MỤC LỤC

LỜI MỞ ĐẦU 1
CHƯƠNG 1. TÌM HIỂU VỀ VI ĐIỀU KHIỂN AVR ATMEGA32 2
1.1. ĐẶT VẤN ĐỀ 2
1.2. CẤU TRÚC VI ĐIỀU KHIỂN ATMEGA32 2
1.2.1. Cấu trúc tổng quát ATMEGA32 2
1.2.2. Một số đặc trưng 5
1.3. TỔNG QUAN AVR 7
1.3.1. ALU – Arithmetic Logic Unit 8
1.3.2. Thanh ghi trạng thái – Status Register 8
1.3.3. Tập các thanh ghi làm việc đa năng 9
1.3.4. Con trỏ ngăn xếp – Stack Pointer (SP) 10
1.3.5. Xử lý reset và ngắt – Reset and Interrupt Handling 11
1.4. TỔ CHỨC BỒ NHỚ TRONG ATMEGA32 – AVR ATMEGA32 13
1.4.1. Hệ thống bộ nhớ lập trình lại Flash 13
1.4.2. Bộ nhớ dữ liệu SRAM 13
1.4.3. Bộ nhớ dữ liệu EEPROM 14
1.5. MÔ TẢ THANH GHI - THANH GHI ĐIỀU KHIỂN NGẮT THÔNG THƯỜNG 14
1.6. I/O PORT 15
1.6.1. Port với chức năng IO thông thường 15
1.6.2. Mô tả thanh ghi trong I/O Port 16
1.7. NGẮT NGOÀI - EXTERNAL INTERRUPTS 17
1.7.1. Thiết lập kiểu ngắt – Interrupt Sense Control 17
1.7.2. Cho phép ngắt – Interrupt Request Enable 18
1.7.3. Cờ ngắt – Interrupt Flag 18
1.8. BỘ ĐỊNH THỜI 8BIT TIMER/COUNTER 0 18
1.8.1. Hoạt động của bộ Timer/Couter 19

1.8.2. Đơn vị đếm 21
1.8.3. Đơn vị so sánh ngõ ra 21
1.9. BỒ ĐỊNH THỜI/ĐẾM TIMER/COUNTER 1 16-BIT 22
1.9.1. Sơ đồ khối và một số đặc điểm 22
1.9.2. Một số định nghĩa 23
1.10. SPI (SERIAL PERIPHERAL INTERFACE) 23
1.10.1. Sơ đồ và định nghĩa 23
1.11. BỘ SO SÁNH TƯƠNG TỰ (ALALOG COMPARATOR) 25
1.12. HỆ THỐNNG XUNG CLOCK 27
1.13. BỘ BIẾN ĐỔI A/D 28
CHƯƠNG 2. GIAO TIẾP QUA CỔNG COM 31
2.1. GIỚI THIỆU 31
2.2. ƯU ĐIỂM CỦA GIAO DIỆN NỐI TIẾP RS232 32
2.3. NHỮNG ĐẶC ĐIỂM CẦN LƯU Ý TRONG CHUẨN RS232 32
2.4. CÁC MỨC ĐIỆN ÁP ĐƯỜNG TRUYỀN 32
2.5. CỔNG RS232 TRÊN PC 33
2.6. QUÁ TRÌNH DỮ LIỆU 34
CHƯƠNG 3. THIẾT KẾ HỆ THỐNG GHÉP NỐI MÁY TÍNH ĐO LƯỜNG NHIỀU KÊNH 36
3.1. THIẾT KẾ HỆ THỐNG 36
3.2. XÂY DỰNG CHƯƠNG TRÌNH PHẦN MỀM 47
3.2.1. Chương trình cho PC 47
3.2.2. Chương trình viết cho vi điều khiển 52
KẾT LUẬN. 56
TÀI LIỆU THAM KHẢO 57

LỜI MỞ ĐẦU

Ngành công nghệ điện tử phát triển nhanh và ngày càng nhiều các sản phẩm xuất hiện trong đời sống, trong sản xuất hay trên thị trường nhằm giúp cho con người cải thiện chất lượng cuộc sống. Trong quá trình công nghiệp hóa hiện đại hóa ngày nay, vấn đề cần thiết là phải áp dụng công nghệ tự động vào trong quá trình sản xuất cũng như quản lý nhằm nâng cao hơn nữa năng suất làm việc của công nhân. Trên tình thần đó em chọn đề tài **"*Thiết kế hệ thống ghép nối máy tính đo lường nhiều kênh*"** làm đồ án tốt nghiệp.

Trong quá trình nhận đề tài với sự nỗ lực của bản thân và sự giúp đỡ tận tình của thầy Nguyễn Văn Dương, em đã hoàn tất xong cuốn đồ án này. Tuy nhiên do thời gian cũng như kinh nghiệm bản thân có hạn nên bản đồ án này không tránh được những sai sót, em rất mong được sự đóng góp ý kiến chỉ bảo của các thầy cô và các bạn.

Cuối cùng em xin chân thành cảm ơn các thầy cô giáo trong khoa Điện - Điện tử của trường Đại Học Dân Lập Hải Phòng đã tạo điều kiện và giúp đỡ để em hoàn thành cuốn đồ án này. Đặc biệt em xin chân thành cảm ơn Nguyễn Văn Dương giảng viên hướng dẫn chính đã tận tình hướng dẫn chỉ bảo em trong suốt quá trình học tại trường cũng như trong thời gian làm đồ án vừa qua.

Sinh Viên

Vũ Duy Đoàn

CHƯƠNG 1.

TÌM HIỂU VỀ VI ĐIỀU KHIỂN AVR ATMEGA32

1.1. Đặt vấn đề

Trên thị trường có hàng trăm loại vi xử lý và vi điều khiển vì thế việc lựa chọn 1 loại cụ thể phù hợp với ứng dụng của ta trở thành một công việc hết sức khó khăn. Thông thường việc lựa chọn phụ thuộc vào một số yếu tố như: tính năng công việc, giá thành, thị trường, khả năng thiết kế, ... Nếu xét trên phương diện giá thành thì họ vi điều khiển AVR có giá thành cao gấp nhiều lần so với vi điều khiển loại cũ như 89C51, nhưng xét trên phương diện chức năng và ứng dụng thì giá thành của AVR lại rẻ hơn rất nhiều. Để có thể có những chức năng như AVR thì 89C51 cần rất nhiều mạch hỗ trợ bên ngoài, giá thành của những mạch bên ngoài sẽ làm tăng giá thành chung và kích cỡ mạch, công suất tiêu thụ vì thế cũng tăng lên rất nhiều. Ngược lại, với AVR được tích hợp nhiều thành phần ngoại vi trên cùng một vỏ chip nên kết cấu mạch nhỏ gọn hơn nhiều theo đó giá thành và công suất tiêu thụ cũng giảm đi.

Ngày nay những ứng dụng điện tử và điều khiển đòi hỏi phải thật nhỏ gọn và có trình độ công nghệ cao. Người làm kĩ thuật luôn luôn tìm tòi, khám phá những thành tựu công nghệ. Vì những lý do trên, em quyết định chọn họ vi điều khiển AVR mà cụ thể là vi điều khiển ATMEGA32 làm dối tượng nghiên cứu phục vụ cho đề tài.

1.2. Cấu trúc vi điều khiển ATMEGA32

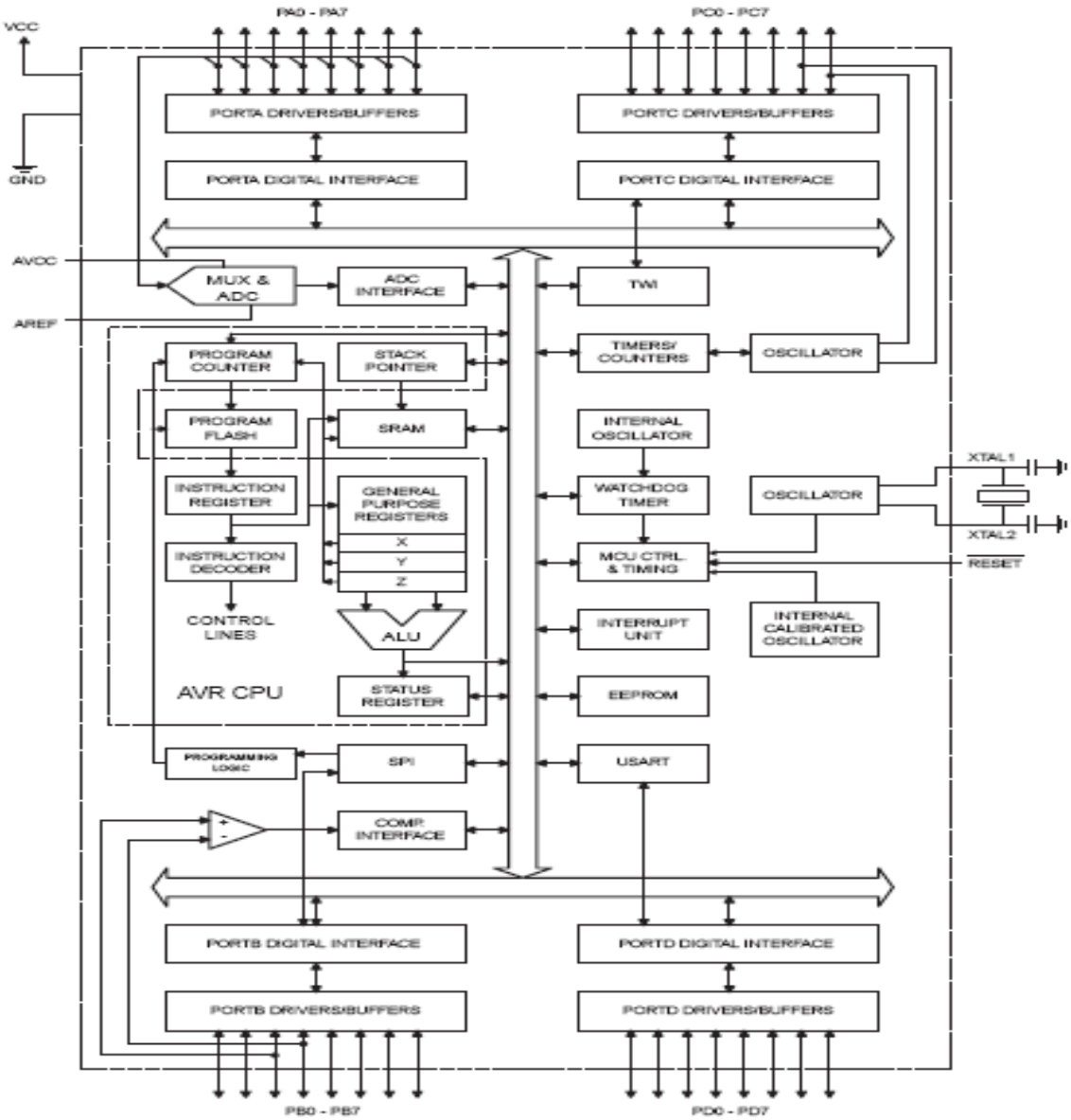
1.2.1. Cấu trúc tổng quát ATMEGA32

ATMEGA32 là loại vi điều khiển CMOS, nguồn thấp, 8 bit, xây dựng trên nền tảng cấu trúc tập lệnh thu gọn tiên tiến cho AVR .

- RISC – Reduced Instruction Set Computer.
- CISC – Complex Instruction Set Computer.

Khả năng thực thi 1MIPS (Mega Instruction Per Second) trên 1MHz.

Bao gồm 32 thanh ghi làm việc (General Purpose Working Register) liên kết trực tiếp với bộ xử lý số học ALU (Arithmetic Logic Unit).

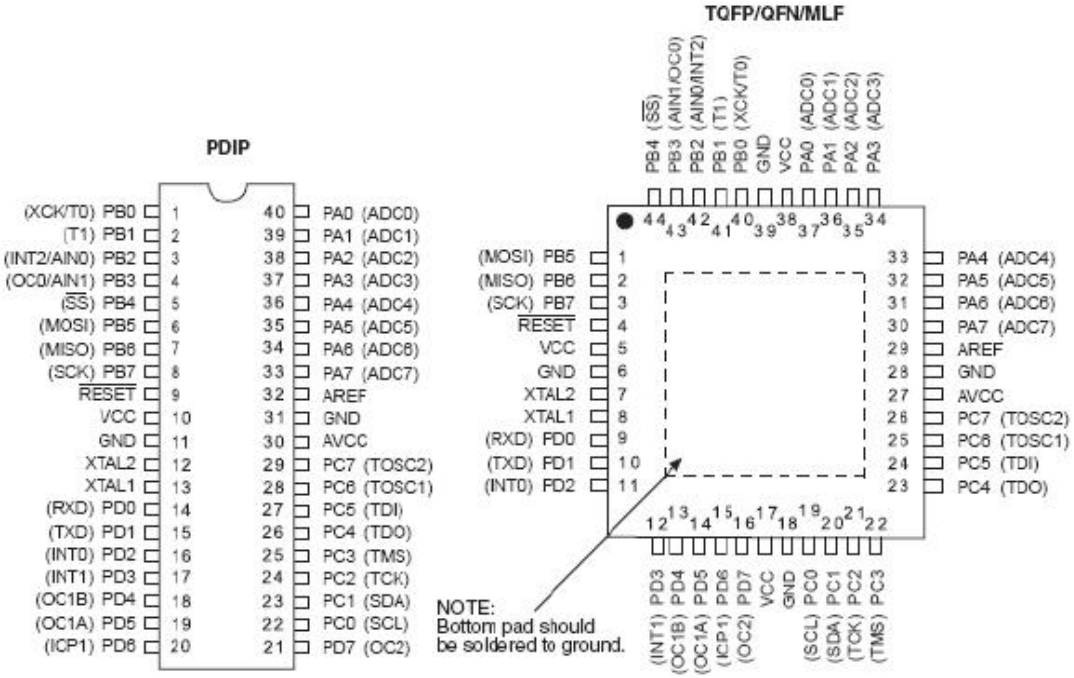


Hình 1.1. Sơ đồ khối của vi điều khiển ATMEGA32

Atmega32 có các tính năng sau:

- 32K byte Flash có khả năng lập trinh được tương thích hoạt động Read-While-Write.

- 1024 byte EEPROM.
- 2K byte SRAM.
- 32 cổng xuất nhập đa dụng.
- Tính năng On-chip debug.
- Ngõ JTAG.
- 3 Timer/Counter.
- Internal và External Interrupt.
- USART.
- TWI.
- 8 kênh ADC 10-bit.
- Watch_dog timer với bộ dao động on_chip riêng biệt.
- SPI.
- Tính năng ISP thông qua cổng SPI hoặc Boot Loader.



Hình 1.2. Sơ đồ chân Atmega32

+ VCC: Điện áp nguồn nuôi.

+ GND: Nối mass.

+ PortA (PA7…PA0): PortA nhận vào tín hiệu tương tự (Analog) và chuyển đổi thành tín hiệu số (Digital). Ngoài ra PortA có thể được tách ra làm vào/ra 2 hướng 2 bits nếu bộ chuyển đổi A/D không được sử dụng. Khi các chân PA0 đến PA7 là các lối vào và được đặt xuống chế độ thấp từ bên ngoài, chúng sẽ là nguồn dòng nếu các điện trở nối lên nguồn dương được kích hoạt. Các chân của PortA ở vào trạng thái có điện trở cao khi tín hiệu Reset ở chế độ tích cực hoặc ngay cả khi không có tín hiệu xung đồng hồ.

Port A cung cấp các đường địa chỉ/dữ liệu vào/ra hoạt động theo kiểu đa hợp kênh khi dùng bộ nhớ SRAM ở bên ngoài.

+ PortB, D: tương tự như PortA.

+ PortC (PC7…PC0): tương tự như PortA. Nhưng nếu cho phép giao diện JTAG, thì các chân PC5, PC3, PC2 sẽ hoạt động ngay cả khi reset lại tín hiệu.

+ Reset: Lối vào đặt lại. Bộ vi điều khiển sẽ được đặt lại khi chân này ở chế độ thấp trong hơn 50ns, các xung ngắn hơn không tạo ra tín hiệu đặt lại.

+ XTAL1: Lối vào bộ khuếch đại đảo và lối vào mạch tạo xung nhịp bên trong.

+ XTAL2: Lối ra bộ khuếch đại đảo: XTAL1 và XTAL2 lần lượt là lối vào và lối ra của một bộ khuếch đại đảo. Bộ khuếch đại này được bố trí để làm bộ tạo dao động trên chip. Một bộ tinh thể thạch anh hoặc một bộ cộng hưởng gốm có thể được sử dụng. Để điều khiển bộ vi điều khiển từ một nguồn xung nhịp bên ngoài, chân XTAL2 để trống, còn chân XTAL1 được nối với bộ dao động bên ngoài.

+ AREF: Là chân chuyển đổi tín hiệu analog cho bộ chuyển đổi A/D.

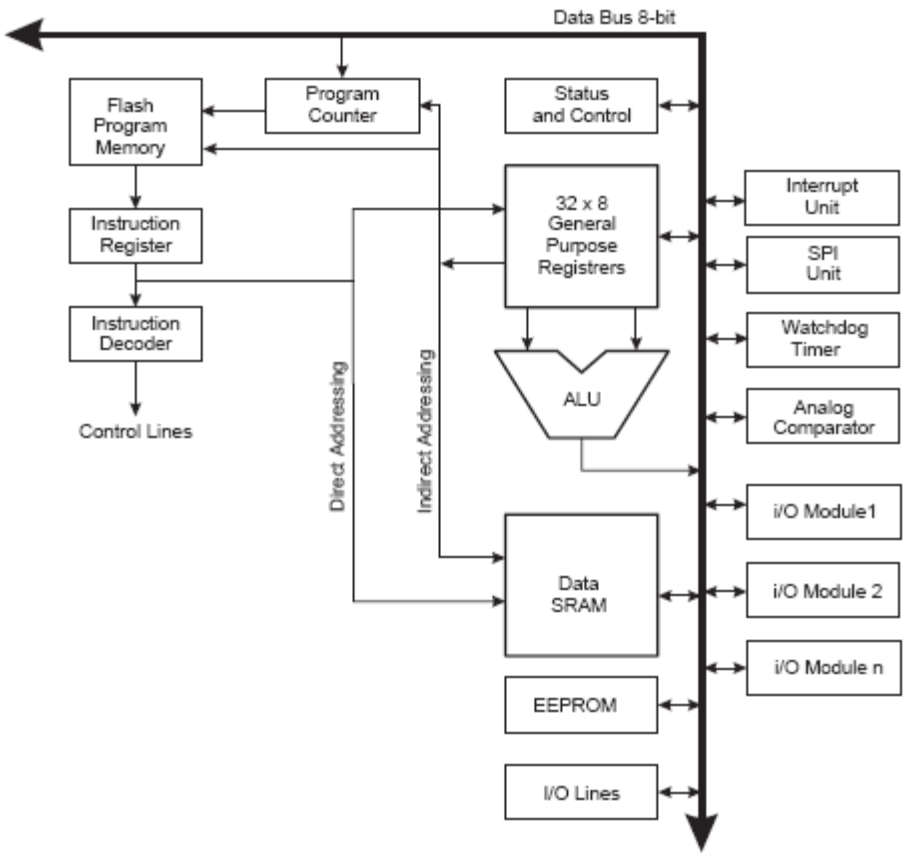
+ AVCC: Là chân nguồn cho PortA và cho bộ chuyển đổi A/D. Nó có thể tự kết nối với nguồn chính ngay cả khi ADC không được sử dụng.

1.2.2. Một số đặc trưng

Hoạt động: Các lệnh được chứa trong bộ nhớ chương trình Flash Memory dưới dạng các thanh ghi 16 bit. Bộ nhớ chương trình được truy cập trong mỗi

chu kỳ xung Clock và một lệnh chứa trong Program memory được nạp vào thanh ghi lệnh (instruction Register), thanh ghi lệnh tác động cũng như lựa chọn tệp thanh ghi cũng như RAM cho ALU thực thi. Trong khi thực thi chương trình, địa chỉ của dòng lệnh đang thực thi được quyết định bởi bộ đếm chương trình – PC (Program Counter).

AVR có ưu điểm là hầu hết các lệnh đều được thực thi trong một chu kỳ xung nhịp, vì thế mà trong một số trường hợp dù nguồn clock của AVR nhỏ hơn một số loại vi điều khiển khác (như PIC) nhưng thời gian thực thi vẫn nhanh hơn.



Hình 1.3. Cấu trúc bên trong của AVR

1.3. Tổng quan AVR

Lõi AVR sử dụng kiến trúc Harvard – với các bus riêng biệt cho chương trình và dữ liệu. Lệnh từ bộ nhớ chương trình thực thi thông qua một ống đơn cấp. Khi một lênh đang thực thi, lệnh tiếp theo sẽ được nhốt (pre-fetch) từ bộ nhớ chương trình, cho phép các lệnh được thực thi trong mỗi chu kì clock.

Các 32 thanh ghi (8-bit) làm việc cho phép truy xuất nhanh trong 1 chu kỳ clock. Trong hoạt động thông thường của ALU, 2 toán hạng xuất ra từ thanh ghi làm việc, lệnh thực thi, và kết quả lưu ngược lại thanh ghi làm việc chỉ trong 1 chu kì clock. 6 trong số 32 thanh ghi được dùng như con trỏ địa chỉ gián tiếp 16-bit sử dụng cho địa chỉ không gian dữ liệu. 1 trong 3 thanh ghi địa chỉ này có thể dùng như con trỏ địa chỉ look-up table trong bộ nhớ Flash.

Bộ ALU hỗ trợ các hoạt động tính toán số học và logic giữa thanh ghi với nhau, hay giữa thanh ghi với hằng số. Các hoạt động từng thanh ghi đơn cũng được thực hiện trong ALU. Sau khi tính toán, thanh ghi trạng thái (Status Register) cập nhật thông tin liên quan đến kết quả tính toán.

Dòng chương trình (Program Flow) được cung cấp bởi các lệnh nhảy có điều kiện hoặc không điều kiện, và có thể định địa chì trực tiếp đến toàn bộ không gian địa chỉ. Hầu hết các lệnh trong AVR đều ở dạng 16-bit. Mỗi địa chỉ bộ nhớ chương trình chứa một lệnh 16 hoặc 32-bit.

Bộ nhớ chương trình chia ra làm 2 phần: Boot Loader và vùng ứng dụng. Cả 2 đều sử dụng các lockbit để bảo vệ đọc/ghi. Lệnh SPM thực thi việc ghi dữ liệu vào vùng flash ứng dụng phải được đặt trong vùng Boot Loader.

Trong quá trình ngắt hay hàm/chương trình con được gọi, địa chỉ trả về của bộ đếm chương trình lưu trong ngăn xếp (stack). Stack được phân bổ hiệu quả trong 1 phần bộ nhớ SRAM, vì vậy, độ lớn của stack chỉ phụ thuộc vào SRAM và việc sử dụng SRAM. Chương trình người dùng cần phải khởi tạo giá trị này cho SP – Con trỏ ngăn xếp (Stack Pointer) trong chương trình sau khi reset và trước khi thực hiện bất kì việc gọi hàm hay chương trình ngắt được thực thi.

Module ngắt linh hoạt có thanh ghi điều khiển riêng trong không gian IO và có bit cho phép ngắt toàn cục trong thanh ghi trạng thái (Status Register). Tất cả các ngắt đều có vector ngắt riêng trong bảng vector ngắt. Các ngắt có ưu tiên ngắt theo đúng vị trí ngắt của nó. Địa chỉ ngắt càng thấp thì độ ưu tiên ngắt càng cao.

1.3.1. ALU – Arithmetic Logic Unit

Bộ ALU hiệu suất cao của AVR hoạt động trong liên kết trực tiếp với 32 thanh ghi làm việc. Trong 1 chu kì clock, hoạt động tính toán số học giữa các thanh ghi hoặc giữa thanh ghi với dữ liệu trực tiếp sẽ được thực thi. Hoạt động của ALU được chia ra làm 3 phần chính: xử lý số học, phép toán logic và các phép toán với bit. Một số bổ sung trong kiến trúc cũng cho phép sử dụng các nhân tử hiệu quả, cho cả không dấu/có dấu và định dạng phân số.

1.3.2. Thanh ghi trạng thái – Status Register

Thanh ghi này chứa kết quả liên quan đến lệnh xử lý số học gần nhất. Kết quả chứa trong thanh ghi này có thể được sử dụng để thực hiện các hoạt động có điều kiện.

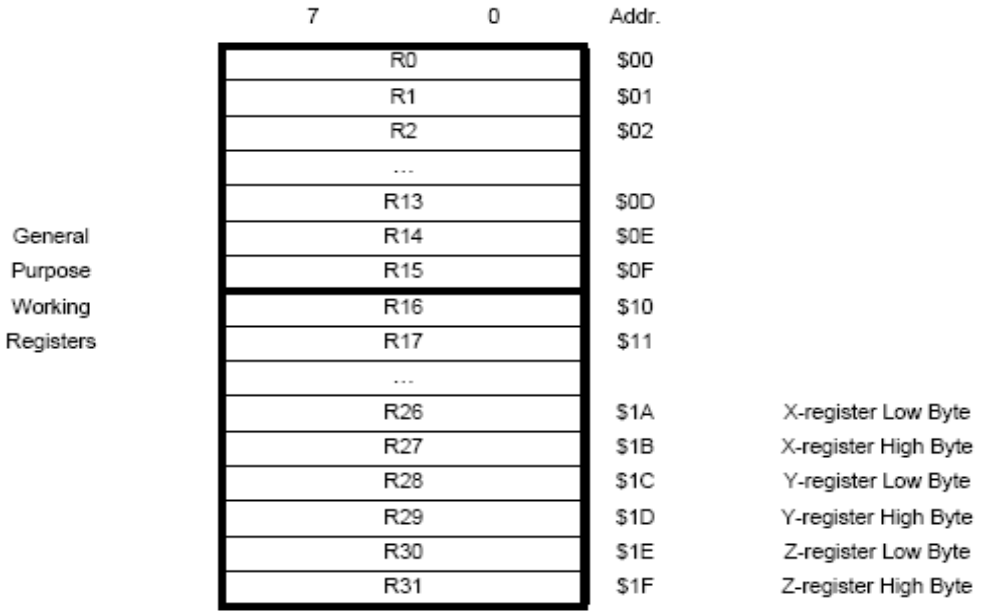
Thanh ghi trạng thái không tự động lưu lại khi nhảy vào interrupt và cũng không tự động phục hồi (restore) khi quay về, cần thực hiện điều này bằng phần mềm.

Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
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-I : cho phép ngắt toàn cục. Không cho phép người sử dụng tự ý xóa. Chỉ bị xóa khi có ngắt mới xuất hiện và được chỉ dẫn bởi RETI cho những ngắt kế tiếp. Có thể được đặt hoặc được xóa bởi SEI và CLI.
- Bit 6-T : bit lưu trữ. Có thể chép từ BST sang BLD và ngược lại.

- Bit 5-H : Cờ nhớ nửa H sử dụng để nhớ nửa hữu ích trong phép tính số BCD.
- Bit 4-S : tín hiêu bit S = N+ V.
- Bit 3-D : 2 cờ tràn.
- Bit 2-N : cờ phủ định.
- Bit 1-Z : cờ zero.
- Bit 0-C : cờ nhớ.

1.3.3. Tập các thanh ghi làm việc đa năng

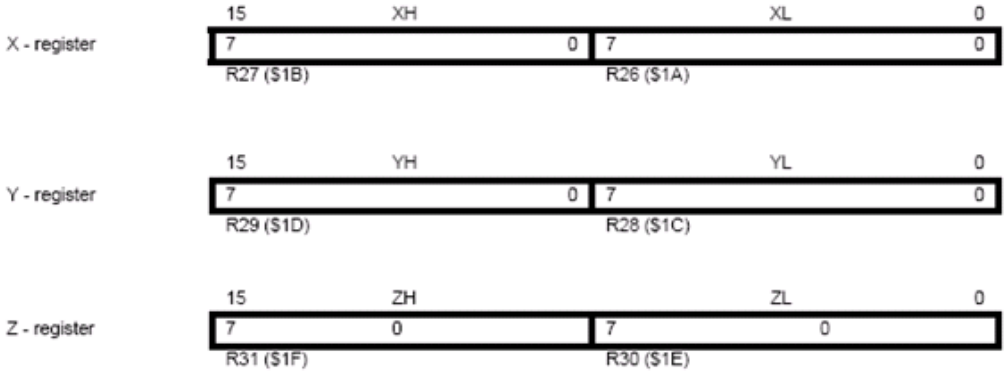


Hình 1.4. Các thanh ghi hỗ trợ làm việc AVR CPU

Tất cả các lệnh đều thực thi trên các thanh ghi làm việc có thể truy xuất trực tiếp đến các thanh ghi, và hầu hết là các lệnh thực thi trong 1 chu kì clock.

Như trên hình 1.4, tất cả các thanh ghi được gán địa chỉ bộ nhớ dữ liệu, ánh xạ chúng trực tiếp đến 32 phân vùng đầu tiên trong không gian dữ liệu. Mặc dù không được hiện thực vật lý như phân vùng SRAM, nhưng tổ chức bộ nhớ này cung cấp khả năng truy xuất phức tạp tuyệt vời của các thanh ghi, như than ghi con trỏ X-, Y- và Z- có thể được set để định vị đến bất kì thanh ghi nào trong tập thanh ghi.

Thanh ghi X, Y và Z: Các thanh ghi làm việc từ R26 đến R31 có các chức năng phụ trợ. Những thanh ghi này là con trỏ địa chỉ 16-bit để định địa chỉ gián tiếp trong không gian dữ liệu.



Trong các chế độ định địa chỉ khác nhau, các thanh ghi này có các chức năng như chuyển dịch cố định, tự động tăng/giảm.

1.3.4. Con trỏ ngăn xếp – Stack Pointer (SP)

Stack được sử dụng với mục đích chính là lưu trữ các dữ liệu tạm thời, các biến cục bộ và lưu trữ giá trị địa chỉ trả về sau khi gọi chương trình ngắt hay hàm/chương trình con (subroutine). Thanh ghi con trỏ địa chỉ luôn luôn trỏ tới đỉnh của stack (Top of the Stack). Lưu ý rằng, stack được hiện thực theo cách giảm từ địa chỉ bộ nhớ cao hơn xuống thấp hơn. Ngầm định rằng lệnh PUSH (cất) sẽ giảm giá trị con trỏ SP.

SP trỏ tới vùng SRAM nơi mà stack của chương trình interrupt và subroutine phân bổ. Không gian cho stack phải được chương trình phần mềm định nghĩa trước khi thực thi subrountine hay ngắt xảy ra (kích hoạt ngắt).

SP cần phải trỏ tới địa chỉ từ trên $60.

SP sẽ giảm đi 1, khi dữ liệu được cất vào stack sau khi thực hiện lệnh PUSH, và stack sẽ giảm đi 2 khi giá trị trả về được cất vào stack khi gọi subrountine hay chương trình ngắt.

SP sẽ tăng lên 1 khi dữ liệu được đẩy (pop) ra khỏi stack với lệnh POP, và SP sẽ tăng lên 2 khi dữ liệu được đẩy ra khỏi stack với lệnh RET khi trở về từ subrountine hay lệnh RETI khi trở về từ chương trình ngắt.

SP trong AVR hiện thực hóa từ 2 thanh ghi 8-bit.

Bit	15	14	13	12	11	10	9	8	
	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	SPH
	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	SPL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	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	0	0	0	0	
	0	0	0	0	0	0	0	0	

1.3.5. Xử lý reset và ngắt – Reset and Interrupt Handling

AVR cung cấp các nguồn ngắt khác nhau. Mỗi vector ngắt và reset này có vector chương trình riêng rẽ trong không gian bộ nhớ chương trình. Tất cả các ngắt đều được gán bit cho phép ngắt riêng biệt cùng với bit cho phép ngắt toàn cục, tất cả các bit trên phải được set lên 1. Phụ thuộc vào giá trị của PC, các ngắt có thể bị cấm tự động khi Boot Lock bit BLB02 và BLB12 được lập trình. Tính năng này đảm bảo chức tính bảo mật phần mềm.

Địa chỉ thấp nhất trong không gian bộ nhớ mặc định là được định nghĩa là vector Reset và vector ngắt. Địa chỉ vector ngắt cũng đồng thời quy định ưu tiên ngắt cho, địa chỉ ngắt thấp hơn có mức ưu tiên ngắt cao hơn và ngược lại. RESET có mức ưu tiên ngắt cao nhất, kế đến là INT0 – External Interrupt 0 Request.

Các vector ngắt có thể được di chuyển lên vị trí bắt đầu của khu vực Boot Flash bằng bit IVSEL trong thanh ghi General Interrupt Control Register (GICR). Vector RESET cũng có thể dời lên vị trí bắt đầu của Boot Flash bằng cách thiết lập bit cầu chì BOOTSRT.

Khi có ngắt xảy ra, bit cho phép ngắt toàn cục (Global Interrrupt Enable bit) sẽ bị xóa tự động bằng phần cứng, do đó tất cả các ngắt đều bị cấm. Phần mềm có thể set bit này lên 1 trở lại để cho phép ngắt trùm (nested interrupt) có

thể xảy ra. Tất cả các ngắt được kích hoạt sau đó có thể "ngắt" trong chương trình xử lý ngắt (Interrupt subrountine) hiện tại. Bit này cũng tự động được set lên 1 trở lại sau khi quay về từ chương trình ngắt khi thực thi lệnh RETI.

Có 2 dạng ngắt cơ bản:

+ Dạng thứ nhất: Ngắt được kích hoạt (trigger) bởi sự kiện đặt cờ ngắt (Interrupt Flag). Với những ngắt này, PC được trỏ tới vector chương trình ngắt nhằm thực thi chương trình xử lý ngắt, và phần cứng sẽ xóa cờ ngắt tương ứng này. Cờ ngắt cũng có thể được xóa bằng phần mềm bằng cách ghi 1 vào vị trí bit của cờ ngắt đó. Nếu một điều kiện ngắt xảy ra trong khi bit cho phép ngắt đó bị xóa, thì cờ ngắt vẫn được đặt và ngắt sẽ xảy ra khi ngắt đó được cho phép trở lại, hoặc phần mềm phải xóa cờ ngắt đó đi bằng cách ghi 1. Tương tự như vậy, nếu một hay nhiều điều kiện ngắt xảy ra trong khi bit cho phép ngắt toàn cục bị xóa, các cờ ngắt tương ứng đó sẽ được set lên, và đến khi nào bit cho phép ngắt toàn cục được đặt trở lại, thì các ngắt đó sẽ được thực thi tương ứng theo thứ tự ưu tiên ngắt.

+ Dạng thứ hai: Ngắt được kích hoạt khi nào điều kiện ngắt vẫn còn hiện hữu. Những ngắt này không cần có cờ ngắt. Nếu điều kiện ngắt ngừng xuất hiện trước khi ngắt được phép hoạt động, ngắt đó sẽ không xảy ra.

Khi AVR thoát ra từ chương trình ngắt, nó luôn quay về chương trình chính và sẽ thực hiện thêm một hay vài lệnh nữa trước khi phục vụ các ngắt khác đang treo.

Lưu ý rằng thanh ghi trạng thái (Status Register) sẽ không tự động lưu lại khi nhảy vào chương trình ngắt, cũng như không được phục hồi lại khi quay về từ chương trình ngắt. Điều này cần thực hiện bằng phần mềm.

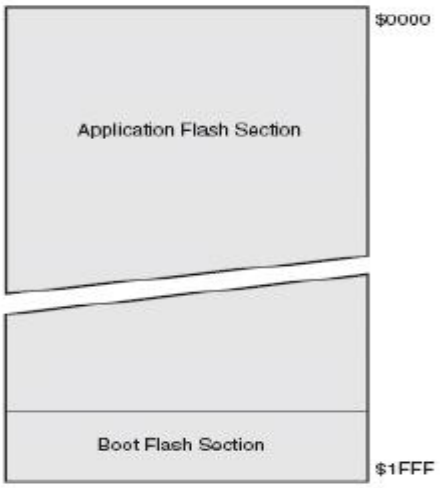
Khi sử dụng lệnh CLI để cấm ngắt, tất cả các ngắt sẽ bị cấm ngay lập tức. Không có ngắt nào xảy ra khi thực hiện lệnh CLI, ngay cả khi ngắt xảy ra đồng thời với lệnh CLI.

1.4. Tổ chức bộ nhớ trong ATMEGA32 – AVR ATMEGA32

Bộ nhớ ATMEGA32 bao gồm 2 phần: Bộ nhớ chương trình và bộ nhớ dữ liệu. Trong ATMEAG32 còn có bộ nhớ EEPROM để lưu trữ dữ liệu.

1.4.1. Hệ thống bộ nhớ lập trình lại Flash

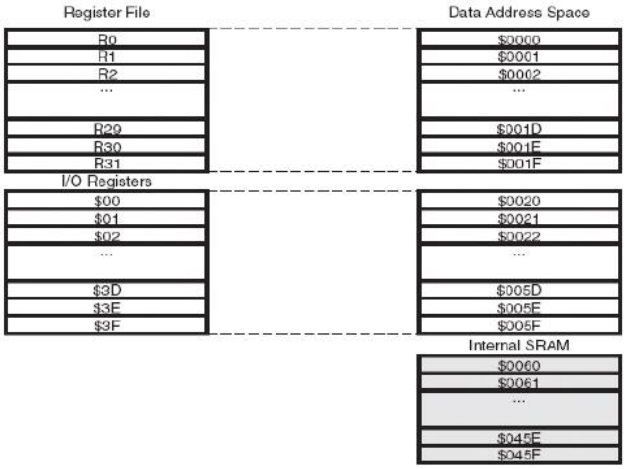
Atmega32 chứa 32K bytes On-Chip trong hệ thống bộ nhớ lập trình lại Flash để lưu trữ process. Bộ nhớ Flash có khả năng ghi/ xóa 1000 lần cho mỗi vòng.



Hình 1.5. Sơ đồ bộ nhớ process

1.4.2. Bộ nhớ dữ liệu SRAM

Có tới 1120 địa chỉ ô nhớ cho bộ nhớ nhập xuất, thanh ghi file và SRAM nội. 96 đường địa chỉ đầu tiên dành riêng cho thanh ghi file và bộ nhớ vào ra và còn 1024 đường địa chỉ còn lại cho SRAM nội.



Hình 1.6. Sơ đồ bộ nhớ dữ liệu

1.4.3. Bộ nhớ dữ liệu EEPROM

Atmega32 chứa 1024byte bộ nhớ dữ liệu EEPROM. Nó được tổ chức như một không gian dữ liệu riêng biệt, mỗi byte đơn có thể đọc và ghi. Đây là bộ nhớ dữ liệu có thể ghi xóa ngay trong lúc vi điều khiển đang hoạt động và không bị mất dữ liệu khi nguồn điện cung cấp bị cắt. Có thể ví bộ nhớ dữ liệu EEPROM giống như là ổ cứng (Hard disk) của máy vi tính. EEPROM được xem như là một bộ nhớ vào ra được đánh địa chỉ độc lập với SRAM, điều này có nghĩa là ta cần sử dụng các lệnh in, out … khi muốn truy xuất tới EEPROM.

1.5. Mô tả thanh ghi - Thanh ghi điều khiển ngắt thông thường

Bit	7	6	5	4	3	2	1	0	
	INT1	INT0	INT2	–	–	–	IVSEL	IVCE	GICR
Read/Write	R/W	R/W	R/W	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

+ Bit 1 – IVSEL: Interrupt Vector Select – Lựa chọn vector ngắt. Khi bit này được xóa, các vector ngắt sẽ nằm ở vị trí bắt đầu của bộ nhớ Flash. Khi bit này được đặt, các vector ngắt sẽ nằm ở vị trí bắt đầu của Boot Loader trong Flash. Vị trí bắt đầu thực sự của Boot Flash do các bit cầu chì BOOTSZ quyết định. Để tránh việc vô tình thay đổi bảng vector ngắt, cần phải theo trình tự sau:

- Ghi 1 vào bit Interrupt Vector Change Enable.

- Trong vòng 4 chu kì, ghi giá trị mong muốn vào IVSEL trong khi ghi 0 vào IVCE.

Ngắt sẽ tự động bị cấm khi thực hiện trình tự trên. Ngắt sẽ bị cấm trong chu kì IVCE được đặt, và nó vẫn tiếp tục bị cấm sau lệnh ghi vào IVSEL. Nếu IVSEL không được ghi thì nó vẫn tiếp tục bị cấm trong 4 chu kì tiếp theo nữa. Bit I trong SREG không bị tác động bởi việc cấm ngắt tự động này.

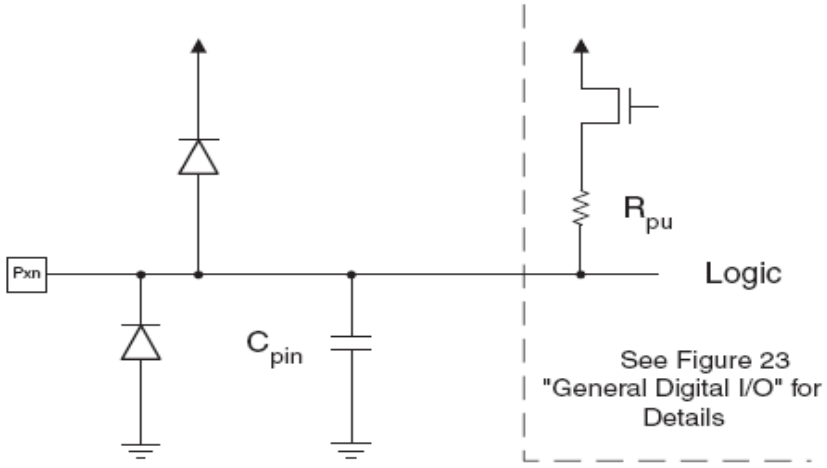
+ Bit 0 – IVCE: Interrupt Vector Change Enable – Cho phép thay đổi vector ngắt. Bit này phải được set lên trước khi ghi giá trị vào bit IVSEL. Bit này được xóa bởi phần cứng trong 4 chu kì sau khi set hoặc xóa khi ghi vào bit IVSEL.

1.6. I/O Port

1.6.1. Port với chức năng IO thông thường

Tất cả các port trên AVR đều có tính năng Đọc - Hiệu chỉnh – Ghi thực (Read – Modify – Write) khi sử dụng chúng như cổng IO số thông thường. Có nghĩa là hướng (Input/Output) của từng pin trên port có thể thiết lập mà không vô tình làm thay đổi hướng đến các pin khác trong port đó bằng lệnh SBI/CBI.

Mỗi bộ đệm ngõ ra có bộ khiển (drive) đối xứng có đặc tính nguồn và bộ thu cao. Bộ khiển trên từng pin có khả năng lái LED trực tiếp. Mỗi pin có điện trở kéo (Pulled-up resistor) có thể lựa chọn cho từng pin riêng biệt với trị số cố định phù hợp với điện áp cấp. Tất cả các IO đều có diode bảo vệ đến nguồn và đất.



Thiết lập thuộc tính cho pin

Có tất cả 4 port A, B, C, D. Mỗi port gồm 8 ngõ I/O (8-bit).

Mỗi port bao gồm 3 thanh ghi có thể truy xuất bit: DDRx, PORTx và PINx (với x có thể là A, B, C hay D)

• DDRx dùng để thiết lập hướng cho pin. Nếu set DDRx.n lên 1, pin đó là Output, nếu là mức 0, pin đó là Input (với n có thể từ 0 đến 7)

• Nếu set PORTx.n lên 1 khi pin đó đang là input, có nghĩa là chức năng pull-up resistor được kích hoạt. Để tắt chức năng này, PORTx.n phải xóa về 0 hoặc pin đó được thiết lập là ngõ Output. Port pin ở trạng thái thả nổi (tri-state) khi reset được kích hoạt và không có clock đang chạy.

• Nếu PORTx.n được set lên 1 trong khi đang là ngõ Output, nghĩa là port pin đó đang được lái lên mức cao (1). Nếu PORTx.n được xóa về 0 trong khi đang là ngõ Output, nghĩa là port pin đó đang được lái xuống mức thấp (0).

• Bit PUD (Pull up Disable) trong thanh ghi SFIOR có thể được set để cấm chức năng điện trở kéo lên trong tất cả các port.

1.6.2. Mô tả thanh ghi trong I/O Port

Special Function I/O Register – SFIOR

Bit	7	6	5	4	3	2	1	0	
	ADTS2	ADTS1	ADTS0	–	ACME	PUD	PSR2	PSR10	SFIOR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• Bit 2 – PUD: Pull-up disable

Nếu bit này được đặt lên 1 thì chức năng điện trở kéo trên tất cả các port sẽ bị cấm, ngay cả khi giá trị DDRx.n và PORTx.n được thiết lập để cho phép điện trở kéo lên ({DDx.n, PORTx.n} = 0b01).

Port A Data Register – PORTA

(* Lấy hình minh họa bên dưới làm mẫu, các thanh ghi khác có cùng chức năng nhưng thuộc về port khác cũng có nội dung tương tự)

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	

Port A Data Direction Register – DDRA

(* Lấy hình minh họa bên dưới làm mẫu, các thanh ghi khác có cùng chức năng nhưng thuộc về port khác cũng có nội dung tương tự)

Bit	7	6	5	4	3	2	1	0	
	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	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	

Port A Input Pins Address – PINA

(* Lấy hình minh họa bên dưới làm mẫu, các thanh ghi khác có cùng chức năng nhưng thuộc về port khác cũng có nội dung tương tự)

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	

Port B Data Register – PORTB

Port B Data Direction Register – DDRB

Port A Input Pins Address – PINB

Port B Data Register – PORTC

Port B Data Direction Register – DDRC

Port A Input Pins Address – PINC

Port B Data Register – PORTD

Port B Data Direction Register – DDRD

Port A Input Pins Address – PIND

1.7. Ngắt ngoài - External Interrupts

Bao gồm INT0, INT1 và INT2.

1.7.1. Thiết lập kiểu ngắt – Interrupt Sense Control

• Với INT1 và INT0: thiết lập kiểu tác động ngắt (Interrupt Sense Control) thông qua thanh ghi MCU Control Register - MCUCR.

Bit	7	6	5	4	3	2	1	0	
	SE	SM2	SM1	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
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	

Table 34. Interrupt 1 Sense Control

ISC11	ISC10	Description
0	0	The low level of INT1 generates an interrupt request.
0	1	Any logical change on INT1 generates an interrupt request.
1	0	The falling edge of INT1 generates an interrupt request.
1	1	The rising edge of INT1 generates an interrupt request.

Table 35. Interrupt 0 Sense Control

ISC01	ISC00	Description
0	0	The low level of INT0 generates an interrupt request.
0	1	Any logical change on INT0 generates an interrupt request.
1	0	The falling edge of INT0 generates an interrupt request.
1	1	The rising edge of INT0 generates an interrupt request.

• Với INT2: thiết lập kiểu tác động ngắt (Interrupt Sense Control) thông qua thanh ghi MCU Control and Status Register - MCUSCR: ISC2= 0: Falling,

ISC2 = 1: Rising.

Bit	7	6	5	4	3	2	1	0	
	JTD	ISC2	–	JTRF	WDRF	BORF	EXTRF	PORF	MCUCSR
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	See Bit Description					

1.7.2. Cho phép ngắt – Interrupt Request Enable

Cả 3 ngắt đều được cho phép ngắt tác động đến ngắt toàn cục thông qua thanh ghi General Interrupt Control Register – GICR.

Bit	7	6	5	4	3	2	1	0	
	INT1	INT0	INT2	–	–	–	IVSEL	IVCE	GICR
Read/Write	R/W	R/W	R/W	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit[7..5] – INT1, INT0, INT2: External Interrupt Request 1/0/2 Enable.

1.7.3. Cờ ngắt – Interrupt Flag

Thông qua thanh ghi General Interrupt Flag Register – GIFR.

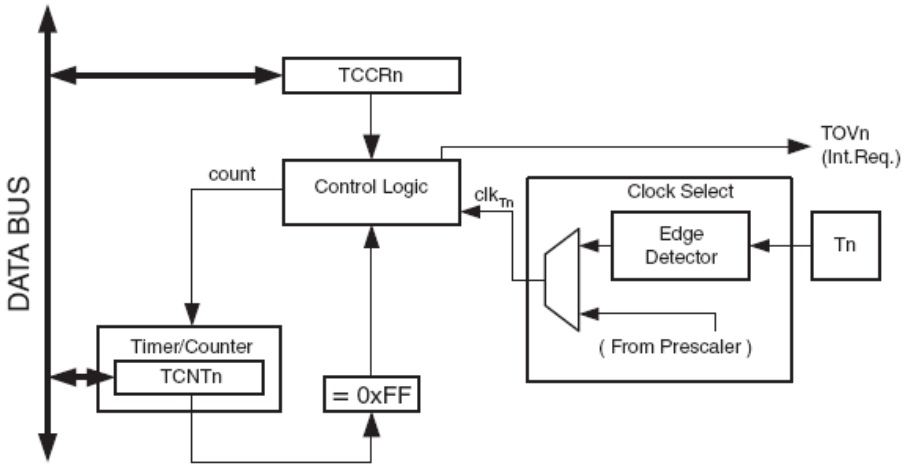
Bit	7	6	5	4	3	2	1	0	
	INTF1	INTF0	INTF2	–	–	–	–	–	GIFR
Read/Write	R/W	R/W	R/W	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

Bit[7..5] – INTF1/0/2 – External Interrupt Flag 1/0/2.

1.8. Bộ định thời 8bit timer/counter 0

Bộ định thời (timer/counter0) là một module định thời/đếm 8 bit, có các đặc điểm sau:

Sơ đồ cấu trúc của bộ định thời như hình 1.7.



Hình 1.7. Sơ đồ cấu trúc bộ định thời

- Bộ đếm một kênh.
- Xóa bộ định thời khi trong mode so sánh (tự động nạp).
- PWM.
- Tạo tần số.
- Bộ đếm sự kiện ngoài.
- Bộ chia tần 10 bit.
- Nguồn ngắt tràn bộ đếm và so sánh.

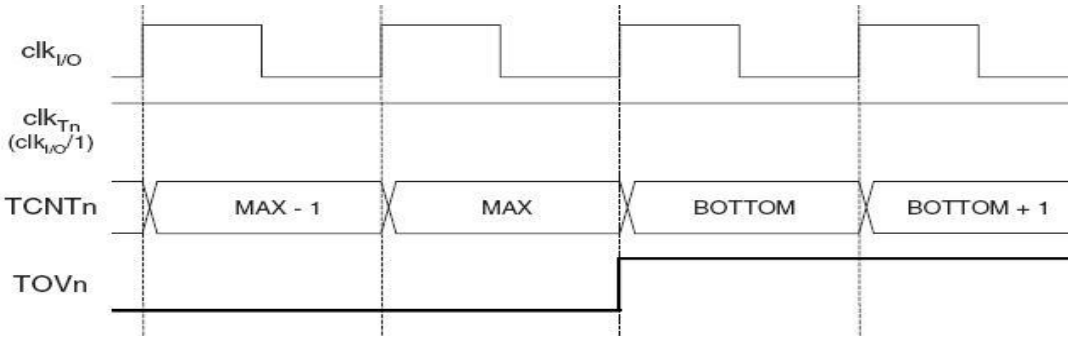
AVR Atmega8 có tích hợp bộ timer/counter. Ta bắt đầu phần này bằng sơ đồ khối sau:

1.8.1. Hoạt động của bộ Timer/Couter

Mạch đếm lên làm thanh ghi TCNTn tăng 1 đơn vị mỗi khi có xung clkTn, khi đạt giá trị lớn nhất (8bit=255), cờ TOVn được set (logic 1) và bộ đếm tràn, giá trị bộ đến TCNTn trở về 00 và tiếp tục đếm.

Xung clkTn có thể được lựa chọn từ nhiều nguồn khác nhau. Khi chọn xung nội (system clock), Timer/Counter là một Timer. Khi chọn xung ngoài (thông qua chân Tn) Timer/Counter là Counter.

Hoạt động này có thể diễn tả bằng giản đồ xung trong hình 1.8.



Hình 1.8. Giản đồ xung hoạt động

Cũng giống như bộ timer/counter trong các vi điều khiển khác, chúng ta quan tâm đến 2 thanh ghi: Timer/Counter Control và Timer/Counter Value. Trong AVR, đó là thanh ghi TCCRn và TCNTn.

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	CS02	CS01	CS00	TCCR0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/(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 T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

Hình 1.9. Thanh ghi TCCRn

Clock Select Bit Description

Bit	7	6	5	4	3	2	1	0	
	TCNT0[7:0]								TCNT0
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	

Hình 1.10. Thanh ghi TCNTn

TCNT0 - Timer/C

TCNT0 và OCR0 là các thanh ghi 8 bit. Các tín hiệu yêu cầu ngắt đều nằm trong thanh ghi TIFR. Các ngắt có thể được che bởi thanh ghi TIMSK.

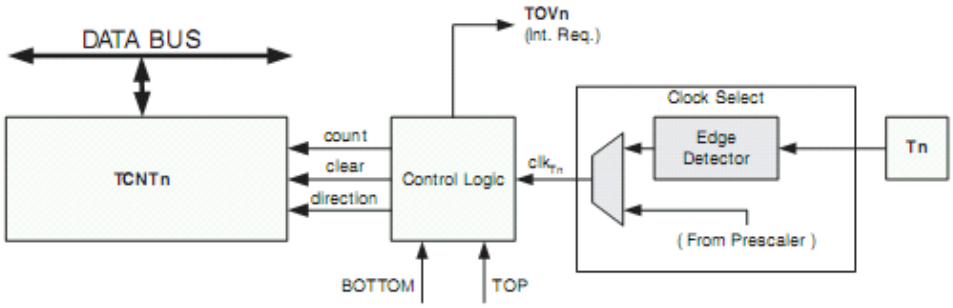
Bộ định thời có thể sử dụng xung clock nội thông qua bộ chia hoặc xung clock ngoài trên chân T0. Khối chọn xung clock điều khiển việc bộ định thời/bộ

đếm sẽ dùng nguồn xung nào để tăng giá trị của nó. Ngõ ra của khối chọn xung clock được xem là xung clock của bộ định thời (clkT0).

Thanh ghi OCR0 luôn được so sánh với giá trị của bộ định thời/bộ đếm. Kết quả so sánh có thể được sử dụng để tạo ra PWM hoặc biến đổi tần số ngõ ra tại chân OC0.

1.8.2. Đơn vị đếm

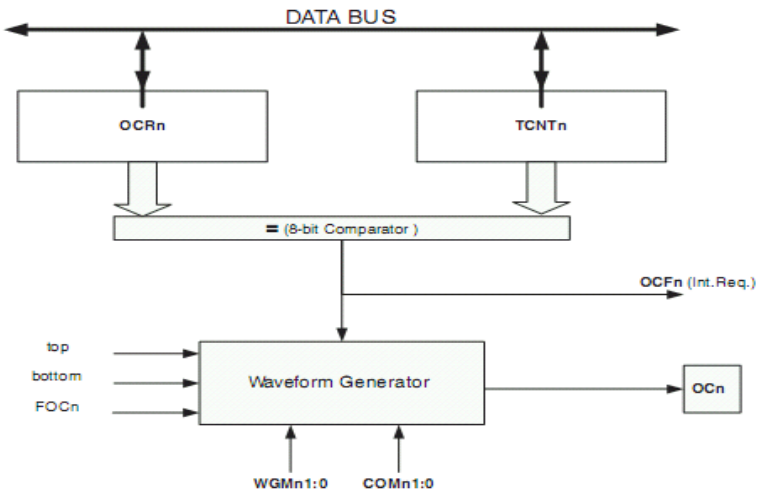
Phần chính của bộ định thời 8 bit là một đơn vị đếm song hướng có thể lập trình được. Cấu trúc của nó như hình 1.11.



Hình 1.11. Đơn vị đếm

- count: tăng hay giảm TCNT0 1
- direction: lựa chọn giữa đếm lên và đếm xuống
- clear: xóa thanh ghi TCNT0
- clkT0: xung clock của bộ định thời
- TOP: báo hiệu bộ định thời đã tăng đến giá trị lớn nhất
- BOTTOM: báo hiệu bộ định thời đã giảm đến giá trị nhỏ nhất (0)

1.8.3. Đơn vị so sánh ngõ ra



Hình 1.12. Sơ đồ đơn vị so sánh ngõ ra

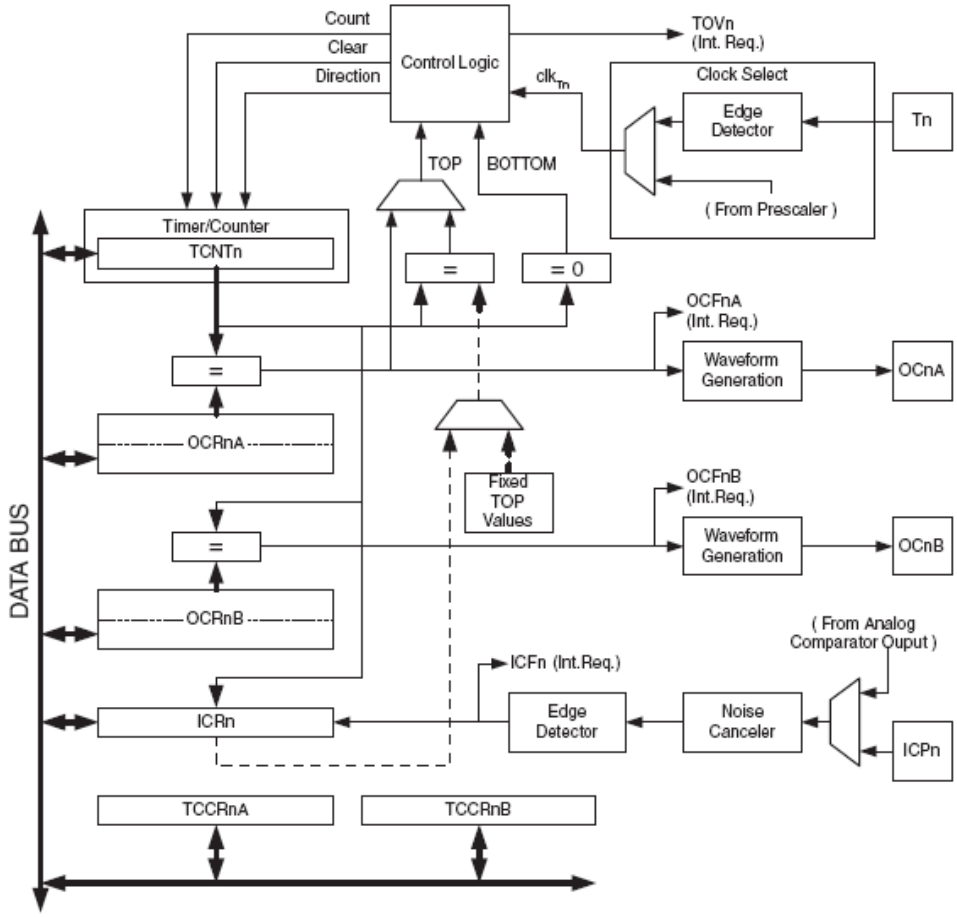
Bộ so sánh 8 bit liên tục so sánh giá trị TCNT0 với giá trị trong thanh ghi so sánh ngõ ra (OCR0). Khi giá trị TCNT0 bằng với OCR0, bộ so sánh sẽ tạo một báo hiệu. Báo hiệu này sẽ đặt giá trị cờ so sánh ngõ ra (OCF0) lên 1 vào chu kỳ xung clock tiếp theo. Nếu được kích hoạt (OCIE0=1), cờ OCF0 sẽ tạo ra một ngắt so sánh ngõ ra và sẽ tự động được xóa khi ngắt được thực thi. Cờ OCF0 cũng có thể được xóa bằng phần mềm.

1.9. Bộ định thời/đếm timer/counter 1 16-bit

1.9.1. Sơ đồ khối và một số đặc điểm

Bộ định thời (timer/counter1) là một module định thời/đếm 16 bit, có các đặc điểm sau:

- True 16-bit Design (i.e., allows 16-bit PWM)
- 2 đơn vị ngõ vào so sánh độc lập(Two Independent Output Compare Units)
- đôi thanh ghi so sánh ngõ ra đệm(Double Buffered Output Compare Registers)
- 1 đơn vị chốt ngõ vào(One Input Capture Unit)
- Bộ chống nhiễu lối vào(Input Capture Noise Canceler)
- Xóa timer trong Compare Match (Clear Timer on Compare Match (Auto Reload))
- Chống nhiễu sọc ngang(Glitch-free, Phase Correct Pulse Width Modulator (PWM)
- Giá trị chu kỳ PWM
- Bộ phát tần số chung
- Bộ đếm sự kiện ngoài
- 4 nguồn ngắt độc lập (TOV1, OCF1A, OCF1B, and ICF1)



Hình 1.13. Sơ đồ khối bộ định thời

1.9.2. Một số định nghĩa

BOTTOM Bộ đếm đạt tới BOTTOM khi co giá trị 0x0000

MAX Bộ đếm đạt tới MAXimum khi khi đạt giá trị 0xFFFF (hệ 10 là 65535).

TOP Bộ đếm đạt tới TOP khi nó bằng với giá trị lớn nhất của chuỗi đếm. Giá trị này có thể được gán bởi các giá trị cố định : 0x00FF, 0x01FF, or 0x03FF,hoặc giá trị trong bộ nhớ của các thanh ghi OCR1A ,ICR1 .

1.10. SPI (SERIAL PERIPHERAL INTERFACE)

1.10.1. Sơ đồ và định nghĩa

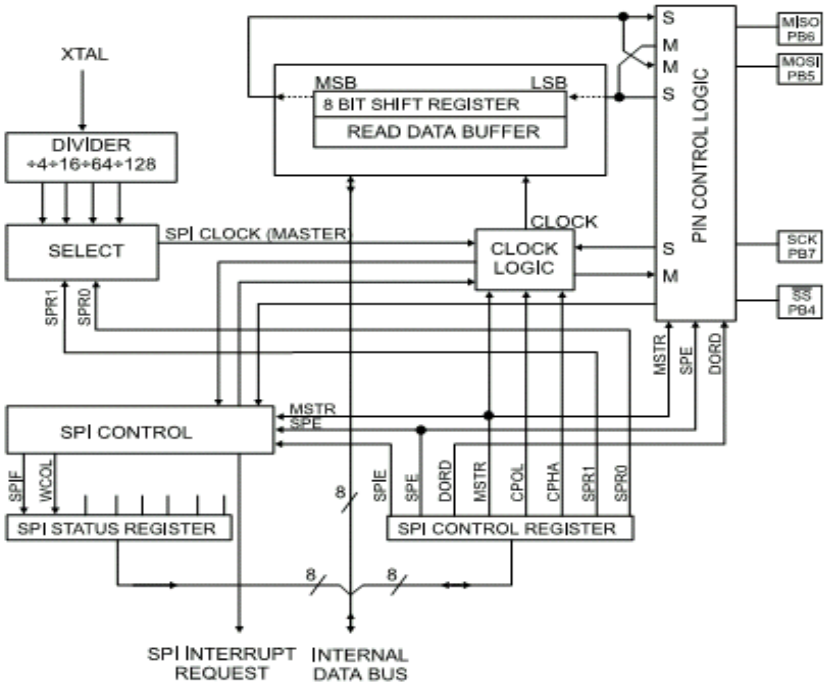
SPI là một giao diện thực hiện việc trao đổi dữ liệu giữa các thiết bị tương thích với khung dữ liệu 8bit và được truyền đồng bộ (cùng xung nhịp đồng hồ).

SPI cho phép truyền dữ liệu nối tiếp đồng bộ giữa thiết bị ngoại vi và vi

điều khiển AVR hoặc giữa các vi điều khiển AVR. SPI có các đặc điểm đặc biệt sau:

- Chế độ song công, truyền dữ liệu đồng bộ 3 dây.
- Có thể giữ vai trò Master hoặc Slave.
- Bit MSB hoặc LSB có thể được truyền trước tùy vào người lập trình.
- Bốn tốc độ truyền có thể lập trình thông qua hai bit.
- Cờ ngắt báo kết thúc truyền.
- Vận hành từ trạng thái ngủ (Được đánh thức từ trạng thái ngủ).

Sơ đồ cấu trúc hình 1.14.

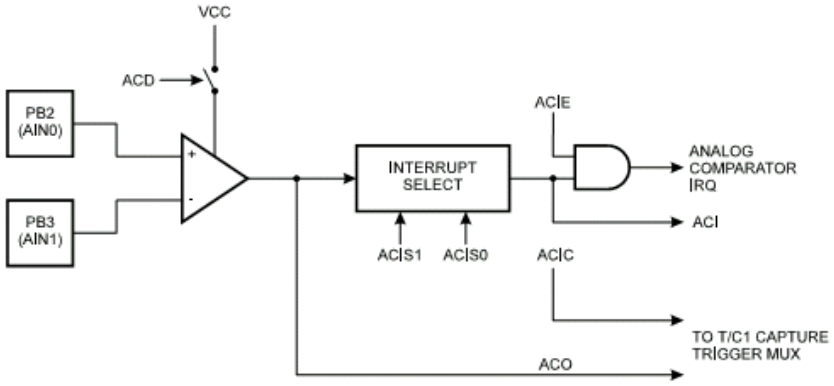


Hình 1.14. Sơ đồ cấu trúc SPI

Để điều khiển khối giao tiếp SPI thì chúng ta có 3 thanh ghi. Đó là 1 thanh ghi điều khiển SPCR (SPI control Register), thanh ghi trang thái SPSR (SPI status Register) và cuối cùng là thanh ghi dữ liệu SPDR (SPI Data Register).

1.11. Bộ so sánh tương tự (alalog comparator)

Bộ so sánh tương tự của AVR có đầu vào là hai chân PB2 và PB3 (như hình 1.15.). Với chân PB2 được nối vào cực dương của bộ so sánh và PB3 được nối vào cực âm của bộ so sánh. Nó tạo ra hai mức logic nếu V+ > V- thì tín hiệu ra là 1 và ngược lại là 0.



Hình 1.15. Sơ đồ khối bộ so sang tương tự

Để điều khiển và qua sát trạng thái của bộ so sánh tương tự ta có một thanh ghi đó là thanh ghi ACSR. Trước khi tìm hiểu về nguyên tắc hoạt động của nó ta sẽ giới thiệu về thanh ghi này.

Thanh ghi ACSR là một thanh ghi 8 bit có địa chỉ trong các thanh ghi I/O là 0x08 và có địa chỉ trong không gian bộ nhớ SRAM là 0x28.Trong 8 bit thì có 7 bit được định nghĩa và bit 6 không được định nghĩa.Nó chỉ có thể đọc và luôn có giá trị logic là 0.

Bit 7-ACD: Analog comparator disable –Đây là bit điều khiển. Bit này ttrực tiếp điều khiển hoạt động của AC(bộ so sánh tương tự). Nếu như bit này được set lên 1 thì nguồn cung cấp cho AC hoạt động bị tắt (turn off) và đồng nghĩa với việc nó không hoạt động. Và nếu nó được xóa thì AC được cấp nguồn và hoạt động bình thường. Chú ý: Ta có thể thay đổi giá trị logic của bit này lúc nào cũng được để ngưng hoạt động của chúng hoặc cho chúng hoạt đông trở lại nhưng khi thay đổi giá trị logic của nó thì ngắt (ngắt của AC) cần bị cấm nếu không nó sẽ sinh ra một ngắt (Cụ thể là bit ACIE cần bị xóa).

Bit 5-ACO: Analog comparator output – Đây là bit trạng thái. Bit này được nối trức tiếp với đầu ra của bộ so sánh tương tự.

Bit 4-ACI: Analog comparator interrupt flag – Đây là bit trạng thái. Cờ báo ngắt của bộ so sanh tương tự.

Nếu như cờ này được đặt và các ngắt được phép thì một chương trình phục vụ ngắt được gọi và chúng đước xóa bằng phần cứng khi chương trinh báo ngắt được phục vụ. Các trường hợp làm thay đổi trạng thái cờ này ngoài việc thay đổi bit ACD sẽ được nói tới trong các bít 0 và 1.

Bit 3-ACIE: AC interrupt enable – Đây là bit điều khiển. Nếu bit này được đặt thì ngắt này được phép và ngược lại.

Bit 2-ACIC: Analog comparator input Capture Enable – Đây là bit điều khiển. Khi bit này được đặt lên 1 thì đầu ra của AC được nối trực tiếp vào đầu vào của chức năng bắt sự kiện của Timer/counter 1. (Đọc thêm timer/counter1). Bit ACIS1 và ACIS0: Ac interrupt mode select – Đây là hai bit điều khiển.

Bảng 1.1. Chế độ ngắt 2 bit ACIS1 và ACIS2

ACIS1	ACIS0	Chế độ ngắt
0	0	Theo mức
0	1	Dành riêng(chưa dùng đến)
1	0	Sườn xuống
1	1	Sườn lên

Chú ý: Các bit này cũng có thể được thay đổi bất cứ khi nào. Nhưng khi thay đổi thì ngắt của nó phải bị cấm.

Ta có thể sử dụng lệnh SBI hoặc CBIU để thay đổi trạng thái các bit trên thanh ghi này trừ bit ACI. Bit này sau khi được đọc cũng sẽ bị xóa (nếu nó được đặt).

Thiết lập port đầu vào cho bộ so sánh tương tự: Hai chân PB2 và PB3 này cần được thiết lập là đầu vào bỏ điện trở treo.

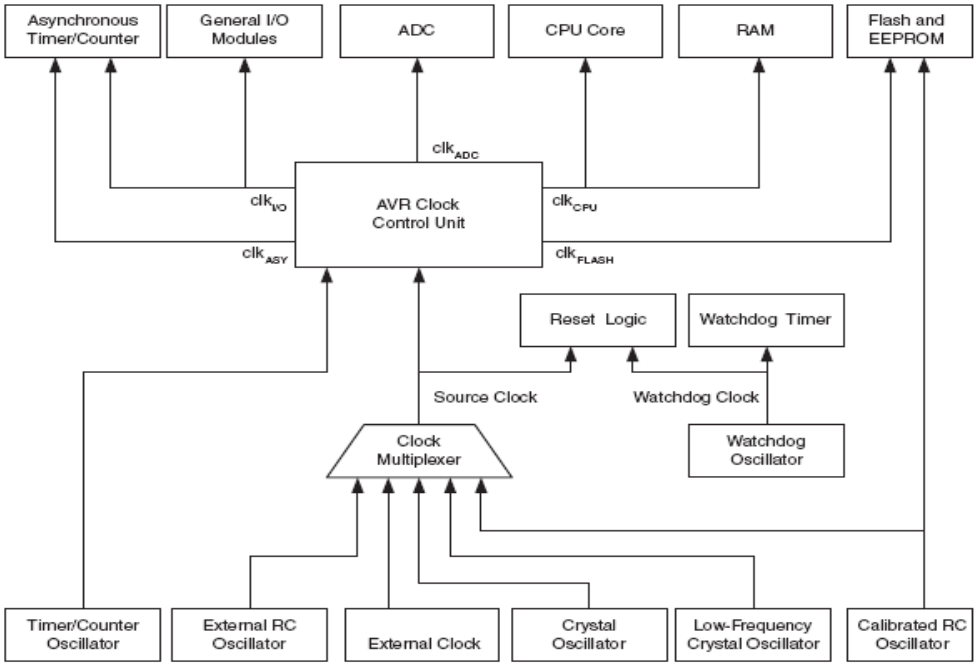
Để lập trình cho AC ta bắt đầu các bước sau:

- Bước 1: Thiết lập các chân đầu vào cho AC.
- Bước 2: Chọn các chế độ cho AC ví như dùng ngắt …
- Bước 3: Khởi động AC bằng cách xóa bit AC

1.12. Hệ thốnng xung clock

Để cấu hình cho chip hoạt động theo chế độ xung clock nào, người ta dùng các bit cầu chì (fuse bit) CKSEL 3, CKSEL2, CKSEL 1.

Ngoài ra khi vi điều khiển được đánh thức từ các chế độ nghỉ sang chế độ hoạt động bình thường, bộ tạo dao động cần có một khoảng thời gian để ổn định, khoảng thời gian này gọi là thời gian khởi động (start-up time). CPU chỉ thực hiện lệnh khi hết khoảng thời gian khởi động này.



Hình 1.16. Hệ thống xung clock

Khi ta reset CPU cũng cần một khoảng thời gian trì hoãn (delay time) để nguồn nuôi đạt mức ổn định trước khi thực bắt đầu thực thi lệnh. Người ta dùng các bit cầu chì CKSEL 0, SUT1, SUT0 để thiết lập thời gian khởi động và thời gian trì hoãn. Khoảng thời gian khởi động và thời gian trì hoãn được đo được đo

bằng một đồng hồ riêng, đó là bộ dao động Watchdog. Tần số của bộ dao động Watchdog phụ thuộc vào điện thế nguồn nuôi và nhiệt độ môi trường. Ở Vcc = 5V và nhiệt độ 25^0C thì tần số của bộ dao động Watchdog là 1 MHz.

Liên quan đến việc thiết lập của hệ thống xung clock người ta còn dùng tới bit cầu chì CKOPT mà vai trò của nó khá linh hoạt tùy theo việc thiết lập xung clock cho hệ thống như thế nào. Hình 1.16. cho thấy ATmega128 có tới 7 bộ tạo xung clock có thể được lựa chọn.

Dưới đây là mô tả cụ thể cho từng trường hợp cấu hình xung clock của hệ thống.

1.13. Bộ biến đổi A/D

Vi điều khiểnAtmega8 có một bộ biến đổi ADC tích hợp trong chip với các đặc điểm:

- Độ phân giải 10 bit
- Sai số tuyến tính: 0.5LSB
- Độ chính xác +/-2LSB
- Thời gian chuyển đổi:65-260µs
- 6 Kênh đầu vào có thể được lựa chọn
- Có hai chế độ chuyển đổi free running và single conversion
- Có nguồn báo ngắt khi hoàn thành chuyển đổi
- Loại bỏ nhiễu trong chế độ ngủ

Tám đầu vào của ADC là tám chân của PORTA và chúng được chọn thông qua một MUX.

Để điều khiển hoạt động vào ra dữ liệu của ADC và CPU chúng ta có 3 thanh ghi: ADMUX là thanh ghi điều khiển lựa chọn kênh đầu vào cho ADC, ADCSRA là thanh ghi điều khiển và thanh ghi trạng thái của ADC, ADCH và ADCL là 2 thanh ghi dữ liệu.

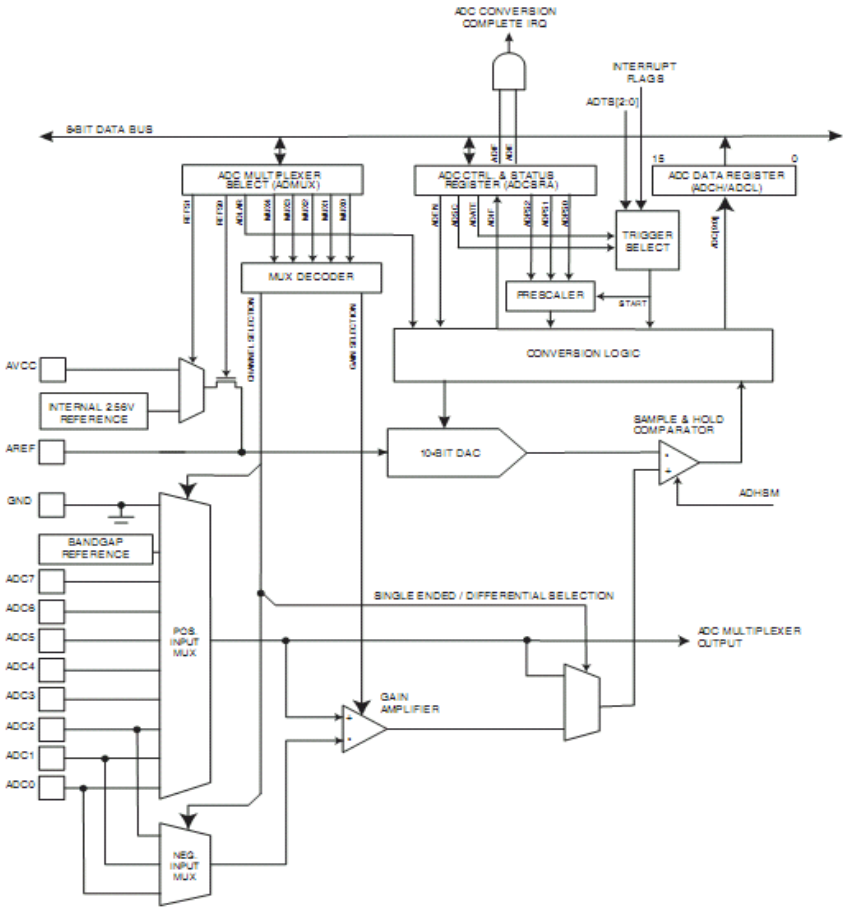
Nguyên tắc hoạt động và lập trình điều khiển

ADC có nhiệm vụ chuyển đổi tín hiệu điện áp tương tự thành tín hiệu số có độ phân giải 10 bit.Với giá trị nhỏ nhất của điện áp đặt ở chân AGND và giá trị cực đại của điện áp tương tự được mắc vào chân AREF. Tám kênh tương tự đầu vào được chọn lựa thông qua ADMUX và ADMUX này được điều khiển bởi thanh ghi ADMUX.

ADC này có thể hoạt động được ở hai chế độ. Đó là chuyển đổi đơn: chỉ chuyển đổi một lần khi có lệnh chuyển đổi và chế độ tự chuyển đổi (Free running mode) đây là chế độ mà ADC tự động chuyển đổi khi được hoạt động và công việc chuyển đổi có tính tuần hoàn (chỉ cần khởi động một lần).

ADC được phép hoạt động nhờ thiết lập bit ADEN. Quá trình chuyển đổi được bắt đầu bằng việc ghi vào bit ADSC mức logic 1 và trong suốt quá trình chuyển đổi bit này luôn được giữ ở mức cao. Khi quá trình chuyển đổi hoàn thành thì bit này được xóa bằng phần cứng và cờ AIDF được bật lên.

Dữ liệu sau khi chuyển đổi được đưa ra thanh ghi dữ liệu ADCL và ADCH, nhưng chú ý khi đọc dữ liệu từ hai thanh ghi này thì đọc ADCL trước rồi mới đọc ADCH. Nếu đọc ADCH trước thì dữ liệu cập nhật có thể ghi đè lên ADCL (Vi điều khiển nghĩ rằng đã đọc xong dữ liệu).



Hình 1.17. Sơ đồ bộ biến đổi A/D

Để điều khiển vào ra dữ liệu với ADC, các bước thực hiện như sau:

- Bước 1: Định nghĩa các cổng vào cho tín hiệu tương tự.Xóa bit tương ứng với chân đó trong thanh ghi DDRA. Sau đó loại bỏ điện trở treo bằng cách xóa bit tương ứng ở thanh ghi PORTA.
- Bước 2: Chọn kênh tương tự vào (chọn chân vào cho ADC) thông qua thanh ghi ADMUX (có thể thay đổi trong quá trình hoạt động).
- Bước 3: Thiết lập các thông số cho ADC.
- Tốc độ chuyển đổi thông qua xung nhip chuyển đổi.
- Chế độ chuyển đổi : đơn hoặc tự động.
- Sử dụng ngắt hoặc không.

Bước 4: Bắt đầu chuyển đổi và đọc dữ liệu.

CHƯƠNG 2.

GIAO TIẾP QUA CỔNG COM

2.1. Giới thiệu

Ngày nay, việc sử dụng máy vi tính làm bộ xử lý trung tâm cho các hệ thống đo lường và điều khiển rất phổ biến. Trong hệ thống đó một yêu cầu đặt ra là ta phải thực hiện giao tiếp giữa thiết bị ngoại vi với máy tính. Chuẩn giao tiếp được coi là đơn giản và dễ dùng đó là RS232. Hầu như các thiết bị đều được giao tiếp với máy tính thông qua chuẩn này. Ở đây em thực hiện giao tiếp giữa máy tính và vi điều khiển theo chuẩn RS232. Vấn đề giao tiếp giữa PC và vi điều khiển rất quan trọng trong các ứng dụng điều khiển, đo lường... Ghép nối qua cổng nối tiếp RS232 là một trong những kỹ thuật được sử dụng rộng rãi để ghép nối các thiết bị ngoại vi với máy tính. Nó là một chuẩn giao tiếp nối tiếp dùng định dạng không đồng bộ, kết nối nhiều nhất là 2 thiết bị, chiều dài kết nối lớn nhất cho phép để đảm bảo dữ liệu là 12.5m đến 25.4m, tốc độ 20kbit/s đôi khi là tốc độ 115kbit/s với một số thiết bị đặc biệt. Ý nghĩa của chuẩn truyền thông nối tiếp nghĩa là trong một thời điểm chỉ có một bit được gửi đi dọc theo đường truyền.

Có hai phiên bản RS232 được lưu hành trong thời gian tương đối dài là RS232B và RS232C. Nhưng cho đến nay thì phiên bản RS232B cũ thì ít được dùng còn RS232C hiện vẫn được dùng và tồn tại thường được gọi là tên ngắn gọn là chuẩn RS232. Các máy tính thường có 1 hoặc 2 cổng nối tiếp theo chuẩn RS232C được gọi là cổng COM. Chúng được dùng ghép nối cho chuột, modem, thiết bị đo lường... Trên main máy tính có loại 9 chân hoặc lại 25 chân tùy vào đời máy và main của máy tính. Việc thiết kế giao tiếp với cổng RS232 cũng tương đối dễ dàng, đặc biệt khi chọn chế độ hoạt động là không đồng bộ và tốc độ truyền dữ liệu thấp.

2.2. Ưu điểm của giao diện nối tiếp RS232

+ Khả năng chống nhiễu của các cổng nối tiếp cao

+ Thiết bị ngoại vi có thể tháo lắp ngay cả khi máy tính đang được cấp điện

+ Các mạch điện đơn giản có thể nhận được điện áp nguồn nuôi qua công nối tiếp

2.3. Những đặc điểm cần lưu ý trong chuẩn RS232

+ Trong chuẩn RS232 có mức giới hạn trên và dưới (logic 0 và 1) là ±12V. Hiện nay đang được cố định trở kháng tải trong phạm vi từ 3000 ôm - 7000 ôm

+ Mức logic 1 có điện áp nằm trong khoảng -3V đến -12V, mức logic 0 từ 3V đến 12V

+ Tốc độ truyền nhận dữ liệu cực đại là 100kbps (ngày nay có thể lớn hơn)

+ Các lối vào phải có điện dung nhỏ hơn 2500pF

+ Trở kháng tải phải lớn hơn 3000 ôm nhưng phải nhỏ hơn 7000 ôm

+ Độ dài của cáp nối giữa máy tính và thiết bị ngoại vi ghép nối qua cổng nối tiếp RS232 không vượt qua 15m nếu chúng ta không sử model

+ Các giá trị tốc độ truyền dữ liệu chuẩn : 50, 75, 110, 750, 300, 600, 1200, 2400, 4800, 9600, 19200, 28800, 38400... 56600, 115200 bps

2.4. Các mức điện áp đường truyền

RS 232 sử dụng phương thức truyền thông không đối xứng, tức là sử dụng tín hiệu điện áp chênh lệch giữa một dây dẫn và đất. Do đó ngay từ đầu tiên ra đời nó đã mang vẻ lỗi thời của chuẩn TTL, nó vẫn sử dụng các mức điện áp tương thích TTL để mô tả các mức logic 0 và 1. Ngoài mức điện áp tiêu chuẩn cũng cố định các giá trị trở kháng tải được đấu vào bus của bộ phận và các trở kháng ra của bộ phát. Mức điện áp của tiêu chuẩn RS232C (chuẩn thường dùng bây giờ) được mô tả như sau:

+ Mức logic 0: +3V , +12V

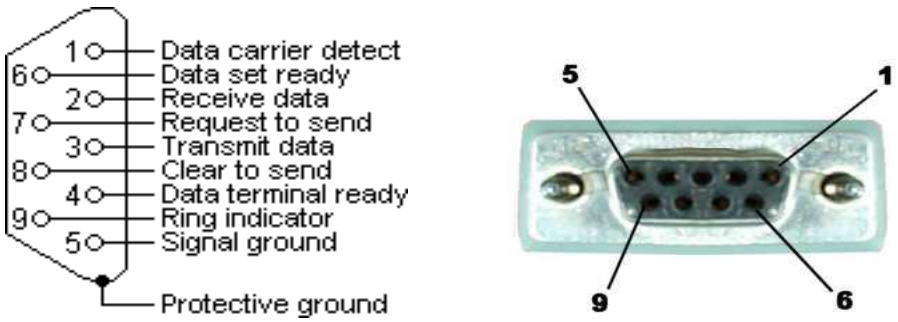
+ Mức logic 1: -12V, -3V

Các mức điện áp trong phạm vi từ -3V đến 3V là trạng thái chuyển tuyến. Chính vì từ -3V tới 3V là phạm vi không được định nghĩa, trong trường hợp thay đổi giá trị logic từ thấp lên cao hoặc từ cao xuống thấp, một tín hiệu phải vượt qua quãng quá độ trong một thơì gian ngắn hợp lý. Điều này dẫn đến việc phải hạn chế về điện dung của các thiết bị tham gia và của cả đường truyền. Tốc độ truyền dẫn tối đa phụ thuộc vào chiều dài của dây dẫn. Đa số các hệ thống hiện nay chỉ hỗ trợ với tốc độ 19,2 kBd .

2.5. Cổng RS232 trên PC

Hầu hết các máy tính cá nhân hiện nay đều được trang bị ít nhất là 1 cổng COM hay cổng nối tiếp RS232. Số lượng cổng COM có thể lên tới 4 tùy từng loại main máy tính. Khi đó các cổng COM đó được đánh dấu là COM 1, COM 2, COM 3... Trên đó có 2 loại đầu nối được sử dụng cho cổng nối tiếp RS232 loại 9 chân (DB9) hoặc 25 chân (DB25). Tuy hai loại đầu nối này có cùng song song nhưng hai loại đầu nối này được phân biệt bởi cổng đực (DB9) và cổng cái (DB25)

Ta xét sơ đồ chân cổng COM 9 chân như trong hình 2.1.



Hình 2.1. Các chân cổng COM 9 chân

Chức năng của các chân như sau:

+ Chân 1: Data Carrier Detect (DCD): Phát tín hiệu mang dữ liệu

+ Chân 2: Receive Data (RxD): Nhận dữ liệu

+ Chân 3: Transmit Data (TxD): Truyền dữ liệu

+ Chân 4: Data Termial Ready (DTR): Đầu cuối dữ liệu sẵn sàng được kích hoạt bởi bộ nhận khi muốn truyền dữ liệu

+ Chân 5: Singal Ground (SG): Mass của tín hiệu

+ Chân 6: Data Set Ready (DSR): Dữ liệu sẵn sàng, được kích hoạt bởi bộ truyền khi nó sẵn sàng nhận dữ liệu

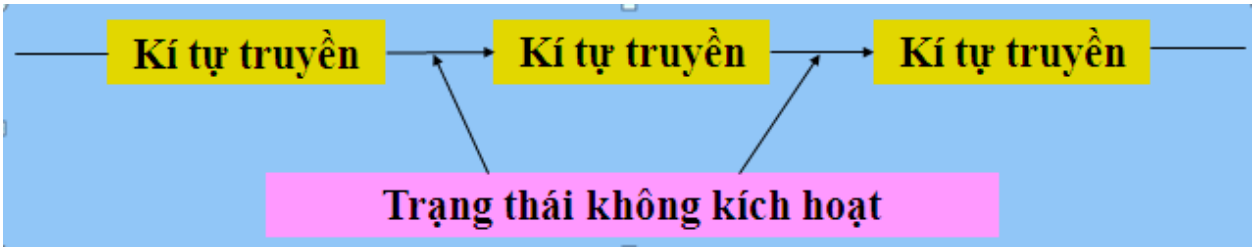
+ Chân 7: Request to Send: Yêu cầu gửi, bô truyền đặt đường này lên mức hoạt động khi sẵn sàng truyền dữ liệu.

+ Chân 8: Clear To Send (CTS): Xóa để gửi, bô nhận đặt đường này lên mức kích hoạt động để thông báo cho bộ truyền là nó sẵn sàng nhận tín hiệu

+ Chân 9: Ring Indicate (RI): Báo chuông cho biết là bộ nhận đang nhận tín hiệu rung chuông

Còn DB25, bây giờ hầu hết các main mới ra đều không có cổng này nữa nên ở đây không đề cập đến.

2.6. Quá trình dữ liệu



Hình 2.2. Khuôn mẫu khung truyền

**** Quá trình truyền dữ liệu***

Truyền dữ liệu qua cổng nối tiếp RS232 được thực hiện không đồng bộ, có khuôn mẫu như hình 2.2. Do vậy nên tại một thời điểm chỉ có một bit được truyền. Bộ truyền gửi một bit bắt đầu (bit start) để thông báo cho bộ nhận biết một kí hiệu sẽ được gửi đến trong lần truyền bit tiếp theo. Bit này luôn bắt đầu bằng mức 0. Tiếp theo đó là các bit dữ liệu (bits data) được gửi dưới dạng mã ASCII (có thể là 5, 6, 7 hay 8 bit dữ liệu) Sau đó là một Parity bit (Kiểm tra bit

chẵn, lẻ hay không) và cuối cùng là bit dừng - bit stop có thể là 1, 1.5 hay 2 bit dừng.

**** Tốc độ Baud***

Đây là một tham số đặc trưng của RS232. Tham số này chính là đặc trưng cho quá trình truyền dữ liệu qua cổng nối tiếp RS232 là tốc độ truyền nhận dữ liệu hay còn gọi là tốc độ bit. Tốc độ bit được định nghĩa là số bit truyền được trong thời gian 1 giây. Tốc độ bit này phải được thiết lập ở bên phát và bên nhận đều phải có tốc độ như nhau (Tốc độ giữa vi điều khiển và máy tính phải chung nhau 1 tốc độ truyền bit). Ngoài tốc độ bit còn một tham số để mô tả tốc độ truyền là tốc độ Baud. Tốc độ Baud liên quan đến tốc độ mà phần tử mã hóa dữ liệu được sử dụng để diễn tả bit được truyền còn tôc độ bit thì phản ánh tốc độ thực tế mà các bit được truyền. Vì một phần tử báo hiệu sự mã hóa một bit nên khi đó hai tốc độ bit và tốc độ baud là phải đồng nhất. Một số tốc độ Baud thường dùng: 50, 75, 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 28800, 38400, 56000, 115200 … Trong thiết bị thường dùng tốc độ là 19200.

Khi sử dụng chuẩn nối tiếp RS232 thì yêu cầu khi sử dụng chuẩn là thời gian chuyển mức logic không vượt quá 4% thời gian truyền 1 bit. Do vậy, nếu tốc độ bit càng cao thì thời gian truyền 1 bit càng nhỏ do vậy thời gian chuyển mức logic càng phải nhỏ. Điều này làm giới hạn tốc Baud và khoảng cách truyền.

**** Bit chẵn lẻ hay Parity bit***

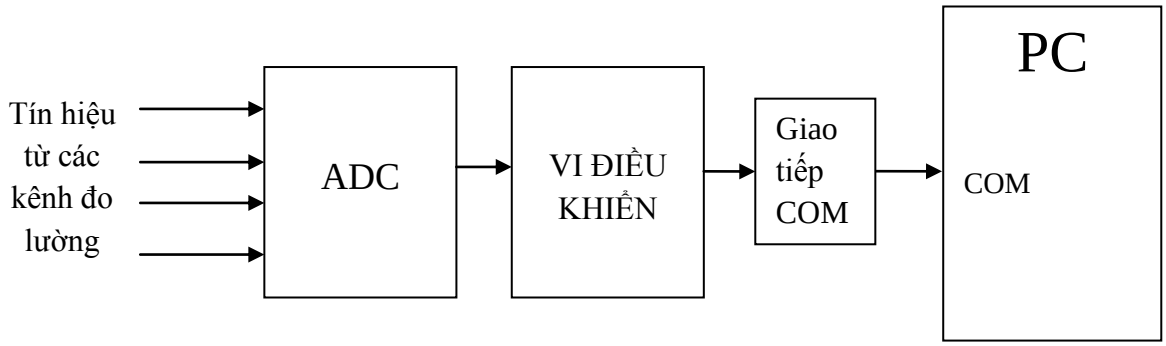
Đây là bit kiểm tra lỗi trên đường truyền. Thực chất của quá trình kiểm tra lỗi khi truyền dữ liệu là bổ xung thêm dữ liệu được truyền để tìm ra hoặc sửa một số lỗi trong quá trình truyền. Do đó trong chuẩn RS232 sử dụng một kỹ thuật kiểm tra chẵn lẻ. Một bit chẵn lẻ được bổ sung vào dữ liệu được truyền để cho số lượng các bit "1" được gửi trong một khung truyền là chẵn hay lẻ. Một bit Parity chỉ có thể tìm ra một số lẻ các lỗi, chẳng hạn như 1, 3, 5, 7, 9... Nếu như một bit chẵn được mắc lỗi thì Parity bit sẽ trùng giá trị với trường hợp không mắc lỗi vì thế không phát hiện ra lỗi. Do đó trong kỹ thuật mã hóa lỗi này không được sử dụng trong trường hợp có khả năng một vài bit bị mắc lỗi.

CHƯƠNG 3.

THIẾT KẾ HỆ THỐNG GHÉP NỐI MÁY TÍNH ĐO LƯỜNG NHIỀU KÊNH

3.1. Thiết kế hệ thống

Sơ đồ khối hệ thống như trong hình 3.1.



Hình 3.1. Sơ đồ khối hệ thống ghép nối máy tính đo lường nhiều kênh

Tín hiệu từ các kênh đo lường dưới dạng là các điện áp tương tự, được cho qua bộ ADC để chuyển thành tín hiệu số. Vi điều khiển ở đây đảm nhận nhiệm vụ nhận dữ liệu từ ADC và thực hiện giao tiếp với máy tính theo chuẩn RS232. Máy tính nhận dữ liệu về, thực hiện hiển thị và lưu trữ.

*** Khối ADC**

Thực hiện chuyển đổi các mẫu điện áp tương tự sang điện áp số. Gọi tín hiệu tương tự là U_A, thì tín hiệu số là U_D được biểu diễn dưới dạng mã nhị phân như:

$$U_D=b_{n-1}.2^{n-1}+b_{n-2}.2^{n-2}+\ldots+b_0 2^0 \qquad (3.1)$$

Trong đó các hệ số b_k là các bít của số nhị phân. Bít b_{n-1} được gọi là bit có ý nghĩa lớn nhất (MSB), mỗi biến đổi giá trị của MSB tương ứng với sự biến đổi của tín hiệu là nửa dải làm việc. Bit b_0 gọi là bit có ý nghĩa nhỏ nhất (LSB), mỗi biến đổi giá trị của LSB tương ứng với một mức lương tử .

Với một mạch biến đổi có N bit đầu ra thì mỗi bước của bậc thang tương ứng một giá trị:

$$Q = U_{LSB} = \frac{U_{Am}}{2^N - 1} \qquad (3.2)$$

Trong đó, U_{AM} là giá trị cực đại cho phép cho phép của điện áp đầu vào ADC. Q hoặc U_{LSB} gọi là mức lượng tử. Do tín hiệu số là rời rạc nên trong quá trình biến đổi AD xuất hiện một sai số gọi là sai số lượng tử hóa được xác định như sau:

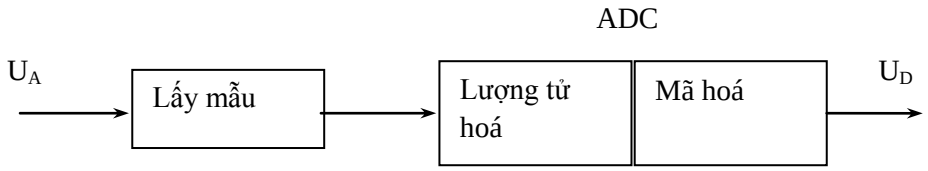
$$\Delta Q = \frac{1}{2} Q \qquad (3.3)$$

Khi chuyển đổi AD phải thực hiện lấy mẫu tín hiệu tương tự. Để đảm bảo khôi phục lại tín hiệu một cách trung thhực thì tần số lấy mẫu f_M phải thỏa mãn điều kiện:

$$f_M \geq 2.f_{Amax} \qquad (3.4)$$

Trong đó, f_{Amax} là tần số cực đại của tín hiệu đầu vào

Quá trình biến đổi A/D gồm 3 bước: lấy mẫu, lượng tử hóa và mã hóa.



Hình 3.2 Sơ đồ khối quá trình chuyển đổi A/D

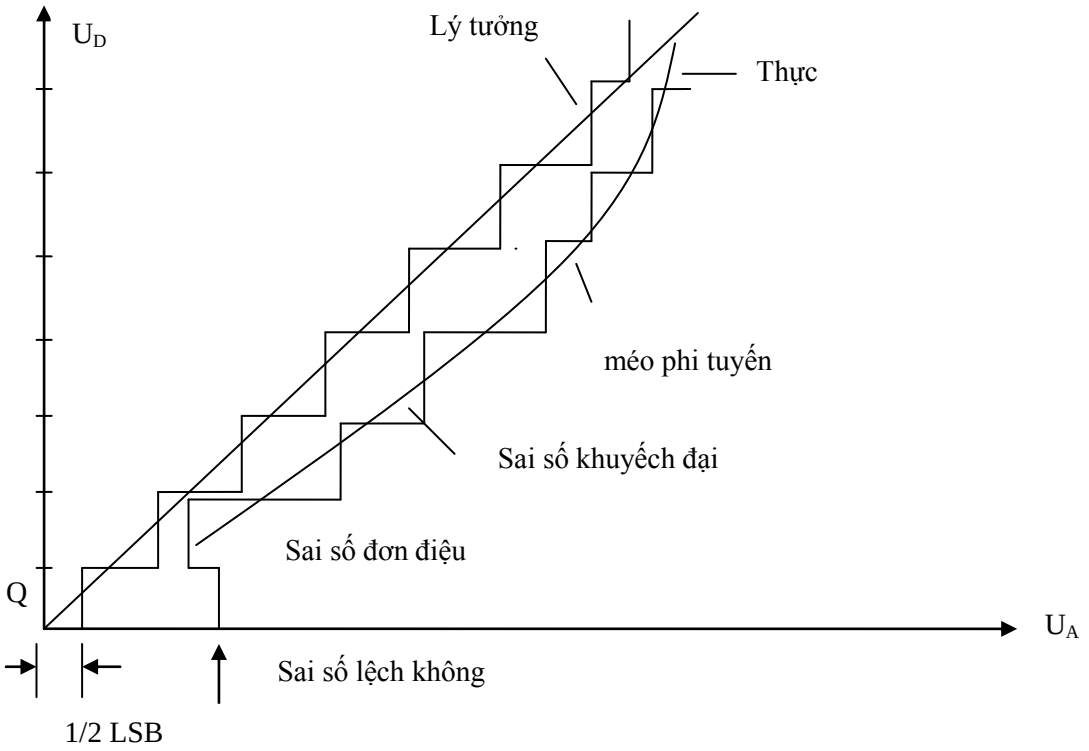
- Lấy mẫu: Mạch lấy mẫu có hai nhiệm vụ:

+ Lấy mẫu tín hiệu tương tự tại các thời điểm khác nhau và cách đều nhau (rời rạc hóa về mặt thời gian).

+ Giữ cho biên độ điện áp tại các thời điểm lấy mẫu không đổi trong suốt quá trình chuyển đổi tiếp theo.

- Lượng tử hoá: là quá trình rời rạc các mẫu về biên độ. Chia khoảng biên độ thành các mức rời rạc gọi là các mức lượng tử, biên độ của các mẫu được làm tròn về các mức lượng tử đó.

- Mã hoá: Mã hoá các mẫu sau khi được lượng tử hoá thành các bits số.



Hình 3.3. Đặc tuyến truyền đạt lý tưởng và thực của bộ chuyển đổi A/D

Tổng quát ta có công thức chuyển đổi A/D đối với mỗi mẫu tín hiệu tương tự:

$$U_{Di} = \text{Round}\left[\frac{U_{Ai}}{U_{ref}}.2^N\right] \qquad (3.5)$$

U_{Ai}: Điện áp tương tự của mẫu thứ i

U_{ref}: Điện áp tham chiếu (điện áp chuẩn cố định), dùng để so sánh với U_{Ai} tạo điện áp số.

U_{Di}: Điện áp số ứng với mẫu U_{Ai}

N: Số bits của bộ chuyển đổi

ở đây yêu cầu $U_A \leq U_{ref}$, nên ta phải lựa chọn U_{ref} thích hợp với mỗi tín hiệu U_A.

Với mỗi giá trị N, thì U_{ref} càng lớn thì sai số lượng tử càng lớn. Với mỗi giá trị U_{ref} thì N càng lớn thì sai số lượng tử càng nhỏ.

Các tham số chính của bộ chuyển đổi A/D

- Dải biến đổi của điện áp tín hiệu tương tự ở đầu vào: là khoảng điện áp mà bộ chuyển đổi A/D có thể thực hiện chuyển đổi được.Khoảng điện áp đó có thể lấytrị số từ 0 đến một trị số dương hay âm nào đó hoặc có thể là điện áp hai cực tính

- Độ chính xác của bộ chuyển đổi A/D: Tham số đầu tiên đặc trưng cho độ chính xác của một ADC là độ phân biệt.Trên đầu ra mỗi bộ ADC là các giá trị số được sắp xếp theo quy luật của một loại mã nào đó. Số các số hạng của mã số ở đầu ra (số bits trong mã nhị phân) tương ứng với dải biến đổi của điện áp vào cho biết mức chính xác của phép chuyển đổi. Một ADC có N bit đầu ra thì nó có thể phân biệt được 2^N mức trong dải biến đổi của nó. Độ phân biệt của một ADC là Q, nó chính là giá trị của một mức lượng tử hóa hoặc còn gọi là 1 LSB. Trong thực tế thường dùng số bit N ở đầu ra để đặc trưng cho độ chính xác với cùng một dải điện áp vào số các số hạng của mã số ở đầu ra càng lớn thì độ chính xác càng cao.

Ngoài ra đặc trưng cho tính chính xác của ADC còn có các tham số khác, đó là :

+ Đường đặc tuyến có sai số lệch không, nghĩa là nó không xuất phát tại giá trị tương ứng là 1/2 LSB. Nó là hình bậc thang không đều do ảnh hưởng của các sai số.

+ Sai số khuyếch đại là sai số giữa độ dốc trung bình của đường đặc tuyến thực với độ dốc trung bình của đường đặc tuyến lý tưởng.

+ Sai số phi tuyến được đặc trưng bởi sự thay đổi độ dốc đường trung bình của đặc tuyến thực trong dảI biến đổi của của điện áp vào. Sai số này làm cho đặc tuyến chuyển đổi có dạng hình bậc thang không đều.

+ Sai số đơn điệu thực chất cũng do tính phi tuyến của đường đặc tính biến đổi gây ra, nhưng nó làm cho độ dốc đường trung bình biến thiên không đơn điệu, thậm chí mất một vài mã mã số.

- Tốc độ chuyển đổi: Tốc độ chuyển đổi cho biết số mẫu chuyển đổi trong 1 giây, được gọi là tần số chuyển đổi f_c. Cũng có thể dùng tham số thời gian chuyển đổi T_c để đặc trưng cho tốc độ chuyển đổi. Vì giữa các lần chuyển đổi còn có một khoảng thời gian cần thiết để cho ADC phục hồi lại trạng thái ban đầu, nên thường $f_c<1/T_c$, ở đây với một bộ ADC tốc độ cao thì phải trả giá bằng độ chính xác giảm hoặc ngược lại.

Các phương pháp biến đổi số tương tự

Có nhiều cách phân loại các phương pháp biến đổi số tương tự. Trong đó có cách phân loại theo quá trìnhchuyển đổi về mặt thời gian, theo cách phân loại này có bốn phương pháp biến đổi A/D:

- Biến đổi song song: trong phương pháp này, tín hiệu được so sánh cùng một lúc với nhiều giá trị chuẩn. Do đó, tất cả các bít được xác định đồng thời và đưa đến đầu ra.

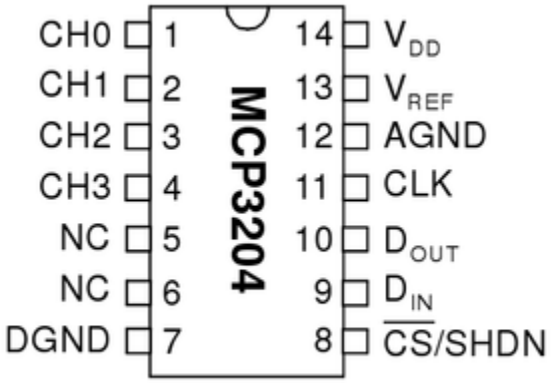
- Biến đổi nối tiếp theo mã đếm: quá trình so sánh được thực hiện lần lượt từng bước theo quy luật của mã đếm. Kết quả chuyển đổi được xác định bằng cách đếm số lượng giá trị chuẩn có thể chứa được trong giá trị tín hiệu tương tự cần chuyển đổi.

- Biến đổi nối tiếp theo mã nhị phân: quá trình so sánh được thực hiện lần lượt từng bước theo qui luật của mã nhị phân. Các đơn vị chuẩn dùng để so sánh lấy các giá trị giảm dần theo quy luật của mã nhi phân. Do đó các bít được xác định lần lượt theo từ bít có ý nghĩa lớn nhất đến bít có ý nghĩa nhỏ nhất.

- Biến đổi song song - nối tiếp kết hợp: trong phương pháp này, qua mỗi bước so sánh có thể xác định được tối thiểu là hai bít đồng thời.

Sử dụng bộ chuyển đổi trong hệ thống

Trong hệ thống ở đây ta sử dụng IC chuyển đổi 12 bits MCP3204 có sơ đồ chân như trong hình 3.4.



Hình 3.4. Sơ đồ chân IC MPC3204

CH0 : Kênh tín hiệu tương tự vào 0

CH1 : Kênh tín hiệu tương tự vào 1

CH2 : Kênh tín hiệu tương tự vào 2

CH3 : Kênh tín hiệu tương tự vào 3

N/C : không nối.

DGND : Digital Ground.

CS: Chọn chíp.

Din : Kết nối tới AVRs MOSI

Dout : Kết nối tới AVRs MISO

CLK : Kết nối tới AVRs SCK

Agnd : Analog Ground

Vref : Điện áp so sánh.

Vdd : Nguồn (5v).

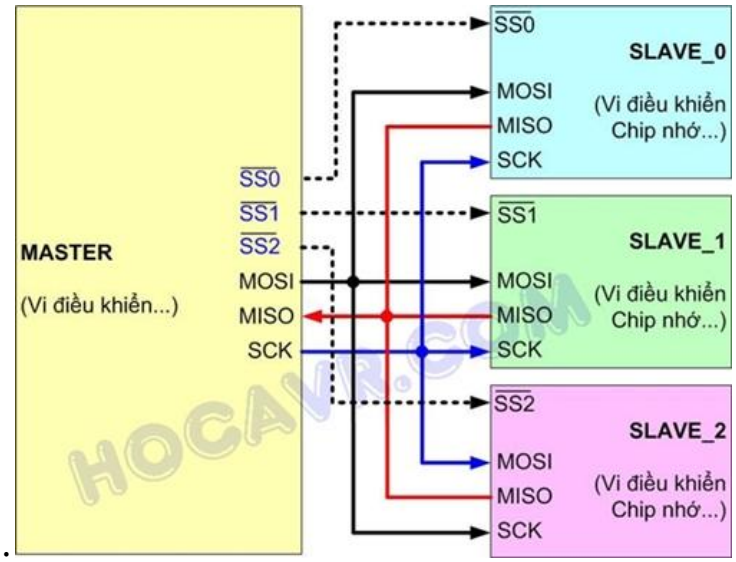
IC MCP3204 thực hiện giao tiếp theo chuẩn SPI (Serial Peripheral Bus) là một chuẩn truyền thông nối tiếp tốc độ cao do hãng Motorola đề xuất. Đây là kiểu truyền thông Master-Slave, trong đó có 1 chip Master điều phối quá trình tuyền thông và các chip Slaves được điều khiển bởi Master vì thế truyền thông chỉ xảy ra giữa Master và Slave. SPI là một cách truyền song công (full duplex) nghĩa là tại cùng một thời điểm quá trình truyền và nhận có thể xảy ra đồng thời. SPI đôi khi được gọi là chuẩn truyền thông "4 dây" vì có 4 đường giao tiếp trong chuẩn này đó là SCK (Serial Clock), MISO (Master Input Slave Output), MOSI (Master Ouput Slave Input) và SS (Slave Select). Hình 1 thể hiện một kết SPI giữa một chip Master và 3 chip Slave thông qua 4 đường.

SCK: Xung giữ nhịp cho giao tiếp SPI, vì SPI là chuẩn truyền đồng bộ nên cần 1 đường giữ nhịp, mỗi nhịp trên chân SCK báo 1 bit dữ liệu đến hoặc đi. Đây là điểm khác biệt với truyền thông không đồng bộ mà chúng ta đã biết trong chuẩn UART. Sự tồn tại của chân SCK giúp quá trình tuyền ít bị lỗi và vì thế tốc độ truyền của SPI có thể đạt rất cao. Xung nhịp chỉ được tạo ra bởi chip Master.

MISO– Master Input / Slave Output: nếu là chip Master thì đây là đường Input còn nếu là chip Slave thì MISO lại là Output. MISO của Master và các Slaves được nối trực tiếp với nhau.

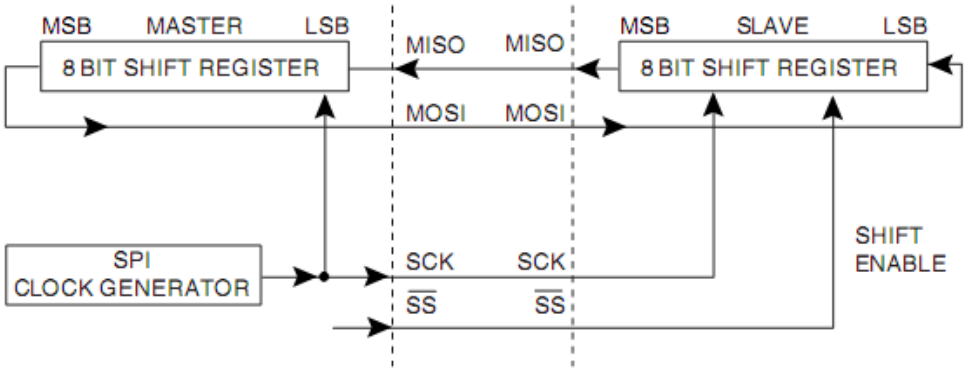
MOSI – Master Output / Slave Input: nếu là chip Master thì đây là đường Output còn nếu là chip Slave thì MOSI là Input. MOSI của Master và các Slaves được nối trực tiếp với nhau.

SS – Slave Select: SS là đường chọn Slave cần giap tiếp, trên các chip Slave đường SS sẽ ở mức cao khi không làm việc. Nếu chip Master kéo đường SS của một Slave nào đó xuống mức thấp thì việc giao tiếp sẽ xảy ra giữa Master và Slave đó. Chỉ có 1 đường SS trên mỗi Slave nhưng có thể có nhiều đường điều khiển SS trên Master, tùy thuộc vào thiết kế của người dùng.



Hình 3.5. Giao diện SPI.

Mỗi chip Master hay Slave có một thanh ghi dữ liệu 8 bits. Cứ mỗi xung nhịp do Master tạo ra trên đường giữ nhịp SCK, một bit trong thanh ghi dữ liệu của Master được truyền qua Slave trên đường MOSI, đồng thời một bit trong thanh ghi dữ liệu của chip Slave cũng được truyền qua Master trên đường MISO. Do 2 gói dữ liệu trên 2 chip được gởi qua lại đồng thời nên quá trình truyền dữ liệu này được gọi là "song công". Hình 3.6. mô tả quá trình truyền 1 gói dữ liệu thực hiện bởi module SPI trong AVR, bên trái là chip Master và bên phải là Slave.

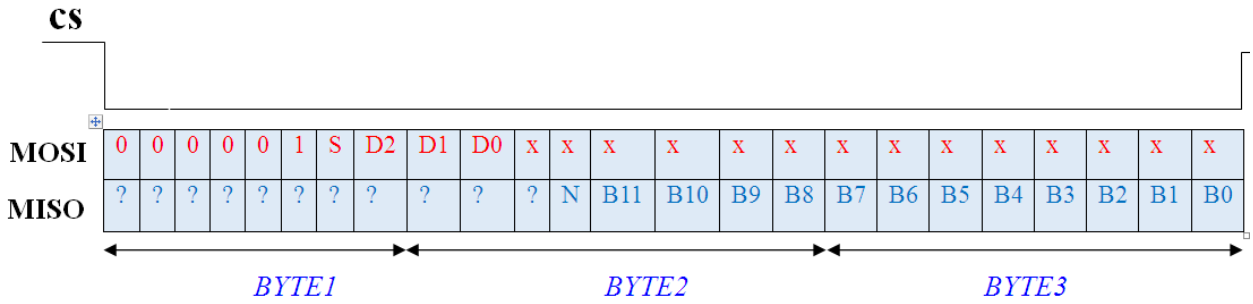


Hình 3.6. Truyền dữ liệu SPI.

Cực của xung giữ nhịp, phase và các chế độ hoạt động: cực của xung giữ nhịp (Clock Polarity) được gọi tắt là CPOL là khái niệm dùng chỉ trạng thái của

chân SCK ở trạng thái nghỉ. Ở trạng thái nghỉ (Idle), chân SCK có thể được giữ ở mức cao (CPOL=1) hoặc thấp (CPOL=0). Phase (CPHA) dùng để chỉ cách mà dữ liệu được lấy mẫu (sample) theo xung giữ nhịp. Dữ liệu có thể được lấy mẫu ở cạnh lên của SCK (CPHA=0) hoặc cạnh xuống (CPHA=1). Sự kết hợp của SPOL và CPHA làm nên 4 chế độ hoạt động của SPI. Nhìn chung việc chọn 1 trong 4 chế độ này không ảnh hưởng đến chất lượng truyền thông mà chỉ cốt sao cho có sự tương thích giữa Master và Slave.

Các gói SPI cho chuyển đổi ADC MCP3204 được tạo thành từ 3byte:



Byte 1

Trong byte này, Master ghi 1 chuỗi bit: '0' '0' '0' '0' '0' '1' 'S' 'D2'

Byte 2

Trong byte này, Master ghi theo trình tự: 'D1' 'D0' 'X' 'X' 'X' 'X' 'X 'X'

Trong đó D2, D1, D0 lựa chon kênh tín tiệu tương tự vào.

	D2	D1	D0
Channel 0	0	0	0
Channel 1	0	0	1
Channel 2	0	1	0
Channel 3	0	1	1

Và Slaver trả về theo trình tự: '?' '?' '?' 'N' 'B11' 'B10' 'B9' 'B8'

Trong đó ? là bit không xác định. 'N' là bit rỗng và luôn bằng “0”

Byte 3

Trong byte này, Master ghi 1 byte không vào slave.

Trong cùng thời gian, Slaver trả về giá trị chuyển đổi các bit B0 tới B7.

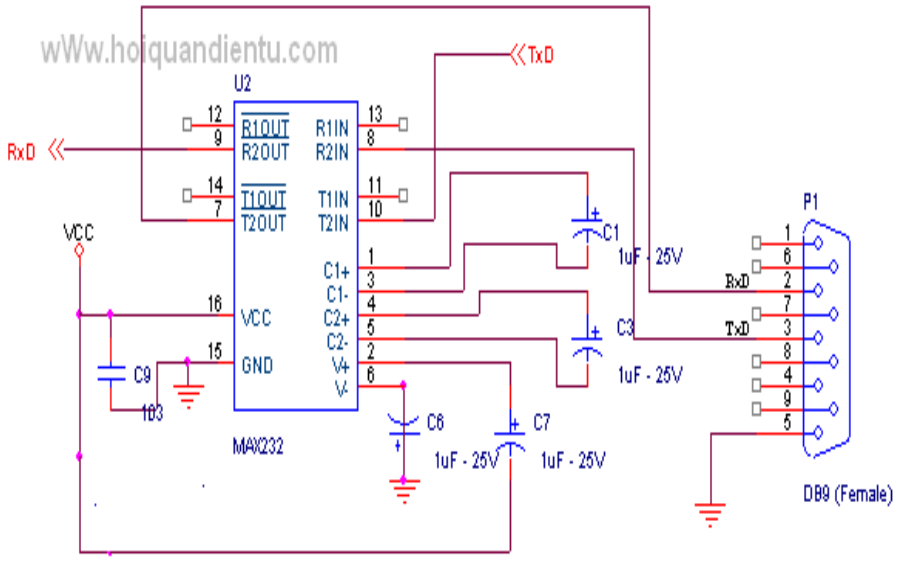
*** Khối Vi điều khiển**

Thực hiện giao tiếp với ADC MCP3204 theo chuẩn SPI để nhận giá trị đo lường và giao tiếp với máy tính theo chuẩn RS232 để nhận tín hiệu điều khiển từ máy tính và truyền dữ liệu về. Ở đây em sử dụng vi điều khiển AVR Atmega32.

*** Khối giao tiếp cổng COM**

Có rất nhiều mạch giao tiếp theo chuẩn RS232 giữa máy tính và các thiết bị khác. Ở đây em dùng mạch chuẩn giao tiếp RS232 với IC Max232.

Max232 là IC chuyên dùng cho giao tiếp giữa RS232 và thiết bị ngoại vi. Max232 là IC của hãng Maxim. Đây là IC chay ổn định và được sử dụng phổ biến trong các mạch giao tiếp chuẩn RS232. Giá thành của Max232 phù hợp và tích hợp trong đó hai kênh truyền cho chuẩn RS232. Dòng tín hiệu được thiết kế cho chuẩn RS232. Mỗi đầu truyền ra và cổng nhận tín hiệu đều được bảo vệ chống lại sự phóng tĩnh điện. Ngoài ra Max232 còn được thiết kế với nguồn +5V cung cấp nguồn công suất nhỏ. Mạch giao tiếp như hình 3.7.

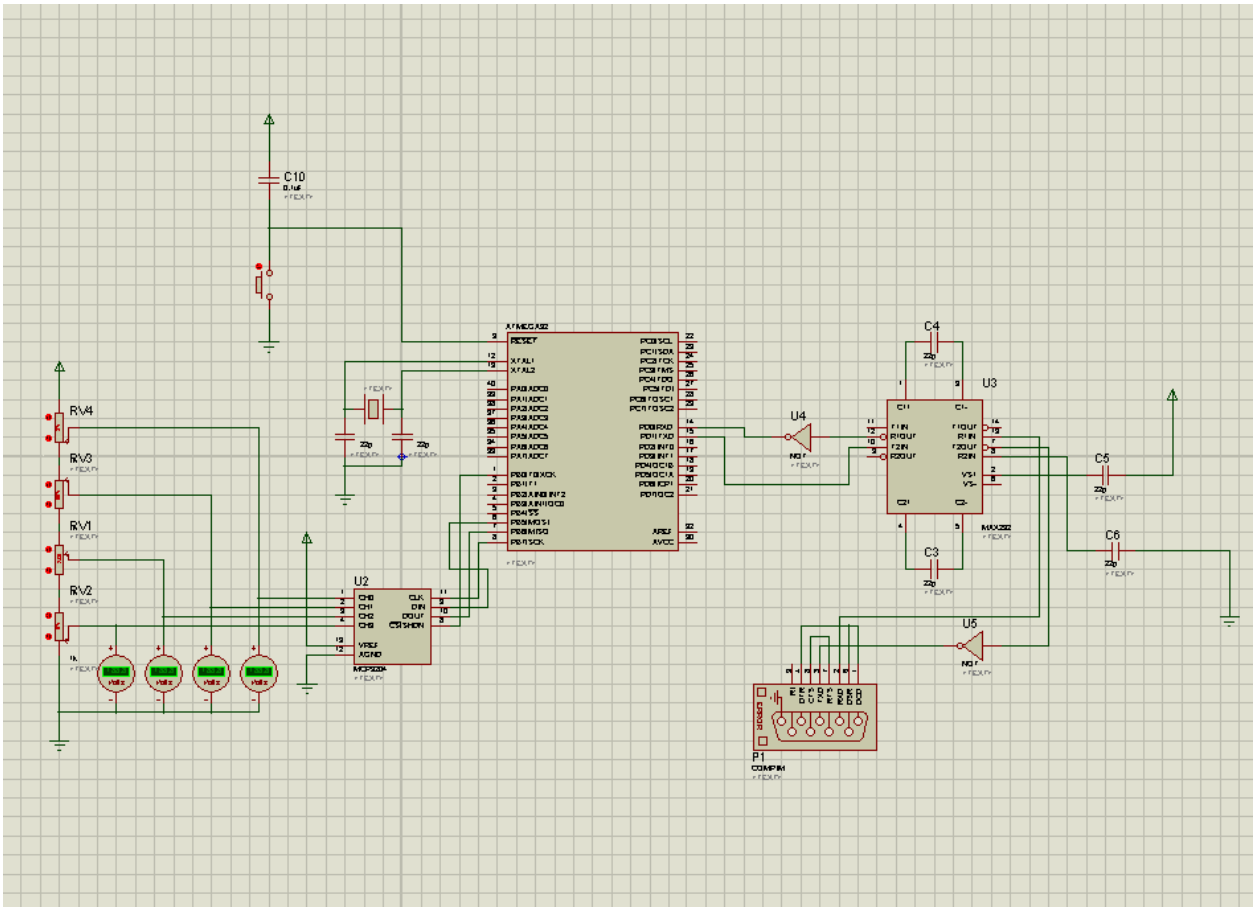


Hình 3.7. Sơ đồ giao tiếp RS 232 dùng MAX 232

- Vi mạch MAX 232 có nhiệm vụ chuyển đổi mức TTL ở lối vào thành mức +3...+15V hoặc -3...-15V thành mức TTL ở phía nhận hay mức +10V hoặc –10V ở phía truyền.

- Vi mạch MAX 232 có hai bộ đệm và hai bộ nhận. Đường dẫn điều khiển lối vào CTS, điều khiển việc xuất ra dữ liệu ở cổng nối tiếp khi cần thiết, được nối với chân 9 của vi mạch MAX 232. Còn chân RST (chân 10 của vi mạch MAX) nối với đường dẫn bắt tay để điều khiển quá trình nhận. Thường thì các đường dẫn bắt tay được nối với cổng nối tiếp qua các cầu nối, để khi không dùng đến nữa có thể hở mạch các cầu này. Cách truyền dữ liệu đơn giản nhất là chỉ dùng ba đường dẫn TxD, RxD và GND (mass).

*** Mạch nguyên lý**



Hình 3.8. Mạch nguyên lý giao tiếp máy tính và vi điều khiển

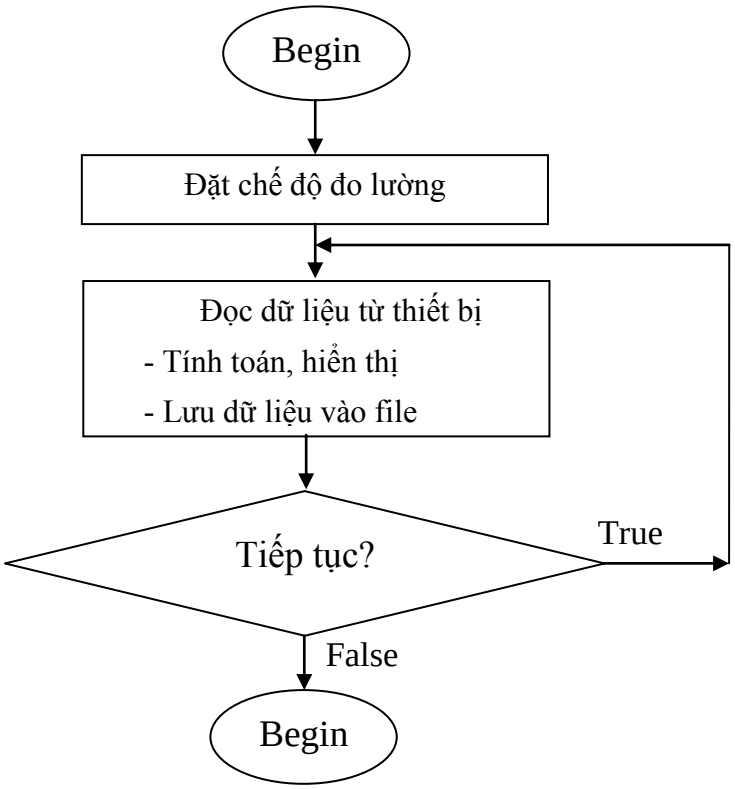
3.2. Xây dựng chương trình phần mềm

3.2.1. Chương trình cho PC

Chương trình được viết bằng Visual Basic, có giao diện như trong hình 3.9.

Hình 3.9. Giao diện chương trình đo lường nhiều kênh

*** Lưu đồ thuật toán**



*** Mã chương trình**

```
Private Declare Sub Sleep Lib "kernel32.dll" (ByVal dwMilliseconds As Long)

Dim Inputuart As String
Private Sub Check1_Click()
End Sub
Private Sub Command1_Click()
End
End Sub

Private Sub Command2_Click()
Timer1.Enabled = True
End Sub

Private Sub Command3_Click()
Timer1.Enabled = False
End Sub

Private Sub Command4_Click()
Me.MSComm1.Output = Chr(0)
End Sub

Private Sub Command5_Click()
Me.MSComm1.Output = Chr(1)
End Sub

Private Sub Command6_Click()
```

```
Me.MSComm1.Output = Chr(2)
End Sub

Private Sub Command7_Click()
Me.MSComm1.Output = Chr(3)
End Sub

Private Sub Form_Load()
MSComm1.CommPort = 4
MSComm1.Settings = "9600,N,8,1"
Me.MSComm1.RThreshold = 1
MSComm1.InputLen = 1
MSComm1.PortOpen = True

Check1.Value = 1
Check2.Value = 0
Check3.Value = 0
Check4.Value = 0

End Sub

Private Sub MSComm1_OnComm()
Dim InputText As String
If Me.MSComm1.CommEvent = comEvReceive Then
   InputText = MSComm1.Input
   txtOutput.Text = txtOutput.Text + InputText
```

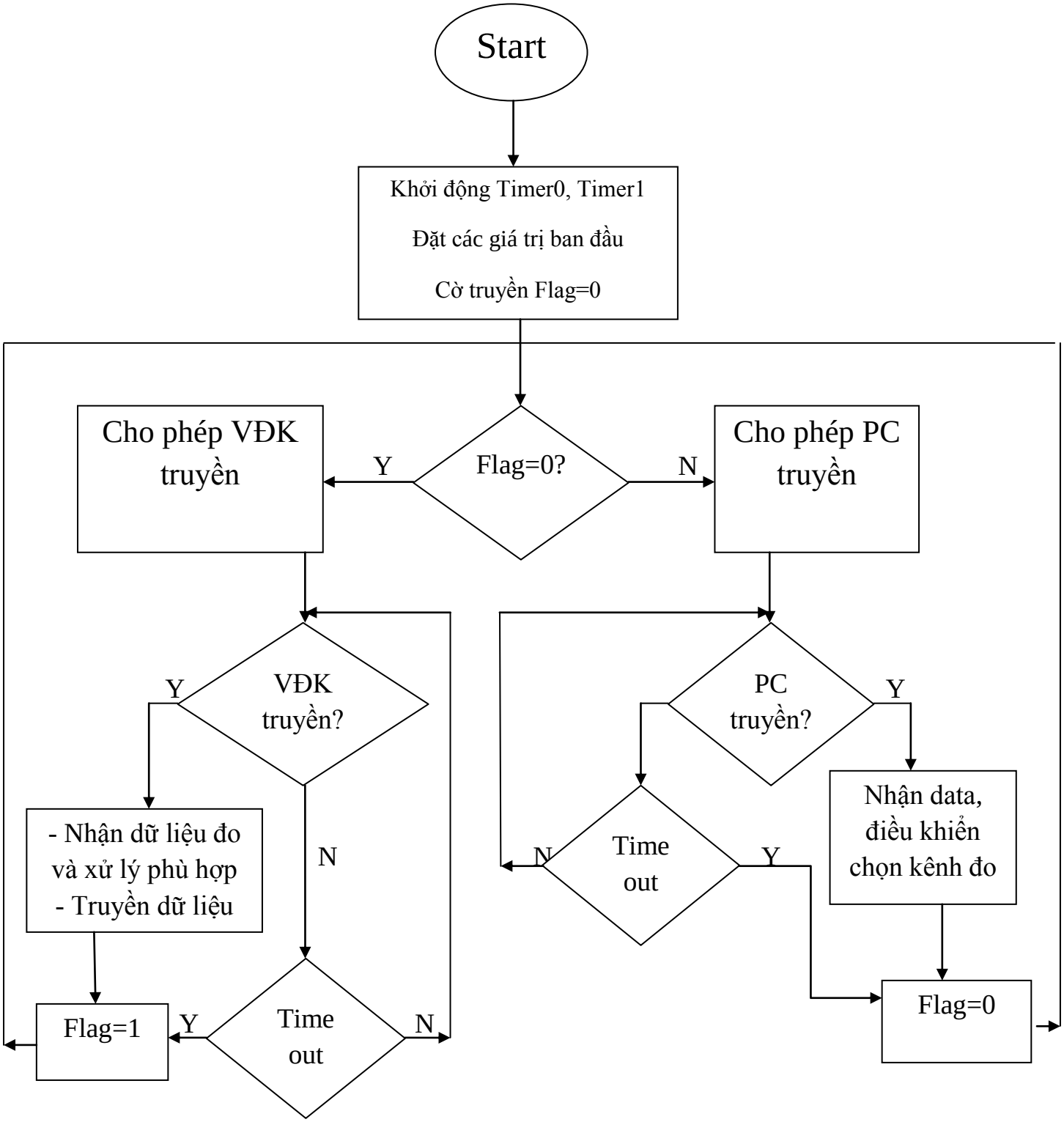
```
    txtOutput.SelStart = Len(txtOutput.Text)
    Sleep (50)
End If
End Sub

Private Sub Text1_Change()
End Sub

Private Sub Text5_Change()
End Sub

Private Sub txtInput_Change()
End Sub

Private Sub txtInput_KeyPress(KeyAscii As Integer)
Me.MSComm1.Output = Chr(Key)
End Sub

Private Sub txtOutput1_Change()
End Sub

Private Sub Timer1_Timer()
If Check1.Value = 1 Then
Me.MSComm1.Output = Chr(0)
End If
If Check2.Value = 1 Then
```

```
Me.MSComm1.Output = Chr(1)

End If

If Check3.Value = 1 Then

Me.MSComm1.Output = Chr(2)

End If

If Check4.Value = 1 Then

Me.MSComm1.Output = Chr(3)

End If

If (Check1.Value = 0) And (Check2.Value = 0) And (Check3.Value = 0) And
(Check4.Value = 0) Then

txtOutput.Text = "Ban hay chon mot kenh!"

End If

End Sub
```

```
 Dim FileNum As Integer
   RecordLen = Len(dataArray(1))
   FileNum = FreeFile
   Open "Data.Dat" For Random As FileNum Len = RecordLen
   For i = 1 To sampleSize - 1
      dataArray(i) = 10 * Sin(10 * 3.1415 * i / sampleSize)
      Put #FileNum, i, dataArray(i)
   Next i

End Sub
```

3.2.2. Chương trình viết cho vi điều khiển

*** Lưu đồ gải thuật cho vi điều khiển.**



*** Mã chương trình**

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stdio.h>
#include "adc_ex.h"

void uart_char_tx(unsigned char chr)
{
  if (chr == '\n')
  uart_char_tx('\r');
  while (bit_is_clear(UCSRA,UDRE))
  {
  //cho den khi bit UDRE=1
  }
  UDR=chr;
}

static FILE uartstd= FDEV_SETUP_STREAM(uart_char_tx,
NULL,_FDEV_SETUP_WRITE);

uint16_t result;
unsigned char u_Data;
```

```
int main(void)
{
        //Initialize External ADC Module
        InitADCEx();
        //Khoi dong Module Uart
   //set baud, 57.6k ung voi f=8Mhz, xem bang 70 trang 165, Atmega32 datasheet
   UBRRH=0;
   UBRRL=103;
        //set khung truyen va kich hoat bo nhan du lieu
   UCSRA=0x00;
   UCSRC=(1<<URSEL)|(1<<UCSZ1)|(1<<UCSZ0);
   UCSRB=(1<<RXEN)|(1<<TXEN)|(1<<RXCIE);//cho phep ca 2 qua trinh nhan va//truyen, va cho  phep ngat sau khi nhan xong
   sei(); //cho phep ngat toan cuc
        while(1)
        {
        }
}
ISR(SIG_UART_RECV)
{
          //trinh phuc vu ngat USART hoan tat nhan
     u_Data=UDR;
          if (u_Data==0)
          {
```

```
        result=ReadADCEx(0);
        fprintf(&uartstd,"Gia tri kenh 0: %i\n", result);
        }
        if (u_Data==1)
        {
        result=ReadADCEx(1);
        fprintf(&uartstd,"Gia tri kenh 1: %i\n", result);
        }
        if (u_Data==2)
        {
        result=ReadADCEx(2);
        fprintf(&uartstd,"Gia tri kenh 2: %i\n", result);
        }
        if (u_Data==3)
        {
        result=ReadADCEx(3);
        fprintf(&uartstd,"Gia tri kenh 3: %i\n", result);
        }
}
```

KẾT LUẬN.

Sau thời gian thực hiện đề tài tốt nghiệp với sự hướng dẫn tận tình của thầy Nguyễn Văn Dương cùng những cố gắng của bản thân em đã hoàn thành đề tài tốt nghiệp **"*Thiết kế hệ thống ghép nối máy tính đo lường nhiều kênh*"**.

Đánh giá kết quả

Bản đồ án đã thiết kế hoàn chỉnh hệ thống ghép nối máy tính đo lường nhiều kênh. Hệ thống đã chạy tốt trên phần mềm mô phỏng Proteus. Tuy nhiên do thời gian làm đồ án và kiến thức bản thân còn hạn chế vì vậy bản đồ án còn nhiều thiếu sót, mới chỉ dừng lại ở mức mô phỏng, và thiết kế chưa thực sự tối ưu. Vậy để phát triển đề tài thành sản phẩm tiêu dùng thực sự yêu thích, có thể được ứng dụng vào thực tế đóng góp cho ngành công nghiệp nước nhà, em rất mong được sự chỉ bảo, góp ý của các Thầy cô và bạn bè.

TÀI LIỆU THAM KHẢO

1. Hồ Trung Mỹ (2007) ***Vi xử lý***. Nhà xuất bản đại học Quốc Gia
2. Datasheet Atmega32.
3. Pham Hùng Kim Khánh(2008) ***Tài liệu lập trình hệ thống.*** Nhà xuất bản Đại học Kỹ thuật Công nghệ TPHCM.
4. ***Giáo trình visual basic 6.0***. Nhà xuất bản đại học FPT.
5. Ngô Diện Tập, ***Vi Điều Khiển trong đo l□ờng và điều khiển tự động***. Nhà xuất bản Khoa Hoc và Kỹ Thuật Hà Nội.
6. Các tài liệu từ internet, từ diễn đàn

   www.hocavr.com/

   http://hoiquandientu.com/

   www.dientuvietnam.net/

   www.vagam.dieukhien.net

   www.duyphi.phpnet.us/index.htm