BỘ CÔNG THƯƠNG

TRƯỜNG CAO ĐẲNG CÔNG NGHIỆP VÀ XÂY DỰNG

BÀI GIẢNG MÔN HỌC

VI ĐIỀU KHIỂN

(Lưu hành nội bộ)

Người biên soạn: **Lưu Thế Mạnh**

Uông Bí, năm 2010

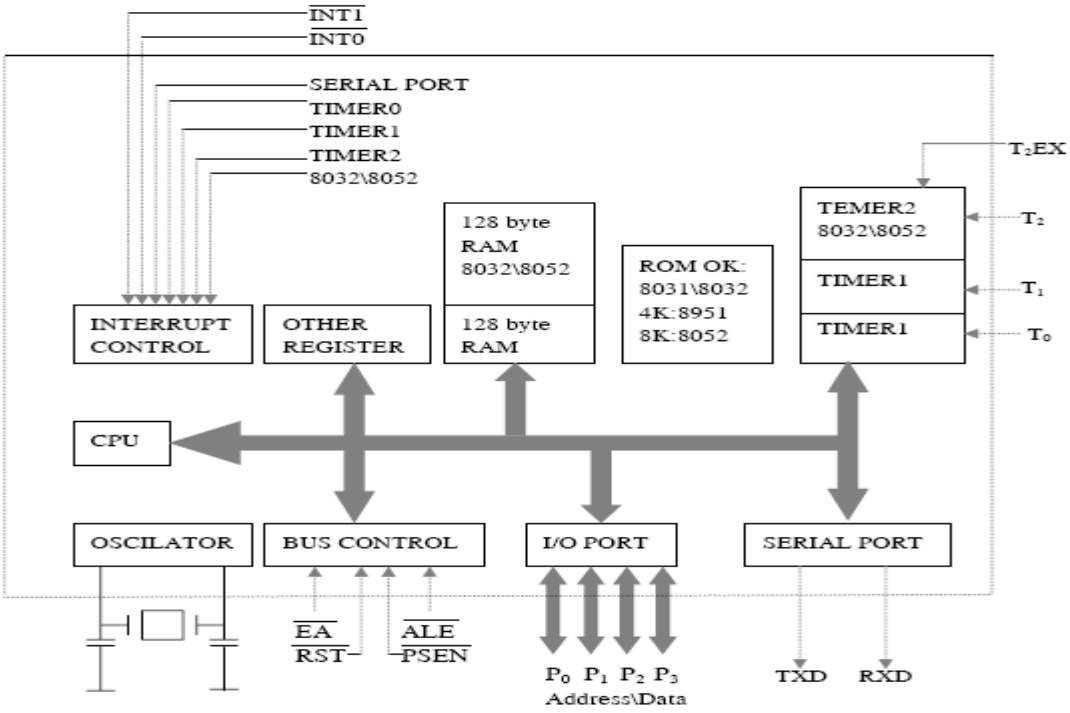
Bài 1 : Giới thiệu về vi điều khiển 8051.

1.1. Mô tả chung

- Họ vi điều khiển 8051 có các đặc trưng được tóm tắt như sau:
  + Bộ nhớ có thể lập trình lại.
  + 4 KB ROM.
  + 128 byte RAM.
  + 4 cổng vào/ra (I/O port) 8-bit.
  + 2 bộ định thời 16-bit.
  + Cổng giao tiếp nối tiếp.
  + Không gian nhớ chương trình (mã) ngoài 64 Kb.
  + Không gian dữ liệu ngoài 64 Kb.
  + Bộ xử lý bit (thao tác trên các bit riêng rẽ).
  + 210 vị trí nhớ được định địa chỉ, mỗi vị trí 1 bit.
  + Nhân/chia trong 4 μs.

1.2. Sơ đồ khối họ vi điều khiển 8051

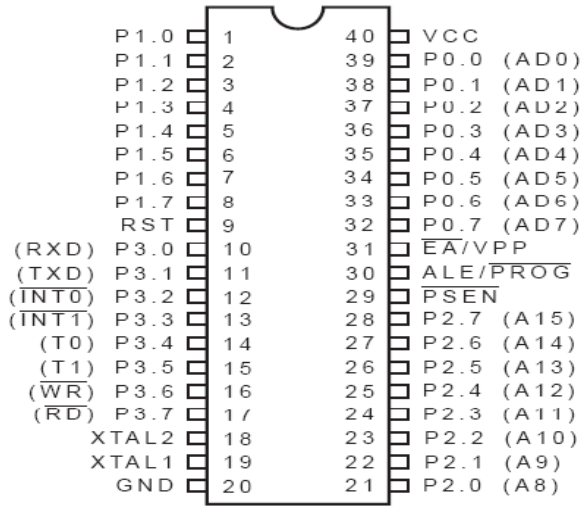
Hình sau đây cho thấy sơ đồ khối của vi điều khiển 8051



Hình 1. Sơ đồ khối họ vi điều khiển 8051

1.3. Sơ đồ chân và chức năng

1.3.1. Sơ đồ chân



Hình 2: Bố trí chân của họ 8051.

1.3.2. Mô tả chân

- Cổng P0: Bình thường đây là cổng ra. Để có thể vừa làm đầu ra, vừa làm đầu vào thì mỗi chân của P0 phải được nối tới một điện trở treo 10 kΩ bên ngoài. Sở dĩ như vậy là vì cổng P0 có dạng cực máng hở, đây là điểm khác với các cổng P1, P2 và P3. Khi nối 8051 tới bộ nhớ ngoài, P0 trở thành bus địa chỉ và bus dữ liệu dồn kênh để tiết kiệm số chân [byte thấp của bus địa chỉ nếu là địa chỉ].

- Cổng P1: P1 chỉ có một công dụng là vào/ra.

- Cổng P2: P2 có 2 công dụng, hoặc làm nhiệm vụ vào/ra hoặc là byte địa chỉ cao của bus địa chỉ 16-bit cho các thiết kế có bộ nhớ chương trình ngoài hoặc các thiết kế có nhiều hơn 256 byte bộ nhớ dữ liệu ngoài.

- Cổng P3: P3 có 2 công dụng. Khi không hoạt động vào/ra, các chân của P3 có nhiều chức năng riêng (mỗi chân có chức năng riêng liên quan đến các đặc trưng cụ thể của 8051).

Bit	Tên	Địa chỉ bit	Chức năng
P3.0	RxD	B0H	Nhận dữ liệu của cổng nối tiếp
P3.1	TxD	B1H	Phát dữ liệu của cổng nối tiếp
P3.2	$\overline{\text{INT0}}$	B2H	Ngắt ngoài 0
P3.3	$\overline{\text{INT1}}$	B3H	Ngắt ngoài 1
P3.4	T0	B4H	Chân vào của bộ định thời/đếm 0
P3.5	T1	B5H	Chân vào của bộ định thời/đếm 1
P3.6	$\overline{\text{WR}}$	B6H	Điều khiển ghi bộ nhớ dữ liệu ngoài
P3.7	$\overline{\text{RD}}$	B7H	Điều khiển đọc bộ nhớ dữ liệu ngoài

Bảng 1: Chức năng các chân của cổng P3.

- Chân cho phép bộ nhớ chương trình $\overline{\text{PSEN}}$: 8051 cung cấp cho ta 4 tín hiệu điều khiển bus. Tín hiệu cho phép bộ nhớ chương trình $\overline{\text{PSEN}}$ (Program Store Enable) là tín hiệu xuất. Đây là tín hiệu cho phép ta truy xuất bộ nhớ chương trình ngoài. Chân này thường nối với chân cho phép xuất $\overline{\text{OE}}$ (Output Enable) của EPROM (hoặc ROM) để cho phép đọc các byte lệnh.

- Chân cho phép chốt địa chỉ ALE: Chân xuất tín hiệu cho phép chốt địa chỉ ALE (Address Latch Enable) để phân kênh (demultiplexing) bus dữ liệu và bus địa chỉ.

- Chân truy xuất ngoài $\overline{\text{EA}}$ (External Access): Chân vào này có thể được nối với 5V (logic 1) hoặc với GND (logic 0).

+ Nếu chân này nối lên 5V, 8051 thực thi chương trình trong ROM nội (chương trình nhỏ hơn 4K/8K).

+ Nếu chân này nối với GND (và chân $\overline{\text{PSEN}}$ cũng ở logic 0), chương trình cần thực thi chứa ở bộ nhớ ngoài.

- Chân RESET (RST): Khởi động lại. Đây là chân vào, mức tích cực cao, bình thường ở mức thấp. Khi có xung cao đặt tới chân này thì bộ vi điều khiển sẽ kết thúc mọi hoạt động hiện tại và tiến hành khởi động lại. Khi reset, mọi giá trị trên các thanh ghi sẽ bị xoá. Lưu ý, để reset có hiệu quả, chân RST cần duy trì trạng thái tích cực mức cao tối thiểu 2 chu kỳ máy.

- Các chân XTAL1 và XTAL2: Mạch dao động bên trong chip 8051 được ghép nối với thạch anh bên ngoài ở hai chân XTAL1 và XTAL2.

1.4. Tổ chức bộ nhớ

Lệnh	Tên	Địa chỉ
ACC*	Thanh ghi tích luỹ (thanh ghi tổng) A	0E0H
B*	Thanh ghi B	0F0H
PSW*	Từ trạng thái chương trình	0D0H
SP	Con trỏ ngăn xếp	81H
DPTR	Con trỏ dữ liệu hai byte	
DPL	Byte thấp của DPTR	82H
DPH	Byte cao của DPTR	83H
P0*	Cổng P0	80H
P1*	Cổng P1	90H
P2*	Cổng P2	0A0H
P3*	Cổng P3	0B0H
IP*	Điều khiển ưu tiên ngắt	0B8H
IE*	Điều khiển cho phép ngắt	A08H
TMOD	Điều khiển chế độ bộ đếm/định thời	89H
TCON*	Điều khiển bộ đếm/định thời	88H
T2CON*	Điều khiển bộ đếm/định thời 2	0C8H
T2MOD	Điều khiển chế độ bộ đếm/định thời 2	0C9H
TH0	Byte cao của bộ đếm/định thời 0	8CH
TL0	Byte thấp của bộ đếm/định thời 0	8AH
TH1	Byte cao của bộ đếm/định thời 1	8DH
TL1	Byte thấp của bộ đếm/định thời 1	8BH
TH2	Byte cao của bộ đếm/định thời 2	0CDH
TL2	Byte thấp của bộ đếm/định thời 2	0CCH
RCAP2H	Byte cao của thanh ghi bộ đếm/định thời 2	0CBH
RCAP2L	Byte thấp của thanh ghi bộ đếm/định thời 2	0CAH
SCON*	Điều khiển nối tiếp	98H
SBUF	Bộ đệm dữ liệu nối tiếp	99H
PCON	Điều khiển công suất	87H

*Bảng 2: Các thanh ghi chức năng đặc biệt của họ vi điều khiển 8051 (*các thanh ghi có thể định địa chỉ theo bit).*

Họ vi điều khiển 8051 có không gian bộ nhớ riêng cho chương trình và dữ liệu. Cả 2 bộ nhớ chương trình và dữ liệu đều đặt bên trong chip, tuy nhiên ta có thể mở rộng bộ nhớ chương trình và bộ nhớ dữ liệu bằng cách sử dụng các chip

nhớ bên ngoài.

- Bộ nhớ nội trong chip bao gồm ROM và RAM. RAM trên chip bao gồm vùng RAM đa chức năng (nhiều công dụng), vùng RAM với từng bit được định địa chỉ (gọi tắt là vùng RAM định địa chỉ bit), các dãy (bank) thanh ghi và các thanh ghi chức năng đặc biệt SFR (Special Function Register). Hai đặc tính đáng lưu ý:

+ Các thanh ghi và các cổng vào/ra được định địa chỉ theo kiểu ánh xạ bộ nhớ (memory mapped) và được truy xuất như một vị trí trong bộ nhớ.

+ Vùng stack thường trú trong RAM trên chip (RAM nội) thay vì ở trong RAM ngoài như đối với các bộ vi xử lý.

- Vùng RAM đa mục đích: Bất kỳ một vị trí nhớ nào trong vùng RAM đa mục đích đều có thể được truy xuất tự do bằng cách sử dụng các kiểu định địa chỉ trực tiếp hoặc gián tiếp.

- Vùng RAM định địa chỉ bit: Ý tưởng truy xuất các bit riêng rẽ thông qua phần mềm là một đặc trưng mạnh của hầu hết các bộ vi điều khiển. Các bit có thể được set, xoá, AND, OR… bằng một lệnh. Hầu hết các bộ vi xử lý yêu cầu một chuỗi lệnh đọc-sửa-ghi để nhận được cùng một kết quả. Ngoài ra 8051 còn có các cổng vào/ra có thể định địa chỉ từng bit, làm đơn giản việc giao tiếp bằng phần mềm với các thiết bị vào/ra đơn bit.

- Các dãy thanh ghi: Dãy các thanh ghi đang được sử dụng được gọi là dãy thanh ghi tích cực. Dãy thanh ghi tích cực có thể được thay đổi bằng cách thay đổi các bit chọn trong từ trạng thái PSW.

- Các thanh ghi chức năng đặc biệt được trình bày trong bảng 2.

1.5. Tập lệnh

- Đặc điểm nổi bật của 8051 là có các lệnh tác động và thực hiện trên từng bit được đánh địa chỉ. 8051 có thể xoá, lập, lấy phần bù, kiểm tra, di chuyển, và thực hiện các phép toán logic trên từng bit.

- Tập lệnh của 8051 được thiết kế nhằm tối ưu cả độ dài của mã lệnh và tốc độ thực hiện. Hầu hết các lệnh có độ dài 1 byte, một số 2 byte và 3 byte là rất ít. Các lệnh thực hiện trong 1 hoặc 2 chu kỳ máy, chỉ có nhân và chia là thực hiện trong 4 chu kỳ máy.

- Có 8 kiểu định địa chỉ:
  + Thanh ghi (register).
  + Trực tiếp (direct).
  + Gián tiếp (indirect).
  + Tức thời (immediate).
  + Tương đối (relative).
  + Tuyệt đối (absolute).
  + Dài (long).
  + Chỉ số (indexed).
- Tổng số lệnh là 111. 111 lệnh chia thành 5 nhóm :
  + Tính toán số học.
  + Tính toán logic (cho biến kiểu byte).
  + Chuyển dữ liệu.

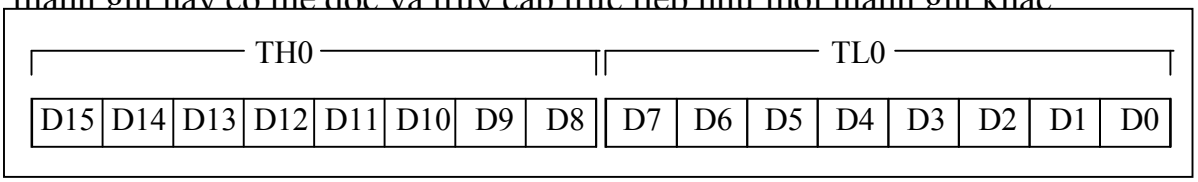
+ Thực hiện trên bit.

+ Lệnh rẽ nhánh chương trình và điều khiển.

1.6. Bộ đếm/bộ định thời

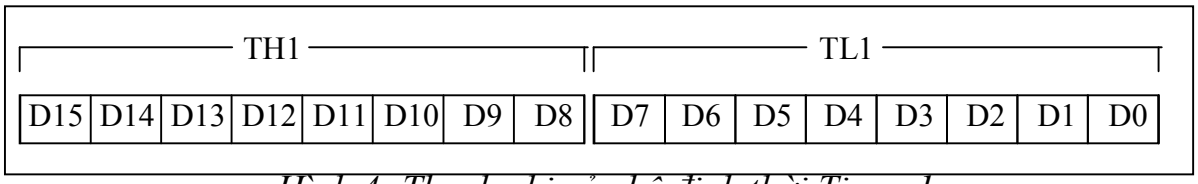
8051 có hai bộ định thời là Timer 0 và Timer 1. Cả hai bộ định thời đều có độ dài 16 bit. Do 8051 có cấu trúc 8 bit, nên mỗi bộ định thời được truy cập dưới dạng hai thanh ghi độc lập là byte thấp và byte cao.

- Thanh ghi của bộ Timer 0: Thanh ghi 16 bit của bộ Timer 0 được truy cập theo 2 byte là byte thấp và byte cao. Thanh ghi byte thấp được gọi là TL0 (Timer 0 Low byte) và thanh ghi byte cao là TH0 (Timer 0 High byte). Các thanh ghi này có thể đọc và truy cập trực tiếp như mọi thanh ghi khác.



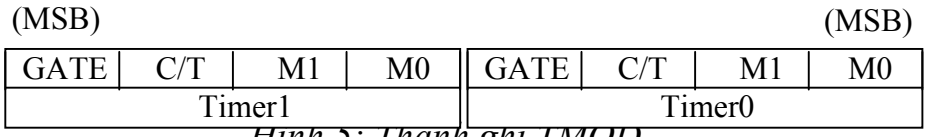
Hình 3: Thanh ghi của bộ định thời Timer 0.

- Thanh ghi của bộ Timer 1: Bộ định thời Timer 1 cũng dài 16 bit và thanh ghi 16 bit cũng được chia ra thành hai byte là TL1 và TH1. Các thanh ghi này được truy cập và được đọc giống như các thanh ghi của bộ Timer 0.



Hình 4: Thanh ghi của bộ định thời Timer 1.

- Thanh ghi chế độ của bộ định thời TMOD: Cả hai bộ định thời đều dùng chung một thanh ghi được gọi là TMOD để thiết lập các chế độ làm việc khác nhau của bộ định thời. Thanh ghi TMOD là thanh ghi 8 bit gồm có 4 bit thấp dành cho bộ Timer 0 và 4 bit cao dành cho Timer 1. Trong đó hai bit thấp của chúng dùng để thiết lập chế độ của bộ định thời, còn hai bit cao dùng để xác định phép toán.



Hình 5: Thanh ghi TMOD.

+ Các bit M0, M1: Là các bit chế độ dùng để chọn chế độ 0, 1 và 2 của các bộ Timer 0 và Timer 1.

M1	M0	Chế độ	Chế độ hoạt động
0	0	0	Chế độ bộ định thời 13 bit. Bộ định thời/bộ đếm 8 bit, định tỷ lệ trước 5 bit.
0	1	1	Chế độ bộ định thời 16 bit, không định tỷ lệ trước.
1	0	2	Chế độ 8 bit tự nạp lại. THx lưu giá trị sẽ tự nạp vào TLx mỗi khi tràn.

1	1	3	Chế độ bộ định thời chia tách

Bảng 3: Các chế độ hoạt động của bộ định thời.

1.7. Ngắt

- 8051 có 5 ngắt dành cho người dùng:
  + Hai ngắt dành cho bộ định thời Timer 0 và Timer 1.
  + Hai ngắt phần cứng dành cho các thiết bị bên ngoài nối tới chân INT0 và INT1 của cổng P3.
  + Truyền thông nối tiếp có một ngắt dành cho cả thu lẫn phát.

Ngắt	Địa chỉ ROM	Chân
Bật lại nguồn (RESET)	0000	9
Ngắt phần cứng ngoài (INT0)	0003	12 (P3.2)
Ngắt bộ Timer0 (TF0)	000B	
Ngắt phần cứng ngoài 1 (INT1)	0013	13 (P3.3)
Ngắt bộ Timer1 (TF1)	001B	
Ngắt COM nối tiếp (RI và TI)	0023	

Bảng 4: Bảng vector ngắt của 8051.

- Cho phép và cấm ngắt: Các ngắt phải được cho phép bằng phần mềm để bộ vi điều khiển có thể đáp ứng được. Thanh ghi cho phép ngắt IE (Interrupt Enable) chịu trách nhiệm về việc cho phép (không che) và cấm (che) các ngắt.

D7							D0
EA	--	ET2	ES	ET1	EX1	ET0	EX0

Hình 6: Thanh ghi cho phép ngắt IE.

  + EA: Nếu EA = 0 thì không ngắt nào được báo nhận.
    Nếu EA = 1 thì từng nguồn ngắt sẽ được mở hoặc cấm bằng cách bật hoặc xoá bit cho phép tương ứng.
  + ET2: Cho phép/cấm ngắt tràn hoặc thu của Timer 2.
  + ES: Cho phép/cấm ngắt cổng nối tiếp.
  + ET1: Cho phép/cấm ngắt tràn của Timer 1.
  + EX1: Cho phép/cấm ngắt ngoài 1.
  + ET0: Cho phép/cấm ngắt tràn của Timer 0.
  + EX0: Cho phép/cấm ngắt ngoài 0.

- Ưu tiên ngắt: Khi 8051 được cấp nguồn thì các mức ưu tiên ngắt được gán theo bảng sau :

Mức ưu tiên từ cao xuống thấp
Ngắt ngoài 0 INT0
Ngắt bộ định thời 0 TF0
Ngắt ngoài 1 INT1
Ngắt bộ định thời 1

TF1
Ngắt truyền thông nối tiếp (RI+TI)

Bảng 5: Mức ưu tiên ngắt của 8051 khi Reset.

- Ta có thể thay đổi trình tự ưu tiên ngắt cho ở bảng trên bằng cách gán mức ưu tiên ngắt cao hơn cho bất kỳ ngắt nào nhờ thanh ghi mức ưu tiên ngắt IP (Interrupt Priority).

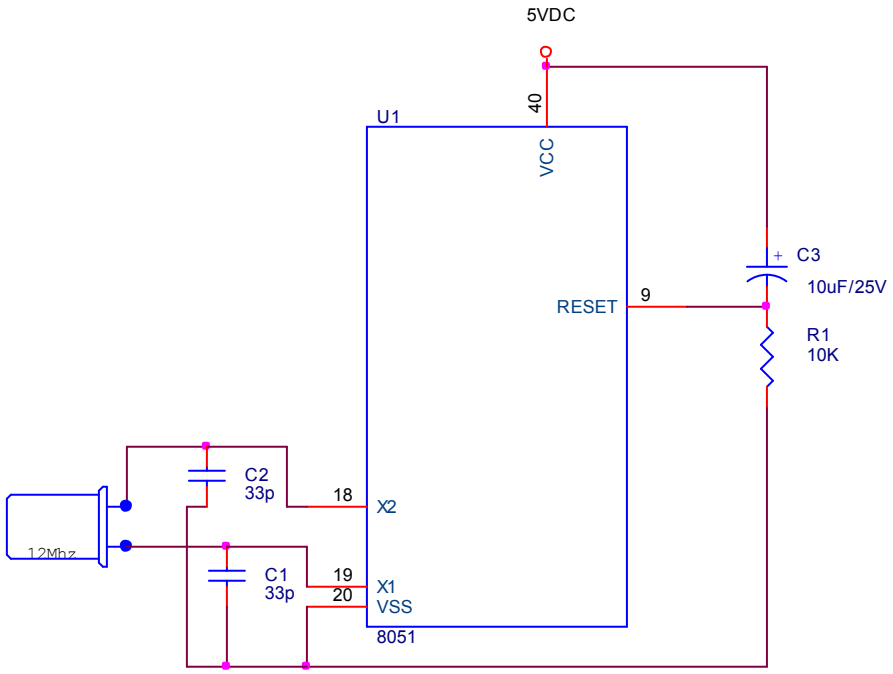
D7							D0
--	--	PT2	PS	PT1	PX1	PT0	PX0

Bit ưu tiên =1 là mức ưu tiên cao, bit ưu tiên = 0 là mức ưu tiên thấp.

+ Bit D7 và D6 hay IP.7 và IP.6 - chưa dùng.
+ Bit D5 hay IP.5 là bit ưu tiên ngắt Timer2 (dùng cho 8052).
+ Bit D4 hay IP.4 là bit ưu tiên ngắt cổng nối tiếp.
+ Bit D3 hay IP.3 là bit ưu tiên ngắt Timer1.
+ Bit D2 hay IP.2 là mức ưu tiên ngắt ngoài 1.
+ Bit D1 hay IP.1 là mức ưu tiên ngắt Timer 0.
+ Bit D0 hay IP.0 là mức ưu tiên ngắt ngoài 0.

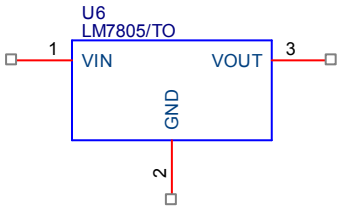
- Trường hợp có nhiều bit ưu tiên ngắt trong thanh ghi IP cùng được đặt lên cao: Khi đó, nếu các ngắt này cùng có mức ưu tiên cao hơn các ngắt khác thì chúng sẽ được phục vụ theo trình tự cho trong bảng ưu tiên ngắt mặc định.

Mạch vi điều khiển đơn giản nhất:

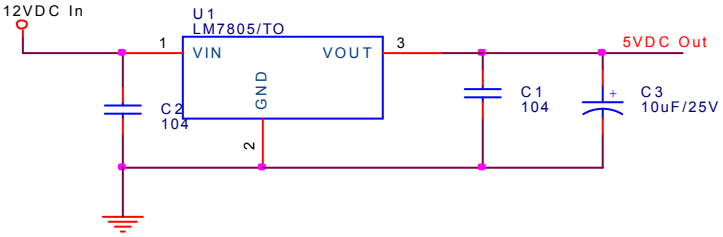


Mạch này chưa có khối nguồn để tạo nguồn 5V các bạn dùng con IC sau:

Sơ đồ chân:



Giới thiệu IC ổn áp 7805 : Đầu vào > 7V đầu ra 5V 500mA. Mạch ổn áp: cần cho VĐK vì nếu nguồn cho VĐK không ổn định thì sẽ treo, không chạy đúng, hoặc reset liên tục, thậm chí là chết chip



Mạch nguồn

Bài 2 : Ngôn ngữ lập trình và phần mềm biên dịch, mô phỏng Keil C.

Phần 1 – Ngôn ngữ C với lập trình vđk

2.1.1. Cấu trúc một chương trình

```
//Đính kèm các file
#include <file.h>
#include <file.c>
//Khai báo biến toàn cục
unsigned char x,y;
int z;
long n=0;
//Khai báo và định nghĩa các hàm
void Hàm1(void)
{
  …//Các câu lệnh
}

void Hàm2(unsigned char x)
{
  …//Các câu lệnh
}

//Hàm chính bắt buộc chương trình nào cũng phảI có
void main(void)
{
  …//Các câu lệnh
}
```

Các câu lệnh trong hàm chính có thể có lời gọi các hàm đã khai báo ở trên hoặc không
Khi có lời gọi hàm nào thì chương trình nhảy đến hàm đó thực hiện hàm đó xong con trỏ lại quay về chương trình chính(hàm main) thực hiện tiếp các hàm hoặc câu lệnh.

Các câu lệnh trong C kết thúc bằng dấu ";"
Các lời giải thích được đặt trong dấu: Mở đầu bằng "/*" kết thúc bằng "*/"
Nếu lời giải thích trên 1 dòng thì có thể dùng dấu: "//"
Khi lập trình nên giải thích các câu lệnh khối lệnh làm gì để về sau khi chương trình lớn dễ sửa lỗi.

2.1.2.Các loại biến trong C:

Dạng biến	Số Bit	Số Byte	Miền giá trị
char	**8**	**1**	**-128 đến +127**
unsigned char	**8**	**1**	**0 đến 255**
short	**16**	**2**	**-32,768 đến +32,767**
unsigned short	**16**	**2**	**0 đến 65,535**
int	**16**	**2**	**-32,768 đến +32,767**
unsigned int	**16**	**2**	**0 đến 65,535**
long	**32**	**4**	**-2,147,483,648 đến+2,147,483,647**
unsigned long	**32**	**4**	**0 đến 4,294,697,295**

Khai báo biến:
Cấu trúc: Kiểubiến Tênbiến;
VD: unsigned char x;
Khi khai báo biến có thể gán luôn cho biến giá trị ban đầu.
VD :
Thay vì: unsigned char x;
x=0;
Ta chỉ cần : unsigned char x=0;
Có thể khai báo nhiều biến cùng một kiểu một lúc
VD: unsigned int x,y,z;

Ngoài ra để dùng cho vi điều khiển trình dịch chuyên dụng còn hỗ trợ các loại biến sau:

Dạng biến	Số Bit	Số Byte	Miền giá trị

bit	**1**	**0**	**0 ; 1**
sbit	**1**	**0**	**0 ; 1**
sfr	**8**	**1**	**0 đến 255**
sf16	**16**	**2**	**0 đến 65,535**

Trong đó bit có thể dùng như các biến của C++ nhưng các loại biến còn lại thì liên quan đến các thanh ghi hoặc địa chỉ cổng của 8051.
Có nghĩa là khi khai báo biến kiểu bit thì không cần định địa chỉ trong RAM cho các biến đó, còn khi khai báo biến kiểu sbit, sfr, sf16 thì phải định rõ địa chỉ trong RAM vì nó là các dạng biến đặc biệt gọi là special function registers (= các thanh ghi co chức năng đặc biệt viết tắt là SFR)

VD:

```
Bit  Kiemtra;
Sfr P10=0x90;
```

Các SFR không cần phải học thuộc chỉ cần biết, và chúng được khai báo trong thư viện
AT89X51.H và AT89X52.H

2.1.3. Hàm trong C:

Hàm trong C có cấu trúc như sau:
Có hai loại hàm:
Hàm trả lại giá trị:

```
Cấu trúc:       Kiểugiátrịhàmtrảlại Tênhàm (Biếntruyềnvàohàm)
                   {
                  // Các câu lệnh xử lí ở đây
                   }

Ví dụ :         unsigned char Cong(unsigned char x, unsigned char y)
                   {
                   // Các câu lệnh xử lí ở đây
                    }
```

Hàm không trả lại giá trị:

```
Cấu trúc:       void Tênhàm (Biêntruyềnvàohàm)
                 {
                   // Các câu lệnh xử lí ở đây
                    }
Ví dụ :         void Cong(unsigned char x, unsigned char y)
                  {
                   // Các câu lệnh xử lí ở đây
                    }
```

Hàm có thể có biến truyền vào hoặc không.
Ví dụ:
Hàm không có biến truyền vào:

```
unsigned char  Tênhàm(void)
```

```
{
    // Các câu lệnh xử lí ở đây
}
```

Hàm có biến truyền vào:

```
void  Tênhàm(unsigned char x)
{
    // Các câu lệnh xử lí ở đây
}
```

Số biến truyền vào tùy ý(miễn đủ bộ nhớ), ngăn cách bởi dấu ","
Ví dụ:

```
Void TênHàm(unsigned char x, unsigned char y, unsigned char z)
{
// Các câu lệnh xử lí ở đây
}
```

NgoàI ra riêng cho vi điều khiển phần phềm Keil C còn có một loại hàm đó là hàm ngắt:
Cấu trúc:

```
Void  Tênhàm(void) interrupt nguồnngắt using băngthanhghi
{
}
```

Hàm ngắt không được phép trả lại giá trị hay truyền tham biến vào hàm.
Tên hàm bất kì
Interrupt là từ khóa chỉ hàm ngắt
Nguồn ngắt từ 0 tới 5 theo bảng vector ngắt

Ngắt do	Cờ	Địa chỉ vector
Reset hệ thống	**RST**	**0000H**
Ngắt ngoài 0	**IE0**	**0003H**
Bộ định thời 0	**TF0**	**000BH**
Ngắt ngoài 1	**IE1**	**0013H**
Bộ định thời 1	**TF1**	**001BH**
Port nối tiếp	**RI hoặc TI**	**0023H**
Bộ định thời 2	**TF2 hoặc EXF2**	**002BH**

Không tính ngắt reset hệ thống bắt đầu đếm từ ngắt ngoài 0 nguồn ngắt là 0.
Băng thanh ghi trên Ram chọn từ 0 đến 3.

2.1.4. Các toán tử cơ bản :

Phép gán: =
VD: x=y; // x phảI là biến y có thể là biến hoặc giá trị nhưng phảI phù hợp kiểu
Phép cộng: +
Phép trừ: -
Phép nhân: *
Phép chia: /
Các toán tử logic:
Bằng : ==
And: &&
Or: ||

Not: !
Dịch trái: <<
Dịch phải: >>

2.1.5. Các cấu trúc lệnh rẽ nhánh, kiểm tra thường dùng:

Câu lệnh rẽ nhánh if:

Cấu trúc: if (Điềukiện) { // Các câu lệnh xử lí }

Giải thích: Nếu Điềukiện đúng thì xử lí các câu lệnh bên trong còn sai thì nhảy qua

Câu lệnh lựa chọn switch:

Cấu trúc:
```
switch(Biến)
 {
case giatri1:  { // Các câu lệnh   break; }
case giatri2:  { // Các câu lệnh   break; }
case giatri3:  { // Các câu lệnh   break; }
…
case giatrin:  { // Các câu lệnh   break; }
 }
```

Giải thích : Tùy vào Biến có giá trị 1 thì thực hiện các câu lệnh sau đó tương ứng rồi thoát khỏi cấu trúc nhờ câu lệnh break;

Biến có giá trị 2 thì thực hiện các câu lệnh sau đó tương ứng rồi thoát

….

Biến có giá trị n thì thực hiện các câu lệnh sau đó tương ứng rồi thoát

Câu lệnh vòng lặp xác định for:

Cấu trúc: for(n=m; n<l; n++) {// Các câu lệnh xử lí }

Giải thích:

Trong đó m,l là giá trị (m>l), còn n là biến

Thực hiện lặp các câu lệnh (l-m) lần

Câu lệnh vòng lặp không xác định while:

Cấu trúc:
```
While( Điều kiện)
{
 //Các câu lệnh
}
```

Giải thích:

Thực hiện lặp các câu lệnh khi điều kiện đúng, nếu câu lệnh sai thì thoát khỏi vòng lặp

2.1.6. Bộ tiền xử lí

#define : Dùng để định nghĩa. Ví dụ:

```
#define dung 1
#define sai 0
```

có nghĩa là dung có giá trị bằng 1. Trong chương trình có thể có đoạn code như sau:

```
 bit kiemtra
if (bit==dung)  { // Các câu lệnh}
if (bit==sai)  { // Các câu lệnh}
```

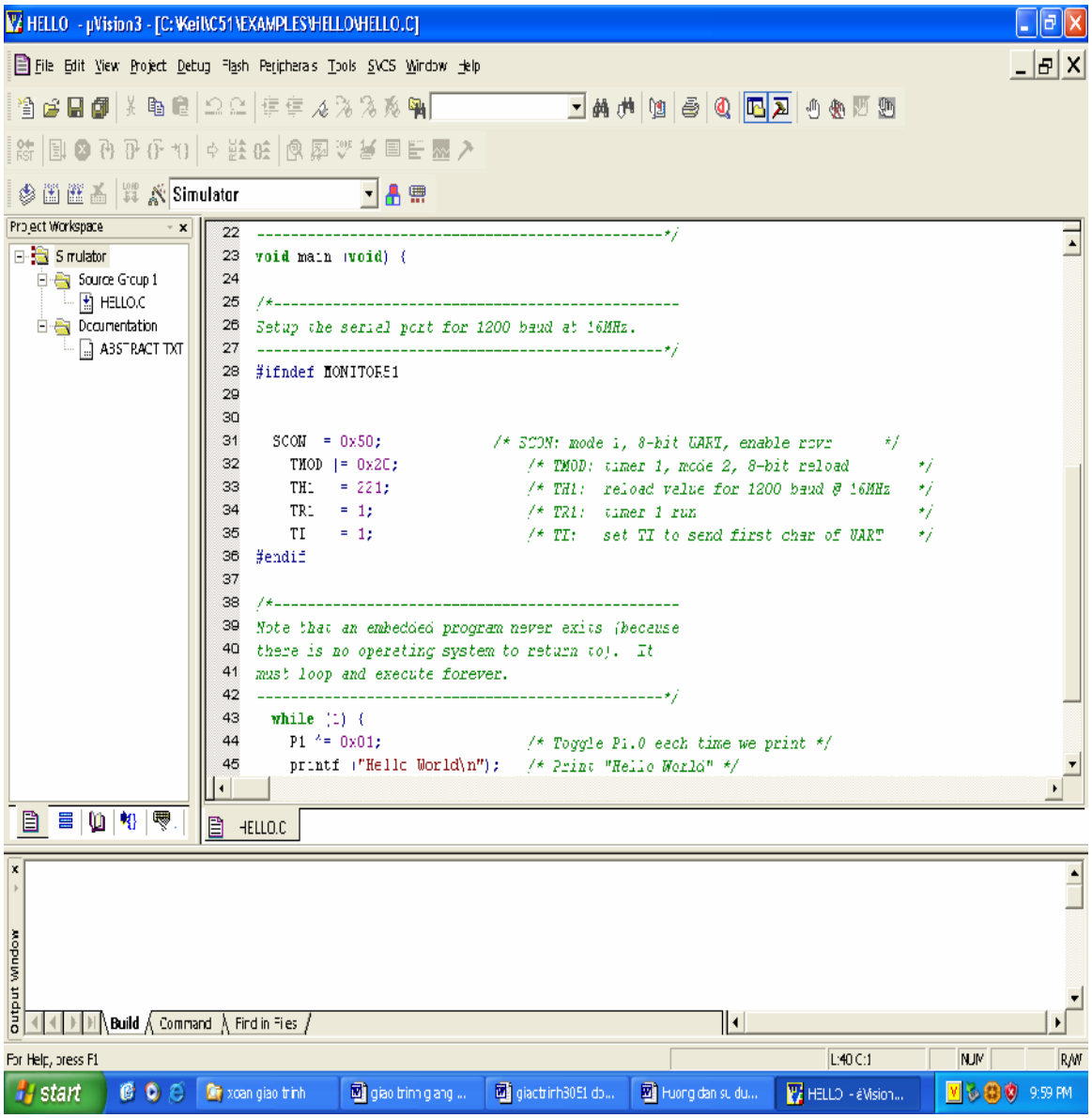
Việc này giúp lập trình dễ sửa lỗi hơn.

Một số web hay :

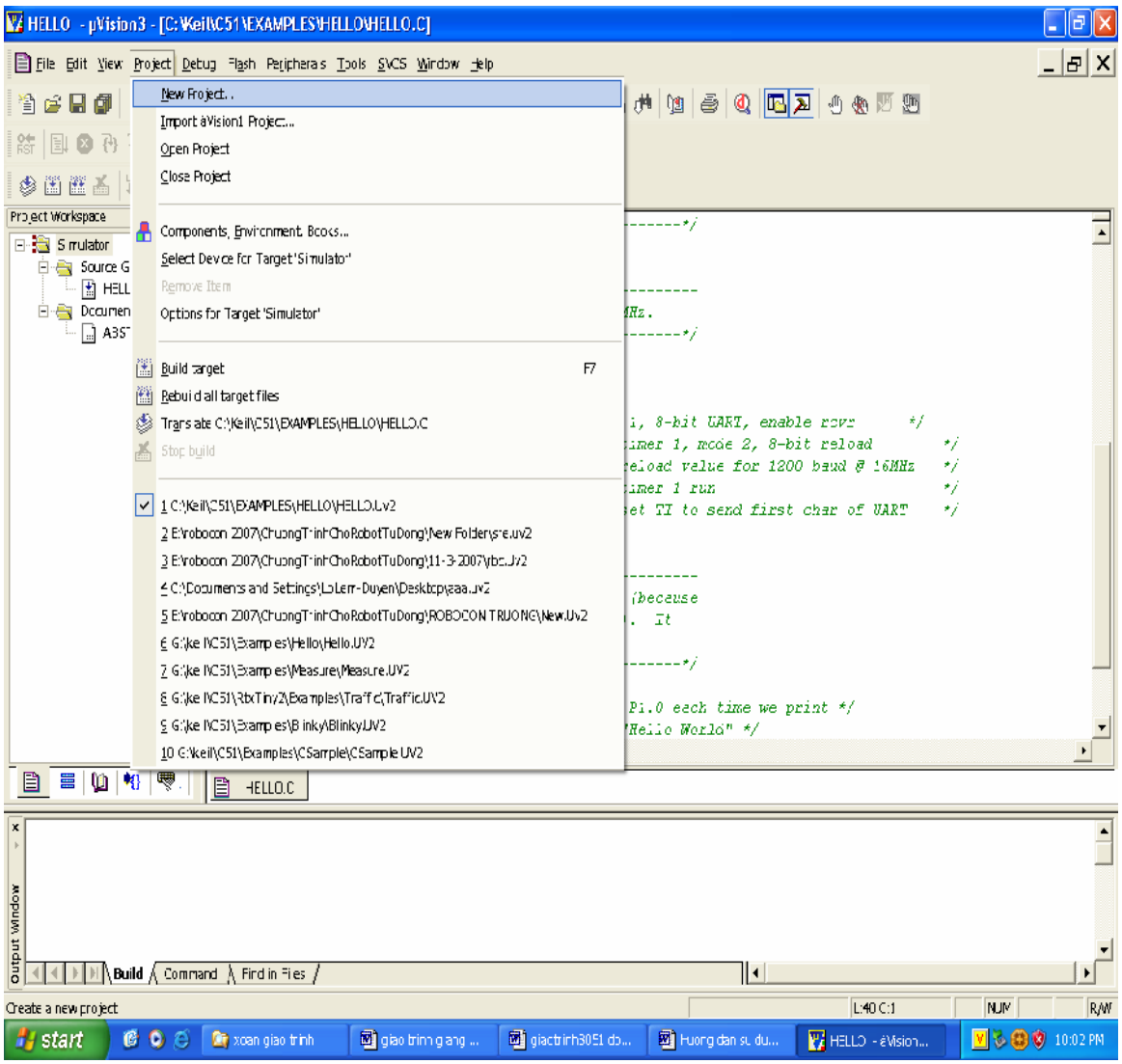
www.dientuvietnam.net
www.svbkol.org
www.diendandientu.com
www.microchip.com
www.kmitl.ac.th
www.ftdichip.com
www.atmel.com
www.keil.com
www.iguanalabs.com
www.ttvnol.com
www.8052.com

Phần 2: Trình biên dịch cách sử dụng Keil C uVision 3.0

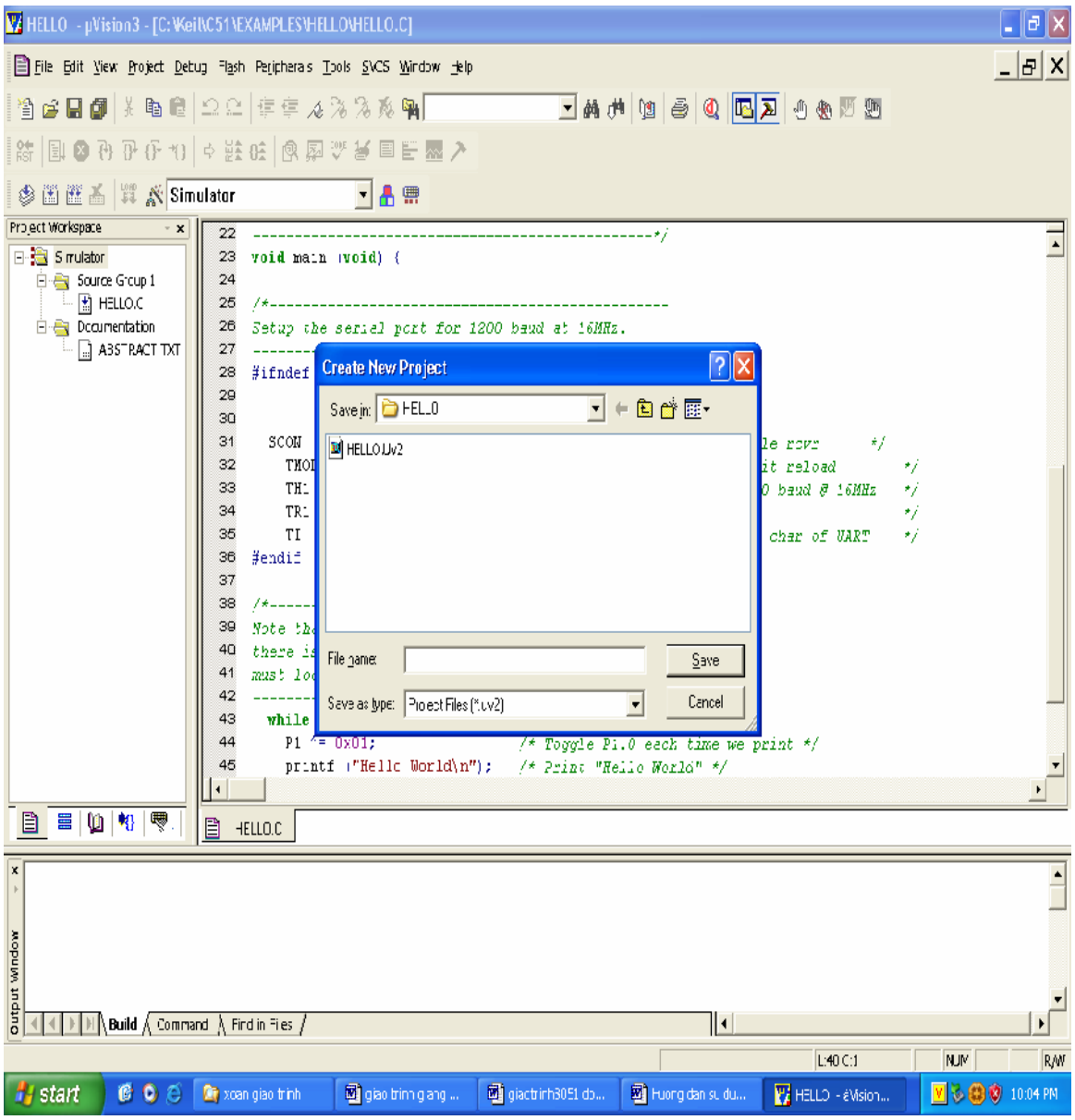
2.2.1.Khởi tạo cho Project:



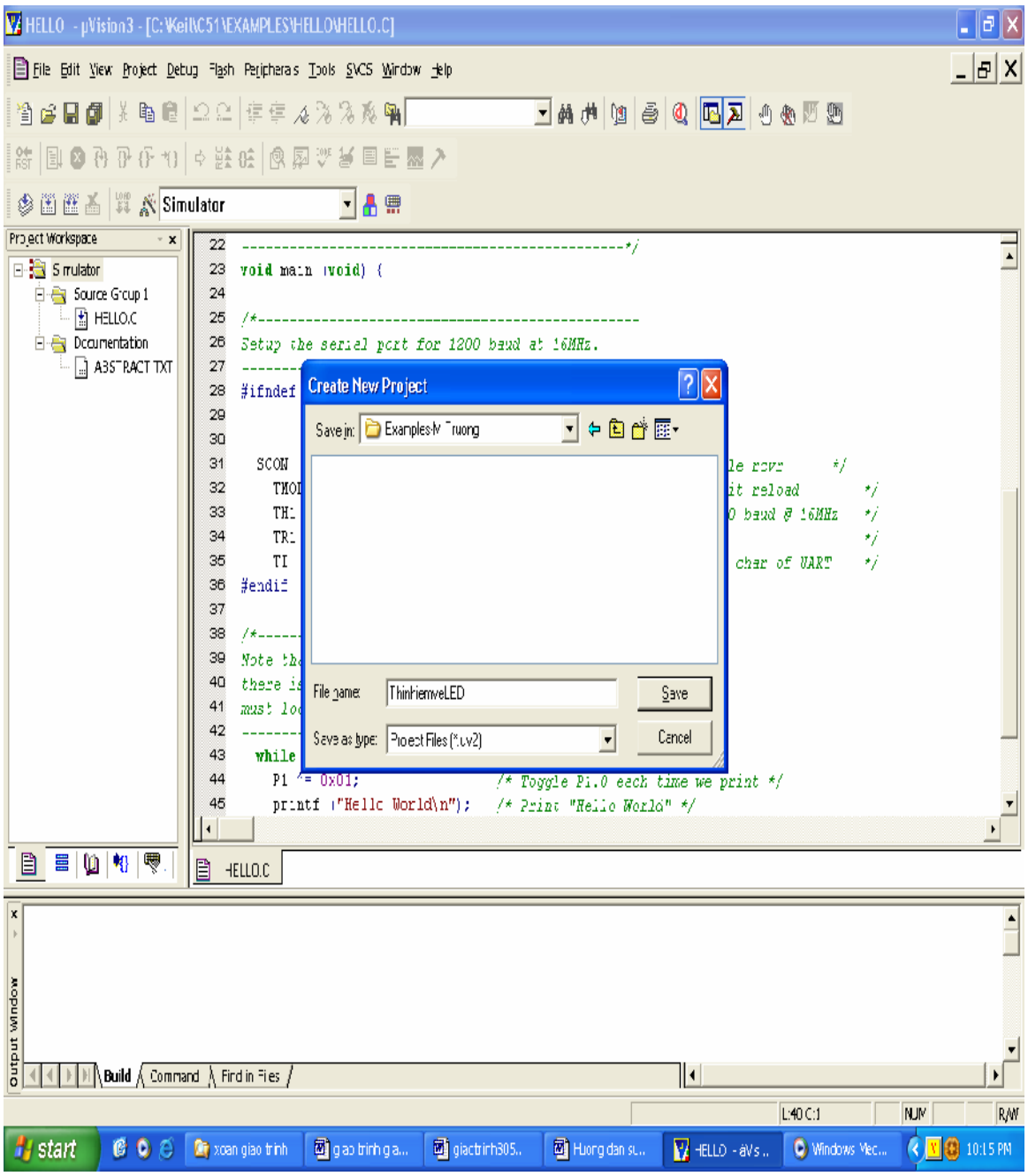
Để tạo 1 project mới chọn Project → New project như sau:



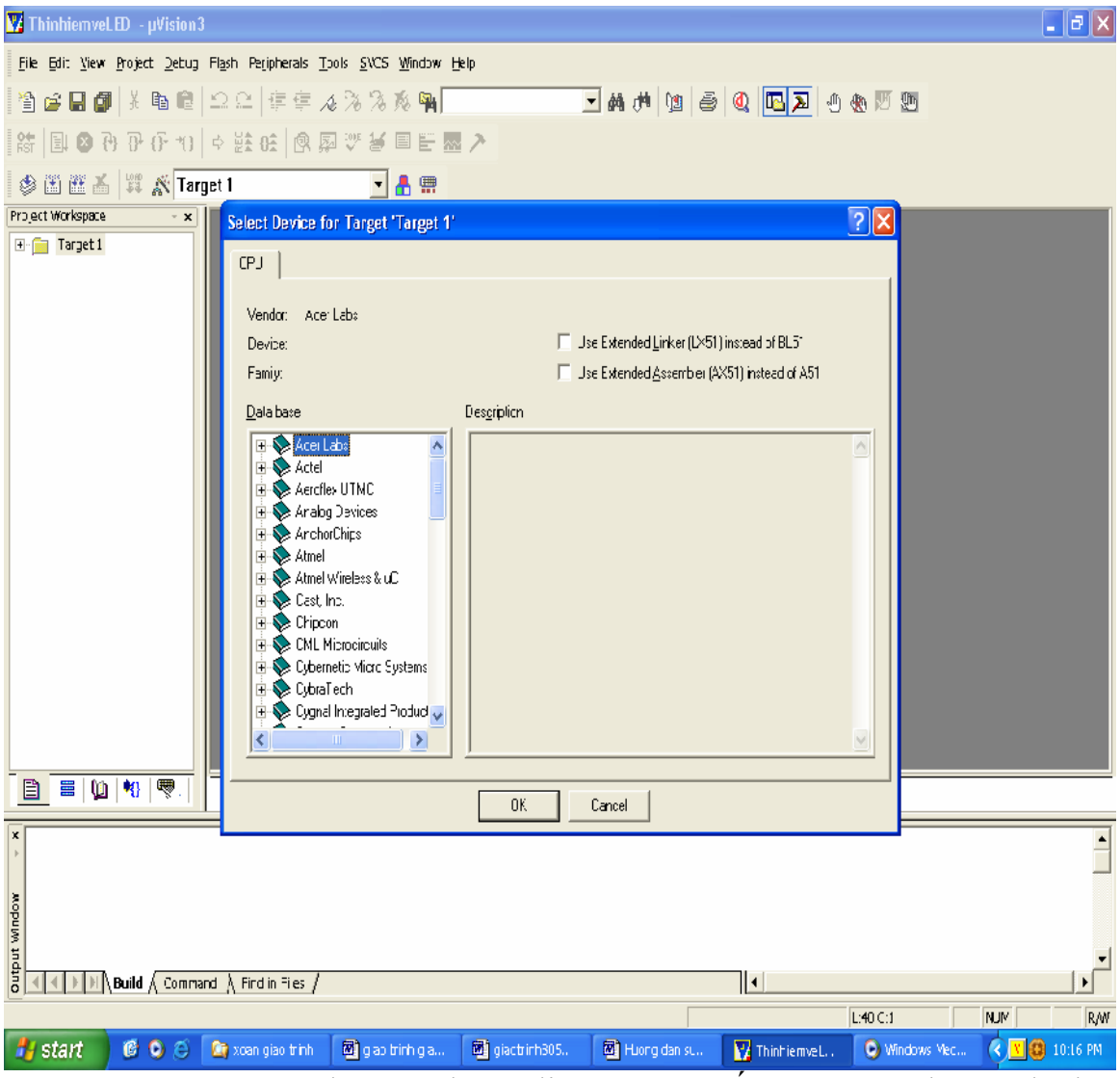
Được hình sau:



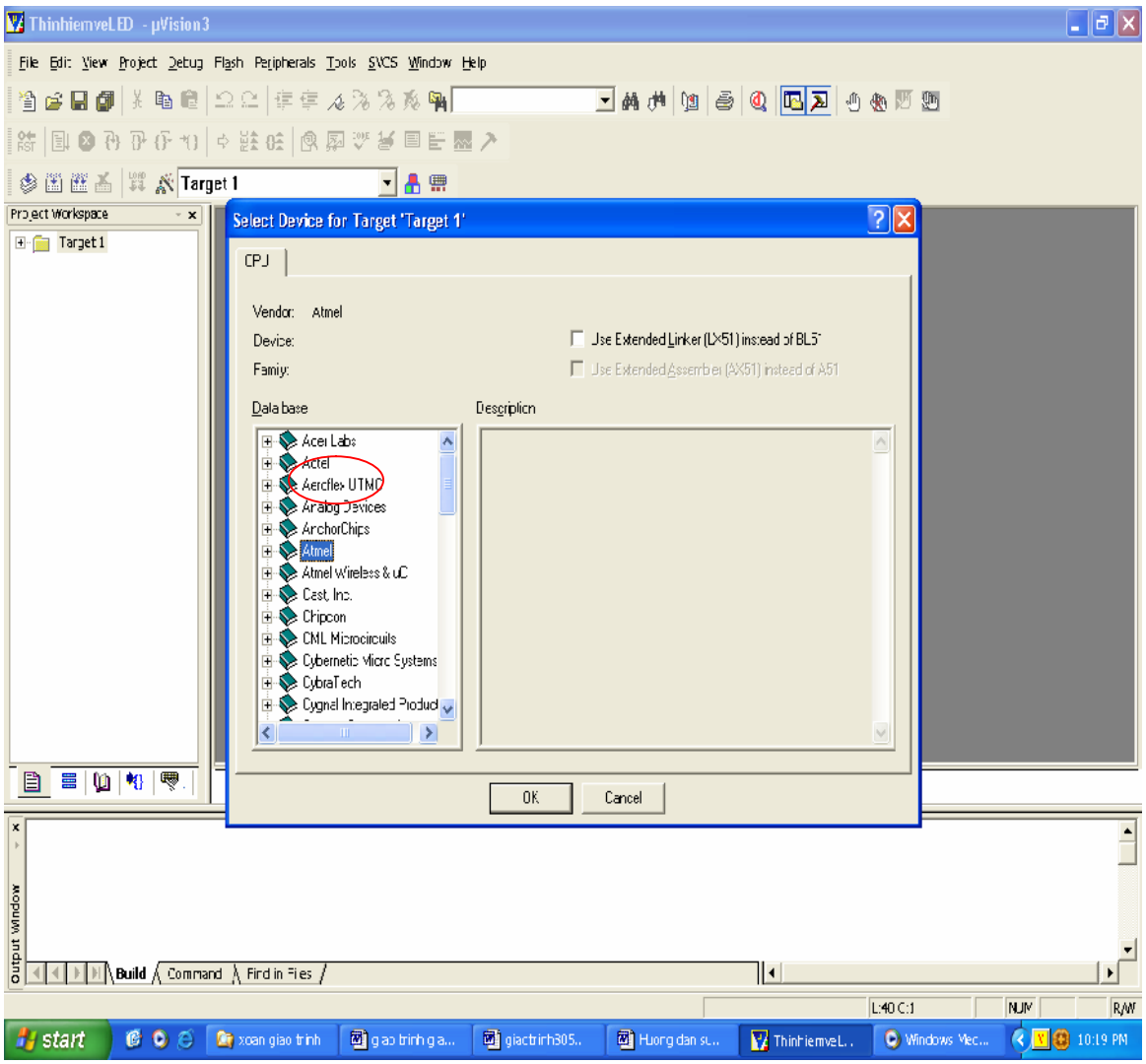
Đánh tên và chuyển đến thư mục bạn lưu project. Bạn nên tạo mỗi một thư mục cho 1 project. Rồi chọn Save.

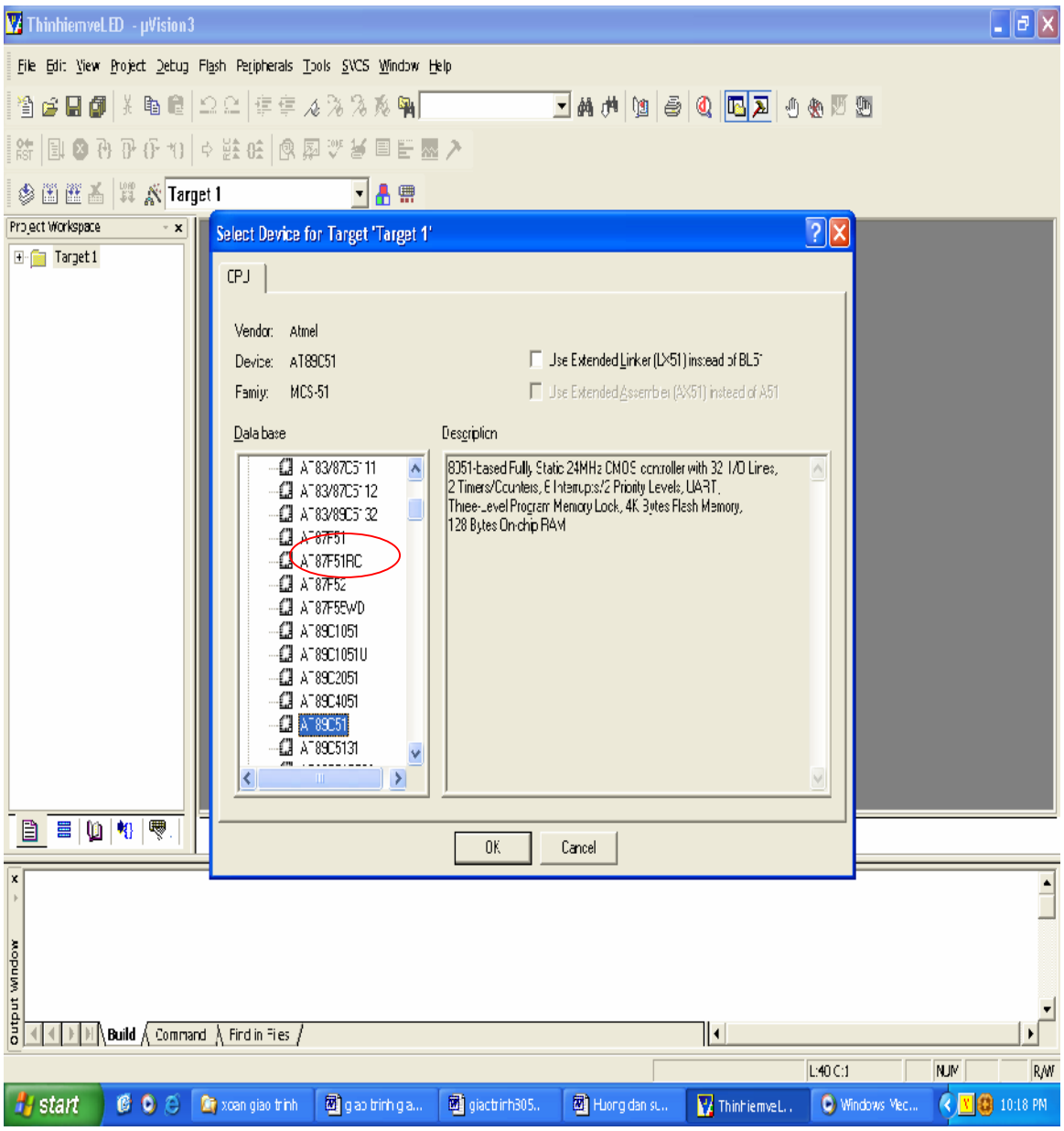


Được hình sau:

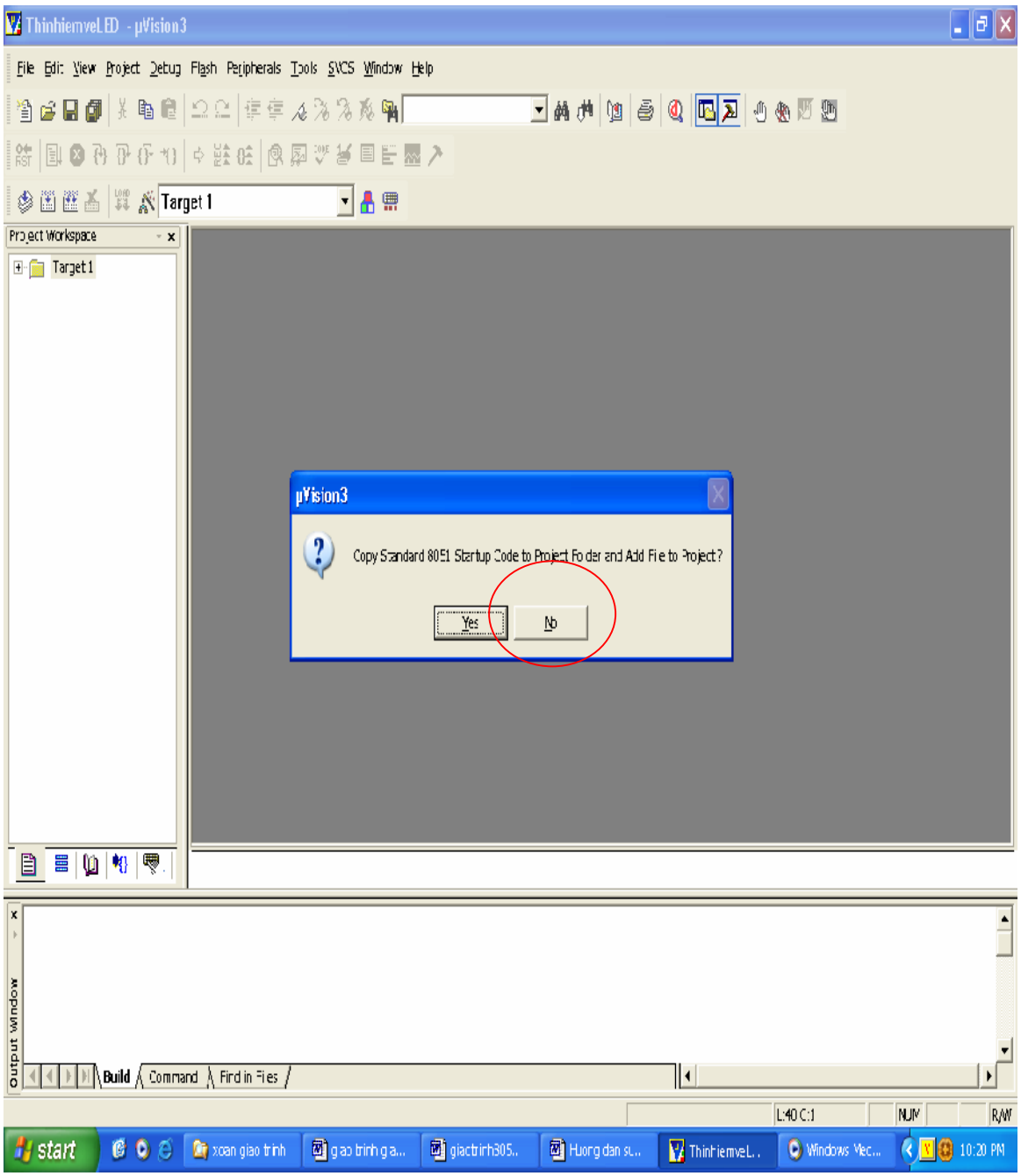


Trong này có 1 loạt các hãng điện tử sản xuất 8051. Bạn lập trình cho con nào thì chọn con đấy ,kích chuột vào các dấu + để mở rộng các con IC của các hãng. Ở đây ta lập trình cho AT89C51 của hãng Atmel nên ta chọn như sau:

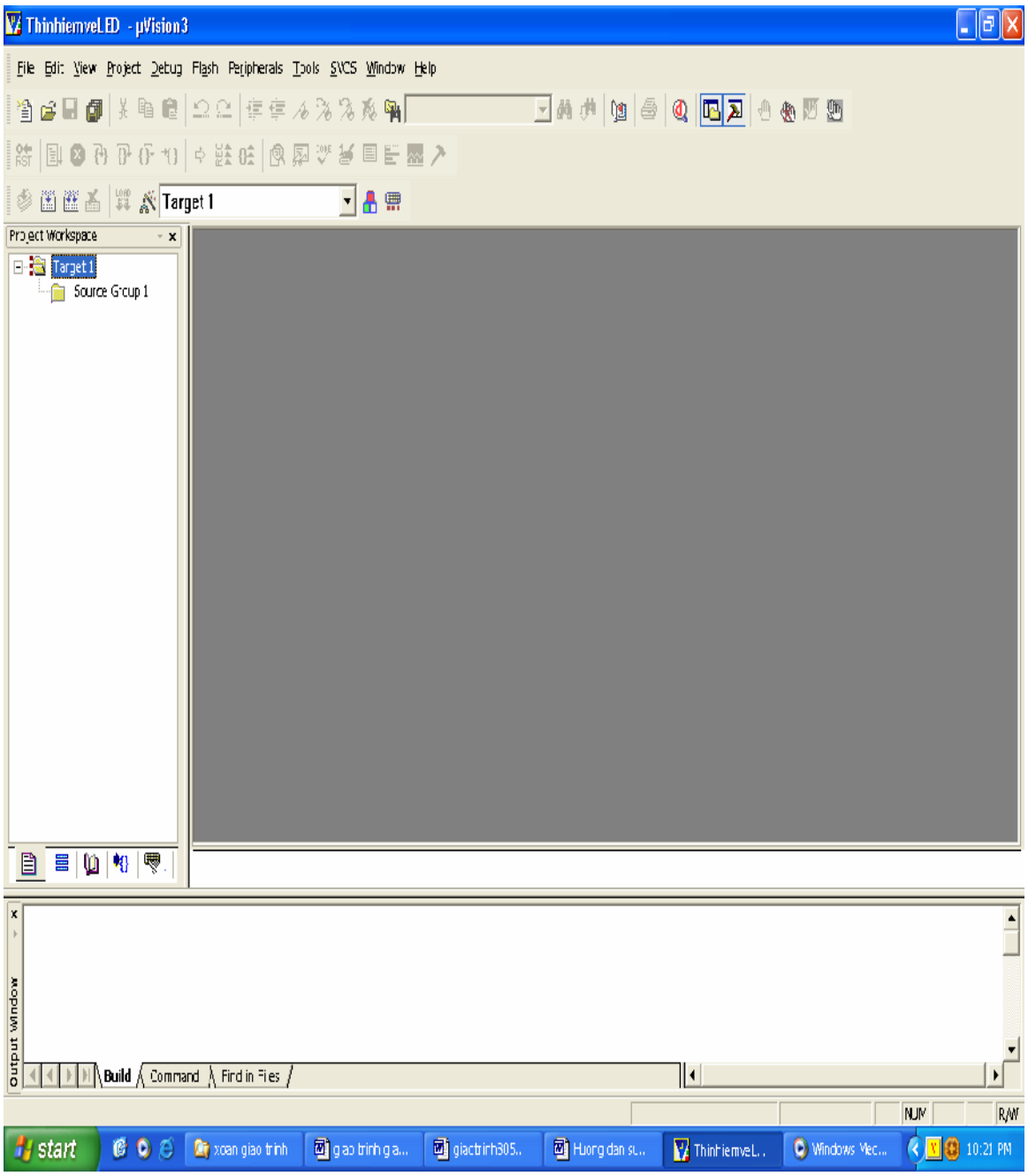

ThinhiemveLED - µVision3
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Target 1
Project Workspace
Target 1
Select Device for Target 'Target 1'
CPU
Vendor: Atmel
Device:
Family:
Use Extended Linker (LX51) instead of BL51
Use Extended Assembler (AX51) instead of A51
Data base
Description
Acer Labs
Actel
Aeroflex UTMC
Analog Devices
AnchorChips
Atmel
Atmel Wireless & uC
Cast, Inc.
Chipcon
CML Microcircuits
Cybernetic Micro Systems
CybraTech
OK
Cancel
Output Window
Build
Command
Find in Files
L:40 C:1
NUM
R/W
start
10:19 PM



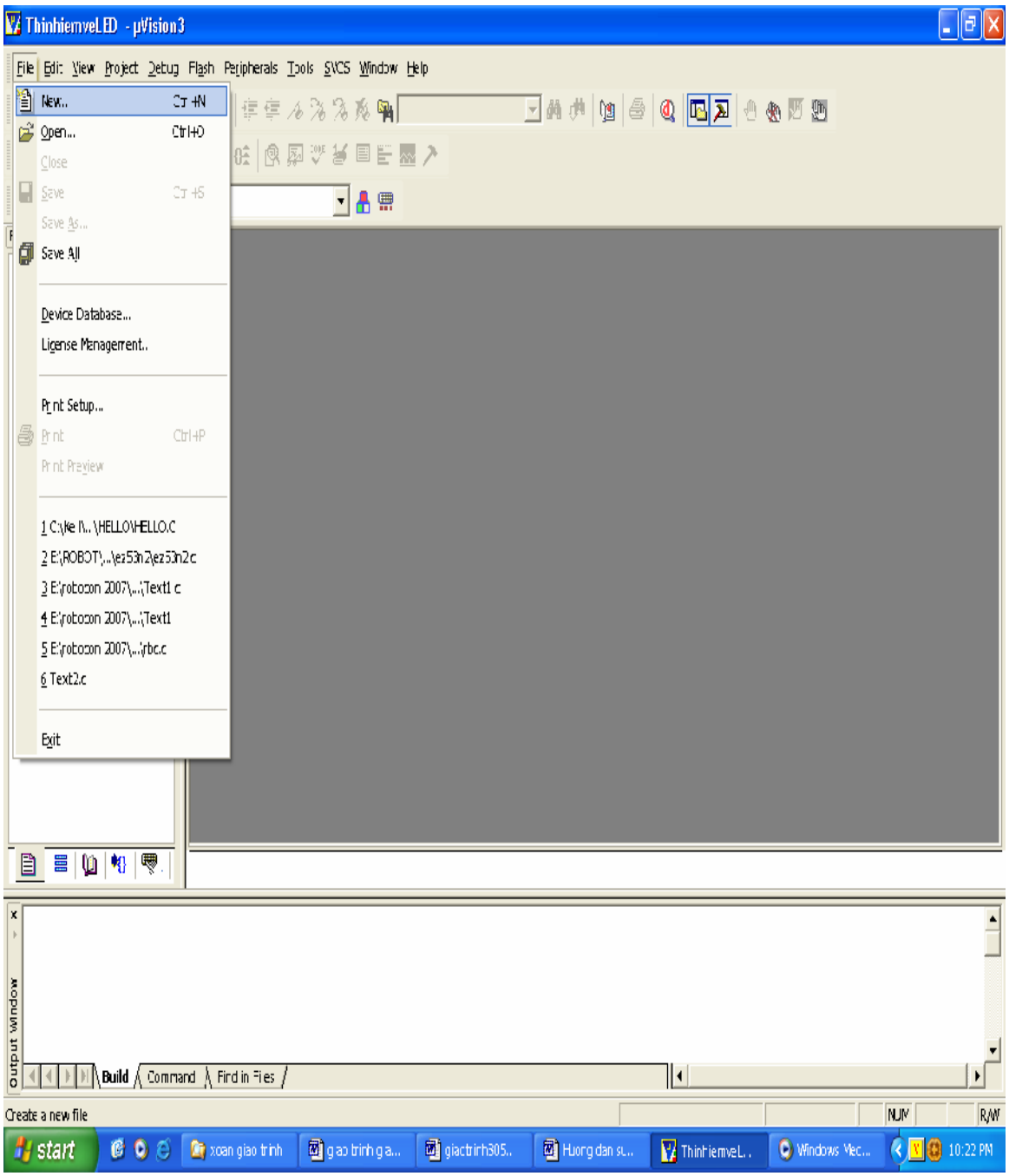
Khi chọn chip thì ngay lập tức cái bảng hiện ra 1 số tính năng của chíp các bạn có thể nhìn thấy. 8051 based Fully Static 24Mhz … . Nhập OK được cửa sổ như sau:



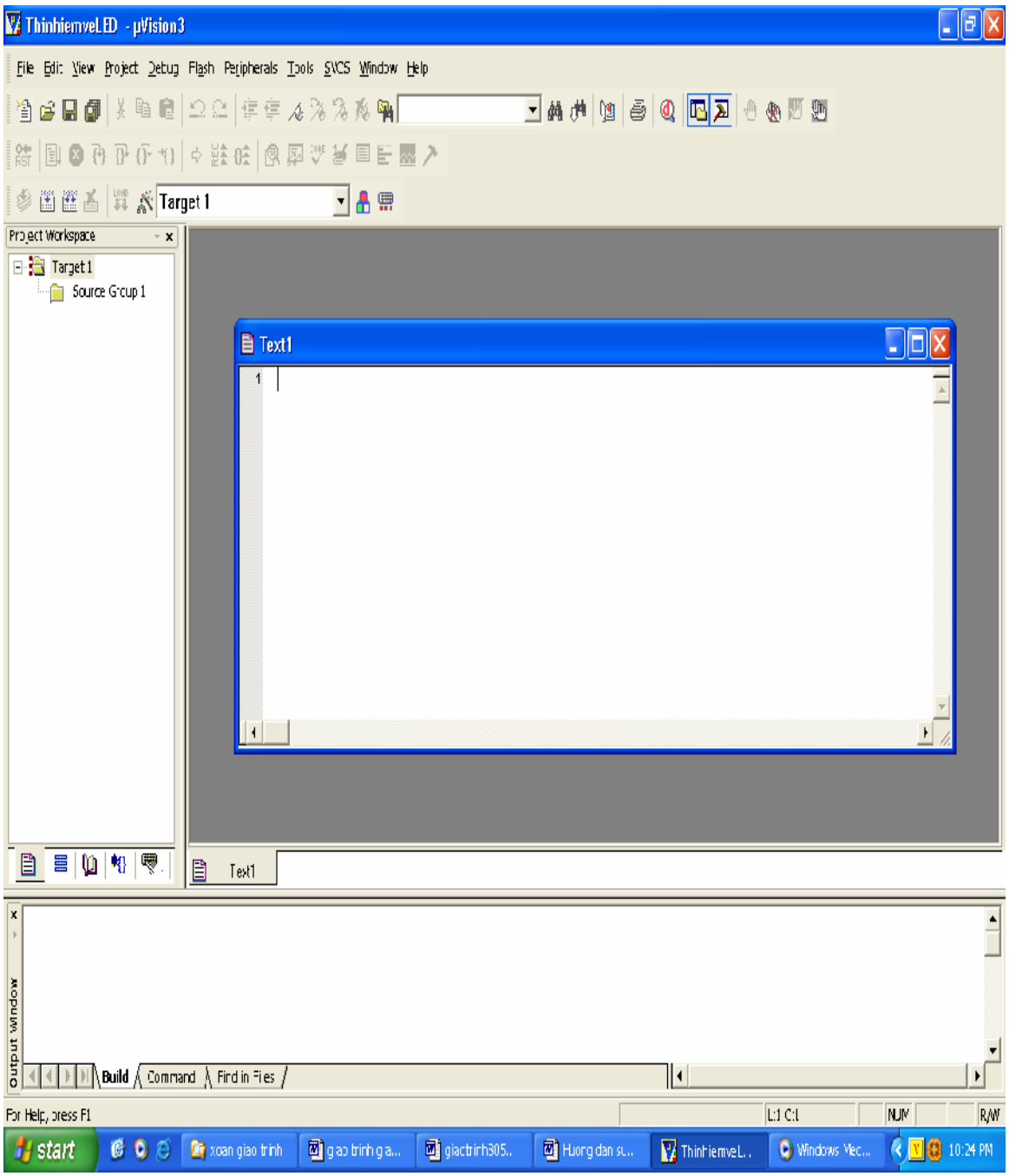
Chúng ta sẽ chọn No. Vì nếu chọn Yes chỉ làm cho file lập trình của bạn thêm nặng . Chúng ta được cửa sổ sau:



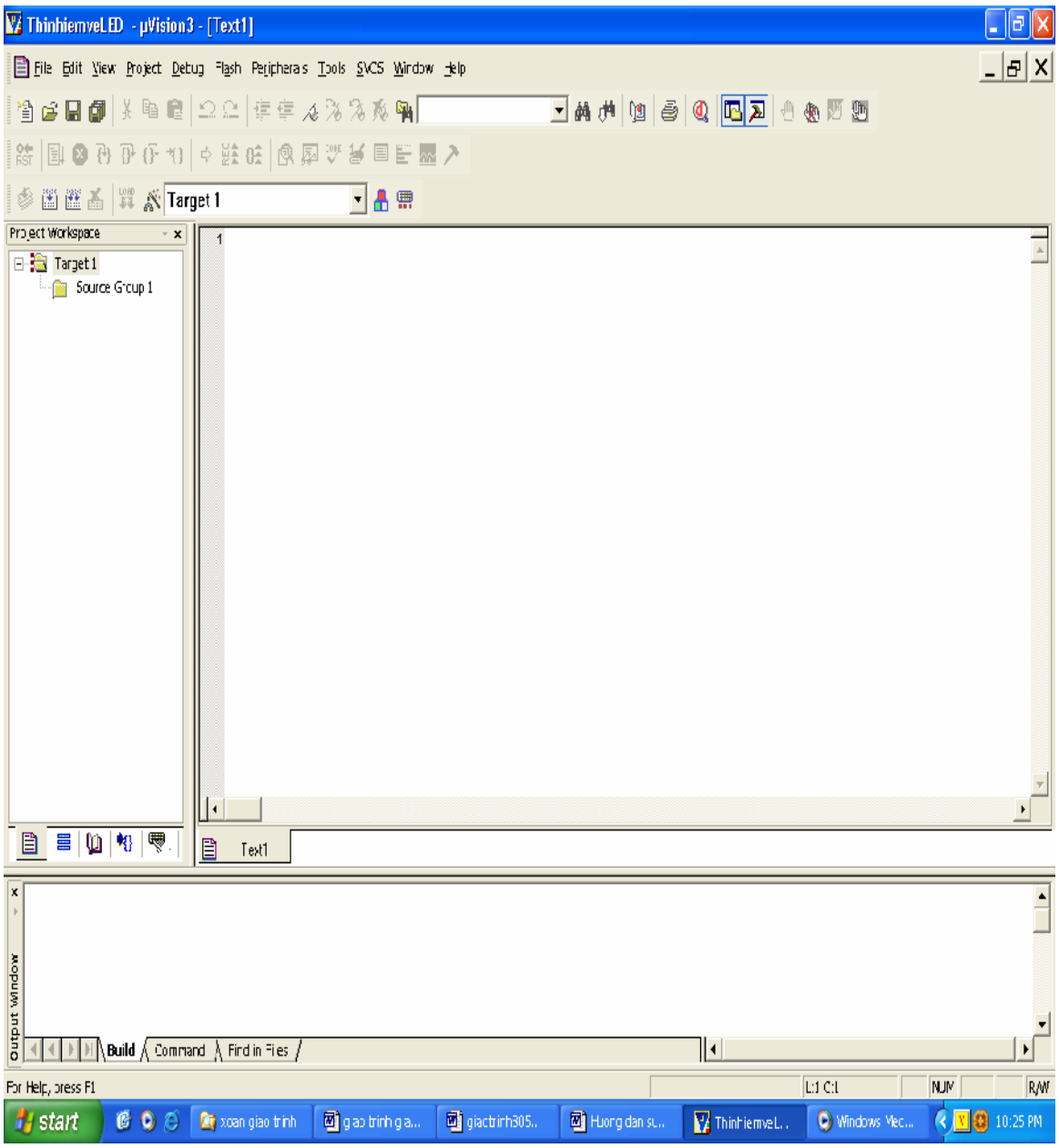
Để tạo 1 file code các bạn chọn File → New hoặc ấn Ctrl+N. Như sau:



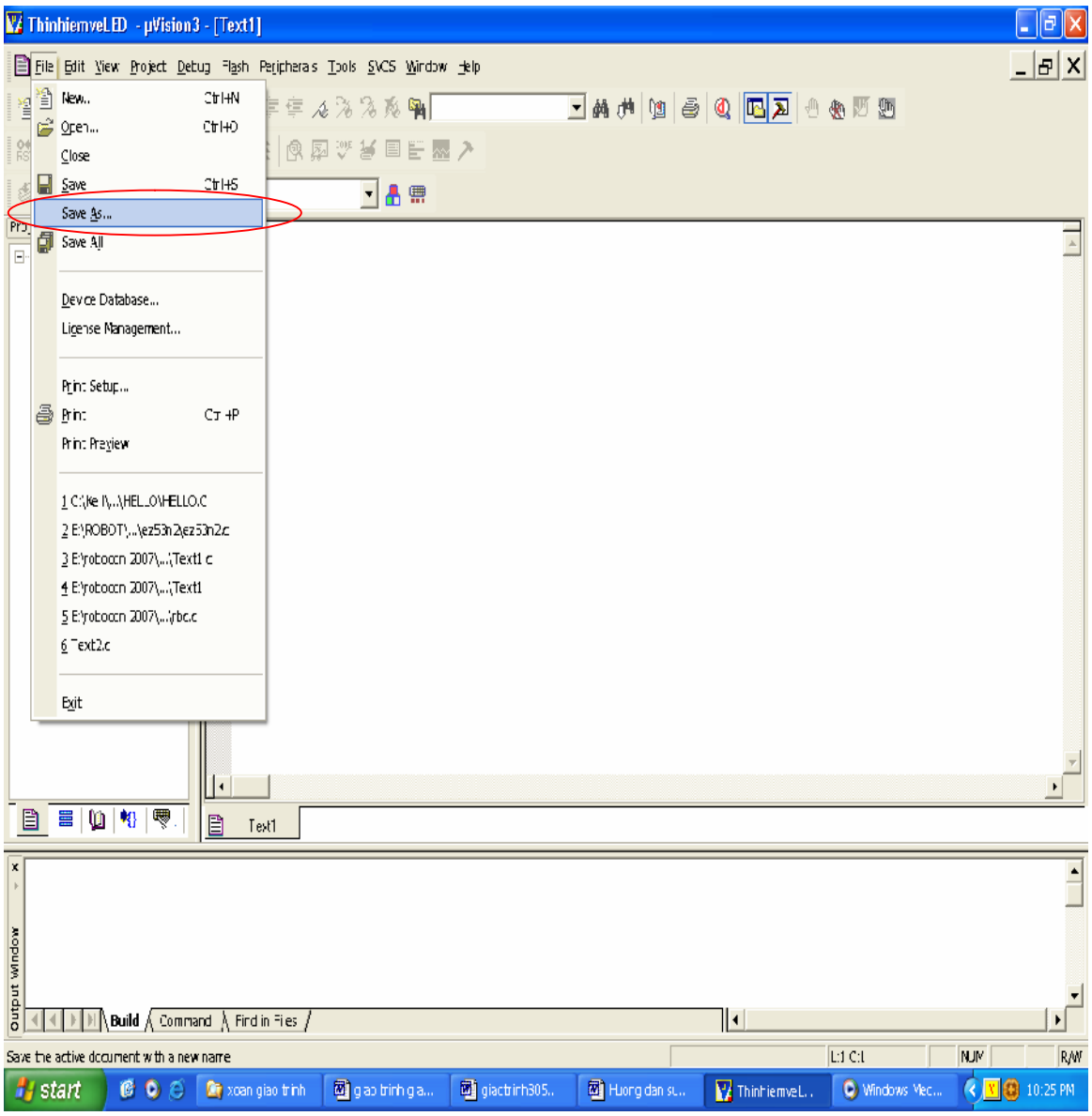
Được cửa sổ như sau:



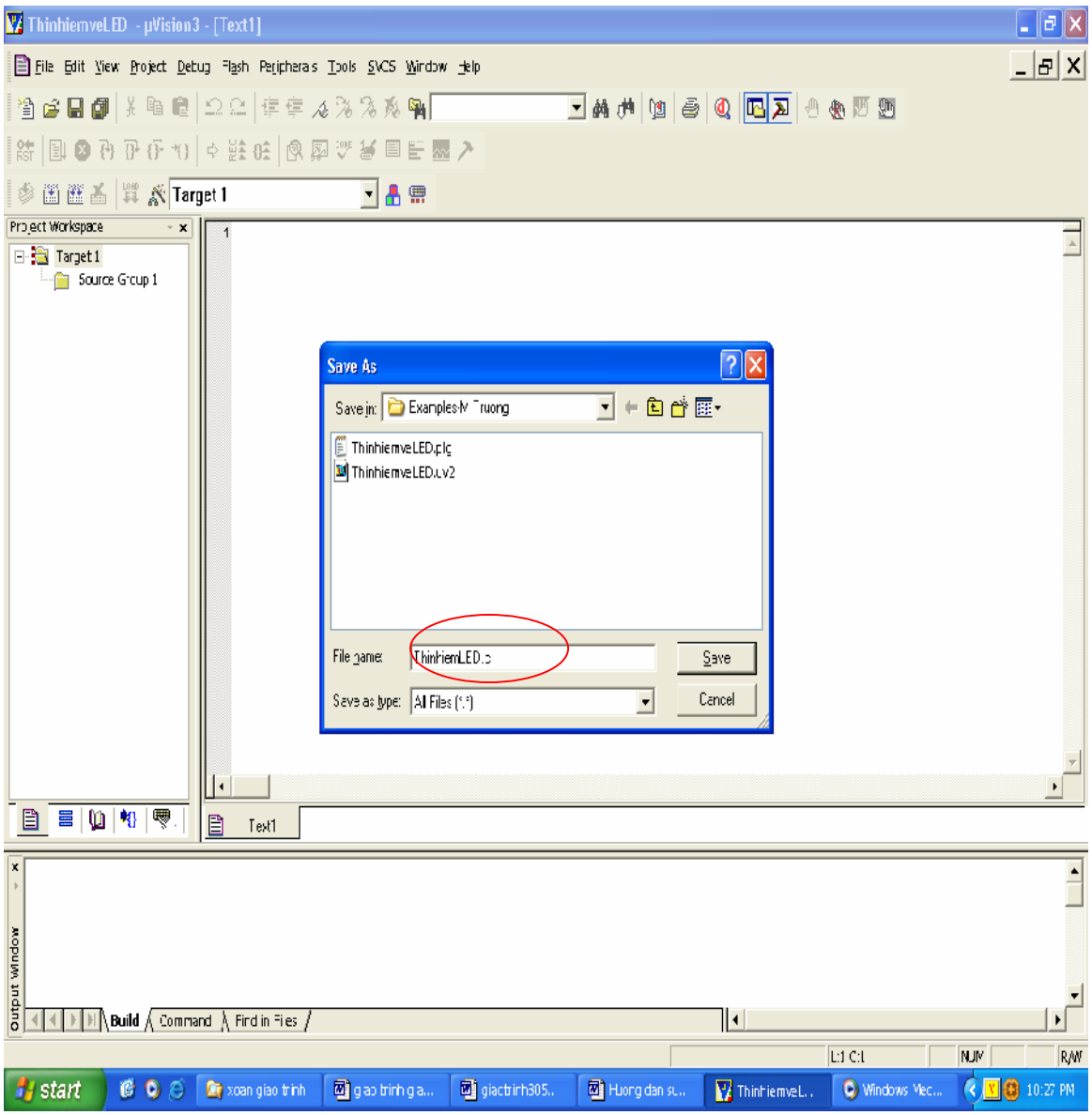
Cho cửa sổ Text 1 to ra được như sau:



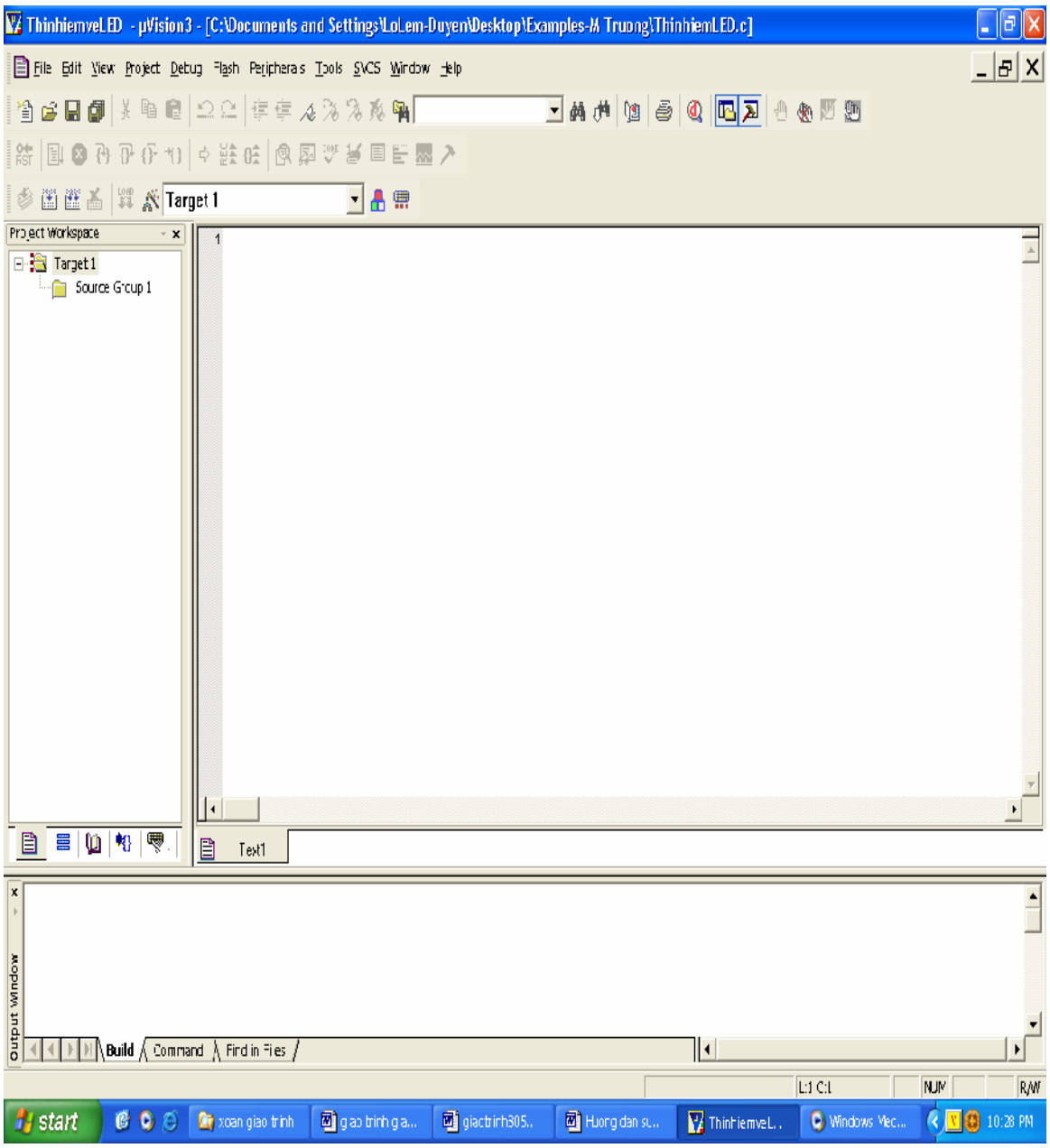
Tiếp theo bạn chọn File → Save As… hoặc Ctrl+S. Để nhớ file mặc dù chưa có gì. Như sau:



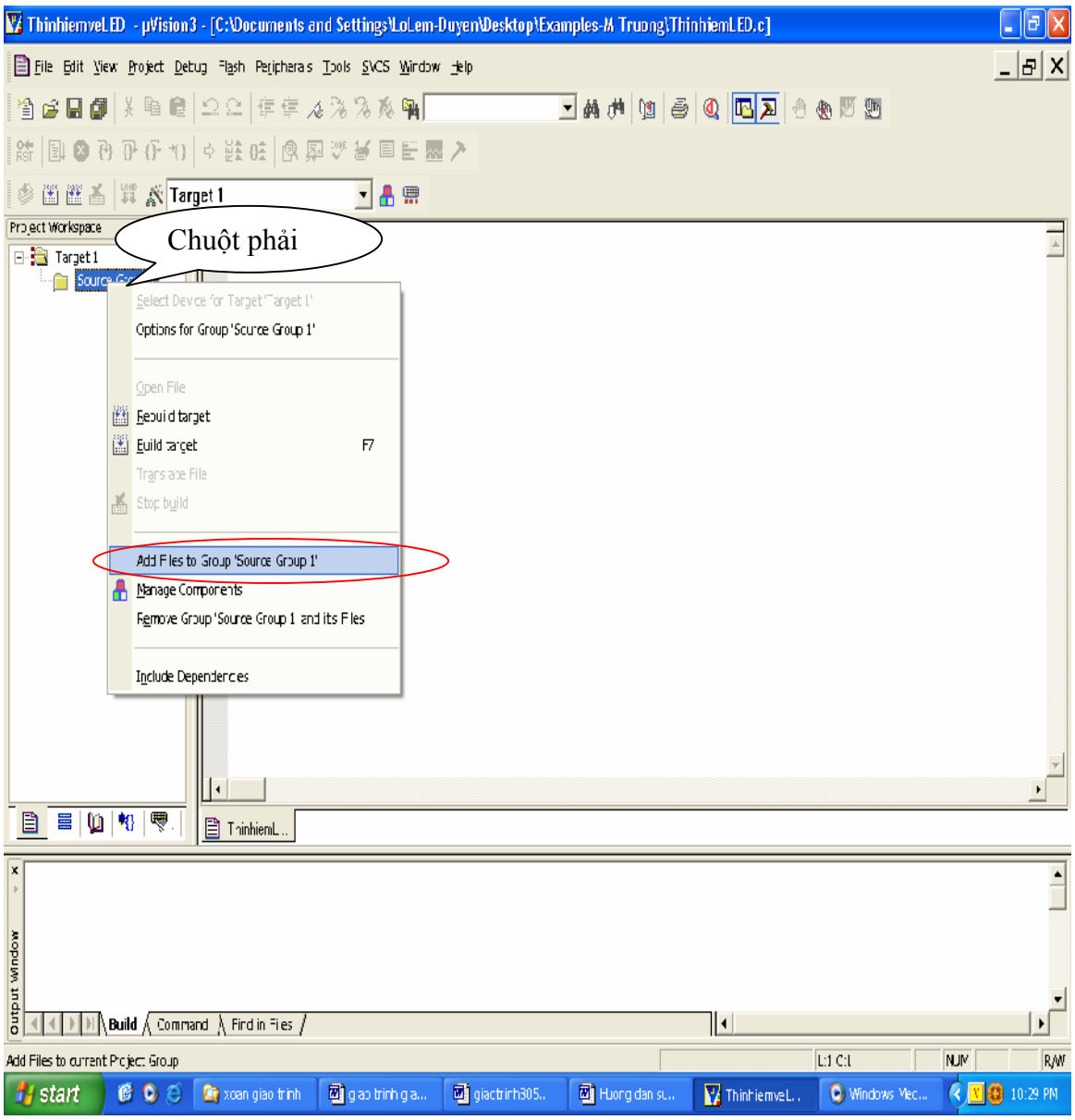
Được cửa sổ sau:



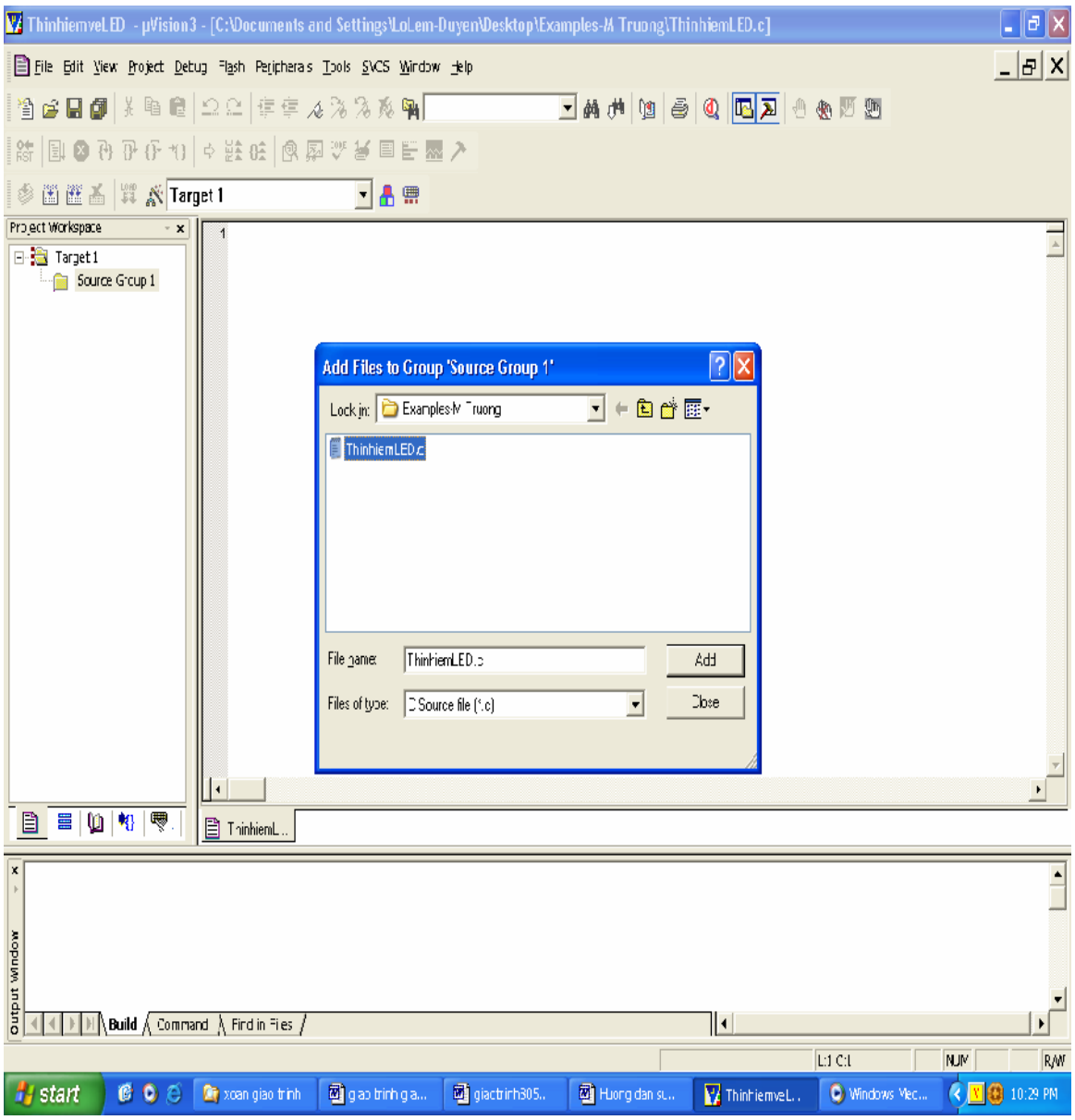
Các bạn nhập tên vào text box file name. Chú ý tên gì cũng được nhưng không được thiếu đuôi mở rộng .C . Và nhấn Save



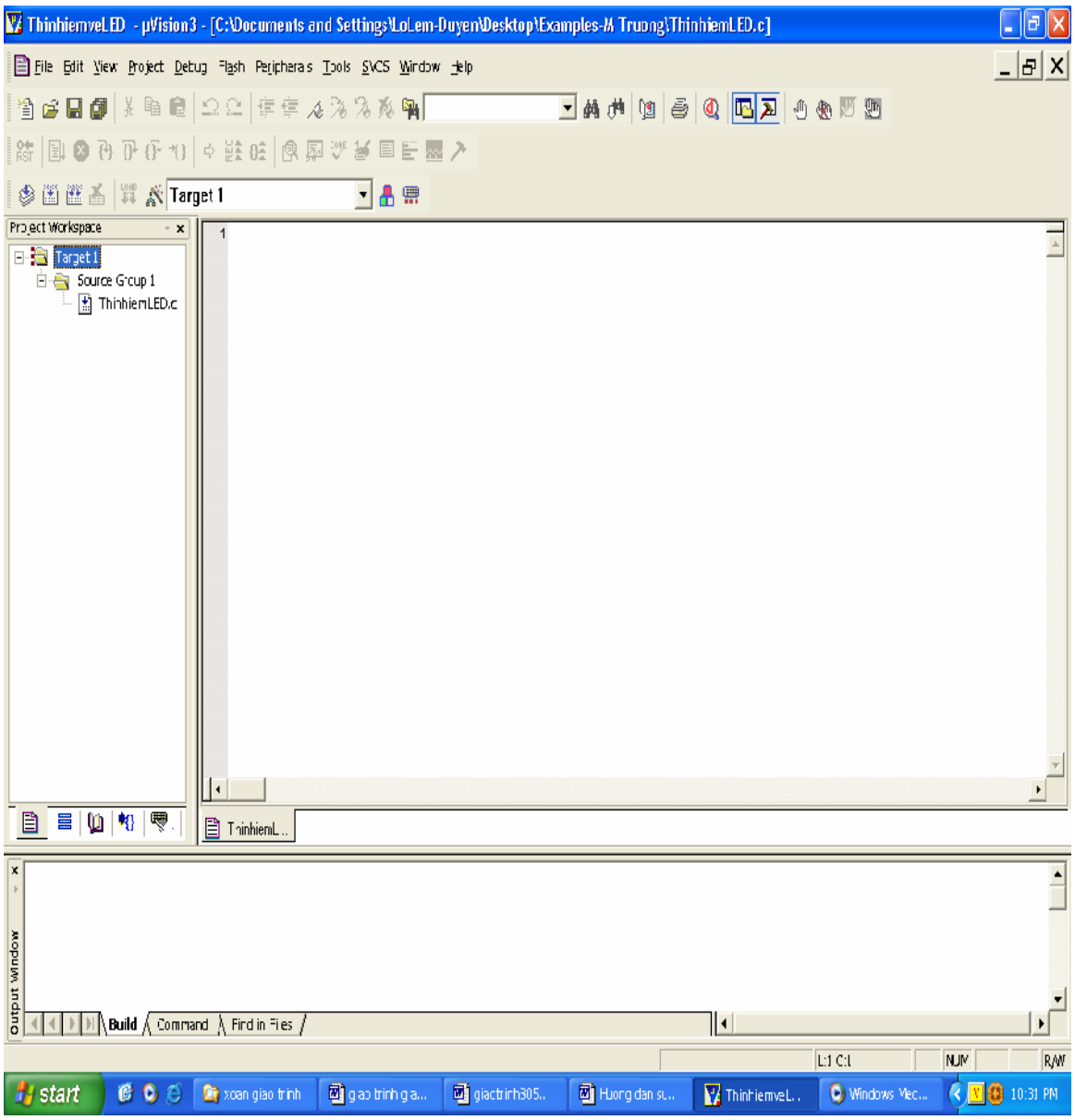
Trong ô bên trái màn hình, cửa sổ project workspace, các bạn mở rộng cái target 1 ra được như sau:



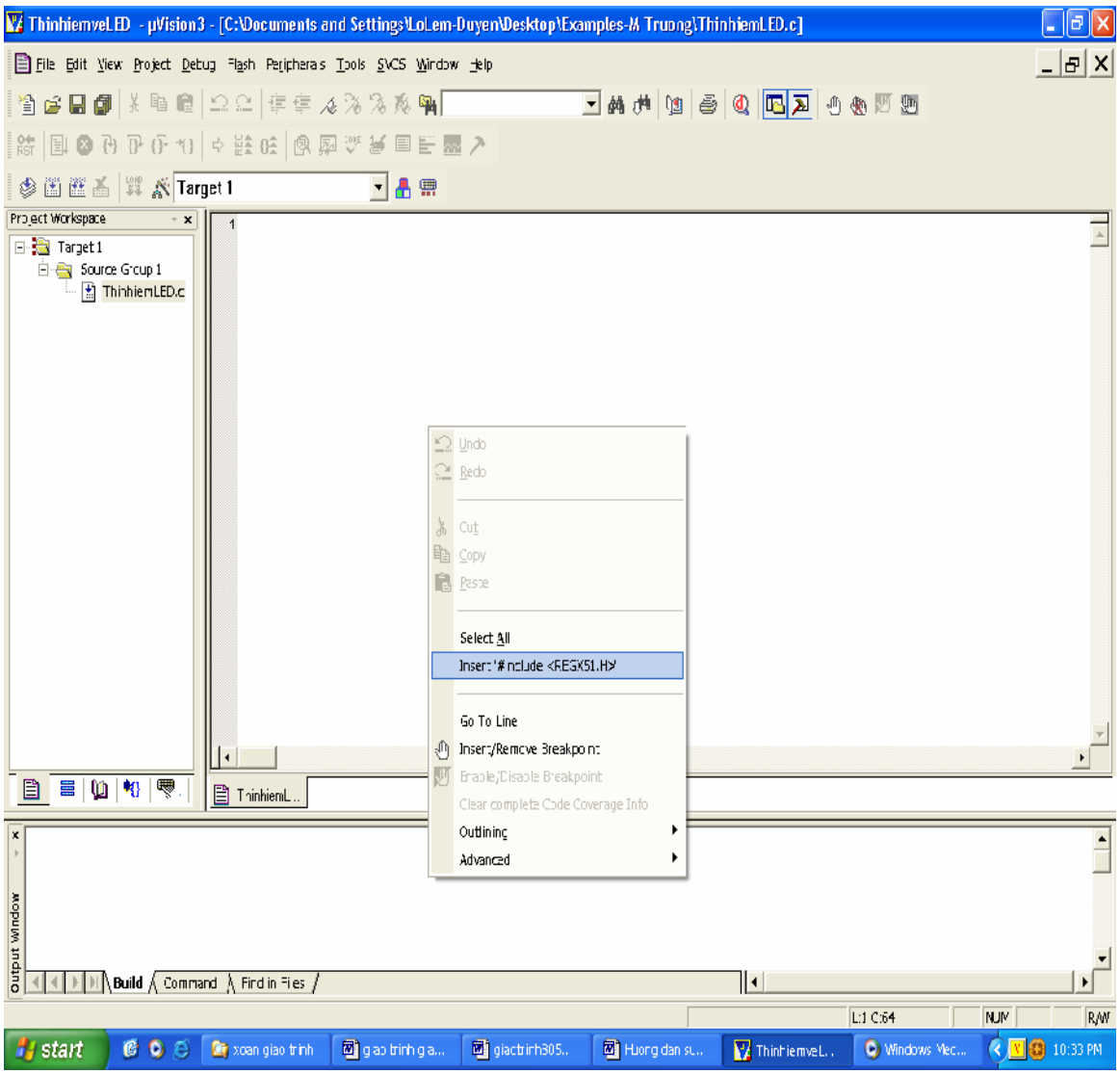
Nhấp chuột phải vào thư mục Suorce Group1 được hình như trên. Chọn Add files to Group "Source Group1" để add file vào project. Được như sau:

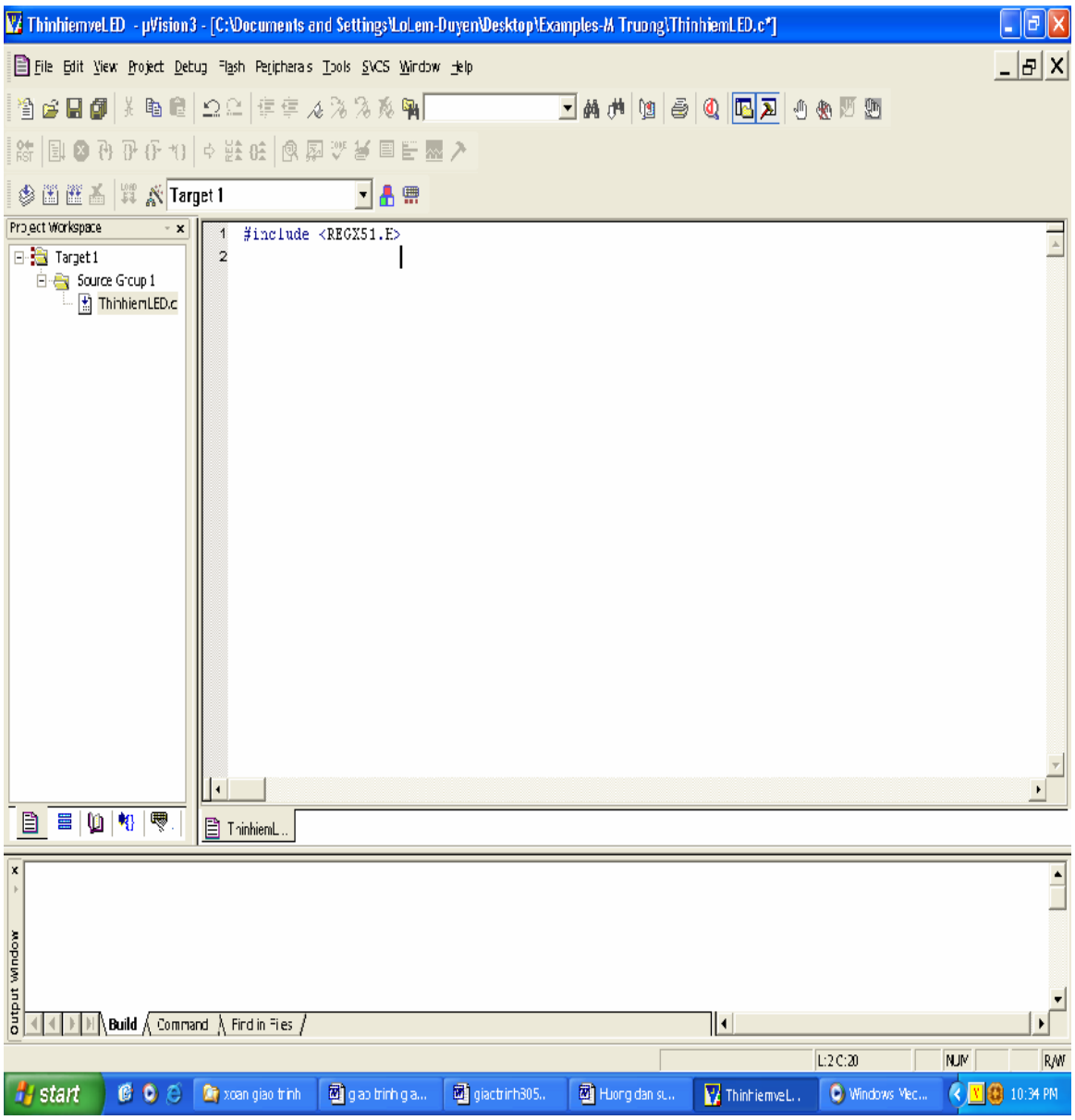


Chọn file .C mà các bạn vừa nhớ. Của tôi là ThinhiemLED.c . Nhấn Add 1 lần rồi ấn Close. Nếu bạn ấn Add 2 lần nó sẽ thông báo là file đã add bạn chỉ việc OK rồi nhấn Close. Được như sau:



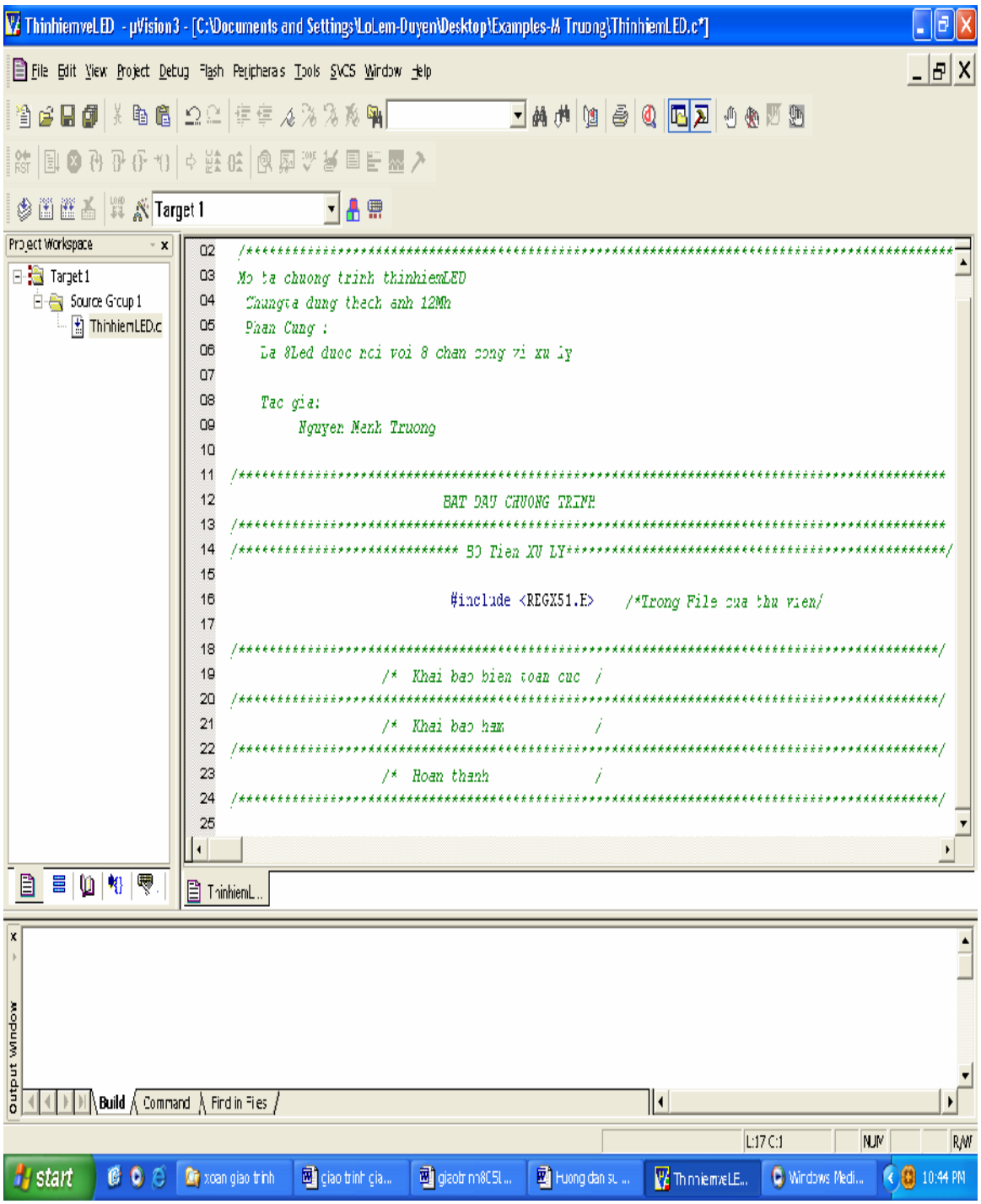
Bây giờ nhìn trong Source Group 1 đã thấy file ThinhiemLED.C . Các bạn nhấp chuột phải vào vùng soạn thảo file ThinhiemLED.C như sau, để thêm file thư viện.Chọn Insert< #include <REGX51.H>


ThinhiemveLED - µVision3 - [C:\Documents and Settings\LoLem-Duyen\Desktop\Examples-M Truong\ThinhiemLED.c]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Target 1
Project Workspace
Target 1
Source Group 1
ThinhiemLED.c
Undo
Redo
Cut
Copy
Paste
Select All
Insert '#include <REGX51.H>'
Go To Line
Insert/Remove Breakpoint
Enable/Disable Breakpoint
Clear complete Code Coverage Info
Outlining
Advanced
Output Window
Build
Command
Find in Files
L:1 C:64
NUM
R/W
start
10:33 PM



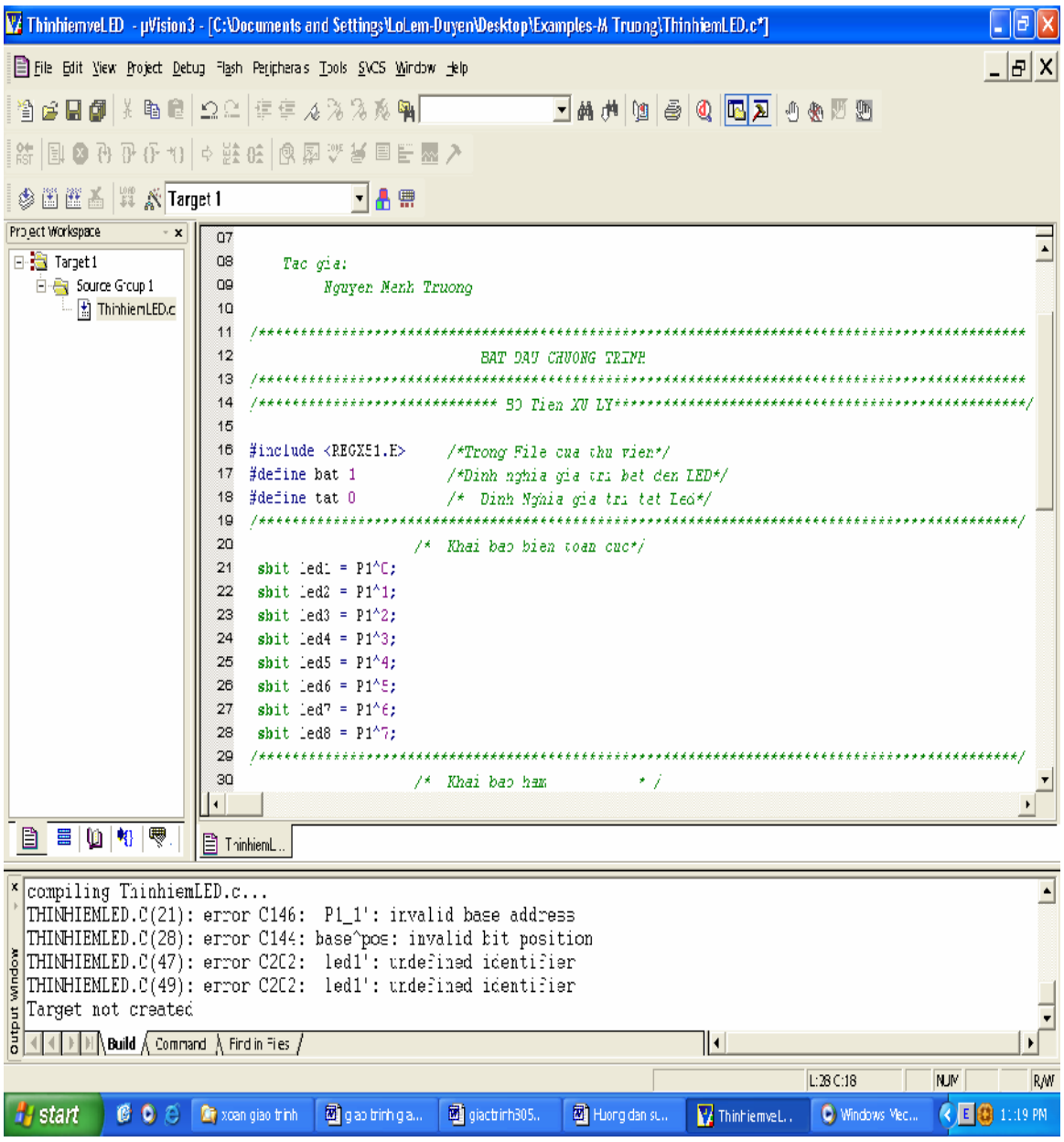
Phần cuối cùng của công việc khởi tạo là các bạn viết lời giải thích cho dự án của mình .Phần này rất cần thiết vì nó để người khác hiểu mình làm gì trong project này và khi mình cần sử dụng lại code đọc lại mình còn biết nó là cái gì.

Các bạn tạo lời giải thích theo mẫu sau:

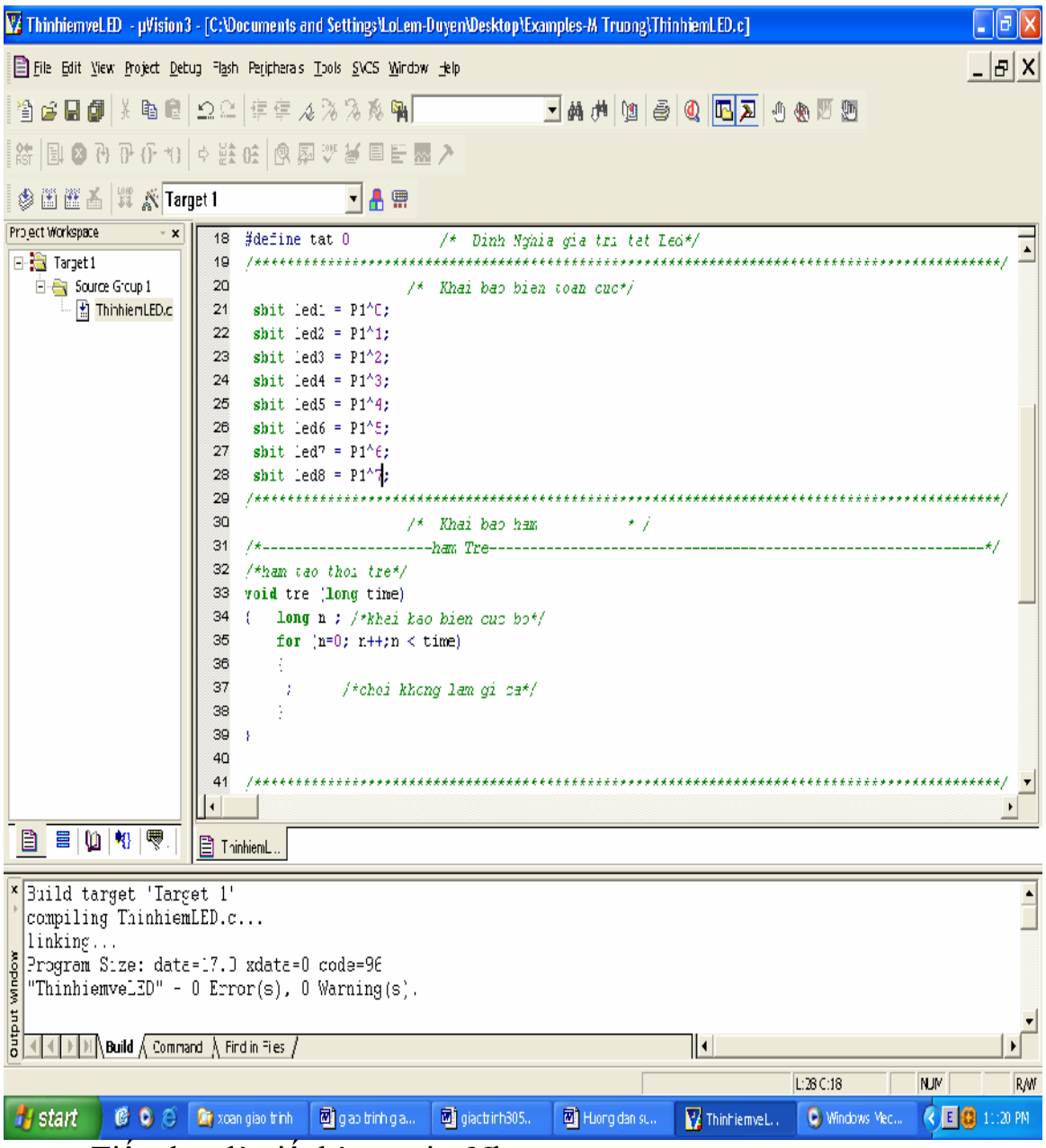


2.2.2 Soạn thảo chương trình:

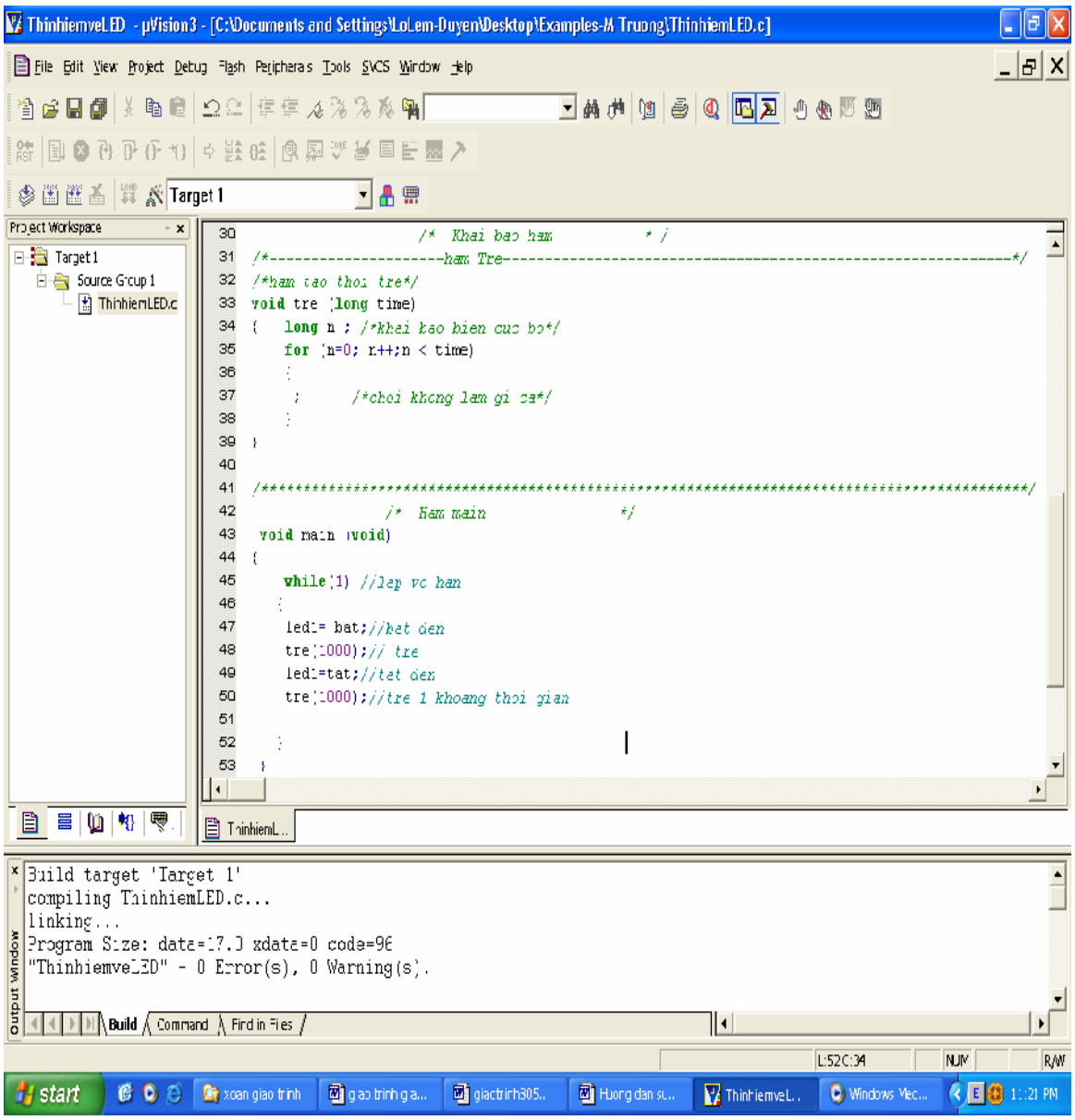
Các bạn viết chương trình thử 1 chương trình như sau làm ví dụ. Khi viết xong 1 dòng lệnh nên giải thích dòng lệnh đó làm gì. Ví dụ:



Các bạn nên chia chương trình như tôi làm. Với 1 file nhỏ thì nó hơi rườm rà. Nhưng với 1 file lớn khoảng 1000 dòng code thì nó lại rất sáng sủa. Các bạn nên tạo 1 file mẫu rồi nhớ vào 1 file text để ở đâu đó mỗi lần dùng chỉ việc copy rồi paste qua chứ không nên mỗi lần tạo một cái như vậy lại phản tác dụng. Phía trên là phần bộ tiền xử lí và khai báo biến. Tiếp theo là viết hàm trễ.



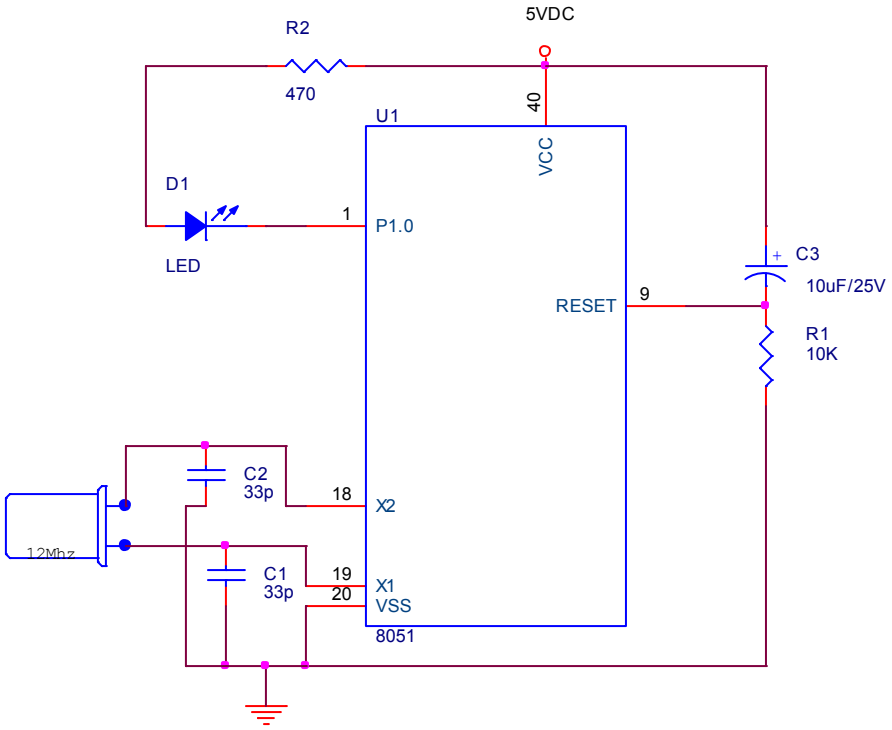
Tiếp theo là viết hàm main. Như sau:



Rồi nhấn Ctrl+S. Hoặc chọn File → Save để nhớ file vừa soạn thảo.

Các bạn nhìn vào code có thể các bạn đã hiểu con AT89C51 nó làm gì nếu các bạn đã nắm vững các bài trước. Còn nếu không hiểu thì tôi sẽ giải thích lại cho các bạn.

Đây là sơ đồ nguyên lí của 1 led. Project là 8 led nhưng tôi chỉ giải thích 1 led là các bạn hiểu. Mục đích là làm con led nhấp nháy.



Biến Led1 được khai báo (gán cho) chấn P0_1 của vi điều khiển bằng câu lệnh sbit Led1=P1^0; . Giá trị bật bat được định nghĩa là 0.

Khi các bạn gán : Led1=bat; trong hàm main thì chân P1_0 của AT89C51 có mức logic là 0V. Theo sơ đồ nguyên lí: 5V → Trở 470 → Led1 → P1_0 (0 V). Có chênh lệch áp → có dòng điện qua led → Led sáng. Các bạn có thể tính tóan chỗ này dễ dàng là tại sao lại là trở 470 Ôm. Điện áp mất ở led là Uak (0,6 đến 0,7V) lấy =0,6V.Điện áp chân P1_0 là 0V. Điện áp hai đầu trở : 5V -0,6V = 4,4V. Dòng qua trở = dòng qua led = 4,4V/470 Ôm xấp xỉ 10 mA. Với dòng 10mA đến 15mA là led đủ dòng để sáng và sáng rất đẹp. Nếu dòng yếu thì led mờ, còn dòng lớn thì các bạn biết sao rồi đấy.

Khi các bạn gán: Led1= tat; tức là chân P1_0 có giá trị 1 tương ứng điện áp của nó là 5V . Hiệu điện thế giữa hai đầu +5V và P1_0 là 0V . Nên không có dòng qua led → Led tắt.Nhưng nếu trong hàm main các bạn viết chỉ có như sau:

```
While(1)
{
Led1=bat;
Led1=tat;
}
```

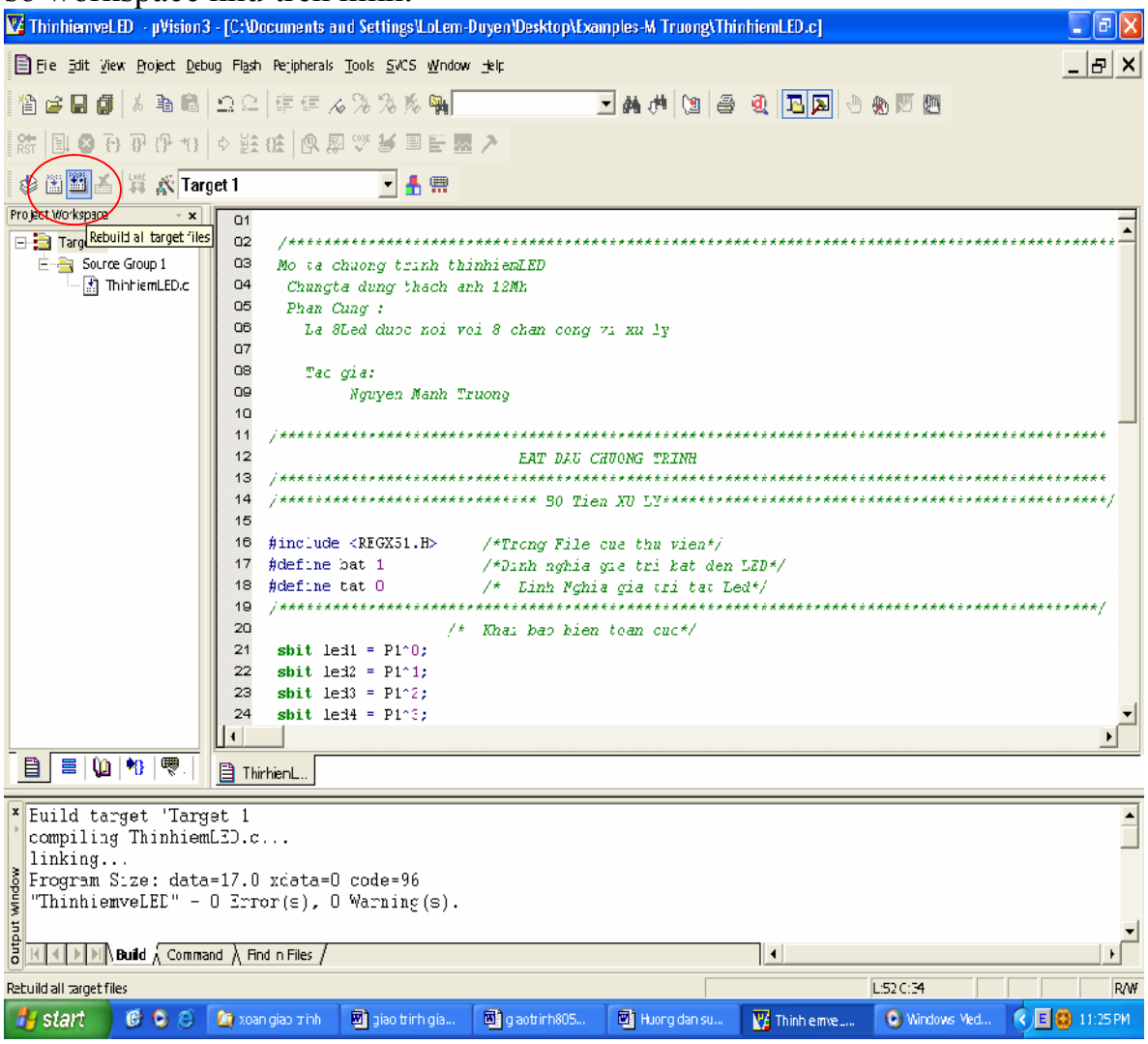
Khi chạy debug thì vẫn thấy led nhấp nháy. Nhưng khi nạp chương trình vào chíp lắp vào mạch thì led không nháy hoặc chỉ sáng mờ hoặc tắt ngóm. Vì lệnh Led1=bat; là lệnh 1 chu kì máy , tần số thạch anh là 12 Mhz, 1 chu kì máy có thời gian là 1uS. Vừa bật lên 1 uS rồi lại tắt ngay. Led không đáp ứng được tần số cao vậy nên không nhấp nháy. Do đó cần tới hàm trễ . Bật led lên trễ 1 thời gian khá lâu(0,5 giấy), rồi tắt led đi khá lâu(0,5s) rồi lại bật lại tạo thành vòng lặp sẽ được led nhấp nháy.

Tác dụng của câu lệnh while(1) . Điều kiện bên trong vòng while là 1 luôn luôn đúng nên nó là vòng lặp vô hạn lần. Nếu không có vòng while(1) thì

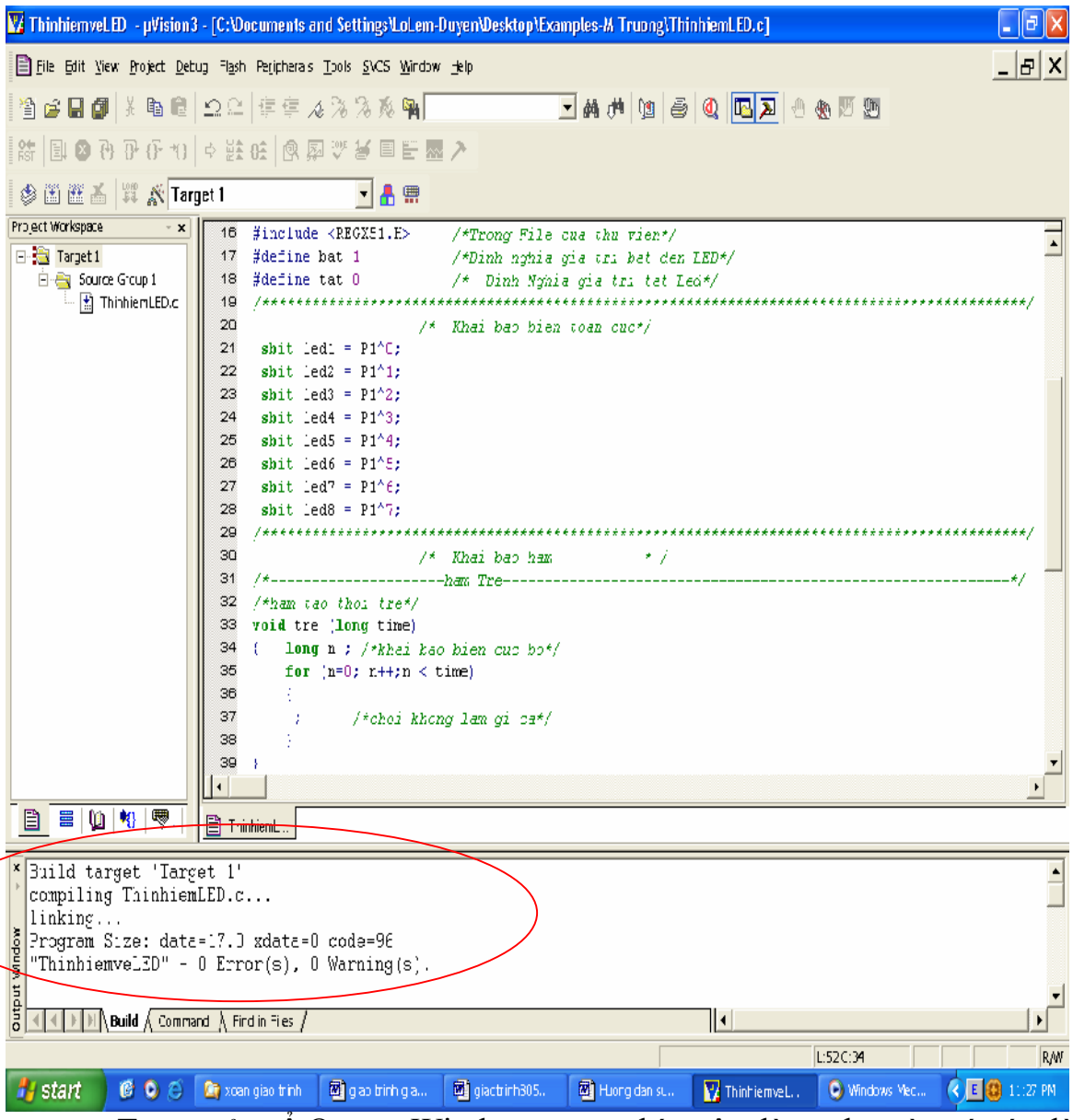
led của các bạn chỉ sáng lên 1 lần rồi tắt.

2.2.3 Dịch chương trình:

Soạn thảo song nhấn Ctrl +S để nhớ . Nhớ xong các bạn biên dịch chương trình bằng cách ấn phím F7 hoặc chọn Build target là biểu tượng ngay trên cửa sổ workspace như trên hình:



Các bạn sẽ thấy như sau:



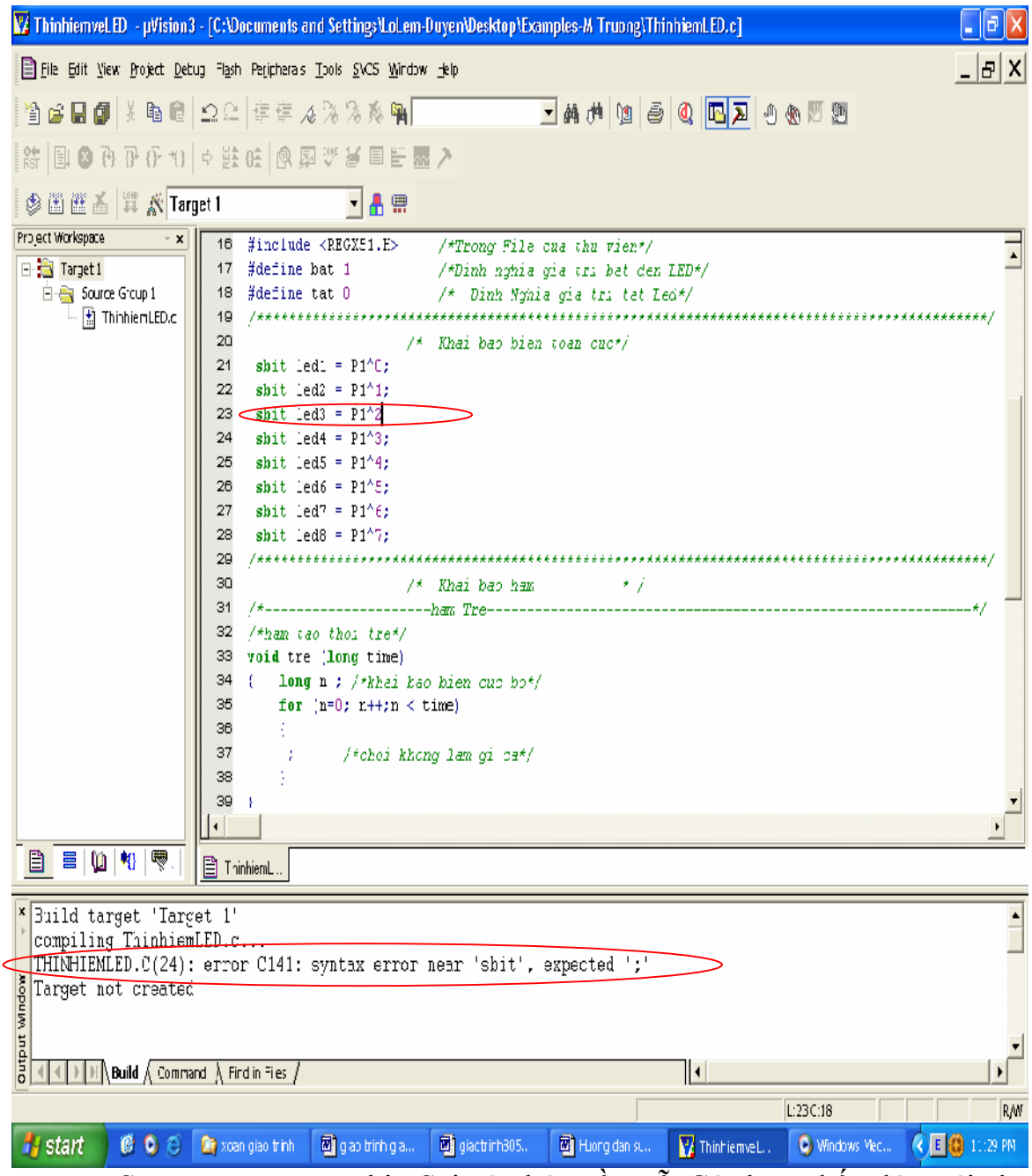
Trong cửa sổ Output Window ngay phía trên dòng chữ này có các dòng chữ Compiling …

Linking…

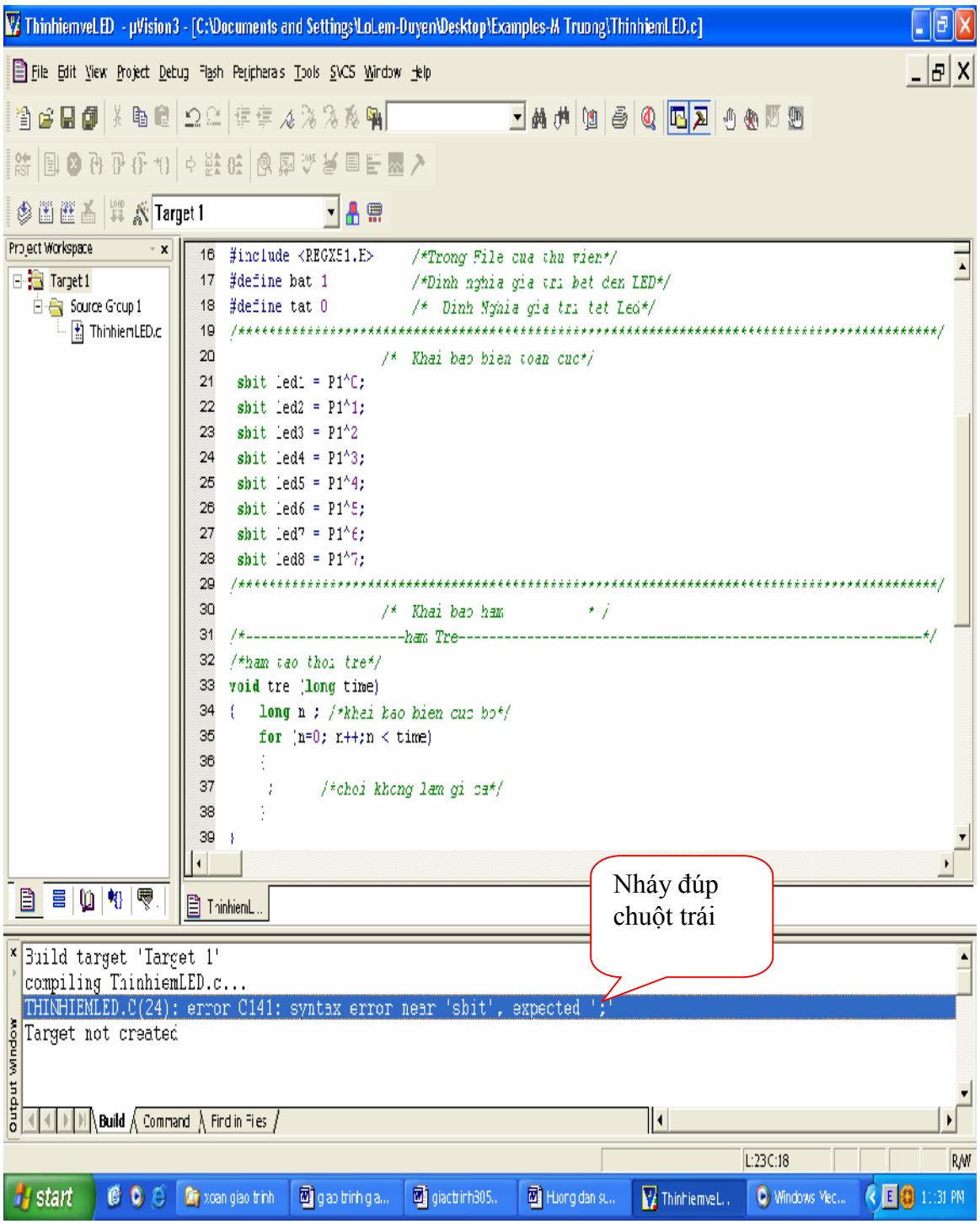
Program Size: data =17.0 code =96

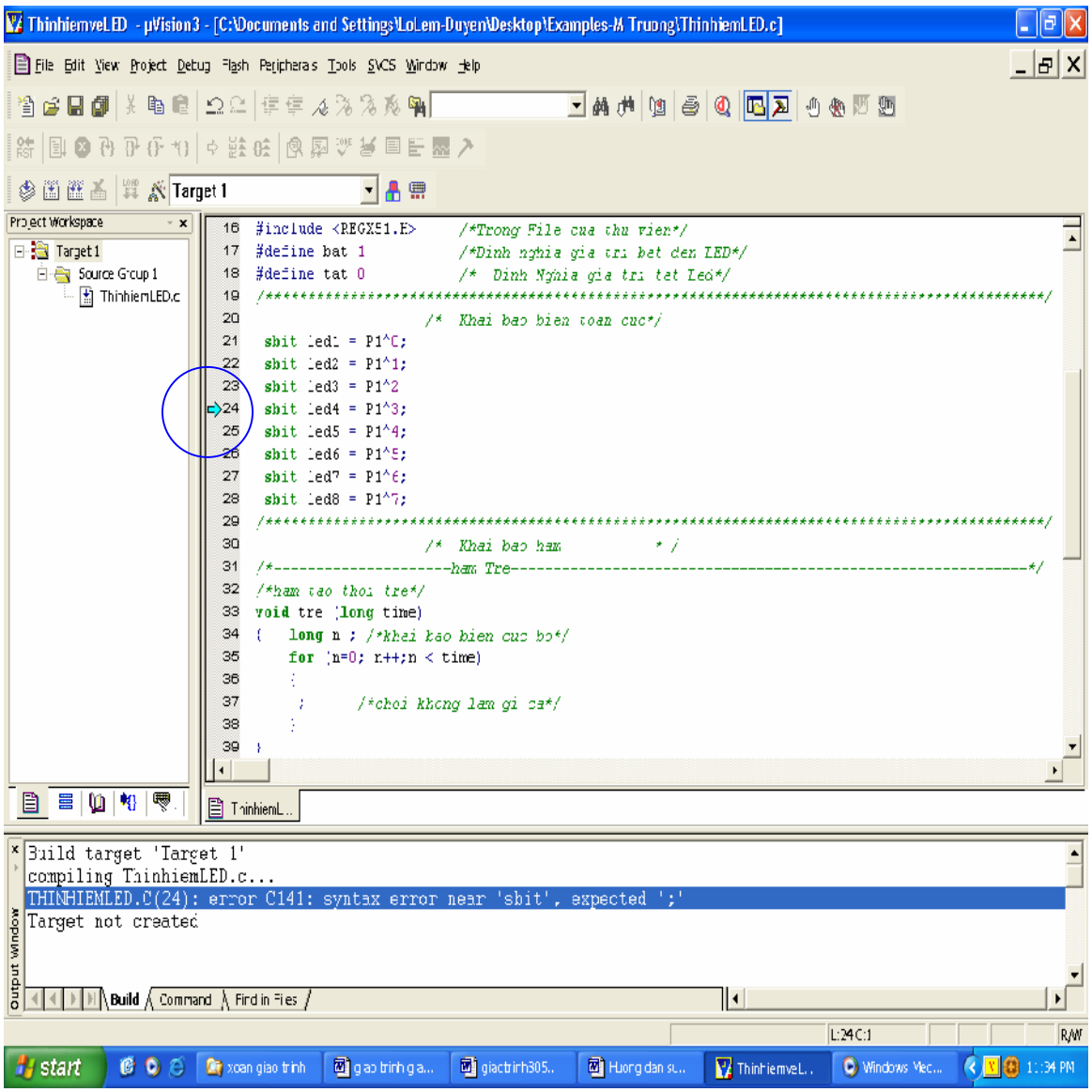
… 0 error , 0 Warning .

Như vậy là OK. Nếu không được như vậy nó sẽ báo lỗi và các bạn kiểm tra xem soạn thảo đúng chưa. Tôi ví dụ xóa 1 dấu ; ở trong hàm main ở dòng : Led1=bat; , giờ bỏ đi thành Led1= bat .Rồi dịch lại (ấn F7) trình biên dịch sẽ báo như sau:

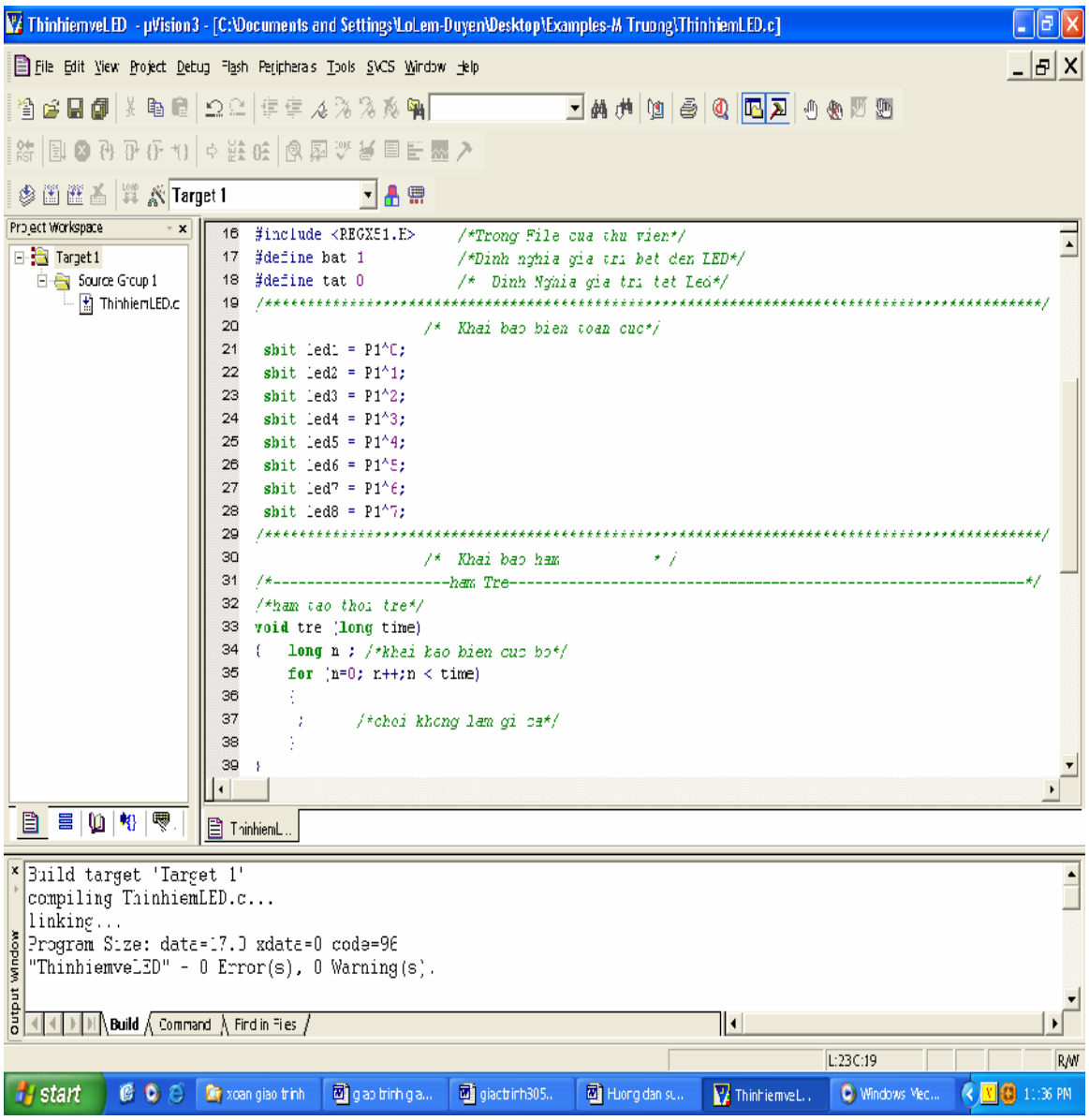


Syntax error near sbit. Sai cú pháp gần trễ. Các bạn nhấp đúp trái chuột vào dòng thông báo này con trỏ sẽ ở ngay dòng dưới dòng có lỗi thêm dấu nhìn dấu mũi tên màu xanh ở hình dưới đây, gõ vào dấu ; và dịch lại là OK.” Trong chương trình lớn đôi khi con trỏ chỉ đến gần chỗ có lỗi thôi và bạn phải tự tìm ra lỗi.


ThinhiemveLED - µVision3 - [C:\Documents and Settings\LoLem-Duyen\Desktop\Examples-M Truong\ThinhiemLED.c]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Target 1
Project Workspace
Target1
Source Group 1
ThinhiemLED.c
#include <REGX51.H> /*Trong File cua thu vien*/
#define bat 1 /*Dinh nghia gia tri bat den LED*/
#define tat 0 /* Dinh Nghia gia tri tat Led*/
/* Khai bao bien toan cuc*/
sbit led1 = P1^0;
sbit led2 = P1^1;
sbit led3 = P1^2
sbit led4 = P1^3;
sbit led5 = P1^4;
sbit led6 = P1^5;
sbit led7 = P1^6;
sbit led8 = P1^7;
/* Khai bao ham */
/*--------------------ham Tre----------*/
/*ham tao thoi tre*/
void tre (long time)
{ long n ; /*khai bao bien cuc bo*/
for (n=0; n++;n < time)
;
; /*choi khong lam gi ca*/
;
}
ThinhiemL...
Nháy đúp chuột trái
Build target 'Target 1'
compiling ThinhiemLED.c...
THINHIEMLED.C(24): error C141: syntax error near 'sbit', expected ';'
Target not created
Output Window
Build
Command
Find in Files
L:23 C:18
R/W
start
xoan giao trinh
giao trinh ga...
giactrinh305...
Huong dan su...
ThinhiemveL...
Windows Vec...
1:31 PM

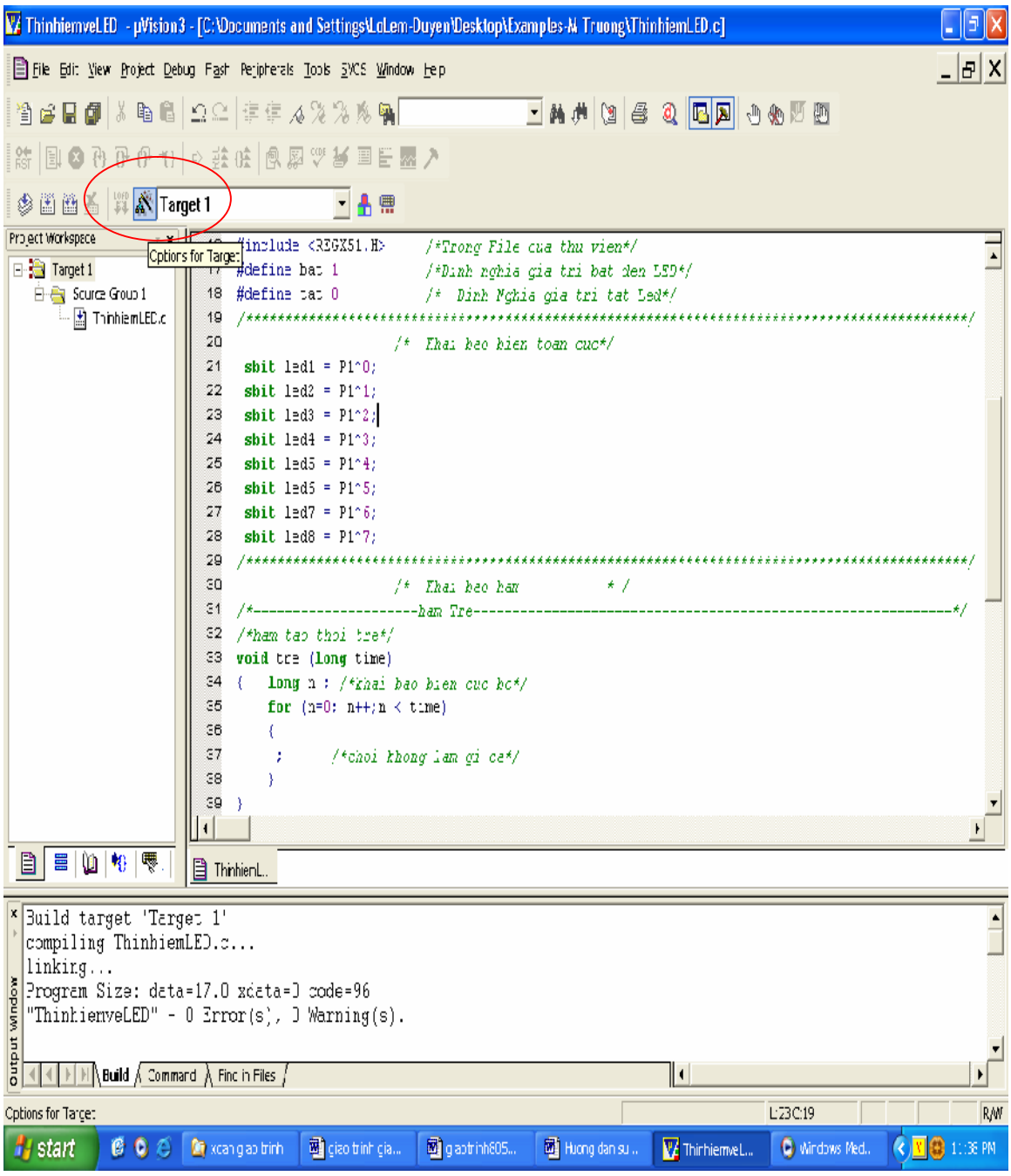


Sau khi dịch lại được hình sau:

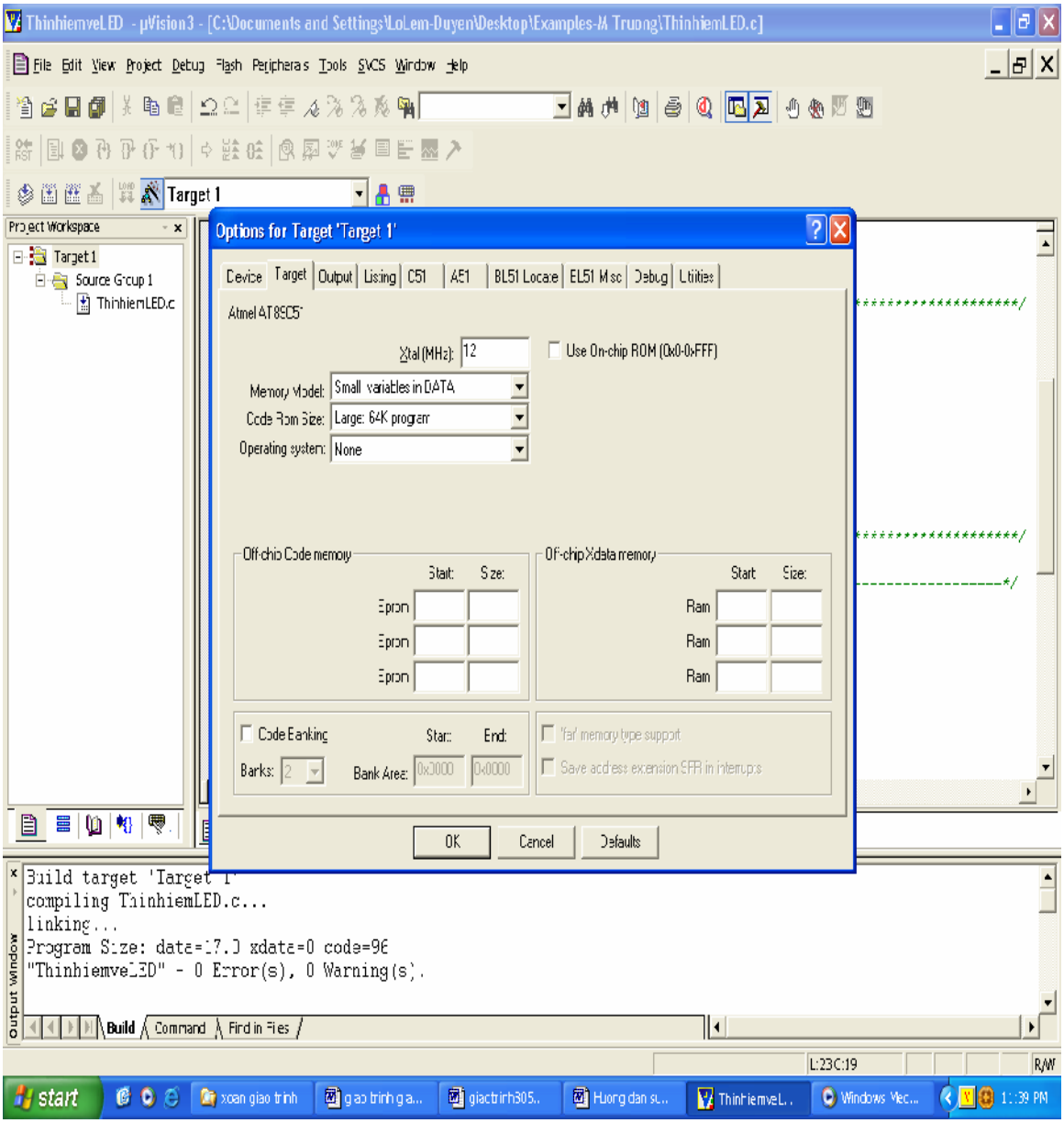


2.2.4. Chạy mô phỏng và sửa lỗi.

Trước khi debug chúng ta khởi tạo như sau. Các ban vào Option for target 1.

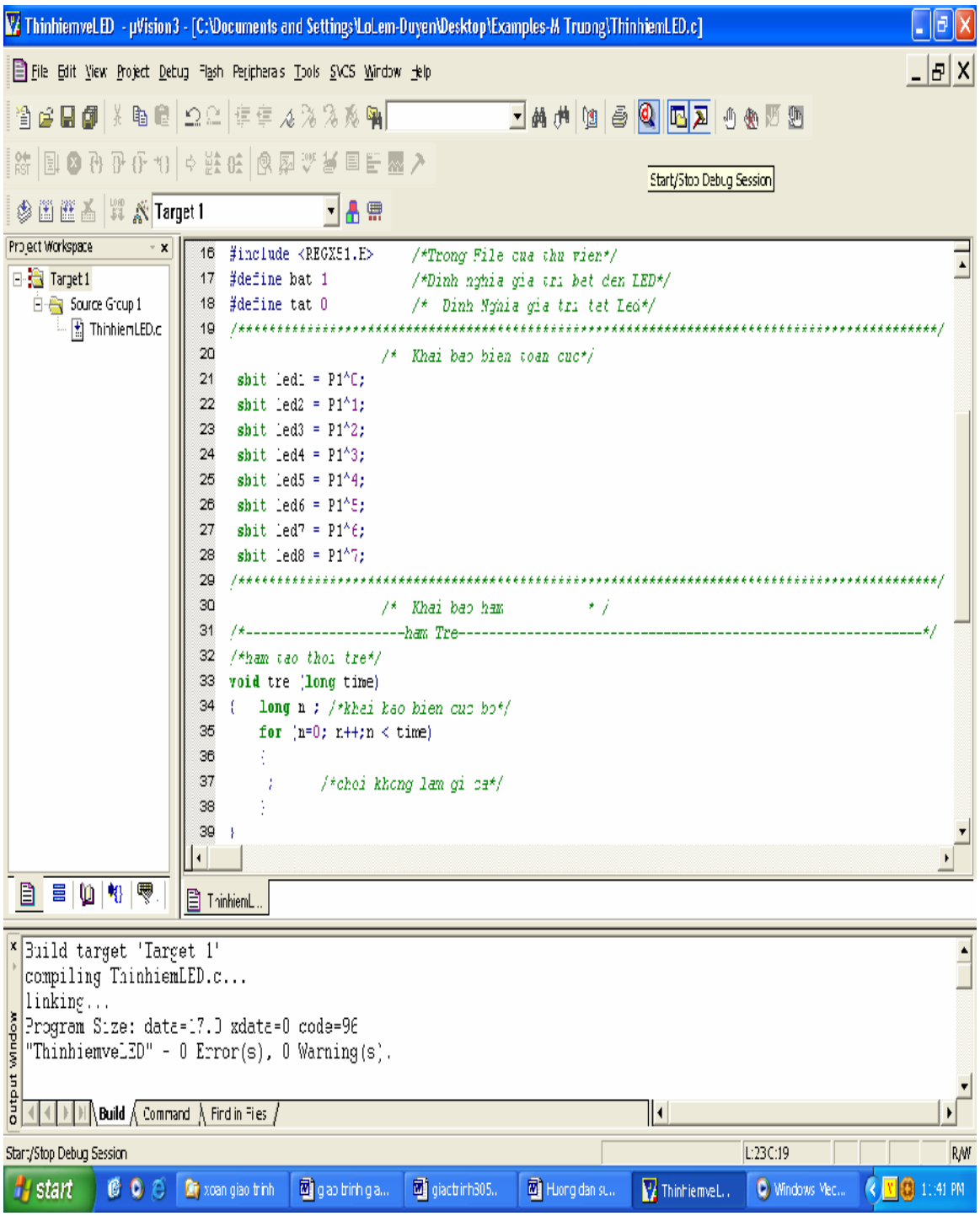


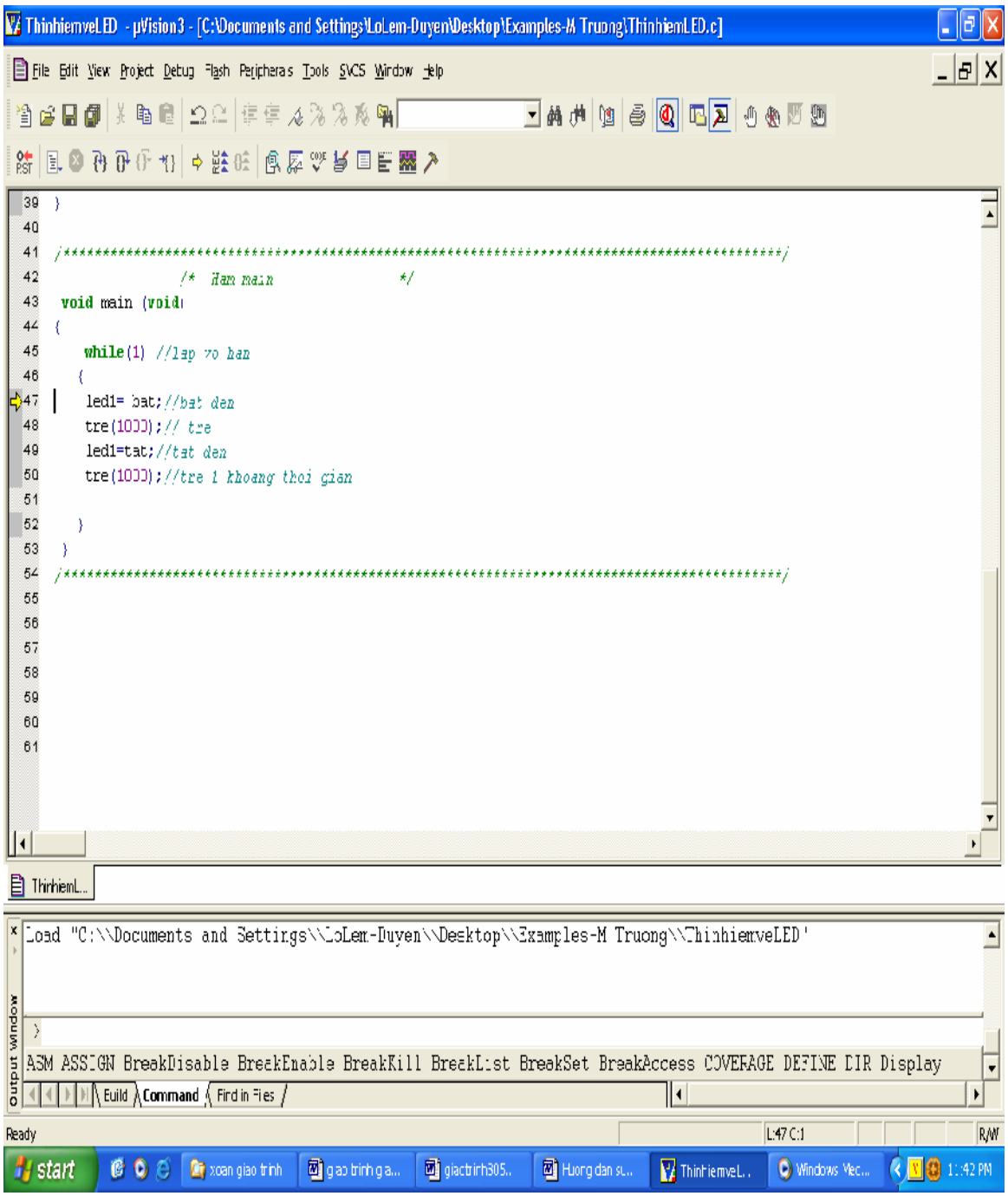
Được bảng sau. Nhập tần số thạch anh là 12 Mhz đúng với tần số thạch anh.

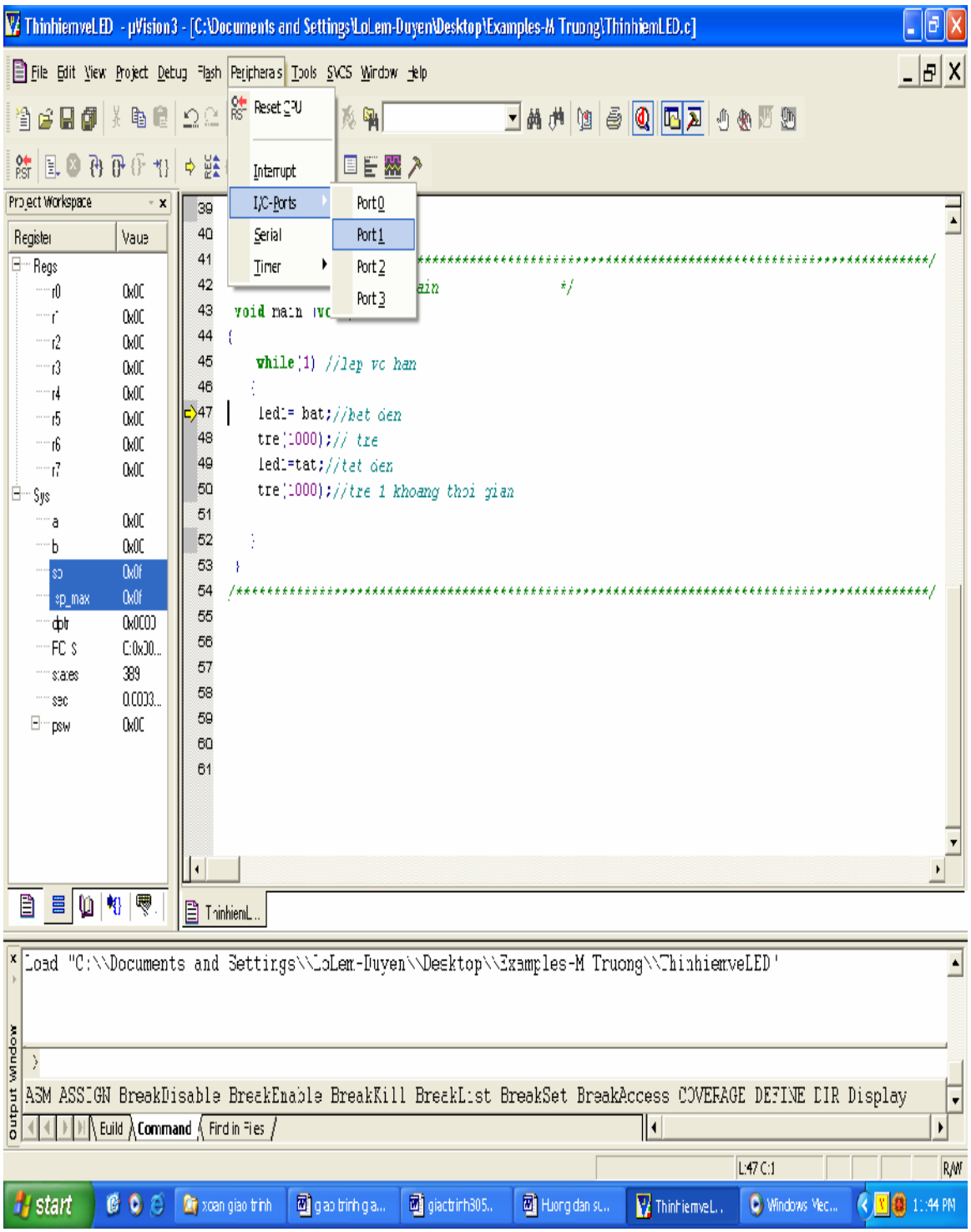


Chọn OK.

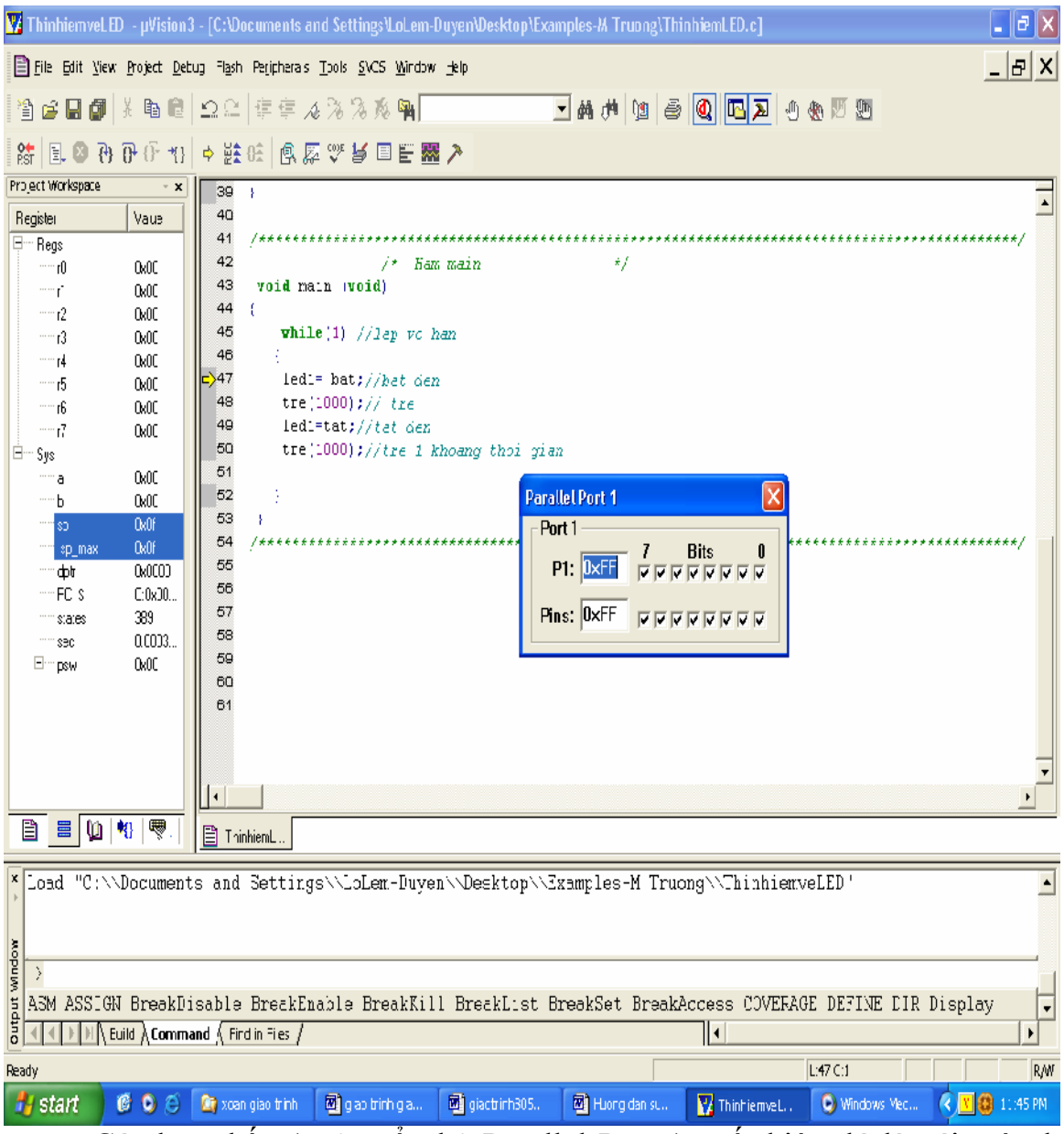
Để debug các bạn nhấn tổ hợp phím Ctrl + F5. Hoặc nhấn vào icon có chữ D mầu đỏ trong cái kính lúp trên thanh công cụ.


ThinhiemveLED - µVision3 - [C:\Documents and Settings\LoLem-Duyen\Desktop\Examples-M Truong\ThinhiemLED.c]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Start/Stop Debug Session
Target 1
Project Workspace
Target1
Source Group 1
ThinhiemLED.c
#include <REGX51.H> /*Trong File cua thu vien*/
#define bat 1 /*Dinh nghia gia tri bat den LED*/
#define tat 0 /* Dinh Nghia gia tri tat Led*/
/* Khai bao bien toan cuc*/
sbit led1 = P1^0;
sbit led2 = P1^1;
sbit led3 = P1^2;
sbit led4 = P1^3;
sbit led5 = P1^4;
sbit led6 = P1^5;
sbit led7 = P1^6;
sbit led8 = P1^7;
/* Khai bao ham */
/*--------------------ham Tre--*/
/*ham tao thoi tre*/
void tre (long time)
{ long n ; /*khai bao bien cuc bo*/
for (n=0; n++;n < time)
;
; /*choi khong lam gi ca*/
;
}
ThinhiemL...
Build target 'Target 1'
compiling ThinhiemLED.c...
linking...
Program Size: data=17.0 xdata=0 code=96
"ThinhiemveLED" - 0 Error(s), 0 Warning(s).
Output Window
Build
Command
Find in Files
Start/Stop Debug Session
L:23 C:19
R/W
start
xoan giao trinh
giao trinh ga...
giactrinh305...
Huong dan su...
ThinhiemveL...
Windows Vec...
1:41 PM


ThinhiemveLED - µVision3 - [C:\Documents and Settings\LoLem-Duyen\Desktop\Examples-M Truong\ThinhiemLED.c]
File Edit View Project Debug Flash Peripheras Tools SVCS Window Help
}
/***/
/* Ham main */
void main (void)
{
while(1) //lap vo han
{
led1= bat;//bat den
tre(1000);// tre
led1=tat;//tat den
tre(1000);//tre 1 khoang thoi gian
}
}
/***/
ThinhiemL...
Load "C:\\Documents and Settings\\LoLem-Duyen\\Desktop\\Examples-M Truong\\ThinhiemveLED"
ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE DEFINE DIR Display
Output Window
Build Command Find in Files
Ready
L:47 C:1
R/W
start
xoan giao trinh
giao trinh ga...
giactrinh305...
Huong dan su...
ThinhiemveL...
Windows Vec...
1:42 PM



Trong menu Peripherals(các thiết bị ngoại vi) chọn IO port , Port 1. Được như sau:



Các bạn thấy 1 cửa sổ nhỏ Parallel Port 1 xuất hiện đó là cái mô phỏng cho cổng 1 của AT89C51. Dấu tích tương đương chân ở mức cao(5V) , không tích chân ở mức thấp (0V). Trong menu peripherals còn các ngoại vi khác như timer , interrupt, serial.

Để chạy chương trình các bạn nhấp chuột phải vào màn hình soạn thảo.

Rồi ấn F11. Mỗi lần ấn sẽ chạy 1 lệnh. Khi debug nếu các bạn chờ hàm delay lâu quá 1000 lần lặp . Các bạn nhấn Ctrl + F11 để bỏ qua hàm.

Hoặc ấn F10 để chạy từng dòng lệnh. Các bạn sẽ thấy chân P1_0 thay đổi giá trị.

Bảng bên trái , ô Project workspace bây giờ có các thanh ghi. Các bạn có thể thấy chúng thay đổi. Nhưng các bạn không cần quan tâm đến các thanh ghi này. Vì mình học ngôn ngữ C mà. Nếu học asembly thì mới phải sử dụng chúng. Cũng mệt đấy. Cái bạn quan tâm nhất là cái sec. Nó cũng thay đổi. Vì thạch anh là 12Mhz, nên mỗi chu kì máy là 10^{-6} giây. Các bạn căn cứ vào đấy để biết lệnh nào mất bao nhiêu chu kì máy, làm thời gian thực thì cần lắm đấy. Thoát khỏi debug lại ấn Ctrl+F5 hoặc ấn vào icon debug.

Đây chỉ là một chương trình nhỏ để chúng ta tập làm quen với ngôn ngữ C trên giao diện của trình biên dịch KIEL uVesion 3

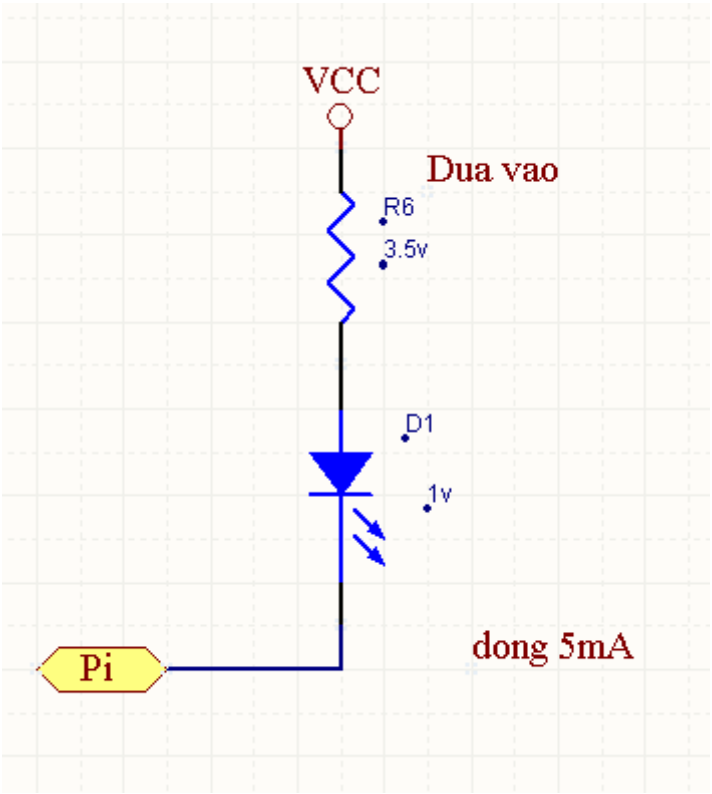
.

Bài 3: Thao tác với cổng vào ra của vi điều khiển

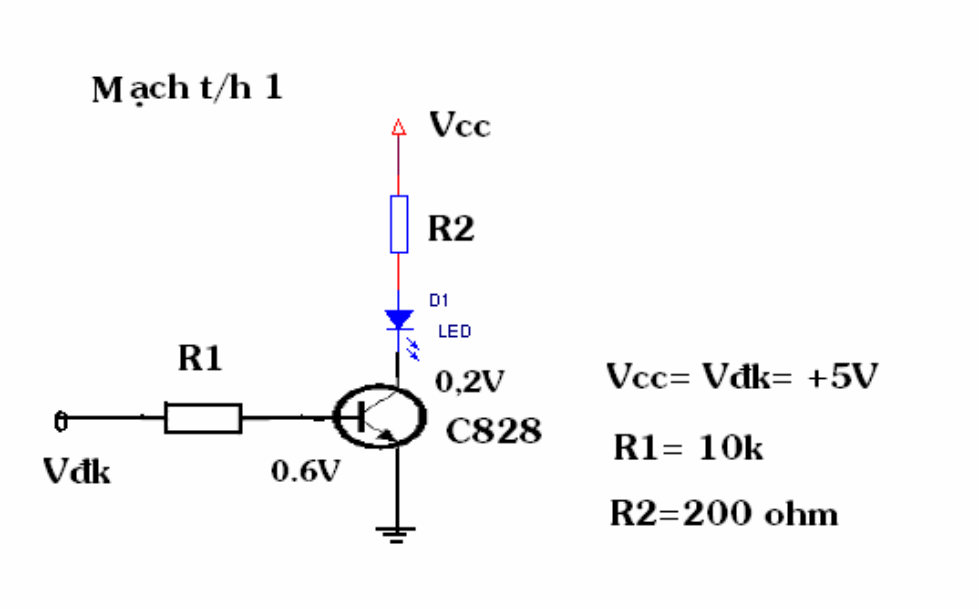
Bài toán ứng dụng ghép nối vđk với led đơn, led 7 thanh, thao tác phím đơn...

Kỹ năng ghép nối led <Light Emiting Diode>

+ Đối với các loại Led công suất nhỏ chúng ta có thể sử dụng trực tiếp chân vào ra của vi điều khiển để điều khiển chúng như sau :



+ Còn đối với Led có công suất lớn; thì chúng ta cần có thêm transitor để nuốt dòng hộ:

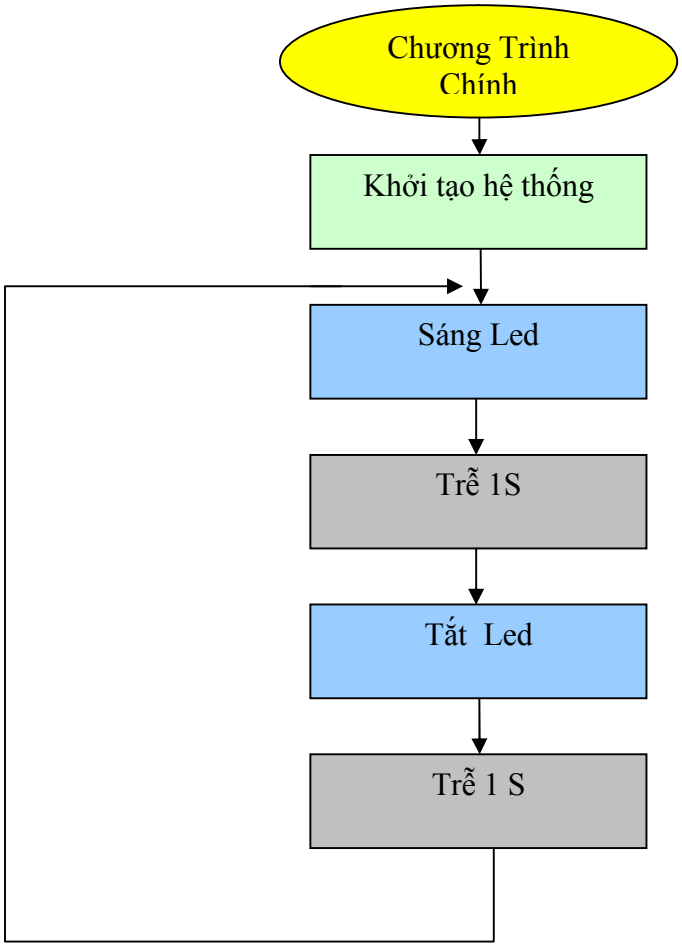


<u>Bài toán 1: Bài toán ghép nối vi điều khiển với các led đơn</u>

Thông qua việc thực hiện bài toán này các bạn sẽ học được cách ghép nối vi điều khiển với led đơn và thao tác lập trình tạo các hiệu ứng như bật tắt led, hay bật tắt led có một quy luật nhất định sẽ giúp các bạn thực hành các lệnh C lập trình và tạo một tư duy giải thuật tốt.

Trong kit có 8 led đơn được ghép nối với 8 chân của cổng P1 (cực Katot nối với chân vđk – nghĩa là khi bạn đưa mức logic 0 ra cổng thì led sẽ sáng).

+ Sơ đồ thuật toán của bài nháy led:



Ví dụ sau sử dụng các mẫu hàm đơn giản tạo một chương trình tạo trễ và làm cho một led1(theo định nghĩa sbit) nhấp nháy theo một thời gian định trước.

```
/*******************Bo tien xu li*********************************/
#include <AT89x51.h>// Dinh kem file thu vien
#define bat 1          // Dinh nghia gia tri bat den led
#define tat 0// Dinh nghia gia tri tat den led
/*****************************************************************/

/*******************Khai bao bien toan cuc************************/
sbit Led1=P1^0; //Khai bao bien Led1 kieu bit chan P1_0
```

```
sbit Led2=P1^1; // ...
sbit Led3=P1^2;
sbit Led4=P1^3;
sbit Led5=P1^4;
sbit Led6=P1^5;
sbit Led7=P1^6;
sbit Led8=P1^7;//Khai bao bien Led8 kieu bit chan P1_7
/*********************Khai bao ham*****************************/
/*----------------Ham tre------------------------------
                        Ham tao thoi gian tre.
                        Dau vao: 1 bien thoi gian.
                        Dau ra:        khong
-----------------------------------------------------*/
void delay (long time)
{
        long n;// Khai bao bien cuc bo
        for(n=0; n<time; n++)//Lap time lan
        {
                ; // Khong lam gi nop
        }
}
/*******************Ham chinh**************************/
void main(void)
{
        while(1)// Lap vo han
        {
                Led1= bat;// Bat led 1
                delay(1000);// Tre 1 khoang thoi gian
                Led1= tat;// Tat led 1
                delay(1000);// Tre 1 khoang thoi gian
        }

}
//======>>>>>====== Doan chuong trinh tao hieu ung nhap nhay voi cac led
====//
//-------------------Chuong trinh chinh-----------------//
void main2(void)
{
 while(1)
  {
 // P1 =0x00 ;
  led1 =bat;
  delay(1000);
  led1 =tat;
  led2 =bat;
```

```
  delay(1000);
  led2 =tat;
  led3 =bat;
  delay(1000);
  led3 =tat;
  led4 =bat;
  delay(1000);
  led4 =tat;
  led5 =bat;
  delay(1000);
  led5 =tat;
  led6 =bat;
  delay(1000);
  led6 =tat;
  led7 =bat;
  delay(1000);
  led7 =tat;
  led8 =bat;
  delay(1000);
  led8 =tat;
  delay(1000);
   }
}
```

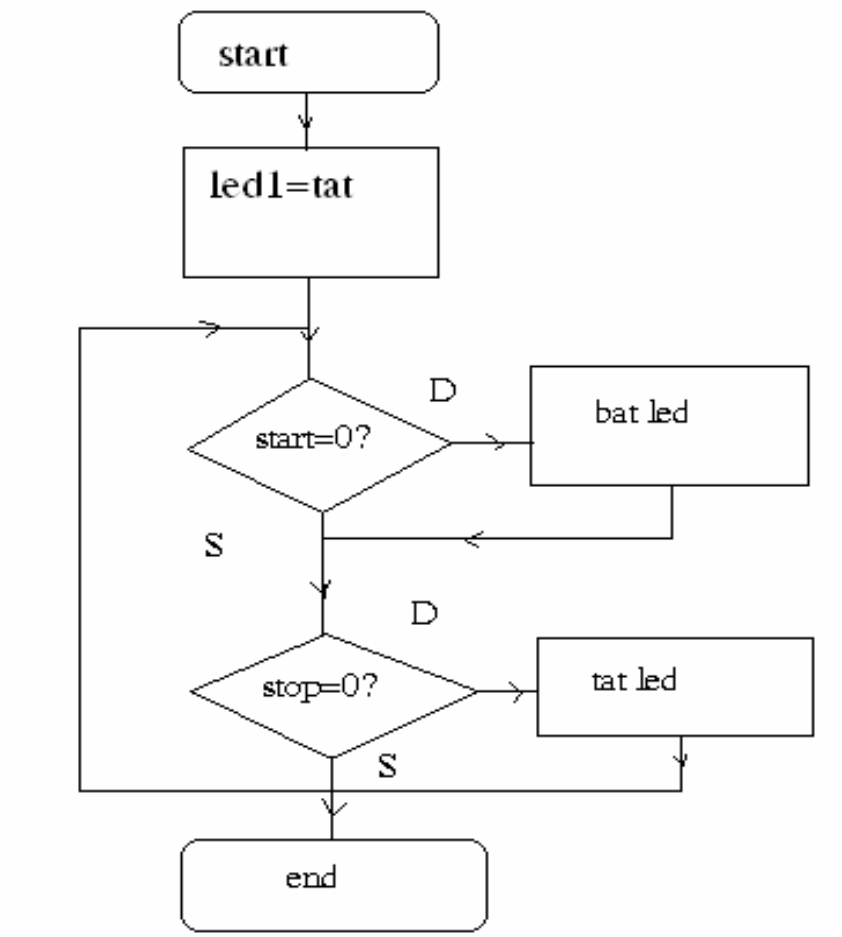
Bài toán 2 : Phối hợp led-công tắc(1)

Với các dữ liệu ban đầu như sau : Led1 –P1.0; led 2—p1.1; led 3—p1.2; led 4—p1.3; led 5—p1.4; led 6—p1.5; led7—p1.6; led 8—p1.7;
Các công tắc tích cực mức 0 được nối với các cổng P3.
Ctac1- p3.0 ... ctac8—p3.7

Ta có bài toán 2 như sau: với led 1(p1.0) và ctac1(p3.0), ctac2 (p3.1)
Ban đầu led1 tắt, khi nhấn ctac1 led 1 sáng cho tới khi nhấn ctac2 thì led 1 mới tắt, cứ như vậy.

*Các bạn tự xây dựng giải thuật và chương trình, sử dụng chương trình biên dịch và mô phỏng **Keil C** để quan sát kết quả, sau cùng các bạn tạo ra file.hex và nạp xuống kit thí nghiệm. Quan sát trực tiếp kết quả trên đối tượng thật.*

Sau đây là một cách giải quyết bài toán: (các bạn tham khảo)



Code for ex2.1:

```
/*******************Bo tien xu li*********************************/
#include <AT89x51.h>// Dinh kem file thu vien
#define bat 0          // Dinh nghia gia tri bat den led
#define tat 1// Dinh nghia gia tri tat den led
/*****************************************************************/

/*******************Khai bao bien toan cuc************************/
sbit led1=P1^0;
sbit start = P3^0;  // cong tac start de bat led
sbit stop  = P3^1;  // cong tac stop de tat led.
/*******************Ham chinh*************************************/
void main(void)
{
   led1 = tat; // ban dau led tat;
 while(1)
  {
   if (( start==0)&& ( stop==1))
```

```
        {  led1=bat;}
   if (( start==1)&& ( stop==0))
        {  led1=tat;}
  }
}
```

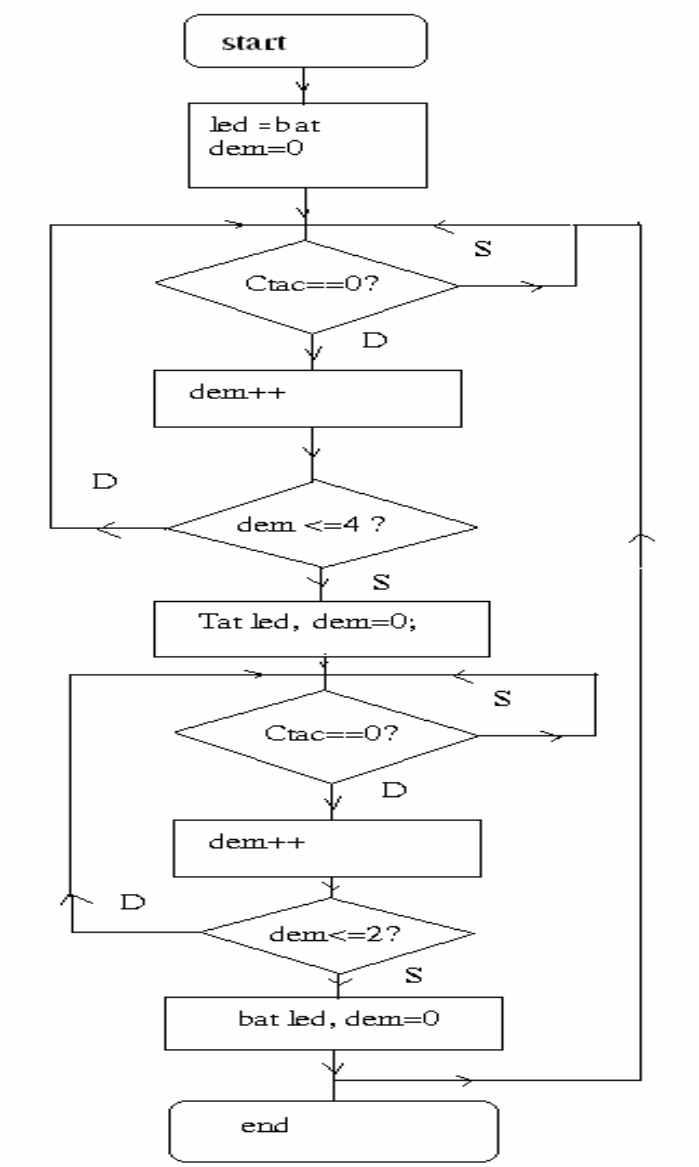
Bài Toán 3:Phối hợp led-công tắc(2)

Yêu cầu của bài toán như sau:

Cũng với một led đơn và một công tắc. Các bạn hãy lập trình như sau : ban đầu led sáng, khi bạn nhấn công tắc 5 lần thì led tắt, và bạn nhấn tiếp 3 lần nữa thì led tắt. Quá trình này được lặp di lặp lại như vậy.

Các bạn hãy tiến hành lập trình sử dụng phần mềm công cụ mô phỏng cùng kit để kiểm chứng chương trình của mình.

Sau đây là một ví dụ (code) cho bài toán này:



Code for ex 2.3:

```
/*******************Bo tien xu li*********************************/
#include <AT89x51.h>// Dinh kem file thu vien
#define bat 0          // Dinh nghia gia tri bat den led
#define tat 1// Dinh nghia gia tri tat den led
/*****************************************************************/

/*******************Khai bao bien toan cuc************************/
sbit led1=P1^0;
sbit ctac = P3^7;  // cong tac start de bat led

void delay(long time)
{
```

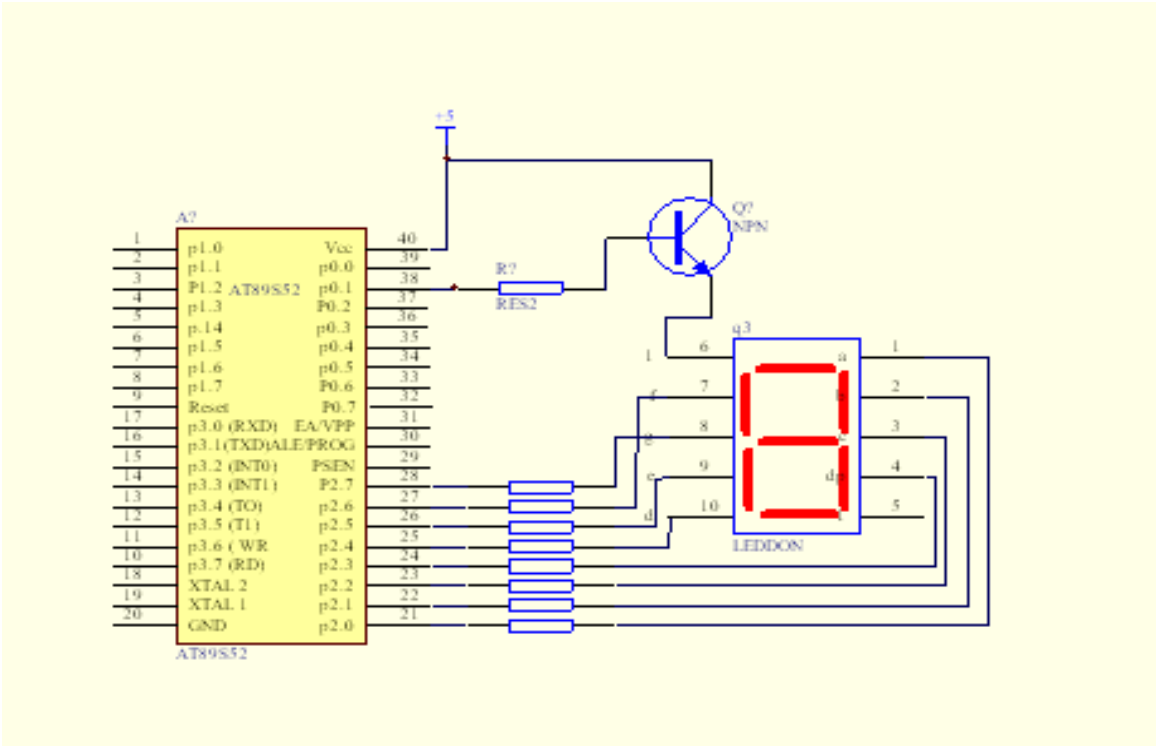
```
  long n1; // khai bao bien cuc bo.
  for (n1=0;n1++;n1<time) // vong lap time lan.
         {
           ;// khong thuc hien gi ca !!!
         }
}
//-------------------Chuong trinh con test phim----------//
unsigned char dem=0;
void phim_an(void)
{
                if (ctac==0)// co phim an
        {
                  while (ctac==0)
                   {
                    delay(15000);
                   }

        dem ++; // tang bien dem
         }
 }
/*******************Ham chinh**************************/


void main(void)
{
   led1 = bat; // ban dau led tat;
        dem =0; //  khoi tao gia tri ban dau
 while(1)
  {
   while(( dem <5) &&( dem >=0))
         {
         phim_an();
         }
          led1=tat;
          dem=0;
         while(( dem <2)&&( dem >=0))
         {
         phim_an();
         }
          led1=bat;
          dem=0;

  }
}
```

Bài toán 4:

Mục đích của bài toán này là giúp các bạn hiểu được cách ghép nối vđk với một bộ hiện thị hết sức thông dụng đó là led 7 thanh.
Học được cách lập trình hiển thị đơn giản:

Đối với đối tượng led 7 thanh các bạn cần làm quen với các dạng của nó, có loại chỉ một led 7 thanh (dạng đơn thường 10 chân với 8 chân dữ liệu và 2 chân A chung hoặc K chung) , có loại led 7 thanh chứa 2 con trong một vỏ và cũng đưa ra 10 chân trong đó có 8 chân dữ liệu và 2 chân còn lại là hai chân (A chung hay K chung) của hai led này, cũng có loại 4 led trong một vỏ và đưa ra 12 chân với 8 chân dữ liệu (cho các led ở các thanh) và 4 chân còn lại là 4 chân chung của 4 led này.

Các bạn cần nắm được thứ tự các chân của chúng từ đó mới có thiết kế chúng với vđk được. Thông thường led 7 đoạn được ghép nối với vđk như hình vẽ dưới:



Như vậy với mỗi cách thiết kế mạch ta có được đường dữ liệu và đường điều khiển khác nhau.
Đường điều khiển chính là điều khiển đóng mở các tran để đưa mức điện áp vào chân chung của các led.
Đường dữ liệu là đường nối với các chân tương ứng với các thanh của led.

Từ cách ghép nối này ta mới tính ra một bảng gọi là ***mã 7 thanh*** ví dụ như ở bảng dưới:

Giả sử ta nối led 7 thanh đơn với vdk như hình vẽ: tức chân điều khiển là chân p0.1

a—p2.0 c---p2.2 , d --- p2.4 , e----p2.5 , f -----p2.6 , g---p2.7

b—p2.1 , dp ----p2.3

Và ta có bài toán nhỏ như sau: Bạn hãy lập trình để đưa ra số 9 ra led 1 và led 3 còn 2 led còn lại tắt.

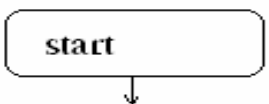
Code for ex:

Các số hiển thị	P2.7 g	P2.6 f	P2.5 e	P2.4 d	P2.3 dp	P2.2 c	P2.1 b	P2.0 a	Số nạp hex Mov P2,#
0	1	0	0	0	1	0	0	0	088
1	1	1	1	1	1	1	0	0	0fc
2	0	1	0	0	1	1	0	0	04c
3	0	1	1	0	1	0	0	0	068
4	0	0	1	1	1	0	0	1	039
5	0	0	1	0	1	0	1	0	02a
6	0	0	0	0	1	0	1	0	00a
7	1	1	1	1	1	0	0	0	0f8
8	0	0	0	0	1	0	0	0	008
9	0	0	0	1	1	0	0	0	018

```
/*******************Bo tien xu li*********************************/
#include <AT89x51.h>// Dinh kem file thu vien
/* --- dinh  nghia cac bit dung------*/
 sbit  led1 = P0^3;
 sbit  led2 = P0^2;
 sbit  led3 = P0^1;
 sbit  led4 = P0^0;
 // duong du lieu la P1
 // voi ma 7 thanh cho cac chu so 0 --9 tuong ung la
 // 0c0h,0f9h,0a4h,0b0h,099h,092h,082h,0f8h,080h,090h
// chuong trinh chinh.
  void main ( void )
  {
    P1= 0x90; // dua so 9 ra duong du lieu
                    led1= led3= 1; // bat led 1 va led 3
                    led2= led4= 0; // tat led 2 va led 4
   while(1) {;}
  }
```

Bài toán tiếp thiết lập vdk thành một bộ đếm số lần ấn phím 0-9-0 với hiển thị 7thanh trên led số 1. (trong phạm vi kiến thức ta làm bộ đếm từ 0-9-0 sau này ta sẽ xét bài toán tạo một bộ đếm với dải rộng hơn)

Các bạn tự lập trình chú ý rằng trong ví dụ đếm phím ấn đã được đề cập trong các bài học trước thì số lần ấn phím lưu trong biến đếm là số thập phân không thể đưa ra hiển thị trên led được mà phải đưa sang mã led 7 thanh do đó



các bạn phải viết chương trình con chuyển đổi này.

Chương trình hiển thị ra led 7 thanh tốt nhất các bạn nên viết thành một chương trình con sử dụng cấu trúc switch ()_case.
Và chương trình kiểm tra công tắc cũng nên xây dựng thành một chương trình con.
Các bạn hãy xây dựng chương trình của riêng mình !!

Code for ex:

```
/*******************Bo tien xu li*********************************/
#include <AT89x51.h>// Dinh kem file thu vien
/* --- dinh  nghia cac bit dung------*/
 sbit  led1 = P0^3;
 sbit  led2 = P0^2;
 sbit  led3 = P0^1;
 sbit  led4 = P0^0;
 sbit  ctac = P3^7;
 // duong du lieu la P1
 // voi ma 7 thanh cho cac chu so 0 --9 tuong ung la
 // 0c0h,0f9h,0a4h,0b0h,099h,092h,082h,0f8h,080h,090h
 void delay(long time)
{
  long n1; // khai bao bien cuc bo.
  for (n1=0;n1++;n1<time) // vong lap time lan.
                      {
                        ;// khong thuc hien gi ca !!!
                      }
}

//-------------------Chuong trinh con test phim----------//
unsigned char dem=0;
unsigned char phim_an(void)
```

```
{
                        if (ctac==0)// co phim an
                    {
                        delay(10000); // chong rung phim.
                        while (ctac==0)
                         {
                          ; // doi cho toi khi nha phim an.
                         }
                         delay(20000); // chong rung phim.
        dem ++; // tang bien dem
                        if (dem==10) dem=0;
                     }
                    return dem;
}
//------------------ Hien thi so lan an phim-------------//
void solan_an(unsigned char solan)
{
  switch (solan)
   {
   case 0: {P1=0xc0;break;}
   case 1: {P1=0xf9;break;}
   case 2: {P1=0xa4;break;}
   case 3: {P1=0xb0;break;}
   case 4: {P1=0x99;break;}
   case 5: {P1=0x92;break;}
   case 6: {P1=0x82;break;}
   case 7: {P1=0xf8;break;}
   case 8: {P1=0x80;break;}
   case 9: {P1=0x90;break;}
    }
}
//-------------------Chuong trinh chinh------------------//
void main(void)
{
P1=0xc0; // ban dau la so 0.
led1= led3= led2=0 ;
led4=1; // bat led 4
while(1)
 {
   phim_an();
   solan_an(dem);
  }
   }
```

Bài 4: Ghép nối và thao tác với màn hình LCD

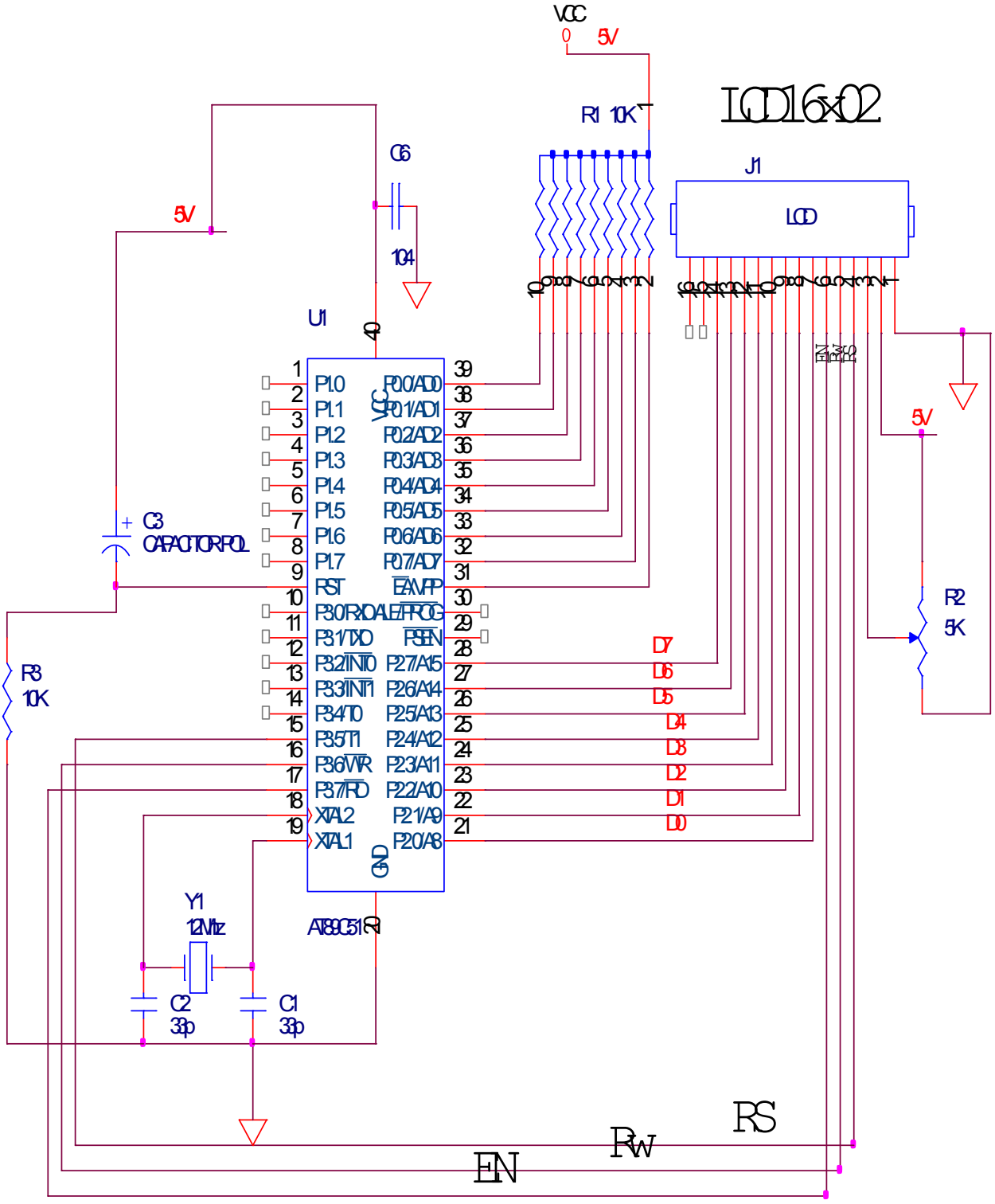
Trong thực tế có rất nhiều loại LCD có kích thước khác nhau (mở rộng theo hàng và cột). Ta đang nghiên cứu chủ yếu hai loại LCD đơn giản là LCD 16x1 và 16x2. Chúng có 16 chân chức năng và ta sẽ lần lượt nghiên cứu chức năng từng chân:

Chân	Ký hiệu	I/O	Mô tả
1	V_{SS}	-	Đất
2	V_{CC}	-	Dương nguồn 5v
3	V_{EE}	-	Cấp nguồn điều khiển phản
4	RS	I	RS = 0 chọn thanh ghi lệnh. RS = 1 chọn thanh dữ liệu
5	R/W	I	R/W = 1 đọc dữ liệu. R/W = 0 ghi
6	E	I/O	Cho phép
7	DB0	I/O	Các bít dữ liệu
8	DB1	I/O	Các bít dữ liệu
9	DB2	I/O	Các bít dữ liệu
10	DB3	I/O	Các bít dữ liệu
11	DB4	I/O	Các bít dữ liệu
12	DB5	I/O	Các bít dữ liệu
13	DB6	I/O	Các bít dữ liệu
14	DB7	I/O	Các bít dữ liệu

Chân 15 và chân 16: ghi là A và K. Nó là anốt và katốt của 1 con led dùng để sáng LCD trong bóng tối. Chúng ta không sử dụng. Nếu các bạn muốn dùng thì nối chân A qua 1 điện trở từ 1K đến 5K lên dương 5V, chân K xuống đất đèn sẽ sáng.

4.1. Lắp mạch theo sơ đồ sau:

Trên ta có cách ghép nối LCD với vi điều khiển như hình vẽ dưới:



- Hướng dẫn khi thực hành trên board trắng: Cắm luôn 8 bit dữ liệu của LCD từ D0 đến D7 vào cổng 2 của 8051 mà không cần câu dây. Để thừa 6 chân ra ngoài là: EN,RW,RS,Ve, Vcc, GND ra ngoài. Rồi dùng dây để câu chân 1 xuống GND, chân 2 lên +5V, chân 3 vào chân giữa của biến trở tinh 5K, 2 chân còn lại của biến trở tinh 1 chân câu lên +5V,1 chân câu xuống 0V.

4.2. Nguyên lí hoạt động của LCD:

- Chân V_{CC}, V_{SS} và V_{EE}: Các chân V_{CC}, V_{SS} và V_{EE}: Cấp dương nguồn - 5v và đất tương ứng thì V_{EE} được dùng để điều khiển độ tương phản của LCD.

- Chân chọn thanh ghi RS (Register Select): Có hai thanh ghi trong LCD, chân RS(Register Select) được dùng để chọn thanh ghi, như sau: Nếu RS = 0 thì thanh ghi mà lệnh được chọn để cho phép người dùng gửi một lệnh chẳng hạn như xoá màn hình, đưa con trỏ về đầu dòng v.v… Nếu RS = 1 thì thanh ghi dữ liệu được chọn cho phép người dùng gửi dữ liệu cần hiển thị trên LCD.

- Chân đọc/ ghi (R/W): Đầu vào đọc/ ghi cho phép người dùng ghi thông tin lên LCD khi R/W = 0 hoặc đọc thông tin từ nó khi R/W = 1.

- Chân cho phép E (Enable): Chân cho phép E được sử dụng bởi LCD để chốt dữ liệu của nó. Khi dữ liệu được cấp đến chân dữ liệu thì một xung mức cao xuống thấp phải được áp đến chân này để LCD chốt dữ liệu trên các chân dữ liêu. Xung này phải rộng tối thiểu là 450ns.

- Chân D0 - D7: Đây là 8 chân dữ liệu 8 bít, được dùng để gửi thông tin lên LCD hoặc đọc nội dung của các thanh ghi trong LCD. Để hiển thị các chữ cái và các con số, chúng ta gửi các mã ASCII của các chữ cái từ A đến Z, a đến f và các con số từ 0 - 9 đến các chân này khi bật RS = 1.

Cũng có các mã lệnh mà có thể được gửi đến LCD để xoá màn hình hoặc đưa con trỏ về đầu dòng hoặc nhấp nháy con trỏ.

- Chú ý: Chúng ta cũng sử dụng RS = 0 để kiểm tra bít cờ bận để xem LCD có sẵn sàng nhân thông tin. Cờ bận là bít D7 và có thể được đọc khi R/W = 1 và RS = 0 như sau:

Nếu R/W = 1, RS = 0 khi D7 = 1 (cờ bận 1) thì LCD bận bởi các công việc bên trong và sẽ không nhận bất kỳ thông tin mới nào. Khi D7 = 0 thì LCD sẵn sàng nhận thông tin mới. Lưu ý chúng ta nên kiểm tra cờ bận trước khi ghi bất kỳ dữ liệu nào lên LCD.

- Sau đây là bảng mã lệnh của LCD:

Mã (Hex)	**Lệnh đến thanh ghi của LCD**
1	Xoá màn hình hiển thị
2	Trở về đầu dòng
4	Giảm con trỏ (dịch con trỏ sang trái)
6	Tăng con trỏ (dịch con trỏ sang phải)
5	Dịch hiển thị sang phải
7	Dịch hiển thị sang trái
8	Tắt con trỏ, tắt hiển thị
A	Tắt hiển thị, bật con trỏ
C	Bật hiển thị, tắt con trỏ
E	Bật hiển thị, nhấp nháy con trỏ
F	Tắt con trỏ, nhấp nháy con trỏ
10	Dịch vị trí con trỏ sang trái
14	Dịch vị trí con trỏ sang phải
18	Dịch toàn bộ hiển thị sang trái
1C	Dịch toàn bộ hiển thị sang phải
80	Ép con trỏ về đầu dòng thứ nhất
C0	Ép con trỏ về đầu dòng thứ hai
38	Hai dòng và ma trận 5 × 7

- Điều khiển LCD qua các bước sau:

Bước 0 : Chuẩn bị phần cứng. Dùng tuốc vít hay cái gì bạn có xoay biến trở 5 K điều chỉnh độ tương phản của LCD. Xoay cho đến khi các ô vuông(các điểm ảnh) của LCD hiện lên thì xoay ngược biến trở lại 1 chút.

Bước 1 : Khởi tạo cho LCD.

Bước 2 : Gán các giá trị cho các bit điều khiển các chân RS,RW,EN cho phù hợp với các chế độ : Hiển thị kí tự lên LCD hay Thực hiện 1 lệnh của LCD.

Bước 3: Xuất byte dữ liệu ra cổng điều khiển 8 bit dữ liệu của LCD.

Bước 4: Kiểm tra cờ bận xem LCD sẵn sàng nhận dữ liệu mới chưa.

Bước 5: Quay vòng lại bước 1.

4.3. Lập trình:

- Để có thể lập trình cho LCD ta thêm vào thư viện **string.h** của trình biên dịch bằng câu lệnh:

```
#include <string.h>
- Khai báo các chân của LCD gắn với các cổng:
/*
RS chon thanh ghi
        =0 ghi lenh
        =1 ghi du lieu
RW doc ghi
        =0 ghi
        =1 doc
E cho phep chot du lieu
        xung cao xuong thap toi thieu 450 ns.
Bit co ban D7
        khi RS=0 RW=1 neu D7=1 LCD ban
                                D7=0 LCD san sang.
*/
sfr LCDdata = 0xA0;// Cong 2 , 8 bit du lieu  P0 co dia chi 0x80, P1 0x90 , P2 0xA0
sbit BF = 0xA7; // Co ban bit 7
sbit RS = P3^5;
sbit RW = P3^4;
sbit EN = P3^3;
```

- Viết 1 số hàm điều khiển LCD như sau:

```
* Hàm kiểm tra LCD có bận hay không:
void wait(void)
{
        long n = 0;
        EN=1;// Dua chan cho fep len cao
        RS=0;// Chon thanh ghi lenh
        RW=1;//      Doc tu LCD
        LCDdata=0xFF;// Gia tri 0xFF
```

```
                while(BF){n++; if(n>100) break;}// Kiem tra co ban
                // Neu ban dem n den 100 roi thoat khoi while
                EN=0;// Dua xung cao xuong thap de chot
                RW=0;// Doc tu LCD
}
* Hàm điều khiển LCD thực hiện 1 lệnh:
void LCDcontrol(unsigned char x)
{
                EN=1;// Dua chan cho phep len cao
                RS=0;// Chon thanh ghi lenh
                RW=0;// Ghi len LCD
                LCDdata=x;// Gia tri x
                EN=0;// Xung cao xuong thap
                wait();// Doi LCD san sang
}
```

Hàm có 1 biến đầu vào là các giá trị trong bảng mã lệnh của LCD.

```
* Hàm khởi tạo LCD:
void LCDinit(void)
{
                LCDcontrol(0x30);//Che do 8 bit.
                LCDcontrol(0x38);// 2 dong va ma tran 5x7
                LCDcontrol(0x0C);// Bat con tro
                LCDcontrol(0x06);// Tang con tro xang fai
                LCDcontrol(0x01);// Xoa man hinh
}
* Hàm lệnh cho LCD hiển thị 1 kí tự:
void LCDwrite(unsigned char c)
{
                EN=1;// Cho fep muc cao
                RS=1;// Ghi du lieu
                RW=0;// Ghi len LCD
                LCDdata=c;// Gia tri C
                EN=0;// Xung cao xuong thap
                wait();// Cho
}
```

Bài toán 1: Các bạn hãy viết chương trình tạo trên LCD hiển thị dòng chữ 'DKS.co., ' và nhấp nháy hai led nối với P1.3 và P1.4

Code for ex:

```
#include <AT89x51.h>
#define       tat     1     /* muc logic 1 */
#define       bat     0     /* muc logic 0 */
```

```
#include <string.h>
sbit led = P1^4;


//-------------------------------------------------------------
sfr LCDdata = 0xA0;
sbit BF = 0xA7; // Co ban bit 7
sbit RS = P3^7;
sbit RW = P3^6;
sbit EN = P3^5;
/*---------Cac chuong trinh con cua lcd ------*/
//---------------Chuong trinh con kiem tra su san sang cua lcd----------------;
void wait(void)
{
		long n = 0;
		EN=1;// Dua chan cho fep len cao
		RS=0;// Chon thanh ghi lenh
		RW=1;//	Doc tu LCD
		LCDdata=0xff;// Gia tri 0xFF
		while(BF){n++; if(n>110) break;}// Kiem tra co ban
		// Neu ban dem n den 100 roi thoat khoi while
		EN=0;// Dua xung cao xuong thap de chot
		RW=0;// Doc tu LCD
}
//------------------Chuong trinh con thiet lap lenh cho LCD----------------------;
void LCDcontrol(unsigned char x)
{
		EN=1;// Dua chan cho fep len cao
		RS=0;// Chon thanh ghi lenh
		RW=0;// Ghi len LCD
		LCDdata=x;// Gia tri x
		EN=0;// Xung cao xuong thap
		wait();// Doi LCD san sang
}
//---------------chuong trinh con thiet lap mot so thong so cua lcd---------------;
void LCDinit(void)
{
		LCDcontrol(0x38);// 2 dong va ma tran 5x7
		LCDcontrol(0xc0);
		LCDcontrol(0x0f);// Bat con tro
		LCDcontrol(0x01);// Xoa man hinh
		LCDcontrol(0x82);
}
//------------------Chuong trinh con thiet lap dulieu cho LCD----------------------;
void LCDwrite(unsigned char c)
{
```

```
                EN=1;// Cho fep muc cao
                RS=1;// Ghi du lieu
                RW=0;// Ghi len LCD
                LCDdata=c;// Gia tri C
                EN=0;// Xung cao xuong thap
                wait();// Cho
}

/*================ Chuong trinh con tre thoi gian ================*/
void delay(long time)                // chuong trinh tre thoi gian.
{
  long n1;                           // khai bao bien cuc bo.
  for (n1=0;n1<time;n1++)    // vong lap time lan.
        {
          ;                                  // khong thuc hien gi ca !!!
        }
}

/* ====================== Chuon trinh chinh
====================*/
 //* Hàm main:
void main(void)
{

        LCDinit();

//      LCDcontrol(0x82);
        LCDwrite('D');
        LCDwrite('K');
        LCDwrite('S');
        LCDwrite('.');
        LCDwrite('C');
        LCDwrite('o');
        LCDwrite(',');
        LCDwrite('.');


        while(1)// Lap vo han
        {
                led= bat;// Bat led 1
                P1_3=tat;
                delay(3000);// Tre 1 khoang thoi gian
                led= tat;// Tat led 1
                P1_3=bat;
                delay(2000);// Tre 1 khoang thoi gian
        }
```

```
}
```

Bài 5: Ngắt của vi điều khiển, thao tác với ngắt ngoài của vi điều khiển

Bài toán ứng dụng: mạch bảo vệ sự cố (quá áp, thấp áp, quá nhiệt, ngắn mạch, báo cháy,...)

5.1. Khái niệm:

Ngắt là sự đáp ứng các sự kiện bên trong hoặc bên ngoài nhằm thông báo cho bộ vi điều khiển biết thiết bị đang cần được phục vụ.

Ví dụ về ngắt: Ta xem xét kĩ ví dụ sau để thấy được khái niệm ngắt.
Lớp đang học bài, mất điện lớp nghỉ học và chuyển sang trò chuyện, có điện lớp lại tiếp tục học bài học từ điểm mà đã bị dừng.

Trong ứng dụng của vi điều khiển mà có sử dụng ngắt thì bao gồm các vấn đề sau: Chương trình chính, chương trình phục vụ ngắt, và các tác động ngắt. Trong ví dụ trên **lớp học** là chương trình chính, chương trình phục vụ ngắt là sự trò chuyện khi lớp không học, còn tác động ngắt ở đây chính là **mất điện.**

5.2. Trình tự thực hiện ngắt của vi điều khiển.

Khi một ngắt được kích hoạt, trình tự thực hiện của bộ vi điều khiển như sau:
B1: Kết thúc lệnh hiện tại trong chương trình chính của vi điều khiển và lưu địa chỉ của lệnh kế tiếp vào ngăn xếp.
B2: Lưu lại trạng thái hiện hành của tất cả các ngắt vào bên trong.
B3: Nhảy tới một vị trí cố định trong bộ nhớ gọi là bảng vectơ ngắt, nơi lưu địa chỉ của trình phục vụ ngắt.
B4: Nhận địa chỉ từ bảng vectơ ngắt rồi nhảy tới địa chỉ đó và bắt đầu thực hiện trình phục vụ ngắt cho tới lệnh cuối cùng.
B5: kết thúc trình phục vụ ngắt và vđk trở về đúng vị trí mà nó bị ngắt lúc trước sau nạp địa chỉ lệnh cần làm kế tiếp từ ngăn xếp và thực hiện lệnh này.

5.2.1.Các ngắt của vi điều khiển 8051

Vđk 8051 có 5 ngắt bao gồm có: 2 ngắt ngoài (INT0, INT1); 2 ngắt định thời (T0, T1) ; và một ngắt truyền thông nối tiếp (RS232).
Bảng vectơ ngắt :

Ngắt	Cờ	Địa chỉ vectơ	*Thứ tự ngắt*
Ngắt ngoài 0	**IE0**	**0003H**	**0**
Bộ định thời 0	**TF0**	**000BH**	**1**
Ngắt ngoài 1	**IE1**	**0013H**	**2**
Bộ định thời 1	**TF1**	**001BH**	**3**
Port nối tiếp	**RI hoặc TI**	**0023H**	**4**
Bộ định thời 2	**TF2 hoặc EXF2**	**002BH**	**5**

Cho phép ngắt và cấm ngắt: (IE – interrupt enable)

EA	--	ET2	ES	ET1	EX1	ET0	EX0

EA (IE.7 bit) là bit cho phép ngắt toàn bộ. Tức là nếu bạn sử dụng một trong 5 ngắt kể trên của vđk thì bạn phải **set** bit này lên với giá trị 1. Ngược lại khi bit EA =0 thì toàn bộ các ngắt của vđk sẽ bị cấm.
ET2 (IE.5 bit) là bit cho phép ngắt của timer 2 (chỉ ở họ 52). Khi bit này = 1 thì ngắt định thời 2 sẽ được cho phép và ngược lại.
ET1, ET0 (IE.3, IE.1 bit) là bit cho phép ngắt định thời 1, 0 của vđk 8051. Khi các bit này =1 thì ngắt của hai bộ định thời này được cho phép và ngược lại.
ES (IE.4 bit) là bit cho phép ngắt truyền thông nối tiếp.
EX1, EX0 (IE.0 và IE.2 bit) là hai bit cho phép của hai ngắt ngoài của vđk.

5.2.2. Ngắt ngoài và cách lập trình

a) Khái niệm:

Bộ vđk 8051 có hai ngắt ngoài là INT0, INT1 với hai chân tác động đầu vào tương ứng là P3.2, P3.3.
Ngắt ngoài là ngắt của vđk mà tác động ngắt ở đây chính là các tác động bên ngoài của vđk tác động vào vđk thông qua các chân ngắt dưới dạng một tín hiệu điện áp dạng xung.
Tác động ngắt ngoài có hai dạng là tác động theo **dạng mức** và **dạng sườn**

- Kích hoạt theo mức: ở chế độ này các chân INT0, INT1 bình thường ở mức cao giống như các chân khác của vđk, khi có tín hiệu mức thấp cấp tới thì tín hiệu này kích hoạt ngắt. lưu ý là trước khi thực hiện lệnh cuối cùng của chương trình phục vụ ngắt thì mức thấp tại các chân ngắt phải được chuyển lên mức cao, nếu không sẽ lại gây ra một ngắt ngay lập tức.
- Kích hoạt theo sườn: bình thường các chân ngắt của vđk ở mức cao, khi có tín hiệu tác động vào chúng có dạng sườn xuống thì sẽ tác động ngắt.

Để sử dụng chế độ ngắt này thì ta phải tác động vào thanh ghi TCON cụ thể là TCON.0 = 1 thì cho phép ngắt ngoài 0 kích hoạt theo sườn, còn TCON.2 =1 thì cho phép ngắt ngoài 1 kích hoạt theo sườn.

b) Cách lập trình

Để lập trình cho ngắt của vđk ta phải thực sự hiểu được bản chất của ngắt và quá trình thực hiện ngắt của vđk diễn ra.
Mẫu hàm viết cho chương trình phục vụ ngắt như sau:

```
Void  Name (void) interrupt x ( x là số thứ thực của ngắt )
{

// chuong trinh phuc vu ngat

}
```

Trong chương trình chính ta phải có các thao tác thiết lập việc cho phép các ngắt được sử dụng.

5.3. Các bài toán ứng dụng ngắt ngoài của vđk

Bài toán1:

Bạn sử dụng ngắt ngoài 0, với nhiệm vụ là bình thường vđk bật một led đơn tại chân p1.0 và khi có ngắt (khi bạn ấn công tắc p3.2) thì tắt led ở p1.0 và bật led ở p1.1. Bỏ tay nhấn công tắc ra thì led p1.0 sáng và led p1.1 tắt. Hãy thử với cả 2 trường hợp là ngắt kích phát sườn và ngắt kích phát mức, nhận xét kết quả thu được.

Code for ex1:

```
#include <AT89x52.h>
#define on 0
#define off 1
sbit led1=P1^0;
sbit led2=P1^1;
void delay(long int n)
{
    long int i;
    for (i=0;i<n;i++){}
}
void INTO_(void) interrupt 0 // Chuong trinh phuc vu ngat
{
    led1 = on; led2= off;
    delay(5000);
    led1 = off; led2 = on;
}
void main(void)
{
    EA = 1; //Cho phep ngat toan cuc
    EX0 = 1; //Cho phep ngat ngoai
    TCON=0x01;//ngat kich phat suon, voi ngat kich phat muc :
TCON=0x00;
    led1=off; led2=on;
    while(1)
    {
    }
}
```

Bài toán2:

- *Sử dụng INT0 và INT1:*

```
/* ----------------------------------------
    De bai yeu cau la: Binh thuong khi chua co ngat thi LCD hien thi dong chu:
     " DKS.co,."
     Khi co ngat ngoai 0 thi thong bao: " Ngat ngoai 0 !"
     Khi co ngat ngoai 1 thi thong bao: " Ngat ngoai 1 !"
    ---------- ----------------------------*/
```

```
#include <AT89x51.h>
#define        tat      1     /* muc logic 1 */
#define        bat      0     /* muc logic 0 */
#include <string.h>
// -----------khoi tao ngat ngoai-----------------//
void khoi_tao_INT0_INT1(void)// Ham khoi tao
{
EA=0;// Cam ngat toan cuc.
EX0=1;// Cho phep ngat ngoai 0.
EX1=1;// Cho phep ngat ngoai 1.
EA=1;// Cho phep ngat toan cuc.
}
sfr LCDdata = 0xA0;
sbit BF = 0xA7; // Co ban bit 7
sbit RS = P3^7;
sbit RW = P3^6;
sbit EN = P3^5;
/*---------Cac chuong trinh con cua lcd ------*/
//---------------Chuong trinh con kiem tra su san sang cua lcd----------------;
void wait(void)
{
                long n = 0;
                EN=1;// Dua chan cho fep len cao
                RS=0;// Chon thanh ghi lenh
                RW=1;//      Doc tu LCD
                LCDdata=0xff;// Gia tri 0xFF
                while(BF){n++; if(n>110) break;}// Kiem tra co ban
                // Neu ban dem n den 100 roi thoat khoi while
                EN=0;// Dua xung cao xuong thap de chot
                RW=0;// Doc tu LCD
}
//------------------Chuong trinh con thiet lap lenh cho LCD----------------------;
void LCDcontrol(unsigned char x)
{
                EN=1;// Dua chan cho fep len cao
                RS=0;// Chon thanh ghi lenh
                RW=0;// Ghi len LCD
                LCDdata=x;// Gia tri x
                EN=0;// Xung cao xuong thap
                wait();// Doi LCD san sang
}
//---------------chuong trinh con thiet lap mot so thong so cua lcd---------------;
void LCDinit(void)
{
```

```
                LCDcontrol(0x38);// 2 dong va ma tran 5x7
                LCDcontrol(0xc0);
                LCDcontrol(0x0f);// Bat con tro
                LCDcontrol(0x01);// Xoa man hinh
                LCDcontrol(0x82);
}
//------------------Chuong trinh con thiet lap dulieu cho LCD----------------------;
void LCDwrite(unsigned char c)
{
                EN=1;// Cho fep muc cao
                RS=1;// Ghi du lieu
                RW=0;// Ghi len LCD
                LCDdata=c;// Gia tri C
                EN=0;// Xung cao xuong thap
                wait();// Cho
}
/*================ Chuong trinh con tre thoi gian ================*/
void delay(long time)                // chuong trinh tre thoi gian.
{
  long n1;                           // khai bao bien cuc bo.
  for (n1=0;n1<time;n1++)    // vong lap time lan.
        {
          ;                                  // khong thuc hien gi ca !!!
        }
}
/* ------------------- Cac chuong trinh ngat -------------------------*/
void INT0_int(void) interrupt 0 //Ngat ngoai 0
{
        P1_0=0; // led bao cho ngat ngoai 0
        P1_7=1;  // led bao cho ngat ngoai 1
        LCDinit();
        LCDcontrol(0x82);
        LCDwrite('N');
        LCDwrite('g');
        LCDwrite('a');
        LCDwrite('t');
        LCDwrite(' ');
        LCDwrite('n');
        LCDwrite('o');
        LCDwrite('a');
        LCDwrite('i');
        LCDwrite('0');
}
//-----------------------------------------------------------------------//
```

```
void INT1_int(void) interrupt 2 //Ngat ngoai 1
{
   P1_0=1; // led bao cho ngat ngoai 0
        P1_7=0;  // led bao cho ngat ngoai 1
        LCDinit();
        LCDcontrol(0x82);
        LCDwrite('N');
        LCDwrite('g');
        LCDwrite('a');
        LCDwrite('t');
        LCDwrite(' ');
        LCDwrite('n');
        LCDwrite('o');
        LCDwrite('a');
        LCDwrite('i');
        LCDwrite('1');
}
/*          ====================          Chuon          trinh          chinh
====================*/
 //* Hàm main:
void main(void)
{
        khoi_tao_INT0_INT1();
        LCDinit();
        P1_0=0;
        P1_7=0;
//      LCDcontrol(0x82);
        LCDwrite('D');
        LCDwrite('K');
        LCDwrite('S');
        LCDwrite('.');
        LCDwrite('C');
        LCDwrite('o');
        LCDwrite(',');
        LCDwrite('.');
        while(1)// Lap vo han
        {
                ;
        }
 }
```

Bài 6 : Bộ định thời của VĐK và ngắt định thời

Bài toán ứng dụng: tạo xung vuông với tần số yêu cầu check với bộ đo tần số; lập trình quét led 7 thanh với các hiệu ứng, điều chế độ rộng xung ứng dụng điều khiển động cơ một chiều và động cơ bước, mạch quáng cáo với led Matrix. Ma trận bàn phím.

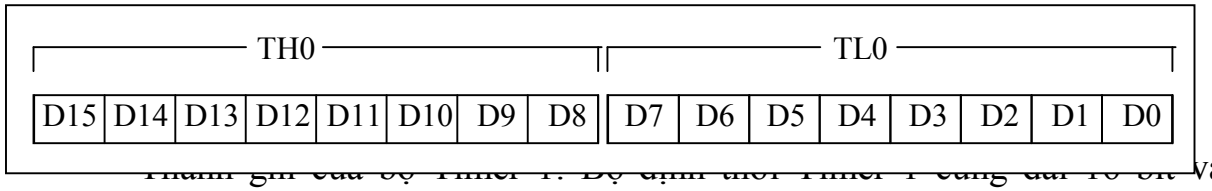
6.1. Cơ sở lý thuyết

6.1.1. Bộ định thời của vi điều khiển là gì.

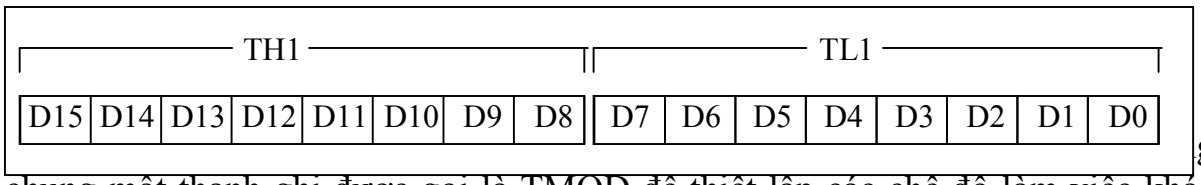
Vi điều khiển 8051 nói chung có 2 bộ định thời chúng dùng để tạo trễ hay làm bộ đếm để đếm các sự kiện bên ngoài vđk.

6.1.2. Thanh ghi chứa, thanh ghi thiết lập chế độ cho bộ định thời

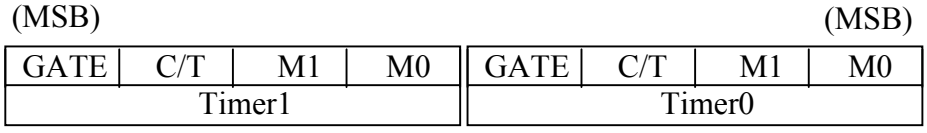
- Thanh ghi của bộ Timer 0: Thanh ghi 16 bit của bộ Timer 0 được truy cập theo 2 byte là byte thấp và byte cao. Thanh ghi byte thấp được gọi là TL0 (Timer 0 Low byte) và thanh ghi byte cao là TH0 (Timer 0 High byte). Các thanh ghi này có thể đọc và truy cập trực tiếp như mọi thanh ghi khác.



Thanh ghi của bộ Timer 1: Bộ định thời Timer 1 cũng dài 16 bit và thanh ghi 16 bit cũng được chia ra thành hai byte là TL1 và TH1. Các thanh ghi này được truy cập và được đọc giống như các thanh ghi của bộ Timer 0.



g chung một thanh ghi được gọi là TMOD để thiết lập các chế độ làm việc khác nhau của bộ định thời. Thanh ghi TMOD là thanh ghi 8 bit gồm có 4 bit thấp dành cho bộ Timer 0 và 4 bit cao dành cho Timer 1. Trong đó hai bit thấp của chúng dùng để thiết lập chế độ của bộ định thời, còn hai bit cao dùng để xác định phép toán.



+ Các bit M0, M1: Là các bit chế độ dùng để chọn chế độ 0, 1 và 2 của các bộ Timer 0 và Timer 1.

M1	M0	Chế độ	Chế độ hoạt động
0	0	0	Chế độ bộ định thời 13 bit. Bộ định thời/bộ đếm 8 bit, định tỷ lệ trước 5 bit.

0	1	1	Chế độ bộ định thời 16 bit, không định tỷ lệ trước.
1	0	2	Chế độ 8 bit tự nạp lại. THx lưu giá trị sẽ tự nạp vào TLx mỗi khi tràn.
1	1	3	Chế độ bộ định thời chia tách

Ta chủ yếu sử dụng chế độ 8 bit tự nạp lại và chế độ 16 bit không tự nạp lại.

Ví dụ: để sử dụng timer0 ở chế độ 16bit và timer1 ở chế độ 8 bit
ta nạp giá trị cho TMOD = 0x 21 (00100001)

6.1.3. Cơ chế tạo trễ của bộ định thời và cách tính toán giá trị nạp cho bộ định thời.

*** <u>Ta xét chế đô 1 :</u>**

- Ở chế độ 1 đó là bộ định thời 16bit, do đó các giá trị trong khoảng từ 0000 đến FFFF có thể sử dụng để nạp cho TH, TL của bộ định thời.
- Sau khi TH, TL được nạp giá trị ban đầu 16 bit thì bộ định thời phải được khởi động với lệnh TR0=1 với timer0 và TR1=1 với timer1, khi này bộ định thời sẽ bắt đầu đếm tăng theo clock (trong thanh ghi chứa) từ giá trị ban đầu cho tới giá trị đỉnh là FFFF. Khi đó bộ định thời sẽ quay vòng từ ffff – 0000 và bật cờ báo tràn TFx=1 (ban đầu TFx =0) khi đó các bạn phải có thao tác để xoá cờ để lần sau còn biết được khi nào bộ đếm tràn.
- Khi bộ đếm tràn thì TH và Tl của bộ định thời sẽ mang giá trị 0 do đó các bạn phải có thao tác nạp lại giá trị ban đầu cho chúng và xoá cờ TF. Để bộ đếm được lặp lại.

Như vậy khoảng thời gian mà bộ định thời tạo trễ được chính là khoảng thời gian nó đếm tăng từ giá trị ban đầu được nạp cho tới giá trị đỉnh FFFF.

+ Tính giá trị nạp vào thanh ghi chứa của bộ định thời từ thời gian muốn trễ (t).
Gọi N = 65536- t/ chu kì máy.
Sau chuyển N sang mã hexa được một số có 4 chữ số dạng x1x2 y1y2
Khi đó giá trị nạp cho thanh ghi chứa như sau: THx =x1x2 còn TLx = y1y2.

Ví dụ các bạn muốn tạo trễ 500us thì là có giả sử chu kỳ máy = 1us
N = 65536—500/1 = 65036 và chuyển sang hex là FE0C
Từ đó TH = 0xfe còn TL =0x0c.

Các bước lập trình cho bộ định thời để tạo trễ : (mode1)

B1: Chọn chế độ 1 cho bộ định thời cần dùng, từ đó xác định giá trị nạp cho thanh ghi TMOD
B2: Tính toán giá trị ban đầu cần nạp cho TH, TL từ thời gian mong muốn trễ.
B3: Khởi động bộ định thời.
B4: kiểm tra trạng thái bật của cờ TF
B5: Dừng bộ định thời

B6: xoá cờ TF cho vòng lặp kế tiếp.
B7: Quay về b2 để nạp lại giá trị ban đầu cho TH, TL.

*** <u>Ta xét chế độ 2 :</u>**

Trong chế độ này rất giống với chế độ 1,

ở chế độ này là bộ định thời 8 bit, do vậy chỉ cho phép các giá trị từ 00- ff được nạp vào thanh ghi TH của bộ định thời.
Sau khi nạp giá trị 8 bit thì vđk sẽ sao nội dung của TH và TL và bộ định thời được khởi động qua lệnh TRx =1
Sau khi được khởi động thì bộ định thời bắt đầu đếm tăng trong thanh ghi TL, từ giá trị ban đầu cho tới giá trị đỉnh FF. và khi đã quay vòng từ FF về 00 thì cờ báo TFx được bật lên =1. Và khi này thanh ghi TL mang giá trị 0 nhưng TL sẽ ngay lập tức được tự động nạp lại với giá trị ban đầu được giữ ở thanh ghi TH.
Như vậy trong chế độ này mỗi khi bộ đếm tràn thì thanh chứa sẽ được vđk nạp lại giá trị ban đầu do đó chế độ này gọi là chế độ tự động nạp lại.

Các bước lập trình cho bộ định thời để tạo trễ : (mode2)
B1: Nạp giá trị cho thanh ghi thiết lập chế độ TMOD với timer0 bạn muốn sử dụng.
B2: Nạp vào thanh ghi TH giá trị đếm ban đầu.
B3: Khởi động bộ định thời.
B4: Kiểm tra cờ báo tràn TF.
B5: Xoá cờ tràn TF, quay về bước 4.
+ Tính giá trị nạp vào thanh ghi chứa của bộ định thời từ thời gian muốn trễ (t).
Gọi N = 256- (t)/ chu kì máy.
Sau chuyển N sang mã hexa được một số ta nạp số này vào thanh ghi TH và cả TL. (nạp vào TL giá trị đếm lần 1 và nạp vào TH cho những lần sau để khi bộ đếm tràn thì phần cứng của vđk sẽ tự động sao giá trị trong TH sang TL).
Ví dụ các bạn muốn tạo trễ 50us thì là có giả sử ckmáy = 1us
N = 256—50 = 206 và chuyển sang hex : ce
Từ đó TH = 0xce còn TL =0xce.

6.1.4. Ngắt của bộ định thời

Ta xét bộ định thời trong chế độ ngắt của chúng.
Khi sử dụng bộ định thời trong hoạt động ngắt cũng giống như ngắt ngoài ta cũng cần thấy được nguyên tắc chung nhất:

- Đó là phải khai báo ngắt của bộ định thời muốn sử dụng.
- Nguồn gây ngắt
- Chương trình chính và chương trình phục vụ ngắt.

Với ngắt định thời việc khai báo ngắt bao gồm việc khai báo ngắt toàn cục là EA = 1 và khai báo ngắt cho từng bộ định thời muốn dùng ET0 =1 (timer0) và

ET1=1 (timer1).
Nguồn báo ngắt định thời (với cả hai chế độ) đều là khi bộ đếm tràn (khi cờ TFx=1) đó là nguồn báo ngắt.
Khi cờ TF=1 thì vđk kết thúc công việc hiện tại ở chương trình chính và chuyển vào chương trình phục vụ ngắt căn cứ theo địa chỉ của chương trình phục vụ ngắt.
Khi thực hiện xong chương trình phục vụ ngắt vđk quay trở về chương trình chính tại nơi bị gián đoạn khi trước.

Tóm lại ngắt định thời là việc ta sử dụng bộ định thời tạo trễ một khoảng thời gian được tính toán trước, sau khoảng thời gian này một công việc ta mong muốn sẽ được thực hiện ở chương trình phục vụ ngắt. Công việc đó có thể coi là độc lập với công việc ở chương trình chính.

Các công việc lập trình mà bạn sử dụng ngắt định thời là:
- Tính toán thời gian trễ mong muốn và công việc muốn thực hiện sau khoảng thời gian đó.
- Khai báo ngắt định thời trong chương trình chính (EA= 1, ETx=1).
- Lựa chọn bộ định thời và chế độ của nó trong thanh ghi TMOD (ở chương trình chính).
- Khởi động bộ định thời (ở chương trình chính).
- Xây dựng chương trình chính và các chương trình con cần thiết.
- Xây dựng chương trình phục vụ ngắt thực hiện 1 công việc mong muốn (xác định rõ địa chỉ ngắt theo thứ tự ngắt trong bảng vectơ ngắt được trình bày trong phần trước).

Chú ý: Trong chương trình phục vụ ngắt các bạn phải xoá cờ báo tràn TFx, và nạp lại giá trị cho bộ định thời với chế độ 1 còn chế độ 2 thì không cần làm.

6.2.Các bài toán minh họa.

Bài toán 1:

Sử dụng bộ định thời với ngắt của nó tạo một xung vuông với tần số 1k đối xứng (50% mức 1 và 50 % mức 0).

Giải quyết bài toán như sau:
Như vậy ta có f = 1khz → T (chu kỳ) = 10^-3s = 1000 us.
⇨ T1= 500us, T0= 500us.
Như vậy chu kỳ tạo xung sẽ chia làm hai phần, do xung là đối xứng nên hai phần này bộ đếm đếm giống nhau (tạo trễ một khoảng thời gian như nhau). Nửa chu kỳ đầu bộ đếm tạo trễ 500 us cho mức logic 0, sau khi bộ đếm đếm tới giá trị đỉnh của hai thanh ghi chứa TH, TL là ffff thì cờ báo TF =1 gây ngắt, vđk phải chuyển vào chương trình phục vụ ngắt và sẽ lật trạng thái xung ra. Trong nửa chu kỳ còn lại cũng như vậy và cứ như thế.
Do thời gian tạo trễ là 500us do đó ta phải lựa chế độ 16 bit.
Ta sử dụng công thức tính để tạo ra giá trị nạp cho TH. TL
Các bạn xây dựng chương trình và quan sát kết quả trên phần mềm mô phỏng,

các đưa ra cửa sổ quan sát Timer0 và pin p1.0

Code for ex 1:

```
#include <REGX51.H>

int check=0; // khai bao bien trang thai cua ctrinh ngat t0;
sbit xung1 = P1^0;
/*--Chuong trinh con nhap gia tri cho timer0 trong che do 16 bit ---*/
void nhap ( void)
  {
    TH0 = 0xfe ;   // lay (65536 - 500) -- hex duoc la fe0c  do  su dung thach anh
12Mhz
         TL0 = 0x0c ;

  }
/*-------Chuong trinh phuc vu ngat timer 0 tao xung --
  voi hai doan chuong trinh ------------------------*/
void timer0_int (void) interrupt 1
 {
     check ++;
          TF0=0;   // xoa co bao tran timer0
        switch ( check)
         {
      case 1:
           {
            xung1 =1; // tao muc 1 ra pin
                 nhap();
                 break;  // thoat ra
           }
          case 2:
          {
            xung1 =0; // tao muc 1 ra pin
                 nhap();
            check =0;  // khoi tao lai bien trang thai.
                 break;  // thoat ra
           }
         }
 }
/*----------Chuong trinh chinh------------*/
void main ( void)
  {
    EA = 1 ; // cho phep ngat toan cuc.
         ET0= 1; // cho phep ngat dinh thoi T0
         TMOD =      0x01; // timer0 voi che do 16 bit ko nap lai
    xung1=0; // gia tri ban dau cua xung vuong muon tao
         nhap();          // khoi tao gia tri ban dau cho bo dem t0.
```

```
        TR0=1;  // khoi dong bo dinh thoi
        while (1) {;} // khong lam gi de doi ngat T0

  }
```

Bài toán 2:

Sử dụng bộ định thời với ngắt của nó tạo một xung vuông với tần số 1k đối xứng (30% mức 1 và 70 % mức 0).

Với bài toán này cũng vậy rất giống với bài toán trên, bạn đã hiểu kĩ bài trển thì trong bài toán này trở nên rất đon giản.
Sự thay đổi duy nhất:
Cũng với một chu kỳ tạo xung (một chu kỳ ngắt) gồm hai phần. Phần 1 bộ định thời sẽ tạo trễ 300us cho mức logic 0, sau khi bộ định thời đếm tràn và gây ngắt thì vdk sẽ chuyển vào chương trình phục vụ ngắt sẽ lật mức đầu ra của xung, và tiếp sau bộ định thời sẽ giữ trễ 700 us cho mức logic 1 và bộ định thời lại đếm tràn cờ tf =1 lại gây ngắt và vđk lại vào chương trình phục vụ ngắt và lật mức của xung. Cứ như vậy sẽ tạo ra được xung mong muốn.

Dưới đây là code giải quyết bài toán:

Code for ex2:

```
#include <REGX51.H>

int check=0; // khai bao bien trang thai cua ctrinh ngat t0;
sbit xung1 = P1^0;
/*--Chuong trinh con nhap gia tri cho timer0 trong che do 16 bit ---*/
void nhap1 ( void)  // tao ra 300 us
  {
    TH0 = 0xfe ;
        TL0 = 0xd4 ;

  }
void nhap2 ( void)  // tao ra 700 us
  {
    TH0 = 0xfd ;
        TL0 = 0x44 ;

  }
/*-------Chuong trinh phuc vu ngat timer 0 tao xung --
  voi hai doan chuong trinh ------------------------*/
void timer0_int (void) interrupt 1
```

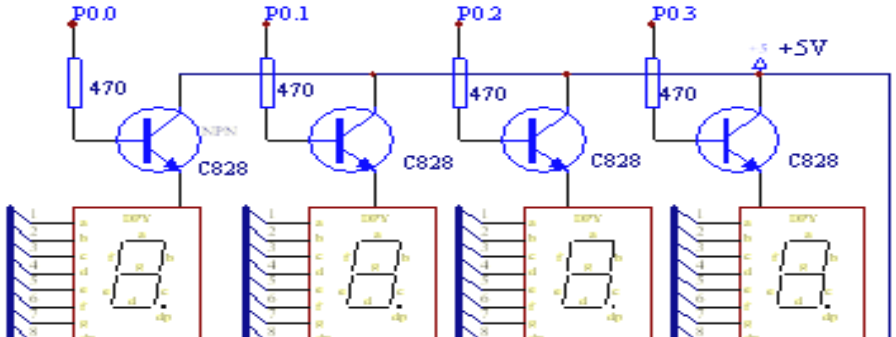
```
{
    check ++;
        TF0=0;   // xoa co bao tran timer0
       switch ( check)
        {
     case 1:
          {
           xung1 =1; // tao muc 1 ra pin
                nhap1();
                break;  // thoat ra
           }
          case 2:
          {
           xung1 =0; // tao muc 1 ra pin
                nhap2();
           check =0;  // khoi tao lai bien trang thai.
                break;  // thoat ra
           }
         }
}
/*----------Chuong trinh chinh------------*/
void main ( void)
 {
  EA = 1 ; // cho phep ngat toan cuc.
       ET0= 1; // cho phep ngat dinh thoi T0
       TMOD =     0x01; // timer0 voi che do 16 bit ko nap lai
  xung1=0; // gia tri ban dau cua xung vuong muon tao
       nhap2();       // khoi tao gia tri ban dau cho bo dem t0.
       TR0=1;  // khoi dong bo dinh thoi
       while (1) {;} // khong lam gi de doi ngat T0
 }
```

Bài toán 3:

Sử dụng bộ định thời với ngắt của nó để quét led 7 thanh tạo ra một bộ chỉ thị số :

Sử dụng kĩ thuật này tạo trên 4 led số 2007 và nhấp nháy.

Kĩ thuật quét led chính là các bạn tạo ra các hình ảnh đứt quãng trên 4 led dựa trên kĩ thuật 24 hình/s hay là độ lưu ảnh trên võng mạc mắt người là < 0.1s



Code for ex3:

```
#include <AT89x51.H>
long code1,j=1, dem, y1,y2,y3,y4;
#define on  1                   // muc logic 1 (+5v)
#define off  0                  // muc logic 0 (0v)
/* --- dinh  nghia cac bit dung------*/
 sbit  led1 = P0^3;
 sbit  led2 = P0^2;
 sbit  led3 = P0^1;
 sbit  led4 = P0^0;
//=------------Chuong trinh con chuyen so thap phan sang ma led 7 thanh-------//
void decode_led7seg1(unsigned char number1)
// chuyen sang ma led ko co dau cham dp.
{
  switch (number1)
   {
   case 0: {code1=0xc0;break;}
   case 1: {code1=0xf9;break;}
   case 2: {code1=0xa4;break;}
   case 3: {code1=0xb0;break;}
   case 4: {code1=0x99;break;}
   case 5: {code1=0x92;break;}
   case 6: {code1=0x82;break;}
   case 7: {code1=0xf8;break;}
   case 8: {code1=0x80;break;}
   case 9: {code1=0x90;break;}
   }
}
 //---------------Chuong trinh tao tre-------------------//
  void delay(long tg)   // chuong trinh tre thoi gian.
   {
   long n1;             // khai bao bien cuc bo.
   for (n1=0;n1<tg;n1++)// vong lap time lan.
      {
```

```
            ;              // khong thuc hien gi ca !!!
       }
     }
 //---------------------//
   void nhap(void)     // thoi gian nay duoc tinh cho ca 4 led
     {
       TH0=0xf0;
       TL0=0x20;

     }
//-----------Chuong trinh phuc vu ngat timer0-------------//
 void timer0(void) interrupt 1 //Ngat timer 0
    {       //---------vong quet 1 cho led1.
   switch (j)
     {
      case 1:
       {
        j++;
        P1=y1;     // dua du lieu ra led1
        led1=on;
        led2=off;
        led3=off;
        led4=off;
        nhap();
        break;
       }
//---------Vong quet 2 cho led2.-------//
      case 2:
       {
         j++;
         P1=y2;     // dua du lieu ra led2
         led2=on;
         led1=off;
         led3=off;
         led4=off;
         nhap();
         break;
       }
//--------Vong quet 3 cho led3--------//
     case 3:
       {
         j++;
         P1=y3;    // dua du lieu ra led3
         led3=on;
         led2=off;
         led1=off;
```

```
        led4=off;
        nhap();
        break;
       }
//--------Vong quet 4 cho led4---------//
     case 4:
       {
        P1=y4;    // dua du lieu ra led4
        led4=on;
        led2=off;
        led3=off;
        led1=off;
        nhap();
        j=1;
        break;
       }
     } // end of switch
   }
/*
--- Voi kit thi ma led la: tuong ung voi cac so tu 0-->9
       0c0h,0f9h,0a4h,0b0h,099h,092h,082h,0f8h,080h,090h
*/
//-------------------------------------------------------------------------//
void main ( void)  // chuong trinh chinh
 {  // begin of main
    EA= 1;          // cho phep ngat toan cuc.
    TMOD=0x01;  // Timer 0 che do 16 bit not auto reload
    ET0=1;      // Cho phep ngat timer 0
    nhap();

          y1=0xa4; // hien thi so 2 ra led1;
          y2=0xc0; // hien thi so 0 ra led2;
          y3=0xc0; // hien thi so 0 ra led3;
          y4=0xf8; // hien thi so 7 ra led4;
          for (;;)
          {
           TR0 = 1 ; // khoi dong bo dinh thoi t0;
           delay(10000);
           TR0 =0;
           delay(8000);
          }
 } // end of main
```

Các bạn phát huy thêm, tự xây dựng cho mình một phương pháp giải quyết vấn đề trên, Các bạn phải thực sự hiểu về cơ chế hoạt động của ngắt định thời và của chính bộ định thời. Và với vi điều khiển có thể có nhiều bộ định thời, chúng

độc lập với nhau, nên hoạt động của vđk vô cùng đa dạng.

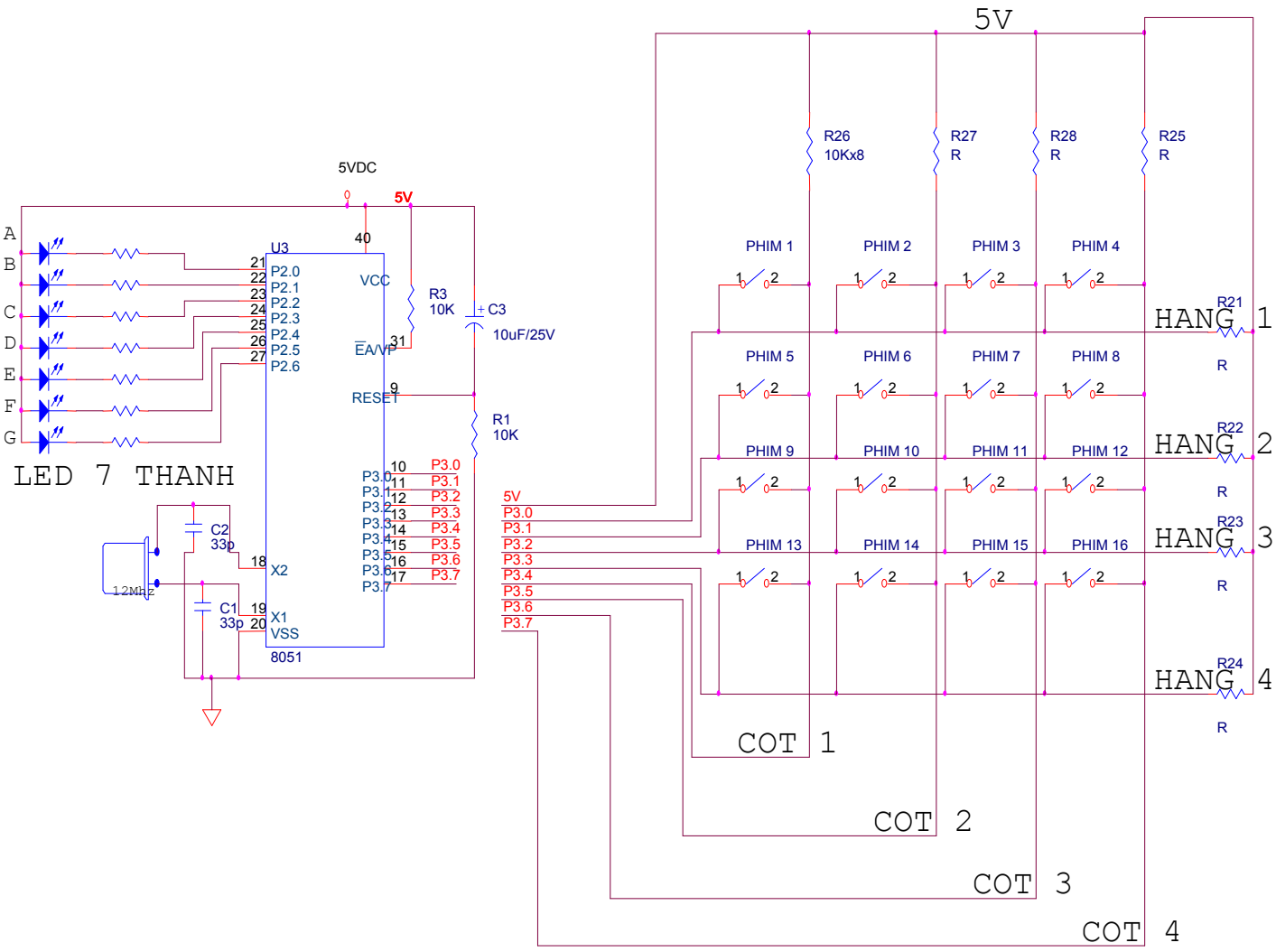
6.3. Bàn phím ma trận 4x4

Nhiệm vụ:

Quét bàn phím 16 phím bấm(4x4), xem phím nào được bấm, các phím được đánh số từ 0 đến 15 rồi hiển thị giá trị ra LCD.

6.3.1. Lắp mạch theo sơ đồ sau

Đây là sơ đồ ***ví dụ*** một cách ghép nối bàn phím ma trận 4x4 với vđk:



6.3.2: Nguyên lí quét phím-thuật toán 1:

- Vì sao mạch phím đấu theo ma trận. Nếu để đọc từ 16 nút bấm bình thường phải dùng 16 chân vi điều khiển. Nếu đấu theo dạng ma trận thì chỉ mất 8 chân ta cũng có thể đọc được 16 phím bấm.

- Có 2 cách quét phím theo cột và theo hàng, giáo trình giới thiệu với các bạn cách quét theo hàng, quét theo cột các bạn có thể làm tương tự.

- Bước 1 : Ta đưa chân nối với Hàng 1 xuống 0V. Rồi ta kiểm tra giá trị logic của các chân nối với cột. Nếu phím 1 được bấm thì Cột 1 sẽ có giá trị bằng 0. Nếu phím 2 được bấm thì Cột 2 sẽ có giá trị bằng 0. Nếu phím 3 được bấm thì Cột 3 sẽ có giá trị bằng 0. Nếu phím 4 được bấm thì Cột 4 sẽ có giá trị bằng 0.

Ta căn cứ vào đó để xác định xem phím nào được bấm.

- Bước 2 : Ta đưa chân nối với Hàng 2 xuống 0V.Rồi ta kiểm tra giá trị logic của các chân nối với các cột. Nếu phím 5 được bấm thì Cột 1 sẽ có giá trị bằng 0. Nếu phím 6 được bấm thì Cột 2 sẽ có giá trị bằng 0. Nếu phím 7 được bấm thì Cột 3 sẽ có giá trị bằng 0. Nếu phím 8 được bấm thì Cột 4 sẽ có giá trị bằng 0. Ta căn cứ vào đó để xác định xem phím nào được bấm.

- Bước 3 : Ta đưa chân nối với Hàng 3 xuống 0V.Rồi ta kiểm tra giá trị logic của các chân nối với các cột . Nếu phím 9 được bấm thì Cột 1sẽ có giá trị bằng 0. Nếu phím 10 được bấm thì Cột 2 sẽ có giá trị bằng 0. Nếu phím 11 được bấm thì Cột 3 sẽ có giá trị bằng 0. Nếu phím 12 được bấm thì Cột 4 sẽ có giá trị bằng 0. Ta căn cứ vào đó để xác định xem phím nào được bấm.

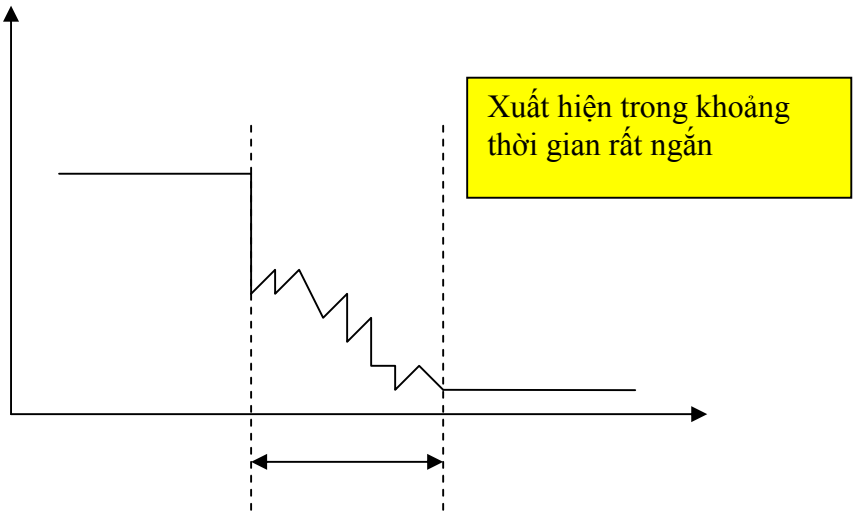
- Bước 4 : Ta đưa chân nối với Hàng 1 xuống 0V. Rồi ta kiểm tra giá trị logic của các chân nối với các cột. Nếu phím 13 được bấm thì Cột 1 sẽ có giá trị bằng 0. Nếu phím 14 được bấm thì Cột 2 sẽ có giá trị bằng 0. Nếu phím 15 được bấm thì Cột 3 sẽ có giá trị bằng 0. Nếu phím 16 được bấm thì Cột 4 sẽ có giá trị bằng 0. Ta căn cứ vào đó để xác định xem phím nào được bấm.

Ta sẽ dùng câu lệnh if để kiểm tra.

6.3.3. Nguyên lý quét phím - thuật toán 2

- Bước 1 : Đặt hàng hoặc cột làm đầu vào mức cao
- Bước 2 : Đặt các cột hoặc hàng làm đầu ra mức thấp. Sau đó kiểm tra các đầu vào (hàng hoặc cột) xem có đầu vào ở mức thấp (=0) không nếu có thì trễ chống rung. Sau đó lại kiểm tra lại các đầu vào để chính xác 1 lần nữa. Sau khi tìm thấy đầu vào ở mức thấp thì chuyển các đầu ra trước đó thành đầu vào mức cao và đầu vào trước đó thành đầu ra mức thấp.
- Bước 3 : Kiểm tra các đầu vào mới xem có đầu nào bị kéo xuống thấp không để xác định nốt hàng cột còn lại.
  - Chúng ta có thể chống rung bằng 2 cách

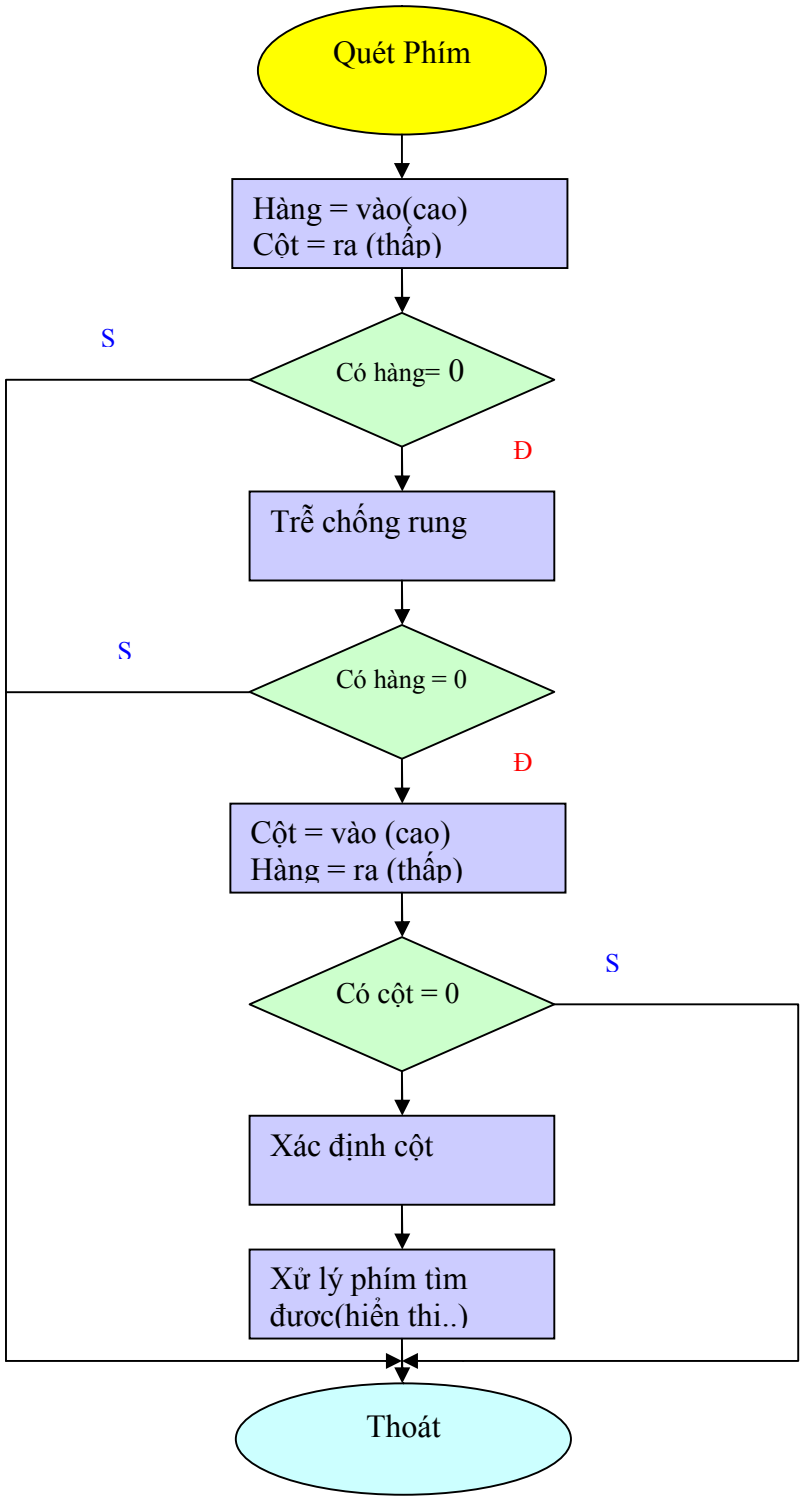
    + Phần cứng



    + Phần mềm

- Sơ đồ thuật toán :

Thuật toán đầu vào



➔ Sau đây là bài toán thực hành với bàn phím ma trận 4x4. Thực hiện đọc 16 phím và hiển thị tên chức năng định trước của chúng trên màn hình LCD.

Trong bài toán đã quy định việc ghép nối phần cứng của bàn phím với vđk, gồm 4 cột và 4 hàng, cùng với modun hiển thị LCD ghép nối phần cứng với vđk vẫn

như trong các bài học trước.

```
// Ban phim ma tran ---------------------------------
// chuong trinh lam quen voi ban phim matran 4*4
// Hien thi phim an  tren LCD

/*-------------Ban phim matran---------------
 ================
|  0  |  1  |  2  |  3 |
 ================
|  4  |  5  |  6  |  7 |
 ================
|  8  |  9  |  +  |  -  |    Cac phim chuc nang voi ten dinh truoc.
 ================
|  *  |  /  |  D  |  = |
 ================
-----------------------------------------*/
#include <REGX51.H>
long codeLCD, hienthi ;
sbit co1 = P1^3;
sbit co2 = P1^2;
sbit co3 = P1^1;
sbit co4 = P1^0;
sbit ha1 = P1^7;
sbit ha2 = P1^6;
sbit ha3 = P1^5;
sbit ha4 = P1^4;

#include < string.h>
sfr LCDdata = 0xA0;// Cong 2 , 8 bit du lieu  P0 co dia chi 0x80, P1 0x90 , P2
0xA0
 sbit BF = 0xA7; // Co ban bit 7 cua p2
 sbit RS = P3^7;
 sbit RW = P3^6;
 sbit EN = P3^5;
/*------chuyen sang ma ASCII cho LCD--------*/
void decode_LCD(unsigned char number2)
{
  switch (number2)
    {
   case 0: {codeLCD='0';break;}
   case 1: {codeLCD='1';break;}
   case 2: {codeLCD='2';break;}
   case 3: {codeLCD='3';break;}
```

```
  case 4: {codeLCD='4';break;}
  case 5: {codeLCD='5';break;}
  case 6: {codeLCD='6';break;}
  case 7: {codeLCD='7';break;}
  case 8: {codeLCD='8';break;}
  case 9: {codeLCD='9';break;}
  case 10: {codeLCD='+';break;}
  case 11: {codeLCD='-';break;}
  case 12: {codeLCD='*';break;}
  case 13: {codeLCD='/';break;}
  case 14: {codeLCD='D';break;}
  case 15: {codeLCD='=';break;}

  }
}

/*---------Cac chuong trinh con cua lcd ------*/
//---------------Chuong trinh con kiem tra su san sang cua lcd----------------;
void wait(void)
{
                    long n = 0;
                    EN=1;// Dua chan cho fep len cao
                    RS=0;// Chon thanh ghi lenh
                    RW=1;//      Doc tu LCD
                    LCDdata=0xff;// Gia tri 0xFF
                    while(BF){n++; if(n>110) break;}// Kiem tra co ban
                    // Neu ban dem n den 100 roi thoat khoi while
                    EN=0;// Dua xung cao xuong thap de chot
                    RW=0;// Doc tu LCD
}
//------------------Chuong trinh con thiet lap lenh cho LCD----------------------;
void LCDcontrol(unsigned char x)
{
                    EN=1;// Dua chan cho fep len cao
                    RS=0;// Chon thanh ghi lenh
                    RW=0;// Ghi len LCD
                    LCDdata=x;// Gia tri x
                    EN=0;// Xung cao xuong thap
                    wait();// Doi LCD san sang
}
//---------------chuong trinh con thiet lap mot so thong so cua lcd--------------;
void LCDinit(void)
{
                    LCDcontrol(0x38);// 2 dong va ma tran 5x7
```

```
                    LCDcontrol(0xc0);
                    LCDcontrol(0x0e);// Bat con tro
                    LCDcontrol(0x01);// Xoa man hinh
}
//------------------Chuong trinh con thiet lap dulieu cho LCD----------------------;
void LCDwrite(unsigned char c)
{
                    EN=1;// Cho fep muc cao
                    RS=1;// Ghi du lieu
                    RW=0;// Ghi len LCD
                    LCDdata=c;// Gia tri C
                    EN=0;// Xung cao xuong thap
                    wait();// Cho
}
/*-----------------Chuong trinh con quet phim------------*/

void quetphim (void)
{
    ha1=0; ha2=ha3=ha4=1;
  if ( co1==0) codeLCD='0';
  if ( co2==0) codeLCD='1';
  if ( co3==0) codeLCD='2';
  if ( co4==0) codeLCD='3';
                    ha2=0; ha1=ha3=ha4=1;
  if ( co1==0) codeLCD='4';
  if ( co2==0) codeLCD='5';
  if ( co3==0) codeLCD='6';
  if ( co4==0) codeLCD='7';
                    ha3=0; ha2=ha1=ha4=1;
  if ( co1==0) codeLCD='8';
  if ( co2==0) codeLCD='9';
  if ( co3==0) codeLCD='+';
  if ( co4==0) codeLCD='-';
    ha4=0; ha2=ha3=ha1=1;
  if ( co1==0) codeLCD='*';
  if ( co2==0) codeLCD='/';
  if ( co3==0) codeLCD='D';
  if ( co4==0) codeLCD='=';
 }
  /*-----------------------------------------*/
//  main program ------>>-----
 main()

{
```

```
 ha1=ha2=ha3=ha4=co1=co2=co3=co4=1;   // thiet lap cot dau vao vdk
                    LCDinit();
                    LCDwrite('P');
                    LCDwrite('h');
                    LCDwrite('i');
                    LCDwrite('m');
                    LCDwrite(' ');
                    LCDwrite('a');
                    LCDwrite('n');
                    LCDwrite(' ');
                    LCDwrite('l');
                    LCDwrite('a');
                    LCDwrite(':');
  while(1)
  {

                    quetphim();
                    LCDcontrol(0x8c);
                    LCDwrite(codeLCD);
                    }
}
```

Bài7: Bộ đếm của VĐK 8051 (Counter)

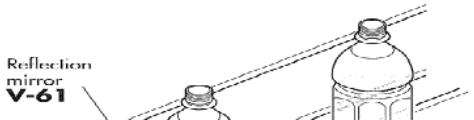
Bài toán ứng dụng: đếm sản phẩm với hiển thị 7 thanh và LCD

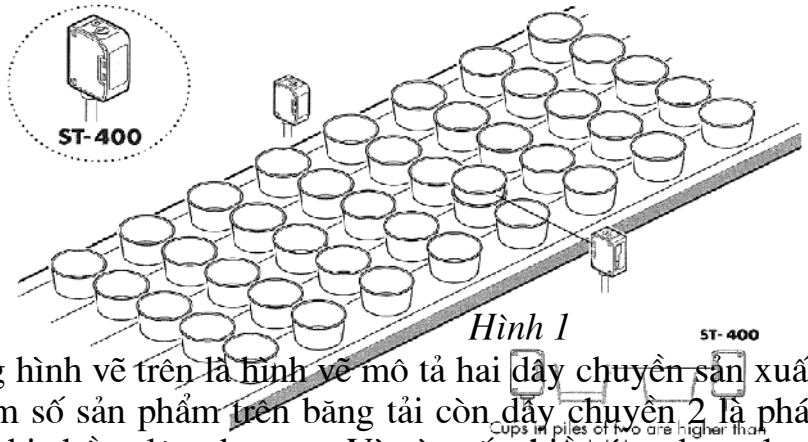
7.1. Ý nghĩa thực tiễn của các bộ đếm (Counter).

Bộ đếm của vdk có tác dụng dùng để đếm các dữ kiện bên ngoài.

Cấu tạo như một bộ đếm bình thường với xung clock cho bộ đếm là xung từ bên ngoài tác động vào.

ỉng dụng của bộ đếm là vô cùng rộng rãi trong thực tế của đời sống sinh hoạt cũng như trong sản xuất. Đặc biệt là trong các dây truyền sản xuất công nghiệp, là đếm sản phẩm trong tất cả các lĩnh vực sản xuất: như đếm bao bì (xi măng, đường, giấy,..), đếm hộp, chai,vv… (như hình 1)



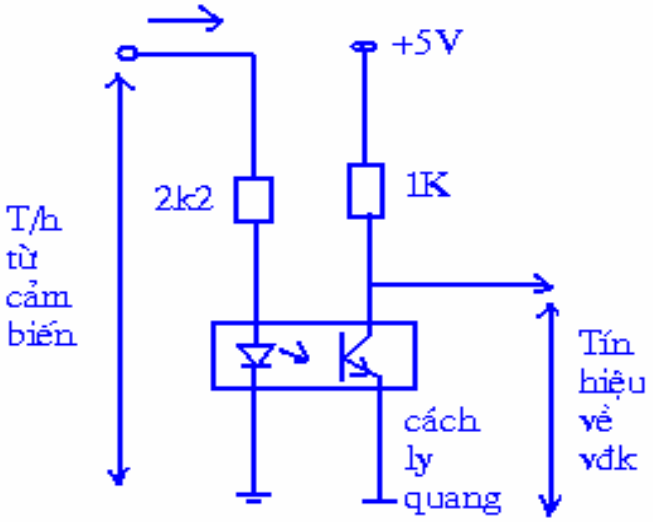


Hình 1

Trong hình vẽ trên là hình vẽ mô tả hai dây chuyền sản xuất. Trong dây truyền 1 là đếm số sản phẩm trên băng tải còn dây chuyền 2 là phát hiện và đếm số sản phẩm bị chồng lên nhau,vv.. Và còn rất nhiều ứng dụng thực tiễn hơn nữa.

Đầu ra của các cảm biến này chính là tín hiệu điện áp dạng **xung vuông** tính hiệu này được chuẩn hoá và đưa tới chân đếm của các bộ vi xử lý. Bộ vi xử lý sẽ xử lý tính toán, hiển thị, truyền thông hay đưa ra các tín hiệu cảnh báo.

Mạch chuẩn hoá tín hiệu từ cảm biến đếm sản phẩm



7.2. Cơ sở lý thuyết và lập trình vi điều khiển 8051 thành một bộ đếm.

Vđk 8051 bình thường có 2 bộ định thời, là 3 bộ với thành viên AT89x52.

Mỗi bộ định thời có hai chức năng song song nhau là định thời và counter.

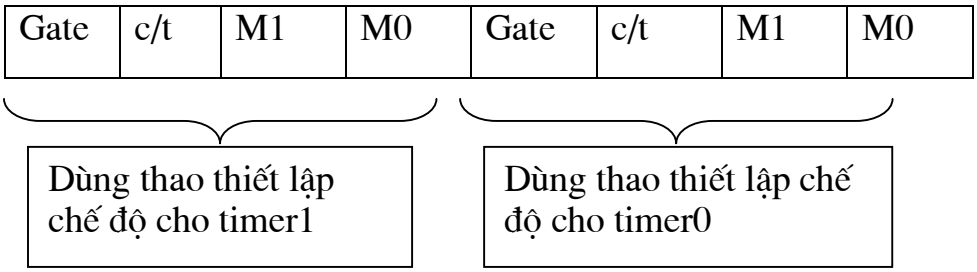
Như vậy sẽ có tối đa là 3 bộ đếm với họ 8051.

→ *Đầu vào của bộ đếm sự kiện ngoài là các chân của vđk tương ứng với các bộ định thời.*

Cụ thể là chân T0 (p3.4) là đầu vào bộ counter 0 ; chân T1 (p3.5) là đầu vào bộ counter 1 ; chân T2 (p1.0) là đầu vào bộ counter 2 (các bạn có thể

tra trong datasheet của vđk này).

→*Thiết lập vđk thành bộ đếm sự kiện ngoài:*
Thông qua thanh ghi điều khiển bộ định thời TMOD:



Gate =0 là mặc định cho ta thao tác bằng phần mềm.
Còn các bit M1 và M0 để thiết lập lên chế độ của bộ định thời nói chung

M1	M0	chế độ
1	0	là chế độ 8 bit của bộ định thời
0	1	là chế độ 16 bit của bộ định thời

(*) Trong chế độ 8 bit thì bộ đếm sẽ đếm từ giá trị ban đầu mặc định là 0000 → 00ff. Tức là giá trị đếm được tối đa trong một lần trước khi tràn của bộ định thời là 256 giá trị. Số đếm được lưu trong thanh ghi **TLX**, còn thanh ghi **THx** không có tác dụng gì trong chế độ đếm này.

(*) Trong chế độ 16 bit thì bộ đếm sẽ đếm từ giá trị ban đầu mặc định là 0000 → ffff. Tức là giá trị đếm được tối đa trong một lần trước khi tràn của bộ định thời là 65536 giá trị. Số đếm được lưu trong thanh ghi **TLX**, và thanh ghi **THx**.
Sau mỗi lần bộ đếm tràn thì cờ báo tràn **tfx =1** (ban đầu mặc định là 0)
Bit cờ này có thể xoá bằng lệnh gán **tfx = 0**, hay set cờ lên với lệnh **tfx = 1** ;
Và bộ đếm lại đếm từ giá trị mặc định trong mỗi chế độ.
→ *Khởi động bộ đếm.*
Bộ đếm của vđk sẽ bắt đầu check tín hiệu đầu vào ở chân đầu vào của bộ đếm khi được cho phép bởi lệnh start bộ đếm chính là lệnh
TRx = 1;
Như vậy khi bit TRx= 1 thì giá trị của bộ đếm trong mỗi chế độ sẽ tăng theo giá trị đầu vào phù hợp.
Và bộ đếm sẽ dừng hoạt động khi có lệnh TRx =0;
→ *Các yêu cầu đếm theo tín hiệu đầu vào.*
Bộ đếm của vđk khi đã được thiết lập và cho phép thì vđk sẽ liên tục kiểm tra tín hiệu đầu vào, nếu phù hợp dạng thì sẽ tăng giá trị của bộ đếm.

Dạng tín hiệu theo yêu cầu của bộ đếm là xung vuông mỗi khi có sườn xuống của xung tác động vào chân đầu vào của bộ đếm thì bộ đếm sẽ tăng giá trị lên một đơn vị tức là **đếm theo sườn xuống** (mức 1 chuyển thành mức 0).

Như vậy khi được thiết lập và khởi động thì vđk hoạt động như một bộ đếm các sự kiện bên ngoài mà hoàn toàn không ảnh hưởng tới các hoạt động

khác của vđk.

Có bạn thường nhầm lần giữa bộ đếm với việc các bạn sử dụng một chân nào đó của vđk sau các bạn check các mức tín hiệu khi có sự chuyển mức các bạn tăng biến đếm lên, và các bạn cho đó là bộ đếm.

Điều này chỉ đúng trong trường hợp nếu như vđk chỉ làm mỗi một nhiệm vụ là một bộ đếm. Việc này các bạn làm là lập trình cho vđk làm chức năng là một bộ đếm đơn thuần, nếu như trong trường hợp các bạn còn sử dụng vđk làm nhiều công việc khác không chỉ là một bộ đếm thì phương pháp các bạn sử dụng sẽ gặp khó khăn và bất lợi. Bộ đếm trong của vđk đếm dữ kiện ngoàI được trình bày ở trên trong th này có vài trò và tác dụng rất lớn.

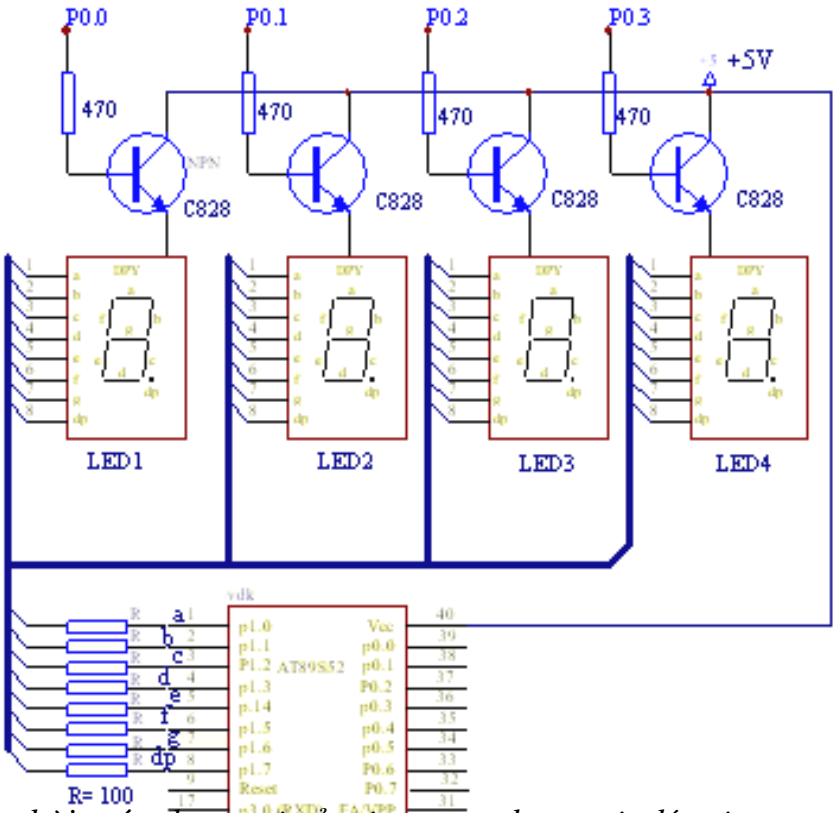
Đó là bộ đếm vẫn hoạt động và hoạt động độc lập với các công việc khác của vđk đang làm.

Khi đó các bạn muốn làm nhiều công việc khác như hiển thì 7 thanh , hay LCD, truyền thông cảnh báo,... các bạn chỉ cần sử dụng các kĩ năng thao tác với các dữ liệu đếm được lưu trong các thanh ghi tương ứng với chế độ đếm của bộ định thời.

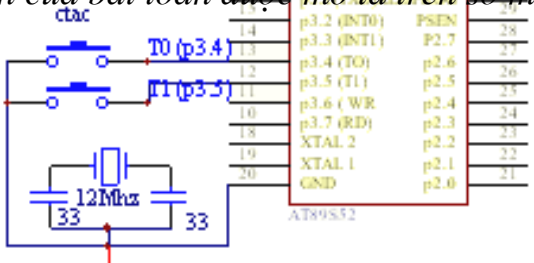
7.3. Bài toán lập trình minh họa

Với bài toán đếm sản phẩm ta có thể mô hình hoá bài toán trở lên đơn giản như sau: đó là đếm số lần các bạn ấn một phím ấn và hiển thị số đếm trên led 7 thanh.

Bài toán được thực hiện trên kit thực hành của trung tâm (Kit for training vđk 8051)



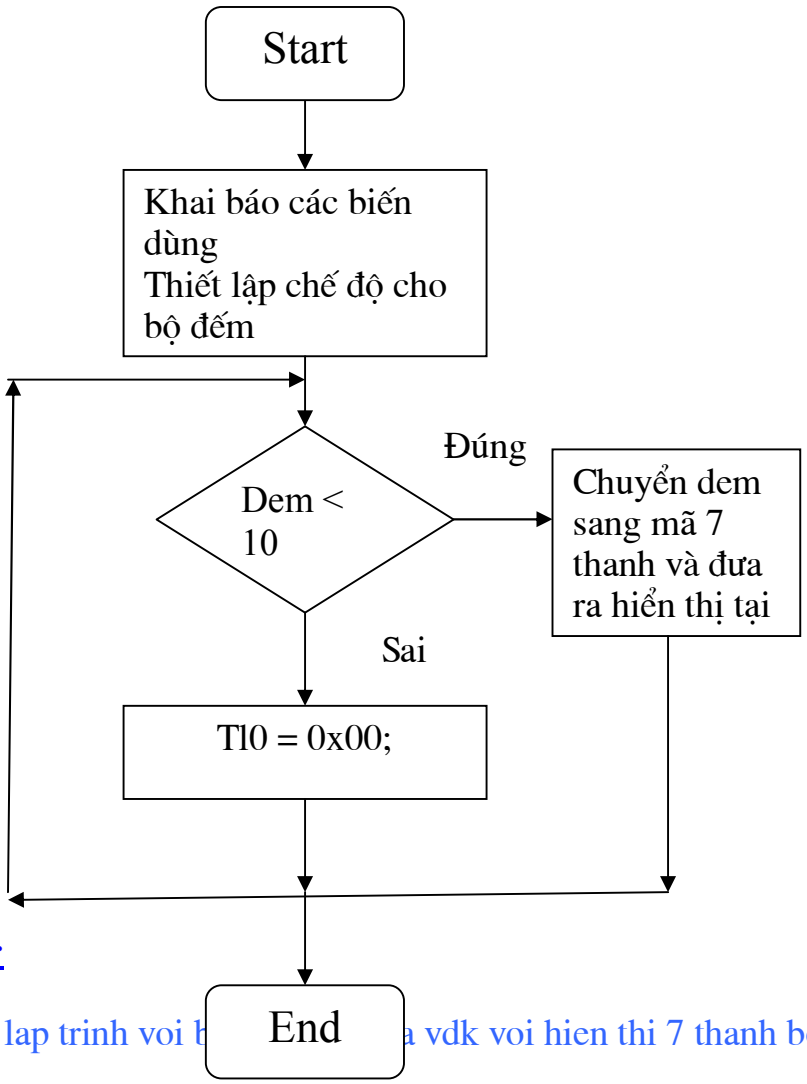
Mô hình của bài toán được mô tả trên sơ mạch nguyên lý trên.



==> Ta xét **bài toán 1:** Thiết lập bộ đếm sử dụng T0 và hiển thị 7 thanh là bộ đếm từ 0 - 9 - 0 :

Trong bài toán này ta chỉ cần sử dụng hiển thị với một led 7 thanh do đó ta không cần phải sử dụng phương pháp quét led 7 thanh.

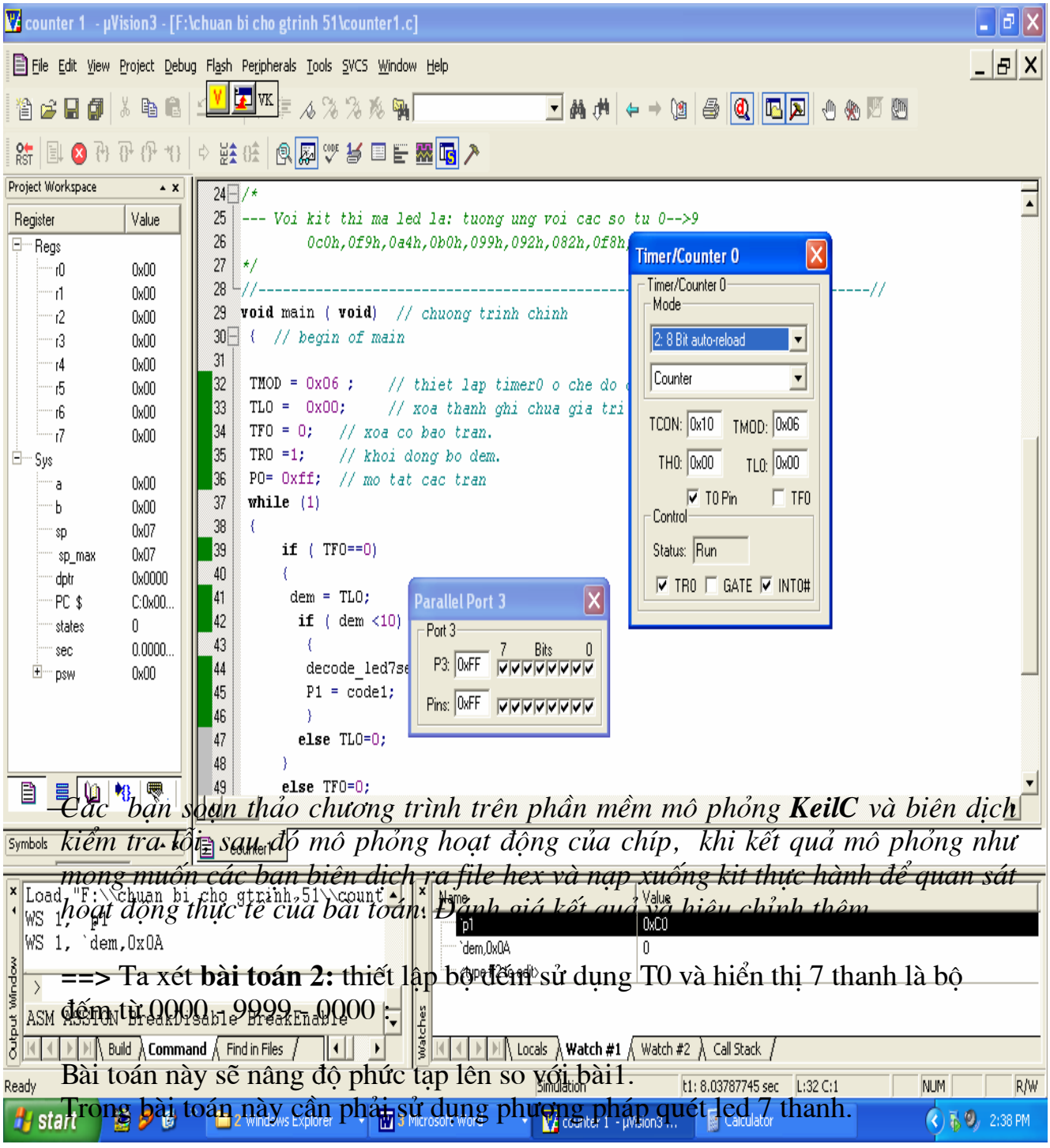
Sơ đồ thuật toán của vd1:



Code for test 1:

```
// Chuong trinh lap trinh voi k a vdk voi hien thi 7 thanh bo dem 0-9-0

#include <REGX51.H>
long  code1, dem;
//=------------Chuong trinh con chuyen so thap phan sang ma led 7 thanh-------//
void decode_led7seg1(unsigned char number1)
// chuyen sang ma led ko co dau cham dp.
{
  switch (number1)
   {
   case 0: {code1=0xc0;break;}
   case 1: {code1=0xf9;break;}
   case 2: {code1=0xa4;break;}
   case 3: {code1=0xb0;break;}
   case 4: {code1=0x99;break;}
```

```
  case 5: {code1=0x92;break;}
  case 6: {code1=0x82;break;}
  case 7: {code1=0xf8;break;}
  case 8: {code1=0x80;break;}
  case 9: {code1=0x90;break;}
   }
}
/*
--- Voi kit thi ma led la: tuong ung voi cac so tu 0-->9
      0c0h,0f9h,0a4h,0b0h,099h,092h,082h,0f8h,080h,090h
*/
//-------------------------------------------------------------------------//
void main ( void)  // chuong trinh chinh
 {  // begin of main

 TMOD = 0x06 ;    // thiet lap timer0 o che do counter 8bit.
 TL0 =        0x00;     // xoa thanh ghi chua gia tri dem duoc
 TF0 = 0;      // xoa co bao tran.
 TR0 =1;       // khoi dong bo dem.
 P0= 0xff;     // mo tat cac tran
 while (1)
 {
          if ( TF0==0)
          {
           dem = TL0;
            if ( dem <10)
              {
          decode_led7seg1(dem);
                  P1 = code1;
                  }
            else TL0=0;
          }
          else TF0=0;

 }  // end of while

 } // end of main
```



*Các bạn soạn thảo chương trình trên phần mềm mô phỏng **KeilC** và biên dịch kiểm tra lỗi, sau đó mô phỏng hoạt động của chíp, khi kết quả mô phỏng như mong muốn các bạn biên dịch ra file hex và nạp xuống kit thực hành để quan sát hoạt động thực tế của bài toán. Đánh giá kết quả và hiệu chỉnh thêm.*

==> Ta xét **bài toán 2:** thiết lập bộ đếm sử dụng T0 và hiển thị 7 thanh là bộ đếm từ 0000 - 9999 - 0000 :

Bài toán này sẽ nâng độ phức tạp lên so với bài1.

Trong bài toán này cần phải sử dụng phương pháp quét led 7 thanh.

Code for test 2:

```
……….
// Chuong trinh lap trinh voi bo dem T1 cua vdk voi hien thi 7 thanh bo dem
0000-9999-0000
#include <AT89x51.H>
long code1, dem;
```

```
#define on  1          // muc logic 1 (+5v)
#define off  0         // muc logic 0 (0v)

// -----  khai bao cac bien cuc bo...
 long y1,y2,y3,y4,x,n;
 long k1,k1_tg,k2,k3,k4,code_led1;
 int j=1,i=1;
 long  n_tg1, n_tg2,bien=0;
/* --- dinh  nghia cac bit dung------*/
 sbit  led1 = P0^3;
 sbit  led2 = P0^2;
 sbit  led3 = P0^1;
 sbit  led4 = P0^0;
//=------------Chuong trinh con chuyen so thap phan sang ma led 7 thanh-------//
void decode_led7seg1(unsigned char number1)
// chuyen sang ma led ko co dau cham dp.
{
  switch (number1)
    {
   case 0: {code1=0xc0;break;}
   case 1: {code1=0xf9;break;}
   case 2: {code1=0xa4;break;}
   case 3: {code1=0xb0;break;}
   case 4: {code1=0x99;break;}
   case 5: {code1=0x92;break;}
   case 6: {code1=0x82;break;}
   case 7: {code1=0xf8;break;}
   case 8: {code1=0x80;break;}
   case 9: {code1=0x90;break;}
    }
}
 //---------------Chuong trinh tao tre-------------------//
  void delay(long tg)   // chuong trinh tre thoi gian.
    {
   long n1;            // khai bao bien cuc bo.
   for (n1=0;n1<tg;n1++)// vong lap time lan.
      {
                        ;            // khong thuc hien gi ca !!!
      }
    }
 //---------------------//
   void nhap(void)
     {
      TH0=0xf0;
```

```
      TL0=0x20;

    }
//-----------Chuong trinh phuc vu ngat timer0-------------//
 void timer0(void) interrupt 1 //Ngat timer 0
   {      //---------vong quet 1 cho led1.
  switch (j)
    {
     case 1:
      {
       j++;
       P1=y1;     // dua du lieu ra led1
       led1=on;
       led2=off;
       led3=off;
       led4=off;
       nhap();
       break;
      }
//---------Vong quet 2 cho led2.-------//
      case 2:
       {
        j++;
        P1=y2;     // dua du lieu ra led2
        led2=on;
        led1=off;
        led3=off;
        led4=off;
        nhap();
        break;
       }
//--------Vong quet 3 cho led3--------//
     case 3:
       {
        j++;
        P1=y3;    // dua du lieu ra led3
        led3=on;
        led2=off;
        led1=off;
        led4=off;
        nhap();
        break;
       }
//--------Vong quet 4 cho led4---------//
```

```
    case 4:
      {
       P1=y4;    // dua du lieu ra led4
       led4=on;
       led2=off;
       led3=off;
       led1=off;
       nhap();
       j=1;
       break;
      }
    } // end of switch
  }
/*
--- Voi kit thi ma led la: tuong ung voi cac so tu 0-->9
      0c0h,0f9h,0a4h,0b0h,099h,092h,082h,0f8h,080h,090h
*/
//---------------------------------------------------------------------------//
void main ( void)  // chuong trinh chinh
 {  // begin of main
    EA= 1;         // cho phep ngat toan cuc.
    TMOD=0x51;  // Timer 0 che do 16 bit not auto reload
    ET0=1;      // Cho phep ngat timer 0
    TH0=0;
    TL0=0;
    nhap();
    TR0=1;
  // ------
 TL1=0;
 TH1=0;
 TR1=1; // start counter
 for(;;)
  {
 if ( TF1==0)
    {
    n=TH1*255+TL1;
 //-------phan tach n ra thanh k1,k2,k3,k4.
     k1=n/1000;
                    decode_led7seg1(k1);
     y1=code1;
                  k2=(n-k1*1000)/100;
                    decode_led7seg1(k2);
     y2=code1;
                  k3=(n-k1*1000-k2*100)/10;
```

```
      decode_led7seg1(k3);
      y3=code1;
                    k4=(n-k1*1000-k2*100-k3*10);
                      decode_led7seg1(k4);
      y4=code1;
                    }
                    else {TF1=0;TH1=0;TL1=0;}
 }  // end of while
 } // end of main
```

==> Ta xét **bài toán 3:** thiết lập bộ đếm sử dụng T0, hiển thị 7 thanh sử dụng ngắt timer1 và hiển thị đồng thời trên LCD bộ đếm sản phẩm trên băng tải từ 0000 - 9999 - 0000 :

Bài toán này ta xây dựng kế thừa của bài toán 2 ở trên. Các bạn phân tích bài toán đặt ra cần phải hiển thị cả sản phẩm đếm được từ bộ counter0 trên LCD, thông qua đây các bạn luyện được nhiều kĩ năng, đặc biệt là phối hợp các công cụ đã được học từ các bài trên trong một ví dụ nhỏ nhưng tổng hợp.
Các bạn tự xây dựng software. Dưới đây là ví dụ về software cho bài toán.

Cũng như trong tinh thần của giáo trình là các bạn hãy tự xây dựng cách giải quyết bài toán sau sử dụng các công cụ mô phỏng KIT để kiểm tra kết quả. Không nên sử dụng ngay code minh hoạ của giáo trình !!!

Code for test 3:

```
#include <AT89x51.H>
#include < string.h>
#define on  1          // muc logic 1 (+5v)
#define off  0         // muc logic 0 (0v)
// -----  khai bao cac bien cuc bo...
 long code1,code2, dem;
 long y1,y2,y3,y4,x,n;
 long k1,k1_tg,k2,k3,k4,code_led1;
 long lc1,lc2,lc3,lc4;
 int j=1;
/* --- dinh  nghia cac bit dung------*/
 sbit  led1 = P0^3;
 sbit  led2 = P0^2;
 sbit  led3 = P0^1;
 sbit  led4 = P0^0;
 sfr LCDdata = 0xA0;// Cong 2 , 8 bit du lieu  P0 co dia chi 0x80, P1 0x90 , P2
0xA0
 sbit BF = 0xA7; // Co ban bit 7 cua p2
 sbit RS = P3^7;
 sbit RW = P3^6;
```

```
 sbit EN = P3^5;
//=------------Chuong trinh con chuyen so thap phan sang ma led 7 thanh-------//
void decode_led7seg1(unsigned char number1)
// chuyen sang ma led ko co dau cham dp.
{
  switch (number1)
    {
   case 0: {code1=0xc0;break;}
   case 1: {code1=0xf9;break;}
   case 2: {code1=0xa4;break;}
   case 3: {code1=0xb0;break;}
   case 4: {code1=0x99;break;}
   case 5: {code1=0x92;break;}
   case 6: {code1=0x82;break;}
   case 7: {code1=0xf8;break;}
   case 8: {code1=0x80;break;}
   case 9: {code1=0x90;break;}
    }
}
/*------chuyen sang ma ASCII cho LCD--------*/
void decode_LCD(unsigned char number2)
{
  switch (number2)
    {
   case 0: {code2='0';break;}
   case 1: {code2='1';break;}
   case 2: {code2='2';break;}
   case 3: {code2='3';break;}
   case 4: {code2='4';break;}
   case 5: {code2='5';break;}
   case 6: {code2='6';break;}
   case 7: {code2='7';break;}
   case 8: {code2='8';break;}
   case 9: {code2='9';break;}
    }
}

/*---------Cac chuong trinh con cua lcd ------*/
//---------------Chuong trinh con kiem tra su san sang cua lcd----------------;
void wait(void)
{
                        long n = 0;
                        EN=1;// Dua chan cho fep len cao
                        RS=0;// Chon thanh ghi lenh
```

```
                    RW=1;//      Doc tu LCD
                    LCDdata=0xff;// Gia tri 0xFF
                    while(BF){n++; if(n>110) break;}// Kiem tra co ban
                    // Neu ban dem n den 100 roi thoat khoi while
                    EN=0;// Dua xung cao xuong thap de chot
                    RW=0;// Doc tu LCD
}
//------------------Chuong trinh con thiet lap lenh cho LCD----------------------;
void LCDcontrol(unsigned char x)
{
                    EN=1;// Dua chan cho fep len cao
                    RS=0;// Chon thanh ghi lenh
                    RW=0;// Ghi len LCD
                    LCDdata=x;// Gia tri x
                    EN=0;// Xung cao xuong thap
                    wait();// Doi LCD san sang
}
//---------------chuong trinh con thiet lap mot so thong so cua lcd---------------;
void LCDinit(void)
{
                    LCDcontrol(0x38);// 2 dong va ma tran 5x7
                    LCDcontrol(0xc0);
                    LCDcontrol(0x0e);// Bat con tro
                    LCDcontrol(0x01);// Xoa man hinh
}
//------------------Chuong trinh con thiet lap dulieu cho LCD----------------------;
void LCDwrite(unsigned char c)
{
                    EN=1;// Cho fep muc cao
                    RS=1;// Ghi du lieu
                    RW=0;// Ghi len LCD
                    LCDdata=c;// Gia tri C
                    EN=0;// Xung cao xuong thap
                    wait();// Cho
}
/*-----------------------------------------*/
  void nhap(void)
    {
     TH1=0xf0;
     TL1=0x20;
    }
//-----------Chuong trinh phuc vu ngat timer0------------//
 void timer1(void) interrupt 3 //Ngat timer 0
   {      //---------vong quet 1 cho led1.
```

```
   switch (j)
     {
      case 1:
       {
       j++;
       P1=y1;     // dua du lieu ra led1
       led1=on;
       led2=off;
       led3=off;
       led4=off;
       nhap();
       break;
       }
//---------Vong quet 2 cho led2.-------//
      case 2:
       {
        j++;
        P1=y2;     // dua du lieu ra led2
        led2=on;
        led1=off;
        led3=off;
        led4=off;
        nhap();
        break;
       }
//--------Vong quet 3 cho led3--------//
     case 3:
       {
        j++;
        P1=y3;    // dua du lieu ra led3
        led3=on;
        led2=off;
        led1=off;
        led4=off;
        nhap();
        break;
       }
//--------Vong quet 4 cho led4---------//
     case 4:
       {
        P1=y4;    // dua du lieu ra led4
        led4=on;
        led2=off;
        led3=off;
```

```
        led1=off;
        nhap();
        j=1;
        break;
      }
    } // end of switch
  }
/*
--- Voi kit thi ma led la: tuong ung voi cac so tu 0-->9
      0c0h,0f9h,0a4h,0b0h,099h,092h,082h,0f8h,080h,090h
*/
//-------------------------------------------------------------------------//
void main ( void)  // chuong trinh chinh
 {  // begin of main
    EA= 1;          // cho phep ngat toan cuc.
    TMOD=0x15;  // Timer 1 che do 16 bit, t0 counter 16 bit
    ET1=1;      // Cho phep ngat timer 0
    nhap();
    TR1=1;
  // ------
 TL0=0;
 TH0=0;
 TR0=1; // start counter
 /*---------------Doan chuong trinh cho LCD----*/

LCDinit();
LCDwrite('S');
LCDwrite('o');
LCDwrite(' ');
LCDwrite('s');
LCDwrite('a');
LCDwrite('n');
LCDwrite(' ');
LCDwrite('p');
LCDwrite('h');
LCDwrite('a');
LCDwrite('m');
LCDwrite(':');

/*-----------------------------------------*/
 while(1)
  {
                    if ( TF0==0)
    {
```

```
                    n=TH0*256+TL0;
//-------phan tach n ra thanh k1,k2,k3,k4.
    k1=n/1000;
                    decode_led7seg1(k1);
    y1=code1;
                    decode_LCD(k1);
                    lc1=code2;
                  k2=(n-k1*1000)/100;
                    decode_led7seg1(k2);
    y2=code1;
                    decode_LCD(k2);
                    lc2=code2;
                  k3=(n-k1*1000-k2*100)/10;
    decode_led7seg1(k3);
    y3=code1;
                    decode_LCD(k3);
                    lc3=code2;
                  k4=(n-k1*1000-k2*100-k3*10);
                    decode_led7seg1(k4);
    y4=code1;
                    decode_LCD(k4);
                    lc4=code2;
                    LCDcontrol(0x8c);
                    LCDwrite(lc1);  // hien thi len LCD so hang nghin.
    LCDwrite(lc2);  // hien thi len LCD so hang tram.
                    LCDwrite(lc3);  // hien thi len LCD so hang chuc.
                    LCDwrite(lc4);  // hien thi len LCD so hang dvi.

                  }
                  else {TF0=0;TH0=0;TL0=0;}
}  // end of while
} // end of main
```

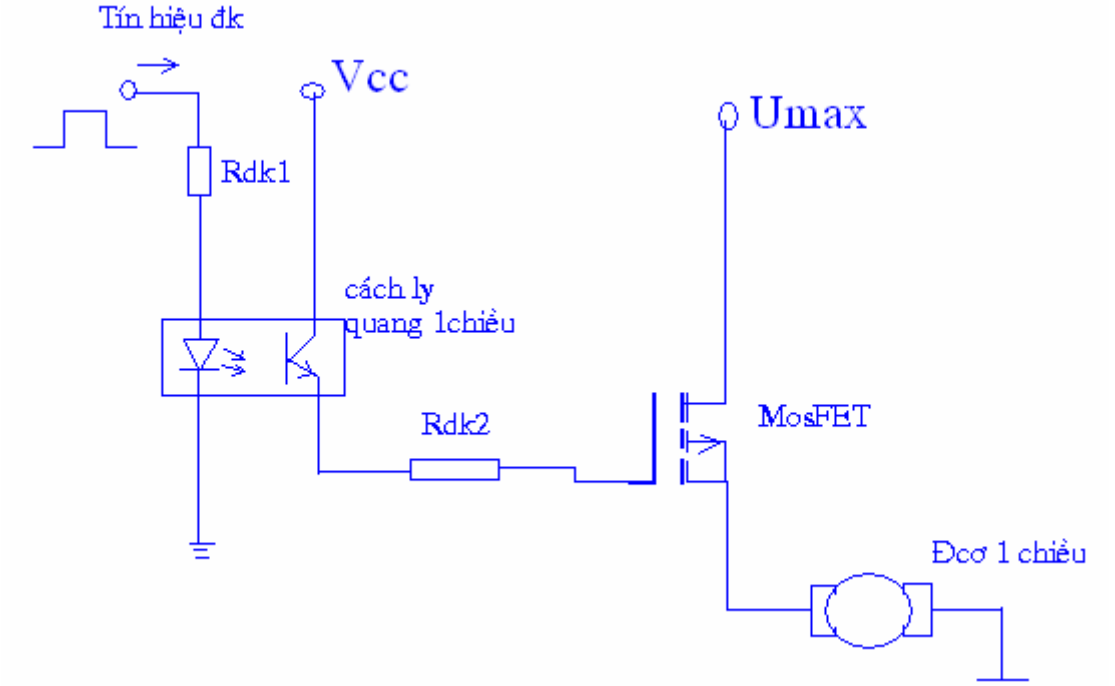
Bài 8: Điều khiển tốc độ động cơ, điều khiển quá trình với băng tải.

8.1.Điều khiển tốc độ động cơ một chiều

Sử dụng bộ định thời cùng ngắt của nó linh hoạt các bạn sẽ có rất nhiều ứng dụng thực tế có ý nghĩa như việc lập trình cho hệ thống mạch của mạch chữ chạy quảng cáo sử dụng led ma trận. Hay điều khiển tốc độ của động cơ một chiều.

Và đây chính là phương pháp xây dựng việc điều chế độ rộng xung (PWM) bằng phần mềm.

Ta có sơ đồ mạch điều khiển động cơ như sau:



Trong sơ đồ trên xung vuông do vi điều khiển tạo ra sẽ đi điều khiển để mở van (tran) và việc đóng mở tran (băm xung) sẽ tạo ra một điện áp Utb đặt lên động cơ. Giá trị điện áp này sẽ ảnh hưởng tới tốc độ quay.

Ta có công thức gần đúng như sau:

Utb = Umax(T1/T)

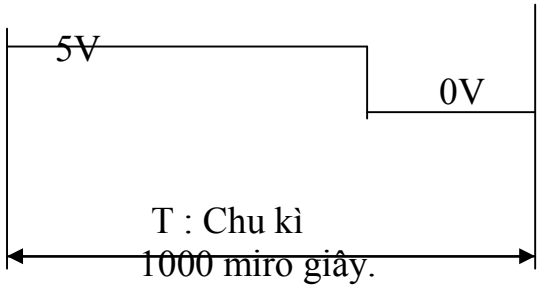
Trong công thức này Umax là giá trị điện áp đặt trên van, còn T1 là thời gian tồn tại mức 1 của xung, T là chu kỳ của xung với hầu hết các van là 1000us (f=1khz).

T và Umax là cố định do đó khi thay đổi T1 trong khoảng từ 0 →T thì sẽ làm cho Utb thay đổi giá trị từ 0 – Umax và Utb càng lớn động cơ quay càng nhanh, động cơ quay lớn nhất khi T1 = T và khi đó Utb = Umax.

chính việc thay đổi T1 sẽ làm thay đổi tốc độ động cơ, và ta thay đổi T1 và tạo ra xung thông qua việc sử dụng bộ định thời và ngắt của nó.

→ ***Đây là phương pháp điều chế độ rộng xung (PWM)***

Tạo xung tần số 1Khz → Chu kì = $1/10^3$ = 0,001 giây= 1 mili giây=1000 uS= 1000 chu kì máy. Với 10 cấp tốc độ, tức là bạn phải tạo ra được xung 10%, 20%, 30%, 40%, …, 90%, 100%. 1 xung như sau:

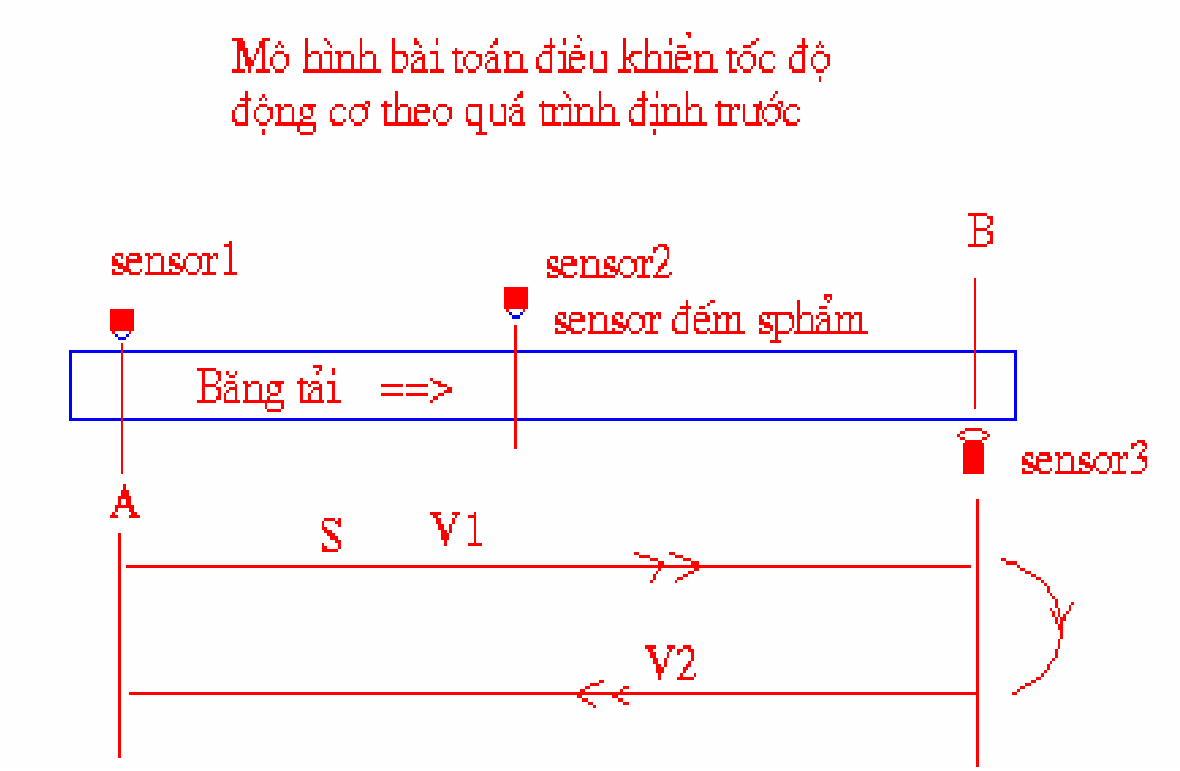


Khoảng thời gian xung kéo dài 5V là T1. Xung 10% tức là T1/ T= 10%=1/10. Xung 20% T2/T=2/10…PWM(Thay đổi độ rộng xung)

8.2. Điều khiển tốc độ động cơ theo quá trình.

// Bai toan 2----------

// bai toan dieu khien dong co theo qua trinh dinh truoc.



Sơ đồ mạch điều khiển tốc độ động cơ một chiều có đảo chiều:

Ta có bảng chân lý của mạch điều khiển động cơ

Bit1(Fet)	Bit2(rơle)	Kết quả
0	0	Quay ngược

0	1	Quay thuận
1	0	Dừng
1	1	Dừng

Trong bài toán này ta xét tốc độ V1= 50% Vmax (tốc độ cực đại) còn V2= 70% Vmax.
Xuất phát từ công thức gần đúng : Utb= Umax(T1/T) Với T= 1000us (chọn f =1 khz).

Dó đó với V1 = 0.5Umax = Umax(T1_1/T) → T1_1= T/2 = 500us.
Với V2= 0.7Umax = Umax(T1_2/T) → T1_2 = 0.7T = 700us.
Từ đây ta sẽ sử dụng timer 1 cho việc điều chế độ rộng xung.
Nên ta có 4 chương trình con nhập giá trị cho timer1 như sau:

```
/*-------------------------------Nhap gia tri voi V1--------*/
void nhapV1-muc1 (void)
 {
   TH1 = 0xfe   ;
    TL1 = 0x0c   ;
 }
//----------------------//
void nhapV1-muc0 (void)
 {
   TH1 = 0xfe   ;
    TL1 = 0x0c   ;
 }
//----------------------//
/*-------------------------------Nhap gia tri voi V2--------*/
void nhapV2_muc1(void)
 {
   TH1 = 0xfd    ;
    TL1 = 0x44    ;
 }
//----------------------//
void nhapV2_muc0(void)
 {
   TH1 = 0xfe    ;
    TL1 = 0xd4    ;
 }
//----------------------//
```

Code for ex2:

```
/*---------------
Bai toan dieu khien toc do dong co theo qua trinh
----------------*/
#include <REGX51.H>
#define on   1;
#define off  0;
```

```
sbit bit1 =P1^0;
sbit bit2 =P1^1;
sbit sensor1 = P3^0;
sbit sensor3 = P3^1;
sbit sensor_dem = P3^4;
sbit start = P3^5; // ctac start
int check1, y1=0,y2=0;
/*-------------cac chuong trinh tao toc do ----------*/
/*----------------------Nhap gia tri voi V1--------*/
void nhap11 (void)  // tao muc logic 1 cho v1
 {
    TH1 = 0xfe   ;
    TL1 = 0x0c   ;
 }
//-----------------------//
void nhap10 (void)  // tao muc logic 0 cho v1
 {
    TH1 = 0xfe   ;
    TL1 = 0x0c   ;
 }
//-----------------------//
/*----------------------Nhap gia tri voi V2--------*/
void nhap21(void)  // tao muc logic 1 cho v2
 {
    TH1 = 0xfd    ;
    TL1 = 0x44    ;
 }
//-----------------------//
void nhap20(void) // tao muc logic 0 cho v2
 {
    TH1 = 0xfe    ;
    TL1 = 0xd4    ;
 }
void stop (void)
{
  TR1=0; // cam t1
  ET1=0;
  bit1= 1 ;
  bit2= 1 ;
 }
/*---------------Chuong trinh stop gap--------------------*/
 void EX0_int (void) interrupt 0
{
  while (1)
```

```
  {
  stop();
  }
}
/*******------ chuong trinh pvu ngat T1-----******/
void timer1 (void) interrupt 3
{  TF1=0; // xoa co tran
   switch(check1)
    {
                    case 1:   // quay thuan... trai -->phai
                     {
                     /*-----tao muc 1----*/
                     if (y1==0)
                      {
                        y1++;
                        bit1=0  ;       // dk FET
                        bit2=1          ;          // dk Role
                        nhap11();     //  tao tre muc 1voi V1
                        break;
      }
                     /*-----tao muc 0----*/
                      if (y1==1)
                      {
                        y1=0;
                        bit1= 1 ;
                        bit2= 1 ;
                        nhap10(); //  tao tre muc 0 voi V1
                        break;
      }
                     break;
                    }
                    case 2:   // quay nguoc... phai -->trai
                     {
                     /*-----tao muc 1----*/
                     if (y2==0)
                      {
                        y2++;
                        bit1= 0 ;
                        bit2= 0 ;
                        nhap21(); //  tao tre muc 1 voi V2
                        break;
      }
                     /*-----tao muc 0----*/
                      if (y2==1)
```

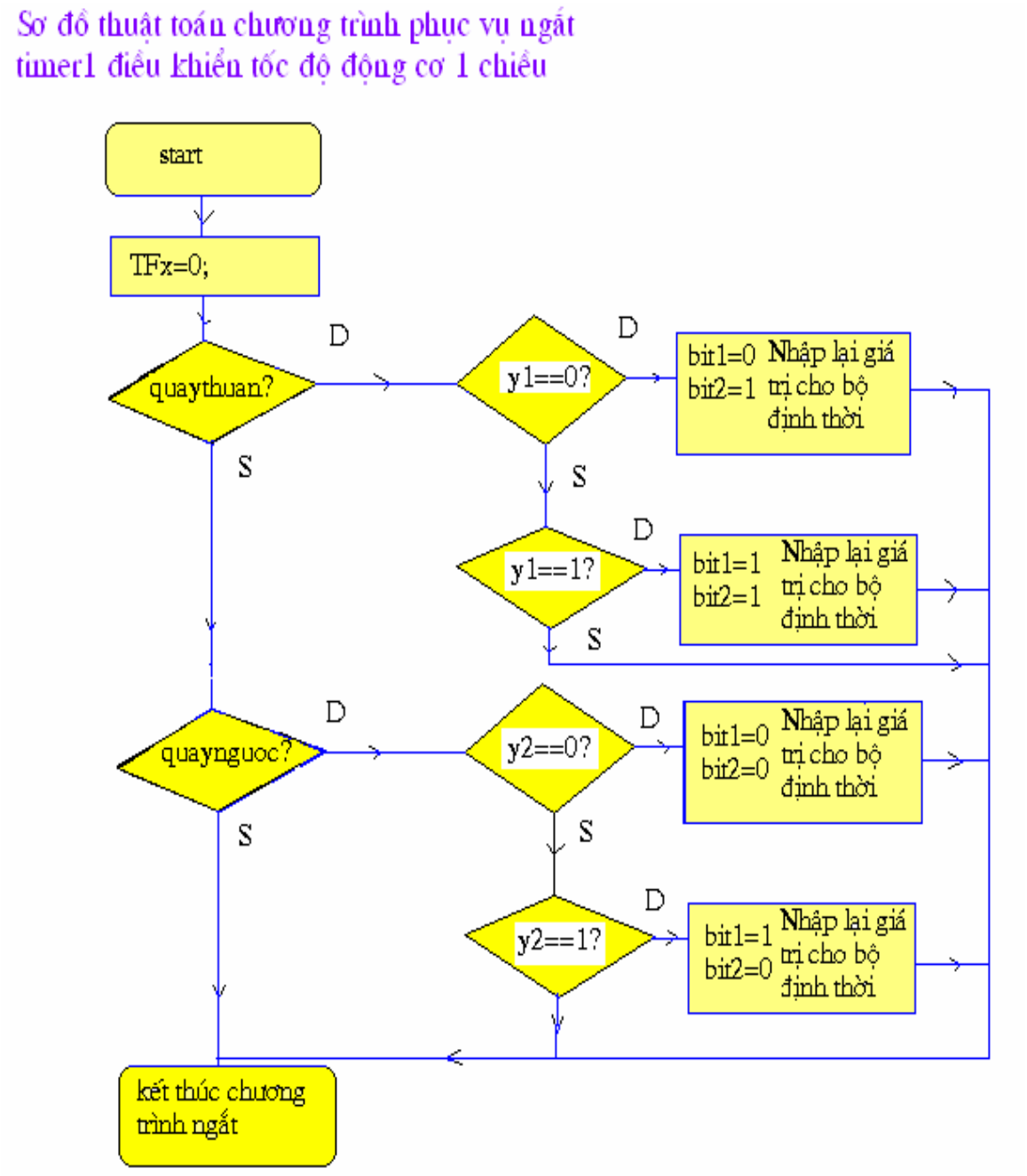
```
                    {
                      y2=0;
                      bit1= 1 ;
                      bit2= 0 ;
                      nhap20(); //  tao tre muc 0 voi V2
                      break;
      }
                     break;
                    }

                   }//  end of switch
} //  end of main
/*------------Chuong trinh chinh --------------*/
void main(void)
{
  EA=1;
  ET1=1;
  EX0=1 ; // cho phep ngat ngoai 0 dung lam Stop.
  TMOD =0x10;  // timer 1 voi ngat 16 bit
  y1=y2=0; // khoi dong cac bien trang thai
   // bat dau quay dong co
 if ( start==0)
 { // bat dau dong co quay thuan voi V1
  check1=1; // dong co quay thuan
  TR1= 1;
 while(1)
  {
                      if(sensor3==0)
                    {
                     stop();   // dung dong co lai
                      check1=2; // de dong co quay nguoc
                      ET1=1;
                      TR1=1;    // dong co bat dau quay nguoc
                    }
                    if(sensor1==0)
                    {
                     stop();   // dung dong co lai
                      check1=1; // de dong co quay nguoc
                      ET1=1;
                      TR1=1;    // dong co bat dau quay nguoc
                    }
                  }
 }
}
```

Sơ đồ thuật toán:

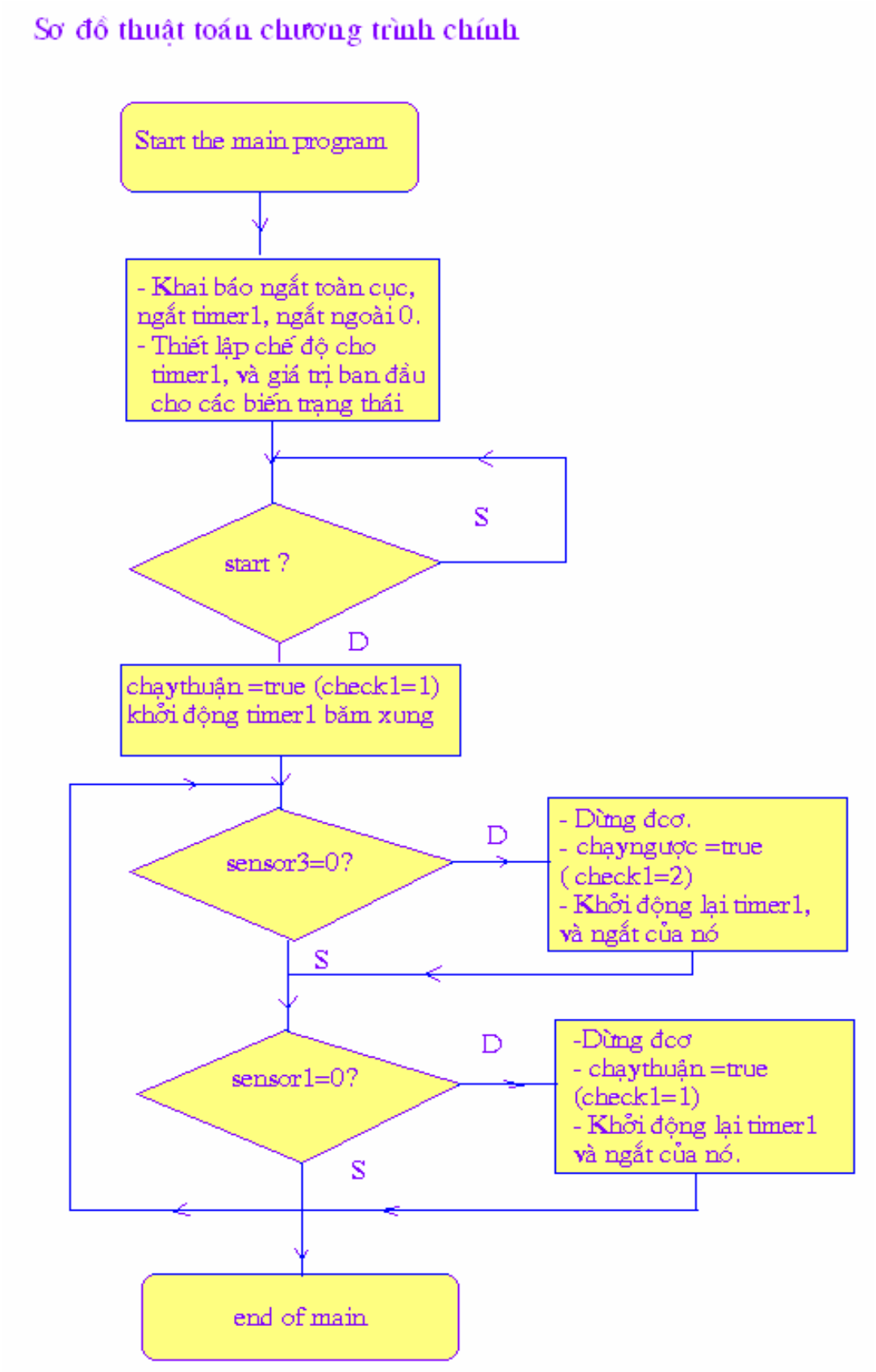
Trong bài toán này ta sử dụng hai sensor1(p3.0) và sensor3 (p3.1)
Một công tắc start (P3.5) và công tắc stop với ngắt ngoài 0.
Yêu cầu của bài toán là đk động cơ với hai cấp tốc độ trên theo quá trình trong hình vẽ.

Sau đây là giải thuật của bài toán:

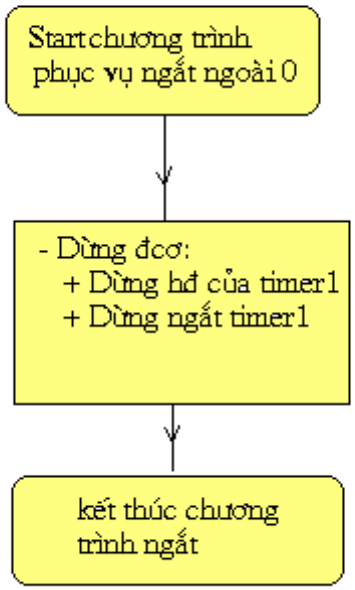
Sơ đồ thuật toán chương trình phục vụ ngắt timer1 điều khiển tốc độ động cơ 1 chiều



Sơ đồ thuật toán chương trình chính



Sơ đồ thuật toán chương trình phục vụ ngắt ngoài 0_ stop đco



→ Hoàn toàn các bạn có thể phát triển bài toán này :
Đó là thêm cả phần đếm sản phẩm chạy trên băng tải và thể hiện kết quả đếm trên led 7 thanh hay trên LCD.