GIÁO TRÌNH VI ĐIỀU KHIỂN AVR

Thực hành trên KIT AVR_DKS_03

MỤC LỤC

Bài 1: Điều khiển IO(Vào- Ra) với led đơn.

Bài 1: Điều khiển IO(Vào- Ra) với led đơn.

Yêu cầu:

- Khởi tạo project bằng CodeVision.
- Nạp chương trình.
- Điều khiển led đơn trên KIT theo ý muốn.

Lý thuyết:

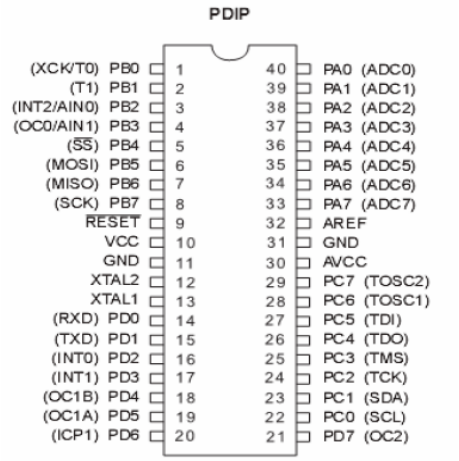
1.Kiến trúc vi điều khiển:

AVR là họ vi điều khiển 8 bit theo công nghệ mới, với những tính năng rất mạnh được tích hợp trong chip của hãng Atmel theo công nghệ RISC, nó mạnh ngang hàng với các họ vi điều khiển 8 bit khác như PIC, Pisoc.Do ra đời muộn hơn nên họ vi điều khiển AVR có nhiều tính năng mới đáp ứng tối đa nhu cầu của người sử dụng, so với họ 8051 89xx sẽ có độ ổn định, khả năng tích hợp, sự mềm dẻo trong việc lập trình và rất tiện lợi.

* Tính năng mới của họ AVR:

- Giao diện SPI đồng bộ.
- Các đường dẫn vào/ra (I/O) lập trình được.
- Giao tiếp I2C.
- Bộ biến đổi ADC 10 bit.
- Các kênh băm xung PWM.
- Các chế độ tiết kiệm năng lượng như sleep, stand by..vv.
- Một bộ định thời Watchdog.
- 3 bộ Timer/Counter 8 bit.
- 1 bộ Timer/Counter 16 bit.
- 1 bộ so sánh analog.
- Bộ nhớ EEPROM.
- Giao tiếp USART..vv.

2. Giới thiệu vi điều khiển Atmega16L:

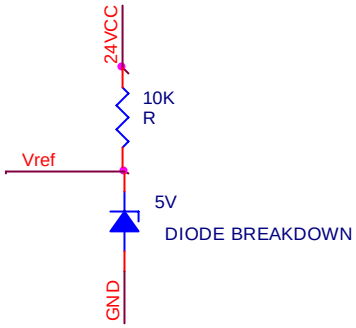


Atmelga16L có đầy đủ tính năng của họ AVR, về giá thành so với các loại khác thì giá thành là vừa phải khi nghiên cứu và làm các công việc ứng dụng tới vi điều khiển. Tính năng**:**

- Bộ nhớ 16K(flash) . - 512 byte (EEPROM). - 1 K (SRAM).
- Đóng vỏ 40 chân , trong đó có 32 chân vào ra dữ liệu chia làm 4 PORT A,B,C,D. Các chân này đều có chế độ pull_up resistors.
- Giao tiếp SPI. - Giao diện I2C. - Có 8 kênh ADC 10 bit.
- 1 bộ so sánh analog. - 4 kênh PWM.
- 2 bộ timer/counter 8 bit, 1 bộ timer/counter1 16 bit.
- 1 bộ định thời Watchdog.
- 1 bộ truyền nhận UART lập trình được.

2.1.Mô tả các chân:

- Vcc và GND 2 chân cấp nguồn cho vi điều khiển hoạt động.
- Reset đây là chân reset cứng khởi động lại mọi hoạt động của hệ thống.
- 2 chân XTAL1, XTAL2 các chân tạo bộ dao động ngoài cho vi điều khiển, các chân này được nối với thạch anh (hay sử dụng loại 4M), tụ gốm (22p).
- Chân Vref thường nối lên 5v(Vcc), nhưng khi sử dụng bộ ADC thì chân này được sử dụng làm điện thế so sánh, khi đó chân này phải cấp cho nó điện áp cố định, có thể sử dụng diode zener:



Hình 2.1. Cách nối chân Vref

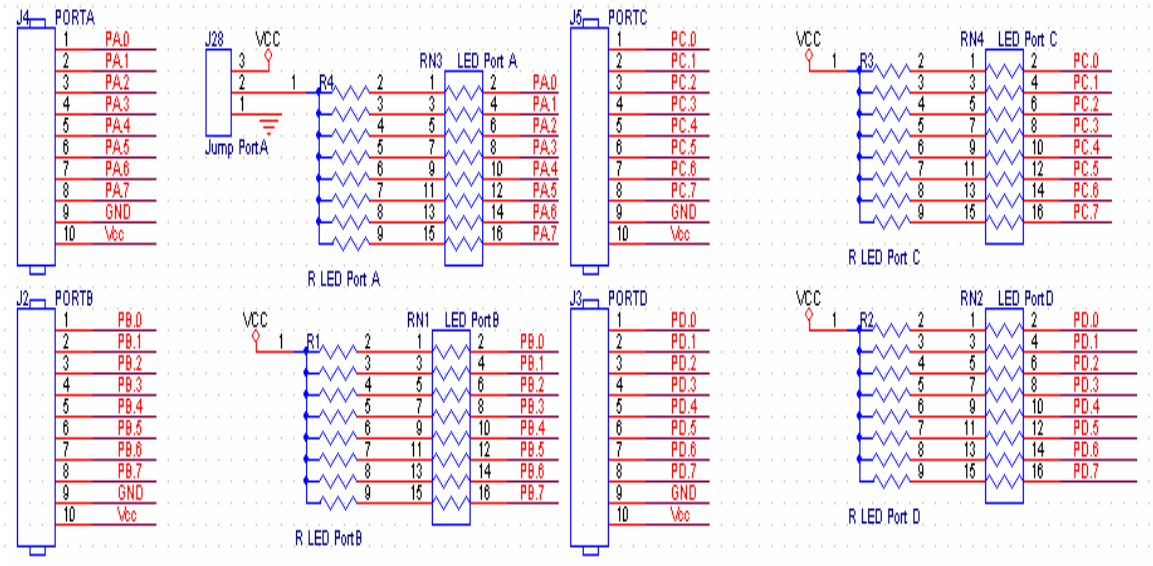
- Chân Avcc thường được nối lên Vcc nhưng khi sử dụng bộ ADC thì chân này được nối qua 1 cuộn cảm lên Vcc với mục đích ổn định điện áp cho bộ biến đổi.

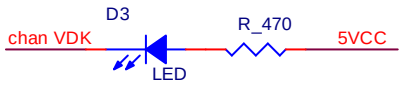
3. Phần mềm lập trình codevision(Hitech):

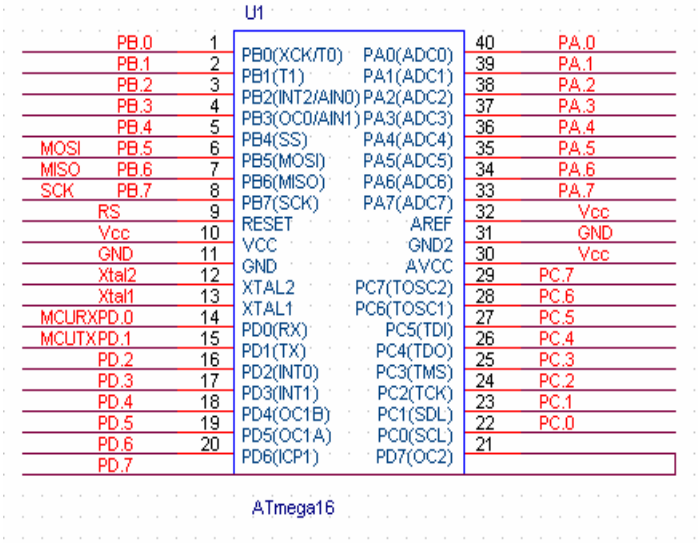
Lựa chọn phần mềm : đây là phần mềm được sử dụng rất rộng dải bởi nó được xây dựng trên nền ngôn ngữ lập trình C, phần mềm được viết chuyên nghiệp hướng tới người sử dụng bởi sự đơn giản, sự hỗ trợ cao các thư viện có sẵn.

3.1.Mô tả phần cứng trên KIT AVR 03:

Các led đơn nối với các cổng vào ra của ATMEGA16L(PORTA-PORTB-PORTC-PORTD). Để led sáng cần đưa mức logic của các chân IO của AVR lên mức cao(5V), để led tắt đưa các chân IO của AVR xuống mức thấp.







3.2.Lập trình:

Thiết lập cổng vào ra:

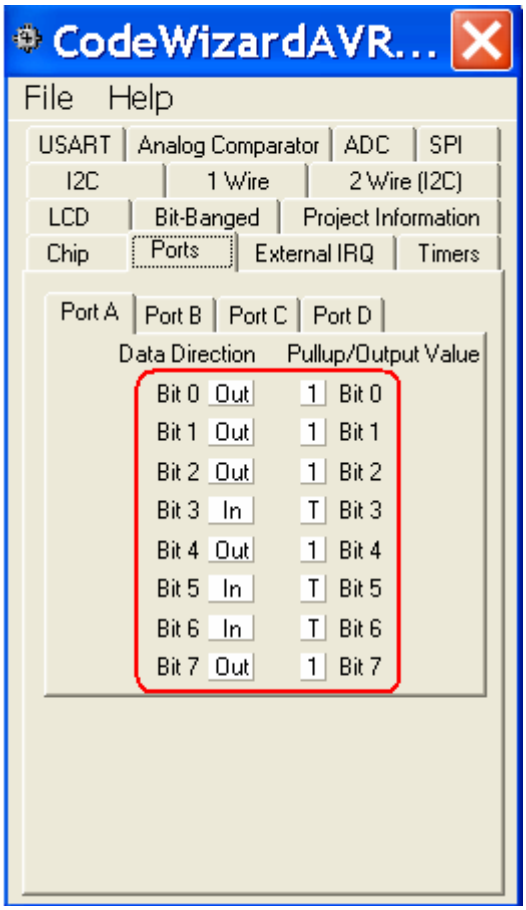
Khi xem xét đến các cổng I/O của AVR thì ta phải xét tới 3 thanh ghi bit ***DDxn,PORTxn,PINxn*.**

-Các bit *DDxn* để truy cập cho địa chỉ xuất nhập *DDRx*. Bit *DDxn* trong thanh ghi *DDRx* dùng để điều khiển hướng dữ liệu của các chân của cổng này.Khi ghi giá trị logic ‘0’ vào bất kì bit nào của thanh ghi này thì nó sẽ trở thành lối vào,còn ghi ‘1’ vào bit đó thì nó trở thành lối ra.

-Các bit *PORTxn* để truy cập tại địa chỉ xuất nhập *PORTx*. Khi *PORTx* được ghi giá trị 1 khi các chân có cấu tạo như cổng ra thì điện trở kéo là chủ động(được nối với cổng).Ngắt điện trở kéo ra, *PORTx* được ghi giá trị 0 hoặc các chân có dạng như cổng ra.Các chân của cổng là 3 trạng thái khi 1 điều kiện reset là tích cực thậm chí xung đồng hồ không hoạt động.

-Các bit *PINxn* để truy cập tại địa chỉ xuất nhập *PINx*. *PINx* là các cổng chỉ để đọc,các cổng này có thể đọc trạng thái logic của *PORTx.PINx* không phải là thanh ghi,việc đọc *PINx* cho phép ta đọc giá trị logic trên các chân của *PORTx*.chú ý *PINx* không phải là thanh ghi,việc đọc *PINx* cho phép ta đọc giá trị logic trên các chân của *PORTx*.

Nếu *PORTxn* được ghi giá trị logic ‘1’ khi các chân của cổng có dạng như chân ra ,các chân có giá trị ‘1’.Nếu *PORTxn* ghi giá trị ‘0’ khi các chân của cổng có dạng như chân ra thì các chân đó có giá trị ‘0’.
Các cổng của AVR đều có thể đọc,ghi. Để thiết lập 1 cổng là cổng vào ,ra thì ta tác động tới các bit *DDxn*, *PORTxn,PINxn*.ta có thể thiết lập để từng bit làm cổng vào,ra cứ không chỉ với cổng,như vậy ta có thể sử lí tới từng bit,đây chính là điểm mạnh của các dòng Vi điều khiển 8 bit.



Ta có thể sử dụng CodeWizardAVR để thiết lập cho các *PORTx* và *Pinx.*
Ví dụ như trên hình:các bit 0,1,2,4,7 của PORTA làm chân ra có trở kéo,còn các bit còn lại làm chân vào. Khi đã thiết lập xong thì các bit 0,1,2,4,7 sẽ có thể xuất dữ liệu ra còn các bit còn lại có thể nhận dữ liệu vào.
Ví dụ :
Ta muốn ghi dữ liệu giá trị logic ’0’ ra PORTA.0 để bật tắt một Led thì:

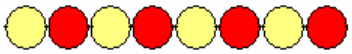
PORTA.0=1;

Ta muốn đọc dữ liệu là một bit từ chân 3 của PORTA:

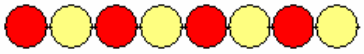
Bit x;

x=PINA.3;

Cũng như vậy khi ta thiết lập PORTA làm cổng ra thì ta có thể xuất dữ liệu ra từ PORTA:

PORTA=0xAA;  **PORTA**

Còn nếu ta thiết lập PORTA làm cổng vào và giá trị hiện thời của PORTA**:**

 **PORTA**

Thì sau câu lệnh đọc giá trị từ PORTA: x=PORTA thì x=0x55. Khi thiết lập PORTA làm cổng ra thì khi reset giá trị của PORTA là PORTA=0xFF;

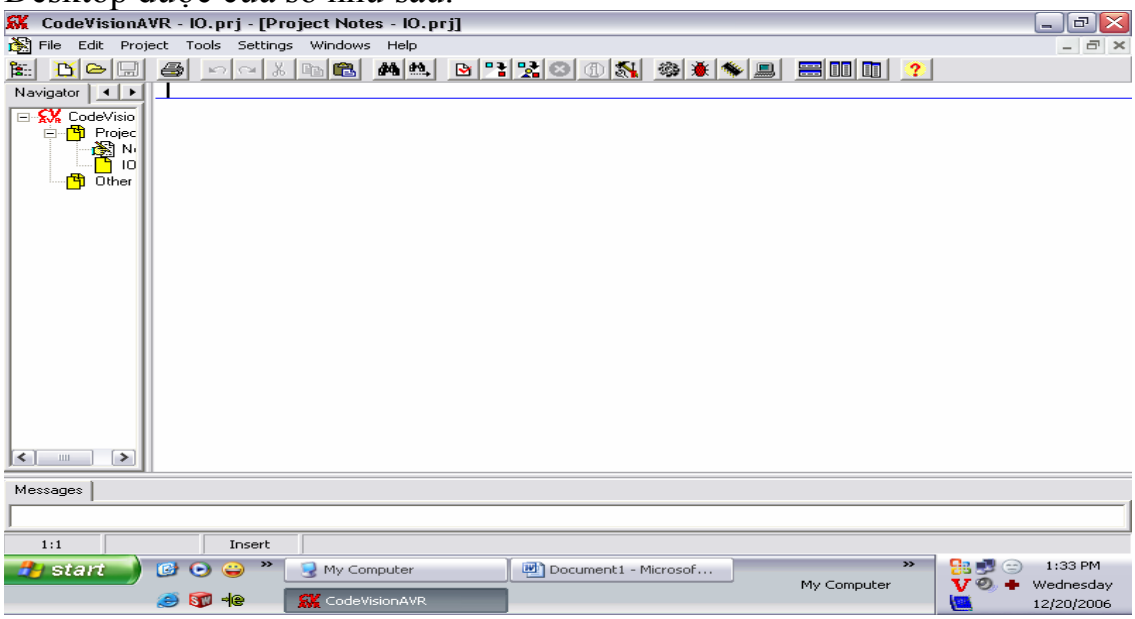
 **PORTA**

Khi thiết lập PORTA làm cổng vào thì khi reset giá trị của PORTA là PORTA=0x00;  **PORTA**

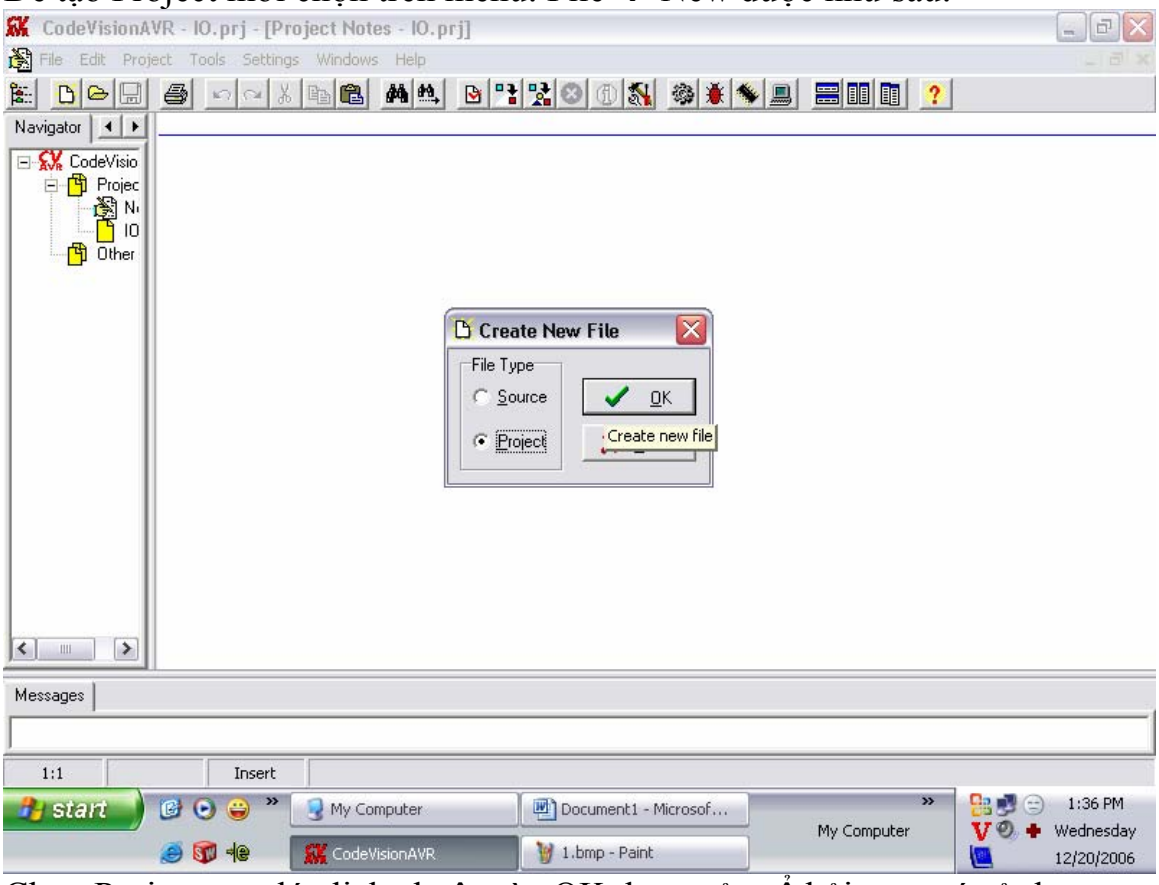
Việc thiết lập cổng vào ra là một việc quan trọng vì tùy theo mục đích sử dụng các cổng nào làm cổng vào ra,thì ta phải thiết lập đúng thì mới có thể sử dụng được, động tác này khác với họ vi điều khiển 8051- AT8951.

CodeVision:

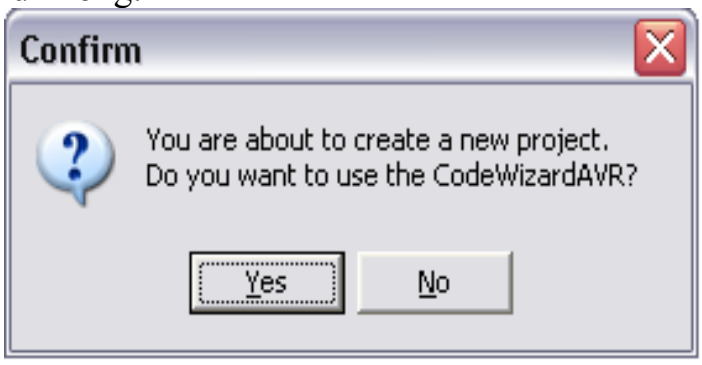
Chạy CodeVision bằng cách click chuột vào ICON của CodeVision trên Desktop được cửa sổ như sau:



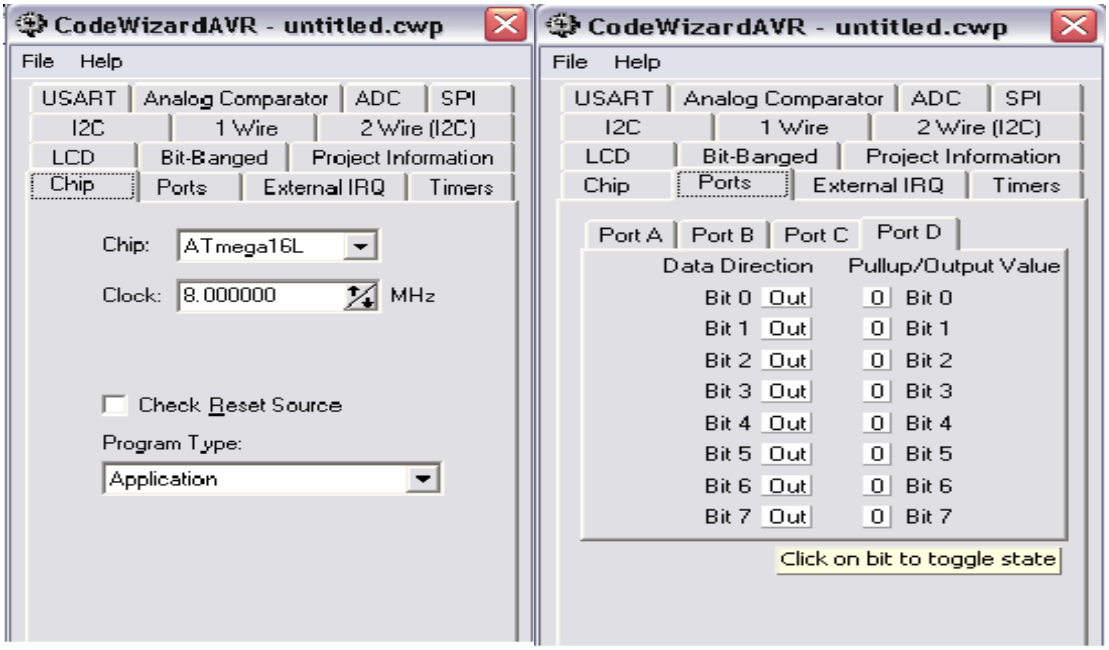
Để tạo Project mới chọn trên menu: File → New được như sau:



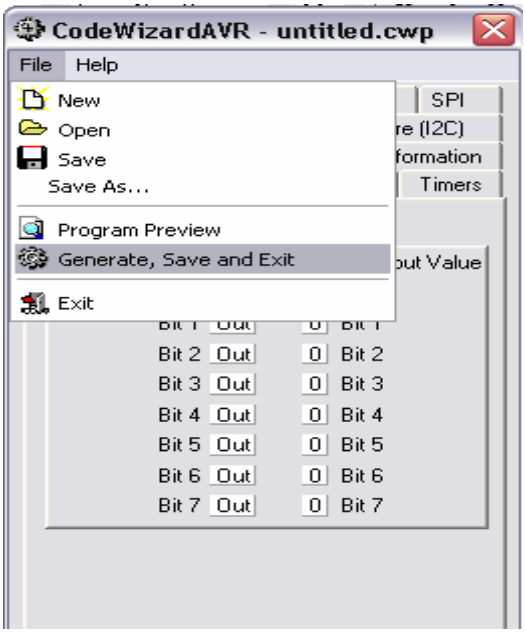
Chọn Project sau đó click chuột vào OK được cửa sổ hỏi xem có sử dụng Code Winzard không:



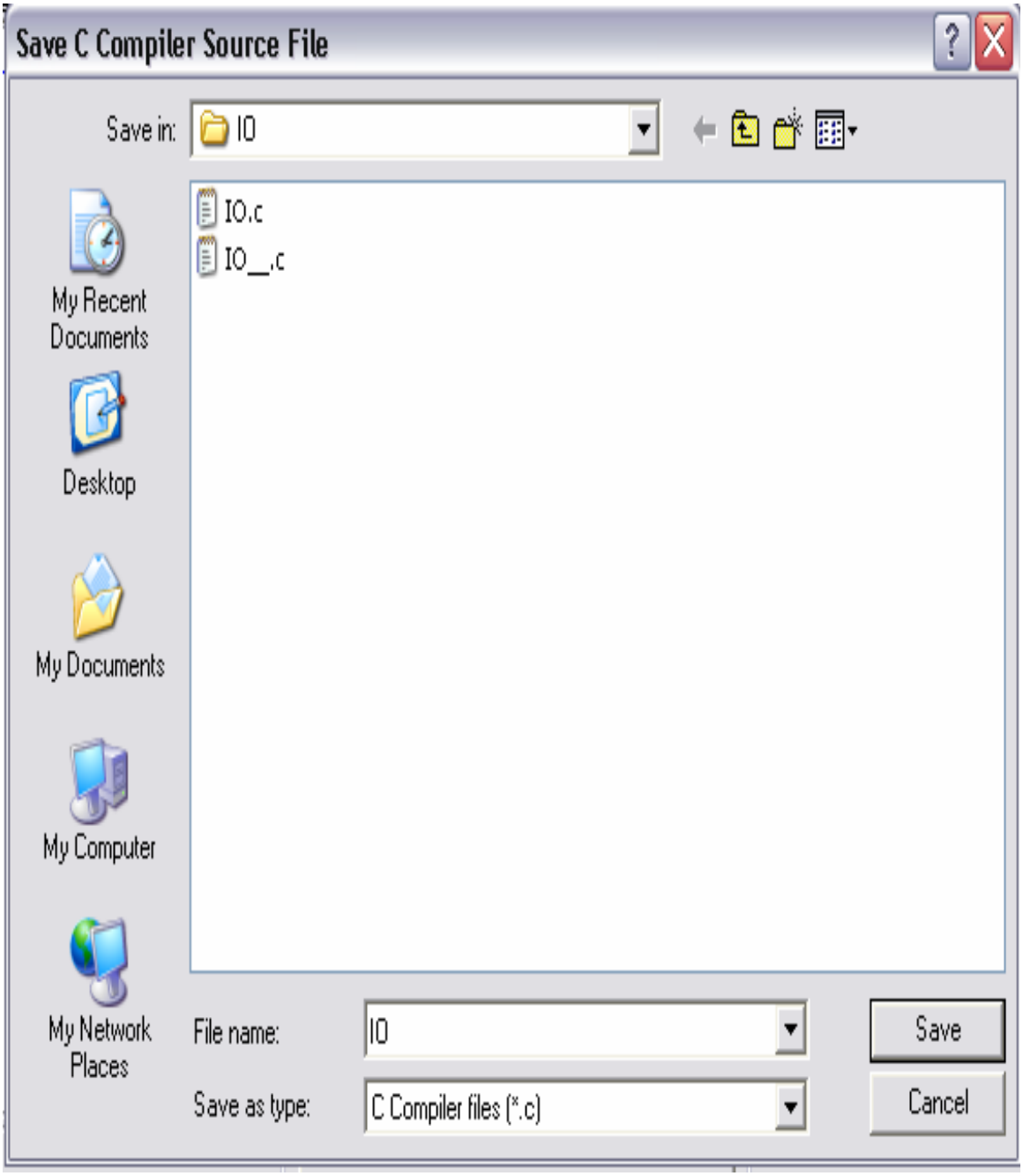
Chọn Yes được cửa sổ CodeWinzardAVR như sau :



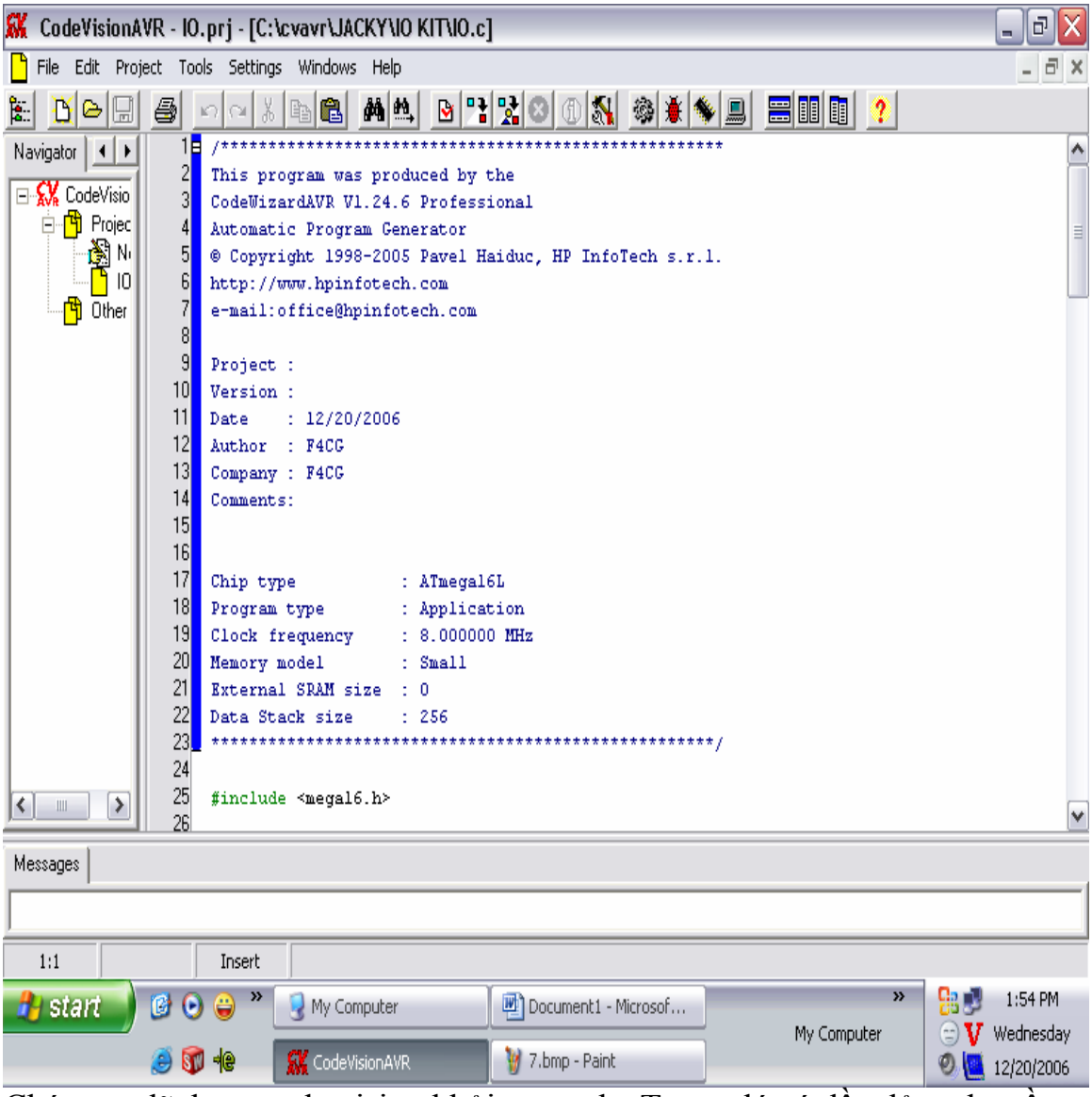
Sử dụng chíp AVR nào và thạch anh tần số bao nhiêu ta nhập vào tab Chip. Để khởi tạo cho các cổng IO ta chuyển qua tab Ports.
Các chân IO của AVR mặc định trạng thái IN, muốn chuyển thành trạng thái OUT để có thể đưa các mức logic ra ta click chuột vào các nút IN (mầu trắng) để nó chuyển thành OUT trong các Tab Port. Sau đó chọn File → Generate, Save and Exit.



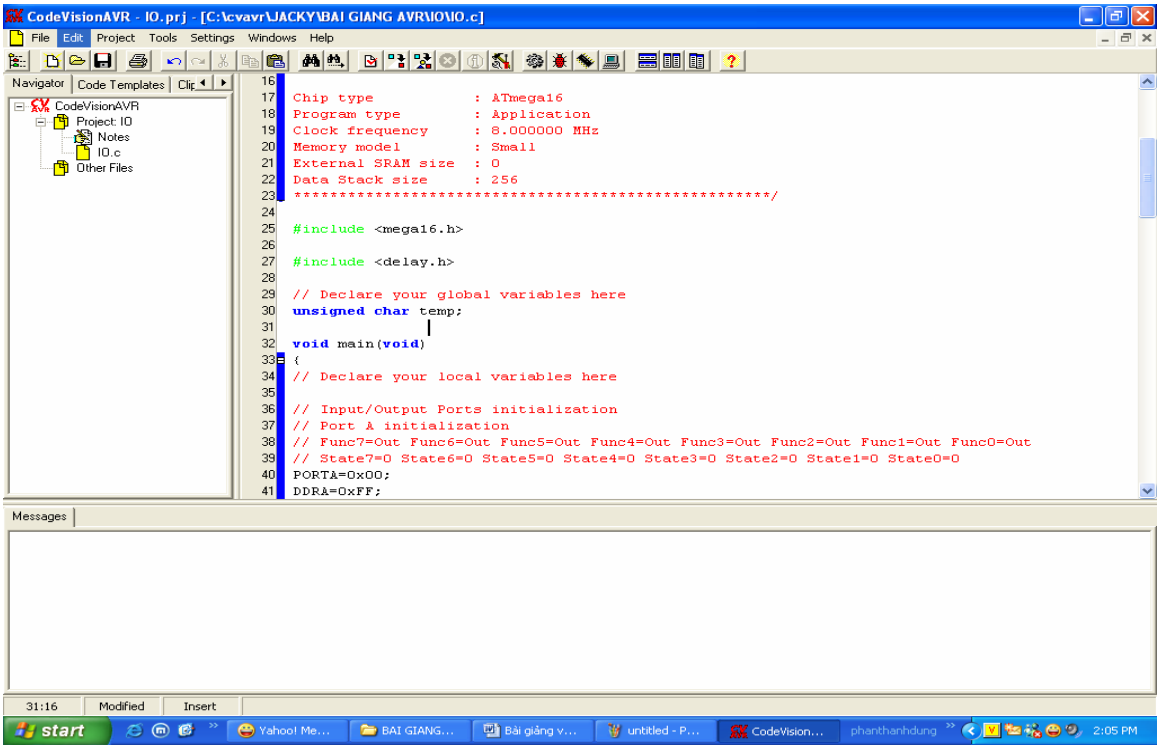
Được cửa sổ yêu cầu nhớ các file của Project. Đây là ví dụ IO nên ta save tên là IO.

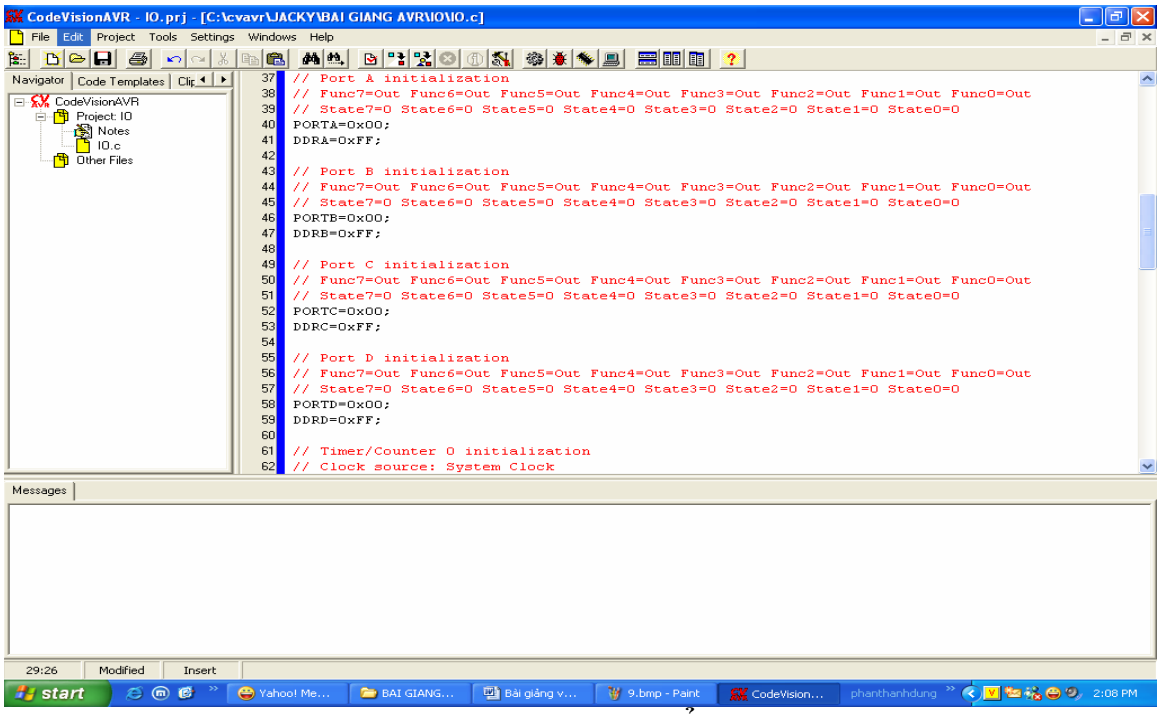


Sau khi nhớ song 3 file : IO.c – IO.prj – IO.cwp được cửa sổ như sau:



Chúng ta đã được code vision khởi tạo code. Trong đó có đầy đủ code cần thiết mà khi nãy chúng ta cấu hình cho cổng IO. Chúng ta bắt đầu soạn code. Để led nhấp nháy chúng ta dùng hàm delay_ms(). Do đó ta thêm thư viện delay.h bằng cách tìm dòng lệnh: #include <mega16.h> ngay đầu chương trình viết ngay dưới dòng lệnh sau:
#include <delay.h>. Để led nhấp nháy ở cổng IO ta đưa ra cổng IO một biến temp có giá trị tăng dần từ 0 đến 255. Do đó ta khai báo thêm một biến unsigned char temp ngay dưới dòng // Declare your global variables here như sau:

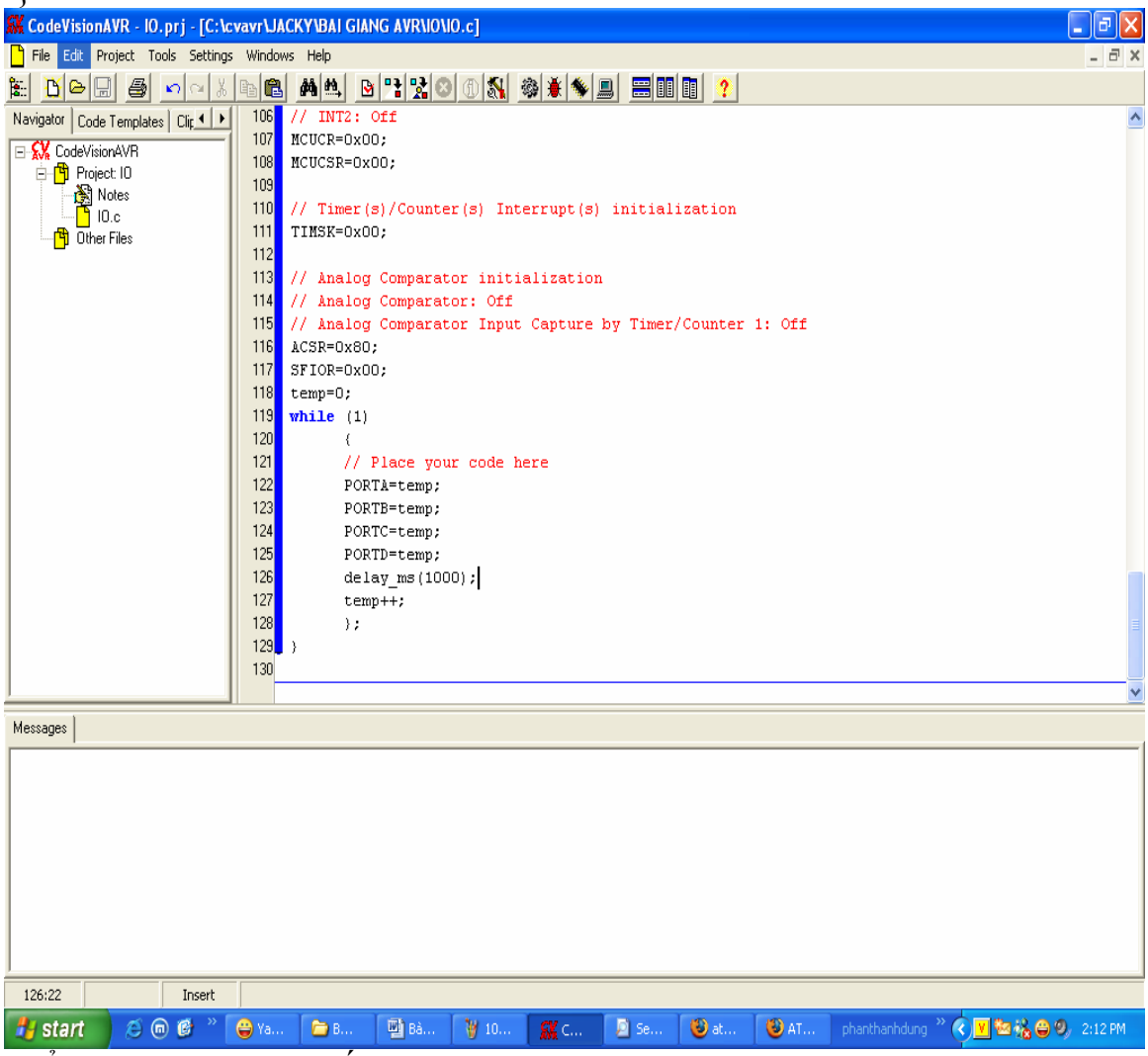




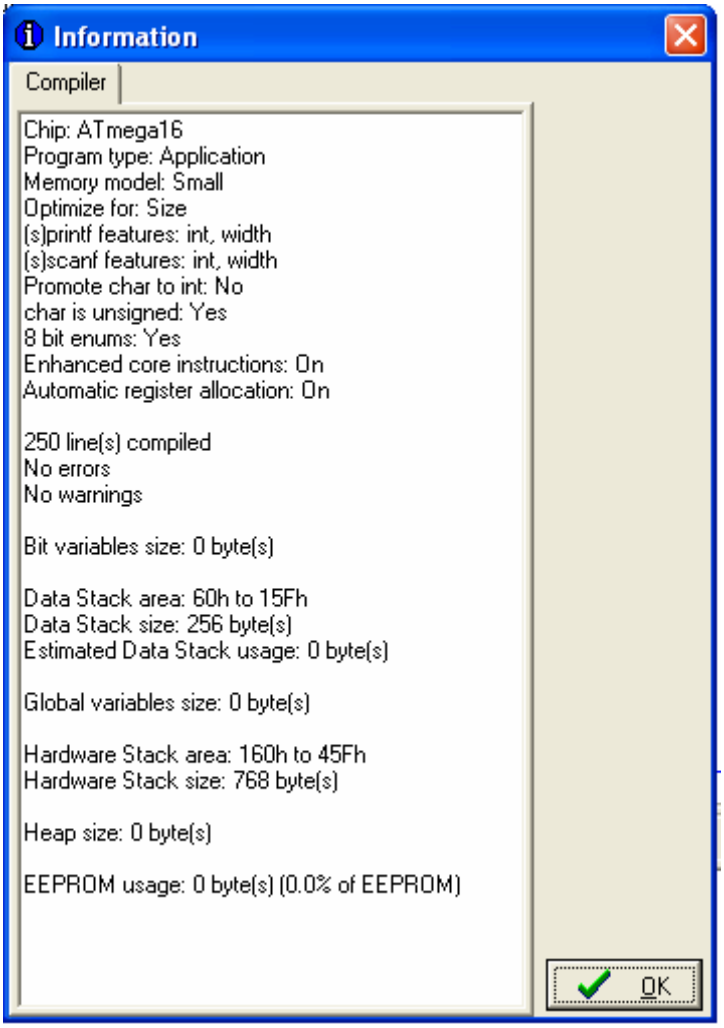
Khởi tạo cho các cổng IO

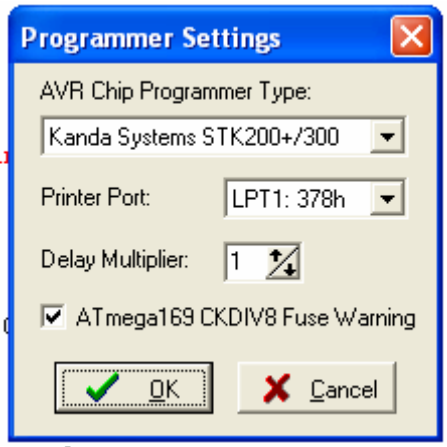
Trong hàm main có vòng while(1). Chúng ta soạn code vào đó như sau:

```
temp=0;
while (1)
{
 // Place your code here
 PORTA=temp;
PORTB=temp;
 PORTC=temp;
PORTD=temp;
 delay_ms(1000);
 temp++;
};
}
```



Để dịch chương trình ấn F9 hoặc vào menu : Project → Compile.
Được cửa sổ Information như sau:

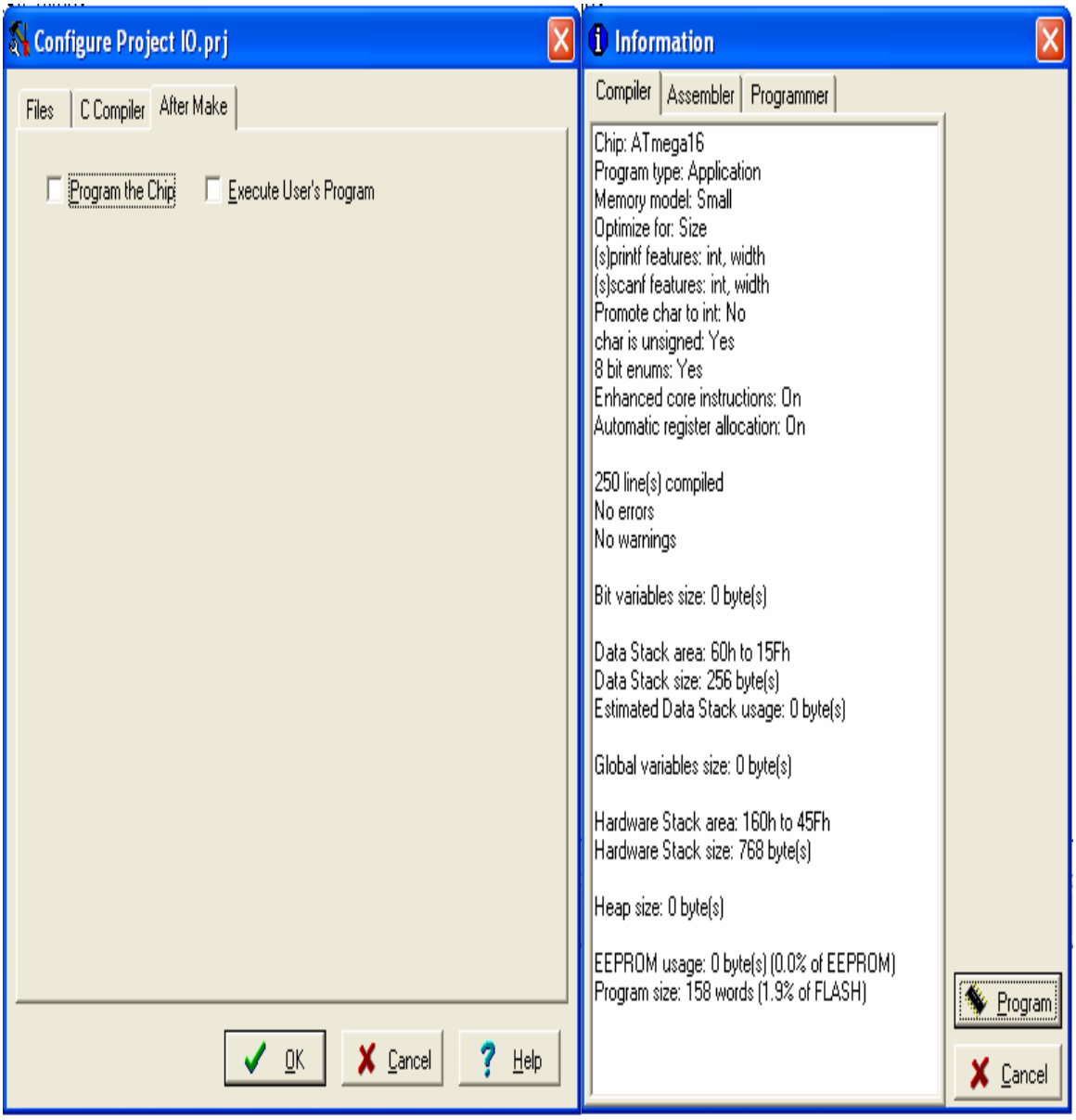




Cấu hình cho mạch nạp

Chương trình không có lỗi. Nhấp OK.
Để nạp chương trình các bạn cần cấu hình cho mạch nạp. Vào menu: Settings → Programmer được cửa sổ như bên cạnh.

Mạch nạp ta dùng STK 200 do đó các bạn chọn Kanda Systems STK200+/300. Nhấp OK. Sau đó các bạn chọn trên menu: Projects → Configure được cửa sổ như sau:



Trong tab After Make các bạn đánh dấu vào Program the Chip và nhấp OK. Nhấn tổ hợp phím Shift + F9 được như hình bên.
Cắm Jump mạch nạp vào .Click vào Program. Đợi nạp xong nhổ jump nạp ra ấn Reset để thấy led chạy.

Bài 2 : Điều khiển vào ra với led 7 đoạn

Bài 2 : Điều khiển vào ra với led 7 đoạn

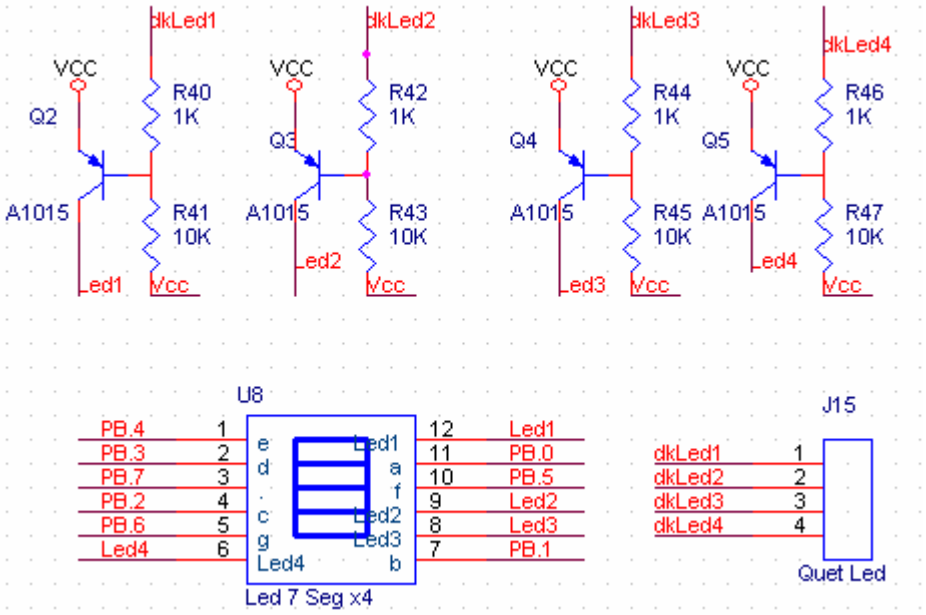
1.Yêu cầu:

Biết phương pháp quét led.

Đưa số bất kỳ ra hàng led.

2.Mô tả:

4 led 7 thanh anot chung, 4 chân anot chung(chân dương) được nối với 4 transitor để ta có thể quét led sử dụng 4 chân của PORTD, các chân điều khiển sáng các thanh còn lại được nối song song nhau và đưa vào PORTB của AVR và có thứ tự như sau: Từ bít 0 → 6 ứng với từ A → G. Bít thứ 7 là dấu chấm.



Vì có 4 led nên ta có thể hiển thị đến hàng nghìn. Do đó đầu vào của ta là một số bất kì lớn tới hàng nghìn. Ta phải tách lấy từng số hàng nghìn, trăm, chục, đơn vị rồi đưa vào 4 biến rồi tùy vào 4 biến số đó mà ta đưa ra từng led. Quét let ta làm như sau: Đưa PORTD.0 xuống 0 để bật nguồn cho led hàng đơn vị, đẩy trị số hàng đơn vị ra PORTB, trễ một khoảng thời gian → đưa PORTD.0 lên một để tắt nguồn led đơn vị, đưa PORTD.1 xuống 0 để bật nguồn cho led hàng chục, đẩy giá trị hàng chục ra PORTB, trễ một

khoảng thời gian, … Cứ làm như vậy đến hàng nghìn. Như vậy tại một thời điểm chỉ có một led sáng chỉ bằng 1/3 thời gian led tắt, nhưng do tần số bật led nhanh, mắt người lưu ảnh nên vẫn thấy led sáng như lúc nào cũng bật nguồn cho led.

3.Thực hành:

Các bước khởi tạo tương tự bài một. Chúng ta soạn thảo code gồm hai hàm như sau và đặt ngay phía trên hàm main như trong hình.

```
void daydulieu(unsigned char x)// Ham dua du lieu ra PORT
{
      switch(x) // Tuy thuoc vao bien dau vao ma dua du lieu ra tu 0...9
      {      //logic 1 tat led, logic 0 bat led
            case 0: { PORTB=0xC0; break; }  // So 0
            case 1: { PORTB=0xF9; break; }  // So 1
            case 2: { PORTB=0xA4; break; }  // So 2
            case 3: { PORTB=0xB0; break; }  // So 3
            case 4: { PORTB=0x99; break; }  // So 4
            case 5: { PORTB=0x92; break; }  // So 5
            case 6: { PORTB=0x82; break; }  // So 6
            case 7: { PORTB=0xF8; break; }  // So 7
            case 8: { PORTB=0x80; break; }  // So 8
            case 9: { PORTB=0x90; break; }  // So 9
      }
}
void hienthi(int n)
{
      int a,b,c,d;
      // Lay cac so cac hang
      a= n/1000;            // lay hang
nghin
      b=(n-a*1000)/100;      // lay hang
tram
      c=(n-a*1000-b*100)/10;  // lay
hang chuc
      d=(n-a*1000-b*100-c*10);// lay
hang don vi

      // Quet led
      PORTD=0xFE;// led dau tien
      daydulieu(d);// day ra hang don vi
      delay_ms(10);// tre
      PORTB=0xFF;// tat toan bo led
      PORTD=0xFD;//led thu hai
      daydulieu(c);// dua ra hang chuc
      delay_ms(10);// tre
      PORTB=0xFF;// tat toan bo led
      PORTD=0xFB;
      daydulieu(b);
      delay_ms(10);
      PORTB=0xFF;
      PORTD=0xF7;
      daydulieu(a);
      delay_ms(10);
      PORTB=0xFF;
}
```

CodeVisionAVR - 7 thanh.prj - [E:\cvavr\JACKY\Led 7 thanh\7 thanh.c]

File Edit Project Tools Settings Windows Help

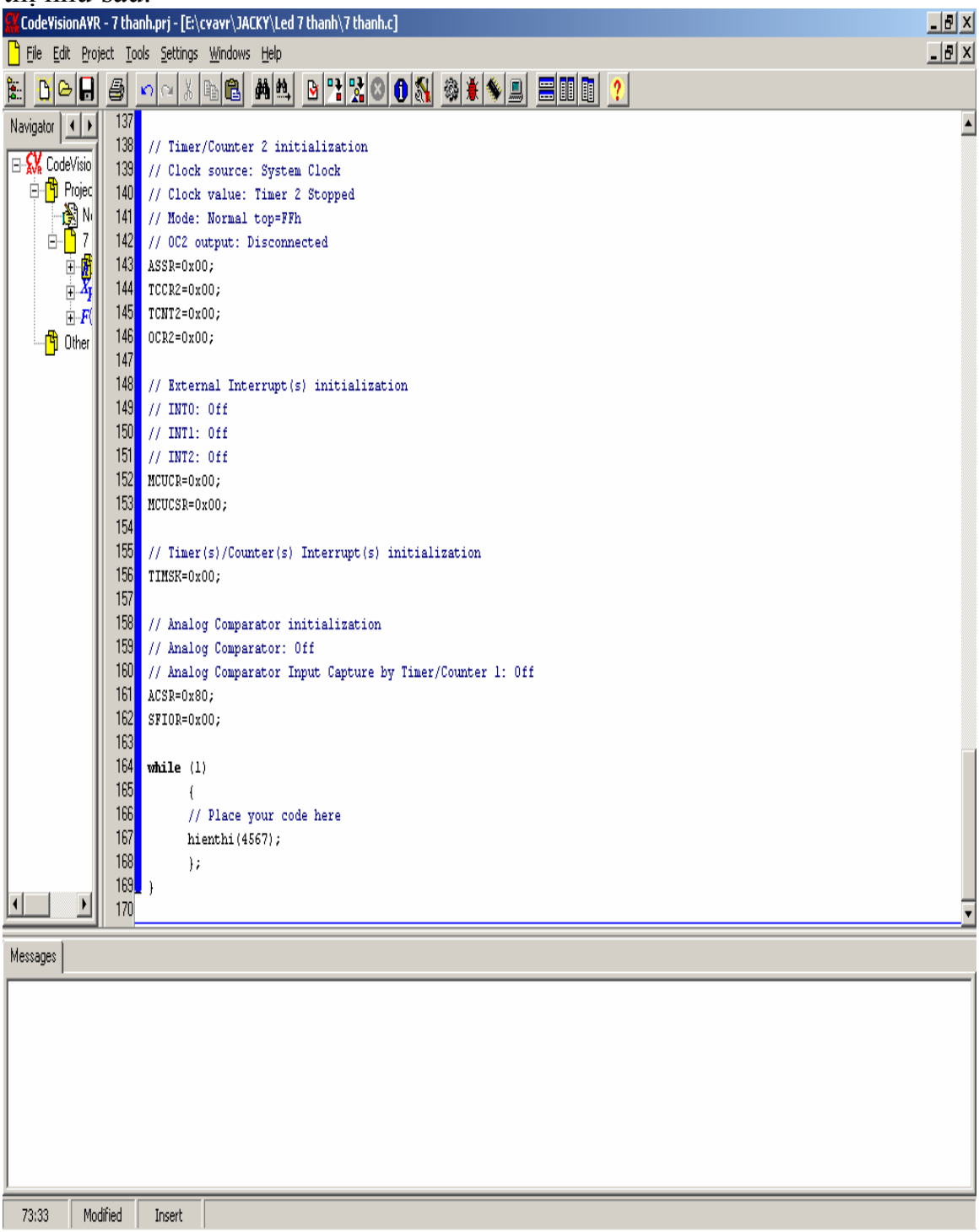
Navigator

```
48
49  void hienthi(int n)
50  {
51          int a,b,c,d;
52          // Lay cac so cac hang
53          a= n/1000;              // lay hang nghin
54          b=(n-a*1000)/100;       // lay hang tram
55          c=(n-a*1000-b*100)/10;  // lay hang chuc
56          d=(n-a*1000-b*100-c*10);// lay hang don vi
57
58          // Quet led
59          PORTD=0xFE;// led dau tien
60          daydulieu(d);// day ra hang don vi
61          delay_ms(10);// tre
62          PORTB=0xFF;// tat toan bo led
63          PORTD=0xFD;//led thu hai
64          daydulieu(c);// dua ra hang chuc
65          delay_ms(10);// tre
66          PORTB=0xFF;// tat toan bo led
67          PORTD=0xFB;
68          daydulieu(b);
69          delay_ms(10);
70          PORTB=0xFF;
71          PORTD=0xF7;
72          daydulieu(a);
73          delay_ms(10);
74          PORTB=0xFF;
75  }
76
77  void main(void)
78  {
79  // Declare your local variables here
80
81  // Input/Output Ports initialization
```

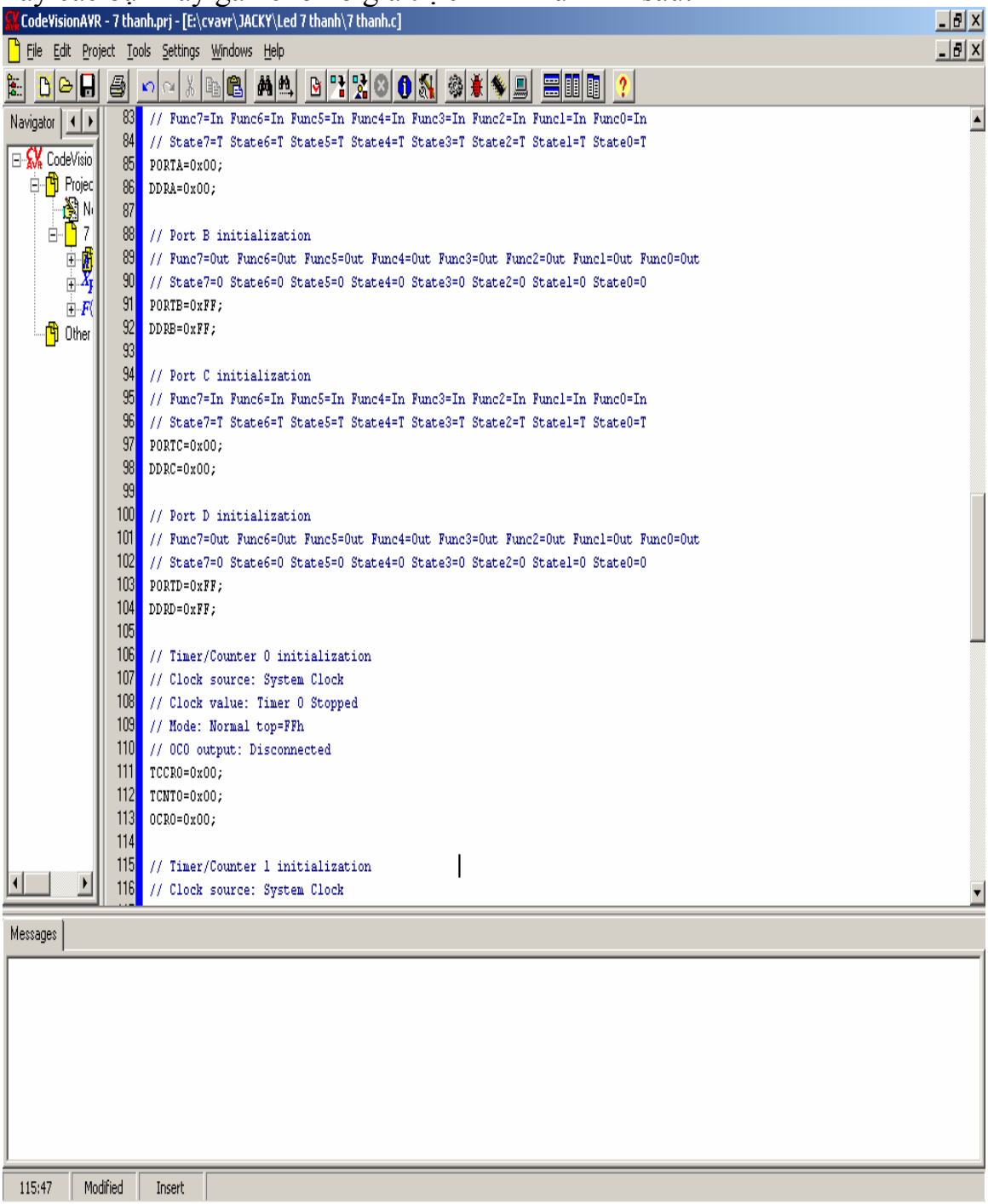
Messages

70:28 Modified Insert

Trong vòng while(1) trong hàm main ta chỉ dùng một câu lệnh gọi hàm hiển thị như sau:



Chú ý: trong bài này vì nếu đưa các PORTD và PORTB lúc khởi tạo bằng 0x00 thì tất cả các led sẽ sáng do đó tại các câu lệnh khởi tạo cho hai PORT này các bạn hãy gán cho nó giá trị 0xFF như hình sau:



Bài 3: Điều khiển IO với LCD

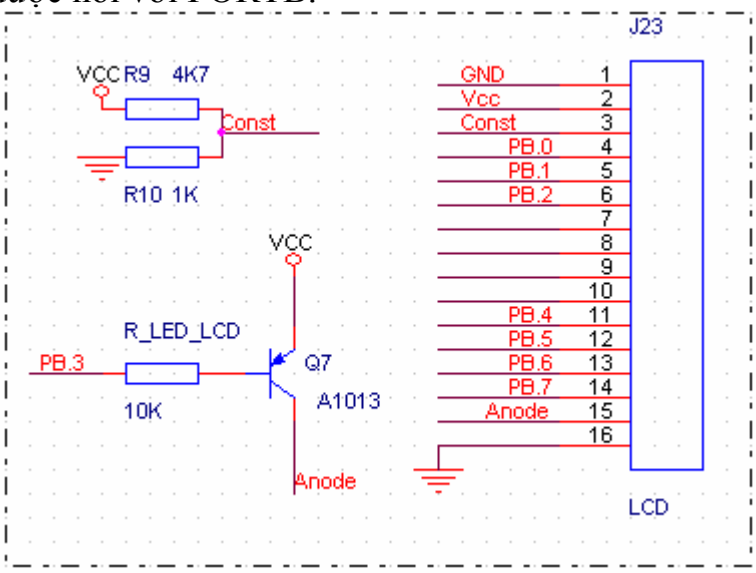
Bài 3: Điều khiển IO với LCD

1.Yêu cầu:

Biết khởi tạo cho LCD với CodeWinzardAVR với bất kỳ cổng nào.
Hiển thị ra LCD các ký tự bất kỳ.

2.Mô tả:

LCD được nối với PORTB.



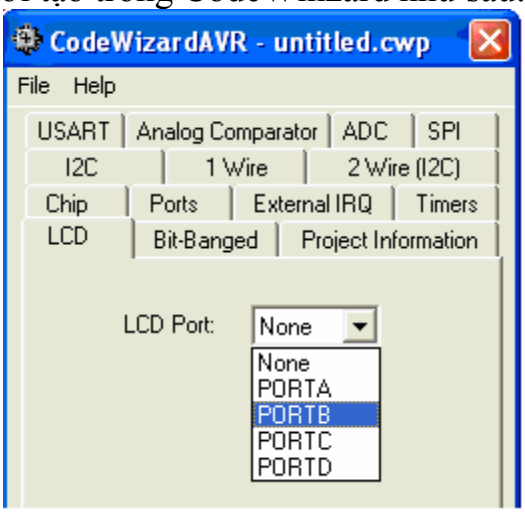
3.Lý thuyết:

Chức năng của LCD trong hầu hết các mạch, các bộ điều khiển đảm nhân vai trò hiển thị các thông số, các thông tin mà chúng ta muốn nhập vào hay các thông tin xử lý mà bộ điều khiển đang hoạt động đực hiển thị ra màn hình, giúp chúng ta giao tiếp gần hơn với quá trình hoạt đông của hệ thống. Loại LCD mà chúng ta sử dụng là loại SD-DM1602A 2 dòng mỗi dòng 16 kí tự, loại này do Trung Quốc sản xuất . Nó có 16 chân như hình vẽ. Trong đó chúng ta có thể thấy 2 chân 1,2 được cấp nguồn cho LCD hoạt động, chân thứ 3 (chân VSS) được nối vào đầu ra của biến trở dùng để điều chỉnh độ tương phản (phải điều chỉnh VSS hợp lý thì LCD mới hiển thị được) 2 chân 15,16 đây là 2 chân cấp nguồn dung để bật đèn của LCD từ chân 4->14 là các chân điều khiển được nối với vi điều khiển, các chân 4,5,6 được để điều khiển hoạt động của LCD, các chân còn lại là 8 bit Data dùng để truyền nhận dữ liệu. Chúng ta có thể giao tiếp Data 8 bit hoặc 4 bit như

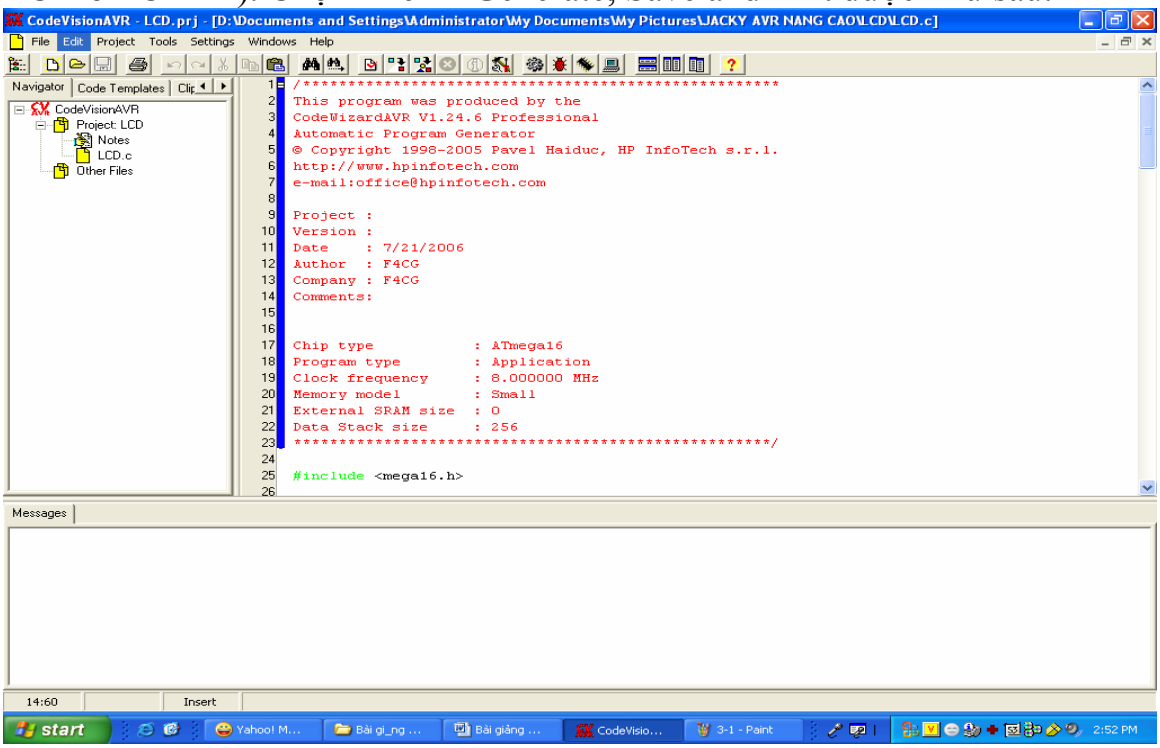
trong mạch của chúng ta truyền Data dưới dạng 4 bit. Việc truyền dưới dạng 4bit hoặc 8 bit phải được thiết lập cả phần cúng và phần mềm.

4.Thực hành:

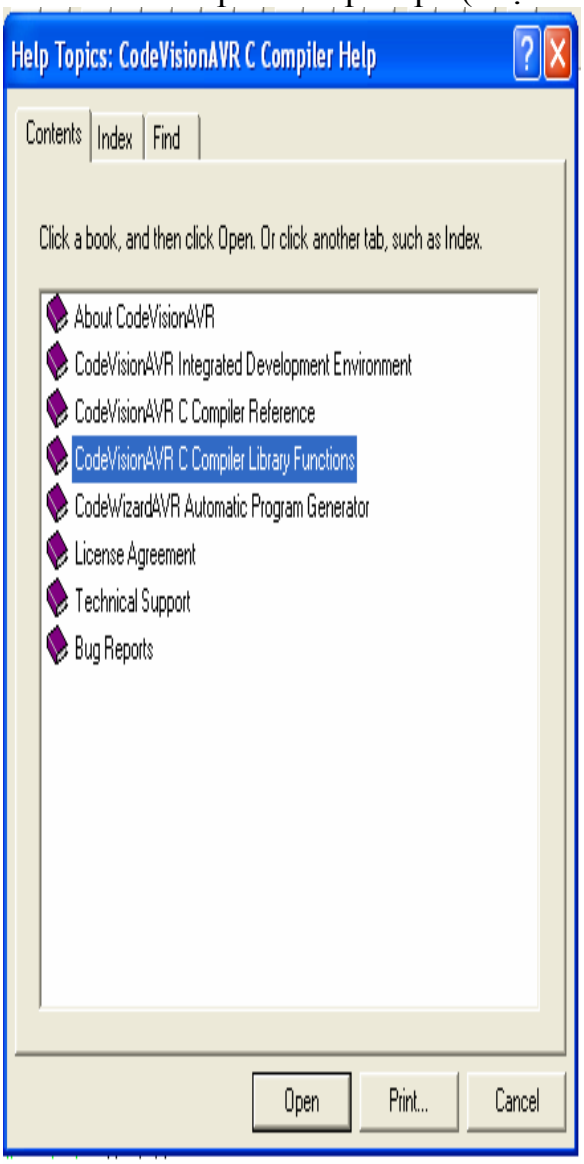
Các bước khởi tạo trong CodeWinzard như sau:

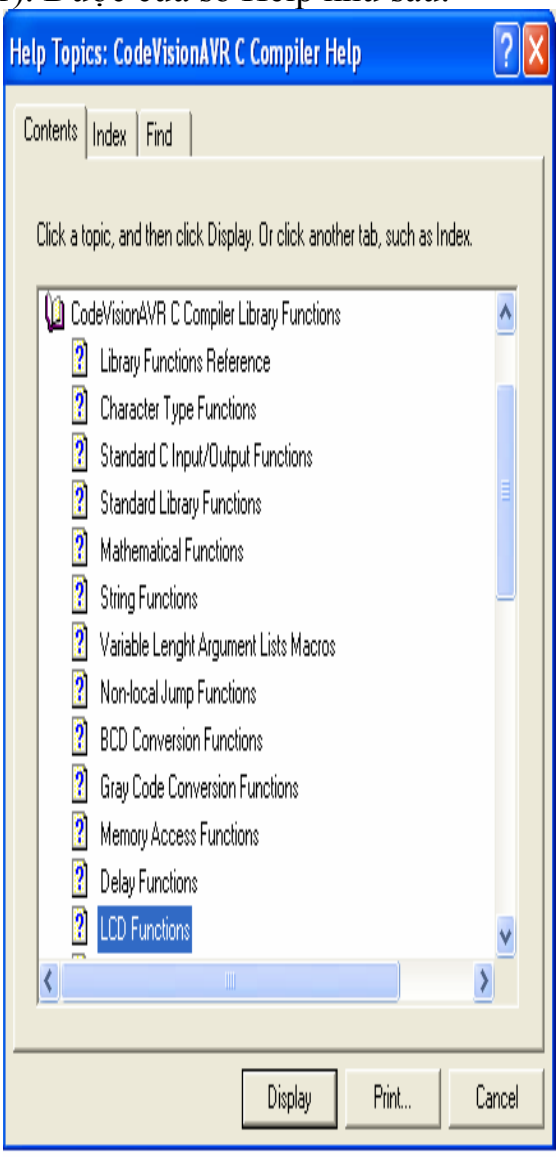


Trong cửa sổ CodeWinzard, chọn tab LCD, trong list mặc định là None, các bạn chuyển thành PORTB cho phù hợp với phần cứng của KIT(thiết kế LCD ở PORTB). Chọn File → Generate, Save and Exit được như sau:

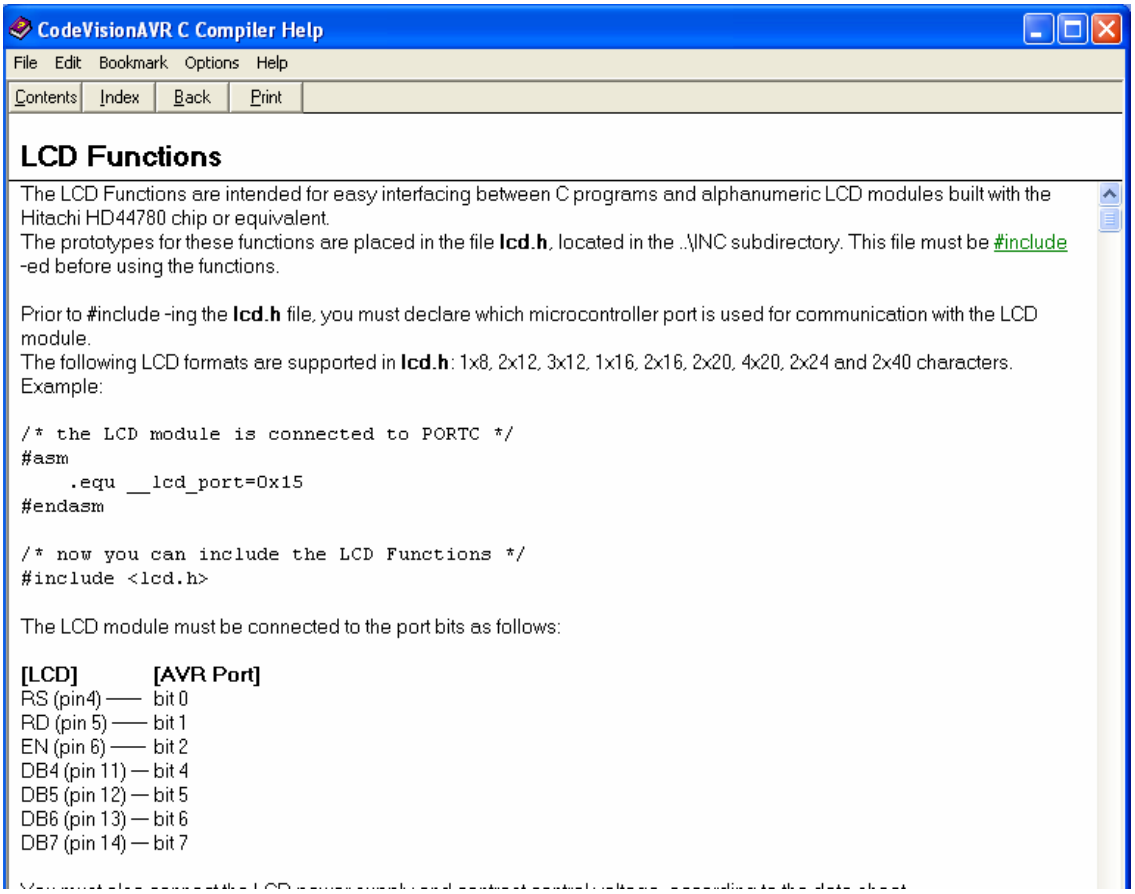


Code cho LCD các bạn có thể tham khảo trong Help bằng cách chọn trên menu Help → Help Topic(hoặc ấn F1). Được cửa sổ Help như sau:





Trong tab Contents, click đúp chuột vào CodeVisionAVR C Compiller Library Functions được như bên cạnh. Nhấp đúp vào LCD Functions để tham khảo các hàm cho LCD.



Trong vòng while(1) trong hàm main ta viết các câu lệnh như sau:

```
while (1)
    {
    // Place your code here
    lcd_gotoxy(0,0);// Dua con tro ve goc, dong 0, cot 0
    lcd_putsf("DKS-MTC-JACKY");// Hien thi dong chu
    lcd_gotoxy(0,1);// Dua con tro ve dong 1, cot 0
    lcd_putsf("Wellcome you");  // Hien thi dong chu
    delay_ms(3000); // Tre 3 s
    lcd_gotoxy(0,0); // Dua con tro ve dong 0 cot 0
    lcd_putsf("embestdks.com"); // Hien thi dong chu
    delay_ms(3000); // Tre 3 s
    };
```

Bài 4: ADC với LM35

Bài 4: ADC với LM35

1.Yêu cầu:

Đo được nhiệt độ từ LM35 hiển thị lên LCD.

2.Lý thuyết:

Đối với ATMEGA 16L: 8 chân của PORTA sử dụng làm 8 kênh đầu vào ADC. Để sử dụng tính năng ADC của Atmega 16L chúng ta cần phải thiết kế phần cứng của Vi điều khiển như sau :

* Chân AVCC chân này bình thường khi thiết kế mạch chúng ta đưa lên Vcc(5V) nhưng khi trong mạch có sử dụng các kênh ADC của phần cứng thì chúng ta phải nối chân này lên Vcc qua 1 cuộn cảm nhằm mục đích cấp nguồn ổn định cho các kênh (đầu vào) của bộ biến đổi.

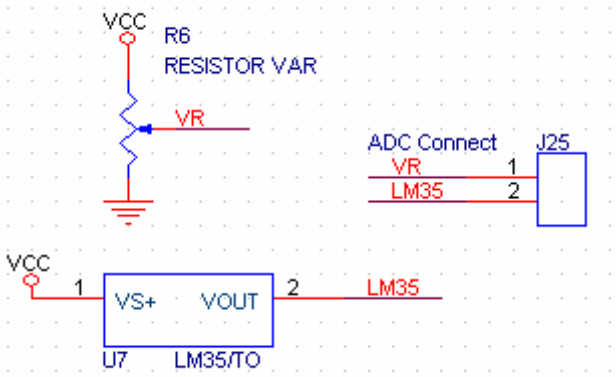
* Chân AREF chân này cần cấp 1 giá trị điện áp ổn định được sử dụng làm điện áp tham chiếu, chính vì vậy điện áp cấp vào chân này cần ổn định vì khi nó thay đổi làm giá trị ADC ở các kênh thu được bị trôi (thay đổi) không ổn định với 1 giá trị đầu vào chúng ta có công thức tính như sau:

$$ADCx=(V\_INT*1024)/ AREF$$

chỉ dựa vào công thức chúng ta củng có thể thấy giá trị ADCx tỉ lệ thuận với điện áp vào V_INT. Giá trị ADC thu được từ các kênh được lưu vào 2 thanh ghi ADCH và ADCL khi sử dụng chúng ta phải đọc giá trị từ các thanh ghi này, khi sử dụng ở ché độ 8 bít thì chỉ lưu vào thanh ghi ADCL.

3.Mô tả:

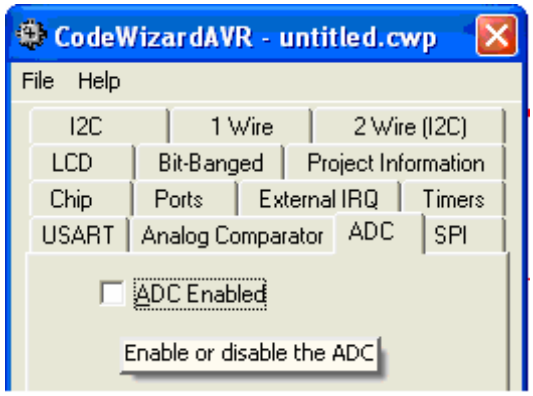
Đầu ra của LM35 và chân 2 biến trở 1K trên Kit được nối vơi 2 jump chờ. Với AMEGA16L có 8 kênh ADC là chức năng thứ 2 của PORTA. Do đó để ADC ta dung dây nối 2 chân đó với 2 bit của PORTA là bit 0 và bit 1..

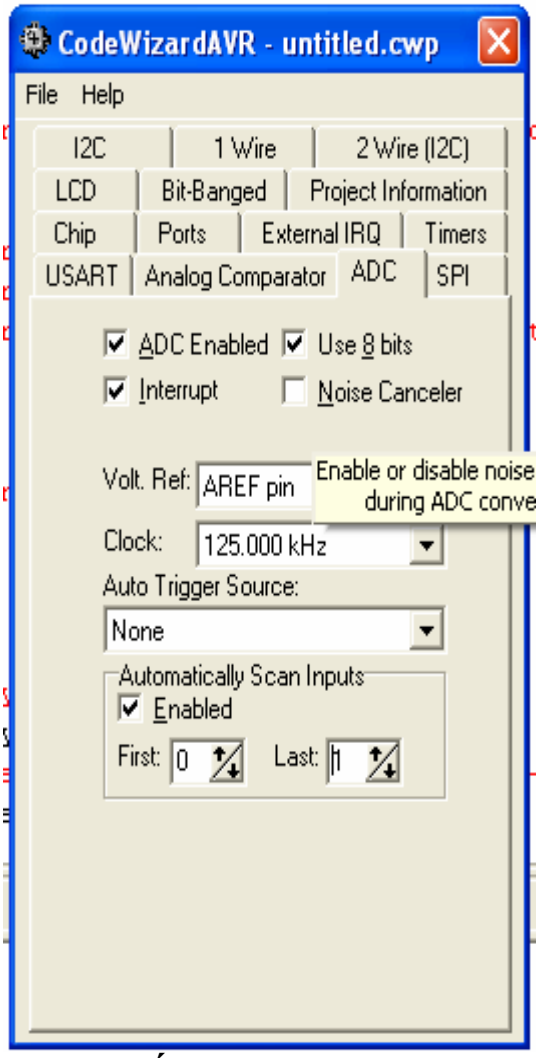


Theo datasheet LM35 thì cứ 10mV tương ứng với 1^0C, ở 0^0C điện áp ra là 0V, tương ứng với giá trị ADC là 0. Với Vref=5V, giá trị của ADC từ 0 đến 256, lấy tròn 250 mức. Mỗi giá trị ADC ứng với 5V/250= 20 mV. Vậy 1 giá trị ADC ứng với 2^0C. Muốn tăng độ phân giải ADC ta giảm Vref.

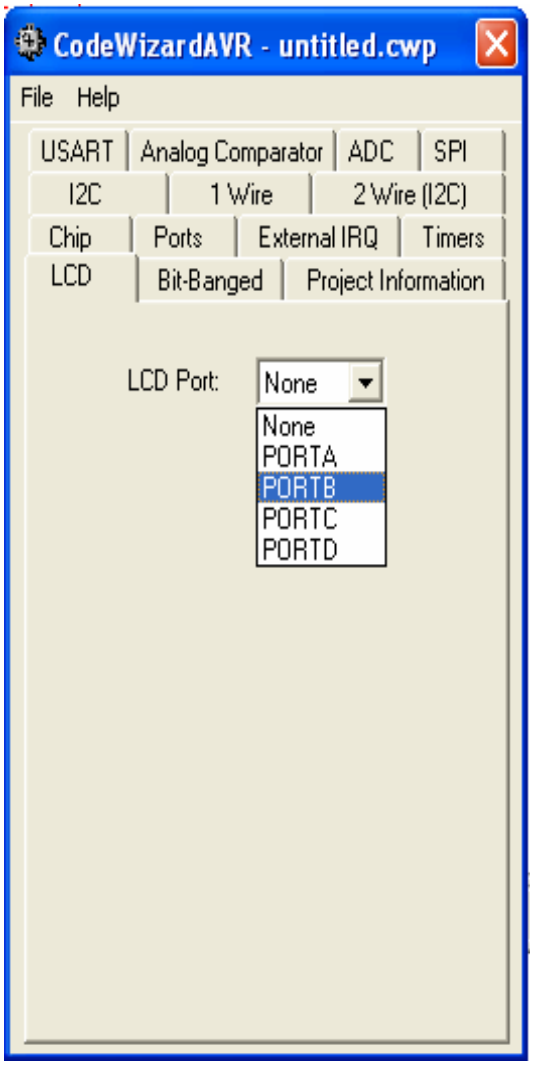
4.Thực hành: Các bước khởi tạo code như sau:

Trong tab ADC check vào ADC enable:





Cấu hình ADC.



Khởi tạo LCD.

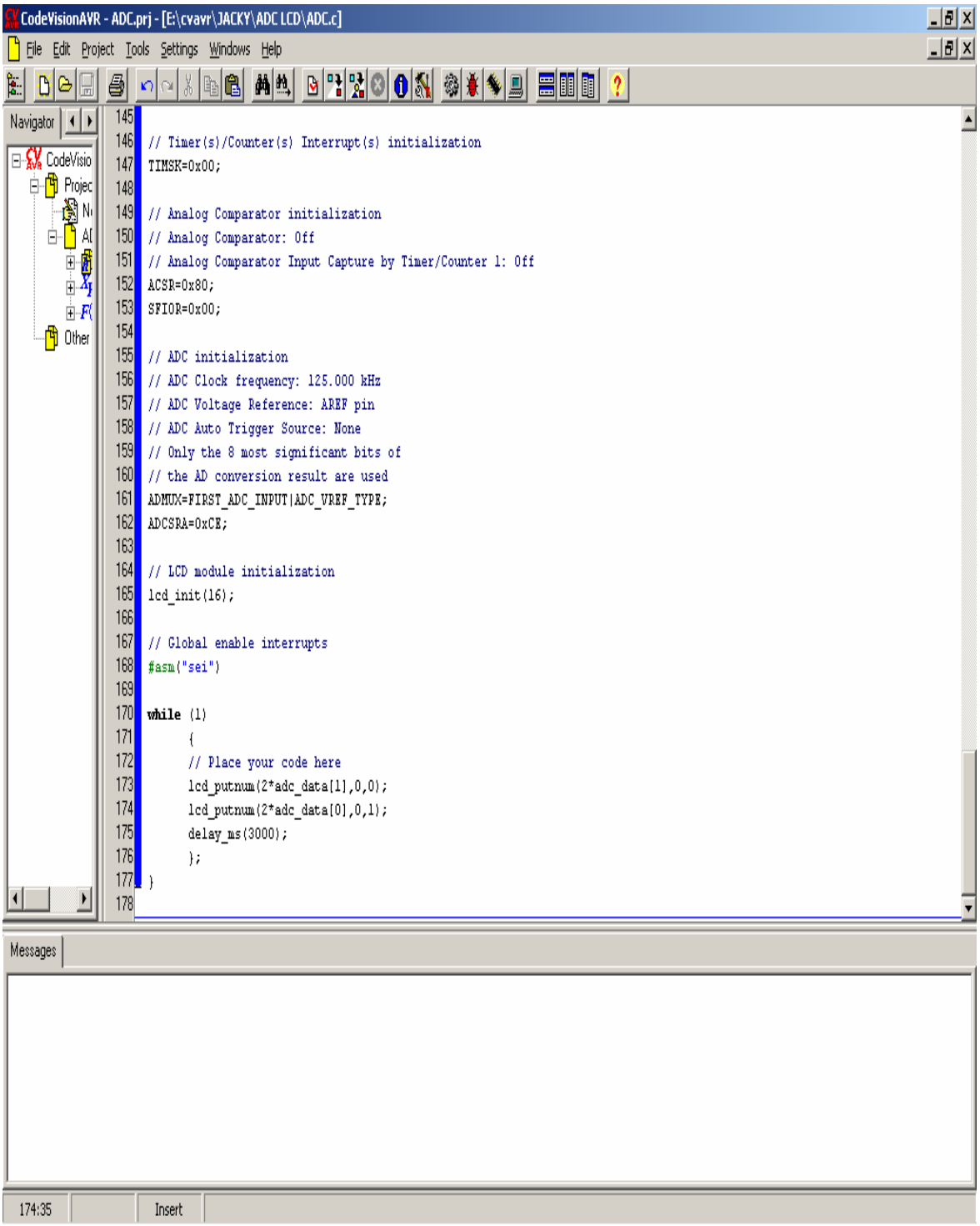
Ta check vào Use 8 bít, để ADC trả về giá trị 8 bít, và ta ADC dùng ngắt check vào Interrupt, về điện áp tham khảo AREF thì lấy điện áp của chân AREF của AVR được nối với 5V. Tần số ADC tùy các bạn thích nhanh hoặc chậm chọn giá trị phù hợp. Trong box Automatically Scan Inputs các bạn check vào Enabled. Vì chúng ta cần ADC 2 kênh, 1 kênh dùng biến trở để test ADC, một kênh từ LM35 đấu với 2 bit 0 và 1 của PORTA do đó chọn First 0, Last 1.
Khởi tạo cho LCD vào PORTB như hình bên cạnh.
Chọn Generate, Save and Exit.

Đê hiển thị được một số bất kỳ lên LCD, trong thư viện hàm không có và ta phải tự viết hàm . Đầu vào là một biến unsigned char, ta phải tách lấy hàng trăm, hàng chục, hàng đơn vị và đưa lần lượt lên LCD.
Code như sau:

```
void lcd_putnum(unsigned char so,unsigned char x,unsigned char y)
{
      unsigned char a,b,c;
      a=so/100;
         // lay fan tram
      b=(so-100*a)/10;
         // lay fan chuc
      c=(so-100*a-10*b);
         // lay hang don vi
      lcd_gotoxy(x,y);
         // ve vi tri x,y
      lcd_putchar(a+48);
         // day ra hang tram, ma ascii
      lcd_putchar(b+48);
         // day ra hang chuc, ma ascii
      lcd_putchar(c+48);
         // day ra hang don vi, ma ascii
}
```

Trong vòng while(1) trong hàm main ta viết như sau:

```
while (1)
    {
          // Place your code here
    lcd_putnum(2*adc_data[1],0,0);
         // dua gia tri ADC tu LM35*2= nhiet do
    lcd_putnum(adc_data[0],0,1);
         // dua gia tri ADC tu bien tro
    delay_ms(3000);
         // tre 3 s, cap nhat du lieu mot lan
    };
```


CodeVisionAVR - ADC.prj - [E:\cvavr\JACKY\ADC LCD\ADC.c]
File Edit Project Tools Settings Windows Help
Navigator
CodeVisio
Projec
Other
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;
// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;
// ADC initialization
// ADC Clock frequency: 125.000 kHz
// ADC Voltage Reference: AREF pin
// ADC Auto Trigger Source: None
// Only the 8 most significant bits of
// the AD conversion result are used
ADMUX=FIRST_ADC_INPUT|ADC_VREF_TYPE;
ADCSRA=0xCE;
// LCD module initialization
lcd_init(16);
// Global enable interrupts
#asm("sei")
while (1)
{
// Place your code here
lcd_putnum(2*adc_data[1],0,0);
lcd_putnum(2*adc_data[0],0,1);
delay_ms(3000);
};
}
Messages
174:35
Insert

Đo nhiệt độ bằng LM35 qua ADC thường có sai số và độ trôi, do đó ta cần hiệu chỉnh nhiệt độ bằng cách so sánh với nhiệt kế.

Bài 5: Giao tiếp I2C với RTC DS1307

Bài 5: Giao tiếp I2C với RTC DS1307

1.Yêu cầu:

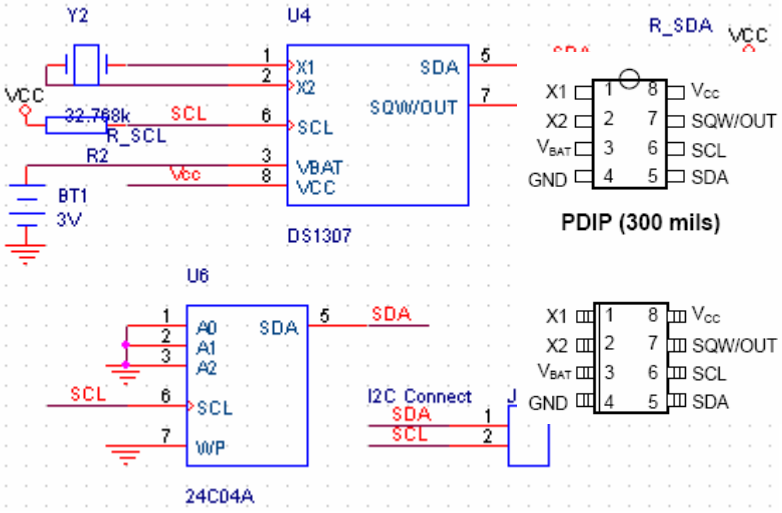
Hiểu được giao tiếp I2C.

Nguyên lí hoạt động của DS1307.

Đọc thời gian và ngày tháng từ DS1307 và hiển thị lên LCD.

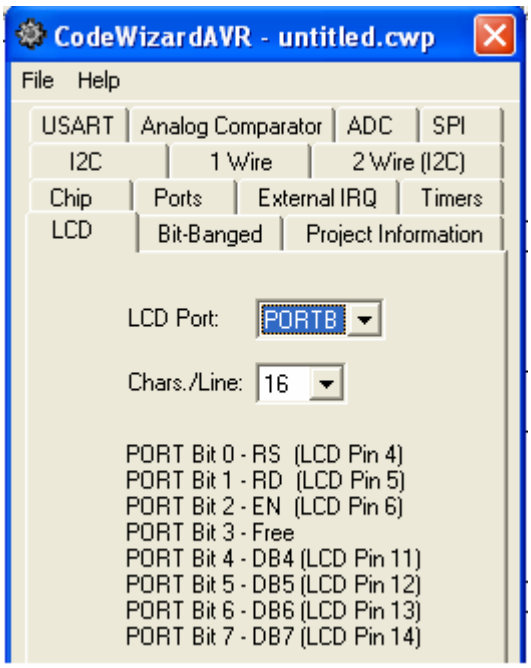
2.Mô tả:

Bus của I2C từ DS1307 và 24Cxx được nối với một jumper giúp ta có thể nối với bất kỳ 2 bít của hai cổng bất kỳ của AVR trên KIT bởi một dây nối.

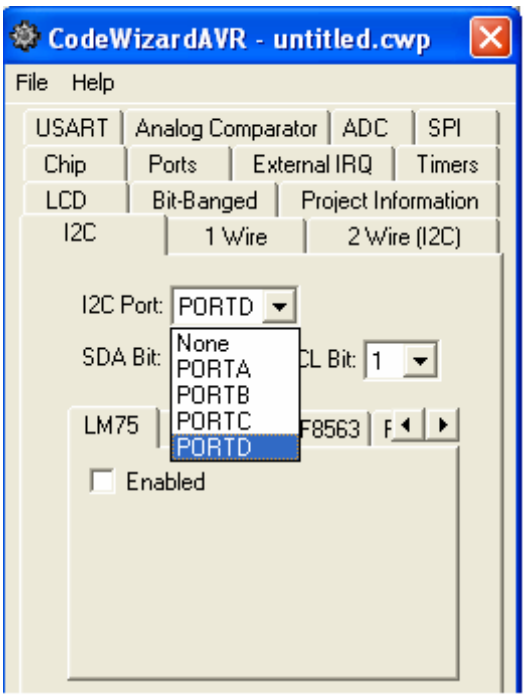


3.Thực hành:

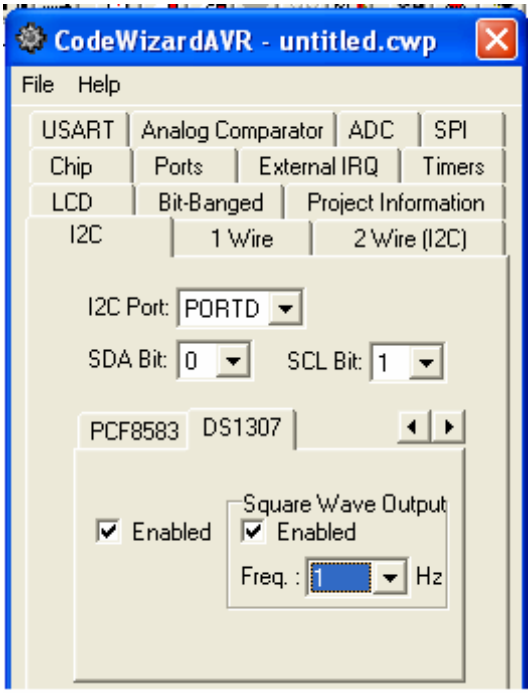
Khởi tạo cho LCD và DS1307 như sau:



Khởi tạo cho LCD- PORT B



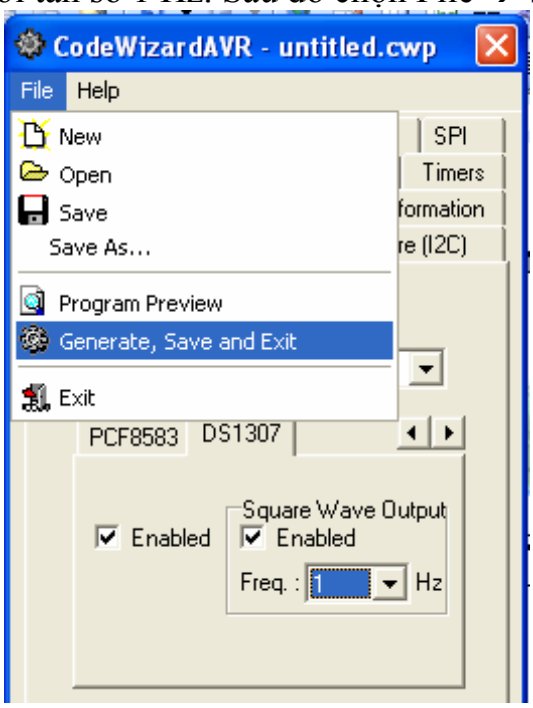
Khởi tạo I2C



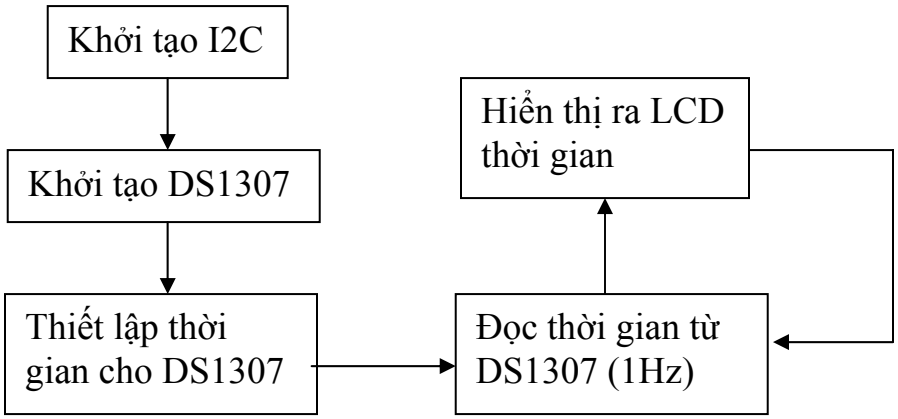
Khởi tạo DS1307

Trong tab các chip ta chọn chíp DS1307, check vào Enabled để xác định sử dụng DS1307 và trong ô Square Wave Output ta check vào ô Enabled, trong

list Freq: Chọn 1 Hz để khởi tạo cho chân output của DS1307 cứ 1 s có một xung ra, trong mạch chân đó nối với 1 led và khởi tạo vừa rồi làm cho led đó nhấp nháy với tần số 1 Hz. Sau đó chọn File → Save, Generate and Exit.



Được cửa sổ soạn thảo code.
Sơ đồ làm việc với DS1307 như sau:



Coding như sau:
Bổ xung thư viện delay.h vào đầu chương trình.

```
16
17  Chip type               : ATmega16L
18  Program type            : Application
19  Clock frequency         : 8.000000 MHz
20  Memory model            : Small
21  External SRAM size      : 0
22  Data Stack size         : 256
23  *****************************************************/
24
25  #include <mega16.h>
26  #include <delay.h>
27  // I2C Bus functions
28  #asm
29      .equ __i2c_port=0x12 ;PORTD
30      .equ __sda_bit=0
31      .equ __scl_bit=1
32  #endasm
```

Ngay trước vòng while(1) trong hàm main bổ xung câu lệnh đặt thời gian và ngày tháng cho RTC. I2C, DS1307, LCD đã khởi tạo bằng CodeWinzard AVR.

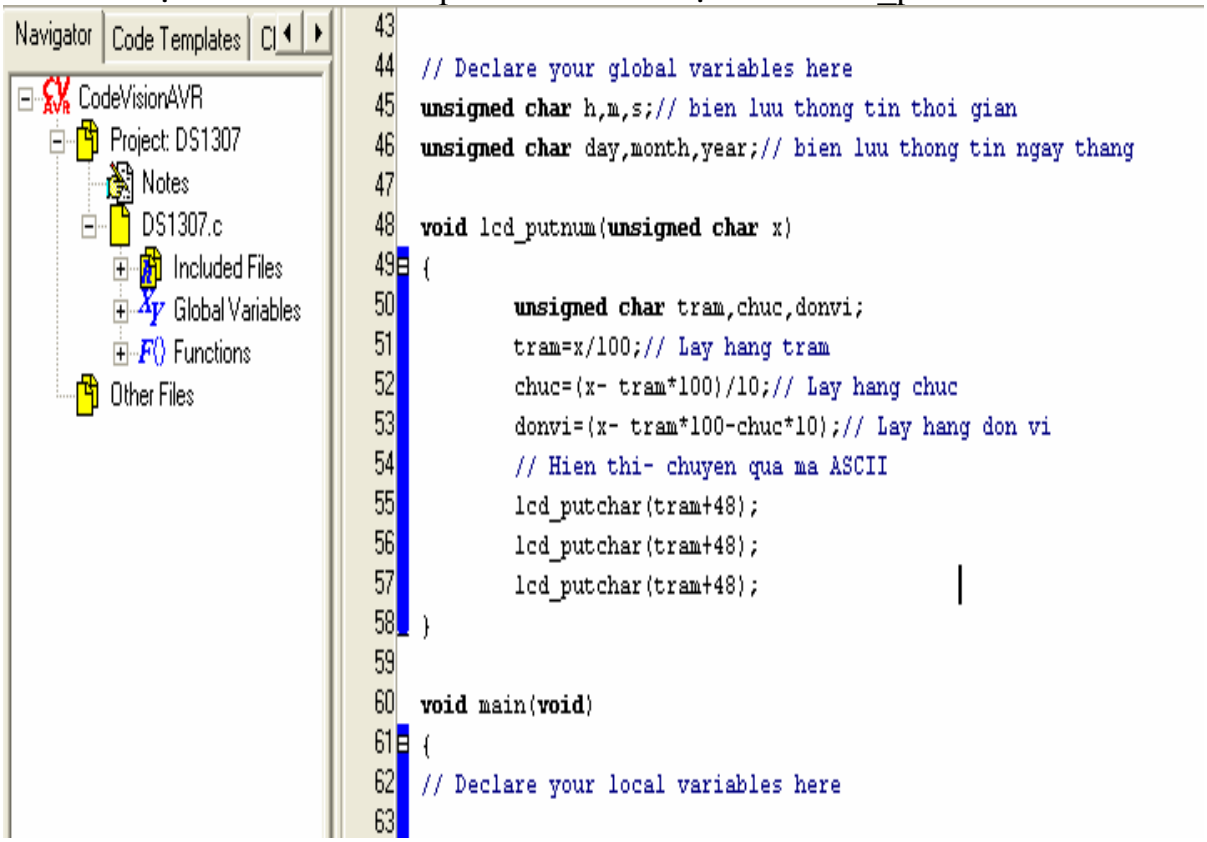
```
127 // Analog Comparator initialization
128 // Analog Comparator: Off
129 // Analog Comparator Input Capture by Timer/Counter 1: Of
130 ACSR=0x80;
131 SFIOR=0x00;
132
133 // I2C Bus initialization
134 i2c_init(); // Khoi tao I2C
135
136 // DS1307 Real Time Clock initialization
137 // Square wave output on pin SQW/OUT: On
138 // Square wave frequency: 1Hz
139 rtc_init(0,1,0); // Khoi tao RTC
140
141 // LCD module initialization
142 lcd_init(16);// Khoi tao LCD
143
144 rtc_set_time(10,10,10); // Khoi tao thoi gian
145 rtc_set_date(1,1,2007); // Khoi tao ngay thang
146
147 while (1)
148       {
149       // Place your code here
```

Để có thể đọc được thời gian ta dùng hàm rtc_get_time() và rtc_get_date có sẵn trong thư viện DS1307.h.(Để tham khảo các hàm có thể mở Help tương tự như tham khảo các hàm của LCD ở bài trước.)

Chúng ta phải khai báo 3 biến để lưu thông tin về thời gian là giờ h; phút m; giây s và 3 biến lưu thông tin về ngày tháng là ngày day; tháng month; năm year ngay phía trước hàm main như sau:

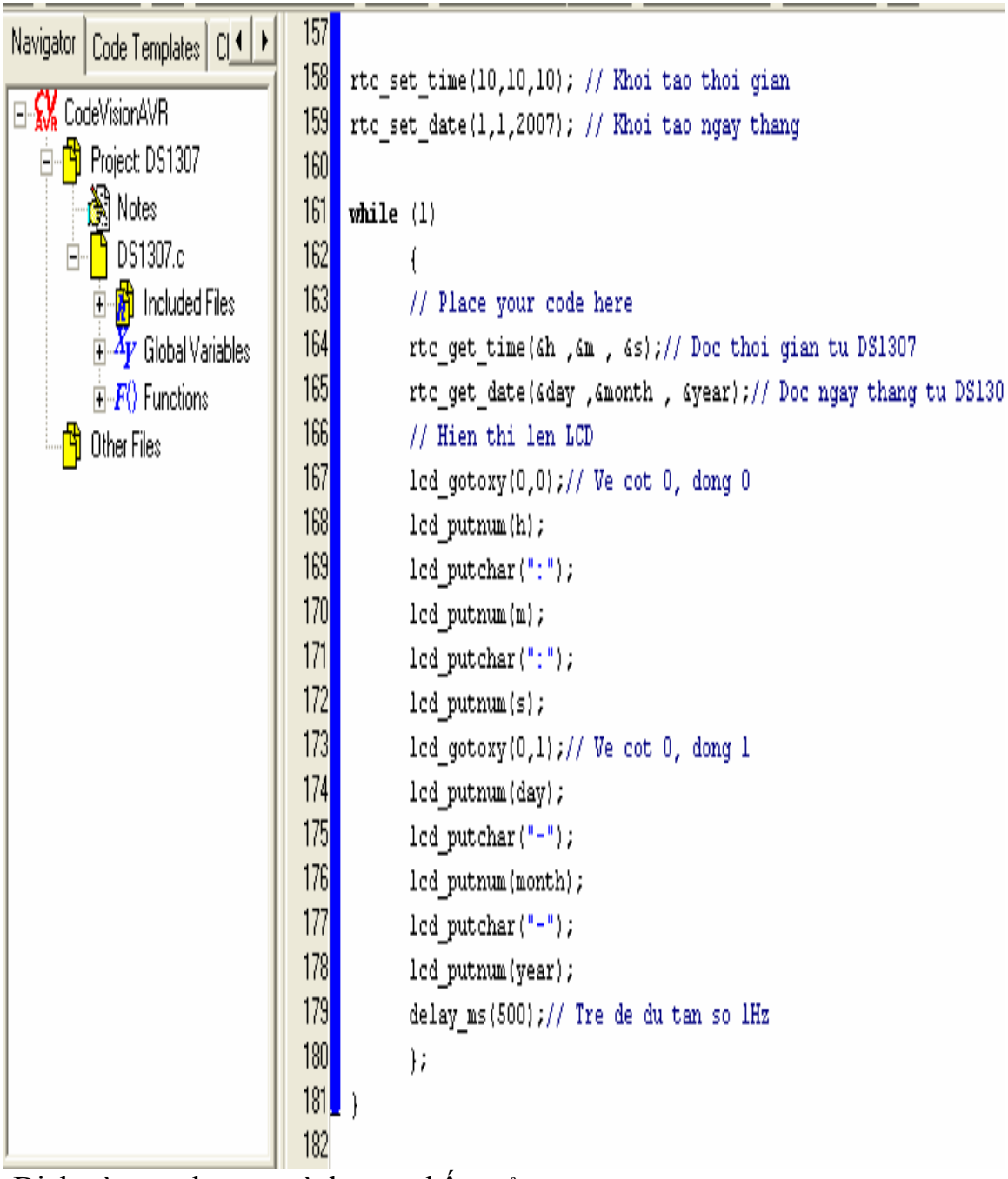
```
39  #asm
40     .equ __lcd_port=0x18 ;PORTB
41  #endasm
42  #include <lcd.h>
43
44  // Declare your global variables here
45  unsigned char h,m,s;// bien luu thong tin thoi gian
46  unsigned char day,month,year;// bien luu thong tin ngay thang
47  void main(void)
48  {
49  // Declare your local variables here
50
51  // Input/Output Ports initialization
```

Để hiển thị các số ra LCD ta phải viết thêm một hàm LCD_putnum như sau:




```
43
44  // Declare your global variables here
45  unsigned char h,m,s;// bien luu thong tin thoi gian
46  unsigned char day,month,year;// bien luu thong tin ngay thang
47
48  void lcd_putnum(unsigned char x)
49  {
50          unsigned char tram,chuc,donvi;
51          tram=x/100;// Lay hang tram
52          chuc=(x- tram*100)/10;// Lay hang chuc
53          donvi=(x- tram*100-chuc*10);// Lay hang don vi
54          // Hien thi- chuyen qua ma ASCII
55          lcd_putchar(tram+48);
56          lcd_putchar(tram+48);
57          lcd_putchar(tram+48);
58  }
59
60  void main(void)
61  {
62  // Declare your local variables here
63
```

Chương trình chính trong vòng while(1) như sau:




```
157
158 rtc_set_time(10,10,10); // Khoi tao thoi gian
159 rtc_set_date(1,1,2007); // Khoi tao ngay thang
160
161 while (1)
162       {
163       // Place your code here
164       rtc_get_time(&h ,&m , &s);// Doc thoi gian tu DS1307
165       rtc_get_date(&day ,&month , &year);// Doc ngay thang tu DS130
166       // Hien thi len LCD
167       lcd_gotoxy(0,0);// Ve cot 0, dong 0
168       lcd_putnum(h);
169       lcd_putchar(":");
170       lcd_putnum(m);
171       lcd_putchar(":");
172       lcd_putnum(s);
173       lcd_gotoxy(0,1);// Ve cot 0, dong 1
174       lcd_putnum(day);
175       lcd_putchar("-");
176       lcd_putnum(month);
177       lcd_putchar("-");
178       lcd_putnum(year);
179       delay_ms(500);// Tre de du tan so 1Hz
180       };
181 }
182
```

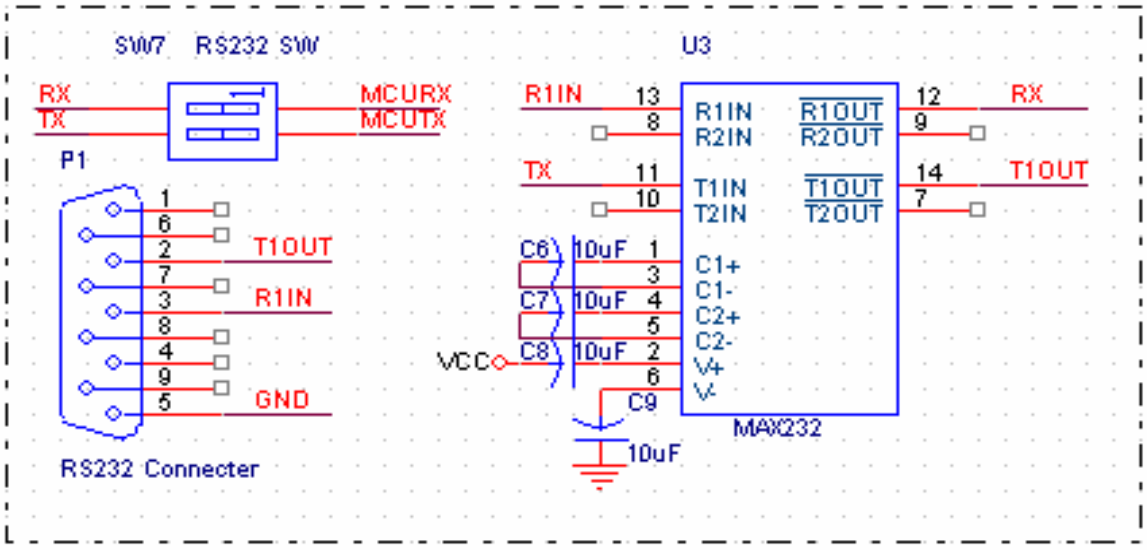
Dịch và nạp chương trình, xem kết quả.

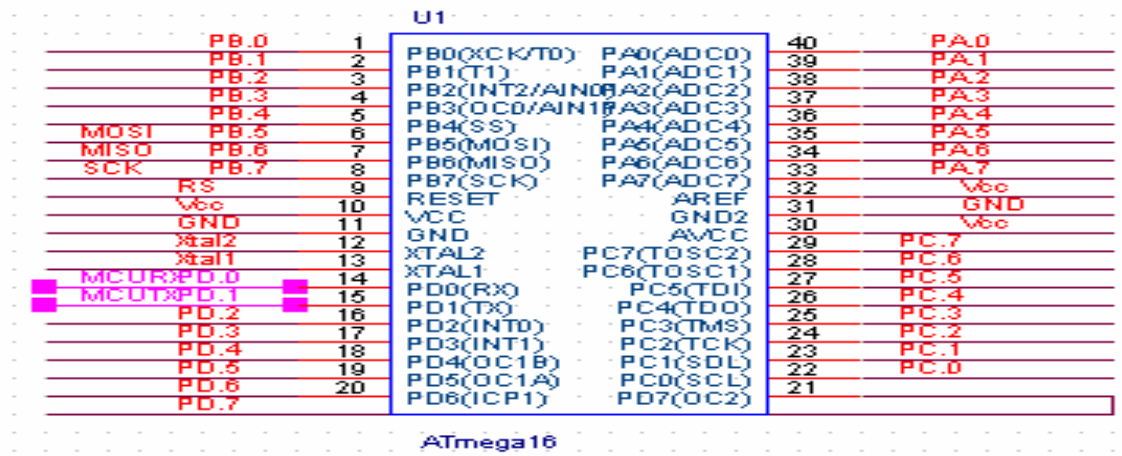
Bài 6: Truyền thông nối tiếp RS232 và Visual Basic.

1.Yêu cầu:

- Biết khởi tạo RS232 trong CodeWinzard AVR.
- Viết chương trình nhận dữ liệu từ cổng COM PC và truyền lên cổng COM đúng dữ liệu đó.
- Các thuộc tính và các control trong Visual Basic 6.0.
- Tự tạo một Project trong Visual Basic 6.0 truyền dữ liệu xuống cổng COM và đọc dữ liệu từ cổng COM lên.

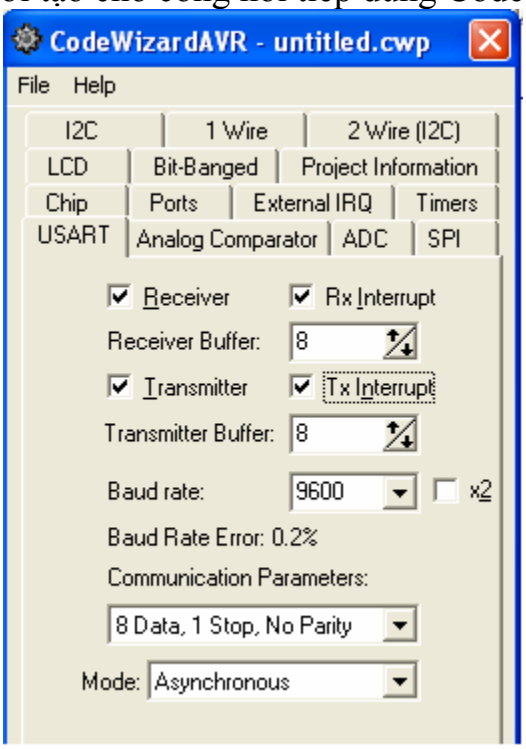
2.Mô tả: Cổng nối tiếp trên KIT.





3.Thực hành:

Các bước khởi tạo cho cổng nối tiếp dùng CodeWinzard như sau:

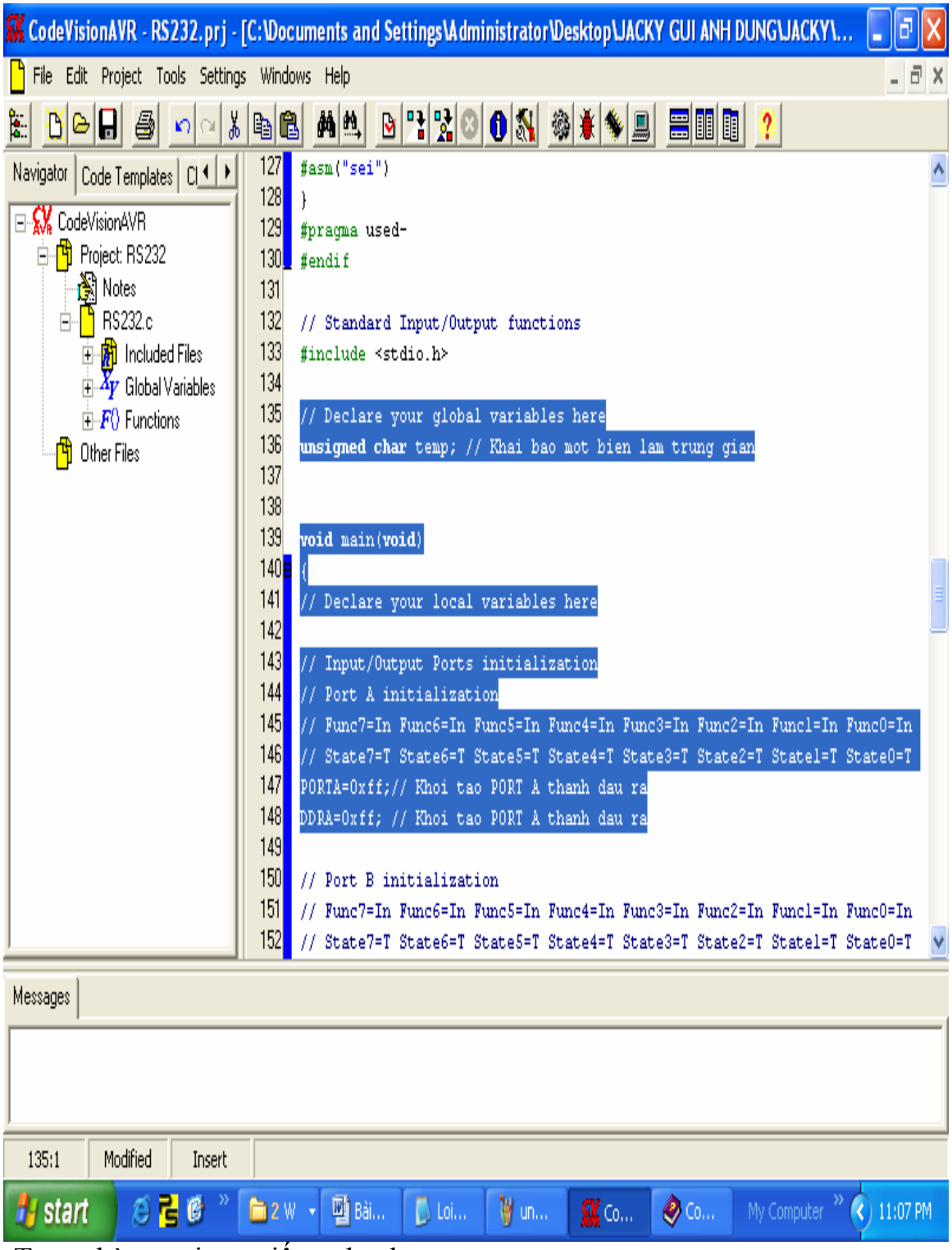


Khởi tạo RS232

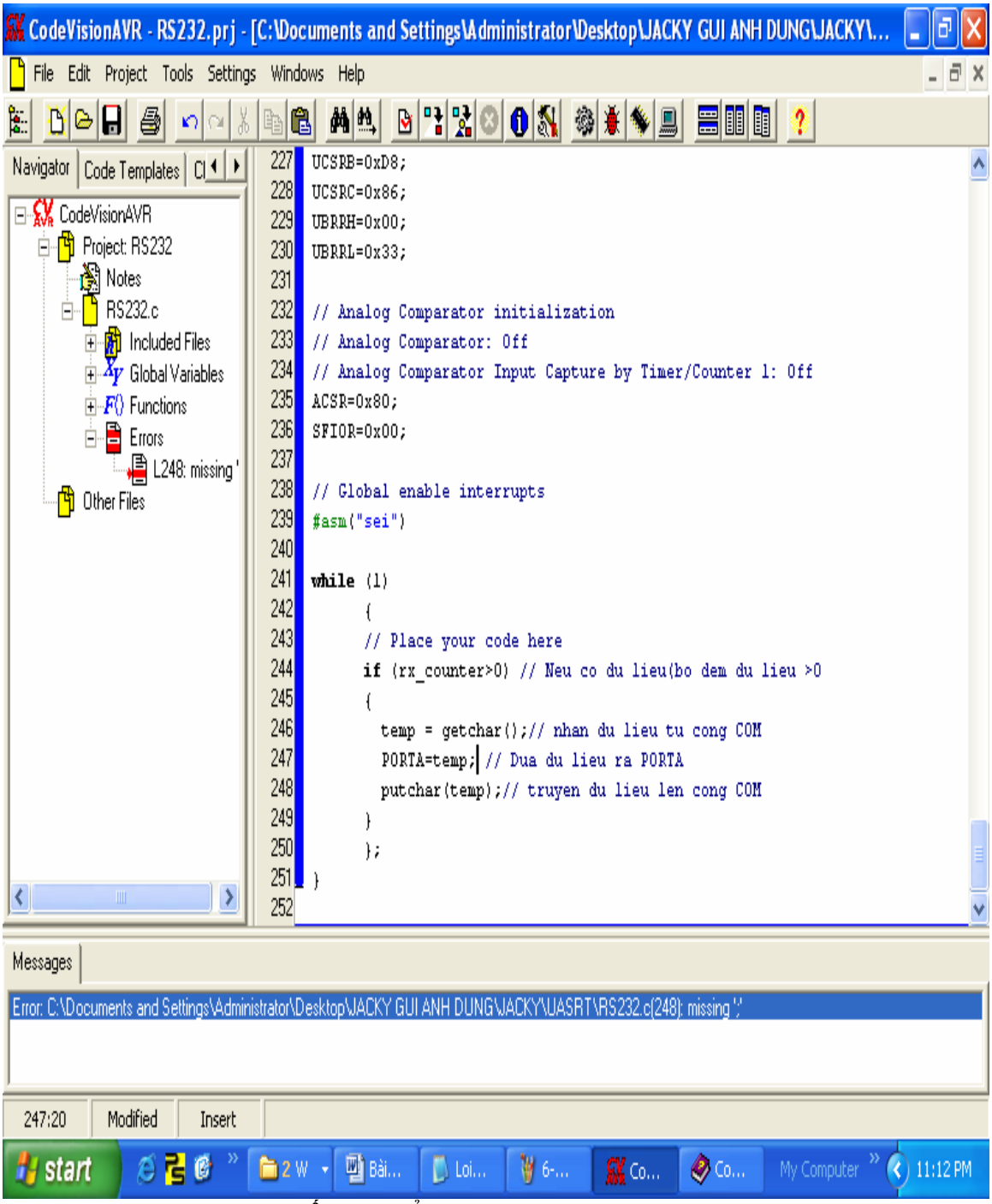
Trong tab USART check vào các ô Receiver để cho phép nhận dữ liệu; Rx Interrupt để nhận dữ liệu sử dụng ngắt; Transmitter để cho phép truyền dữ liệu; Tx Interrupt để truyền dữ liệu sử dụng ngắt.

Các thông số còn lại: Receiver Buffer và Transmitter Buffer là bộ nhớ đệm nhận và đệm truyền. Trong ứng dụng đơn giản chúng ta để mặc định là 8, trong các ứng dụng truyền số lượng thông tin lớn ta có thể tăng bộ đệm để tránh mất thông tin. Tốc độ baud mặc định là 9600 (bit/s). Các thông số của bộ truyền: 8 bit, 1 bit dừng(stop), không ưu tiên. Chế độ truyền không đồng bộ.

Theo yêu cầu là nhận dữ liệu và truyền lên dữ liệu đó ta viết code như sau. Trước tiên ta khai báo một biến trung gian để truyền nhận dữ liệu và khởi tạo cho PORTA là đầu ra như sau:



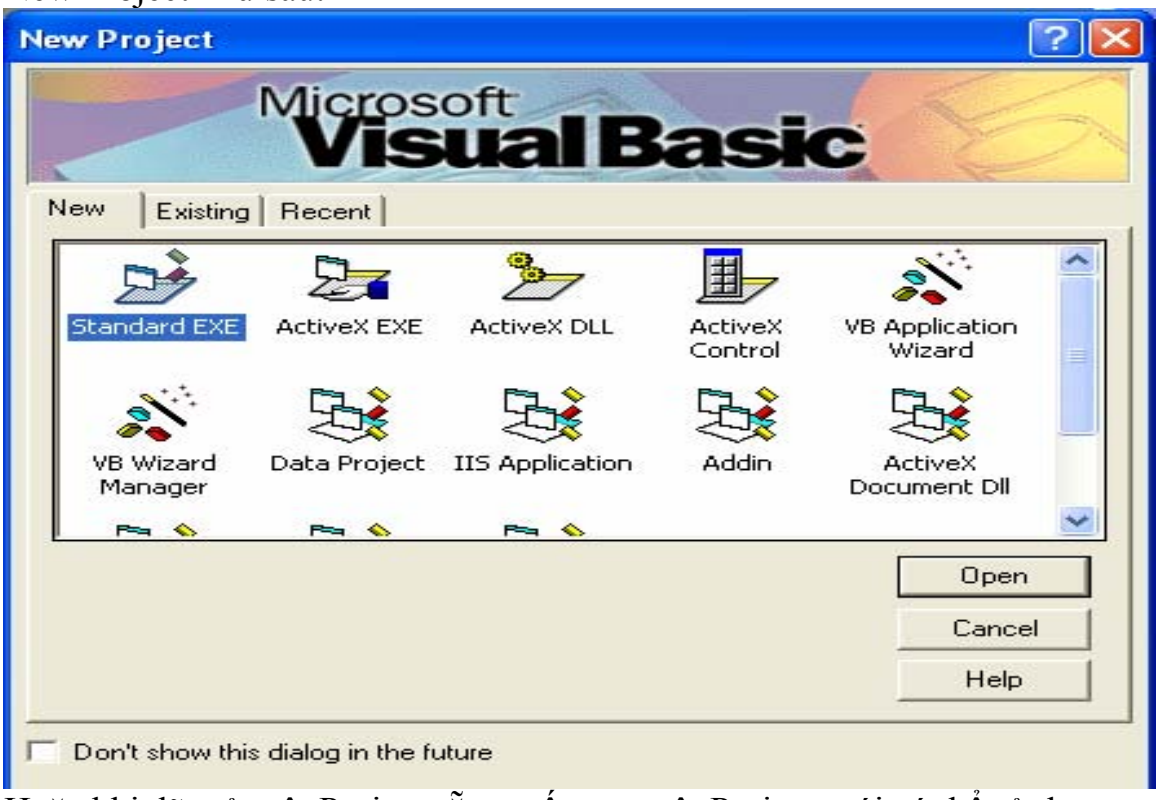
Trong hàm main ta viết code như sau:



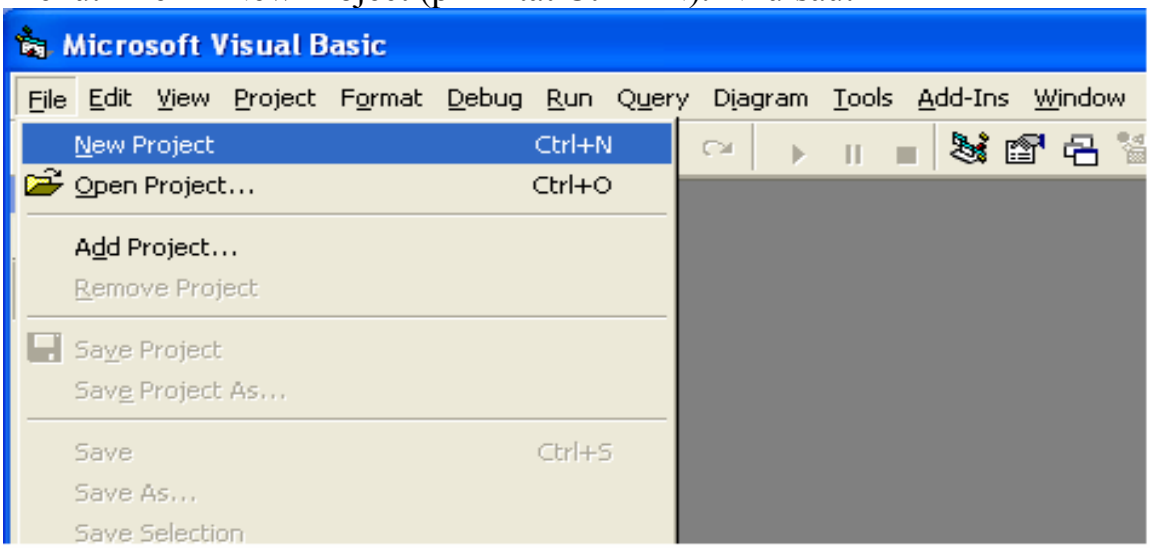
Chọn File → Save All. Ấn F9 để dịch chương trình. Nạp chương trình vào AVR.

4.Visual Basic và các control đơn giản.

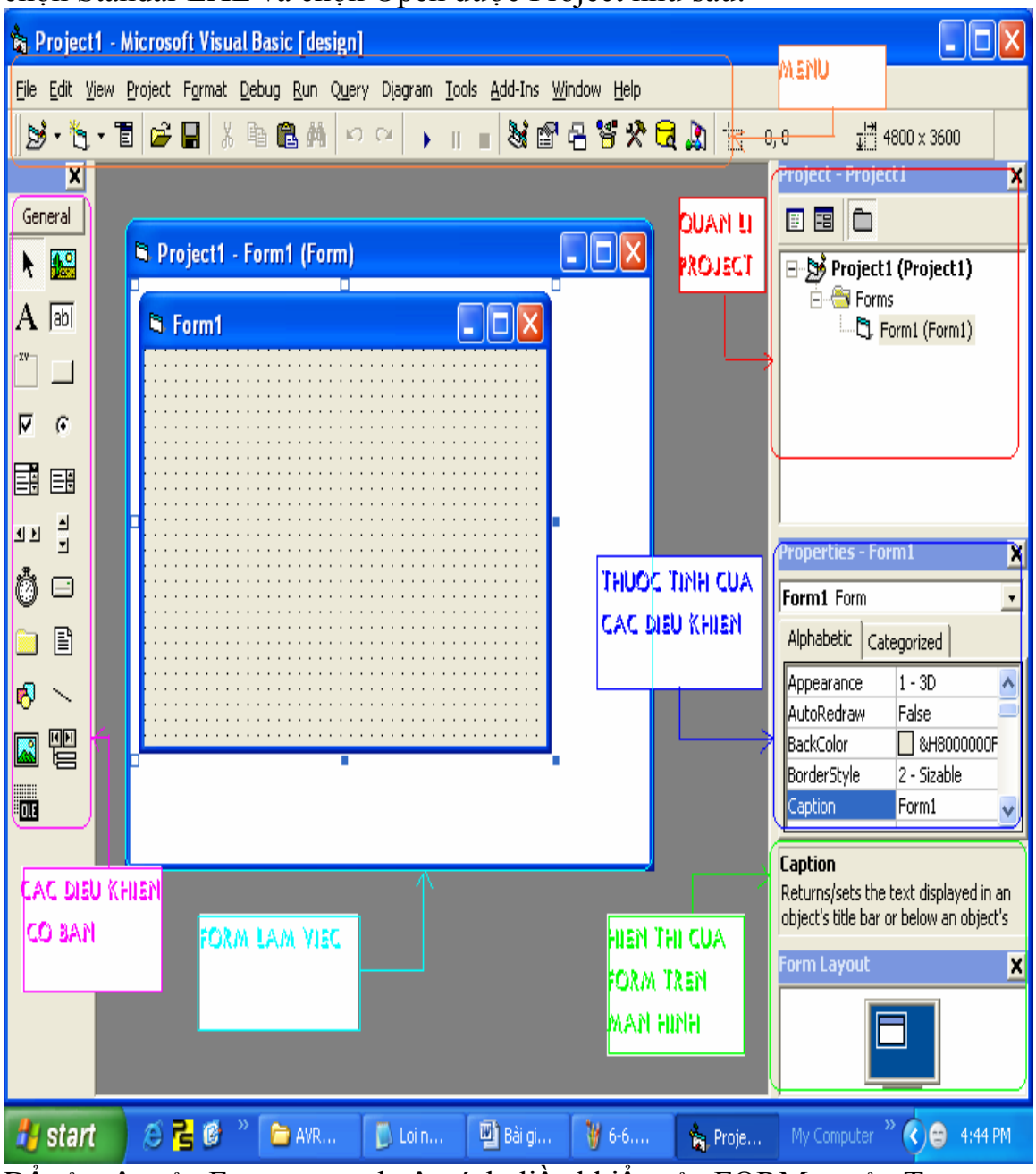
Khởi tạo Project trong VB. Kích đúp và biểu ICON của VB được cửa sổ New Project như sau:



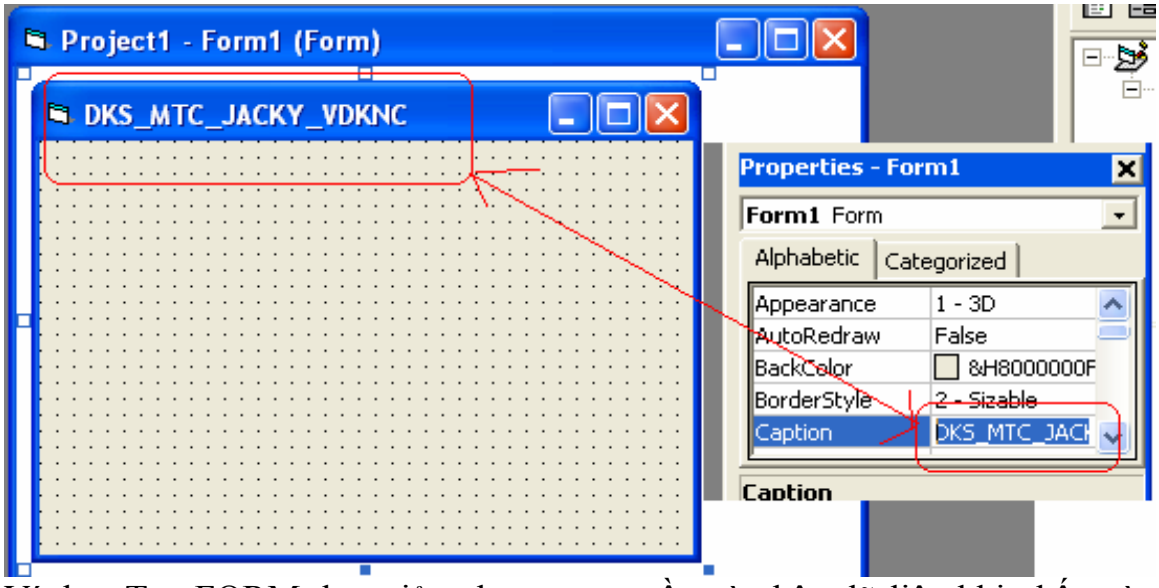
Hoặc khi đã mở một Project sẵn muốn tạo một Project mới có thể sử dụng Menu: File → New Project (phím tắt Ctrl + N). Như sau:



Trong cửa sổ New Project có 3 tab: New để tạo Project mới; Existing để mở một Project có sẵn; Recent: để mở các Project gần đây. Trong tab new có nhiều loại Project : Standar Exe, ActiveX exe, ActiveX DLL, … . Chúng ta chọn Standar EXE và chọn Open được Project như sau:

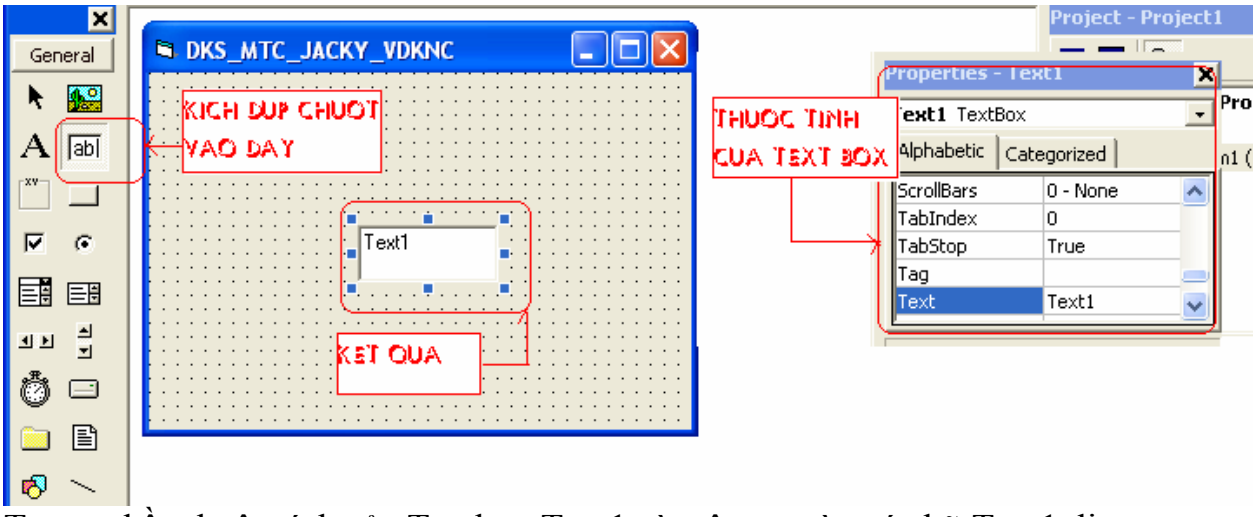


Để sửa tên của Form trong thuộc tính điều khiển của FORM ta sửa Text trong ô Caption như sau:

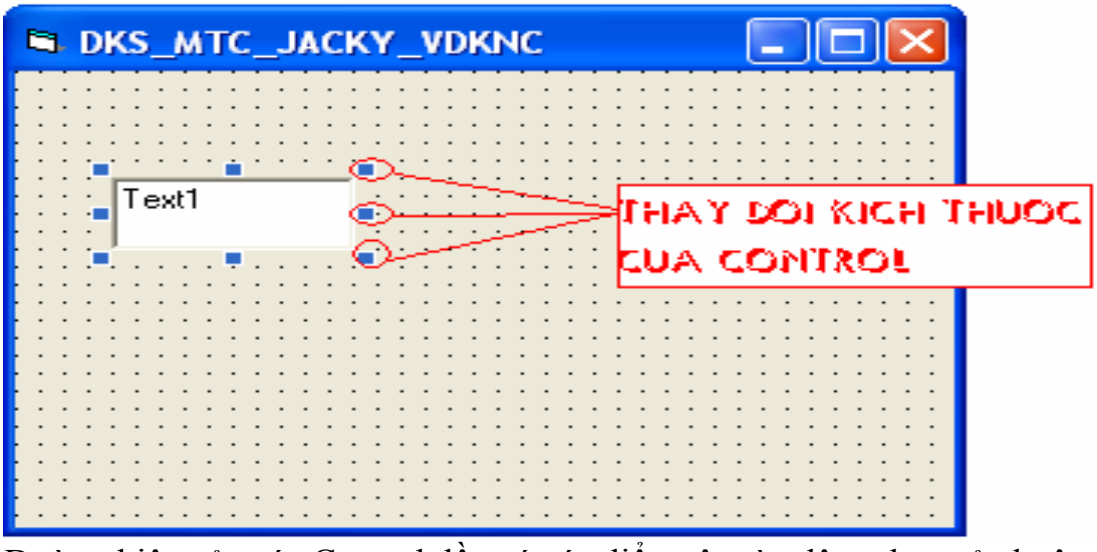


Ví dụ : Tạo FORM đơn giản như sau: truyền và nhận dữ liệu khi nhấp vào một nút. Đầu vào sẽ có 1 tham số a để truyền, đầu ra có 1 thông số- nhận dữ liệu- như vậy ta sẽ dùng 2 textbox control, ngoài ra ta cần sử dụng 3 nút bấm button để xác định sự kiện truyền, nhận và thoát.

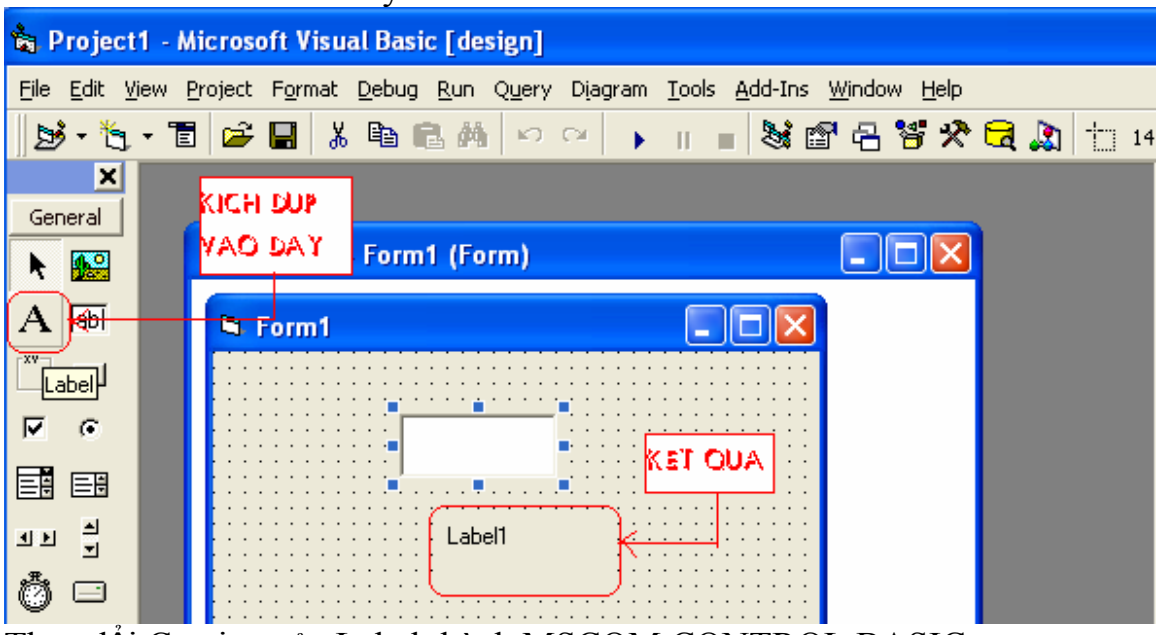
Để có thể đưa một control vào trong FORM, ở phần CAC DIEU KHIEN CO BAN ta chỉ cần nhấp đúp vào các control mới dùng. Ví dụ lấy textbox control.



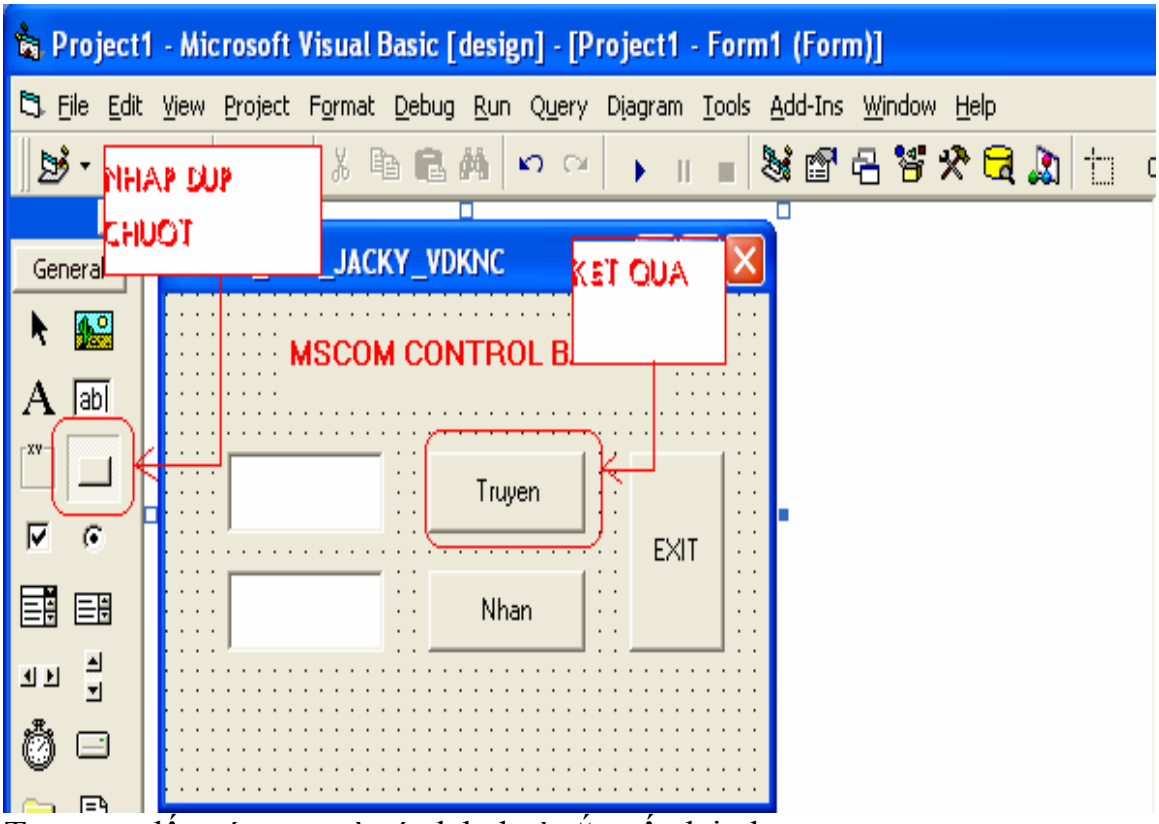
Trong phần thuộc tính của Textbox Text1, tìm ô text và xoá chữ Text1 đi. Để ô Text 1 thành trắng, để di chuyển các control ta nhấp trái chuột và dữ chặt và di chuyển tới vị trí thích hợp.



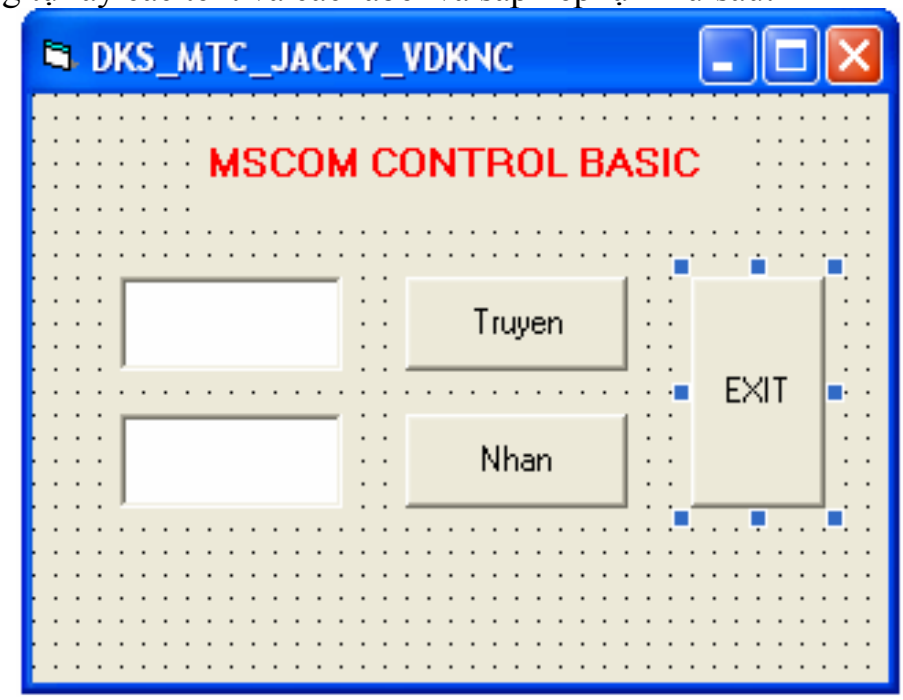
Đường biên của các Control đều có các điểm tô màu đậm, đưa trỏ chuột tới đó trỏ chuột biến thành mũi tên, nhấp trái chuột và dữ chặt để thay đổi kích thước của các control. Lấy LABEL như sau:



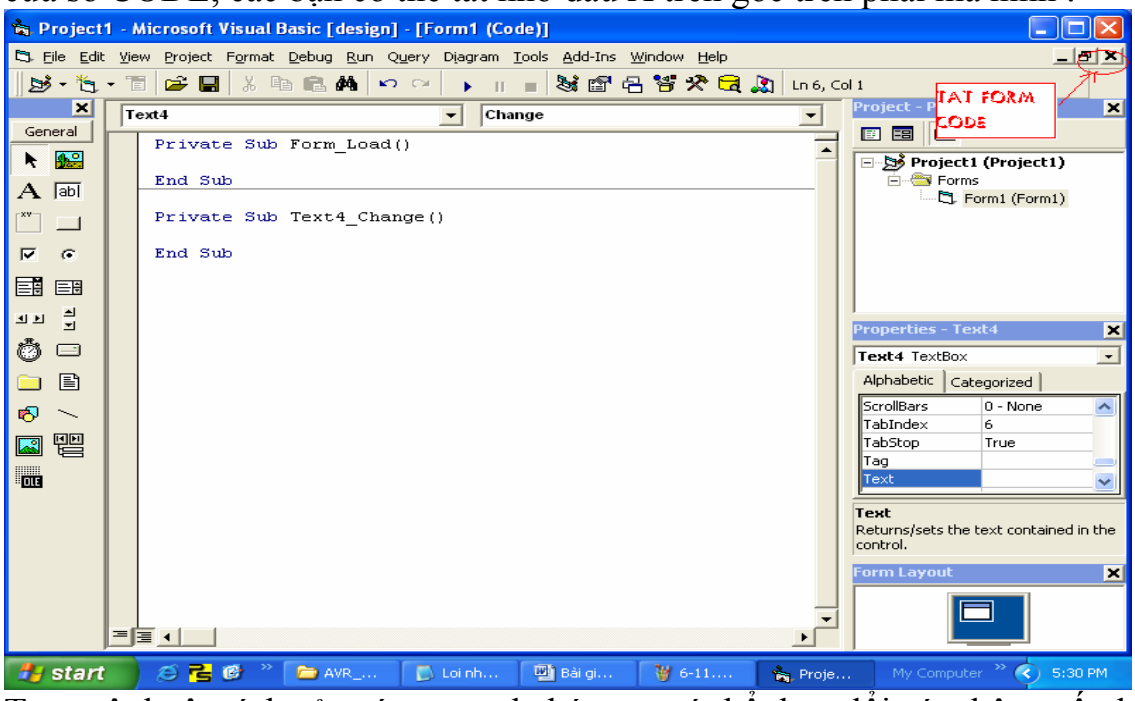
Thay đổi Caption của Label thành MSCOM CONTROL BASIC .
Lấy các button và sửa các thuộc tính tương tự như sau:



Tương tự lấy các text và các label và sắp xếp lại như sau:



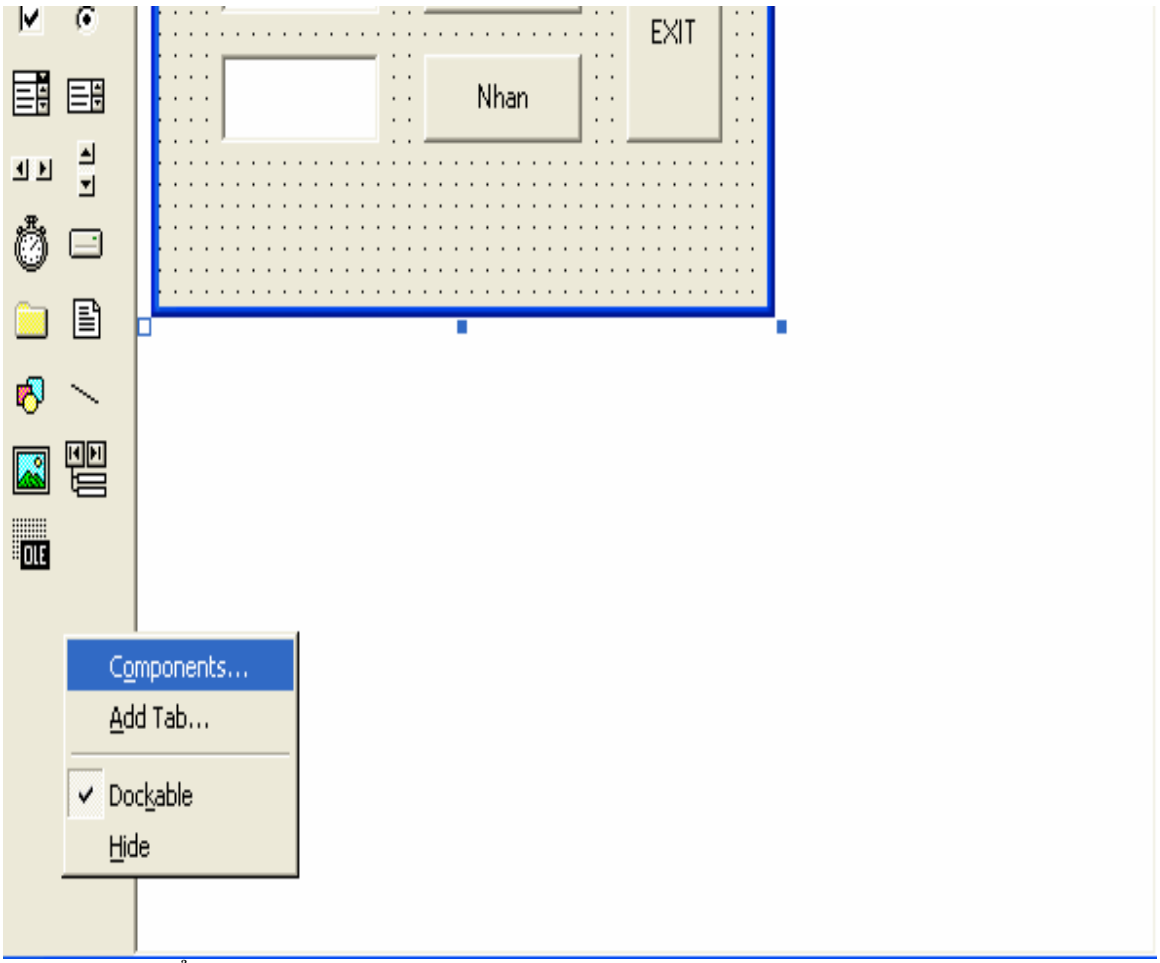
Trong trường hợp các bạn kích đúp chuột vào một điều khiển nó sẽ hiện ra cửa sổ CODE, các bạn có thể tắt nhờ dấu X trên góc trên phải mà hình :



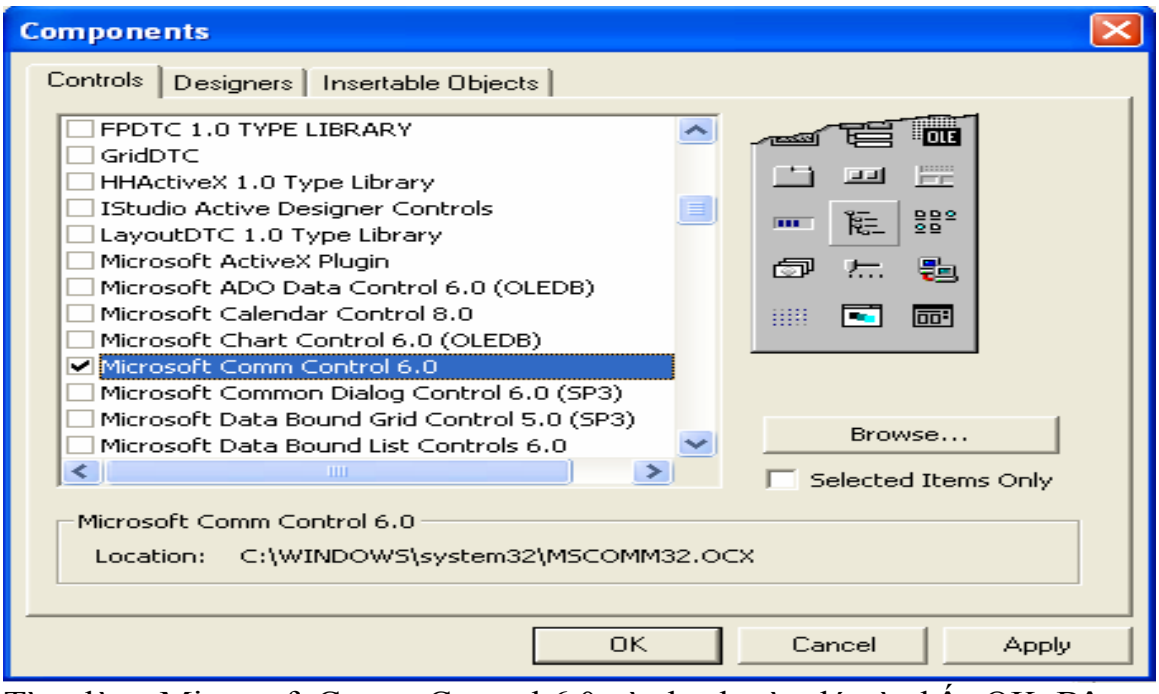
Trong ô thuộc tính của các control chúng ta có thể thay đổi các thông số như tên của các control ví dụ: Name, Font chữ hiển thị, mầu sắc chữ, mầu nền, v.v.Như vậy ta đã tạo ra một FORM các tham số a,b hiện thị bởi các textbox1,2. Nút truyền là Command1, nút nhận là thoát là Command2, nút thoát là Command3.

Form chạy như sau: Nhập thông số vào các text 1, nhấn nút Truyền thì dữ liệu trong text1 được truyền ra cổng COM. Nhấn nút nhận thì dữ liệu nhận được sẽ hiển thị lên text 2. Phím thoát để thoát khỏi chương trình.

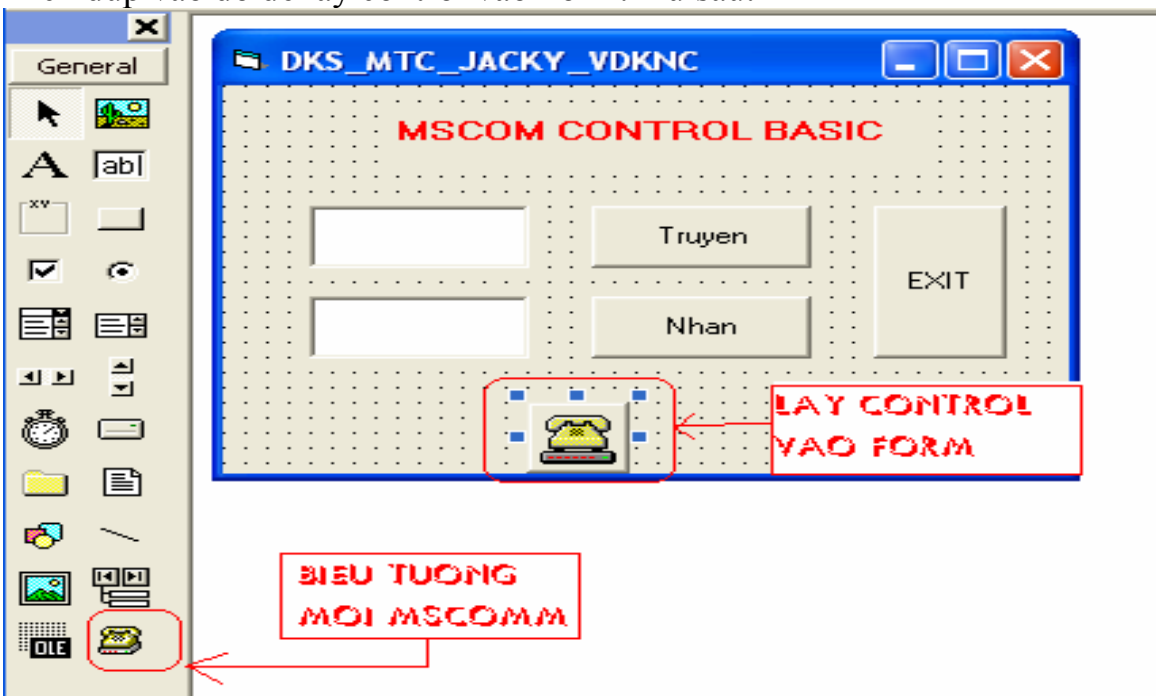
Vì Control để điều khiển cổng COM – MSCOM không phải control cơ bản nên nó không hiển thị trên tools, chúng ta phải lấy trong thư viện ra. Như sau: kích chuột phải vào thanh các control đơn giản chọn Component… .



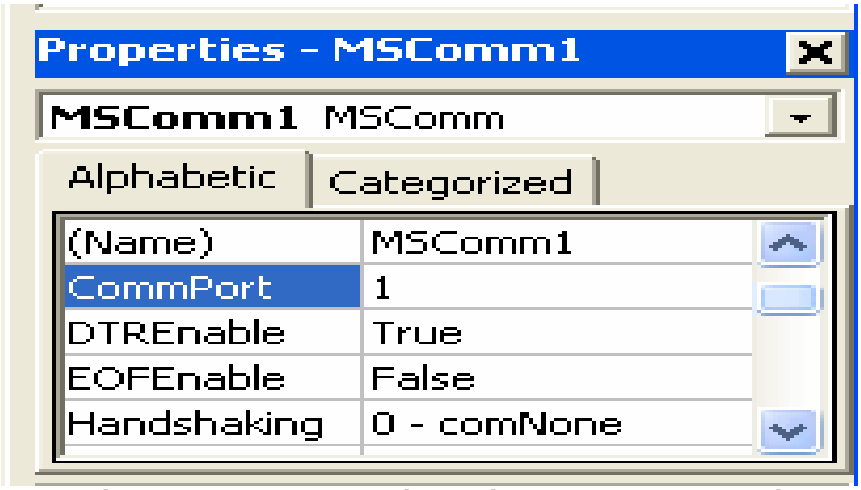
Được cửa sổ Components như sau:



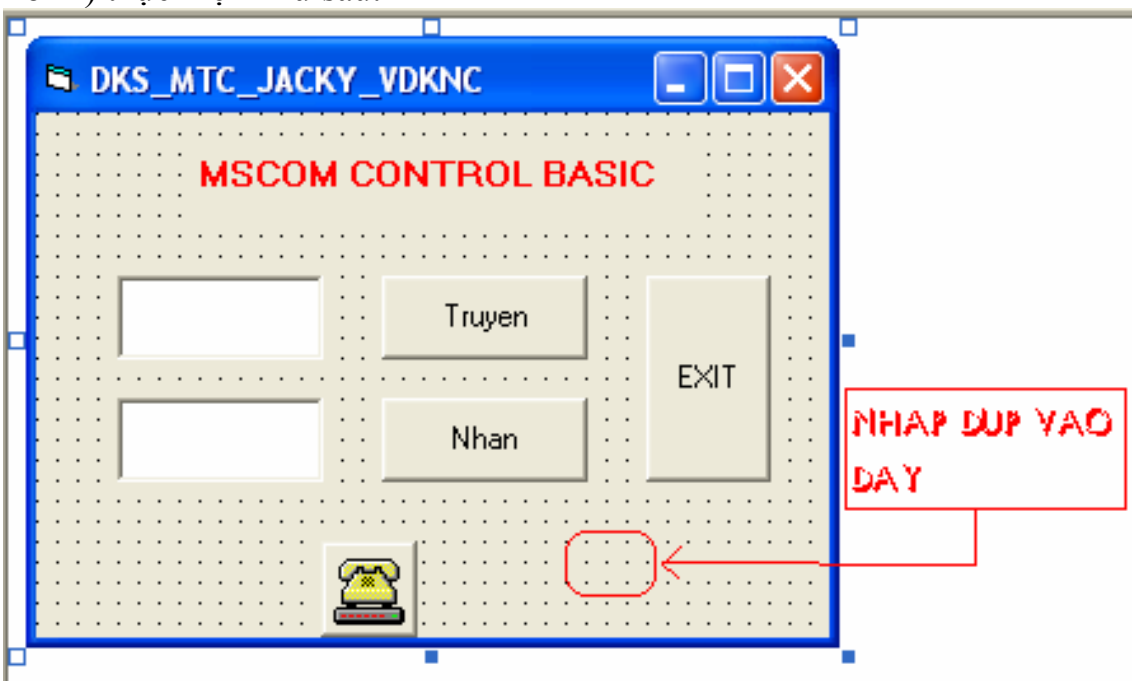
Tìm dòng Microsoft Comm Control 6.0 và check vào đó và nhấn OK. Bây giờ trên thanh công cụ có thêm một biểu tượng mới là MSCOMM control. Kích đúp vào đó để lấy control vào Form.như sau:



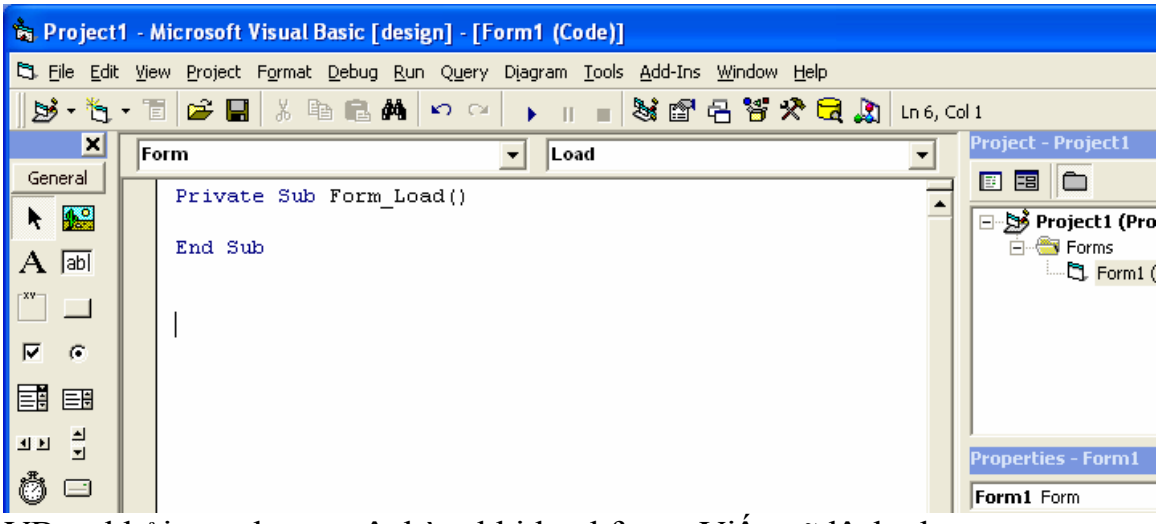
Thuộc tính mặc định cho MSCOMM như sau:



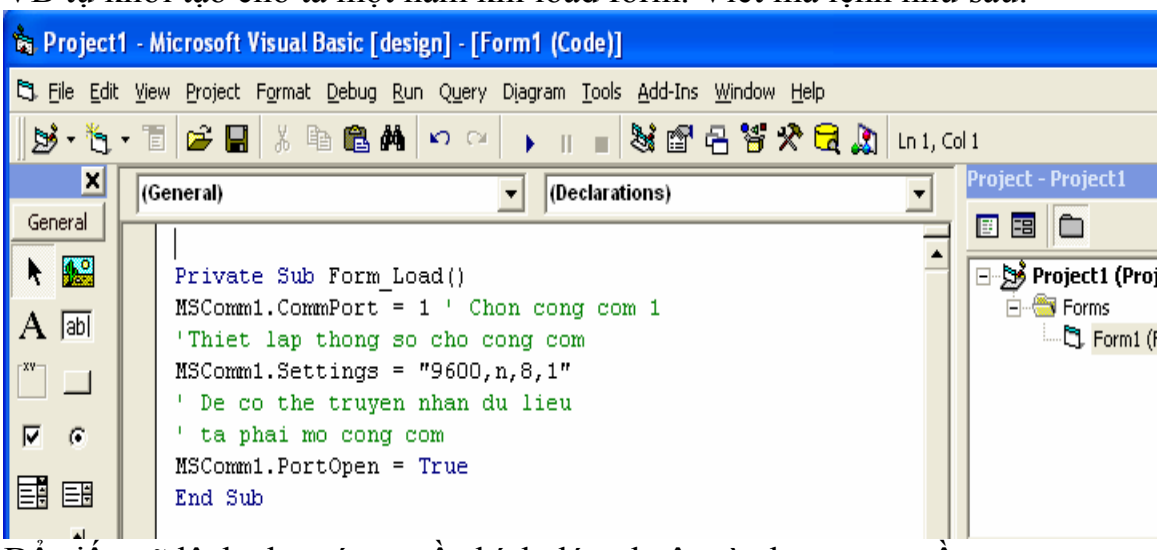
Để viết Code cho đối tượng nào ta chỉ cần nhấp đúp chuột vào đối tượng đó cửa sổ viết code sẽ hiện ra. Khi chạy chương trình thì trước hết ta cần khởi tạo cho control MSCOMM. Như vậy ta phải khởi tạo trong hàm Form_Load. Ta chuyển trỏ chuột để nó đánh dấu Form (Nhấp đúp vào khoảng trống trên Form) thực hiện như sau:



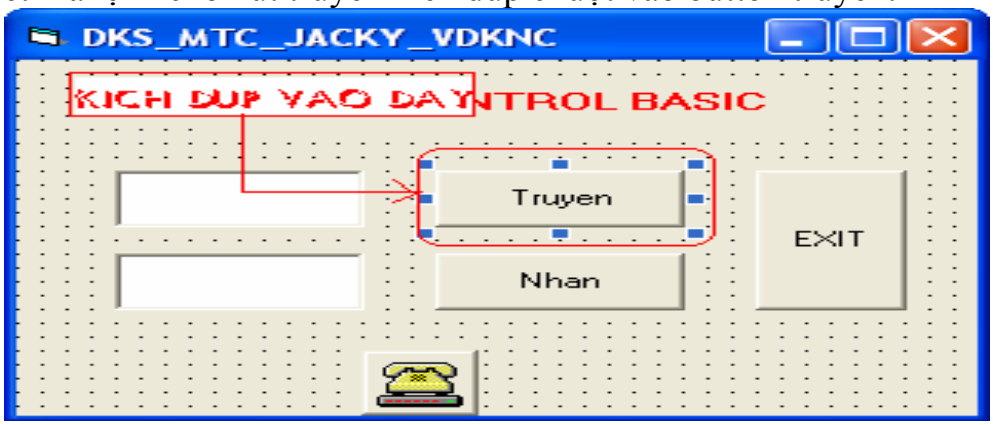
Ta được cửa sổ soạn code như sau:



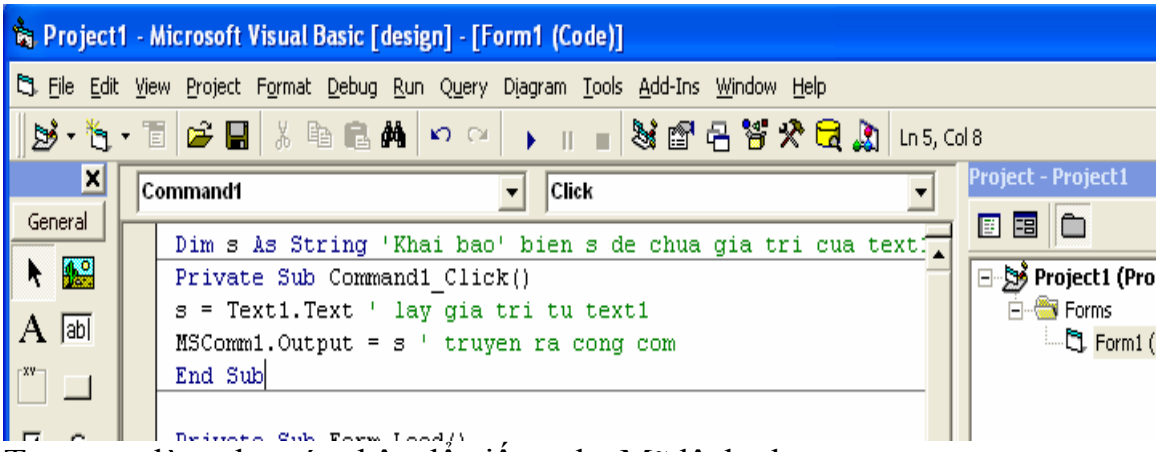
VB tự khởi tạo cho ta một hàm khi load form. Viết mã lệnh như sau:



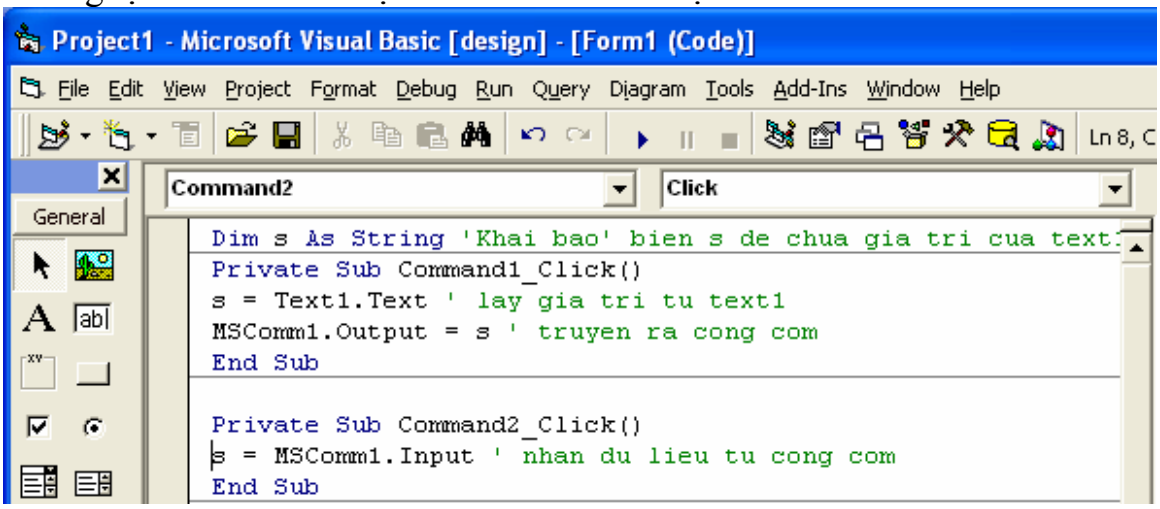
Để viết mã lệnh cho nút truyền kích đúp chuột vào button truyền:



Mã lệnh như sau:



Tương tự làm cho nút nhận để viết code. Mã lệnh như sau:

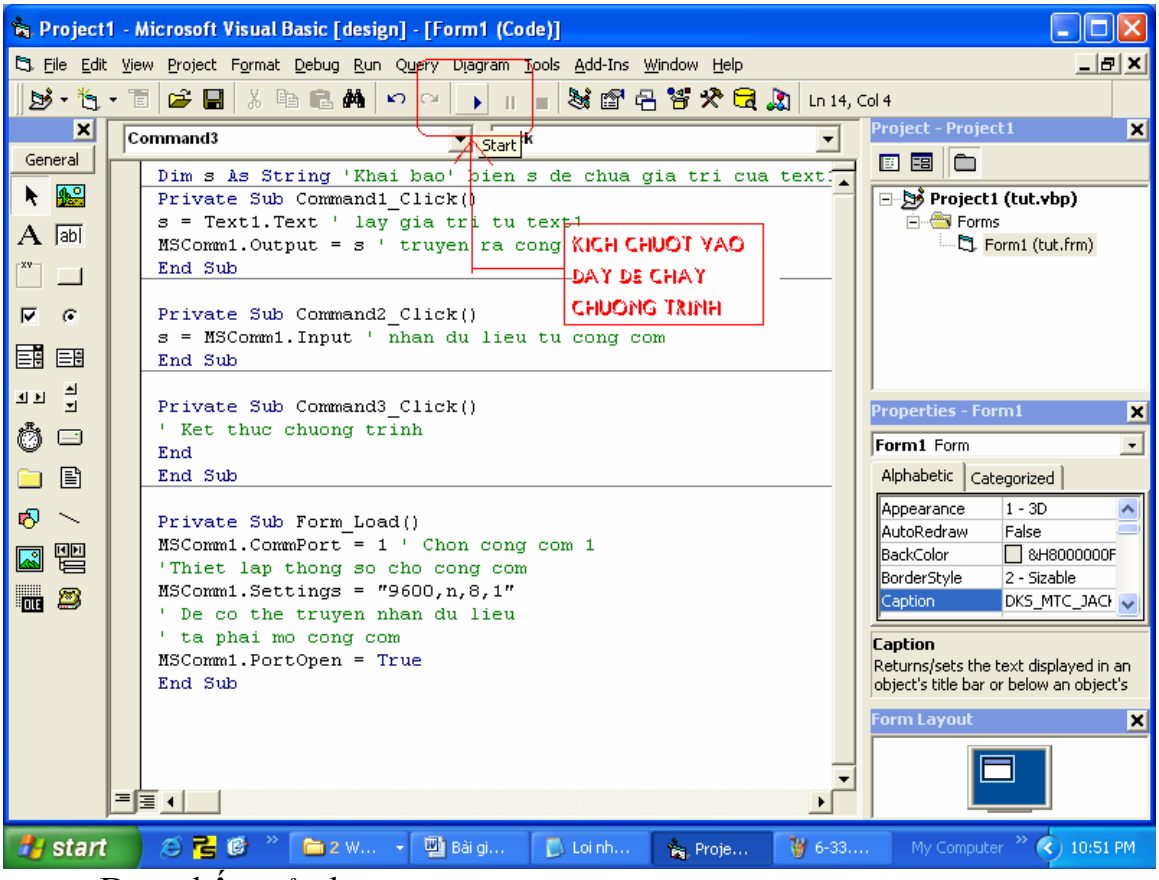


Tương tự làm cho nút EXIT:

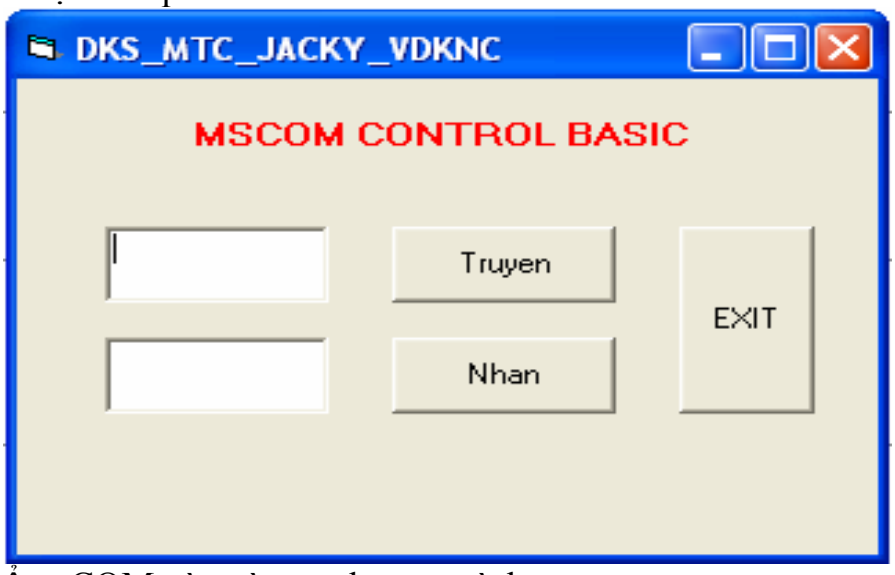
```
Private Sub Command2_Click()
s = MSComm1.Input ' nhan du lieu tu cong com
End Sub

Private Sub Command3_Click()
' Ket thuc chuong trinh
End
End Sub
```

Chọn File → Save Project và File Save Form với tên là tut. Để lưu lại Project vừa tạo.

Chọn File → Make tut.exe để tạo file thực thi và chạy như phần mềm thông thường.



Được kết quả như sau:



Cắm cổng COM vào và test chương trình.

BÀI 7: ĐO LƯỜNG SỬ DỤNG MÁY TÍNH

BÀI 7: ĐO LƯỜNG SỬ DỤNG MÁY TÍNH

1.Yêu cầu:

Điều khiển led đơn trên KIT AVR 03 bằng máy tính.

Đo kết quả ADC từ biến trở và cảm biến nhiệt LM35 hiển thị lên máy tính.

- Điều khiển led:

Tạo một Form bằng VB như sau:

Trong FORM có: 1 đối tượng picturebox chứa logo của DKS. Có 10 đối tượng button trong đó 8 đối tượng button Led1...Led8 là một mảng button có tên từ Command1(0) ... Command1(7). **Muốn tạo một mảng button ta chỉ việc lấy ra 8 đối tượng button và sửa tên tất cả chúng thành Command 1.** Hai button còn lại là Phản hồi và Exit. Có một textbox để hiển thị dữ liệu phản hồi.

Hoạt động của phần mềm như sau:

Khi nhấn vào button Led 1 thì truyền dữ liệu là 0x01 xuống cổng nối tiếp của PC, AVR nhận được và đưa dữ liệu đó ra cổng của AVR để 1 led trên Kit sáng. Tương tự cho bấm các nút Led còn lại. Đồng thời AVR gửi luôn giá trị vừa nhận được lên PC. Và khi bấm nút phản hồi thì dữ liệu đó hiện ra trên Textbox. Khi nhấn nút Exit thì thoát khỏi phần mềm.

Thực hành:
Phần mềm trên VB Code như sau:

```
Private Sub Command1_Click(Index As Integer)
If Index = 0 Then
MSComm1.Output = Chr$(1)
End If
If Index = 1 Then
MSComm1.Output = Chr$(2)
End If
If Index = 2 Then
MSComm1.Output = Chr$(4)
End If
If Index = 3 Then
MSComm1.Output = Chr$(8)
End If
If Index = 4 Then
MSComm1.Output = Chr$(16)
End If
If Index = 5 Then
MSComm1.Output = Chr$(32)
End If
If Index = 6 Then
MSComm1.Output = Chr$(64)
End If
If Index = 7 Then
MSComm1.Output = Chr$(128)
End If
End Sub

Private Sub Command2_Click()
MSComm1.PortOpen = False
End
End Sub

Private Sub Command3_Click()
If MSComm1.Input = "" Then
Exit Sub
Else
Text1.Text = Asc(MSComm1.Input)
End If
End Sub
```

```
Private Sub Form_Load()
MSComm1.CommPort = 1
MSComm1.Settings = "9600,n,8,1"
MSComm1.PortOpen = True
End Sub
```

Firm ware:

Khởi tạo trong CodeVision AVR cho phép cổng nối tiếp hoạt động, PORT D là out put như các bài trước đã học. Sau đó lập trình cho hàm main như sau:

```
227 UCSRC=0x86;
228 UBRRH=0x00;
229 UBRRL=0x33;
230
231 // Analog Comparator initialization
232 // Analog Comparator: Off
233 // Analog Comparator Input Capture by Timer/Counter 1: Off
234 ACSR=0x80;
235 SFIOR=0x00;
236
237 // Global enable interrupts
238 #asm("sei")
239
240 while (1)
241       {
242       // Place your code here
243       if(rx_counter >0) // Co du lieu nhan trong bo nho dem chua
244       {//Neu co
245           temp=getchar();//Lay du lieu vao bien temp
246           PORTA=temp;//Dua ra PORT D
247           putchar(temp);//Gui lai du lieu len may tinh
248       }
249       };
250 }
251
```

Trong hàm main có sử dụng thêm một biến temp nên dĩ nhiên các bạn phải khai báo thêm biến đó ở phía ngoài hàm main.

Nạp chương trình vào chip AVR

Kết nối dây cổng Com từ KIT và cổng Com máy tính và test kết quả.

Đo ADC từ biến trở và LM35.
Trên VB tạo ra một giao diện phần mềm như sau:

Form gồm có:

4 label để hiển thị như hình.
2 text box để hiển thị dữ liệu.
2 button: Thu du lieu và Thoat khỏi phần mềm.

Code trên VB như sau:

```
Private Sub Command1_Click()
If MSComm1.Input = "" Then
Exit Sub
Else
Text1.Text = Asc(MSComm1.Input)
Text2.Text = Asc(MSComm1.Input)
End If
End Sub

Private Sub Command2_Click()
MSComm1.PortOpen = False
```

End
End Sub

Private Sub Form_Load()
MSComm1.CommPort = 1
MSComm1.Settings = "9600,n,8,1"
MSComm1.PortOpen = True
End Sub

Firm Ware:

Khởi tạo bằng CodeWinzard AVR cho cổng nối tiếp USART hoạt động, cho phép ADC hoạt động(interrupt) như các bài trước sau đó viết code cho hàm main như sau:

```
256 ACSR=0x80;
257 SFIOR=0x00;
258
259 // ADC initialization
260 // ADC Clock frequency: 125.000 kHz
261 // ADC Voltage Reference: AREF pin
262 // ADC Auto Trigger Source: None
263 // Only the 8 most significant bits of
264 // the AD conversion result are used
265 ADMUX=FIRST_ADC_INPUT|ADC_VREF_TYPE;
266 ADCSRA=0xCE;
267
268 // Global enable interrupts
269 #asm("sei")
270
271 while (1)
272       {
273       // Place your code here
274
275       putchar(adc_data[0]);
276       putchar(adc_data[1]);
277       delay_ms(1000);
278       };
279 }
280
```

Dịch nạp chương trình và test

.

Bài 8: ĐIỀU KHIỂN STEP MOTOR.

Bài 8: ĐIỀU KHIỂN STEP MOTOR.

1.Yêu cầu :

Hiểu nguyên lí điều khiển động cơ bước đơn cực.

Điều khiển được bằng AVR.

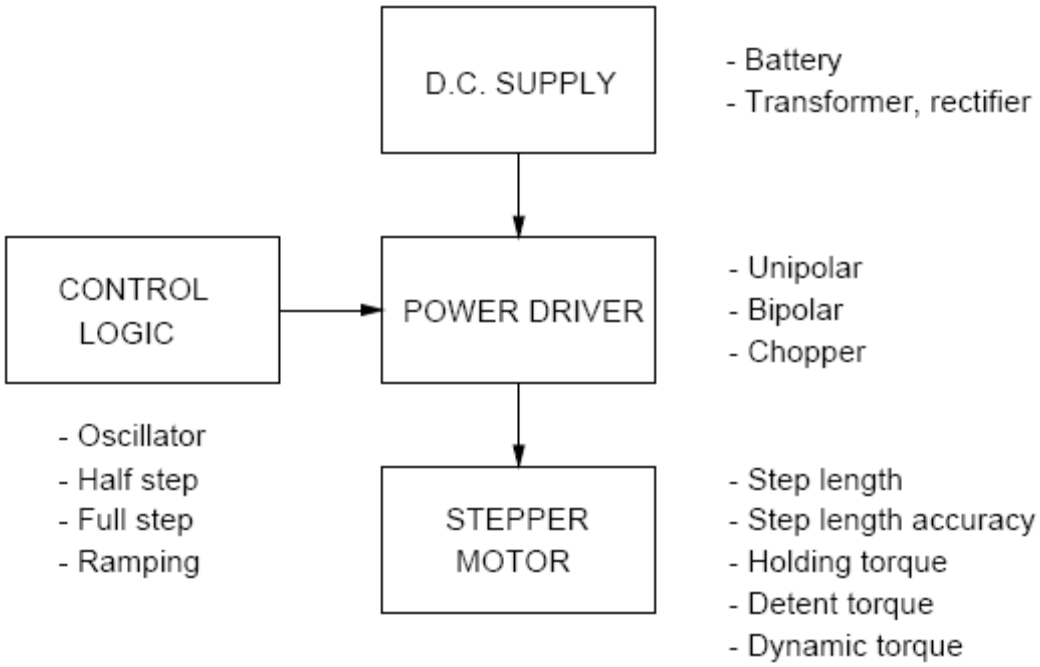
2.Lý thuyết:

2.1.Giới thiệu về động cơ bước:

Động cơ bước thực chất là một động cơ đồng bộ dùng để biến đổi các tín hiệu điều khiển dưới dạng các xung điện rời rạc kế tiếp nhau thành các chuyển động góc quay hoặc các chuyển động của roto và có khả năng cố định roto vào những vị trí cần thiết. Động cơ bước làm việc được là nhờ có bộ chuyển mạch điện tử đưa các tín hiệu điều khiển vào stato theo một thứ tự và một tần số nhất định. Tổng số góc quay của roto tương ứng với số lần chuyển mạch, cũng như chiều quay và tốc độ quay của roto, phụ thuộc vào thứ tự chuyển đổi và tần số chuyển đổi. Khi một xung điện áp đặt vào cuộn dây stato (phần ứng) của động cơ bước thì roto (phần cảm) của động cơ sẽ quay đi một góc nhất định, góc ấy là một bước quay của động cơ. Khi các xung điện áp đặt vào các cuộn dây phần ứng thay đổi liên tục thì roto sẽ quay liên tục. (Nhưng thực chất chuyển động đó vẫn là theo các bước rời rạc).

2.2.Hệ thống điều khiển động cơ bước.

Một hệ thống có sử dụng động cơ bước có thể được khái quát theo sơ đồ sau.



D.C.SUPPLY: Có nhiệm vụ cung cấp nguồn một chiều cho hệ thống. Nguồn một chiều này có thể lấy từ pin nếu động cơ có công suất nhỏ. Với các động cơ có công suất lớn có thể dùng nguồn điện được chỉnh lưu từ nguồn xoay chiều.

CONTROL LOGIC: Đây là khối điều khiển logic. Có nhiệm vụ tạo ra tín hiệu điều khiển động cơ. Khối logic này có thể là một nguồn xung, hoặc có thể là một hệ thống mạch điện tử. Nó tạo ra các xung điều khiển. Động cơ bước có thể điều khiển theo cả bước hoặc theo nửa bước.

POWER DRIVER: Có nhiệm vụ cấp nguồn điện đã được điều chỉnh để đưa vào động cơ. Nó lấy điện từ nguồn cung cấp và xung điều khiển từ khối điều khiển để tạo ra dòng điện cấp cho động cơ hoạt động.

STEPPER MOTOR: Động cơ bước. Các thông số của động cơ gồm có: Bước góc, sai số bước góc, mômen kéo, mômen hãm, mômen làm việc.

Đối với hệ điều khiển động cơ bước, ta thấy đó là một hệ thống khá đơn giản vì không hề có phần tử phản hồi. Điều này có được vì động cơ bước trong quá trình hoạt động không gây ra sai số tích lũy, sai số của động cơ do sai số trong khi chế tạo. Việc sử dụng động cơ bước tuy đem lai độ

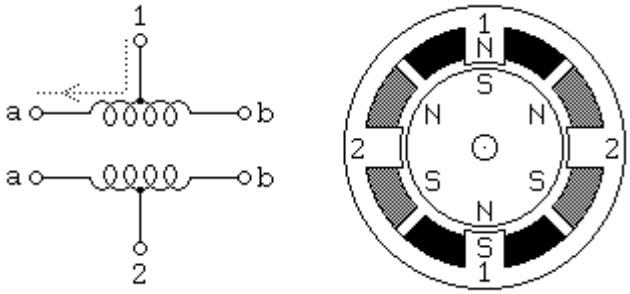
chính xác chưa cao nhưng ngày càng được sử dụng phổ biến. Vì công suất và độ chính xác của bước góc đang ngày càng được cải thiện.

Bước góc của động cơ bước được chế tạo theo bảng tiêu chuẩn sau:

Step angle	Steps per revolution
0.9°	400
1.8°	200
3.6°	100
3.75°	96
7.5°	48
15.0°	24

3.Nguyên tăc điều khiển động cơ bước đơn cực:

Động cơ bước đơn cực, (có thể là động cơ vĩnh cửu hoặc động cơ hỗn hợp) có 5,6 hoặc 8 dây ra thường được quấn như sơ đồ dưới. Khi dùng, các đầu nối trung tâm thường được nối vào cực dương nguồn cấp, và hai đầu còn lại của mỗi mấu lần lượt nối đất để đảo chiều từ trường tạo bởi quận đó.



Hình 1-5 : Động cơ đơn cực.

Tín hiệu điều khiển. Điều khiển đủ bước (full step) :

```
   Winding 1a 10001000100010001000100010001
   Winding 1b 00100010001000100010001000100
   Winding 2a 01000100010001000100010001000
  Winding 2b 00010001000100010001000100010
         time --->
  Winding 1a 11001100110011001100110011001
  Winding 1b 00110011001100110011001100110
  Winding 2a 01100110011001100110011001100
  Winding 2b 10011001100110011001100110011
```

time --->

Điều khiển nửa bước (half step)

Winding 1a 11000001110000011100000111
Winding 1b 00011100000111000001110000
Winding 2a 01110000011100000111000001
Winding 2b 00000111000001110000011100
time --->

4.Mạch điều khiển động cơ bước:

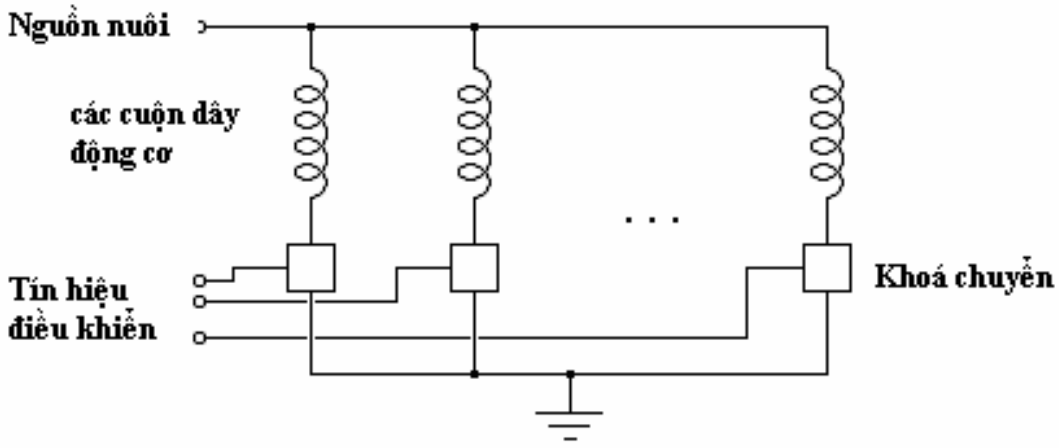
Mạch điều khiển động cơ bước bao gồm một số chức năng sau đây:

Tạo các xung với những tần số khác nhau.

Chuyển đổi các phần cho phù hợp với thứ tự kích từ.

Làm giảm các dao động cơ học.

Đầu vào của mạch điều khiển là các xung. Thành phần của mạch là các bán dẫn, vi mạch. Kích thích các phần của động cơ bước theo thứ tự 1-2-3-4 do các transistor công suất T1 đến T4 thực hiện.Với việc thay đổi vị trí bộ chuyển mạch, động cơ có thể quay theo chiều kim đồng hồ hoặc ngược lại.

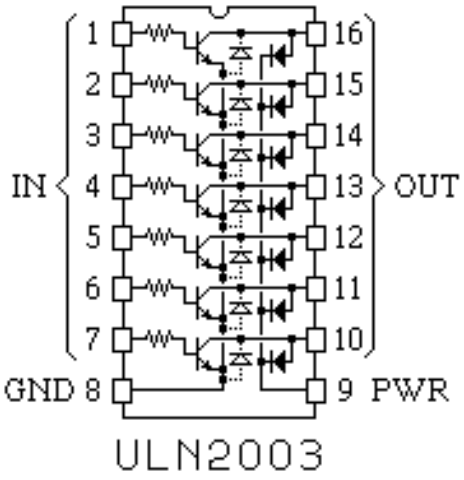


Điện áp được cấp qua các khoá chuyển để nuôi các cuộn dây, tạo ra từ trường làm quay rotor. Các khoá ở đây không cụ thể, có thể là bất cứ thiết bị

đóng cắt nào điều khiển được như rơle, transitor công suất... Tín hiệu điều khiển có thể được đưa ra từ bộ điều khiển như vi mạch chuyên dụng, máy tính. Với động cơ nhỏ có dòng cỡ 500 mili Ampe, có thể dùng IC loại dãy darlington collector hở như :

ULN2003, ULN2803 (Allegro Microsystem)

DS2003 (National Semiconductor), MC1413 (Motorola)



IC họ ULN200x có đầu vào phù hợp TTL, các đầu emitor được nối với chân 8. Mỗi transitor darlington được bảo vệ bởi hai diode. Một mắc giữa emitor tới collector chặn điện áp ngược lớn đặt lên transitor. Diode thứ hai nối collector với chân 9. Nếu chân 9 nối với cực dương của cuộn dây, tạo thành mạch bảo vệ cho transitor.

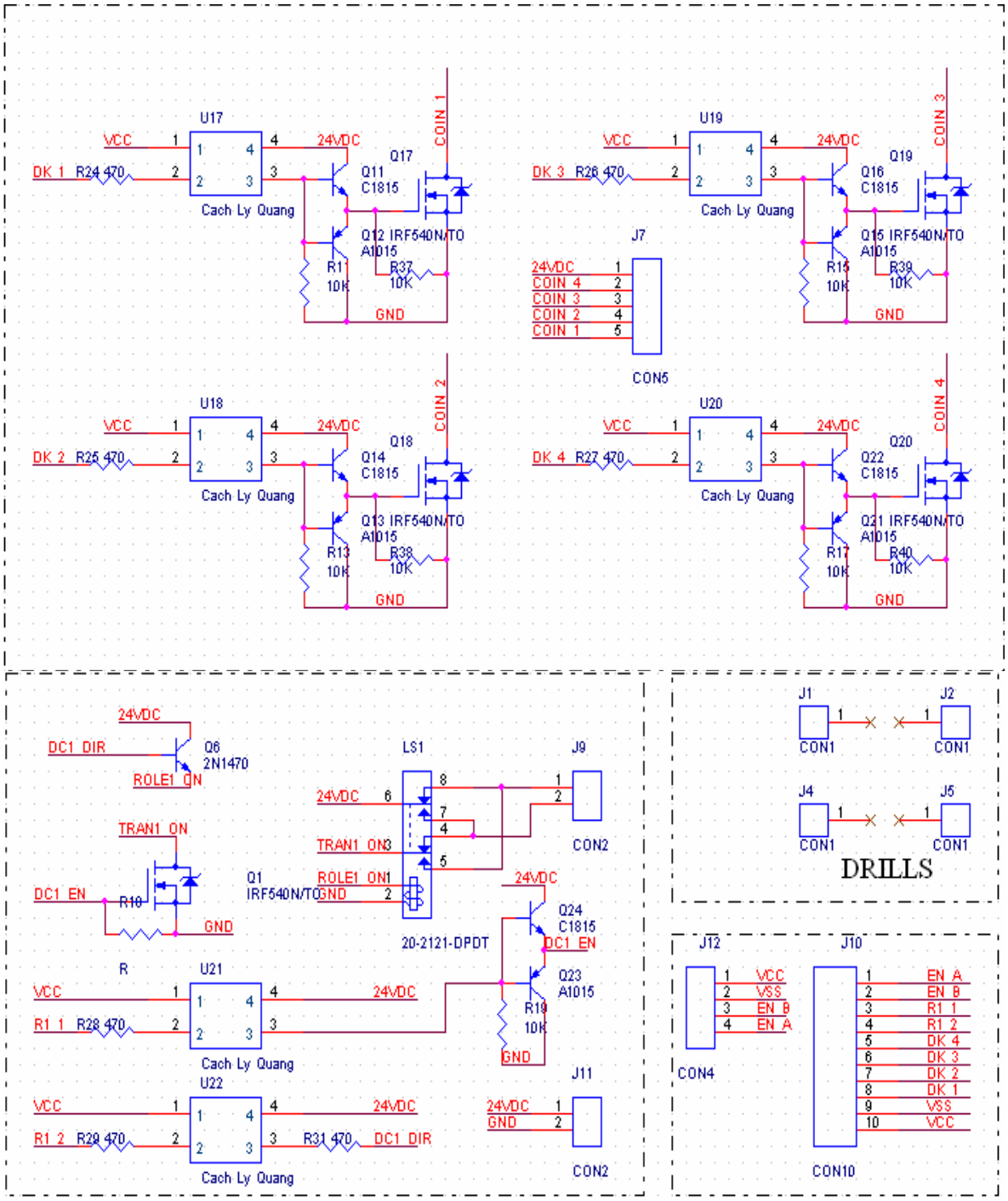
Với các động cơ lớn có dòng > 0.5A các IC họ ULN không đáp ứng được ta có thể dùng các Tranzitor trường(IRF).Một số loại IRF thông dụng:

IRF540 tranzitor ngược có thể chịu dòng đến 20A

IRF640 tranzitor ngược có thể chịu dòng đến 18A

IRF250 tranzitor ngược có thể chịu dòng đến 30A .

Sơ đồ mạch được thiết kế như sau:



5.Code :

```
#include <mega16.h>
#include <delay.h>

// Khai bao bien
unsigned char   stepA[] = {0xFF,0xFE,0xFD,0xFB,0xF7},
                stepB[] = {0xFF,0xEF,0xDF,0xBF,0x7F},
                stepC[] = {0xFF,0xEF,0xDF,0xBF,0x7F};
unsigned char   indexA, indexB, indexC;
unsigned char   n_data;
unsigned char   n_step=10;
unsigned int    n_step3=5000,n_i;
//--------------------
// Declare your global variables here
void main(void)
{
// Declare your local variables here
// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTA=0xFF;
DDRA=0xFF;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTB=0xFF;
DDRB=0xFF;

// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTC=0xFF;
DDRC=0xFF;

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0xFF;
DDRD=0xFF;
```

```
// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
```

```
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

while (1)
      {
      // Place your code here
        if(indexA ++>3) indexA = 1;
        if(indexB ++>3) indexB = 1;
        if(indexC ++>3) indexC = 1;
        PORTA = stepA[indexA] & stepB[indexB];
        PORTC = stepC[indexC];
        //-------
        delay_ms(500);
      }}
```