ỦY BÂN NHÂN DÂN TP THỦ ĐỨC
TRƯỜNG TRUNG CẤP NGHỀ ĐÔNG SÀI GÒN

GIÁO TRÌNH

Tên mô đun: Vi điều khiển cơ bản

NGHỀ: ĐIỆN TỬ CÔNG NGHIỆP
TRÌNH ĐỘ TRUNG CẤP

(Ban hành kèm theo Quyết định số: ...../QĐ-TCN ngày ..... tháng ... năm 20... của Hiệu trưởng Trường trung cấp nghề Đông Sài Gòn)

TP Thủ Đức, năm 2023
(Lưu hanh nội bộ)

8051 Cơ Bản

MỤC LỤC

Bài 1: Giới thiệu về vi điều khiển

Chú ý : Đây hoàn toàn là phần lí thuyết, mình đã rút gọn tối đa, do đó các bạn nên đọc hết. Chưa nên thực hành vội vì tôi sẽ hướng dân sau.

1.1 Tiêu chuẩn trong lựa chọn một bộ vi điều khiển

Là khả năng sẵn sàng đáp ứng về số lượng trong hiện tại và tương lai. Đối với một số nhà thiết kế điều này là quan trong hơn cả . Hiện nay, các bộ vi điều khiển 8 bit đứng đầu là họ 8051 có số lương lớn nhất các nhà cung cấp đa dạng (nhiều nguồn). Nhà cung cấp có nghĩa là nhà sản xuất bên cạnh nhà sáng chế của bộ vi điều khiển. Trong trường hợp 8051 thì nhà sáng chế của nó là Intel, nhưng hiện nay có rất nhiều hãng sản xuất nó (cũng như trước kia đã sản xuất).

Các hãng này bao gồm: Intel, Atmel, Philips/signe-tics, AMD, Siemens, Matra và Dallas, Semicndictior.

Bảng địa chỉ của một số hãng sản xuất các thành viên của họ 8051.

Hãng	Địa chỉ Website
Intel	www.intel.com/design/mcs51
Antel	www.atmel.com
Plips/ Signetis	www.semiconductors.philips.com
Siemens	www.sci.siemens.com
Dallas Semiconductor	www.dalsemi.com

8051 là một bộ xử lý 8 bit có nghĩa là CPU chỉ có thể làm việc với 8 bit dữ liệu tại một thời điểm. Dữ liệu lớn hơn 8 bit được chia ra thành các dữ liệu 8 bit để cho xử lý. 8051 có tất cả 4 cổng vào - ra I/O mỗi cổng rộng 8 bit. Các nhà sản xuất đã cho xuất xưởng chỉ với 4K byte ROM trên chíp.

Bảng các đặc tính của 8051 đầu tiên.

Đặc tính	Số lượng
ROM trên chíp	4K byte
RAM	128 byte
Bộ định thời	2
Các chân vào - ra	32
Cổng nối tiếp	1
Nguồn ngắt	6

1.2.Vi điều khiển và vi xử lí:

Xin nhắc đến cái máy tính của bạn, con chíp Intel hay ADM của bạn là 1 bộ vi xử lí, nó không có RAM, ROM,cổng IO và các thiết bị ngoại vi on Chip. Còn vi điều khiển chứa 1 bộ vi xử lí và RAM,ROM, cổng IO, và có thể có các thiết bị ngoại vi.

1.3. Sơ đồ chân vi điều khiển 8051:

Là IC đóng vỏ dạng DIP có 40 chân, mỗi chân có một kí hiệu tên và có các chức năng như sau:

Chân 40: nối với nguồn nuôI +5V.

Chân 20: nối với đất(Mass, GND).

Chân 29 (PSEN)(program store enable) là tín hiệu điều khiển xuất ra của 8051, nó cho phép chọn bộ nhớ ngoài và được nối chung với chân của OE (Outout Enable) của EPROM ngoài để cho phép đọc các byte của chương trình. Các xung tín hiệu PSEN hạ thấp trong suốt thời gian thi hành lệnh. Những mã nhị phân của chương trình được

đọc từ EPROM đi qua bus dữ liệu và được chốt vào thanh ghi lệnh của 8051 bởi mã lệnh.(chú ý việc đọc ở đây là đọc các lệnh (khác với đọc dữ liệu), khi đó VXL chỉ đọc các bit opcode của lệnh và đưa chúng vào hàng đợi lệnh thông qua các Bus địa chỉ và dữ liệu)

Chân 30 (ALE : Adress Latch Enable) là tín hiệu điều khiển xuất ra của 8051, nó cho phép phân kênh bus địa chỉ và bus dữ liệu của Port 0.

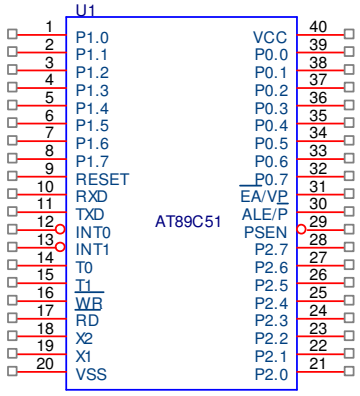
Chân 31 (EA : Eternal Acess) được đưa xuống thấp cho phép chọn bộ nhớ mã ngoàI đối với 8051.

Đối với 8051 thì : EA = 5V : Chọn ROM nội. EA = 0V : Chọn ROM ngoại.

32 chân còn lại chia làm 4 cổng vào ra:

Vào ra tức là có thể dùng chân đó để đọc mức logic (0;1 tương ứng với 0V ; 5V)vào hay xuất mức logic ra(0;1)

P0 từ chân 39 → 32 tương ứng là các chân P0_0 → P0_7
P1 từ chân 1 → 8 tương ứng là các chân P1_0 → P1_7
P2 từ chân 21 →28 tương ứng là các chân P2_0 → P2_7
P3 từ chân 10 → 17 tương ứng là các chân P3_0 → P3_7



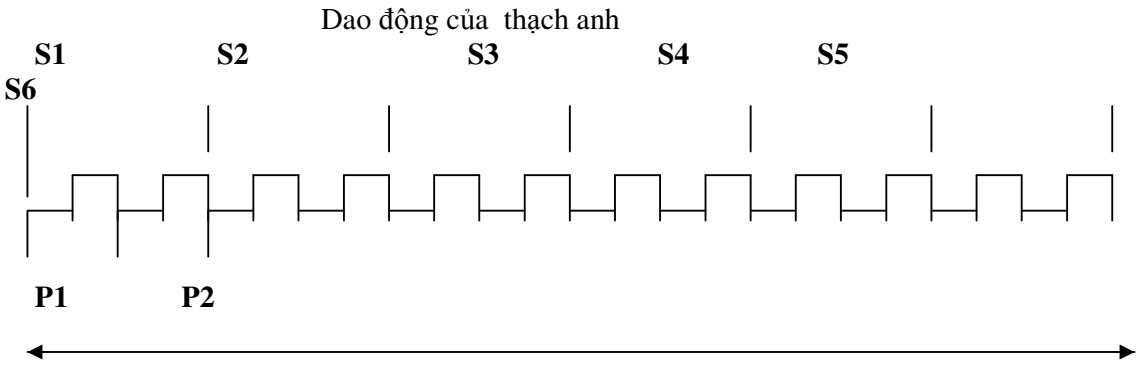
P0	P1	P2	P3	Port's Bit
P0.0	P1.0	P2.0	P3.0	D0
P0.1	P1.1	P2.1	P3.1	D1
P0.2	P1.2	P2.2	P3.2	D2
P0.3	P1.3	P2.3	P3.3	D3
P0.4	P1.4	P2.4	P3.4	D4
P0.5	P1.5	P2.5	P3.5	D5
P0.6	P1.6	P2.6	P3.6	D6
P0.7	P1.7	P2.7	P3.7	D7

Riêng cổng 3 có 2 chức năng ở mỗi chân như trên hình vẽ:

P3.0 – RxD : chân nhận dữ liệu nối tiếp khi giao tiếp RS232(Cổng COM).
P3.1 _ TxD : phân truyền dữ liệu nối tiếp khi giao tiếp RS232.
P3.2 _ INTO : interrupt 0 , ngắt ngoài 0.
P3.3 _ INT1: interrupt 1, ngắt ngoài 1.
P3.4 _T0 : Timer0 , đầu vào timer0.
P3.5_T1 : Timer1, đầu vào timer 1.
P3.6_ WR: Write, điều khiển ghi dứ liệu.
P3.7 _RD: Read , điều khiển đọc dữ liệu.

Chân 18, 19 nối với thạch anh tạo thành mạch tạo dao động cho VĐK

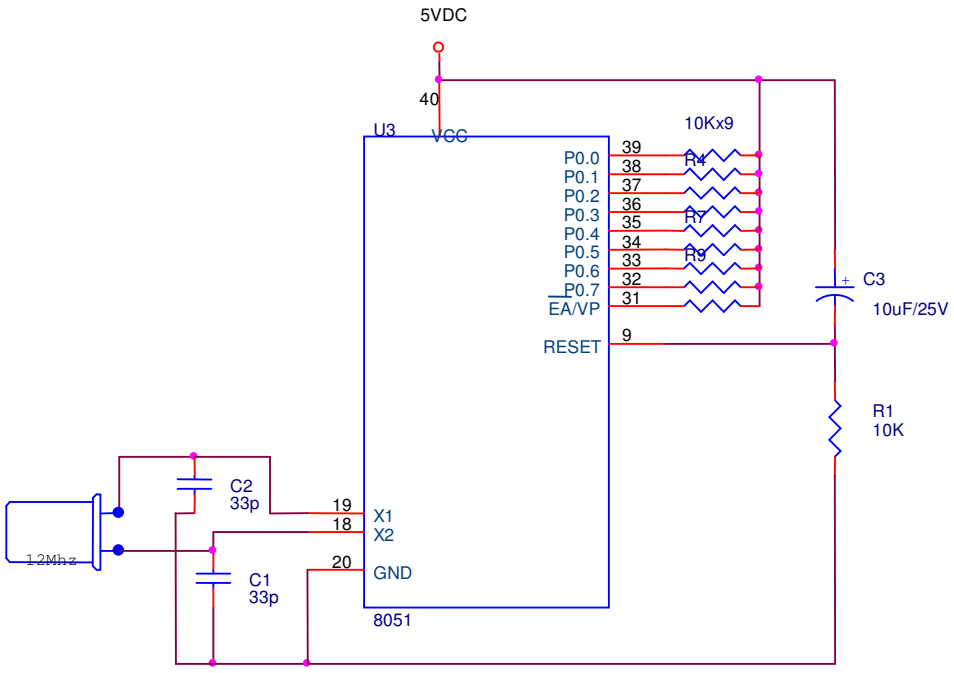
Tần số thạch anh thường dùng trong các ứng dụng là : 11.0592Mhz(giao tiếp với cổng com máy tính) và 12Mhz Tần số tối đa 24Mhz. Tần số càng lớn VĐK xử lí càng nhanh.



Riêng cổng 3 có thêm chức năng như dưới đây

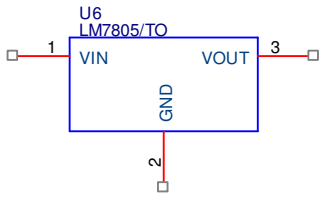
P3.0 – RxD : chân nhận dữ liệu nối tiếp khi giao tiếp RS232(Cổng COM).
P3.1 _ TxD : phần truyền dữ liệu nối tiếp khi giao tiếp RS232.
P3.2 _ INTO : interrupt 0 , ngắt ngoài 0.
P3.3 _ INT1: interrupt 1, ngắt ngoài 1.
P3.4 _T0 : Timer0 , đầu vào timer0.
P3.5_T1 : Timer1, đầu vào timer 1.
P3.6_ WR: Write, điều khiển ghi dứ liệu.
P3.7 _RD: Read , điều khiển đọc dữ liệu.

Chân 18, 19 nối với thạch anh tạo thành mạch tạo dao động cho VĐK
Tần số thạch anh thường được dùng trong các ứng dụng là : 11.0592Mhz(giao tiếp với cổng com máy tính) và 12Mhz
Tần số tối đa 24Mhz. Tần số càng lớn VĐK xử lý càng nhanh.



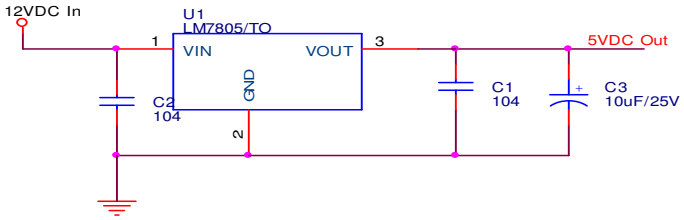
Mạch này chưa có khối nguồn để tạo nguồn 5V các bạn dùng con IC sau:

Sơ đồ chân:

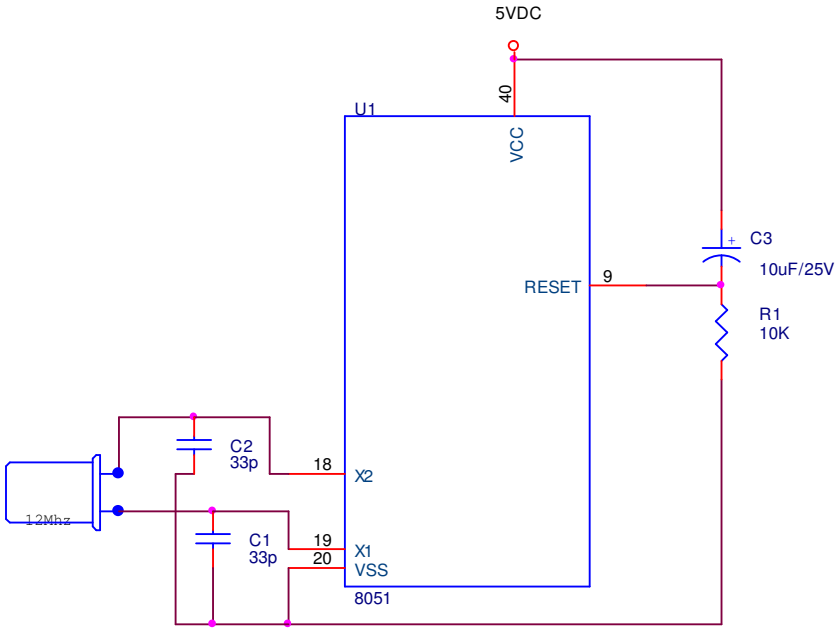


Giới thiệu IC ổn áp 7805 : Đầu vào > 7V đầu ra 5V 500mA. Mạch ổn áp: cần cho VĐK vì nếu nguồn cho VĐK không ổn định thì sẽ treo VĐK, không chạy đúng, hoặc reset liên tục, thậm chí là chết chíp.

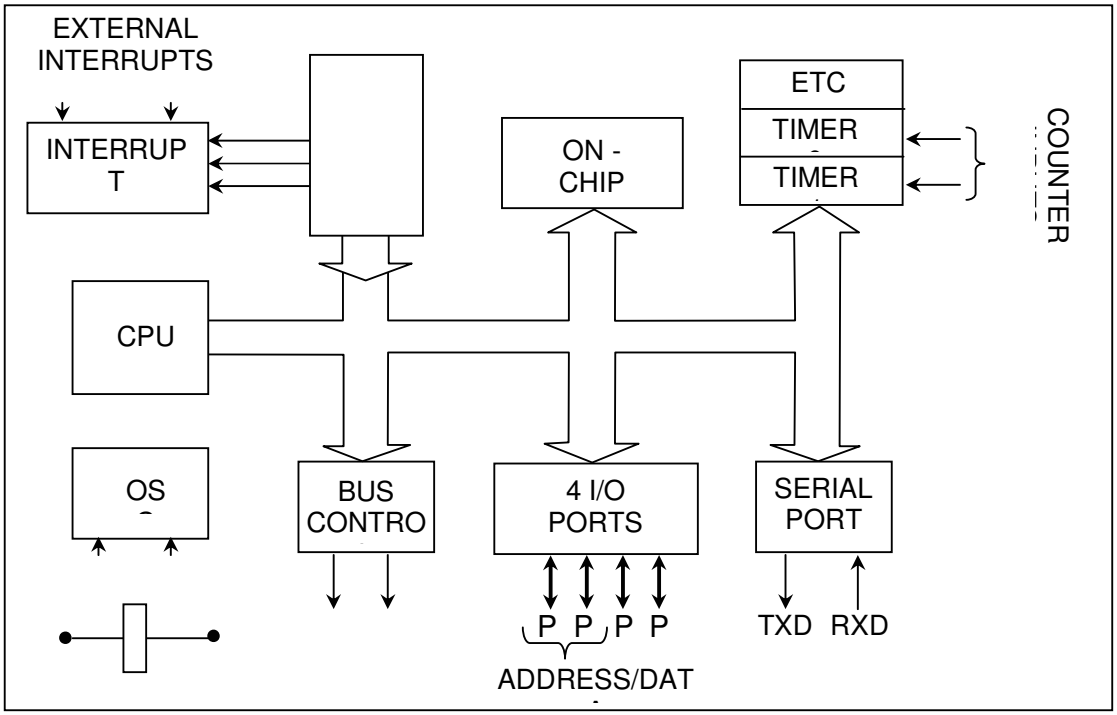
Mạch nguồn



1.4. Mạch vi điều khiển cơ bản



1.5. Cấu trúc vi điều khiển 89c51



Chú ý:Sơ đồ khối bên trong con 8051 có những tài nguyền . Interrupt, Ram, Timer, Serial prort.

1.6.Các thành viên khác của họ 8051:

Có hai bộ vi điều khiển thành viên khác của họ 8051 là 8052 và 8031.

Bộ vi điều khiển 8052: 8052 có tất cả các đặc tính chuẩn của 8051 ngoài ra nó có thêm 128 byte RAM và một bộ định thời nữa. Hay nói cách khác là 8052 có 256 byte RAM và 3 bộ định thời. Nó cũng có 8K byte ROM. Trên chíp thay vì 4K byte như 8051.

Bảng : So sánh các đặc tính của các thành viên họ 8051.

Đặc tính	**8051**	**8052**
ROM trên chip	**4K byte**	**8K byte**
RAM	**128 byte**	**256 byte**
Bộ định thời	**2**	**3**
Cổng nối tiếp	**1**	**1**
Nguồn ngắt	**6**	**8**

Do vậy tất cả mọi chương trình viết cho 8051 đều chạy trên 8052 nhưng điều ngược lại là không đúng.Đặc biệt : Một nhà sản xuất chính của họ 8051 khác nữa là Philips Corporation. Hàng này có một dải lựa chọn dộng lớn cho các bộ vi điều khiển

họ 8051. Nhiều sản phẩm của hãng đã có kèm theo các đặc tính như các bộ chuyển đổi ADC, DAC, chân PWM, cổng I/0 mở rộng .

Update sản phẩm 8051 mới tại các trang web của các nhà sản xuất địa chỉ đã có ở phần giới thiệu. Chủ yếu:www.atmel.com

1.7.Ram nội và các thanh ghi

Cỏc cú địa chỉ và FFH

F0	**F7**	**F6**	**F5**	**F4**	**F3**	**F2**	**F1**	**F0**
E0	**E7**	**E6**	**E5**	**E4**	**E3**	**E2**	**E1**	**E0**
D0	**D7**	**D6**	**6D**	**6C**	**6B**	**6A**	**69**	**68**
B8	**-**	**-**	**-**	**BC**	**BB**	**BA**	**B9**	**B8**
B0	**B7**	**B6**	**B5**	**B4**	**B3**	**B2**	**B1**	**B0**
A8	**AF**	**AE**	**AD**	**AC**	**AB**	**AA**	**A9**	**A8**
A0	**A7**	**A6**	**A5**	**A4**	**A3**	**A2**	**A1**	**A0**
99	**Không định địa chỉ từng bít**							
98	**9F**	**9E**	**9D**	**9C**	**9B**	**9A**	**99**	**98**
90	**97**	**96**	**95**	**94**	**93**	**92**	**91**	**90**
8D	**Không định địa chỉ từng bít**							
8C	**Không định địa chỉ từng bít**							
8B	**Không định địa chỉ từng bít**							
8A	**Không định địa chỉ từng bít**							
89	**Không định địa chỉ từng bít**							
88	**8F**	**8E**	**8D**	**8C**	**8B**	**8A**	**89**	**88**
87	**Không định địa chỉ từng bít**							
83	**Không định địa chỉ từng bít**							
82	**Không định địa chỉ từng bít**							
81	**Không định địa chỉ từng bít**							
80	**87**	**86**	**85**	**84**	**83**	**82**	**81**	**80**
THANH GHI CHỨC NĂNG ĐẶC BIỆT								

thanh ghi SFR nằm giữa 80H các địa chỉ này

ở trên 80H, vì các địa chỉ từ 00 đến 7FH là địa chỉ của bộ nhớ RAM bên trong 8051. Không phải tất cả mọi địa chỉ từ 80H đến FFH đều do SFR sử dụng, nhưng vị trí ngăn nhớ từ 80H đến FFH chưa dùnglà để dữ trữ và lập trình viên 8051 cũng không được sử dụng.

Bảng : chức năng của thanh ghi chức năng đặc biệt SFR

SFR định địa chỉ từng bit(những thanh ghi cần nhớ đối khi lập trình cơ bản C)

Thanh ghi / Bit	Ký hiệu	Chức năng					
TMOD		Chọn model cho bộ định thời 1					
7	GATE	Bít điều khiển cổng. Khi được set lên 1, bộ định thời chỉ hoạt động trong khi INT1 ở mức cao					
6	C/T	Bít chọn chức năng đếm hoặc định thời:					
		1= đếm sự kiện					
		0= định thời trong một khoảng thời gian					
5	M1	Bit chọn chế độ thứ nhất					
4	M0	Bit chọn chế độ thứ 2					
		M1	M0	Chõ đé	Chức năng		
		0	0	0	Chế độ định thời 13 bit		
		0	1	1	Chế độ định thời 16 bit		
		1	0	2	Chế độ tự động nạp lại 8 bit		
		1	1	3	Chế độ định thời chia xẻ		
3	GATE	Bit điều khiển cổng cho bộ định thời 0					
2	C/T	Bit chọn chức năng đếm / định thời cho bộ định thời 0					
1	M1	Bit chọn chế độ thứ nhất cho bộ định thời 0					
0	M0	Bit chọn chế độ thứ 2 cho bộ định thời 0					
TF1 TR1	TF1	TR0	IE1	IT1	IE0	IT0	
Thanh ghi / Bit	Ký hiệu	Chức năng					
TCON		Điều khiển bộ đinh thời					
TCON.7	TF1	Cờ tràn của bộ định thời 1. Cờ này được set bởi phần cứng khi có tràn, được xoá bởi phần mềm, hoặc bởi phần cứng khi bộ vi xử lý trỏ đến trình phục vụ ngắt					
TCON.6	TR1	Bit điều khiển hoạt động của bộ định thời 1. Bit này được set hoặc xoá bởi phần mềm để điều khiển bộ định thời hoạt động hay ngưng					
TCON.5	TF0	Cừ tràn của bộ định thời 0					
TCON.4	TR0	Bit điều khiển hoạt động của bộ định thời 0					
TCON.3	IE1	Cừ ngắt bên ngoài 1 (kích khởi cạnh). Cờ này được set bởi phần cứng khi có cạnh âm (cuống) xuất hiện trên chân INT1, được xoá bởi phần mềm, hoặc phần cứng khi CPU trỏ đến trình phục vụ ngắt					

TCON.2	IT1	Cờ ngắt bên ngoài 1 (kích khởi cạnh hoặc mức). Cờ này được set hoặc xoá bởi phần mềm khi xảy ra cạnh âm hoặc mức thấp tại chân ngắt ngoài
TCON.1	IE0	Cờ ngắt bên ngoài 0 (kích khởi cạnh)
TCON.0	IT0	Cờ ngắt bên ngoài 0 (kích khởi cạnh hoặc mức)

EA	ET2	ES	ET1	EX1	EX0	ET0

IE	Điều khiển các nguồn ngắt (0: không cho phép; 1: cho phép)
IE.7	EA	Cho phép/ không cho phép toàn cục
IE.6	---	Không sử dụng
IE.5	ET2	Cho phép ngắt do bộ định thời 2
IE.4	ES	Cho phép ngắt do port nối tiếp
IE.3	ET1	Cho phép ngắt cho bộ định thời 1
IE.2	EX1	Cho phép ngắt từ bên ngoài (ngắt ngoài 1)
IE.1	EX0	Cho phép ngắt từ bên ngoài (ngắt ngoài 0)
IE.0	ET0	Cho phép ngắt do bộ định thời 0

CHÚ Ý

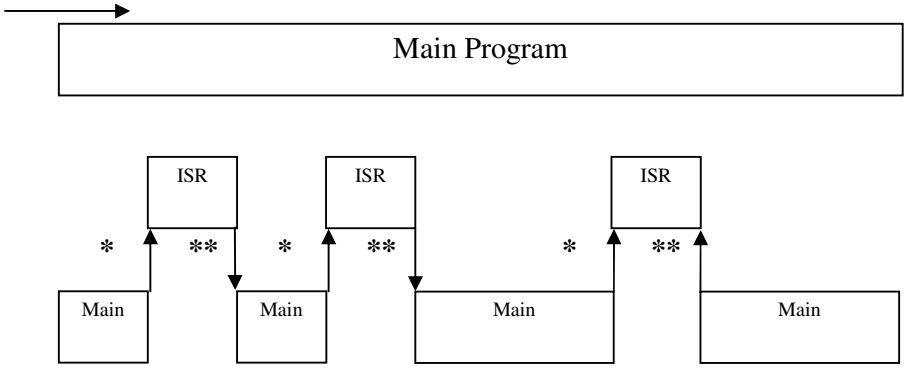
3 thanh ghi này cũng rất cơ bản, nhớ tên thanh ghi, tên các bít trong thanh ghi, chức năng từng thanh ghi và từng bít trong thanh ghi.

1.8.Giới thiệu sơ qua các nguồn 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

Timer

Program excution without interrut



Program excution with interrupt.

Một chương trình chính không có ngắt thì chạy liên tục, còn chương trình có ngắt thì cứ khi nào điều kiện ngắt được đảm bào thì con trỏ sẽ nhảy sang hàm ngắt thực hiện xong hàm ngắt lại quay về đúng chỗ cũ và thực hiện tiếp chương trình. Ta có 1 ví dụ như sau: Bạn đang ăn cơm , có tiếng điện thoại , bạn đạt bát cơm ra nghe điện thoại , nghe xong lại quay về bưng bát cơm lên ăn tiếp. Thì quá trình ăn cơm của bạn là chương trình chính,có điện thoại gọi đến là điều kiện ngắt, bạn ra nghe điện thoại là thực hiện chương trình ngắt(Interrupt Service Rountine),quay về ăn cơm tiếp là tiếp tục thực hiện chương trình chính.

Ngắt đối với người mới học vi điều khiển là rất khó hiểu, vì đa số các tài liệu đều không giải thích ngắt để làm gì. Có nhiều loại ngắt khác nhau nhưng tất cả đều có chung 1 đặc điểm, ngắt dùng cho mục đích đa nhiệm. Đa tức là nhiều, nhiệm tức là nhiệm vụ. Thực hiện nhiều nhiệm vụ .Các bạn nhìn vào tiền trình của hàm main với chương trình có ngắt :

Chương trình chính đang chạy, ngắt xảy ra, thực hiện hàm ngắt rồi quay lại chương trình chính. Chương trình trong vi điều khiển khác với ví dụ ăn cơm nghe điện thoại của tôi ở chỗ , thời gian thực hiện hàm chính là rất lớn,thời gian thực hiện hàm ngắt là rất nhỏ, cho nên thời gian thực thi hàm ngắt không ảnh hưởng nhiều lắm đên chức năng hàm chính. Như vậy trong hàm ngắt các bạn làm 1 việc, trong hàm chính các bạn làm 1 việc

như vậy coi như các bạn làm được 2 việc(đa nhiệm) trong 1 quang thời gian tương đối ngắn cõ mS, chứ thực ra tại 1 thời điểm vi điều khiển chỉ thực thi 1 lệnh.

Ví dụ : Bạn thử nghĩ xem làm thế nào để vừa điều chế xung PWM để điều chỉnh tốc độ động cơ , vừa đọc các cảm biến đầu vào mà tốc độ động cơ phụ thuộc đầu vào cảm biến.

BÀI 2: NGÔN NGỮ LẬP TRÌNH C

2.1.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.

2.2.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 : Kiểu biến Tên biến

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**

VD:
Ngoài ra ,chúng ta có thể định nghĩa biến kiểu bít hay kiểu SFR (special function register)
Bit Kiemtra;
Sfr P10=0x90;
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

2.3.Hàm trong C:

Hàm trong C có cấu trúc như sau
Có 2 loại hàm
Hàm trả lại giá trị:
Cấu trúc: Kiểu giá trị hàm trả lạii Tên hàm (Biến truyền vào hàm)

```
{
// Các lệnh xử lý ở đây
}
```

VD :

```
unsigned char Cong(unsigned char x, unsigned char y)
{
// Các lệnh xử lý ở đây
}
```

Hàm không trả lại giá trị
Cấu trúc:

```
void Tên hàm (Biến truyền vào hàm)
{
// Các câu lệnh xử lý ở đây
}
```

VD dụ :

```
void Cong(unsigned char x, unsigned char y)
{
// Các câu lệnh xử lý ở đây
}
```

Hàm có thể truyền vào biến hoặc không
VD
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
Băng thanh ghi trên ram chọn từ 0 đến 3

2.4.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: >>

2.5.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 giatrị1: { // Các câu lệnh   break; }
  case giatrị2: { // Các câu lệnh   break; }
  case giatrị3: { // Các câu lệnh   break; }
  …
  case giatrịn: { // Các câu lệnh   break; }
  }
```
Giải thích : Tùy vào Biến có giatrị1 thì thực hiện các câu lệnh sau đó tương ứng rồi thoát khỏi cấu trúc nhờ câu lệnh break;

Biến có giatrị2 thì thực hiện các câu lệnh sau đó tương ứng rồi thoát

….

Biến có giatrịn 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 thi thoát khỏi vòng lặp

2.6. Bộ tiền xử lý:

#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.

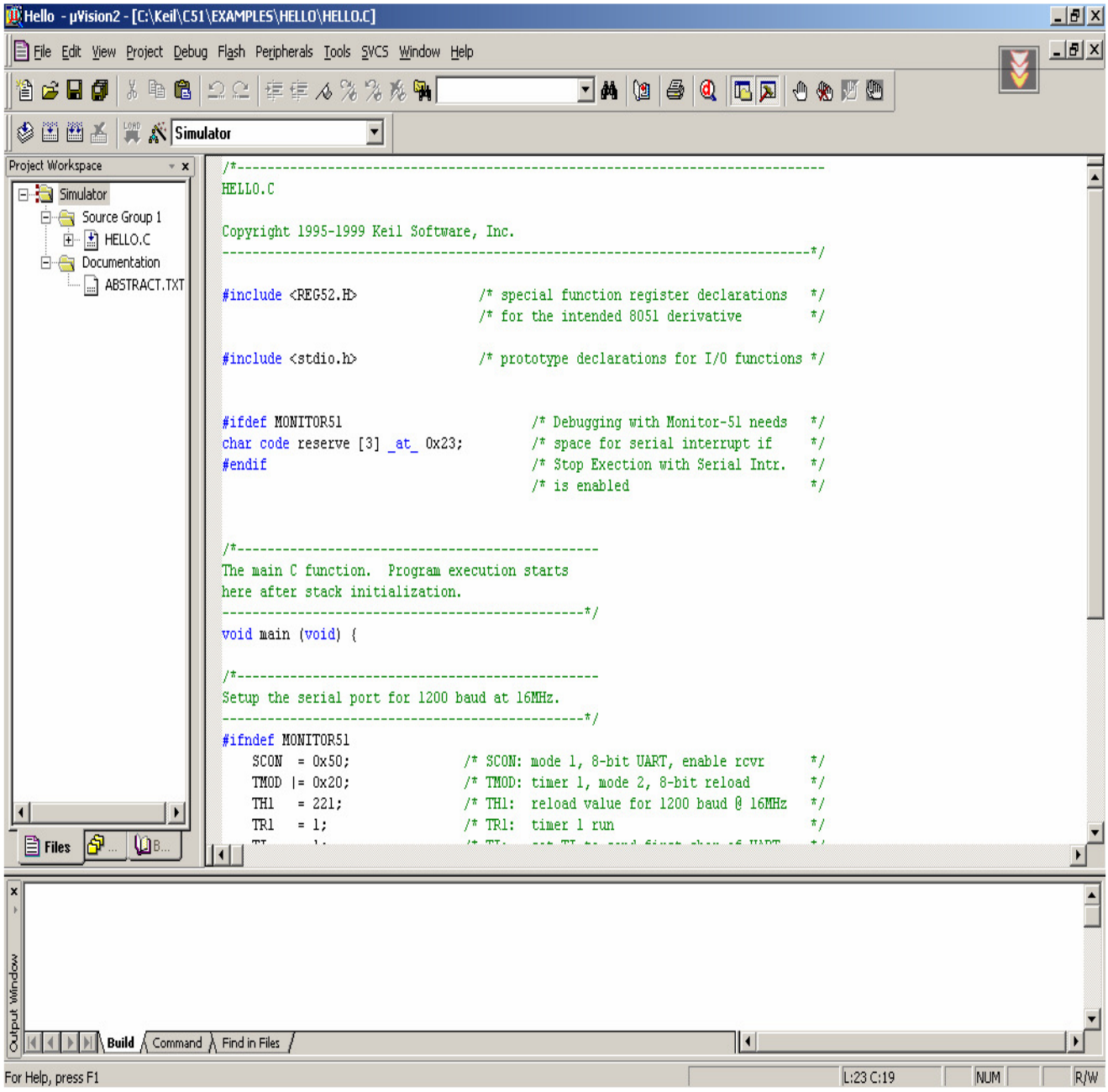
Một số web hay :

www.dientuvietnam.net
www.svbkol.org
www.diendandientu.com
www.atmel.com
www.keil.com
www.iguanalabs.com

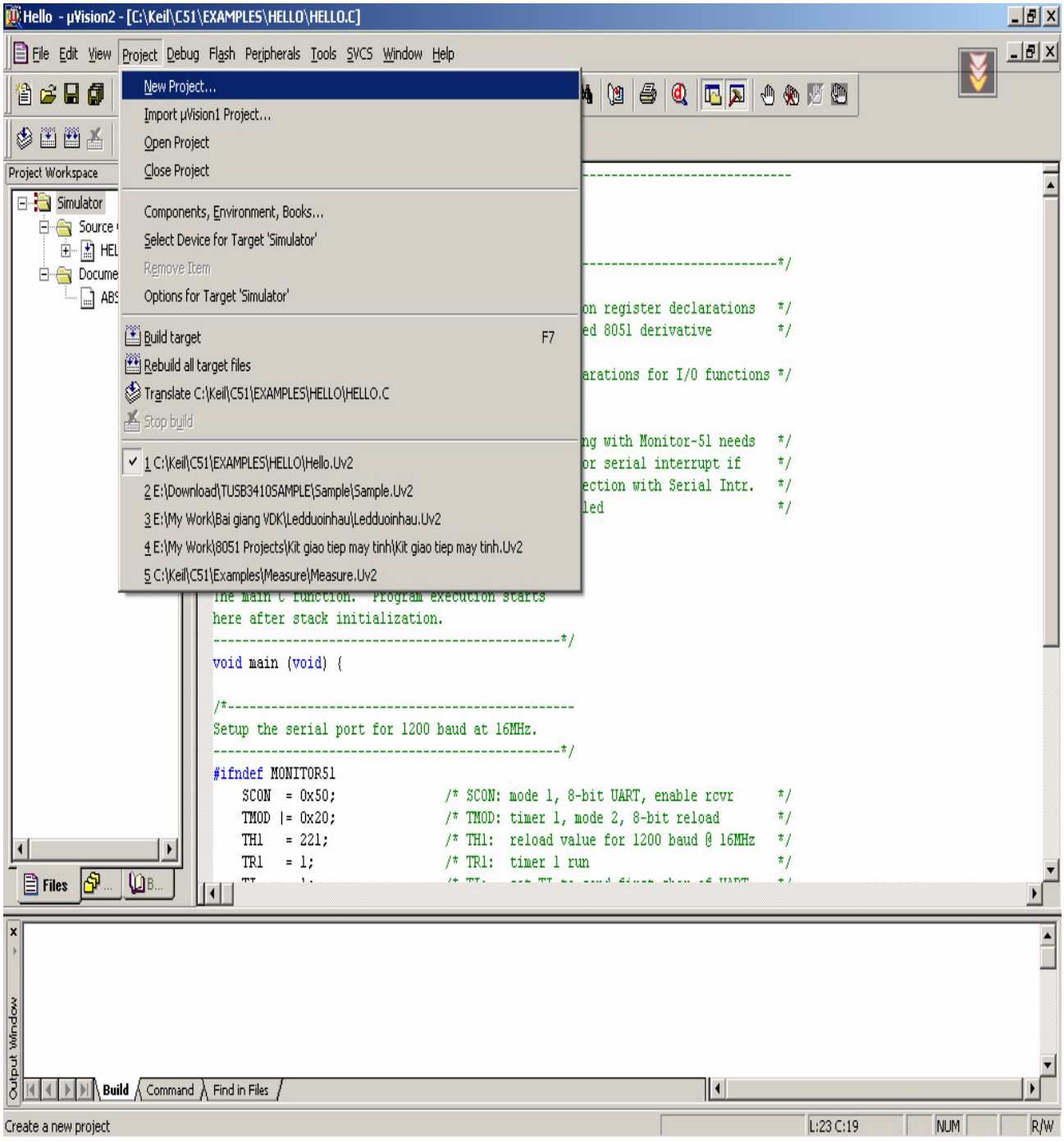
www.microchip.com
www.elechtro-tech-online.com
www.diendansv.hutech.edu.vn
www.ttvnol.com
www.8052.com
www.kmitl.ac.th
www.ftdichip.com

Bài 2 (Tiếp)
Phần 2: Sử dụng Keil C.

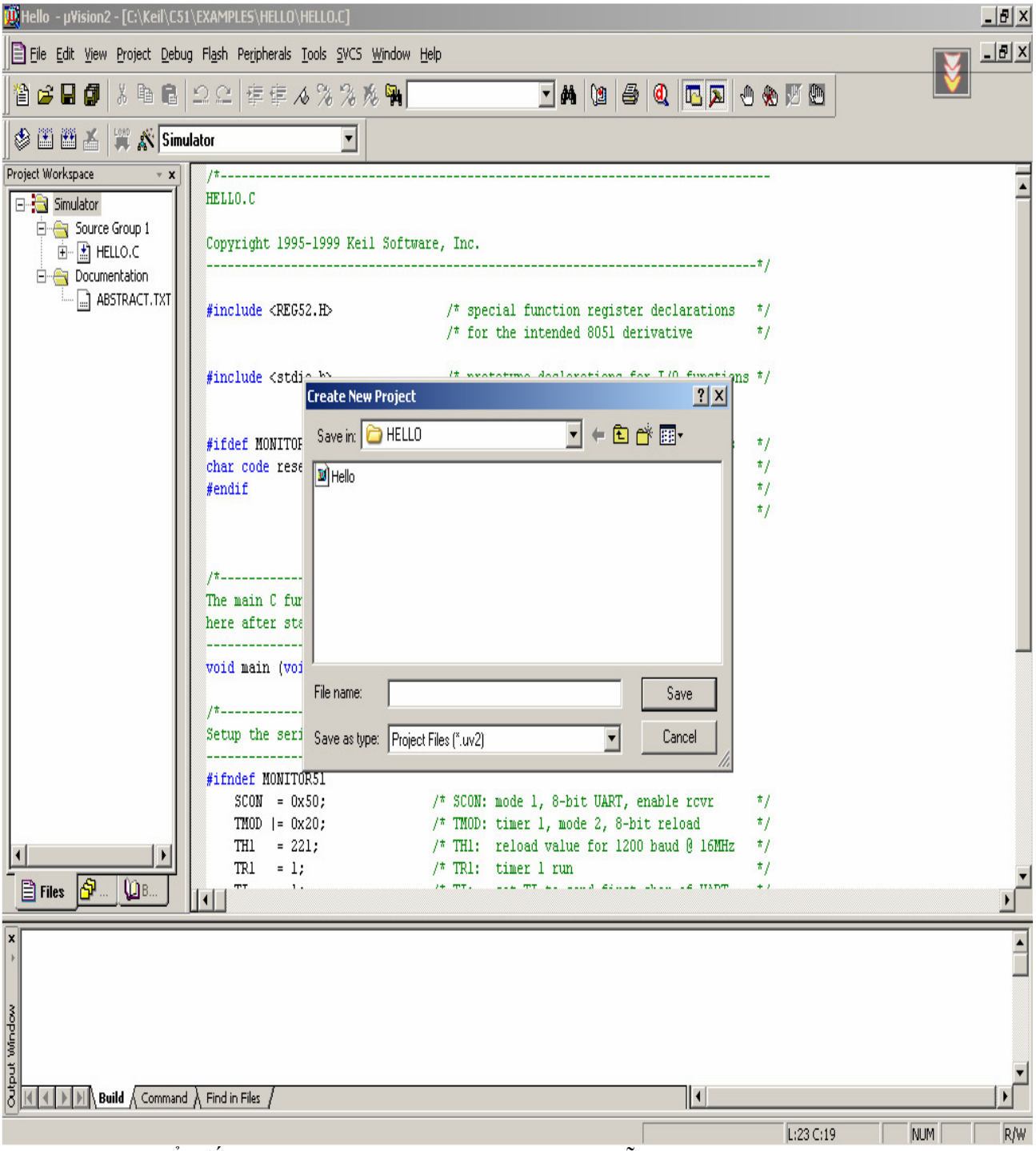
Sau khi cài đặt
1> Khởi tạo cho Project:



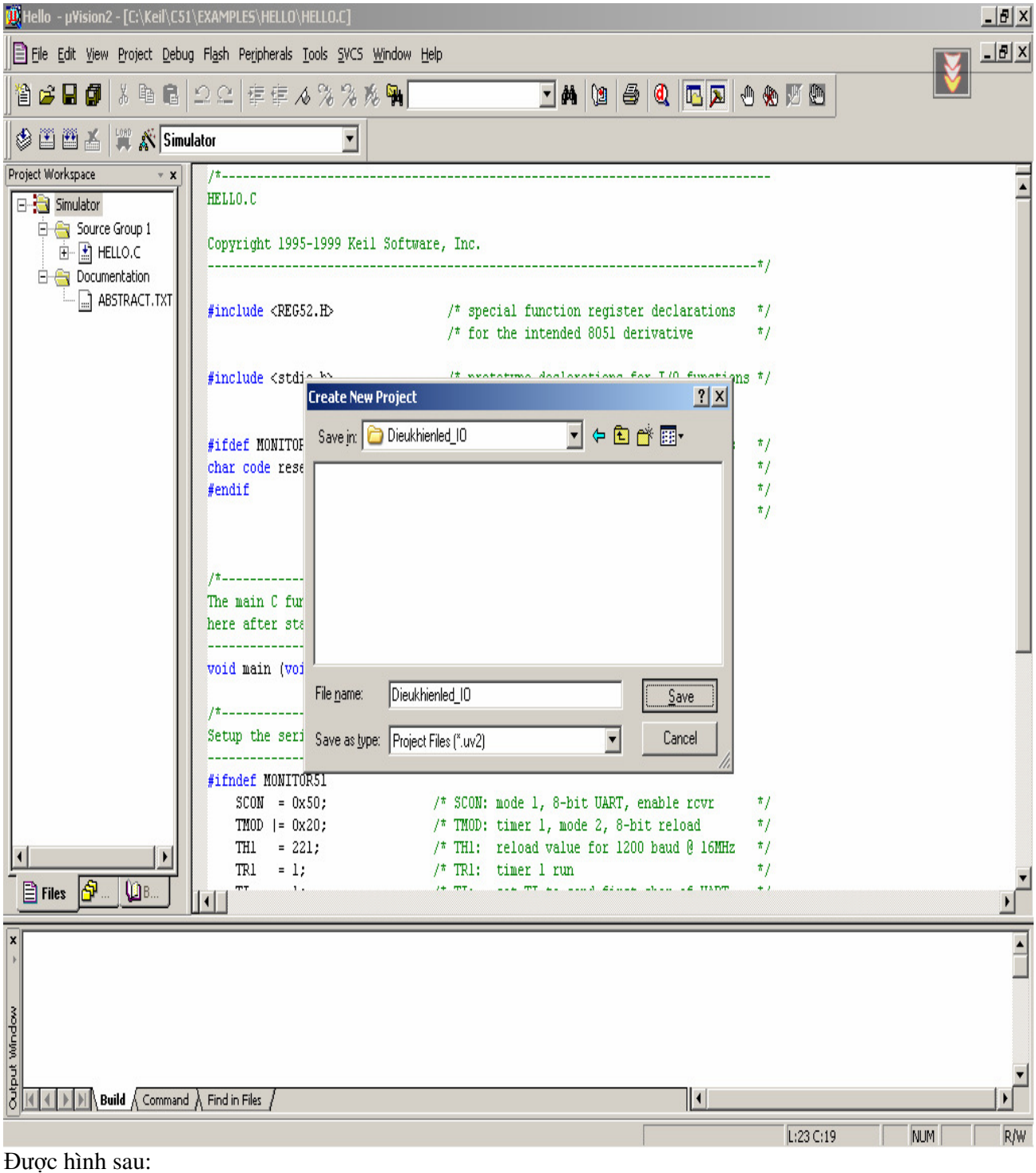
Để tạo 1 project mới chọn Project → New project như sau:



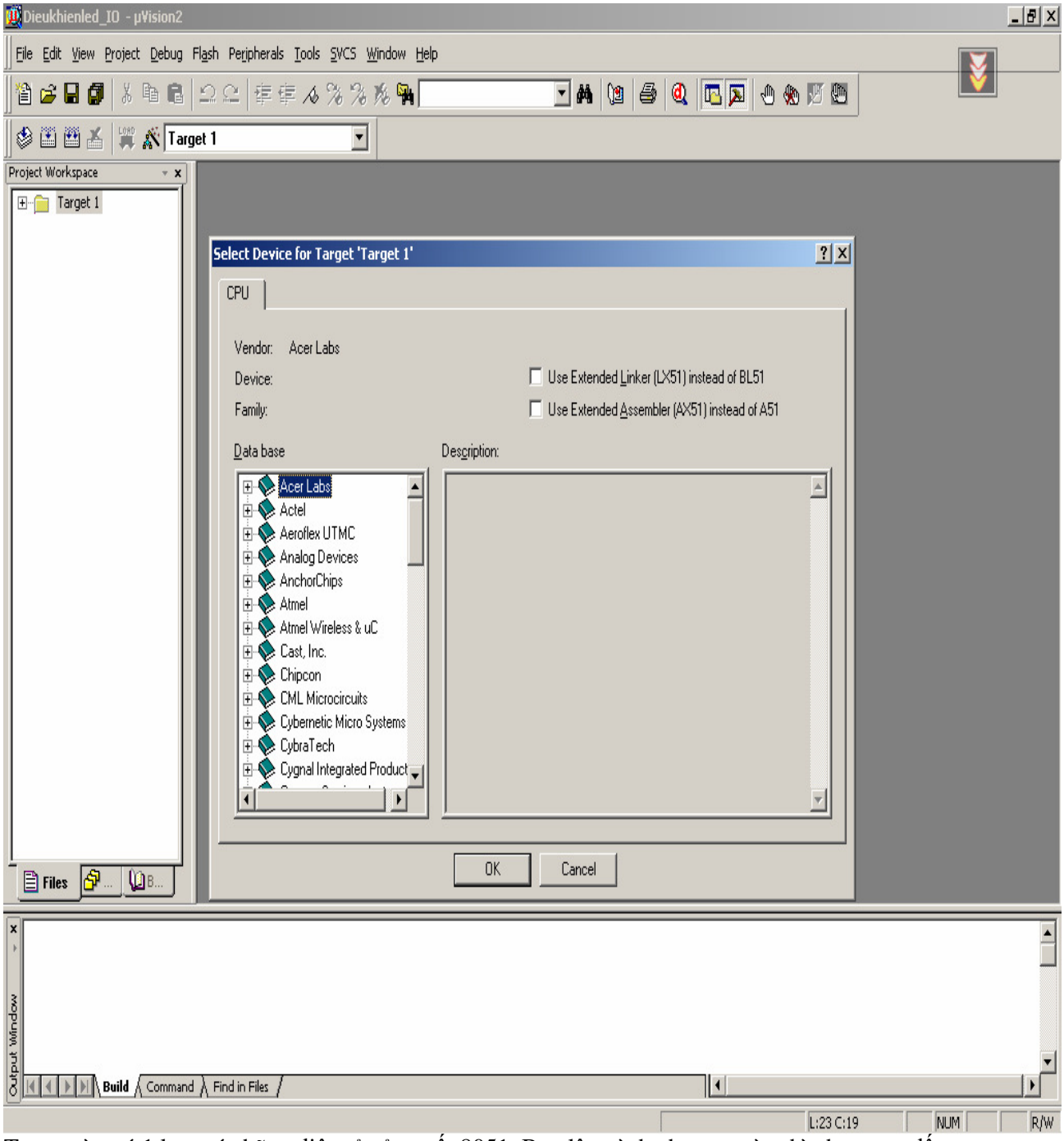
Được hình sau:



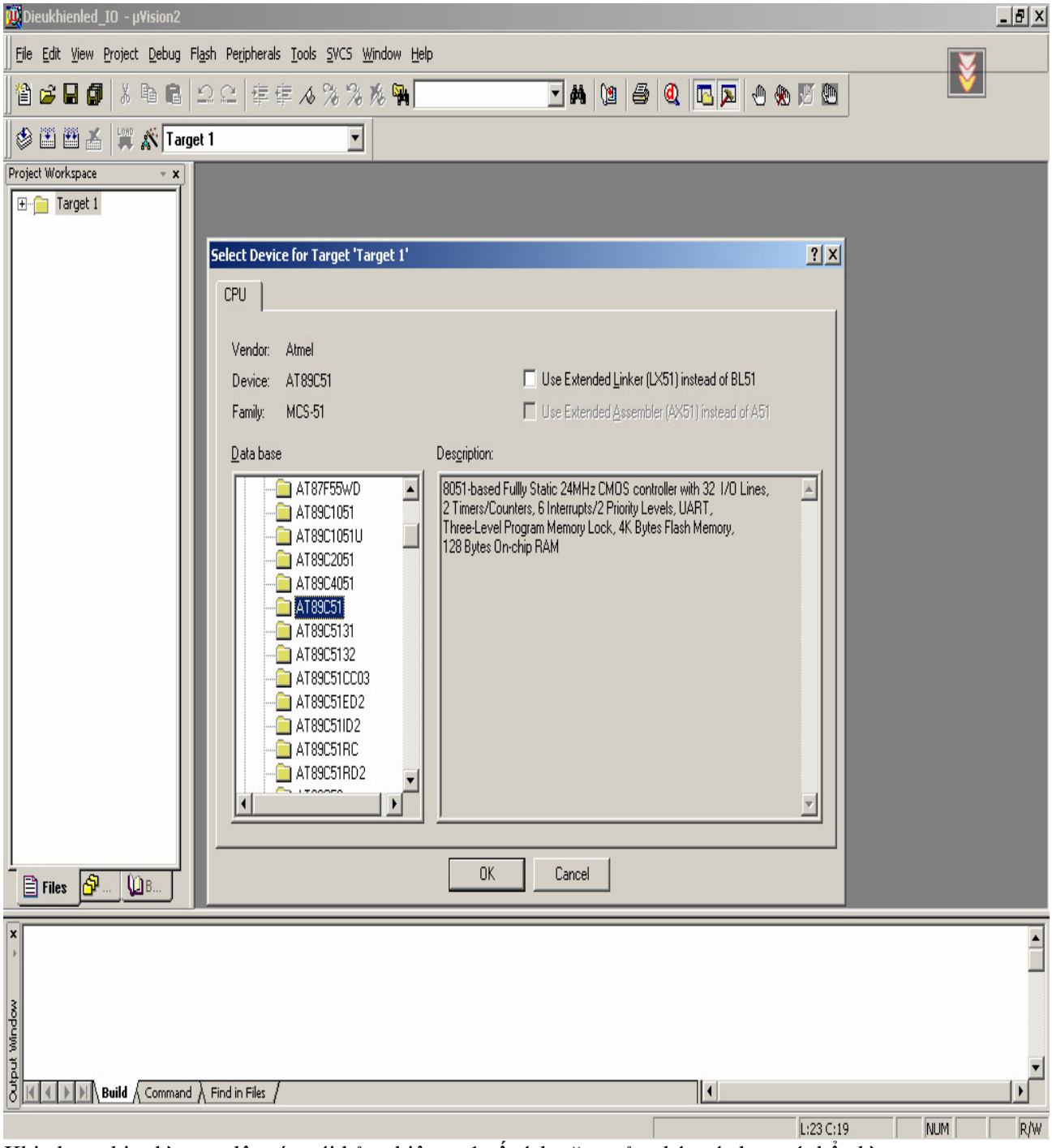
Đánh tên và chuyển đến thư mục bạn lưu project. Bạn nên tạo mỗi một thư mục cho 1 project.
Rồi chọn Save.



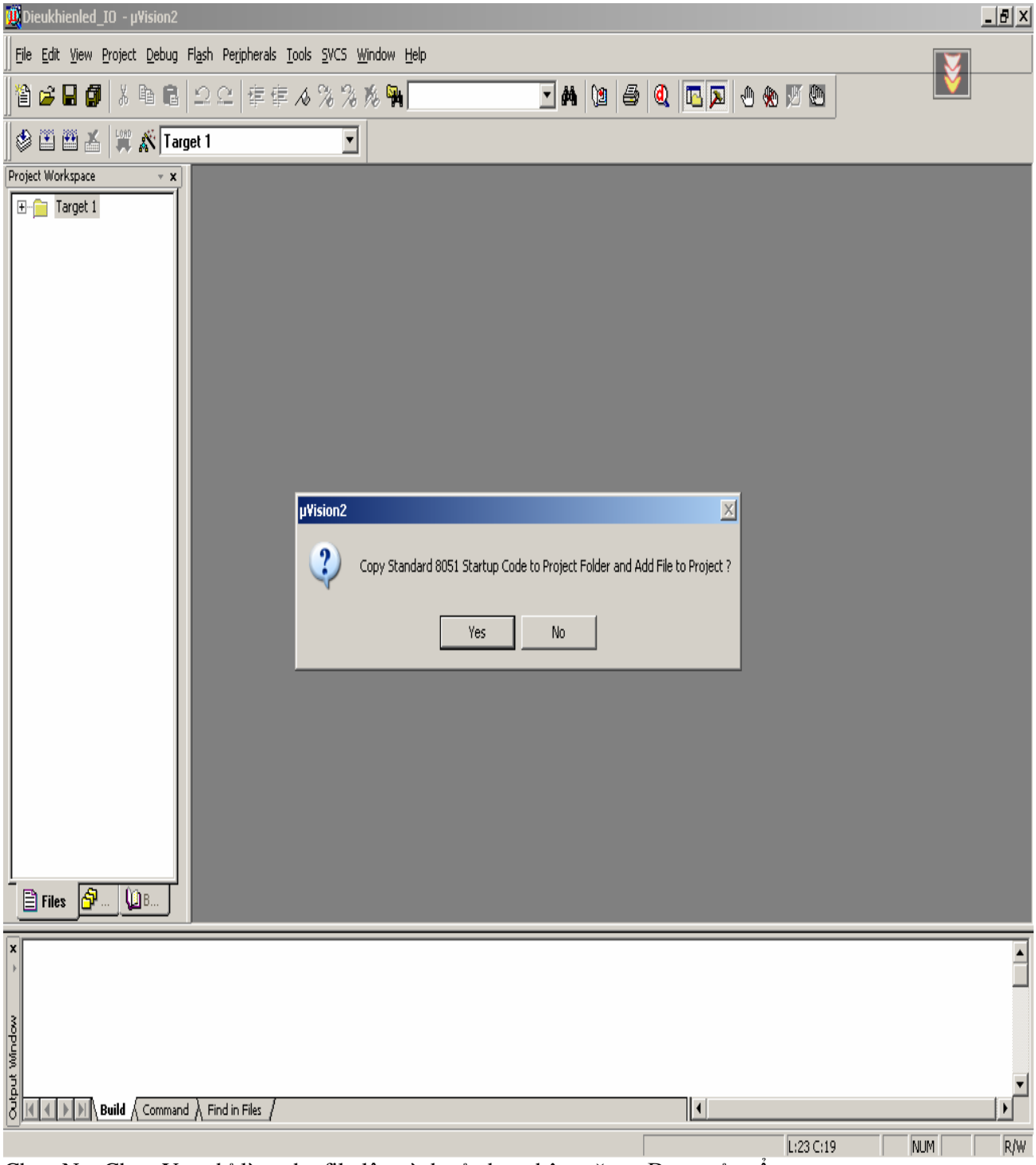
Được hình sau:



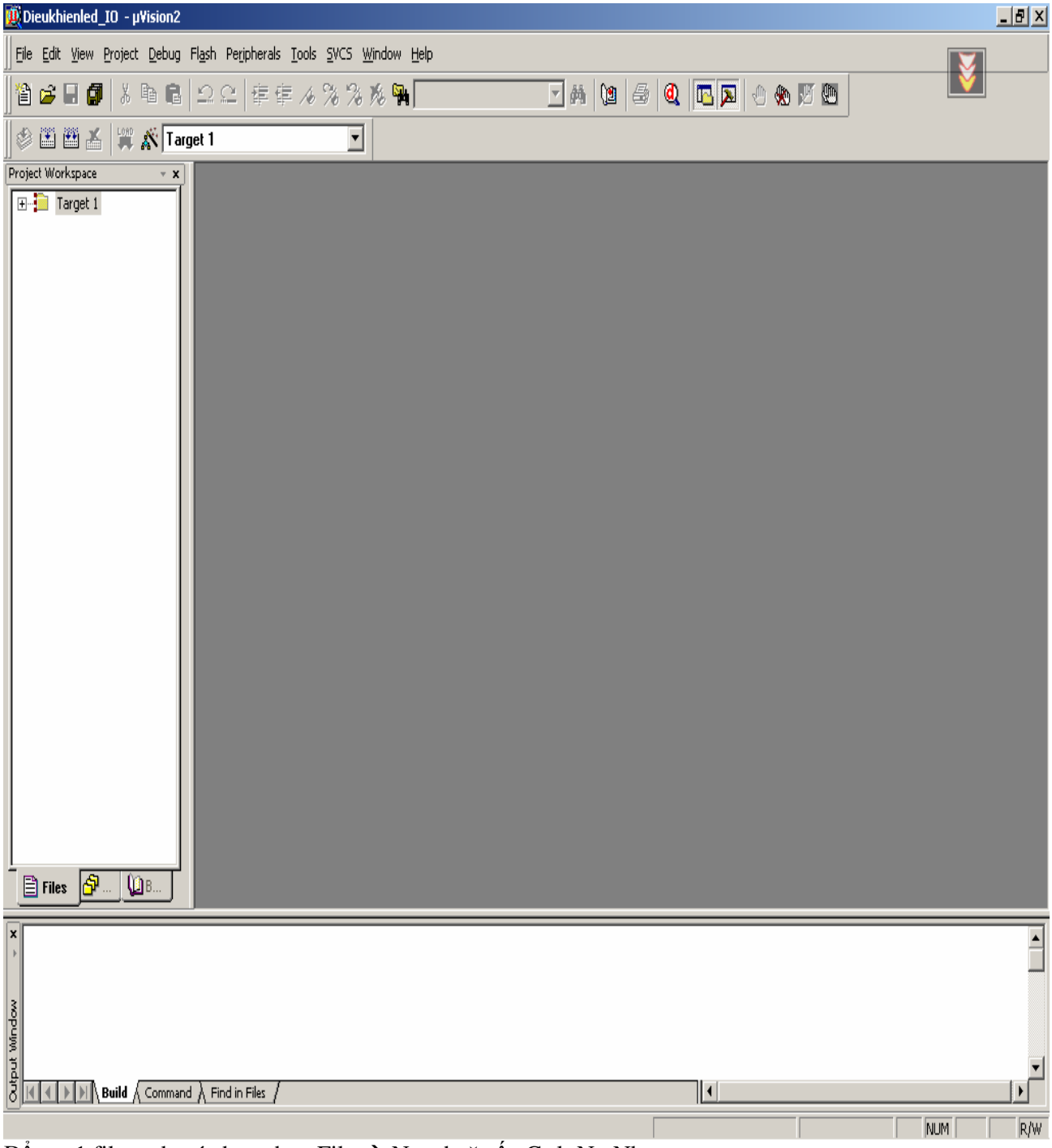
Trong này có 1 loạt các hãng điện tử sản xuất 8051. Bạn lập trình cho con nào thì chọn con đấy ,kích chuột vào các dấu + để mở rộng các con IC của các hãng. Ở đây ta lập trình cho AT89C51 của hãng Atmel nên ta chọn như sau:



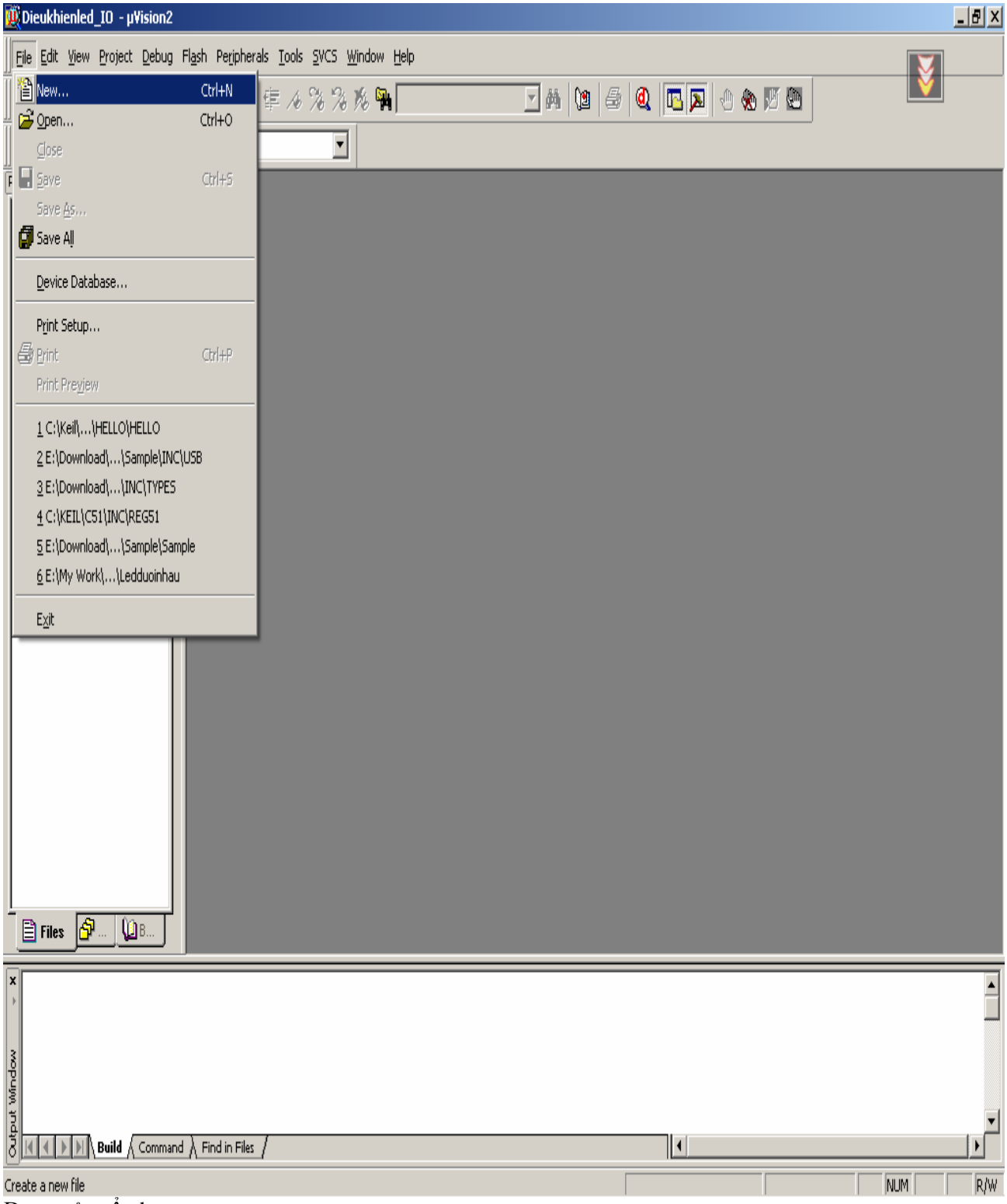
Khi chọn chip thì ngay lập tức cái bảng hiện ra 1 số tính năng của chíp các bạn có thể nhìn thấy. 8051 based Fully Static 24Mhz … . Nhap OK được cửa sổ như sau:



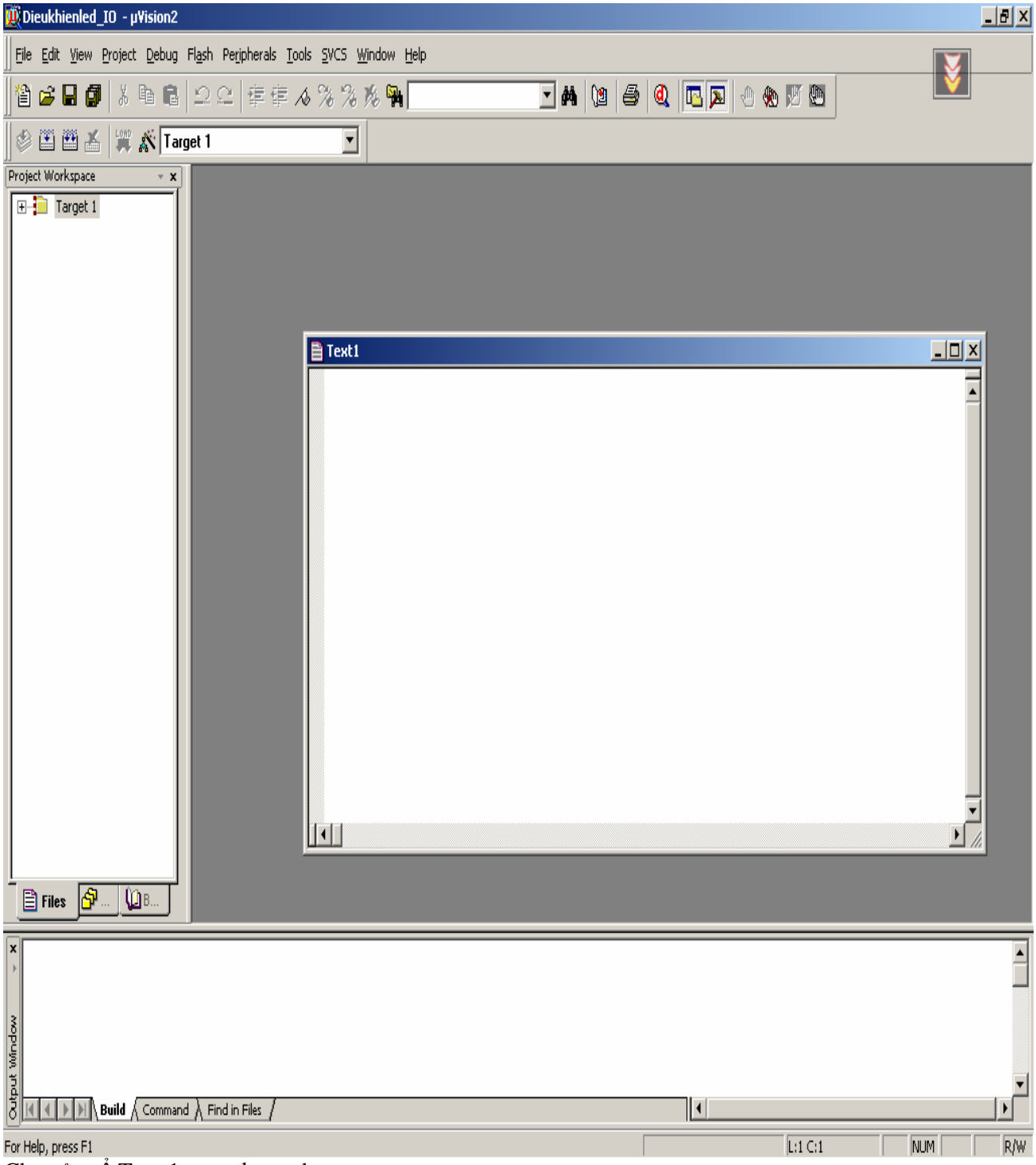
Chọn No. Chọn Yes chỉ làm cho file lập trình của bạn thêm nặng . Được cửa sổ sau:



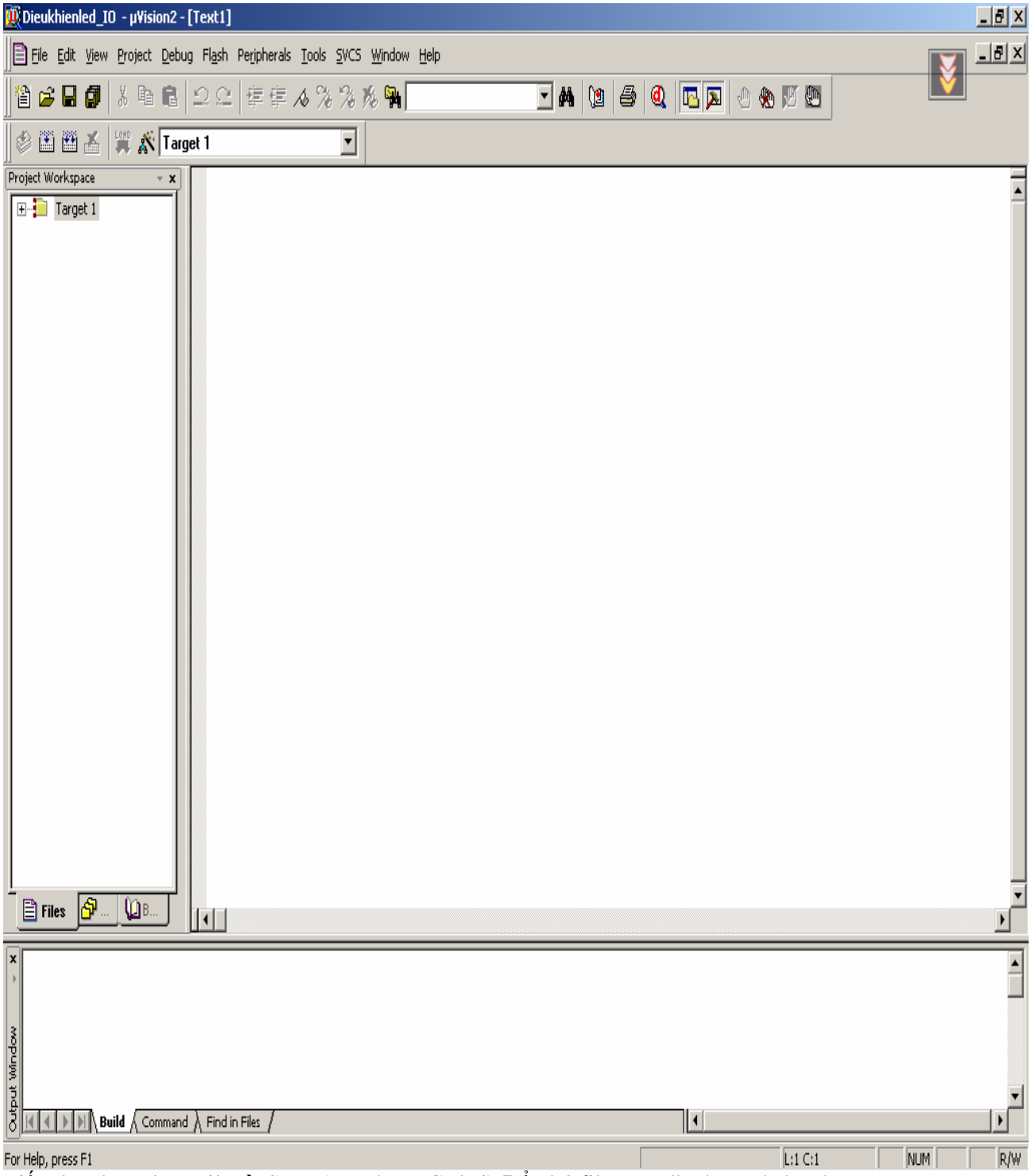
Để tạo 1 file code các bạn chọn File → New hoặc ấn Ctrl+N. Như sau:



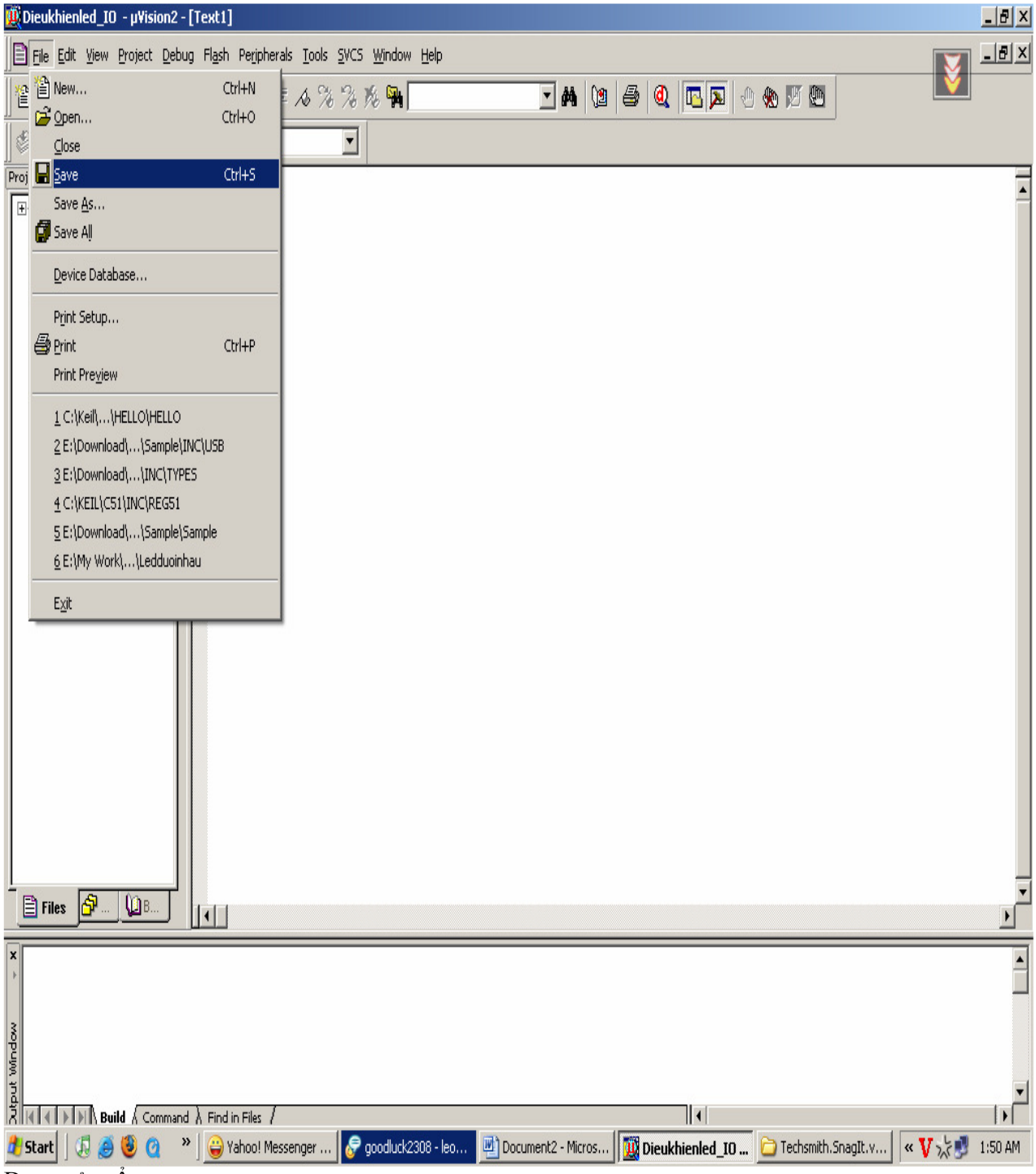
Được cửa sổ như sau:



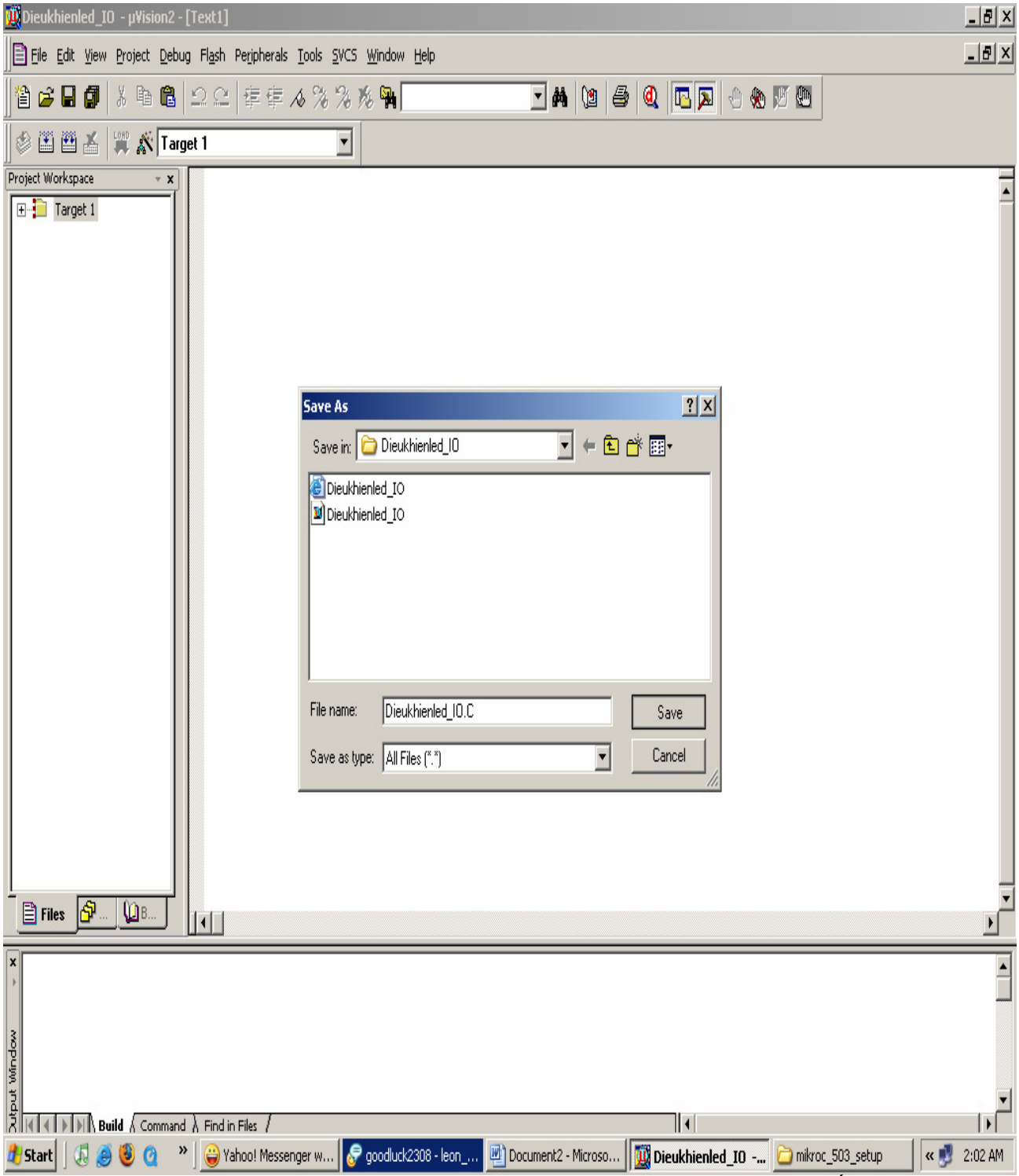
Cho cửa sổ Text 1 to ra được như sau:



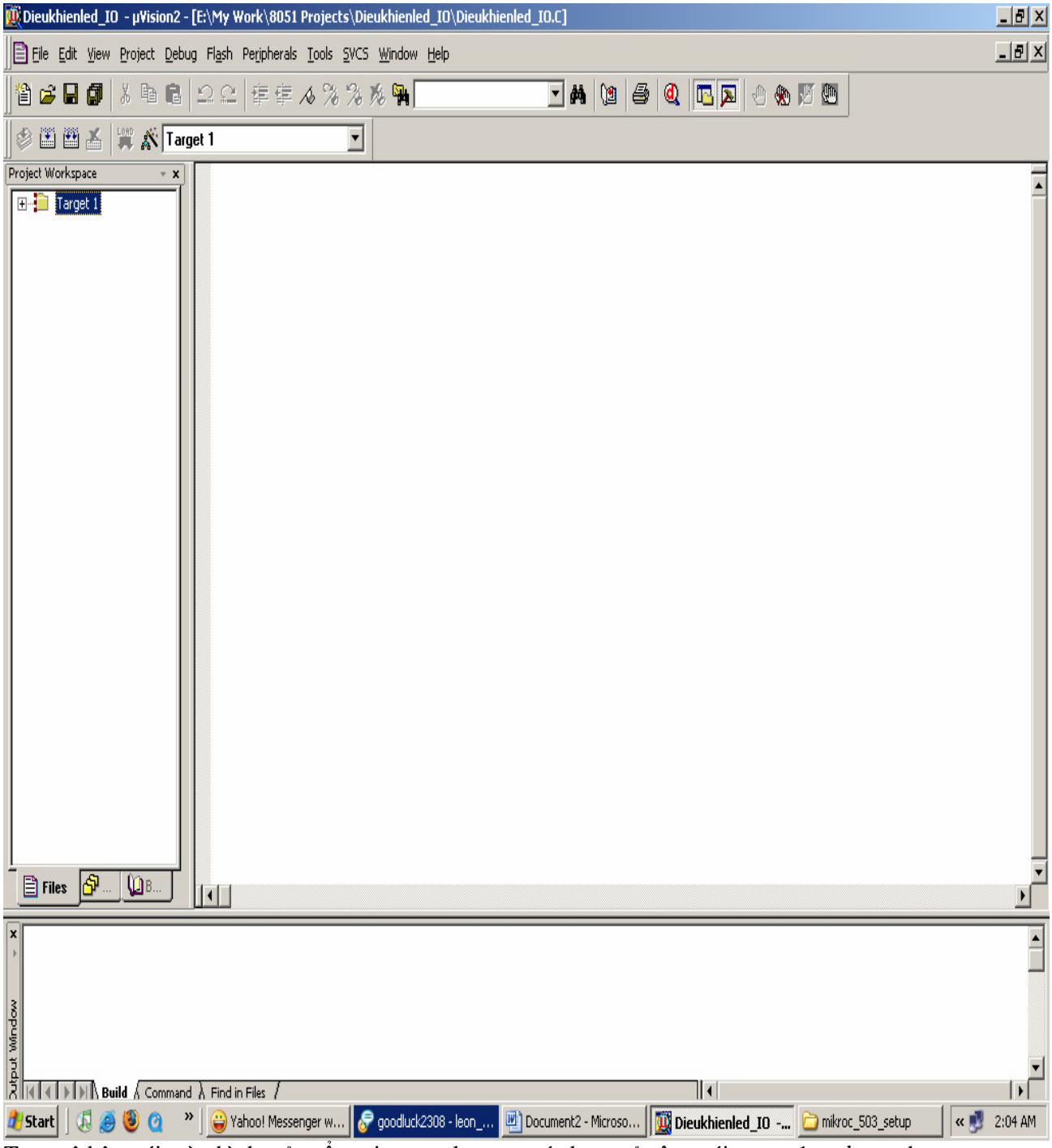
Tiếp theo bạn chọn File → Save As… hoặc Ctrl+S. Để nhớ file mặc dù chưa có gì. Như sau:



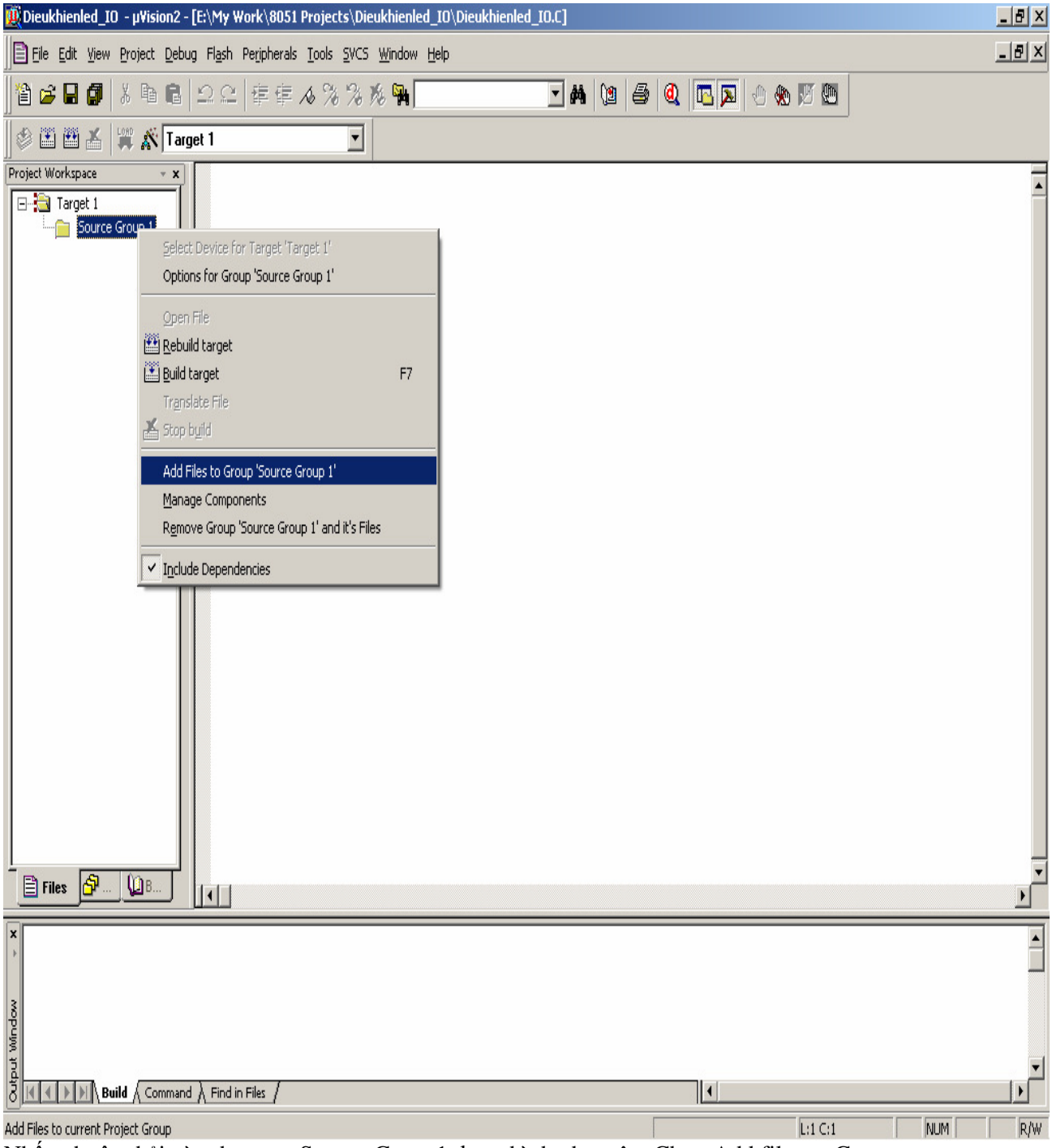
Được cửa sổ sau:



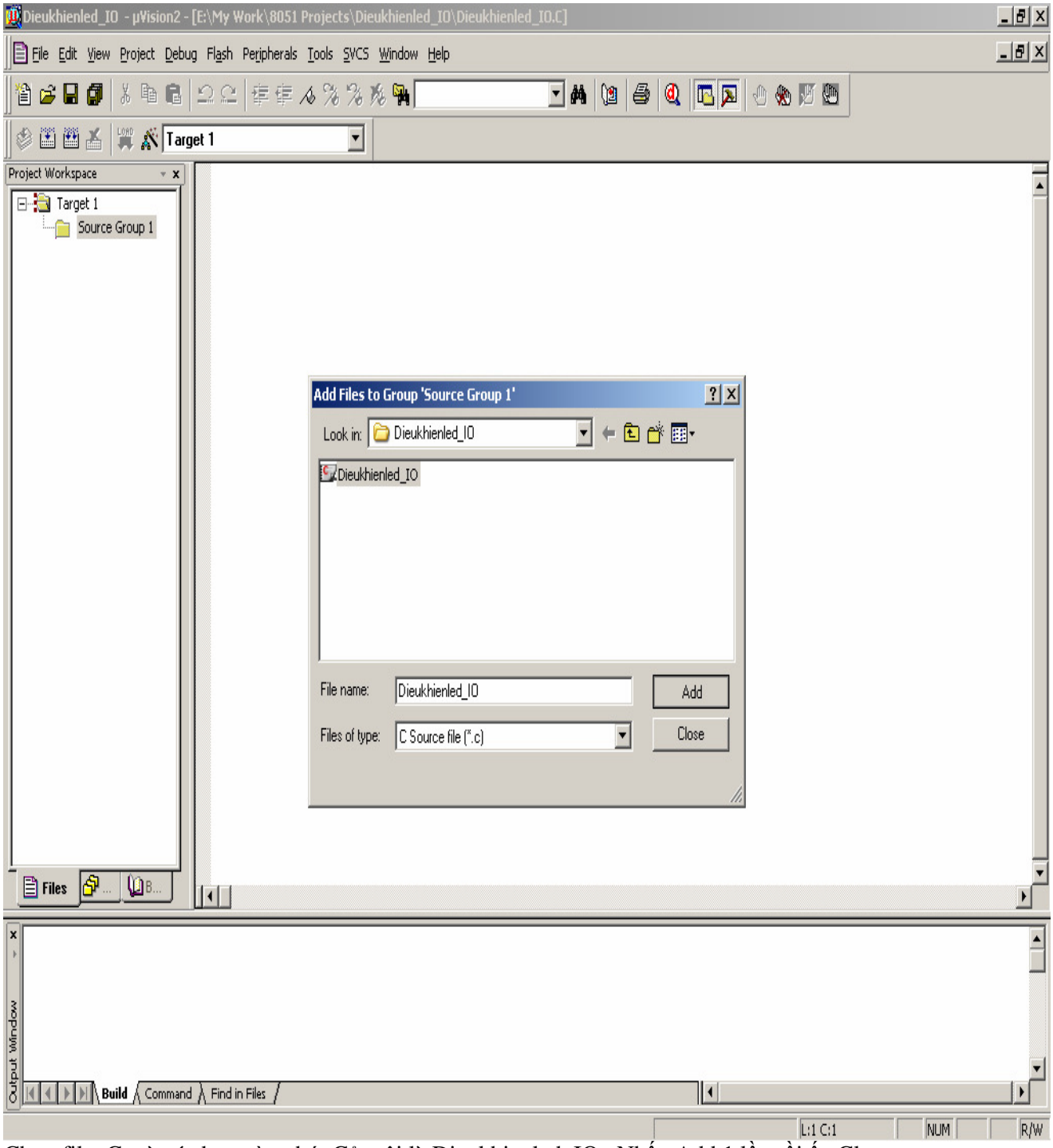
Các bạn nhập tên vào text box file name. Chú ý tên gì cũng được nhưng không được thiếu đuôi mở rộng .C . Nhấn Save. Được cửa sổ sau:



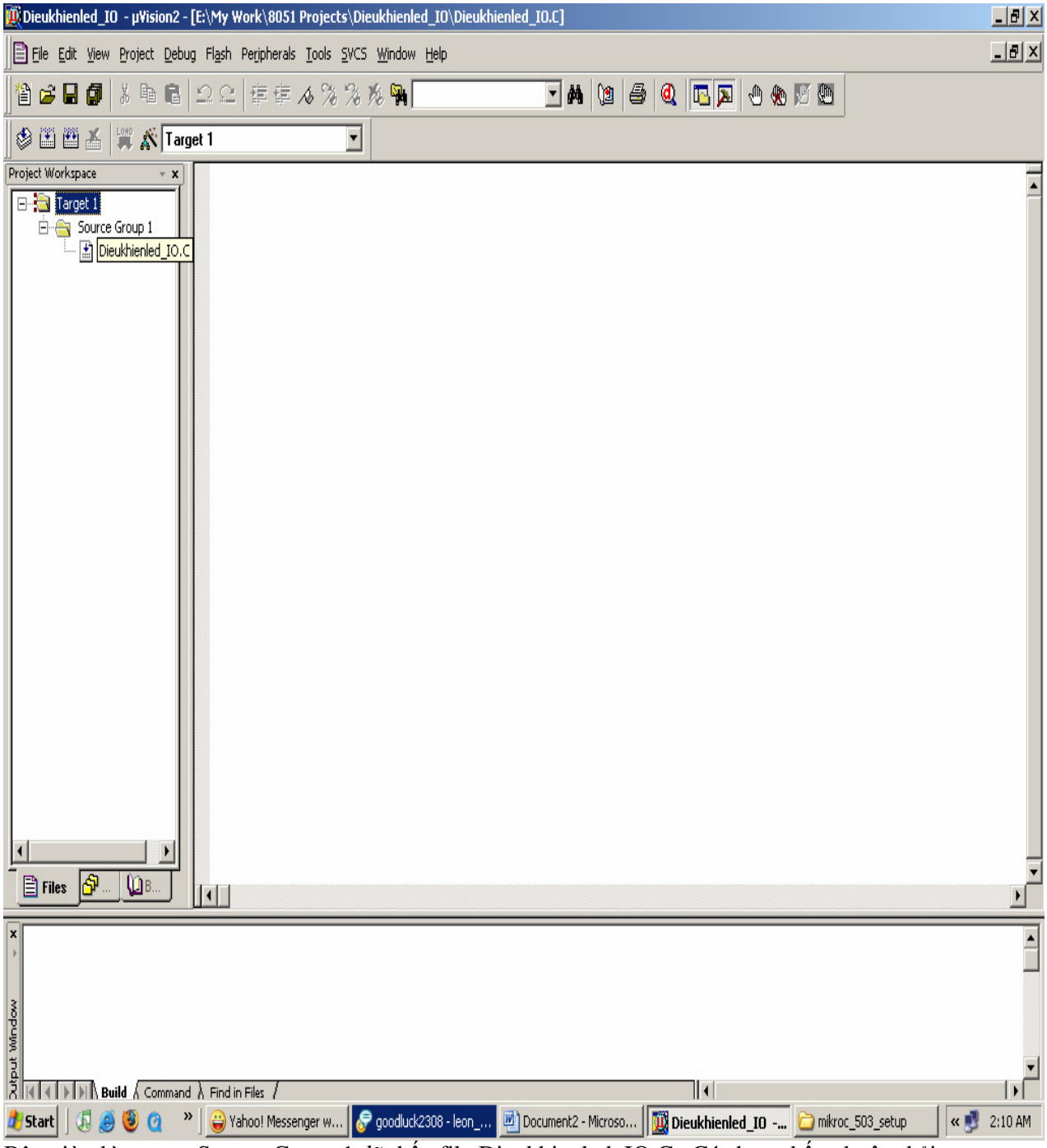
Trong ô bên trái màn hình, cửa sổ project workspace, các bạn mở rộng cái target 1 ra được như sau:



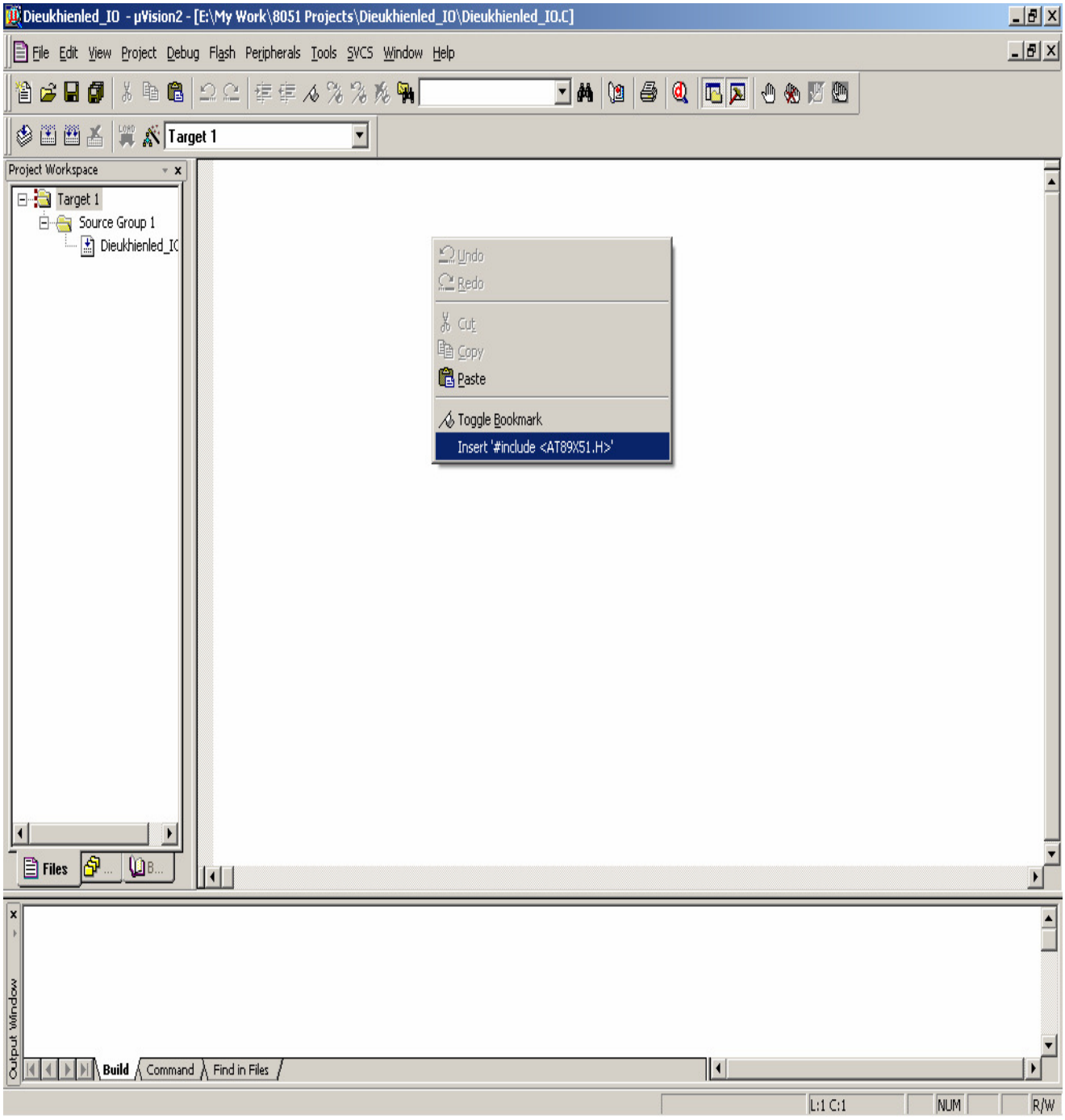
Nhấp chuột phải vào thư mục Suorce Group1 được hình như trên. Chọn Add files to Group "Source Group1" để add file vào project. Được như sau:



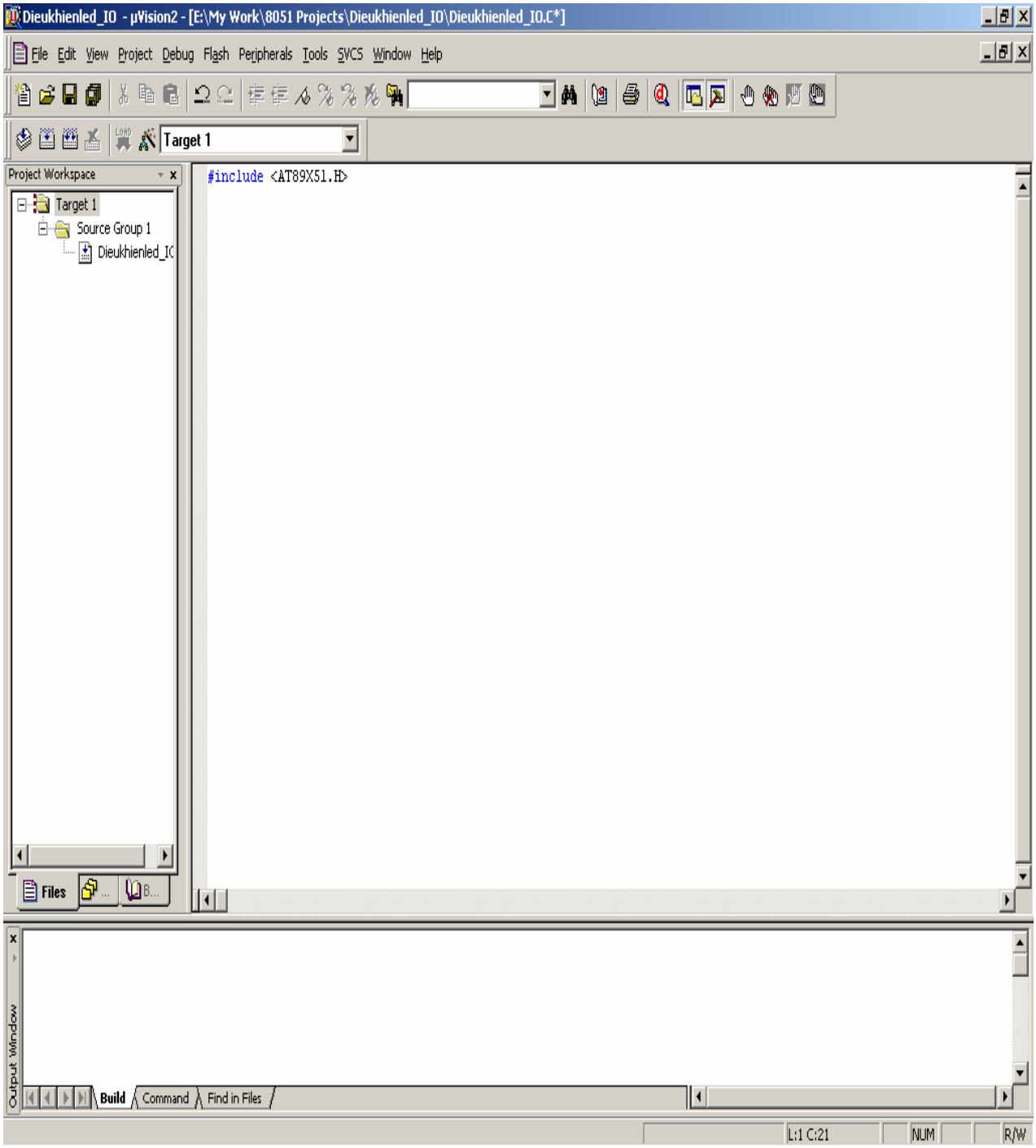
Chọn file .C mà các bạn vừa nhớ. Của tôi là Dieukhienled_IO . Nhấn Add 1 lần rồi ấn Close. Nếu bạn ấn Add 2 lần nó sẽ thông báo là file đã add bạn chỉ việc OK rồi nhấn Close. Được như sau:



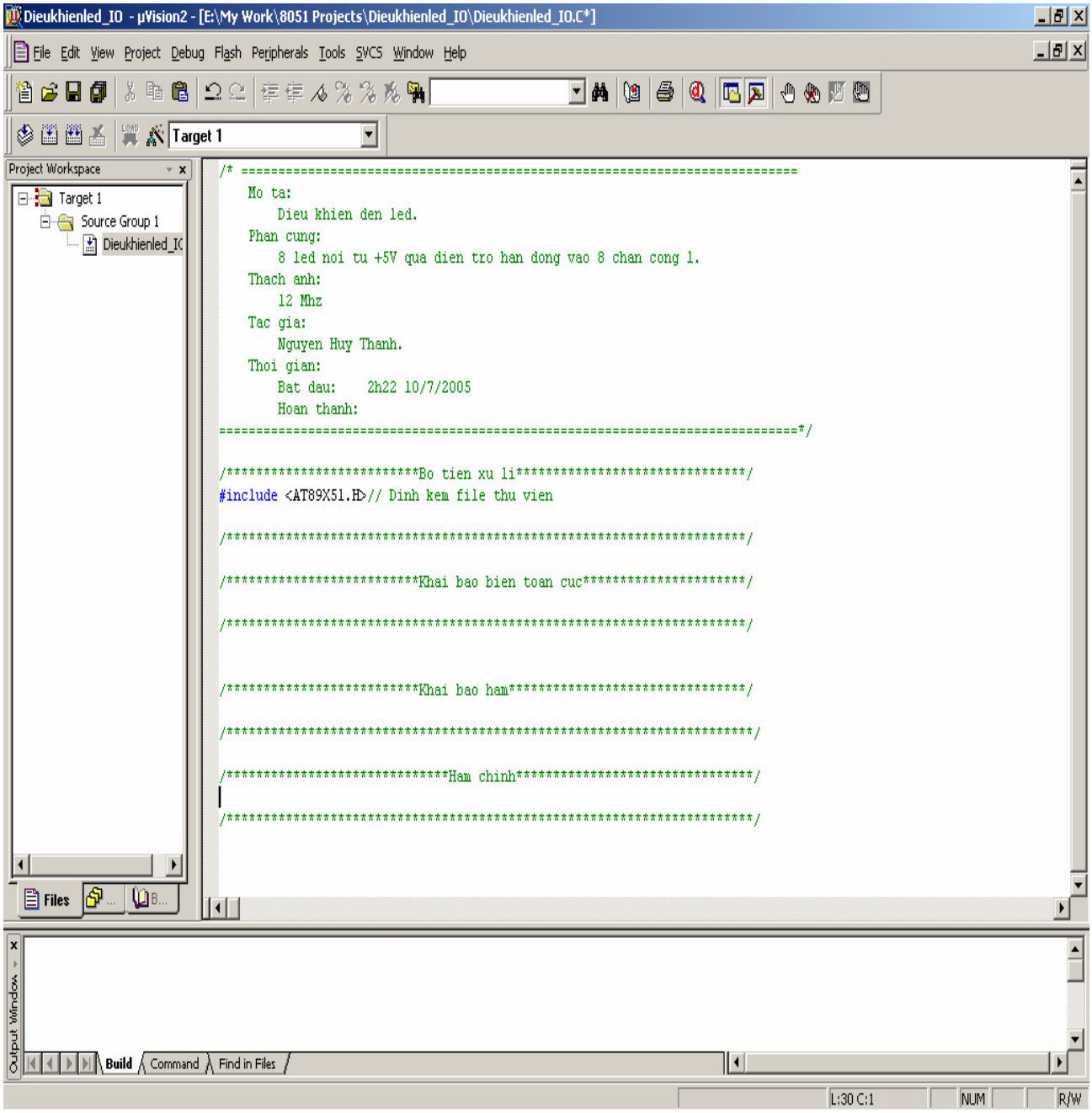
Bây giờ nhìn trong Source Group 1 đã thấy file Dieukhienled_IO.C . Các bạn nhấp chuột phải vào vùng soạn thảo file Dieukhienled_IO.C như sau, để thêm file thư viện.Chọn Insert '#include <AT89X51.H>"



Các bạn đã biết file đó là gì nếu các bạn đã học bài trước kí theo yêu cầu của tôi. Được như sau:

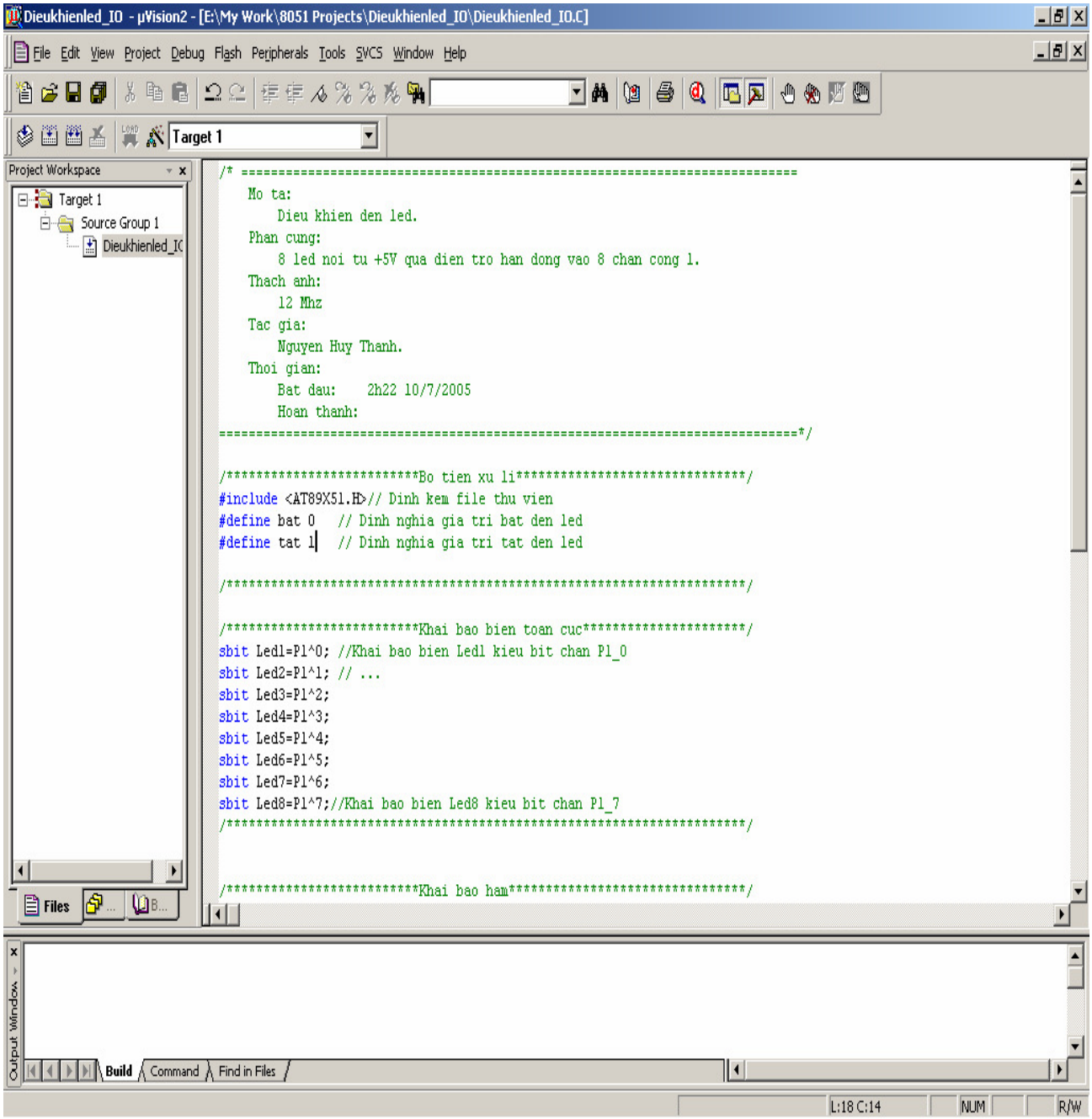


Phần cuối cùng của công việc khởi tạo là các bạn viết lời giải thích cho dự án của mình .Phần này rất cần thiết vì nó để người khác hiểu mình làm gì tron project này và khi mình cần sử dụng lại code đọc lại mình còn biết nó là cái gì.
Các bạn tạo lời giải thích theo mẫu sau:

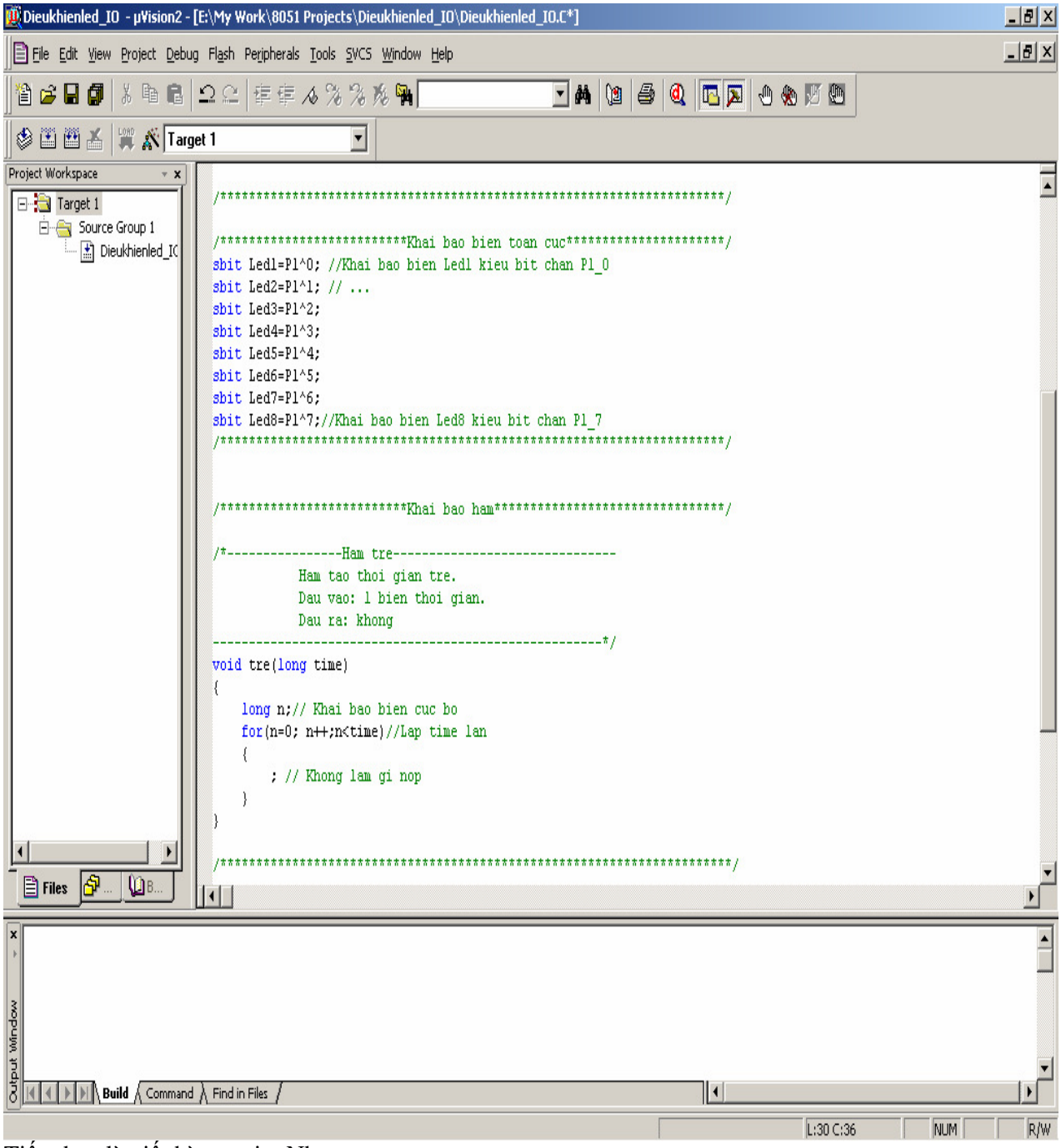


2> Soạn thảo chương trình:

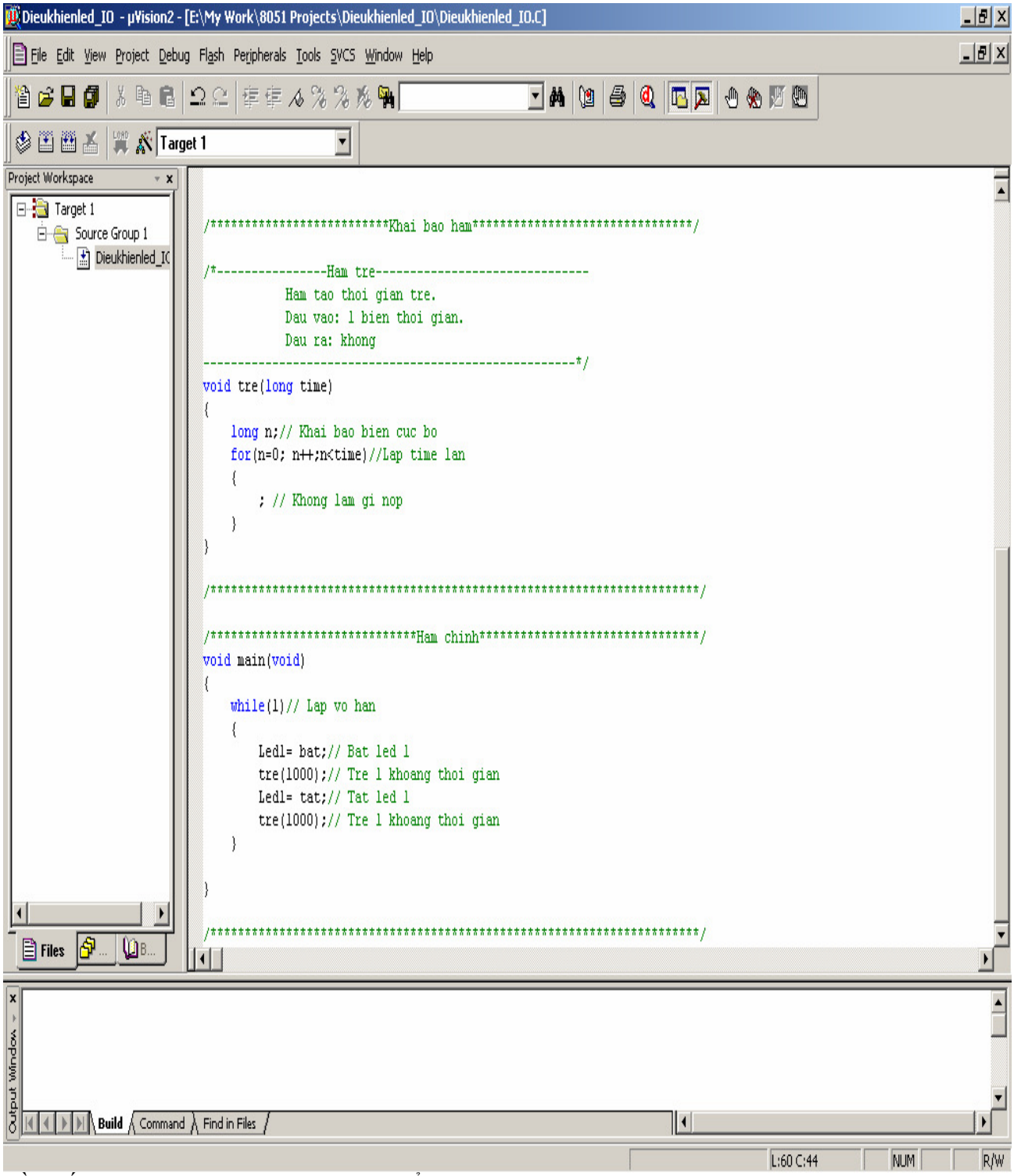
Các bạn viết chương trình của bài 3 vào đây làm ví dụ. Khi viết xong 1 dòng lệnh nên giải thích dòng lệnh đó làm gì. Như sau:



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.



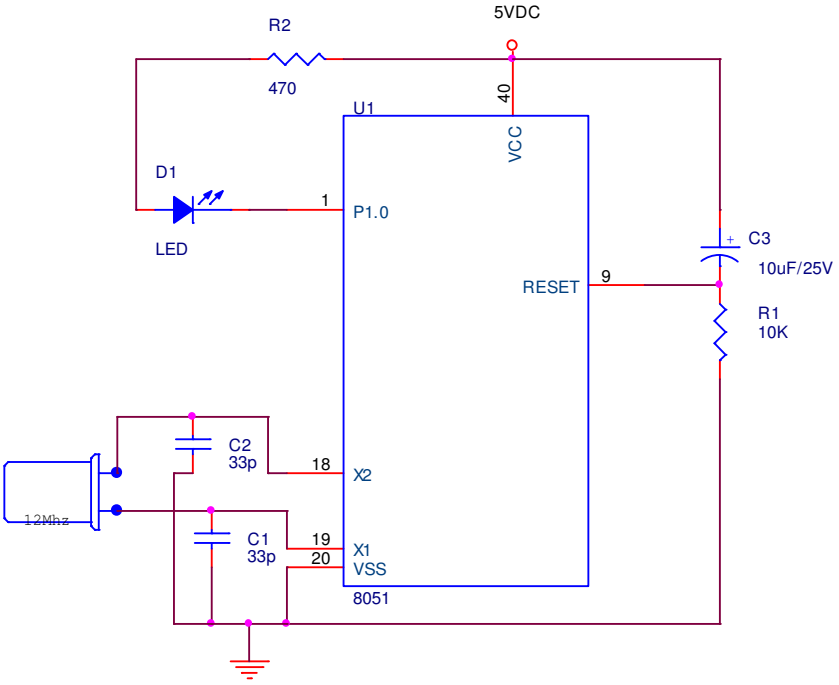
Tiếp theo là viết hàm main. Như sau:



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(vì để phục vụ cho bài 3) 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:

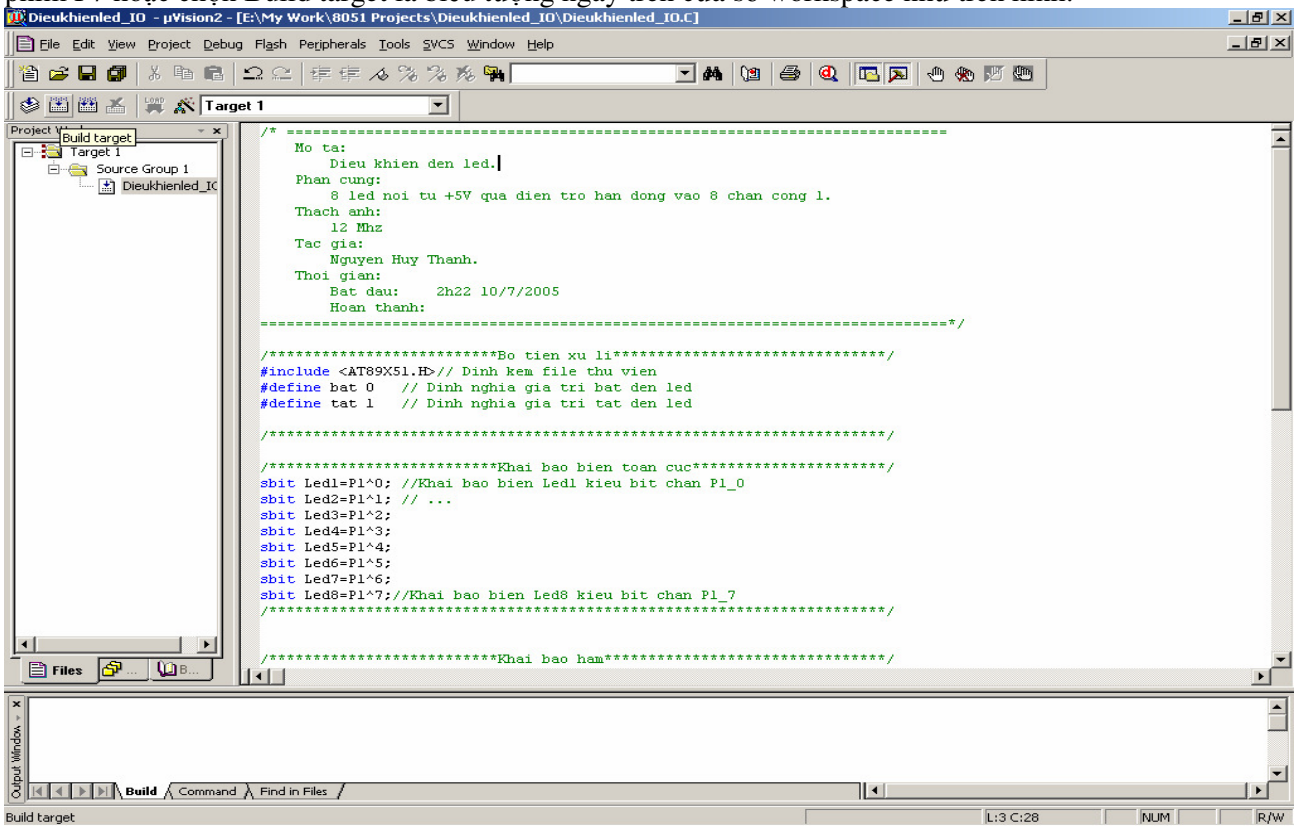
```
  While(1)
{
  Led1=bat;
  Led1=tat;
}
```

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

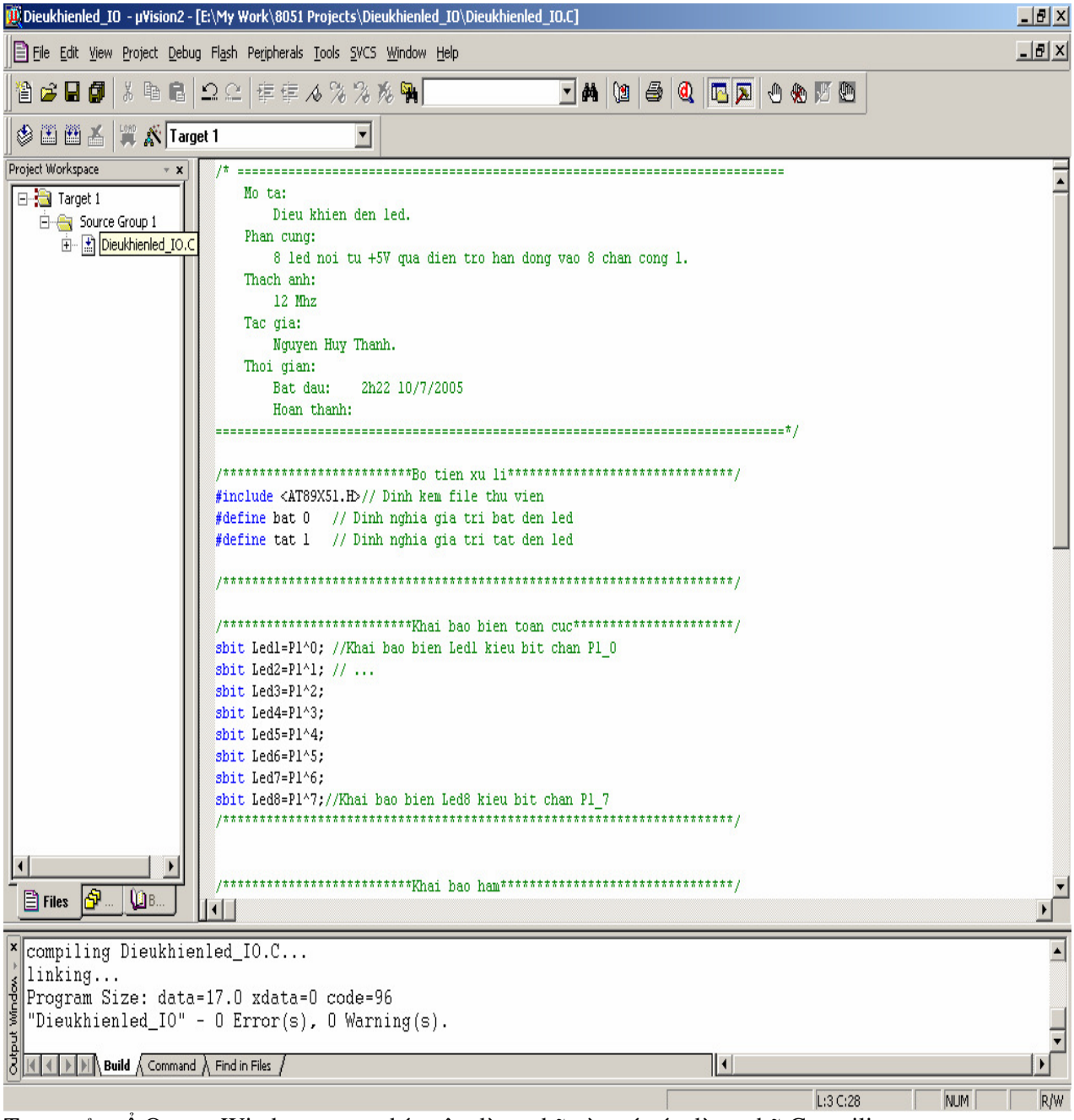
vì hết chương trình rồi còn đâu. Phần này nếu bạn nào mới học điện tử thể nào cũng có câu hỏi thắc mắc cứ nhắn tin hoặc gửi mail.

3> Dịch chương trình:

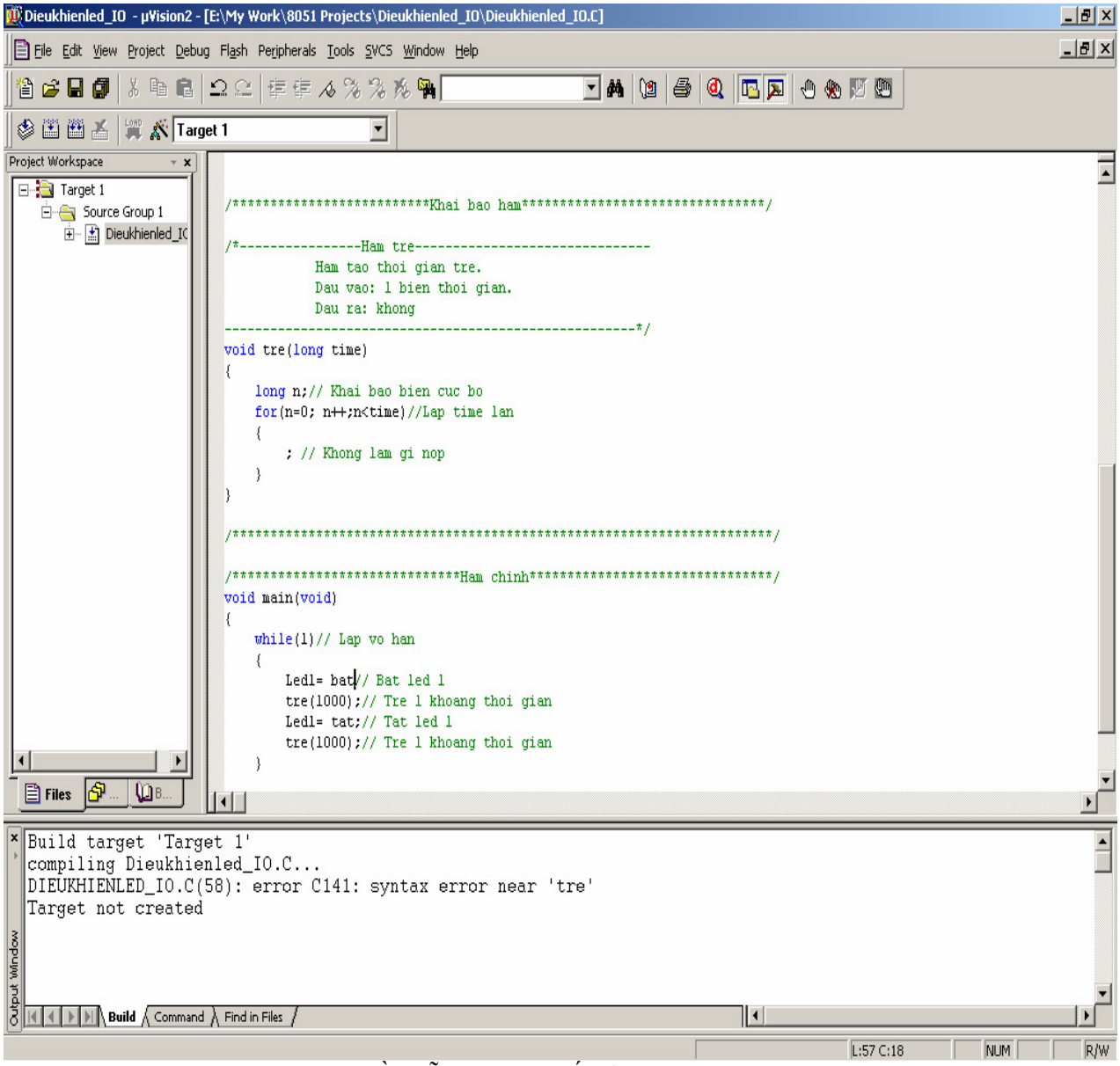
Soạn thảo song nhấn Ctrl +S để nhớ . Nhớ xong các bạn biên dịch chương trình bằng cách ấn phím F7 hoặc chọn Build target là biểu tượng ngay trên cửa sổ workspace như trên hình:



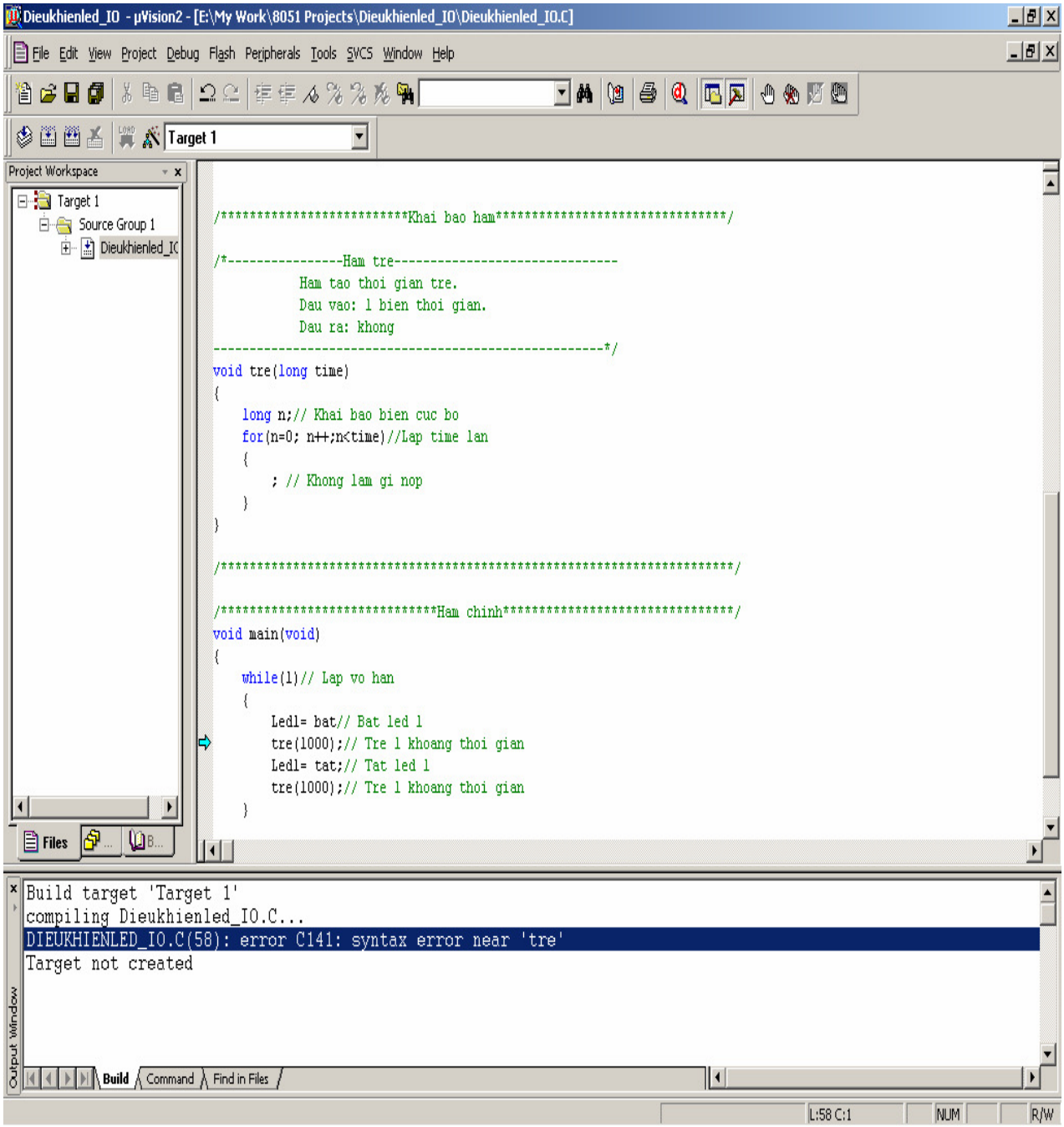
Các bạn sẽ thấy như sau:



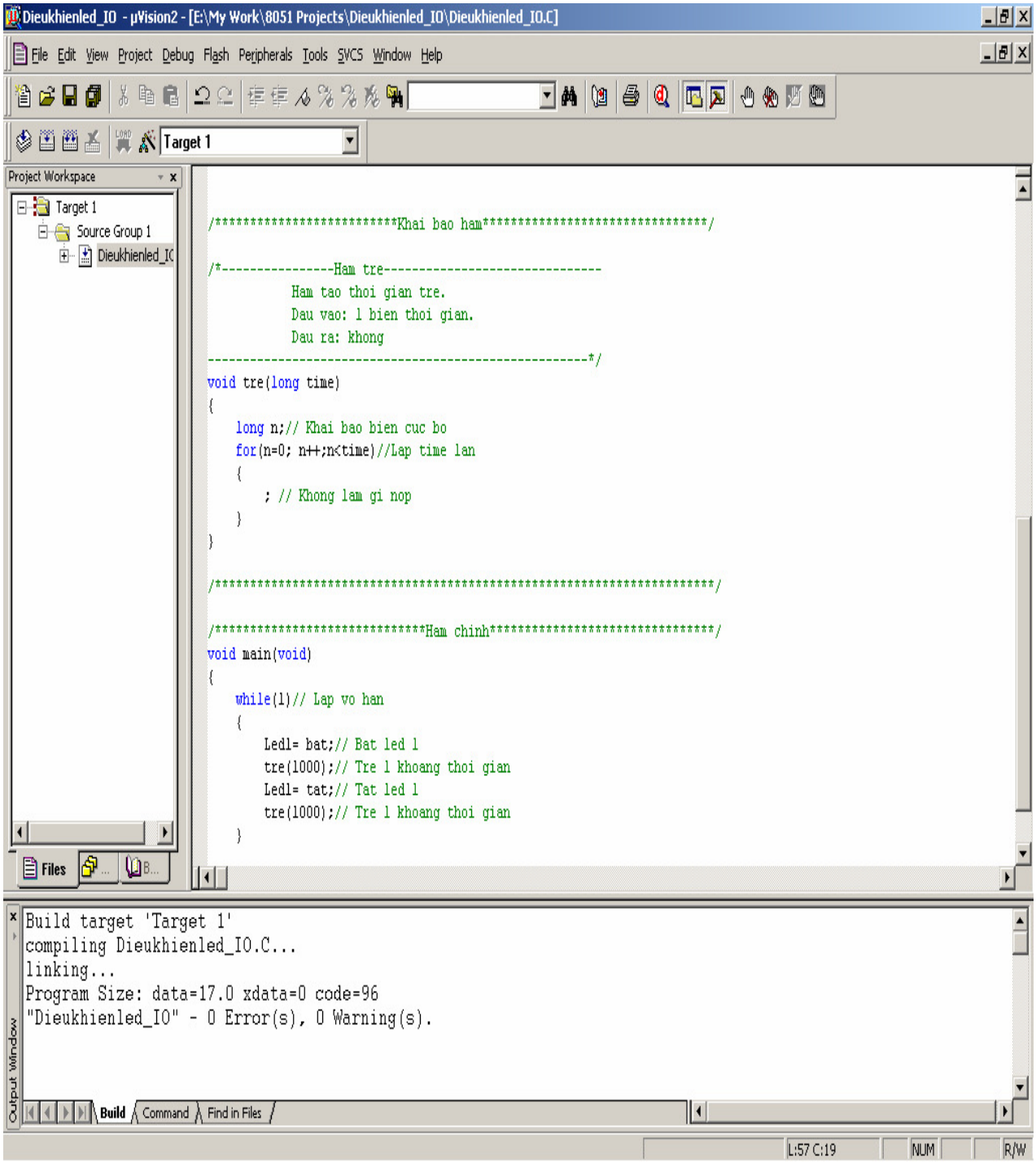
Trong cửa sổ Output Window ngay phía trên dòng chữ này có các dòng chữ Compiling …
Linking…
Program Size: data =17.0 code =96
… 0 error , 0 Warning .
Như vậy là OK. Nếu không được như vậy nó sẽ báo lỗi và các bạn kiểm tra xem soạn thảo đúng chưa. Tôi ví dụ xóa 1 dấu ; ở trong hàm main ở dòng : Led1=bat; , giờ bỏ đi thành Led1= bat .Rồi dịch lại (ấn F7) trình biên dịch sẽ báo như sau:



Syntax error near tre. 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."

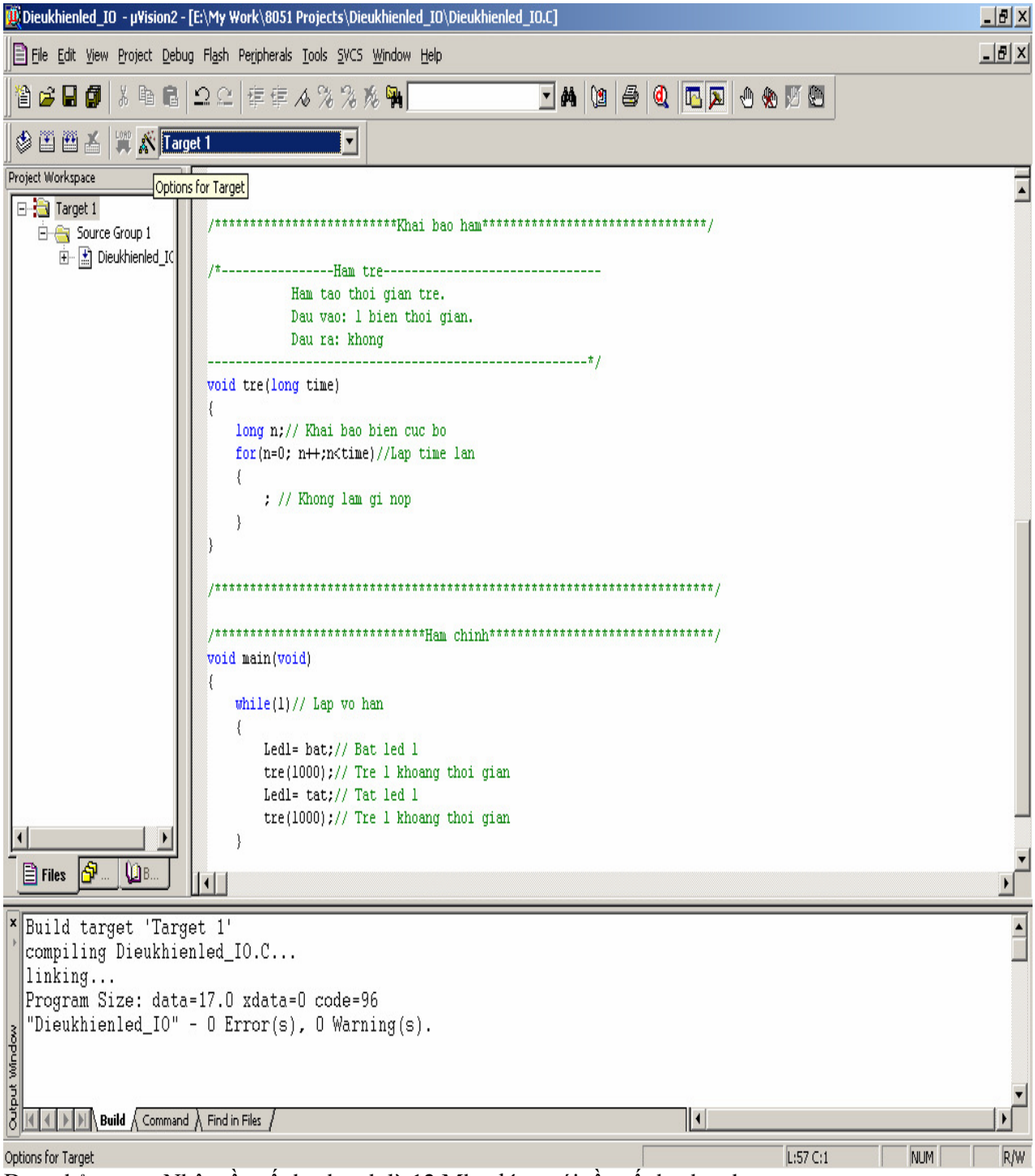


Sau khi dịch lại được hình sau:

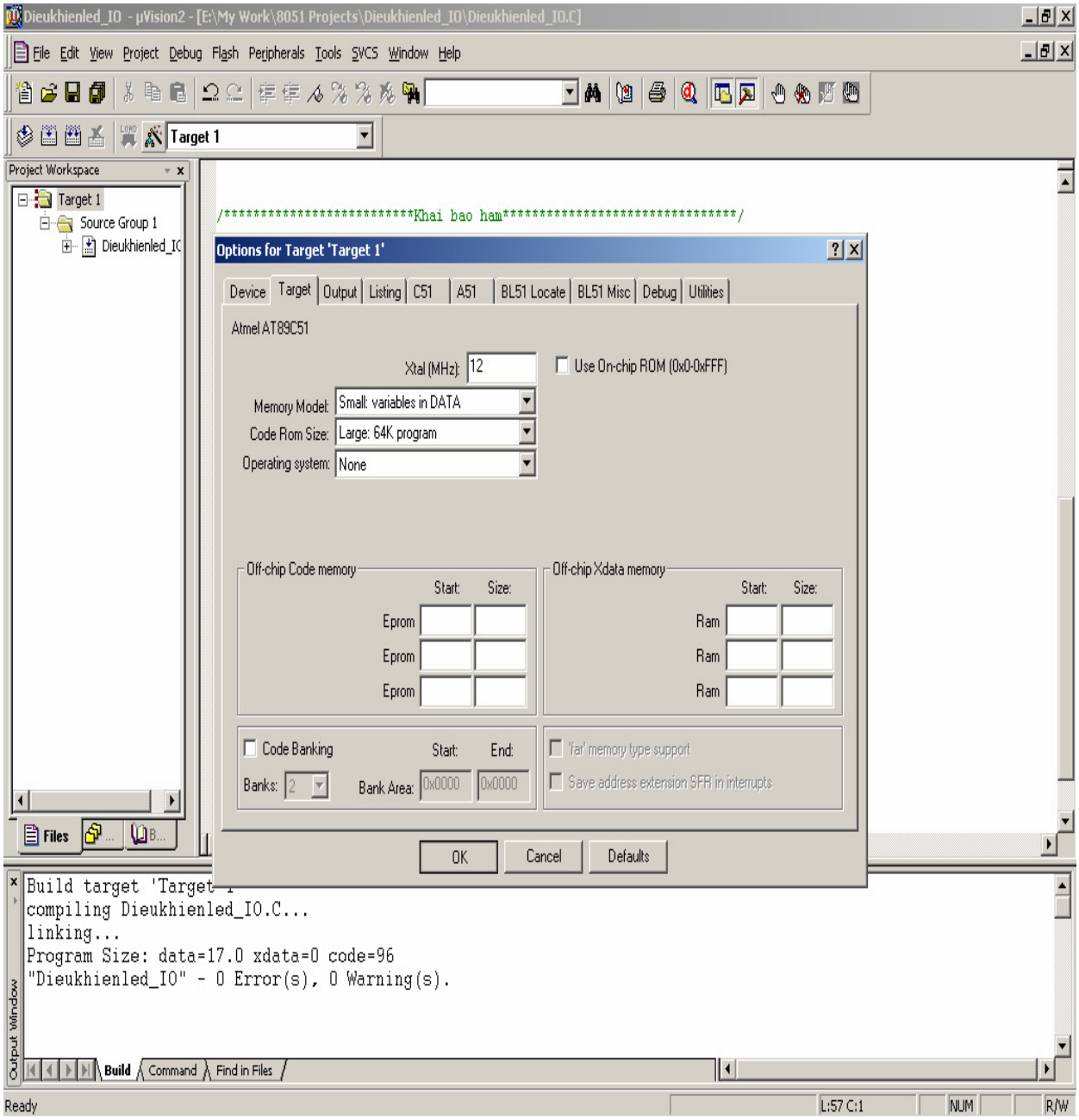


4> Chạy mô phỏng và sửa lỗi.

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

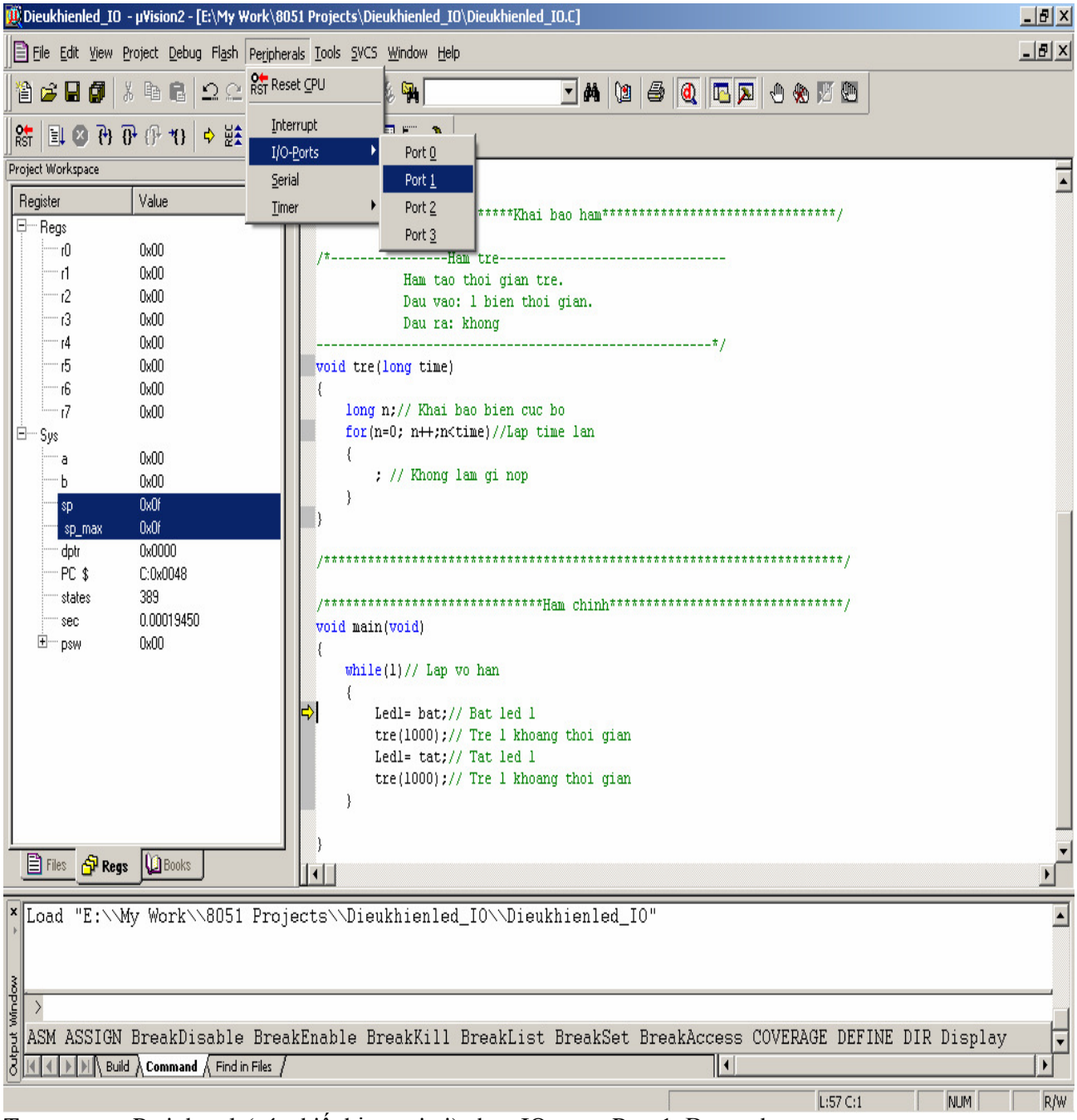


Được bảng sau. Nhập tần số thạch anh là 12 Mhz đúng với tần số thạch anh.

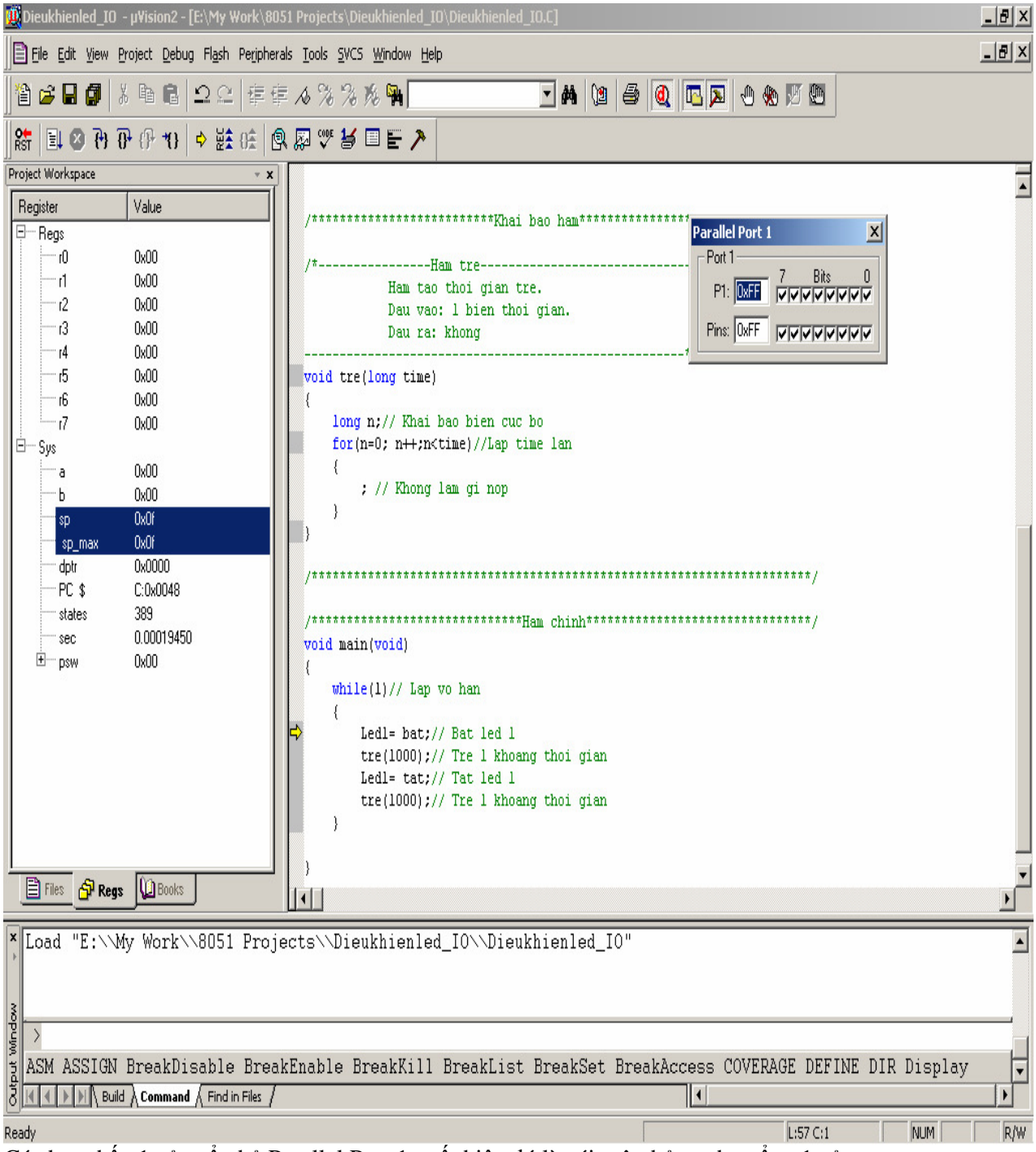


Chọn OK.

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. Được cửa sổ sau:



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. Các bài sau dùng đến các bạn nhớ lấy trong này.

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

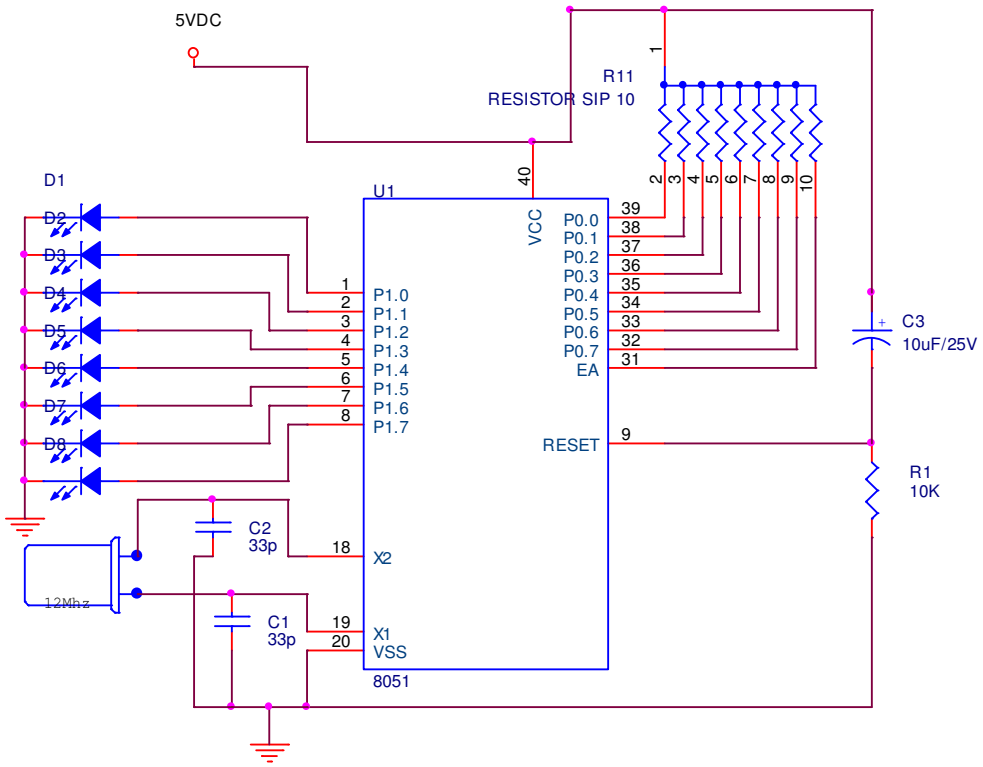
Rồi ấn F11. Mỗi lần ấn sẽ chạy 1 lệnh. Khi debug nếu các bạn chờ hàm delay lâu quá 1000 lần lặp . Các bạn nhấn Ctrl + F11 để bỏ qua hàm.

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.

Bài 3: Điều khiển IO.

3.1.Lắp mạch :

-Khối nguồn 5V các bạn để như lắp mạch đèn nháy.
-Các bạn lắp mạch theo sơ đồ sau:



- Hướng dẫn lắp mạch:

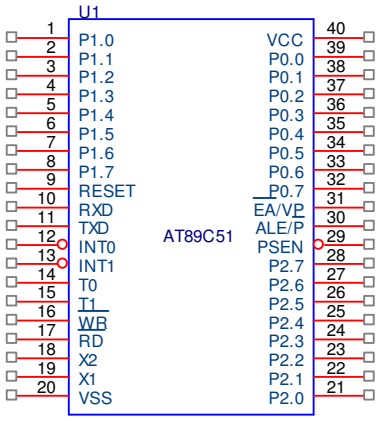
* Qui định : nếu linh kiện nào tôi không nhắc đến chiều thì các bạn lắp linh kiện chiều nào cũng được.

Hàng dọc ngoài là nguồn 5 V.

Hàng dọc trong là GND.

Thứ tự chân các bạn phải nhớ không giống thứ tự sắp xếp tronh hình vẽ mạch. Cách đếm chân các bạn để con chip xuôi mà các bạn nhìn dòng chữ AT89C51/52 xuôi, chân 1 là chân gần dấu tròn và hình tam giác nhỏ mầu trắng. Đếm từ trên xuống dưới trái qua phải. Sơ đồ chân con IC thật sẽ đếm đúng thế này. Khi lắp mạch phải đếm chân.

Không nên lắp nguồn 12V vào vội chỉ khi nạp chíp xong lắp vào mạch mới lắp nguồn.
Khi muốn gỡ chíp ra phải rút nguồn ra rồi mới gỡ chíp.



* Chuẩn bị board như sau:

Lắp 4 dây nối ngắn nối các sông.
Lắp 1 dây nối nối hàng trên cùng với hàng dưới cùng làm nguồn +5V.
Lắp 1 dây nối nối hàng gần trên cùng với hàng gần dưới cùng làm GND.
Lắp chíp như hình.

* Bước 1 lắp mạch dao động:

Lắp 1 con tụ 33pF từ chân 19 xuống chân 20.(Đừng hỏi tại sao).
Lắp 1 con tụ 33pF từ chân 18 xuống chân 20.
Lấy dây câu từ chân 20 xuống GND.
Lắp 1 con thạch anh 12M vào chân 18 và 19.

* Bước 2 lắp mạch reset.

Lắp 1 con trở 10K(nâu đen cam) từ chân 9 xuống GND.
Lắp 1 con tụ 10uF/50V cực dương lên +5V, cực âm vào chân 9.

* Bước 3 lắp trở băng.

Để chíp chạy với ROM trong chân EA phải lên +5V qua 1 điện trở.
Cổng 0 là cổng có cực máng hở muốn thực hiện được IO thì phải có điện trở treo.(Thấy cái này lạ xem lại bài 2). Nên lắp 1 con trở băng 10 chân vào cổng 0 và chân EA.
Chân 1 của trở băng (Chân đầu tiên có nốt tròn mầu trắng) vào chân 40. Chân 10 của trở băng vào chân 31.
Nếu không có trở băng các bạn có thể thay trở băng 10 chân bằng 9 con trở thường vì trở băng 10 chân chính là 9 con trở đấu chung 1 đầu như trong sơ đồ mạch phía trên.

* Bước 4 lắp led:

Để dễ dàng khi lắp mạch nên chúng ta lắp led theo sơ đồ như trên.
Lắp chân dài của 1 led vào chân 1 vi điều khiển chân ngắn vào GND.
Tương tự lắp 7 led còn lại vào chân 2 đến chân 8.
* Bước 5 câu 1 dây nhỏ từ chân 40 lên nguồn 5V.

3.2. Nguyên lí hoạt động:

Led nối từ chân vđk xuống đất vậy nếu chân vi điều khiển 5V thì led sẽ sáng, nếu chân vi điều khiển 0V thì led sẽ tối.

Điện áp 5V vì sao led không cháy mà lại còn sáng yếu?

Vì vi điều khiển 8051 chỉ có thể cung cấp dòng nhỏ không đủ 10mA ở 1 chân nên led sáng yếu. Còn nếu muốn led sáng đẹp thì lắp như sau từ dương 5V → Chân dài của led→ Chân ngắn của led →Chân vi điều khiển. Cái này sẽ nói sau.
* Bước 5 lắp nguồn:
Lắp 1 dây từ chân 40 lên hàng nguồn 5V.

3.3. Lập trình :

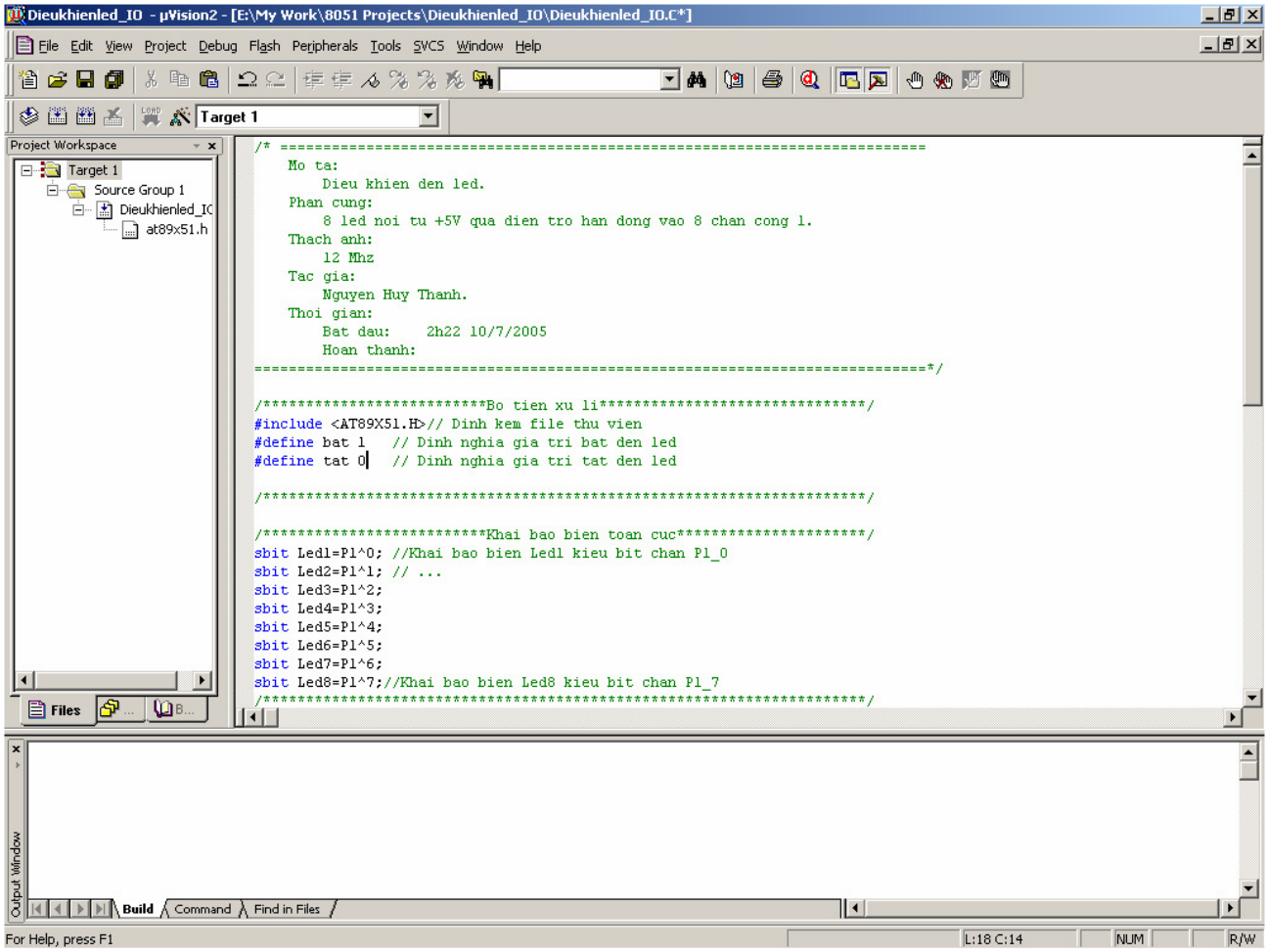
Trước hết điều khiển 1 led. Để điều khiển 1 led thì các bạn chỉ việc gán chân nối với led đó bằng 0 hoặc 1, thì điện áp ở chân đó sẽ là 0V hoặc 5V, tùy vào điện áp đèn sẽ sáng hoặc tối.
Code như sau:

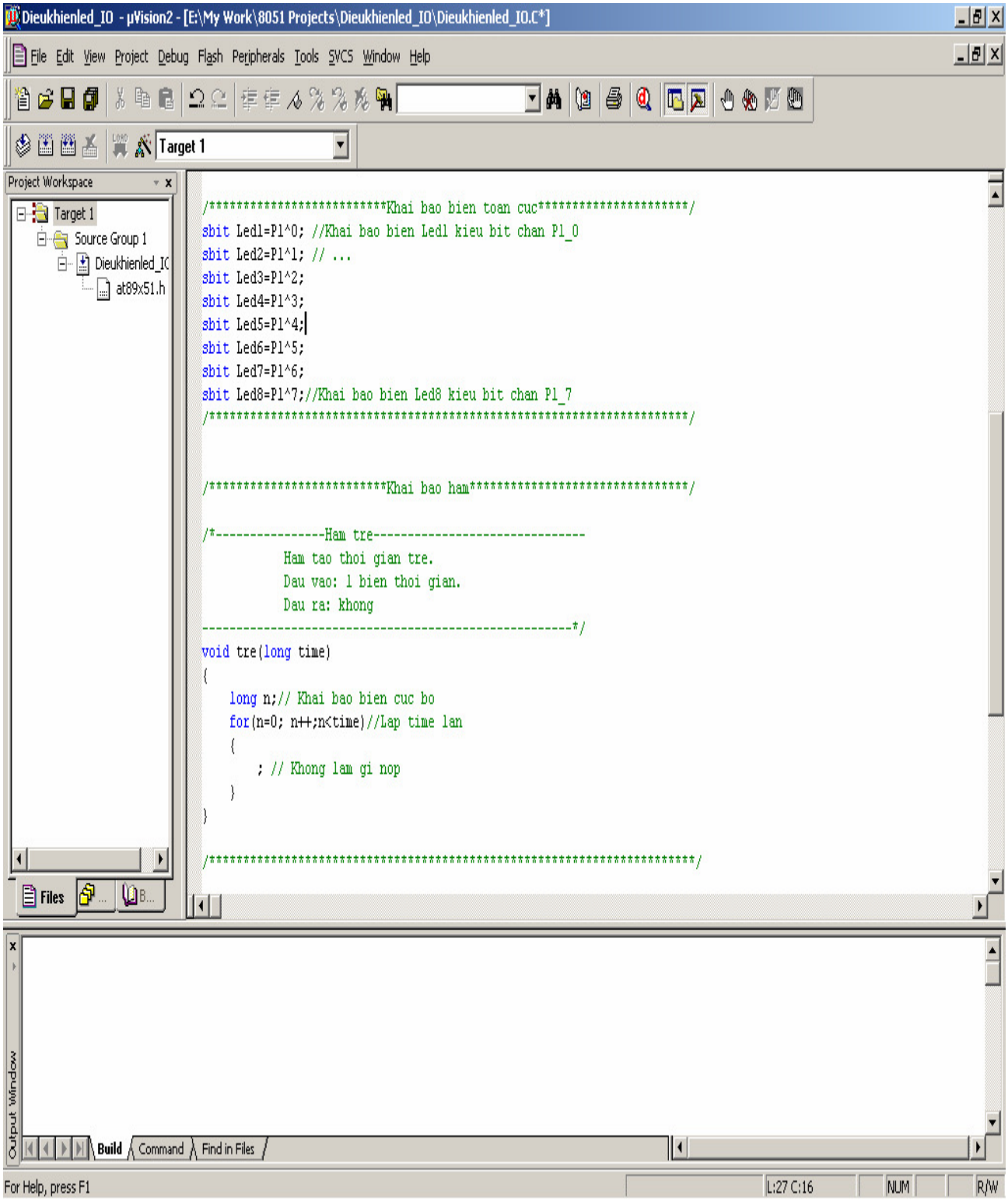
```
/* ======================================================
        Mo ta:
                Dieu khien den led.
        Phan cung:
                8 led noi tu +5V qua dien tro han dong vao 8 chan cong 1.
   Thach anh:
                12 Mhz
        Tac gia:
                Nguyen Huy Thanh.
        Thoi gian:
                Bat dau:        1h03 16/7/2005
                Hoan thanh:     1h07 16/7/2005
======================================================*/

/******************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.
```

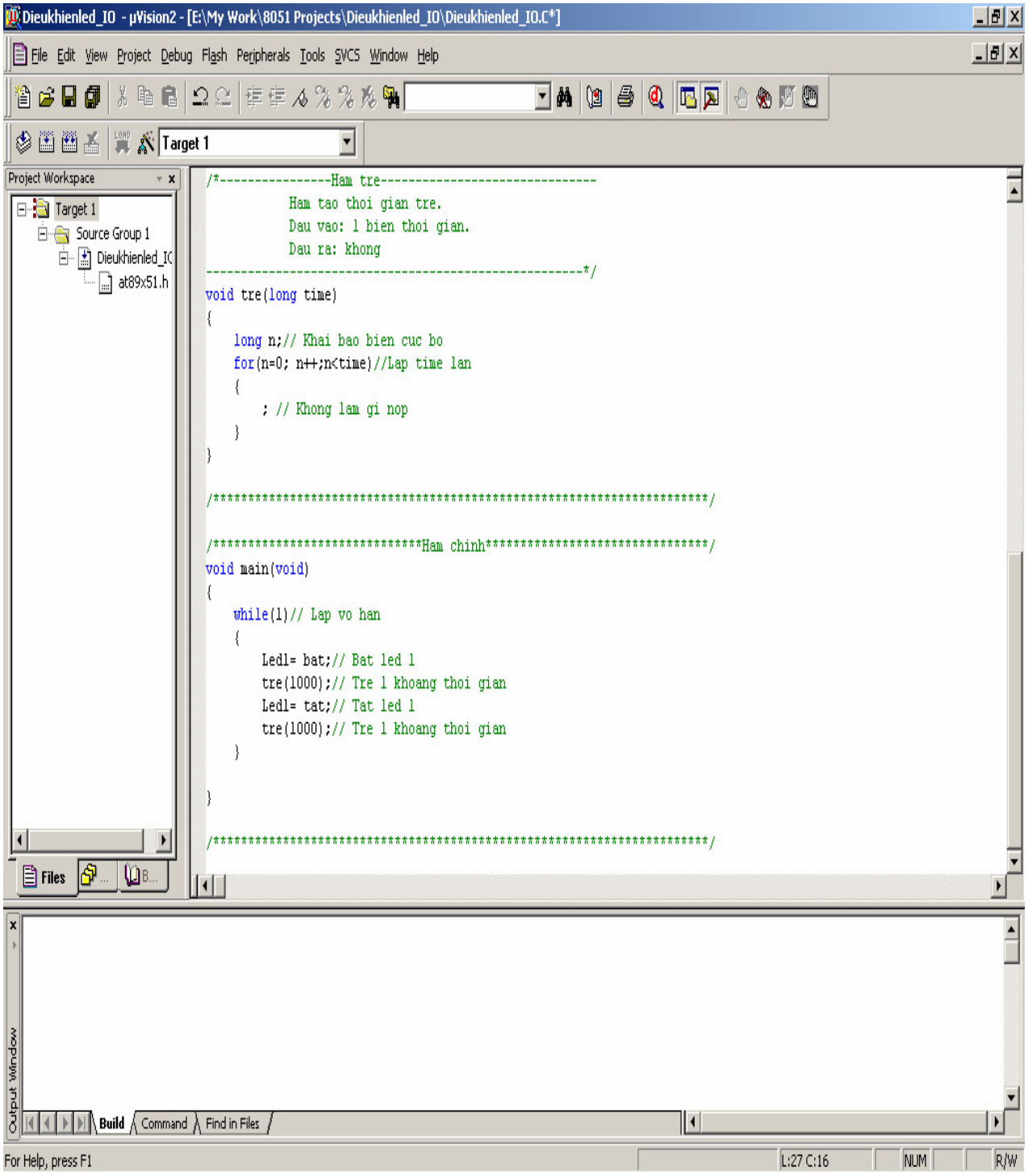
```
                        Dau ra:khong
-----------------------------------------------------*/
void tre(long time)
{
        long n;// Khai bao bien cuc bo
        for(n=0; n<time; n++)//Lap time lan
        {
                ; // Khong lam gi nop
        }
}

/************************************************************/

/********************Ham chinh**************************/
void main(void)
{
        while(1)// Lap vo han
        {
                Led1= bat;// Bat led 1
                tre(1000);// Tre 1 khoang thoi gian
                Led1= tat;// Tat led 1
                tre(1000);// Tre 1 khoang thoi gian
        }

}

/************************************************************/
```

Nên tự gõ chứ không nên copy paste.

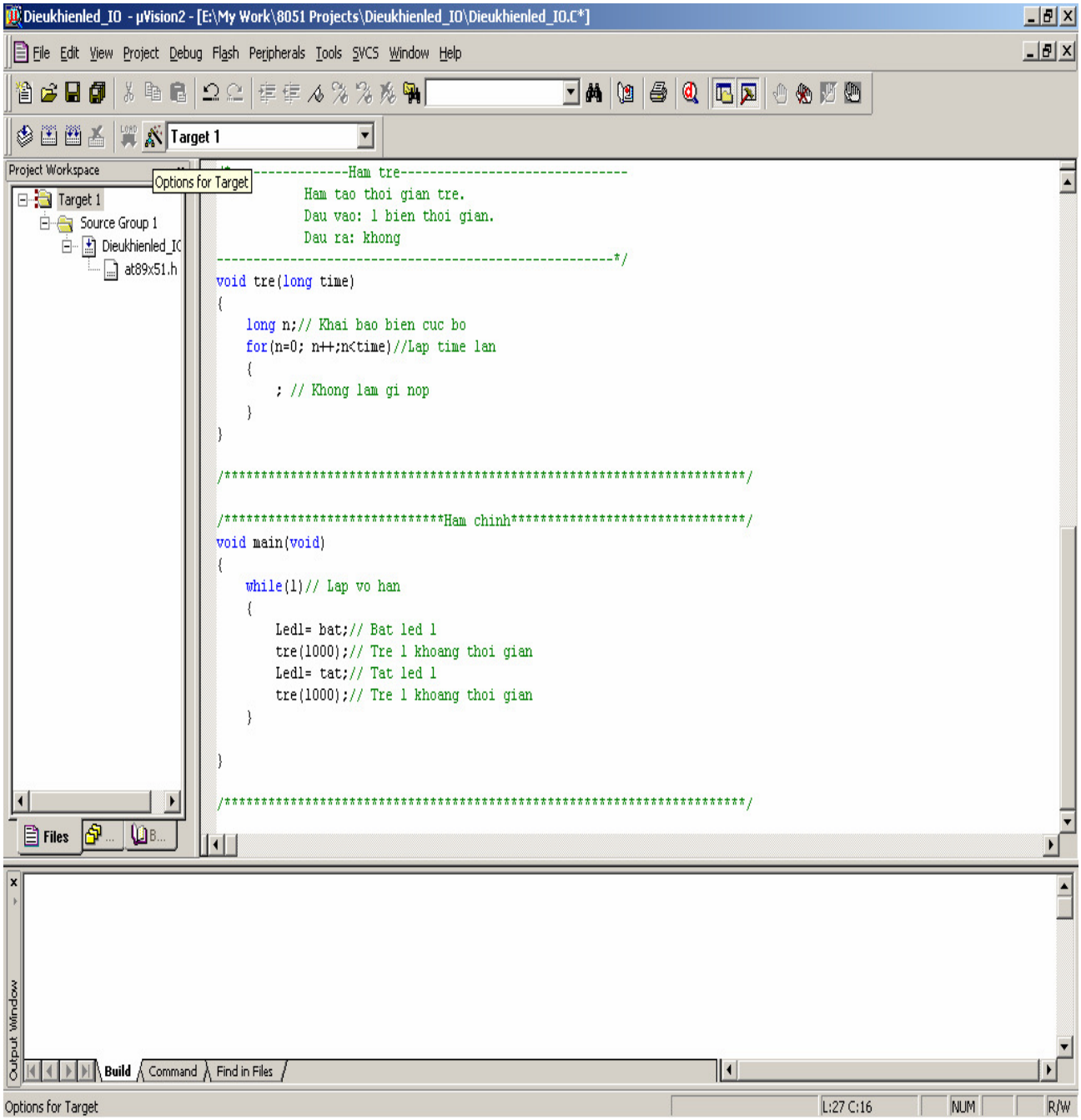
* CHÚ Ý: Code này khác code trong bài 3 sử dụng Keil C đấy nhé. Chỗ #define bat 1 và #define tat 0. Vì led lắp kiểu khác mà.


Dieukhienled_IO - µVision2 - [E:\My Work\8051 Projects\Dieukhienled_IO\Dieukhienled_IO.C*]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Target 1
Project Workspace
Target 1
Source Group 1
Dieukhienled_IC
at89x51.h
Files
/* ==
Mo ta:
Dieu khien den led.
Phan cung:
8 led noi tu +5V qua dien tro han dong vao 8 chan cong 1.
Thach anh:
12 Mhz
Tac gia:
Nguyen Huy Thanh.
Thoi gian:
Bat dau: 2h22 10/7/2005
Hoan thanh:
==*/
/*************************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
/***/
Output Window
Build Command Find in Files
For Help, press F1
L:18 C:14
NUM
R/W

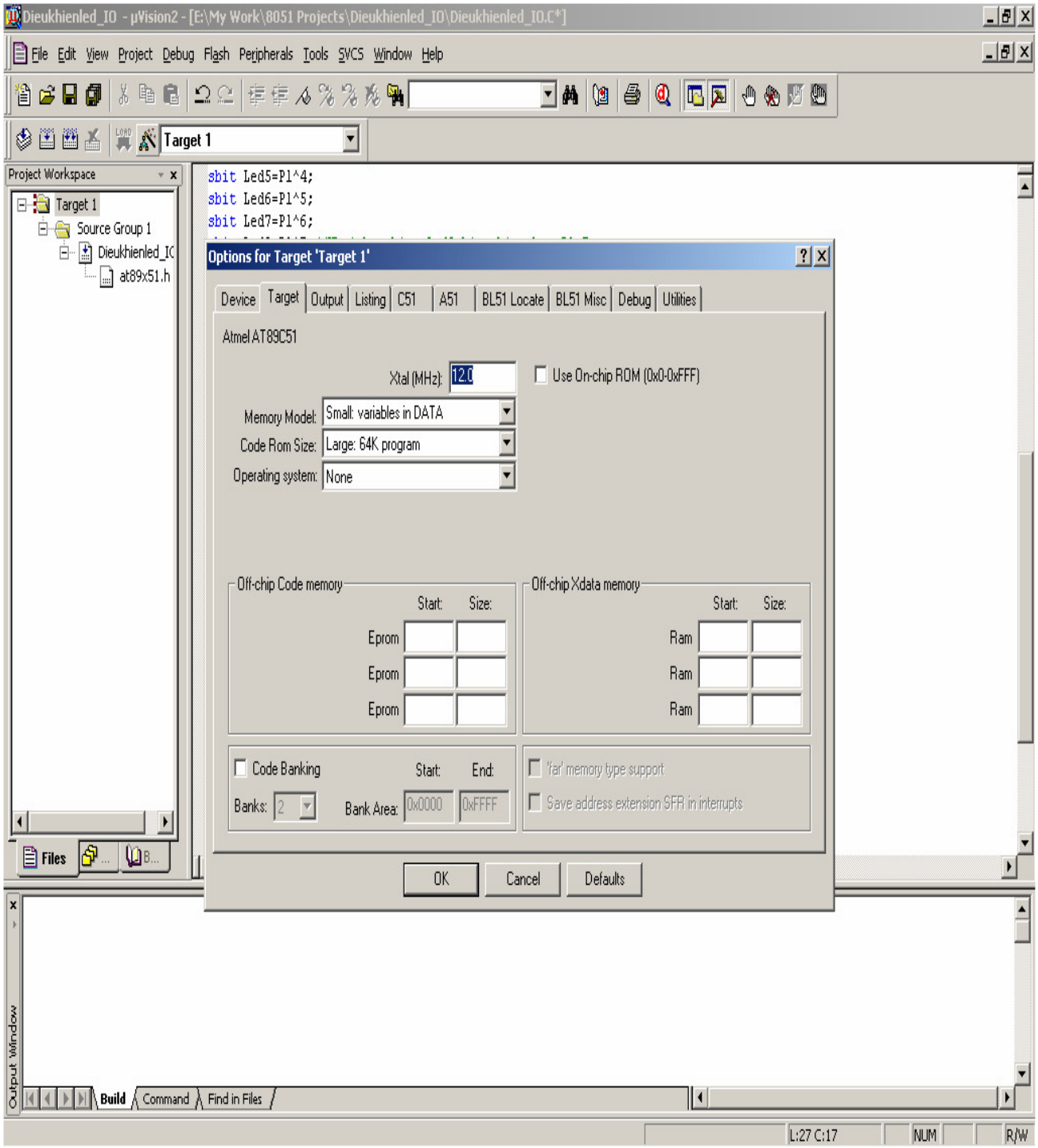

Dieukhienled_IO - µVision2 - [E:\My Work\8051 Projects\Dieukhienled_IO\Dieukhienled_IO.C*]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Target 1
Project Workspace
Target 1
Source Group 1
Dieukhienled_IC
at89x51.h
Files
/*************************Khai bao bien toan cuc**********************/
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 tre(long time)
{
long n;// Khai bao bien cuc bo
for(n=0; n++;n<time)//Lap time lan
{
; // Khong lam gi nop
}
}
Output Window
Build
Command
Find in Files
For Help, press F1
L:27 C:16
NUM
R/W



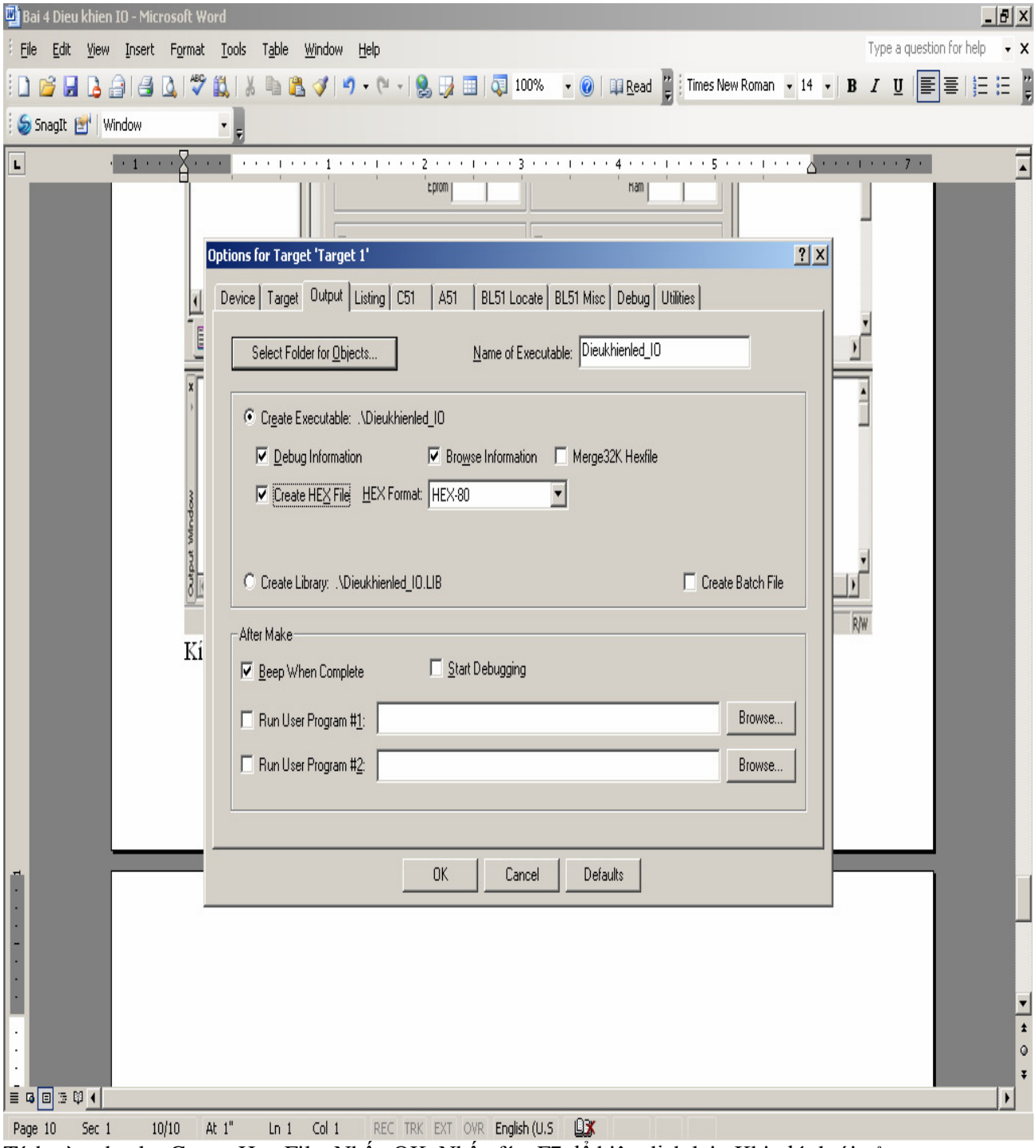
* Để có thể nạp chương trình vào chíp thì fải tạo ra file .hex. Để tạo ra file .hex làm như sau . Vào Option for target chỗ chỉnh tần số thạch anh.



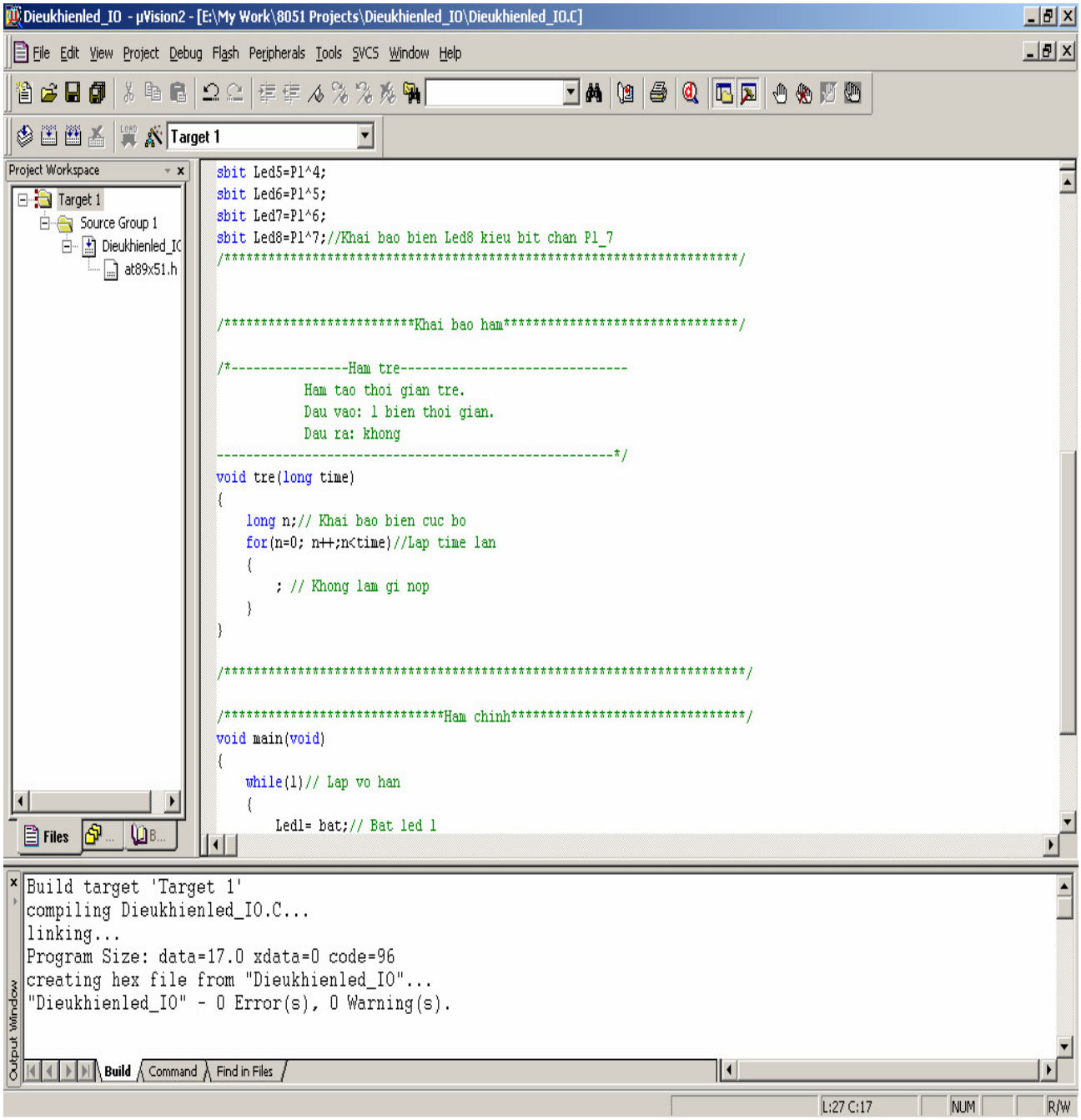
Được hình sau:



Kích vào tab Output. Được hình sau:



Tích vào check : Create Hex File. Nhấp OK. Nhấn fím F7 để biên dịch lại . Khi đó dưới cửa sổ output window được chữ Creating hex file…



Creating hex file from "Dieukhienled_IO" như ngay trên hình.

3.4.Nạp chíp:

Cắm đầu cổng COM vào cổng COM máy tính.

Cắm nguồn vào mạch nạp.

Cho chíp vào socket 40 chân màu xanh. Rất chú ý phải cho đúng chiều chíp không là hỏng. Chiều chíp giống chiều con chíp có sẵn trong mạch(Chíp MASTER).

Mở phần mềm EZDL4 lên. Thấy có chữ identifing target chip …. Nháy. Gạt cần nhở ở socket 40 chân để kẹp chíp cho tiếp xúc.

Trên EZDL4 sẽ thấy chữ AT89C51 hoặc AT89C52 tùy các bạn dùng loại chíp nào.

Kích vào Send. Chọn đường dẫn đến thư mục lưu project của bạn chọn file : Dieukhienled_IO.hex . Nhấn OK. Chờ mạch báo chữ Complete thì gạt nẫy trên socket lấy chíp ra cắm vào mạch.

3.5. Kết quả:

Cắm nguồn vào mạch . Chú ý không cắm ngược âm dương.

Nếu mạch không chạy rút nguồn ra kiểm tra các chỗ sau:

1) Kiểm tra mạch dao động xem tụ đúng giá trị không, lắp đúng sơ đồ không, lắp có lỏng không.
2) Kiểm tra xem tụ ở mạch reset chân 9 lắp đúng cực không, có lỏng không. Dùng đồng hồ đo chân 9 nếu 0V hoặc xấp xỉ 0V là OK.
3) Trở băng lắp đúng chiều chưa, chân 1 vào chân 40. Dùng đồng hồ đo chân 31(EA) xem có bằng 5V không , nếu 5V thì OK.
4) Kiểm tra chân 20 nối GND chưa, chân 40 nối +5V chưa dùng đồng hồ để đo điện áp.
5) Kiểm tra xem lắp đúng chiều led chưa.
6) Các bạn có thể lắp lỏng chíp hoặc do chất lượng board lấy tay tì mạnh con chíp xuống board .

Nếu vẫn không chạy thì liên lạc với tôi.

3.6. Điều khiển 8 led từng chiếc 1:

Các bạn có thể sửa lại hàm main như sau:

```
void main(void)
{
	while(1)// Lap vo han
	{
		Led1= bat;// Bat led 1
		tre(1000);// Tre 1 khoang thoi gian
		Led1= tat;// Tat led 1
		tre(1000);// Tre 1 khoang thoi gian
		Led2= bat;// Bat led 2
		tre(1000);// Tre 1 khoang thoi gian
		Led2= tat;// Tat led 2
		tre(1000);// Tre 1 khoang thoi gian
		Led3= bat;// Bat led 3
		tre(1000);// Tre 1 khoang thoi gian
		Led3= tat;// Tat led 3
		tre(1000);// Tre 1 khoang thoi gian
		Led4= bat;// Bat led 4
		tre(1000);// Tre 1 khoang thoi gian
		Led4= tat;// Tat led 4
		tre(1000);// Tre 1 khoang thoi gian
		Led5= bat;// Bat led 5
		tre(1000);// Tre 1 khoang thoi gian
		Led5= tat;// Tat led 5
		tre(1000);// Tre 1 khoang thoi gian
		Led6= bat;// Bat led 6
		tre(1000);// Tre 1 khoang thoi gian
```

```
            Led6= tat;// Tat led 6
            tre(1000);// Tre 1 khoang thoi gian
            Led7= bat;// Bat led 7
            tre(1000);// Tre 1 khoang thoi gian
            Led7= tat;// Tat led 7
            tre(1000);// Tre 1 khoang thoi gian
            Led8= bat;// Bat led 8
            tre(1000);// Tre 1 khoang thoi gian
            Led8= tat;// Tat led 8
            tre(1000);// Tre 1 khoang thoi gian
        }

}
```

Để điều khiển 8 led.
Với chương trình này các bạn có thể cho thứ tự các led tắt bật khác nhau để có các kiểu nháy khác nhau.

3.7. Điều khiển out cả cổng:

Nếu các bạn nhàm chán với việc điều khiển từng chân 1 viết code rất tốn công các bạn có thể xuất giá trị ra cả cổng.
Trước hết các bạn cần nắm các điều như sau:

- 1 cổng có 8 bit tổ hợp 8bit có $2^8 = 256$ trạng thái. Khi các bạn đưa ra cổng 1 giá trị a(thập phân) từ 0 đến 255 thì số a sẽ được đổi ra hệ nhị phân rồi đưa ra các bit(chân) của cổng. Ví dụ:

Nếu có lệnh: P1=1; vì $1_{(10)} = 0000\ 0001_{(2)}$ nên chân P1_0(bit 0) sẽ bằng 1(5V) còn lại các từ P1_1(bit 1) đến P1_7(bit 7) sẽ bằng 0(0V).

P1=10; vì $10_{(10)} = 0000\ 1001_{(2)}$ thì sẽ có P1_0 và P1_3 bằng 1(5V) còn lại các chân khác sẽ là 0(0V).

- Các bạn có thể đưa ra cổng 1 giá trị số hex từ 0 đền ff tương ứng từ 0 đến 255.

Các số cơ sở trong hệ hex.

(HEX)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
(10)											10	11	12	13	14	15

Cách đổ số hex ra số thập phân: có số hex : $N_{(16)}$=abf1 đổi ra hệ số 10
$N_{(10)}=1.16^0 + 15.16^1 + 11.16^2 + 10.16^3$= Bấm máy tính hộ nhé.
Đổi số nhị phân sang hex: Gộp 4 số nhị phân thành 1 số hex
Ví dụ: $0010\ 0001_{(2)} = 21_{(16)}$ 4 số đầu có bit 1 = 1 nên $1x2^1= 2$
4 số sau có bit 0 =1 nên $1x2^0=1$.
"Các bạn thấy vất vả với phần này dù bạn có hiểu hay không hiểu 1 lát nữa sẽ biết cách làm liền."
Cách đưa ra như sau:
Ví dụ lệnh P1=1; tương đương với P1=0x01;
P1=10; tương đương với P1=0x0A;

Chương trình xuất ra cả cổng tương đương với chương trình điều khiển 8 led từng cái 1 như sau:

```
void main(void)
{
    while(1)// Lap vo han
    {
```

```
                P1=0x01;// Bat led 1
                tre(1000);// Tre 1 khoang thoi gian
                P1=0x00;// Tat led 1
                tre(1000);// Tre 1 khoang thoi gian
                P1=0x02;// Bat led 2
                tre(1000);// Tre 1 khoang thoi gian
                P1=0x00;// Tat led 2
                tre(1000);// Tre 1 khoang thoi gian
                P1=0x04;// Bat led 3
                tre(1000);// Tre 1 khoang thoi gian
                P1=0x00;// Tat led 3
                tre(1000);// Tre 1 khoang thoi gian
                P1=0x08;// Bat led 4
                tre(1000);// Tre 1 khoang thoi gian
                P1=0x00;// Tat led 4
                tre(1000);// Tre 1 khoang thoi gian
                P1=0x10;// Bat led 5
                tre(1000);// Tre 1 khoang thoi gian
                P1=0x00;// Tat led 5
                tre(1000);// Tre 1 khoang thoi gian
                P1=0x20;// Bat led 6
                tre(1000);// Tre 1 khoang thoi gian
                P1=0x00;// Tat led 6
                tre(1000);// Tre 1 khoang thoi gian
                P1=0x40;// Bat led 7
                tre(1000);// Tre 1 khoang thoi gian
                P1=0x00;// Tat led 7
                tre(1000);// Tre 1 khoang thoi gian
                P1=0x80;// Bat led 8
                tre(1000);// Tre 1 khoang thoi gian
                P1=0x00;// Tat led 8
                tre(1000);// Tre 1 khoang thoi gian
        }

}
```

Như vậy gõ code vẫn mỏi tay lắm để đạt được mục đích 8 đèn nháy liên tiếp các bạn có thể làm như sau:

```
/******************Ham chinh*****************************/
void main(void)
{
unsigned char n; // Khai bao them bien n cho vong for
        while(1)// Lap vo han
        {
                P1=0x01;// Bat led 1
                for(n=0 ; n<8;n++)// Lap 8 lan
                {
                        P1=P1<<1; // Dich bit xang trai
```

```
                    tre(1000);
                }
            }

        }

        /*************************************************************/
```

Debug quan sát sự thay đổi của cổng 1 để thấy được lợi hại của phép dịch bit xang trái.

Để hiểu thao tác xuất ra cổng , chân, các các bạn làm cho mình 1 ví dụ nữa như sau:

```
/***********************Ham chinh*********************************/
void main(void)
{
unsigned char n; // Khai bao them bien n cho vong for
        while(1)// Lap vo han
        {
            P1=0x01;// Bat led 1
            for(n=0 ; n<256;n++)// Lap 8 lan
            {
                P1=n; // Dich bit xang trai
                tre(5000);
            }
        }

}

/*************************************************************/
```

Debug để thấy sự thay đổi các cổng.

3.8. Kinh nghiệm :

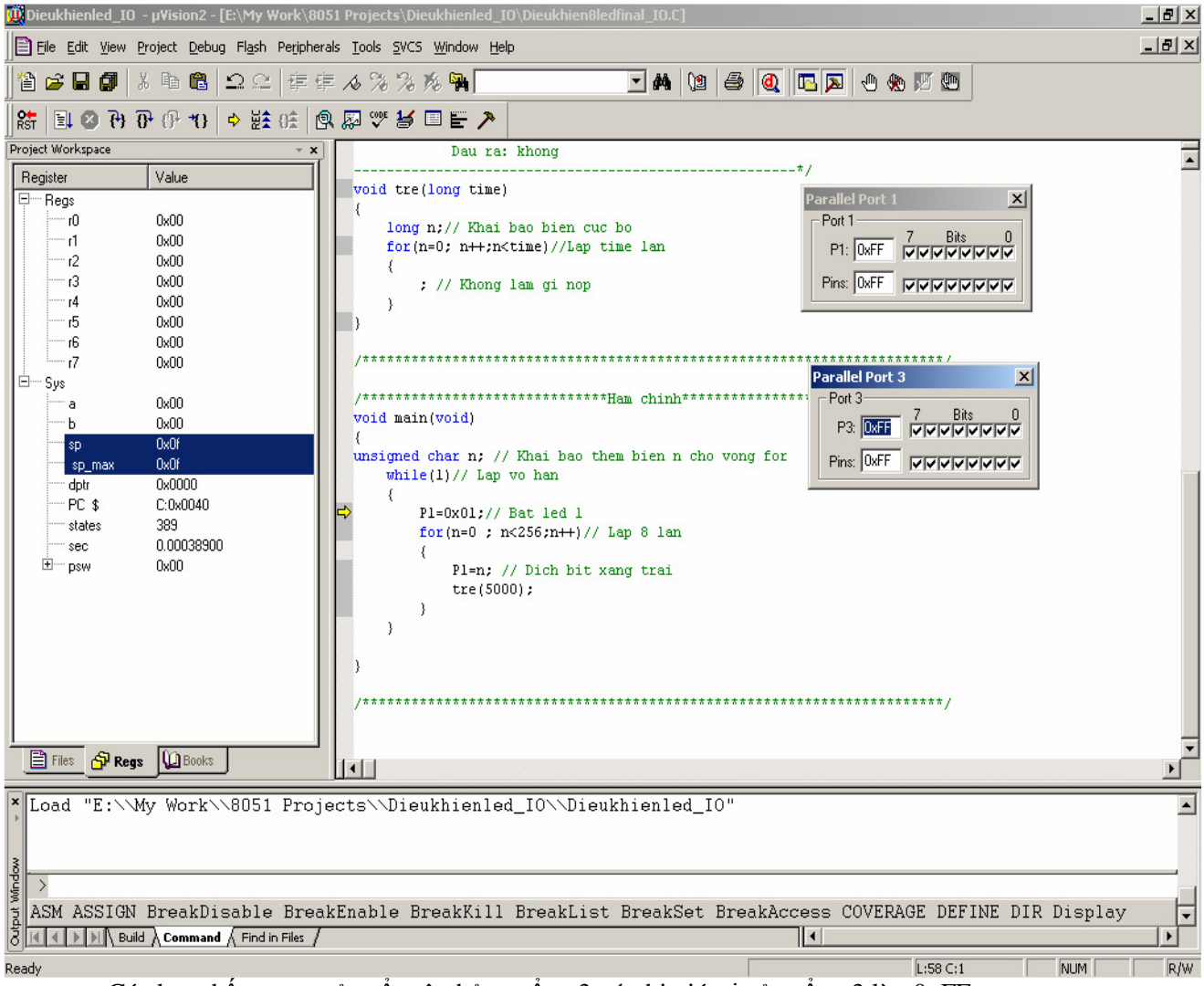
*Để chuyển đổi giữa các hệ sô nhanh các bạn dùng cái calculator có ngay trong window. Chọn Start → All programs → Accessories → Calculator.

Trong Calculator các bạn chọn menu View → Chọn scienctific. Cách chuyển đổi tự khám phá nhé.

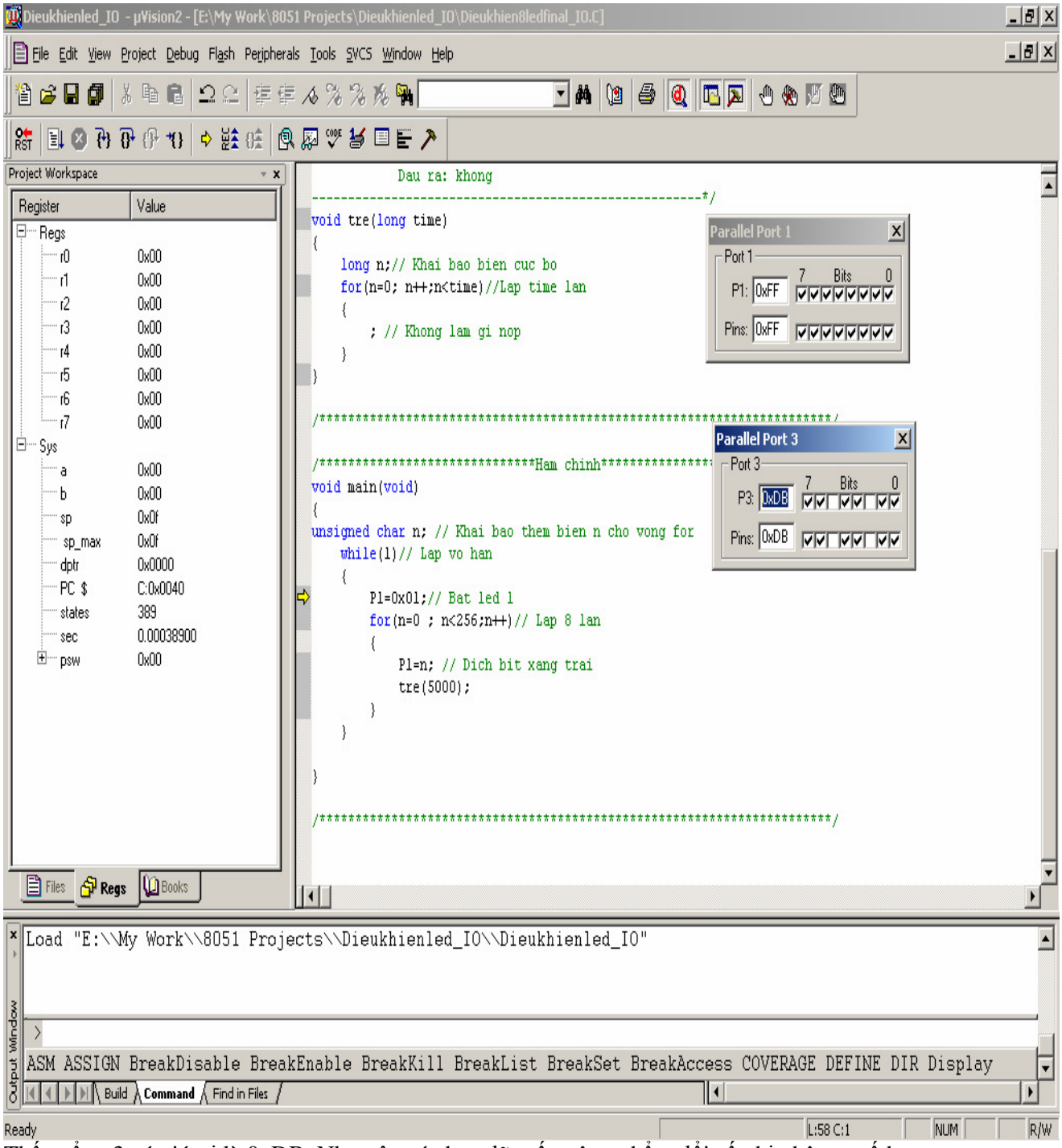
* Để tính ra số hex nhanh nhất .

Tôi ví dụ muốn đưa chân P3.5 và chân P3.2 xuống 0(0V) còn các chân còn lại ta làm như sau:

Cứ cho P3= 100 ; hay 1 giá trị bất kì .Vì code dịch không lỗi cú pháp mới debug được nên phải cho giá trị bất kì vào. Dịch chương trình rồi nhấn Debug. Ra được như sau:



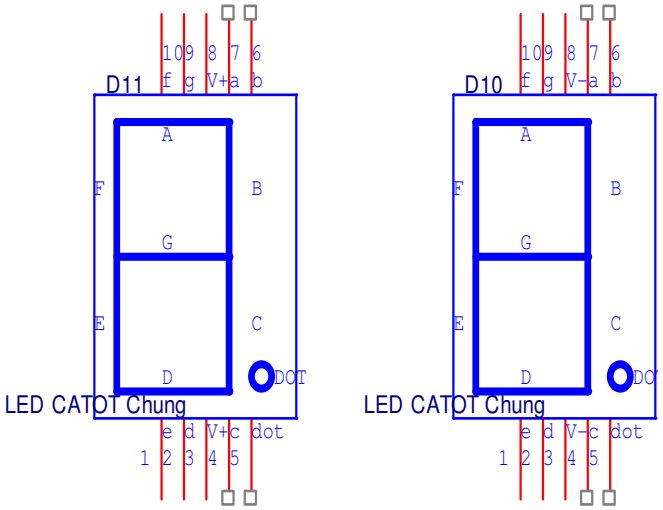
Các bạn thấy trong cửa sổ mô phỏng cổng 3 có ghi giá trị của cổng 3 là : 0xFF tương ứng tất cả 8 bit là 1111 1111. Giờ các bạn bỏ dấu tích trên chân 3.5 và 3.2 đi . Đếm từ trái qua phải nhé.



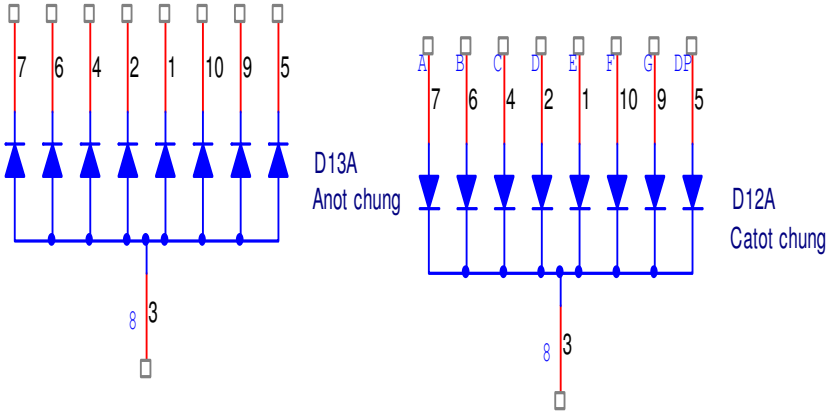
Thấy cổng 3 có giá trị là 0xDB. Như vậy các bạn đỡ mất công nhẩm đổi số nhị phân ra số hex. Nếu không dùng cách này để là như trên các bạn phải nghĩ trong đầu ra được dãy số 1101 1011 rồi đổi qua số HEX kết quả sẽ là 0xDB nhưng mệt lắm.

Bài 4: Điều khiển Led 7 thanh Anot chung

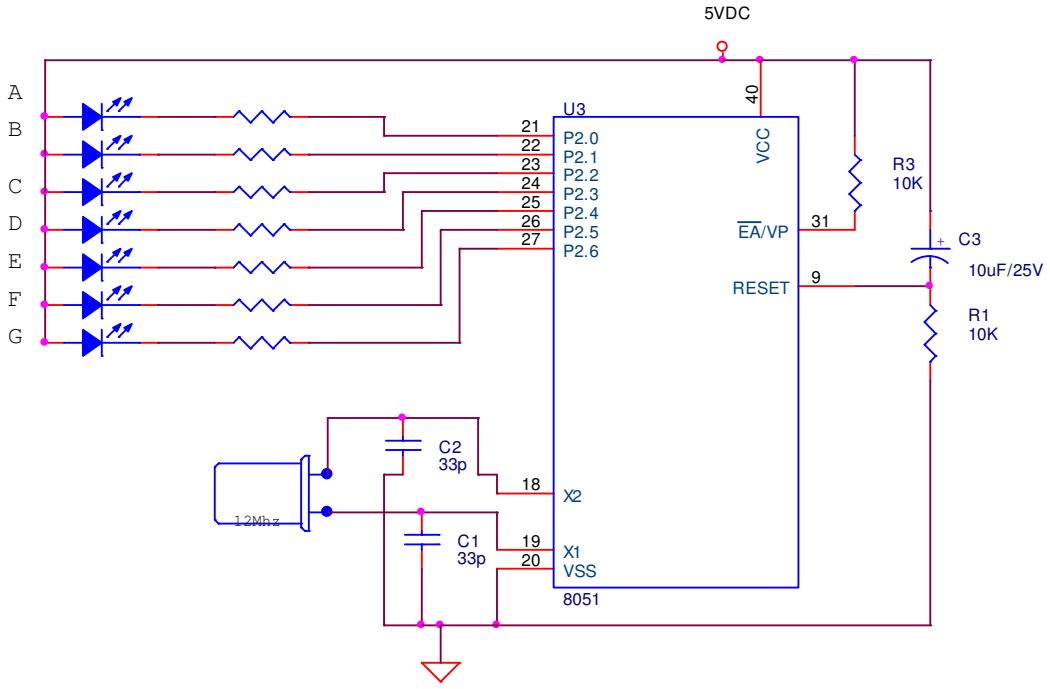
4.1 Lắp mạch :



Có hai loại led 7 thanh: Anốt chung và Catốt chung. Hình trên là sơ đồ chân của hai loại led. Nó có cấu tạo như sau:



Chỉ là 8 con led đấu chung 1 đầu: Anốt hoặc Catốt.
Mạch lắp sử dụng led Anốt chung như sau:



- Lắp mạch: Mạch bài trước(Điều khiển led đơn) để nguyên,chỉ lắp thêm vào.

+ Bước1: Lắp 7 điện vào 7 chân cổng P2 từ P2.0 đến P2.7(Từ chân 21 → Chân 27) của vi điều khiển. Chú ý đếm bit của cổng 2 từ dưới lên trên.

+ Bước2: Lắp led 7 thanh 5 chân phía trên sông, 5 chân phía dưới sông(Giống lắp AT89C51).

+ Bước 3: Dùng dây điện câu từ các chân theo sơ đồ. Chân điều khiển thanh A vào điện trở nối vào P2.0 và cứ thế đến chân điều khiển thanh G vào điện trở nối vào P2.7. Bỏ qua chân điều khiển dấu chấm(Dot chân 5 của led 7 thanh).

+ Bước 4: Dùng dây câu chân 3(hoặc chân 8, hoặc cả hai chân 3 và 8) lên +5V. Lắp mạch nên để dây câu khác màu và rõ ràng tránh nhầm lẫn.

- Test led:

Cho nguồn vào(Dĩ nhiên led đơn bài trước vẫn nhấp nháy).

Rút 1 đầu điện trở nối với chân P2.0 của VĐK cắm xuống đất thấy led 7 thanh sáng đèn A,cắm lại điện trở về vị trí cũ. Tương tự test 6 thanh còn lại. Thanh nào không sáng thì kiểm tra lại xem cắm dây tiếp xúc chưa. Vẫn không sáng thì led đã hỏng thay led khác vào.

4.2. Nguyên lí hoạt động:

Khi cắm nguồn vào mạch tất cả các chân của các cổng IO của VĐK là 5V(Nếu cổng 0 không lắp điện trở treo thì sẽ là 0V). Nhìn sơ đồ mạch không có chênh lệch điện áp nên không có đèn nào sáng. Chúng ta muốn sáng thanh nào chỉ việc đưa ra điện áp 0V ở chân vi điều khiển nối với thanh đó.

	Thanh hiện	Thanh tắt	Giá trị(P2)
Để hiện thị số 1:	B,C	các thanh còn lại	1111 1001
Để hiện thị số 2:	A,B,D,E,G	các thanh còn lại	1010 0100
….			
Để hiện thị số 8:	Tất cả các thanh	không thanh nào	1000 0000

gfe dcba

Bít thứ 8 P2.7 không dùng.

Ngoài ra led 7 thanh còn có thể hiện thị 1 số chữ

Đê hiển thị chữ B: Giống sô 8

Hiển thị chữ A: A,B,C,E,F,G D 1000 1000

4.3.Lập trình :

Cách 1: Lập trình dễ hiểu không cần phải tính toán nhưng phải viết và copy, past và sửa nhìu.

Code như sau:

```
#include <AT89X52.H>
/* Khai bao cac bien bit gan voi chan vi dieu khien*/
sbit ThanhA = P2^0;
sbit ThanhB = P2^1;
sbit ThanhC = P3^5;
sbit ThanhD = P3^4;
sbit ThanhE = P3^3;
sbit ThanhF = P2^2;
sbit ThanhG = P2^3;
/* Khai bao bien */
long n;// Cho vong for

/* Khai bao ham */

/* Ham tre */
void delay(long time)
{
        for(n=0; n<time; n++)
        {
                ;
        }
}
```

```
/* Ham tat tat ca cac thanh */
void tat(void)
{
ThanhA =1;
ThanhB =1;
ThanhC =1;
ThanhD =1;
ThanhE =1;
ThanhF =1;
ThanhG =1;
}

/* Cac ham hien thi chu va so */

void so1(void)
{
tat();
ThanhA =1;
ThanhB =0;
ThanhC =0;
ThanhD =1;
ThanhE =1;
ThanhF =1;
ThanhG =1;
}

void so2(void)
{
tat();
ThanhA =0;
ThanhB =0;
ThanhC =1;
ThanhD =0;
ThanhE =0;
ThanhF =1;
ThanhG =0;
}
void so3(void)
{
tat();
ThanhA =0;
ThanhB =0;
ThanhC =0;
ThanhD =0;
ThanhE =1;
ThanhF =1;
ThanhG =0;
}

void so4(void)
{
tat();
ThanhA =1;
ThanhB =0;
ThanhC =0;
ThanhD =1;
ThanhE =1;
ThanhF =0;
ThanhG =0;
}

void so5(void)
{
tat();
ThanhA =0;
ThanhB =1;
ThanhC =0;
ThanhD =0;
ThanhE =1;
ThanhF =0;
ThanhG =0;
}

void so6(void)
{
tat();
ThanhA =0;
ThanhB =1;
ThanhC =0;
ThanhD =0;
ThanhE =0;
ThanhF =0;
ThanhG =0;
}

void so7(void)
{
tat();
```

```
ThanhA =0;
ThanhB =0;
ThanhC =0;
ThanhD =1;
ThanhE =1;
ThanhF =1;
ThanhG =1;
}

void so8(void)
{
tat();
ThanhA =0;
ThanhB =0;
ThanhC =0;
ThanhD =0;
ThanhE =0;
ThanhF =0;
ThanhG =0;
}

void so9(void)
{
tat();
ThanhA =0;
ThanhB =0;
ThanhC =0;
ThanhD =0;
ThanhE =1;
ThanhF =0;
ThanhG =0;
}

void chuA(void)
{
tat();
ThanhA =0;
ThanhB =0;
ThanhC =0;
ThanhD =1;
ThanhE =0;
ThanhF =0;
ThanhG =0;
}

void chuB(void)
{
tat();
ThanhA =0;
ThanhB =0;
ThanhC =0;
ThanhD =0;
ThanhE =0;
ThanhF =0;
ThanhG =0;
}

void chuC(void)
{
tat();
ThanhA =0;
ThanhB =1;
ThanhC =1;
ThanhD =0;
ThanhE =0;
ThanhF =0;
ThanhG =1;
}

void chuD(void)
{
tat();
ThanhA =0;
ThanhB =0;
ThanhC =0;
ThanhD =0;
ThanhE =0;
ThanhF =0;
ThanhG =1;
}

void chuE(void)
{
tat();
ThanhA =0;
ThanhB =1;
ThanhC =1;
ThanhD =0;
```

```
ThanhE =0;
ThanhF =0;
ThanhG =0;
}

void chuF(void)
{
tat();
ThanhA =0;
ThanhB =1;
ThanhC =1;
ThanhD =1;
ThanhE =0;
ThanhF =0;
ThanhG =0;
}
void chuG(void)
{
tat();
ThanhA =0;
ThanhB =1;
ThanhC =0;
ThanhD =0;
ThanhE =0;
ThanhF =0;
ThanhG =1;
}

void chuH(void)
{
tat();
ThanhA =1;
ThanhB =0;
ThanhC =0;
ThanhD =1;
ThanhE =0;
ThanhF =0;
ThanhG =0;
}

void chuI(void)
{
tat();
ThanhA =1;
ThanhB =1;
ThanhC =1;
ThanhD =1;
ThanhE =0;
ThanhF =0;
ThanhG =1;
}

void chuL(void)
{
tat();
ThanhA =1;
ThanhB =1;
ThanhC =1;
ThanhD =0;
ThanhE =0;
ThanhF =0;
ThanhG =1;
}

void chuO(void)
{
tat();
ThanhA =0;
ThanhB =0;
ThanhC =0;
ThanhD =0;
ThanhE =0;
ThanhF =0;
ThanhG =1;
}

void chuP(void)
{
tat();
ThanhA =0;
ThanhB =0;
ThanhC =1;
ThanhD =1;
ThanhE =0;
ThanhF =0;
ThanhG =0;
}
```

```
void chuR(void)
{
tat();
ThanhA =0;
ThanhB =0;
ThanhC =0;
ThanhD =1;
ThanhE =0;
ThanhF =0;
ThanhG =0;
}

void chuS(void)
{
tat();
ThanhA =0;
ThanhB =1;
ThanhC =0;
ThanhD =0;
ThanhE =1;
ThanhF =0;
ThanhG =0;
}
```

```
void chuU(void)
{
tat();
ThanhA =1;
ThanhB =0;
ThanhC =0;
ThanhD =0;
ThanhE =0;
ThanhF =0;
ThanhG =1;
}

void chuY(void)
{
tat();
ThanhA =1;
ThanhB =0;
ThanhC =0;
ThanhD =0;
ThanhE =1;
ThanhF =0;
ThanhG =0;
}
```

```
/* Ham chinh */
void main(void)
{
while(1)
{
so0();
delay(20000);
so1();
delay(20000);
so2();
delay(20000);
so3();
delay(20000);
so4();
delay(20000);
so5();
delay(20000);
so6();
delay(20000);
so7();
delay(20000);
so8();
delay(20000);
so9();
delay(20000);
chuA();
delay(20000);
chuB();
delay(20000);
chuC();
delay(20000);
chuD();
delay(20000);
chuE();
delay(20000);
chuF();
delay(20000);
chuG();
delay(20000);
chuH();
delay(20000);
chuI();
delay(20000);
chuL();
```

```
delay(20000);
chuO();
delay(20000);
chuP();
delay(20000);
chuR();
delay(20000);
chuS();
delay(20000);
chuU();
delay(20000);
chuY();
delay(20000);
}
}
```

Cách 2: Các bạn viết 1 chương trình đơn giản rồi dùng công cụ Debug để xem số hex rồi viết vào rất ngắn gọn.

Ví dụ:

Hàm hiển thị số 1:

```
void so1(void)
{
tat();
P2=0xF5;
}
```

Các bạn debug cho hiển thị cổng P2 lên. Để dấu tích ở các đèn tắt(1) , bỏ dấu tích ở các đèn cần bật(0). Rồi đọc giá trị hex như tôi hướng dẫn ở bài trước.

4.4.Nạp chíp:

Đã hướng dẫn. Nếu các bạn gặp phải vấn đề của mạch nạp. Thường là:

+ Treo chip Master: Rút nguồn mạch nạp ra, đợi chút rồi cắm lại.
+ Không nhận cổng COM. Các bạn nhấp chuột phải vào Mycomputer, chọn Properties → Device manage. Nhấn vào Scan for hardware change để máy tính nhận lại cổng COM. Rồi khởi động lại máy.
+ Lý do khác các bạn thử kiểm tra phần cứng mạch nạp.
+ Vẫn không được tôi chịu.

Từ bài sau sẽ không có phần này.

4.5.Kết quả:

Nhìn vào hàm main các bạn thấy chương trình sẽ hiển thị số 1 ở led 7 thanh, trễ 1 khoảng thời gian, rồi nhảy sang số 2 đến sô 9 , rồi đến chữ. Nếu các bạn không có hàm trễ thì led sẽ hiển thị ra sao?

Led sẽ hiện số 8. Vì tất cả 8 led được bật tắt quá nhanh (chỉ cần >24 Hz) do hiện tượng lưu ảnh mắt các bạn sẽ nhìn thấy tất cả các led sáng chứ không hiện số.

4.6.Kinh nghiệm :

Sau khi đã thực hiện theo các rút gọn như viết hàm số 1 ở trên các bạn áp dụng cấu trúc lệnh switch case để viết lại chương trình thì chương trình sẽ rất gọn.

```
void Hienthiled(unsigned char x) // Co 1 bien dau vao de xac dinh xem la hien thi so nao
{
        switch(x)
        {
                case 1: { tat(); P2=0xF5; break;} // So 1
                case 2: { tat(); P2=0xFF; break;}// So 2
                …
                case 9: { tat(); P2=0xFF; break;}// So 9
                case 10: { tat(); P2=0xFF; break;}// Chu A
                ….
                case 20: { tat(); P2=0xFF; break; }// Chu Y
        }
}
```

Các giá trị ở trên chỉ là ví dụ các bạn đã rút gọn và tự copy vào. Với hàm hiển thị led các bạn đã viết để hiện các số và các chữ giờ hàm main chỉ cần như sau:

```
void main (void)
{
while(1)
{
        for(n=0; n<20; n++)
        {
                Hienthiled(n);
                delay(20000);
        }
}
}
```

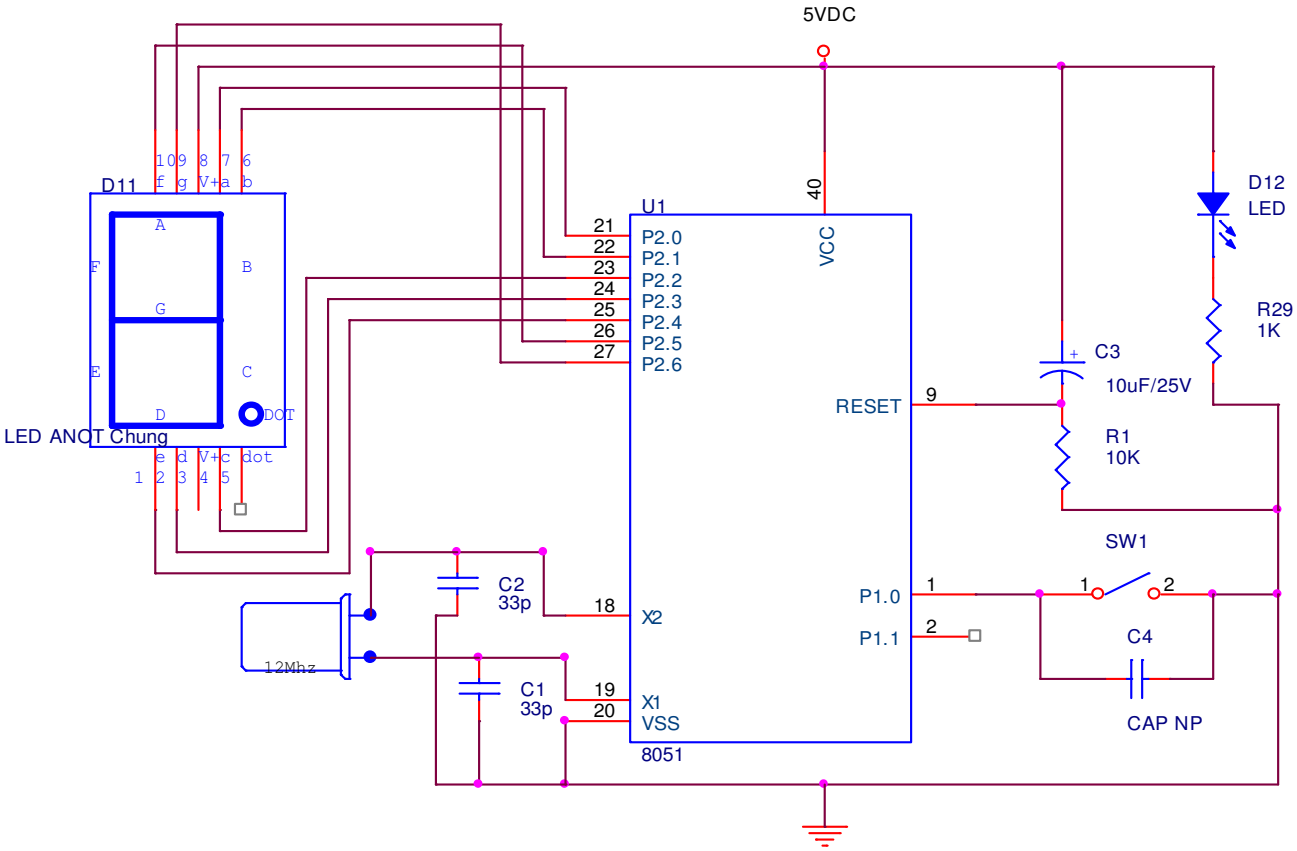
Bài 5:Đọc phím

Phần 1: Đọc 1 phím.

Nhiệm vụ:

Đếm số lần phím bấm giới hạn từ 0 đến 9 hiển thị ra led 7 thanh.

5.1.Lắp mạch như sau:



- Mạch bài 5 led 7 thanh giữ nguyên.

Lắp mạch: quá đơn giản.

Bước 1: Lắp 1 nút bấm ở đâu đó trên mạch.

Bước 2: Dùng dây câu 1 chân nút bấm với P1.0(Thay thế led bài 4).

Bước 3: Dùng dây câu 1 chân nút bấm xuống đất.

Bước 4: Lắp 1 tụ 104 giữa 2 chân nút bấm.

Bước 5: Lắp trở vào chân P1.0 và led từ +5V vào đầu trở còn lại(Chú ý đúng chiều led.

- Test nút bấm: Cắm nguồn vào, nhấn nút đèn sáng. Nhả nút đèn tắt.

5.2.Nguyên lí hoạt động:

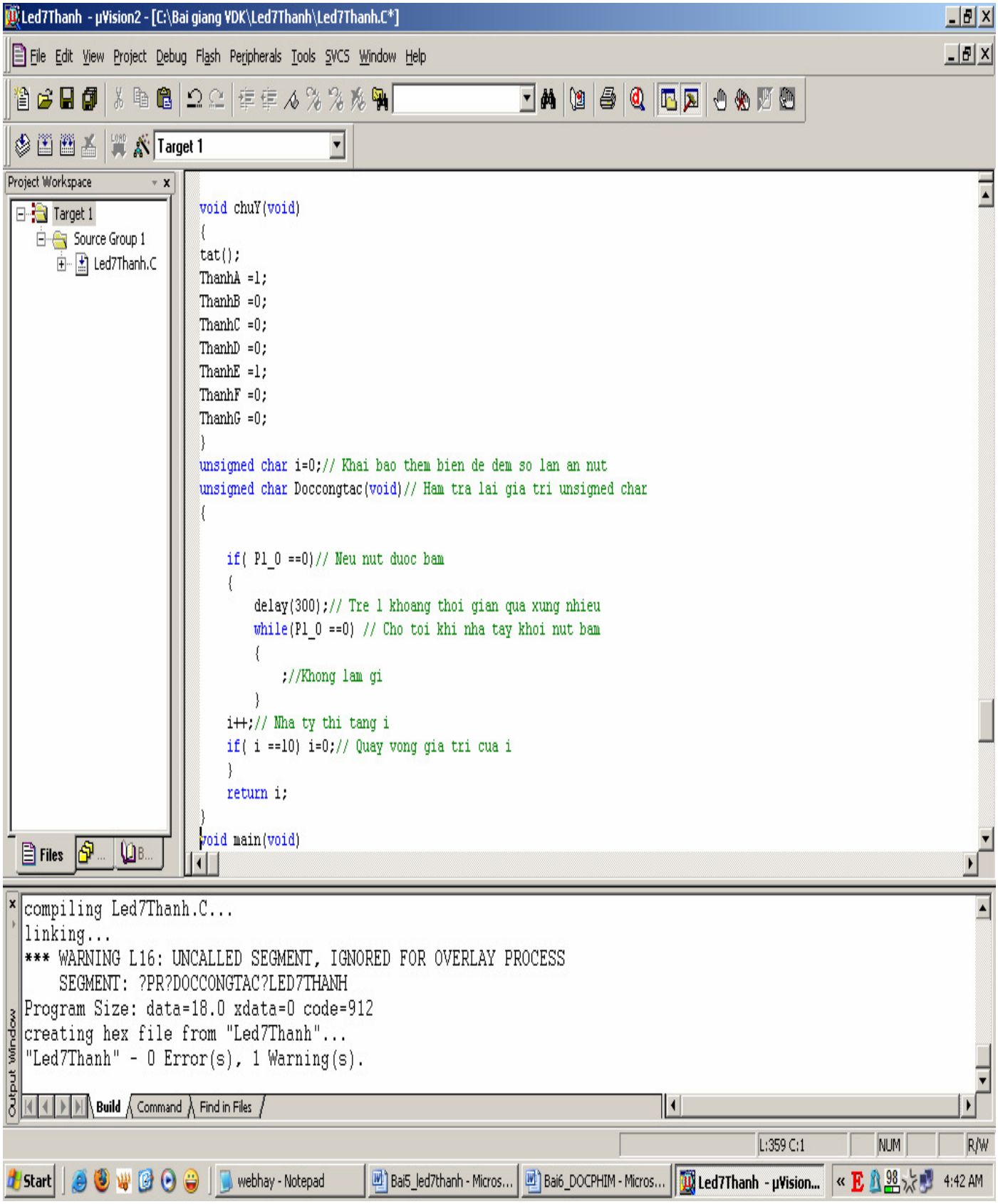
- Phần nút bấm: (khi không có tụ 104) ban đầu chân P1.0 ở mức cao +5V, nếu bấm nút 2 đầu nút bấm thông với nhau. Chân P1.0 thông với GND. Led sáng do có chênh áp. Chân P1.0 thông đất.

Nếu có tụ 104 tụ điện được nạp điện, khi bấm nút tụ điện sẽ phóng điện từ cực dương sang cực âm làm chân P1.0 thông với GND nhưng lâu về 0 V hơn 1 chút(trễ cứng).

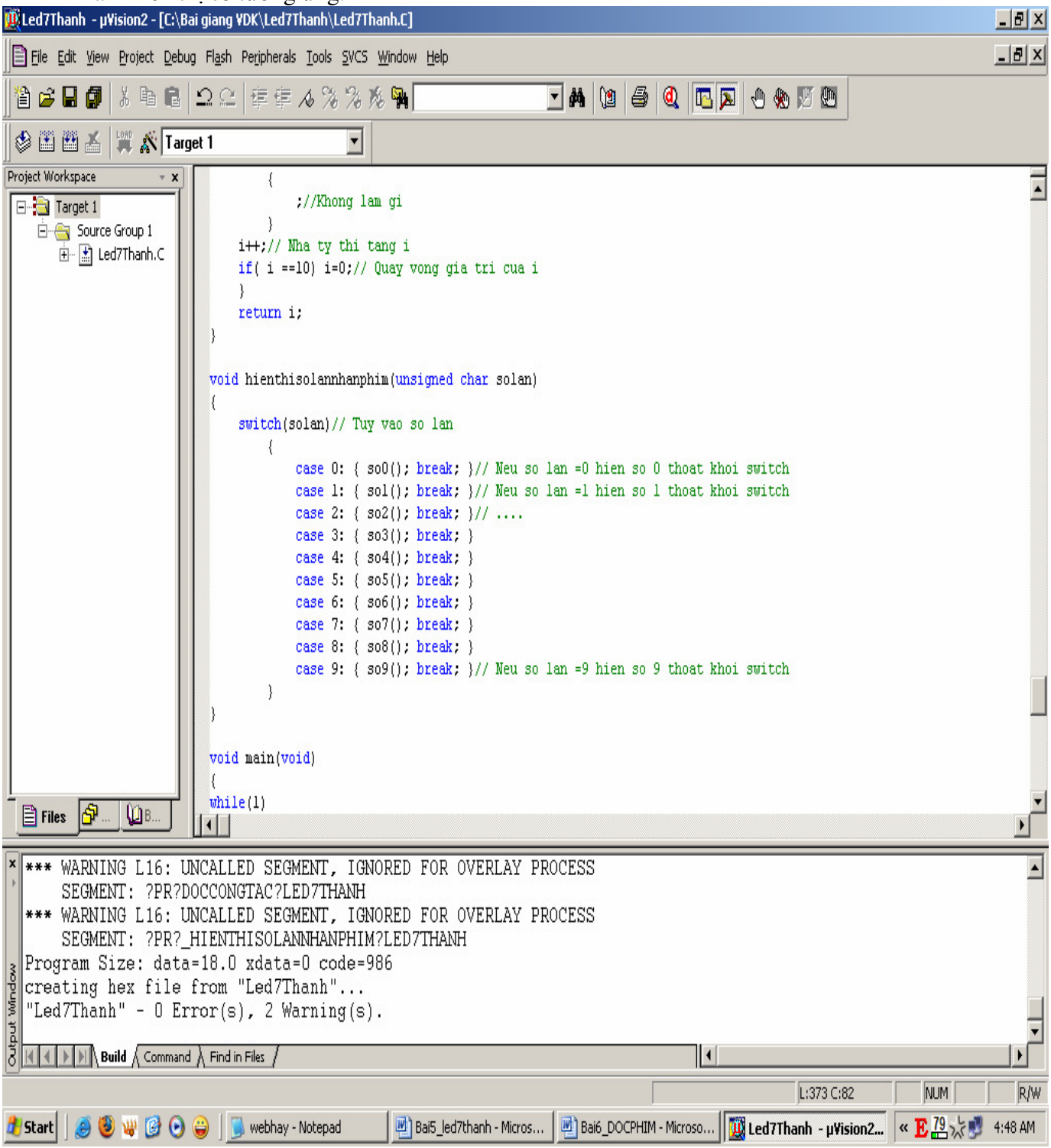
- Khi bấm nút theo nguyên lí thi bấm 1 cái là xuống 0 liền, nhưng do tiếp điểm cơ khí của nút bấm nên khi bấm nút nó sẽ có 1 số xung điện chứ không phải là bấm cái là nó xuống 0 luôn. Tụ 104 để giảm nhiễu đó. Tụ 104 cũng có thể bỏ đi không lắp vì ta có thể khử nhiễu bằng phần mềm.

5.3.Lập trình:

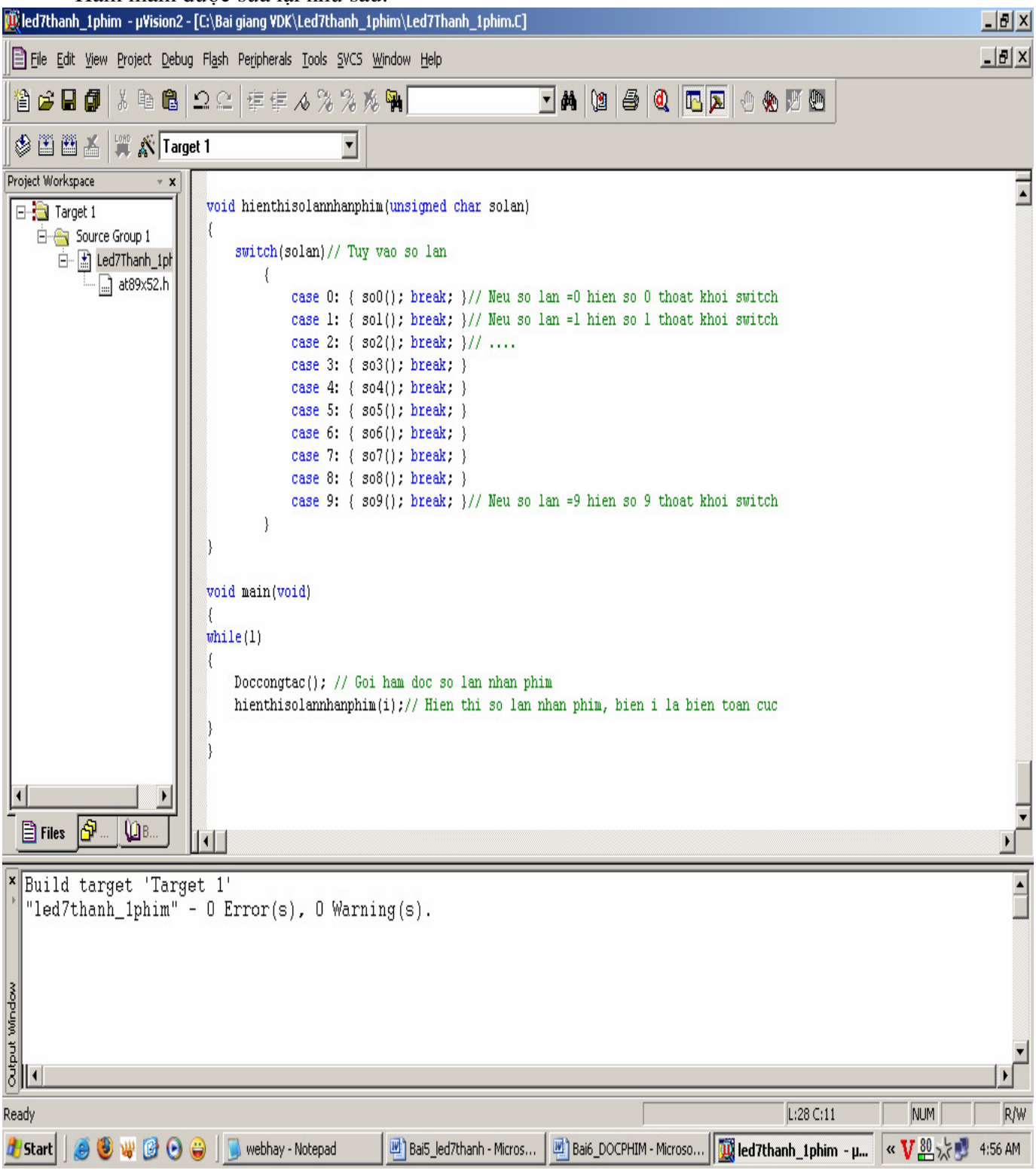
Code bài 5 giữ nguyên: soạn thêm một số hàm như sau hàm đọc phím bấm.
Hàm đọc số lần ấn phím


Led7Thanh - µVision2 - [C:\Bai giang VDK\Led7Thanh\Led7Thanh.C*]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Target 1
Project Workspace
Target 1
Source Group 1
Led7Thanh.C
Files
void chuY(void)
{
tat();
ThanhA =1;
ThanhB =0;
ThanhC =0;
ThanhD =0;
ThanhE =1;
ThanhF =0;
ThanhG =0;
}
unsigned char i=0;// Khai bao them bien de dem so lan an nut
unsigned char Doccongtac(void)// Ham tra lai gia tri unsigned char
{
if(P1_0 ==0)// Neu nut duoc bam
{
delay(300);// Tre 1 khoang thoi gian qua xung nhieu
while(P1_0 ==0) // Cho toi khi nha tay khoi nut bam
{
;//Khong lam gi
}
i++;// Nha ty thi tang i
if(i ==10) i=0;// Quay vong gia tri cua i
}
return i;
}
void main(void)
compiling Led7Thanh.C...
linking...
*** WARNING L16: UNCALLED SEGMENT, IGNORED FOR OVERLAY PROCESS
SEGMENT: ?PR?DOCCONGTAC?LED7THANH
Program Size: data=18.0 xdata=0 code=912
creating hex file from "Led7Thanh"...
"Led7Thanh" - 0 Error(s), 1 Warning(s).
Output Window
Build
Command
Find in Files
L:359 C:1
NUM
R/W
Start
webhay - Notepad
Bai5_led7thanh - Micros...
Bai6_DOCPHIM - Micros...
Led7Thanh - µVision...
4:42 AM

Hàm hiển thị số tương ứng.



Hàm main được sửa lại như sau:



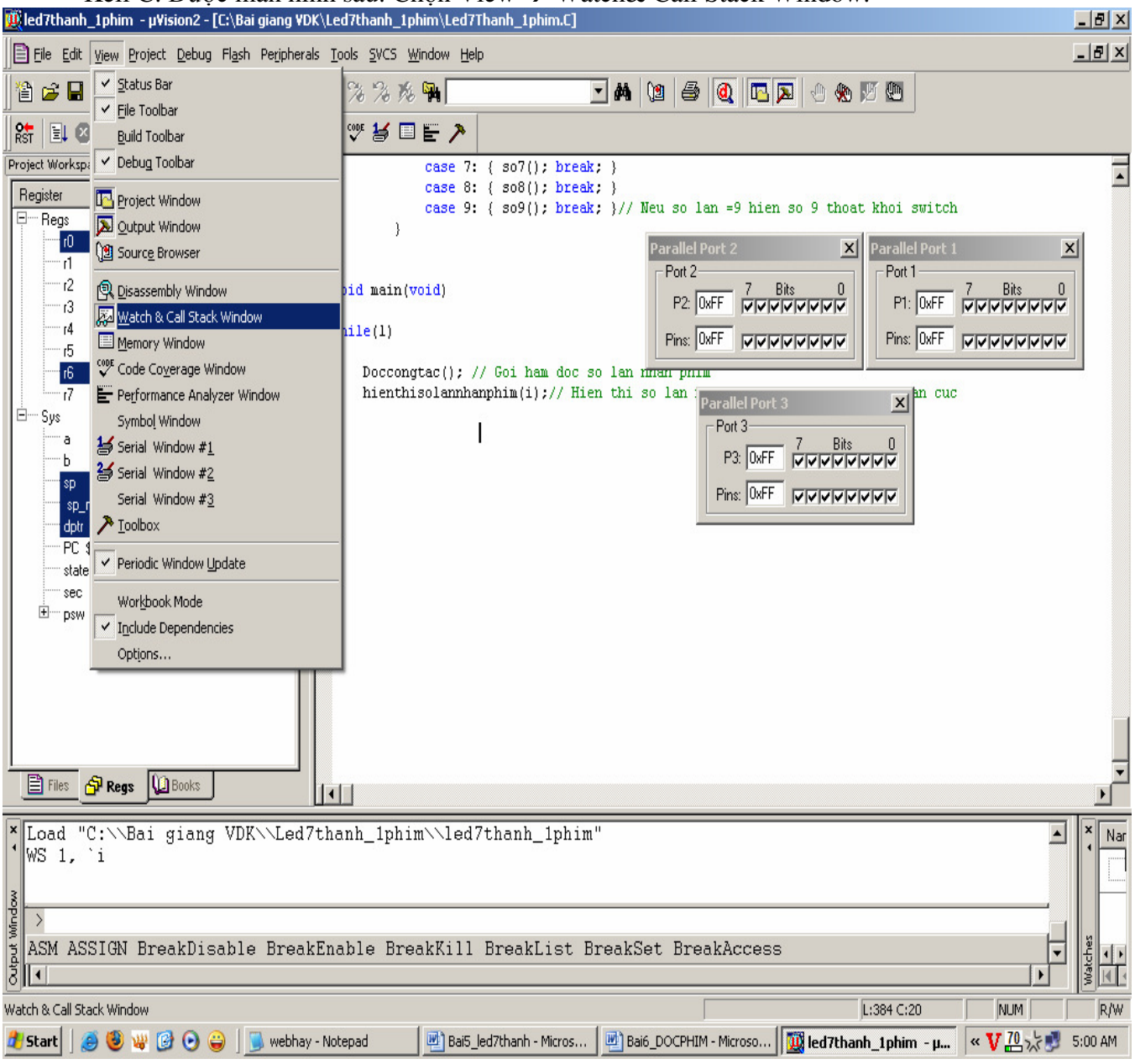
Đây là code các hàm bổ sung:

```
unsigned char i=0;// Khai bao them bien toan cuc de dem so lan an nut
unsigned char Doccongtac(void)// Ham tra lai gia tri unsigned char
{

        if( P1_0 ==0)// Neu nut duoc bam
        {
                delay(300);// Tre 1 khoang thoi gian qua xung nhieu
                while(P1_0 ==0) // Cho toi khi nha tay khoi nut bam
                {
                        ;//Khong lam gi
                }
        i++;// Nha ty thi tang i
        if( i ==10) i=0;// Quay vong gia tri cua i
        }
        return i;
}

void hienthisolannhanphim(unsigned char solan)
{
        switch(solan)// Tuy vao so lan
                {
                        case 0: { so0(); break; }// Neu so lan =0 hien so 0 thoat khoi switch
                        case 1: { so1(); break; }// Neu so lan =1 hien so 1 thoat khoi switch
                        case 2: { so2(); break; }// ....
                        case 3: { so3(); break; }
                        case 4: { so4(); break; }
                        case 5: { so5(); break; }
                        case 6: { so6(); break; }
                        case 7: { so7(); break; }
                        case 8: { so8(); break; }
                        case 9: { so9(); break; }// Neu so lan =9 hien so 9 thoat khoi switch
                }
}

void main(void)
{
while(1)
{
        Doccongtac(); // Goi ham doc so lan nhan phim
        hienthisolannhanphim(i);// Hien thi so lan nhan phim, bien i la bien toan cuc
}
}
```

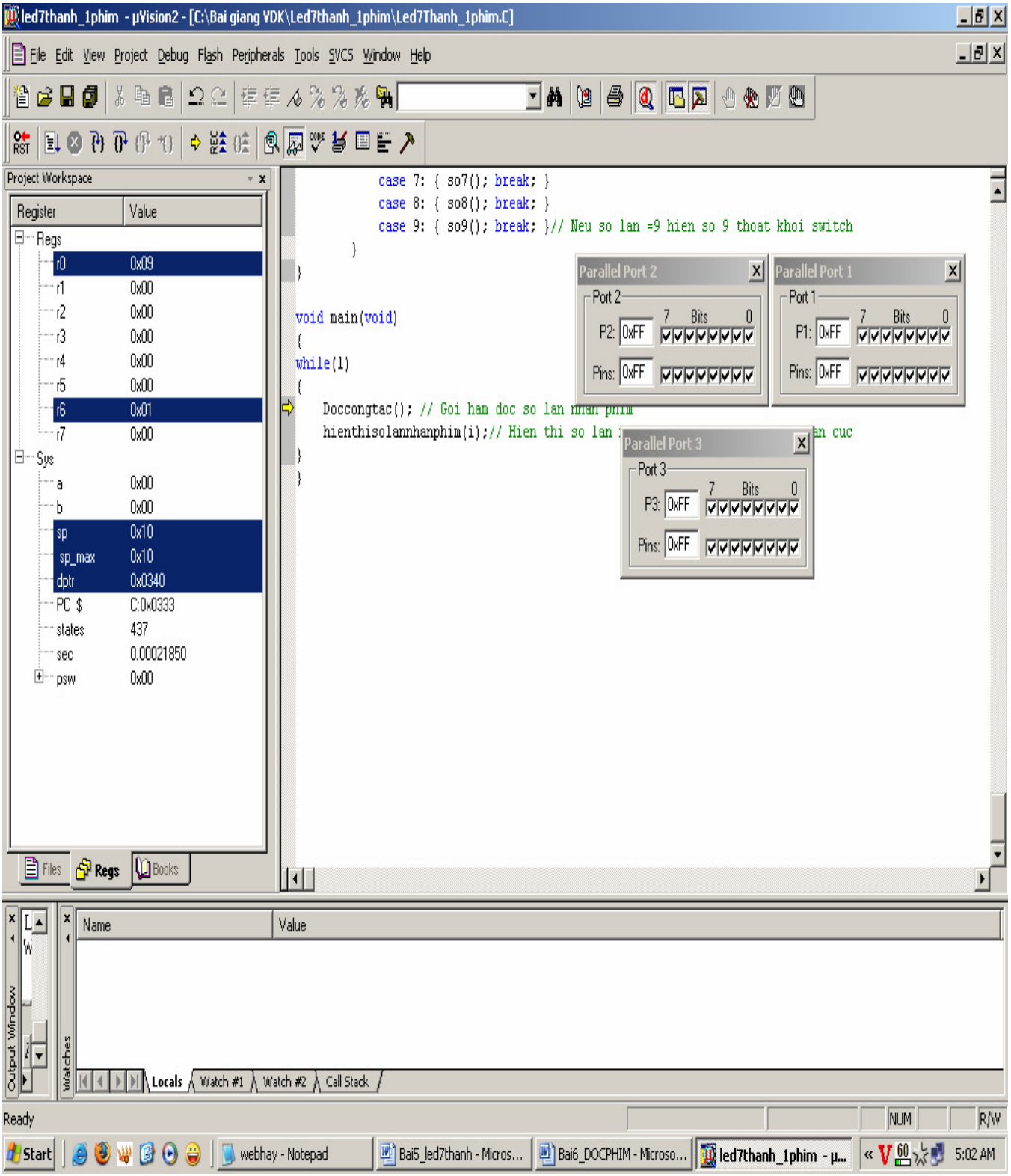
5.4) Kinh nghiệm:

- Xem các giá trị của biến trong Debug.

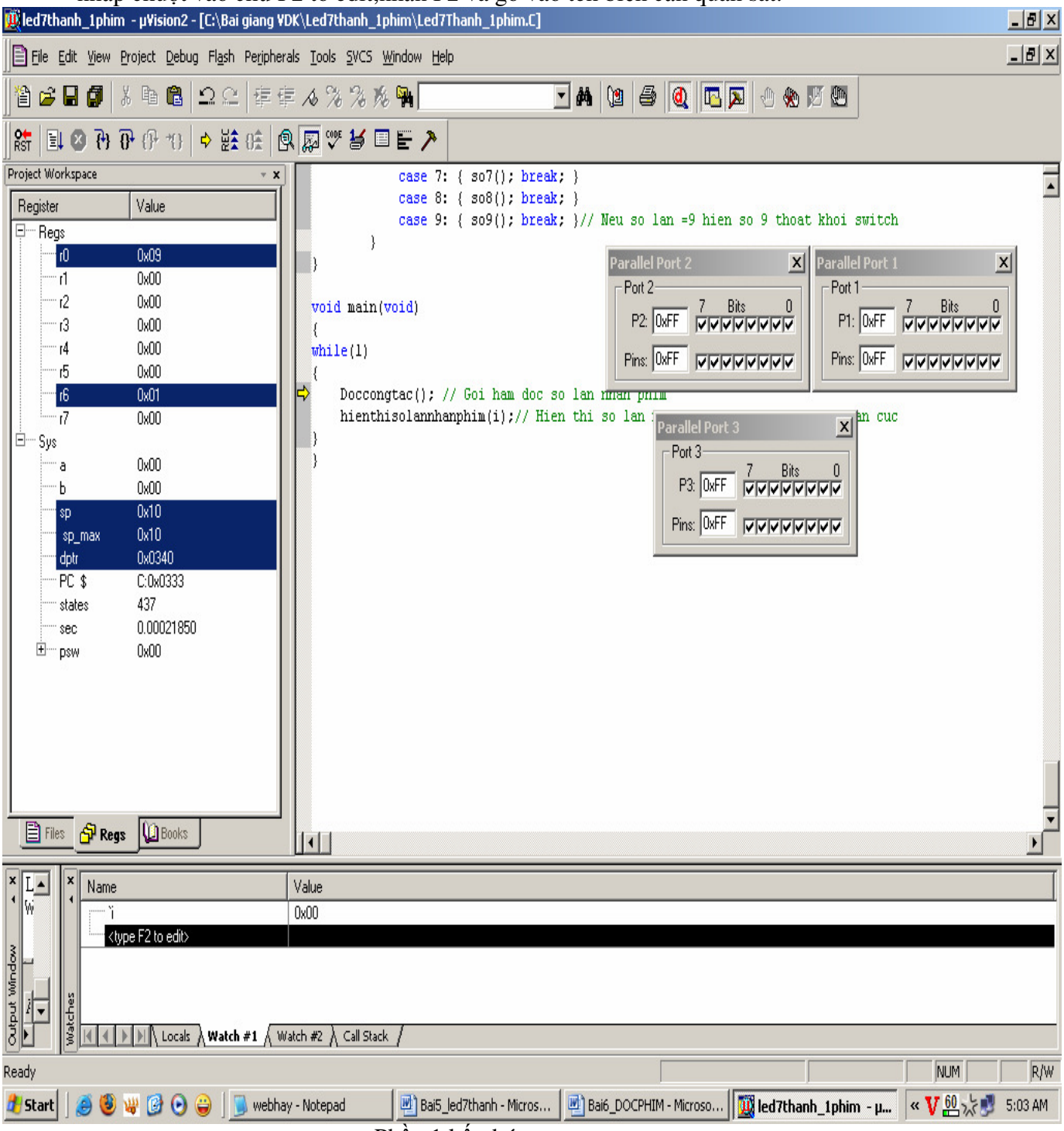
Sau khi viết xong chương trình và biên dịch chương trình các bạn vào công cụ Debug của Keil C. Được màn hình sau: Chọn View → Watch& Call Stack Window.



Được như sau:

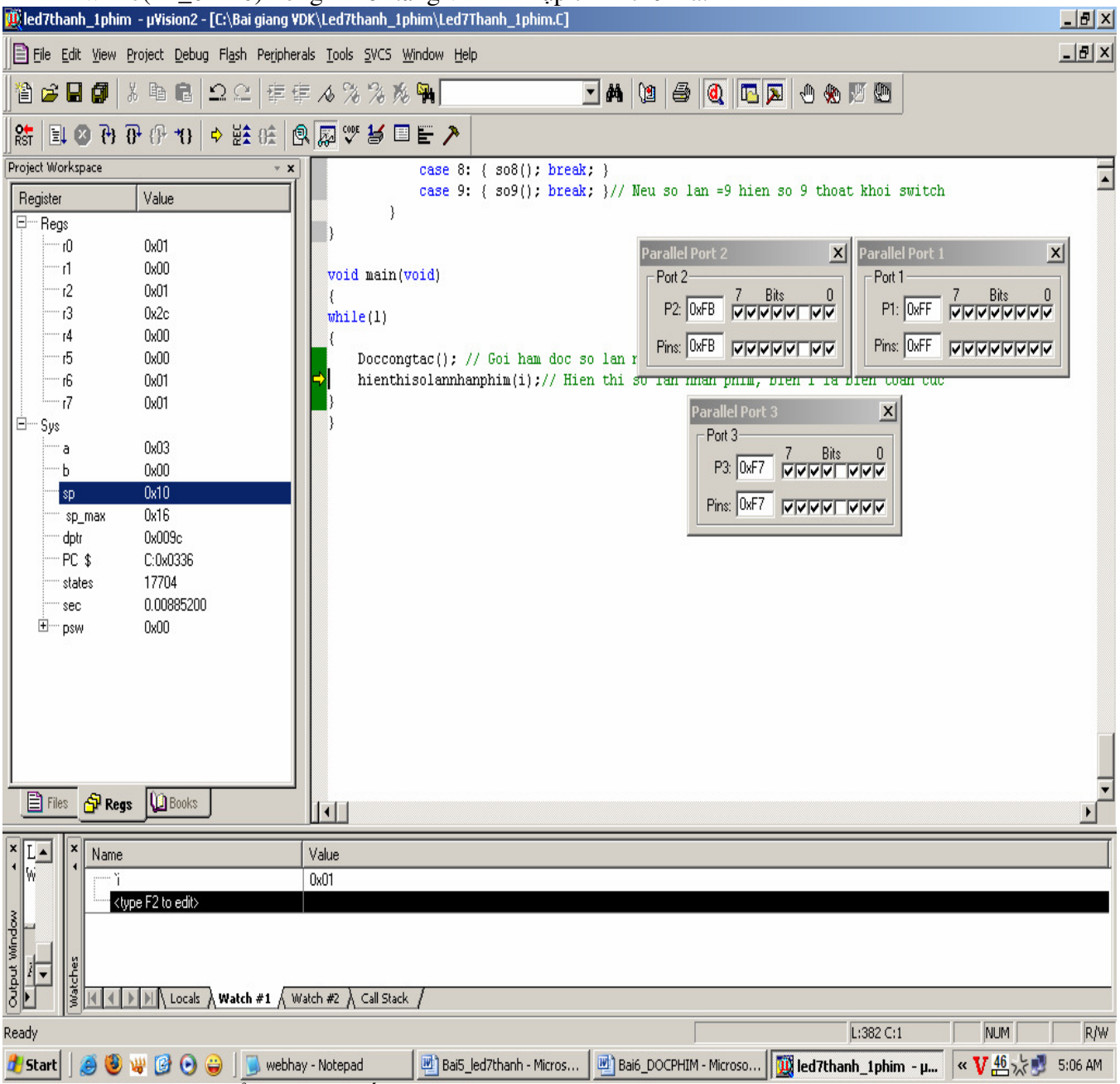

led7thanh_1phim - µVision2 - [C:\Bai giang VDK\Led7thanh_1phim\Led7Thanh_1phim.C]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Project Workspace
Register
Value
Regs
r0 0x09
r1 0x00
r2 0x00
r3 0x00
r4 0x00
r5 0x00
r6 0x01
r7 0x00
Sys
a 0x00
b 0x00
sp 0x10
sp_max 0x10
dptr 0x0340
PC $ C:0x0333
states 437
sec 0.00021850
psw 0x00
Files
Regs
Books
case 7: { so7(); break; }
case 8: { so8(); break; }
case 9: { so9(); break; }// Neu so lan =9 hien so 9 thoat khoi switch
}
}
void main(void)
{
while(1)
{
Doccongtac(); // Goi ham doc so lan nhan phim
hienthisolannhanphim(i);// Hien thi so lan
}
}
Parallel Port 2
Port 2
P2: 0xFF
Pins: 0xFF
7 Bits 0
Parallel Port 1
Port 1
P1: 0xFF
Pins: 0xFF
Parallel Port 3
Port 3
P3: 0xFF
Pins: 0xFF
Output Window
Watches
Name
Value
Locals
Watch #1
Watch #2
Call Stack
Ready
NUM
R/W
Start
webhay - Notepad
Bai5_led7thanh - Micros...
Bai6_DOCPHIM - Microso...
led7thanh_1phim - µ...
5:02 AM

Trong khung phía dưới chính là Watch& Call Stack Window. Chuyển sang tab watch#1, nhấp chuột vào chữ F2 to edit,nhấn F2 và gõ vào tên biến cần quan sát.



Phần 1 kết thúc.

Chúng ta quan sát biến I, nên gõ I vào và được hình như trên. Các bạn nhấn F11 để chạy mô phỏng. Mỗi lần ấn phím, cho chân P1.0 xuống 0 rồi lên 1, tương đương với ấn phím và nhả phím I sẽ tăng 1 như sau: (Khi ấn phím phải chờ hàm delay(300) và hàm while(P1_0 ==0) xong I mới tăng vì mình lập trình thế mà.



Giá trị của I hiển thị theo số hex.

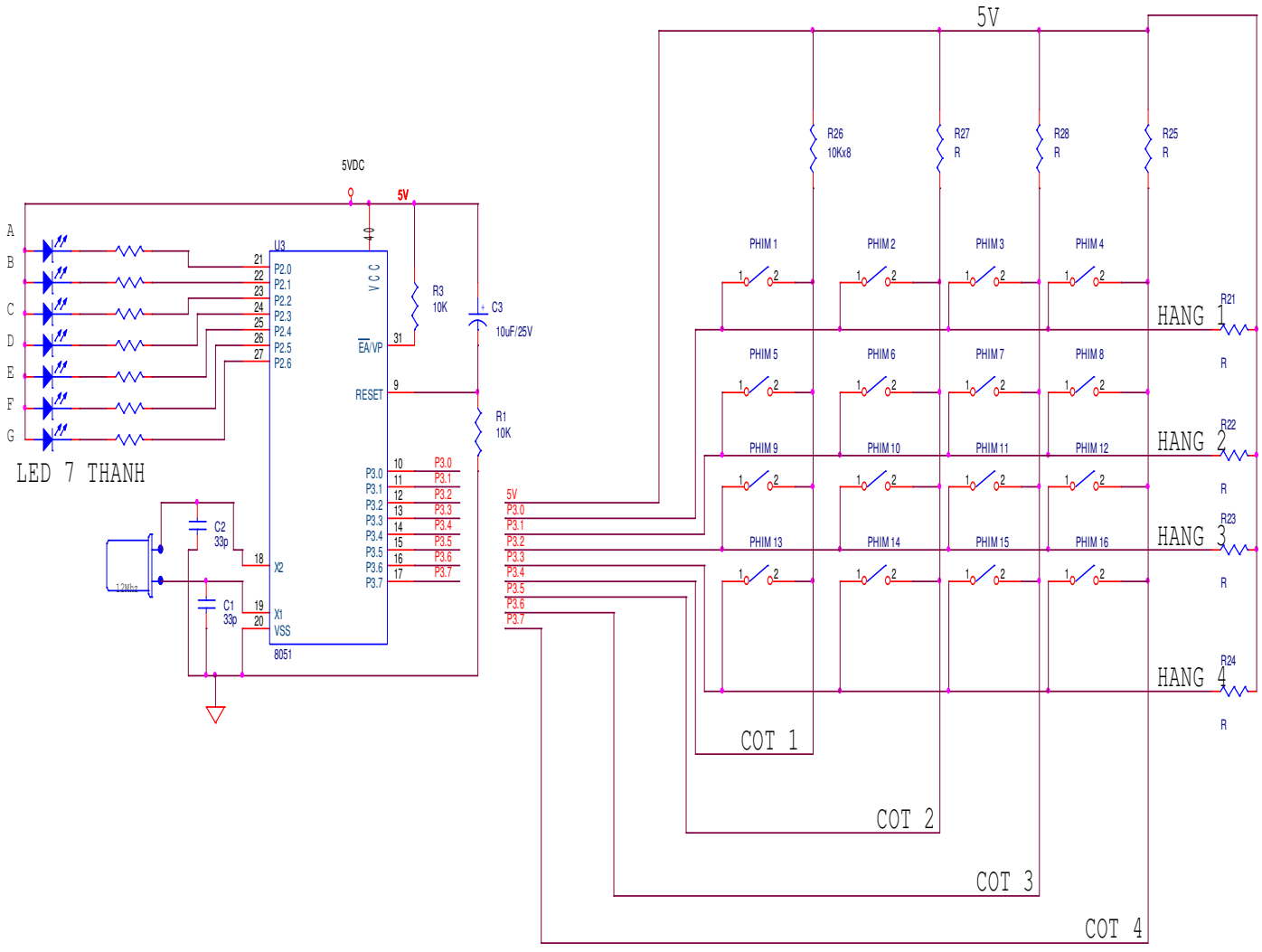
Bài 6:Đọc bàn phím

Phần 2: Đọc ma trận phím.

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 led 7 thanh.

6.1. Lắp mạch theo sơ đồ sau:



- Lắp mạch:mạch bài led 7 thanh giữ nguyên và với bàn phím các bạn hàn được chỉ cần câu vào cổng 3 đúng thứ tự chân và câu chân 5V lên +5V.

6.2 Nguyên lí quét phím:

- 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, tôi chọn cách quét theo hàng, quét theo cột các bạn có thể làm tương tự.

- Bước 1 : Ta đưa chân P3.0 nối với Hàng 1 xuống 0V.Rồi ta kiểm tra giá trị logic của các chân P3.4,P3.5,P3.6,P3.7.Nếu phím 1 được bấm thì Cột 1_ P3.4 sẽ có giá trị bằng 0. Nếu phím 2 được bấm thì Cột 2_ P3.5 sẽ có giá trị bằng 0. Nếu phím 3 được bấm thì Cột 3_ P3.6 sẽ có giá trị bằng 0. Nếu phím 4 được bấm thì Cột 4_ P3.7 sẽ có giá trị bằng 0. Ta căn cứ vào đó để xác định xem phím nào được bấm.

- Bước 2 : Ta đưa chân P3.1 nối với Hàng 2 xuống 0V.Rồi ta kiểm tra giá trị logic của các chân P3.4,P3.5,P3.6,P3.7.Nếu phím 5 được bấm thì Cột 1_ P3.4 sẽ có giá trị bằng 0. Nếu phím 6 được bấm thì Cột 2_ P3.5 sẽ có giá trị bằng 0. Nếu phím 7 được bấm thì Cột 3_ P3.6 sẽ có giá trị bằng 0. Nếu phím 8 được bấm thì Cột 4_ P3.7 sẽ có giá trị bằng 0. Ta căn cứ vào đó để xác định xem phím nào được bấm.

- Bước 3 : Ta đưa chân P3.2 nối với Hàng 3 xuống 0V.Rồi ta kiểm tra giá trị logic của các chân P3.4,P3.5,P3.6,P3.7.Nếu phím 9 được bấm thì Cột 1_ P3.4 sẽ có giá trị bằng 0. Nếu phím 10 được bấm thì Cột 2_ P3.5 sẽ có giá trị bằng 0. Nếu phím 11 được bấm thì Cột 3_ P3.6 sẽ có giá trị bằng 0. Nếu phím 12 được bấm thì Cột 4_ P3.7 sẽ có giá trị bằng 0. Ta căn cứ vào đó để xác định xem phím nào được bấm.

- Bước 4 : Ta đưa chân P3.3 nối với Hàng 1 xuống 0V.Rồi ta kiểm tra giá trị logic của các chân P3.4,P3.5,P3.6,P3.7.Nếu phím 13 được bấm thì Cột 1_ P3.4 sẽ có giá trị bằng 0. Nếu phím 14 được bấm thì Cột 2_ P3.5 sẽ có giá trị bằng 0. Nếu phím 15 được bấm thì Cột 3_ P3.6 sẽ có giá trị bằng 0. Nếu phím 16 được bấm thì Cột 4_ P3.7 sẽ có giá trị bằng 0. Ta căn cứ vào đó để xác định xem phím nào được bấm.

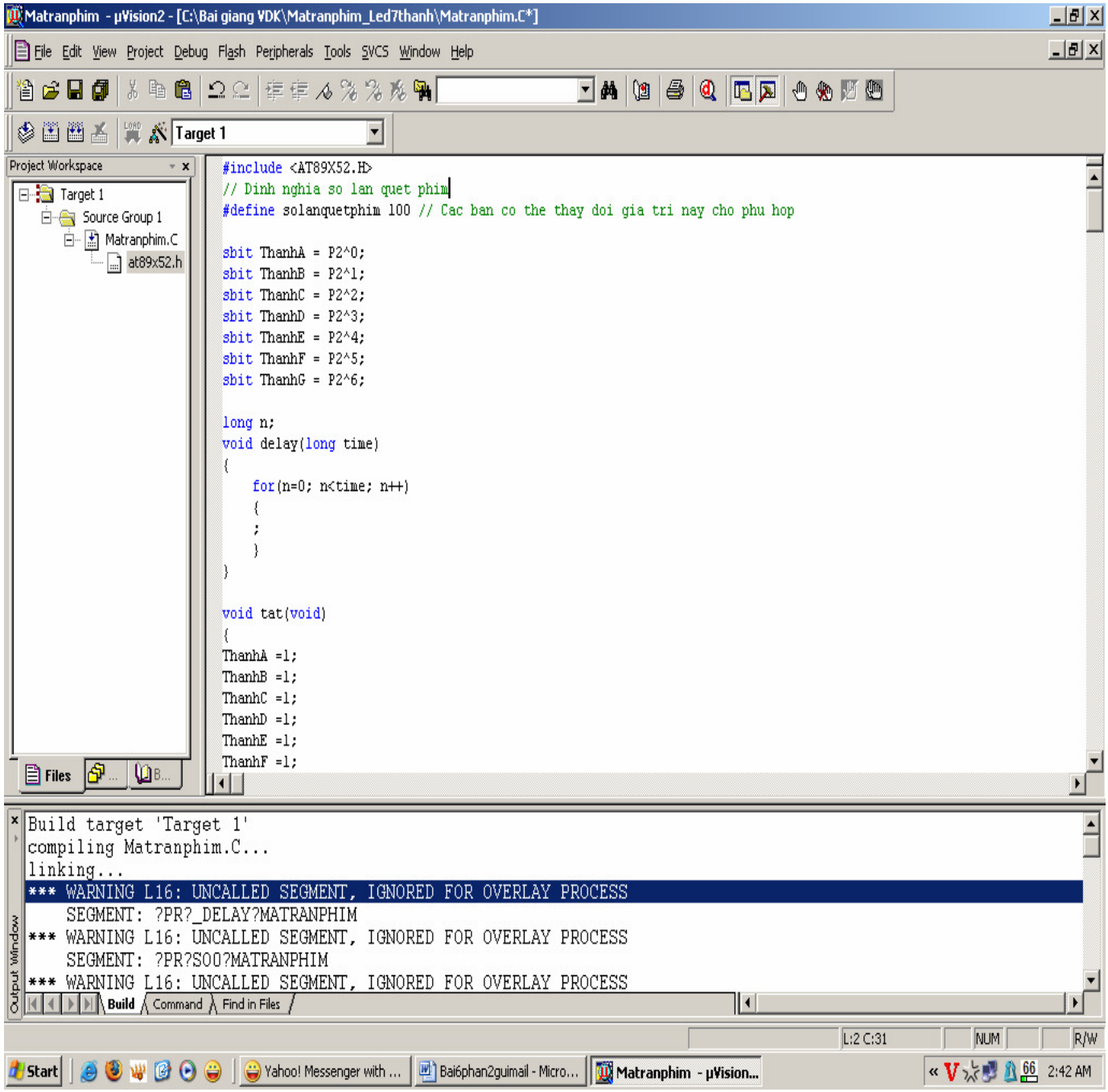
Ta sẽ dùng câu lệnh if để kiểm tra.

6.3. Lập trình:

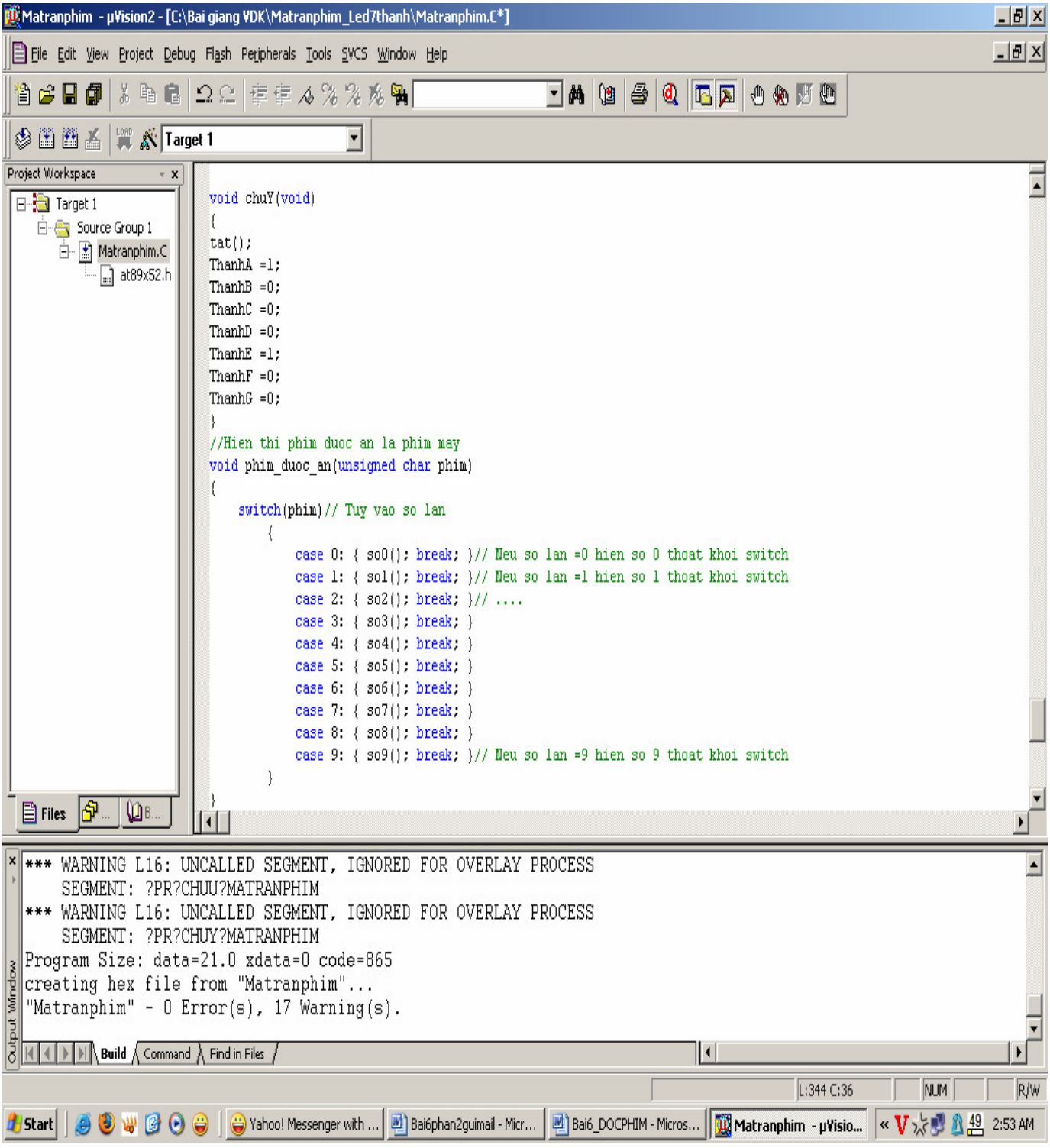
- Tạo 1 project mới, copy phần hiển thị các sô 0…9 các chữ A…Y của bài trước. Rồi bổ sung các hàm sau. Hàm hiện thị phím ấn.

```
void phim_duoc_an(unsigned char phim)
{
	switch(phim)// Tuy vao so lan
		{
			case 0: { so0(); break; }// Neu so lan =0 hien so 0 thoat khoi switch
			case 1: { so1(); break; }// Neu so lan =1 hien so 1 thoat khoi switch
			case 2: { so2(); break; }// ....
			case 3: { so3(); break; }
			case 4: { so4(); break; }
			case 5: { so5(); break; }
			case 6: { so6(); break; }
			case 7: { so7(); break; }
			case 8: { so8(); break; }
			case 9: { so9(); break; }// Neu so lan =9 hien so 9 thoat khoi switch
```

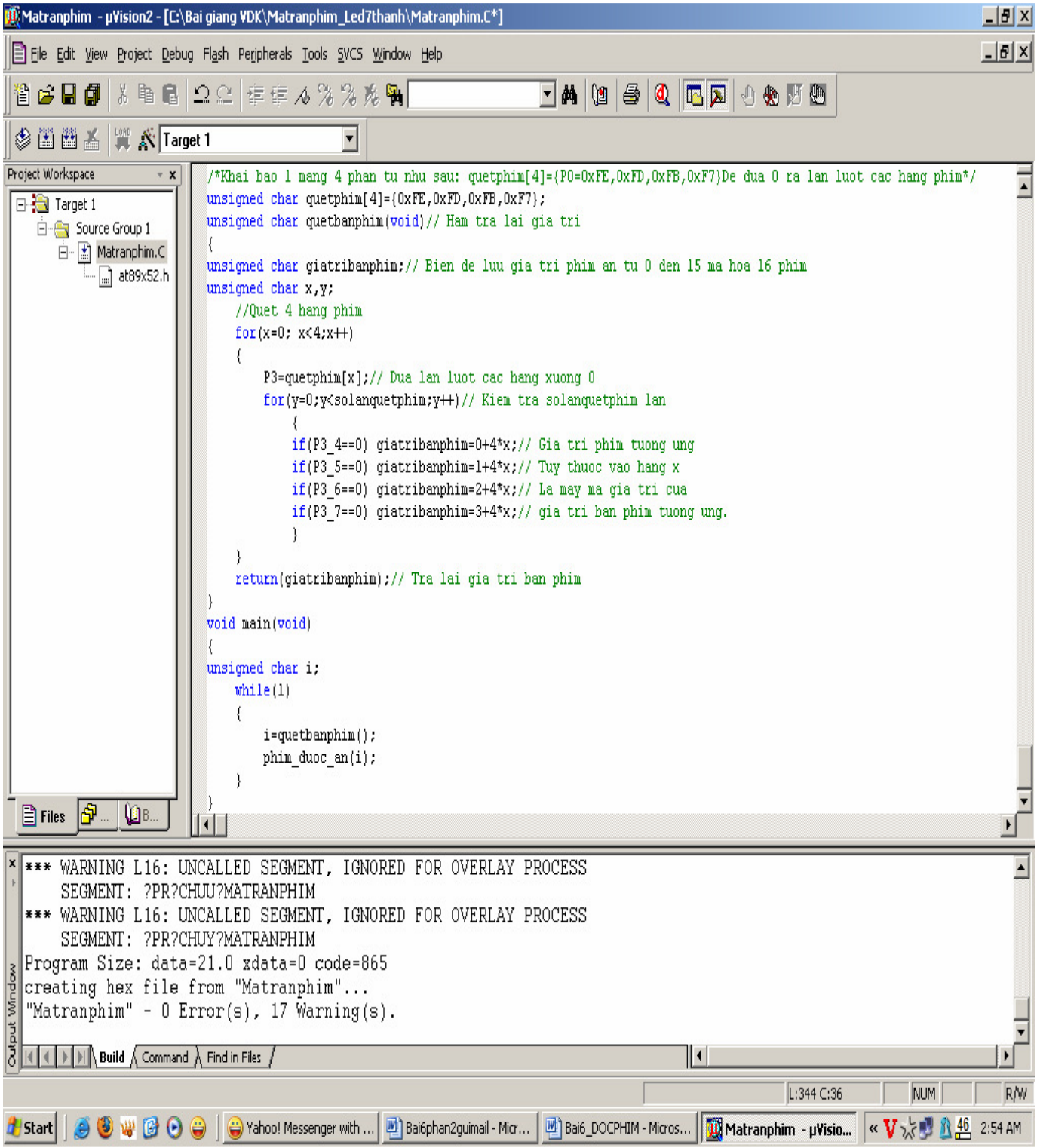
```
                }
}
```
Hàm quét phím:
```
/*Khai bao 1 mang 4 phan tu nhu sau: quetphim[4]={P0=0xFE,0xFD,0xFB,0xF7}
De dua 0 ra lan luot cac hang phim, khi do neu nut nao duoc an thi chan vi dieu khien se
xuong 0.Chu y fai kiem tra phim khoang 100 lan.*/
unsigned char quetphim[4]={0xFE,0xFD,0xFB,0xF7};
// Dinh nghia so lan quet phim
#define solanquetphim 100 // Cac ban co the thay doi gia tri nay cho phu hop
unsigned char quetbanphim(void)
{
unsigned char giatribanphim;// Bien de luu gia tri phim an tu 0 den 15 ma hoa 16 phim
unsigned char x,y;
        //Quet 4 hang phim
        for(x=0; x<4;x++)
        {
                P3=quetphim[x];// Dua lan luot cac hang xuong 0
                for(y=0;y<solanquetphim;y++)// Kiem tra solanquetphim lan
                        {
                        if(P3_4==0) giatribanphim=0+4*x;// Gia tri phim tuong ung
                        if(P3_5==0) giatribanphim=1+4*x;// Tuy thuoc vao hang x
                        if(P3_6==0) giatribanphim=2+4*x;// La may ma gia tri cua
                        if(P3_7==0) giatribanphim=3+4*x;// gia tri ban phim tuong ung.
                        }
        }
        return(giatribanphim);
}
```
Hàm Main.
```
void main(void)
{
unsigned char i;
        while(1)
        {
                i=quetbanphim();
                phim_duoc_an(i);
        }
}
```
Thêm câu lệnh #define vào đầu chương trình:



Viết hàm phím được ấn:



Viết hàm quét bàn phím và hàm main.


Matranphim - µVision2 - [C:\Bai giang VDK\Matranphim_Led7thanh\Matranphim.C*]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Target 1
Project Workspace
Target 1
Source Group 1
Matranphim.C
at89x52.h
Files
/*Khai bao 1 mang 4 phan tu nhu sau: quetphim[4]={P0=0xFE,0xFD,0xFB,0xF7}De dua 0 ra lan luot cac hang phim*/
unsigned char quetphim[4]={0xFE,0xFD,0xFB,0xF7};
unsigned char quetbanphim(void)// Ham tra lai gia tri
{
unsigned char giatribanphim;// Bien de luu gia tri phim an tu 0 den 15 ma hoa 16 phim
unsigned char x,y;
//Quet 4 hang phim
for(x=0; x<4;x++)
{
P3=quetphim[x];// Dua lan luot cac hang xuong 0
for(y=0;y<solanquetphim;y++)// Kiem tra solanquetphim lan
{
if(P3_4==0) giatribanphim=0+4*x;// Gia tri phim tuong ung
if(P3_5==0) giatribanphim=1+4*x;// Tuy thuoc vao hang x
if(P3_6==0) giatribanphim=2+4*x;// La may ma gia tri cua
if(P3_7==0) giatribanphim=3+4*x;// gia tri ban phim tuong ung.
}
}
return(giatribanphim);// Tra lai gia tri ban phim
}
void main(void)
{
unsigned char i;
while(1)
{
i=quetbanphim();
phim_duoc_an(i);
}
}
*** WARNING L16: UNCALLED SEGMENT, IGNORED FOR OVERLAY PROCESS
SEGMENT: ?PR?CHUU?MATRANPHIM
*** WARNING L16: UNCALLED SEGMENT, IGNORED FOR OVERLAY PROCESS
SEGMENT: ?PR?CHUY?MATRANPHIM
Program Size: data=21.0 xdata=0 code=865
creating hex file from "Matranphim"...
"Matranphim" - 0 Error(s), 17 Warning(s).
Output Window
Build
Command
Find in Files
L:344 C:36
NUM
R/W
Start
Yahoo! Messenger with ...
Bai6phan2guimail - Micr...
Bai6_DOCPHIM - Micros...
Matranphim - µVisio...
2:54 AM

5.4. Kinh nghiệm:

- Các bạn chạy Debug xem sự thay đổi giá trị của biến giá trị bàn phím.
- Các bạn mới dùng 10 phím từ 0…9 . Bài tập cho các bạn là dùng phím 16 để chọn mode: nhấn phím 16 thì thay đổi chế độ hiện số thành hiện chữ và phím 1 tương đương chữ A,phím 2 tương đương chữ B và cứ như vậy.
- Các bạn thử thay đổi giá trị #define solanquetphim 100 xem sao.
- Đừng lo lắng về các warning. Các warning nó cảnh báo là bạn có khai báo 1 số hàm mà bạn không dùng đến. Như hàm delay(); hàm chữ A, vv.

Bài 6:Điều khiển LCD 16x2

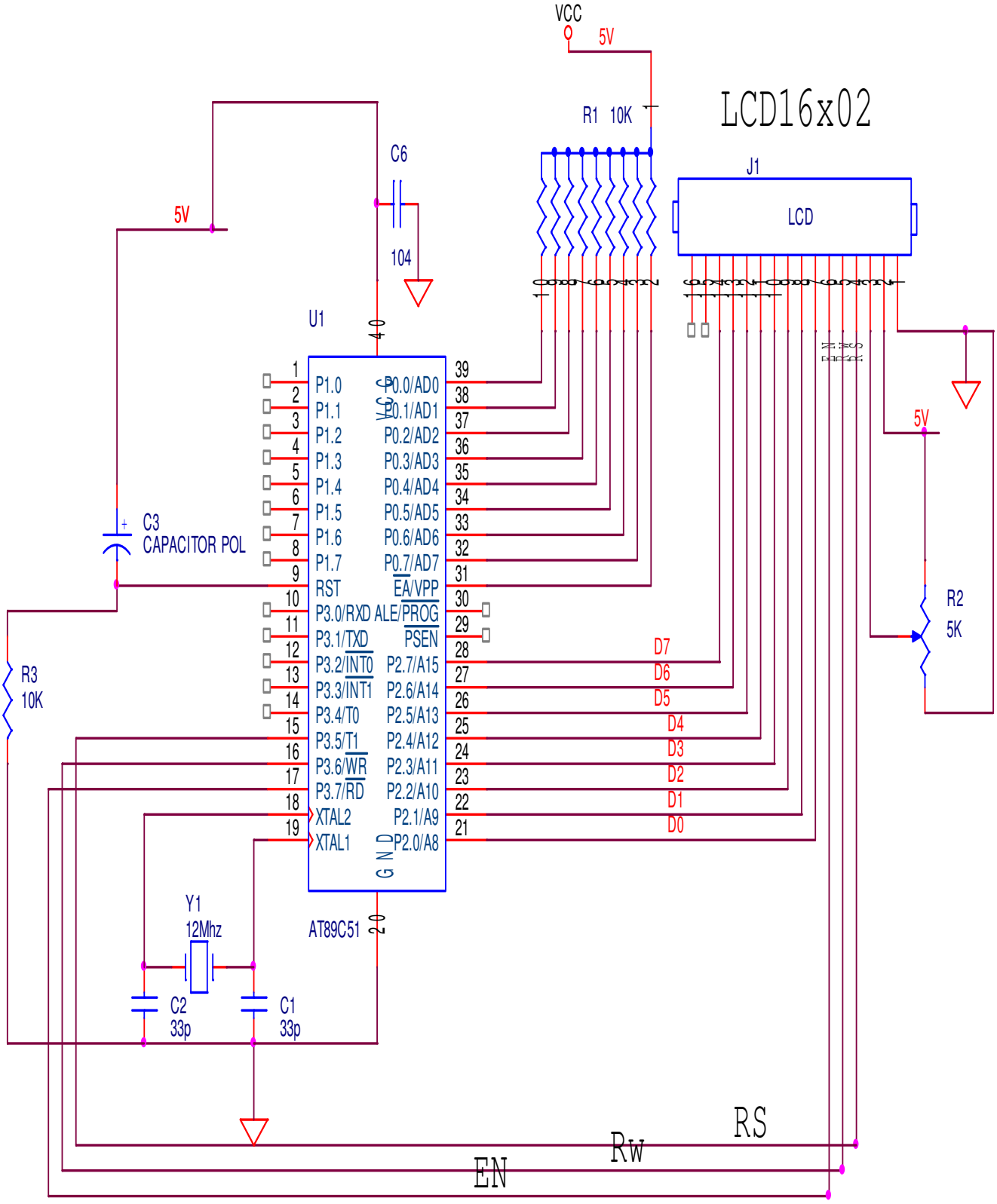
Nhiệm vụ: Điều khiển hiển thị LCD 16x2 dòng chữ "www.EmbestDKS.com" chạy trên màn hình LCD.

Có 16 chân như sau:

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.

6.1.Lắp mạch theo sơ đồ sau:


VCC
5V
LCD16x02
R1 10K
C6
104
J1
LCD
5V
U1
40
P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
RST
P3.0/RXD
P3.1/TXD
P3.2/INT0
P3.3/INT1
P3.4/T0
P3.5/T1
P3.6/WR
P3.7/RD
XTAL2
XTAL1
VCC
GND
P0.0/AD0
P0.1/AD1
P0.2/AD2
P0.3/AD3
P0.4/AD4
P0.5/AD5
P0.6/AD6
P0.7/AD7
EA/VPP
ALE/PROG
PSEN
P2.7/A15
P2.6/A14
P2.5/A13
P2.4/A12
P2.3/A11
P2.2/A10
P2.1/A9
P2.0/A8
AT89C51
20
C3
CAPACITOR POL
R3
10K
Y1
12Mhz
C2
33p
C1
33p
R2
5K
D7
D6
D5
D4
D3
D2
D1
D0
RS
Rw
EN

- Hướng dẫn: 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 lên +5V,1 chân xuống 0V.

6.2.Nguyên lý 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.

6.3.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 fep 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 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
}
```

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(0x30);
        LCDcontrol(0x30);
        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
}
```
Hàm có 1 biến đầu vào là mã của kí tự trong bảng ASCII.

* Hàm lệnh cho LCD hiển thị 1 xâu kí tự (dòng chữ):

```
void LCDputs(unsigned char *s,unsigned char row)
{
unsigned char len;
if(row==1) LCDcontrol(0x80);// Ep con tro ve dau dong 1
else LCDcontrol(0xC0);// Ep con tro ve dau dong 2
                len=strlen(s);// Lay do dai bien duoc tro boi con tro
                while(len!=0)// Khi do dai van con
                {
                   LCDwrite(*s);// Ghi ra LCD gia tri duoc tro boi con tro
                        s++;// Tang con tro
                        len--;// Tru do dai
                }
}
```

Hàm có hai biến đầu vào là : xâu kí tự cần hiển thị và dòng cần hiển thị xâu đó(1 hoặc 2).

*s là con trỏ, trỏ tới biến s

6.3.1. Định nghĩa con trỏ

Bộ nhớ của VĐK các bạn tưởng tượng như 1 cái tủ nhiều ngăn. Khi khai báo 1 biến, ví dụ biến kiểu unsigned char i; thì vđk lưu biến I vào 1 ngăn trong tủ_ 1 ô nhớ trong bộ nhớ, dĩ nhiên để xác định các ngăn tủ người ta đánh số cho từng ngăn, còn vđk cấp cho các ô nhớ trong bộ nhớ 1 địa chỉ để xác định ô nhớ đó.Ví dụ tiếp: I có giá trị là 100, thì nội dung của ô nhớ lưu biến I là 100,i=100, còn con trỏ trỏ đến I có giá trị là địa chỉ của ô nhớ chứa biến I đó.

6.3.2. Cách sử dụng con trỏ:

Để khai báo con trỏ có thêm dấu * trước tên biến. *I là biến kiểu con trỏ, trỏ tới biến i(unsigned char). I mang giá trị từ 0 đến 255, *I mang địa chỉ của ô nhớ chứa i.

* Hàm hiển thị 1 số integer:

```
void LCDwritei(int d)
{
unsigned char i,j,k,l;
i=d%10;// Chia lay phan du, duoc chu so hang don vi
d=d/10;// Chia lay phan nguyen, duoc nhung chu so da bo hang don vi
j=d%10;// Duoc chu so hang chuc
d=d/10;// Nhung chu so da bo hang don vi va hang chuc
k=d%10;// Duoc hang tram
l=d/10;// Duoc hang nghin
LCDwrite(48+l);// Hien thi ki tu trong bang ascii
LCDwrite(48+k);// Trong bang ascii so 0 co co so thu tu la 48
LCDwrite(48+j);
LCDwrite(48+i);
```

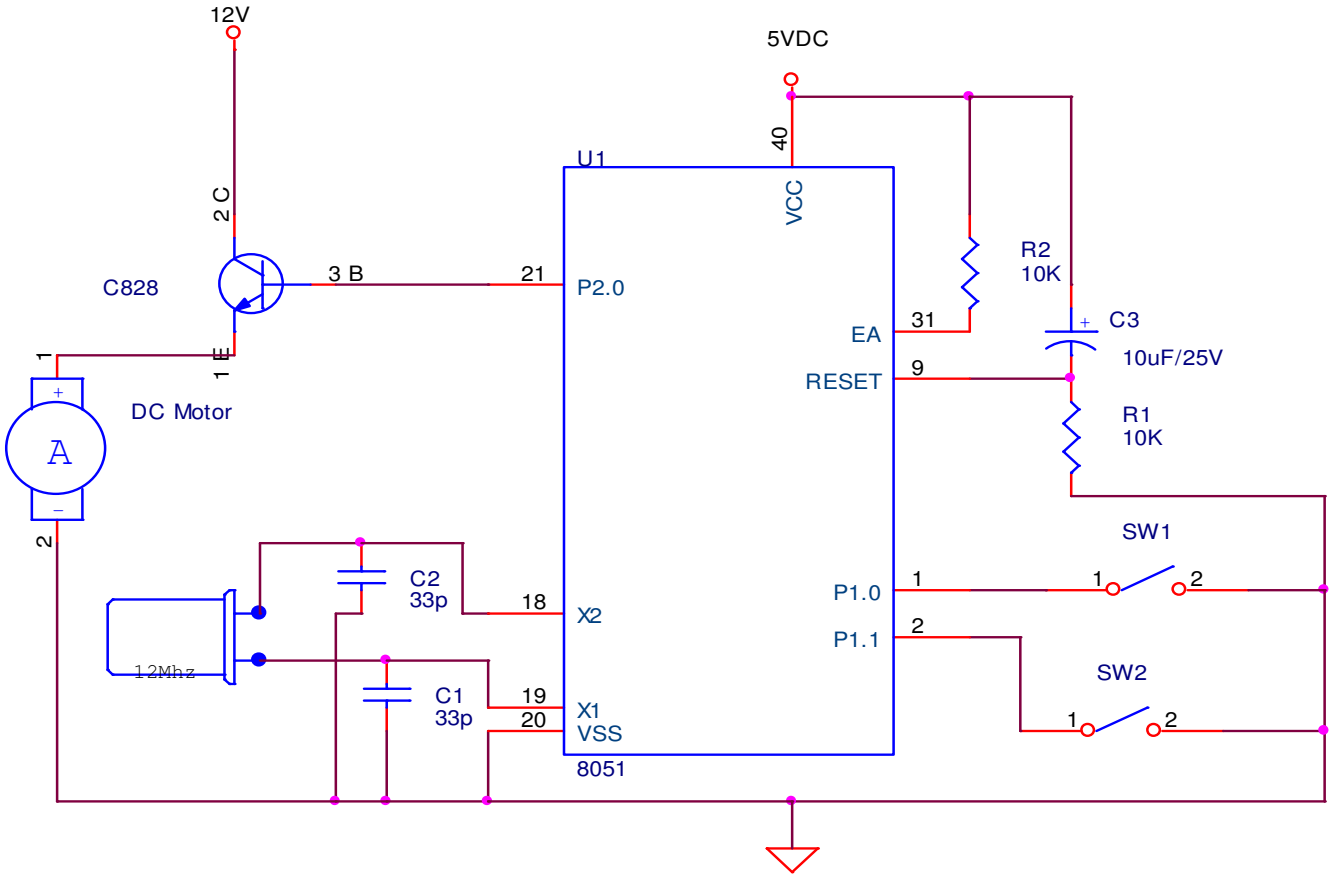
```
}
        Hàm có 1 biến đầu vào là số int lớn đến hàng nghìn cần hiển thị.
* Hàm trễ:
void delay(long time)
{
        long n;
        for(n=0;n<time;n++) ;
}
* Hàm main:
void main(void)
{
char x;
        LCDinit();
        LCDputs("8052 MCU",1);
        delay(30000);
        while(1)
        {
        for(x=0;x<16;x++)// Dich 16 lan.
        {
        LCDputs("8052 MCU",1);
        LCDcontrol(0x18);// Dich hien thi sang trai.
        delay(5000);// Tre
        }
        }
}
```

Bài 7.Điều chế độ rộng xung

Nhiệm vụ:

Tạo ra xung có độ rộng thay đổi, 10 cấp, tần số 1Khz, để điều khiển tốc độ động cơ (10 cấp tốc độ).

7.1. Lắp mạch theo sơ đồ sau:



- Hướng dẫn: Chân của C828 là ECB, nếu cầm xuôi transitor nhìn vào mặt có chữ, tính từ bên trái sang.

7.2.Lập trình:

- Cách tạo xung có độ rộng thay đổi bằng VĐK.

+ Cách 1: Như các bạn điều khiển nhấp nháy 1 con led, đó là tạo ra 1 xung ở 1 chân của vi điều khiển, nhưng xung đó có độ rộng cố định, tần số lớn, cách bạn có thể điều chỉnh lại hàm delay để tần số của nó đúng 1 Khz. Tuy nhiên vì là dùng hàm delay nên trong thời gian có xung lên 1(5V) và thời gian không có xung(0V) vi điều khiển không làm gì cả, hơn nữa tạo xung bằng việc delay mà các bạn có nhu cầu cần 2 bộ phát

xung ở 2 kênh, có cùng tần số mà khác độ rộng xung thì trở nên rất khó khăn. Cho nên chúng ta dùng bộ định thời Timer của vi điều khiển trong trường hợp này rất tiện.

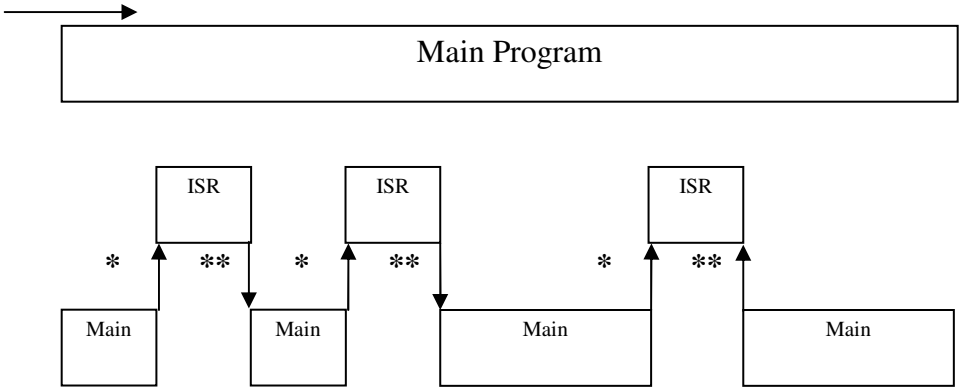
+ Cách 2: Dùng ngắt Timer của bộ vi điều khiển.

Trước hết nhắc lại về ngắt của vi điều khiển:

+ Ngắt là gì ? để trả lời câu hỏi này tôi xin trích đoạn về ngắt trong bài 2 ví dụ cho ngắt timer:

Timer

Program excution without interrut



Program excution with interrupt.

Một chương trình chính không có ngắt thì chạy liên tục, còn chương trình có ngắt thì cứ khi nào điều kiện ngắt được đảm bảo thì con trỏ sẽ nhảy sang hàm ngắt thực hiện xong hàm ngắt lại quay về đúng chỗ cũ thực hiện tiếp chương trình chính. Tôi có 1 ví dụ như sau: Bạn đang ăn cơm , có tiếng điện thoại , bạn đạt bát cơm ra nghe điện thoại , nghe xong lại quay về bưng bát cơm lên ăn tiếp. Thì quá trình ăn cơm của bạn là chương trình chính,có điện thoại gọi đến là điều kiện ngắt, bạn ra nghe điện thoại là thực hiện chương trình ngắt(Interrupt Service Rountine),quay về ăn cơm tiếp là tiếp tục thực hiện chương trình chính.

Ngắt đối với người mới học vi điều khiển là rất khó hiểu, vì đa số các tài liệu đều không giải thích ngắt để làm gì. Có nhiều loại ngắt khác nhau nhưng tất cả đều có chung 1 đặc điểm, ngắt dùng cho mục đích đa nhiệm. Đa tức là nhiều, nhiệm tức là nhiệm vụ. Thực hiện nhiều nhiệm vụ .

Các bạn nhìn vào tiền trình của hàm main với chương trình có ngắt :
Chương trình chính đang chạy, ngắt xảy ra, thực hiện hàm ngắt rồi quay lại chương trình chính. Chương trình trong vi điều khiển khác với ví dụ ăn cơm nghe điện thoại của tôi ở chỗ, thời gian thực hiện hàm chính là rất lớn,thời gian thực hiện hàm ngắt là rất nhỏ, cho nên thời gian thực hiện hàm ngắt không ảnh hưởng gì đến thời gian thực hiện hàm chính Như vậy trong hàm ngắt các bạn làm 1 việc, trong hàm chính cóc bạn làm 1 việc

như vậy coi như các bạn làm được 2 việc(đa nhiệm) trong 1 quãng thời gian tương đối ngắn cỡ mS, chứ thực ra tại 1 thời điểm vi điều khiển chỉ thực thi 1 lệnh.
Ví dụ : Bạn thử nghĩ xem làm thế nào để vừa điều chế xung PWM để điều chỉnh tốc độ động cơ , vừa đọc các cảm biến đầu vào mà tốc độ động cơ phụ thuộc đầu vào cảm biến.

Vậy ngắt là 1 điều kiện nào đó xảy ra ngẫu nhiên mà vi điều khiển có thể biết do phần cứng của vi điều khiển, rồi ta căn cứ vào đó để lập trỡnh.

* Ví dụ: Với ngắt bộ định thời timer, hay bộ đếm counter là khi tràn bộ đếm thì phần cứng của vi điều khiển sẽ bảo có ngắt xẩy ra và nhảy đến chương trình phục vụ ngắt(ISR_ Interrupt Sevice Rountine) 1 cách tự động.

Với ngắt ngoài, chân P3.2 chẳng hạn, nếu ta khai báo trước chân sử dụng chân P3.2 sử dụng cho ngắt ngoài chứ không phải sử dụng cho mục đích IO thiừ cứ khi cú 1 xung xuất hiện từ mạch ngoại vi truyền vào chõn P3.2 thì phần cứng của vi điều khiển nhận ra và chuyển tới chương trình phục vụ ngắt.

Với ngắt nối tiếp thì cứ khi có kí tự truyền từ máy tính xuống vi điều khiển thì sẽ có hiện tượng ngắt xảy ra.

- Hàm ngắt:

Cấu trúc:

```
Void  Tênhàm(void) interrupt nguồnngắt using băngthanhghi
{
// Chuong  trinh phuc vu ngat o day
}
```

Chỳ ý về hàm ngắt:

+ Hàm ngắt không được phép trả lại giá trị hay truyền biến vào hàm.
+ Tên hàm bất kì.
+ interrupt là từ khóa phân biệt hàm ngắt với hàm thường.
+ Nguồn ngắt từ 0 tới 5 theo bảng vector ngắt.
+ Băng thanh ghi trên ram chọn từ 0 đến 3.

Tựy theo bạn viết hàm ngắt cho nguồn nào bạn chọn nguồn ngắt từ bảng sau:

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**

Riêng ngắt Reset không tính, bắt đầu đếm từ 0 và từ ngắt ngoài 0. Ví dụ: tôi cần viết hàm ngắt cho bộ định thời timer 1 hàm ngắt sẽ là.

```
void timer1_isr(void) interrupt 3 using 0
{
// Lenh can thuc hien.
}
```

- Về using 0: Có 4 băng thanh ghi bạn có thể chọn cho chương trình phục vụ ngắt, cái này cũng không quan trọng. Trong hàm ngắt các bạn có thể bỏ đi từ using 0, khi đó vi điều khiển sẽ tự sắp xếp là dùng băng thanh ghi nào.
- Hàm ngắt khác hàm bình thường chỗ nào. Hàm bình thường ví dụ hàm delay, cứ khi bạn gọi nó thì nó sẽ được thực hiện, có nghĩa là nó có vị trí cố định trong tiến trình hàm main, có nghĩa là bạn biết nó xảy ra khi nào. Còn hàm ngắt thì không có tiến trình cố định, điều kiện ngắt có thể xảy ra bất kì lúc nào trong tiến trình hàm main và cứ khi nào có điều kiện ngắt thì hàm ngắt sẽ được gọi tự động.
- Để sử dụng ngắt ta phải làm các công việc sau:

1) Khởi tạo ngắt: dùng ngắt nào thì cho phép ngắt đó hoạt động bằng cách gán giá trị tương ứng cho thanh ghi cho phép ngắt IE(Interrupt Enable):

EA	ET2	ES	ET1	EX1	EX0	ET0

IE		Điều khiển các nguồn ngắt (0: không cho phép; 1: cho phép)
IE.7	EA	Cho phép/ không cho phép toàn cục
IE.6	---	Không sử dụng
IE.5	ET2	Cho phép ngắt do bộ định thời 2
IE.4	ES	Cho phép ngắt do port nối tiếp
IE.3	ET1	Cho phép ngắt cho bộ định thời 1
IE.2	EX1	Cho phép ngắt từ bên ngoài (ngắt ngoài 1)
IE.1	EX0	Cho phép ngắt từ bên ngoài (ngắt ngoài 0)
IE.0	ET0	Cho phép ngắt do bộ định thời 0

IE là thanh ghi có thể xử lí từng bít. Ví dụ : bạn muốn cho phép ngắt timer 1 bạn dùng lệnh: ET1=1; Không cho phép nữa bạn dùng lệnh : ET1=0; Hoặc bạn có thể dùng lệnh IE= 0x08; thì bit 3 của thanh ghi IE tức(IE) sẽ lên 1. Nhưng cách thứ nhất tiện hơn.

2) Cấu hình cho ngắt: Trong 1 ngắt nó lại có nhiều chế độ ví dụ: với ngắt timer. Bạn phải cấu hình cho nó chạy ở chế độ nào, chế độ timer hay counter, chế độ 16 bit, hay 8 bit,… bằng cách gán các giá trị tương ứng cho thanh ghi TMOD(Timer MODe).

TMOD		Chọn model cho bộ định thời 1			
7	GATE	Chọn model cho bộ định thời 1			
6	C/T	Bít chọn chức năng đếm hoặc định thời:			
5	M1	Bit chọn chế độ thứ nhất			
4	M0	Bit chọn chế độ thứ 2			
		M1	M0	Chế độ	Chức năng
		0	0	0	Chế độ định thời 13 bit
		0	1	1	Chế độ định thời 16 bit
		1	0	2	Chế độ tự động nạp lại 8 bit
		1	1	3	Chế độ định thời chia xẻ
3	GATE	Bit điều khiển cổng cho bộ định thời 0			
2	C/T	Bit chọn chức năng đếm / định thời cho bộ định thời 0			
1	M1	Bit chọn chế độ thứ nhất cho bộ định thời 0			
0	M0	Bit chọn chế độ thứ 2 cho bộ định thời 0			

Ví dụ tôi cấu hình cho bộ định thời 1 chế độ timer,với bộ đếm 8 bit tự động nạp lại(auto reload) dùng lệnh sau: TMOD=0x20.

Các bạn đừng lo vì việc phải nhớ bảng thanh ghi này, các bạn không phải nhớ nói trắng ra như vậy, chuyển sang phần lập trình các bạn sẽ được hướng dẫn làm thế nào để không phải nhớ, nhưng chỉ lập trình với C mới làm được còn lập trình Asem thì bắt buộc phải nhớ .

3)Bắt đầu chương trình có ngắt:

-Trước khi bắt đầu cho chạy chương trình ta phải cho phép ngắt toàn cục được xảy ra bằng cách gán EA(Enable All interrupt) bằng 1, thì ngắt mới xảy ra.

-Thường thì ngay vào đầu chương trình(hàm main) trước vòng while(1) chúng ta đặt công việc khởi tạo, cấu hình và cho phép kiểm tra ngắt. Ví dụ với bộ định thời timer ta gán các giá trị phù hợp cho thanh ghi TCON(Timer CONtrol).

TCON		Điều khiển bộ đinh thời
TCON.7	TF1	Cờ tràn của bộ định thời 1. Cờ này được set bởi phần cứng khi có tràn, được xoá bởi phần mềm, hoặc bởi phần cứng khi bộ vi xử lý trỏ đến trình phục vụ ngắt
TCON.6	TR1	Bit điều khiển hoạt động của bộ định thời 1. Bit này được set hoặc xoá bởi phần mềm để điều khiển bộ định thời hoạt động hay ngưng
TCON.5	TF0	Cừ tràn của bộ định thời 0
TCON.4	TR0	Bit điều khiển hoạt động của bộ định thời 0
TCON.3	IE1	Cừ ngắt bên ngoài 1 (kích khởi cạnh). Cờ này được set bởi phần cứng khi có cạnh âm (cuống) xuất hiện

		trên chân INT1, được xoá bởi phần mềm, hoặc phần cứng khi CPU trỏ đến trình phục vụ ngắt
TCON.2	IT1	Cờ ngắt bên ngoài 1 (kích khởi cạnh hoặc mức). Cờ này được set hoặc xoá bởi phần mềm khi xảy ra cạnh âm hoặc mức thấp tại chân ngắt ngoài
TCON.1	IE0	Cờ ngắt bên ngoài 0 (kích khởi cạnh)
TCON.0	IT0	Cờ ngắt bên ngoài 0 (kích khởi cạnh hoặc mức)

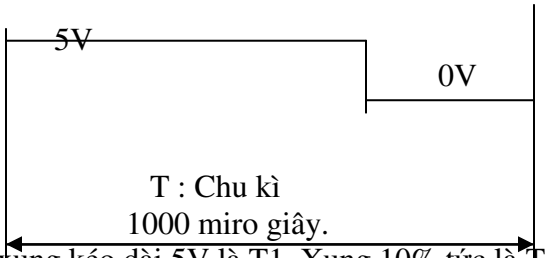
Ví dụ để chạy bộ định thời timer 1 ta dùng câu lệnh: TR1=0;
TR1(Timer Run 1). Còn bạn nào thích khó thì:TCON=0xxx;
Còn các loại ngắt khác quá trình tương tự, đây là khóa học cơ bản chỉ làm việc với ngắt timer, trong khóa nâng cao sẽ có các ngắt còn lại, tuy nhiên làm việc được với ngắt timer thì các ngắt khác các bạn cũng có thể làm tương tự, các bạn làm đến ngắt nào thì dùng tài liệu tra bảng thanh ghi của ngắt đó. Tài liệu tôi sẽ gửi cùng bài này.
- Quay trở lại bài học:

Sau khi khởi tạo song và cho ngắt timer 1 chạy thì điều gì xảy ra?

Khi bắt đầu cho timer 1 chạy thì bộ đếm của timer sẽ đếm dao động của thạch anh, cứ 12 dao động của thạch anh(1 chu kì máy), bộ đếm của timer 1 TL1(Timer Low1) sẽ tăng 1,có thể nói timer 1 đếm số chu kì máy. Đối với chế độ 8 bít.
TL1 là 1 thanh ghi 8 bít, là bộ đếm của bộ định thời rõ rồi. Nó đếm được từ 0, đến 255. Nếu nó đếm đến 256 thì bộ đếm tràn, TL1 quay vòng lại bằng 0, và cờ ngắt TF1(Timer Flag 1) tự động được gán lên 1(bằng phần cứng của vi điều khiển) như 1 công tắc tự động bật, và ngắt xảy ra.

Còn với chế độ 16 bít, bộ đếm của bộ định thời còn 1 thanh ghi 8 bít nữa là TH1(Timer high 1), nếu cấu hình cho timer 1 hoạt động ở chế độ 16 bit thì khi TL1 tràn nó sẽ đếm sang TH1(TH1 sẽ tăng 1). Như vậy ta có thể đếm: 2^{16} chu kì máy(2 thanh ghi 8+8=16 bít).
Chú ý là khi bộ đếm tràn ngắt sẽ xảy ra. Nếu ta cần đếm 256 chu kì máy thì khi khởi tạo ta cho TL1=0; , còn nếu không muốn đếm 256 chu kì mày mà ta chỉ cần đếm 100 thôi ngắt đã xảy ra rồi thì ta fải làm như sau: 256-100 = 156; và khi khởi tạo ta gán : TL1=155; vì đếm từ 155 đến 255 là đủ 100 lần thì ngắt xảy ra.

Với yêu cầu của bài. 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)
Bây giờ tôi mới xin nói về phần 2.

7.3) Nguyên lí hoạt động:

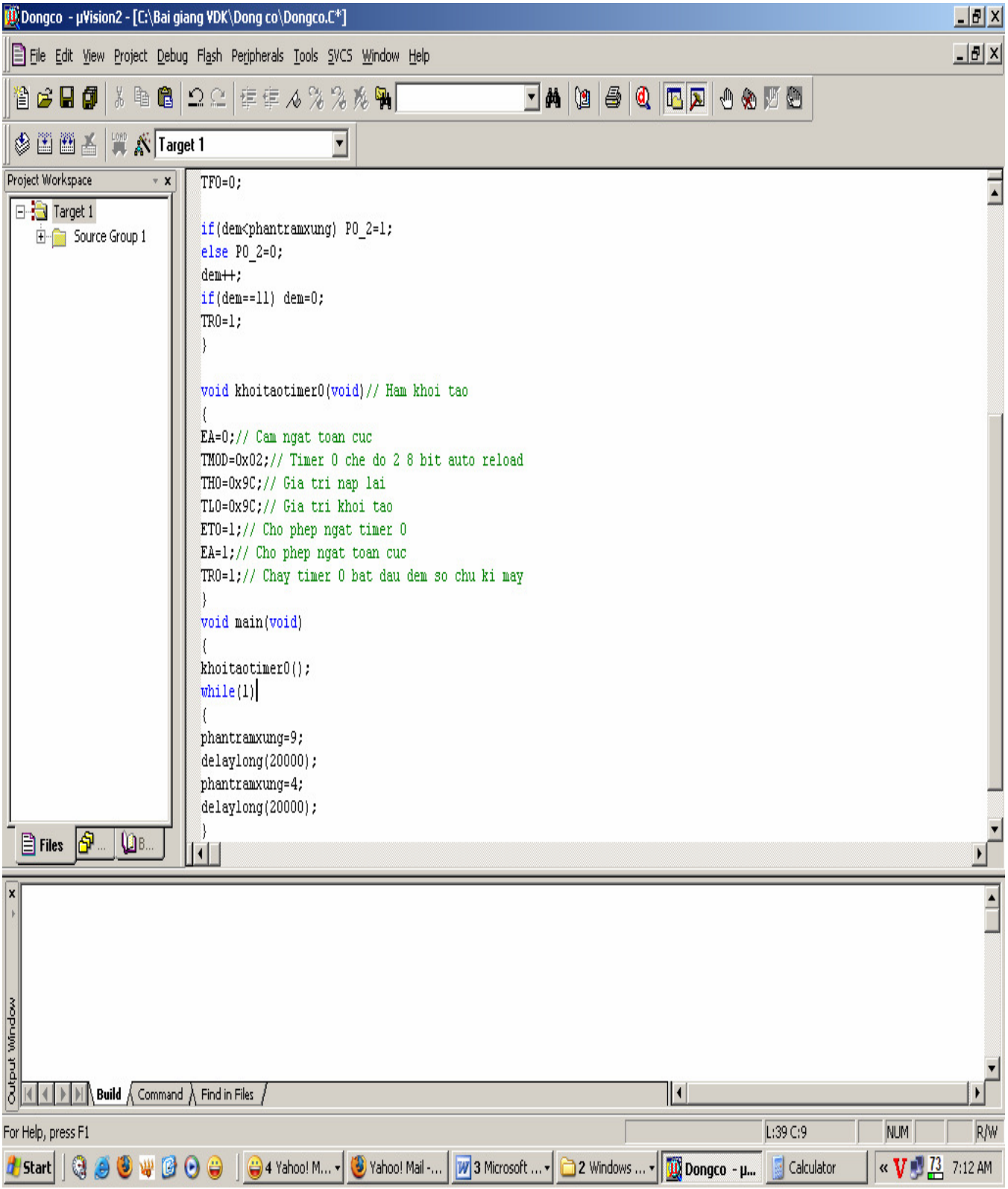
- Xung PWM: Đưa ra mở transitor, xung với độ rộng lớn hơn transitor sẽ mở lâu hơn động cơ sẽ quay nhanh hơn,dĩ nhiên không tuyến tính. Không có xung động cơ sẽ không quay, có xung 100% động cơ sẽ quay max.Tuy nhiên xung phải lớn hơn 1 mức nào đó thì mới đủ khởi động cho động cơ. Các đặc tính này các bạn tham khảo trong giáo trình về máy điện, khí cụ điện, nếu các bạn cần thông số chính xác.

Để có thể thay đổi 10 cấp tốc độ với chu kì 1000uS, ta khởi tạo cho ngắt timer: 100 uS ngắt 1 lần. Trong hàm ngắt kiểm tra xem ta cần cấp xung bao nhiêu % thì ta sẽ gán giá trị cho nó. Cụ thể như sau:

* Hàm khởi tạo ngắt.

Dùng ngắt timer 0, 100 uS ngắt 1 lần, dùng chế độ 2 8 bit tự động nạp lại của timer (vì mình chỉ cần đếm đến 100).TL0 nạp bằng 156. Đối với chế độ 2 khi tràn bộ đếm TL0 sẽ quay vòng giá trị bằng 0, nhưng sau đó nó lại được nạp giá trị lưu trong TH0(giá trị nạp lại), do đó ta chỉ cần gán giá trị choTL0 và TH0 trong hàm khởi tạo, còn ở các chế độ khác 16 bit, 2 timer counter 8 bit, khi tràn bộ đếm TL0 không được nạp lại mà ta phải tự gán lại giá trị cho nó trong hàm ngắt.

```
void khoitaotimer0(void)// Ham khoi tao
{
EA=0;// Cam ngat toan cuc
TMOD=0x02;// Timer 0 che do 2 8 bit auto reload
TH0=0x9B;// Gia tri nap lai 155 doi ra so hex
TL0=0x9B;// Gia tri khoi tao 155 doi ra so hex
ET0=1;// Cho phep ngat timer 0
EA=1;// Cho phep ngat toan cuc
TR0=1;// Chay timer 0 bat dau dem so chu ki may
}
```


Dongco - µVision2 - [C:\Bai giang VDK\Dong co\Dongco.C*]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Target 1
Project Workspace
Target 1
Source Group 1
Files
TF0=0;
if(dem<phantramxung) P0_2=1;
else P0_2=0;
dem++;
if(dem==11) dem=0;
TR0=1;
}
void khoitaotimer0(void)// Ham khoi tao
{
EA=0;// Cam ngat toan cuc
TMOD=0x02;// Timer 0 che do 2 8 bit auto reload
TH0=0x9C;// Gia tri nap lai
TL0=0x9C;// Gia tri khoi tao
ET0=1;// Cho phep ngat timer 0
EA=1;// Cho phep ngat toan cuc
TR0=1;// Chay timer 0 bat dau dem so chu ki may
}
void main(void)
{
khoitaotimer0();
while(1)
{
phantramxung=9;
delaylong(20000);
phantramxung=4;
delaylong(20000);
}
Output Window
Build
Command
Find in Files
For Help, press F1
L:39 C:9
NUM
R/W
Start
4 Yahoo! M...
Yahoo! Mail -...
3 Microsoft ...
2 Windows ...
Dongco - µ...
Calculator
7:12 AM

* Hàm ngắt:

```
unsigned char dem=0;// Khai bao bien dem de dem tu 1 den 10
unsigned char phantramxung;// Bien chua phan tram xung(0...10)

void timer0(void) interrupt 1 //Ngat timer 0
{
TR0=0;// Dung chay timer 0
TF0=0;// Xoa co, o che do co tu duoc xoa,che do khac can toi cu viet vao day
dem++;
if(dem<phantramxung) P2_0=1;// Neu bien dem < phan tram xung thi dua gia tri 1 ra
chan, xung 5V
else P2_0=0;// Neu dem = phan tram xung
if(dem==10) dem=0;// Neu dem du 10 thi gan lai bang 0 de bat dau chu ki moi
TR0=1;// Cho chay timer
}
```

Để có thể thay đổi độ rộng xung thì ta lưu độ rộng xung vào 1 biến, vì hàm ngắt không cho truyền biến vào ta khai báo biến đó là biến toàn cục để có thể gán giá trị ở mọi hàm. 100 uS ngắt 1 lần để xác định đủ chu kì 1000 uS ta cần đếm từ 1 đến 10 ta khai báo biến đếm.

```
void timer0(void) interrupt 1 //Ngat timer 0
{
TR0=0;// Dung chay timer 0
TF0=0;// Xoa co, o che do co tu duoc xoa,che do khac can toi cu viet vao day
TH0=0xAB;
TL0=0xAB;
….
TR0=1;// Cho chay timer
}
```

Cấu trúc hàm ngắt timer nào cũng phải theo, do chế độ 2 tự động nạp lại nên kô cần gán giá trị cho TH0 và TL0.

Về biến dem sẽ đếm từ 1 đến 10 nếu bằng 10 kết thúc 1 chu kì 10*100 =1000 uS, ta gán lại nó bằng 0 để sang chu kì mới.

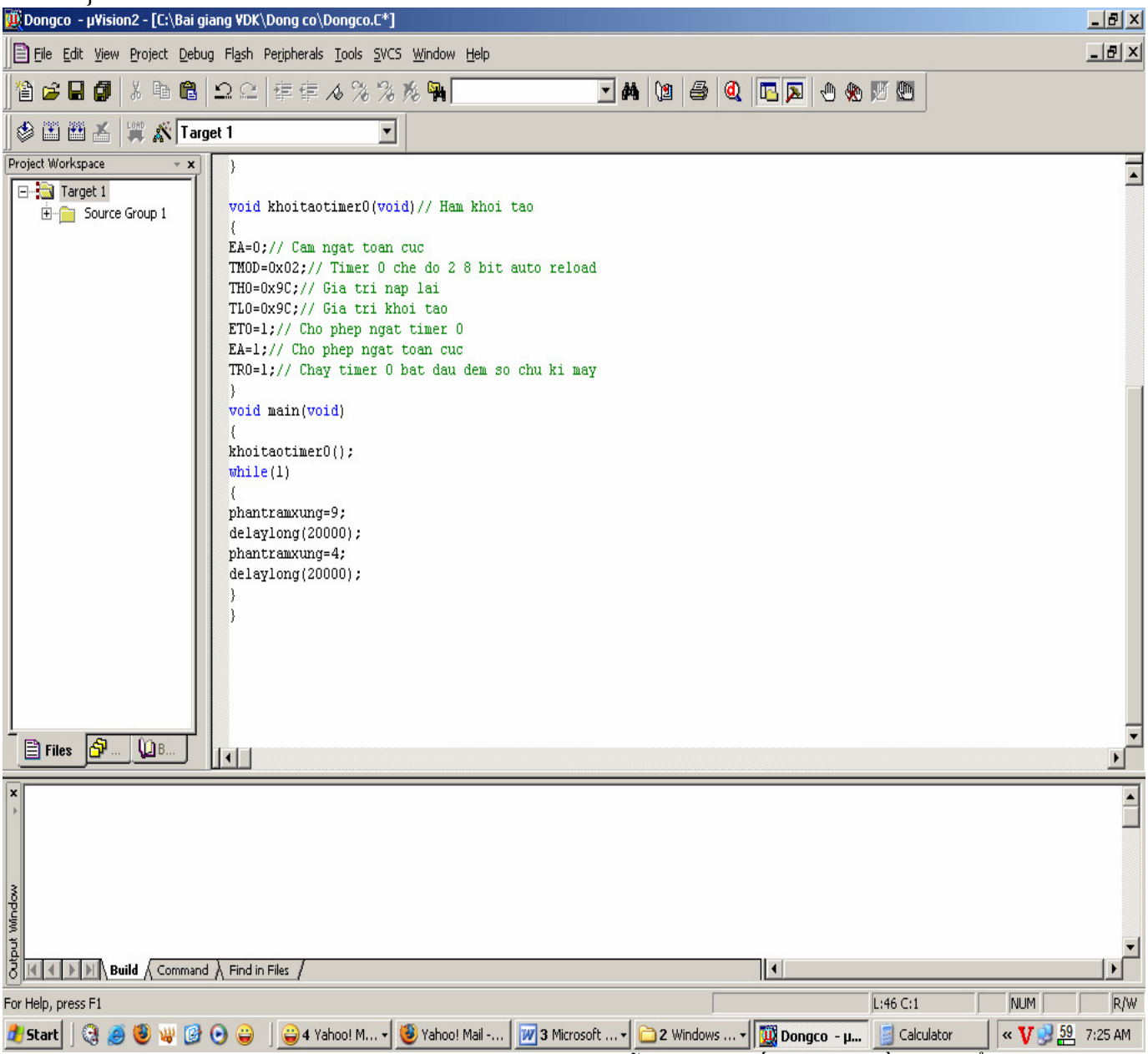
```
if(dem<phantramxung) P2_0=1;// Neu bien dem < phan tram xung thi dua gia tri 1 ra
chan, xung 5V
else P2_0=0;// Neu dem = phan tram xung
```

Câu lệnh này kiểm tra nếu đếm nhỏ hơn phantramxung thì sẽ đưa ra cổng giá trị 1, bằng hoặc lớn hơn sẽ đưa ra giá trị 0. Khi vào chương trình chính ta chỉ việc thay đổi giá trị biến phantramxung thì độ rộng xung sẽ thay đổi.

* Hàm main:

```
void main(void)
{
khoitaotimer0();
while(1)
{
```

```
phantramxung=9;
delaylong(20000);
phantramxung=4;
delaylong(20000);
}
}
```

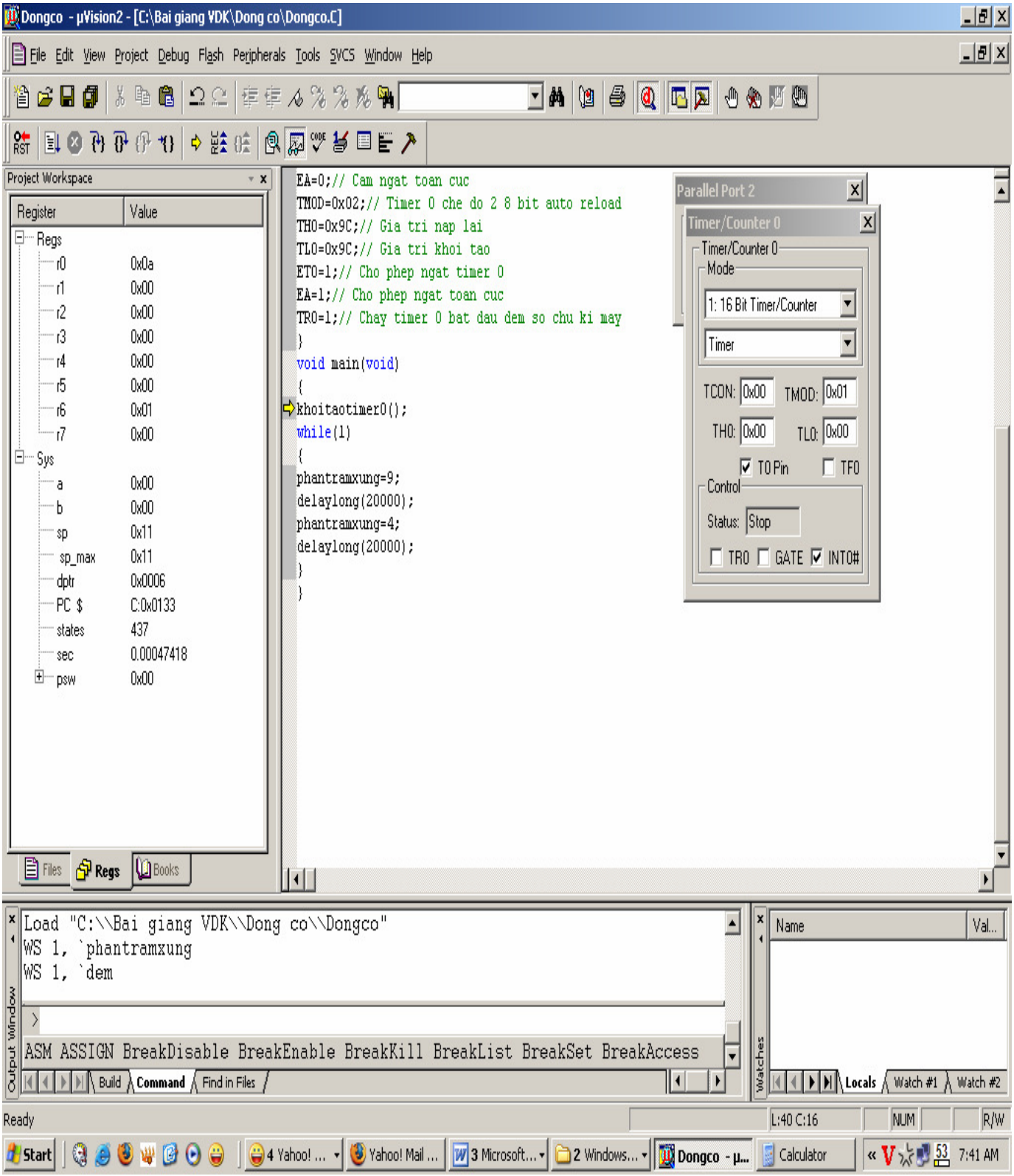


Giả sử khi các bạn gán phantramxung=4; Thì cứ mỗi 100uS ngắt xảy ra 1 lần, và kiểm tra biến đếm. Lần đầu dem=1 <4 nên giá trị P2_0 = 1 mức cao, lần thứ 2 , 200 uS, dem =2<4 P2_0 = 1 mức cao, lần thứ 3, 300uS, dem=3<4, P2_0=1 mức cao, lần thứ 4, 400uS, dem =4 <4 sai, P2_0=0, bắt đầu xuống mức thấp, có xung từ cao xuống thấp, dem = 5<4 sai ,

P2_0=0 mức thấp, …, dem =10 <4 sai P2_0 mức thấp đủ 1000 uS , 400uS cao, 600uS thấp quay vòng dem=0, ngắt lần thứ 11, dem=1 < 4 , P2_0=1 mức cao, có xung thấp lên cao….
Để PWM 2 chân P2_0 và P3_5, các bạn khai báo thêm 1 biến phantramxung2 và đưa thêm dòng lệnh sau vào hàm ngắt.

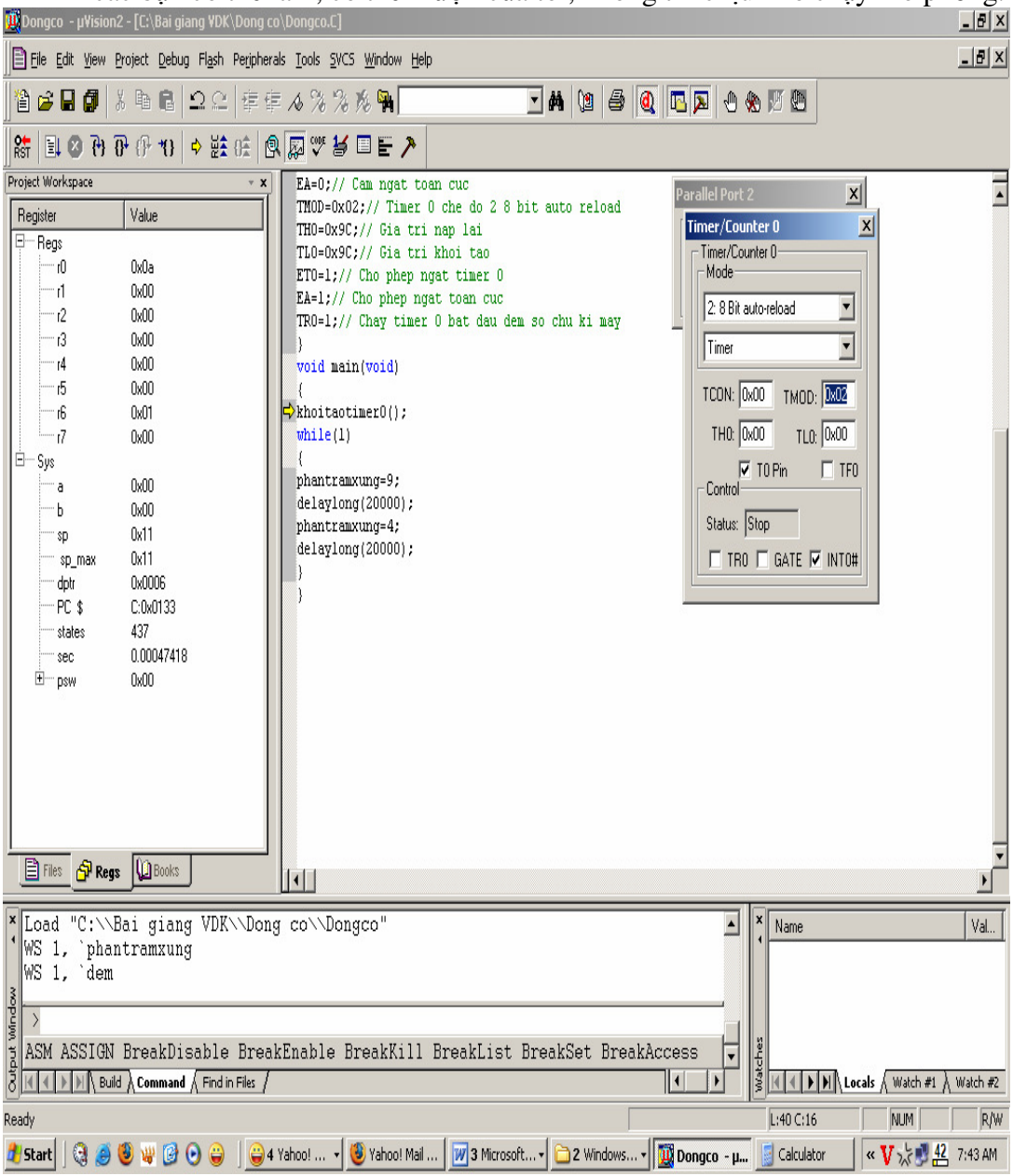
if(dem<phantramxung) P3_5=1;// Neu bien dem < phan tram xung thi dua gia tri 1 ra chan, xung 5V
else P3_5=0;// Neu dem = phan tram xung

Chú ý: Thực ra 1 chu kí như ta vừa làm không chính xác 100% là 1Khz, vì ta chưa tính đến độ dài của hàm ngắt,mỗi lần ngắt 100uS, 10 lần là 1000uS đã đủ, còn thời gian thực hiện hàm ngắt nữa, như vậy là chu kì của ta lớn hơn 1000uS, tần số sẽ <1Khz, nhưng thực sự sai số đó không đáng kể. Nếu các bạn muốn chính xác tôi cũng chiều lòng các bạn. Các bạn chạy debug, để thạch anh đúng 12Mhz, quan sat dòng sec xem hàm ngắt diễn ra trong bao nhiêu chu kì máy, khi nạp giá trị cho TL0 và TH0 các bạn lấy 155 trừ đi giá trị đó được gía trị a gán vào, như vậy a+thời gian thực hiện hàm ngắt đúng đủ 100uS.


Dongco - µVision2 - [C:\Bai giang VDK\Dong co\Dongco.C]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Project Workspace
Register
Value
Regs
r0 0x0a
r1 0x00
r2 0x00
r3 0x00
r4 0x00
r5 0x00
r6 0x01
r7 0x00
Sys
a 0x00
b 0x00
sp 0x11
sp_max 0x11
dptr 0x0006
PC $ C:0x0133
states 437
sec 0.00047418
psw 0x00
Files
Regs
Books
EA=0;// Cam ngat toan cuc
TMOD=0x02;// Timer 0 che do 2 8 bit auto reload
TH0=0x9C;// Gia tri nap lai
TL0=0x9C;// Gia tri khoi tao
ET0=1;// Cho phep ngat timer 0
EA=1;// Cho phep ngat toan cuc
TR0=1;// Chay timer 0 bat dau dem so chu ki may
}
void main(void)
{
khoitaotimer0();
while(1)
{
phantramxung=9;
delaylong(20000);
phantramxung=4;
delaylong(20000);
}
}
Parallel Port 2
Timer/Counter 0
Timer/Counter 0
Mode
1: 16 Bit Timer/Counter
Timer
TCON: 0x00
TMOD: 0x01
TH0: 0x00
TL0: 0x00
T0 Pin
TF0
Control
Status: Stop
TR0
GATE
INT0#
Load "C:\\Bai giang VDK\\Dong co\\Dongco"
WS 1, `phantramxung
WS 1, `dem
ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess
Build
Command
Find in Files
Output Window
Name
Val...
Locals
Watch #1
Watch #2
Watches
Ready
L:40 C:16
NUM
R/W
Start
4 Yahoo! ...
Yahoo! Mail ...
3 Microsoft...
2 Windows...
Dongco - µ...
Calculator
7:41 AM

Chỉ vào list mode: Chọn chế độ, rồi quan sát giá trị ở TMOD rồi quay lại điền vào chương trình.

Chú ý: Vô cùng quan trọng các bạn chỉ được chạy với động cơ loại nhỏ, nếu động cơ loại to phải có mạch điều khiển riêng không là sẽ cháy chip. Nếu không có mạch điều khiển các bạn có thể làm, có thể mượn của tôi, không thì chịu khó chạy mô phỏng.



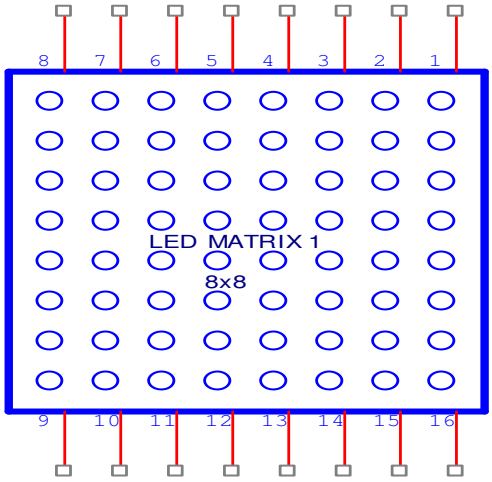
Bài 8:Led ma trận

Nhiệm vụ:

Điều khiển Led ma trận 8x8. Hiện thị dòng chữ chạy “MTC”

Chuẩn bị:

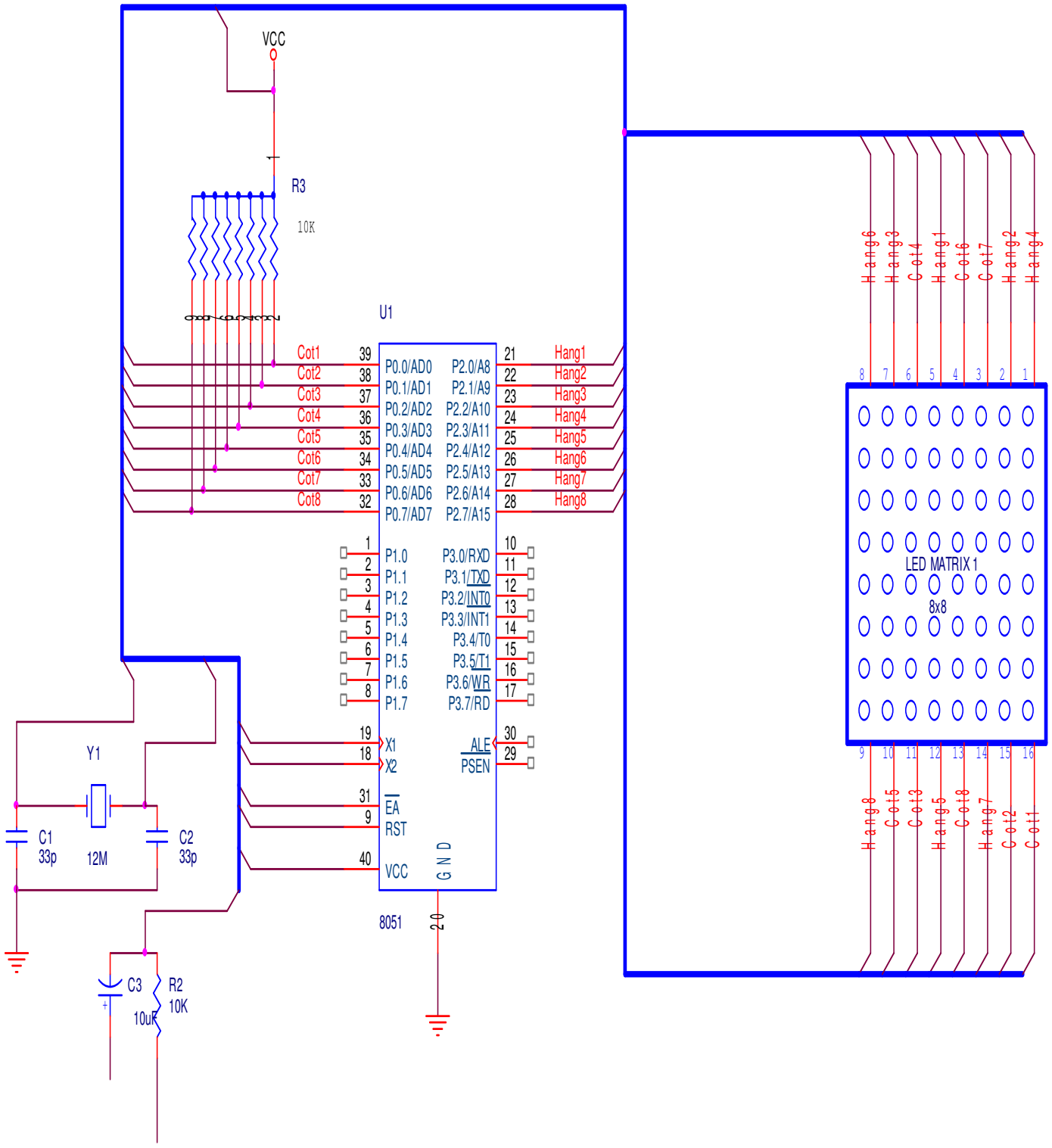
Led ma trận 8x8



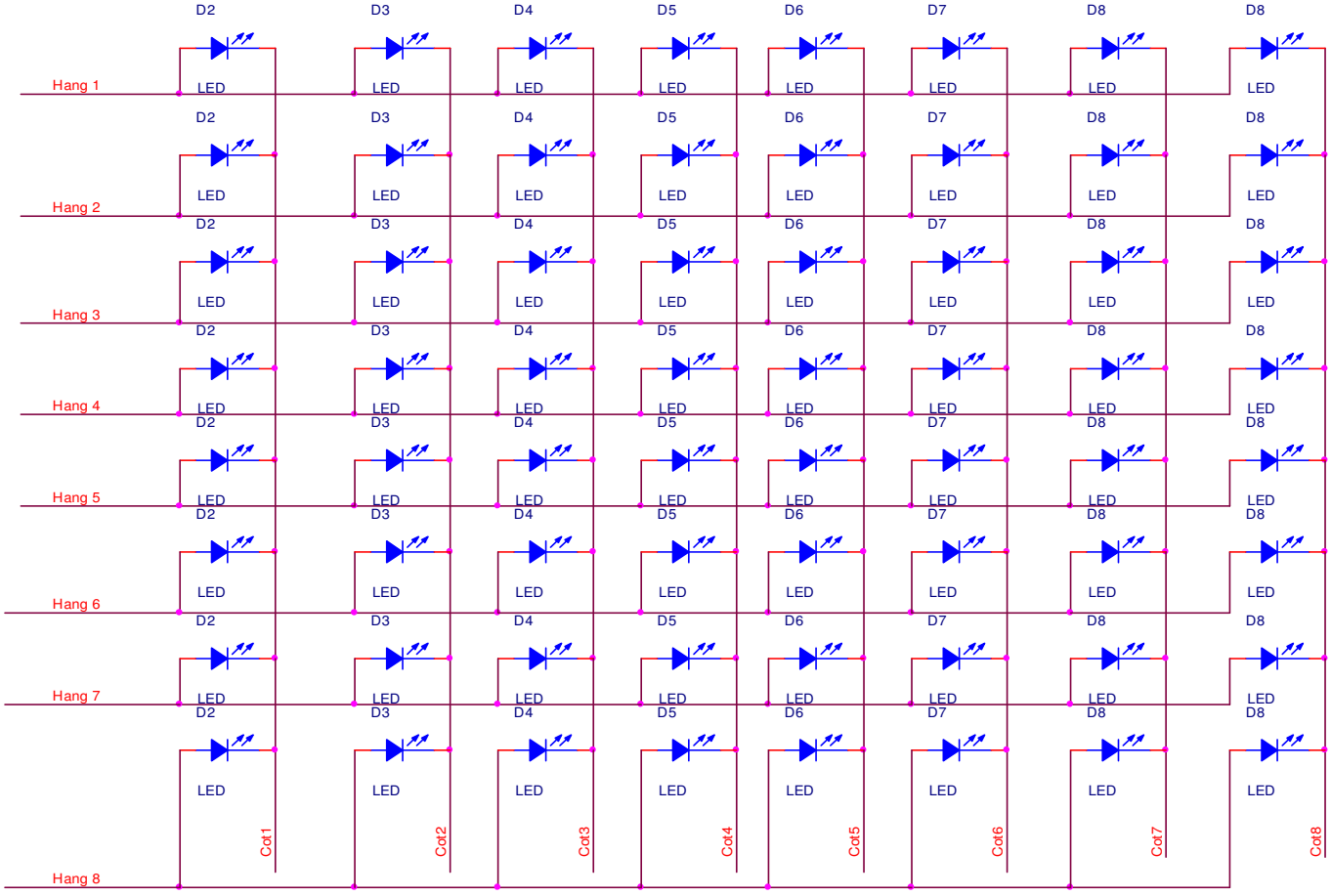
Sơ đồ chân led ma trận 8x8:

Chân	Cột	Hàng	Chân	Cột	Hàng
1		4	9		8
2		2	10	5	
3	7		11	3	
4	6		12		5
5		1	13	8	
6	4		14		7
7		3	15	2	
8		6	16	6	

8.1. Lắp mạch theo sơ đồ sau:


VCC
R3
10K
U1
Cot1
Cot2
Cot3
Cot4
Cot5
Cot6
Cot7
Cot8
P0.0/AD0
P0.1/AD1
P0.2/AD2
P0.3/AD3
P0.4/AD4
P0.5/AD5
P0.6/AD6
P0.7/AD7
P2.0/A8
P2.1/A9
P2.2/A10
P2.3/A11
P2.4/A12
P2.5/A13
P2.6/A14
P2.7/A15
Hang1
Hang2
Hang3
Hang4
Hang5
Hang6
Hang7
Hang8
P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
P3.0/RXD
P3.1/TXD
P3.2/INT0
P3.3/INT1
P3.4/T0
P3.5/T1
P3.6/WR
P3.7/RD
X1
X2
ALE
PSEN
EA
RST
VCC
G N D
8051
Y1
C1
33p
12M
C2
33p
C3
10uF
R2
10K
LED MATRIX 1
8x8

8.2. Nguyên lý hoạt đéng:



Muốn cho led sáng, cấp điện dương 5V vào hàng, 0V vào cột, dòng 10mA đến 15 mA.
Ví dụ: muốn đèn led ở vị trí 5x4 sáng, ta đưa điện áp cột 4(P0_3) xuống 0V, điện áp hàng 5(P2_5) lên 5V.
Hiển thị chữ: thống kê các điểm sáng thành chữ rồi cho các hàng cột điện áp tương ứng. Có thể dùng công cụ debug để lấy giá trị cổng tương ứng với các led sáng.
Giống như quét bàn phím, đưa điện áp 0V ra từng cột nối với cổng 0. Như vậy sẽ có 8 giá trị: 0xFE, 0xFD, 0xFB, 0xF7, 0xEF, 0xDF, 0xBF, 0x7F phảI đưa vào 1 mảng 8 phần tử, rồi sau đó đưa vào 1 vòng for tăng dần 1 biến để tăng phần tử mảng cot[8].
Với mỗi lần 1 chân cổng 0 xuống 0V ta dùng cổng 2 đưa ra 1 giá trị 8 bít để điều khiển trong 1 cột nhứng đèn nào sáng. Ví dụ muốn hàng 1 và hàng 3 sáng thì hàng 1 và 3 có giá trị 5V còn các hàng khác 0V, ta được giá trị 8 bít sau: 0x05 (1010 000).
Tại mỗi thời điểm chỉ có một số đèn trên 1 cột sáng, nhưng do ta quét 8 cột với tần số nhanh, vì mắt có hiện tượng lưu ảnh nên ta thấy trong 1 thời điểm ta thấy toàn bộ kí tự.
Với 8 cột lần lượt bằng 0V ta phảI đưa ra tương ứng 8 giá trị 8 bit ra cổng 2, do đó ta fảI lưu 8 giá trị đó vào 1 mảng 8 kí tự_ kytu1[8], ta sẽ viết các ký tự trên 7 cột. Để mỗi kí tự sẽ cách nhau 1 cột không sáng. Ta khai báo mảng kytu1[9] có 9 phần tử và phần tử đầu tiên có giá trị đẩy ra cổng 2 là 0x00 để tắt toàn bộ cột đó.

Quá trình điều khiển hiển thị như sau:

Cột 1, hàng 1, cột 2 hàng 2, …, cột 8 , hàng 8.

Để làm chữ chạy:

Thêm 1 biến vào để điều khiển thứ tự hiển thị hàng.

Hiện 1 chữ trên led như trên đã đưa ra:

Cột 1, hàng 1, cột 2 hàng 2, …, cột 8 , hàng 8.

Muốn chữ đó dịch chuyển sang tráI ta hiển thị như sau:

Cột 1, hàng 2, cột 2 hàng 3, …, cột 7, hàng 8,cột 8 , hàng 1 ký tự sau.

Cột 1, hàng 3, cột 2 hàng 4, …, cột 7 hàng 1 ký tự sau,cột 8 , hàng 2 ký tự sau.

8.3) Code:

```
#include <REGX51.H>
/* Cot tu P0.0 den P0.7
Hang tu P2.0 den P2.7
De quet dua muc logic 0 lan luot ra cong 0
*/
/* Ham tre */
void delay(long time)
{
long n;
for(n=0; n<time; n++)
{
;
}
}
unsigned char kytu1[9];// Mang 9 phan tu chua gia tri cac hang day ra cong 2
unsigned char k=0;// Bien xac dinh cac ky tu
/* Ham nap gia tri hien thi cac ky tu vao mang kytu1
co 8 gia tri dua ra va 1 gia tri khong bat den nao de cac ky tu cach nhau 1 cot */
void mahoa(unsigned char x)
{
switch(x)
{
// Dau trang
case 0: { kytu1[0]=0x00; kytu1[1]=0x00; kytu1[2]=0x00; kytu1[3]=0x00;
kytu1[4]=0x00; kytu1[5]=0x00; kytu1[6]=0x00; kytu1[7]=0x00; kytu1[8]=0x00;
break; }
// Chu M
case 1: { kytu1[0]=0x00; kytu1[1]=0xFF; kytu1[2]=0x02; kytu1[3]=0x04;
kytu1[4]=0x08; kytu1[5]=0x04; kytu1[6]=0x02; kytu1[7]=0xFF; kytu1[8]=0x00;
break;
}
// Chu T
case 2: { kytu1[0]=0x00; kytu1[1]=0x01; kytu1[2]=0x01; kytu1[3]=0x01;
```

```
kytu1[4]=0xFF; kytu1[5]=0x01; kytu1[6]=0x01; kytu1[7]=0x01; kytu1[8]=0x00;
break;
}
// Chu C
case 3: { kytu1[0]=0x00; kytu1[1]=0x7E; kytu1[2]=0x81; kytu1[3]=0x81;
kytu1[4]=0x81; kytu1[5]=0x81; kytu1[6]=0x42; kytu1[7]=0x00; kytu1[8]=0x00;
break; }
// Dau trang
case 4: { kytu1[0]=0x00; kytu1[1]=0x00; kytu1[2]=0x00; kytu1[3]=0x00;
kytu1[4]=0x00; kytu1[5]=0x00; kytu1[6]=0x00; kytu1[7]=0x00; kytu1[8]=0x00;
break; }
}
}
/* Ham quet led ma tran_ vua hien thi vua dich ky tu dan sang trai*/
void hienthi(void)
{
unsigned char n,m,lap;
unsigned char cot[8]={0xFE,0xFD,0xFB,0xF7,0xEF,0xDF,0xBF,0x7F}; // Cac
phan tu
quet cot
for(m=0; m<8 ; m++)// Dich hien thi
{
for(lap=0; lap<10; lap ++) // Lap hien thi
{
for(n=0; n<8 ; n++)// Quet cot
{
if((n+m)<9 )// Neu n+m < 9 hien thi ky tu 1
{
mahoa(k); // Nap cac gia tri ma hoa ky tu dua ra cac hang (Cong 2)
P0=cot[n]; // Day gia tri 0V ra cong 0 (cac cot)
P2=kytu1[n+m];// Day cac gia tri cac hang (ma hoa ky tu) ra cong 2(cac hang)
delay(45);// Tre du de led sang
}
if((n+m) > 7)// Neu n+m >7 hien thi ky tu 2
{
mahoa(k+1);// Nap gia tri ma hoa ky tu tiep de dua ra cac hang(Cong 2)
P0=cot[n];// Day gia tri logic 0V ra cong 0(cac cot)
P2=kytu1[n+m-8];// Day cac gia tri cac hang (ma hoa ky tu) ra cong 2(cac hang)
delay(45);// Tre du de led sang
```

```
}
P0=0xFF;// Day cac cot len cao
P2=0x00;// Dua cac hang xuong thap de tat toan bo cac led.
}
}
}
}
void main(void)
{
while(1)// Vong lap vo han.
{
hienthi(); // Hien thi 2 ky tu dau tien dau trang va chu M
k=k+1; // Tang k de hien thi chu M va chu T lan tiep
if(k==4) k=0;// Quay vong hien thi
}
}
```

Chú ý:

Mạch led sáng đều tuy nhiên sáng yếu, do lắp trên board chúng ta không lắp các linh kiện khuyêch đại dòng v_ ổn dòng để led sáng đẹp, đều. Khi l_m mạch các bạn thêm các linh kiện khuyêch đại hoặc ổn dòng để đèn sáng đều nh– ULN2003 hoặc ULN 2803 để đ–a giá trị logic ra h_ng, dùng transitor hoặc dùng luôn ULN để quét cột.