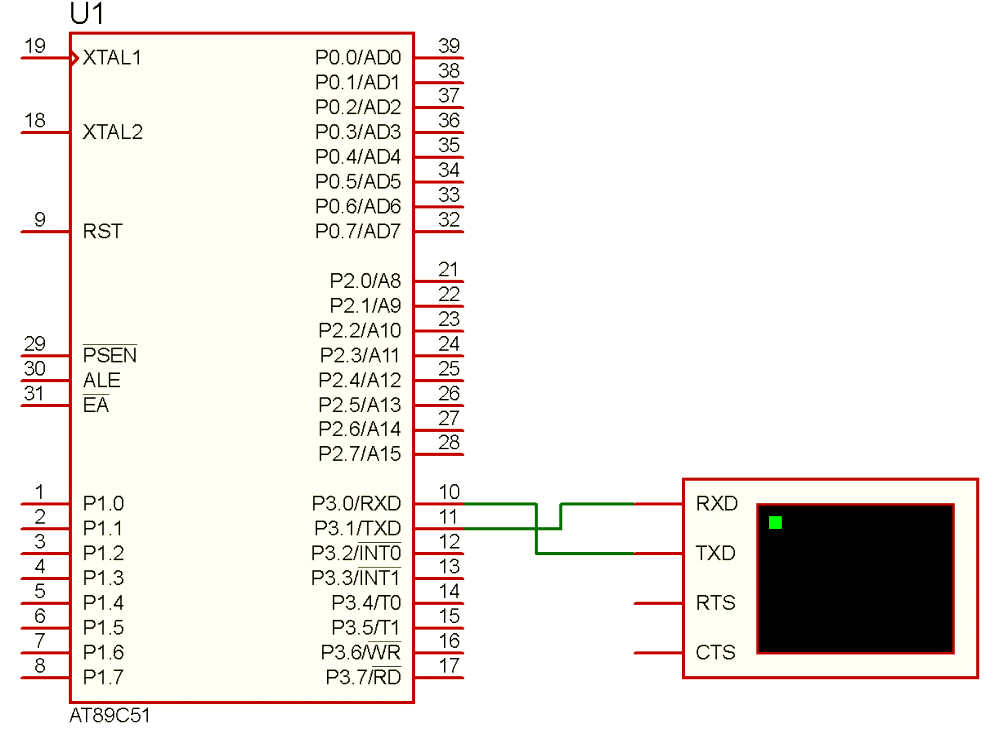
     SBUF là thanh ghi 8 bit, được sử dụng chung cho cả truyền và nhận dữ liệu trong UART.

     Để truyền dữ liệu, chúng ta chỉ cần ghi dữ liệu vào thanh ghi SBUF.

     Để nhận dữ liệu chúng ta đọc giá trị từ thanh ghi SBUF.

**Bài toán 1:** Lập trình truyền thông chuẩn UART tốc độ Baud là 9600 để truyền các ký tự từ a đến k.

**+ Sơ đồ mạch nguyên lý**



*Hình 3. 17. Sơ đồ mạch nguyên lý*

**+ Chương trình điều khiển**

#include <REGX51.H>

void maint()

{

char i ;

// khoi tao UART 0 mode 1, 9600

SM0= 0; SM1= 1; // chon uart mode 1

TMOD =0x20; // 0010xxxx time 1 hoat dong o che do 8 bit

TH1= 0xFD; // tra bang toc do 9600

TR1=1; //Time bat dau chay

TI=1; // San sang gui du lieu

for(i='a';i<='k'; i++)

{

while(TI==0);

TI=0;

SBUF =i ; //truyen ky tu a den ky tu z qua chuan uart

}

while(1)

{} }

**Bài toán 2:**

# **CHƯƠNG 4: THIẾT KẾ HỆ THỐNG NHÚNG**

## **4.1. Thiết kế hệ thống hiện thị 8 ký tự dùng vi điều khiển AT89C51**

### **4.1.1. Yêu cầu thiết kế**

- Hệ thống điều khiển được 8 ký tự dùng trên hai cổng P2

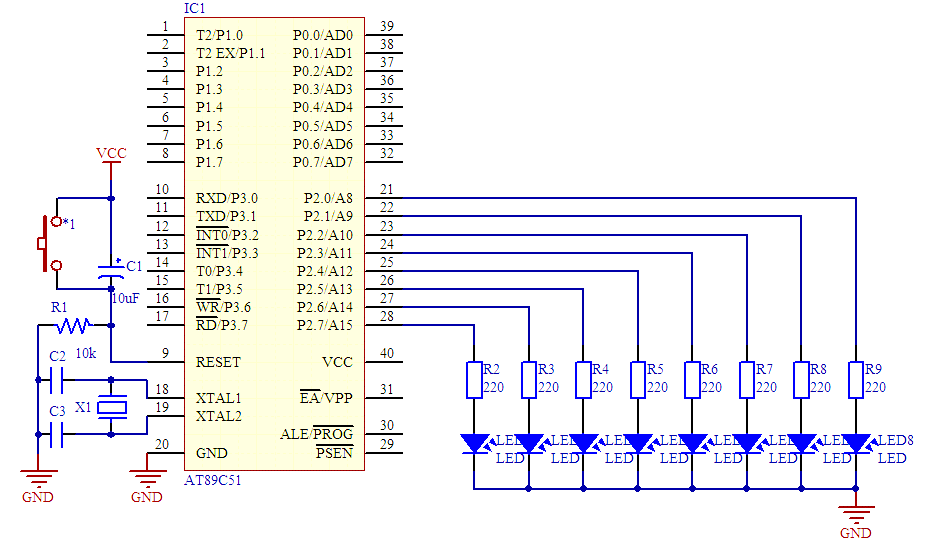
- Các ký tự sáng và giữ nguyên trạng thái lần lượt từ trái sang phải thời gian trễ là 0.5s.

- Các ký tự tắt và giữ nguyên trạng thái lần lượt từ phải qua trái thời gian trễ 0.5s

- Các ký tự sáng xen kẽ các ký tự chẵn sáng rồi đến ký tự lẻ sáng

- Tất cả các ký tự cùng nhấp nháy 5 xung nhịp.

**4.1.2. Thiết kế sơ đồ nguyên lý**

****

*Hình 4.1. Sơ đồ mạch nguyên lý*

**Chọn linh kiện trong hệ thống**

- IC1 là chip vi điều khiển AT89C51

* Điện trở R1, R2, R3, R4, R5, R6, R7 có giá trị 220
* Trở treo RN1 có giá trị 10
* Thạch anh X1 có giá trị 12Mhz
* Tụ gốm không phân cực C2, C3 có giá trị 33pF
* Điện trở R9 có giá trị 10
* Tụ C1 là tụ phân cực giá trị 10uF

**4.1.3. Xây dựng chương trình điều khiển**

*#include <REGX51.H>*

*void delay(unsigned char t){*

*//Chuong trinh tao tre 50ms*

*unsigned char i;*

*for(i=0;i<t;i++){*

*TH0=0x3c;*

*TL0=0xb0;*

*TR0=1;*

*while(!TF0);*

*TF0=TR0=0;*

*}}*

*unsigned char w; //khai bao bien*

*unsigned char m1[8]={0xfe,0xfc,0xf8,0xf0,0xe0,0xc0,0x80,0x00}; //mang bit*

*void nhay\_thuan(){//ctr con nhay lan luot*

*P2=0xFF; //sang tat ca*

*delay(50);*

*for(w=0;w<8;w++){*

*P2=m1[w]; //xuat mang m1 ra P2 chieu thuan*

*delay(10); //0.5s*

*}}*

*void nhay\_nghich(){//ctr con nhay lan luot*

*P2=0xFF; //sang ca tat*

*delay(50);*

*for(w=0;w<8;w++){*

*P0=m1[7-w]; //xuat mang m1 ra P2 chieu nguoc*

*delay(50); //0.5s*

*}}*

*void xen\_ke(){//ctr con nhay xen ke*

*P2=0x00; //tat ca tat*

*delay(50);*

*for(w=0;w<10;w++){*

*P2=0xAA; // 0xAA = 01010101*

*delay(50);*

*P2=0x55; // 0x55 = 10101010*

*delay(50);*

*}}*

*void nhap\_nhay(){//ctr con nhap nhay*

*P2=0xFF; //sang tat ca*

*delay(50);*

*//Nhap nhay tat ca 15 lan*

*for(w=0;w<15;w++){*

*P2=0xFF; //sang*

*delay(10);*

*P2=0x00; //tat*

*delay(50);*

*}}*

*void main(){ //Chuong trinh chinh*

*P2=0xFF; //Tat ca sang*

*delay(50); //Tre 1s do 20x50=1000ms=1s*

*while(1){*

*nhay\_thuan();*

*nhay\_nghich();*

*xen\_ke(); //Goi ctr con nhay xen ke*

*nhap\_nhay(); //Goi ctr con nhap nhay*

*}}*

**4.2. Thiết kế hệ thống giao tiếp bàn phím 4x4 dùng AT89C51**

**4.2.1. Yêu cầu hệ thống**

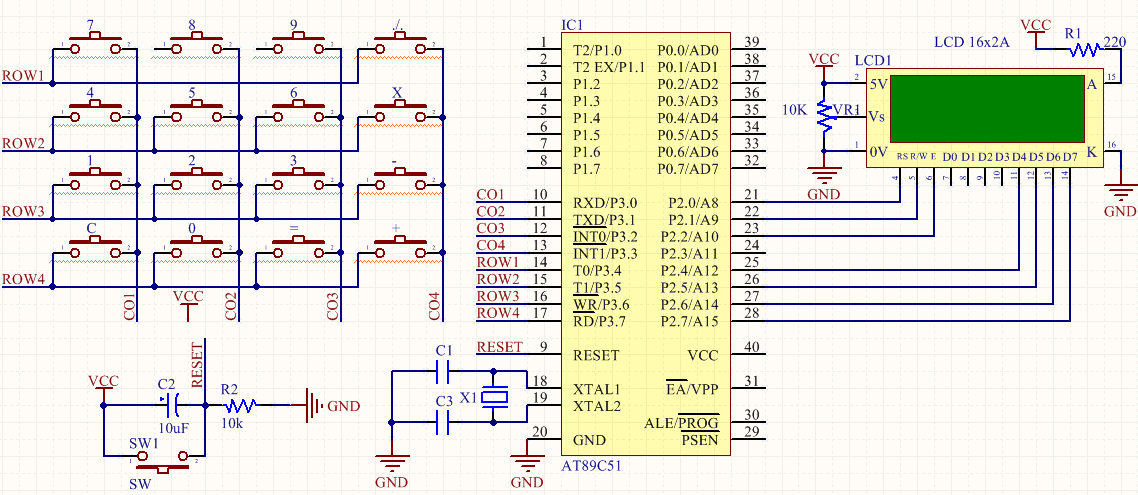
- Hệ thống dùng vi điều khiện họ 8051

- Hệ thống hiện số phím khi chúng ta nhấn vào phím đã quy định trước.

- Khi không nhấn màn hình ở trạng thái chờ

- Hiện thị phím bấm lên LCD

**4.2.2. Thiết kế sơ đồ nguyên lý**

****

*Hình 4.2. Sơ đồ mạch nguyên lý*

**Chọn linh kiên trong hệ thống**

- IC1 là chip vi điều khiển AT89C51

- Điện trở R1 có giá trị 220, điện trở R2 có giá trị 10k,

- Tụ C1, C2 là tụ gốm không phân cực có giá trị 33pF

- Tụ C2 là tụ phân cực có giá trị 10uF

- Bộ tạo dao động thạnh anh X1 có giá trị 12Mhz

- Màn hình hiển thị LCD 16x2

**4.2.3. Xây dựng chương trình điều khiển**

*#include <REGX51.H>*

*//Khai bao chan quet nut nhan*

*#define col0 P3\_0 //cot*

*#define col1 P3\_1*

*#define col2 P3\_2*

*#define col3 P3\_3*

*#define row0 P3\_4 //hang*

*#define row1 P3\_5*

*#define row2 P3\_6*

*#define row3 P3\_7*

*//Khai bao chan giao tiep LCD16x2 4bit*

*#define LCD\_RS P2\_0*

*#define LCD\_RW P2\_1*

*#define LCD\_EN P2\_2*

*#define LCD\_D4 P2\_4*

*#define LCD\_D5 P2\_5*

*#define LCD\_D6 P2\_6*

*#define LCD\_D7 P2\_7*

*/\*\*Ham delay\*\*/*

*void delay\_us(unsigned int t){*

*unsigned int i;*

*for(i=0;i<t;i++);*

*}*

*void delay\_ms(unsigned int t){*

*unsigned int i,j;*

*for(i=0;i<t;i++)*

*for(j=0;j<125;j++);*

*}*

*/\*Ham quet phim nhan 4x4\*/*

*unsigned char check\_but(){ //Kiem tra nut nhan*

*row0=row1=row2=row3=0;*

*if(!col0 || !col1 || !col2 || !col3 )return 1;*

*return 0;*

*}*

*void scan\_row(unsigned char r){ //Quet hang*

*row0=row1=row2=row3=1;*

*if(r==0)row0=0;*

*else if(r==1)row1=0;*

*else if(r==2)row2=0;*

*else if(r==3)row3=0;*

*}*

*unsigned char check\_col()*

*{*

*unsigned char c=0;*

*if(!col0)c=1;*

*else if(!col1)c=2;*

*else if(!col2)c=3;*

*else if(!col3)c=4;*

*return c;*

*}*

*unsigned char get\_key(){*

*unsigned char row,col;*

*if(check\_but()){*

*delay\_ms(5);*

*if(check\_but()){*

*for(row=0;row<4;row++){*

*scan\_row(row); //Quet hang*

*col=check\_col(); //Lay vi tri cot*

*if(col>0)return ((row\*4)+col);*

*} }}*

*return 0; //Khong co nut nao*

*}*

*/\*Ctr giao tiep LCD 16x2 4bit\*/*

*void LCD\_Enable(void){*

*LCD\_EN =1;*

*delay\_us(3);*

*LCD\_EN=0;*

*delay\_us(50);*

*}*

*//Ham Gui 4 Bit Du Lieu Ra LCD*

*void LCD\_Send4Bit(unsigned char Data){*

*LCD\_D4=Data & 0x01;*

*LCD\_D5=(Data>>1)&1;*

*LCD\_D6=(Data>>2)&1;*

*LCD\_D7=(Data>>3)&1;*

*}*

*// Ham Gui 1 Lenh Cho LCD*

*void LCD\_SendCommand(unsigned char command){*

*LCD\_Send4Bit(command >>4)*

*LCD\_Enable();*

*LCD\_Send4Bit(command);*

*LCD\_Enable();*

*}*

*void LCD\_Clear(){*

*LCD\_SendCommand(0x01);*

*delay\_us(10);*

*}*

*// Ham Khoi Tao LCD*

*void LCD\_Init(){*

*LCD\_Send4Bit(0x00);*

*delay\_ms(20);*

*LCD\_RS=0;*

*LCD\_RW=0;*

*LCD\_Send4Bit(0x03);*

*LCD\_Enable();*

*delay\_ms(5);*

*LCD\_Enable();*

*delay\_us(100);*

*LCD\_Enable();*

*LCD\_Send4Bit(0x02);*

*LCD\_Enable();*

*LCD\_SendCommand( 0x28 );*

*LCD\_SendCommand( 0x0c);*

*LCD\_SendCommand( 0x06 );*

*LCD\_SendCommand(0x01);*

*}*

*void LCD\_Gotoxy(unsigned char x, unsigned char y){*

*unsigned char address;*

*if(!y)address=(0x80+x);*

*else address=(0xc0+x);*

*delay\_us(1000);*

*LCD\_SendCommand(address);*

*delay\_us(50);*

*}*

*void LCD\_PutChar(unsigned char Data)*

*{*

*LCD\_RS=1;*

*LCD\_SendCommand(Data);*

*LCD\_RS=0 ;*

*}*

*void LCD\_Puts(char \*s)*

*{*

*while (\*s){*

*LCD\_PutChar(\*s);*

*s++;*

*}}*

*unsigned char code ma\_nut[16]=*

*{*

*'7','8','9',':',*

*'4','5','6','x',*

*'1','2','3','-',*

*'C','0','=','+'*

*};*

*void show\_button(unsigned char key){*

*LCD\_Puts(" Phim ");*

*LCD\_PutChar(ma\_nut[key-1]);*

*}*

*void main(){ //Chuong trinh chinh*

*unsigned char but;*

*LCD\_Init();*

*delay\_ms(200);*

*LCD\_Puts("Quet phim 4x4..");*

*delay\_ms(500);*

*LCD\_Clear();*

*LCD\_Gotoxy(0,0);*

*LCD\_Puts(" HO MAU VIET ");*

*while(1){*

*LCD\_Gotoxy(0,1);*

*but=get\_key();*

*if(but>0)show\_button(but);*

*else LCD\_Puts(" No Phim ");*

*delay\_ms(200);*

*}}*

**4.2.4. Đánh giá kết quả**



*Hình 4.3. Kết quả kiểm tra hệ thống*

**4.3. Thiết kế hệ thống hiện thị trên LED ma trận 8x8**

**4.3.1. Yêu cầu**

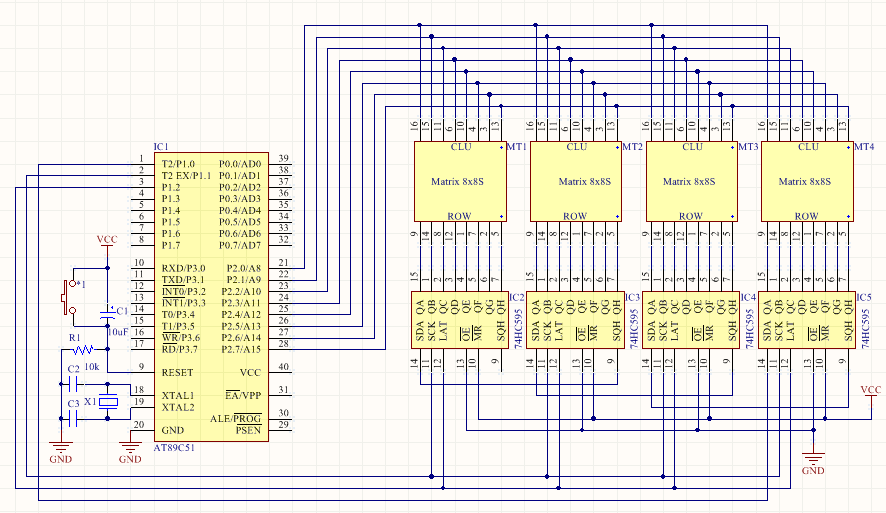
- Hệ thống hiển thị được dòng chữ “ KTYS-K11”

- Vi điều khiển được dùng trong hệ thống là họ 8051

- Hệ thống có phần hiện thị là 4 LED ma trận 8x8

- Hệ thống dùng IC giải mã là 74HC595

**2.6.2. Thiết kế sơ đồ mạch nguyên lý**

****

*Hình 4.4. Sơ đồ mạch nguyên lý*

**Chọn linh kiện cho hệ thống**

- Vi điều khiển chọn AT89C51

- LED ma trận chọn 8x8 màu đỏ

- Bộ giải mã LED ma trận chọn 74HC595

- Điện trở R1 chọn 10K

- Tụ C2, C3 tụ gốm có giá trị 33 uF

- Thạch anh chọn giá trị 12Mhz

**4.3.2. Xây dựng chương trình điều khiển**

#include<reg51.h>

#include<stdio.h>

#include<intrins.h>

sbit DATA = P1^0;

sbit sck = P1^1;

sbit strobe = P1^2;

void delay(unsigned int t)

{

unsigned int i;

for(i=0;i<t;i++);

}

int m,n,k;

unsigned char hang[104]={0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,

0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,

0x00,0x00,0xe7,0xc3,0x99,0x3c,0xff,0xff, //k

0xfc,0xfc,0x00,0x00,0xfc,0xfc,0xff,0xff, //t

0xfc,0xf9,0xf3,0x07,0x07,0xf3,0xf9,0xfc, //y 0xff,0xff,0x20,0x20,0x24,0x24,0x04,0x04, //s

0xff,0xff,0xef,0xef,0xef,0xef,0xff,0xff, //-

0x00,0x00,0xe7,0xdb,0xbd,0x7e,0xff,0xff, //k

0xff,0x33,0x39,0x00,0x00,0x3f,0x3f,0xff,//1 0xff,0x33,0x39,0x00,0x00,0x3f,0x3f,0xff, //1

0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff ,

0xff, 0xff,0xff,0xff,0xff,0xff,0xff,0xff ,

0xff, 0xff,0xff,0xff,0xff,0xff,0xff,0xff };

unsigned char cot[8]={0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80};

void convert(unsigned char data\_out)

{

unsigned char i,tg;

for(i=0;i<=7;i++)

{

tg = data\_out;

tg = tg&0x80;

if(tg == 0x80)

DATA = 1;

else

DATA = 0;

data\_out \*= 2;

sck = 1;

\_nop\_();

\_nop\_();

sck = 0;

}

}

void clear()

{

convert(0xff);

convert(0xff);

convert(0xff);

convert(0xff);

}

void main()

{

while(1)

{

for(m=0;m<=72;m++)

{

for(k=0;k<25;k++)

for(n=m;n<=m+7;n++)

{

convert(hang[n]);

convert(hang[n+8]);

convert(hang[n+16]);

convert(hang[n+24]);

P2 = cot[n-m];

strobe = 1;

\_nop\_();

\_nop\_();

strobe = 0;

delay(100);

clear();

}}}}

**4.3.3. Đánh giá kết quả**



*Hình 4.5. Sơ đồ mô phỏng*

**4.4. Thiết kế đồng hồ thời gian thực hiện thị thời gian**

**4.4.1. Yêu cầu thiết kế**

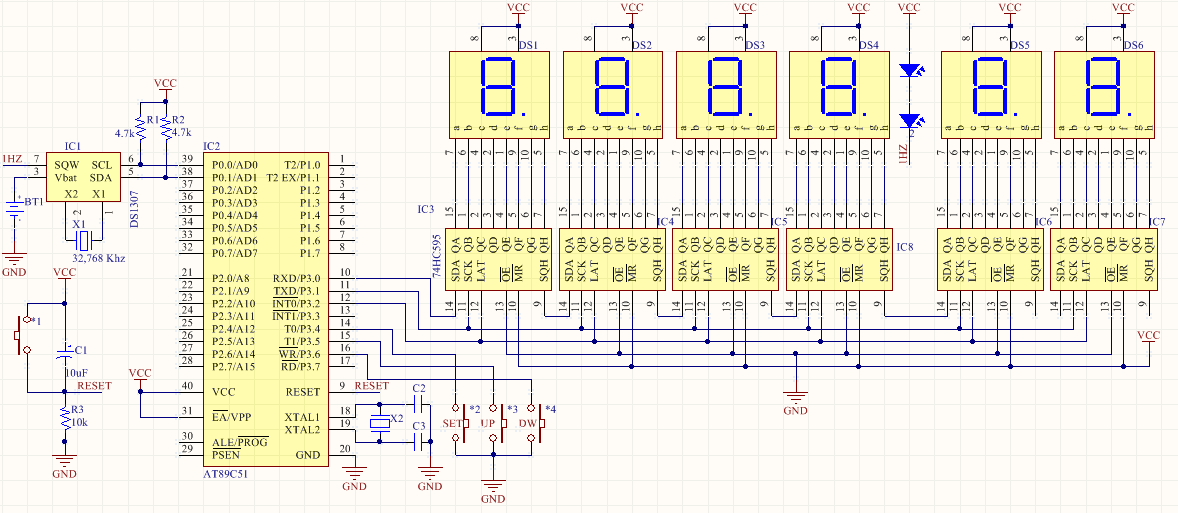
- Vi điều khiển dùng trong hệ thống là AT89C51

- Hệ thống có nút để cài đặt thời gian khi cần thiết

- Thời gian hiện thị trên LED 7 đoạn

- Hệ thống có pin lưu thời gian khi mất điện

**4.4.2. Thiết kế sơ đồ mạch nguyên lý**



*Hình 4.6. Sơ đồ mạch nguyên lý*

**Chọn linh kiện cho hệ thống**

- Vi điều khiển chọn chip AT89C51

- IC thời gian thực chọn DS1307

- Hiển thị thời gian trên LED 7 đoạn

- Giải mã 7 thanh dùng ICHC595

- Điện trở R3 có giá trị 10K, R1, R2 có giá trị 4.7K

- Tụ C1 có giá trị 10uF, tụ C2, C3 tụ gốm không phân cực có giá trị 33uF.

**4.4.3. Xây dựng chương trình điều khiển**

#include<reg52.h>

sbit DATA1 = P3^0;

sbit SCK1 = P3^1;

sbit LACH1 = P3^2;

sbit set=P3^4;

sbit up=P3^5;

sbit down=P3^6;

unsigned char hour, minute, second,c;

unsigned char ma[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};

unsigned char kt=0,a=0,i,k,x=0;

bit set0=1,set1=1,up1=1,up0=1,down0=1,down1=1;

sbit SCL = P1^0;

sbit SDA = P1^1;

void delay(){

unsigned char i;

for (i = 0; i < 4; i++){};

}

void SCL\_high(){

SCL = 1;

delay();

}

void SCL\_low(){

SCL = 0;

delay();

}

void I2C\_Start(){

SDA = 1;

SCL = 1;

SDA = 0;

delay();

SCL = 0;

SDA = 1;

}

void I2C\_Stop(){

SDA = 0;

SCL\_high();

SDA = 1;

}

bit I2C\_Write(unsigned char dat){

unsigned char i;

bit outbit;

for (i = 1; i <= 8; i++){

outbit=dat&0x80;

SDA = outbit;

dat = dat << 1;

SCL\_high();

SCL\_low();

}

SDA = 1;

SCL\_high();

outbit = SDA;

SCL\_low();

return(outbit);

}

unsigned char I2C\_Read(bit ack){

unsigned char i, dat;

bit inbit;

dat = 0;

for(i=1;i<=8;i++) {

SCL\_high();

inbit = SDA;

dat = dat << 1;

dat = dat | inbit;

SCL\_low();

}

if (ack) SDA = 0;

else SDA = 1;

SCL\_high();

SCL = 0;

SDA = 1;

delay();

return(dat);

}

void rtc\_write(unsigned char add, unsigned char dat){

I2C\_Start();

I2C\_Write(0xd0);

I2C\_Write(add);

I2C\_Write(((dat/10)<<4)|(dat%10));

I2C\_Stop();

}

unsigned char rtc\_read(unsigned char add){

unsigned char dat;

I2C\_Start();

I2C\_Write(0xd0);

I2C\_Write(add);

I2C\_Start();

I2C\_Write(0xd1);

dat = I2C\_Read(0);

I2C\_Stop();

dat = (dat & 0x0f) + (dat>>4)\*10;

return (dat);

}

void writeds(){

rtc\_write(0x00, second);

rtc\_write(0x01, minute);

rtc\_write(0x02, hour);

}

void readds()

{

if(kt==0)

{

if(a==1)

{writeds();a=0;

}

second = rtc\_read(0x00);

minute = rtc\_read(0x01);

hour = rtc\_read(0x02);

}}

void hienthi(unsigned char x)

{

unsigned int i,temp;

for(i=0;i<8;i++)

{

temp=x;

temp=temp&0x80;

if(temp==0x80)

DATA1=1;

else

DATA1=0;

x\*=2;

SCK1=0;

SCK1=1;

}}

void quetled()

{

if(kt==0)

{

hienthi(ma[second%10]);

hienthi(ma[second/10]);

hienthi(ma[minute%10]);

hienthi(ma[minute/10]);

hienthi(ma[hour%10]);

hienthi(ma[hour/10]);

LACH1=0;

LACH1=1;

}

if(kt==1)

{

hienthi(0xff);

hienthi(0xff);

hienthi(0xff);

hienthi(0xff);

hienthi(ma[hour%10]);

hienthi(ma[hour/10]);

LACH1=0;

LACH1=1;

}

if(kt==2)

{

hienthi(0xff);

hienthi(0xff);

hienthi(ma[minute%10]);

hienthi(ma[minute/10]);

hienthi(0xff);

hienthi(0xff);

LACH1=0;

LACH1=1;

}

if(kt==3)

{

hienthi(ma[second%10]);

hienthi(ma[second/10]);

hienthi(0xff);

hienthi(0xff);

hienthi(0xff);

hienthi(0xff);

LACH1=0;

LACH1=1;

}}

void ktphim()

{

set0=set1;set1=set;

if((set0==1)&&(set1==0))

{

kt++;

if(kt>3)kt=0;

}

switch(kt)

{

case 0:break;

case 1:

{

up0=up1;up1=up; a=1;

if((up0==1)&&(up1==0))

{

hour++;

if(hour>23)hour=0;

}

down0=down1;down1=down;

if((down0==1)&&(down1==0))

{

hour--;

if(hour==-1)

hour=23;

}

break;

}

case 2:

{

up0=up1;up1=up; a=1;

if((up0==1)&&(up1==0))

{

minute++;

if(minute>59)

minute=0;

}

down0=down1;down1=down;

if((down0==1)&&(down1==0))

{

minute--;

if(minute==-1)

minute=59;

}

break;

}

case 3:

{

up0=up1;up1=up;a=1;

if((up0==1)&&(up1==0))

{

second++;

if(second>59)

second++;

}

down0=down1;down1=down;

if((down0==1)&&(down1==0))

{

second=0;

}

break;

} }}

void main()

{

I2C\_Start();

I2C\_Write(0xD0);

I2C\_Write(0x07);

I2C\_Write(0x10);

I2C\_Stop();

while(1)

{

ktphim();

readds();

quetled();

}**}**

**4.4.4. Đánh giá kết quả**

****

*Hình 4.7. Sơ đồ mạch mô phỏng*

**4.5. Thiết kế hệ thống điều khiển và xác định tốc độ động cơ**

**4.5.1. Yêu cầu hệ thống**

- Hệ thống dùng vi điều khiển họ 8051

- Hệ thống xác định được số vòng quay/phút

- Hệ thống có các nút chức năng điều khiển động cơ quay thuận, quay nghịch, tăng tốc, giảm tốc, và tắt động cơ.

**4.5.2. Thiết kế sơ đồ mạch nguyên lý**

****

*Hình 4.8. Sơ đồ mạch nguyên lý*

**Chọn linh kiện dùng trong hệ thống**

- Vi điều khiển dùng chip AT89C51

- Điều khiển động cơ dùng mạch cầu L298

- Hiện thị dùng LCD có kích thức 16x2

- Điện trở treo có giá trị 10K

**5.5.3. Xây dựng chương trình điều khiển**

#include "REGX52.H"

//-----Khoi tao bien va cac gia tri gan-----//

sbit RS\_LCD = P2^2;

sbit RW\_LCD = P2^3;

sbit E\_LCD = P2^4;

sbit l0 = P2^0;

sbit l1 = P2^1;

unsigned char a[] = { 200, 300, 400, 500, 600, 700, 800, 900, 1000} ;

int count1=0,count2=0;

int dem=0;

char x1,x0;

unsigned char xung;

unsigned char phantramxung;

unsigned char chay=0;//

//----Ham tre -----//

void delay\_short()

{

unsigned int x;

for(x=0;x<3;x++);

}

void delay(unsigned int time)

{

unsigned int q;

unsigned char e;

for (q=1; q<=time; q++)

{

for (e=1; e<=150; e++)

{;}

}}

void timer()interrupt 1{

TR0=0;

TR1=0;

dem++;

TH0=0x3C;

TL0=0xB0;

if(dem==12){

count1=TL1;

count2=TH1;

TL1=TH1=0;

dem=0;

}

TR0=1;

TR1=1;

}

void timer2(void) interrupt 5{ //Ngat timer 2

TR2=0;

TF2=0; // Dung chay timer 2

chay++;

if(chay<=phantramxung) {

l1=x1;

l0=x0; // Neu bien dem < phan tram xung thi dua gia tri 1 ra chan

}

else {

l1 =0;

l0=0;

}

if(chay==10) chay=0;

TR2=1; // Cho chay timer

}

char tangtoc(void){

if(phantramxung>0){

if (P1\_0==0){

while(P1\_0==0){;}

phantramxung++;

if (phantramxung>9)

}

}

return phantramxung;

}

//ham giam tang toc

char giam(void){

if (P1\_1==0){

while(P1\_1==0){;}

phantramxung--;

if (phantramxung<=1) phantramxung=1;

}

return phantramxung;

}

char quaythuan(void){//ham khoi dong dong co

if(P1\_2==0){

P1\_5=0;

P1\_6=1;

x1=1;

x0=0;

phantramxung=6;

}

return phantramxung;

}

char quaynguoc(void){

if(P1\_4==0){

P1\_5=1;

P1\_6=0;

x1=0;

x0=1;

phantramxung=6;

}

return phantramxung;

}

char dung(void){//ham khoi dong dong co

if(P1\_3==0){

P1\_5=1;

P1\_6=1;

phantramxung=0;

}

return phantramxung;

}

void kt\_ban()

{

unsigned char x;

P0 = 0xff;

RS\_LCD = 0;

RW\_LCD = 1;

do

{

E\_LCD = 1;

E\_LCD = 0;

delay(3);

x=P0;

x=x&0x80;

}

while(x!=0x80);

}

void ghi\_lenh(unsigned char lenh)

{

kt\_ban();

P0 = lenh;

RS\_LCD = 0; // Chon thanh ghi lenh

RW\_LCD = 0; // Chon che do ghi du lieu tu 8051 vao LCD

E\_LCD = 1;

delay\_short();

E\_LCD = 0;

}

void ghi\_kytu(unsigned char kytu)

{

kt\_ban();

P0 = kytu;

RS\_LCD = 1; // Chon che do doc du lieu tu 8051

RW\_LCD = 0;

E\_LCD =1;

delay\_short();

E\_LCD = 0;

}

void ghi\_nhanh ( unsigned char kytu)

{

kt\_ban();

// delay(150);

P0 = kytu;

RS\_LCD = 1;

RW\_LCD = 0;

E\_LCD = 1;

delay\_short();

E\_LCD = 0;

}

void ghi\_chuoi(char \*str)

{

while(\*str)

{

delay(50);

ghi\_kytu(\*str);

str++;

}

}

void ghi\_chuoi1(char \*str)

{

while(\*str)

{

delay(50);

ghi\_kytu(\*str);

str++;

ghi\_lenh(0x07);

}

}

void ghi\_nhanh\_chuoi(char \*str)

{

while(\*str){

ghi\_nhanh(\*str);

str++;

}

}

void hienthiso(long n, long m){

int a, b, c, d, e;

char f=0;

n=m\*255+n;

e=n/10000;

n=n\*4.128;

n=n-10000\*e;

a=n/1000;n=n-1000\*a;

b=n/100;n=n-b\*100;

c=n/10; n=n-c\*10;

d=n;

ghi\_lenh(0xc0);

ghi\_lenh(0x0c);

ghi\_nhanh\_chuoi(" ");

ghi\_nhanh(a+48); // Doi ra hang nghin, ma ascii

ghi\_nhanh(b+48); // Doi ra hang tram, ma ascii

ghi\_nhanh(c+48); // Doi ra hang chuc, ma ascii

ghi\_nhanh(d+48); // Doi ra hang don vi, ma ascii

ghi\_nhanh\_chuoi(" V/Phut ");

delay(500);

}

void setting\_LCD()

{

ghi\_lenh(0x38);

ghi\_lenh(0xC0); // Xoa man hinh

ghi\_lenh(0x01);

ghi\_lenh(0x06); // Co dich hien thi

}

//ham gioi thieu

void gioi\_thieu()

{

ghi\_lenh(0x01);

ghi\_lenh(0x0c);

ghi\_chuoi("THIET KE ");

ghi\_lenh(0xc0);

ghi\_lenh(0x0c);

ghi\_chuoi("HE THONG ");

ghi\_lenh(0x01);

ghi\_lenh(0x0c);

ghi\_nhanh\_chuoi("DIEU KHIEN TOC ");

delay(200);

ghi\_lenh(0xc0);

ghi\_lenh(0x0c);

ghi\_chuoi("DO DONG CO DC ");

delay(1000);

ghi\_lenh(0x01);

ghi\_lenh(0x0c);

ghi\_chuoi(" START!! ");

delay(500);

}

void main()

{

TMOD=0x51;

IE=0xAA;

TH0=0x3C;

TL0=0xB0;

TH2=0xFF;

TL2=0x9C;

T2MOD=0xc9;

T2CON=0x04;

RCAP2H=0xFF;

RCAP2L=0x9C;

TR2=1;

TR0=1;

TR1=1;

setting\_LCD();

gioi\_thieu();

ghi\_lenh(0x01);

ghi\_lenh(0x0c);

ghi\_chuoi("TOC DO HIEN TAI:");

while(1)

{

hienthiso(count1,count2);

quaythuan();

tangtoc();

giam();

dung();

quaynguoc();

}}

**5.5.4. Đánh giá kết quả**

****

*Hình 4.9. Kết quả mô phỏng hệ thống*