BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG……..

Luận văn

Xây dựng hệ thống điều khiển thời gian thực hiển thị trên Led Matrix P10

LỜI MỞ ĐẦU

Ngày nay nhân loại đang trải qua những sự phát triển về mọi mặt. Trong đó điện tử tự - động hóa đóng một vai trò không nhỏ. Điện tử góp phần vào quá trình tự động hóa mọi thứ giúp con người hiện đại hóa cuộc sống.

Vận dụng những kiến thức đã học được trong quá trình học tập ở trường em thực hiện đồ án tốt nghiệp này. Đồ án này chủ yếu được áp dụng chủ yếu dựa vào vi điều khiển. Mà thực tế là IC8051, nhằm mục đích giúp em hiểu tường tận hơn về vi điều khiển, cách đọc, biết và nhận biết về các chân IC mà em đã được học từ giảng viên trong trường, tìm hiểu và nghiên cứu qua sách cũng như cách thức vận dụng nó trong thực tế.

Trong thực tế. các ứng dụng của vi điều khiển rất đa dạng và phong phú. Từ những ứng dụng đơn giản chỉ có vài thiết bị ngoại vi cho đến những hệ thống điều khiển phức tạp. Tuy nhiên do phạm vi trình độ của em còn hạn chế, nên việc nhiên cứu và tìm hiểu về vi điều khiển còn nhiều điều chưa biết. Trong bài viết của em, em xin giới thiệu ứng dụng họ IC8051 để hiện thị bộ đếm thời gian thực sử dụng DS1307 quét hiển thị trên LED matrix.

Tuy nhiên trong quá trình viết do trình độ hiểu biết của em còn nhiều hạn chế, nên còn xảy ra nhiều sai sót mong thầy và các bạn góp ý bổ sung để em được hiểu biết hơn trong quá trình học tập tiếp theo.

Em xin chân thành cảm ơn!

LỜI CẢM ƠN

Em xin chân thành cám ơn thầy Ths.Nguyễn Trọng Thắng đã tận tình hướng dẫn và tạo điều kiện thuận lợi cho em có thể hoàn thành tốt đề tài này.

Em xin chân thành cám ơn các thầy trong khoa điện tử cùng các bạn sinh viên trong lớp đã đóng góp ý kiến và kinh nghiệm trong quá trình thực hiện đề tài này.

Sinh viên thực hiện

Phạm Minh Tuấn

CHƯƠNG 1

HỆ THỐNG THỜI GIAN THỰC

1.1 Hệ thống thời gian thực

1.1.1 Giới thiệu về hệ thống thời gian thực

Trong những năm gần đây, các hệ thống điều khiển theo thời gian thực là một trong những lĩnh vực thu hút nhiều sự chú ý trong giới khoa học nghiên cứu về khoa học máy tính. Trong đó, vấn đề điều hành thời gian thực và vấn đề lập lịch là đặc biệt quan trọng. Một số ứng dụng quan trọng của hệ thống thời gian thực (RTC) đã và đang được ứng dụng rộng rãi hiện nay là các dây chuyền sản xuất tự động, rô bốt, điều khiển không lưu, điều khiển các

thí nghiệm tự động, truyền thông, điều khiển trong quân sự… Thế hệ ứng dụng tiếp theo của hệ thống này sẽ là điều khiển rô bốt có hoạt động giống con người, hệ thống kiểm soát thông minh trong các nhà máy công nghiệp, điều khiển các trạm không gian, thăm dò đáy đại dương…

1.1.2 Khái niệm hệ thống thời gian thực:

Một số hệ thống thời gian thực (RTS – Realtime Systems) có thể được hiểu như là một mô hình xử lý mà tính đúng đắn của hệ thống không chỉ phụ thuộc vào kết quả tính toán logic mà còn phụ thuộc vào thời gian mà kết quả này phát sinh ra.

Hệ thống thời gian thực được thiết kế nhằm cho phép trả lời (Response) lại các yếu tố kích thích phát sinh từ các thiết bị phần cứng trong một ràng buộc thời gian xác định. Ở đây ta có thể hiểu thế nào là một RTS bằng cách hiểu thế nào là một tiến trình, một công việc thời gian thực. Nhìn chung, trong những RTS chỉ có một số công việc được gọi là công việc thời gian thực, các công việc này có mức độ khẩn cấp riêng phải hoàn tất, ví dụ một tiến trình đang cố gắng điều khiển hoặc giám sát một sự kiện đang xảy ra trong thế giới thực. Bởi vì mỗi sự kiện xuất hiện trong thế giới thực nên tiến trình giám sát sự kiện này phải xử lý theo kịp với những thay đổi của sự kiện này. Sự thay đổi của sự kiện trong thế giới thực xảy ra rất nhanh, mỗi tiến trình giám sát sự kiện này phải thực hiện việc xử lý trong một khoảng thời gian ràng buộc gọi là deadline, khoảng thời gian ràng buộc này được xác định bởi thời gian bắt đầu và thời gian hoàn tất công việc. Trong thực tế, các yếu tố kích thích xảy ra trong thời gian ngắn vào khoảng vài mili giây, thời gian mà hệ thống trả lời lại yếu tố kích thích đó tốt nhất vào khoảng dưới một giây, thương vào khoảng vài chục mili giây, khoảng thời gian này bao gồm thời gian tiếp nhận kich thích, xử lý thông tin và trả lời lại kích thích. Một số yếu tố khác cần quan tâm trong RTS là những công việc thời gian thực này có tuần hoàn hay

không? Công việc tuần hoàn thi ràng buộc thời gian ấn định theo từng chu kỳ xác định. Công việc không tuần hoàn xảy ra với ràng buộc thời gian vào lúc bắt đầu và lúc kết thúc công việc, ràng buộc này chỉ được xác định vào lúc bắt đầu công việc. Các biến cố kích hoạt công việc không tuần hoàn thường dựa trên kỹ thật xử lý ngắt của hệ thống phần cứng.

Về mặt cấu tạo, RTS thường được cấu thành từ các thành tố chính sau:

- Đồng hồ thời gian thực: Cung cấp thông tin thời gian thực.
- Bộ điều khiển ngắt: Quản lý các biến cố không theo chu kỳ.
- Bộ định biểu: Quản lý các quá trình thực hiện.
- Bộ quản lý tài nguyên: Cung cấp các tài nguyên máy tính.
- Bộ điều khiển thực hiện: Khởi động các tiến trình.

  Các yếu tố trên có thể được phân định là thành phần cứng hay mềm tùy thuộc vào hệ thống và ý nghĩa sử dụng. Thông thường, các RTS được kết hợp vào phần cứng có khả năng tốt hơn so với hệ thống phần mềm có chức năng tương ứng và tránh được chi phí quá đắt cho việc tối ưu hóa phần mềm. Ngày nay, chi phí phần cứng ngày càng rẻ , chọn lựa ưu tiên phần cứng là một xu hướng chung.

1.1.3 Các loại hệ thống thời gian thực:

Các RTS thường được phân thành hai loại sau Soft reatime system và Hard realtime system: Đối với Soft realtime system, thời gian trả lời của hệ thống cho yếu tố kích thích là quan trọng, tuy nhiên trong trường hợp ràng buộc này bị vi phạm, tức là thời gian trả lời của hệ thống vượt quá giới hạn trễ cho phép, hệ thống vẫn cho phép tiếp tục hoạt động bình thường, không quan tâm đến các tác hại do sự vi phạm này gây ra (thường thì những tác hại này không đáng kể).

Ngược lại với Soft realtime system là Hard realtime system, trường hợp này người ta quan tâm khắc khe đến các hậu quả do sự vi phạm giới hạn thời gian để cho phép bởi vì những hậu quả này có thể là rất tồi tệ, thiệt hại về vật chất, có thể gây ra những ảnh hưởng xấu đến với đời sống con người. Một số ví dụ cho loại này là hệ thống điều khiển không lưu, một phân phối đường bay, thời gian cất cánh, hạ cánh không hợp lý, không đúng lúc có thể gây ra tai nạn máy bay mà hậu quả của nó khó có thể lường trước được.

Trong thực tế thì có nhiều loại RTS bao gồm cả hai loại soft và hard. Trong cả hai loại này, máy tính thường can thiệp trực tiếp hoặc gián tiếp đến các thiết bị vật lý để kiểm soát cũng như điều khiển sự hoạt động của thiết bị này. Đứng trên góc độ này, người ta thường chia RTS ra làm hai loại sau:

(1) Embedded system: Bộ vi xử lý điều khiển là một phần trong toàn bộ thiết bị, nó được sản xuất trọn gói từ yếu tố cứng đến yêu tố mềm từ nhà máy, người sử dụng không biết về chi tiết của nó và chỉ sử dụng thông qua các nút điều khiển, các bảng số. Với hệ thống này, ta sẽ không thấy được những thiết bị như trong máy tính bình thường như bàn phím, màn hình… mà thay vào đó là các nút điều khiển, các bảng số, đèn tín hiệu hay các màn hình chuyên dụng đặc trưng cho từng hệ thống. Máy giặt là một ví dụ. Người sử dụng chỉ việc bấm nút chọn chương trình giặt, xem kết quả qua hệ thống đèn báo hiệu… Bộ vi xử lý trong Embedded system này đã được lập trình trước và gắn chặt vào ngay từ khi sản xuất và không thể lập trình lại. Những chương trình này chạy độc lập, không có sự giao tiếp với hệ điều hành (HĐH) cũng như không cho phép người sử dụng can thiệp vào.

(2) Loại hai này là bao gồm những hệ thống có sự can thiệp của máy tính thông thường. Thông qua máy tính ta hoàn toàn có thể kiểm soát cũng như điều khiển mọi hoạt động của thiết bị phần cứng của hệ thống này.

Những chương trình điều khiển này có rất nhiều loại, phục vụ cho nhiều mục đích khác nau và có thể được viết lại cho phù hợp với yêu cầu thực tế. Hiển nhiên thì loại hệ thống này hoạt động được phải cần một HĐH điều khiển máy tính. HĐH này phải có khả năng nhận biết được thiết bị phần cứng, có khả năng hoàn tất công việc trong giới hạn thời gian nghiêm ngặt. HĐH này phải là HĐH hỗ trợ xử lú thời gian thực – Realtime operating system (RTOS).

1.1.4 Hệ điều hành cho hệ thống thời gian thực

Trong lĩnh vực công nghệ thông tin, người ta nói về hệ thống thông tin thời gian thực khi hệ thống đó điều khiển một vật thể vật lý với một tốc độ phù hợp với sự tiến triển của tiến trình chủ. Một ví dụ dễ hiểu (hệ thống thông tin điều khiển màn hình hiển thị giờ chính xác của các tàu điện ngầm sẽ đến và đi tại một gare nhất định). Hệ thống thông tin thời gian thực khác với những hệ thống thông tin khác bởi sự gò bó về thời gian, do đó, việc tuân thủ các nguyên tắc cũng quan trọng như độ chính xác của kết quả, nói một cách khác, hệ thống không chỉ đơn giản là đưa ra kết quả chính xác mà nó còn phải thực hiện một xử lý trong một thời gian rất ngắn. Hệ thống thông tin thời gian thực ngày nay được ứng dụng trong rất nhiều lĩnh vực như: trong ngành công nghiệp sản xuất, kiểm soát tiến trình (trong nhà máy, hay trong viện hạt nhân, trong hệ thống hàng không, thông qua các hệ thống dẫn đường tích hợp trên máy bay và vệ tinh). Sự phát triển của hệ thống thông tin thời gian thực yêu cầu mỗi phần tử của hệ thống phải ở thời gian thực, và một hệ thống được thiết kế theo cách như vậy được gọi là hệ điều hành thời gian thực.

Để đảm bảo tuân thủ đúng sự giới hạn về thời gian, hệ thống cần phải:

- Có những dịch vụ khác nhau và những thuật toán có thể xử lý trong khoảng thời gian hạn chế. Một hệ điều hành thời gian thực phải được thiết

kế làm sao cho các dịch vụ của nó có thể truy cập vào phần cứng với một khoảng thời gian ngắn nhất.

- Có những kết hợp thích hợp để đảm bảo cho những xử lý của mọi thành phần không vượt quá thời gian cho phép.

Một số ví dụ cho hệ điều hành thời gian thực:

- Adeos
- ART Linux
- ChorusOS
- eCos
- ELinOS
- FreeRTOS
- iRmx
- ITRON
- Linux
- LynxOS
- MicroC/OS-II
- Nucleus
- OS-9
- OSE
- OSEK/VDX
- pSOS
- PikeOS
- QNX
- RedHawk
- RSX-11
- VxWorks
- Windows CE
- Xenomai

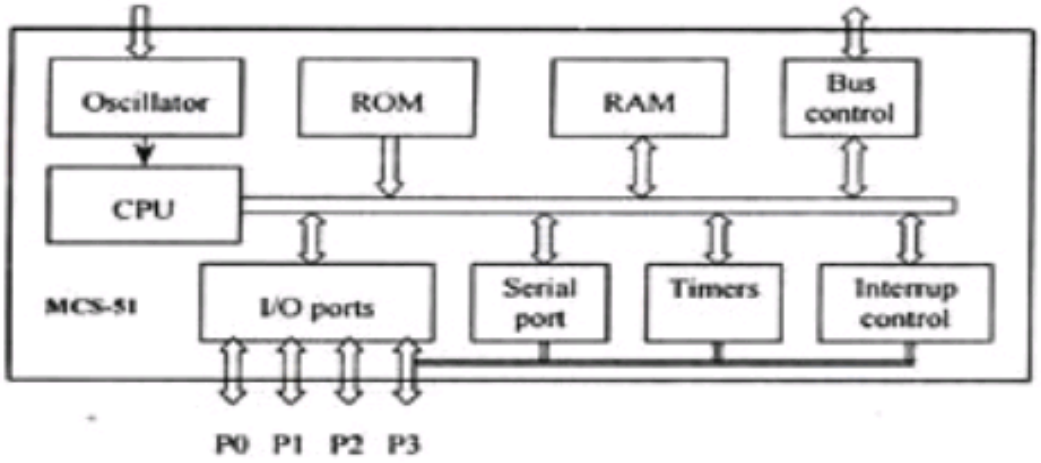
CHƯƠNG 2

GIỚI THIỆU CÁC LINH KIỆN DÙNG TRONG HỆ THỐNG

2.1 VI ĐIỀU KHIỂN

2.1.1Giới thiệu họ vi điều khiển

Bộ điều khiển đơn chip 8051 được công ty INTEL chế tạo vào năm 1980 là sản phẩm đầu tiên của bộ vi điều khiển MCS-51. Ngày nay, họ MCS-51 đã có trên 250 biến thể khác nhau và được hầu hết các công ty bán dẫn hàng đầu trên thế giới chế tạo, với số lượng tiêu thụ trên 4 tỷ mỗi năm. Họ MCS-51 có khả năng ứng dụng rất rộng rãi, chũng có mặt trong rất nhiều sản phẩm dân dụng như máy giặt, máy điều hòa nhiệt độ, lò vi sóng, nồi cơm điện…, các thiết bị điện tử y tế và viễn thong, các thiết bị đo lường và điều khiển sử dụng trong công nghiệp,v.v… Dưới đây là cấu trúc cơ bản của các bộ vi điều khiển MCS-51:



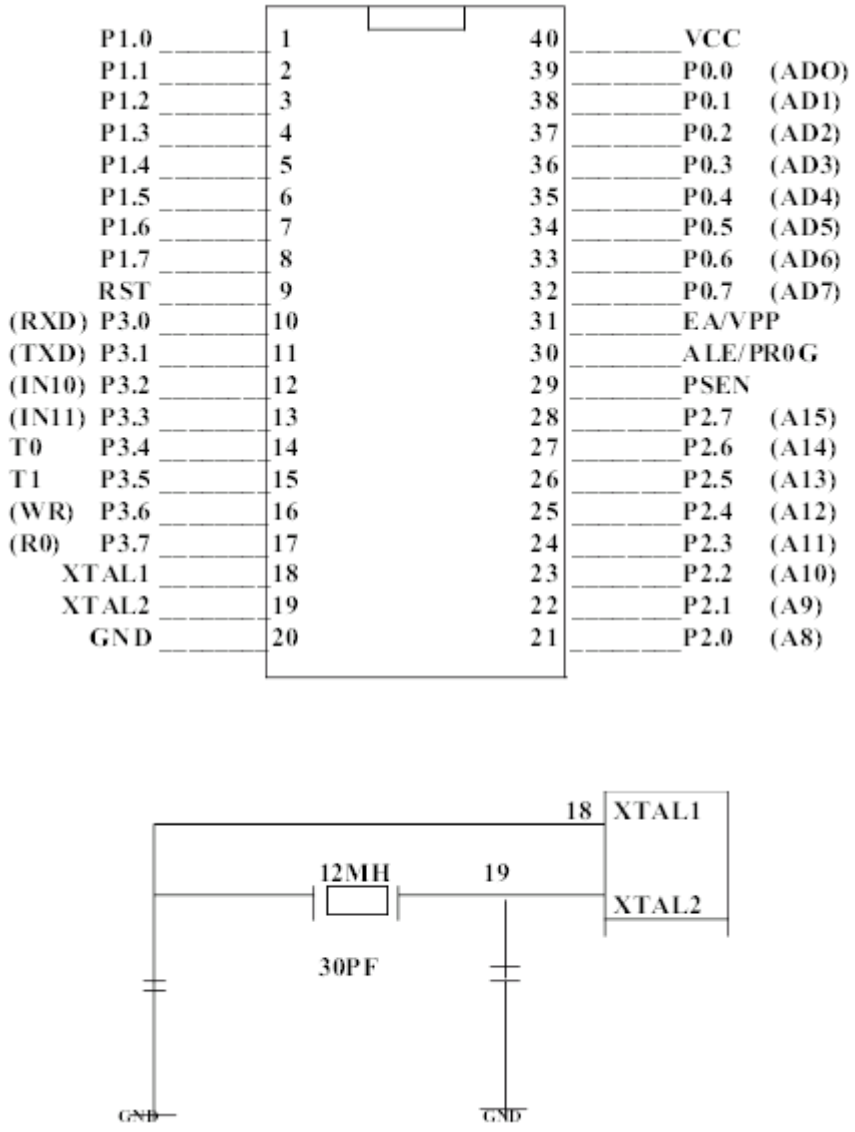
Hình 2.1: Cấu trúc cơ bản của MCS-51

Mỗi vi mạch MCS-51 bao gồm trong nó bộ xử lý trung tâm (CPU), bộ nhớ chỉ đọc (ROM), bộ nhớ đọc ghi (RAM), các cổng vào ra song song 8 bit (I/O Port), cổng vào ra nối tiếp (Serial Port), các bộ điếm và định thời (Timer), khối điểu khiển ngắt (Interrupt control), khối điều khiển bus (Bus

control) và mạch tạo xung nhịp (Oscillator). Giao tiếp giữa CPU và các khối bên trong của MCS-51 được thực hiện qua các bus nội bộ gồm bus dư liệu 8 bit, bus địa chỉ và các tín hiệu điều khiển khác. Cấu trúc trên cho phép coi MCS-51 như là một máy tính đơn chip 8 bit.

2.1.2 Sơ đồ và chức năng các chân

Sơ đồ các chân ra trên vỏ của các vi mạch MCS-51 như hình đưới đây



Hình 2.2: sơ đồ chân của họ MCS-51

- Các chân XTAL1 (19) và XTAL2 (18) để mắc thạch anh cho mạch tạo xung nhịp của MCS-51.

- Chân RESET (9) là tín hiệu vào tích cực mức cao để thiết lập lại trạng thái ban đầu cho MCS-51.

- Chân /EA (31) là tín hiệu vào, khi nối /EA với đất thi MCS-51 làm việc với các bộ nhớ ROM, RAM bên ngoài.

- Chân ALE (30) là tín hiệu ra dùng để chốt 8 bit địa chỉ thấp (AO A7) khi sử dụng bộ nhớ ngoài.

- Chân /PSEN (29) là tín hiệu ra tích cực mức thấp dùng để đọc mã lệnh từ bộ nhớ chương trình bên ngoài khi /EA được nối với đất, khi /EA được nối với +5v thì /PSEN luôn không tích cực ở mức cao.

- Các chân cổng 0: P0.7 P0.0 (32 39) được dùng làm cổng vào ra khi /EA được nối với +5v. Khi /EA nối đất thì cổng 0 được sử dụng làm bus địa chỉ và số liệu cho bộ nhớ ngoài. Khi đó, ở nửa đầu của chu kỳ lệnh truy nhập bộ nhớ ngoài, MCS-51 đã ra cổng 0 8 bit địa chỉ thấp (A0 A7), sau đó cổng 0 trở thành bus số liệu 8 bit, do đó phải dùng ALE để chốt 8 bit địa chỉ thấp vào thanh chốt địa chỉ phần thấp.

- Các chân cổng 2: P2.0 P2.7 (21 28) được dùng làm cổng vào ra khi /EA được nối với +5v. Khi /EA được nối đất thì cổng 2 được sử dụng để đưa ra 8 bit địa chỉ cao (A8 A15) cho bộ nhớ ngoài.

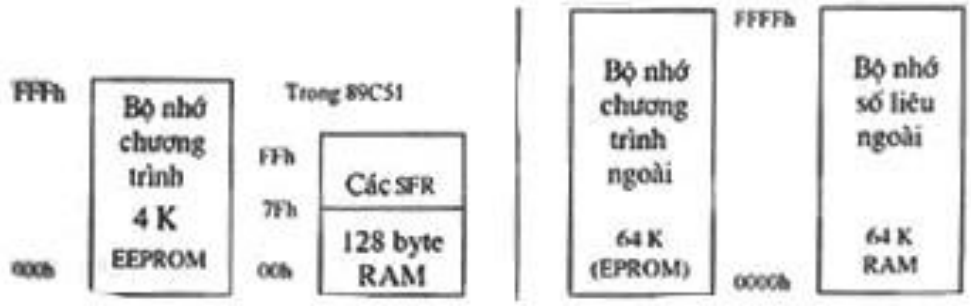
- Các chân cổng 3: P3.0 P3.7 (10 17) có thể được dùng làm cổng vào ra hoặc dùng cho chức năng khác như sau: P3.0 (RxD) có thể được dùng để nhận số liệu nối tiếp P3.1 (TxD) có thể được dùng để phát số liệu nối tiếp P3.2 (INTO) có thể được dùng để nhận ngắt ngoài 0; P3.3 (INT1) có thể được dùng để nhận ngắt ngoài 1; P3.4 (T0) có thể được

dùng để nhận xung clock Timer 0; P3.5 (T1) có thể được dùng để nhận xung clock cho Timer 1; P3.6 (/WR) khi /EA nối đất thì nó được dùng để đưa ra tín hiệu điều khiển đọc RAM ngoài.

- Các chân cổng 1: P1.0 P1.7 (1 8) đối với nhóm 8051 chỉ được sử dụng làm cổng vào ra. Đối với nhóm 8052 thi chân P1.0 (1) có thể được dùng để nhận xung clock T2 cho Timer 2, còn chân P1.1 (2) có thể được dùng làm đầu vào nạp lại cho T2EX cho Timer 2.
- Chân GND (20) là để nối đất, còn chân Vcc (40) là để cấp nguồn cho vi mạch MCS-51
- Tất cả 32 chân của 4 cổng P0 P3 đều có thể dùng để làm các cổng vào ra số liệu song song 8bit hoặc dùng làm các tín hiệu vào ra độc lập nhau.

2.1.3 Tổ chức bộ nhớ

Họ MCS-51 có không gian nhớ riêng cho chương trình và số liệu ở cả bên trong và bên ngoài. Tổ chức bộ nhớ của 89S52 như trên hình sau:



Hình 2.3: Sơ đồ tổ chức bộ nhớ

Khi /EA được nối với đất +5v thì bộ nhớ ngoài không được dung, MCS-51 chỉ truy nhập EPROM trong để đọc mã chương trình và cất số liệu vào RAM trong. Khi /EA được nối đất thi bộ nhớ chương trình ROM trong không được sử dụng, MCS-51 đọc mã chương trình từ bộ nhớ chương trình ngoài bằng tín hiệu /PSEN, còn bộ nhớ số liệu ngoài được truy nhập bằng các

tín hiệu /WR và /RD, do có bộ nhớ chương trình và bộ nhớ ngoài có thể dùng chung bus địa chỉ A0 A15.

Bộ nhớ số liệu trong của họ MCS-51 có địa chỉ từ 00h đến FFh, trong đó nhóm 8052 có đủ 256 byte RAM, nhóm 8051 chỉ có 128 byte RAM ở các địa chỉ thấp từ 00h đến 7fh, vùng địa chỉ cao từ 80h đến FFh được dành cho các thanh ghi chức năng đặc biệt SFR. Tổ chức vùng 128 byte thấp bộ nhớ số liệu RAM trong của họ MCS-51 như trên hình vẽ, nó được chia thành 3 miền.

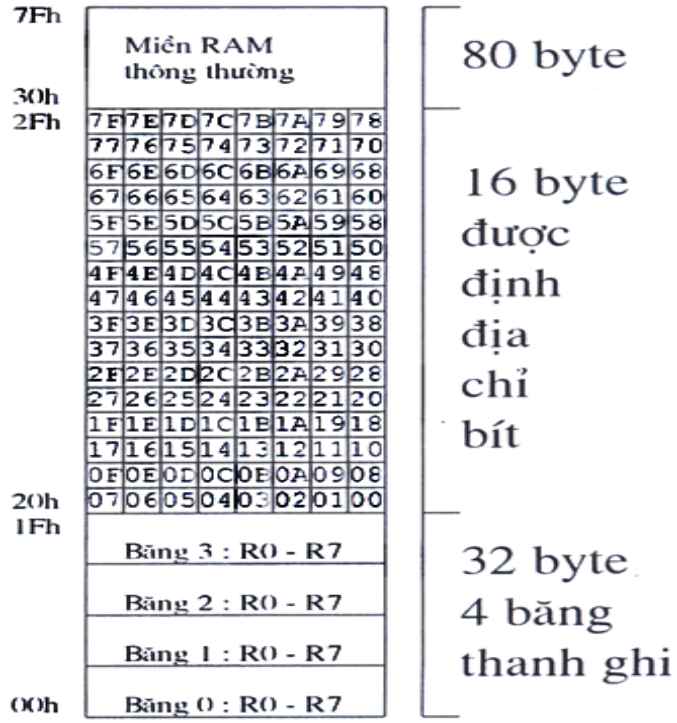
- Miền các băng thanh ghi chiếm địa chỉ từ 00h đến 1fh có 32 byte chia thành 4 băng, mỗi băng có 8 thanh ghi được đánh số từ R0 đến R7.

Tại mỗi thời điểm chỉ có một băng thanh ghi có thể truy nhập và được gọi là băng tích cực. Để chọn băng tích cực cần nạp giá trị thích hợp cho các bít RS0 và RS1 của thanh ghi từ trạng thái PSW, mặc định bằng 0 là tích cực.

Miền RAM được định địa chỉ bít có 16 byte 8 bít = 128 bít, chiếm địa chỉ từ 20h đến 1fh. Mỗi bít ở miền này được định địa chỉ riêng từ 00h đến 7fh nên có thể truy nhập đến từng bít riêng rẽ bằng các lệnh xử lý bít. Vùng RAM được định ịa chỉ bít và các lệnh xử lý bít là một trong những đặc tính nổi bật đem lại sức mạnh cho họ bộ vi điều khiển MCS-51.

- Miền RAM thông thường có 80 byte chiếm địa chỉ từ 30h đến 7fh. Các thanh ghi chức năng đặc biệt (viết tắt theo tiếng Anh là SFR) là tập các thanh ghi bên trong của bộ vi điều khiển. Họ MCS-51 định địa chỉ cho tất cả các SFR ở vùng 128 byte cao của bộ nhớ số liệu trong (xem hình 2), mỗi SFR có tên gọi và địa chỉ riêng, một số SFR có định địa chỉ cho từng bít. Khi bật nguồn hoặc RESET, tất cả các SFR đều được nạp giá trị đầu, sau đó chương trình cần nạp lại giá trị cho các SFR cần dùng theo yêu cầu sử dụng.

Tổ chức 128 byte thấp trong RAM:



Hình 2.4: Sơ đồ tổ chức 128 byte thấp trong ram họ 8051

Việc truy nhập đến các SFR chỉ có thể thực hiện bằng phương pháp địa chỉ trực tiếp với tên gọi hoặc địa chỉ của SFR là toán hạng của lệnh. Với các SFR có định địa chỉ bít, có thể truy nhập và thay đổi trực tiếp từng bít.của nó bằng các lệnh xừ lý bít. Bảng 2 cho biết thông tin chủ yếu về các SFR.

Ở nhóm 8051vùng 128 byte cao của bộ nhớ số liệu trong chỉ có các SFR, không tồn tại các ô nhớ khác ở vùng nhớ này. Ở nhóm 8052 bộ nhớ số liệu trong có 256 byte RAM, các ô nhớ của vùng RAM 128 byte cao chỉ có thể truy nhập được bằng phương pháp địa chỉ gián tiếp, còn các SFR cũng có địa chỉ nằm trong vùng đó nhưng chỉ truy nhập được bằng phương pháp địa chỉ trực tiếp, vì thế việc truy nhập chúng không bị xung đột và nhầm lẫn.

2.1.4 Phầm mềm lập trình vi điều khiển

Có thể viết trên ngôn ngữ Assembler hoặc các ngôn ngữ bậc cao khác như C, Basic, Forth… Tập lệnh Assembler của họ MCS-51 có 83 lệnh, được

chia thành 5 nhóm là các lệnh số học, các lệnh logic, các lệnh chuyển số liệu, các lệnh xử lý bít và các lệnh rẽ nhánh. Các lệnh xứ lý bít là điểm mạnh cơ bản của họ MCS-51, vì chúng làm cho chương trình ngắn gọn hơn và chạy nhanh hơn. Chương jtrình Assembler được viết trên máy tính, sau đó phải dịch ra mã máy của họ

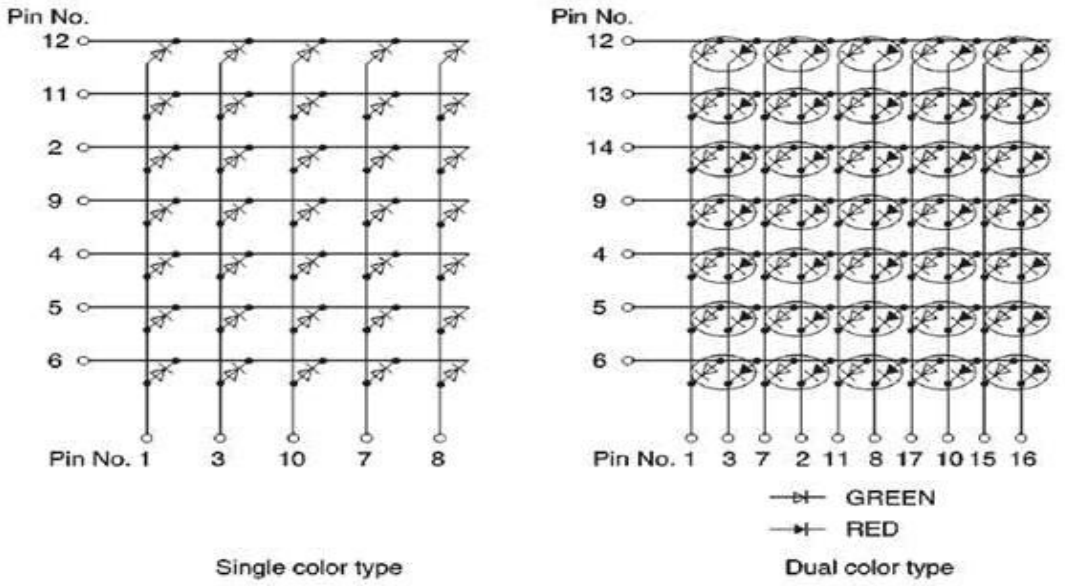
MCS-51 bằng trình biên dịch ASM51, rồi mới nạp. Chương trình mã máy vào bộ nhớ cho trình EEPROM (hoặc EPROM) ở bên trong hoặc bên ngoài MCS-51.

Khi lập trình bằng ngôn ngữ bậc cao như C, Basic, Forth.... cũng phải dịch chúng ra mã máy của họ MCS-51 bằng các trình biên dịch tương ứng, sau đó nạp chương trình mã máy vào bộ nhớ chương trình. Nói chung, chương trình viết trên ngôn ngữ Assembler khó hơn viết trên ngôn ngữ bậc cao, nhưng khi dịch ra mã máy sẽ ngắn gọn hơn và chạy nhanh hơn các chương trình viết trên ngôn ngữ bậc cao. Để viết và nạp phần mềm cho MCS-51, bạn phải có các công cụ là máy vi tính, trình biên dịch ngôn ngữ sử dụng ra mã máy của họ MCS-51 và bộ nạp chương trình mã máy từ máy tính vào bộ nhớ chương trình EEPROM trong Mcs-51 hoặc bộ nhớ EPROM ngoài.

2.2 LEDMATRIX

2.2.1 Hình dạng và cấu tạo của LEDMATRIX

Ma trận led bao gồm nhiều led đơn bố trí thành hàng và cột trong một vỏ. Các tín hiệu điều khiển cột được nối với Anode của tất cả các led trên cùng một cột. Các tín hiệu điều khiển hàng cũng được nối với Cathode của tất cả các led trên cùng một hàng như hình vẽ:



Hình 2.5: Sơ đồ kết nối của ledmatrix

2.2.2 Nguyên lý hoạt động

Khi có một tín hiệu điều khiển ở cột và hàng, các chân Anode của các led trên cột tương ứng được cấp điện áp cao, đồng thời các chân Cathode của các led trên hàng tương ứng được cấp điện áp thấp. Tuy nhiên lúc đó chỉ có một led sáng, vì nó có đồng thời điện thế cao trên Anode và điện thế thấp trên Cathode. Như vậy khi có một tín hiệu điều khiển hàng và cột, thi tại một thời điểm chỉ có duy nhất một led tại chỗ gặp nhau của một hàng và cột là sáng. Các bảng quang báo với số lượng led lớn hơn cũng được kết nối theo câu trúc như vậy.

Trong trường hợp ta muốn cho sáng đồng thời một số led rời rạc trên ma trận, để hiển thị một ký tự nào đó, nếu trong hiển thị tĩnh ta phải cấp áp

cao cho Anode và áp thấp cho Cathode, cho các led tương ứng mà ta muốn sáng. Nhưng khi đó một số led ta không muốn cũng sẽ sáng, miễn là nó nằm tại vị trí gặp nhau của các cột và hàng mà ta cấp nguồn. Vì vậy trong điều khiển led ma trận ta không thể sử dụng phương pháp hiện thị tĩnh mà phải sử dụng phương pháp quét (hiển thị động), có nghĩa là ta phải tiến hành cấp tín hiệu điều khiển theo dạng xung quét trên các hàng và cột có led cần hiển thị. Để đảm bảo cho mắt nhìn thấy các led không bị nháy, thì tần số quét nhỏ nhất cho mỗi chu kỳ là khoảng 20hz(50ms). Trong lập trình điều khiển led ma trận bằng vi xử lý ta cũng phải sử dụng phương pháp quét như vậy.

Ma trận led có thể là loại chỉ hiển thi được một màu hoặc hiển thị được 2 mày trên một điểm, khi đó led có số chân ra tương ứng: đối với ma trận led 8x8 hiển thị một màu, thi số chân ra là 16, trong đó 8 chân dùng để điều khiển hàng và 8 chân còn lại dùng để điều khiển cột. Đối với loại 8x8 có 2 màu thi số chân ra của led là 24 chân, trong đó 8 chân dùng để điều khiển hàng (hoặc cột) chung cho cả hai màu, 16 chân còn lại thi 8 chân dùng để điều khiển hàng (hoặc cột) màu thứ nhất, 8 chân còn lại dùng để điều khiển màu thứ 2.

2.2.3 LED Matrix – Module P10

Dựa trên nguyên tắc như quét màn hình tivi, máy tính, ta có thể thực hiện việc hiển thị ma trận đèn bằng cách quét theo hàng và quét theo cột. Mỗi Led trên ma trận Led có thể coi như một điểm ảnh. Địa chỉ của mỗi điểm ảnh này được xác định đồng thời bởi mạch giải mã hàng và giải mã cột, điểm ảnh này sẽ được xác định nhờ dữ liệu đưa ra từ mạch điều khiển. Như vậy tại mỗi thời điểm chỉ có trạng thái của một điểm ảnh xác định. Tuy nhiên khi xác định địa chỉ và trạng thái của điểm ảnh tiếp theo thì các điểm ảnh còn lại sẽ chuyển về trạng thái tắt.Vì thế để hiển thị được toàn bộ hình ảnh mà ta muốn thì ta phải quét ma trận nhiều lần với tốc độ quét rất lớn, lớn hơn nhiều lần thời gian kịp tắt của đèn. Mắt người chỉ nhận biết được tối đa 24 hình/s do đó nếu tốc

độ quét lớn mắt người sẽ không nhận biết được sự gián đoạn hay là nhấp nháy của đèn Led(đánh lừa cảm giác mắt). Ứng dụng trong hiển thị Led matrix để đảm bảo phù hợp các thông số về điện của từng Led đơn người ta không điều khiển theo chu trình như màn hình tivi (CRT) bởi như vậy để đảm bảo độ sáng của toàn bộ bảng led thì dòng tức thời qua từng led là vô cùng lớn do đó có thể đánh thủng lớp tiếp giáp của led .Trên thực tế người ta có thể ghép chung anot hoặc catot của 1 hàng hoặc 1 cột . Khi đó công việc điều khiển sẽ là chuyển dữ liệu ra các cột và cấp điện cho hàng .Như vậy tài 1 thời điểm sẽ có 1 hàng được điều khiển sáng theo dữ liệu đưa ra. Ngoài ra để đảm bảo độ sáng của bảng thông tin là tốt nhất, đặc biệt với những bảng cỡ lớn theo chiều dọc (có nhiều hàng), thời gian sáng của 1 hàng lúc này sẽ bị giảm đi rất nhiều nếu dữ nguyên kiểu quét 1 hàng .Để khắc phục điều này người ta sử dụng phương pháp điều khiển cho 2 hoặc 4 hàng cùng sáng, từ đó giúp giảm dòng tức thời qua từng led mà vẫn đảm bảo độ sáng tối ưu .Và trong đồ án này module P10 được sử dụng hoạt động trên phương pháp điều khiển cùng lúc 4 hàng cùng sáng tại 1 thời điểm, sau 4 lần quét ta sẽ có 1 khung hình hoàn thiện. 2. Module P10 a.

Thông số Module LED 16x32:

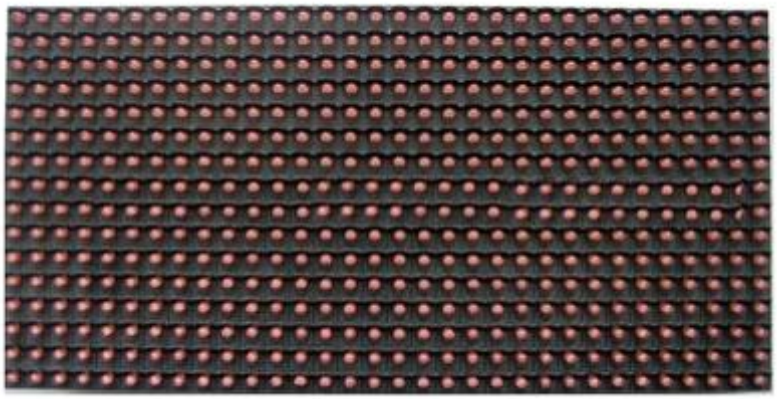
Mã sản phẩm : BW-PH10-4SS

Cách sử dụng Bảng ngoài trời Độ phân giải (mm) 10mm Module dày 30,5mm Kích thước (mm) 320 * 160 Pixel Density (pexel / m) 10.000 Hiển thị một màu Màu đỏ Độ phân giải (pixel) 32 * 16 Trọng lượng (G) 425 Khoảng cách (m) ≥ 12,5 Góc nhìn (°) lựa chọn Nghiêng 110 ± 5 độ, thẳng 60 độ. Nhiệt độ hoạt động (° C) Làm việc Nhiệt độ: -20 °C ~ 50°C Nhiệt độ lưu trữ: -40°C ~ 85 ° C Độ ẩm hoạt động 10 ~ 95% Công suất Trung bình (W / m²) 100 ~ 300 Công suất tiêu thụ tối đa (W / m²) ≤ 500 Chế độ kiểm soát Không đồng bộ Chế độ quét 1/4 quét bởi áp Constant Cân bằng trắng Độ sáng

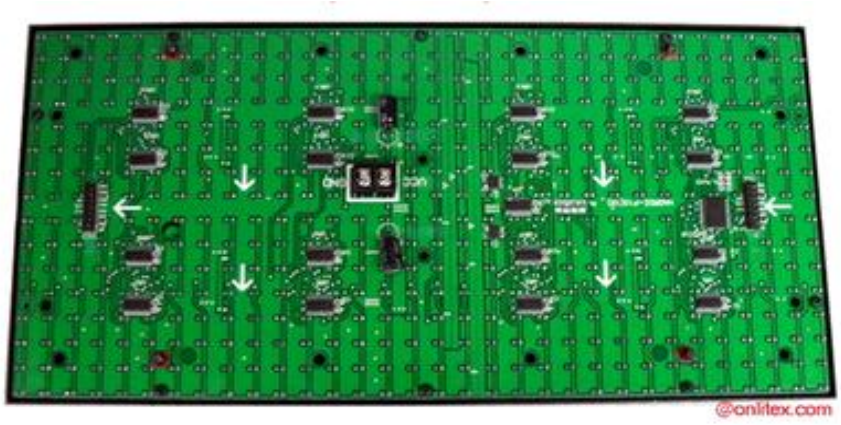
(cd / m²) ≥ 2000 Lớp chống thấm nước IP51 MTTF ≥ 10.000 Tuổi thọ (giờ) ≥ 100,000 Nguồn điện sử dụng 5V/20A chuyên dụng.

Hình ảnh thực tế:

mặt trước



mặt sau

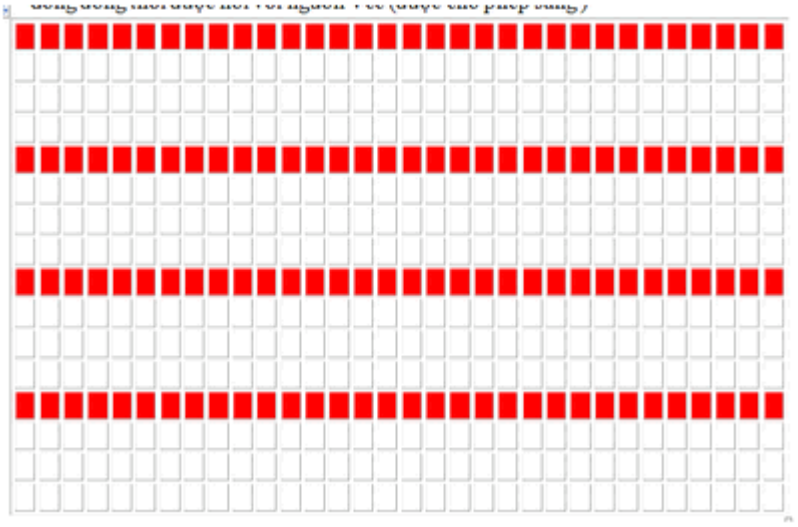


Hình 2.6: Modul P10

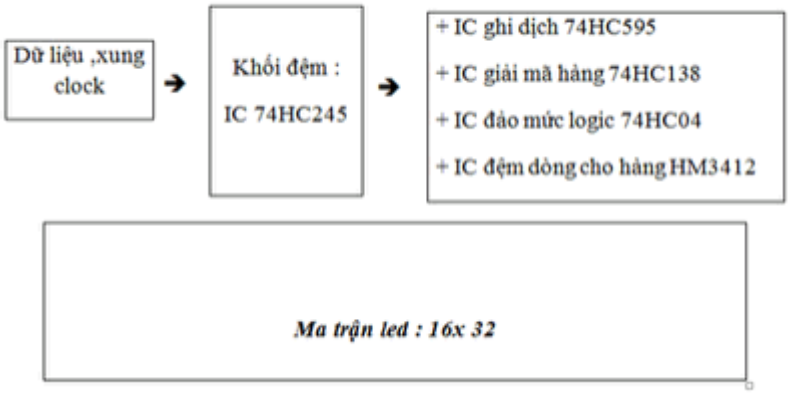
Nguyên lý hoạt động:

Giản đồ xung điều khiển mudue : Các đường điều khiển gồm : - Tín hiệu OE : tích cực mức logic cao (5V) cho phép chốt hàng (hàng tương ứng với 2 tín hiệu A,B được nối đất) - Tín hiệu chọn hàng : A,B là 2 đường tín hiệu cho phép chọn hàng hiển thị

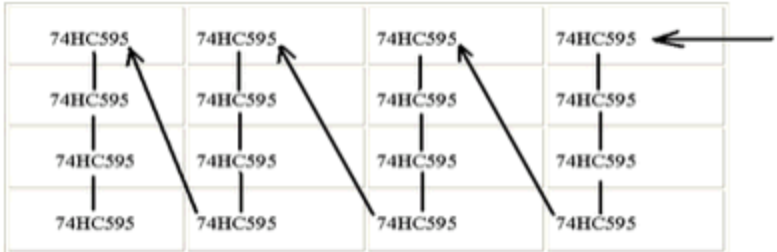
- Tín hiệu CLK : Tín hiệu cho phép chốt dữ liệu ra cột . - Tín hiệu SCK : xung đưa dữ liệu ra IC ghi dịch . - Tín hiệu DATA: đưa dữ liệu cần hiển thị ra bảng led. - Sơ đồ quét của mudule : + Quét theo tỉ lệ ¼ + Tất cả module có 16 dòng,32 cột .Tại 1 thời điểm nhất định sẽ có 4 dòng đồng thời được nối với nguồn Vcc (được cho phép sáng)



Sơ đồ khối của module :



Sơ đồ dịch dữ liệu :



Lý do chọn loại modul này:

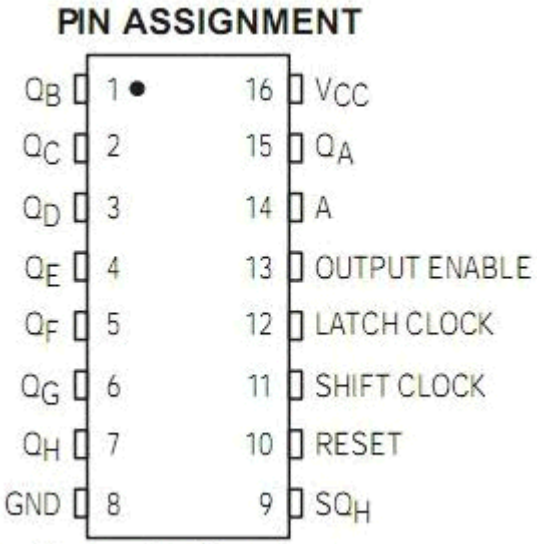
P10 – 1R là loại module LED rất phổ biến trên thị trường và đang được sử dụng rộng rãi tại Việt Nam. +Cách điều khiển đơn giản. +Phù hợp với các bảng thông tin điện tử cỡ vừa và nhỏ. + Cấu tạo đơn giản, rễ dàng lắp đặt , sửa chữa . + Mở rộng kích thước bảng đơn giản, không cần thay đổi phần cứng . + Độ sáng phù hợp với các bảng thông tin ngoài trời . +Sử dụng, lắp đặt đơn giản. +Giá thành không quá đắt (245.000VND / 1module - giá bán lẻ)

2.3 IC 74HC595

2.3.1 Chức năng:

Là một IC ghi dịch 8 bit kết hợp chốt dữ liệu, đầu vào nối tiếp đầu ra song song. Chức năng thường được dung trong các mạch quét led 7 thanh, led matrix… để tiết kiệm số chân VDK tối đa (3 chân). Có thể mở rộng số chân vi điều khiển bao nhiêu tùy thích mà không IC nào có thể làm được bằng cách nối tiếp đầu vào dữ liệu các ic với nhau.

2.3.2 Sơ đồ chân:



Hình 2.7: Sơ đồ chân 74HC595

Giải thích ý nghĩa hoạt động của một số chân quan trọng:

(input)

Chân 14 : đầu vào dữ liệu nối tiếp . Tại 1 thời điểm xung clock chỉ đưa vào được 1 bit

(output)

QA=>QH : trên các chân (15,1,2,3,4,5,6,7)

Xuất dữ liệu khi chân chân 13 tích cực ở mức thấp và có một xung tích cực ở sườn âm tại chân chốt 12

(output-enable)

Chân 13 : Chân cho phép tích cực ở mức thấp (0) .Khi ở mức cao, tất cả các đầu ra của 74595 trở về trạng thái cao trở, không có đầu ra nào được cho phép.

(SQH)

Chân 9: Chân dữ liệu nối tiếp . Nếu dùng nhiều 74595 mắc nối tiếp nhau thì chân này đưa vào đầu vào của con tiếp theo khi đã dịch đủ 8bit.

(Shift clock)

Chân 11: Chân vào xung clock . Khi có 1 xung clock tích cực ở sườn dương(từ 0 lên 1) thì 1bit được dịch vào ic.

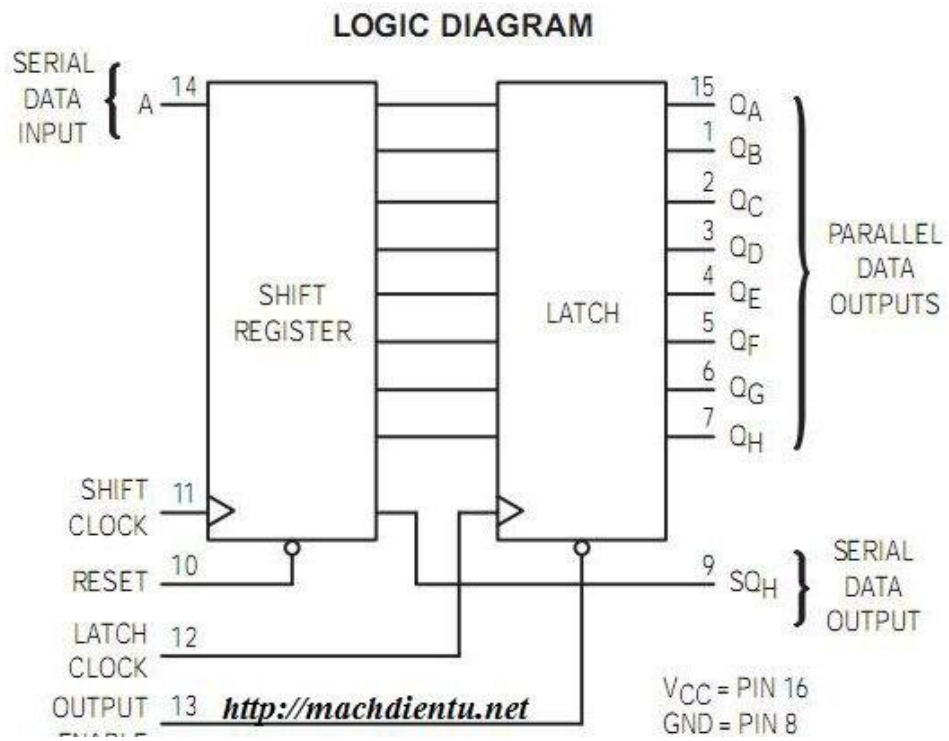
(Latch clock)

Chân 12 : xung clock chốt dữ liệu . Khi có 1 xung clock tích cực ở sườn dương thì cho phép xuất dữ liệu trên các chân output . lưu ý có thể xuất dữ liệu bất kỳ lúc nào bạn muốn ,ví dụ đầu vào chân 14 dc 2 bit khi có xung clock ở chân 12 thì dữ liệu sẽ ra ở chân Qa và Qb (chú ý chiều dịch dữ liệu từ Qa=>Qh)

(Reset)

Chân 10: khi chân này ở mức thấp(mức 0) thì dữ liệu sẽ bị xóa trên chip)

Sơ đồ hoạt động của chip:



Hình 2.8: sơ đồ chức năng các chân

2.3.3 Bảng thông số chip:

Symbol	Parameter	Value	Unit
V_{CC}	DC Supply Voltage (Referenced to GND)	– 0.5 to + 7.0	V
V_{in}	DC Input Voltage (Referenced to GND)	– 0.5 to V_{CC} + 0.5	V
V_{out}	DC Output Voltage (Referenced to GND)	– 0.5 to V_{CC} + 0.5	V
I_{in}	DC Input Current, per Pin	± 20	mA
I_{out}	DC Output Current, per Pin	± 35	mA
I_{CC}	DC Supply Current, V_{CC} and GND Pins	± 75	mA
P_D	Power Dissipation in Still Air, Plastic DIP† SOIC Package† TSSOP Package†	750 500 450	mW
T_{stg}	Storage Temperature	– 65 to + 150	°C
T_L	Lead Temperature, 1 mm from Case for 10 Seconds (Plastic DIP, SOIC or TSSOP Package)	260	°C

Đây là ic đầu ra hoạt động ở 2 mức 0 &1 dòng ra tầm 35mA . điện áp hoạt động <=7V . Công suất trung bình 500mW

Dựa vào bảng tính toán được các thông số khi thiết kế mạch

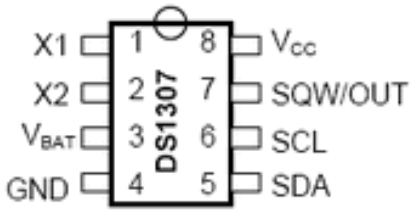
2.3.4 Tần số đáp ứng:

Symbol	Parameter		Min	Max	Unit
V_{CC}	DC Supply Voltage (Referenced to GND)		2.0	6.0	V
V_{in}, V_{out}	DC Input Voltage, Output Voltage (Referenced to GND)		0	V_{CC}	V
T_A	Operating Temperature, All Package Types		– 55	+ 125	°C
t_r, t_f	Input Rise and Fall Time (Figure 1)	V_{CC} = 2.0 V V_{CC} = 4.5 V V_{CC} = 6.0 V	0 0 0	1000 500 400	ns

Tại 6V thì tần số vào đáp ứng khoảng 400ns . Dựa vào đó chúng ta se đưa được ra tần số quét hợp lý.

2.4 DS1307 IC thời gian thực

DS1307 là chip đồng hồ thời gian thực (RTC: Real-time clock), khái niệm thời gian thực ở đây được dung với ý nghĩa thời gian tuyệt đối mà con người đang sử dụng, tính bằng giây, phút, giờ… DS1307 là một sản phẩm của Dallas Semiconductor (một công ty thuộc Maxim Intergrated Products). Chip này có 7 thanh ghi 8bit chứa thời gian là : giây, phút, giờ, thứ , ngày, tháng, năm. Ngoài ra DS1307 còn có 1 thanh ghi điều khiển ngõ ra phụ và 56 thanh ghi trống có thể dung như RAM. DS1.07 xuất hiện ở 2 gói SOIC và DIP có 8 chân như trong hình vẽ dưới:



Các chân của DS1307 được mô tả như sau:

• X1 và X2: là 2 ngõ kết nối với 1 thạch anh 32.768KHz làm nguồn tạo dao động cho chip.

• VBAT: cực dương của một nguồn pin 3V nuôi chip.

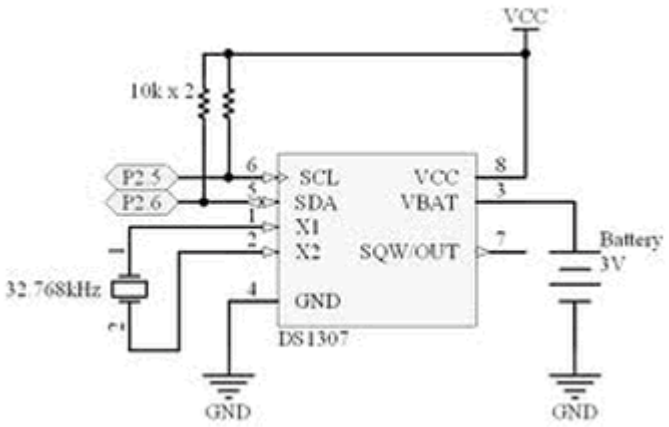
• GND: chân mass chung cho cả pin 3V và Vcc.

• Vcc: nguồn cho giao diện I2C, thường là 5V và dùng chung với vi điều khiển. Chú ý là nếu Vcc không được cấp nguồn nhưng VBAT được cấp thì DS1307 vẫn đang hoạt động (nhưng không ghi và đọc được).

• SQW/OUT: một ngõ phụ tạo xung vuông (Square Wave / Output Driver), tần số của xung được tạo có thể được lập trình. Như vậy chân này hầu như không liên quan đến chức năng của DS1307 là đồng hồ thời gian thực, chúng ta sẽ bỏ trống chân này khi nối mạch.

• SCL và SDA là 2 đường giao xung nhịp và dữ liệu của giao diện I2C.

• Có thể kết nối DS1307 bằng một mạch điện đơn giản như trong hình sau:



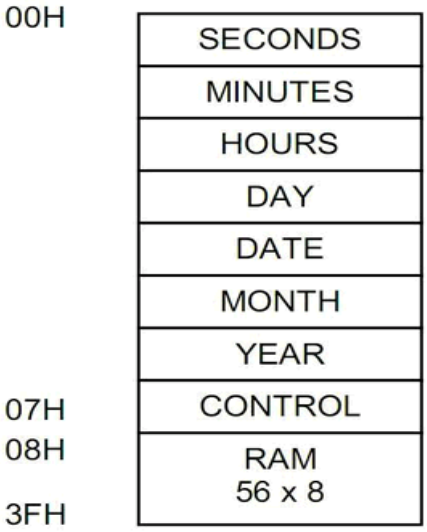
Cấu tạo bên trong DS1307 bao gồm một số thành phần như mạch nguồn, mạch dao động, mạch điều khiển logic, mạch giao diện I2C, con trỏ địa chỉ và các thanh ghi (hay RAM). Sử dụng DS1307 chủ yếu là ghi và đọc các thanh ghi của chip này. Vì thế có 2 vấn đề cơ bản đó là cấu trúc các thanh ghi và cách truy xuất các thanh ghi này thông qua giao diện I2C.

Như đã trình bày, bộ nhớ DS1307 có tất cả 64 thanh ghi 8-bit được đánh địa chỉ từ 0 đến 63 (từ 00H đến 3FH theo hệ HexaDecimal). Tuy nhiên, thực chất chỉ có 8 thanh ghi đầu là dùng cho chức năng "đồng hồ" (RTC) còn lại 56 thanh ghi bỏ trống có thể được dùng chứa biến tạm như RAM nếu muốn. Bảy thanh ghi đầu tiên chứa thông tin về thời gian của đồng hồ bao gồm: giây (SECONDS), phút (MINUETS), giờ (HOURS), thứ (DAY), ngày (DATE), tháng (MONTH) và năm (YEAR). Việc ghi giá trị vào 7 thanh ghi này tương đương với việc "cài đặt" thời gian khởi động cho RTC. Việc đọc giá trị từ 7 thanh ghi là đọc thời gian thực mà chip tạo ra. Ví dụ, lúc khởi động chương trình, chúng ta ghi vào thanh ghi "giây" giá trị 42, sau đó 12s chúng ta đọc thanh ghi này, chúng ta thu được giá trị 54. Thanh ghi thứ 8 (CONTROL) là thanh ghi

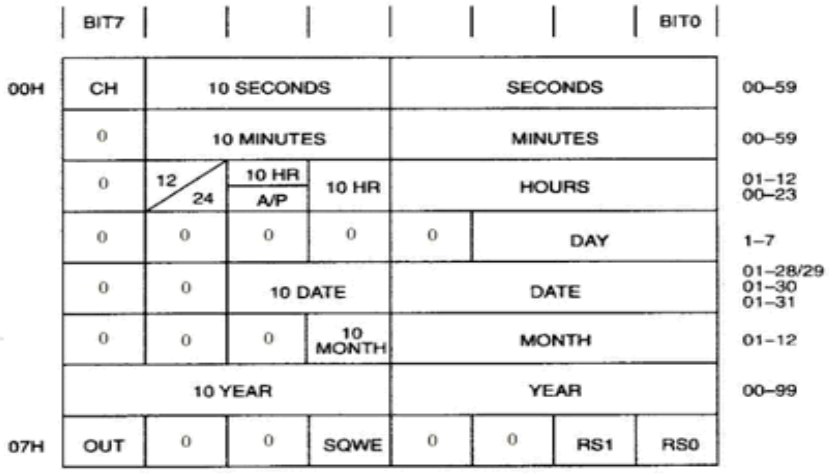
điều khiển xung ngõ ra SQW/OUT (chân 6). Tuy nhiên, do chúng ta không dùng chân SQW/OUT nên có thể bỏ qua thanh ghi thứ 8.

Vì 7 thanh ghi đầu tiên là quan trọng nhất trong hoạt động của DS1307, chúng ta sẽ khảo sát các thanh ghi này một cách chi tiết. Trước hết hãy quan sát tổ chức theo từng bit của các thanh ghi này như trong hình:

Tổ chức bộ nhớ của DS1307



Tổ chức các thanh ghi thời gian:



Hình 2.8: Tổ chức bộ nhớ và tổ chức thanh ghi thời gian DS1307

Thanh ghi giây (SECONDS): thanh ghi này là thanh ghi đầu tiên trong bộ nhớ của DS1307, địa chỉ của nó là 0x00. Bốn bit thấp của thanh ghi này chứa mã BCD 4-bit của chữ số hàng đơn vị của giá trị giây. Do giá trị cao nhất của chữ số hàng chục là 5 (không có giây 60) nên chỉ cần 3 bit (các bit SECONDS 6:4) là có thể mã hóa được (số 5 =101, 3 bit). Bit cao nhất, bit 7, trong thanh ghi này là 1 điều khiển có tên CH (Clock halt – treo đồng hồ), nếu bit này được set bằng 1 bộ dao động trong chip bị vô hiệu hóa, đồng hồ không hoạt động. Vì vậy, nhất thiết phải reset bit này xuống 0 ngay từ đầu. Thanh ghi phút (MINUTES): có địa chỉ 01H, chứa giá trị phút của đồng hồ. Tương tự thanh ghi SECONDS, chỉ có 7 bit của thanh ghi này được dùng lưu mã BCD của phút, bit 7 luôn luôn bằng 0.

Thanh ghi giờ (HOURS): có thể nói đây là thanh ghi phức tạp nhất trong DS1307. Thanh ghi này có địa chỉ 02H. Trước hết 4-bits thấp của thanh ghi này được dùng cho chữ số hàng đơn vị của giờ. Do DS1307 hỗ trợ 2 loại hệ thống hiển thị giờ (gọi là mode) là 12h (1h đến 12h) và 24h (1h đến 24h) giờ, bit6 (hình 4) xác lập hệ thống giờ. Nếu bit6=0 thì hệ thống 24h được chọn, khi đó 2 bit cao 5 và 4 dùng mã hóa chữ số hàng chục của giá trị giờ. Do giá trị lớn nhất của chữ số hàng chục trong trường hợp này là 2 (=10, nhị phân) nên 2 bit 5 và 4 là đủ để mã hóa. Nếu bit6=1 thì hệ thống 12h được chọn, với trường hợp này chỉ có bit 4 dùng mã hóa chữ số hàng chục của giờ, bit 5 (màu orange trong hình 4) chỉ buổi trong ngày, AM hoặc PM. Bit5 =0 là AM và bit5=1 là PM. Bit 7 luôn bằng 0.

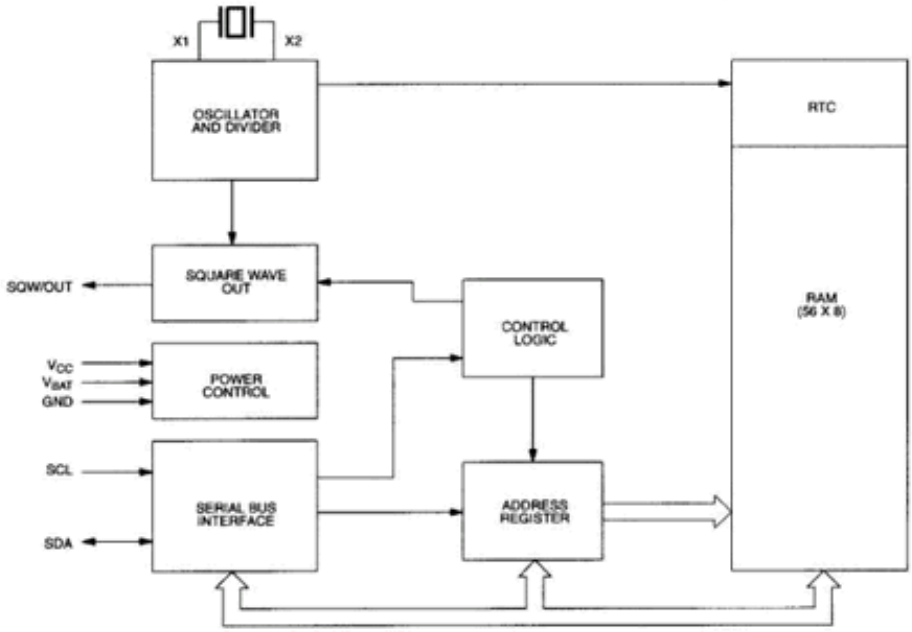
Thanh ghi thứ (DAY – ngày trong tuần): nằm ở địa chỉ 03H. Thanh ghi DAY chỉ mang giá trị từ 1 đến 7 tương ứng từ Chủ nhật đến thứ 7 trong 1 tuần. Vì thế, chỉ có 3 bit thấp trong thanh ghi này có nghĩa.

Các thanh ghi còn lại có cấu trúc tương tự, DATE chứa ngày trong tháng (1 đến 31), MONTH chứa tháng (1 đến 12) vàYEAR chứa năm (00 đến

99). Chú ý, DS1307 chỉ dùng cho 100 năm, nên giá trị năm chỉ có 2 chữ số, phần đầu của năm do người dùng tự thêm vào (ví dụ 20xx).

Ngoài các thanh ghi trong bộ nhớ, DS1307 còn có một thanh ghi khác nằm riêng gọi là con trỏ địa chỉ hay thanh ghi địa chỉ (Address Register). Giá trị của thanh ghi này là địa chỉ của thanh ghi trong bộ nhớ mà người dùng muốn truy cập.

Cấu trúc DS1307:



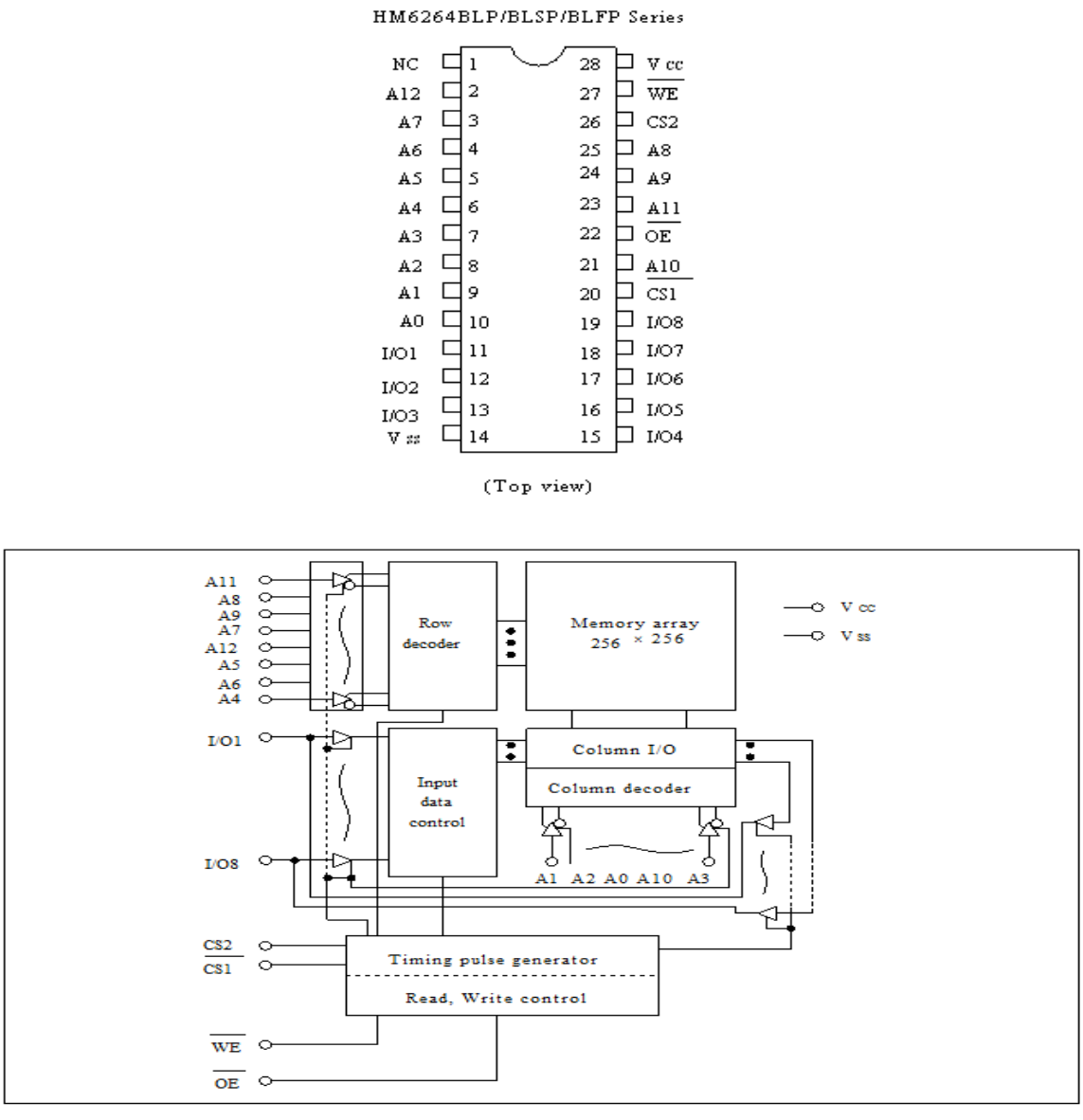
Hình 2.9: Sơ đồ khối DS1307

DS1307 có thể hoạt động ở 2 chế độ sau:

• Ở chế độ slave nhận (chế độ DS1307 ghi): chuỗi dữ liệu và chuỗi xung clock sẽ được nhận thông qua SDA và SCL. Sau mỗi byte được nhận thì 1 bit ACKnowledge sẽ được truyền. Các điều kiện START và STOP sẽ được nhận dạng khi bắt đầu và kết thúc 1 truyền 1 chuỗi, nhận dạng địa chỉ được thực hiện bởi phần cứng sau khi chấp nhận địa chỉ của slave và bit một chiều.

• Chế độ slave phát (chế độ DS1307 đọc): byte đầu tiên slave nhận được tương tự như chế độ slave ghi. Tuy nhiên trong chế độ này thì bit chiều lại chỉ chiều chuyền ngược lại. Chuỗi dữ liệu được phat đi trên SDA bởi DS1307 trong khi chuỗi xung clock vào chân SCL.

2.5 IC HM6264

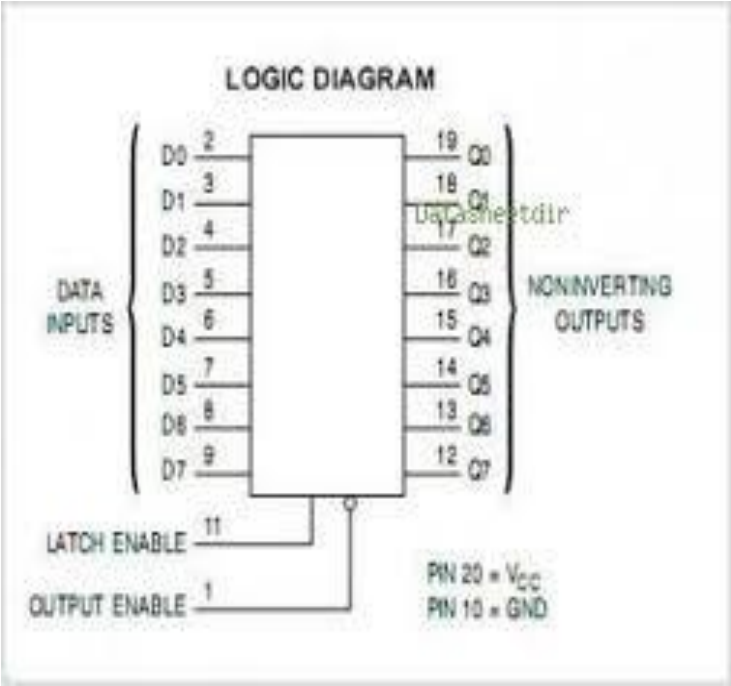


Hình 2.10: Sơ đồ khối của HM6264

Mang chức năng như một RAM ngoài để nhớ cho chip 8051

- Dung lượng 8Kx8
- 8 chân dữ liệu
- 13 chân địa chỉ
- Hai chân chọn chip
- Chân điều khiển đọc
- Chân điều khiển ghi

2.6 IC 74HC573



Hình 2.11: Sơ đồ chức năng các chân 74HC573

Là một IC chốt dữ liệu (mục đích là nhằm tiết kiệm được chân của VĐK). Chốt dữ liệu là lưu giữ trạng thái cổng ra cố định khi cổng vào thay đổi. Nó thực hiện khi chân chôt LE của nó ở mức thấp(0) có nghĩa là đầu ra giữ nguyên trạng thái trước đó mà ko quan tâm đến trạng thái đầu vào như thế nào.

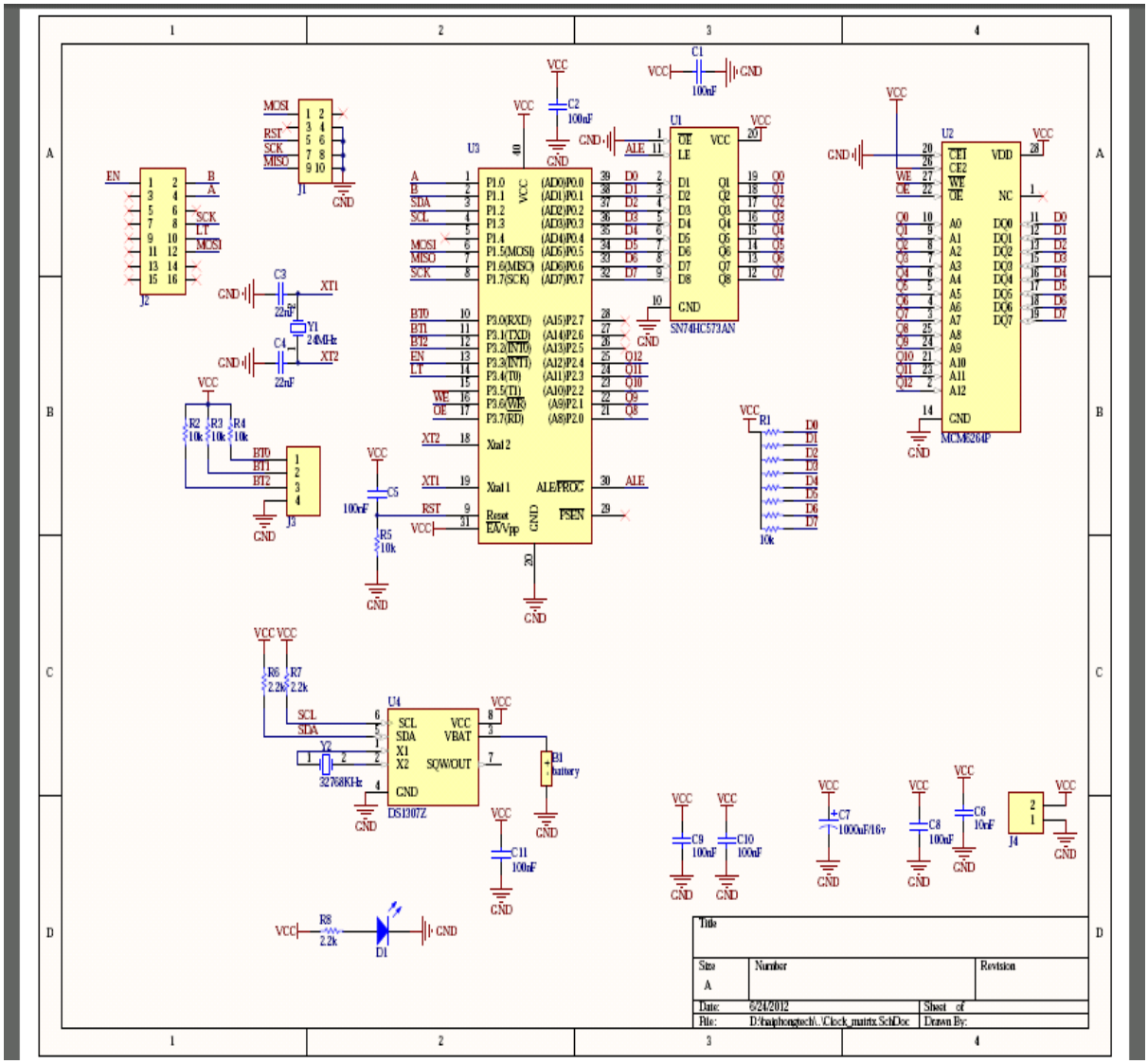
IC này chủ yếu được dùng nhiều cho ứng dụng chốt dữ liệu với các ứng dụng hiển thị led 7 thanh, dĩ nhiên có thể cho các ứng dụng khác tùy vào người dùng có thể phối ghép.

giả sử ta muốn đk 24 led chỉ bằng 1 cổng P2, ở đây mình mắc 8 cổng dữ liệu vào của 3 IC 74HC573 vào Port P2 của 8051. Vậy ta phải làm thế nào? Khi bạn cho chạy dữ liệu ở cổng P2, đồng thời 3 cổng vào của 3 IC 753 cũng thay đổi trạng thái. IC nào chân LE tích cực thì đầu ra sẽ thay đổi đầu vào. Các IC còn lại ko dc tích cực chân LE thì đầu ra sẽ giữ nguyên trạng thái trước đó của nó. Đấy chính là chốt. Nó là nguyên lý hoạt động của con Triger D.

CHƯƠNG 3

THIẾT KẾ VÀ THI CÔNG

3.1 SƠ ĐỒ NGUYÊN LÝ



Hình 3.1: Sơ đồ nguyên lý

VĐK thực hiện theo các bước sau:

+ b1. VĐK Xuất địa chỉ ra PORT_0 =00h (8 bit đia chỉ thấp A0..A7) và 8 bit địa chỉ cao = 40h (A8..A15). Với địa chỉ này, trên sớ đồ giải mã ngõ ra

PORT_C của IC 74138 ở mức thấp (một ngõ vào của cổng NOR sẽ ở mức thấp) (1)và khi đó, chân WR\ của VĐK sẽ ở mức cao.

+ b2. VĐK xuất tiếp dữ liệu 10101010b ra PORT_0 (AD0..AD7) và chân WR\ sẽ tự động tích cực mức thấp (2)

+ Kết hợp (1) và (2), 2 ngõ vào của cộng NOR ở mức thấp, nên ngõ ra của cổng NOR ở mức cao => chân LE của IC OU22 được tích cực mức cao làm cho IC 74HC573 mở cổng, kết quả là dữ liệu 10101010b sẽ xuất ra Q0..Q7 trên IC HM6264. Từ đây dữ liệu được truyền đi vào IC dịch 74HC595 (trên led matrix) , và xuất ra màn hình ledmatrix để hiển thị.

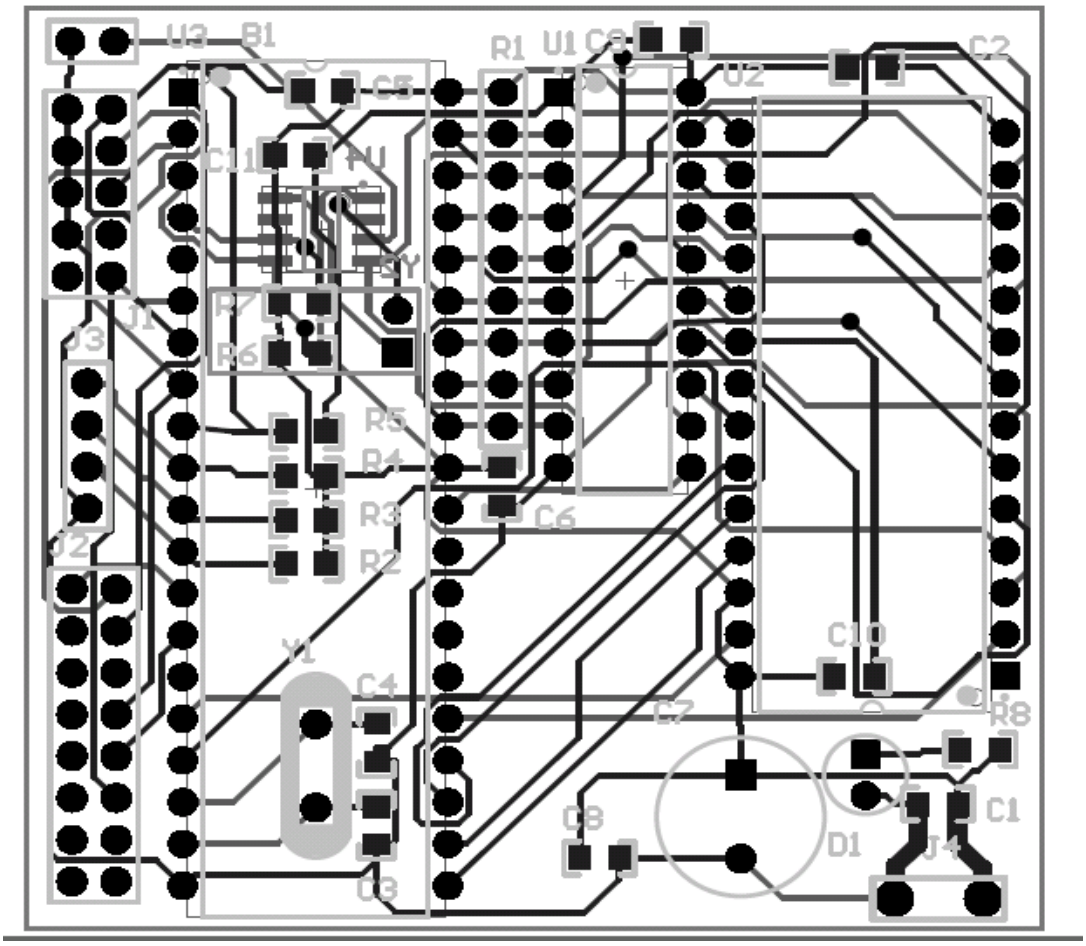
Chân 16,17 của vi điều khiển được nối với 22, 27 của IC HM6264 để đọc thông tin và ghi dữ liệu từ con Ram này.

Chân 18, 19 của vi điều khiển được nối vào thạch anh để tạo xung điều khiển. Chân 9 được nối vào tụ điện và một điện trở sau đó nối vào nguồn và đất (mục đích để reset vi điều khiển khi có sự cố ngắt điện).

Chân 5,6 nối với chân 3,4 tức cổng P1.2 & P1.3 của vi điều khiển,chân 8 của DS1307 được nối với nguồn 5v, chân 4 nối GND, các chân 1 và 2 được nối với thạch anh để tạo xung dao động. Chân 3 được nối với pin Cmos mục đích là lưu dữ liệu khi xảy ra trường hợp mất điện (vì DS1307 không có pin nhớ bên trong).

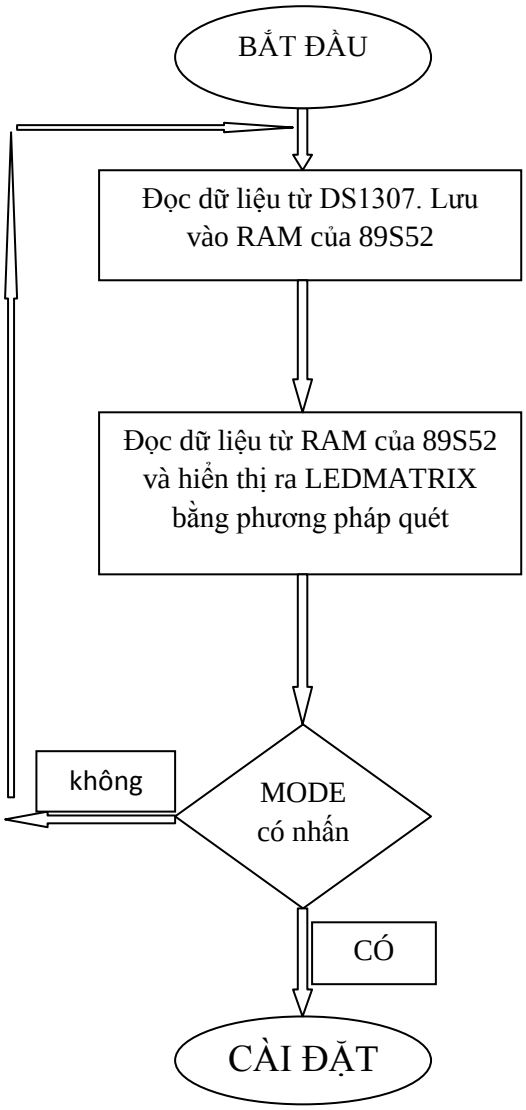
Các chân còn lại của vi điều khiển được đưa ra bên ngoài dưới dạng kết nối modul để kết nối với Modul LEDMAXTRIX, nút bấm hoặc kết nốt trực tiếp với bộ kit nạp chương trình cho vi điều khiển

3.2 SƠ ĐỒ MẠCH IN

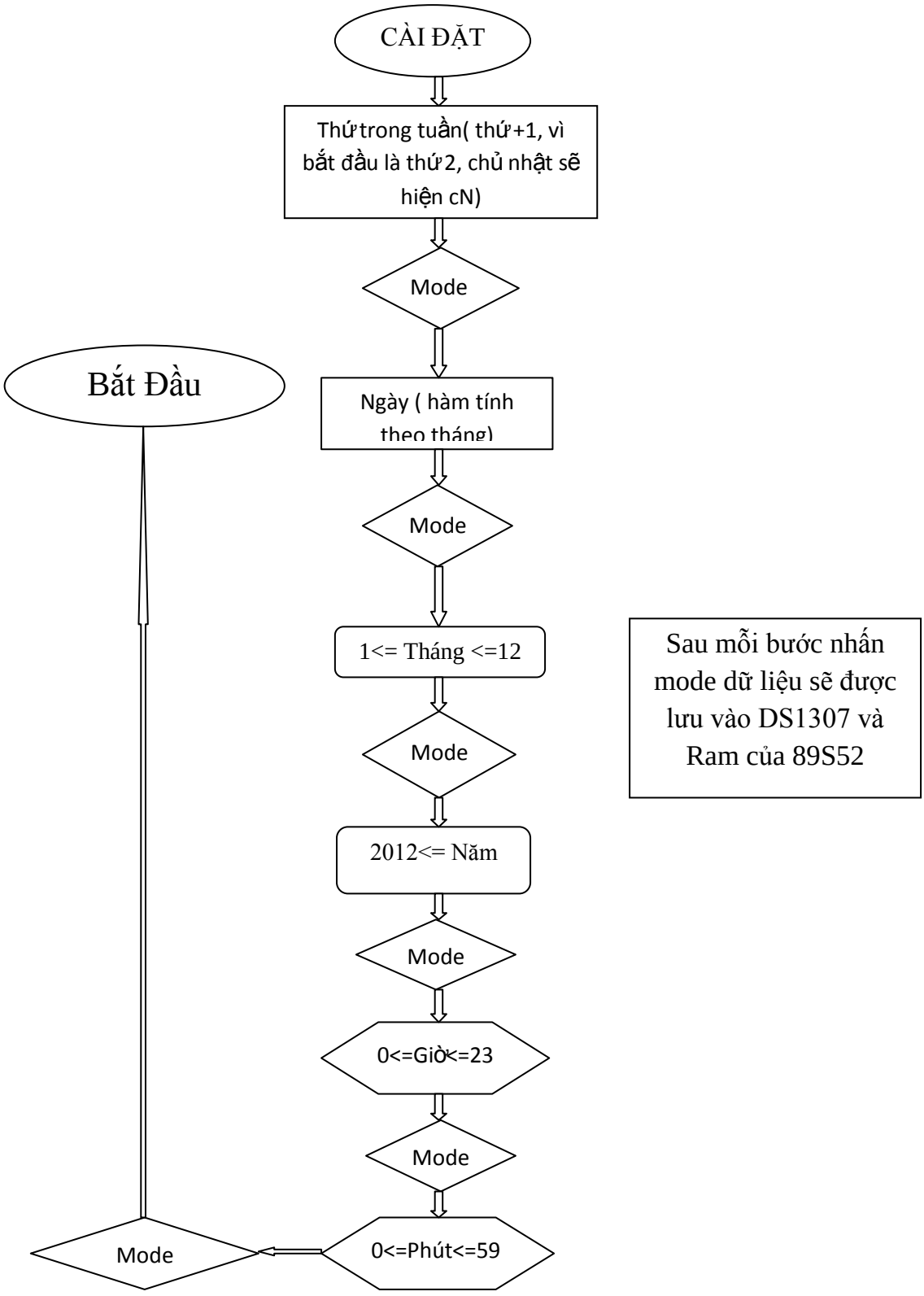


Hình 3.2: Sơ đồ mạch in

3.3 LƯU ĐỒ THUẬT TOÁN CHƯƠNG TRÌNH CHÍNH

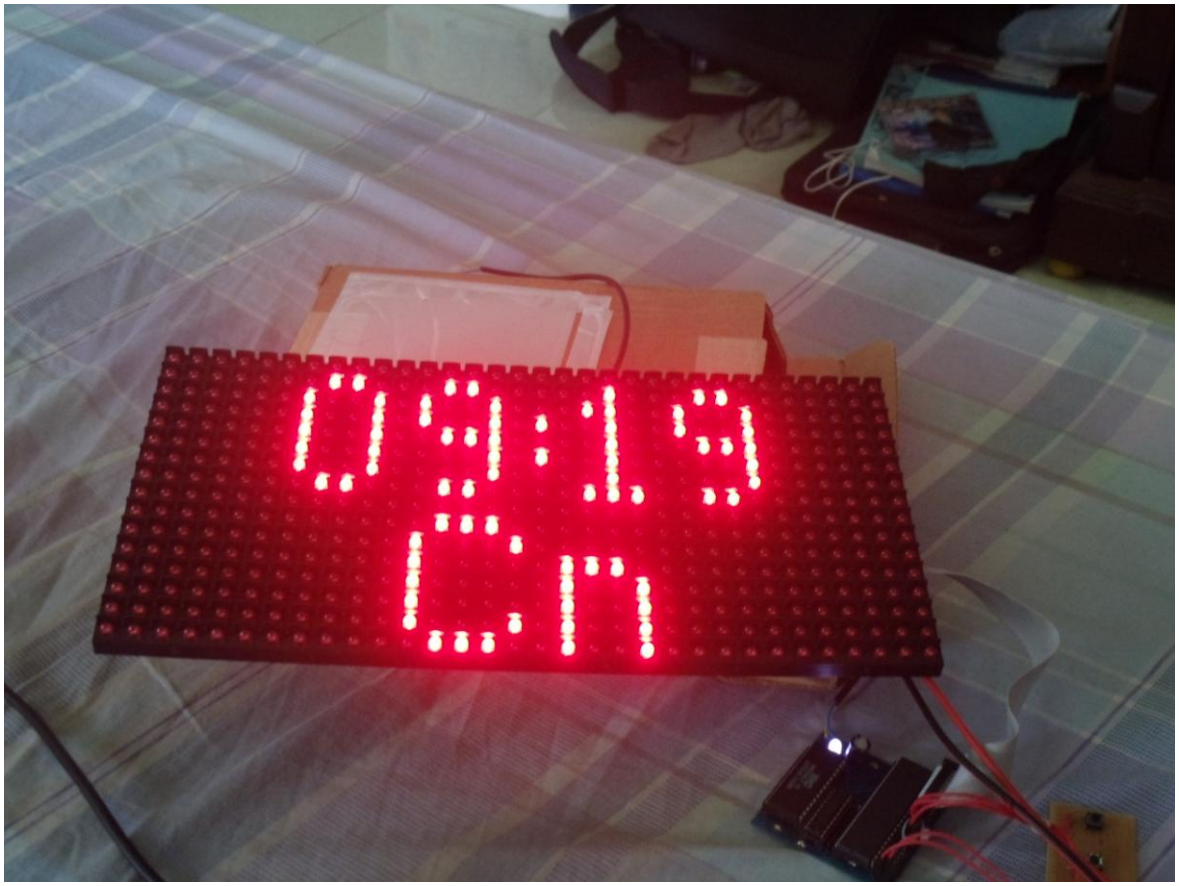


Hình 3.3a: Lưu đồ thuật toán chương trình chính

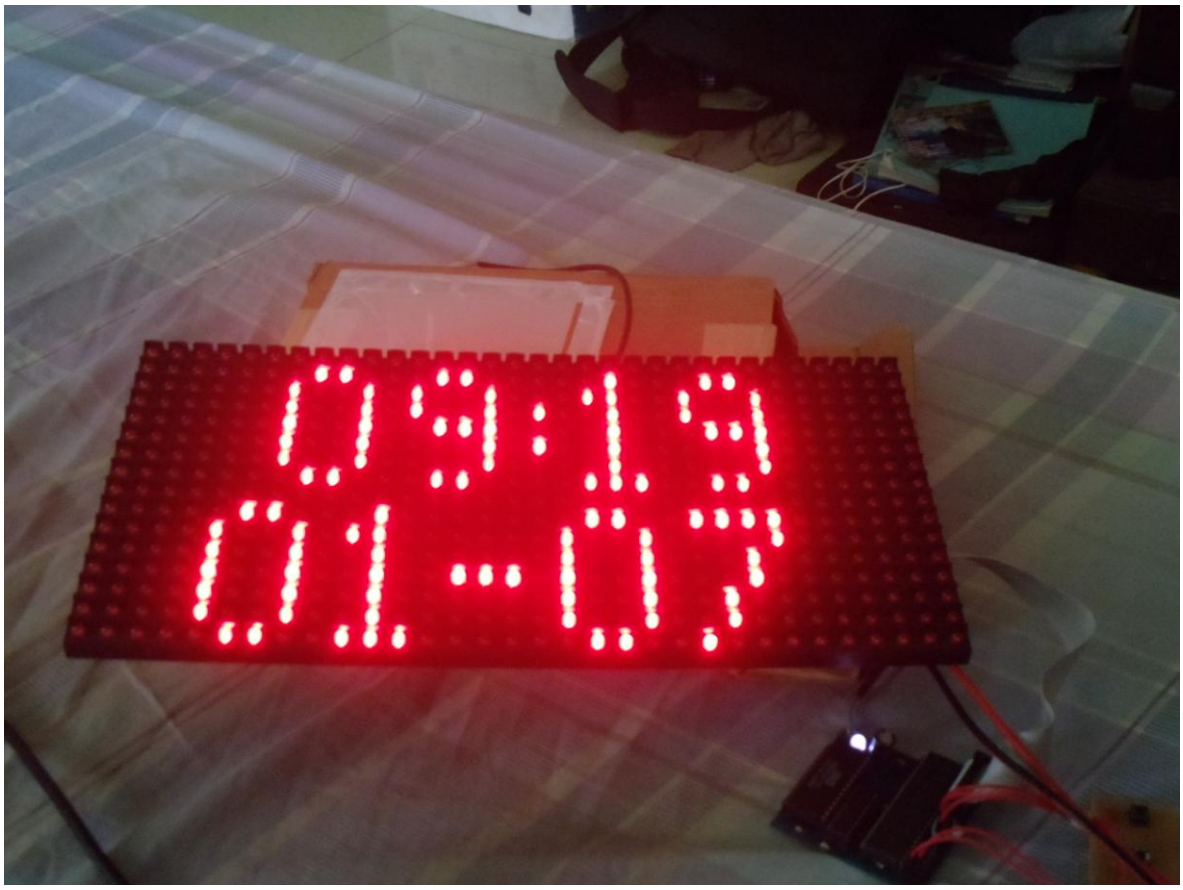


Hình 3.3b: Lưu đồ thuật toán chương trình cài đặt

3.4 HÌNH ẢNH SẢN PHẨM



Hình 3.4a: Giờ - Phút - Thứ (thứ trong tuần)



Hình 3.4b: Giờ - Phút - Ngày - Tháng

3.5 CHƯƠNG TRÌNH ĐIỀU KHIỂN

3.5.1 Khai báo biến và hàm

```
#include <REGX52.H>
#define    M_SDA          P1_2
#define    M_SCL          P1_3
#define    mosi           P1_5
#define    clk            P1_7
#define    latch          P3_4
#define    LS0            P1_0
#define    LS1            P1_1
#include <SPI.c>
#include <DS1307.c>
#define   up              P3_0
#define   down            P3_1
#define   mode            P3_2
#define   screen_light    P3_3   //On/Off Screen
unsigned int i=0,ay=0,ax=0;
unsigned int t=0,y=0,j=0;
unsigned int Yi=408,Xi=16,YB=32,XB=16,With=0,WithB=0;
unsigned int EndByte=0,EndByteB=0,End4=0,Ad1=0,Ad2=0,Ad3=0,Ad4=0;
unsigned int Dk2=0,Dk1=0,step=0;
unsigned int Colum_Counter=0, Row_Counter=0,
Bit_Counter=0,Shift_Time=0;
```

```
bit Enable_Shift=0,first=0,enable_delay=0,in_mode=0, in_up=0, in_down=0,
save=0;
unsigned char m=0, number=0;
unsigned char time_delay=0;
void display(unsigned char ascii_text);
void matrix_goto_xy(unsigned char xx, unsigned char yy);
void Read_Time();
void Display_Time(unsigned char H,unsigned char M);
void matrix_clear();
```

3.5.2 Khởi tạo các tham số

```
unsigned char xdata Matrix[64],
Matrix_Temp[64],Matrix_G[64],Matrix_Colum[16];
char Seconds=0,Minute=0,Hour=0,Data=0,Day=0,Month=0,Year=0,
Data_Temp=0;
char Data_AL=0,Month_AL=0;
void init()//khoi tao cho ngat
{
EA=1;//cho phep ngat toan cuc
ET0=1;
ET1=1;
//----------------thiet lap tna so bus----------------------
TMOD=0x11;//chon timer 1 che do 8 bit tu lap lai
TH1=0xFE;//chon thoi gian ngat la 1ms
```

```
TL1=0x18;
TH0=0xFC;//chon thoi gian ngat la 1ms
TL0=0x18;
TR0=1;//khoi dong timer 0
TR1=1;//khoi dong timer 1
}
void delay_ms(unsigned int time)//tao tre thoi gian
{
while(time--)
{
unsigned char temp=11;
while(temp--);
}
}
void init_matrix(void)  {
  //--------------------------------Khoi tao tham so matrrix-----------------------
  With=Yi/8;
  WithB=YB/8;
  EndByteB=WithB*XB;//Vi tri Byte cuoi cung cua man hinh
  EndByte=With*Xi;//Vi tri Byte cuoi cung cua chuoi
  End4=WithB*16-WithB;//Byte dau tien cua hang cuoi cung trong 1 modul
  Dk2=WithB*4;
  Dk1=End4-(Dk2-WithB)+WithB;
```

```
  step=WithB*4;//Buoc nhay toi 74HC595 trong cung 1 cot
  Ad1=0;//Dia chi tang 1
  Ad2=Ad1+WithB*16;//Dia chi tang 2
  Ad3=Ad2+WithB*16;//Dia chi tang 3
  Ad4=Ad3+WithB*16;//Dia chi tang 4
}
```

3.5.3 Chương trình chính

```
void Display_Data(unsigned char Da,unsigned char Mo)  {// Hien thi thang-ngay
  unsigned char n_c=0;
  unsigned char N_1=0,N_2=0,N_3=0,N_4=0;

  N_1=Da/10;// Lay phan nguyen
  N_2=Da%10;// Lay phan du
  N_3=Mo/10;// Lay phan nguyen
  N_4=Mo%10;// Lay phan du
  for(n_c=0;n_c<64;n_c++)
    {
    Matrix_Temp[n_c]=0;// Copy de dich
    }
  display(N_1+48);
  display(N_2+48);
  display('-');
```

```
  display(N_3+48);
  display(N_4+48);
  for(n_c=0;n_c<64;n_c++)
    {
    Matrix_G[n_c]=Matrix_Temp[n_c];// Copy de dich
    }
}
void Display_Time(unsigned char H,unsigned char M)  {// Hien thi gio-phut
  unsigned char n_c=0;
  unsigned char N_1=0,N_2=0,N_3=0,N_4=0;
  N_1=H/10;// Lay phan nguyen
  N_2=H%10;// Lay phan du
  N_3=M/10;// Lay phan nguyen
  N_4=M%10;// Lay phan du
  for(n_c=0;n_c<64;n_c++)
    {
    Matrix_Temp[n_c]=0;// Copy de dich
    }
  display(N_1+48);
  display(N_2+48);
  display(':');
  display(N_3+48);
  display(N_4+48);
```

```
  for(n_c=0;n_c<64;n_c++)
    {
    Matrix[n_c]=0;// Copy de dich
    }
   for(n_c=0;n_c<32;n_c++)
    {
    Matrix[n_c]=Matrix_Temp[n_c];// Hien thi luon tren man hinh_ko can
dich
    }
}
void Display_Day(unsigned char D) {// Hien thi thu trong tuan
  unsigned char n_c=0;
  for(n_c=0;n_c<64;n_c++)

    {
    Matrix_Temp[n_c]=0;// Copy de dich
    }
  if(D<7)
  {
   display('T');
   display('H');
   display('U');
   display(' ');
```

```
    display(D+1+48);// Cong them 1 la vi tinh tu thu 2
   }
   else
   {
    matrix_goto_xy(15,11);
    display('C');
    display('N');
   }
   for(n_c=32;n_c<64;n_c++)
     {
     Matrix_G[n_c]=Matrix_Temp[n_c];// Copy de dich
     }
}
unsigned char Convert_Decimal(unsigned char Bcd) {
   unsigned char Temp_Bcd=0;
   Temp_Bcd=Bcd>>4;// Dich 4 bit cao sang phai
   return(Temp_Bcd*10+(Bcd & 0x0F));// Tra ve gia tri sau khi convert
}
unsigned char Convert_Binary(unsigned char Decimal) {
   unsigned char Temp_1=0,Temp_2=0;
   Temp_1=(Decimal/10)<<4;
   Temp_2=Decimal%10;
   return(Temp_1|Temp_2);// Tra ve gia tri sau khi convert
```

```
}
void Read_Time(void)
{
  Day=Convert_Decimal(read_DS1307(0x03));// Hien thi thu trong tuan
  Data=Convert_Decimal(read_DS1307(0x04));// Hien thi ngay trong thang
  Month=Convert_Decimal(read_DS1307(0x05));// Hien thi thang trong nam
  Year=Convert_Decimal(read_DS1307(0x06));// Hien thi nam
  Hour=Convert_Decimal(read_DS1307(0x02));// Hien thi gio
  Minute=Convert_Decimal(read_DS1307(0x01));// Hien thi phut
  Seconds=Convert_Decimal(read_DS1307(0x00));// Hien thi giay
}
void shift_l(void)   {// Ham dich sang trai
  unsigned int shift_i=0,s_i=0;
  unsigned char Data_A=0,Data_B=0;
  for(s_i=0;s_i<EndByteB;s_i++)
    {
    Matrix_Temp[s_i]=0x00;// Lam sach bo dem nhap
    }
  for(s_i=8;s_i<XB;s_i++)
    {
    if(number==0)// Neu ko yeu cau banner
      {
      Yi=32;
```

```
        YB=32;
        init_matrix();
        Matrix_Colum[s_i]=Matrix_G[s_i*With];// Copy cot dau tien trong
mang goc-Tranh cut dau du lieu
        }
      }
while(Colum_Counter < With+WithB)// Neu chua dich ra day man hinh
{
  if(Bit_Counter<8 && Enable_Shift==1)//Neu dich chua het 1 cot(8bit)
     {
     for(shift_i=32;shift_i<EndByteB;shift_i++)// Chi cho dich phan duoi
        {
        if(((shift_i+1)%WithB)!=0)// Neu ma ko phai byte ngoai cung
           {
           Data_A=Matrix_Temp[shift_i]<<1;// Dich sang trai 1 bit
           Data_B=Matrix_Temp[shift_i+1]>>7;// Dich byte lien ke sang phai 7
bit
           Matrix_Temp[shift_i]=Data_A|Data_B;// Cong 2 byte lien ke
           }
        else if(Colum_Counter<With)// Neu dich chua ra het man hinh
           {
           Data_A=Matrix_Temp[shift_i]<<1;// Dich sang trai 1 bit
```

```
        Data_B=Matrix_Colum[(((shift_i+1)/WithB)-1)]>>7;// Dich byte lien
ke cua mang goc sang phai 7 bit
        Matrix_Temp[shift_i]=Data_A|Data_B;// Cong 2 byte lien ke
        Matrix_Colum[(((shift_i+1)/WithB)-
1)]=Matrix_Colum[(((shift_i+1)/WithB)-1)]<<1;// Dich sang trai 1 bit
        }
      else// Neu dich ra het man hinh
        {
        if(first==0 && number==0)// Yeu cau dung giua man hinh
          {
          delay_ms(800);//Tre hien thi
          first=1;
          }
        Matrix_Temp[shift_i]=Matrix_Temp[shift_i]<<1;// Dich sang trai 1
bit
        }
      }
    Bit_Counter++;// Tang so bit da dich duoc
    for(s_i=32;s_i<EndByteB;s_i++)// Chi cho dich phan duoi
      {
      Matrix[s_i]=Matrix_Temp[s_i];// Dua ra man hinh
      }
    Enable_Shift=0;// Tam dung dich
```

```
    }
  else if(Bit_Counter==8)
    {
    Bit_Counter=0;// Xoa
    Colum_Counter++;// Tang so cot da dich duoc
    if(Colum_Counter<With)// Neu chua dich het mang goc
      {
      for(s_i=0;s_i<XB;s_i++)
        {
        if(number==0)
          {
          Matrix_Colum[s_i]=Matrix_G[s_i*With+Colum_Counter];// Copy
cot tiep theo mang goc
          }
        }
      }
    }
}

Colum_Counter=0;
Bit_Counter=0;
first=0;
}
```

```
void shift_up(void)// Ham dich len tren
{
 unsigned int shift_i=0,s_i=0;
 number=1;
 for(s_i=32;s_i<64;s_i++)// Chi cho dich phan duoi
        {
         Matrix[s_i]=0;
        }
 for(shift_i=60, s_i=32;shift_i<64;shift_i++, s_i++)// Chi cho dich phan duoi
        {
         Matrix[shift_i]=Matrix_G[s_i];
        }
 while(Row_Counter<8)
 {
 if(Enable_Shift==1)
 {
  for(shift_i=32;shift_i<60;shift_i++)// Chi cho dich phan duoi
        {
         Matrix[shift_i]=Matrix[shift_i+WithB];// Byte tren copy byte duoi
        }
  for(shift_i=60, s_i=32;shift_i<64;shift_i++, s_i++)// Chi cho dich phan duoi
        {
```

```
        Matrix[shift_i]=Matrix_G[s_i+Row_Counter*WithB];// Copy cot tiep theo mang goc
       }
  Row_Counter++;
  Enable_Shift=0;// Tam dung dich
  }
 }
 number=0;
 Enable_Shift=0;// Tam dung dich
 Row_Counter=0;
 delay_ms(1000);
}
void Caculation_AL(void)// Ham tinh AL
{
if(Year<16)// Neu chua sang nam 2016_Chi lam lich het nam 2015
   {
   //----------------------Tinh Ngay AL_Thang AL----------------------Bat dau tu nam 2011 nen se tru di gia tri 11 khi truy xuat mang AL[]____
   if((AL[(unsigned char)(Year-11)*48+Month*4-3]-Data)>0)// Neu Ngay DL cua ngay AL dau thang - Ngay DL hien tai >0
       {
       Data_AL=(AL[(unsigned char)(Year-11)*48+Month*4-4]+Data)-1;// Ngay AL= Ngay AL dau thang DL + Ngay DL hien tai - 1
```

```
        Month_AL=AL[(unsigned char)(Year-11)*48+Month*4-2];// Thang AL bang thang AL dau thang DL
        }
    else// Neu Ngay DL cua ngay AL dau thang - Ngay DL hien tai <=0
        {
        Data_AL=(Data-AL[(unsigned char)(Year-11)*48+Month*4-3])+1;// Ngay AL= Ngay DL hien tai - Ngay DL cua ngay AL dau thang AL + 1
        Month_AL=AL[(unsigned char)(Year-11)*48+Month*4-1];// Thang AL bang thang AL dau thang AL
        }
    }
}
void matrix_goto_xy(unsigned char xx, unsigned char yy)
{
 ax=xx;
 ay=yy;
}
void display(unsigned char ascii_text)
{
 unsigned char S_bit=0, E_bit=0, offset_ay=0;// Vi tri bit 1 dau tien va cuoi cung cua ki tu
 unsigned char D_temp=0, DD_temp=0, i_temp=0, j_temp=0;
 char D_ax=0;
 bit S_stop=0, E_stop=0;
```

```
 for(i_temp=0; i_temp<8; i_temp++)
 {
  D_temp=D_temp | (text[ascii_text-32].F[i_temp]);;// Cong don 8 byte du lieu tu tren xuong de tim S_bit va E_bit
 }
//-------------------------------------------------------------------------------------------------
 if(D_temp!=0x00)// Neu D_temp ko phai ki tu space
 {
  for(i_temp=0; i_temp<8; i_temp++)
  {
   DD_temp=D_temp & (0x80 >> i_temp);// Kiem tra S_bit. Chieu kiem tra tu phai qua trai
   if(DD_temp && S_stop==0)// Neu co bit = 1 thi do la S_bit
   {
    S_bit=i_temp;// Vi tri bit 1 dau tien cua byte du lieu
    S_stop=1;
   }
   DD_temp=D_temp & (1 << i_temp);// Kiem tra E_bit. Chieu kiem tra tu trai qua phai
   if(DD_temp && E_stop==0)// Neu co bit = 1 thi do la E_bit
   {
    E_bit=((8 - i_temp)-S_bit);// Vi tri bit 1 cuoi cung tien cua byte du lieu
    E_stop=1;
```

```
   }
  }
 offset_ay=1;
 }
 else
 {
  offset_ay=3;// Day la dau space nen tang khoang cach
 }
//-------------------------------------------------------------------------------------
--------------
  if(ax <= XB)// Neu hien thi cho phep qua len tren modul
  {
  for(j_temp=0, D_ax=ax; j_temp<8 && D_ax>0; j_temp++, D_ax--)// Dong
nhat cac byte
   {
    Matrix_Temp[(ax * WithB) - (j_temp * WithB) + (ay >>
3)]=(Matrix_Temp[(ax * WithB) - (j_temp * WithB) + (ay >> 3)]) |
(((text[ascii_text-32].F[7-j_temp])<<S_bit)>>(ay%8));// Chuyen MSB thanh
LSB
    Matrix_Temp[(ax * WithB) - (j_temp * WithB) + (ay >> 3) +
1]=(Matrix_Temp[(ax * WithB) - (j_temp * WithB) + (ay >> 3) + 1]) |
(((text[ascii_text-32].F[7-j_temp])<<S_bit)<<(8 - ay%8));// Chuyen LSB
thanh MSB
   }
```

```
  }
  else// Neu hien thi vuot qua xuong duoi modul
  {
  for(j_temp=0, D_ax=8-(ax-XB); j_temp<8 && D_ax>0; j_temp++, D_ax--
)// Dong nhat cac byte
   {
    Matrix_Temp[(XB * WithB) - (j_temp * WithB) + (ay >>
3)]=(Matrix_Temp[(XB * WithB) - (j_temp * WithB) + (ay >> 3)]) |
(((text[ascii_text-32].F[(8-(ax-XB))-j_temp])<<S_bit)>>((ay%8)));// Chuyen
MSB thanh LSB
    Matrix_Temp[(XB * WithB) - (j_temp * WithB) + (ay >> 3) +
1]=(Matrix_Temp[(XB * WithB) - (j_temp * WithB) + (ay >> 3) + 1]) |
(((text[ascii_text-32].F[7-j_temp])<<S_bit)<<((8 - ay%8)));// Chuyen LSB
thanh MSB
   }
  }
  // Vi tri con tro(ay) sau khi da hien thi tu dong tang theo chieu dai bit LSB
cua byte du lieu truoc do
  ay=(ay + E_bit) + offset_ay;// Gia tri 1 chinh la khoang cach hieu chinh, co
the thay doi gia tri nay de co khoang cach giua cac ki tu mong muon
}
void matrix_clear(void)
{
 unsigned char n_c=0;
 for(n_c=0;n_c<64;n_c++)
```

```
 {
  Matrix[n_c]=0x00;// Xoa man hinh
  Matrix_Temp[n_c]=0x00;
 }
}
void setup_time(void)// Ham cai dat thoi gian
{
 unsigned char n_c=0, D=1, day=0, dat=0, month=0, year=12, hour=0,
minute=0, again=0;
 unsigned char N_1=0,N_2=0,N_3=0,N_4=0, N_5=0, N_6=0;

 matrix_clear();// Xoa man hinh
 matrix_goto_xy(7,8);// Hien thi hang tren
 display('C');
 display('A');
 display('T');
 matrix_goto_xy(15,8);// Hien thi hang duoi
 display('D');
 display('A');
 display('T');
for(n_c=0;n_c<64;n_c++)
  {
   Matrix[n_c]=Matrix_Temp[n_c];// Dua ra man hinh
```

```
  }
delay_ms(1000);// Tre hien thi
//-------------------------------------------------------
 in_mode=0;// Xac nhan vao setup
 matrix_clear();
 delay_ms(100);
 // Hien thi thu 2 dau tien
    matrix_goto_xy(9,5);
    display('T');
    display('H');
    display('U');
    display(' ');
    display(1+1+48);// Cong them 1 la vi tinh tu thu 2
    for(n_c=0;n_c<64;n_c++)
    {
     Matrix[n_c]=Matrix_Temp[n_c];// Dua ra man hinh
    }
//-------------------------------------------------------
 while(in_mode==0)// Cai dat thu trong tuan
 {
  if(in_up==1 || in_down==1)// Neu an tang
  {
  if(in_up==1 && D<7) D++;// Gioi han tren la 6
```

```
else if(in_down==1 && D>1) D--;// Gioi han duoi la 1
in_up=0;
in_down=0;
matrix_clear();
matrix_goto_xy(9,5);
if(D<7)
{
 display('T');
 display('H');
 display('U');
 display(' ');
 display(D+1+48);// Cong them 1 la vi tinh tu thu 2
 day=D;
}
else if(D>=7)// Neu la chu nhat
{
 matrix_goto_xy(9,11);
 display('C');
 display('N');
 day=D;
 D=0;
}
for(n_c=0;n_c<64;n_c++)
```

```
  {
   Matrix[n_c]=Matrix_Temp[n_c];// Dua ra man hinh
  }
  }
 }
write_DS1307(0x03,day);// Luu vao DS1307
//-------------------------------------------------------------
 in_mode=0;// Chuyen sang cai dat ngay-thang-nam
  matrix_clear();
  matrix_goto_xy(7,5);
  N_1=dat/10;// Lay phan nguyen
  N_2=dat%10;// Lay phan du
  N_3=month/10;// Lay phan nguyen
  N_4=month%10;// Lay phan du
  N_5=year/10;// Lay phan nguyen
  N_6=year%10;// Lay phan du

  display(N_1+48);
  display(N_2+48);
  display('-');
  display(N_3+48);
  display(N_4+48);
  matrix_goto_xy(15,7);// Hien thi nam xuong hang duoi
```

```
  display(2+48);// Bat dau tu 2012
  display(0+48);
  display(1+48);
  display(2+48);
  for(n_c=0;n_c<64;n_c++)
  {
   Matrix[n_c]=Matrix_Temp[n_c];// Dua ra man hinh
  }
//-------------------------------------------------------------
 while(in_mode==0 || again<3)// Cai dat ngay-thang-nam
 {
  if(in_up==1 || in_down==1)// Neu thay doi gia tri
  {
  if(in_up==1 && dat<31 && again==0) dat++;// Gioi han tren la ngay 31DL
  else if(in_down==1 && dat>1 && again==0) dat--;// Gioi han duoi la ngay mung 1DL
  else if(in_up==1 && month<11 && again==1) month++;// Gioi han tren la thang 12
  else if(in_down==1 && month>1 && again==1) month--;// Gioi han duoi la thang 1
  else if(in_up==1 && year<99 && again==2) year++;// Gioi han tren la nam 2099
  else if(in_down==1 && year>12 && again==2) year--;// Gioi han duoi la nam 2012
```

```
in_up=0;
in_down=0;
matrix_clear();
matrix_goto_xy(7,5);
N_1=dat/10;// Lay phan nguyen
N_2=dat%10;// Lay phan du
N_3=month/10;// Lay phan nguyen
N_4=month%10;// Lay phan du
N_5=year/10;// Lay phan nguyen
N_6=year%10;// Lay phan du
display(N_1+48);
display(N_2+48);
display('-');
display(N_3+48);
display(N_4+48);
matrix_goto_xy(15,7);// Hien thi nam xuong hang duoi
display(2+48);// 20XX
display(0+48);
display(N_5+48);
display(N_6+48);

for(n_c=0;n_c<64;n_c++)
{
```

```
   Matrix[n_c]=Matrix_Temp[n_c];// Dua ra man hinh
  }
  }
 if(in_mode==1)// Neu yeu cau chuyen sang thang-nam
 {
  again++;
  in_mode=0;
  if(again==3) break;
 }
}
write_DS1307(0x04,Convert_Binary(dat));// Ngay trong thang(1-->31)
write_DS1307(0x05,Convert_Binary(month));// Thang trong nam(1-->12)
write_DS1307(0x06,Convert_Binary(year));// Nam
//-------------------------------------------------------------
  in_mode=0;// Chuyen sang cai dat gio - phut - giay
  again=0;
  matrix_clear();
  matrix_goto_xy(7,5);
  N_1=hour/10;// Lay phan nguyen
  N_2=hour%10;// Lay phan du
  N_3=minute/10;// Lay phan nguyen
  N_4=minute%10;// Lay phan du
  display(N_1+48);
```

```
 display(N_2+48);
 display(':');
 display(N_3+48);
 display(N_4+48);
 for(n_c=0;n_c<64;n_c++)
 {
  Matrix[n_c]=Matrix_Temp[n_c];// Dua ra man hinh
 }
while(in_mode==0 || again<2)// Cai dat gio - phut. Giay mac dinh la 00
{
 if(in_up==1 || in_down==1)// Neu thay doi gia tri
 {
 if(in_up==1 && hour<24 && again==0) hour++;// Gioi han tren la 24h
 else if(in_down==1 && hour>1 && again==0) hour--;// Gioi han duoi la
00h
 else if(in_up==1 && minute<59 && again==1) minute++;// Gioi han tren la
59p
 else if(in_down==1 && minute>1 && again==1) minute--;// Gioi han duoi
la 1p
 in_up=0;
 in_down=0;
 matrix_clear();
 matrix_goto_xy(7,5);
```

```
 N_1=hour/10;// Lay phan nguyen
 N_2=hour%10;// Lay phan du
 N_3=minute/10;// Lay phan nguyen
 N_4=minute%10;// Lay phan du
 display(N_1+48);
 display(N_2+48);
 display(':');
 display(N_3+48);
 display(N_4+48);
 for(n_c=0;n_c<64;n_c++)
 {
  Matrix[n_c]=Matrix_Temp[n_c];// Dua ra man hinh
 }
 }
if(in_mode==1)// Neu yeu cau chuyen sang thang-nam
{
 again++;
 in_mode=0;
 if(again==2) break;
}
}
write_DS1307(0x02,Convert_Binary(hour));// Gio
write_DS1307(0x01,Convert_Binary(minute));// Phut
```

```
write_DS1307(0x00,0x00 & 0x7F);// Giay
in_mode=0;
again=0;
matrix_clear();// Xoa man hinh
matrix_goto_xy(10,7);// Hien thi nam xuong hang duoi
display('O');
display('K');
display(' ');
display('!');
for(n_c=0;n_c<64;n_c++)
  {
   Matrix[n_c]=Matrix_Temp[n_c];// Dua ra man hinh
  }
delay_ms(700);// Tre hien thi
matrix_clear();// Xoa man hinh
}
void timer_1(void) interrupt 3//Ngat dinh thoi timer 1
{
TF1=0;
TH1=0xEC;//chon thoi gian ngat la 1ms
TL1=0x18;
// Place your code here
    Shift_Time++;
```

```
  if(Shift_Time==8 && number==0)
     {
     Shift_Time=0;
     Enable_Shift=1;// Cho phep dich
     }
  else if(Shift_Time==40 && number>0)
     {
     Shift_Time=0;
     Enable_Shift=1;// Cho phep dich
     }
if(mode==0 && time_delay==0)
{
 in_mode=1;// Neu an mode
}
if(up==0 && time_delay==0)
{
 in_up=1;// Neu an up
}
if(down==0 && time_delay==0)
{
 in_down=1;// Neu an down
}
```

```
if((mode==0 || up==0 || down==0) && time_delay==0)// Neu co su kien an nut
{
 enable_delay=1;
 time_delay=90;
}
else if(time_delay>0)
{
 time_delay--;
 if(time_delay==0) enable_delay=0;
}
}
void timer_0(void) interrupt 1 //Ngat dinh thoi timer 0
{
TF0=0;
TH0=0xFC;//chon thoi gian ngat la 1ms
TL0=0x18;// Place your code here
P3_3=1;
  if(t<Dk2)
    {
    for(j=0;j<WithB;j++)
      {
      for(y=0;y<Dk1+1;y=y+step)
```

```
            {
            spi(~Matrix[End4+j-y-t]);
//spi(~XBYTE[End4+j-y-t+baseaddr]);
            }
          }
        P3_3=0;
        latch=1;// Chot
        latch=0;// Chot
        if(m==0)
          {
          LS0=1;
          LS1=1;
          }
        else if(m==1)
          {
          LS0=1;
          LS1=0;
          }
        else if(m==2)
          {
          LS0=0;
          LS1=1;
          }
```

```
      else if(m==3)
        {
        LS0=0;
        LS1=0;
        }
      P3_3=1;
      m++;
      if(m==4)
        {
        m=0;
        }
      t=t+WithB;
      }
    else
      {
      t=0;
      }
      P3_3=0;
}
void main(void)
{
P0=0xFF;
delay_ms(1000);
```

```
init();// Khoi tao timer
init_matrix();
/*
  write_DS1307(0x00,0x00 & 0x7F);// Giay
  delay_ms(100);
  write_DS1307(0x01,0x48);// Phut
  delay_ms(100);
 write_DS1307(0x02,0x22);// Gio
  delay_ms(100);


   write_DS1307(0x03,0x02);// Ngay trong tuan(1-->7)
   delay_ms(100);
   write_DS1307(0x04,0x06);// Ngay trong thang(1-->31)
   delay_ms(100);
   write_DS1307(0x05,0x06);// Thang trong nam(1-->12)
   delay_ms(100);
   write_DS1307(0x06,0x12);// Nam
   delay_ms(100);
*/
while(1)
{
if(in_mode==1)
      {
       matrix_goto_xy(7,6);
```

```
 setup_time();
}
else
{
Read_Time();
Data_Temp=Data;
if(Data==1 && Hour==0 && Minute==1 && Seconds<35 &&
save==0)// Neu la ngay 1 DL dau thang, 0h1p
 {
  write_DS1307(0x00,Convert_Binary(Data+4) & 0x7F);// Bu sai so 4s
  save=1;
 }
 Caculation_AL();
 matrix_goto_xy(7,6);
 Display_Time(Hour,Minute);// Hien thi gio - phut
 matrix_goto_xy(15,4);
 Display_Data(Data_AL,Month_AL);// Hien thi ngay - thang DL
 shift_up();
 matrix_goto_xy(15,5);
 Display_Day(Day);// Hien thi thu trong tuan
 shift_l();
 matrix_goto_xy(15,4);
 Display_Data(Data,Month);// Hien thi ngay - thang DL
 shift_up();
}}}
```

KẾT LUẬN

Cuộc sống con người phát triển ngày càng hiện đại vì vậy thời gian rất quý báu đối với mỗi người. Đồng hồ chính là thước đo thời gian không thể thiếu, vì vậy em chọn đề tài này để nghiên cứu ứng dụng trong thực tiễn cuộc sống, để có thể làm ra những chiếc đồng hồ đa dạng hơn, hiện đại hơn.

Đề tài em lựa chọn làm đồ án tốt nghiệp lần này là điều khiển hiển thị trên modul LED MATRIX P10 một màu loại dùng ngoài trời của Trung Quốc sản xuất. Nhưng do kiến thức còn hạn hẹp nên không thể tránh được những thiếu sót trong quá trình làm đề tài. Em rất mong nhận được những lời chỉ bảo từ thầy cô trong hội đồng.

Từ đề tài này chúng ta có thể phát triển lên một cái đồng hồ đa chức năng. Đồng hồ đã chức năng đó có thể hiển thị Giờ - Phút - Thứ - Ngày - Tháng là cái cơ bản thêm vào đó có cả dương lịch và âm lịch. Ngoài ra có cảm biến nhiệt độ để đo nhiệt độ thời tiết và nhiều chức năng hơn như hẹn giờ, chuông báo giờ…

TÀI LIỆU THAM KHẢO

1. Phạm Quang Trí***, Giáo trình vi xử lý*** – Lý thuyết và thực hành, Trường ĐHCN TP.HCM
2. Tống Văn On (2001) ***, Họ vi điều khiển 8051***, NXB Lao động – Xã hội, Hà Nội
3. Phạm Văn Ất (1999), ***Kỹ thuật lập trình C- cơ sở và nâng cao,*** NXB Khoa học và kỹ thuật.
4. Trang wed www.google.com
5. Trang wed www.alldatasheet.com

MỤC LỤC