|  |
| --- |
| TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI  **TRƯỜNG ĐIỆN – ĐIỆN TỬ**  logo_128  **ĐỒ ÁN II**  **Đề tài:**  **MẠCH ĐO NHIỆT ĐỘ SỬ DỤNG CẢM BIẾN LM35**  Sinh viên thực hiện: Nguyễn Xuân Hoàng -20203433  Nguyễn Bình Độ - 20203361  Hà Quang Huy - 20203719  Giảng viên hướng dẫn: Thầy Tào Văn Cường    Hà Nội, 2024 |

**LỜI MỞ ĐẦU**

Trong thời kì hội nhập ngày nay, để có thể bắt kịp các nước phát triển và vươn lên, Việt Nam cần một đội ngũ kỹ sư lành nghề, có hiểu biết sâu rộng và có khả năng giải quyết các vấn đề phức tạp. Trong hoàn cảnh công nghệ hiện đại ngày càng được sử dụng rộng rãi trong đời sống, việc sử dụng nhuần nhuyễn công nghệ như là một công cụ để học tập và làm việc là ưu tiên hàng đầu. Tuy nhiên, là một sinh viên thuộc Đại học Bách khoa Hà Nội, khoa Điện tử viễn thông, em không những cần học cách sử dụng các thành tựu công nghệ của thế giới như một công cụ sao cho hiệu quả mà còn cần phải hiểu rõ cách mà chúng hoạt động, tìm hiểu các thành phần từ những bộ phận nhỏ nhất, nhằm có thể tối ưu hóa, cải tiến chúng, tăng năng suất, góp phần đẩy mạnh cuộc cách mạng khoa học công nghệ.

Từ máy tính cá nhân, máy tính bảng, điện thoại thông minh, vô tuyển đến máy giặt, tủ lạnh, hay Router, bộ phát sóng Wifi đều có ít nhất một vài vi xử lý được cài đặt trong chúng. Thực tế cuộc sống ta có thể thấy, các vi xử lý hiện hữu khắp nơi từ những đồ dùng quen thuộc hằng ngày đến đến những hệ thống lớn. Các bộ vi xử lý, tuy nhỏ bé nhưng có vai trò vô cùng to lớn, được thiết kế lập trình để điều khiển các hoạt động của một bộ phận nhất định trong hệ thống. Để thiết kế một hệ thống hoạt động tốt, hiệu quả cần có những bộ vi xử lý phù hợp với nhu cầu và lập trình để thực hiện các chức năng phù hợp. Người thiết kế và cài đặt những vi xử lý này đa phần là những kỹ sư của ngành Điện tử viễn thông.

Với môn học Đồ án thiết kế II, dưới sự chỉ dẫn của thầy Tào Văn Cường, chúng em quyết định chọn đề tài “Thiết kế mạch đo nhiệt độ sử dụng Kit AVR” do cảm thấy rằng đề tài này vô cùng hữu ích đối với cuộc sống, cũng như là một cơ hội tốt để em tiếp tục cải thiện kiến thức của mình về chuyên ngành Điện tử kĩ thuật máy tính. Qua quá trình học tập, nghiên cứu để thực hiện đề tài này, em không những nắm bắt được những kiến thức cơ bản về vi xử lý, cảm biến nhiệt độ, về cách thức lập trình cho vi xử lý trong máy tính, hoàn thiện và hiểu kĩ hơn về kiến thức đã được học trong nhiều môn khác trong chương trình học, mà em còn hiểu rõ hơn vai trò của bản thân mình đối với cuộc cách mạng công nghệ đang diễn ra và được đẩy mạnh hơn bao giờ hết. Đây chính là một trong những động lực để em ra sức học tập và đóng góp vào cuộc cách mạng cộng nghệ của đất nước và thế giới.

Qua đây chúng em xin phép bày tỏ lòng biết ơn đến sự dạy dỗ và chỉ bảo của thầy Tào Văn Cường, người trực tiếp hướng dẫn, chỉ bảo và giúp đỡ em học hỏi và tìm hiểu để có thể hoàn thành đồ án này. Chúng em xin chân thành cảm ơn sự hướng dẫn tận tình của thầy

# DANH MỤC HÌNH VẼ

[Hình 1.1.2 Cấu trúc các khối Atmega16 3](#_Toc60003919)

[Hình 1.1.3a Sơ đồ khối AVR MCU 4](#_Toc60003920)

[Hình 1.1.3b Các thanh ghi chức năng chung 5](#_Toc60003921)

[Hình 1.1.4 Cấu hình chân Atmega16 7](#_Toc60003922)

[Hình 1.1.5a Cấu trúc khối timer/counter1 9](#_Toc60003923)

[Hình 1.1.5b Khối bộ đếm 10](#_Toc60003924)

[Hình 1.2.1 Sơ đồ chân LM35 11](#_Toc60003925)

[Hình 1.2.2 LCD 16x2 12](#_Toc60003926)

[Hình 1.3.1 Mạch Kit phát triển và các phụ kiện đi kèm 12](#_Toc60003927)

[Hình 1.3.4 Sơ đồ nguyên lý mạch Kit 14](#_Toc60003928)

[Hình 2. 1a Sơ đồ khối 19](file:///D:\word\baocao_doan2.docx#_Toc60003967)

[Hình 2. 1b Sơ đồ nguyên lí mạch kit 20](#_Toc60003968)

[Hình 2.2 Đi dây mạch in AVR 20](#_Toc60003969)

# DANH MỤC KÝ HIỆU VÀ CHỮ VIẾT TẮT

|  |  |
| --- | --- |
| AVR | Alf and Vegard’s RISC processor |
| RISC | Reduced instruction set computer |
| LCD | Liquid Crystal Display |
| MIPS | Million Instructions Per Second |
| ADC | Analog to Digial Converter |
| OOP | Object Orriented Programming |
| CPU | Central Processsing Unit |

Mục lục

[DANH MỤC HÌNH VẼ i](#_Toc60009662)

[DANH MỤC KÝ HIỆU VÀ CHỮ VIẾT TẮT ii](#_Toc60009663)

[CHƯƠNG 1: CƠ SỞ LÍ THUYẾT TỔNG QUAN 1](#_Toc60009664)

[1.1. Tìm hiểu vi điều khiển Atmega16 1](#_Toc60009665)

[1.1.1. Tổng quan về Atmega16 1](#_Toc60009666)

[1.1.2. Cấu trúc Atmega16 3](#_Toc60009667)

[1.1.3. ARV CPU Core 4](#_Toc60009668)

[1.1.4. Các cổng I/O (xuất nhập) 7](#_Toc60009669)

[1.1.5. Bộ định thời, bộ đếm Timer/Counter 9](#_Toc60009670)

[1.2. Các linh kiện được sử dụng 11](#_Toc60009671)

[1.2.1. Cảm biến nhiệt độ LM35 11](#_Toc60009672)

[1.2.2. Màn hình hiển thị LCD 12](#_Toc60009673)

[1.3. Mạch Kit cho vi điều khiển AVR 14](#_Toc60009674)

[1.3.1. Tổng quan về mạch Kit AVR 14](#_Toc60009675)

[1.3.2. Cấu trúc mạch Kit 15](#_Toc60009676)

[1.3.3. Các thông số kỹ thuật chính 15](#_Toc60009677)

[1.3.4. Sơ đồ nguyên lý mạch Kit 16](#_Toc60009678)

[CHƯƠNG 2: THIẾT KẾ MẠCH 19](#_Toc60009683)

[2.1. Sơ đồ khối 19](#_Toc60009684)

[2.2. Mạch in Kit AVR 20](#_Toc60009685)

[CHƯƠNG 3: KẾT QUẢ THỰC HIỆN 21](#_Toc60009687)

[3.1. Kết quả thực hiện đề tài 21](#_Toc60009688)

[3.2. Yêu cầu phi chức năng 21](#_Toc60009689)

[3.3. Hướng phát triển 21](#_Toc60009690)

[KẾT LUẬN 23](#_Toc60009694)

[DANH MỤC TÀI LIỆU THAM KHẢO 24](#_Toc60009695)

[PHỤ LỤC 25](#_Toc60009696)

[Phụ lục 1: Source code 25](#_Toc60009697)

# Chương 1: Cơ sở lí thuyết tổng quan

Ở chương này chúng em chủ yếu nói về cấu trúc, một số chức năng chính của vi điều khiển ATMEGA16 (họ vi điều khiển AVR) sử dụng trong đề tài và các phần mềm, linh kiện được sử dụng

## Tìm hiểu vi điều khiển Atmega16

### Tổng quan về Atmega16

Atmega16 là họ IC vi điều khiển do hãng Atmel sản xuất. Các sản phẩm Atmega thích hợp cho những ứng dụng điều khiển. Việc xử lý trên byte và các phép toán số học ở cấu trúc dữ liệu nhỏ được thực hiện bằng nhiều chế độ truy xuất dữ liệu nhanh trên RAM nội. Tập lệnh cung cấp một bảng tiện dụng của những lệnh số học 8 bit gồm cả lệnh nhân và lệnh chia. Nó cung cấp những hổ trợ mở rộng trên chip dùng cho những biến một bit như là kiểu dữ liệu riêng biệt cho phép quản lý và kiểm tra bit trực tiếp trong hệ thống điều khiển.

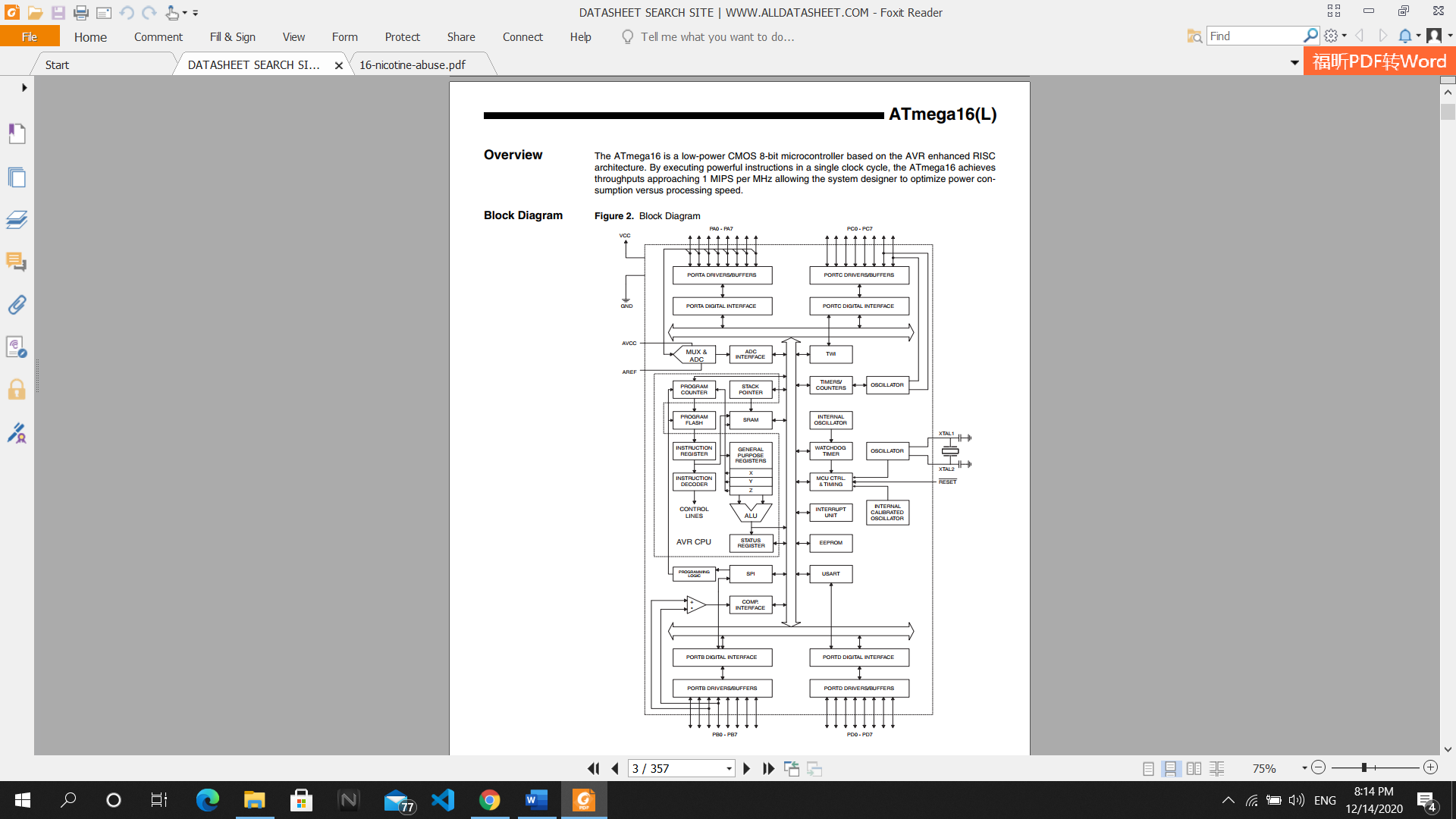
Các đặc điểm của chip Atmega được tóm tắt như sau:

* 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 ( thật ra là Advanced RISC Architecture )
* 130 lệnh thực thi trong vòng 1 chu kì chip
* 32 x 8 thanh ghi công dụng chung ( chắc là 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 cách vào chương trình khởi động chip
* 512 byte EEPROM có thể ghi và xóa 100,000 lần
* 1k byte ram nhớ tĩnh 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 đặt 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 atme

### Cấu trúc Atmega16

ATmega16 là vi điều khiển CMOS 8-bit công suất thấp dựa trên lõi AVR nâng cao.

Kiến trúc RISC. Bằng cách thực hiện các hướng dẫn mạnh mẽ trong một chu kỳ đồng hồ, ATmega16 đạt được thông lượng lên tới gần 1 MIPS trên mỗi MHz cho phép hệ thống thiết kế để tối ưu hóa mức tiêu thụ điện năng so với tốc độ xử lý.

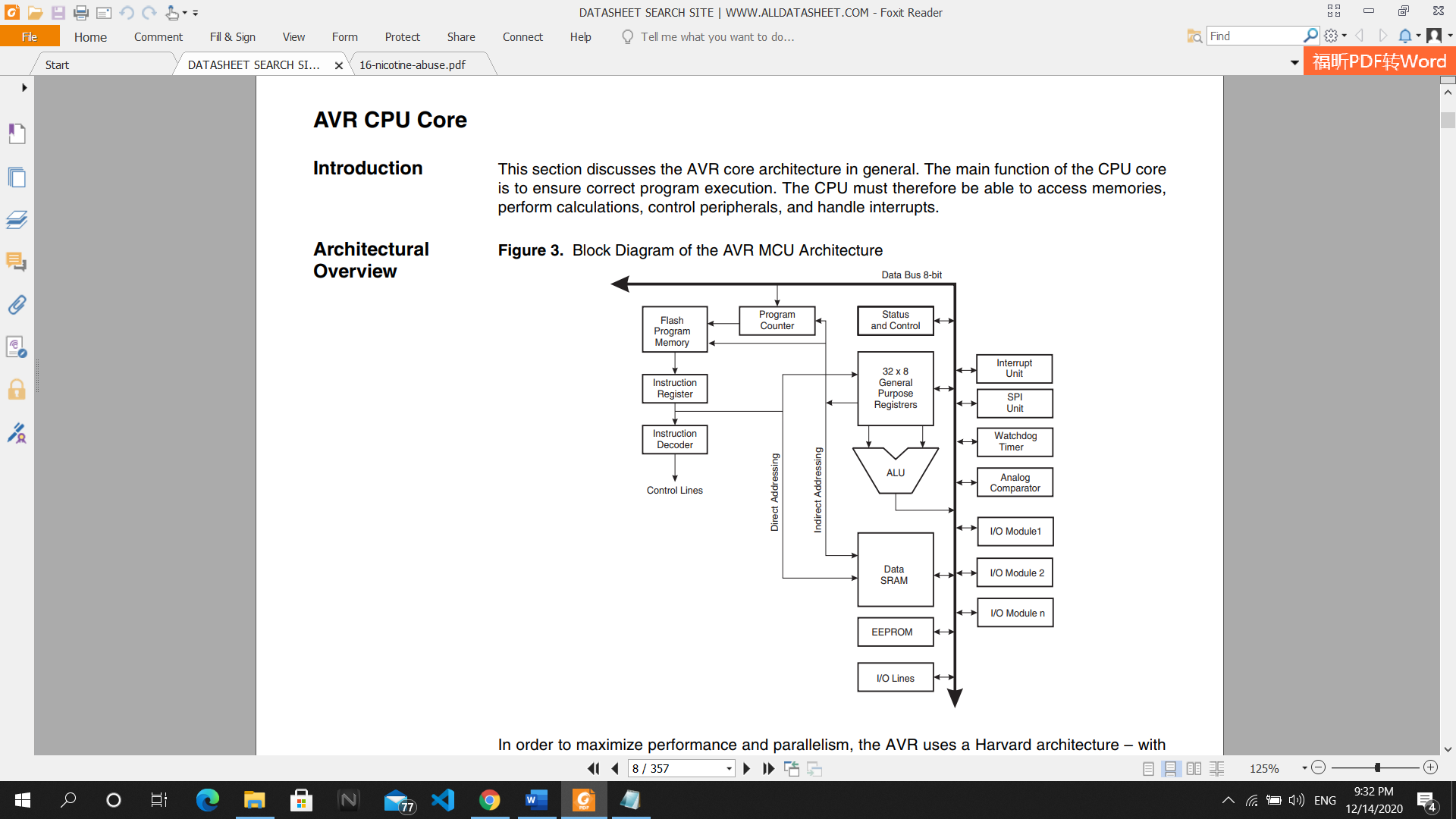


Hình 1.1.2. Cấu trúc các khối Atmega16

Lõi AVR kết hợp một tập lệnh phong phú với 32 thanh ghi làm việc cho mục đích chung.Tất cả 32 thanh ghi được kết nối trực tiếp với đơn vị Logic số học (ALU), cho phép hai thanh ghi độc lập được truy cập trong một lệnh duy nhất được thực thi trong một đồng hồ đi xe đạp. Cấu trúc này nhanh hơn mười ngàn lần CISC

### ARV CPU Core

Lõi CPU đảm bảo thực hiện đúng chương trình. Do đó, CPU phải có khả năng truy cập bộ nhớ, thực hiện các phép tính, điều khiển thiết bị ngoại vi và xử lý ngắt.



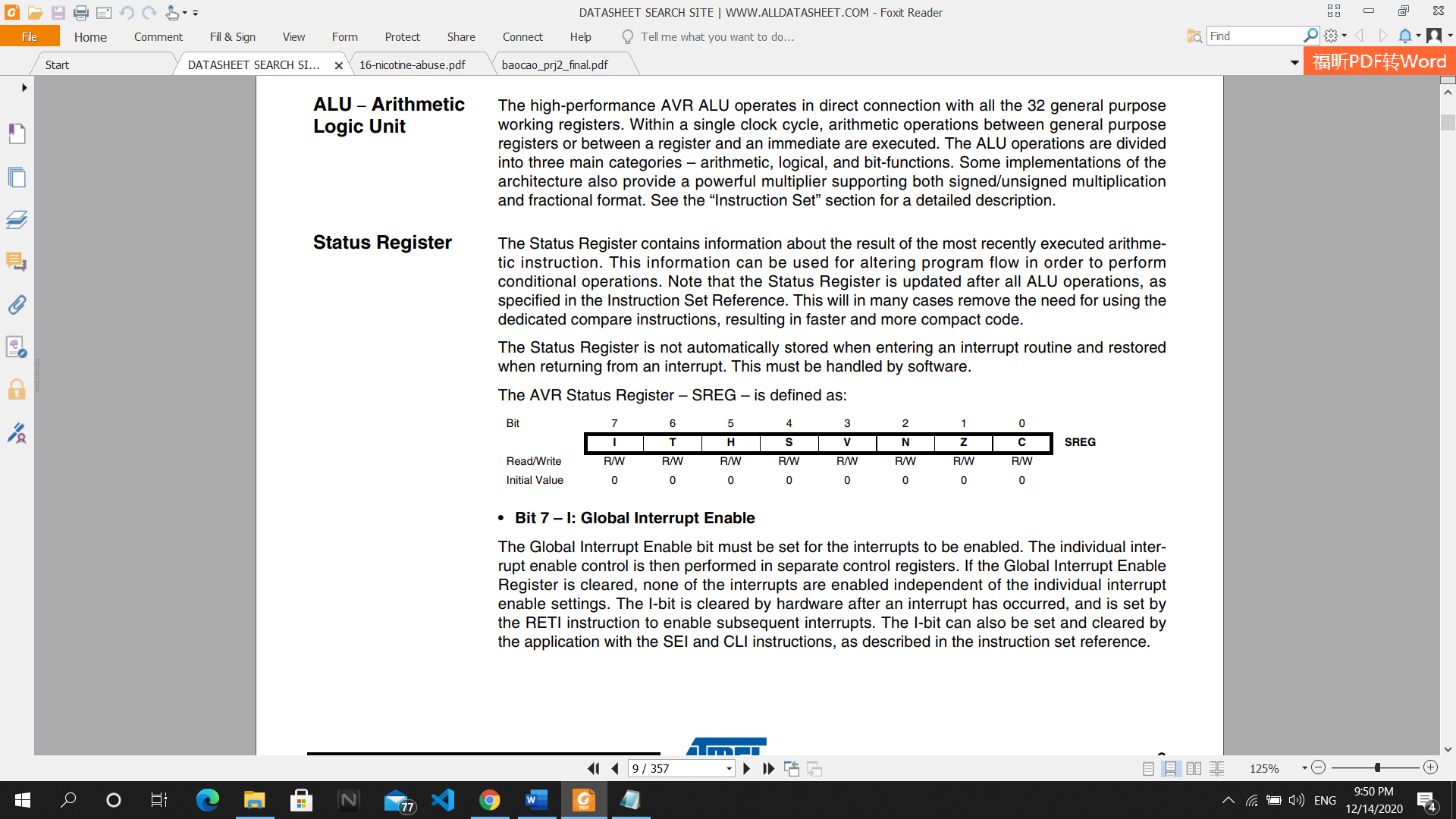
Hình 1.1.3a Sơ đồ khối AVR MCU

**ALU**

ALU làm việc trực tiếp với các thanh ghi chức năng chung. Các phép toán được  
thực hiện trong một chu kì xung clock. Hoạt động của ALU được chia thành 3 loại: đại  
số, logic, theo bit.

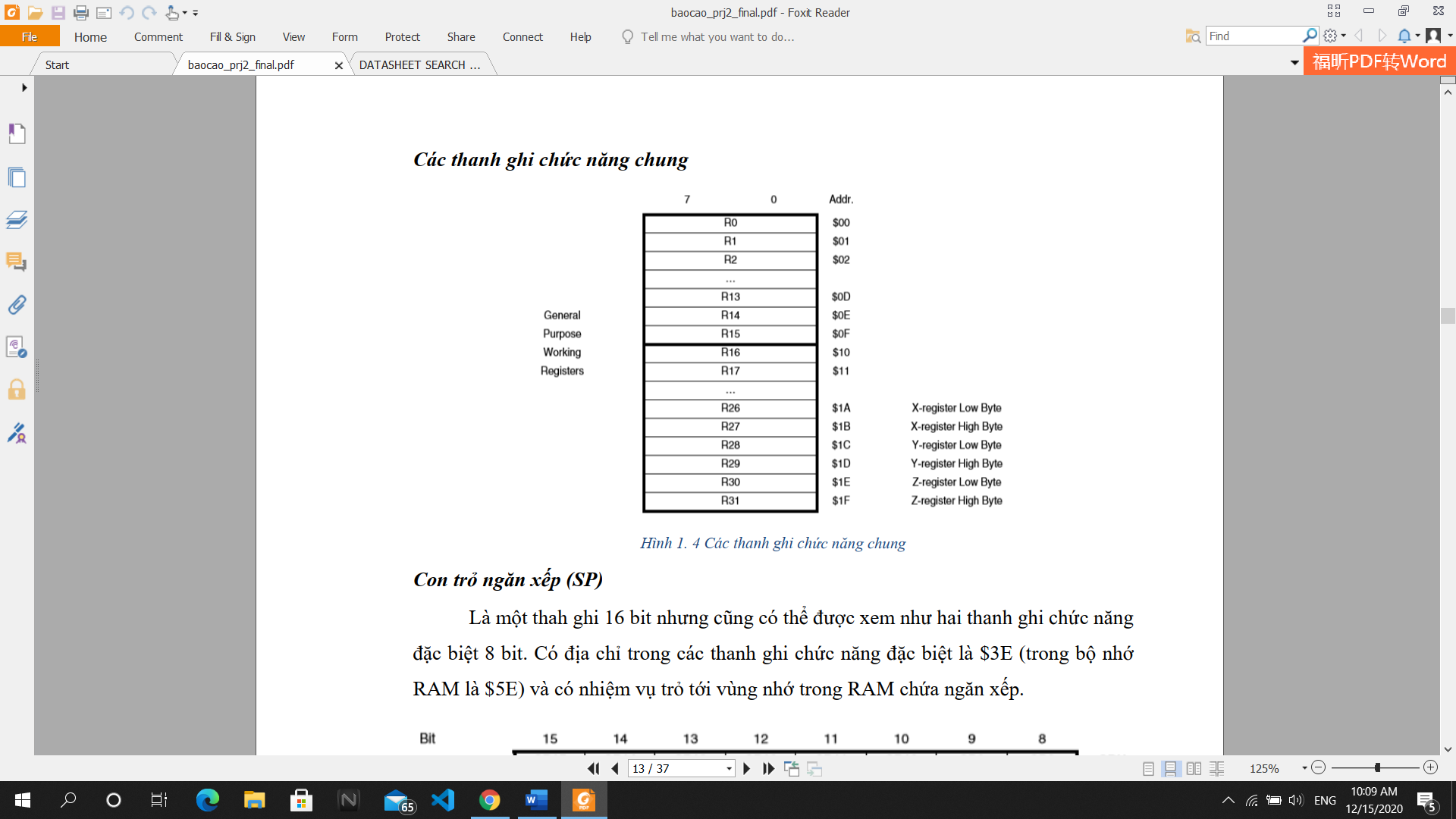
**Thanh ghi trạng thái**

Thanh ghi trạng thái AVR-SREG được định nghĩa:



* **Bit 7-I:** cho phép ngắt toàn cục
* **Bit 6-T:** bộ nhớ sao chép bit
* **Bit 5-H:** cờ half carry
* **Bit 4-S:** bit dấu,S=N XOR V
* **Bit 3-V:** bổ sung 2 cờ tràn
* **Bit 2-N:** cờ Negative
* **Bit 1-Z:** cờ zero
* **Bit 0-C:** cờ carry

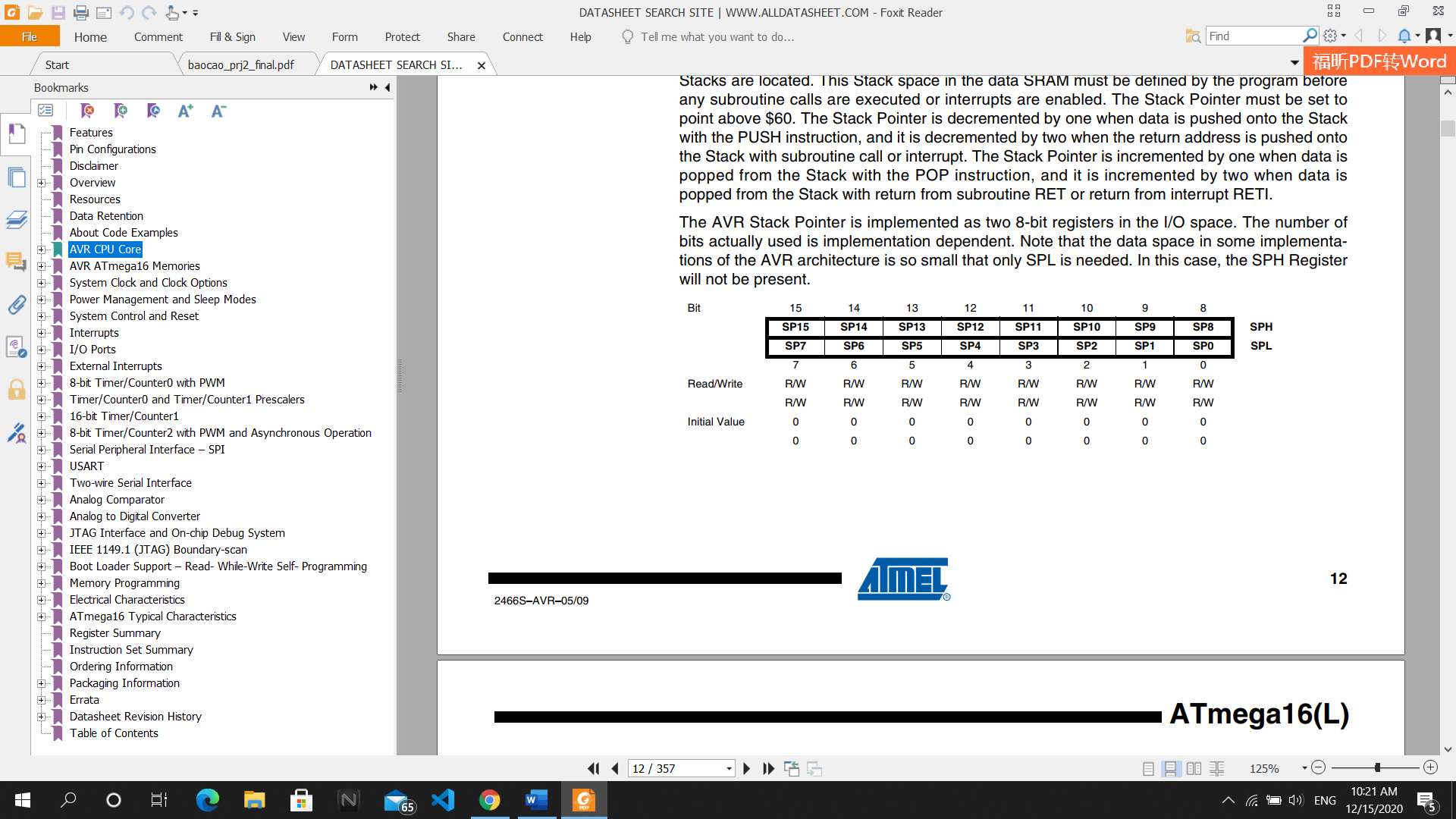
**Các thanh ghi chức năng chung**



Hình 1.1.3b. Các thanh ghi chức năng chung

**Con trỏ ngăn xếp**

Là thanh ghi 16-bit nhưng cũng có thể được coi như là hai thanh ghi 8-bit. Có địa chỉ trong các thanh ghi chức năng đặc biệt như là $3E ( trong bộ nhớ ram là $5E) và có nhiệm vụ trỏ tới bộ nhớ RAM chứa ngăn xếp



**Quản lý ngắt**

Khi có tín hiệu báo ngắt CPU sẽ tạm dừng công việc đang thực hiệnlại và lưu vị trí đang thực hiện chương trình (con trỏ PC) vào ngăn xếp sau đó trỏ tới vector phục vụ ngắt và thực hiện chương trình phục vụ ngắt đó cho tới khi gặp lệnh RETI (Return from Interupt) thì CPU lại lấy PC từ ngăn xếp ra và tiếp tục thực hiện chương trình mà trước khi có ngắt nó đang thực hiện.

Trong trường hợp mà có nhiều ngắt yêu cầu cùng một lúc thì CPU sẽ lưu các cờ báo ngắt đó lại, và thực hiện lần lượt các ngắt theo mức ưu tiên. Trong khi đang thực hiện ngắt mà xuất hiện ngắt mới thì sẽ xảy ra hai trường hợp. Trường hợp ngắt này có ưu tiên cao hơn thì nó sẽ được phục vụ, và ngược lại, nếu có mức ưu tiên thấp hơn thì

nó sẽ bị bỏ qua.

Bộ nhớ ngăn xếp là vùng bất kì trong SRAM từ địa chỉ 0x60 trở lên. Để truy nhập vào SRAM thông thường thì ta dùng con trỏ X, Y, Z và truy cập vào SRAM theo kiểu ngăn xếp thì ta dùng con trỏ SP. Con trỏ nagy là một thanh ghi 16 bit và được truy cập như hai thanh ghi 8 bit chung có địa chỉ: SPL: 0x3D/0x5D (IO/SRAM) và SPH:0x3E/0x5D.

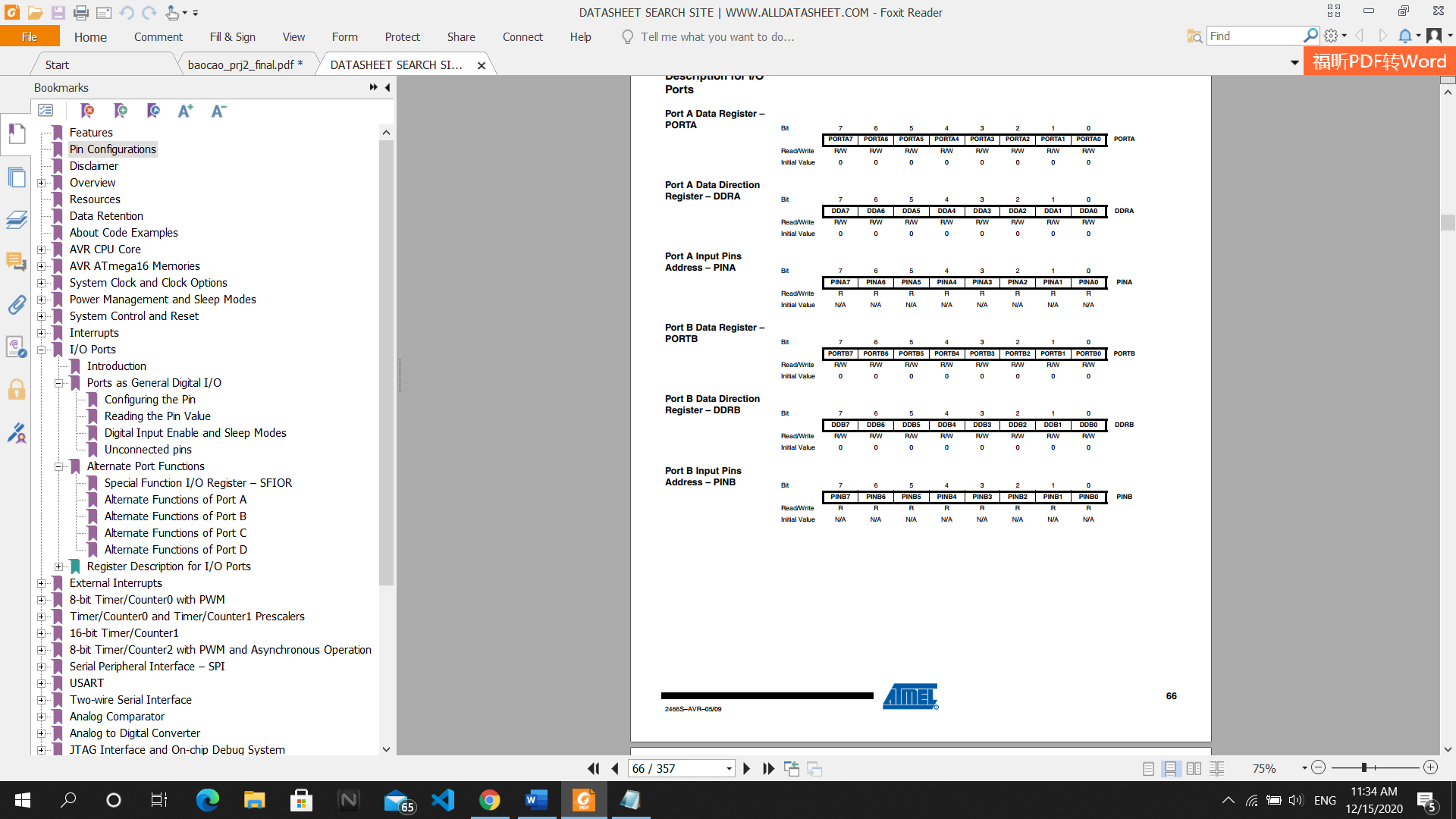
Khi chương trình phục vụ ngắt hoặc chương trình con thì con trỏ PC được lưu vào ngăn xếp trong khi con trỏ ngăn xếp bị giảm hai vị trí. Và con trỏ ngăn xếp sẽ giảm 1 khi thực hiện lệnh PUSH. Ngược lại thực hiện lệnh POP thì con trỏ ngăn xếp sẽ tăng 1 và khi thực hiện lệnh RET hoặc RETI thì con trỏ ngăn xếp sẽ tăng 2. Như vậy, con trỏ ngăn xếp cần được chương trình đặt trước giá trị khởi tạo ngăn xếp trước khi một chương trình con được gọi hoặc các ngắt được cho phép phục vụ. Và giá trị ngăn xếp ít nhất cũng phải lớn hơn 60H (0x60) vì 5FH trỏ lại là vùng các thanh ghi

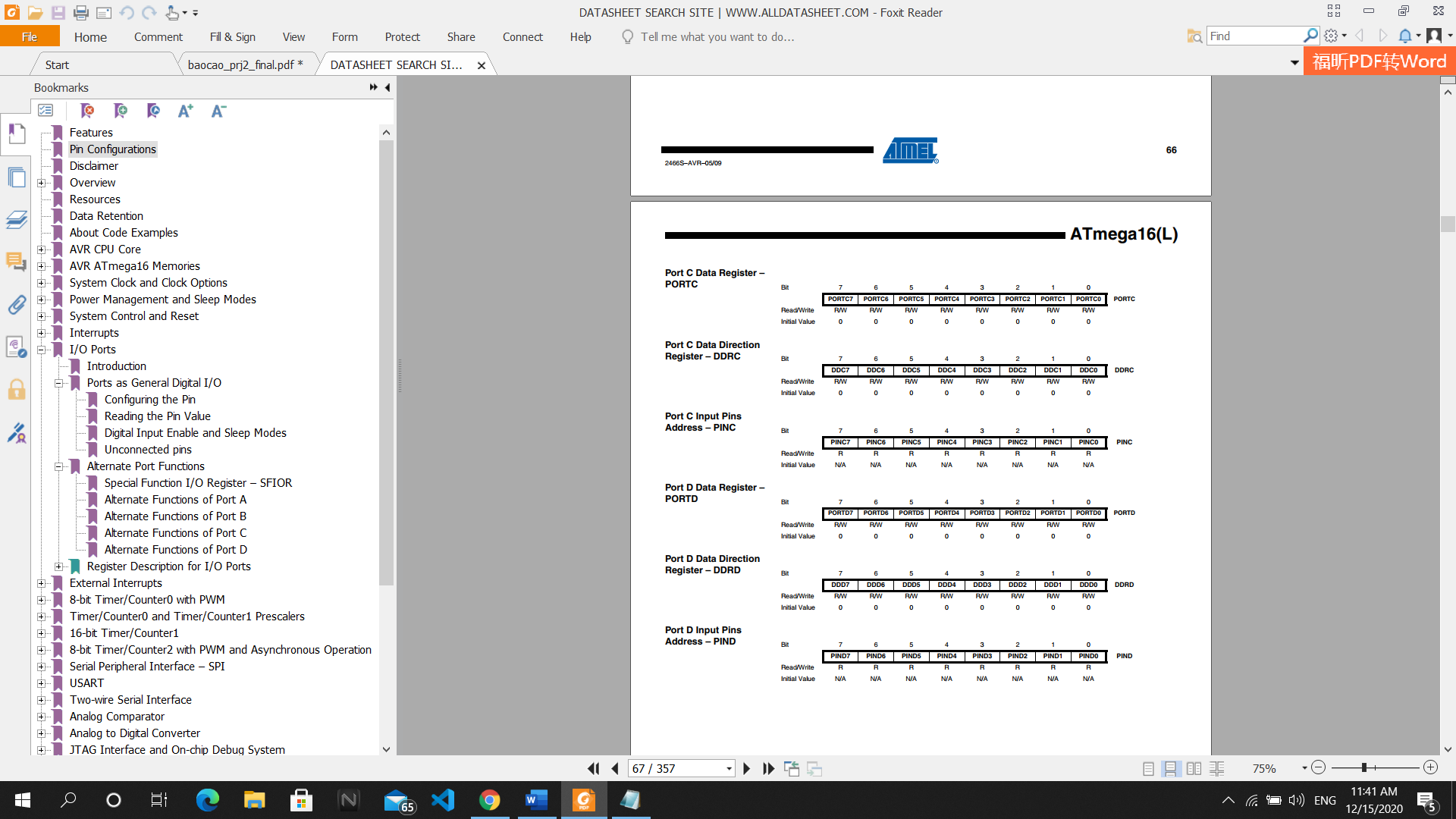
### Các cổng I/O (xuất nhập)



Hình 1.1.4. Cấu hình chân Atmega16

**Các thanh ghi mô tả các cổng I/O:**



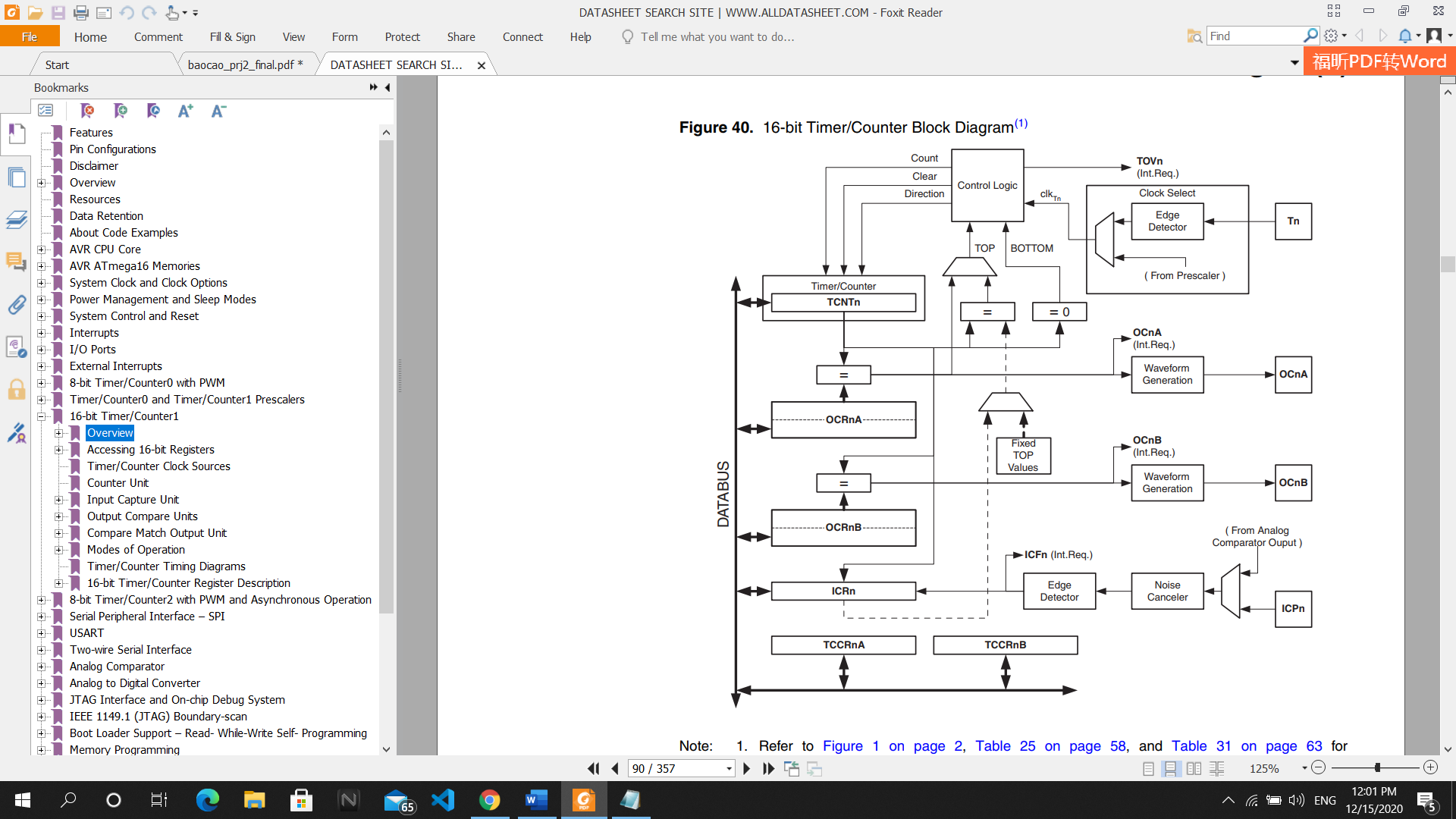


**Chức năng các chân:**

* Chân VCC (chân số 10) là VCC cấp điện áp nguồn cho vi điều khiển. Nguồn điện cấp trong khoảng +5V± 0.5.
* Chân GND (chân số 11 và 31) nối đất (hay nối Mass). Khi thiết kế cần sử dụng một mạch ổn áp để bảo vệ cho vi điều khiển, các đơn giản nhất là dùng IC ổn áp 7805
* Port A (PA): Port A gồm 8 chân (từ chân 33 đến chân 40), có chức năng đầu vào cho chuyển đổi ADC.
* Port B (PB): Port B gồm 8 chân (từ chân 1 đến chân 8), ngoài có chức năng làm các đường xuất/nhập thì còn nhiều chức năng khác như giao tiếp SPI, so sánh điện áp tương tự đầu vào, tạo xung PWM timer 0, đếm xung ngoại
* Port C (PC): Port C gồm 8 chân (từ chân 22 đến chân 29) nếu giao tiếp JTAG được kích hoạt, điện trở trên các chân PC5 (TDI), PC3 (TMS), PC2 (TCK) sẽ được kích hoạt ngay cả khi khởi động lại (reset), ngoài ra còn có giao tiếp I2C
* Port D (PD): Port D gồm 8 chân (từ chân 14 đến 21): chức năng xuất nhập
* Chân RESET (RST): ngõ vào RST ở chân 9 là ngõ vào Reset dùng để thiết lập trạng thái ban đầu cho vi điều khiển. Hệ thống sẽ được thiết lập lại các giá trị ban đầu nếu ngõ này ở mức 1 tối thiểu 2 chu kì clock.
* Chân XTAL1 và XTAL2: Hai chân này có vị trí chân là 12 và 13 được sử dụng để nhận nguồn xung clock từ bên ngoài để hoạt động, thường được ghép nối với thạch anh và các tụ để tạo nguồn xung clock ổn định.
* Chân AVCC: Nguồn cấp cho cổng A và bộ chuyển đổi ADC, chân này được nối với nguồn VCC bên ngoài, ngay cả khi bộ chuyển đổi ADC không được sử dụng. Nếu bộ chuyển đổi ADC không được sử dụng, chân AVCC nên được nối với nguồn qua bộ lọc.
* Chân AREF: AREF là chân chuẩn analog cho bộ chuyển đổi ADC.

### Bộ định thời, bộ đếm Timer/Counter

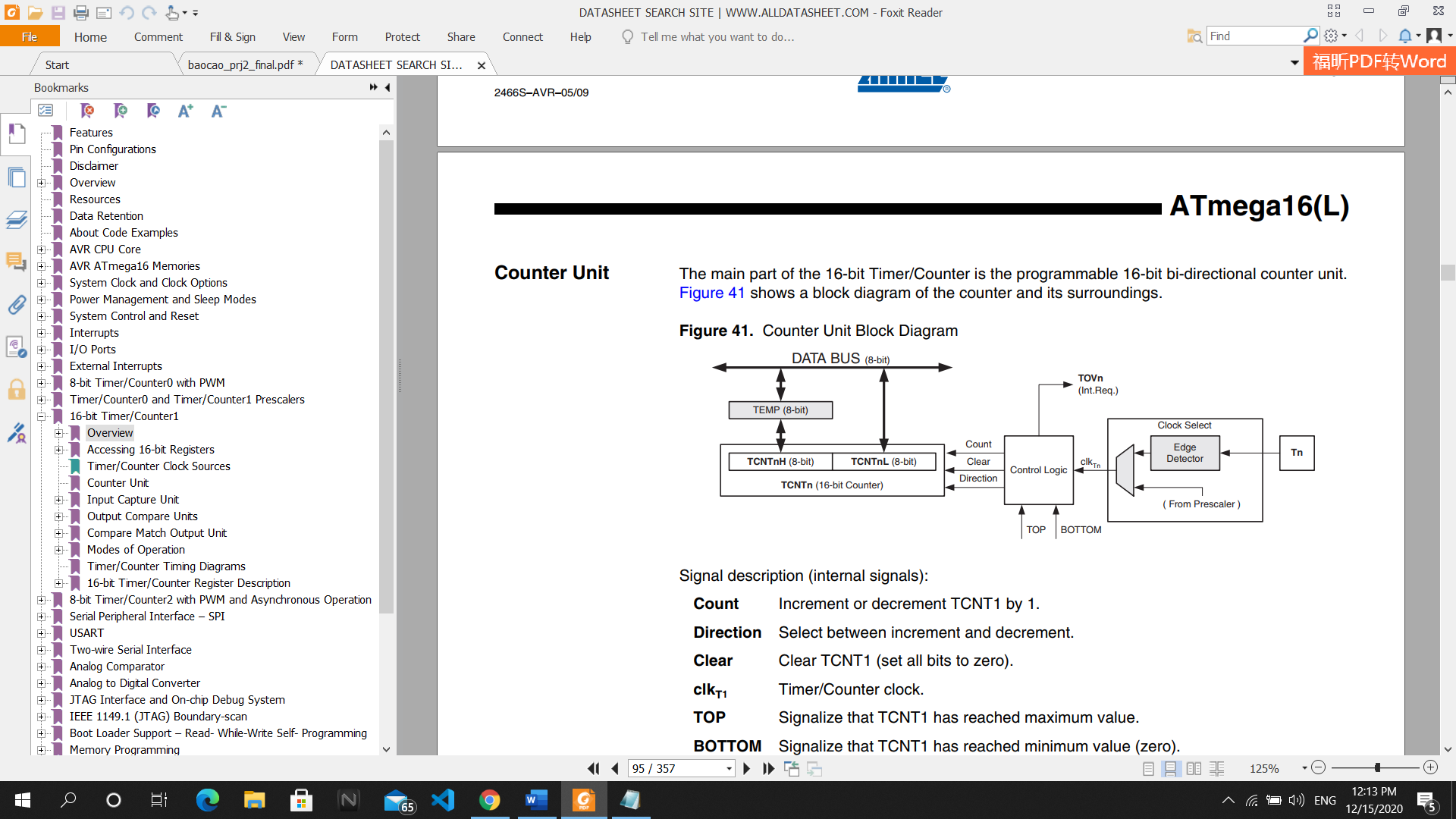
Atmega cung cấp cho lập trình viên 3 bộ timer/counter đó là: timer/counter0 8-bit, timer/counter1 16-bit, timer/counter 8-bit. Trong đề tài này chúng em sử dụng timer/counter1



Hình 1.1.5a. Cấu trúc khối timer/counter1

**Các thanh ghi chính**: Timer/Counter (TCNT1), thanh ghi so sánh :Output Compare Registers (OCR1A/B) và thanh ghi Input Capture Register (ICR1)

**Bộ đếm**: Phần chính của bộ định thời / bộ đếm 16 bit là bộ định hướng 16 bit có thể lập trình được bộ đếm



Hình 1.1.5b. Khối bộ đếm

**Count:** tăng hoặc giảm TCNT1 đi 1.

**Direction:** chọn tăng hay giảm.

**Clear:** xóa TCNT1 (đặt tất cả các bit về 0).

**clkT1:** Timer/Counter clock.

**TOP:** giá trị lớn nhất của bộ định thời/bộ đếm.

**BOTTOM:** giá trị nhỏ nhất của bộ định thời (0).

**Các chế độ hoạt động**

* Normal: đây là chế độ mặc định, bộ đếm sẽ đếm từ 0x0000 đến 0xFFFF
* CTC(compare macth mode): xóa timer ở chế độ so sánh
* Fast PWM
* Phase Correct PWM Mode
* Phase and Frequency Correct PWM Mode

## Các linh kiện được sử dụng

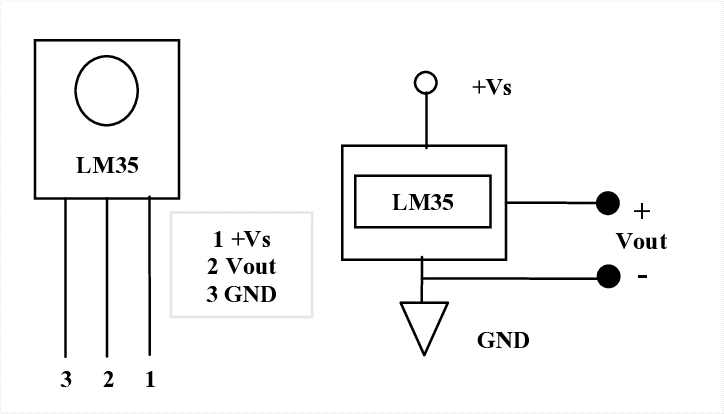
### Cảm biến nhiệt độ LM35

Cảm Biến Nhiệt Độ LM35 được sản xuất bởi hãng National Semiconductor dải đo từ 0 độ đến 100 độ C [3]. LM35 là cảm biến tiêu hao điện năng thấp sử dụng điện áp 5V. Cảm biến gồm có 3 chân, 2 chân nguồn, 1 chân tín hiệu ra dạng Analog.

Chân dữ liệu của LM35 là chân ngõ ra điện áp dạng tuyến tính. Chân số 2 cảm biến xuất ra cứ 1mV = 0.1°C (10mV = 1°C). Để lấy dữ liệu ở dạng °C chỉ cần lấy điện áp trên chân OUT đem chia cho 10.

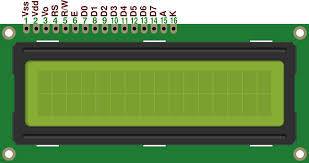
Chân 1 cấp điện áp 5V, chân 3 cấp GND, chân 2 là chân OUTPUT dữ liệu dạng điện áp. Các thông số kỹ thuật cơ bản của LM35 như sau [3]:

* Điện áp đầu vào từ 4V đến 30V.
* Điện áp ra: -1V đến 6V.
* Công suất tiêu thụ là 60uA.
* Độ phân giải điện áp đầu ra là 10mV/oC.
* Độ chính xác cao ở 25 C là 0.5 C.
* Trở kháng đầu ra thấp 0.1 cho 1mA tải.
* Độ chính xác thực tế: 1/4°C ở nhiệt độ phòng và 3/4°C ngoài khoảng -55°C tới 150°C



Hình 1.2.1 Sơ đồ chân LM35

### Màn hình hiển thị LCD

Màn hình text LCD1602 xanh lá sử dụng driver HD44780, có khả năng hiển thị 2 dòng với mỗi dòng 16 ký tự, màn hình có độ bền cao, rất phổ biến, nhiều code mẫu và dễ sử dụng thích hợp cho những người mới học và làm dự án.

Hình 1.2.2. LCD 16x2

Các thông số kỹ thuật chính của LCD1602 như sau:

* Điện áp hoạt động là 5V.
* Kích thước: 80 x 36 x 12.5 mm
* Chữ đen, nền xanh lá
* Khoảng cách giữa hai chân kết nối là 0.1 inch tiện dụng khi kết nối với Breadboard.
* Tên các chân được ghi ở mặt sau của màn hình LCD hổ trợ việc kết nối, đi dây điện.
* Có đèn led nền, có thể dùng biến trở hoặc PWM điều chình độ sáng để sử dụng ít điện năng hơn.
* Có thể được điều khiển với 6 dây tín hiệu
* Có bộ ký tự được xây dựng hổ trợ tiếng Anh và tiếng Nhật, xem thêm HD44780 datasheet để biết thêm chi tiết.

Bảng 1.1. Các chân của LCD1602

|  |  |  |  |
| --- | --- | --- | --- |
| **Chân** | **Ký hiệu** | **Mô tả** | **Giá trị** |
| 1 | VSS | GND | 0V |
| 2 | VCC |  | 5V |
| 3 | V0 | Độ tương phản |  |
| 4 | RS | Lựa chọn thanh ghi | RS=0 (mức thấp) chọn thanh ghi lệnh  RS=1 (mức cao) chọn thanh ghi dữ liệu |
| 5 | R/W | Chọn thanh ghi đọc/viết dữ liệu | R/W=0 thanh ghi viết  R/W=1 thanh ghi đọc |
| 6 | E | Enable |  |
| 7 | DB0 | Chân truyền dữ liệu | 8 bit: DB0DB7 |
| 8 | DB1 |
| 9 | DB2 |
| 10 | DB3 |
| 11 | DB4 |
| 12 | DB5 |
| 13 | DB6 |
| 14 | DB7 |
| 15 | A | Cực dương led nền | 0V đến 5V |
| 16 | K | Cực âm led nền | 0V |

## Mạch Kit cho vi điều khiển dòng AVR

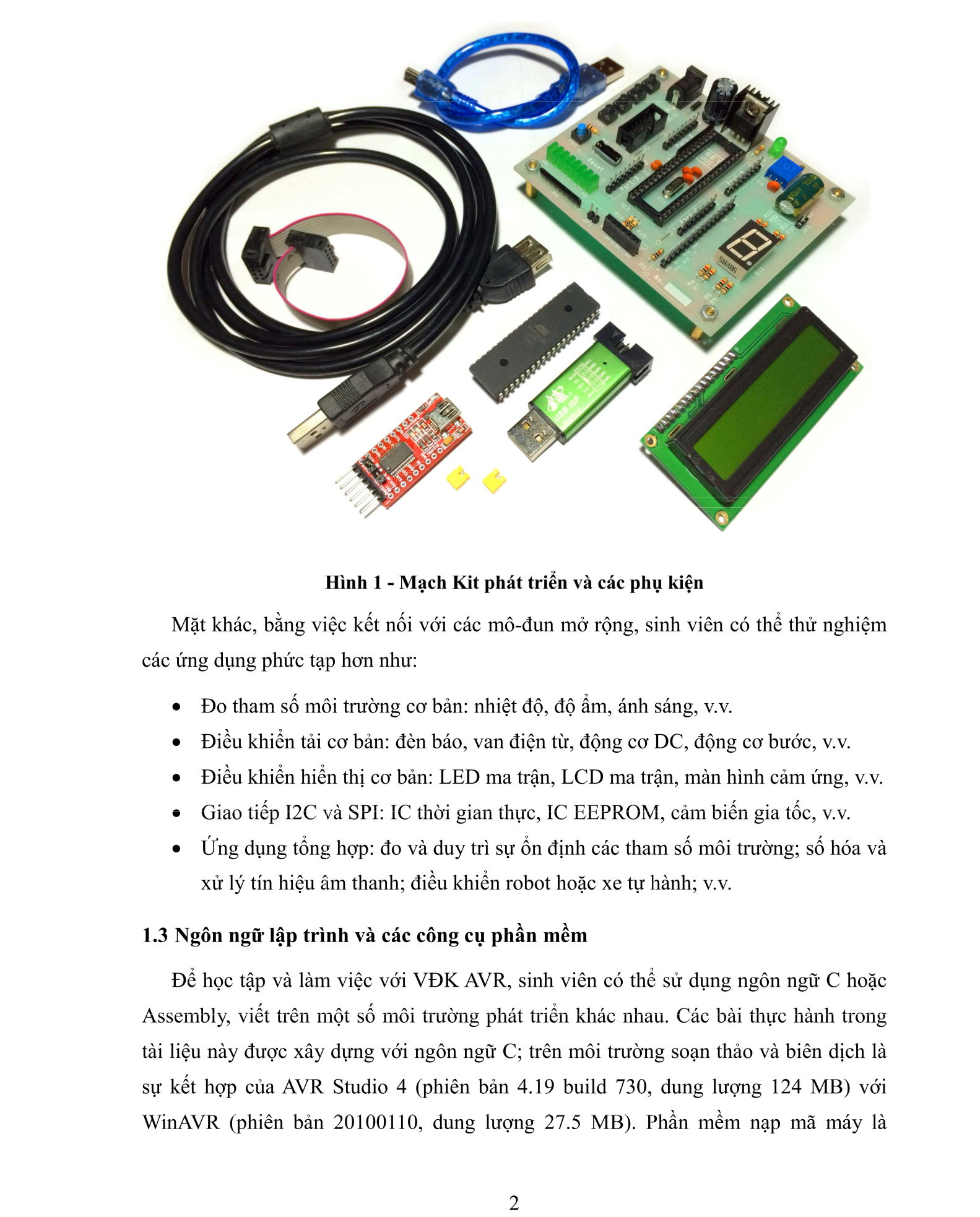
* + 1. ***Tổng quan về mạch Kit AVR***

AVR là một dòng vi điều khiển 8 bit khá mạnh và thông dụng tại thị trường Việt Nam. Với tốc độ xung nhịp tới 16Mhz, bộ nhớ chương trình tối đa tới 256 kB, và rất nhiều chức năng ngoại vi tích hợp sẵn, vi điều khiển họ AVR có thể đáp ứng tốt cho nhiều ứng dụng trong thực tế, từ đơn giản đến phức tạp.

Với bộ Kit này có thể thử nghiệm các ứng dụng cơ bản như:

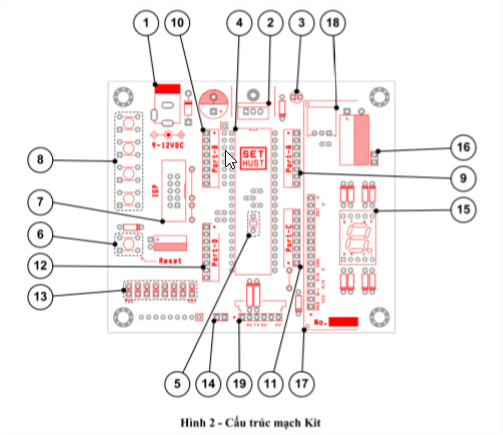
* Điều khiển cổng ra số, với LED đơn và LED 7 thanh
* Đọc trạng thái logic đầu vào số, từ bàn phím và giác cắm mở rộng
* Đo điện áp tương tự, với biến trở vi chỉnh và bộ ADC 10-bit
* Điều khiển màn hình tinh thể lỏng, với màn hình LCD dạng text
* Giao tiếp với máy tính qua chuẩn UART – USB
* Thử nghiệm các ngắt ngoài, thử khả năng điều khiển chế độ rộng xung

Nhiều ứng dụng điều khiển các chức năng tích hợp sẵn trong vi điều khiển như: vận hành các bộ định thời (Timer) và bộ đếm (Counter), đọc ghi EEPROM, lập trình các ngắt chương trình, thiết lập Watchdog, v.v.

****

Hình 1.3.1. Mạch Kit phát triền và các phụ kiện đi kèm

* + 1. ***Cấu trúc mạch Kit***



Hình 1.3.2 Cấu trúc mạch Kit

Hình 1.6 mô tả cấu trúc mạch Kit phát triển với chi tiết từng thành phần được ghi rõ tại Bảng 1.2.

Bảng 1.2. Các linh kiện chính và chức năng tương ứng

|  |  |  |
| --- | --- | --- |
| **STT** | **Tên linh kiện** | **Chức năng** |
| 1 | Giắc cắm nguồn | Nhận nguồn điện 9-12 VDC cấp cho mạch Kit |
| 2 | IC ổn áp 7805 | Hạ 9-12 VDC xuống 5 VDC và giữ ổn định mức điện áp này. |
| 3 | LED báo nguồn | Báo nguồn |
| 4 | Vi điều khiển họ AVR | Điều khiển hoạt động của toàn mạch theo mã nguồn do người dùng lập trình và nạp xuống |
| 5 | Thạch anh | Quyết định tần số xung nhịp cấp cho vi điều khiển |
| 6 | Nút ấn Reset | Khởi động lại vi điều khiển |
| 7 | Giắc ISP | Kết nối mạch nạp để nạp mã nguồn cho vi điều khiển |
| 8 | Nhóm 4 phím ấn | Nhận lệnh điều khiển từ người sử dụng |
| 9 | Giắc cắm 8 chân | Nối tới 8 chân vào\ra đa năng (ứng với Port-A) của vi điều khiển |
| 10 | Giắc cắm 8 chân | Nối tới 8 chân vào\ra đa năng (ứng với Port-B) của vi điều khiển |
| 11 | Giắc cắm 8 chân | Nối tới 8 chân vào\ra đa năng (ứng với Port-C) của vi điều khiển |
| 12 | Giắc cắm 8 chân | Nối tới 8 chân vào\ra đa năng (ứng với Port-D) của vi điều khiển |
| 13 | Dãy LED đơn | Báo trạng thái logic của 8 chân ở Port-D (sáng-0,tắt-1) |
| 14 | Jumper dãy LED đơn | Cho phép hoặc vô hiệu hóa Led đơn |
| 15 | LED 7 thanh | Hiển thị số 0-9 và một vài kí tự do người dùng định nghĩa |
| 16 | Jumper LED 7 thanh | Cho phép hoặc vô hiệu hóa LED 7 thanh |
| 17 | Jack cắm LCD | Kết nối màn hình LCD dạng text (1602) |
| 18 | Biến trở vi chỉnh | Điều chỉnh trơn và liên tục từ 0 -5 VDC, mức điện áp đầu vào ADC0 của bộ ADC(chân PA0) |
| 19 | Giắc UART-USB | Kết nối mô-đun chuyển đổi UART-USB |

* + 1. ***Các thông số kỹ thuật chính***

Các thông số kỹ thuật của mạch Kit gồm có:

* Điện áp nguồn
  + Tiêu chuẩn : 9-12 VDC
  + Giới hạn : 7-12 VDC
* Dòng điện tiêu thụ:
  + Khi không có mô-đun mở rộng, toàn bộ LED chỉ thị I/O tắt: 18mA
  + Khi có LCD và mô-đun USB, các LED chỉ thị I/O bị vô hiệu hóa : 22mA
  + Khi có LCD và mô-đun USB, toàn bộ LED chỉ thị I/O sáng: 80mA
* Mạch có khả năng tự bảo vệ khi bị lắp ngược cực tính nguồn
* Mức logic các cổng I/O: TTL(5V)
* Loại vi điều khiển được hỗ trợ: ATmega16,ATmega32, và tương đương
* Cổng I/O mở rộng: 4 giắc cắm(loại 8 chân) ứng với 4 Port(8bit mỗi Port)
* Hỗ trợ mô-đun USB: UART-USB hay COM-USB(mức 5VDC)
* Xung nhịp tích hợp sẵn: thạch anh 8Mhz

Để làm việc với vi điều khiển AVR, có thể sử dụng ngôn ngữ C hoặc Assembly, viết trên một số môi trường phát triển khác nhau. Đề tài này được xây dựng bằng ngôn ngữ C, trên môi trường soạn thảo và biên dịch là sự kết hợp của AVR Studio 4 (phiên bản 4.19 build 730, dung lượng 124MB) với WinAVR (phiên bản 20100110, dung lượng 27.5 MB). Phần mềm nạp mã máy là PROGISP (phiên bản 1.72, dung lượng khoảng 3-4MB). Phần mềm nhận dữ liệu từ cổng USB hay COM ảo là Terminal. Phần mềm mô phỏng mạch là Proteus 8 Professional.

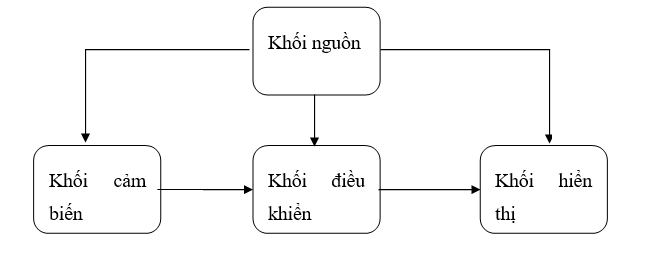
* + 1. ***Sơ đồ nguyên lý mạch Kit***

# 

Hình 1.3.4 Sơ đồ nguyên lý mạch Kit

# CHƯƠNG 2: THIẾT KẾ MẠCH

2.1. Sơ đồ khối



Hình 2. 1a Sơ đồ khối

Sơ đồ khối gồm 4 khối chính:

* khối nguồn: cung cấp nguồn đầu vào cho mạch
* khối cảm biến: lấy thông số nhiệt độ đo đầu vào từ bên ngoài
* khối vi điều khiển: vi điều khiển atmega xử lí đầu vào, tính toán các thông số sau đó gửi cho khối hiển thị
* khối hiển thị: LCD hiển thị các thông số

A diagram of a computer

Description automatically generated

Hình 2.1b Sơ đồ nguyên lí đo nhiệt độ độ ẩm sử dụng Kit AVR

2.2. Mạch in Kit AVR

A blue circuit board with green and blue lines

Description automatically generated

Hình 2. 2 Đi dây mạch in Kit AVR

# CHƯƠNG 3: KẾT QUẢ ThỰC HIỆN

3.1. Kết quả thực hiện đề tài

Mạch có thể đo và hiển thị nhiệt độ trên màn LCD

A circuit board with a red light

Description automatically generated

3.2. Yêu cầu phi chức năng

Mạch chạy cần ổn định, đáp ứng nhanh các thao tác của người sử dụng, ngoài ra mạch cần phải gọn gàng ít kết nối dây nhiều đảm bảo tính thẩm mỹ cho mạch

3.3. Hướng phát triển

Đề tài “Thiết kế mạch đo nhiệt độ sử dụng Kit AVR” là một đề tài không lớn về mặt quy mô, nhưng có khả năng phát triển và mở rộng rất lớn. Cụ thể:

* Kit AVR có thể thực hiện các ứng dụng phức tạp hơn như:
* Đo nhiệt độ thêm cả độ ẩm môi trường và cảnh báo vượt độ ẩm nhiệt độ.
* Điều khiển tái cơ bản: đèn báo, van điện tử, động cơ DC.
* Điều khiển hiển thị : Led ma trận, LCD ma trận.

Kit AVR có ứng dụng tổng hợp: đo và duy trì sự ổn định các tham số môi trường, số hóa và xử lí tín hiệu âm thanh, điều khiển robot, xe tự hành.

# KẾT LUẬN

Trong quá trình nghiên cứu, tìm hiểu cũng như thực hiện đề tài này, nhóm em đã cố gắng tiếp thu nhiều kiến thức mới liên quan đến lập trình cho vi xử lý Atmega16, cải thiện khả năng sử dụng phần mềm Proteus và Altium. Do đây là một đề tài phổ biến nên tài liệu cô cung cấp và các mô-đun liên quan được viết rất dễ hiểu nên có phần thuận lợi cho em trong quá trình thực hiện. Mặc dù đã có nhiều cố gắng ôn lại các kiến thức đã được học từ trước, kết hợp tra cứu các tài liệu chuyên ngành nhưng do hạn chế về thời gian, khả năng và kinh nghiệm nên mạch kết quả chưa được chỉnh chu và hoàn thiện như mong đợi. Trong quá trình thực hiện em còn gặp một số lỗi về phần mềm nên còn tốn nhiều thời gian không tránh khỏi những thiếu sót nhất định. Tuy vậy, em đã rất nổ lực và cố gắng hoàn thành đề tài đúng thời hạn và đáp ứng đa phần các yêu cầu kĩ thuật đã đề ra. Mong cô thông cảm.

Để hoàn thành Đồ án II này, một lần nữa nhóm em xin chân thành cảm ơn thầy Tào Văn Cường đã hướng dẫn, cung cấp tài liệu đầy đủ và chi tiết cho nhóm. Chúng em xin chân thành cảm ơn.

# DANH MỤC TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | Atmel, 8-bit AVR Microcontroller with 16K Bytes In-System Programmable Flash. |
| [2] | "C (programming language)," 10 July 2020. [Online]. Available: https://en.wikipedia.org/wiki/C\_(programming\_language). |
| [3] | "LM35 Precision Centigrade Temperature Sensors," Texas Instruments, 1999. |
| [4] | "LCD1602 Module," Sunflower, March 2018. [Online]. Available: http://wiki.sunfounder.cc/index.php?title=LCD1602\_Module. |

# phụ lục

### Phụ lục 1: Source code

**LCD\_16x2\_C\_file.c**

#include "LCD\_16x2\_H\_file.h" /\* Include LCD header file \*/

void LCD\_Command (char cmd) /\* LCD command write function \*/

{

LCD\_Data\_Port = cmd; /\* Write command data to LCD data port \*/

LCD\_Command\_Port &= ~((1<<RS)|(1<<RW)); /\* Make RS LOW (command reg.), RW LOW (Write) \*/

LCD\_Command\_Port |= (1<<EN); /\* High to Low transition on EN (Enable) \*/

\_delay\_us(1);

LCD\_Command\_Port &= ~(1<<EN);

\_delay\_ms(3); /\* Wait little bit \*/

}

void LCD\_Char (char char\_data) /\* LCD data write function \*/

{

LCD\_Data\_Port = char\_data; /\* Write data to LCD data port \*/

LCD\_Command\_Port &= ~(1<<RW); /\* Make RW LOW (Write) \*/

LCD\_Command\_Port |= (1<<EN)|(1<<RS); /\* Make RS HIGH (data reg.) and High to Low transition on EN (Enable) \*/

\_delay\_us(1);

LCD\_Command\_Port &= ~(1<<EN);

\_delay\_ms(1); /\* Wait little bit \*/

}

void LCD\_Init (void) /\* LCD Initialize function \*/

{

LCD\_Command\_Dir |= (1<<RS)|(1<<RW)|(1<<EN); /\* Make LCD command port direction as o/p \*/

LCD\_Data\_Dir = 0xFF; /\* Make LCD data port direction as o/p \*/

\_delay\_ms(20); /\* LCD power up time to get things ready, it should always >15ms \*/

LCD\_Command (0x38); /\* Initialize 16X2 LCD in 8bit mode \*/

LCD\_Command (0x0C); /\* Display ON, Cursor OFF command \*/

LCD\_Command (0x06); /\* Auto Increment cursor \*/

LCD\_Command (0x01); /\* Clear LCD command \*/

LCD\_Command (0x80); /\* 8 is for first line and 0 is for 0th position \*/

}

void LCD\_String (char \*str) /\* Send string to LCD function \*/

{

int i;

for(i=0;str[i]!=0;i++) /\* Send each char of string till the NULL \*/

{

LCD\_Char (str[i]); /\* Call LCD data write \*/

}

}

void LCD\_String\_xy (char row, char pos, char \*str) /\* Send string to LCD function \*/

{

if (row == 1)

LCD\_Command((pos & 0x0F)|0x80); /\* Command of first row and required position<16 \*/

else if (row == 2)

LCD\_Command((pos & 0x0F)|0xC0); /\* Command of Second row and required position<16 \*/

LCD\_String(str); /\* Call LCD string function \*/

}

void LCD\_Clear(void)

{

LCD\_Command(0x01); /\* clear display \*/

LCD\_Command(0x80);

}

**LCD\_16x2\_C\_file.h**

#ifndef LCD\_16x2\_H\_H\_ /\* Define library H file if not defined \*/

#define LCD\_16x2\_H\_H\_

#define F\_CPU 8000000UL /\* Define CPU Frequency e.g. here its 8MHz \*/

#include <avr/io.h> /\* Include AVR std. library file \*/

#include <util/delay.h> /\* Include Delay header file \*/

#define LCD\_Data\_Dir DDRC /\* Define LCD data port direction \*/

#define LCD\_Command\_Dir DDRD /\* Define LCD command port direction register \*/

#define LCD\_Data\_Port PORTC /\* Define LCD data port \*/

#define LCD\_Command\_Port PORTD /\* Define LCD data port \*/

#define EN PD7 /\* Define Enable signal pin \*/

#define RW PD5 /\* Define Read/Write signal pin \*/

#define RS PD6 /\* Define Register Select (data reg./command reg.) signal pin \*/

void LCD\_Command (char); /\* LCD command write function \*/

void LCD\_Char (char); /\* LCD data write function \*/

void LCD\_Init (void); /\* LCD Initialize function \*/

void LCD\_String (char\*); /\* Send string to LCD function \*/

void LCD\_String\_xy (char,char,char\*); /\* Send row, position and string to LCD function \*/

void LCD\_Clear(void); /\* Clear LCD\*/

#endif /\* LCD\_16x2\_H\_FILE\_H\_ \*/

**main.c**

#define F\_CPU 8000000UL

#include <avr/io.h>

#include <util/delay.h>

#include <stdio.h>

#include "LCD\_16x2\_H\_file.h"

#define DEGREE\_SYMBOL 0xDF

void ADC\_Init(){

ADMUX = (1 << REFS0); /\* Vref: Avcc, ADC channel: 0 \*/

ADCSRA = (1 << ADEN) | (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0); /\* Enable ADC, with freq/128 \*/

}

int ADC\_Read(int channel)

{

ADMUX &= 0xF0; /\* Clear the previous channel selection \*/

ADMUX |= channel & 0x0F; /\* set input channel to read \*/

ADCSRA |= (1<<ADSC); /\* Start ADC conversion \*/

while (ADCSRA & (1 << ADSC)); /\* Wait until end of conversion by polling ADC interrupt flag \*/

return ADCW; /\* Return ADC word \*/

}

int main()

{

char Temperature[10];

int Temp;

LCD\_Init(); /\* initialize 16x2 LCD\*/

ADC\_Init(); /\* initialize ADC\*/

LCD\_Clear();

LCD\_String\_xy(1, 0, "Temperature:");

while(1)

{

\_delay\_ms(1000);

Temp = (ADC\_Read(7) \* 5.0 / 1024.0) \* 100.0; /\* convert ADC value to temperature in Celsius \*/

sprintf(Temperature, "%d%cC", 70-Temp, DEGREE\_SYMBOL); /\* format temperature value as a string \*/

LCD\_String\_xy(2, 0, Temperature);

\_delay\_ms(500);

}

return 0;

}