

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM**

**Giới thiệu về IC dịch HC595**

**TT Kiến Trúc Và Tổ Chức Máy Tính**

**GVHD: Huỳnh Hoàng Hà**

**Mã học phần: COOL325364\_23\_1\_11**

**Ngày, tiết học: Thứ sáu tiết 1-5**

**SVTH: Nguyễn Nhật Trí**

**MSSV: 21139064**

**TP. Thủ Đức - 11/2023**

1. **Giới thiệu:**

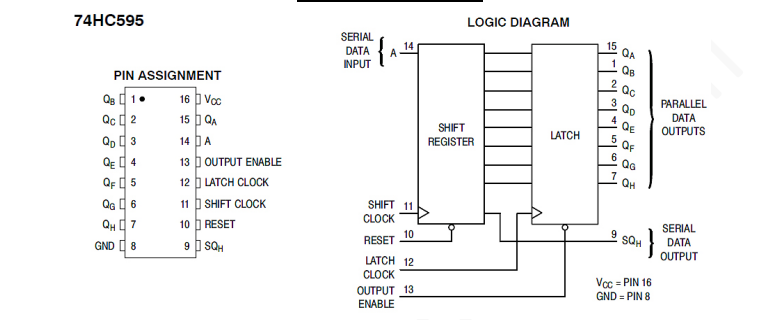
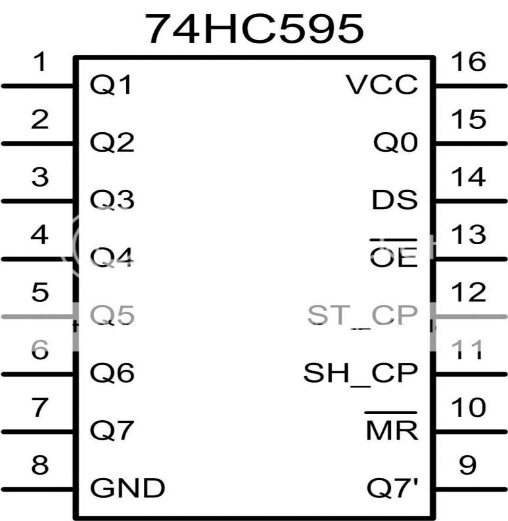
74HC595 là một thanh ghi dịch (shift register) hoạt động trên giao thức nối tiếp vào song song ra (Serial IN Parallel OUT).

Nó nhận dữ liệu nối tiếp từ vi điều khiển và sau đó gửi dữ liệu này qua các chân song song.

Có thể tăng 8 chân đầu ra bằng cách sử dụng chip đơn.

Cũng có thể kết nối song song nhiều hơn 1 thanh ghi dịch.

Giả sử đã kết nối ba thanh ghi dịch với bộ vi điều khiển, các chân đầu ra được tăng lên 8 x 3 = 24.



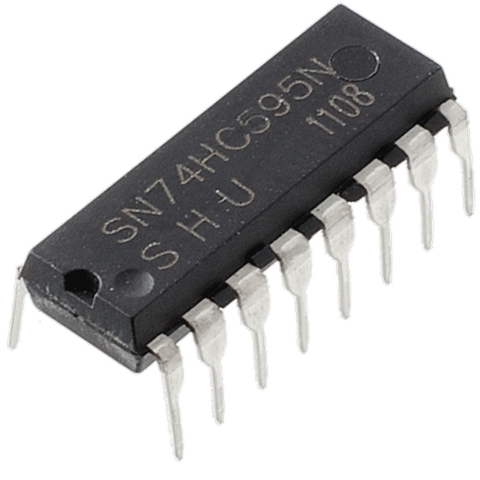
Hình 1 – sơ đồ chân của 74HC595. Hình 2 – sơ đồ logic.

1. **Mô tả chân:**

|  |  |  |
| --- | --- | --- |
| Số chân | Tên chân | Mô tả |
| 1, 2, 3, 4, 5, 6, 7 | Chân output (Q1 đến Q7) | 74HC595 có 8 chân đầu ra, trong đó có 7 chân này. Chúng có thể được kiểm soát nối tiếp. |
| 8 | Ground | Nối đất. |
| 9 | (Q7’) Serial Output | Chân này được sử dụng để kết nối nhiều hơn một 74hc595 dưới dạng xếp tầng. |
| 10 | (MR) Master Reset | Reset tất cả các đầu ra ở mức thấp. Trong quá trình sử dụng nên giữ ở mức cao để hoạt động bình thường. |
|  |  |  |
| 11 | (SH\_CP) Clock | Đây là chân đồng hồ mà tín hiệu đồng hồ phải được cung cấp từ vi điều khiển hoặc vi xử lý |
| 12 | (ST\_CP) Latch | Chân Latch dùng để cập nhật dữ liệu vào các chân đầu ra. Nó kích hoạt mức cao |
| 13 | (OE) Output Enable | Chân OE được sử dụng để tắt đầu ra. Phải giữ ở mức thấp để hoạt động bình thường |
| 14 | (DS) Serial Data | Đây là chân mà dữ liệu được gửi đến, dựa trên đó 8 đầu ra được điều khiển |
| 15 | (Q0) Output | Chân đầu ra đầu tiên |
| 16 | Vcc | Chân này cấp nguồn cho IC, thường sử dụng + 5V |

1. **Tính năng:**

* Nó là một thanh ghi dịch với đầu vào nối tiếp 8 bit và đầu ra nối tiếp 8 bit hoặc song song 3 trạng thái.
* Điện áp hoạt động của IC này là từ 2V đến 6V.
* Điện áp đầu ra bằng điện áp hoạt động của IC này.
* Nó dựa trên công nghệ logic CMOS và do đó tiêu thụ điện năng rất thấp là 80uA.
* Dòng điện đầu ra sorce/sink là 35mA.
* Nó có một đặc điểm là khả năng chống nhiễu cao.
* Nó có thể kết hợp nhiều IC xếp tầng thông qua chân 9 để có nhiều đầu ra hơn.
* Tần số xung nhịp tối đa là 25Mhz ở 4,5V.
* Mạch Schmitt trigger có trên tất cả các đầu vào.



Hình 3 - ảnh thực tế.

1. **Nơi ứng dụng:**

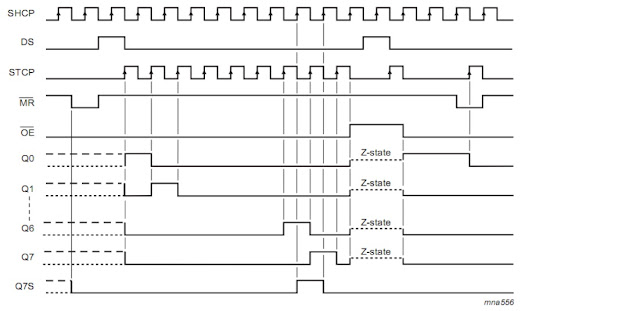
Bạn có bao giờ tự hỏi, làm thế nào một kỹ sư nhúng điều khiển hàng trăm điốt phát sáng nối tiếp hoặc song song với sự trợ giúp của một bộ vi điều khiển có rất ít chân I / O đa năng? Ngoài ra, Bạn muốn điều khiển nhiều hơn 8 động cơ servo và bạn có 2-3 chân GPIO của một bộ vi điều khiển.

Bạn sẽ giải quyết vấn đề này như thế nào? Bạn điều khiển Ma trận LED có các kích thước khác nhau như 8 × 8, 16 × 16, 32 × 32 như thế nào với các chân GPIO tối thiểu của vi điều khiển? Câu trả lời đơn giản là sử dụng một thanh ghi dịch 74HC595.

Trong hầu hết các ứng dụng, bạn cần nhiều đầu ra hơn cho các đèn LED giao tiếp các thiết bị khác như led bảy đoạn, 16 đoạn, đèn flash LED, v.v. IC này rất tiện lợi để sử dụng.

Để tăng các chân đầu ra, bạn có thể giao tiếp IC này với các vi điều khiển khác nhau như Arduino Uno , PIC Microcontroller, Atmel, v.v. Bạn có thể sử dụng IC này trong việc thiết kế các dự án yêu cầu điều khiển nhiều đầu ra.

1. **Giản đồ thời gian (timing diagram) về cách hoạt động của IC.**



Hình 4 – giản đồ thời gian.

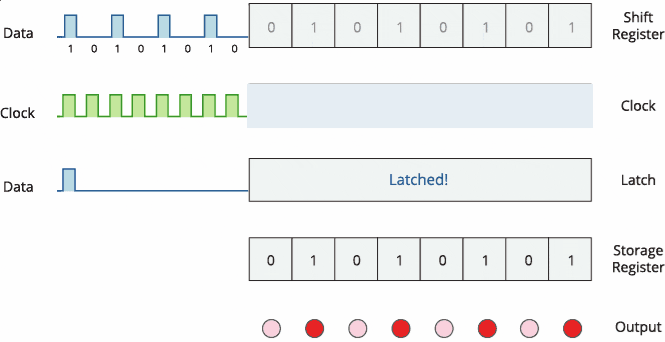
1. **Bảng chân trị:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Đầu vào | | | | | Đầu ra | | Chức năng |
| SH\_CP | ST\_CP |  |  | DS | Q7’ | Qn |
| X | X | L | L | X | L | n.c | MR mức thấp chỉ tác động lên thanh ghi dịch |
| X |  | L | L | X | L | L | Thanh ghi dịch trống được đưa vào thanh ghi lưu trữ |
| X | X | H | L | X | L | Z | Thanh ghi dịch được xóa; đầu ra song song ở trạng thái trở kháng cai |
|  | X | L | H | H | Q6’ | n.c | Logic mức cao được dịch vào thanh ghi dịch giao đoạn 0, dữ liệu thanh ghi của tất cả giai đoạn được dịch, VD: trạng thái thanh ghi dịch giai đoạn thứ 6 được xuất ra song song Q7’ |
| X |  | L | H | X | n.c | Qn’ | Dữ liệu của các giai đoạn thanh ghi dịch (bên trong Qn’) được chuyển vào thanh ghi lưu trữ và trạng thái ra song song |
|  |  | L | H | X | Q6’ | Qn’ | Dữ liệu của thanh ghi dịch được dịch, các trạng thái trước của thanh ghi dịch được truyền vào thanh ghi lưu trữ và các bước ra song song |

\*n.c viết tắt của Not connected.

7. **Cách sử dụng thanh ghi dịch 74HC595:**

Thanh ghi dịch 74HC595 bên trong bao gồm hai thanh ghi như thanh ghi dịch và thanh ghi lưu trữ. Cả hai đều rộng 8-bit. Cái đầu tiên chịu trách nhiệm chấp nhận đầu vào dữ liệu trên mọi cạnh dương của xung clock và nó tiếp tục nhận dữ liệu. Nhưng dữ liệu từ thanh ghi dịch chỉ chuyển sang thanh ghi lưu trữ khi mình áp dụng tín hiệu cao hoạt động để chốt chân đầu vào.



Hình 5 – Nguyên lí.

* Nó có tám đầu ra và 3 chân đầu vào bao gồm chân dữ liệu, chân lưu trữ và chân clock thanh ghi dịch. Kết nối chân 8 với đất và chân 16 với nguồn điện áp + 5V.
* Chân cho phép đầu ra (~ OE) phải được nối đất để kích hoạt các chân đầu ra của thanh ghi dịch. Chân thiết lập lại chính sẽ xóa bộ nhớ của thanh ghi dịch nếu nó được áp dụng với tín hiệu thấp. Đó là lý do tại sao nó nên được giữ ở mức cao.
* Khi quá trình chuyển đổi cạnh dương xảy ra trên chân 11, thanh ghi dịch sẽ chấp nhận các đầu vào được áp dụng trên đường dữ liệu.
* Các đầu ra của thanh ghi lưu trữ được kết nối với các chân đầu vào của chốt D / thanh ghi lưu trữ.
* Các đầu vào này được cập nhật trên đầu ra chốt khi xảy ra quá trình chuyển đổi cạnh dương tại chân 12.

Quan trọng nhất, Nếu bạn cần ghép nhiều IC với nhau thì chân 9 được kết nối với chân dữ liệu của một IC thanh ghi dịch khác.

* 1. **LED matrix 8x8:**

**8.1 CODE**

#include<reg51.h>

#include<intrins.h>

sbit SRCLK=P3^6;

sbit RCLK=P3^5;

sbit SER=P3^4;

//--����Ҫʹ�õ�IO��--//

#define COMMONPORTS     P0

//--��ѡͨ����--//

unsigned char code TAB[8]  = {0x7f,0xbf,0xdf,0xef,0xf7,0xfb,0xfd,0xfe};

//--��������--//

unsigned char code CHARCODE[18][8]=

{

{0x00,0x00,0x3e,0x41,0x41,0x41,0x3e,0x00}, //0

{0x00,0x00,0x00,0x00,0x21,0x7f,0x01,0x00}, //1

{0x00,0x00,0x27,0x45,0x45,0x45,0x39,0x00}, //2

{0x00,0x00,0x22,0x49,0x49,0x49,0x36,0x00}, //3

{0x00,0x00,0x0c,0x14,0x24,0x7f,0x04,0x00}, //4

{0x00,0x00,0x72,0x51,0x51,0x51,0x4e,0x00}, //5

{0x00,0x00,0x3e,0x49,0x49,0x49,0x26,0x00}, //6

{0x00,0x00,0x40,0x40,0x40,0x4f,0x70,0x00}, //7

{0x00,0x00,0x36,0x49,0x49,0x49,0x36,0x00}, //8

{0x00,0x00,0x32,0x49,0x49,0x49,0x3e,0x00}, //9

{0x00,0x00,0x7F,0x48,0x48,0x30,0x00,0x00}, //P

{0x00,0x00,0x7F,0x48,0x4C,0x73,0x00,0x00}, //R

{0x00,0x00,0x7F,0x49,0x49,0x49,0x00,0x00}, //E

{0x00,0x00,0x3E,0x41,0x41,0x62,0x00,0x00}, //C

{0x00,0x00,0x7F,0x08,0x08,0x7F,0x00,0x00}, //H

{0x00,0x00,0x00,0xFF,0xFF,0x00,0x00,0x00}, //I

{0x00,0x7F,0x10,0x08,0x04,0x7F,0x00,0x00}, //N

{0x7C,0x48,0x48,0xFF,0x48,0x48,0x7C,0x00}  //��

};

void delay(unsigned int time)

{

  unsigned int i,j;

  for(i=0;i<time;i++)

    for(j=0;j<121;j++);

}

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

Tên hàm : Hc595SendByte(unsigned char dat)

Mô tả : Gửi một byte dữ liệu tới thanh ghi dịch 74HC595.

Đầu vào : dat - Byte dữ liệu cần gửi.

Đầu ra : Không có

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

void Hc595SendByte(unsigned char dat)

{

    unsigned char a;

    SRCLK=0;

    RCLK=0;

    for(a=0;a<8;a++)

    {

        SER=dat>>7;

        dat<<=1;

        SRCLK=1;

        \_nop\_();

        \_nop\_();

        SRCLK=0;

    }

    RCLK=1;

    \_nop\_();

    \_nop\_();

    RCLK=0;

}

void main()

{

    unsigned char tab, j;

    unsigned int  i;

    while(1)

    {

        for(i= 0; i<50; i++

        {

            for(tab=0;tab<8;tab++)

            {

                Hc595SendByte(0x00);

                COMMONPORTS = TAB[tab];

                Hc595SendByte(CHARCODE[j][tab]);

                delay(2);

            }

        }

        j++;

        if(j == 18)

        {

            j = 0;

        }

    }

}

**Giải thích code:**

void Hc595SendByte(unsigned char dat)

{

    unsigned char a;

    SRCLK=0;

    RCLK=0;

    for(a=0;a<8;a++)

    {

        SER=dat>>7;

        dat<<=1;

        SRCLK=1; //

        SRCLK=0;

    }

    RCLK=1;

    RCLK=0;

}

Trong vòng lặp for , biến dat sẽ được dịch sang trái sau mỗi vòng lặp. Điều này có nghĩa là mỗi bit của biến dat sẽ được chuyển sang vị trí kế tiếp .

Sau đó, bit có trọng số cao nhất của biến dat (bit 7) sẽ được chuyển vào đầu vào SER (Serial Data Input) của thanh ghi.

Tiếp theo, tín hiệu SRCLK (Shift Register Clock) sẽ được gán giá trị 1 để thực hiện chuyển bit hiện tại trong SER vào bộ nhớ 8 bit của thanh ghi. Khi SRCLK là 1, dữ liệu sẽ được chuyển từ SER vào thanh ghi. Sau đó, SRCLK sẽ được gán giá trị 0 để chuẩn bị cho lần chuyển bit tiếp theo.

Sau khi đã chuyển 8 bit từ SER vào thanh ghi, tín hiệu RCLK (Register Clock) sẽ được gán giá trị 1 để thực hiện chốt dữ liệu trong thanh ghi. Khi RCLK là 1, dữ liệu trong thanh ghi sẽ được chốt, và các giá trị từ Q0 đến Q7 sẽ được đưa ra qua các ngõ ra tương ứng. Sau đó, RCLK sẽ được gán giá trị 0 để chuẩn bị cho lần chốt dữ liệu tiếp theo.

Quá trình này sẽ lặp lại trong vòng lặp for, cho phép dịch và chuyển dữ liệu từ biến dat vào thanh ghi 8 bit sử dụng các tín hiệu SRCLK và RCLK (cột bằng 0 và hàng bằng 1 led sáng).

unsigned char code TAB[8]  = {0x7f,0xbf,0xdf,0xef,0xf7,0xfb,0xfd,0xfe}; //cot

Mảng TAB chứa các cột trong led ma trận 8x8, chứa các bit 0.

//hang (quet led, khi cot bang 0 va hang bang 1 se thi led se sang)

unsigned char code CHARCODE[18][8]=

{

{0x00,0x00,0x3e,0x41,0x41,0x41,0x3e,0x00}, //0

{0x00,0x00,0x00,0x00,0x21,0x7f,0x01,0x00}, //1

Mảng CHARCODE chứa các mã khi kết hợp với mảng TAB sẽ ra kết quả là các kí tự từ 0-9, A-N.

void main()

{

    unsigned char tab, j;

    unsigned int  i;

    while(1)

    {

        for(i= 0; i<50; i++ )

        {

            for(tab=0;tab<8;tab++)

            {

                Hc595SendByte(0x00);

                COMMONPORTS = TAB[tab];

                Hc595SendByte(CHARCODE[j][tab]);

                delay(2);

            }

        }

        j++;

        if(j == 18)

        {

            j = 0;

        }

    }

}

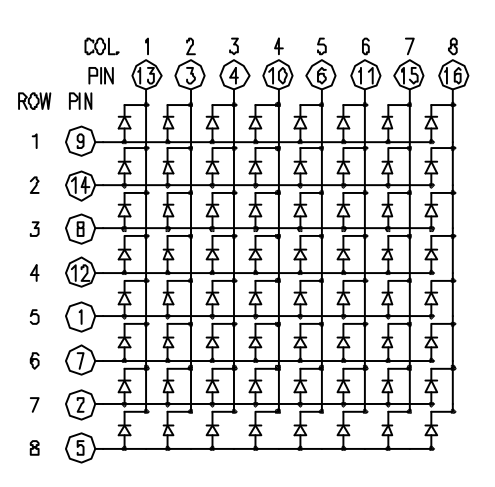
Trong hàm main():

Trong vòng lặp for, giá trị của mảng COMMONPORTS sẽ được gửi đến port 0. Để thực hiện việc gửi này, chúng ta sử dụng hàm Hc595SendByte() để gửi một chuỗi giá trị của một ký tự.

Khi kết hợp COMMONPORTS với các giá trị gửi bằng hàm Hc595SendByte(), nếu bit tương ứng trong mảng COMMONPORTS là 0 và bit CHARCODE[j][tab] là 1, thì đèn LED sẽ được bật.

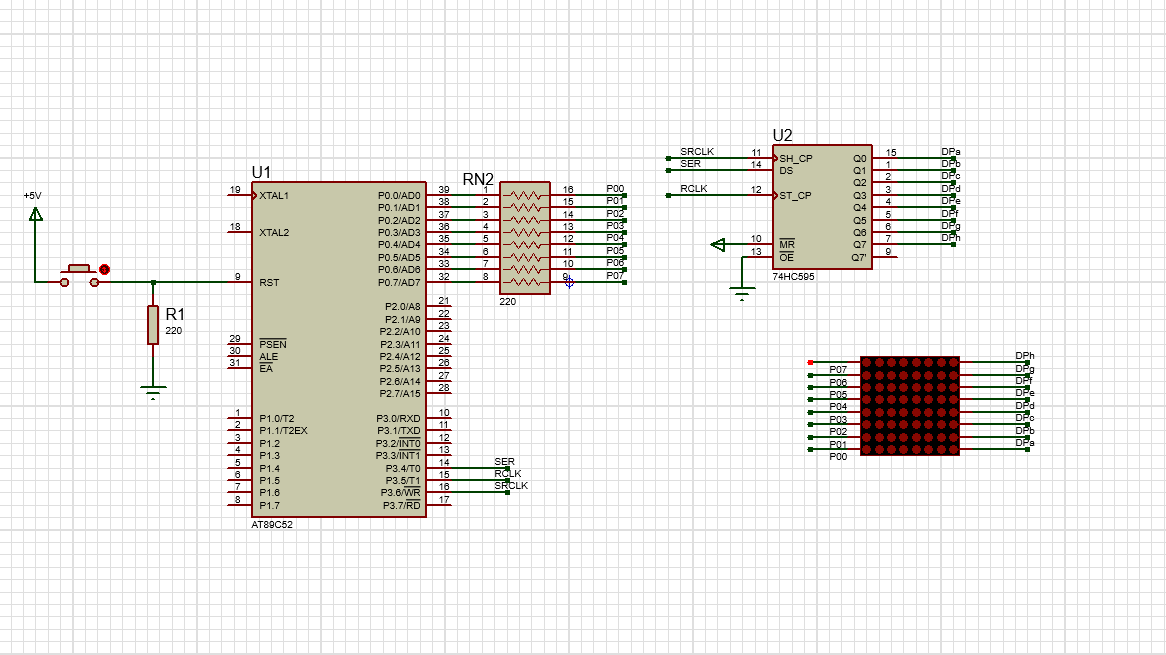
Điều này có nghĩa là khi bit COMMONPORTS tại vị trí tương ứng là 0 và bit CHARCODE[j][tab] tại vị trí tương ứng là 1, thì ta có điều kiện để bật đèn LED.

Quá trình này sẽ được lặp lại trong vòng lặp for, để kiểm tra và điều khiển LED dựa trên giá trị của mảng COMMONPORTS và giá trị gửi bằng hàm Hc595SendByte().



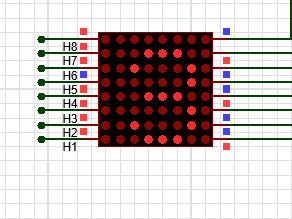
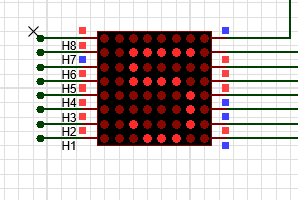
Hình 6 – LED matrix 8x8.

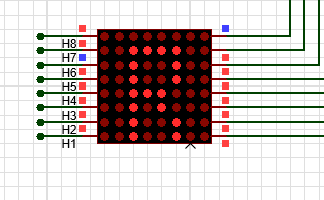
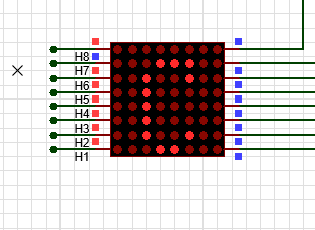
**8.2 Mô phỏng Proteus:**



Hình 7 – Mạch mô phỏng.

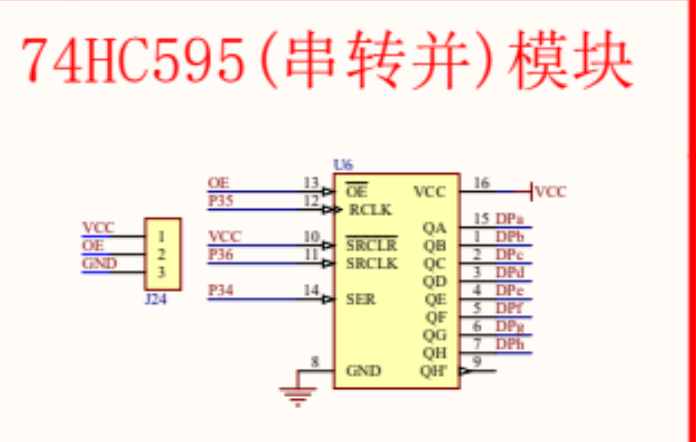
**8.3 OUTPUT trên LED matrix 8x8:**

****

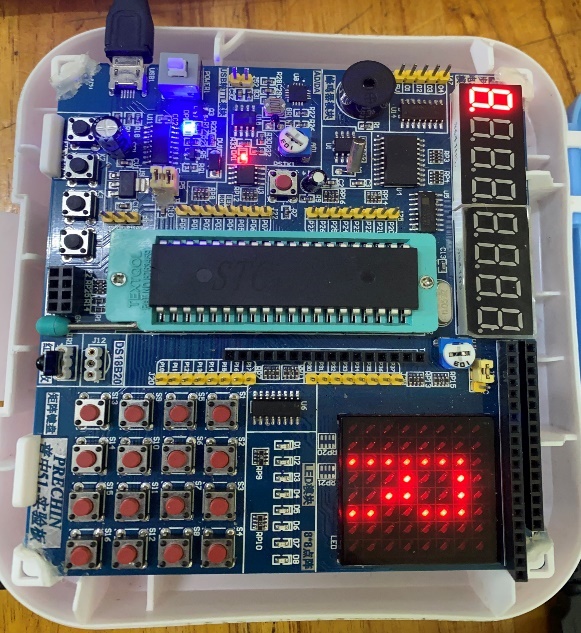
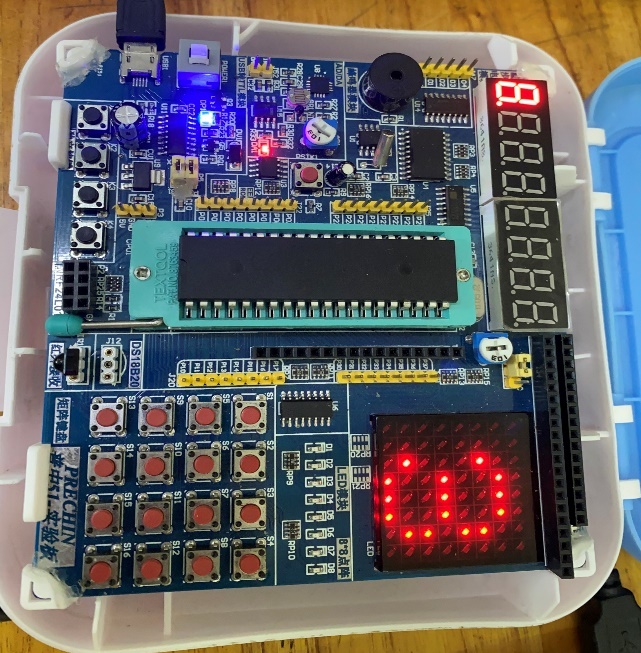
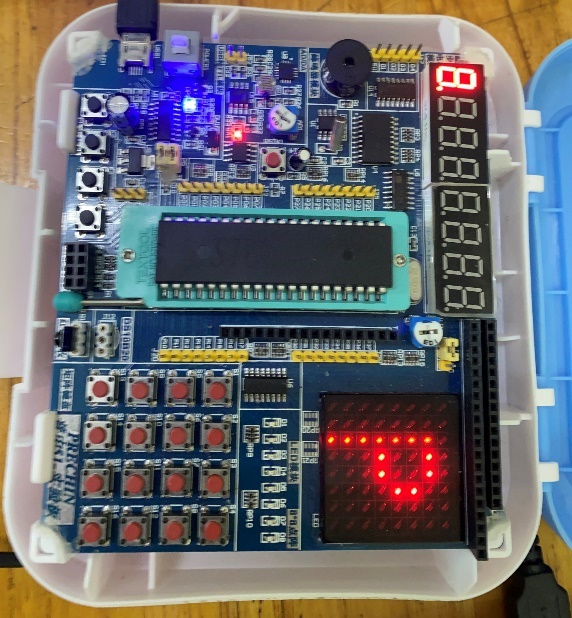


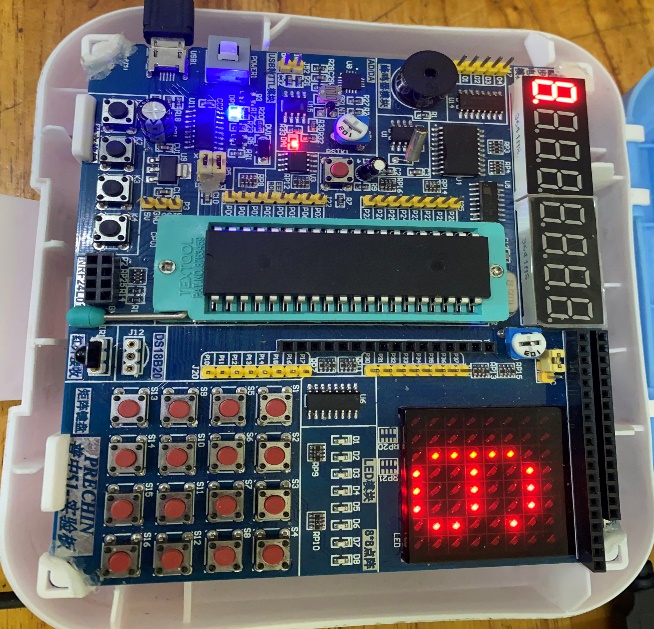
Hình 8 – Kết quả mô phỏng

**8.4 Kit thực tế:**

****

Hình 9 - Schematic

****

****

Hình 10 – Kết quả trên kit