TRƯỜNG ĐẠI HỌC VINH

VIỆN KỸ THUẬT CÔNG NGHỆ

=====\*\*\*=====



**BÀI TIỂU LUẬN**

MÔN: **THIẾT KẾ HỆ THỐNG NHÚNG**

SV thực hiện: Lê Khắc Quân

MSSV: 1755252021600046

Lớp: 58K – Kỹ thuật ĐK & TĐH

GV hướng dẫn: TS. Mai Thế Anh

Học phần: Thiết Kế Hệ Thống Nhúng

*Nghệ An, 2021*

LỜI NÓI ĐẦU

Hệ thống nhúng (tiếng Anh: embedded system) là một thuật ngữ để chỉ một hệ thống có khả năng tự trị được nhúng vào trong một môi trường hay hệ thống mẹ. Đó là các hệ thống tích hợp cả phần cứng và phần mềm phục vụ các bài toán chuyên dụng trong nhiều lĩnh vực công nghiệp, tự động hoá điều khiển, quan trắc và truyền tin. Đặc điểm của các hệ thống nhúng là hoạt động ổn định và có tính năng tự động hoá cao. Hệ thống nhúng thường được thiết kế để thực hiện một chức năng chuyên biệt nào đó. Khác với các máy tính đa chức năng, chẳng hạn như máy tính cá nhân, một hệ thống nhúng chỉ thực hiện một hoặc một vài chức năng nhất định, thường đi kèm với những yêu cầu cụ thể và bao gồm một số thiết bị máy móc và phần cứng chuyên dụng mà ta không tìm thấy trong một máy tính đa năng nói chung. Vì hệ thống chỉ được xây dựng cho một số nhiệm vụ nhất định nên các nhà thiết kế có thể tối ưu hóa nó nhằm giảm thiểu kích thước và chi phí sản xuất. Các hệ thống nhúng thường được sản xuất hàng loạt với số lượng lớn. Hệ thống nhúng rất đa dạng, phong phú về chủng loại. Đó có thể là những thiết bị cầm tay nhỏ gọn như đồng hồ kĩ thuật số và máy chơi nhạc MP3, hoặc những sản phẩm lớn như đèn giao thông, bộ kiểm soát trong nhà máy hoặc hệ thống kiểm soát các máy năng lượng hạt nhân. Xét về độ phức tạp, hệ thống nhúng có thể rất đơn giản với một vi điều khiển hoặc rất phức tạp với nhiều đơn vị, các thiết bị ngoại vi và mạng lưới được nằm gọn trong một lớp vỏ máy lớn. Các thiết bị PDA hoặc máy tính cầm tay cũng có một số đặc điểm tương tự với hệ thống nhúng như các hệ điều hành hoặc vi xử lý điều khiển chúng nhưng các thiết bị này không phải là hệ thống nhúng thật sự bởi chúng là các thiết bị đa năng, cho phép sử dụng nhiều ứng dụng và kết nối đến nhiều thiết bị ngoại vi.

BÀI TẬP TIỂU LUẬN

Câu 1:

Sử dụng vi điều khiển AT89C52, mô phỏng trên phần mềm proteus và lập trình thực hiện nhiệm vụ sau:

1. Hiển thị số 00 lên 2 LED 7 thanh nối vào cổng P2 theo phương pháp quét LED.

**Bài Làm:**

- Chương trình trên KeilC:

#include <REGX51.H>

void delay(int n)

{

int i;

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

}

void hienthi(int chuc,donvi)

{

P3\_1 = 1;

P2 = chuc;

P3\_1 = 0;

P3\_0 = 1;

P2 = donvi;

P3\_0 = 0;

}

void main()

{

while(1)

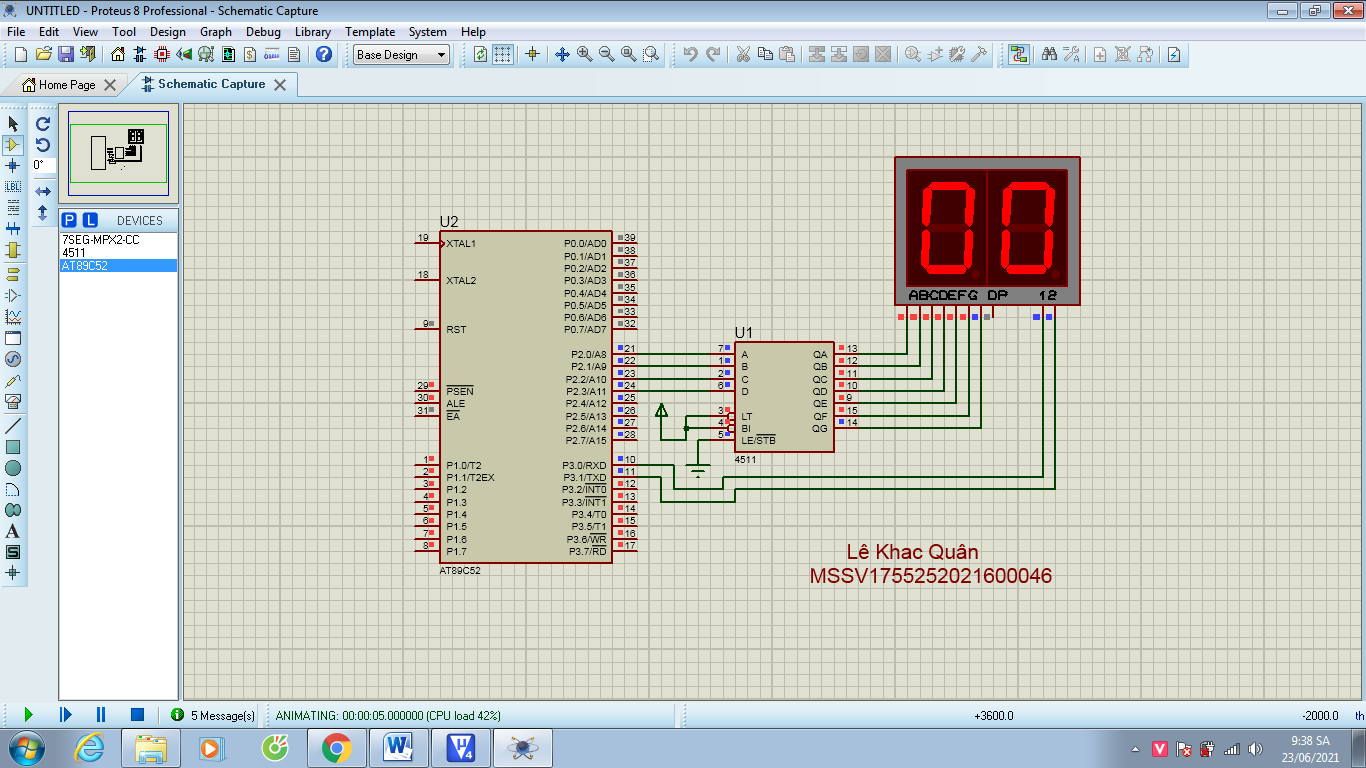
{

hienthi(0,0);

}

}

- Mô phỏng trên Proteus:



2. Tăng số đếm sau mỗi 500ms, nếu số đếm bằng “SBD+20” thì dừng lại (sử dụng timer để định thời gian).

**Bài Làm:**

- Chương trình trên KeilC:

#define ledchuc P3\_0

#define leddonvi P3\_1

#define leddata P2

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

void hienthi(char so, unsigned int time)

{

char chuc,donvi,i;

time = time/10;

chuc = so/10;

donvi = so%10;

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

{

ledchuc = 0;

leddata = so[chuc];

delay\_ms(10);

ledchuc = 1;

leddonvi = 0;

leddata = so[donvi];

delay\_ms(10);

leddonvi = 1;

}

}

void main()

{

while(1)

{

int c;

for(c=0;c<51;c++)

{

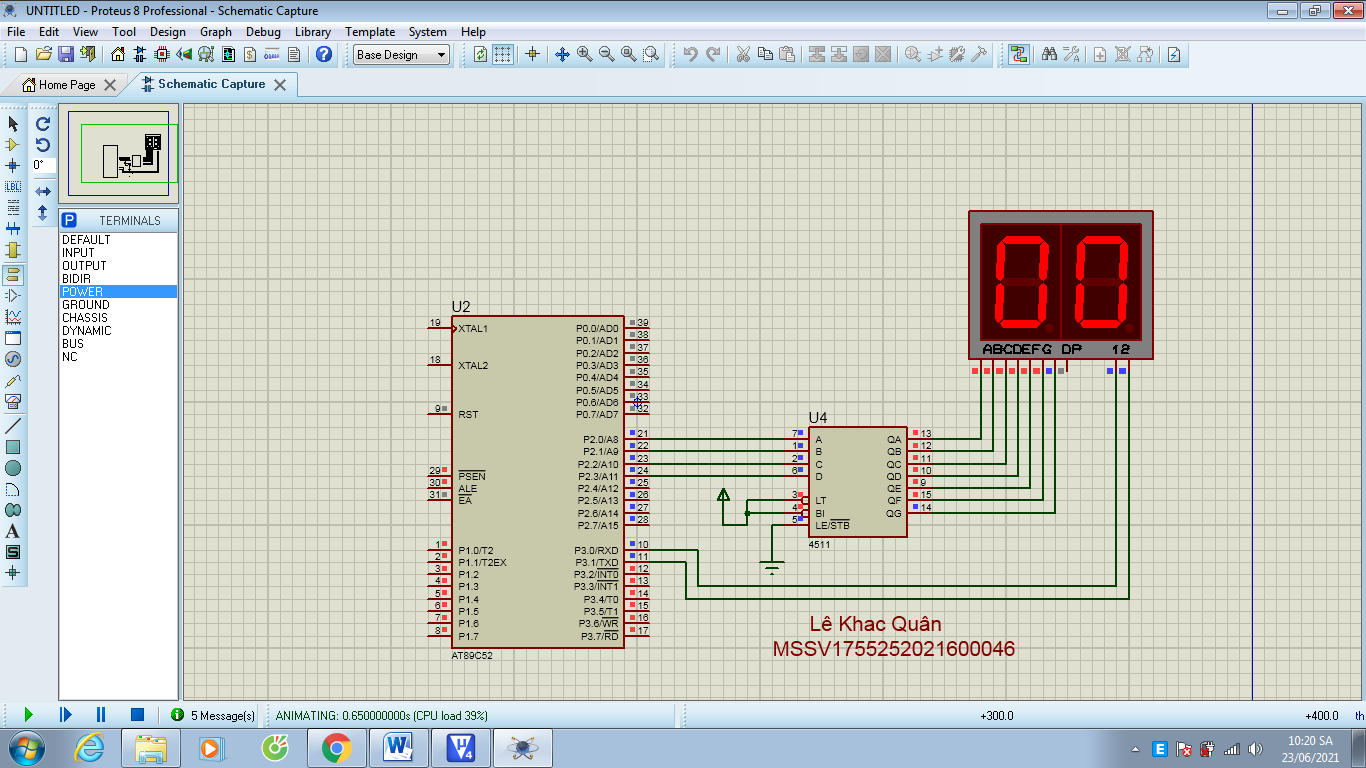
hienthi(c,1000);

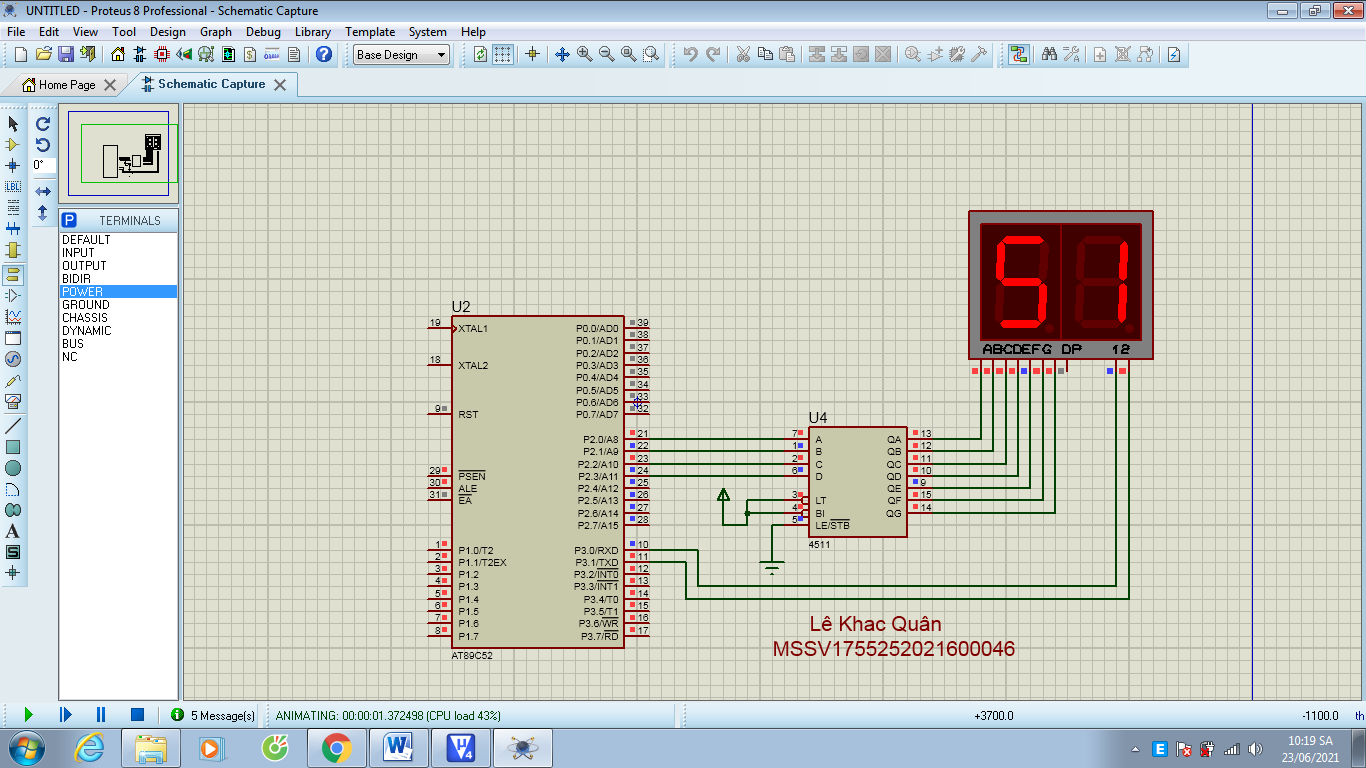
}

}

}

- Mô phỏng trên Proteus: Đếm 00-51(SBD 31+20).





Câu 2:

Sử dụng vi điều khiển AT89C52, mô phỏng trên phần mềm Proteus và lập trình thực hiện các nhiệm vụ sau:

1. Cấu hình ngắt ngoài INT0 ở chế độ ngắt sườn xuống.

**Bài Làm:**

- Chương trình trên KeilC:

#include <REGX52.H>

#define LED\_PORT P2

sbit key = P3\_2 // noi nut bam voi chanP3.2

void delay\_ms(unsigned int t)

{

unsigned int x;

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

{;}

}

// ham gan gia tri dem voi ma hex

unsigned char code code7seg[] = {0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8, 0x80, 0x90};

void main()

{

EX0 = 1; // cho phep ngat ngoai 0

EA = 1; // cho phep su dung ngat ngoai

}

void trinhNgat() interrupt 0 // khai bao trinh phuc vu ngat 0

{

unsigned char dem;

while(1)

{

if(key==0)

{

// cau lenh chong doi phim

delay\_ms(100);

if(key==0)

{

dem++;

delay\_ms(10000000);

if(dem>9)

{

dem = 0;

}

LED\_PORT = code7seg[dem];

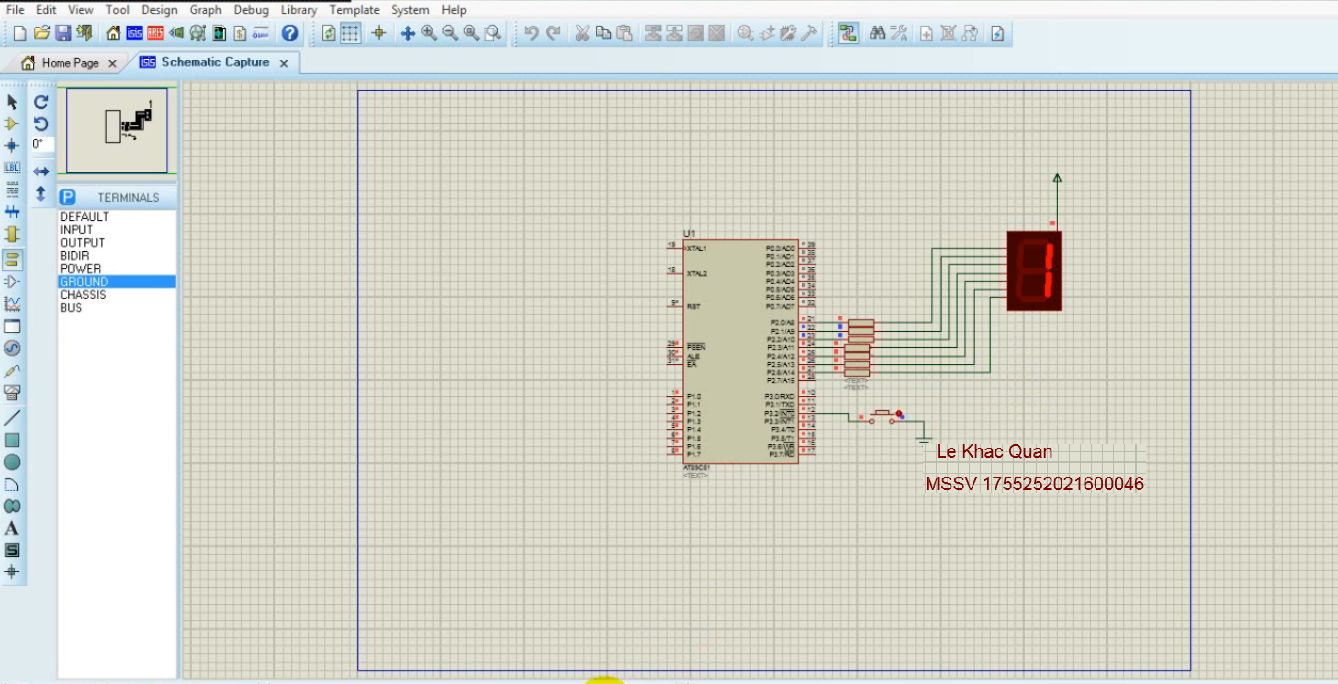
}

}

}

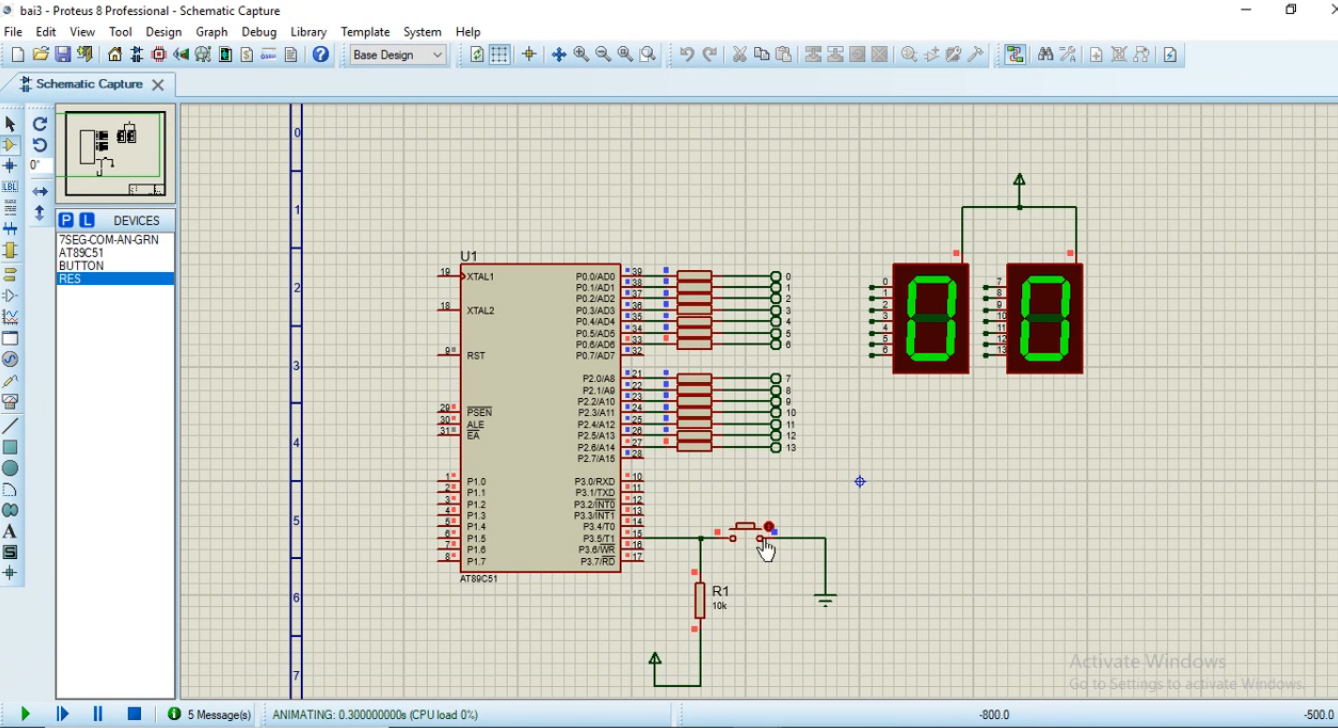
}

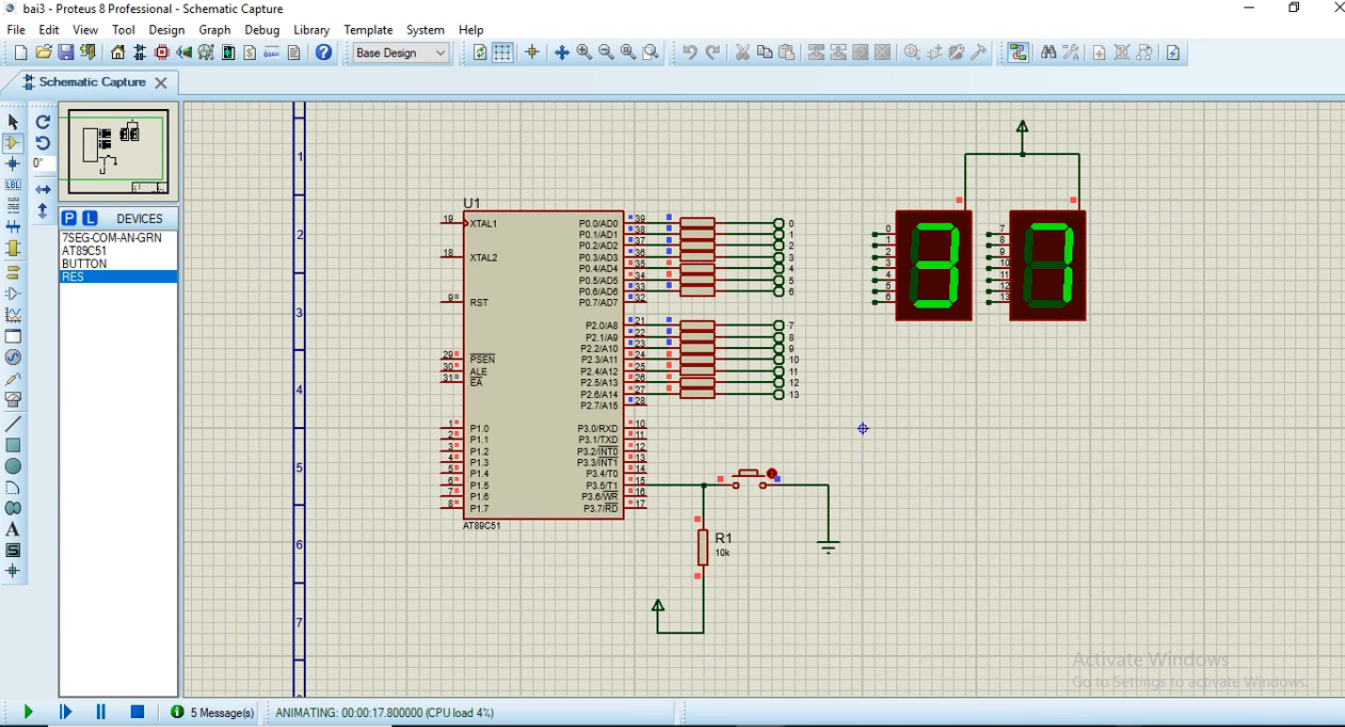
- Mô phỏng trên proteus:

****

2. Nếu sô lần nút bấm nút nối vào chân INT0 được bấm, hiển thị kết quả lên 2 LED 7 thanh nối vào cổng P2 theo phương pháp quét LED (nếu số lần bấm bằng SBD+10 thì quay về bằng 0).

- Mô phỏng trên Proteus:





Câu 3:

Sử dụng vi điều khiển AT89C52 thực hiện các nhiệm vụ sau:

1. Vẽ sơ đồ mô phỏng trên Proteus ghép nối với LCD theo chế độ 4bit, hiển thị họ tên, mã số sinh viên lên LCD.

**Bài Làm:**

- Chương trình trên mikroC:

sbit LCD\_RS at P2\_5\_bit;

sbit LCD\_RW at P2\_6\_bit;

sbit LCD\_EN at P2\_7\_bit;

sbit LCD\_D4 at P3\_4\_bit;

sbit LCD\_D5 at P3\_5\_bit;

sbit LCD\_D6 at P3\_6\_bit;

sbit LCD\_D7 at P3\_7\_bit;

void main()

{

LCD\_RW = 0;

Lcd\_Init();

Lcd\_Cmd(\_LCD\_CLEAR);

Lcd\_Cmd(\_LCD\_CURSOR\_OFF);

Lcd\_Out(1,1,"LEKHACQUAN");

Lcd\_Out(2,1,"1755252021600046");

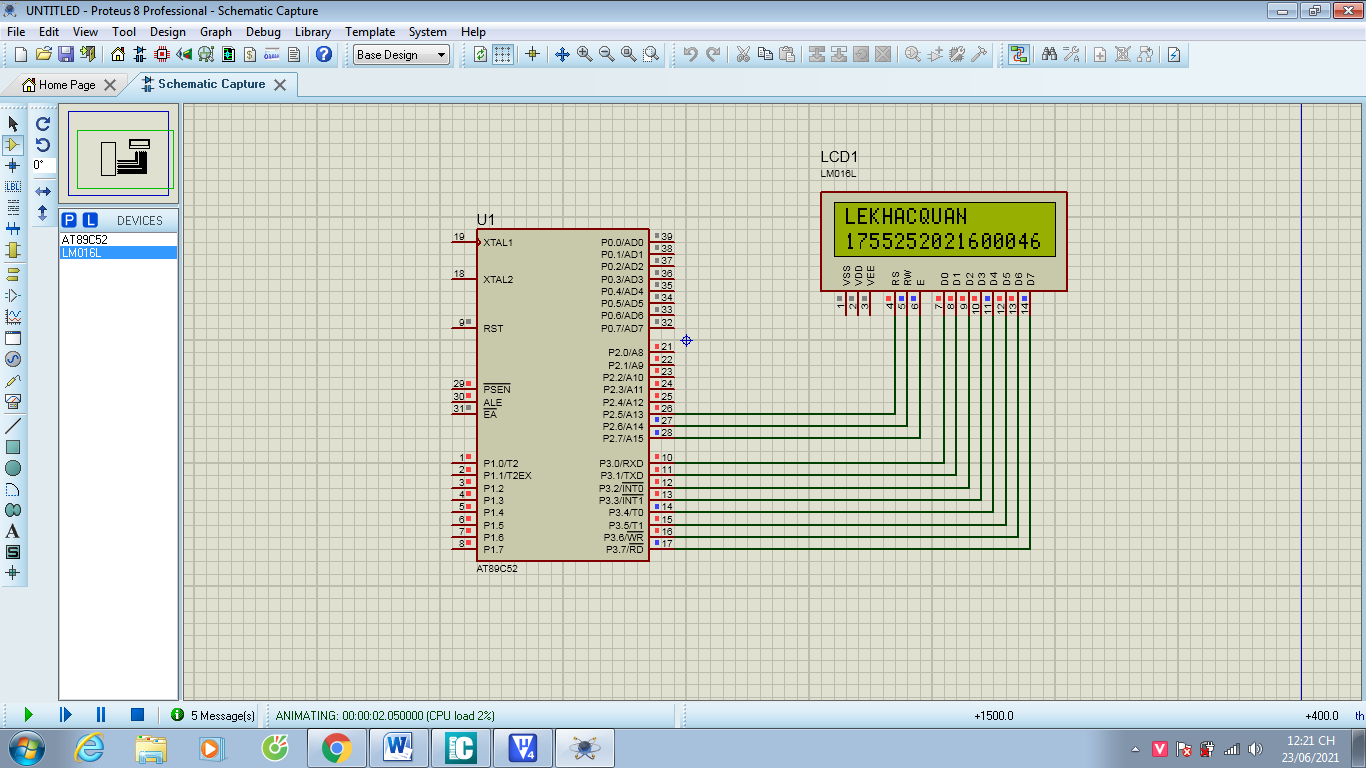
while(1)

{

}

}

- Mô phỏng trên Proteus:



2. Vẽ sơ đồ mô phỏng trên Proteus, lập trình hiển thị “ họ tên, mã số sinh viên” qua chuẩn truyền thong UART.

**Bài Làm:**

- Chương trình trên mikroC:

char rd;

void main(){

UART1\_Init(1200);

Delay\_ms(100);

UART1\_Write\_Text("LeKhacQuan1755252021600046");

while(1)

{

if(UART1\_Data\_Ready()==1)

{

rd=UART1\_Read();

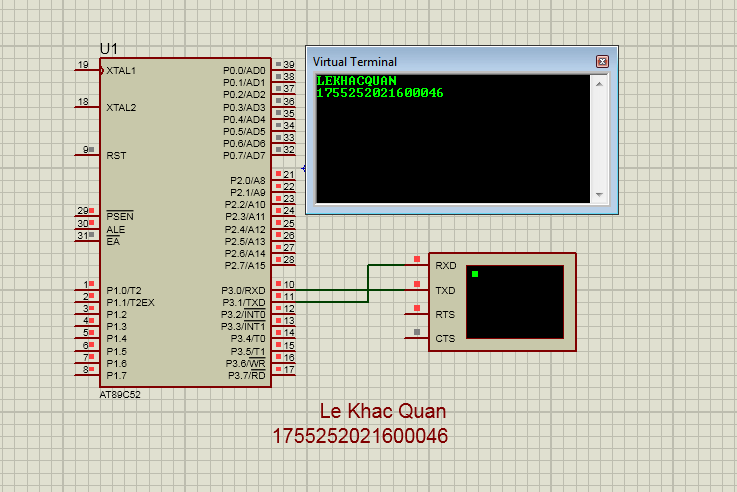
UART\_Write(rd);

}

}

}

- Mô phỏng trên Proteus:



Câu 4:

Sử dụng vi điều khiển AT89C52 vẽ sơ đồ mô phỏng trên proteus ghép nối với LED D1 qua cổng P1.2, BUTTON B1 qua cổng P1.3. Sử dụng hệ điều hành RTX51 lập trình ngắt USART, tast BUTTON, tast LED. Thực hiện gửi “signal” từ ngắt USART và task BUTTON đến tast LED. Task LED thực hiện đảo trạng thái của Led D1 khi nhận được tín hiệu task khác gửi tới.

**Bài Làm:**

- Chương trình trên KeilC:

#include <REGX51.H>

#include <RTX51TNY.H> //Su dung thu vien RTX51 Tiny Real-Time

#define INIT 0 //Dinh nghia INIT = 0

#define DO 1 //Dinh nghia DO = 1

#define BUTT 2 //Dinh nghia BUTTTON = 2

sbit LED\_DO = P1^0; //Dinh nghia chan LED\_DO

sbit BUTTON = P1^1; //Dinh nghia chan BUTTON

void USART(void) interrupt 4 //Ngat nhan USART

{

if(RI) //Flag nhan duoc ki tu

{ //Clear flag

RI=0; //Nhan ki tu

isr\_send\_signal(DO); //Gui signal cho task DO

}

}

//=========Ham Start up==========

void Startup(void) \_task\_ INIT

{

SCON=0x52; //USART che do 1

TMOD=0x21; //Timer 1 mode 2

TH1=TL1=-3; //baudrate 9600

TR1=1;

IE=0x90; //Ngat USART

os\_create\_task (DO); //Tao Task\_Led\_Do

os\_create\_task (BUTT); //Tao Task BUTTON

os\_delete\_task (INIT); //Xoa Task hien tai (Task 0)

}

void Task\_Led\_Do(void) \_task\_ DO

{

while(1)

{

os\_wait2(K\_SIG ,50); //Cho signal voi time out 50 ticks

LED\_DO ^= 1; //Dao trang thai Led Do

}

}

void Task\_BUTTON(void) \_task\_ BUTT

{

while(1)

{

if(BUTTON == 0) //Nhan nut nhan = 0

{

os\_send\_signal(DO); //Gui signal cho task DO

while(BUTTON==0); //Cho nut nhan = 1(Chong nhieu)

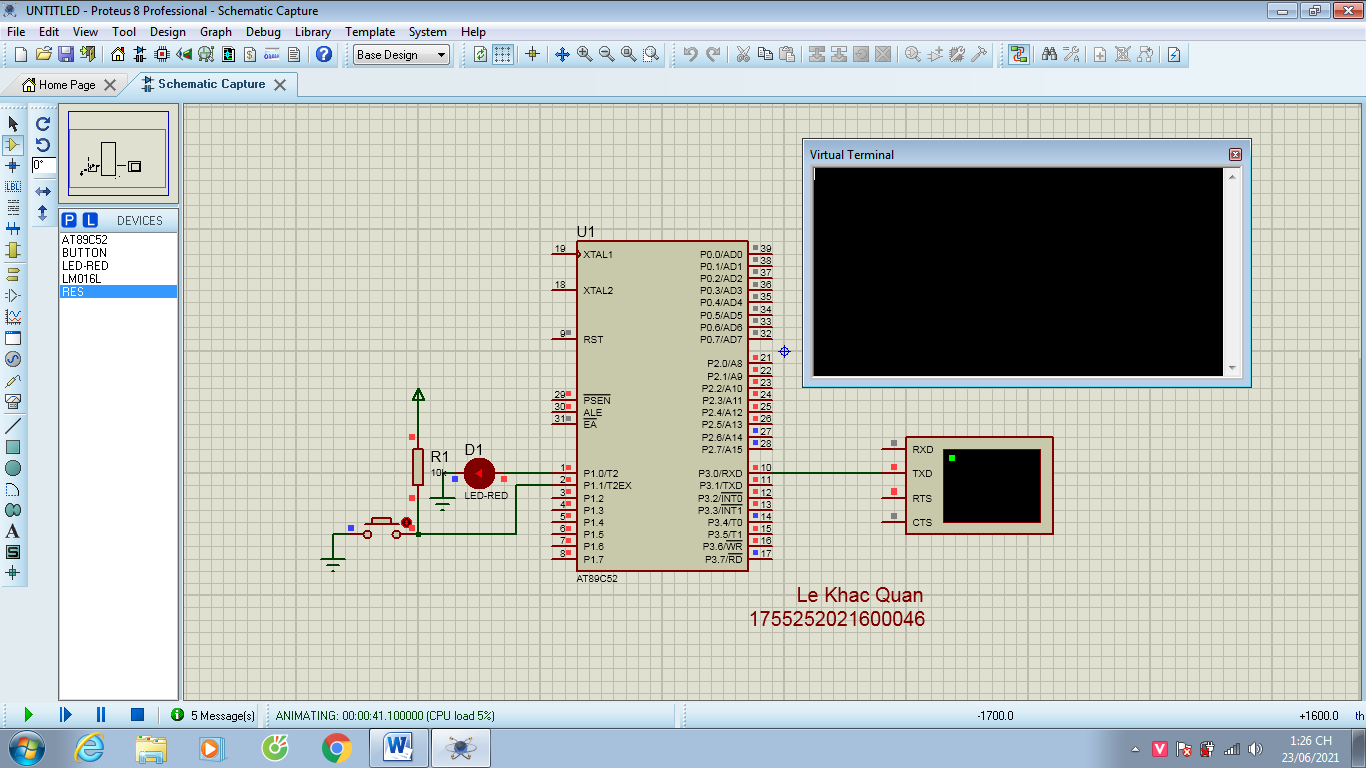
}

os\_wait2(K\_TMO, 10); //Cho 10 ticks = 100ms

}

}

- Mô phỏng trên Proteus:



Câu 5:

Hãy trình bày:

1. Quy trình thiết kế hệ thống nhúng sử dụng vi điều khiển

2. Hệ điều hành thời gian thực (RTOS). Ưu điểm, nhược điểm và ứng dụng của hệ điều hành thời gian thực trong thiết kế các hệ thống nhúng.

**Bài làm:**

1. Quy trình thiết kế hệ thống nhúng sử dụng vi điều khiển

a. Thiết kế phần cứng hệ thống nhúng  
- Bộ xử lý chức năng đơn chuyên dụng  
- Bộ xử lý chức năng đơn tiêu chuẩn – ngoại vi  
- Bộ nhớ  
- Giao diện  
- Công nghệ IC

b. Lập trình cho hệ thống nhúng

- Các phần mềm lập trình  
- Kiểu Biến  
- Từ khóa  
- Hằng số  
- Các Toán tử  
- Các cấu trúc điều khiển  
- Hàm  
- Tiền xử lý và thư viện

2. Hệ điều hành thời gian thực RTOS

- Định nghĩa:

Một hệ điều hành thời gian thực (RTOS) là một hệ điều hành (OS) nhằm phục vụ các ứng dụng thời gian thực, với việc xử lý dữ liệu đầu vào mà không có sự chậm trễ của bộ đệm (buffer). Các yêu cầu về thời gian xử lý (bao gồm cả sự chậm trễ của hệ điều hành) được tính bằng phần mười của giây hoặc bằng thời gian ngắn hơn nữa. Một hệ thống thời gian thực là một hệ thống giới hạn thời gian với các các ràng buộc thời gian cố định được định nghĩa rõ ràng. Quá trình xử lý phải được thực hiện trong một khoản thời gian cố định, nếu không thì hệ thống sẽ gặp sự cố. Việc này có thể được thực hiện thông qua cơ chế hướng sự kiện (event-driven) hoặc chia sẻ thời gian (time-sharing). Các hệ thống hướng sự kiện sẽ chuyển đổi giữa các tác vụ (task) nhiệm vụ dựa trên độ ưu tiên của chúng trong khi các hệ thống chia sẻ thời gian sẽ chuyển đổi các tác vụ dựa trên ngắt của xung nhịp. Phần lớn các hệ điều hành thời gian thực đều sử dụng giải thuật pre-emptive scheduling (lập lịch trước).

- Đặc tính:

Một đặc điểm chính của một RTOS là mức độ nhất quán của nó đối với khoảng thời gian cần thiết để nhận và hoàn thành một tác vụ trong một ứng dụng; sự thay đổi là rung pha. Một hệ điều hành thời gian thực "cứng" có ít jitter hơn một hệ điều hành thời gian thực "mềm". Mục tiêu thiết kế chính không phải là thông lượng cao, mà là sự bảo đảm về một loại hiệu suất mềm hoặc cứng. Một RTOS thường hoặc nói chung đáp ứng được một thời hạn quy định là một hệ điều hành thời gian thực mềm, nhưng nếu nó có thể đáp ứng được một thời hạn xác định thì đó là một hệ điều hành thời gian thực cứng. Một RTOS có một thuật toán tiên tiến cho việc lập thời biểu (scheduling). Tính linh hoạt của bộ lập thời biểu cho phép dàn xếp hệ thống máy tính rộng rãi hơn về các ưu tiên quá trình, nhưng hệ điều hành thời gian thực thường xuyên hơn dành cho một bộ nhỏ các ứng dụng. Các yếu tố chính trong một hệ điều hành thời gian thực là độ trễ ngắt tối thiểu và độ trễ chuyển luồng (thread) tối thiểu; một hệ điều hành thời gian thực được đánh giá cao hơn về mức độ nhanh nhạy hoặc có thể dự đoán được nó như thế nào so với số lượng công việc nó có thể thực hiện trong một khoảng thời gian nhất định.

- Các yêu cầu với RTOS:

• Kích thước nhỏ (lưu trữ toàn bộ trong ROM)

• Sử dụng hệ thống ngắt

• Không nhất thiết phải có các cơ chế bảo vệ

• Tăng tốc độ truyền thông giữa các quá trình

• Khi các quá trình ứng dụng đang thực hiện thì các yêu cầu hệ thống điều hành có thể được thực hiện thông qua các lời gọi hàm thay vì sử dụng cơ chế ngắt mềm

- Quá trình lập lịch:

Mục đích: Đảm bảo yêu cầu về thời gian

• Lập lịch trước khi chạy (static): Xác định chính xác giản đồ thời gian cho các nhiệm vụ tại thời điểm thiết kế

• Lập lịch khi chạy chương trình (dynamic): Lập lịch được thực hiện tự động bởi RTOS, dựa trên sự ưu tiên.

- Các kỹ thuật lập lịch:

a. First Come First Serve (FCFS)

• Các quá tình được xử lý theo thứ tự mà nó xuất hiện yêu cầu và cho đến khi hoàn thành

• Cơ chế lập lịch này thuộc loại không ngắt được và có ưu điểm là dễ dàng thực thi

• Không phù hợp cho hoạt động đáp ứng thời gian thực

b. Shortest Job First (SJF)

• Tác vụ có thời gian thực thi ngắn nhất sẽ có quyền ưu tiên cao nhất và sẽ được phục vụ trước

• Không biết trước được thời gian thực thi của các tác vụ tham gia trong chương trình và thông thường phải áp dụng cơ chế tiên đoán và đánh giá dựa vào kinh nghiệm về các tác vụ thực thi trong hệ thống

• Có thể áp dụng cho các tác vụ cả loại ngắt được và không ngắt được. c. Rate Monotonic (RM)

• Phương pháp này dựa trên một số giả thiết sau:

- Tất cả các tác vụ tham gia hệ thống phải có deadline kiểu chu kỳ

- Tất cả các tác vụ độc lập với nhau

- Thời gian thực hiện của các tác vụ biết trước và không đổi

- Thời gian chuyển đổi ngữ cảnh thực hiện là rất nhỏ và có thể bỏ qua • Thuật toán RM được thực thi theo nguyên lý gán mức ưu tiên cho các tác vụ dựa trên chu kỳ của chúng (chu kỳ nhỏ thì mức ưu tiên cao) • Với các tác vụ chu kỳ không thay đổi thì RM sẽ là phương pháp lập lịch cho phép ngắt và mức ưu tiên cố định

• Sử dụng deadline của tác vụ như điều kiện ưu tiên để xử lý điều phối hoạt động

• Tác vụ có deadline gần nhất sẽ có mức ưu tiên cao nhất và các tác vụ có deadline xa nhất sẽ nhận mức ưu tiên thấp nhất

d. Minimum Laxity First (MLF)

• Cơ chế lập lịch này sẽ ưu tiên tác vụ nào còn ít thời gian còn lại để thực hiện nhất trước khi nó phải kết thúc để đảm bảo yêu cầu thực thi đúng

• Cơ chế lập lịch gán quyền ưu tiên động và dễ đạt được sự tối ưu về hiệu suất thực hiện và sự công bằng trong hệ thống

e. Round Robin (RR)

• Mỗi một tác vụ được xử lý/phục vụ trong một khoảng thời gian nhất định và lặp lại theo một chu trình xuyên suốt toàn bộ các tác vụ tham gia trong hệ thống

• Khoảng thời gian phục vụ cho mỗi tác vụ trong quá trình là một sự thoả hiệp giữa thời gian thực hiện của các tác vụ và thời gian thực hiện một chu trình