TRƯỜNG ĐẠI HỌC HÀNG HẢI VIỆT NAM

KHOA CÔNG NGHỆ THÔNG TIN

-----*-----**

BÁO CÁO BÀI TẬP LỚN

HỌC PHẦN "HỆ THỐNG NHÚNG"

Đề tài:

XÂY DỰNG MA TRẬN HIỂN THỊ CHỮ DÙNG DIODE PHÁT QUANG TRÊN PROTEUS

GVHD:	*Nguyễn Trọng Đức*
Sinh viên thực hiện:	*Nguyễn Quang Hưng - 74423*
	Trần Duy Phong - 75664
	Lê Thị Hoài Phương - 73892
	Lê Thị Hương - 73843

Hải Phòng, tháng 06 năm 2020

TRƯỜNG ĐẠI HỌC HÀNG HẢI

KHOA CÔNG NGHỆ THÔNG TIN

BỘ MÔN KỸ THUẬT PHẦN MỀM

-----*-----**

BÀI TẬP LỚN

HỌC PHẦN: HỆ THỐNG NHÚNG

Mã đề tài: 02

1. Tên đề tài

Xây dựng ma trận hiển thị chữ dụng diot phát quang trên proteus.

2. Mục đích

Phân tích và thiết kế hệ thống ma trận

3. Công việc cần thực hiện

- Tìm hiểu các vấn đề liên quan
- Mô phỏng trên phần mềm Proteus
- Làm báo cáo bài tập lớn
- Bảo vệ bài tập lớn

4. Yêu cầu

 Kết quả làm bài tập lớn: Báo cáo bài tập lớn

5. Tài liệu tham khảo

- Giáo trình "Hệ thống nhúng" – Trường đại học Hàng Hải Việt Nam

Hải Phòng, tháng 06 năm 2020

NGƯỜI HƯỚNG DẪN

Nguyễn Trọng Đức

MỤC LỤC

LỜI NÓI ĐẦU

Ngày nay thế giới đang sống trong kỷ nguyên công nghệ thông tin, nơi mà các hoạt động của công nghệ, kỹ thuật được ứng dụng rộng rãi trên mọi lĩnh vực trong cuộc sống thường nhật như : giao thông công cộng, giáo dục, an ninh,.... Trong xu thế phát triển đó, mọi thứ đều trở nên dần tự động hóa bằng các thiết bị điện tử. Các thiết bị đó dần thay thế mọi hoạt động của con người. Theo dòng phát triển của công nghệ, công nghệ bán dẫn đã và đang phát triển rất mạnh. Thành tựu của nó là các hệ thống nhúng, chúng đóng góp vai trò rất quan trọng trong các hệ thống điều khiển.

Với mục đích nắm hiểu rõ, nắm bắt công nghệ dòng vi xử lý của vi điều khiển và khai thác các ứng dụng của nó trong cuộc sống, từ đó gắn liền đươc lý thuyết với thực tế để thấy được những tính năng ưu việt của dòng vi điều khiển, ta đi tìm hiểu đề tài : **"XÂY DỰNG MA TRẬN HIỂN THỊ CHỮ DÙNG DIODE PHÁT QUANG TRÊN PROTEUS "**

Tuy nhiên do kiến thức chuyên môn còn hạn chế nên còn xảy ra nhiều sai sót. Chúng em mong thầy và các bạn góp ý bổ sung để chúng em có cơ hội tiếp thu và bổ sung kiến thức phục vụ cho quá trình học tập.

Chúng em xin chân thành cảm ơn!

CHƯƠNG 1. PHÂN TÍCH HỆ THỐNG

1.1. KHẢO SÁT VÀ PHÂN TÍCH BÀI TOÁN

a. Giới thiệu về hệ thống

Ngày nay, các bộ vi điều khiển đang có ứng dụng ngày càng rộng rãi trên nhiều lĩnh vực kĩ thuật và đời sống xã hội. Hầu hết các thiết bị từ đơn giản đến phức tạp như: " thiết bị điều khiển tự động, thiết bị văn phòng ...thiết bị trong gia đình" đều có dùng các thiết bị vi

điều khiển. Để thu hút khách hàng thì những đèn LED trang trí, biển hiệu là điều không thể thiếu tại mỗi cửa hàng, siêu thị, trung tâm mua sắm,...

Qua đó, để thấy rõ được ứng dụng của vi điều khiển, nhóm em xin được trình bày về đề tài : "Xây dựng ma trận hiển thị chữ dùng diode phát quang trên proteus."

b. Nhu cầu ý tưởng

Led matrix có rất nhiều công dụng như để chiếu sáng, quảng cáo, trang trí,… Giá thành rẻ, dễ điều khiển, độ bền cao.

Từ vấn đề đặt ra ở trên, chúng em đã nghĩ đến việc áp dụng những gì được học từ môn hệ thống nhúng để thiết kế ra một hệ thống đèn Led matrix có thể hiển thị các chữ cái A-Z.

Ở báo cáo này nhóm em sẽ thực hiện hiển thị một số chữ cái để mô phỏng cho đề tài đã chọn.

c. Mục tiêu đề tài

Bằng kiến thức hiện có, thực hiện mô phỏng bằng công cụ Proteus, thiết kế mạch in cho một đề tài tùy chọn và vận dụng lập trình hợp ngữ, cụ thể là mô phỏng hiển thị kí tự chữ cái trên màn hình led matrix.

Đề tài sẽ giới thiệu rõ về cách ghép nối giữa vi xử lý với các thiết bị ngoại vi, hiểu được cách chọn địa chỉ, cách giải mã địa chỉ và nguyên lý hoạt động của led matrix,… **d. Kết Luận**

Như vậy một sản phẩm hỗ trợ cho việc quảng bá thương hiệu là rất thiết thực và tính khả thi cũng rất cao.

=> Do đó, đề tài "Xây dựng ma trận hiển thị chữ dùng diode phát quang trên proteus" là một để tài vô cùng thực tế và hữu ích.

1.2 . LỰA CHỌN GIẢI PHÁP

1.2.1 . Giải pháp công nghệ

Qua phân tích trên , nhóm chúng em đưa ra giải pháp xây dựng hệ thống ma trận : hiển thị các ký tự trong bảng mã ASCII bằng matrix 8x8.

1.2.2. Giải pháp thiết kế

Đầu vào hệ thống lấy dữ liệu từ 74HC595 là thời gian cho nhiệm cung cấp dữ liệu cho hệ thống , để có thể in ra được ký tự trên matrix.

Xử lý , điều khiển dùng vi điều khiển AT89C52, lập trình vi điều khiển để xử lý độc ghi ký tự, tính toán hiển thị ký tự .

Để hiển thị dùng matrix 8x8 : lấy dữ liệu ra từ vi điều khiển thông báo dữ liệu : chữ cái, ký tự,…..

1.2.3 . Xác định bài toán và giới hạn của đề tài

Hệ thống đồng hồ thời gian thực hiển thị lịch âm dương :

Đầu vào hệ thống lấy dữ liệu từ 74HC595 , là IO thời gian thực nhằm cung cấp dữ liệu cho hệ thống .

Đầu ra được hiển thị trên Matrix 8x8 .

Hiển thị chính xác các ký tự được truyền vào như chữ cái, số liệu,…. Làm việc với điện áp cấp từ pin

Làm việc được lâu dài và ổn định

Quan sát dễ dàng , có thể quan sát được dữ liệu và khoảng cách tương đối xa .

CHƯƠNG 2. THIẾT KẾ HỆ THỐNG

1.1. CÁC MODULE TRONG HỆ THỐNG .

1.1.1 . Module khối nguồn .

Đây là module cấp nguồn cho hệ thống nhằm Cung cấp điện áp chuẩn + 5V .

Yêu cầu đối với khối này :

Có thể lấy nguồn từ điện áp xoay chiều (hoặc ping đế cấp nguồn cho hệ thống .

Điện áp đầu ra của khối (điện áp đầu vào của hệ thống luôn ổn định tại mọi thời điểm. Mạch ổn áp cần cho vị điều khiển vì nếu nguồn cho vi điều khiển không ổn định thì sẽ treo VĐK , không chạy đúng hoặc reset liên tục thậm chí là chết chíp . Với yêu cầu như trên ta lựa chọn mạch biến đổi điện áp xoay chiều thành 1 chiều qua máy biến thể . Sử dụng IC7805 để ổn áp .

1.1.2 . Module Reset

Khối RESET có tác dụng đưa vi điều khiển về trạng thái ban đầu . Khi nút Reset được ẩn điện áp + 5V từ nguồn được nối vào chân Reset của vi điều khiển được chạy thang xuống đất lúc này điện áp tại chân vì điều khiển thay đổi đột ngột về 0 , VĐK nhận biết được sự thay đổi này và khai động lại trạng thái ban đầu cho hệ thống .

1.1.3 . Module điều khiển .

Module dùng để điều khiển hệ thống . Yêu cầu :

Thiết lập được cho vi điều khiển là thể làm việc bình thường hay chuyển sang chế độ cài đặt thời gian

Điều chỉnh tang (giảng thời gian , ngày tháng năm Kết thúc việc thiết lập thời gian và điều hình thời gian bằng một nút điều khiển .

Vậy ta lựa chọn bộ điều khiển gồm 4 nút ấn .

1.1.4 . Module tạo xung giao động .

Đây là bộ dao động thạch anh có tác dụng tạo xung nhịp với tần số 12MHz cho VĐK hoạt động . Hai đầu này được nối vào 2chân XTALL và XTAL2 của VĐK.

1.1.5 . Module hiển thị .

Khối Hiển thị : Lấy tín hiệu ra từ chân IC để hiển thị thời gian , ngày Tháng năm. Khối hiển thị yêu cầu :

Sử dụng nguồn chung toàn hệ thống hoặc có thể dùng nguồn riêng tùy người thiết kế .

Độ sáng đủ lớn để có thể quan sát được trong phạm vi trong phòng góc nhìn rộng màu sắc của số khi hiển thị dề quan sát , Lựa chọn hiển thị qua led 7 thanh , màu đỏ

1.1.6 . Module xử lý .

Dùng VDK để lấy tín hiệu từ khối dieu khiển , tạo xung giao động , tạo thời gian thực . và đưa ra khối hiển thị. Yêu cầu:

Tốc độ xử lý nhanh , chính xác .

Bộ nhớ không cần lớn .

Lựa chọn vị điều khiển AT89C52 .

2.2. LỰA CHỌN LINH KIỆN

2.2.1. Vi Điều Khiển AT89C52

a) Cầu tạo và chức năng các khối của AT89C52 .

CPU (CPU central processing unto 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 bằng 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 .
  - 3 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 thế nạp các chương trình cho chíp mà không cần các bộ nạp chuyên dụng).
- Bộ chia tần số với hệ số chia là 12 .
- 4 cổng xuất nhập với 32 chân .

b) Chức năng các chân của AT89C52

- Port 0 gồm 8 chân , ngoài chức năng xuất nhập , port 1 còn là bus đa hợp dữ liệu và địa chỉ AD0-AD7, chức năng này sẽ được sử dụng khi 89c52 giao tiếp với các thiết bị ngoại có kiến trúc Bus như các vi mạch nhớ , mạch PIO ...
- Port 1 (P1.0 APP1.7q : Chức năng duy nhất của Port 1 là chức năng xuất nhập cũng như các Port khác . Portl có thể xuất nhập theo bit và theo byte .
- Port 2 (P2.0- > P2.7q ; Port 2 ngoài chức năng là cổng vào / ra như Port 0 và 1 còn là byte cao của bus địa chi khi sử dụng bộ nhớ ngoài .
- Port 3 : Mọi chân trên Port 3 ngoài chức năng xuất nhập còn có một chức năng riêng.

Chân PSEN : là chân điều khiển đọc chương trình bộ nhớ ngoài .

Chân ALE : ALE là tín hiệu điều khiển chốt địa chỉ có tần số bằng 1/6 tần số dao động của vi điều khiển . Tín hiệu ALE được dùng để cho phép vi mạch chốt bên ngoài như 7473

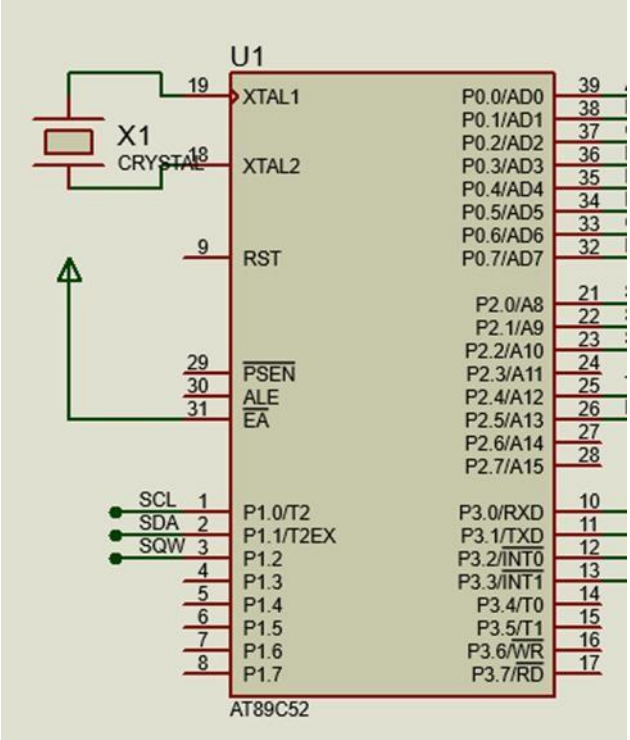
Chân / EA : Tín hiệu / EA cho phép chọn bộ nhớ chương trình là bộ nhớ trong hay ngoài . EA A1 thì thực hiện chương trình trong RAM nội . EA A0thực hiện RAM ngoài .

RST (reset: Ngõ vào reset trên chân số 9. khi RST AI thì bộ vi điều khiến sẽ được khại động lại thiết lập ban đầu .

XTAL1 , XTAL2 : 2 chân này được nối song song với thạch anh tần số max 33 Mhz . Để tạo dao động cho bộ vi điều khiển .

Vcc , GND : cung cấp nguồn nuôi cho bộ vi điều khiển . cấp qua chân 20 và 40 .

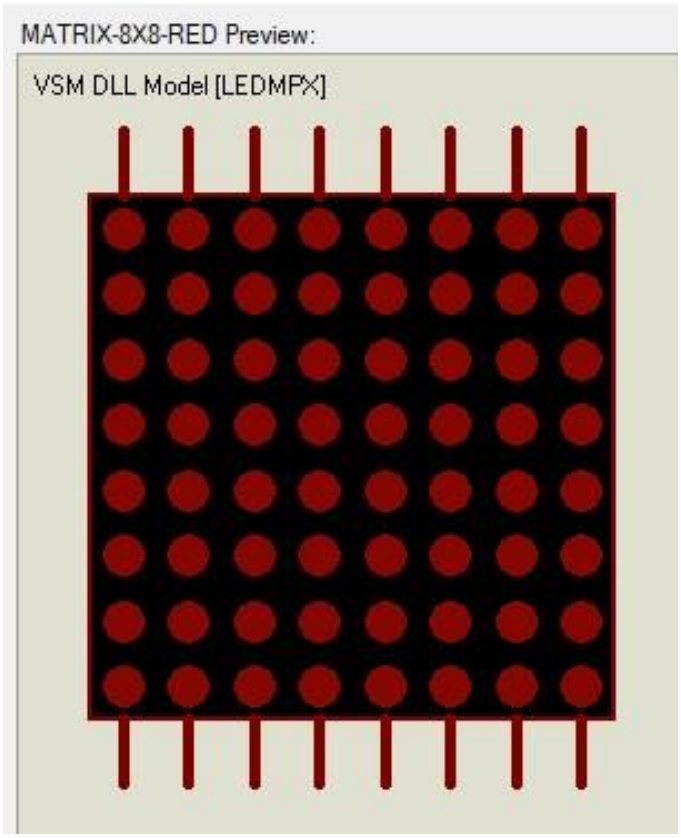
c) Sơ đồ AT89C52 trong mạch

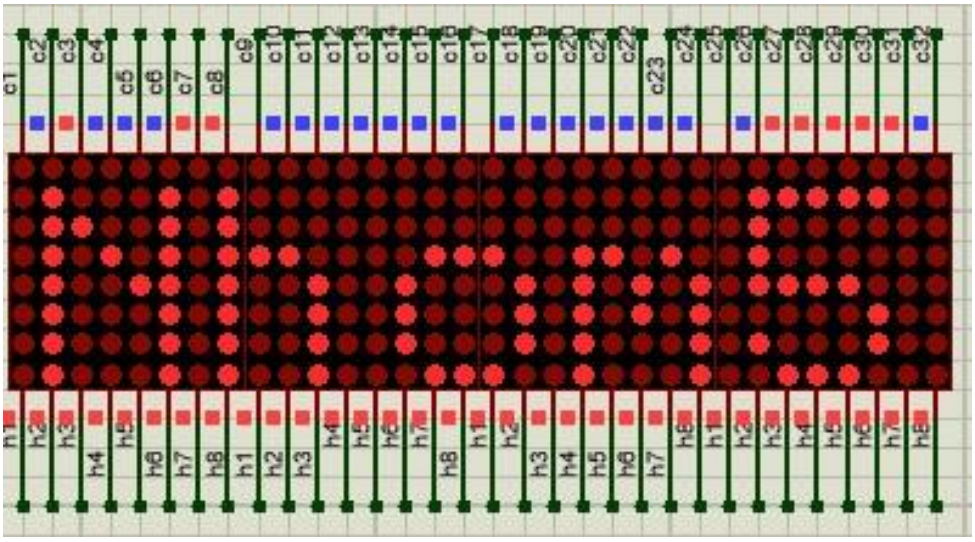


2.2.2. Matrix 8x8

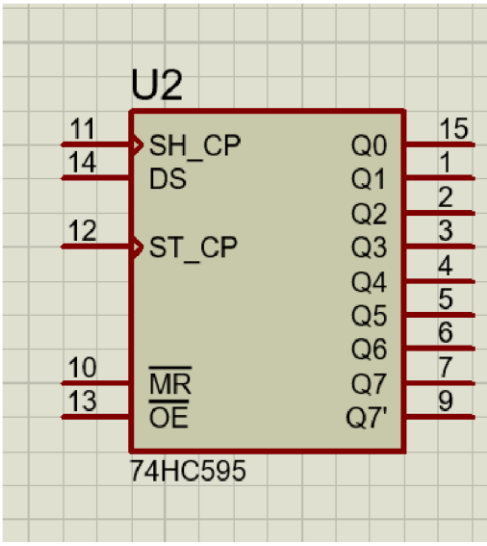
a) Các khái niệm cơ bản :

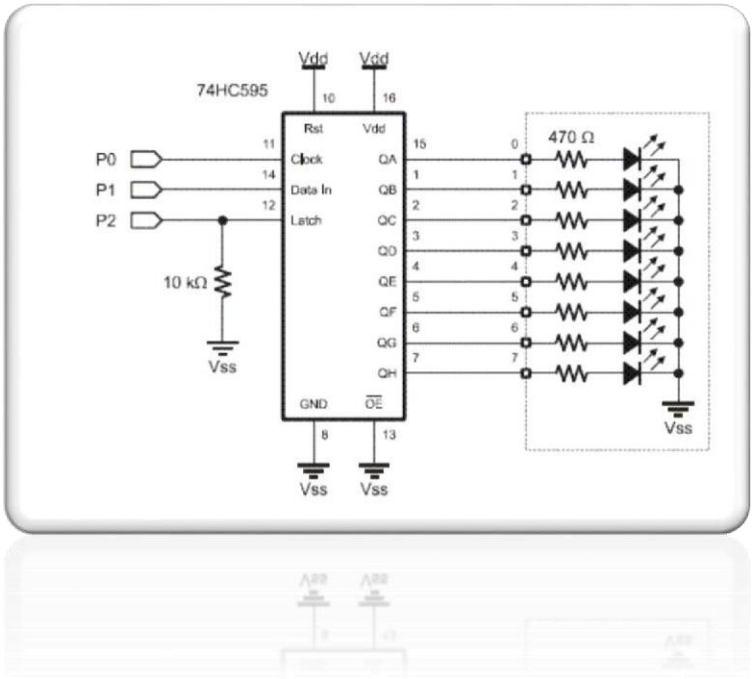
Ma trận LED gồm 8 cột và 8 hàng. Tại mỗi điểm giao giữa các hàng và các cột có gắn vào đó một con Diode phát quang. Nên do đó phải đưa tín hiệu vào hàng ở mức 1 và cột ở mức 0 thì LED mới sáng được.


MATRIX-8X8-RED Preview:
VSM DLL Model [LEDMPX]



2.2.3. IC 74HC595.





IC 74HC595 còn được gọi là IC dịch chốt với mối quan hệ "vào nối tiếp và ra song song 8 bit". Để thực sự hiểu rỏ IC này chúng ta sẽ phải trả lời một số câu hỏi cơ bản nhất của vấn đề. Trước tiên, ***thế nào là dịch và chốt?*** Nói một chút lý thuyết kỹ thuật số về 2 thuật ngữ dịch và chốt trước khi tìm hiểu về 595 để ta nắm chắc vấn đề hơn.

a. Thế nào là dịch và chốt?

Dịch là gì?

Thuật ngữ "Dịch" được dùng cho IC này thực chất là cách nói của dân kỹ thuật về đặt tính ngõ vào nối tiếp của IC. Vậy "đặt tính ngõ vào nối tiếp của IC" là như thế nào? Có thể nói 1 cách đơn giản là ta có thể đưa lần lượt nhiều giá trị logic vào 1 ngõ vào của IC đó để nó lưu vào bộ nhớ chờ đến khi có lệnh xử lí. Có thể hiểu ngõ vào nối tiếp tức là các giá trị được nạp vào IC một cách từ từ ở từng thời điểm khác nhau trên cùng 1 ngõ vào, còn nếu nói vào song song thì có nghĩa là sẽ có nhiều ngõ vào và các giá trị sẽ được nạp vào IC cùng 1 lúc. Thuật ngữ này đối với ngõ ra cũng tương tự!

Chốt là gì?

- Chốt là khi thoa mản 1 điều kiện nào đó nó sẽ cho phép IC giữ nguyên giá trị ngõ ra, không cho nó thay đổi mặc dù tín hiệu ngõ vào có thay đổi thế nào.

=> Vậy tới đây chúng ta cũng đã hình dung được phần nào chức năng của 1 IC dịch chốt rồi nhé. Bây giờ ta sẽ tìm hiểu cụ thể vấn đề với IC 74HC595. b. Tìm Hiểu Cơ Bản

Các chân từ 1 tới 7 và chân số 15 là ngõ ra của IC (ứng với Q0 , Q1 ,…,Q7)

Chân DS (chân số 14) là ngõ vào của IC (đây là IC vào nối tiếp nên ta chỉ cần 1 ngõ vào là đủ).

Chân 16 - VCC là chân cấp nguồn dương (từ 2V đến 6V)

=> Chân số 8 GND là chân cấp Ground – cực (-) của nguồn.

Chân SHCP là chân đưa xung clock (xung nhịp) vào IC và khi có cạnh lên của xung thì IC đưa tín hiệu ở ngõ vào vào bộ nhớ của IC để chờ xử lý. Có thể hơi khó hiểu khi đọc đến đây nhưng các bạn đừng lo lắng, vì đây là muốn các bạn tập làm quen với các thuật ngữ kỹ thuật, cứ đọc để biết là có nó trên đời. Sau đây là phần giải thích:

Đầu tiên, thế nào là xung clock, thực ra xung clock là 1 chuỗi tín hiệu logic 0 và 1 có thể là 1 xen kẽ với 0 cũng có thể là 0,1 ngẫu nhiên nhưng nói chung nó là 1 chuỗi tín hiệu logic.

Còn cạnh lên và cạnh xuống của xung thì các bạn có thể thấy trên hình, cạnh lên là khi xung clock chuyển trạng thái từ 0 lên 1, còn cạnh xuống là thời điểm khi chuyển từ 1 xuống 0.

Vậy khi có cạnh lên của xung tại chân **SHCP** thì 1 tín hiệu logic từ ngõ vào của IC sẽ được lưu trữ vào trong IC để chờ tín hiệu cho phép xử lý. Bộ nhớ tối đa của IC là 8 bit, nếu

vượt quá ngưỡng này thì giá trị mới sẽ được đưa vào IC và đồng thời giá trị cũ nhất của IC sẽ được xoá đi.

Chân STCP là chân đưa xung clock vào IC để khi có cạnh lên của xung thì IC đưa toàn bộ 8bit data đã được lưu (đã nói ở chân SHCP) ra ngõ ra của IC.

Chân MR| là chân reset IC (tức là trả IC về trạng thái ban đầu – giống như khi ta ghost máy tính vậy – khi chân này tích cực thì toàn bộ bộ nhớ của IC sẽ bị xoá tất cả bằng 0, tuy nhiên lưu ý là lúc này tín hiệu ở ngõ ra không bị xoá mà vẫn giữ nguyên giá trị trước đó) và chân này tích cực mức thấp (LOW active) có nghĩa là muốn reset IC thì phải đưa 0V vào chân này.

Chân EO| là chân Output Enable chân khi được tích cực thì mới cho phép ta điều chỉnh được giá trị ngõ ra. Khi tên chân IC mà có dấu gạch trên đầu tức là nó tích cực thấp (LOW active) tức là muốn tích cực chân này thì ta phải đưa 0v (GND) vào chân này. Còn nếu khi chân này không được tích cực (tức là đưa mức logic 1 vào chân này thì ngõ ra bị đưa lên trạng thái trở kháng cao – chỉ cần biết là vậy sau này các bạn sẽ được rõ hơn về trạng thái trở kháng cao – cao siêu lắm ^^).

Bây giờ chúng ta đã đủ kiến thức để tìm hiểu chân số 9 chân **Q7S**. Chữ S ở đây là viết tắt cho từ Serial (nối tiếp) chân này thường được dùng khi ta nối tiếp các IC 74HC595 với nhau (chân Q7S của con trước nối vào chân DS của con sau) chân này sẽ có giá trị của bit trọng số cao của bộ nhớ IC (Bit mới được đưa vào sẽ nằm ở vị trí LSB – trong số thấp) nếu mắc nối tiếp các IC 74HC595 lại với nhau theo cách như vậy thì khi bit MSB bị đẩy ra khỏi bộ nhớ của IC sẽ không mất đi mà trước đó nó đã được sao chép qua IC phía sau.

Nguyên lý:

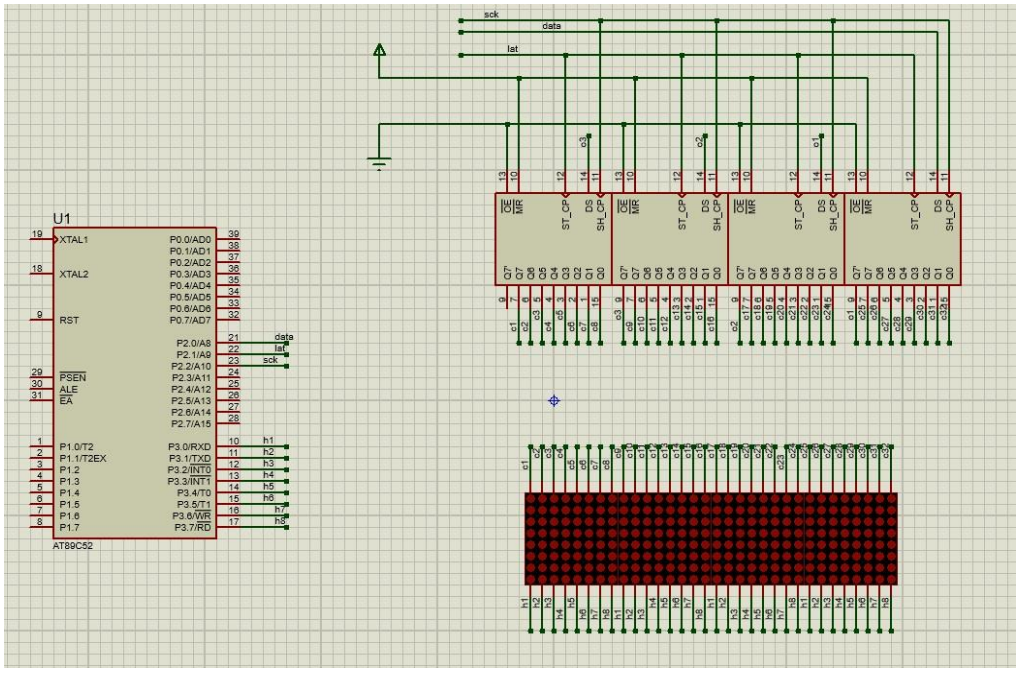
Chú ý là ta đang mở rộng ngõ ra 16bit nên khi nạp dữ liệu các bạn cũng phải nạp vào 16 bit. như ta thấy để chỉ led **D1** sáng các led còn lại đều tắt, tương ứng dữ liệu ngõ ra là

1000 0000 0000 0000. Nhưng khi nạp thì ta phải nạp lần lượt giá trị là 0000 0000 0000 0001. Vì 595 sẽ truyền bit đầu tiên là LSB (bit có trọng số thấp nhất).

Đầu tiên ta cho **DATA** đang ở mức 0, ta sẽ tạo 1 xung tại chân SHCP để nạp bit 0 đầu tiên vào 595 bằng cách nhấn thả nút **BUTTON1**. Lúc này 595 lưu trử giá trị bit 0 đầu tiên. Tiếp tục ta lại nạp thêm 1 bit 0 thứ 2 vào 595 cũng bằng cách nhấn thả nút **BUTTON 1** thêm lần nữa. Lúc này 595 đang lưu giá trị là 00. Và ta lại tiếp tục nạp lần lượt thêm bit 0 thứ 3, thứ 4.... thứ 15 vào IC bằng cách nhấn nút **BUTTON 1** 13 lần (chú ý **DATA** luôn ở mức 0). Lúc này IC đang lưu 15bit là 0000 0000 0000 000. Và bit thứ 16 chúng ta cần nạp là bit 1, bằng cách đưa **DATA** lên mức 1, sau đó nhấn nút **BUTTON 1** để nạp bit 1 đó vào IC.

Lúc này IC đã lưu đủ 16bit là 00000000 00000001. Để đưa dữ liệu ra ngõ ra của 595 và chốt lại, ta cần cấp 1 xung tại chân STCP, bằng cách nhấn thả nút BUTTON 2. Lúc này ta sẽ thấy led D1 sáng và các led còn lại đều tắt.

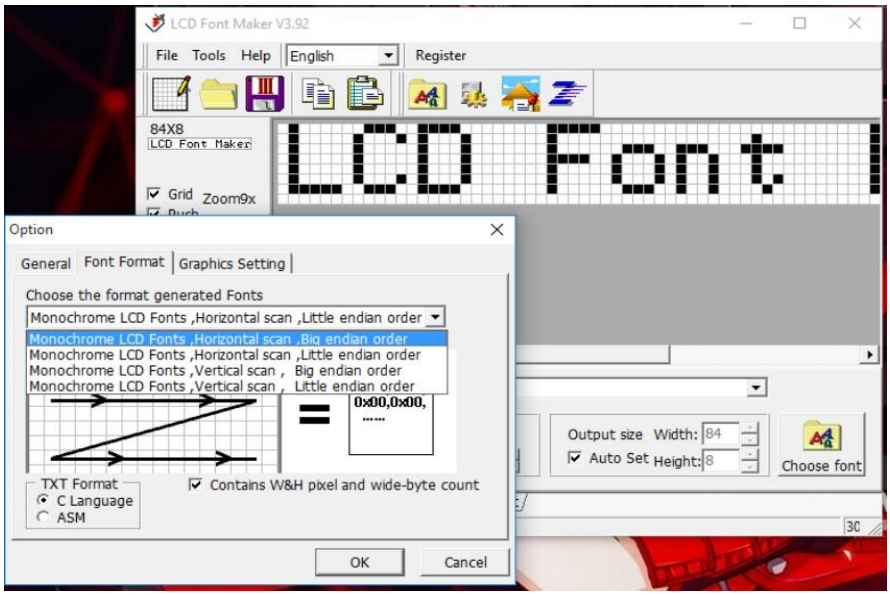
Sơ Đồ Nguyên Lý



3.2 Xây dựng bộ mã ASCII

Chúng ta sẽ lại dùng phần mềm LCD font maker để tạo font

Lưu ý: Các chân có trọng số cao (Q7 -> Q0) nối với các led từ trái sang phải. Do vậy led bên trái sẽ có trọng số cao hơn bên phải. Hãy setup ở phần mềm LCD font maker sao cho pixel bên trái là bit cao hơn.



Như vậy chúng ta sẽ tạo được bộ font theo đúng thứ tự của mã ASCII như sau: unsigned char code font[][7]=

```
{
        0x00,0x00,0x00,0x00,0x00,0x00,0x00, //  32
        0x10,0x38,0x38,0x10,0x10,0x00,0x10, //! 33
        0x6C,0x6C,0x48,0x00,0x00,0x00,0x00, //" 34
        0x00,0x28,0x7C,0x28,0x28,0x7C,0x28, //# 35
        0x20,0x38,0x40,0x30,0x08,0x70,0x10, //$ 36
        0x64,0x64,0x08,0x10,0x20,0x4C,0x4C, //% 37
        0x20,0x50,0x50,0x20,0x54,0x48,0x34, //& 38
        0x30,0x30,0x20,0x00,0x00,0x00,0x00, //' 39
        0x10,0x20,0x20,0x20,0x20,0x20,0x10, //( 40
        0x20,0x10,0x10,0x10,0x10,0x10,0x20, //) 41
        0x00,0x28,0x38,0x7C,0x38,0x28,0x00, //* 42
        0x00,0x10,0x10,0x7C,0x10,0x10,0x00, //+ 43
        0x00,0x00,0x00,0x00,0x00,0x30,0x30, //, 44
        0x00,0x00,0x00,0x7C,0x00,0x00,0x00, //- 45
        0x00,0x00,0x00,0x00,0x00,0x30,0x30, //. 46
        0x00,0x04,0x08,0x10,0x20,0x40,0x00, /// 47
        0x38,0x44,0x4C,0x54,0x64,0x44,0x38, //0 48
        0x00,0x04,0x08,0x10,0x20,0x40,0x00, //1 49
        0x38,0x44,0x04,0x18,0x20,0x40,0x7C, //2 50
        0x38,0x44,0x04,0x38,0x04,0x44,0x38, //3 51
        0x08,0x18,0x28,0x48,0x7C,0x08,0x08, //4 52
        0x7C,0x40,0x40,0x78,0x04,0x44,0x38, //5 53
        0x18,0x20,0x40,0x78,0x44,0x44,0x38, //6 54
        0x7C,0x04,0x08,0x10,0x20,0x20,0x20, //7 55
        0x38,0x44,0x44,0x38,0x44,0x44,0x38, //8 56
```

```
0x38,0x44,0x44,0x3C,0x04,0x08,0x30, //9 57
0x00,0x00,0x30,0x30,0x00,0x30,0x30, //: 58
0x00,0x00,0x30,0x30,0x00,0x30,0x30, //; 59
0x08,0x10,0x20,0x40,0x20,0x10,0x08, //< 60
0x00,0x00,0x7C,0x00,0x00,0x7C,0x00, //= 61
0x20,0x10,0x08,0x04,0x08,0x10,0x20, //> 62
0x38,0x44,0x04,0x18,0x10,0x00,0x10, //? 63
0x38,0x44,0x5C,0x54,0x5C,0x40,0x38, //@ 64
0x38,0x44,0x44,0x44,0x7C,0x44,0x44, //A 65
0x78,0x44,0x44,0x78,0x44,0x44,0x78, //B 66
0x38,0x44,0x40,0x40,0x40,0x44,0x38, //C 67
0x78,0x44,0x44,0x44,0x44,0x44,0x78, //D 68
0x7C,0x40,0x40,0x78,0x40,0x40,0x7C, //E 69
0x7C,0x40,0x40,0x78,0x40,0x40,0x40, //F 70
0x38,0x44,0x40,0x5C,0x44,0x44,0x3C, //G 71
0x44,0x44,0x44,0x7C,0x44,0x44,0x44, //H 72
0x38,0x10,0x10,0x10,0x10,0x10,0x38, //I 73
0x04,0x04,0x04,0x04,0x44,0x44,0x38, //J 74
0x44,0x48,0x50,0x60,0x50,0x48,0x44, //K 75
0x40,0x40,0x40,0x40,0x40,0x40,0x7C, //L 76
0x44,0x6C,0x54,0x44,0x44,0x44,0x44, //M 77
0x44,0x64,0x54,0x4C,0x44,0x44,0x44, //N 78
0x38,0x44,0x44,0x44,0x44,0x44,0x38, //O 79
0x78,0x44,0x44,0x78,0x40,0x40,0x40, //P 80
0x38,0x44,0x44,0x44,0x54,0x48,0x34, //Q 81
0x78,0x44,0x44,0x78,0x48,0x44,0x44, //R 82
0x38,0x44,0x40,0x38,0x04,0x44,0x38, //S 83
0x7C,0x10,0x10,0x10,0x10,0x10,0x10, //T 84
0x44,0x44,0x44,0x44,0x44,0x44,0x38, //U 85
```

0x44,0x44,0x44,0x44,0x44,0x28,0x10, //V 86

0x44,0x44,0x54,0x54,0x54,0x54,0x28, //W 87

0x44,0x44,0x28,0x10,0x28,0x44,0x44, //X 88

0x44,0x44,0x44,0x28,0x10,0x10,0x10, //Y 89

0x78,0x08,0x10,0x20,0x40,0x40,0x78, //Z 90

0x38,0x20,0x20,0x20,0x20,0x20,0x38, //[91

0x00,0x40,0x20,0x10,0x08,0x04,0x00, //\ 92

0x38,0x08,0x08,0x08,0x08,0x08,0x38, //] 93

0x10,0x28,0x44,0x00,0x00,0x00,0x00, //^ 94

0x00,0x00,0x00,0x00,0x00,0x00,0x7C, //_ 95

0x30,0x30,0x10,0x00,0x00,0x00,0x00, //` 96

0x00,0x00,0x38,0x04,0x3C,0x44,0x3C, //a 97

0x40,0x40,0x78,0x44,0x44,0x44,0x78, //b 98

0x00,0x00,0x38,0x44,0x40,0x44,0x38, //c 99

0x04,0x04,0x3C,0x44,0x44,0x44,0x3C, //d 100

0x00,0x00,0x38,0x44,0x78,0x40,0x38, //e 101

0x18,0x20,0x20,0x78,0x20,0x20,0x20, //f 102

0x00,0x3C,0x44,0x44,0x3C,0x04,0x38, //g 103

0x40,0x40,0x70,0x48,0x48,0x48,0x48, //h 104

0x10,0x00,0x10,0x10,0x10,0x10,0x18, //i 105

0x08,0x00,0x18,0x08,0x08,0x48,0x30, //j 106

0x40,0x40,0x48,0x50,0x60,0x50,0x48, //k 107

0x20,0x20,0x20,0x20,0x20,0x20,0x38, //l 108

0x00,0x00,0x68,0x54,0x54,0x44,0x44, //m 109

0x00,0x00,0x70,0x48,0x48,0x48,0x48, //n 110

0x00,0x00,0x38,0x44,0x44,0x44,0x38, //o 111

0x00,0x78,0x44,0x44,0x44,0x78,0x40, //p 112

0x00,0x3C,0x44,0x44,0x44,0x3C,0x04, //q 113

0x00,0x00,0x58,0x24,0x20,0x20,0x70, //r 114

0x00,0x00,0x38,0x40,0x38,0x04,0x38, //s 115

0x00,0x20,0x78,0x20,0x20,0x28,0x10, //t 116

0x00,0x00,0x48,0x48,0x48,0x58,0x28, //u 117

0x00,0x00,0x44,0x44,0x44,0x28,0x10, //v 118

0x00,0x00,0x44,0x44,0x54,0x7C,0x28, //w 119

0x00,0x00,0x48,0x48,0x30,0x48,0x48, //x 120

0x00,0x48,0x48,0x48,0x38,0x10,0x60, //y 121

0x00,0x00,0x78,0x08,0x30,0x40,0x78, //z 122

0x18,0x20,0x20,0x60,0x20,0x20,0x18, //{ 123

0x10,0x10,0x10,0x10,0x10,0x10,0x10, //| 124

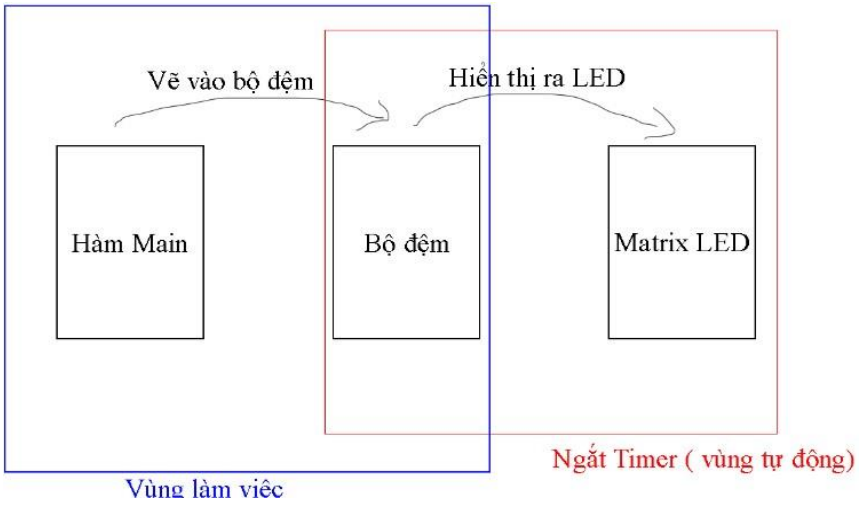
0x30,0x08,0x08,0x0C,0x08,0x08,0x30, //} 125

0x00,0x00,0x28,0x50,0x00,0x00,0x00, //~ 126

}

3.3 Xây Dựng module ngắt hiển thị dữ liệu từ bộ đệm

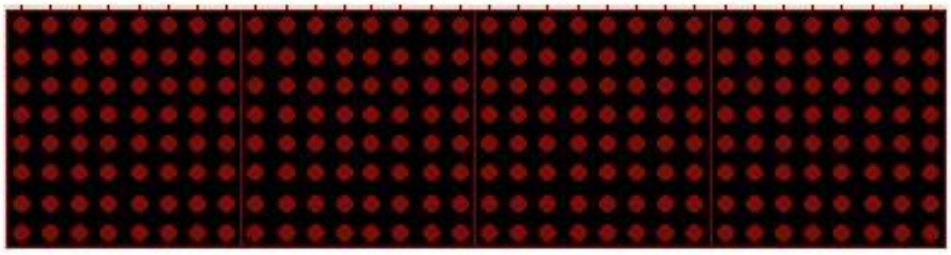
3.3.1 Mối quan hệ giữa bộ đệm và LED

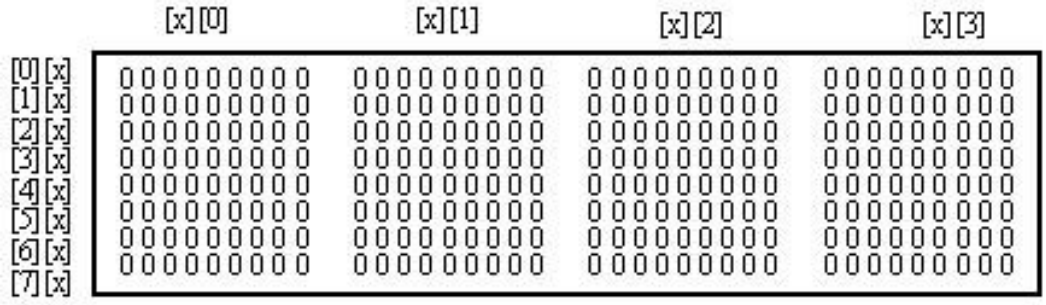


3.3.2 Khởi tạo bộ đệm

Do có 8×32 con LED nên sẽ cần 8×32 bit dữ liệu cho bộ đệm. Mà cứ 8 bit là 1 byte nên ta cần 32byte cho bộ đệm. Chúng ta sẽ tiến hành khởi tạo với mảng 2 chiều dựa theo sơ đồ dưới đây:

unsigned char Buffer_display[8][4];





Bộ đệm hiển thị Buffer_display[8][4]

3.3.3 Xây dựng hàm ngắt để truyền dữ liệu

Như ta đã biết chỗ để truyền dữ liệu ra là 4 con ic74hc595 nối tiếp nên trước tiên dựng 1 hàm đẩy dữ liệu ra HC595

```
#define DAT P2_1
#define LAT P2_2
#define SCK P2_3
void hc595_out(unsigned char byte)
{
    int i;
    for(i=0;i<8;i++)
    {
        DAT = byte & (0x80>>i);
        SCK=0;SCK=1;
    }
}
```

Hàm ngắt (Timer) để đẩy dữ liệu từ bộ đệm ra

```
void T1_ISR() interrupt 3 // Dung timer 1 de quet led matrix
{
 static unsigned char z ; // Bien dem
 P3=0xFF;              //tat het LED
 hc595_out(Buffer_display[z][0]); //lay du lieu trong bo dem hien thi ra man hinh matrix
 hc595_out(Buffer_display[z][1]); //lay du lieu trong bo dem hien thi ra man hinh matrix
 hc595_out(Buffer_display[z][2]); //lay du lieu trong bo dem hien thi ra man hinh matrix
 hc595_out(Buffer_display[z][3]); //lay du lieu trong bo dem hien thi ra man hinh matrix
 LAT=0;LAT=1; //chot du lieu
 P3=maquet[z];         //cho phep LED sang
 z++;if(z==8)z=0;
 TH1=0xFC; // Nap gia tri cho TH1
 TL1=0x17; // Nap gia tri cho TL1
}
```

3.3.4 Xây dựng hàm truyền dữ liệu vào bộ đệm

```
void Matrix_guikitu(int vitri,unsigned char txt)
{
      int i;
      for(i=0;i<7;i++)
      {
          Buffer_display[i][vitri] = font[txt-32][i];
      }
}
```

Hàm này yêu cầu nạp vào biến txt chính là kí tự mà ta muốn in ra, vòng lặp for tới 7 vì mỗi kí tự được tạo với font gồm 7 byte data.

Chúng ta phải trừ txt đi 32 vì mảng font của ta bắt đầu từ 0 nhưng mã ascii lại bắt đầu từ 32. Do vậy phải trừ đi để 2 font này trùng khớp nhau.

Biến vitri ở đây để xác định tấm matrix (bắt đầu từ 0) sẽ được truyền dữ liệu ra.

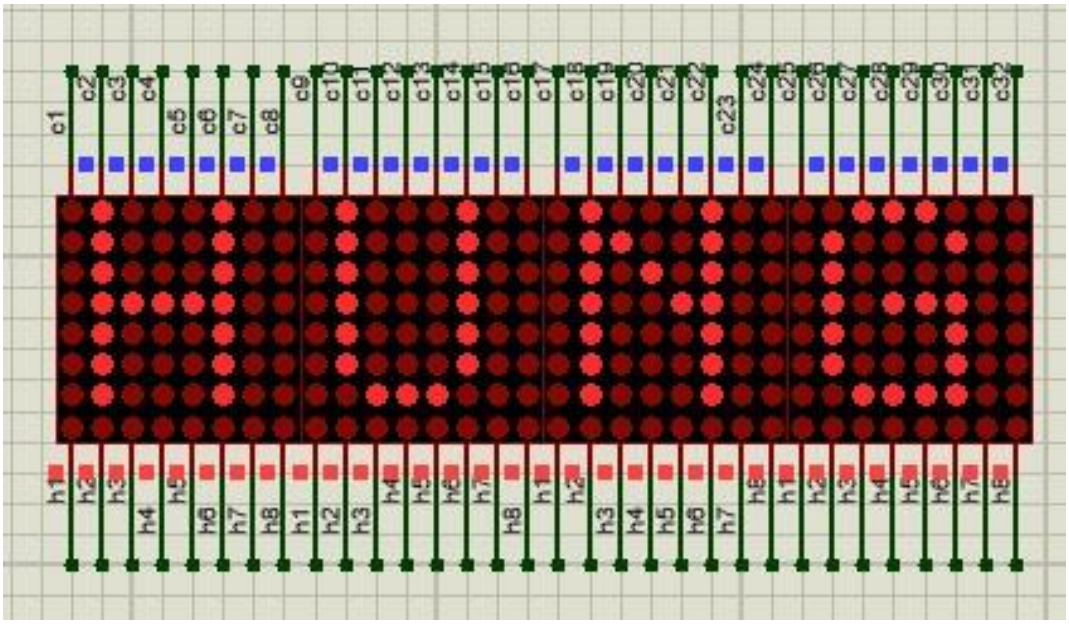
Qua đó chúng ta có thể xây dựng 1 hàm gửi chuỗi ký tự như sau:

```
void Matrix_guichuoi(unsigned char *s)
{
    int vitri=0;
    while(*s)
        {
            Matrix_guikitu(vitri++,*s);
            if(vitri==4)return;
            s++;
        }
}
```

- Có thể hiểu hàm trên là hàm gửi 1 ký tự nhưng lặp lại nhiều lần, vòng lặp while sẽ lặp cho tới khi gặp NULL thì thoát.
- Mỗi lần truyền dữ liệu ra biến vitri đẩy sang tấm matrix bên cạnh.

- Sẽ đặt 1 if để kiểm tra khi in đủ 4 ký tự sẽ tự động thoát.

Ví Dụ: Matrix_guichuoi(“HUNG”);



Full Code:

```
#include <REGX52.H>

#define DAT P2_0
#define LAT P2_1 #define SCK P2_2 unsigned char code maquet[8]={0xFE,0xFD,0xFB,0xF7,0xEF,0xDF,0xBF,0x7F}; unsigned char code font[][7]= {
        0x00,0x00,0x00,0x00,0x00,0x00,0x00, //  32
        0x10,0x38,0x38,0x10,0x10,0x00,0x10, //! 33
        0x6C,0x6C,0x48,0x00,0x00,0x00,0x00, //" 34
        0x00,0x28,0x7C,0x28,0x28,0x7C,0x28, //# 35
        0x20,0x38,0x40,0x30,0x08,0x70,0x10, //$ 36
        0x64,0x64,0x08,0x10,0x20,0x4C,0x4C, //% 37
        0x20,0x50,0x50,0x20,0x54,0x48,0x34, //& 38
        0x30,0x30,0x20,0x00,0x00,0x00,0x00, //' 39
        0x10,0x20,0x20,0x20,0x20,0x20,0x10, //( 40
        0x20,0x10,0x10,0x10,0x10,0x10,0x20, //) 41
        0x00,0x28,0x38,0x7C,0x38,0x28,0x00, //* 42
        0x00,0x10,0x10,0x7C,0x10,0x10,0x00, //+ 43
        0x00,0x00,0x00,0x00,0x00,0x30,0x30, //, 44
```

```
0x00,0x00,0x00,0x7C,0x00,0x00,0x00, //- 45
0x00,0x00,0x00,0x00,0x00,0x30,0x30, //. 46
0x00,0x04,0x08,0x10,0x20,0x40,0x00, /// 47
0x38,0x44,0x4C,0x54,0x64,0x44,0x38, //0 48
0x00,0x04,0x08,0x10,0x20,0x40,0x00, //1 49
0x38,0x44,0x04,0x18,0x20,0x40,0x7C, //2 50
0x38,0x44,0x04,0x38,0x04,0x44,0x38, //3 51
0x08,0x18,0x28,0x48,0x7C,0x08,0x08, //4 52
0x7C,0x40,0x40,0x78,0x04,0x44,0x38, //5 53
0x18,0x20,0x40,0x78,0x44,0x44,0x38, //6 54
0x7C,0x04,0x08,0x10,0x20,0x20,0x20, //7 55
0x38,0x44,0x44,0x38,0x44,0x44,0x38, //8 56
0x38,0x44,0x44,0x3C,0x04,0x08,0x30, //9 57
0x00,0x00,0x30,0x30,0x00,0x30,0x30, //: 58
0x00,0x00,0x30,0x30,0x00,0x30,0x30, //; 59
0x08,0x10,0x20,0x40,0x20,0x10,0x08, //< 60
0x00,0x00,0x7C,0x00,0x00,0x7C,0x00, //= 61
0x20,0x10,0x08,0x04,0x08,0x10,0x20, //> 62
0x38,0x44,0x04,0x18,0x10,0x00,0x10, //? 63
0x38,0x44,0x5C,0x54,0x5C,0x40,0x38, //@ 64
0x38,0x44,0x44,0x44,0x7C,0x44,0x44, //A 65
0x78,0x44,0x44,0x78,0x44,0x44,0x78, //B 66
0x38,0x44,0x40,0x40,0x40,0x44,0x38, //C 67
0x78,0x44,0x44,0x44,0x44,0x44,0x78, //D 68
0x7C,0x40,0x40,0x78,0x40,0x40,0x7C, //E 69
0x7C,0x40,0x40,0x78,0x40,0x40,0x40, //F 70
0x38,0x44,0x40,0x5C,0x44,0x44,0x3C, //G 71
0x44,0x44,0x44,0x7C,0x44,0x44,0x44, //H 72
0x38,0x10,0x10,0x10,0x10,0x10,0x38, //I 73
0x04,0x04,0x04,0x04,0x44,0x44,0x38, //J 74
0x44,0x48,0x50,0x60,0x50,0x48,0x44, //K 75
0x40,0x40,0x40,0x40,0x40,0x40,0x7C, //L 76
0x44,0x6C,0x54,0x44,0x44,0x44,0x44, //M 77
0x44,0x64,0x54,0x4C,0x44,0x44,0x44, //N 78
0x38,0x44,0x44,0x44,0x44,0x44,0x38, //O 79
0x78,0x44,0x44,0x78,0x40,0x40,0x40, //P 80
0x38,0x44,0x44,0x44,0x54,0x48,0x34, //Q 81
0x78,0x44,0x44,0x78,0x48,0x44,0x44, //R 82
0x38,0x44,0x40,0x38,0x04,0x44,0x38, //S 83
0x7C,0x10,0x10,0x10,0x10,0x10,0x10, //T 84
```

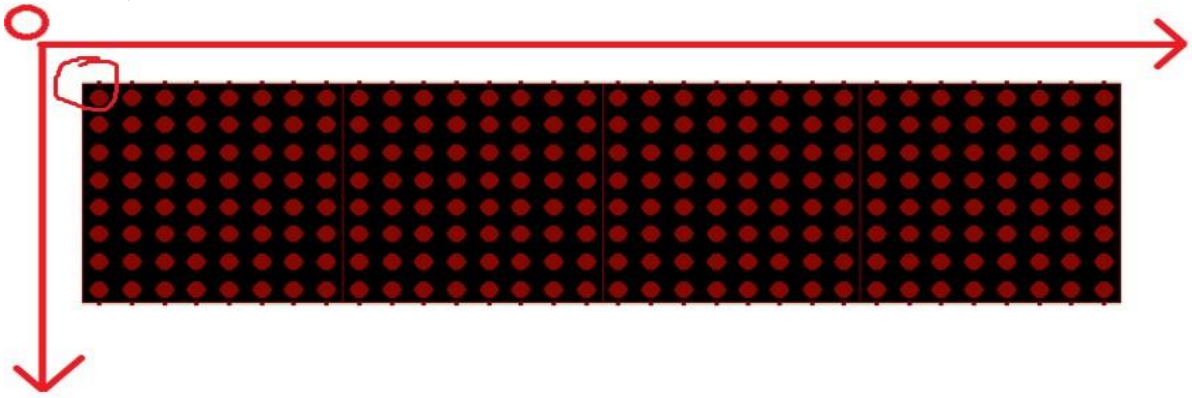
```
0x44,0x44,0x44,0x44,0x44,0x44,0x38, //U 85
0x44,0x44,0x44,0x44,0x44,0x28,0x10, //V 86
0x44,0x44,0x54,0x54,0x54,0x54,0x28, //W 87
0x44,0x44,0x28,0x10,0x28,0x44,0x44, //X 88
0x44,0x44,0x44,0x28,0x10,0x10,0x10, //Y 89
0x78,0x08,0x10,0x20,0x40,0x40,0x78, //Z 90
0x38,0x20,0x20,0x20,0x20,0x20,0x38, //[ 91
0x00,0x40,0x20,0x10,0x08,0x04,0x00, //\ 92
0x38,0x08,0x08,0x08,0x08,0x08,0x38, //] 93
0x10,0x28,0x44,0x00,0x00,0x00,0x00, //^ 94
0x00,0x00,0x00,0x00,0x00,0x00,0x7C, //_ 95
0x30,0x30,0x10,0x00,0x00,0x00,0x00, //` 96
0x00,0x00,0x38,0x04,0x3C,0x44,0x3C, //a 97
0x40,0x40,0x78,0x44,0x44,0x44,0x78, //b 98
0x00,0x00,0x38,0x44,0x40,0x44,0x38, //c 99
0x04,0x04,0x3C,0x44,0x44,0x44,0x3C, //d 100
0x00,0x00,0x38,0x44,0x78,0x40,0x38, //e 101
0x18,0x20,0x20,0x78,0x20,0x20,0x20, //f 102
0x00,0x3C,0x44,0x44,0x3C,0x04,0x38, //g 103
0x40,0x40,0x70,0x48,0x48,0x48,0x48, //h 104
0x10,0x00,0x10,0x10,0x10,0x10,0x18, //i 105
0x08,0x00,0x18,0x08,0x08,0x48,0x30, //j 106
0x40,0x40,0x48,0x50,0x60,0x50,0x48, //k 107
0x20,0x20,0x20,0x20,0x20,0x20,0x38, //l 108
0x00,0x00,0x68,0x54,0x54,0x44,0x44, //m 109
0x00,0x00,0x70,0x48,0x48,0x48,0x48, //n 110
0x00,0x00,0x38,0x44,0x44,0x44,0x38, //o 111
0x00,0x78,0x44,0x44,0x44,0x78,0x40, //p 112
0x00,0x3C,0x44,0x44,0x44,0x3C,0x04, //q 113
0x00,0x00,0x58,0x24,0x20,0x20,0x70, //r 114
0x00,0x00,0x38,0x40,0x38,0x04,0x38, //s 115
0x00,0x20,0x78,0x20,0x20,0x28,0x10, //t 116
0x00,0x00,0x48,0x48,0x48,0x58,0x28, //u 117
0x00,0x00,0x44,0x44,0x44,0x28,0x10, //v 118
0x00,0x00,0x44,0x44,0x54,0x7C,0x28, //w 119
0x00,0x00,0x48,0x48,0x30,0x48,0x48,          //x          120
0x00,0x48,0x48,0x48,0x38,0x10,0x60,          //y          121
0x00,0x00,0x78,0x08,0x30,0x40,0x78, //z 122
0x18,0x20,0x20,0x60,0x20,0x20,0x18, //{ 123
0x10,0x10,0x10,0x10,0x10,0x10,0x10, //| 124
```

```
            0x30,0x08,0x08,0x0C,0x08,0x08,0x30, //} 125
            0x00,0x00,0x28,0x50,0x00,0x00,0x00, //~ 126
};
unsigned  char  Buffer_display[8][4];
void delay(unsigned int t) //hàm delay{
unsigned int x,y;       for(x=0;x<t;x++){
for(y=0;y<123;y++);
       }
  }
void hc595_out(unsigned char byte){
         int i;
         for(i=0;i<8;i++){
                  DAT = byte & (0x80>>i);
                  SCK=0;SCK=1;
         }
} void T1_ISR() interrupt 3 // Dung timer 1 de quet led
matrix
{
 static unsigned char z ; // Bien dem  P3=0xFF;              //tat het LED
hc595_out(Buffer_display[z][0]); //lay du lieu trong bo dem hien thi ra man hinh matrix  1
hc595_out(Buffer_display[z][1]); //lay du lieu trong bo dem hien thi ra man hinh matrix  2
hc595_out(Buffer_display[z][2]); //lay du lieu trong bo dem hien thi ra man hinh matrix  3
hc595_out(Buffer_display[z][3]); //lay du lieu trong bo dem hien thi ra man hinh matrix  4
 LAT=0;LAT=1; //chot du lieu
 P3=maquet[z];         //cho phep LED sang
z++;if(z==8)z=0;
 TH1=0xFC; // Nap gia tri cho TH1
 TL1=0x17; // Nap gia tri cho TL1
} void Matrix_guikitu(int vitri,unsigned char
txt){
```

```
        int i;
for(i=0;i<7;i++){
                Buffer_display[i][vitri] = font[txt-32][i];
        }
} void Matrix_guichuoi(unsigned char
*s){   int vitri=0;   while(*s){
                Matrix_guikitu(vitri++,*s);
        if(vitri==4)
                        return;
                        s++;
        }
} void
main()
{
 TMOD=0x10; // khoi tao ngat T1, 16bit
 ET1=1;     // cho phep ngat T1
 TF1=0;     // xoa co ngat T1
 TR1=1;     // khoi dong T1
 EA = 1;    // cho phep ngat toan cuc  Matrix_guichuoi("HUNG");  while(1){}}
```

3.4 Mở Rộng (điều khiển matrix bằng tọa độ xOy)

3.4.1 Truyền dữ liệu và biểu diễn dữ liệu



Đầu tiên chúng ta sẽ khai báo hai biến toàn cục

MatrixX, MatrixY

Viết 1 hàm để set vị trí con trỏ màn hình

```
void Matrix_chonvitri(int x,int y)
{
    MatrixX=x;
    MatrixY=y;
}
```

Tiếp theo chúng ta sẽ tạo 1 hàm để tắt hoặc bật led trên matrix dựa theo tọa độ:

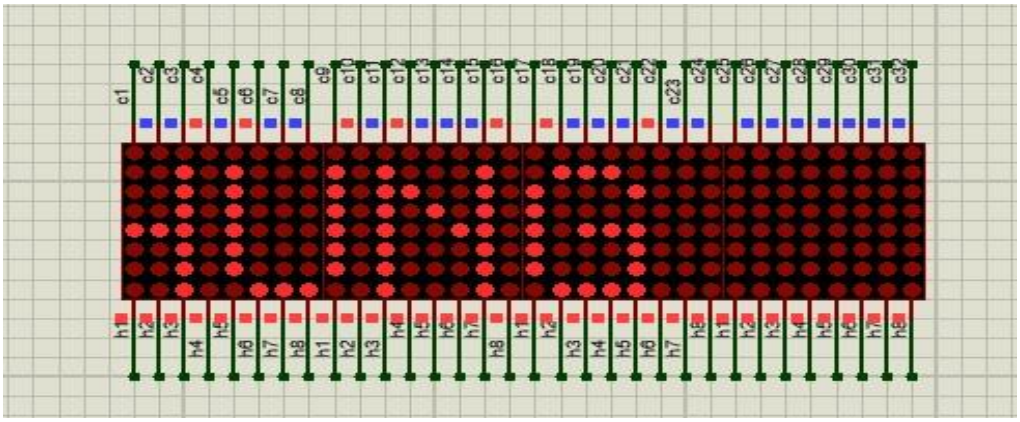
```
void Matrix_setpx(int x,int y,char color)
{
        if(x>31 || y>7 || x<0 || y < 0)return;
    if(color)Buffer_display[y][x/8] |= (0x80 >> (x%8));
    else Buffer_display[y][x/8] &= ~(0x80 >> (x%8));
}
```

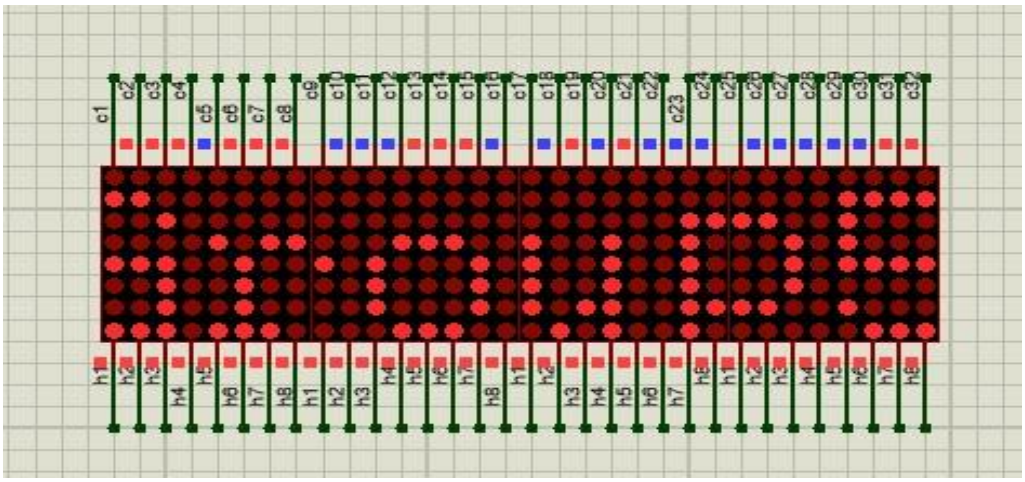
Câu lệnh if(x>31 || y>7 || x<0 || y < 0)return; nghĩa là nếu ký tự ta nhập vào vượt ra ngoài matrix thì sẽ tự động thoát .

Tương tự hàm gửi kí tự sẽ được sửa lại như sau:

```
void Matrix_guikitu(unsigned char txt)
{
    int x,y;
    for(y=MatrixY;y<7+MatrixY;y++)
    {
        for(x=MatrixX;x<6+MatrixX;x++)
        {
            if ( (font[txt-32][y-MatrixY] & (0x80>>(x-MatrixX)))  != 0)Matrix_setpx(x,y,1);
            else Matrix_setpx(x,y,0);
        }
    }
    MatrixX+=6;
}
```

Ví Dụ: Khi mình để tọa độ là (-3,0) và ký tự được truyền vào là HUNG dựa vào tọa độ ta sẽ được kết quả:





=> Như vậy thông qua việc dung tọa độ xOy chúng ta có thể truyền vào nhiều ký tự và có thể nâng cấp lên cho ký tự chuyển động từ mọi phía.

Full Code:

```
#include <REGX52.H>

#define DAT P2_0

#define LAT P2_1

#define SCK P2_2 int MatrixX,MatrixY; unsigned char code
maquet[8]={0xFE,0xFD,0xFB,0xF7,0xEF,0xDF,0xBF,0x7F}; unsigned char
code font[][7]= {
        0x00,0x00,0x00,0x00,0x00,0x00,0x00, //  32
        0x10,0x38,0x38,0x10,0x10,0x00,0x10, //! 33
        0x6C,0x6C,0x48,0x00,0x00,0x00,0x00, //" 34
        0x00,0x28,0x7C,0x28,0x28,0x7C,0x28, //# 35
        0x20,0x38,0x40,0x30,0x08,0x70,0x10, //$ 36
```

```
0x64,0x64,0x08,0x10,0x20,0x4C,0x4C, //% 37
0x20,0x50,0x50,0x20,0x54,0x48,0x34, //& 38
0x30,0x30,0x20,0x00,0x00,0x00,0x00, //' 39
0x10,0x20,0x20,0x20,0x20,0x20,0x10, //( 40
0x20,0x10,0x10,0x10,0x10,0x10,0x20, //) 41
0x00,0x28,0x38,0x7C,0x38,0x28,0x00, //* 42
0x00,0x10,0x10,0x7C,0x10,0x10,0x00, //+ 43
0x00,0x00,0x00,0x00,0x00,0x30,0x30, //, 44
0x00,0x00,0x00,0x7C,0x00,0x00,0x00, //- 45
0x00,0x00,0x00,0x00,0x00,0x30,0x30, //. 46
0x00,0x04,0x08,0x10,0x20,0x40,0x00, /// 47
0x38,0x44,0x4C,0x54,0x64,0x44,0x38, //0 48
0x00,0x04,0x08,0x10,0x20,0x40,0x00, //1 49
0x38,0x44,0x04,0x18,0x20,0x40,0x7C, //2 50
0x38,0x44,0x04,0x38,0x04,0x44,0x38, //3 51
0x08,0x18,0x28,0x48,0x7C,0x08,0x08, //4 52
0x7C,0x40,0x40,0x78,0x04,0x44,0x38, //5 53
0x18,0x20,0x40,0x78,0x44,0x44,0x38, //6 54
0x7C,0x04,0x08,0x10,0x20,0x20,0x20, //7 55
0x38,0x44,0x44,0x38,0x44,0x44,0x38, //8 56
0x38,0x44,0x44,0x3C,0x04,0x08,0x30, //9 57
0x00,0x00,0x30,0x30,0x00,0x30,0x30, //: 58
0x00,0x00,0x30,0x30,0x00,0x30,0x30, //; 59
0x08,0x10,0x20,0x40,0x20,0x10,0x08, //< 60
0x00,0x00,0x7C,0x00,0x00,0x7C,0x00, //= 61
0x20,0x10,0x08,0x04,0x08,0x10,0x20, //> 62
0x38,0x44,0x04,0x18,0x10,0x00,0x10, //? 63
0x38,0x44,0x5C,0x54,0x5C,0x40,0x38, //@ 64
0x38,0x44,0x44,0x44,0x7C,0x44,0x44, //A 65
0x78,0x44,0x44,0x78,0x44,0x44,0x78, //B 66
0x38,0x44,0x40,0x40,0x40,0x44,0x38, //C 67
0x78,0x44,0x44,0x44,0x44,0x44,0x78, //D 68
0x7C,0x40,0x40,0x78,0x40,0x40,0x7C, //E 69
0x7C,0x40,0x40,0x78,0x40,0x40,0x40, //F 70
0x38,0x44,0x40,0x5C,0x44,0x44,0x3C, //G 71
0x44,0x44,0x44,0x7C,0x44,0x44,0x44, //H 72
0x38,0x10,0x10,0x10,0x10,0x10,0x38, //I 73
0x04,0x04,0x04,0x04,0x44,0x44,0x38, //J 74
0x44,0x48,0x50,0x60,0x50,0x48,0x44, //K 75
0x40,0x40,0x40,0x40,0x40,0x40,0x7C, //L 76
```

```
0x44,0x6C,0x54,0x44,0x44,0x44,0x44, //M 77
0x44,0x64,0x54,0x4C,0x44,0x44,0x44, //N 78
0x38,0x44,0x44,0x44,0x44,0x44,0x38, //O 79
0x78,0x44,0x44,0x78,0x40,0x40,0x40, //P 80
0x38,0x44,0x44,0x44,0x54,0x48,0x34, //Q 81
0x78,0x44,0x44,0x78,0x48,0x44,0x44, //R 82
0x38,0x44,0x40,0x38,0x04,0x44,0x38, //S 83
0x7C,0x10,0x10,0x10,0x10,0x10,0x10, //T 84
0x44,0x44,0x44,0x44,0x44,0x44,0x38, //U 85
0x44,0x44,0x44,0x44,0x44,0x28,0x10, //V 86
0x44,0x44,0x54,0x54,0x54,0x54,0x28, //W 87
0x44,0x44,0x28,0x10,0x28,0x44,0x44, //X 88
0x44,0x44,0x44,0x28,0x10,0x10,0x10, //Y 89
0x78,0x08,0x10,0x20,0x40,0x40,0x78, //Z 90
0x38,0x20,0x20,0x20,0x20,0x20,0x38, //[ 91
0x00,0x40,0x20,0x10,0x08,0x04,0x00, //\ 92
0x38,0x08,0x08,0x08,0x08,0x08,0x38, //] 93
0x10,0x28,0x44,0x00,0x00,0x00,0x00, //^ 94
0x00,0x00,0x00,0x00,0x00,0x00,0x7C, //_ 95
0x30,0x30,0x10,0x00,0x00,0x00,0x00, //` 96
0x00,0x00,0x38,0x04,0x3C,0x44,0x3C, //a 97
0x40,0x40,0x78,0x44,0x44,0x44,0x78,          //b          98
0x00,0x00,0x38,0x44,0x40,0x44,0x38,          //c          99
0x04,0x04,0x3C,0x44,0x44,0x44,0x3C, //d 100
0x00,0x00,0x38,0x44,0x78,0x40,0x38, //e 101
0x18,0x20,0x20,0x78,0x20,0x20,0x20, //f 102
0x00,0x3C,0x44,0x44,0x3C,0x04,0x38, //g 103
0x40,0x40,0x70,0x48,0x48,0x48,0x48, //h 104
0x10,0x00,0x10,0x10,0x10,0x10,0x18, //i 105
0x08,0x00,0x18,0x08,0x08,0x48,0x30, //j 106
0x40,0x40,0x48,0x50,0x60,0x50,0x48, //k 107
0x20,0x20,0x20,0x20,0x20,0x20,0x38, //l 108
0x00,0x00,0x68,0x54,0x54,0x44,0x44, //m 109
0x00,0x00,0x70,0x48,0x48,0x48,0x48, //n 110
0x00,0x00,0x38,0x44,0x44,0x44,0x38, //o 111
0x00,0x78,0x44,0x44,0x44,0x78,0x40, //p 112
0x00,0x3C,0x44,0x44,0x44,0x3C,0x04, //q 113
0x00,0x00,0x58,0x24,0x20,0x20,0x70, //r 114
0x00,0x00,0x38,0x40,0x38,0x04,0x38, //s 115
0x00,0x20,0x78,0x20,0x20,0x28,0x10, //t 116
```

```
            0x00,0x00,0x48,0x48,0x48,0x58,0x28, //u 117
            0x00,0x00,0x44,0x44,0x44,0x28,0x10, //v 118
            0x00,0x00,0x44,0x44,0x54,0x7C,0x28, //w 119
            0x00,0x00,0x48,0x48,0x30,0x48,0x48, //x 120
                  0x00,0x48,0x48,0x48,0x38,0x10,0x60, //y 121
                  0x00,0x00,0x78,0x08,0x30,0x40,0x78, //z 122
                  0x18,0x20,0x20,0x60,0x20,0x20,0x18, //{ 123
            0x10,0x10,0x10,0x10,0x10,0x10,0x10, //| 124
            0x30,0x08,0x08,0x0C,0x08,0x08,0x30, //} 125
            0x00,0x00,0x28,0x50,0x00,0x00,0x00, //~ 126
}; unsigned char Buffer_display[8][4];
void delay(unsigned int t) //hàm
delay{      unsigned int x,y;
for(x=0;x<t;x++){
for(y=0;y<123;y++);
}
  }
void hc595_out(unsigned char byte){
        int i;
        for(i=0;i<8;i++){
                DAT = byte & (0x80>>i);
                SCK=0;SCK=1;
        }
} void T1_ISR() interrupt 3 // Dung timer 1 de quet led
matrix
{
 static unsigned char z ; // Bien dem  P3=0xFF;              //tat het LED
hc595_out(Buffer_display[z][0]); //lay du lieu trong bo dem hien thi ra man hinh matrix  1
hc595_out(Buffer_display[z][1]); //lay du lieu trong bo dem hien thi ra man hinh matrix  2
hc595_out(Buffer_display[z][2]); //lay du lieu trong bo dem hien thi ra man hinh matrix  3
hc595_out(Buffer_display[z][3]); //lay du lieu trong bo dem hien thi ra man hinh matrix  4
 LAT=0;LAT=1; //chot du lieu
```

```
 P3=maquet[z];        //cho phep LED sang
z++;if(z==8)z=0;
 TH1=0xFC; // Nap gia tri cho TH1
 TL1=0x17; // Nap gia tri cho TL1
}
void Matrix_chonvitri(int x,int y){
        MatrixX=x;
        MatrixY=y;
} void Matrix_setpx(int x,int y,char color){   if(x>31 ||
y>7 || x<0 || y < 0)return;
        if(color)Buffer_display[y][x/8] |= (0x80 >>
(x%8)); else Buffer_display[y][x/8] &= ~(0x80 >>
(x%8));
} void Matrix_guikitu(unsigned char txt){      int x,y;
for(y=MatrixY;y<7+MatrixY;y++){
for(x=MatrixX;x<6+MatrixX;x++){                    if ( (font[txt-32][y-
MatrixY] & (0x80>>(x-MatrixX)))  !=
0)Matrix_setpx(x,y,1);
                        else Matrix_setpx(x,y,0);
                }
         }
         MatrixX+=6; //sau khi in xong thi tang con tro vi tri len
} void Matrix_guichuoi(unsigned char
*s){
        while(*s){
                Matrix_guikitu(*s);
                s++;
        }
```

```
} void
main()
{
 TMOD=0x10; // khoi tao ngat T1, 16bit
 ET1=1;     // cho phep ngat T1
 TF1=0;     // xoa co ngat T1
 TR1=1;     // khoi dong T1
 EA = 1;    // cho phep ngat toan cuc
 Matrix_chonvitri(0,1);
 Matrix_guichuoi("Group5");
while(1){}
}
```

CHƯƠNG 4: KẾT LUẬN

Về cơ bản, nhóm đã hoàn thành việc phân tích và thiết kế hệ thống ma trận hiển thị chữ dùng diot phát quang trên proteus, cụ thể

- *Về mặt lí thuyết :*
  - Tìm hiểu các vấn đề liên quan
  - Nắm bắt được các linh kiện có trong hệ thống, cách thức hoạt động của các linh kiện
  - Mô phỏng thành công trên phần mềm Proteus
- *Các chương trình được sử dụng :*
  - Proteus 8.8 – Xây dựng ma trận hiển thị chữ dùng diot phát quang trên proteus - Keil Uvision 5 – Lập trình vi điều khiển *Hạn chế :*
  - Thời gian thực hiện ngắn, còn nhiều thiếu sót
  - Đôi chỗ còn chưa hoàn chỉnh
- *Hướng phát triển*
  - Hoàn thành những phần còn thiếu
  - Khắc phục thiếu sót