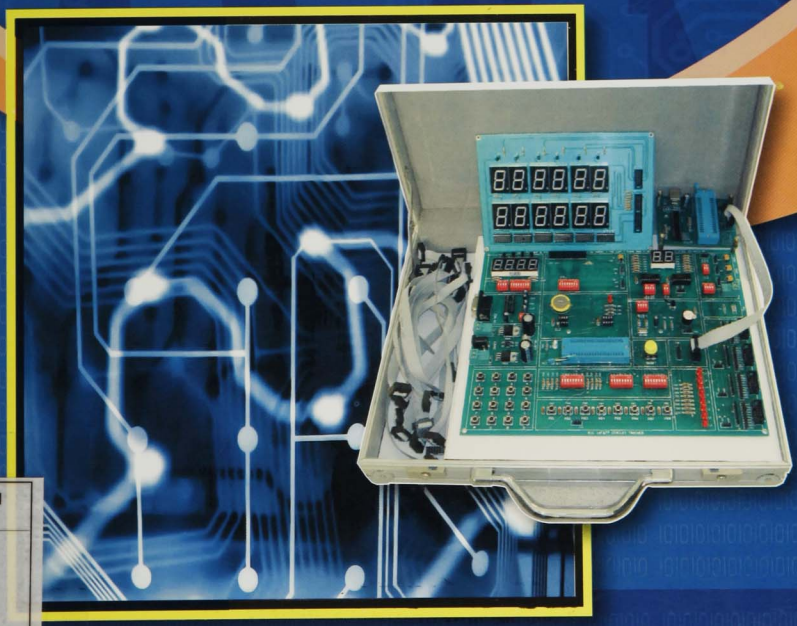
ĂN HÒA

HƯỚNG DẪN THỰC HÀNH VI ĐIỀU KHIỂN AVR



NHÀ XUẤT BẢN XÂY DỰNG

BÙI HỒNG HUẾ - BÙI XUÂN HÒA

HƯỚNG DẪN THỰC HÀNH
VI ĐIỀU KHIỂN AVR

NHÀ XUẤT BẢN XÂY DỰNG
HÀ NỘI - 2012

LỜI NÓI ĐẦU

Hiện nay, các bộ vi điều khiển đã và đang được ứng dụng vào lĩnh vực điều khiển động các quá trình công nghệ như: Tự động hoá quá trình sản xuất, điều khiển hệ thố trạm bơm, điều khiển các thiết bị thuỷ lực và khí nén, điều khiển nhiệt độ, đo lường đi tử... Do giá thành hạ, chi phí thấp nên ngay cả các thiết bị văn phòng, các thiết bị tro gia đình cũng đều có dùng đến các bộ vi điều khiển. Nhưng vấn đề đặt ra là làm thế nà để mau chóng tiếp cận, nghiên cứu ứng dụng các bộ vi điều khiển vào trong lĩnh v điều khiển, tự động hoá phục vụ cho sản xuất và nhu cầu sinh hoạt của chúng ta. l ràng không phải là chỉ nghiên cứu thuần tuý về mặt lí thuyết mà cần có những kh nghiệm cụ thể.

Xuất phát từ thực tế đó, chúng tôi mạnh dạn biên soạn cuốn tài liệu **Hướng dẫn th hành Vi điều khiển AVR** *cùng với Bộ thực tập Vi điều AVR khiển kèm theo nhằm gi cho các em học sinh, sinh viên tiếp cận với các ứng dụng của Vi điều khiển nhanh nh một cách nhanh nhất. Tài liệu tập trung giới thiệu những kiến thức cơ bản nhất về c trúc và lập trình hệ vi điều khiển cùng với các bài tập thực hành. Để tìm hiểu sâu thê các học viên phải đọc các tài liệu lí thuyết về vi điều khiển.*

Tài liệu cũng có thể dùng làm sách tham khảo cho các giáo viên dạy nghề điện, c sinh viên không chuyên điện nhưng có liên quan đến chuyên ngành tự động hoá.

Do bộ tài liệu đề cập đến nhiều vấn đề mới, viết cho nhiều đối tượng ở các trình khác nhau nên không tránh khỏi những thiếu sót. Kính mong các bạn đọc gần xa, c chuyên gia tham gia đóng góp ý kiến để lần tái bản sau được hoàn thiện hơn.

Các tác giả

Phần I

TỔNG QUAN VỀ ATEMEGA 16 VÀ HƯỚNG DẪN CÀI ĐẶT VÀ SỬ DỤNG PHẦN MỀM CODE VISION AVR

I. TỔNG QUAN VỀ ATEMEGA16

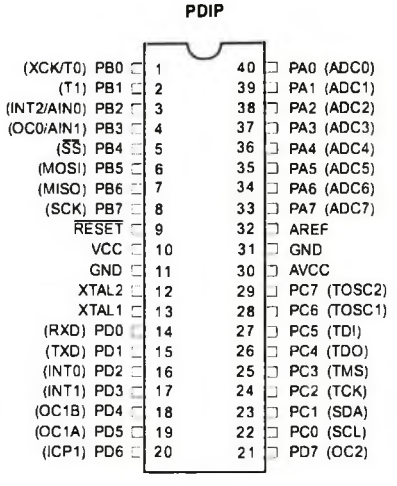
Các tính năng của Atmega16:

- Hiệu xuất cao (high performance), là loại vi điều khiển AVR 8 bit công suất thấp
- Cấu trúc lệnh đơn giản, thời gian thực thi lệnh như nhau.
  - 130 lệnh thực thi trong vòng 1 chu kì chip.
  - 32 thanh ghi công dụng chung 8 bit.
  - Đầy đủ các xử lí tĩnh.
  - Hỗ trợ 16 MIPS khi hoạt động ở tần số 16 MHz.
  - Tích hợp bộ nhân 2 thực hiện trong 2 chu kì chip.
- Bộ nhớ chương trình và dữ liệu không bay hơi (nonvolatile).
  - 16k byte trong hệ thống flash khả trình có thể nạp và xóa 1,000 lần.
  - Tùy chọn khởi động phần mã với các bit nhìn độc lập trong hệ thống bằng cá vào chương trình khởi động chip.
  - 512 byte EEPROM có thể ghi và xóa 100,000 lần.

- 1k byte ram nhớ tinh trong (internal SRAM).
- Lập trình khóa cho phần mềm bảo mật.

- Tính năng ngoại vi
  - 2 bộ định thời/bộ đếm (timers/counters) 8 bit với các chế độ đếm riêng rẽ và kiểu so sánh.
  - 1 bộ định thời/bộ đếm (timer/counter) 16 bit với các chế độ đếm riêng rẽ, kiểu so sánh và kiểu bắt sự kiện.
  - Bộ đếm thời gian thực với máy giao động riêng rẽ.
  - 4 kênh băm xung PWM.
  - 8 kênh ADC 10 bit.
  - Byte định hướng 2 đường giao tiếp nối tiếp.
  - Giao tiếp USART nối tiếp khả trình.
  - Giao tiếp SPI nối tiếp chủ/tớ (master/slave).
  - Bộ định thời khả trình giám sát xung nhịp của chip 1 cách riêng rẽ.
  - Tích hợp bộ so sánh tín hiệu tương tự.
- Giao tiếp JTAG
- Các tính năng đặc biệt của vi điều khiển
  - Chế độ bật nguồn reset và phát hiện Brown-out khả trình.
  - Tích hợp mạch dao động RC bên trong.
  - Các ngắt trong và ngoài.
  - 6 chế độ nghỉ: rảnh rỗi, giảm nhiễu ADC, tiết kiệm năng lượng, nguồn thấp, Standby và Extended Standby.
- Vào/ra và các gói dữ liệu
  - 32 chân vào ra khả trình.
  - 40-pin PDIP and 44-lead TQFP.
- Điện áp sử dụng
  - 2.7 - 5.5V dùng với atmega16L.
  - 4.5 - 5.5V dùng với atmega16.
- Tốc độ xung nhịp dùng cho chip
  - 0 - 8 MHz cho atmega16L.
  - 0 - 16 MHz cho atmega16.

1. Sơ đồ chân



Atmega16 gồm có 40 chân:

- Chân 1 đến 8 : Cổng nhập xuất dữ liệu song song B (PORTB) nó có thể được s dụng các chức năng đặc biệt thay vì nhập xuất dữ liệu.
- Chân 9 : RESET để đưa chip về trạng thái ban đầu.
- Chân 10 : VCC cấp nguồn nuôi cho vi điều khiển.
- Chân 11,31 : GND 2 chân này nối với nhau và nối đất.
- Chân 12,13 : 2 chân XTAL2 và XTAL1 dùng để đưa xung nhịp từ bên ngc vào chip.
- Chân 14 đến 21: Cổng nhập xuất dữ liệu song song D (PORTD) nó có thể được dụng các chức năng đặc biệt thay vì nhập xuất dữ liệu.
- Chân 22 đến 29: Cổng nhập xuất dữ liệu song song C (PORTC) nó có thể được dụng các chức năng đặc biệt thay vì nhập xuất dữ liệu.
- Chân 30: AVCC cấp điện áp so sánh cho bộ ADC.
- Chân 32: AREF điện áp so sánh tín hiệu vào ADC.
- Chân 33 đến 40: Cổng vào ra dữ liệu song song A (PORTA) ngoài ra nó còn đu tích hợp bộ chuyển đổi tín hiệu tương tự sang tín hiệu số ADC (analog to digi converter.

2. Các thanh ghi của vi điều khiển Atmega16

a. Nhóm các thanh ghi truy xuất EFPROM

Bit	15	14	13	12	11	10	9	8	
	-	-	-	-	EEAR11	EEAR10	EEAR9	EEAR8	EEARH
	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEARL
	7	6	5	4	3	2	1	0	
Read/write	R	R	R	R	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
initial Value	0	0	0	0	X	X	X	X	
	X	X	X	X	X	X	X	X	

1. Thanh Ghi EEAR (EEARH và EEARL)

EEAR là thanh ghi 16 bit lưu giữ địa chỉ của các ô nhớ của EEPROM, thanh ghi EEAR được kết hợp từ 2 thanh ghi 8 bit là EEARH và thanh ghi EEARL. Vì bộ nhớ EEPROM của ATmega128 có dung lượng 4 Kbyte = 4096 byte = 212 byte nên ta chỉ cần 12 bit của thanh ghi EEAR , 4 bit từ 15 -12 được dự trữ, ta nên ghi 0 vào các bit dự trữ này.

2. Thanh Ghi EEDR

Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	EEDR
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Đây là thanh ghi dữ liệu của EEPROM, là nơi chứa dữ liệu ta định ghi vào hay lấy ra từ EEPROM.

3. Thanh Ghi EECR

Bit	7	6	5	4	3	2	1	0	
	-	-	-		EERIE	EEMWE	EEWE	EERE	EECR
Read/write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	X	0	

Đây là thanh ghi điều khiển EEPROM, ta chỉ sử dụng 4 bit đầu của thanh ghi này, 4 bit cuối là dự trữ, ta nên ghi 0 vào các bit dự trữ. Sau đây ta xét chức năng của từng bit.

Bit 3 - EERIE: EEPROM Ready Interrupt Enable: Đây là bit cho phép EEPROM ngắt CPU, khi bit này được set thành 1 và ngắt toàn cục được cho thì EEPROM sẽ tạo ra một ngắt với CPU khi bit EEWE được xóa, điều này có nghĩa là khi các ngắt được cho phép và quá trình ghi vào ROM vừa xong thì sẽ tạo ra một ngắt với CPU, chương trình sẽ nhảy tới véc tơ ngắt có địa chỉ là 002C để thực thi chương trình phục vụ ngắt (ISR). Khi bit EERIE là 0 thì ngắt không được cho phép.

Bit 2 - EEMWE: EEPROM Master Write Enable: Khi bit EEMWE và bit EEWE là 1 sẽ ra lệnh cho CPU ghi dữ liệu từ thanh ghi EEDR vào EEPROM, địa chỉ của ô nhớ cần ghi trong EEPROM được lưu trong thanh ghi EEAR. Khi bit này là 0 thì không cho phép ghi vào EEPROM. Bit EEMWE sẽ được xóa bởi phần cứng sau 4 chu kì máy.

Bit 1 - EEWE: EEPROM Write Enable : Bit này vừa đóng vai trò như một bit cờ, vừa là bit điều khiển việc ghi dữ liệu vào EEPROM. Ở vai trò của một bit điều khiển nếu bit EEMWE đã được set lên 1 thì khi ta set bit EEWE lên 1 sẽ bắt đầu quá trình ghi dữ liệu vào EEPROM. Trong suốt quá trình ghi dữ liệu vào EEPROM bit EEWE luôn giữ là 1. Ở vai trò của một bit cờ khi quá trình ghi dữ liệu vào EEPROM hoàn tất, phần cứng sẽ tự động xóa bit này về 0. Lúc này ta có thể bắt đầu ghi dữ liệu vào EEPROM.

Bit 0 - EERE: EEPROM Read Enable : Khi bit này là 1, sẽ cho phép đọc dữ liệu từ EEPROM, dữ liệu từ EEPROM có địa chỉ lưu trong thanh ghi EEAR lập tức được chuyển vào thanh ghi EEDR. Khi bit EERE là 0 thì không cho phép đọc EEPROM.

b. Nhóm các thanh ghi vào ra

1. Thanh Ghi DDRx

Đây là thanh ghi 8 bit (có thể đọc ghi) có chức năng điều khiển hướng của cổng (là lối ra hay lối vào). Khi một bit của thanh ghi này được set lên 1 thì chân tương ứng với nó được cấu hình thành ngõ ra. Ngược lại, nếu bit của thanh ghi DDRx là 0 thì chân tương ứng với nó được thiết lập thành ngõ vào. Lấy ví dụ: Khi ta set tất cả 8 bit của thanh ghi DDRA đều là 1, thì 8 chân tương ứng của portA là PA1 tới PA7 (tương ứng với các chân số 50 tới 44 của vi điều khiển) được thiết lập thành ngõ ra.

Bit	7	6	5	4	3	2	1	0	
	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	LSB	DDRA
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
initial Value	0	0	0	0	0	0	0	0	

Thanh ghi DDRA

2. Thanh Ghi PORTx

PORTx là thanh ghi 8 bit có thể đọc ghi. Đây là thanh ghi dữ liệu của PORTx, nếu thanh ghi DDRx thiết lập cổng là lối ra, khi đó giá trị của thanh ghi PORTx cũng là giá trị của các chân tương ứng của PORTx, nói cách khác, khi ta ghi một giá trị logic lên 1 bit của thanh ghi này thì chân tương ứng với bit đó cũng có cùng mức logic. Khi thanh ghi DDRx thiết lập cổng thành lối vào thì thanh ghi PORTx đóng vai trò như một thanh ghi điều khiển cổng. Cụ thể, nếu một bit của thanh ghi này được ghi thành 1 thì điện trở treo (pull-up resistor) ở chân tương ứng với nó sẽ được kích hoạt, ngược lại nếu bit được ghi thành 0 thì điện trở treo ở chân tương ứng sẽ không được kích hoạt, cổng ở trạng thái cao trở (Hi-Z).

Bit	7	6	5	4	3	2	1	0	
	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	PORTA
Read/ write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
initial Value	0	0	0	0	0	0	0	0	

Thanh ghi PORTA

3 .Thanh Ghi PINx

PINx không phải là một thanh ghi thực sự, đây là địa chỉ trong bộ nhớ I/O kết nối trực tiếp tới các chân của cổng. Khi ta đọc PORTx tức ta đọc dữ liệu được chốt trong PORTx, còn khi đọc PINx thì giá trị logic hiện thời ở chân của cổng tương ứng được đọc. Vì thế đối với thanh ghi PINx ta chỉ có thể đọc mà không thể ghi.

Bit	7	6	5	4	3	2	1	0	
	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	PINA
Read/ write	R	R	R	R	R	R	R	R	
initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Thanh ghi PINA

c. Các thanh ghi của bộ định thời

1. Thanh ghi TCCR1A (Timer/Counter1 Control Register)

Bit	7	6	5	4	3	2	1	0	
	COM1A1	COM1A0	COM1B1	COM1B0	COM1C1	COM1C0	WGM11	WGM10	TCCR1A
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Bit 7:6 - COMnA1:0: Compare Output Mode for Channel A

Bit 5:4 - COMnB1:0: Compare Output Mode for Channel B

Bit 3:2 - COMnC1:0: Compare Output Mode for Channel C

Bit 1:0 - WGMn1:0: Waveform Generation Mode

Bit 7:2 - COMnX1:0 (X=A, B, C): Compare Output Mode for Channel X :

Để tìm hiểu cách thức sử dụng ta theo dõi các bảng dưới đây:

COMnA1/COMnB1/COMnC1	COMnA0/COMnB0/COMnC0	Description
0	0	Hoạt động ở chế độ thông thường các cổng OCnA/OCnB/OCnC không kết nối
0	1	Chuyển OCnA/OCnB/OCnC ở chế độ so sánh khớp
1	0	Xóa OCnA/OCnB/OCnC khỏi chế độ so sánh khớp (đầu ra mức thấp)
1	1	Đặt OCnA/OCnB/OCnC ở chế độ so sánh khớp (đầu ra mức cao)

Bảng 58. Hành vi của các chân OCnX (X=A, B, C; n=1, 3) phụ thuộc vào các thiết lập của các bit COMnA1:0, COMnB1:0, COMnC1:0 trong chế độ non-PWM

COMnA1/COMnB1/COMnC1	COMnA0/COMnB0/COMnC0	Description
0	0	Hoạt động ở chế độ thông thường các cổng OCnA/OCnB/OCnC không kết nối
0	1	WGMn3:0=15; chuyển OCnA ở chế độ so sánh khớp OCnB/OCnC không kết nối (cổng hoạt động ở chế độ bình thường) Ở các chế độ cài đặt WGMn khác, các cổng hoạt động bình thường OCnA/OCnB/OCnC không kết nối
1	0	Xóa OCnA/OCnB/OCnC khỏi chế độ so sánh khớp, đặt OCnA/OCnB/OCnC ở TOP
1	1	Đặt OCnA/OCnB/OCnC khỏi chế độ so sánh khớp, xóa OCnA/OCnB/OCnC khỏi TOP

Bảng 59. Hành vi của các chân OCnX (X = A, B, C; n = 1, 3) phụ thuộc vào các thiết lập của các bit COMnA1:0, COMnB1:0, COMnC1:0 trong chế độ Fast-PWM

COMnA1/COMnB1/COMnC1	COMnA0/COMnB0/COMnC0	Description
0	0	Hoạt động ở chế độ thông thường các cổng OCnA/OCnB/OCnC không kết nối
0	1	Chuyển OCnA/OCnB/OCnC ở chế độ so sánh khớp
1	0	Xóa OCnA/OCnB/OCnC khỏi chế độ so sánh khớp (đầu ra mức thấp)
1	1	Đặt OCnA/OCnB/OCnC ở chế độ so sánh khớp (đầu ra mức cao)

Bảng 60. Hành vi của các chân OCnX (X=A, B, C; n=1, 3) phụ thuộc vào các thiết lập của các bit COMnA1:0, COMnB1:0, COMnC1:0 tromg chế độ PWM hiệu chỉnh pha và tần số

Table 61. Waveform Generation Mode Bit Description

Mode	WGMn3	WGMn2 (CTCn)	WGMn1 (PWMn1)	WGMn0 (PWMn0)	Timer/Counter Mode of Operation	TOP	Update of OCRnx at	TOVn Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCRnA	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	TOP	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	TOP	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	TOP	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICRn	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCRnA	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICRn	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCRnA	TOP	BOTTOM
12	1	1	0	0	CTC	ICRn	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICRn	TOP	TOP
15	1	1	1	1	Fast PWM	OCRnA	TOP	TOP

Bảng 61. Lựa chọn các chế độ thực thi của bộ định thời 1(3)

2. Thanh ghi TCCR1B

7	6	5	4	3	2	1	0	
ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	

Bit 7 - ICNCn: Input Capture Noise Canceler

Bit 6 - ICESn: Input Capture Edge Select

Bit 5 - Reserved Bit

Bit 4:3 - WGMn3:2: Waveform Generation Mode

Bit 2:0 - CSn2:0: Clock Select

Bit 7 - ICNCn: Input Capture Noise Canceler (viết tắt: ICNC): Việc set bit này tới 1 sẽ kích hoạt chức năng chống nhiễu của bộ chống nhiễu lối vào (ICNC). Khi chức năng ICNC được kích hoạt thì ngõ vào từ chân ICPn sẽ được lọc. Chức năng lọc đòi hỏi 4 mẫu có giá trị bằng nhau liên tiếp ở chân ICPn cho sự thay đổi ngõ ra của nó (xem chi tiết về khối Input Capture).

Bit 6 - ICESn: Input Capture Edge Select: Bit này lựa chọn cạnh ở chân Input Capture Pin (ICPn) dùng để bắt "sự kiện trigger" (Trigger event (10)). Khi bit ICESn được thiết lập thành 0 thì một cạnh dương xuống (falling(3)) được dùng như một trigger (tín hiệu nảy). Ngược lại, khi bit này được set thành 1 thì một cạnh âm lên (rising(4)) được dùng như một trigger. Khi xảy ra sự kiện Input capture(2) (theo thiết lập của bit ICESn là 1 hay 0) thì giá trị của bộ đếm được ghi vào thanh ghi Input Capture Register ICRn (n = 1, 3), và khi đó cờ ICFn (Input Capture Flag) được set. Điều này sẽ tạo ra một ngắt Input capture nếu ngắt này được cho phép. Khi thanh ghi ICRn được sử dụng như một giá trị TOP thì chân ICPn không được kết nối và vì thế chức năng Input capture không được cho phép.

Table 62. Clock Select Bit Description

CSn2	CSn1	CSn0	Description
0	0	0	No clock source. (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/1 (No prescaling
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on Tn pin. Clock on falling edge
1	1	1	External clock source on Tn pin. Clock on rising edge

Bảng 62. Lựa chọn tốc độ xung clock

Bit 5 : Dự trữ.

Bit 4:3 - WGMn3:2: Waveform Generation Mode: Đã nói ở phần thanh ghi TCCR1A.

Bit 2:0 - CSn2:0: Clock Select: Dùng để lựa chọn tốc độ xung clock (xem bảng 62). Để cấm bộ định thời hoạt động ta chỉ cần cho {CSn2, CSn1, CSn0} = {0, 0, 0}.

3. Thanh ghi TCCR1C

7	6	5	4	3	2	1	0	
FOC1A	FOC1B	-	-	-	-	-	-	TCCR1C
W	W	W	R	R	R	R	R	

Bit 7 - FOCnA: Force Output Compare for Channel A

Bit 6 - FOCnB: Force Output Compare for Channel B

Bit 5 - FOCnC: Force Output Compare for Channel C

Bit 4:0 - Reserved Bits

Các bit FOCnA/FOCnB/FOCnC chỉ hoạt động khi các bit WGMn3:0 chỉ định chế độ Non-PWM. Khi các bit FOCnA/FFOCnB/FOCnC được set thành 1 thì ngay lập tức một sự kiện "So sánh khớp cưỡng chế" (Forced Compare Match(1)) xảy ra trong bộ tạo sóng. Ngõ ra OCnA/OCnB/OCnC được thay đổi theo thiết lập của các bit COMnX 1:0 (n=1, 3; X=A, B, C), nghĩa là bình thường sự kiện "so sánh khớp" chỉ xảy ra khi khi giá trị bộ định thời (thanh ghi TCNTn (n=1, 3)) bằng với giá trị thanh ghi OCRnX(n=1,3; X=A,B,C), nhưng khi các bit FOCnX(n=1, 3; X=A, B, C) được set thành 1 thì sự kiện "so sánh khớp" sẽ xảy ra mặc dù giá trị của bộ định thời không bằng với giá trị của thanh ghi OCRnX(n=1,3; X=A,B,C). Chú ý là các bit FOCnA/FOCnB/FOCnC cũng hoạt động như là những que dò (strobe), vì thế nó là giá trị hiện thời của các bit COMnX1:0 xác định tác động của "so sánh cưỡng chế" (forced compare). Các que dò FOCnA/FOCnB/FOCnC không tạo ra bất kì ngắt nào và cũng không xóa bộ định thời trong chế độ CTC sử dụng thanh ghi OCRnA như là giá trị TOP. Các bit FOCnA/FOCnB/FOCnC chỉ có thể ghi, khi đọc các bit này ta luôn nhận được giá trị 0.

Bit 4:0 dự trữ, phải ghi thành 0 khi ghi vào thanh ghi TCCRnC.

4. Thanh Ghi Timer/Counter1 - TCNT1H and TCNT1L

7	6	5	4	3	2	1	0	
TCNT1[15:8]								TCNT1H
TCNT1[7:0]								TCNT1L
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Thanh ghi bộ định thời TCNT1 là thanh ghi 16 bit được kết hợp từ hai thanh ghi TCNT1H và thanh ghi TCNT1L. Thanh ghi TCNT1 có thể đọc hay ghi. Để cả 2 byte của TCNT 1 được đọc hay ghi đồng thời người ta dùng một thanh ghi tạm 8 bit byte cao 8-bit Temporary High Byte Register (TEMP). Thanh ghi TEMP được chia sẽ cho tất cả các thanh ghi 16 bit khác. Không nên chỉnh sửa thanh ghi TCNTn (n =1,3) khi nó đang đếm để tránh bị hỏng Compare Match giữa TCNTn và một trong những thanh ghi OCRnX(n=1,3. X=A,B,C).

5. Thanh Ghi Output Compare Register 1 A-OCR1AH and OCR1AL

Bit	7	6	5	4	3	2	1	0	
	OCR1A[15:8]								OCR1AH
	OCR1A[7:0]								OCR1AL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

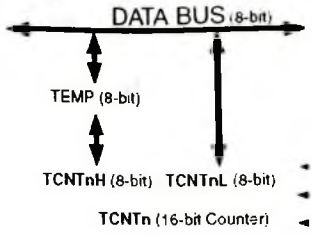
6. Thanh Ghi Output Compare Register 1 B-OCR1BH and OCR1BL

Bit	7	6	5	4	3	2	1	0	
	OCR1B[15:8]								OCR1BH
	OCR1B[7:0]								OCR1BL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

7. Thanh Ghi Output Compare Register 1 C-OCR1CH and OCR1CL

Bit	7	6	5	4	3	2	1	0	
	OCR1C[15:8]								OCR1CH
	OCR1C[7:0]								OCR1CL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Thanh ghi output compare register (OCR1A/OCR1B/OCR1C) là thanh ghi 16 bit, giá trị của nó được liên tục so sánh với bộ đếm (TCNT1). Khi có sự bằng nhau của hai thanh ghi này sẽ tạo ra một ngắt so sánh hay một dạng sóng ở chân ngõ ra so sánh OcnX (X=A,B,C). Giống như thanh ghi TCNT1, thanh ghi OCRnX (X=A,B,C) cũng là thanh ghi 16 bit nên để cả hai byte cao và thấp của thanh ghi được ghi hay đọc đồng thời khi CPU cần truy xuất thanh ghi này, người ta dùng thanh ghi tạm byte cao (TEMP), thanh ghi TEMP luôn lưu giữ byte cao của các thanh ghi 16 bit khi các thanh ghi này cần dùng tới nó (xem hình 3.1). Chú ý là khi ghi một giá trị vào thanh ghi OCRnX trong lúc bộ đếm đang chạy, thì giá trị của thanh ghi OCRnX có thể cập nhật tức thời, nhưng cũng có thể chỉ được cập nhật khi bộ đếm đạt tới một giá trị nào đó (bảng 61), chẳng hạn, giá trị TOP, BOTTOM...



Hình 3.1. Thanh ghi TEMP

8. Thanh Ghi Input Capture Register 1-ICR1H and ICR1L

Bit	7	6	5	4	3	2	1	0	
	ICR1[15:8]								ICR1H
	ICR1[7:0]								ICR1L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Thanh ghi Input capture (ICR1n) sẽ cập nhật giá trị của bộ đếm TCNTn mỗi khi xảy ra sự kiện ở chân ICPn. Ngoài ra thanh ghi này còn được sử dụng để định nghĩa giá trị TOP của bộ đếm. Người ta cũng sử dụng thanh ghi TEMP khi cần truy xuất thanh ghi ICRn (n=1, 3).

9. Thanh Ghi Timer/Counter Interrupt Mask Register-TIMSK (Interrupt for Timer/counter 1)

Bit	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit 5 - TICIE1: Timer/Counter1, Input Capture Interrupt Enable

Bit 4 - OCIE1A: Timer/Counter1, Output Compare A Match Interrupt Enable

Bit 3 - OCIE1B: Timer/Counter1, Output Compare B Match Interrupt Enable

Bit 2 - TOIE1: Timer/Counter1, Overflow Interrupt Enable

Bit 5 - TICIE1: Timer/Counter1, Input Capture Interrupt Enable: Khi bit này được set thành 1 và ngắt toàn cục (global interrupt) được cho phép thì ngắt bắt mẫu ngõ vào bộ Timer/couter1 (Timer/Counter1 Input Capture interrupt) được cho phép. Vector ngắt tương ứng sẽ được thực thi khi cờ ICF1 trong thanh ghi TIFR được set.

Bit 4 - OCIE1A: Timer/Counter1, Output Compare A Match Interrupt Enable: Khi bit này được set thành 1 và ngắt toàn cục (global interrupt) được cho phép thì ngắt so sánh ngõ ra 1A (Timer/Counter1 Output Compare A Match Interrupt) được cho phép. Vector ngắt tương ứng sẽ được thực thi khi cờ OCF1A trong thanh ghi TIFR được set.

Bit 3 - OCIE1B: Timer/Counter1, Output Compare B Match Interrupt Enable: Khi bit này được set thành 1 và ngắt toàn cục (global interrupt) được cho phép thì ngắt so sánh ngõ ra 1B (Timer/Counter1 Output Compare B Match Interrupt) được cho phép. Vector ngắt tương ứng sẽ được thực thi khi cờ OCF1B trong thanh ghi TIFR được set.

Bit 2 - TOIE1: Timer/Counter1, Overflow Interrupt Enable: Khi bit này được set thành 1 và ngắt toàn cục (global interrupt) được cho phép thì ngắt cờ tràn bộ định thời 1 (Timer/Counter1 overflow interrupt) được cho phép. Vector ngắt tương ứng sẽ được thực thi khi cờ TOV1 trong thanh ghi TIFR được set.

10. Thanh Ghi Extended Timer/Counter Interrupt Mask Register-ETIMSK (Interrupt for Timer/counter 3)

Bit	7	6	5	4	3	2	1	0	
	-	-	TICIE3	OCIE3A	OCIE3B	TOIE3	OCIE3C	TOIE1C	ETIMSK
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit 7:6 - Reserved Bits

Bit 5 - TICIE3: Timer/Counter3, Input Capture Interrupt Enable

Bit 4 - OCIE3A: Timer/Counter3, Output Compare A Match Interrupt Enable

Bit 3 - OCIE3B: Timer/Counter3, Output Compare B Match Interrupt Enable

Bit 2 - TOIE3: Timer/Counter3, Overflow Interrupt Enable

Bit 1 - OCIE3C: Timer/Counter3, Output Compare C Match Interrupt Enable

Bit 0 - OCIE1C: Timer/Counter1, Output Compare C Match Interrupt Enable

Thanh ghi ETIMSK liên quan đến cả hai bộ định thời 1 và 3.

Bit 7:6 - Reserved Bits: Dự trữ, phải ghi các bit này thành 0 khi ghi vào thanh ghi ETIMSK

Bit 5 - TICIE3: Timer/Counter3, Input Capture Interrupt Enable: Khi bit này được set thành 1 và ngắt toàn cục (global interrupt) được cho phép thì ngắt bắt mẫu ngõ vào bộ Timer/couter 3 (Timer/Counter3 Input Capture interrupt) được cho phép. Vector ngắt tương ứng sẽ được thực thi khi cờ ICF3 trong thanh ghi ETIFR được set.

Bit 4 - OCIE3A: Timer/Counter3, Output Compare A Match Interrupt Enable: Khi bit này được set thành 1 và ngắt toàn cục (global interrupt) được cho phép thì ngắt so sánh ngõ ra 3A (Timer/Counter1 Output Compare A Match Interrupt) được cho phép. Vector ngắt tương ứng sẽ được thực thi khi cờ OCF3A trong thanh ghi ETIFR được set.

Bit 3 - OCIE3B: Timer/Counter3, Output Compare B Match Interrupt Enable: Khi bit này được set thành 1 và ngắt toàn cục (global interrupt) được cho phép thì ngắt so sánh ngõ ra 3B (Timer/Counter3 Output Compare B Match Interrupt) được cho phép. Vector ngắt tương ứng sẽ được thực thi khi cờ OCF3B trong thanh ghi ETIFR được set.

Bit 2 - TOIE3: Timer/Counter3, Overflow Interrupt Enable: Khi bit này được set thành 1 và ngắt toàn cục (global interrupt) được cho phép thì ngắt cờ tràn bộ định thời 3 (Timer/Counter3 overflow interrupt) được cho phép. Vector ngắt tương ứng sẽ được thực thi khi cờ TOV4 trong thanh ghi ETIFR được set.

Bit 1 - OCIE3C: Timer/Counter3, Output Compare C Match Interrupt Enable: Khi bit này được set thành 1 và ngắt toàn cục (global interrupt) được cho phép thì ngắt so sánh ngõ ra 3C (Timer/Counter3 Output Compare C Match Interrupt) được cho phép. Vector ngắt tương ứng sẽ được thực thi khi cờ OCF3C trong thanh ghi ETIFR được set.

Bit 0 - OCIE1C: Timer/Counter1, Output Compare C Match Interrupt Enable: Khi bit này được set thành 1 và ngắt toàn cục (global interrupt) được cho phép thì ngắt so sánh ngõ ra 1C (Timer/Counter1 Output Compare C Match Interrupt) được cho phép. Vector ngắt tương ứng sẽ được thực thi khi cờ OCF1C trong thanh ghi ETIFR được set.

11. Thanh Ghi Timer/Counter Interrupt Flag Register- TIFR

Bit	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOF0	TIFR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit 5 - ICF1: Timer/Counter1, Input Capture Flag

Bit 4 - OCF1A: Timer/Counter1, Output Compare A Match Flag

Bit 3 - OCF1B: Timer/Counter1, Output Compare B Match Flag

Bit 2 - TOV1: Timer/Counter1, Overflow Flag

Thanh ghi TIFR liên quan tới bộ định thời 1 và 2.

Bit 5 - ICF1: Timer/Counter1, Input Capture Flag: Cờ này được set khi xảy ra sự kiện bắt mẫu ngõ vào (Input Capture) của chân ICP1. Khi thanh ghi ICR1 (Input Capture Register) được thiết lập bởi các bit WGMn3:0 để sử dụng như một giá trị TOP thì cờ ICF1 sẽ được set khi bộ đếm đạt tới giá trị TOP. Cờ ICF1 sẽ tự động xóa khi ngắt tương ứng được thực thi, hoặc có thể xóa hay set bằng cách ghi một giá trị logic vào vị trí của nó.

Bit 4 - OCF1A: Timer/Counter1, Output Compare A Match Flag: Cờ này được set ngay sau khi giá trị bộ đếm (TCNT1) bằng với giá trị thanh ghi OCR1A (Output Compare Register A). Chú ý là một so sánh cưỡng bức (FOC1A) sẽ không set cờ này. Cờ OCF1A sẽ tự động xóa khi ngắt tương ứng được thực thi, hoặc có thể xóa hay set bằng cách ghi một giá trị logic vào vị trí của nó.

Bit 3 - OCF1B: Timer/Counter1, Output Compare B Match Flag: Cờ này được set ngay sau khi giá trị bộ đếm (TCNT1) bằng với giá trị thanh ghi OCR1B (Output Compare Register B). Chú ý là một so sánh cưỡng bức (FOC1B) sẽ không set cờ này. Cờ OCF1B sẽ tự động xóa khi ngắt tương ứng được thực thi, hoặc có thể xóa hay set bằng cách ghi một giá trị logic vào vị trí của nó.

Bit 2 - TOV1: Timer/Counter1, Overflow Flag: Việc thiết lập cờ này phụ thuộc vào thiết lập của các bit WGMn3:0, trong chế độ bình thường và CTC cờ TOV1 được set khi bộ định thời tràn. Xem lại bảng 61 và mục "Các chế độ thực thi" để biết các trường hợp khác.

12. Thanh Ghi Extended Timer/Counter Interrupt Flag Register-ETIFR

Bit	7	6	5	4	3	2	1	0	
	-	-	ICF3	OCF3A	OCF3B	TOV3	OCF3C	OCF1C	ETIFR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit 7:6 - Reserved Bits

Bit 5 - ICF3: Timer/Counter3, Input Capture Flag

Bit 4 - OCF3A: Timer/Counter3, Output Compare A Match Flag

Bit 3 - OCF3B: Timer/Counter3, Output Compare B Match Flag

Bit 2 - TOV3: Timer/Counter3, Overflow Flag

Bit 1 - OCF3C: Timer/Counter3, Output Compare C Match Flag

Bit 0 - OCF1C: Timer/Counter1, Output Compare C Match Flag

Bit 7:6 - Reserved Bits: Dự trữ, phải ghi 0 khi ghi vào thanh ghi ETIFR.

Bit 5 - ICF3: Timer/Counter3, Input Capture Flag: Cờ này được set khi xảy ra sự kiện bắt ngõ vào (Input Capture) của chân ICP3. Khi thanh ghi ICR3 (Input Capture Register) được thiết lập bởi các bit WGMn3:0 để sử dụng như một giá trị TOP thì cờ ICF3 sẽ được set khi bộ đếm đạt tới giá trị TOP. Cờ ICF3 sẽ tự động xóa khi ngắt tương ứng được thực thi, hoặc có thể xóa hay set bằng cách ghi một giá trị logic vào vị trí của nó.

Bit 4 - OCF3A: Timer/Counter3, Output Compare A Match Flag: Cờ này được set ngay sau khi giá trị bộ đếm (TCNT3) bằng với giá trị thanh ghi OCR3A (Output Compare Register A). Chú ý là một so sánh cưỡng bức (FOC3A) sẽ không set cờ này. Cờ OCF3A sẽ tự động xóa khi ngắt tương ứng được thực thi, hoặc có thể xóa hay set bằng cách ghi một giá trị logic vào vị trí của nó.

Bit 3 - OCF3B: Timer/Counter3, Output Compare B Match Flag: Cờ này được set ngay sau khi giá trị bộ đếm (TCNT3) bằng với giá trị thanh ghi OCR3B (Output Compare Register B). Chú ý là một so sánh cưỡng bức (FOC3B) sẽ không set cờ này. Cờ OCF3B sẽ tự động xóa khi ngắt tương ứng được thực thi, hoặc có thể xóa hay set bằng cách ghi một giá trị logic vào vị trí của nó.

Bit 2 - TOV3: Timer/Counter3, Overflow Flag: Việc thiết lập cờ này phụ thuộc vào thiết lập của các bit WGMn3:0, trong chế độ bình thường và CTC cờ TOV3 được set

khi bộ định thời tràn. Xem lại bảng 52 và mục "Các chế độ thực thi" để biết các trường hợp khác.

Bit 1 - OCF3C: Timer/Counter3, Output Compare C Match Flag: Cờ này được set ngay sau khi giá trị bộ đếm (TCNT3) bằng với giá trị thanh ghi OCR3C (Output Compare Register C). Chú ý là một so sánh cưỡng bức (FOC3C) sẽ không set cờ này. Cờ OCF3C sẽ tự động xóa khi ngắt tương ứng được thực thi, hoặc có thể xóa hay set bằng cách ghi một giá trị logic vào vị trí của nó.

Bit 0 - OCF1C: Timer/Counter1, Output Compare C Match Flag: Cờ này được set ngay sau khi giá trị bộ đếm (TCNT1) bằng với giá trị thanh ghi OCR1C (Output Compare Register C). Chú ý là một so sánh cưỡng bức (FOC1C) sẽ không set cờ này. Cờ OCF1C sẽ tự động xóa khi ngắt tương ứng được thực thi, hoặc có thể xóa hay set bằng cách ghi một giá trị logic vào vị trí của nó.

13. Thanh Ghi Special Function IO Register-SFIOR

Bit	7	6	5	4	3	2	1	0	
	TSM	-	-	-	ACME	PUD	PSR0	PSR321	SFIOR
Read/Write	R/W	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit 7 - TSM: Timer/Counter Synchronization Mode

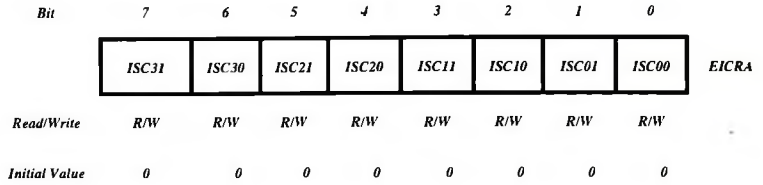
Bit 0 - PSR321: Prescaler Reset Timer/Counter3, Timer/Counter2, and Timer/Counter1

Bit 7 - TSM: Timer/Counter Synchronization Mode: Ghi bit này thành 1 sẽ kích hoạt chế độ "Đồng bộ bộ định thời". Trong chế độ này giá trị ghi vào hai bit PSR0 và PSR321 được giữ, vì thế nó giữ cho tín hiệu reset của bộ chia trước (prescaler(8)) tương ứng được xác nhận (do đó bộ chia trước prescaler vẫn ở trạng thái Reset). Điều này để chắc chắn là các bộ Timer/couter tương ứng được tạm dừng để có thể được cấu hình với giá trị như nhau mà không làm hỏng các cấu hình sẵn có khác. Khi TMS là 0 thì các bit PSR0 và PSR321 được xóa bởi phần cứng và các bộ định thời (1,2,3) bắt đầu đếm đồng hời. (Xem thêm mục : Chế Độ Đồng Bộ Bộ Định Thời).

Bit 0 - PSR321: Prescaler Reset Timer/Counter3, Timer/Counter2, and Timer/Counter1: Khi bit này là 1 thì bộ chia trước (prescaler) của ba bộ định thời 1,2,3 lược reset. Bit PSR321 được xóa bởi phần cứng ngoại trừ trường hợp bit TSM được set. Chú ý là ba bộ định thời 1, 2, 3 cùng chia sẽ một bộ chia trước (prescaler) nên việc reset bộ chia trước (prescaler) sẽ tác động lên cả ba bộ định thời này.

d. Các thanh ghi điều khiển ngắt

1. Thanh ghi External Interrupt Control Register A-EICRA



Bits 7.0 - ISC31, ISC30-ISC00, ISC00: External Interrupt 3 - 0 Sense Control Bits

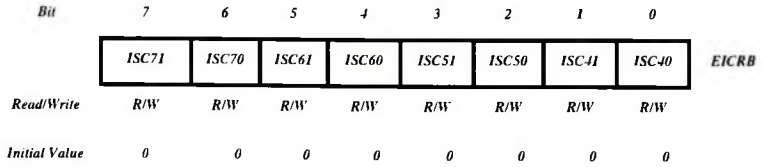
Tám bit của thanh ghi EICRA sẽ điều khiển kiểu bắt mẫu cho 4 ngắt INT3, INT2, INT1, INT0. Quy định cụ thể được thể hiện trong Bảng 48.

ISCn1	ISCn0	Kiểu bắt mẫu
0	0	Mức thấp sẽ tạo yêu cầu ngắt
0	1	Dự trữ
1	0	Cạnh xuống (Falling) sẽ tạo yêu cầu ngắt
1	1	Cạnh lên (Rising) sẽ tạo yêu cầu ngắt

n = 3, 2, 1, 0

Bảng 48. Điều khiển kiểu bắt mẫu ngắt

2. Thanh Ghi External Interrupt Control Register B-EICRB



Bits 7.0 - ISC71, ISC70 - ISC41, ISC40: External Interrupt 7 - 4 Sense Control Bits.

Tám bit của thanh ghi EICRA sẽ điều khiển kiểu bắt mẫu cho 4 ngắt INT7, INT6. INT5, INT4. Quy định cụ thể được thể hiện trong Bảng 50.

ISCn1	ISCn0	Kiểu bắt mẫu
0	0	Mức thấp sẽ tạo yêu cầu ngắt
0	1	Bất cứ sự thay đổi mức logic nào ở chân INTn sẽ tao ra một yêu cầu ngắt
1	0	Cạnh xuống (Falling) sẽ tạo yêu cầu ngắt
1	1	Cạnh lên (Rising) sẽ tạo yêu cầu ngắt

n = 7,6, 5, 4

Bảng 50. Điều khiển kiểu bắt mẫu ngắt

3. Thanh Ghi External Interrupt Mask Register-EIMSK

Bit	7	6	5	4	3	2	1	0	
	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0	EIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bits 7.0 - INT7 - INT0: External Interrupt Request 7 - 0 Enable : Khi cho hép ngắt toàn cục (set bit I trong thanh ghi SREG thành 1) thì các ngắt vẫn chưa thể hực thi, để ngắt có thể thực thi ta cần phải cho phép nó, 8 bit trong thanh ghi EIMSK ẽ quyết định 8 ngắt ngoài tương ứng (từ INT7 ...INT0) có được cho phép hay không. hi một trong số 8 bit (từ INT7 ...INT0) được set thành 1 và ngắt toàn cục được cho hép thì ngắt ngoài tương ứng được cho phép. Còn tín hiệu ngắt là mức hay cạnh sẽ o các thanh ghi EICRA và EICRB (nêu ở trên) qui định. Kích hoạt bất cứ chân (Pin) ào trong 8 chân của ngắt ngoài cũng tạo ra yêu cầu ngắt ngay cả khi chân được thiết ip thành ngõ ra.

4. Thanh Ghi External Interrupt Flag Register-EIFR

Bit	7	6	5	4	3	2	1	0	
	INTF7	INTF6	INTF5	INTF4	INTF3	INTF2	INTF1	INTF0	EIFR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
nitial Value	0	0	0	0	0	0	0	0	

Bits 7.0 - INTF7 - INTF0: External Interrupt Flags 7...0 : Đây là tám cờ ngắt ơng ứng với tám ngắt ngoài INT7..INT0. Khi có tín hiệu yêu cầu ngắt ngoài thì cờ ngắt

tương ứng sẽ được set thành 1, nếu ngắt tương ứng được cho phép thì MCU sẽ nhảy tới bảng véc tơ ngắt, cờ ngắt sẽ được xóa khi chương trình phục vụ ngắt (ISR) được thực thi. Ngoài ra ta cũng có set hay xóa cờ ngắt bằng cách ghi trực tiếp một giá trị logic vào nó.

5. Thanh Ghi MCU Control Register- MCUCR

Bit	7	6	5	4	3	2	1	0	
	SRE	SRW10	SE	SM1	SM0	SM2	IVSEL	IVCE	MCUR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Trong phần này ta chỉ quan tâm tới hai bit là: IVCE (Interrupt Vector Select) và bit IVSEL (Interrupt Vector Change Enable) của thanh ghi MCUCR. Bit này liên quan đến việc thiết lập vị trí bảng véc tơ ngắt.

Bit 1 - IVSEL: Interrupt Vector Select: Khi bit này là 0 vị trí của bảng véc tơ ngắt được đặt ở phần đầu bộ nhớ chương trình. Khi bit này là 1 bảng véc tơ ngắt được di chuyển tới phần đầu của vùng nhớ Boot Loader.

Bit 0 - IVCE: Interrupt Vector Change Enable : Bit này phải được ghi thành 1 để cho phép thay đổi bit IVSEL. Bit IVCE được xóa sau 4 chu kì máy sau khi nó được set hay bit IVSEL được ghi. Trong lúc bit ICVE đang set các ngắt sẽ bị cấm cho tới khi bit IVSEL được ghi, nếu bit IVSEL không được ghi thì các ngắt vẫn bị cấm trong 4 cho kì máy liên tiếp (sau 4 chu kì máy thì bit IVCE sẽ tự động bị xóa nên các ngắt được cho phép trở lại).

e. Các thanh ghi của bộ ADC

1. Thanh ghi ADC Multiplexer Selection- ADMUX

Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit 7:6 - REFS1:0: Reference Selection Bits

Bit 5 - ADLAR: ADC Left Adjust Result

Bits 4:0 - MUX4:0: Analog Channel and Gain Selection Bits

Bit 7:6 - REFS1:0: Reference Selection Bits: hai bit này dùng để lựa chọn điện thế tham chiếu là một trong 3 nguồn: AVCC, Điện thế tham chiếu nội 2.56v và VREF như bảng 97. Nếu chọn điện thế VREF thì các tùy chọn còn lại không được sử dụng để tránh bị ngắn mạch, điều này có nghĩa là nếu ta chọn điện thế tham chiếu là VREF rồi, thì trong suốt quá trình hoạt động của bộ ADC ta không được lựa chọn điện thế tham chiếu khác, vì nếu không, nguồn điện thế VREF bên ngoài do chưa được tháo đi sẽ làm hỏng chip do ngắn mạch.

Table 97. Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal Vref turned off
0	1	AVCC with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 2.56V Voltage Reference with external capacitor at AREF pin

Bảng 97. Lựa chọn điện thế tham chiếu

Table 98. Input Channel and Gain Selections

MUX4..0	Single Ended Input	Positive Differential Input	Negative Differential Input	Gain
00000	ADC0	N/A		
00001	ADC1			
00010	ADC2			
00011	ADC3			
00100	ADC4			
00101	ADC5			
00110	ADC6			
00111	ADC7			
01000		ADC0	ADC0	10x
01001		ADC1	ADC0	10x

Bảng 98. Lựa chọn kiểu ngõ vào và độ lợi

01010[1]		ADC0	ADC0	200x
01011		ADC1	ADC0	200x
01100		ADC2	ADC2	10x
01101		ADC3	ADC2	10x
01110		ADC2	ADC2	200x
01111		ADC3	ADC2	200x
10000		ADC0	ADC1	1x
10001		ADC1	ADC1	1x
10010	N/A	ADC2	ADC1	1x
10011		ADC3	ADC1	1x
10100		ADC4	ADC1	1x
10101		ADC5	ADC1	1x
10110		ADC6	ADC1	1x
10111		ADC7	ADC1	1x
11000		ADC0	ADC2	1x
11001		ADC1	ADC2	1x
11010		ADC2	ADC2	1x
11011		ADC3	ADC2	1x
11100		ADC4	ADC2	1x
11101		ADC5	ADC2	1x
11110	1.23V (V_{BG})	N/A		
11111	0V (GND)			

Bảng 98. Lựa chọn kiểu ngõ vào và độ lợi (tiếp theo)

Bit 5 - ADLAR: ADC Left Adjust Result: Bit này lựa chọn cách bố trí dữ liệu trong hai thanh ghi dữ liệu ADCH và ADCL. Xem phần mô tả hai thanh ghi dữ liệu ADCH và ADCL để biết chi tiết.

Bits 4:0 - MUX4:0: Analog Channel and Gain Selection Bits: Các bit này lựa chọn kiểu ngõ vào (đơn hay vi sai) và độ lợi, xem bảng 98.

2. Thanh ghi ADC Control and Status Register A- ADCSRA

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit 7 - ADEN: ADC Enable

Bit 6 - ADSC: ADC Start Conversion

Bit 5 - ADFR: ADC Free Running Select

Bit 4 - ADIF: ADC Interrupt Flag

Bit 3 - ADIE: ADC Interrupt Enable

Bits 2:0 - ADPS2:0: ADC Prescaler Select Bits

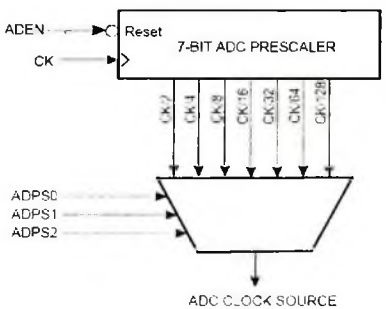
Bit 7 - ADEN: ADC Enable: Bit này là 1 sẽ cho phép bộ ADC hoạt động, ngược lại, sẽ ngừng bộ ADC ngay cả khi nó đang trong quá trình biến đổi.

Bit 6 - ADSC: ADC Start Conversion: Ghi bit này thành 1 để bắt đầu quá trình chuyển đổi. Trong chế độ chuyển đổi từng bước, sau mỗi lần chuyển đổi hoàn thành bit này bị xóa về 0, ta phải set lại bit này để bắt đầu một biến đổi tiếp theo. Trong chế độ chuyển đổi liên tục, ta chỉ cần set bit này một lần.

Bit 5 - ADFR: ADC Free Running Select: Set bit này lên 1 để lựa chọn chế độ hoạt động biến đổi liên tục. Bit này là 0 sẽ cho phép chế độ biến đổi từng bước.

Bit 4 - ADIF: ADC Interrupt Flag: Bit này sẽ được set thành 1 khi một chu trình biến đổi ADC hoàn thành, bit này được xóa bởi phần cứng khi trình phục vụ ngắt tương ứng được thực thi. Chú ý là khi ta chỉnh sửa thanh ghi ADCSRA (như dùng các lệnh CBI, SBI) thì bit này sẽ bị xóa. Vì vậy, để xóa bit này bởi phần mềm, ta chỉ cần ghi giá trị 1 vào nó.

Bit 3 - ADIE: ADC Interrupt Enable: Bit này cho phép ngắt ADC, khi bit ADIE (cho phép ngắt ADC) và bit I (cho phép ngắt toàn cục) trong thanh ghi SREG được set lên 1 sẽ cho phép ngắt ADC hoạt động.



Hình 109. Nguồn clock ADC

Bits 2:0 - ADPS2:0: ADC Prescaler Select Bits: Vì tần số clock ADC được lấy từ xung clock hệ thống (hình 109), nên các bit ADPS2:0 sẽ cho phép chia xung clock hệ thống với các hệ số xác định (bảng 99) trước khi đưa vào nguồn clock ADC. Với độ phân giải 10 bit, tần số clock ADC khoảng từ 50 – 200 KHz, nên tùy theo tần số clock hệ thống mà ta lựa chọn hệ số chia thích hợp.

Table 99. ADC Prescaler Selections

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

Bảng 99. Lựa chọn các hệ số chia cho nguồn clock ADC

c. Thanh ghi ADC Data Register-ADCL and ADCH

Đây là hai thanh ghi chứa kết quả ADC, tùy theo thiết lập của bit ADLAR (trong thanh ghi ADMUX) mà 10 bit dữ liệu ADC có thể được bố trí về phía phải hay trái của hai thanh ghi ADCH và ADCL, cụ thể như sau:

ADLAR = 0

Bit	15	14	13	12	11	10	9	8	
	-	-	-	-	-	-	ADC9	ADC8	ADCH
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	

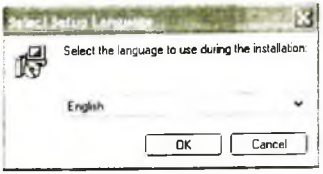
ASDLAR = 1

Bit	15	14	13	12	11	10	9	8	
	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
	ADC1	ADC0	-	-	-	-	-	-	ADCL
	7	6	5	4	3	2	1	0	

II. HƯỚNG DẪN CÀI ĐẶT VÀ SỬ DỤNG PHẦN MỀM CODE VISION AVR

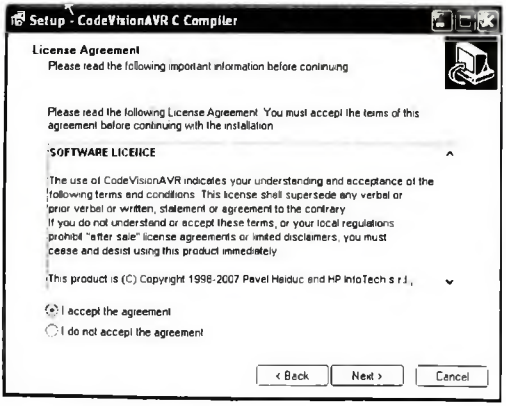
1. Hướng dẫn cài đặt Code Vision AVR

Mở đĩa Code Vision AVR . Nháy đúp vào file setup.exe, Window sẽ có thông báo sau:

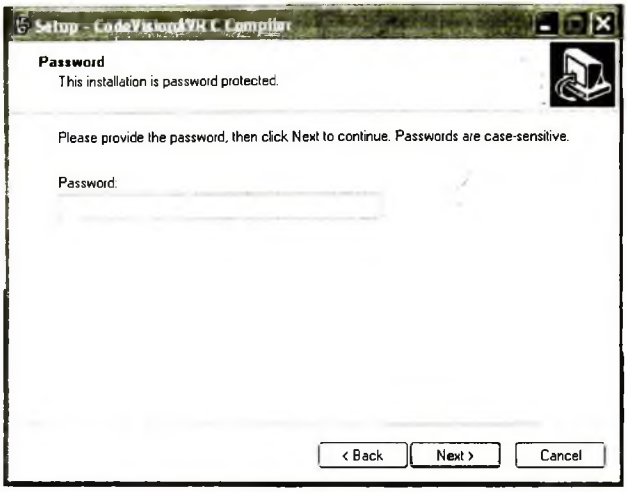


Nhấn ***OK*** để chọn ngôn ngữ. Window sẽ thông báo tiếp như sau:

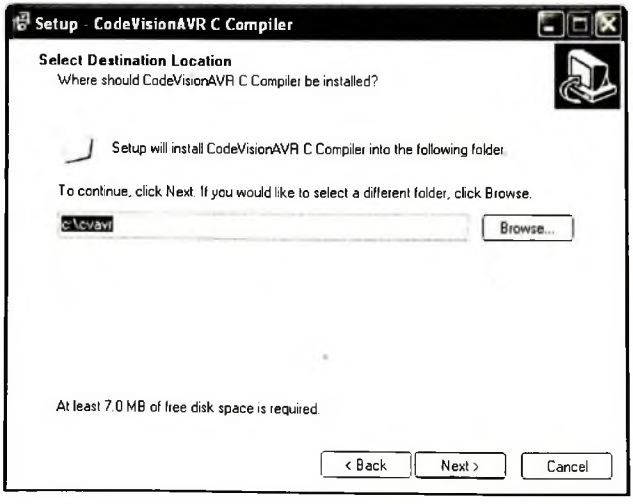
Nhấn ***next*** để tiếp tục.



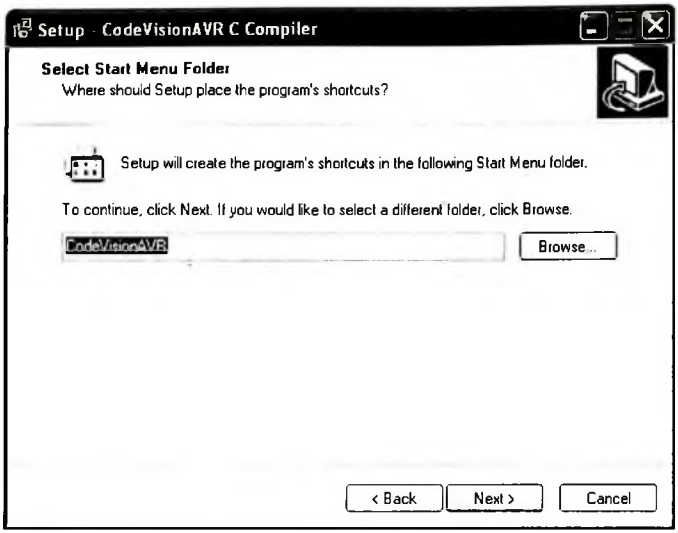
Chọn ***I accept the agreement***, sau đó nhấn ***next*** để tiếp tục. Window có thông báo tiếp như sau:



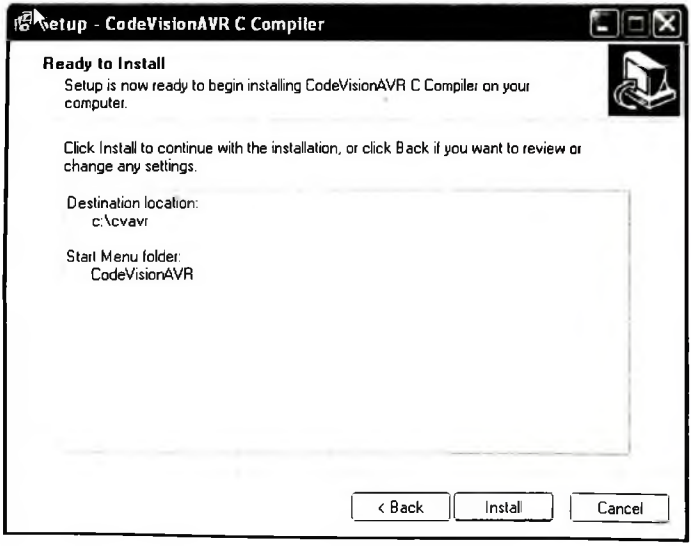
Điền Password bạn được cấp vào đây. Nhấn ***next*** để tiếp tục.



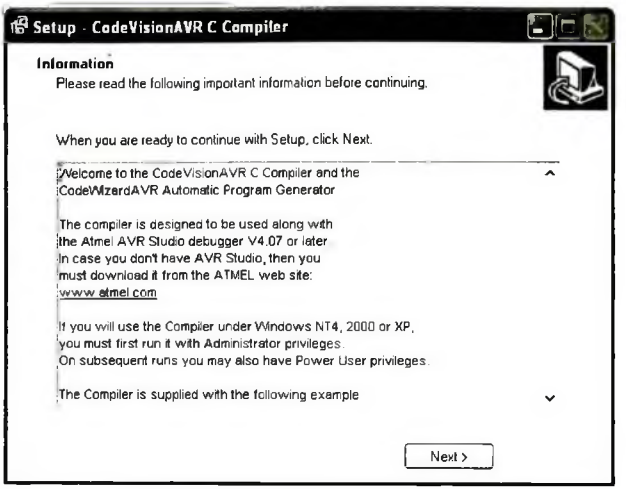
Nhấn ***next*** để tiếp tục.



Nhấn ***next*** để tiếp tục.



Nhấn ***Install*** để tiếp tục.

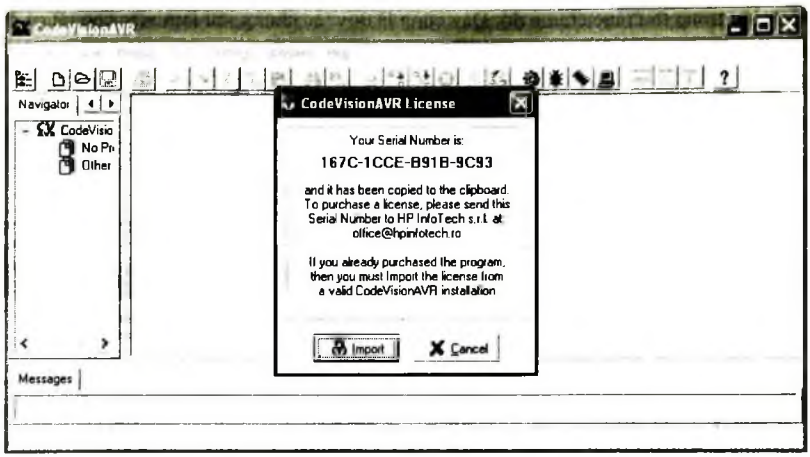


Nhấn ***next*** để tiếp tục.

Nhấn ***Finish*** để kết thúc.

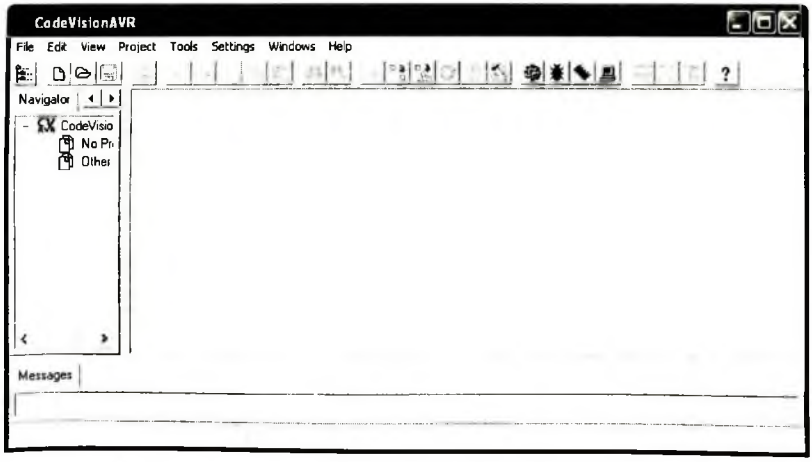
Sau khi cài đặt xong ta tiến hành Crack như sau:

Vào thanh công cụ ***Start/Programs/Code Vision AVR***. Window sẽ có thông báo như sau:



Quay trở lại với giao diện của Code Vision AVR, click chuột vào ***Import*** chọn đường dẫn đến file License, nhấn OK để hoàn tất việc cài đặt.

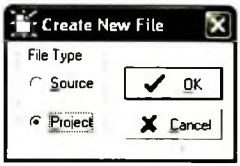
Sau khi cài đặt xong thì giao diện chính của Code Vision AVR như sau:



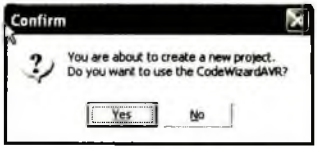
2. Tạo một Project trong Code Vision AVR

Ta cần chú ý khi tạo một Project mới thì ta nên tạo một thư mục mới có tên liên quan đến dự án định làm, rồi lưu tất cả các file liên quan vào trong thư mục đó. Khi biên dịch, Code Vision AVR sẽ tạo ra rất nhiều file liên quan khác nhau các file này sẽ tự động lưu trong thư mục chung đó. Đây cũng là quy tắc chung khi làm việc cho tất cả các phần mềm lập trình.

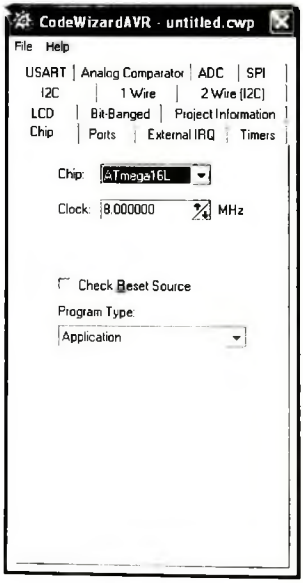
Từ giao diện chương trình chọn *New*, có thông báo sau:



Chọn Project, nhấn OK (file tạo ra mà một dự án mới).

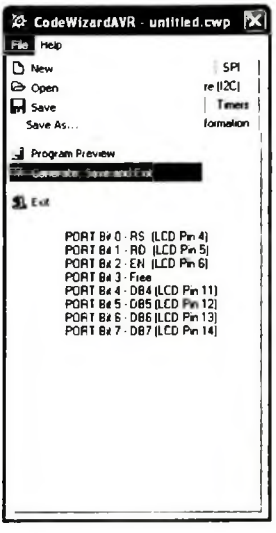


Nhấn ***Yes*** để tiếp tục, cửa sổ CodeWizardAVR hiện ra.

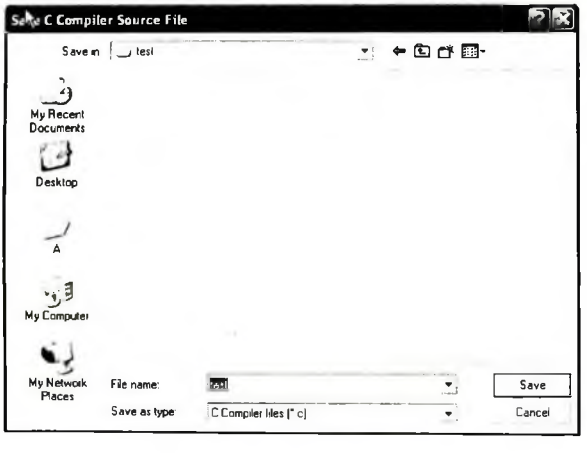


Ta phải khai báo các cấu hình, các thư viện sử dụng co dự án, như loại chip, tần số thạch anh, các cổng vào ra, giao tiếp I2C, giao tiếp LCD, giao tiếp máy tính, time... Sau khi khai báo xong cấu hình cho dự án ta lưu lại như sau:

Chọn ***file/Generate, Save and Exit***



Tạo một forder mới và lưu cả 3 định dạng file vào trong đó, *test.c*, *test.prj*, *test.cwp*.



Sau khi save xong thì trong *file test.c* có sẵn các hàm mà ta đã đặt cấu hình trước, công việc của người lập trình là viết các đoạn code vào vòng while{ }.

3. Cài đặt và sử dụng AVR 910 USB Programmer.

AVR 910 USB Programmer dùng để nạp chương trình sau khi đã dịch ra file hex xuống AVR, sử dụng kết nối USB 2.0 và Code Vision AVR.

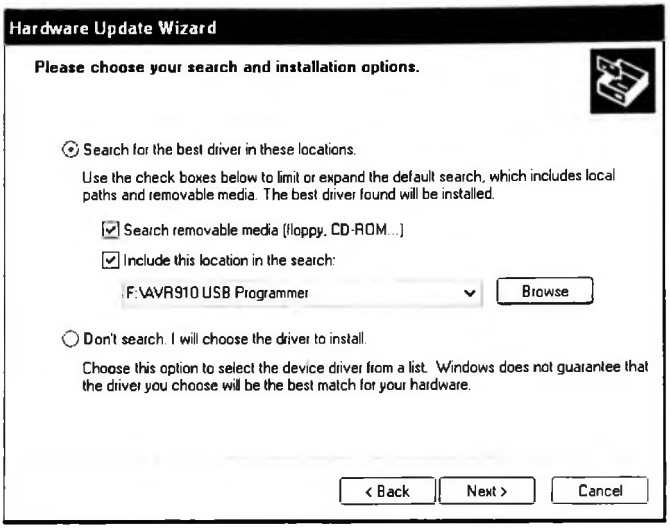
3.1. Cài đặt

Kết nối mạch nạp với máy tính thông qua cổng USB máy sẽ có thông báo sau:

Nhấn ***Next***

Tìm đường dẫn trong file cài đặt trên đĩa CD.

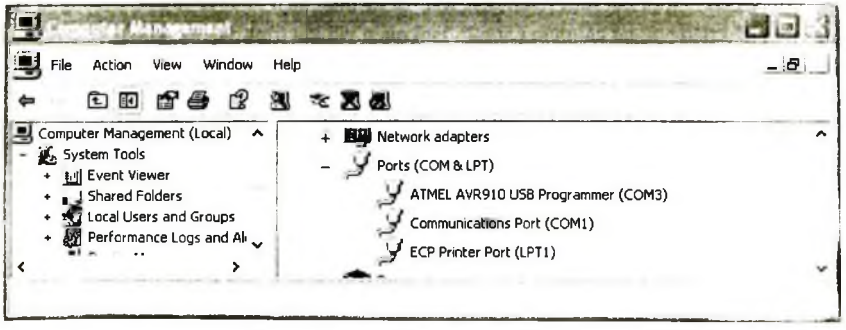
Nhấn *Next*.



Chờ cho Window tìm kiếm và cài đặt·

Chọn ***Finish*** để kết thúc cài đặt.

Để kiểm tra lại quá trình cài đặt ta làm như sau: Quay trở về Desktop nhấn phải chuột vào biểu tượng My Computer chọn Magane, chọn Device Magane.



Phần II

CÁC BÀI THỰC HÀNH CƠ BẢN

BÀI THỰC HÀNH SỐ 1
NHÁY LED

I. MỤC ĐÍCH

Học xong bài này học sinh cần phải nắm rõ các nội dung sau:

- Cách tạo một Project mới trong Code Vision AVR.
- Hiểu được các khai báo các thư viện dùng trong chương trình.

  VD: *include <mega16.h>* hay *include<delay.h>*...
- Hiểu được cách đặt cấu hình vào ra cho các PORT.
- Hiểu được cách xuất dữ liệu ra các PORT.
- Dịch và nạp chương trình cho chíp.

II. TÓM TẮT LÍ THUYẾT

- Nháy Led là một bài thực hành đơn giản và dễ thực hiện nhất vì vậy nó thường là sự lựa chọn đầu tiên cho tất cả những ai mới làm quen với lập trình vi điều khiển.

- Các bước của chương trình nháy Led như sau:

  B1: Chuyển chân PORTC.0 lên mức 1

  B2: Gọi hàm delay.

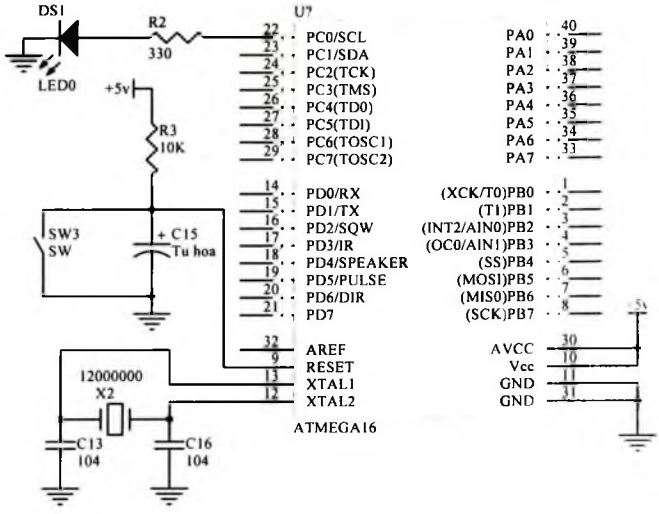
  B3: Chuyển chân PORTC.0 xuống mức 0.

  B3: Gọi hàm delay.

Lặp lại quá trình.

Sơ đồ kết nối:

Sơ đồ nguyên lý như sau:



Hình 1.1

III. CÁC BƯỚC TIẾN HÀNH

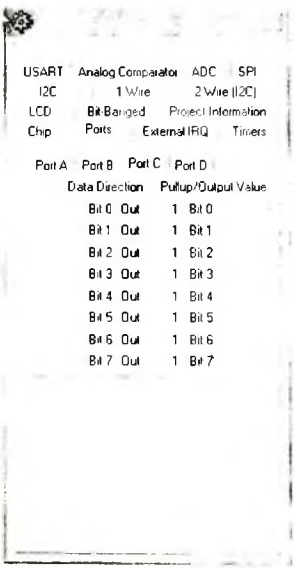
Khởi động phần mềm lập trình Code Vision AVR

- Tạo một Project đặt tên là nháy led lưu vào trong thư mục ***nhayled.***

- Trong phần khai báo ta chọn các chân của PortC là các chân OUT có trở kéo lên 5V. Lần lượt lưu các file vào trong Forder chứa Project.

- Mở file ***nhayled.c*** ta thấy rất nhiều các khai báo, các hàm, ta chỉ viết code vào đoạn code vào vòng while như hình dưới.

```
while (1)
      {
      // Place your code here

      };
}
```



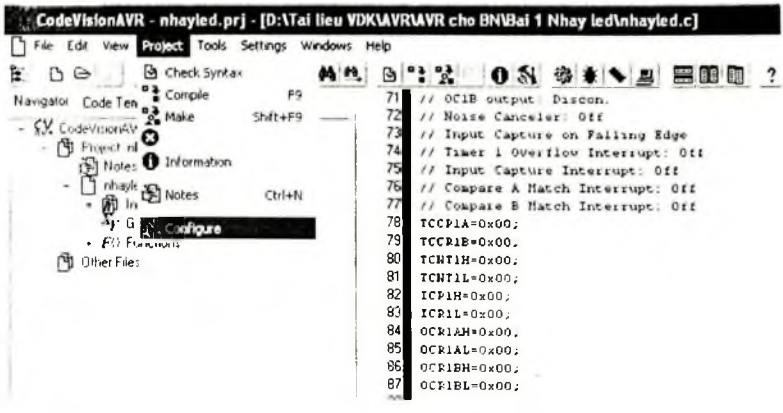
Soạn thảo chương trình.

Phần khai báo thư viện cho project

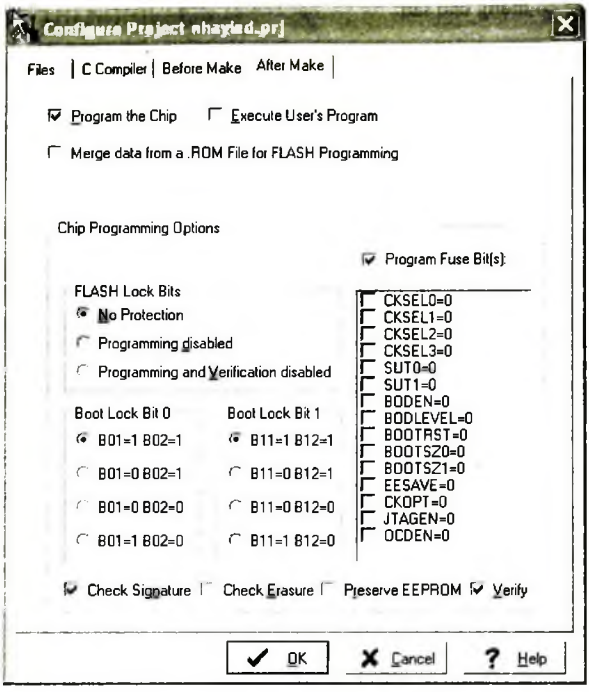
```
#include <mega16.h>
#include <delay.h>
// Declare your global variables here
while (1) {
        PORTC.0=1;
        delay_ms(500);
        PORTC.0=0;
        delay_ms(500);
        // Place your code here
        };
```

Biên dịch và nạp chương trình.

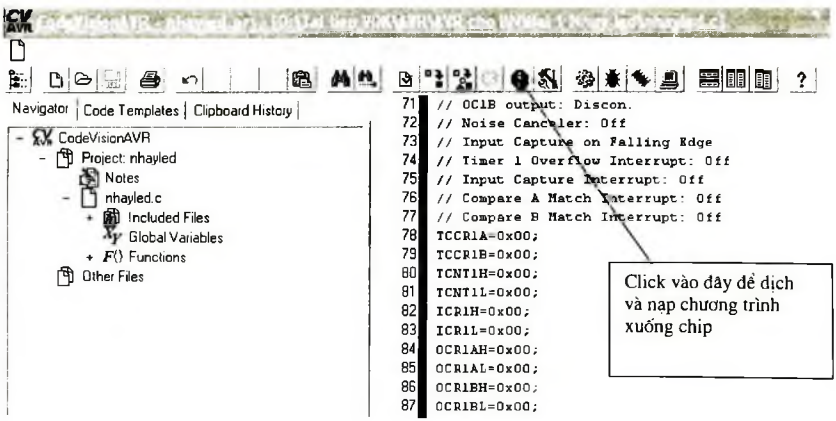
- Nối mạch nạp AVR 910 USB Programmer với Main chính AVR qua đường ISP.
- Trên thanh công cụ



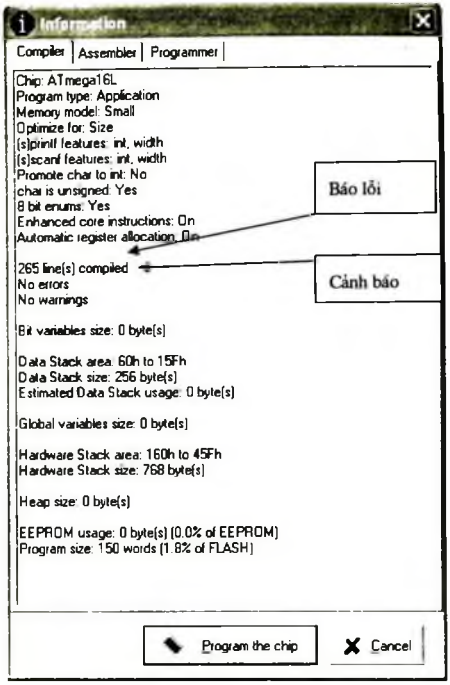
Một cửa sổ mới hiện ra, ta chọn vào Tab ***After Make***, đánh dấu chọn vào ô ***Program the chip***, nhấn OK để chấp nhận.



Quay trở lại với giao diện làm việc, ta tiến hành dịch và nạp chương trình.



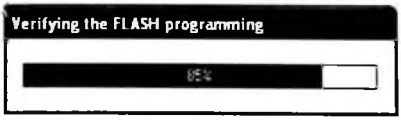
- Window sẽ có thông báo sau:



- Nếu chương trình có lỗi thì sẽ được thông báo ở đây.

- Click vào Program the chip để nạp chương trình.

- Nếu quá trình dịch thành công thì window sẽ có thông báo như sau.



Hoạt động thử.

- Ghép môđun Led đơn với Main chính qua PORTC, (chú ý địa chỉ các chân vào ra).

- Theo dõi sự hoạt động của mạch.

IV. BÁO CÁO THÍ NGHIỆM

Số thứ tự và tên bài.

Mục đích thí nghiệm.

Chương trình đã soạn thảo để thí nghiệm.

Kết quả thí nghiệm.

Nhận xét và kết luận.

V. CÂU HỎI KIỂM TRA

Vẽ lưu đồ thuật toán của chương trình và giải thích.

Nếu muốn thay đổi tốc độ nháy của Led thì ta phải làm như thế nào?

BÀI THỰC HÀNH SỐ 2
ĐIỀU KHIỂN VÀO RA

Bài tập ứng dụng:

Lập trình kiểm tra một tín hiệu vào và bật tắt một con Led. Ví dụ PORTA.0 nối với nút Start, PORTA.1 nối với nút Stop, điều khiển bật tắt một con LED ở PORTC.0.

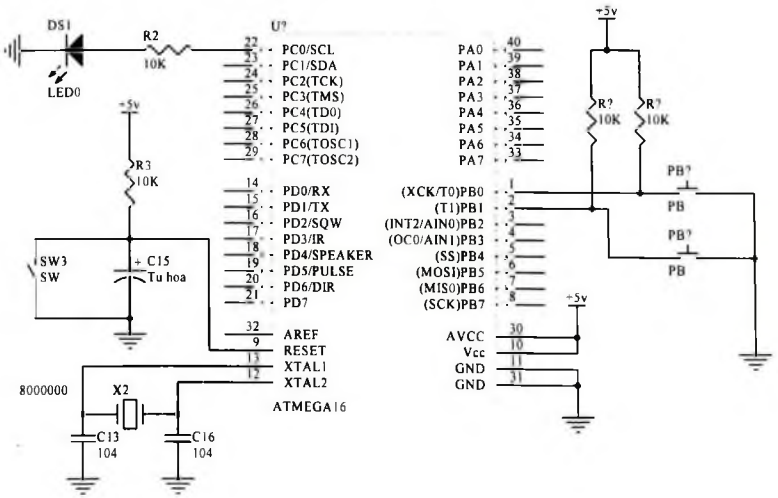
I. MỤC ĐÍCH

Học xong bài này học sinh cần nắm được các nội dung sau:

- Hiểu được cách đặt cấu hình vào ra cho các PORT.
- Hiểu cách kiểm tra tín hiệu vào tại một chân.
- Hiểu được cấu trúc và hoạt động của lệnh "if".

II. TÓM TẮT LÍ THUYẾT

- Sơ đồ nguyên lý.



Mô tả: Bình thường các chân PORTB.0, PORTB.1 ở mức cao, khi nhấn nút thì điện p tại các chân này chuyển xuống mức thấp và nhiệm vụ của chương trình là kiểm tra em lúc nào thì điện áp tại hai chân này xuống mức 0.

II. CÁC BƯỚC TIẾN HÀNH

Khởi động phần mềm lập trình Code Vision AVR.

Tạo một dự án mới và đặt tên cho dự án.

- Khai báo PORTB là PORT nhận dữ liệu, PORTC là PORT xuất dữ liệu.
- Soạn thảo chương trình theo nội dung sau:

```
while (1) {
if (PINA.0==0)                    // Để kiểm tra tín hiệu vào thì ta dùng PIN.
        {PORTC.0=0;
        delay_us(5);  //Chờ 5us chống nẩy phím.
        }
if (PINA.1==0)
        { PORTC.0=1;
        delay_us(5);
        } }}
```

Hoạt động thử.

- Tắt công tắc nguồn.
- Ghép môđun bàn phím với Main chính qua PORTB, môđun Led đơn qua PORTC, (chú ý địa chỉ các chân vào ra).
- Bật công tắc nguồn cho Main.
- Nhấn đồng thời SHIFT+F9 để dịch và nạp chương trình xuống Main.
- Nhấn nút Start nối với PORTA.0.
- Nhấn nút Stop nối với PORTA.1.
- Theo dõi sự hoạt động của Led.

IV. BÁO CÁO THÍ NGHIỆM

Số thứ tự và tên bài.

Mục đích thí nghiệm

Chương trình đã soạn thảo để thí nghiệm

Kết quả thí nghiệm:

Nhận xét và kết luận.

V. CÂU HỎI KIỂM TRA

1. Giải thích hoạt động của chương trình. Viết lưu đồ thuật toán của chương trình.

BÀI THỰC HÀNH SỐ 3
ĐIỀU KHIỂN ĐỘNG CƠ MỘT CHIỀU

Bài tập ứng dụng:

Lập trình điều khiển động cơ hoạt động như sau: Nhấn nút PA0, động cơ chạy thuận, nhấn nút PA1 thì động cơ chạy ngược, nhấn nút PA2 thì động cơ chạy nhanh, nhấn nút PA3 thì động cơ chạy chậm, nhấn nút PA4 thì động cơ dừng.

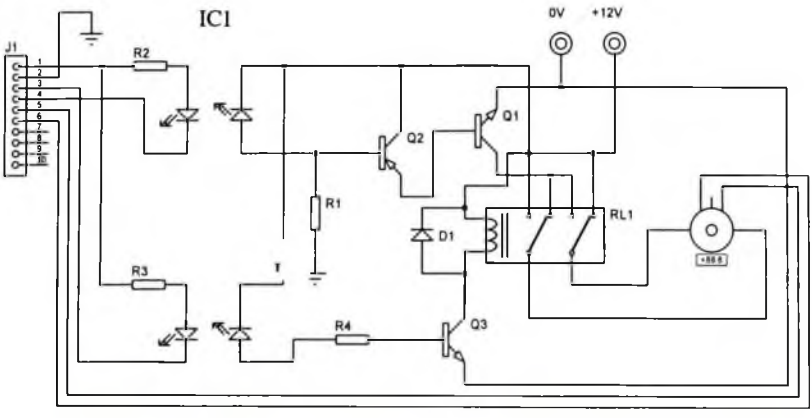
I. MỤC ĐÍCH

Học xong bài này học sinh cần nắm được các nội dung sau:

- Hiểu được cách điều khiển động cơ trên môđun động cơ DC.
- Hiểu cách điều khiển động cơ bằng tín hiệu PWM (độ rộng xung).
- Hiểu được cấu trúc và cách hoạt động của lệnh "SWITCH".
- Hiểu được cách dùng các lệnh toán học trong chương trình.

II. TÓM TẮT LÍ THUYẾT

- Sơ đồ nguyên lý (hình 2-1).



Hình 2-1

III. CÁC BƯỚC TIẾN HÀNH

Khởi động phần mềm lập trình Code Vision AVR.

Tạo một dự án mới và đặt tên cho dự án dkdcdc.

- Soạn thảo chương trình theo nội dung sau:

```
#include <mega16.h>
#include <delay.h>
unsigned char i=0,j=3,k=3;
void right();
void left();
void main(void) {
    PORTA=0xFF;
    DDRA=0x00;
    PORTB=0x00;
    DDRB=0x00;
    PORTC=0x00;
    DDRC=0x00;
    PORTD=0xFF;
    DDRD=0xFF;
    TCCR0=0x00;
    TCNT0=0x00;
    OCR0=0x00;
    TCCR1A=0x00;
    TCCR1B=0x00;
    TCNT1H=0x00;
    TCNT1L=0x00;
    ICR1H=0x00;
    ICR1L=0x00;
    OCR1AH=0x00;
    OCR1AL=0x00;
```

```
        OCR1BH=0x00;
        OCR1BL=0x00;
        ASSR=0x00;
        TCCR2=0x00;
        TCNT2=0x00;
        OCR2=0x00;
        MCUCR=0x00;
        MCUCSR=0x00;
        TIMSK=0x00;
        ACSR=0x80;
        SFIOR=0x00;
while (1){
 if (PINA.0==0) i=1;
if (PINA.1==0) i=2;
 switch(i) {
          case 1: right(); break;
 case 2: left(); break;}
 if (PINA.2==0)
 { j=5;
 k=1;
 }
 if (PINA.3==0)
 {j=1;
 k=5;
 }
        if (PINA.4==0){
                PORTD.0=1;
                i=0;
                }
```

```
        }
                }
    void right(){
     PORTD.1=0;
    PORTD.0=0;
    delay_ms(j);
    PORTD.0=1;
    delay_ms(k);
            }
    void left(){
    PORTD.1=1;
    PORTD.0=0;
    delay_ms(j);
    PORTD.0=1;
     delay_ms(k);
            }
```

Hoạt động thử.

- Tắt công tắc nguồn.
- Ghép môđun bàn phím với Main chính qua PORTA, môđun động cơ DC với PORTD, (chú ý địa chỉ các chân vào ra).
- Bật công tắc nguồn cho Main.
- Nhấn đồng thời SHIFT + F9 để dịch và nạp chương trình xuống Main.
- Nhấn nút nối với PORTA.0 để chạy thuận.
- Nhấn nút nối với PORTA.1 để chạy ngược.
- Nhấn nút nối với PORTA.2 để chạy nhanh.
- Nhấn nút nối với PORTA.3 để chạy chậm.
- Nhấn nút nối với PORTA.4 để dừng động cơ.
- Theo dõi sự hoạt động của Led.

IV. BÁO CÁO THÍ NGHIỆM

Số thứ tự và tên bài.

Mục đích thí nghiệm

Chương trình đã soạn thảo để thí nghiệm

Kết quả thí nghiệm:

Nhận xét và kết luận.

V. CÂU HỎI KIỂM TRA

1. Giải thích hoạt động của chương trình. Viết lưu đồ thuật toán của chương trình.

BÀI THỰC HÀNH SỐ 4
ĐIỀU KHIỂN ĐỘNG CƠ BƯỚC

Bài tập ứng dụng:

Lập trình đảo chiều quay và thay đổi tốc độ động cơ bước.

I. MỤC ĐÍCH

Học xong bài này học sinh cần nắm được các nội dung sau:

- Hiểu được cấu tạo, nguyên lí hoạt động của động cơ bước.

- Hiểu được cách khai báo và sử dụng các biến, cách khai báo và cách gọi các chương trình con.

- Hiểu được cấu trúc và cách dùng lệnh "*Switch*".

- Hiểu được phương pháp tạo xung điều khiển động cơ bước bằng phần mềm.

- Xây dựng được chương trình đảo chiều quay và thay đổi tốc độ động cơ bước.

II. TÓM TẮT LÍ THUYẾT

Hiện nay trong một số hệ thống điều khiển tự động có sử dụng một loại động cơ đặc biệt chạy bằng xung điện. Khác với động cơ thông thường là động cơ này không quay liên tục mà nó chỉ dịch chuyển từng "bước" vì vậy còn gọi là động cơ bước. Ưu điểm nổi bật của nó là có thể điều khiển, khống chế để trục động cơ bước quay một góc $\propto$ nào đó.

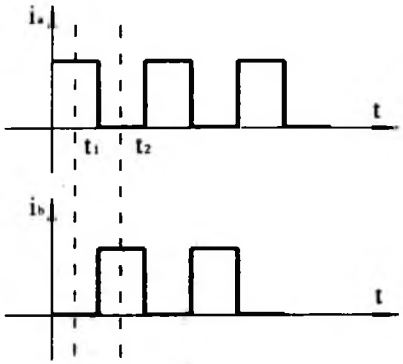
Hình 4-1

Sơ lược cấu tạo:

Động cơ bước rất có cấu tạo giống động cơ điện một chiều nhưng có điểm khác biệt là không có cổ góp và chổi than, rô to không có cuộn kích từ. Thông thường, động cơ bước gồm một rô to sắt mềm trên đó xẻ nhiều rãnh tạo thành các răng. Stato cũng được xẻ rãnh tạo thành các cực từ. Trên các cực từ có quấn dây (Xem hình 4-1).

Nguyên lí làm việc của động cơ bước:

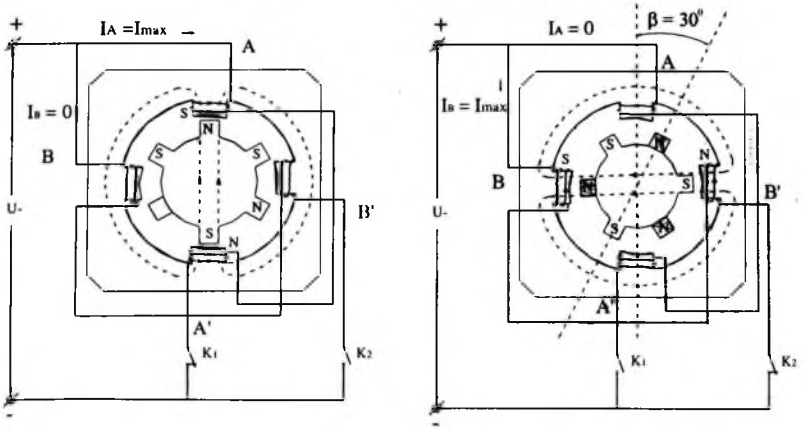
Xét một động cơ bước 2 pha đơn giản stato có 2 cực, rô to kiểu nam châm vĩnh cửu gồm 6 cực đặt xen kẽ lẫn nhau. Góc hợp bởi 2 cực từ kề nhau trên rô to là 60^0. Dòng điện 2 pha được tạo ra bằng cách đóng mở không đồng thời các công tắc K_1 và K_2 xung dòng điện chạy trong 2 cuộn dây (2 pha) được minh hoạ trên hình 4-2.



Hình 4-2

Xét tại thời điểm t_1, khoá K_1 đóng , K_2 mở; dòng điện pha A đặt giá trị cực đại còn dòng điện pha B bằng không. Từ trường stato là từ trường của pha A (có phương thẳng đứng. Từ trường này sẽ hút các cực trái dấu về vị trí trùng phương với nó. Hình 4-3a

Xét tại thời điểm t_2, khoá K_1 mở, K_2 đóng; dòng điện pha A đặt giá trị không còn dòng điện pha B đạt giá trị cực đại. Từ trường stato là từ trường của pha B (có phương nằm ngang). Từ trường này sẽ hút các cực trái dấu về vị trí trùng phương với nó. Hình 4-3b.



a) *b)*

Hình 4-3

Nhận xét:

Sau một xung điện, rô to đã quay được một góc là 30^0. **Góc này bằng một nửa góc** hợp bởi 2 cực từ trái dấu kề nhau trên rô to.

Xét ở các thời điểm trên các xung điện tiếp theo ta cũng có kết quả tương tự.

Nếu gọi số cặp cực trên stato là a, số cực trên rô to là n thì để quay hết một vòng rô to cần dịch chuyển k bước. Với k = n × a (bước) - k còn được gọi là độ phân giải của động cơ bước.

Góc quay của một bước chuyển là: $=\frac{360^0}{k}$

Ví dụ. Trong động cơ trên ta có:

Độ phân giải k = 6 × 2 = 12.

Góc quay của một bước chuyển là: $\frac{360^0}{12}=30^0$

Đặc điểm của động cơ bước:

- Sử dụng nguồn điện áp xung, tốc độ quay phụ thuộc vào tần số xung điện nên việc thay đổi và kiểm soát tốc độ đơn giản.

- Động cơ này hoạt động ổn định, sau mỗi bước động cơ tự "hãm" và dừng lại để dịch chuyển bước tiếp theo vì vậy điểm dừng có độ chính xác cao, thích hợp với hệ thống tự động điều khiển

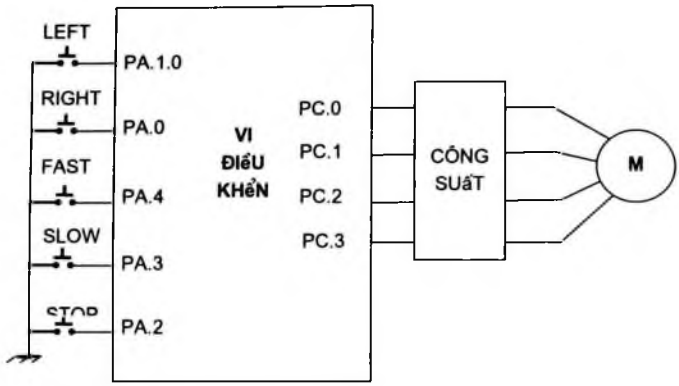
- Tính năng ưu điểm như động cơ một chiều nhưng lại không có chổi than, không phát sinh tia lửa điện.

Tuy nhiên, nguồn cung cấp cho động cơ bước là nguồn xung, không sẵn có ở lưới điện công nghiệp nên cần phải trang bị thiết bị tạo nguồn xung.

Sơ đồ kết nối phần cứng:

- Các nút ấn chức năng được nối vào các chân PA.0, PA.1, PA.2, PA.3, PA.4 tín hiệu được tạo ra là mức logic 0V.

- Động cơ bước M được điều khiển bằng các chân PC.0, PC.1, PC.2, PC.3 thông qua mạch công suất. Có hai cách điều khiển mô tơ đó là: Cả bước (Full step) và nửa bước (Hafl step).



Mã điều khiển

Xung thứ	1	2	3	4	5	6	7	8	9
Biểu diễn số nhị phân ra portC	1	1	1	0	1	1	1	0	1
	1	1	0	1	1	1	0	1	1
	1	0	1	1	1	0	1	1	1
	0	1	1	1	0	1	1	1	0
Biểu diễn số HEX	07H	0BH	0DH	0EH	07H	0BH	0DH	0EH	07H

II. CÁC BƯỚC TIẾN HÀNH

Khởi động phần mềm lập trình Code Vision AVR.

- Tạo file ***dongcobuoc.c*** và lưu trong thư mục ***dongcobuoc.***

Soạn thảo chương trình nguồn.

- Khai báo các biến và các chương trình con:

```
#include <mega16.h>
#include <delay.h>
unsigned char mode=0,i=10;
void run_right();
void run_lefl();
void stop();
```

- Soạn thảo chương trình theo nội dung sau:

```
        while (1) {
         if(PINA.0==0) mode=1;
         if(PINA.1==0) mode=2;
        if(PINA.2==0) mode=3;
         if(PINA.3==0) i++;
         if(PINA.4==0) i--;
         switch(mode) {
         case 1: run_right(); break;
        case 2: run_lefl(); break;
        case 3: stop(); break;
        PORTC=0xff;
        }}}
        void run_right(){
 PORTC=0x07;
 delay_ms(i);
PORTC=0x0B;
delay_ms(i);
 PORTC=0x0D;
 delay_ms(i);
PORTC=0x0E;
 delay_ms(i);
}
        void run_lefl(){
PORTC=0x0E;
 delay_ms(i);
PORTC=0x0D;
delay_ms(i);
PORTC=0x0B;
delay_ms(i);
```

```
PORTC=0x07;
delay_ms(i);
}
        void stop(){ PORTC=0xff;}
        //End Programmer.
```

Biên dịch chương trình nguồn sang File dongcobuoc.hex

Hoạt động thử.

- Tắt công tắc nguồn.
- Ghép nối môđun động cơ bước với Main chính qua PORTC, môđun nút ấn qua PORTA, (chú ý địa chỉ các chân vào ra).
- Bật công tắc nguồn.
- Dịch và nạp chương trình.
- Ấn nút "PA0" động cơ chạy thuận.
- Ấn nút "PA1" động cơ chạy ngược.
- Ấn nút "PA2" động cơ dừng.
- Ấn nút "PA3" động cơ chạy chậm.
- Ấn nút "PA4" động cơ chạy nhanh.
- Theo dõi sự hoạt động của động cơ.

IV. BÁO CÁO THÍ NGHIỆM

Số thứ tự và tên bài.

Mục đích thí nghiệm.

Chương trình đã soạn thảo để thí nghiệm.

Kết quả thí nghiệm.

Nhận xét và kết luận.

V. CÂU HỎI KIỂM TRA

Vẽ lưu đồ và giải thích hoạt động của chương trình điều khiển động cơ bước.

Viết lại chương trình để khởi động động cơ bước, sau một thời gian động cơ tự động ăng tốc.

Viết chương trình điều khiển động cơ bước quay 10 vòng thì dừng.

BÀI THỰC HÀNH SỐ 5
HIỂN THỊ LED 7 THANH

Bài tập ứng dụng:

Lập trình hiển thị số đếm từ 0 đến 99.

I. MỤC ĐÍCH

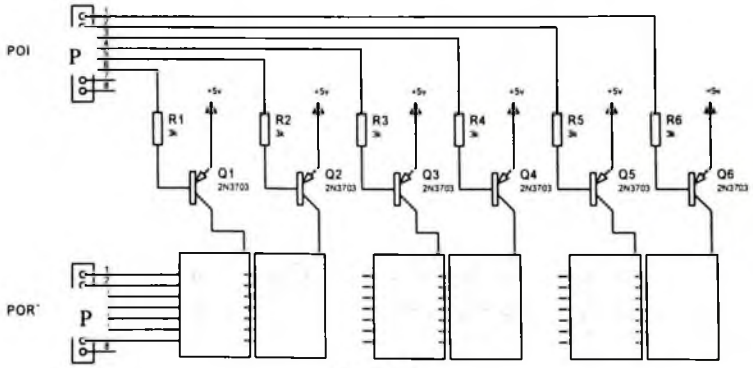
Học xong bài này học sinh cần nắm được cá nội dung sau:

- Hiểu được cấu tạo và phần cứng và cách hiển thị các Led 7 thanh.
- Hiểu được cách định nghĩa các chân out: VD *"#define L0 PORTC.5"*
- Hiểu được cách khai báo và gọi các giá trị trong một mảng.
- Hiểu và sử dụng các lệnh toán học như chia lấy phần nguyên, phần dư...
- Hiểu được cách đổi kiểu bằng phương pháp gán trong lập trình C.
- Xây dựng được chương trình hiển thị số đếm bằng LED 7 thanh.

II. TÓM TẮT LÍ THUYẾT

Phương quét dữ liệu:

Xem sơ đồ kết nối phần cứng để hiển thị 6 số bằng phương pháp quét như hình 5.1



Hình 5.1

Mạch điện dùng 6 Tranzitor được điều khiển bằng PC để xác địch địa chỉ. Dữ liệu xuất ra tại PD đã được mã hoá thành các số hệ 10. Khi quét dữ liệu được xuất ra đồng thời trên tất cả các chân của Led 7 thanh nhưng chỉ có Led được chọn mới có thể hiển thị được giá trị xuất ra. Như vậy tại một thời điểm chỉ có một con Led sáng.

Phương pháp quét:

Phương pháp này dựa trên hiện tượng lưu ảnh của mắt, nếu tần số sáng tắt của đèn LED lớn hơn 25Hz thì ta cảm thấy là đèn sáng liên tục.

Phương pháp quét được mô tả như sau:

Các LED 7 thanh được sử dụng trong mạch là loại anốt chung. Vì vậy để LED sáng thì các chân điều khiển (chân dữ liệu a,b,c,d...phải ở mức thấp còn chân COM của LED phải được nối lên mức 1(nối qua Transistor). Các Transistor sử dụng trong mạch là loại PNP, vì vậy mức tín hiệu đầu ra của PD phải là mức 0 thì LED mới sáng.

II. CÁC BƯỚC TIẾN HÀNH

Khởi động phần mềm lập trình Code Vision AVR.

- Tạo ***file quetled7thanh.c*** lưu trong thư mục ***quetled7thanh***.

- Soạn thảo chương trình nguồn

Khai báo các chương trình con và các biến, các định nghĩa.

```
#include <mega16.h>
#include <delay.h>
#define L0 PORTC.5
#define L1 PORTC.4
#define L2 PORTC.3
#define L3 PORTC.2
#define L4 PORTC.1
#define L5 PORTC.0
int a[]={0x12,0x7e,0x0b,0x4a,0x66,0xc2,0x82,0x7a,0x02,0x42};
int tang,temp,trieu,tnghin,nghin,tram,chuc,donvi,i,j=0;
void display();
void convert();
```

- Soạn thảo chương trình theo nội dung sau:

```
while (1){
convert();
display();
        if (PINA.1==0) tang=0,j=0;
}}
void convert(){
trieu=tang/1000000;
        temp=tang%1000000;
        tnghin=temp/100000;
        temp=temp%100000;
        nghin=temp/1000;
        temp=temp%1000;
        tram=temp/100;
        temp=temp%100;
        chuc=temp/10;
        donvi=temp%10;
        for (i=0;i<=9;i++){
        if (i==trieu) trieu=a[i];
         if (i==tnghin) tnghin=a[i];
         if (i==nghin) nghin=a[i];
         if (i==tram) tram=a[i];
         if (i==chuc) chuc=a[i];
        if (i==donvi) donvi=a[i];
         }}
void display(){
        PORTD=trieu;
        L0=0;
delay_us(100);
```

```
L0=1;
PORTD=tnghin;
L1=0;
delay_us(100);
L1=1;
PORTD=nghin;
L2=0;
delay_us(100);
L2=1;
PORTD=tram;
L3=0;
delay_us(100);
L3=1;
PORTD=chuc;
L4=0;
delay_us(100);
L4=1;
PORTD=donvi;
L5=0;
delay_us(100);
L5=1;
if(PINA.0==0)
{tang=j++;
delay_us(50); } }
```

Hoạt động thử.

- Lắp Bus dữ liệu đun Led 7 thanh với PD, bus địa chỉ nối với PC. PA nối với môđun ím ấn.

- Bật công tắc nguồn.

- Dịch và nạp chương trình.

- Nhấn nút ấn reset ứng với PA1.

- Nhấn nút tăng ứng với PA0.

- Theo dõi sự hoạt động của mạch.

IV. BÁO CÁO THÍ NGHIỆM

Số thứ tự và tên bài.

Mục đích thí nghiệm.

Chương trình đã soạn thảo để thí nghiệm

Kết quả thí nghiệm:

Nhận xét và kết luận.

V. CÂU HỎI KIỂM TRA

Vẽ lưu đồ và giải thích hoạt động của chương trình.

Viết lại chương trình thực hiện công việc như sau. Khi nhấn nút ứng với PA0 thì mạch tự động đếm từ 0 đến 100. Nhấn nút ứng với PA1 thì reset về 0.

BÀI THỰC HÀNH SỐ 6
GIẢI MÃ BÀN PHÍM

Bài tập ứng dụng:

Lập trình giải mã bàn phím, với mỗi phím nhấn sẽ hiển thị số tương ứng trên LED thanh.

MỤC ĐÍCH

Học xong bài này học sinh cần nắm được các nội dung sau:

- Hiểu được cấu trúc phần cứng, và cách quét các phím trên matrix phím.
- Hiểu được cách giải mã bàn phím sử dụng phương pháp quét theo cột, theo hàng.
- Xây dựng được chương trình giải mã các phím nhấn bố trí trên ma trận 4 × 4.

. TÓM TẮT LÍ THUYẾT

Sử dụng nút ấn để nối với các đầu vào sẽ tốn nhiều đầu vào (giả sử có n nút ấn sẽ n n đầu vào). Để giảm bớt các đầu vào ta dùng các phím nhấn bố trí và kết nối thành a trận m × n.

Ví dụ: Có 12 nút ấn bố trí thành ma trận 3 × 4 nối như sơ đồ hình 6.1 ta chỉ cần 3 + 4 = 7 ầu vào.

Để xác định phím được nhấn ta sử dụng phương pháp quét hàng hoặc quét cột ư sau:

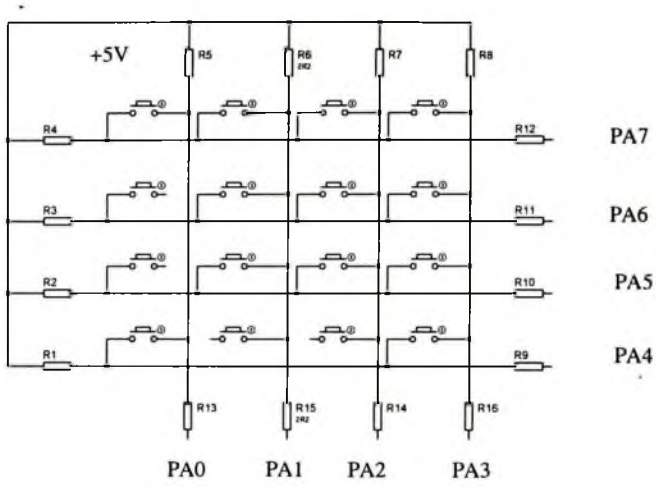
Phương pháp quét cột:

Ban đầu ở trạng thái bình thường tất cả các chân PA0 đến PA7 đều ở mức cao. Để ểm tra phím được nhấn ta lần lượt đặt các chân PA4, PA5, PA6, PA7 ở mức thấp và ần lượt kiểm tra từng cột. Kiểm tra nếu thấy cột nào ở mức thấp ta sẽ kết luận phím nằm àng và cột tương ứng được nhấn.

Ví dụ: Đầu tiên ta cho hàng PA7 = 0 sau đó lần lượt kiểm tra (quét) các cột PA0, A1, PA2, PA3.

Phương pháp quét hàng:

Tương tự như trên, bây giờ ta lần lượt đặt các chân PA0, PA1, PA2, PA3 ở mức thấp ần đó sẽ kiểm tra lần lượt từng hàng.



Hình 6.1

III. CÁC BƯỚC TIẾN HÀNH

Khởi động phần mềm lập trình Code Vision AVR.

- Tạo một dự án mới đặt tên là ***quetbanfim.c***, trong thư mục ***quetbanfim***

Soạn thảo chương trình nguồn

- Khai báo các biến, các chương trình con và các định nghĩa.

```
#include <mega16.h>
#include <delay.h>
#define L0 PORTC.5
#define L1 PORTC.4
#define L2 PORTC.3
#define L3 PORTC.2
#define L4 PORTC.1
#define L5 PORTC.0
#define data PORTD
```

```
int a[]={0x12,0x7e,0x0b,0x4a,0x66,0xc2,0x82,0x7a,0x02,0x42};
unsigned char temp,chuc,donvi,i;
void scan();
void convert();
void display();
```

- Soạn thảo chương trình theo nội dung sau:

```
while (1) {
 scan();
convert();
 display();
}}
void display(){
        data=chuc;
        L4=0;
        delay_us(100);
        L4=1;
        data=donvi;
        L5=0;
        delay_us(100);
        L5=1;
}
void convert(){
        chuc=temp/10;
        donvi=temp%10;
        for (i=0;i<=9;i++){
                if (i==chuc) chuc=a[i];
                if (i==donvi) donvi=a[i];
        }}
void scan(){
```

```
PORTA.0=0;
        if (PINA.4==0) temp=0,delay_us(50);
        if (PINA.5==0) temp=1,delay_us(50);
        if (PINA.6==0) temp=2,delay_us(50);
        if (PINA.7==0) temp=3,delay_us(50);
PORTA.0=1;
PORTA.1=0;
        if (PINA.4==0) temp=4,delay_us(50);
        if (PINA.5==0) temp=5,delay_us(50);
        if (PINA.6==0) temp=6,delay_us(50);
        if (PINA.7==0) temp=7,delay_us(50);
PORTA.1=1;
PORTA.2=0;
        if (PINA.4==0) temp=8,delay_us(50);
        if (PINA.5==0) temp=9,delay_us(50);
        if (PINA.6==0) temp=10,delay_us(50);
        if (PINA.7==0) temp=11,delay_us(50);
PORTA.2=1;
PORTA.3=0;
        if (PINA.4==0) temp=12,delay_us(50);
        if (PINA.5==0) temp=13,delay_us(50);
        if (PINA.6==0) temp=14,delay_us(50);
        if (PINA.7==0) temp=15,delay_us(50);
PORTA.3=1;
}
```

Hoạt động thử.

- Ghép nối môđun Led thanh với Main chính như sau: Bus dữ liệu nối với PD, bus địa chỉ nối với PC.

- Ghép môđun bàn phím với PA của Main chính, (chú ý địa chỉ các chân vào ra).

- Bật công tắc nguồn.

- Dịch và nạp chương trình xuống Main.
- Lần lượt ấn các phím trên matrix phím.
- Quan sát hiển thị trên Led 7 thanh.

. BÁO CÁO THÍ NGHIỆM

Số thứ tự và tên bài.

Mục đích thí nghiệm.

Chương trình đã soạn thảo để thí nghiệm

Kết quả thí nghiệm:

Nhận xét và kết luận.

. CÂU HỎI KIỂM TRA

Vẽ lưu đồ và giải thích hoạt động của chương trình trên.

Viết lại chương trình trên bằng phương pháp quét hàng.

BÀI THỰC HÀNH SỐ 7
HIỂN THỊ LCD

Bài tập ứng dụng:

Lập trình hiển thị các đoạn text lên LCD điều khiển qua nút ấn nối với PA.

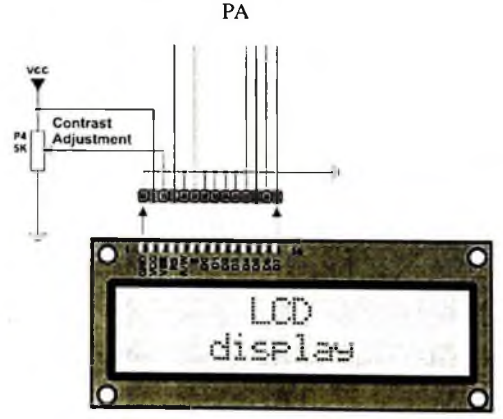
I. MỤC ĐÍCH

Học xong bài này học sinh cần nắm được các nội dung sau:

- Hiểu được nguyên lý hoạt động của LCD.
- Hiểu được cách gửi lệnh, gửi dữ liệu lên LCD.
- Cách dùng lệnh "*Breack;*"
- Cách khai báo và sử dụng các đoạn *text*.

II. TÓM TẮT LÍ THUYẾT

Sơ kết nối của LCD. Như hình 7.1



Hình 7.1

Một số đặc điểm của LCD

- LCD có hai chế độ làm việc:
  + Chế độ 8 bít: Chế độ này thường dùng cho 8051.
  + Chế độ 4 bít: Cách này thường dùng cho các loại vi điều khiển cấp cao như AVR, PIC hay dsPIC.
- LCD có 3 đường điều khiển.
  + Đường E khởi tạo quá trình truyền dữ liệu từ VĐK tới LCD:
    - Nếu E = 1 cho phép
    - Nếu E = 0 không cho phép.
  + Đường RS quy định gửi lệnh hay gửi dữ liệu lên LCD:
    - RS = 1 dữ liệu (ký tự) đang được viết lên LCD.
    - RS = 0 lệnh đang được viết lên LCD.
  + Đường R/W quy định hướng truyền dữ liệu.
    - R/W = 0 vi điều khiển đang viết dữ liệu lên LCD.
    - R/W = 1 vi điều khiển đang đọc dữ liệu từ LCD.
- Để viết dữ liệu lên LCD cần thực hiện các bước sau:
  + Đặt R/W = 0.
  + Đặt RS = 0 hay RS = 1 tùy thuộc vào mục đích gửi lệnh hay gửi dữ liệu.
  + Đặt dữ liệu vào bus dữ liệu (P2) nếu quá trình là quá trình viết.
  + Đặt E = 1.
  + Đặt E = 0.
  + Đọc dữ liệu tại bus dữ liệu (nếu quá trình là quá trình đọc). Việc đọc dữ liệu từ LCD thực hiện tương tự như trên nhưng trường hợp này ít dùng.
- Các ký tự hiển thị trên LCD được lưu trong DDRAM (Display Data RAM).
- Việc đọc viết ký tự lên LCD tốn khoảng 40us đến 120us. Trong khoảng thời gian này LCD "bận", không giao tiếp với bên ngoài. Do vậy trong chương trình ta cần thực hiện delay để chờ LCD thực hiện công việc của mình.
- Dưới đây là bảng mã lệnh của LCD.

Mã HEXA	Lệnh đến thanh ghi LCD
1	Xóa màn hình hiển thị
2	Trở về đầu dòng
4	Dịch con trỏ sang trái
6	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 hiển thị nhấp nháy con trỏ
10	Dịch vị trí con trỏ sang trái
14	Dịch vị trí con trỏ sang trái
18	Dịch toàn bộ hiển thị sang trái
1C	Dịch toàn bộ hiển thị sang.phải
80	Đưa con trỏ về đầu dòng thứ 1
C0	Đưa con trỏ về đầu dòng thứ 2
38	Hai dòng và ma trận 5x7

III. CÁC BƯỚC TIẾN HÀNH

Khởi động phần mềm lập trình Code Vision AVR.

- Tạo một file ***lcd.c*** trong thư mục ***lcd.***

Soạn thảo chương trình nguồn

- Khai báo các biến và các sử dụng như sau:

```
#include <mega16.h>
#include <delay.h>
// Alphanumeric LCD Module functions
#asm
.equ __lcd_port=0x15 ;PORTC
#endasm
```

```
#include <lcd.h>
char i;
char a[]={'N','G','O','C',' ','H','U','Y',' ','C','o',',',',','L','t','d'};
//-----------------------------------------------------------------------
// Declare your global variables here
void text1();
void text2();
void text3();
```

- Soạn thảo chương trình theo nội dung sau:

```
while (1) {
for(i=0;i<16;i++){
lcd_gotoxy(i,0);
lcd_putchar(a[i]);
delay_ms(100);
}
break;
}
while (2){
if (PINA.0==0) text1();
if (PINA.1==0) text2();
if (PINA.2==0) text3();
delay_ms(100) ;
}}
void text1(){
lcd_gotoxy(0,1);
lcd_putsf("Kit AVR ");
delay_us(50);
}
void text2(){
```

```
        lcd_gotoxy(0,1);
    lcd_putsf("Phat tren tai ");
    delay_us(50);
    }
    void text3(){
        lcd_gotoxy(0,1);
        lcd_putsf("Cty Ngoc Huy ");
    delay_us(50);
    }
```

Hoạt động thử.

- Tắt công tắc nguồn.
- Lắp ghép mộđun LCD với main chính qua PC, (chú ý địa chỉ các chân vào ra).
- Bật công tắc nguồn.
- Dịch và nạp chương trình xuống Main.
- Tác động vào các nút ấn ở PA.
- Theo dõi quan sát hiển thị trên LCD.

IV. BÁO CÁO THÍ NGHIỆM

Số thứ tự và tên bài.

Mục đích thí nghiệm

Chương trình đã soạn thảo để thí nghiệm

Kết quả thí nghiệm:

Nhận xét và kết luận.

V. CÂU HỎI KIỂM TRA

Vẽ lưu đồ và giải thích hoạt động của chương trình.

BÀI THỰC HÀNH SỐ 8
LẬP TRÌNH ĐIỀU KHIỂN MÔĐUN ADC-DAC

Bài tập ứng dụng:

Lập điều khiển môđun ADC hoạt động, đọc dữ liệu về từ PD và đưa ra PC.

I. MỤC ĐÍCH

Học xong bài này học sinh cần nắm được các nội dung sau:

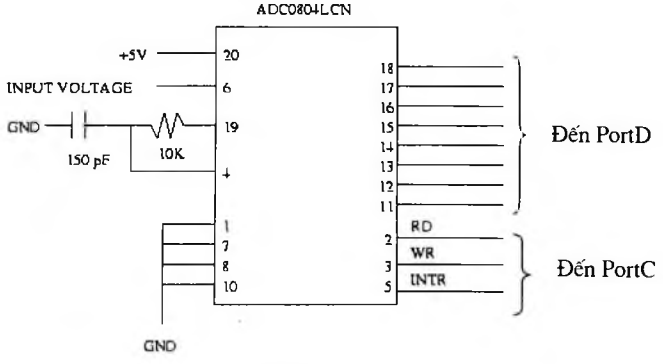
- Hiểu được cơ chế hoạt động của chíp ADC 0804 và DAC 0808
- Thiết lập được các chế độ đọc và ghi lên ADC.

II. TÓM TẮT LÍ THUYẾT

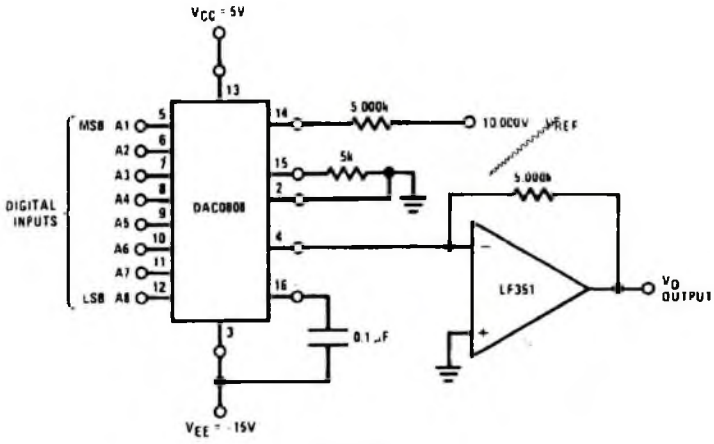
- ADC là một mạch được sử dụng rất rộng rãi trong lĩnh vực điều khiển đo lường. Nó được dùng để chuyển đổi tín hiệu tương tự (analog) thành tín hiệu số (digital).

Ví dụ: Đầu ra của cảm biến nhiệt LM35, hay cảm biến trọng lượng là một tín hiệu điện áp thay đổi tuyến tính theo giá trị tác động lên cảm biến (nhiệt độ, khối lượng). Vì vi điều khiển chỉ làm việc với các con số 0 và 1 nên bắt buộc ta phải chuyển đổi tín hiệu này thành các con số 0 và 1.

Sơ đồ bố trí chân ADC0804: Hình 8-1: Các chân data được nối với PortD, các chân điều khiển nối với PortC.



Hình 8-1



Hình 8-2

Sơ đồ bố trí các chân của IC DAC 0808: Hình 8-2

Nguyên tắc điều khiển IC ADC như sau: Làm theo đúng thứ tự sau.

+ Đặt chân RD ở mức cao.

+ Đặt chân CS ở mức thấp.

+ Chờ một lúc.

+ Đặt chân RW ở mức thấp.

+ Chờ một lúc.

+ Chuyển chân RW lên mức cao.

+ Chờ một lúc.

+ Chuyển chân CS lên mức cao.

Để đọc data từ ADC ta làm như sau:

+ Đặt chân CS ở mức cao.

+ Chờ một lúc.

+ Đặt chân RD ở mức thấp.

+ Đọc data.

+ Chuyển chân CS xuống mức thấp.

III. CÁC BƯỚC TIẾN HÀNH

Khởi động phần mềm lập trình Code Vision AVR.

- Tạo một file ***adc.c*** trong thư mục ***adc.***

Soạn thảo chương trình nguồn

- Các khai báo, định nghĩa.

```
#include <mega16.h>
#include <delay.h>
#define ADC_DONE PORTC.3 //INTR
#define ADC_CS PORTC.2
#define ADC_RD PORTC.0
#define ADC_WR PORTC.1
#define ADC_DATA PIND
void convert();
void read();
unsigned char voltage;
```

- Soạn thảo chương trình theo nội dung sau:

```
while (1){
convert();
delay_ms(1);
read();
 delay_ms(1);
PORTA=voltage;
        }}
void convert(){
        ADC_RD=1;
        ADC_CS=0;
        delay_us(100);
        ADC_WR=0;
        delay_us(100);
        ADC_WR=1;
```

```
        delay_us(100);
        ADC_CS=1;
        }
void read(){
        ADC_CS=1;
        delay_us(100);
        ADC_RD=0;
        voltage = ADC_DATA;
        delay_us(100);
        ADC_CS=0;
        }
```

Hoạt động thử.

- Tắt công tắc nguồn.

- Ghép môđun ADC như sau: Bus dữ liệu nối với PD, bus địa chỉ nối với PC, (chú ý địa chỉ các chân vào ra).

- Nối PA ra môđun Led đơn.

- Chuyển JUM trên môđun ADC sang đo điện áp.

- Bật công tắc nguồn.

- Dịch và nạp chương trình xuống Main.

- Điều chỉnh chiết áp.

- Theo dõi quá trình hiển thị của các Led.

IV. BÁO CÁO THÍ NGHIỆM

Số thứ tự và tên bài.

Mục đích thí nghiệm

Chương trình đã soạn thảo để thí nghiệm

Kết quả thí nghiệm:

Nhận xét và kết luận.

V. CÂU HỎI KIỂM TRA

Vẽ lưu đồ thuật toán và giải thích hoạt động của chương trình.

BÀI THỰC HÀNH SỐ 9
LẬP TRÌNH VỚI NGẮT CỦA BỘ ĐỊNH THỜI

Bài tập ứng dụng:

Tạo một xung bằng tín hiệu ngắt tại chân RD7 và một xung bằng vòng while tại chân RD0.

I. MỤC ĐÍCH

Học xong bài này học sinh cần nắm được các nội dung sau:

- Hiểu được ý nghĩa và cơ chế làm việc của ngắt bộ định thời trong AVR.
- Hiểu được cách đặt các thông số cho thanh ghi cờ ngắt, thanh ghi mặt nạ.
- Xây dựng được chương trình ứng dụng sử dụng ngắt của bộ định thời.

II. TÓM TẮT LÍ THUYẾT

Ngắt là sự đáp ứng các sự kiện bên trong hoặc bên ngoài nhằm thông báo cho bộ vi điều khiển biết thiết bị đang cần được phục vụ. Ở phương pháp ngắt, mỗi khi có 1 thiết bị cần được phục vụ thì thiết bị sẽ báo cho bộ vi điều khiển bằng cách gửi một tín hiệu ngắt. Khi nhận được tín hiệu này, bộ vi điều khiển ngừng mọi công việc đang thực hiện để chuyển sang phục vụ thiết bị. Chương trình đi cùng với ngắt được gọi là *trình phục vụ ngắt* ISR (Interrupt Service Routine) hay còn gọi là *bộ quản ngắt* (Interrupt handler).

Cho phép ngắt và cấm ngắt:

Khi Reset thì tất cả mọi ngắt đều bị cấm (bị che), có nghĩa là không có ngắt nào được bộ vi điều khiển đáp ứng nếu chúng được kích hoạt. Các ngắt phải được cho phép bằng phần mềm để bộ vi điều khiển có thể đáp ứng được.

Thanh ghi điều khiển – $TCCR_x$:

Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bít 3,6 - WG00-WG01: Đây là các bít chọn chế độ trong Timer. Các giá trị được mô tả trong bảng sau. Bảng chọn chế độ time.

Mode	WGM01 (CTC0)	WGM00 (PWM0)	Timer/Counter Mode of Operation	TOP	Update of OCR0	TOV0 Flag Set-on
0	0	0	Normal	0xFF	Immediate	MAX
1	0	1	PWM, Phase Correct	0xFF	TOP	BOTTON
2	1	0	CTC	OCR0	Immediate	MAX
3	1	1	Fast PWM	0xFF	TOP	MAX

Bít 5-4 : COM00-COM01: Quy định giá trị đầu ra trong các phép so sánh

Bít 2: 0 - CS2:0: Đây là các bít quy định xung cấp cho hoạt động của timer. Bảng dưới đây mô tả toàn bộ các giá trị. Bảng chọn chế độ xung clock.

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/(No prescalling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge
1	1	1	External clock source on T0 pin. Clock on raising edge

Thanh ghi cờ ngắt - TIFR

Bit	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR
Read/Write	R/W	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bít 1-ocfx: Khi hai giá trị bằng nhau bít này được set lên bằng 1.

Bít 1-tvox: Khi bộ đếm vượt quá giá trị top thì bít này được set bằng 1.

Thanh ghi mặt nạ ngắt - TIMSK.

Bit	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
Read/Write	R/W	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bít 1-OCIEx: Khi bít này được set lên bằng 1 thì cho phép ngắt so sánh Bít 0-TOIEx: Khi bít này được set lên bằng 1 thì cho phép ngắt tràn.

III. CÁC BƯỚC TIẾN HÀNH

Khởi động phần mềm lập trình Code Vision AVR.

- Tạo file ***ngattime.c*** trong thư mục ***ngattime.***

Soạn thảo chương trình nguồn.

- Soạn thảo chương trình theo nội dung sau:

```
#include <mega16.h>
#include <delay.h>
#define xtal 1000000
#define fmove 2
bit a;
int count=0;
interrupt [TIM0_OVF] void timer0_ovf_isr(void) { //Trình phục vụ ngắt
        count++;
        if (count==2000)
        {a=~a;
        PORTD.7=a;
        count=0;
        } }
void main(void) {
        PORTA=0x00;
        DDRA=0x00;
        DDRB=0xFF;
        PORTC=0x00;
        DDRC=0x00;
        PORTD=0xFF;
        DDRD=0xFF;
        TCCR0=0x01;
```

```
        TCNT0=0x00;
        OCR0=0x00;
        TCNT0=0x10000-(xtal/1024/fmove);
        TIFR=0;
        TIMSK=0x01; //Cho phép ngắt tràn timer0
        #asm("sei")
    while (1) {
    PORTD.0=0;
    delay_ms(500);
    PORTD.0=1;
    delay_ms(500);
    }}.
```

Biên dịch chương trình nguồn sang File ngattime.hex

Hoạt động thử.

- Ghép môđun Led đơn với main chính qua PD.
- Bật công tắc nguồn.
- Dịch và nạp chương trình xuống Main chính.
- Theo dõi sự hoạt động của LED tại PD0 và xung ra tại chân PD7.

IV. BÁO CÁO THÍ NGHIỆM

Số thứ tự và tên bài.

Mục đích thí nghiệm.

Chương trình đã soạn thảo để thí nghiệm.

Kết quả thí nghiệm.

Nhận xét và kết luận.

V. CÂU HỎI KIỂM TRA

Vẽ lưu đồ và giải thích hoạt động của chương trình.

Viết lại chương trình trên với 2 xung tạo ra đều dùng ngắt timer.

BÀI THỰC HÀNH SỐ 10
LẬP TRÌNH VỚI NGẮT NGOÀI

Bài tập ứng dụng:

Viết chương trình bật tắt liên tục một LED ra tại chân PD0, khi có tín hiệu ngắt ngoài thì bật tắt một LED khác tại chân PD7cho bộ đếm sử dụng ngắt ngoài 1 làm đầu vào đếm, ngắt ngoài 0 làm đầu vào reset. Hiển thị ra Led 7 thanh.

I. MỤC ĐÍCH

- Hiểu được ý nghĩa và cơ chế làm việc của ngắt ngoài INT0 và INT1 trong 8051
- Xây dựng được chương trình ứng dụng sử dụng ngắt ngoài.

II. TÓM TẮT LÍ THUYẾT

III. CÁC BƯỚC TIẾN HÀNH

Khởi động phần mềm lập trình Code Vision AVR.

- Tạo một file ***ngatngoai.c*** trong thư mục ***ngatngoai***.

Soạn thảo chương trình nguồn

- Soạn thảo chương trình theo nội dung sau:

```
#include <mega16.h>
#include <delay.h>
bit a;
interrupt [EXT_INT1] void ext_int1_isr(void) //tai chan PD3
        { a=~a;
        PORTC.7=a;
        }
void main(void) {
        PORTA=0x00;
        DDRA=0x00;
```

```
PORTB=0x00;
DDRB=0x00;
PORTC=0xFF;
DDRC=0xFF;
PORTD=0x00;
DDRD=0x00;
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;
GICR|=0x80;
MCUCR=0x08;
MCUCSR=0x00;
GIFR=0x80;
TIMSK=0x00;
ACSR=0x80;
```

```
        SFIOR=0x00;
        #asm("sei")
    while (1) {
    PORTC.0=1;
    delay_ms(500);
    PORTC.0=0;
    delay_ms(500);
    }}
```

Hoạt động thử.

- Tắt công tắc nguồn.
- Ghép môđun Led đơn với Main chính qua PORTC, môđun phím ấn với PORTD. (Chú ý địa chỉ các chân vào ra).
- Bật công tắc nguồn.
- Dịch và nạp chương trình xuống Main.
- Nhấn phím kích hoạt ngắt tại chân PD.3.
- Theo dõi sự hoạt động của LED.

IV. BÁO CÁO THÍ NGHIỆM

Số thứ tự và tên bài.

Mục đích thí nghiệm

Chương trình đã soạn thảo để thí nghiệm

Kết quả thí nghiệm:

Nhận xét và kết luận.

V. CÂU HỎI KIỂM TRA

Vẽ lưu đồ và giải thích hoạt động của chương trình trên.

BÀI THỰC HÀNH SỐ 11
LẬP TRÌNH HIỂN THỊ MATRIX 8X8

Bài tập ứng dụng:

Lập trình hiển thị chữ như sau: ấn nút ở PA0 thì hiển thị chữ "B", ấn nút ở PA1 th hiển thị "C", ấn nút ở PA2 thì hiển thị chữ "D" lên matrix8x8.

I. MỤC ĐÍCH

Học xong bài này thì học sinh cần nắm được các điểm sau:

- Hiểu được cấu trúc phần cứng của matrix 8x8 từ đó mở rộng ra các matrix có kích thước lớn hơn.

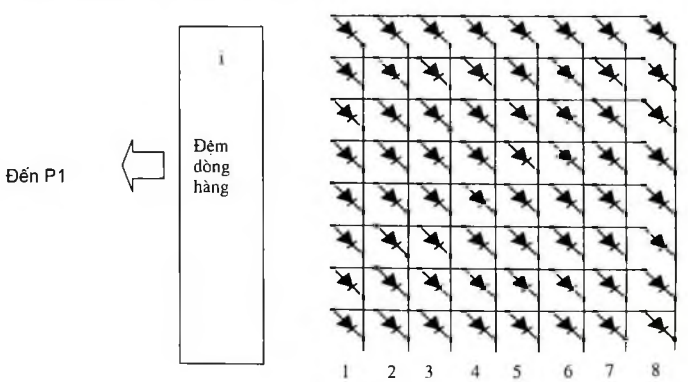
- Hiểu được các khai báo các mảng, vòng for lồng trong nhau.

- Hiểu được cách quét dữ liệu theo cột.

II. TÓM TẮT LÍ THUYẾT

1. Mô tả thiết bị phần cứng.

Dưới đây là sơ đồ nguyên lý của Matrix 8 × 8.



Hình 12.1

Nếu coi mỗi Led là một điểm sáng thì bảng hiển thị của chúng ta coi như một ma trậ gồm 8 hàng và 8 cột. Mỗi Led tương ứng với một điểm sáng hoặc tắt. Nhìn vào sơ đồ t thấy rằng mỗi Led ij sẽ sáng khi i = 1, j = 0 (i chạy từ 0 – 8).

Giá trị xuất ra ở PC tại một thời điểm chỉ có một cột ở mức thấp các cột còn lại phả để lên mức cao.

Ý tưởng:

- Tạo một bảng dữ liệu mã hoá font chữ, khi hoạt động thì CPU sẽ đọc giá trị tron bảng mã này. Bảng mã này có cấu trúc gồm nhiều byte dữ liệu được sắp xếp nối tiế nhau có mục đích.

- Cách hiển thị chữ như sau: Byte dữ liệu đầu tiên được gửi ra PC cột j1=0, tất các cá cột còn lại ở mức cao. Chờ một lúc, byte tiếp theo được gửi ra PC cột j2=0, cột j chuyển lên mức cao, tất cả các cột còn lại đều ở mức cao. Cứ như vậy cho đến cột thứ 8

- Như vậy con số 0 chạy từ j1 đến j8 thì hiển thị được một ảnh, thời gian hiển thị nà là rất ngắn chỉ khoảng 40ms. Như vậy mắt ta không thể quan sát kịp nếu chỉ quét mộ lần. Để mắt nhìn thấy được ảnh hiện lên liên tục thì ta phải quét >24 ảnh/1s.

Kết luận:

Tại một thời điểm chỉ có duy nhất 1 cột mà ở đó có các Led sáng tương ứng với byt dữ liệu được đọc ra. Sau khi các Led trên cột này tắt đi thì Led tương ứng trên cột kế tiế sẽ sáng. Quá trình cứ tiếp tục cho tới cột cuối cùng thì ta đã thực hiện được 1 lần qué tức là ta đã tạo được 1 ảnh cần hiển thị.

2. Phương pháp hiển thị chữ động

Để tạo chữ "A" ở dạng ảnh động ta sẽ cho chữ "A" xuất hiện ở dạng ảnh tĩnh trong thời gian nào đó rồi chuyển sang hiển thị bắt đầu từ kế tiếp (cột 2) tức chữ "A" đã c chuyển sang phải được 1 cột. Để làm được việc này thì khi bit 1 chuyển sang cột 2 t mới đọc byte dữ liệu đầu tiên (có giá trị là 83h) và đọc các byte kế tiếp như trường hợ chữ "A" nằm ở cột 1.

III. CÁC BƯỚC TIẾN HÀNH

Khởi động phần mềm lập trình Code Vision AVR

- Tạo một file ***matrix8x8.c*** trong thư mục ***matrix8x8***.

Soạn thảo chương trình nguồn

- Các khai báo

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

```
unsigned char k,l,m,n=0;
unsigned char a[]={0xfe,0xfd,0xfb,0xf7,0xef,0xdf,0xbf,0x7f};
unsigned char font[]={0xff,0x80,0xB6,0xB6,0xB6,0xBE,0xff,0xff,
    0xff,0xBE,0x80,0xB6,0xB6,0xC9,0xff,0xff,//B
        0xff,0xC1,0xBE,0xBE,0xBE,0xDD,0xff,0xff,//C
    0xff,0xBE,0x80,0xBE,0xBE,0xC1,0xff,0xff};//D
```

- Soạn thảo chương trình theo nội dung sau:

```
while (1){
    for (k=0;k<=8;k++){
        for (l=0,m=n;l<8;l++,m++){
            PORTD=a[l];
            PORTC=font[m];
            delay_us(50);
    PORTC=0xff;
    if (PINA.0==0) n=8;
    if (PINA.1==0) n=16;
    if (PINA.2==0) n=24;
            }}}} // End Programmer
```

Hoạt động thử.

- Tắt công tắc nguồn.

- Ghép môđun matrix với Main chính như sau: Bus dữ liệu nối với PC, bus địa chỉ nối với PD. Môđun nút ấn nối với Main chính qua PA, (chú ý địa chỉ các chân vào ra).

- Bật công tắc nguồn.

- Nạp chương trình xuống Main chính.

- Tác động vào các nút ấn ở PA.

- Quan sát sự hoạt động của mạch.

IV. BÁO CÁO THÍ NGHIỆM

Số thứ tự và tên bài.

Mục đích thí nghiệm

Chương trình đã soạn thảo để thí nghiệm

Kết quả thí nghiệm:

Nhận xét và kết luận.

V. CÂU HỎI KIỂM TRA

Vẽ lưu đồ và giải thích hoạt động của chương trình trên.

BÀI THỰC HÀNH SỐ 12
LẬP TRÌNH GIAO TIẾP I2C VỚI DS1307

Bài tập ứng dụng:

Viết chương trình thiết lập giao tiếp I2C, thiết lập các thông số cho DS1307, đọc thời gian và hiển thị ra LCD

I. MỤC ĐÍCH YÊU CẦU

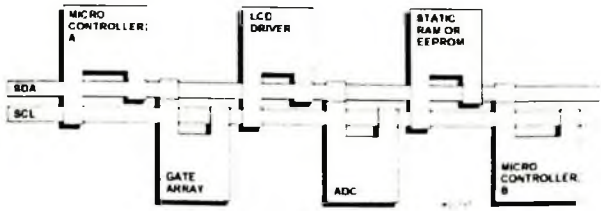
Học xong bài này học sinh cần nắm được các nội dung sau:

- Hiểu được nguyên lý truyền nhận trên bus I2C.
- Hiểu được cấu tạo và các điều kiện hoạt động của DS1307.
- Biết khai báo các chế độ hoạt động của giao tiếp I2C.
- Biết viết các chương trình con gửi và đọc dữ liệu lên đường truyền I2C.
- Xây dựng được chương trình viết và đọc thời gian lên DS1307.

II. TÓM TẮT LÍ THUYẾT

1. Cơ bản về giao tiếp I2C

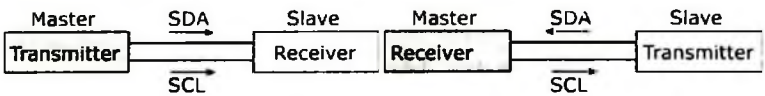
I2C là tên viết tắt của cụm từ Inter - Intergrated Circuit - Bus giao tiếp giữa các IC với nhau và nó trở thành một chuẩn công nghiệp cho các giao tiếp điều khiển, được sử dụng làm bus giao tiếp ngoại vi cho rất nhiều loại IC khác nhau như các loại Vi điều khiển 8051, PIC, AVR, ARM, chíp nhớ như RAM tĩnh (Static Ram), EEPROM, bộ chuyển đổi tương tự số (ADC), số tương tự (DAC), IC điều khiển LCD, LED...



Hình 12.1: BUS I2C và các thiết bị ngoại vi

Một giao tiếp I2C gồm có 2 dây: Serial Data (SDA) và Serial Clock (SCL). SDA là đường truyền dữ liệu 2 hướng, còn SCL là đường truyền xung đồng hồ và chỉ theo một hướng. Như hình vẽ trên, khi một thiết bị ngoại vi kết nối vào đường I2C thì chân SDA của nó sẽ nối với dây SDA của bus, chân SCL sẽ nối với dây SCL.

Mỗi dây SDA hay SCL đều được nối với điện áp dương của nguồn cấp thông qua một điện trở kéo lên (pullup resistor). Sự cần thiết của các diện trở kéo này là vì chân giao tiếp I2C của các thiết bị ngoại vi thường là dạng cực máng hở (opendrain or opencollector). Giá trị của các điện trở này khác nhau tùy vào từng thiết bị và chuẩn giao tiếp, thường dao động trong khoảng 1k đến 10k. Trở lại với hình 1.1, ta thấy có rất nhiều thiết bị (ICs) cùng được kết nối vào một bus I2C, tuy nhiên sẽ không xảy ra chuyện nhầm lẫn giữa các thiết bị, bởi mỗi thiết bị sẽ được nhận ra bởi một địa chỉ duy nhất với một quan hệ chủ/tớ tồn tại trong suốt thời gian kết nối. Mỗi thiết bị có thể hoạt động như là thiết bị nhận dữ liệu hay có thể vừa truyền vừa nhận. Hoạt động truyền hay nhận còn tùy thuộc vào việc thiết bị đó là chủ (master) hay tớ (slave). Một thiết bị hay một IC khi kết nối với bus I2C, ngoài một địa chỉ (duy nhất) để phân biệt, nó còn được cấu hình là thiết bị chủ (master) hay tớ (slave). Tại sao lại có sự phân biệt này ? Đó là vì trên một bus I2C thì quyền điều khiển thuộc về thiết bị chủ (master). Thiết bị chủ nắm vai trò tạo xung đồng hồ cho toàn hệ thống, khi giữa hai thiết bị chủ/tớ giao tiếp thì thiết bị chủ có nhiệm vụ tạo xung đồng hồ và quản lý địa chỉ của thiết bị tớ trong suốt quá trình giao tiếp. Thiết bị chủ giữ vai trò chủ động, còn thiết bị tớ giữ vai trò bị động trong việc giao tiếp.



Hình 12.2: Truyền nhận dữ liệu giữa master/slave

Nhìn hình trên ta thấy xung đồng hồ chỉ có một hướng từ chủ đến tớ, còn luồng dữ liệu có thể đi theo hai hướng, từ chủ đến tớ hay ngược lại tớ đến chủ. Về dữ liệu truyền trên bus I2C, một bus I2C chuẩn truyền 8#bit dữ liệu có hướng trên đường truyền với tốc độ là 100Kbits/s – Chế độ chuẩn (Standard mode). Tốc độ truyền có thể lên tới 400Kbits/s – Chế độ nhanh (Fast mode) và cao nhất là 3,4Mbits/s – Chế độ cao tốc (High#speed mode). Một bus I2C có thể hoạt động ở nhiều chế độ khác nhau:

- Một chủ một tớ (one master - one slave)
- Một chủ nhiều tớ (one master - multi slave)
- Nhiều chủ nhiều tớ (Multi master - multi slave)

Dù ở chế độ nào, một giao tiếp I2C đều dựa vào quan hệ chủ/tớ. Giả thiết một thiết bị A muốn gửi dữ liệu đến thiết bị B, quá trình được thực hiện như sau:

- Thiết bị A (Chủ) xác định đúng địa chỉ của thiết bị B (tớ), cùng với việc xác định địa chỉ, thiết bị A sẽ quyết định việc đọc hay ghi vào thiết bị tớ.

- Thiết bị A gửi dữ liệu tới thiết bị B.

- Thiết bị A kết thúc quá trình truyền dữ liệu.

Khi A muốn nhận dữ liệu từ B, quá trình diễn ra như trên, chỉ khác là A sẽ nhận dữ liệu từ B. Trong giao tiếp này, A là chủ còn B vẫn là tớ.

2. Truyền dữ liệu lên bus

Nguyên tắc truyền một byte dữ liệu lên bus I2C gồm có các bước sau:

- Gửi bít Start từ Master tới slave, đợi cho đến khi truyền xong.

- Gửi địa chỉ của slave lên đường truyền. Dùng để chọn Slave nào hoạt động, đợi cho đến khi truyền xong.

- Gửi địa chỉ cần lưu dữ liệu tới, đợi cho đến khi truyền xong.

- Gửi dữ liệu cần truyền tới Slave, đợi cho đến khi truyền xong.

- Tiếp tục gửi dữ liệu...

- Khi muốn kết thúc gửi bít Stop lên đường truyền.

3. Nhận dữ liệu từ bus

Quá trình nhận dữ liệu từ slave phải tuân thủ theo các bước sau:

- Gửi bít Start từ Master tới slave, đợi cho đến khi truyền xong.

- Gửi địa chỉ của slave (bit 0 = 0) lên đường truyền. Dùng để chọn Slave nào hoạt động, đợi cho đến khi truyền xong.

- Gửi địa chỉ của dữ liệu cần nhận, đợi cho đến khi truyền xong.

- Gửi bit Restart, đợi cho đến khi truyền xong.

- Gửi địa chỉ của slave lên đường truyền (bit 0 =1 báo rằng hoạt động sắp tới là đọc). Đợi cho đến khi truyền xong.

- Đọc dữ liệu từ Slave, đợi cho đến khi đọc xong.

- Phát bít ACK báo tiếp tục nhận dữ liệu, đợi cho đến khi truyền xong.

- Đọc dữ liệu từ Slave, đợi cho đến khi đọc xong.

- Phát bít ACK báo tiếp tục nhận dữ liệu, đợi cho đến khi truyền xong.

- .....

- Đọc dữ liệu từ Slave, đợi cho đến khi đọc xong.

- Phát bít NACK báo rằng qua trình nhận dữ liệu đã kết thúc, đợi cho đến khi truyền xong.

- Phát bít Stop để kết thúc.

4. IC thời gian thực DS1307

- DS1307 là con Real Time clock của hãng Dallas và được dùng khá phổ biến hiện nay. Cũng giống như các RTC khác, DS1307 có các đặc tính sau:

+ Dùng chuẩn giao tiếp I2C.

+ Hiển thị giờ phút giây, ngày, tháng năm (đến tận năm 2099).

+ Dùng thạch anh 32.768 kHz làm bộ tạo dao động.

+ Sử dụng Pin 3V để hoạt động khi mất điện.

+ Có 56 byte Ram cho người sử dụng.....

- Một số đặc điểm mà các bạn cần phải chú ý:

+ DS1307 có 7 byte dữ liệu nằm từ địa chỉ 0x00 tới 0x06, 1 byte điều khiển, và 56 byte lưu trữ (dành cho người sử dụng).

+ Khi xử lý dữ liệu từ DS1307, nhà sản xuất đã tự chuyển cho ta về dạng số BCD, ví dụ như ta đọc được dữ liệu từ địa chỉ 0x04 (tương ứng với day- ngày trong tháng) và tại 0×05 (tháng) là 0×15, 0×11 như thế có nghĩa là lúc đó là ngày 15-11 chứ ko phải là ngày 21 tháng 17.

+ Lưu ý đến vai trò của chân **SQW/OUT**. Đây là chân cho xung ra của DS1307 có 4 chế độ 1Hz, 4.096HZ, 8.192Hz, 32.768Hz... các chế độ này được quy định bởi các bít của thanh ghi Control Register (địa chỉ 0×07).

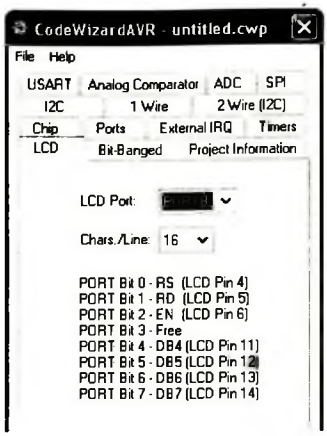
+ Địa chỉ trên mạng I2C của DS1307là **0xD0.**

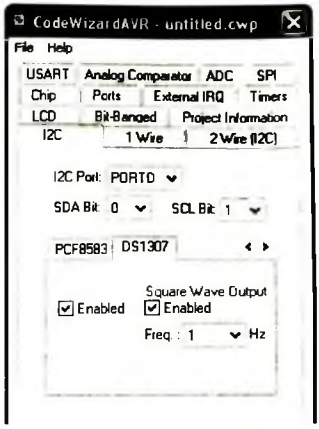
III. CÁC BƯỚC TIẾN HÀNH

1. Khởi tạo cho LCD và DS1307 như sau

Trong Tab ta chọn chíp DS1307 tích vào ô Enable để sử dụng DS1307 trong ô Square Wave Output ta cũng tích vào ô Enable, tại ô Freq ta chọn tần số 1Hz để khởi tạo cho chân output của DS1307 cứ 1s có 1 xung ra. Sau đó chọn File/ Save, Grenate and Exit

Sơ đồ làm việc với DS1307 như sau:





2. Ngay trước vòng While (1) trong hàm Main ta bổ sung câu lệnh thời gian và ngày tháng cho RTC. I2C, LCD, DS1307 đã được khởi tạo sẵn từ trước bằng CodeWinzard:

```
.....
// I2C Bus initialization
i2c_init();
// DS1307 Real Time Clock initialization
// Square wave output on pin SQW/OUT: On
// Square wave frequency: 1Hz
rtc_init(0,1,0);
// LCD module initialization
lcd_init(16);
rtc_set_time (0,0,0);// khoi tao 0h 0p 0s
rtc_set_date (10,10,2011);// ngay 10 thang 10 nam 2011
.....
```

Để đọc được thời gian ta dùng các hàm rtc_get_time và rtc_get_date có sẵn trong thư viện DS1307.h

3. Tiếp theo chúng ta cần khai báo 3 biến lưu thông tin về thời gian là giờ □ h; phút □ m; giây □ s; và 3 biến lưu thông tin về thời gian là ngày □ day; tháng □ month; năm □ year ; ngay trước hàm Main như sau:

```
.....
unsigned h,m,s; //khai bao gio phut giay
unsigned day,month,year;// khai bao ngay thang nam
void main(void)
.....
```

4. Tiếp theo để hiển thị giờ và ngày ra LCD ta cần viết thêm chương trình hiển thị ra LCD:

```
.....
// Declare your global variables here
unsigned h,m,s; //khai bao gio phut giay
unsigned day,month,year;// khai bao ngay thang nam
void lcd_putnum (unsigned char x)
{
unsigned char chuc,tram,donvi;
tram=x/100; //lay hang tram
chuc=(x-tram*100)/10; //lay hang chuc
donvi=(x-tram*100-chuc*10); //lay hang don vi
// chuyen qua ma ASII
lcd_putchar (tram+48);
lcd_putchar (chuc+48);
lcd_putchar (donvi+48) ;
}
void main(void)
.....
```

5. Lập trình cho vòng lặp While (1)

```
.....
```

```
while (1)
{
// Place your code here
rtc_get_time(&h,&m,&s); // doc gio tu DS1307
rtc_get_date(&date,&month,&year); // Doc nagy tu DS1307
// hien thi ra LCD
lcd_gotoxy(0,0);
lcd_putnum(h);
lcd_putchar(':');
lcd_putnum(m);
lcd_putchar(':');
lcd_putnum(s);
lcd_putchar(':');
lcd_gotoxy(0,1);
lcd_putnum(date);
lcd_putchar('/');
lcd_putnum(month);
lcd_putchar('/');
lcd_putnum(year);
delay_ms(500);
};
}
```

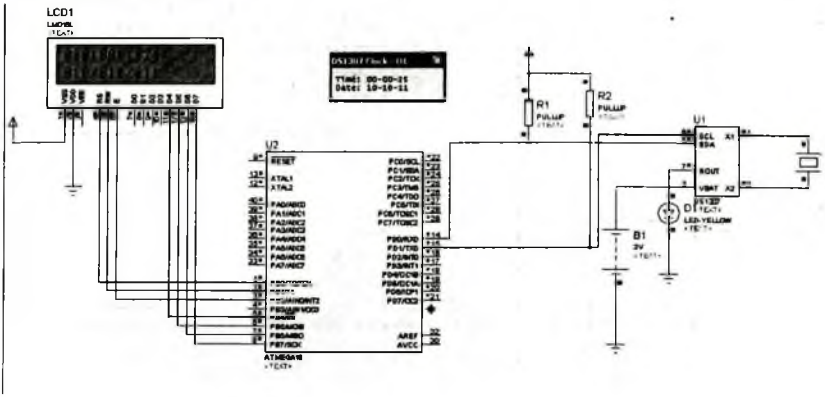
6. Biên dịch chương trình nguồn sang File I2C..hex.

7. Hoạt động thử

- Tắt công tắc nguồn.
- Bật Switch chọn kết nối I2C trên Main chính.
- Kết nối môđun LCD Main như sau: Bus data nối với PortD,
- Nạp chương trình xuống Main chính.
- Bật công tắc nguồn.

- Quan sát sự hoạt động của mạch.
- Khi hoạt động thì led nối với chân SQW có xung 1 giây.

Kết quả mô phỏng :



IV. BÁO CÁO THÍ NGHIỆM

1. Số thứ tự và tên bài
2. Mục đích thí nghiệm
3. Chương trình đã soạn thảo để thí nghiệm
4. Kết quả thí nghiệm
5. Nhận xét và kết luận.

V. CÂU HỎI KIỂM TRA

1. Hãy nêu phương thức hoạt động của bus I2C.
2. Vẽ lưu đồ và giải thích hoạt động của chương trình trên.

BÀI THỰC HÀNH SỐ 13
LẬP TRÌNH GIAO TIẾP MÁY TÍNH

Bài tập ứng dụng:

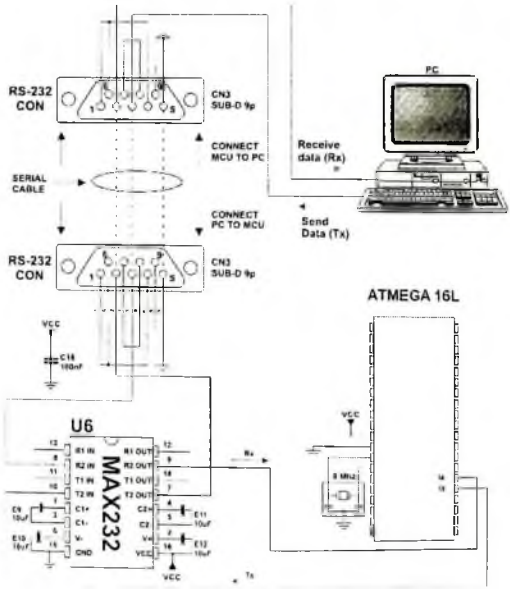
Viết chương trình thiết lập giao tiếp máy tính, truyền một dữ liệu từ máy tính xuống vi điều khiển, ấn một nút trên Main thì truyền một dữ liệu lên máy tính.

I. MỤC ĐÍCH YÊU CẦU

Học xong bài này học sinh cần nắm được các nội dung sau:

- Hiểu cách thiết lập các thông số cho chuẩn giao tiếp RS232 giữa vi điều khiển và PC.
- Biết cách gửi một giá trị lên máy tính và cách đọc một giá trị từ máy tính xuống.

II. TÓM TẮT LÍ THUYẾT



Chuẩn giao tiếp RS232

Ghép nối qua cổng nối tiếp RS-232 là một trong những kỹ thuật được sử dụng rộng rãi nhất để ghép nối các thiết bị ngoại vi với máy tính. Qua cổng nối tiếp có thể ghép nối chuột, modem, thậm chí cả máy in, bộ biến đổi AD, các thiết bị đo lường.

Việc truyền dữ liệu xảy ra trên 2 đường dẫn qua chân cắm ra TxD, gửi dữ liệu của nó đến thiết bị khác. Trong khi đó dữ liệu mà máy tính nhận được dẫn đến chân RxD. Các tín hiệu khác đóng vai trò như tín hiệu hỗ trợ khi trao đổi thông tin và vì thế không phải trong mọi ứng dụng đều dùng đến.

Mức tín hiệu trên chân ra RxD tùy thuộc vào đường dẫn TxD và thông tin thường nằm trong khoảng – 12V.. + 12V các bit dữ liệu được đảo ngược lại. Mức điện áp ở mức cao nằm trong khoảng – 3V và – 12V và mức thấp nằm trong khoảng từ + 3Vvà +12V. Trạng thái tĩnh trên đường dẫn có mức điện áp – 12V.

Lập trình truyền thông nối tiếp trên Atmega16

Atmega16 hỗ trợ rất tốt cho truyền thông nối tiếp USART. Các đặc điểm nổi bật là:

- Hoạt động toàn phần Full-duplex.

- Hỗ trợ cả hoạt động đồng bộ và không đồng bộ.

- Định dạng khung có thể lập trình (với 5,6,7,8 hoặc 9 Data bits và 1 hoặc 2 Stop Bits), tự dò lỗi định dạng khung và lỗi Data Over Run.

- Hỗ trợ truyền thông điều khiển ngắt.

- Hỗ trợ truyền thông đa xử lý.

Các đặc điểm của Bộ thu (Receiver):

- Dò trạng thái rỗi của đường truyền.

- Dò lỗi khung, nhiễu, và lỗi overrun.

- Báo trạng thái thanh ghi dữ liệu của bộ thu đầy.

Các đặc điểm của bộ phát (Transmitter) :

- Báo trạng thái thanh ghi dữ liệu bộ phát rỗng.

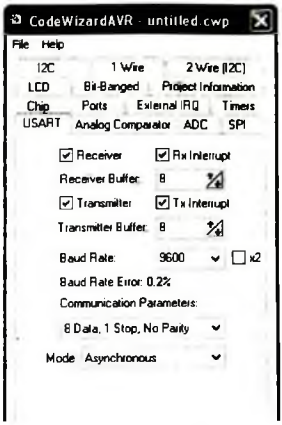
- Báo hoàn thành quá trình truyền.

- Ngắt quá trình truyền.

- Phát trạng thái rỗi lên đường truyền.

III. CÁC BƯỚC TIẾN HÀNH

1. Khởi tạo cho chương trình CodeWinzard



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.

2. 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:

.....

```
#include <stdio.h>
unsigned char temp; bien nho trung gian
// Declare your global variables here
void main(void)
```

.....

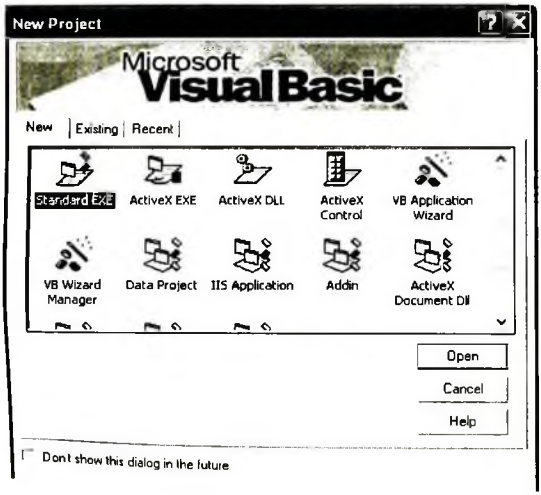
3. Lập trình cho hàm Main:

```
.....
#asm("sei")
while (1)
{
// Place your code here
if (rx_counter>0)
{
temp= getchar();
PORTA= temp;
putchar (temp);
};
}
END
```

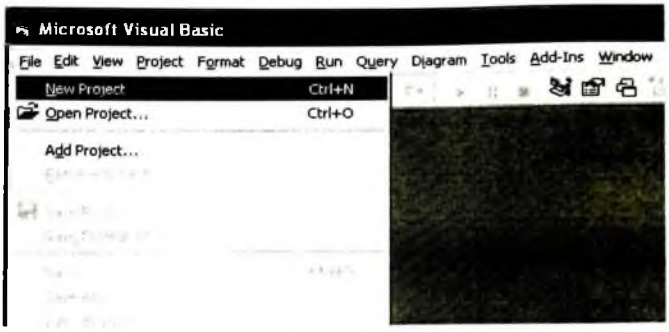
Lưu và nạp chương trình vào vi điều khiển

4. Lập trình truyền nhận với Visual Basic trên máy tính

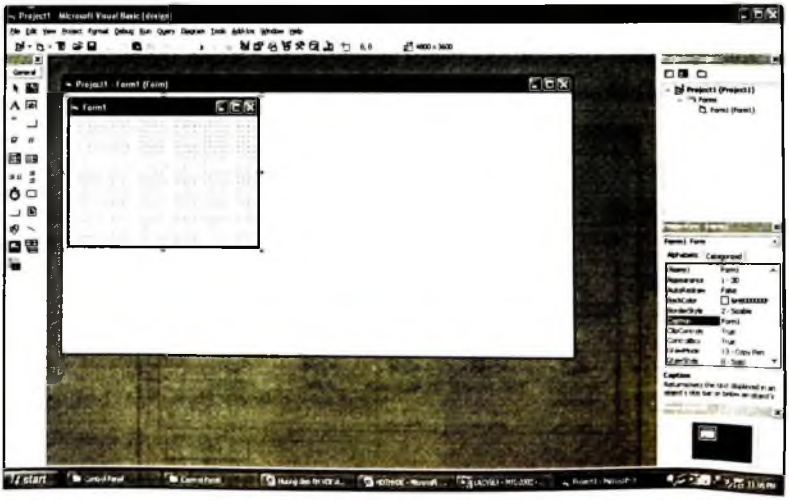
Khởi tạo Project trong VB. Kích đúp và biểu ICON của VB được cửa sổ New Project như sau:



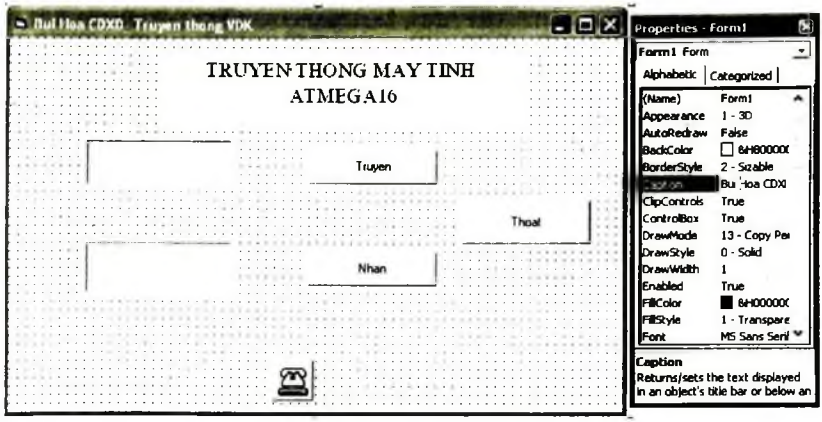
Hoặc khi đã mở một Project sẵn muốn tạo một Project mới có thể sử dụng Menu: File/New Project (phím tắt Ctrl + N). Như sau:



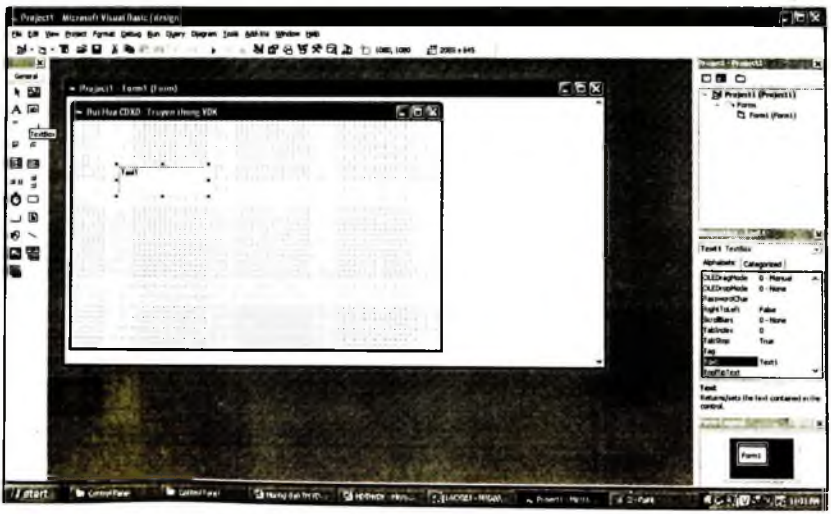
Trong cửa sổ New Project có 3 tab: New để tạo Project mới; Existing để mở một Project có sẵn; Recent: để mở các Project gần đây. Trong tab new có nhiều loại Project : Standar Exe, ActiveX exe, ActiveX DLL, ... Chúng ta chọn Standar EXE và chọn Open được Project như sau:



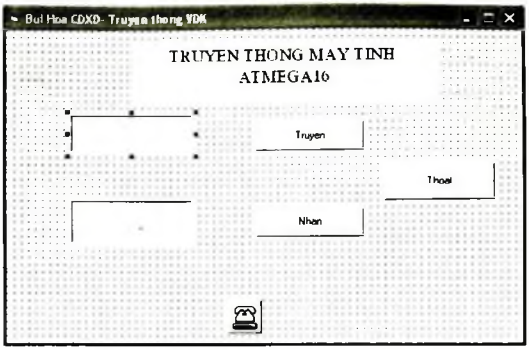
Để sửa tên của Form trong thuộc tính điều khiển của Form ta sửa Text trong mục Caption như sau :



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.

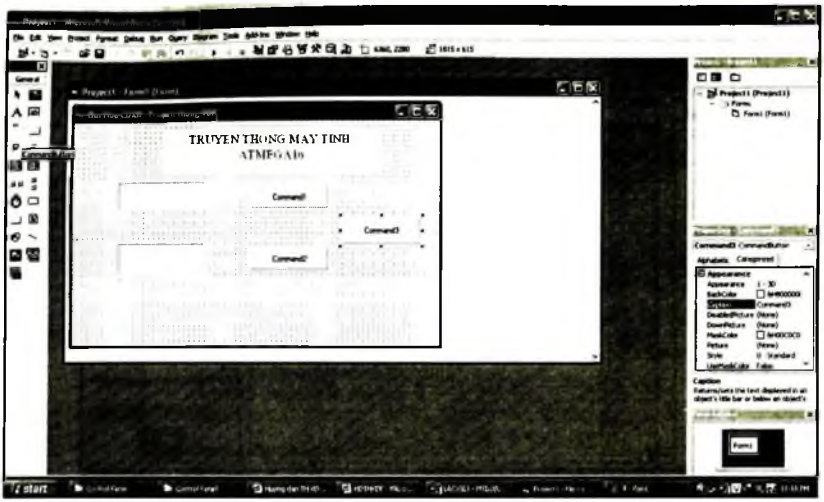


Trong phần thuộc tính của Textbox Text1, tìm ô text và xoá chữ Text1 đi. Để ô Text 1 thành trắng, để di chuyển các control ta nhấp trái chuột và dữ chặt và di chuyển tới vị trí thích hợp.

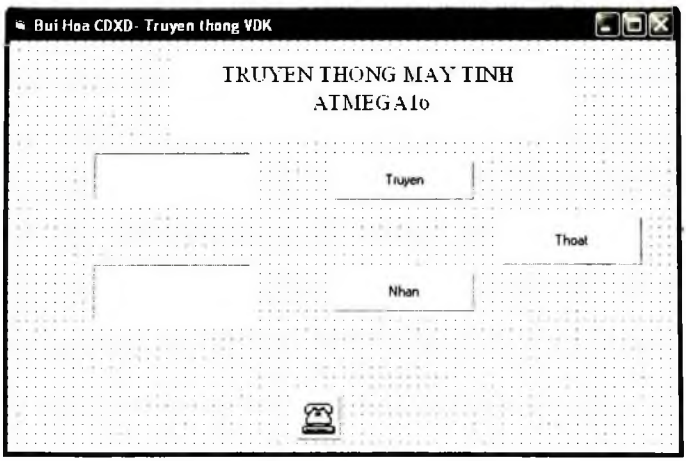


Đường biên của các Control đều có các điểm tô màu đậm, đưa trỏ chuột tới đó trỏ chuột biến thành mũi tên, nhấp trái chuột và dữ chặt để thay đổi kích thước của các control. Lấy LABEL như sau:

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



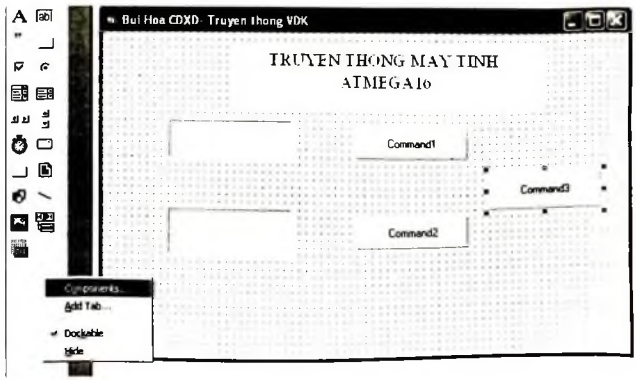
Tương tự lấy các text và các label và sắp xếp lại như sau:



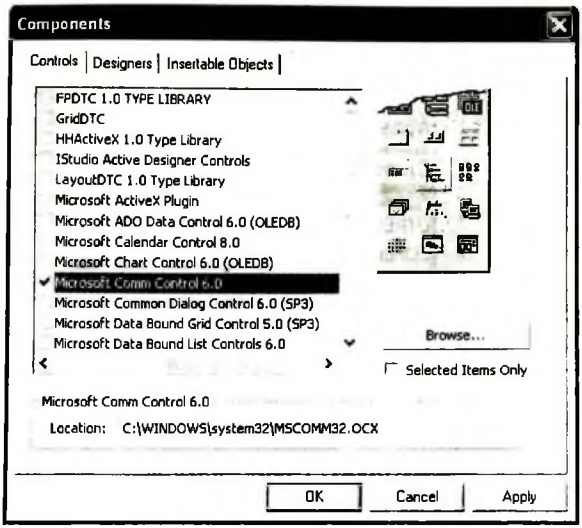
Trong trường hợp các bạn kích đúp chuột vào một điều khiển nó sẽ hiện ra cửa sổ CODE, các bạn có thể tắt nhờ dấu X trên góc trên phải mà hình:

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 hiển thị bởi các textbox1,2. Nút truyền là Command1, nút nhận 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.

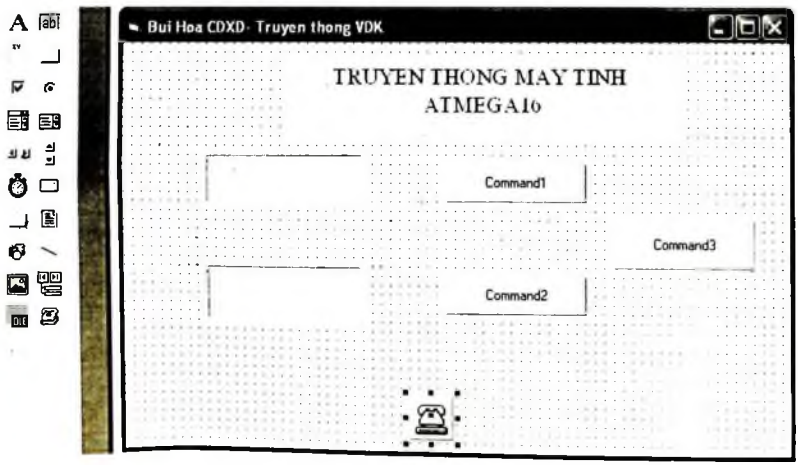


Được cửa sổ Components như sau:

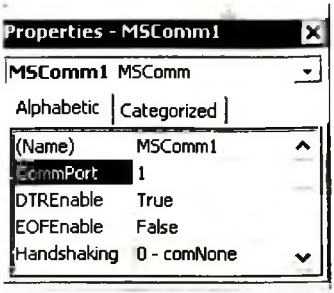


Tìm dòng Microsoft Comm Control 6.0 và check vào đó và nhấn OK. Bây giờ trên thanh công cụ có thêm một biểu tượng mới là MSCOMM control.

Kích đúp vào đó để lấy control vào Form.như sau:

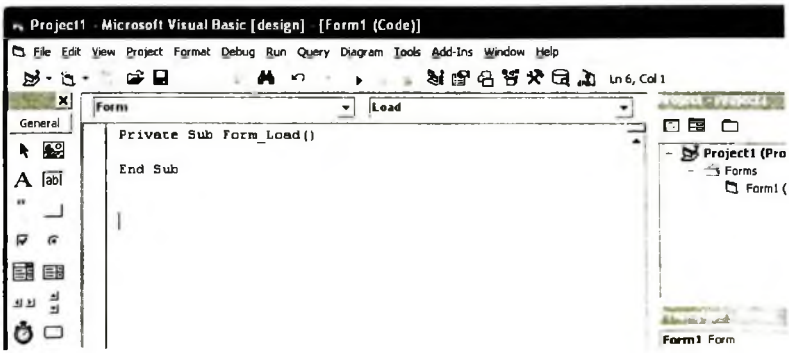


Thuộc tính mặc định cho MSCOMM như sau:

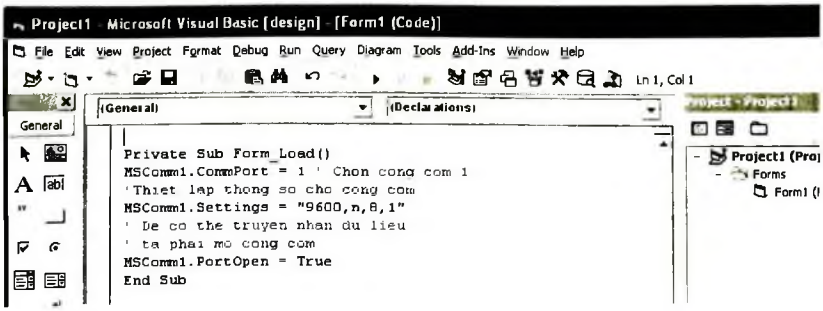


Để viết Code cho đối tượng nào ta chỉ cần nhấp đúp chuột vào đối tượng đó cửa sổ viết code sẽ hiện ra. Khi chạy chương trình thì trước hết ta cần khởi tạo cho control MSCOMM. Như vậy ta phải khởi tạo trong hàm Form_Load. Ta chuyển trỏ chuột để nó đánh dấu Form (Nhấp đúp vào khoảng trống trên Form)

Ta được cửa sổ soạn code như sau:

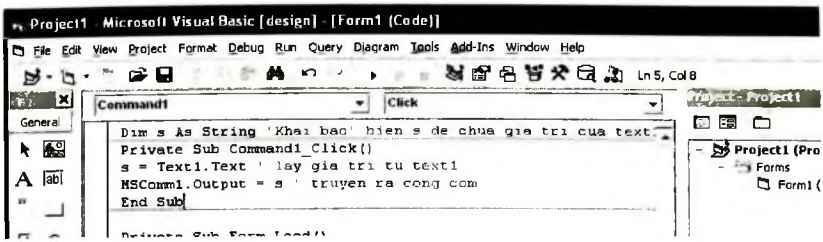


VB tự khởi tạo cho ta một hàm khi load form. Viết mã lệnh như sau:

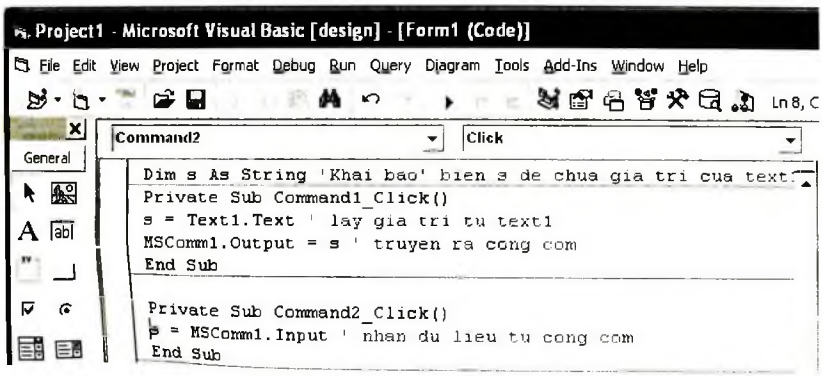


Để viết mã lệnh cho nút truyền kích đúp chuột vào button truyền:

Mã lệnh như sau:



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

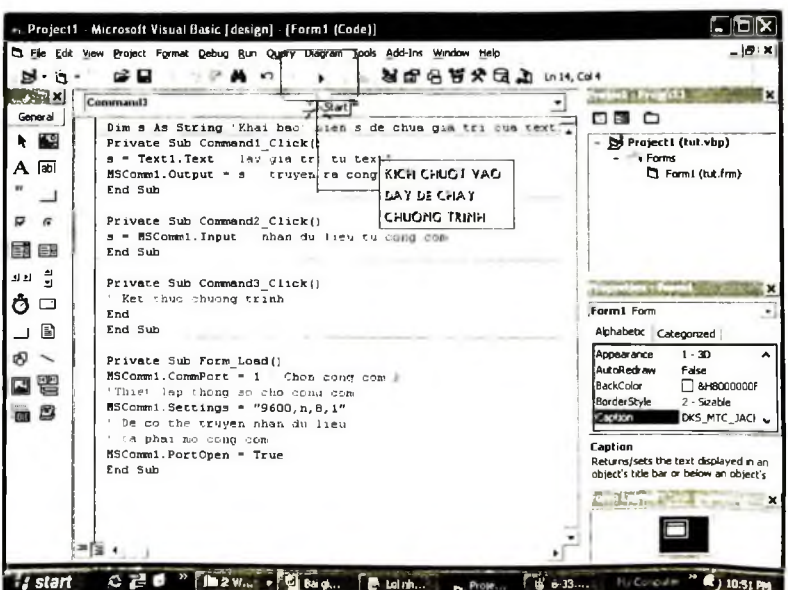


Tương tự làm cho nút EXIT:

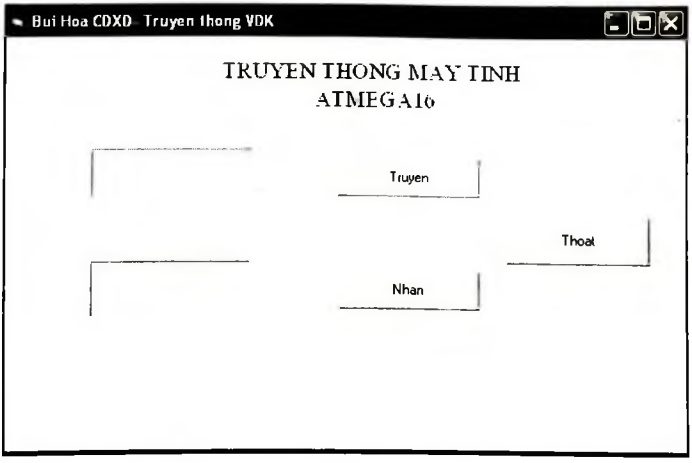
```
Private Sub Command2_Click()
s = MSComm1.Input ' nhan du lieu tu cong com
End Sub

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.

IV. BÁO CÁO THÍ NGHIỆM:

1. Số thứ tự và tên bài.
2. Mục đích thí nghiệm.
3. Chương trình đã soạn thảo để thí nghiệm.
4. Kết quả thí nghiệm:
5. Nhận xét và kết luận.

V. CÂU HỎI KIỂM TRA

1. Hãy nêu phương thức hoạt động của giao thức 232.
2. Vẽ lưu đồ và giải thích hoạt động của chương trình trên.

MỤC LỤC

HƯỚNG DẪN THỰC HÀNH VI ĐIỀU KHIỂN AVR

Chịu trách nhiệm xuất bản:
TRỊNH XUÂN SƠN

Biên tập:	NGUYỄN THỊ BÌNH
Chế bản điện tử:	TRẦN KIM ANH
Sửa bản in:	NGUYỄN THỊ BÌNH
Trình bày bìa:	VŨ BÌNH MINH

In 300 cuốn khổ 19 x 27cm tại Xưởng in Nhà xuất bản Xây dựng. Giấy chấp nhận đăng ký kế hoạch xuất bản số 18-2012/CXB/987-160/XD ngày 29-12- 2011. Quyết định xuất bản số 185/QĐ-XBXD ngày 16-7-2012. In xong nộp lưu chiểu tháng 8 -2012.