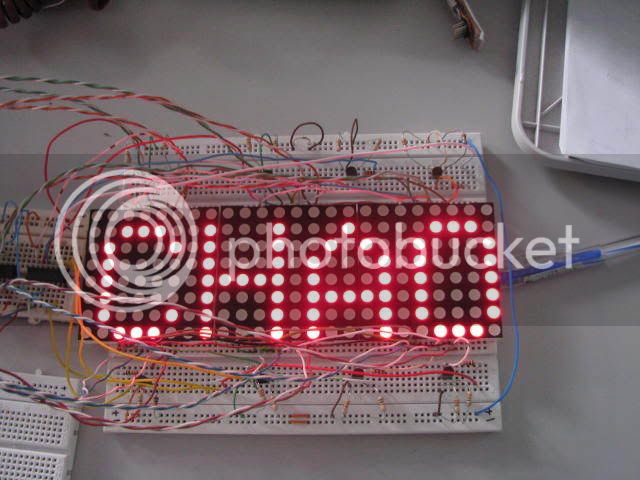
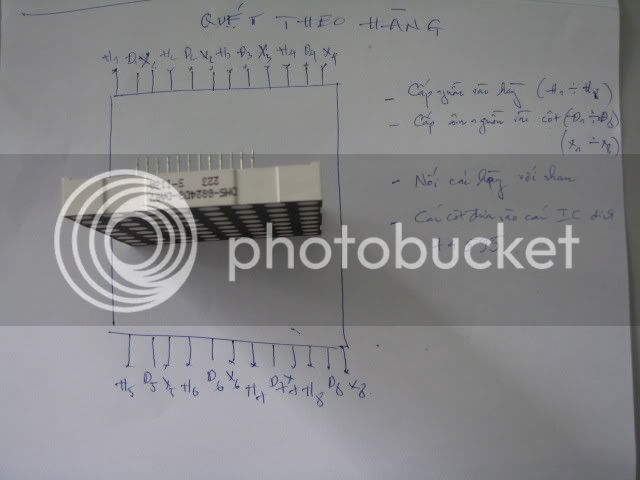
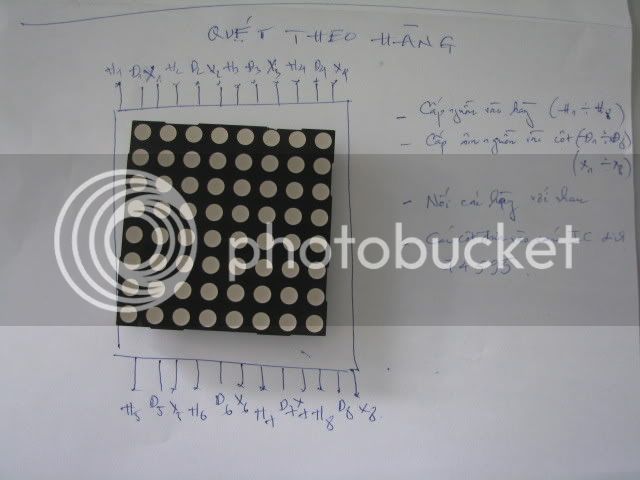
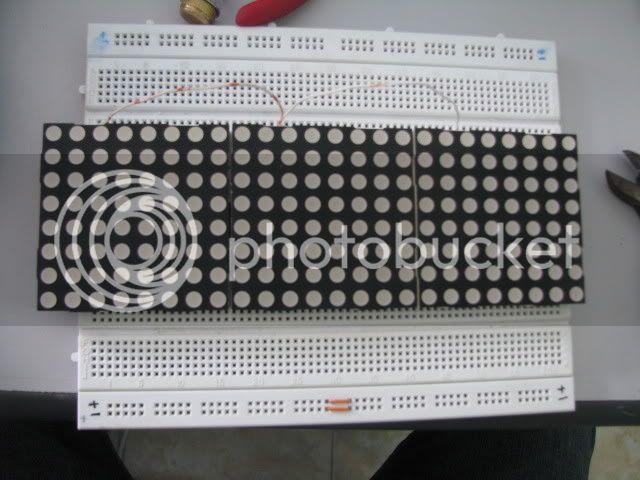
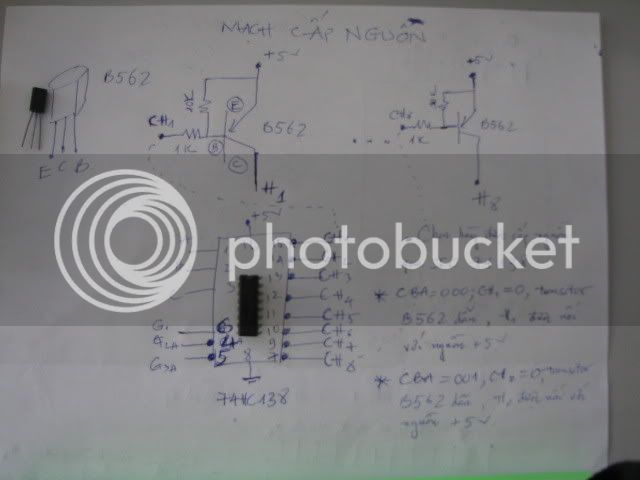
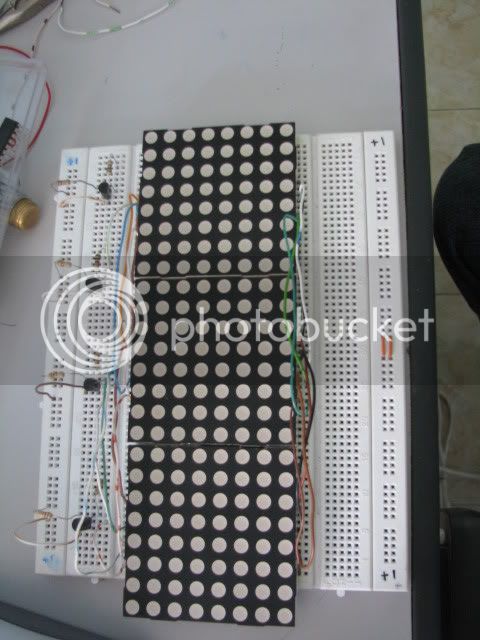
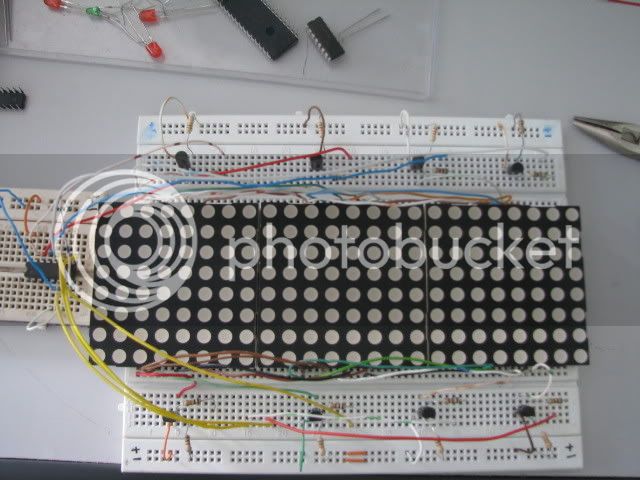
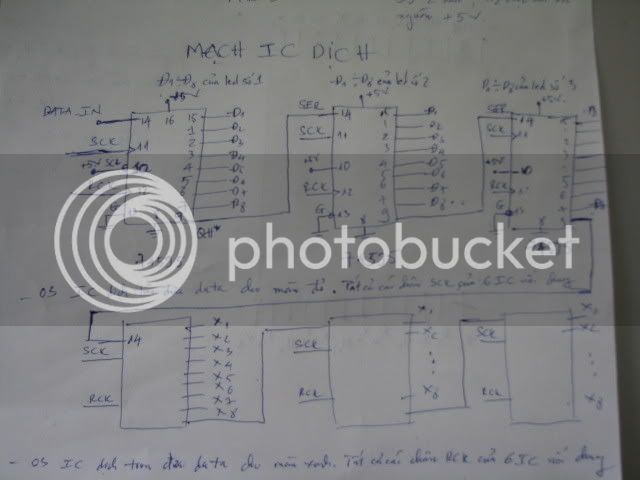
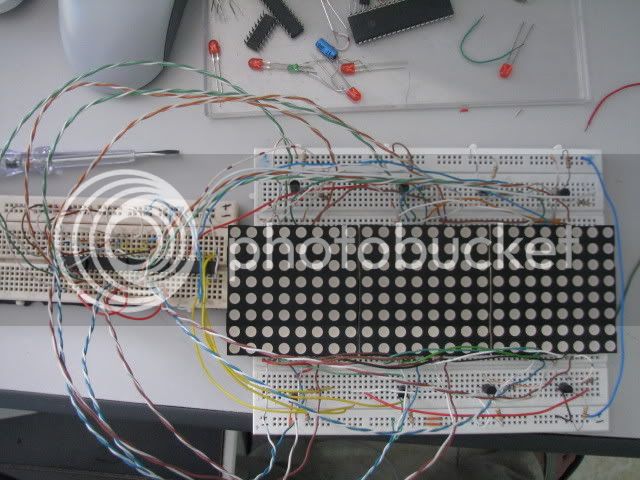
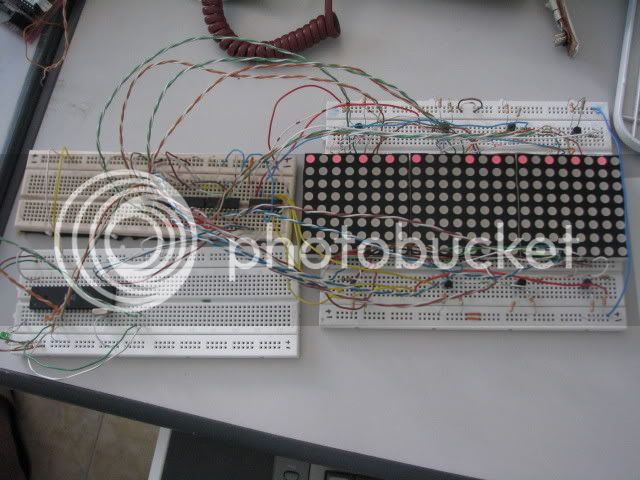
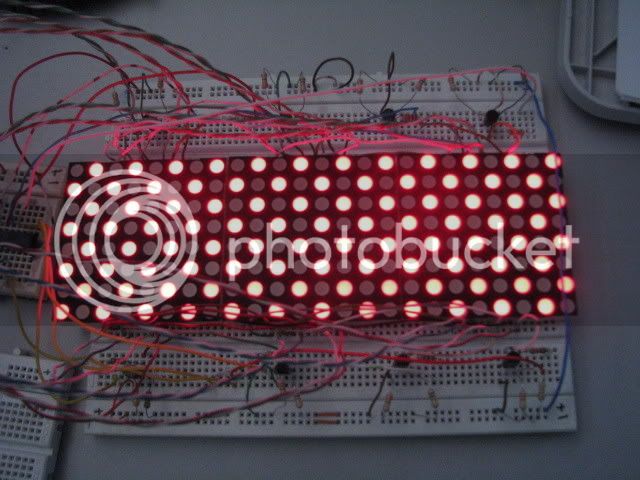
Có nhiều bác gọi điện thoại hỏi em về việc thiết kế led ma trận, sơ đồ mạch thiết kế như thế nào, dùng những IC nào, quét theo hàng hay theo cột, làm sao hiển thị được chữ A, bộ font chữ tạo ra như thế nào…  
Để bớt phải đóng góp tiền cho VNPT trong thời buổi lạm phát này, em viết một tutorial tạm gọi là chi tiết. Hi vọng tutorial này sẽ giúp đỡ các bác đang nghiên cứu về led ma trận tiết kiệm được thời gian và công sức hơn.  
  
Chúng ta bắt đầu làm việc ngay bây giờ. Trước tiên hướng tới thành quả cuối cùng để lấy hứng thú làm việc.  
  
  
Em cũng đưa vào một hiệu ứng đơn giản là chữ “CHAO” sẽ cuộn tròn. Các bác có thể xem hiệu ứng này ở file avi download ở link sau:  
<http://rapidshare.com/files/133093020/Picture_028.rar>  
  
Chúng ta sẽ cần 02 miếng breadboard màu trắng để cắm 3 led ma trận 2 màu 8x8 vào. Các bác cắm sát vào phía bên trái của breadboard nhé. Khi cắm vào sẽ trông như sau:  
  


Phù vậy là xong được 30% công việc rồi. 30% nữa là cắm dây để kết nối mạch và 30% là để viết chương trình. 10% để gọi ai đó đến xem thành quả của chúng ta.  
Ta sẽ đánh dấu các chân của led ma trận nhỉ? Các bác quay cái phía có chữ của led ma trận lên trên nhé. Lưu ý led ma trận của chúng ta là 2 màu có anot chung, một đầu cathot là của led màu xanh và đầu kia màu đỏ. 24 chân của led ma trận 2 màu 8x8 này em đánh dấu như sau:  
  
  
Từ trái sang phải, trên xuống dưới thứ tự chân các led ma trận là:  
- H1 Đ1 X1 H2 Đ2 X2 H3 Đ3 X3 H4 Đ4 X4  
- H5 Đ5 X5 H6 Đ6 X6 H7 Đ7 X7 H8 Đ8 X8  
Để các led đơn trên led ma trận sáng lên ta sẽ cấp dương nguồn vào các chân H1 – H8 và âm nguồn vào các chân Đ1 - Đ8, X1 - X8 tương ứng. Ví dụ ta muốn sáng led đỏ ở hàng thứ 5 và cột thứ 3 thì ta làm sao đó để cấp dương nguồn vào chân H5 và âm nguồn vào chân Đ3.  
Chúng ta sẽ dùng nguyên lý quét theo hàng, tức là đầu tiên sẽ cấp nguồn vào chân H1, cùng lúc đó dữ liệu ta sẽ đưa ra các chân Đ1 – Đ8 và X1 – X8. Sau đó không cấp nguồn cho chân H1 nữa mà cấp nguồn cho chân H2 và dữ liệu tương ứng tiếp theo sẽ đưa ra các chân Đ1 – Đ8, X1 – X8. Cứ tiếp tục như thế cho đến chân H8 rồi quay trở lại chân H1.  
Ta sẽ nối các chân H1 của tất cả các led ma trận lại với nhau nhé. Nối như thế này, rất đơn giản:  
  
Sau đó các bác kiên nhẫn nối các chân H2, H3…H8 lại với nhau. Bây giờ ta nói đến chuyện làm cách nào để cấp nguồn tuần tự từng hàng nhé.

Ta sẽ sử dụng 8 transitor B564, nối các chân C của 8 transitor này với các chân H1-H8, chân E nối lên dương nguồn 5V, chân B là chân điều khiển. Khi chân B ở mức 1, transitor B562 không dẫn, H1 không có dương nguồn. Khi B được kéo xuống mức 0, transitor B562 sẽ dẫn, lúc đó H1 sẽ được nối với nguồn 5V.  
  
Ta có 8 chân cần cấp nguồn lần lượt H1 – H8 tức là có 8 chân nối với các chân B (qua điện trở 1k) của transitor B562 CH1-CH8. Ta có thể nối thẳng các chân điều khiển này vào vi xử lý (ở đây em sẽ dùng con có sẵn là ATMEGA8518). Nhưng như vậy thì phí chân MCU quá, để chân MCU còn làm việc khác khi mở rộng mạch sau này. Do đó em em sẽ nối các chân CH1 - CH8 với con 74HC138. Qua con IC giải mã 3 ra 8 này em chỉ tốn 4 chân, 3 chân điều khiển tín hiệu A, B, C và một chân điều khiển tín hiệu cho phép cho nó. Nguyên lý làm việc của con này bác nào chưa hiểu thì xem thêm datasheet nhé.  
Rồi cắm 8 chú B562 này vào breadboard, mỗi chú có thêm 2 em điện trở đi kèm. Em bám vào chân B 1k, còn em lên dương nguồn thì 10k. Cắm cũng chẳng lâu đâu, xíu xiu thôi mà:  
  
Cả chú 74HC138 luôn, nhưng hết chỗ cho chú rồi, cho chú sang board khác vậy  
  
Chú em phụ cắm led cho em vừa cắm vừa tủm tỉm cười nè  
  
Và dưới đây là schematic của tutorial này nhé:  
(em đang không vào được mạng qua đường chính thống, phải đi qua proxy nên không xài chức năng upload file của diễn đàn được, các bác chịu khó down rapidshare)  
<http://rapidshare.com/files/133103844/Orcad.rar>

Vậy ta đã giải quyết xong vấn đề cấp nguồn, giờ còn việc đưa data ra các chân Đ1 - Đ8, X1 – X8 của 3 con led ma trận. Ta sẽ cần 6 con IC dịch 74HC595. Nguyên lý làm việc của con này mong các bác tham khảo ở datasheet. Em chỉ tóm tắt thế này. Chân DATA\_IN là đường dữ liệu vào (bit dữ liệu nối tiếp). Một xung SCK sẽ làm bit này đi vào thanh ghi dịch trong bụng con 74HC595. Sau 8 bit liên tục thì thanh ghi dịch này sẽ đầy. Tiếp tục đưa bit thứ 9 vào thì bit đầu tiên sẽ bị lòi ra chân số 9 và bit này lại là bit đầu tiên của IC 74HC595 tiếp theo (chân 9 của IC trước nối chân 14 của IC sau). Ta sẽ phải đưa liên tục 8 x 6 = 48 bit để xuất đủ dữ liệu cho 2 màu xanh và đỏ.  
  
Tuy nhiên cần lưu ý là thanh ghi dịch mới chỉ thay đổi trong bụng con 74HC595 thôi chứ chưa đưa ra các chân. Muốn đưa dữ liệu từ các thanh ghi dịch này ra các chân 74HC595 thì cần đưa một xung vào chân RCK (chân số 12).  
  
  
  
Các bác để ý là 6 chân SCK của 6 IC 74HC595 nối nhau và đưa về MCU, 6 chân RCK cũng tương tự. Chân G (enable) được nối mass và chân MR (chân reset) nối lên nguồn.  
  
Trong tutorial này, để tiết kiệm thời gian cho em và cho các bác, em chỉ cắm 3 con 74HC595 phục vụ cho việc sáng các led đỏ. Các bác có thể làm với cả 6 IC dịch để sáng được cả các led xanh nếu muốn.  
Các bác xem hình ảnh khi cắm xong 3 con 74HC595 này, hơi rối một chút nhưng bác sỹ bảo không sao:  
  
  
  
Rồi, đưa cả con MCU vào luôn nhé, ở đây em dùng ATMEGA8515. Schematic các bác xem ở bài trước. Bác nào xài PIC thông cảm nhé, bữa nào em rảnh em làm lại cái này với PIC vậy  
  
  
  
Oái, sao tự nhiên nó sáng mấy cái led vậy, chắc con ATMEGA8515 của em đang chạy chương trình cũ của nó. Các bác đừng quan tâm nhé.

Vậy là phần cứng đã xong, em để 1,2 ngày để các bác viết thử phần mềm. Các bác cần lưu ý là em không sử dụng điện trở hạn dòng nhé. Thông thường người ta sẽ hạn dòng bằng điện trở 220-330 ohm ở các chân Đ1 - Đ8 trước khi nối vào IC 74HC595. Vì em sử dụng phương pháp quét hàng, mỗi hàng chỉ sáng trong thời gian rất ngắn nên sẽ không ảnh hưởng gì đến tuổi thọ của led. Tuy nhiên khi lần đầu tiên làm với led ma trận các bác sẽ không tránh khỏi việc ngâm điện 5V giữa 2 đầu của led. Em mách các bác một chiêu nho nhỏ, thay vì nối thẳng chân E của các transistor B562 lên nguồn, các bác hãy nối qua một điện trở khoảng 330 ohm. Sau khi chương trình của ta tốt rồi, ta sẽ bỏ các điện trở này đi.  
  
Các bài viết tiếp theo của em sẽ viết về cách hiển thị dữ liệu lên led ma trận, cách tạo phông chữ bằng phần mềm Microsoft Excel, cách tạo các hiệu ứng dịch chữ, tăng giảm độ sáng, cách làm chữ hiện lên, biến mất... Các bác từ từ đón xem.

Hôm nay ta bắt đầu viết những chương trình đầu tiên, từ cái dễ đến cái khó. Không thể nào có thể viết một phát mà có ngay một dòng chữ chuyển động ngay được. Giống như ta tập đi, từng bước một.  
Ban đầu ta sẽ làm sao để bảng led ma trận của ta hiển thị được cứ một điểm sáng lại có một điểm tắt như sau:  
  
  
Chương trình em sẽ viết rất đơn giản như sau:

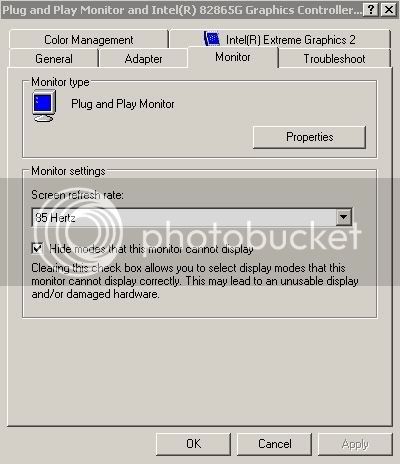
*#include <mega8515.h>  
#include <delay.h>  
#define SCK PORTB.7 //Chân cấp xung để đưa data vào trong thanh ghi dịch  
#define DATA PORTB.5 //Chân đưa data vào thanh ghi dịch  
#define SCL PORTB.3 //Khi data đã vào trong thanh ghi dịch, đưa một xung ra chân này để đưa data từ thanh ghi dịch ra các chân của IC74HC595  
  
//Bốn chân sau đây điều khiển việc cấp nguồn ra các hàng của led ma trận  
#define A PORTC.0;  
#define B PORTC.1;  
#define C PORTC.2;  
#define OE1 PORTC.3;  
  
unsigned char i,j;  
void Data\_in(unsigned char k){ // k là bit dữ liệu vào có giá trị là 0 hoặc 1;  
DATA=k; // DATA đưa dữ liệu vào thanh ghi dịch  
SCK=0; // Tạo ra một xung ở chân SCK  
SCK=1;  
}  
void LatchData(){  
SCL=0; // Sau khi đưa đủ dữ liệu vào các thanh ghi dịch  
SCL=1; // ta tạo một xung để đưa dữ liệu từ thanh ghi dịch ra các chân của 74HC595  
}  
void main(void){  
DDRB=0xFF; // Cho PORTB, PORTC là output hết  
DDRC=0xFF;  
SCK=1; // Ban đầu đưa các chân này lên 1  
SCL=1; // Để tạo xung ta sẽ cho nó băng 0 rồi lại bằng 1  
while (1){  
for (i=0;i<=7;i++){ // 8 hàng, từ hàng 0 đến hàng 7 (H1 - H8)  
// Ta có tất cả 8 hàng và 24 cột. Do đó phải đưa 24 bit dữ liệu ra các thanh ghi dịch (vòng lặp for gọi 12 lần, mỗi lần đưa ra 1 bit 1, 1 bit 0)  
for (j=1;j<=12;j++){  
Data\_in(0);  
Data\_in(1);  
}  
// Đưa ra thanh ghi dịch xong rồi, đưa data ra các chân của 74HC595  
LatchData();  
// PortC sẽ đưa ra các số từ 0 đến 7 ở các chân PORTC.0, PORTC.1,.PORTC.2,... Ta cộng thêm 8 để muốn chân PORTC.3 sẽ luôn bằng 1. Con 74HC138 sẽ ở trạng thái tích cực để giải mã từ 3 ra 8.  
PORTC=i+8;  
// Delay một chút để tạo hiện tượng lưu ảnh của mắt. Các bác hãy thay đổi giá trị này để thấy rõ hiệu ứng quét hàng. Thay số 3 bằng 50 chẳng hạn.  
delay\_ms(3);  
}  
};  
}*

Vậy là xong bài lập trình đầu tiên. Dễ như ăn cháo các bác nhỉ?  
  
Đây là bài đầu tiên em chú thích kỹ, từ bài tiếp theo em sẽ chỉ đưa ra các phần chính. Còn chương trình cụ thể các bác hãy download ở link bên dưới.

Vậy là qua bài đầu, các bác đã nắm được một phần nguyên lý quét led rồi nhỉ? Rất đơn giản ta đưa data vào một hàng qua cơ chế thanh ghi dịch và ngay sau khi đưa data vào ta cấp nguồn vào cho hàng đó. Cứ tuần tự như thế, với một tốc độ quét hợp lý hình ảnh sẽ được hiện ra.  
  
Bài vừa rồi ta đã dùng cơ chế truyền dữ liệu SPI từ ATMEGA8515 ra các thanh ghi dịch bằng phần mềm. Tức là ta phải làm tuần tự 2 việc: 1. đưa data vào ngõ vào của thanh ghi dịch, 2. Cấp cho nó một xung clock.  
  
MCU ATMEGA8515 có sẵn phần cứng SPI, mỗi lần đưa ra ngoài được 8 bit. Có 4 chân được dành cho giao tiếp SPI là PORTB.7 (chân cấp xung clock). PORTB.6 (nhận data về). PORTB.5 (xuất data ra). PORTB.4 (tích cực mức thấp để chọn chip). Ở đây vì chỉ xuất data ra và đám IC dịch luôn sẵn sàng để nhận dữ liệu nên ta chỉ cần 2 chân. PORTB.7 (chân clock) và PORTB.5 (chân data out).  
  
Chương trình đầu ta sẽ chỉnh sửa/bổ sung một chút như sau:

*// SPI functions, khai báo thêm hàm spi.h  
#include <spi.h>  
//trong hàm main, khai báo thêm phần SPI  
// SPI initialization  
// SPI Type: Master  
// SPI Clock Rate: 4000,000 kHz  
// SPI Clock Phase: Cycle Half  
// SPI Clock Polarity: Low  
// SPI Data Order: MSB First  
SPCR=0x50;  
SPSR=0x00;  
while (1){  
for (i=0;i<=7;i++){ //có 8 hàng  
if (i&1){ //nếu là hàng lẻ 1,3,5,7 thì sáng trước, tắt sau  
spi(0b10101010); //mỗi hàm spi đưa ra được 8 bit  
spi(0b10101010); //do đó sẽ gọi 3 lần  
spi(0b10101010);  
} else {//nếu là hàng chẵn 0,2,4,6 thì sáng sau, tắt trước  
spi(0b01010101);  
spi(0b01010101);  
spi(0b01010101);  
}  
LatchData();  
PORTC=i+8;  
delay\_ms(2);  
}*

Vậy là xong bài 2, sử dụng spi bằng phần cứng của MCU.  
  
Các bác có để ý em sửa lại delay là 2ms không? Các bác sẽ thấy hiện tượng hơi chớp ở bài 1 không còn nữa.  
Bác nào tính cho em tần số quét ở bài 1 và bài 2 nào. Đây là công việc quan trọng trước khi ta vào bài 3. Thay vì dùng hàm delay, ta sẽ dùng timer để quét led.

Chưa có ai tính giúp em tần số quét nhỉ? em tính thử xem nhé. Nếu em delay 3ms, tính thêm các câu lệnh rườm rà nữa là khoảng 3.5 ms. Như vậy để quét hết 8 hàng ta sẽ cần 3.5 x 8 = 28ms. Như vậy tần số quét là 1s = 1000 ms / 28 = 35.7. Tức là 1 giây màn hình led của ta sẽ được quét khoảng 36 lần. Ta sẽ cảm thấy hiện tượng chớp chớp trên màn hình led.  
  
Nếu delay là 2 ms, các câu lệnh nữa là 2.5. Quét hết 8 hàng cần 2.5 x 8 =20 ms. Tần số quét sẽ là 1000 / 20 = 50 Hz. Hiện tượng chớp không còn nữa.  
  
Điều em muốn nói ở đây là các bác phải phân biệt tần số quét và khái niệm số hình trên giây ví dụ 24 hình /s. Tần số quét của ta phải từ 50 Hz trở lên thì ta mới không thấy hiện tượng chớp tắt. Các bác có thể kiểm tra điều này bằng cách chuột phải lên destop (màn hình nền windows) ->properties -> settings -> advanced -> Monitor -> Screen refresh rate bác sẽ thấy windows cho phép ta chọn tốc độ quét của màn hình trong khoảng 60 đến 120 Hz. Các bác hãy để thử tốc độ quét thấp nhất 60Hz (hiện cao hơn tần số 50 Hz của ta) thì sẽ thấy màn hình máy tính rung rung...  
  
  
  
Vậy ta chuẩn bị sang bài 3 được rồi nhỉ? Sử dụng timer để quét led thay cho hàm delay.

Hôm nay ta bắt đầu bài 3: Thay thế hàm delay bằng ngắt Timer1. Sở dĩ ta phải dùng ngắt Timer thay cho hàm delay bởi trong khi delay thì ta sẽ không làm được gì cả. Còn sử dụng Timer thì trong thời gian chờ đợi ta có thể chuẩn bị dữ liệu, tính toán hiệu ứng cho hàng tiếp theo.  
Em sẽ khai báo ngắt Timer1 như sau:

*// Timer/Counter 1 initialization  
// Clock source: System Clock  
// Clock value: 15.625 kHz  
// Mode: CTC top=OCR1A  
// OC1A output: Discon.  
// OC1B output: Discon.  
// Noise Canceler: Off  
// Input Capture on Falling Edge  
// Timer 1 Overflow Interrupt: Off  
// Input Capture Interrupt: Off  
// Compare A Match Interrupt: On  
// Compare B Match Interrupt: Off  
TCCR1A=0x00;  
TCCR1B=0x0D;  
TCNT1H=0x00;  
TCNT1L=0x00;  
ICR1H=0x00;  
ICR1L=0x00;  
OCR1AH=0x00;  
OCR1AL=0x25; // Quyet dinh toc do quet  
OCR1BH=0x00;  
OCR1BL=0x00;*

Timer1 có tần số hoạt động là 15625Hz tức là nó sẽ đếm 15625 lần trong vòng 1s. Ở đây, nó cứ đếm từ 0 đến 0x25 = 37 thì ngắt Timer1 sẽ được gọi. Như vậy ngắt Timer1 sẽ được gọi với tần số là 15625/37=422 lần/s.  
Chúng ta phải nhớ kích hoạt ngắt nhé:

*// Timer(s)/Counter(s) Interrupt(s) initialization  
TIMSK=0x40;  
// Global enable interrupts  
#asm("sei")*

Chương trình ngắt Timer1 được viết như sau:

*// Timer 1 output compare A interrupt service routine  
interrupt [TIM1\_COMPA] void timer1\_compa\_isr(void)  
{ if (r & 1){  
spi(0b10101010);  
spi(0b10101010);  
spi(0b10101010);}  
else {  
spi(0b01010101);  
spi(0b01010101);  
spi(0b01010101);  
}  
LatchData();  
PORTC=r+8;  
r++;  
if (r==8) r=0;  
}*

Như vậy mỗi khi ngắt Timer1 được gọi thì một hàng sẽ được quét. Ta có tất cả 8 hàng. Do đó tần số quét sẽ là: 422/8=53Hz. Với tần số quét này màn hình led sẽ không bị chớp.  
  
Các bác có thể thấy trong vòng lặp while (1) không còn câu lệnh nào cả. Việc quét led hoàn toàn thực hiện tự động nhờ Timer1.

*while (1){  
}*

Các bác nghỉ lấy sức, qua bài 4 chúng ta sẽ học cách tạo bộ font chữ bằng chương trình Microsoft Excel như thế nào.  
  
Source code của bài 3 đây:  
  
<http://rapidshare.com/files/134768565/bai3.doc>

Code:

3. Thay delay bang ngat Timer 1

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

This program was produced by the

CodeWizardAVR V1.25.8 Standard

Automatic Program Generator

© Copyright 1998-2007 Pavel Haiduc, HP InfoTech s.r.l.

http://www.hpinfotech.com

Project :

Version :

Date : 7/27/2008

Author : Nguyen Hai Ha

Company : www.softviet.net

Comments:

Chip type : ATmega8515L

Program type : Application

Clock frequency : 16,000000 MHz

Memory model : Small

External SRAM size : 0

Data Stack size : 128

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

#include <mega8515.h>

#include <delay.h>

// SPI functions

#include <spi.h>

#define SCK PORTB.7

#define DATA PORTB.5

#define SCL PORTB.3

#define B PORTC.1;

#define C PORTC.2;

#define OE PORTC.3;

unsigned char r=0;

void LatchData(){

SCL=0;

SCL=1;

}

// Timer 1 output compare A interrupt service routine

interrupt [TIM1\_COMPA] void timer1\_compa\_isr(void)

{ if (r & 1){

spi(0b10101010);

spi(0b10101010);

spi(0b10101010);}

else {

spi(0b01010101);

spi(0b01010101);

spi(0b01010101);

}

LatchData();

PORTC=r+8;

r++;

if (r==8) r=0;

}

void main(void){

// SPI initialization

// SPI Type: Master

// SPI Clock Rate: 4000,000 kHz

// SPI Clock Phase: Cycle Half

// SPI Clock Polarity: Low

// SPI Data Order: MSB First

SPCR=0x50;

SPSR=0x00;

// Timer/Counter 1 initialization

// Clock source: System Clock

// Clock value: 15.625 kHz

// Mode: CTC top=OCR1A

// OC1A output: Discon.

// OC1B output: Discon.

// Noise Canceler: Off

// Input Capture on Falling Edge

// Timer 1 Overflow Interrupt: Off

// Input Capture Interrupt: Off

// Compare A Match Interrupt: On

// Compare B Match Interrupt: Off

TCCR1A=0x00;

TCCR1B=0x0D;

TCNT1H=0x00;

TCNT1L=0x00;

ICR1H=0x00;

ICR1L=0x00;

OCR1AH=0x00;

OCR1AL=0x25; // Quyet dinh toc do quet

OCR1BH=0x00;

OCR1BL=0x00;

DDRB=0xFF;

DDRC=0xFF;

SCL=1;

// Timer(s)/Counter(s) Interrupt(s) initialization

TIMSK=0x40;

// Global enable interrupts

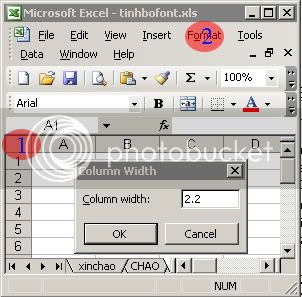
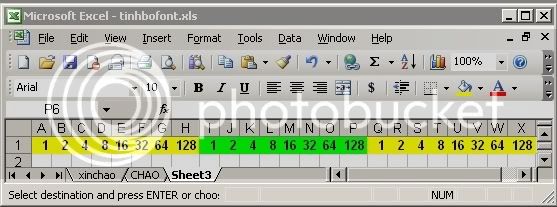
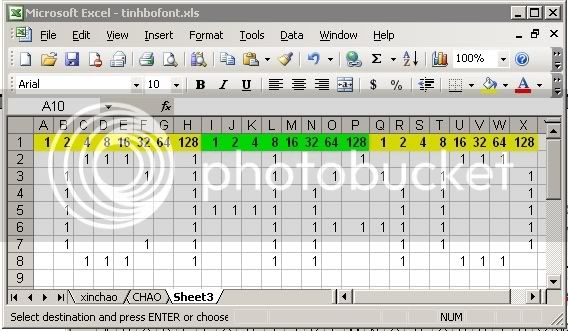
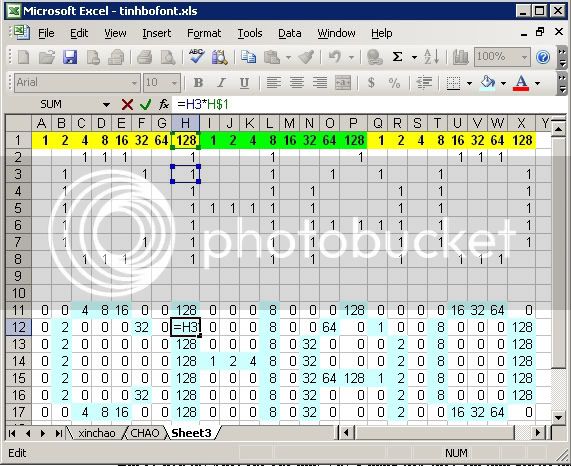
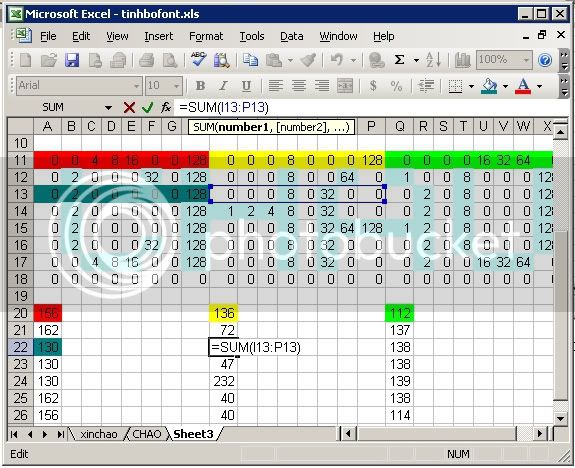
#asm("sei")

while (1){

}

}

Chúc vui vẻ!

Bây giờ ta bắt đầu tạo chữ “CHAO” bằng chương trình Excel nhé.  
Trước hết em nhắc lại một xíu về số nhị phân nhé. Một byte có 8 bit. Mỗi bit có thể biểu diễn được hai trạng thái 0 và 1. 0 ứng với led tắt, 1 ứng với led sáng. Chúng ta đang làm một bảng led ma trận gồm 3 panels led 8x8. Như vậy, một hàng led ma trận của chúng ta hiện nay có 24 cột. Rõ ràng ta sẽ tốn 3 bytes = 24 bits để biểu diễn dữ liệu cho một hàng. Ta sẽ truyền dữ liệu cho các IC dịch 74HC595 của chúng ta như sau:  
- Muốn cho các led của một hàng sáng hết ta sẽ truyền 3 bytes đều mang giá trị là 0b1111111 (hệ nhị phân) = 255 (hệ cơ số 10).  
- Muốn cho một hàng không có led nào sáng ta sẽ truyền 3 bytes đều mang giá trị là 0b00000000 = 255.  
- Muốn cho một cột sáng xong lại đến một cột tắt ta truyền 3 bytes là 0b10101010 = 2^7 + 2^5 + 2^3 + 2^1 = 128 + 32 + 8 + 2 = 178;  
Bác nào con lăn tăn về việc chuyển đối số nhị phân thì xem lại sách vở một chút nhé. Các bác chỉ cần nhớ là một byte có 8 bits: bit 7, bit 6, bit 5… và bit 0. Khi chuyển đổi từ nhị phân sang thập phân cứ hễ bit nào bằng 1 thì ta sẽ phải cộng thêm 2 lũy thừa của bit đó. Như ở ví dụ trên số nhị phân 0b10101010 có các bit số 7, 5, 3 và 1 bằng 1 nên ta sẽ phải cộng 128, 32, 8 và 2 lại với nhau.  
Như vậy ta đã sẵn sàng mở Excel ra rồi nhỉ? Nhấn vào (1) để chọn toàn bộ các ô trong sheet sau đó chọn (2) Format - Column – Width nhập vào 2.2:  
  
  
  
Hàng đầu tiên nhập vào các giá trị của các trọng số 0 đến 7 của 3 bytes ứng với 24 cột. 1 tức là 2 mũ 0, 2 là 2 mũ 1, 4= 2^2 … và 128=2^7  
  
  
  
Bây giờ là lúc các bác có thể thỏa sức sáng tạo ra bộ font mà các bác muốn. Lưu ý là ta chỉ có 8 dòng để tạo bộ font thôi. Em thấy cần 7 dòng là sẽ được một bộ font cân đối. Vậy ta cứ để dư một dòng vậy. Ở chỗ nào sau này các bác muốn led sáng thì gõ số 1 vào. Chỗ nào led tắt thì để trống. Đây là sản phẩm của em sau 2 phút:  
  
  
  
Em có hoa tay phết các bác nhỉ. Vậy 8 hàng tiếp theo em tính giá trị thập phân tại các bit 1 nhé:  
  
  
  
Việc tiếp theo là em sẽ tính toán lại các byte (24 bytes) đại diện cho chữ “CHAO” này. Cũng rất rất đơn giản thôi. Ta cứ cộng mỗi 8 số lại với nhau như hình dưới.  
  
  
  
Như vậy ta có được 24 bytes dùng để hiện chữ “CHAO” rồi. Các byte lần lượt là 156, 136, 112 (cho việc hiện hàng 1), 162, 72, 137 (hàng 2)… 156, 40, 114 (hàng 7), 0, 0, 0 (hàng 8).  
  
Việc cuối cùng là ta đưa nó vào trong một mảng, mảng này sẽ được cất trên ROM của chú AVR khi nạp chương trình luôn.

Code:

XINCHAO[]={156,136,112,162,72,137,130,40,138,130,47,138,130,232,139,162,40,138,156,40,114,0,0,0};

Vậy là xong cái việc tạo font. Sau này các bác có thể tự tạo cho mình bất cứ font chữ hay hình ảnh nào rồi.  
  
Bài tiếp theo em sẽ hướng dẫn các bác đưa cái sản phẩm tim óc nãy giờ của chúng ta lên màn hình led nhé.

Qua bài trước các bác đã nắm được phương pháp "tạo hình" rồi. Tức là nội dung sẽ hiển thị lên bảng led thế nào do ta thiết kế. Nó có thể là kí tự, bông hoa hay bất cứ cái gì. Và công việc bây giờ là hiển thị nó lên màn hình.  
  
Data của chúng ta ban đầu lưu ở trong Flash ROM. Do đó em đã khai báo

Code:

unsigned char flash XINCHAO[]={156,136,112,162,72,137,130,40,138,130,47,138,130,232,139,162,40,138,156,40,114,0,0,0};

Em sẽ khai báo một mảng trong RAM, mảng này sẽ là bộ nhớ màn hình. Timer 1 của chúng ta sẽ liên tục đọc mảng này và đưa ra màn hình led. Sau này ta muốn làm hiệu ứng hay bất cứ điều gì thì ta chỉ việc thay đổi cái mảng này

Code:

unsigned char DMem[24]; // Bo nho man hinh, 3 byte dau cho hang 1, 3 byte tiep theo cho hang 2 ...

Ban đầu em lấy âm bản của data lưu trong flash. Các bác có thể không thèm lấy âm bản và coi nó như là một hiệu ứng cũng được. Lúc đó chữ "CHAO" sẽ không sáng nhưng background của nó thì lại sáng.

Code:

for (i=0;i<=23;i++){

DMem[i]=~XINCHAO[i];

}

Trong chương trình ngắt Timer1 ta đưa bộ nhớ màn hình ra màn hình led. Lưu ý quét mỗi hàng thì ta lấy ra 3 bytes . Em không muốn sử dụng mảng 2 chiều cho bộ nhớ màn hình là có nguyên nhân. Các bác thử đoán xem nhé.

Code:

// Timer 1 output compare A interrupt service routine

interrupt [TIM1\_COMPA] void timer1\_compa\_isr(void){

spi(DMem[r\*3+2]); //byte thứ 2 ở hàng thứ r

spi(DMem[r\*3+1]); // byte thứ 1 ở hàng thứ r

spi(DMem[r\*3]); // byte thứ 0 ở hàng thứ r

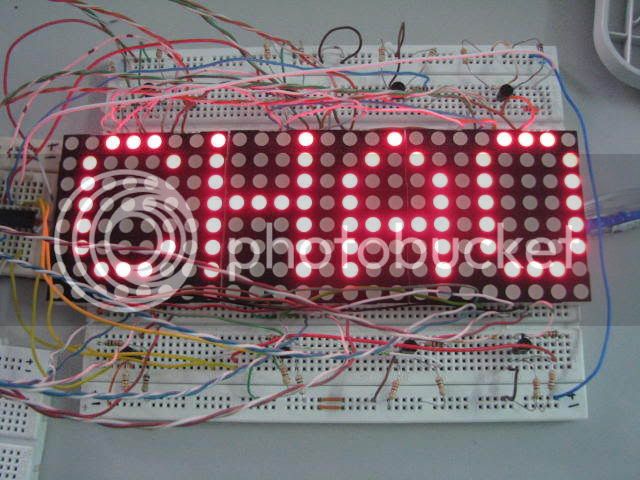
LatchData();

PORTC=r+8; // để Enable chú 74138

r++;

if (r==8) r=0;

}

Vậy là xong, các bác hãy biên dịch và nạp vào AVR. Kết quả sẽ như bài viết đầu tiên của em  
  
  
  
Đây là full source code của chúng ta cho đến thời điểm này

Code:

6. Hien chu CHAO (giong chuong trinh XINCHAO)

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

This program was produced by the

CodeWizardAVR V1.25.8 Standard

Automatic Program Generator

© Copyright 1998-2007 Pavel Haiduc, HP InfoTech s.r.l.

http://www.hpinfotech.com

Project :

Version :

Date : 7/27/2008

Author : Nguyen Hai Ha

Company : www.softviet.net

Comments:

Chip type : ATmega8515L

Program type : Application

Clock frequency : 16,000000 MHz

Memory model : Small

External SRAM size : 0

Data Stack size : 128

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

#include <mega8515.h>

#include <delay.h>

// SPI functions

#include <spi.h>

#define SCK PORTB.7

#define DATA PORTB.5

#define SCL PORTB.3

#define B PORTC.1;

#define C PORTC.2;

#define OE PORTC.3;

char lenX=9;

char NoLed=3;

unsigned char flash XINCHAO[]={156,136,112,162,72,137,130,40,138,130,47,138,130,232,139,162,40,138,156,40,114,0,0,0};

unsigned char DMem[24]; // Bo nho man hinh, 3 byte dau cho hang 1, 3 byte tiep theo cho hang 2 ...

unsigned char r=0;

unsigned char i;

void LatchData(){

SCL=0;

SCL=1;

}

// Timer 1 output compare A interrupt service routine

interrupt [TIM1\_COMPA] void timer1\_compa\_isr(void){

spi(DMem[r\*3+2]);

spi(DMem[r\*3+1]);

spi(DMem[r\*3]);

LatchData();

PORTC=r+8;

r++;

if (r==8) r=0;

}

void main(void){

// Timer/Counter 1 initialization

// Clock source: System Clock

// Clock value: 15.625 kHz

// Mode: CTC top=OCR1A

// OC1A output: Discon.

// OC1B output: Discon.

// Noise Canceler: Off

// Input Capture on Falling Edge

// Timer 1 Overflow Interrupt: Off

// Input Capture Interrupt: Off

// Compare A Match Interrupt: On

// Compare B Match Interrupt: Off

TCCR1A=0x00;

TCCR1B=0x0D;

TCNT1H=0x00;

TCNT1L=0x00;

ICR1H=0x00;

ICR1L=0x00;

OCR1AH=0x00;

OCR1AL=0x25; // Quyet dinh toc do quet

OCR1BH=0x00;

OCR1BL=0x00;

DDRB=0xFF;

DDRC=0xFF;

SCL=1;

// Timer(s)/Counter(s) Interrupt(s) initialization

TIMSK=0x40;

// Global enable interrupts

#asm("sei")

// SPI initialization

// SPI Type: Master

// SPI Clock Rate: 4000,000 kHz

// SPI Clock Phase: Cycle Half

// SPI Clock Polarity: Low

// SPI Data Order: MSB First

SPCR=0x50;

SPSR=0x00;

for (i=0;i<=23;i++){

DMem[i]=~XINCHAO[i];

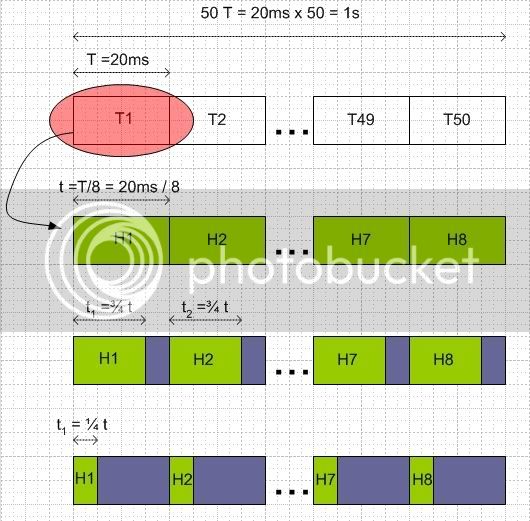
}

while (1){

}

}

Bài này đã xong, các bác hãy tùy biến flash data để tạo ra các hình ảnh lạ mắt trên màn hình led nhé. Bài sau em hướng dẫn hiệu ứng làm chữ "CHAO" của ta tối dần và sáng dần. Các bác đón xem nhé.

Thay đổi độ sáng màn hình led  
  
Một bảng led chuyên nghiệp sẽ có thêm phần điều chỉnh độ sáng. Ban ngày thì cần sáng hơn nhưng buổi tối thì giảm xuống để không gây cảm giác chói mắt khó chịu. Việc điều khiển được độ sáng có thể dùng để tạo hiệu ứng chữ hiện ra từ từ hoặc biến mất từ từ. Có các IC chuyên dụng cho phép thay đổi dòng cấp cho led để thay đổi độ sáng. Còn ở đây chúng ta sẽ dùng phần mềm.  
Nguyên lý thay đổi độ sáng như sau:  
  
  
  
Giả sử tần số quét của ta là 50Hz, tức là 1s ta sẽ quét 50 lần do đó 1 lần quét của chúng ta sẽ mất T =20 ms và 50T=1s.  
  
Trong thời gian T ta sẽ phải sáng lần lượt tất cả các hàng, ở đây số hàng của ta là 8. Do đó mỗi hàng sẽ sàng trong thời gian là t=T/8 = 2.5 ms. Đây sẽ là mức sáng nhất ta có thể cấp cho màn hình led.  
  
Vậy nếu chúng ta có thể chỉ cho mỗi hàng sáng trong một thời gian ít hơn t=2.5ms (nhưng vẫn giữ nguyên tần số quét 50Hz) thì độ sáng sẽ giảm.  
  
Hình trên mô tả việc thay đổi độ sáng bằng 3/4 hay 1/4 so với ban đầu.  
  
Đây là bài tập nên em không có source code cho phần này mà chỉ mô tả nguyên tắc làm. Các bác hãy sửa lại chương trình theo ý tưởng thay đổi độ sáng này nhé.

Qua các bài viết trước em nghĩ chúng ta đã hiểu khá nhiều về led ma trận rồi. Sau bài viết này em xin dừng lại ở đây. Bài viết cuối cùng em trình bày về hiệu ứng cho chữ "CHAO" chay từ trên xuống dưới. Còn hiệu ứng chạy từ phải qua trái em đã từng giới thiệu rồi các bác quan tâm có thể tham khảo ở đây:  
  
<http://dientuvietnam.net/forums/showthread.php?p=90829>  
  
Từ trước đến nay khi quét thì khi quét đến hàng 1 thì ta sẽ đưa data của hàng 1 ra, quét đến hàng 2 thì data của hàng 2.... và quét đến hàng cuối cùng thì ta sẽ đưa data của hàng cuối cùng.  
  
Đọc đến chỗ này có bác lại bảo em lẩm cẩm. Quét đến hàng 1 thì không đưa data của hàng 1 ra thì đưa cái gì không lẽ đưa ra hàng 8. Ấy, chính là ở chỗ đó. Nếu quét đến hàng 1 ta đưa data của hàng thứ 8, đến hàng 2 ta đưa data của hàng thứ 7... đến hàng thứ 8 ta lại đưa data của hàng thứ 1. Vậy ta được cái gì nhỉ?  
  
Hihihi, ta được cái chữ "CHAO" bị lộn ngược đúng không. Mấu chốt là ở chỗ đó. Ta muốn dịch chuyển cái chữ "CHAO" từ dưới lên trên ta cũng sẽ làm tương tự. Lúc quét hàng 1 ta đưa data hàng 2, quét hàng 2 lại đưa data hàng 3.... Rõ ràng như vậy thì chữ "CHAO" của ta bị xích lên trên....  
  
Và đây là source code:

Code:

7. Chạy chữ CHAO từ trên xuống

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

This program was produced by the

CodeWizardAVR V1.25.8 Standard

Automatic Program Generator

© Copyright 1998-2007 Pavel Haiduc, HP InfoTech s.r.l.

http://www.hpinfotech.com

Project :

Version :

Date : 7/27/2008

Author : Nguyen Hai Ha

Company : www.softviet.net

Comments:

Chip type : ATmega8515L

Program type : Application

Clock frequency : 16,000000 MHz

Memory model : Small

External SRAM size : 0

Data Stack size : 128

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

#include <mega8515.h>

#include <delay.h>

// SPI functions

#include <spi.h>

#define SCK PORTB.7

#define DATA PORTB.5

#define SCL PORTB.3

#define B PORTC.1;

#define C PORTC.2;

#define OE PORTC.3;

char lenX=9;

char NoLed=3;

unsigned char Pattern1[8]={0,1,2,3,4,5,6,7};

unsigned char flash XINCHAO[]={156,136,112,162,72,137,130,40,138,130,47,138,130,232,139,162,40,138,156,40,114,0,0,0};

unsigned char DMem[24]; // Bo nho man hinh, 3 byte dau cho hang 1, 3 byte tiep theo cho hang 2 ...

unsigned char r=0;

unsigned char i,k;

void LatchData(){

SCL=0;

SCL=1;

}

// Timer 1 output compare A interrupt service routine

interrupt [TIM1\_COMPA] void timer1\_compa\_isr(void){

spi(DMem[r\*3+2]);

spi(DMem[r\*3+1]);

spi(DMem[r\*3]);

LatchData();

PORTC=Pattern1[r]+8;

r++;

if (r==8) r=0;

}

void main(void){

// Timer/Counter 1 initialization

// Clock source: System Clock

// Clock value: 15.625 kHz

// Mode: CTC top=OCR1A

// OC1A output: Discon.

// OC1B output: Discon.

// Noise Canceler: Off

// Input Capture on Falling Edge

// Timer 1 Overflow Interrupt: Off

// Input Capture Interrupt: Off

// Compare A Match Interrupt: On

// Compare B Match Interrupt: Off

TCCR1A=0x00;

TCCR1B=0x0D;

TCNT1H=0x00;

TCNT1L=0x00;

ICR1H=0x00;

ICR1L=0x00;

OCR1AH=0x00;

OCR1AL=0x25; // Quyet dinh toc do quet

OCR1BH=0x00;

OCR1BL=0x00;

DDRB=0xFF;

DDRC=0xFF;

SCL=1;

// Timer(s)/Counter(s) Interrupt(s) initialization

TIMSK=0x40;

// Global enable interrupts

#asm("sei")

// SPI initialization

// SPI Type: Master

// SPI Clock Rate: 4000,000 kHz

// SPI Clock Phase: Cycle Half

// SPI Clock Polarity: Low

// SPI Data Order: MSB First

SPCR=0x50;

SPSR=0x00;

for (i=0;i<=23;i++){

DMem[i]=~XINCHAO[i];

}

while (1){

k=Pattern1[7];

for (i=7;i>=1;i--){

Pattern1[i]=Pattern1[i-1];

}

Pattern1[0]=k;

delay\_ms(300);

}

}

Vậy còn hiệu ứng gì nữa không nhỉ? Thêm một vài cái nếu sử dụng nguyên tắc này. Xóa chữ "CHAO" từ trên xuống hay dưới lên này, Hiện chữ từ trên xuống, dưới lên, trong ra ngoài, ngoài vào trong... Nói chung tùy vào óc tưởng tượng của các bác.  
  
Em xin kết thúc tutorial này tại đây. Chúc các bác thành công.