

**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**VIỆN ĐIỆN TỬ-VIỄN THÔNG**

**BÁO CÁO ĐỒ ÁN II**

*Đề tài:* **THIẾT KẾ HỆ THỐNG ĐIỀU KHIỂN ĐỘNG CƠ MỘT CHIỀU**

*Giảng viên hướng dẫn: Vũ Hồng Vinh*

|  |  |
| --- | --- |
| **Họ tên sinh viên** | **MSSV** |
| 1. Mai Văn Lâm | 20182625 |
| 1. Trần Xuân Mạnh | 20182679 |
| 1. Lê Anh Thắng | 20172812 |
| 1. Lê Văn Phương | 20172760 |
|  |  |
|  |  |

*Hà Nội, 2022*

**BẢNG PHÂN CÔNG NHIỆM VỤ**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TT | Họ và tên | Nhiệm vụ | Tự đánh gía | Ghi chú |
| 1 | Mai Văn Lâm | Tìm hiểu code, mô phỏng Proteus |  |  |
| 2 | Lê Anh Thắng | Tìm hiểu lý thuyết |  |  |
| 3 | Lê Văn Phương | Tìm hiểu lý thuyết |  |  |
| 4 | Trần Xuân Mạnh | Tổng hợp và hoàn thiện báo cáo |  |  |

**LỜI MỞ ĐẦU**

Ngày nay khi khoa học công nghệ đang phát triển mạnh mẽ. Động cơ một chiều có nhiều ứng dụng trong lĩnh vực điều khiển và sản xuất nhất là trong công nghiệp. Trong đó nó đòi hỏi là động cơ phải có nhiều cấp tốc độ có thể tăng giảm dễ dàng, độ ổn định và tốc độ cao nên động cơ một chiều đã được sử dụng khá phổ biến như: truyền động cho một số máy như máy nghiền, máy nâng vận chuyển, điều khiển băng tải, điều khiển các robot…Với sự ra đời và phát triển của vi xử lý, vi điều khiển thì vấn đề điều khiển động cơ DC không còn là vấn đề khó khăn nữa. Động cơ có thể điều khiển với nhiều cấp tốc độ khác nhau và điều khiển dừng, đảo chiểu, nhanh chậm dễ dàng được. Vì vậy đây cũng chính là ý tưởng để nhóm thực hiện đề tài thiết kế “Mạch điều khiển Động Cơ DC”.

Chúng em xin được gửi lời cảm ơn chân thành đến các Thầy cô trong Viện Điện Tử - Viễn Thông đã giảng dạy và truyền đạt kiến thức chuyên ngành cho chúng em trong thời gian vừa qua. Đặc biệt người thực hiện xin gửi lời cảm ơn sâu sắc tới thầy Vũ Hồng Vinh vì sự tận tình hướng dẫn cũng như đã tạo những điều kiện thuận lợi nhất cho chúng em để có thể thực hiện và hoàn thành tốt đề tài này. Chúng em cũng không quên cảm ơn các bạn trong lớp đã trao đổi, góp ý để người thực hiện hoàn thành đề tài này một cách tốt đẹp và đúng thời gian. Mặc dù đã có nhiều cố gắng và nỗ lực thực hiện, nhưng do kiến thức cũng như khả năng bản thân còn nhiều hạn chế nên trong quá trình thực hiện đề tài không thể tránh khỏi những sai phạm, thiếu sót…Rất mong nhận được sự góp ý, chỉ dẫn từ thầy cô và các bạn sinh viên.

Nhóm chúng em xin chân thành cảm ơn!

Hà Nội, năm 2022

*Sinh viên thực hiện*

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |

Mục lục

[DANH MỤC HÌNH VẼ i](#_Toc99399347)

[PHẦN MỞ ĐẦU ii](#_Toc99399348)

[CHƯƠNG 1: CƠ SỞ LÍ THUYẾT TỔNG QUAN 1](#_Toc99399349)

[1.1. Tìm hiểu vi điều khiển Atmega16 1](#_Toc99399350)

[1.1.1. Tổng quan về Atmega16 1](#_Toc99399351)

[1.1.2. Cấu trúc Atmega16 3](#_Toc99399352)

[1.1.3. ARV CPU Core 4](#_Toc99399353)

[1.1.4. Các cổng I/O (xuất nhập) 7](#_Toc99399354)

[1.1.5. Bộ định thời, bộ đếm Timer/Counter 9](#_Toc99399355)

[1.2. PWM(Pulse Width Modulation)- điều chế độ rộng xung 11](#_Toc99399356)

[1.2.1. Tổng quan về PWM 11](#_Toc99399357)

[1.2.2. Điều chế xung PWM bằng chế độ Fast PWM của vi điều khiển Atmega16 12](#_Toc99399358)

[1.3. Các phần mềm sử dụng 14](#_Toc99399359)

[1.3.1. Trình biên dịch CodevisionAVR 14](#_Toc99399360)

[1.3.2. Phần mềm mô phỏng Proteus 15](#_Toc99399361)

[1.3.3. Phần mềm vẽ mạch Altium 15](#_Toc99399362)

[1.3.4. Phần mềm nạp code ProgISP 16](#_Toc99399363)

[1.4. Các linh kiện được sử dụng 16](#_Toc99399364)

[1.4.1. Module điều khiển động cơ L298N 16](#_Toc99399365)

[1.4.2. Động cơ DC 12V 18](#_Toc99399366)

[1.4.3. Màn hình hiển thị LCD 18](#_Toc99399367)

[CHƯƠNG 2: THIẾT KẾ MẠCH 19](#_Toc99399368)

[2.1. Sơ đồ khối 19](#_Toc99399369)

[2.2. Sơ đồ nguyên lí mạch kit trên Altium 20](#_Toc99399370)

[2.3. Mạch in 21](#_Toc99399371)

[CHƯƠNG 3: YÊU CẦU MẠCH 22](#_Toc99399372)

[3.1. Yêu cầu chức năng 22](#_Toc99399373)

[3.2. Yêu cầu phi chức năng 22](#_Toc99399374)

[3.3. Ứng dụng 22](#_Toc99399375)

[CHƯƠNG 4: KẾT QUẢ THỰC HIỆN 23](#_Toc99399376)

[4.1. Kết quả mô phỏng 23](#_Toc99399377)

[4.2. Kết quả thực tế trên mạch 23](#_Toc99399378)

[KẾT LUẬN 24](#_Toc99399379)

[DANH MỤC TÀI LIỆU THAM KHẢO 25](#_Toc99399380)

[PHỤ LỤC 26](#_Toc99399381)

[Phụ lục 1: Source code 26](#_Toc99399382)

# DANH MỤC HÌNH VẼ

[Hình 1. 1 Cấu trúc các khối Atmega16 3](#_Toc60003919)

[Hình 1. 2 Sơ đồ khối AVR MCU 4](#_Toc60003920)

[Hình 1. 3 Các thanh ghi chức năng chung 5](#_Toc60003921)

[Hình 1. 4 Cấu hình chân Atmega16 7](#_Toc60003922)

[Hình 1. 5 Cấu trúc khối timer/counter1 9](#_Toc60003923)

[Hình 1. 6 Khối bộ đếm 10](#_Toc60003924)

[Hình 1. 7 Đồ thị dạng xung điều chế PWM 11](#_Toc60003925)

[Hình 1. 8 Các chế độ hoạt động của timer/counter1 12](#_Toc60003926)

[Hình 1. 9 Mô tả hoạt động Fast PWM 12](#_Toc60003927)

[Hình 1. 10 Giao diện của CodevisionAVR 14](#_Toc60003928)

[Hình 1. 11 Giao diện Proteus 15](#_Toc60003929)

[Hình 1. 12 Giao diện Altium Designer 18 15](#_Toc60003930)

[Hình 1. 13 Giao diện ProgISP 1.72 16](#_Toc60003931)

[Hình 1. 14 Module L298N 17](#_Toc60003932)

[Hình 1. 15 Động cơ 365 12VDC 18](#_Toc60003933)

[Hình 1. 16 LCD 16x2 18](#_Toc60003934)

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

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

[Hình 2. 3 Mạch in PCB 20](#_Toc60003969)

[Hình 4. 1 Kết quả mô phỏng bằng phần mềm Proteus 22](#_Toc60003987)

[Hình 4. 2 Kết quả thực tế 22](#_Toc60003988)

# 

# PHẦN MỞ ĐẦU

Đối với sinh viên học ngành điện tử viễn thông, việc lập trình, thiết kế và làm mạch là một kỹ năng cơ bản không thể thiếu trong trong ngành nghề của mình. Học phần đồ án thiết kế I là cơ hội để mỗi sinh viên có thể luyện tập, mài dũa kỹ năng đó.

Sau thời gian học tập 3 năm tại trường, được sự chỉ bảo hướng dẫn của các thầy cô trong viên Điện tử Viễn thông, chúng em đã tích lũy được vốn kiến thức nhất định. Được sự đồng ý của thầy Dương Thanh Phương, chúng em được giao đề tài nghiên cứu đồ án thiết kế II: *“Điều khiển động cơ 1 chiều”*

Đồ án thiết kế II của em gồm 4 chương:

Chương 1: Trình bày cơ sở lí thuyết Atmega16, các linh kiện sử dụng, các phần mềm sử dụng

Chương 2: Thiết kế mạch

Chương 3: Yêu cầu mạch

Chương 4: Kết quả thực hiện

Trong quá trình thực hiện đề tài, do còn nhiều hạn chế về mặt thời gian, kiến thức cũng như công cụ và kinh nghiệm nên đề tài vẫn còn những thiếu sót nhất định, mong được thầy cô và các bạn đánh giá, góp ý một cách khách quan để chúng em có thể rút ra kinh nghiệm và hoàn thiện đề tài hơn nữa. Để hoàn thành bài báo cáo này, chúng em xin chân thành gửi lời cảm ơn tới sự quan tâm và hướng dẫn tận tình của Thầy.

# 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.

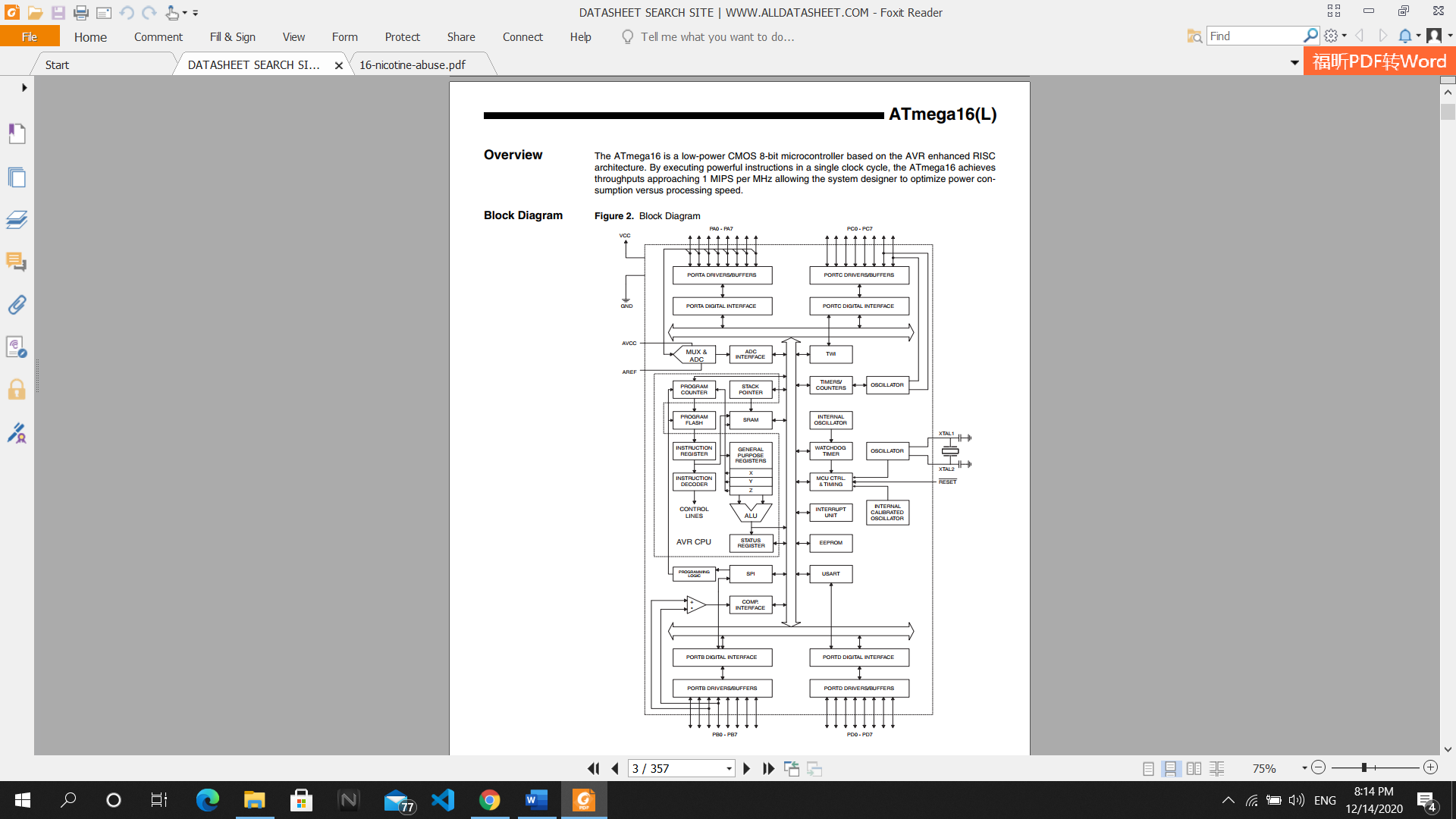
Các đặc điểm của chip Atmega được tóm tắt như sau:

* 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 atmega16

### 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.

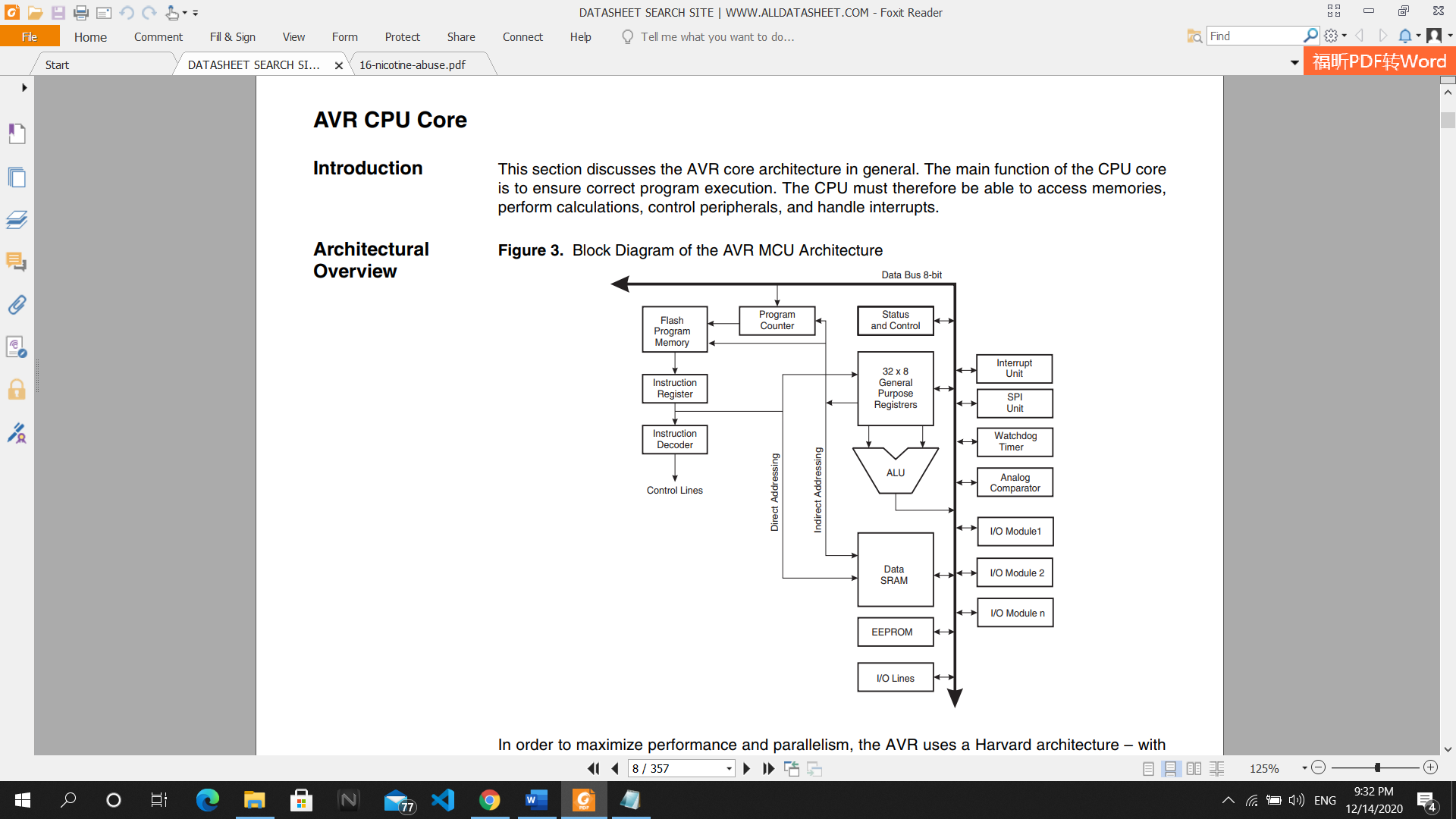


Hình 1. 1 Cấu trúc các khối Atmega16

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.



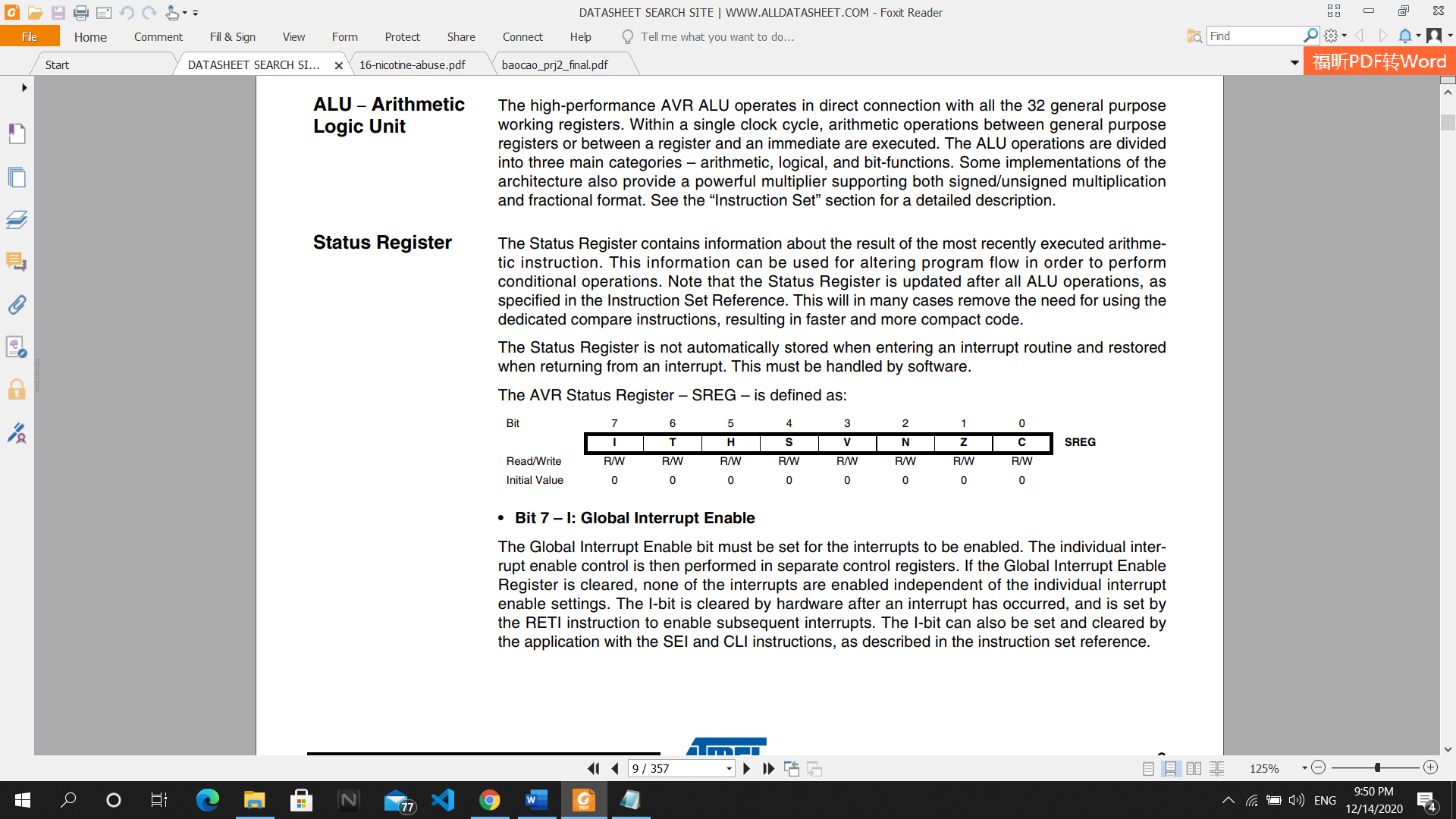
Hình 1. Sơ đồ khối AVR MCU

**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.

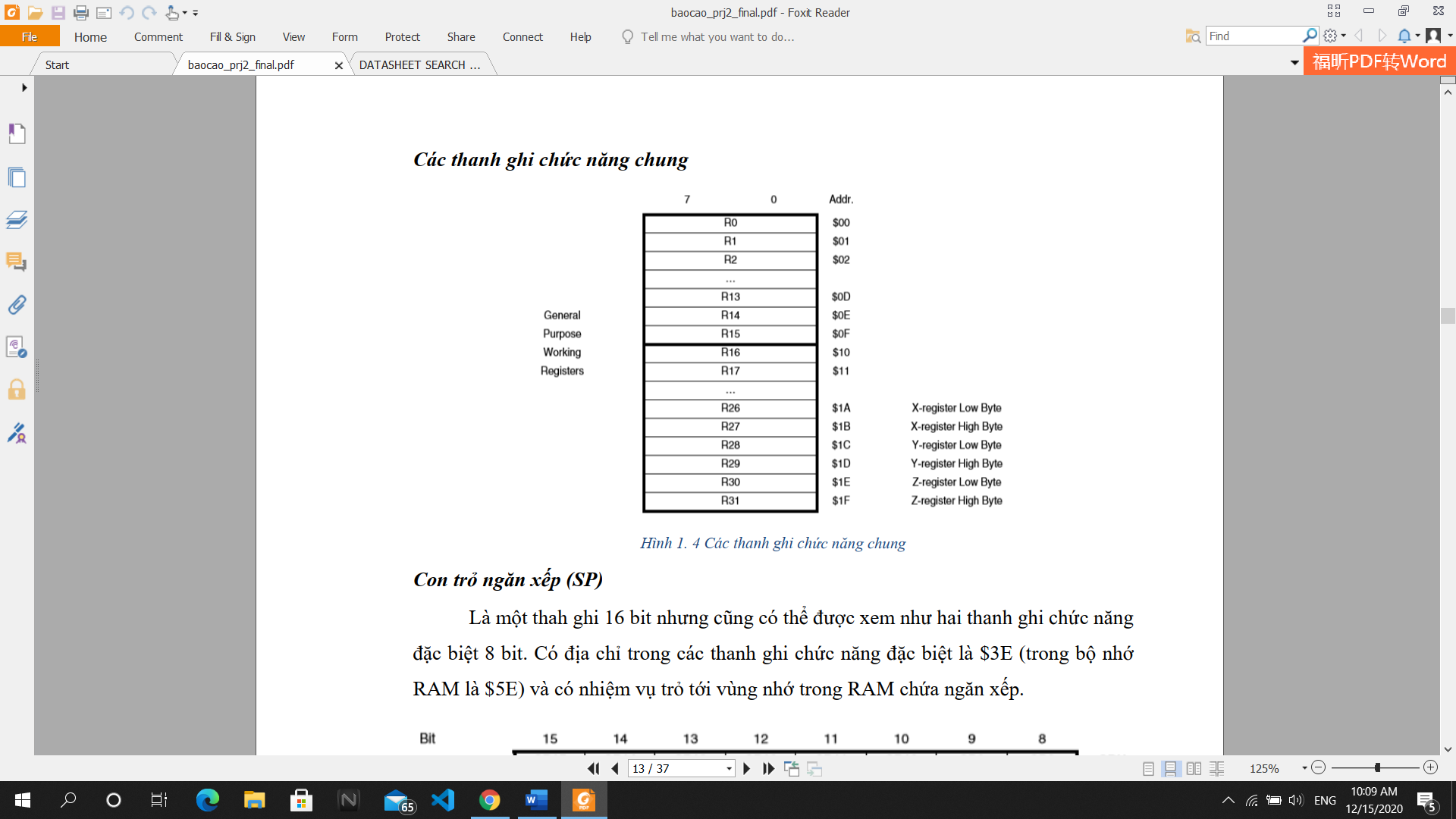
**Thanh ghi trạng thái**

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

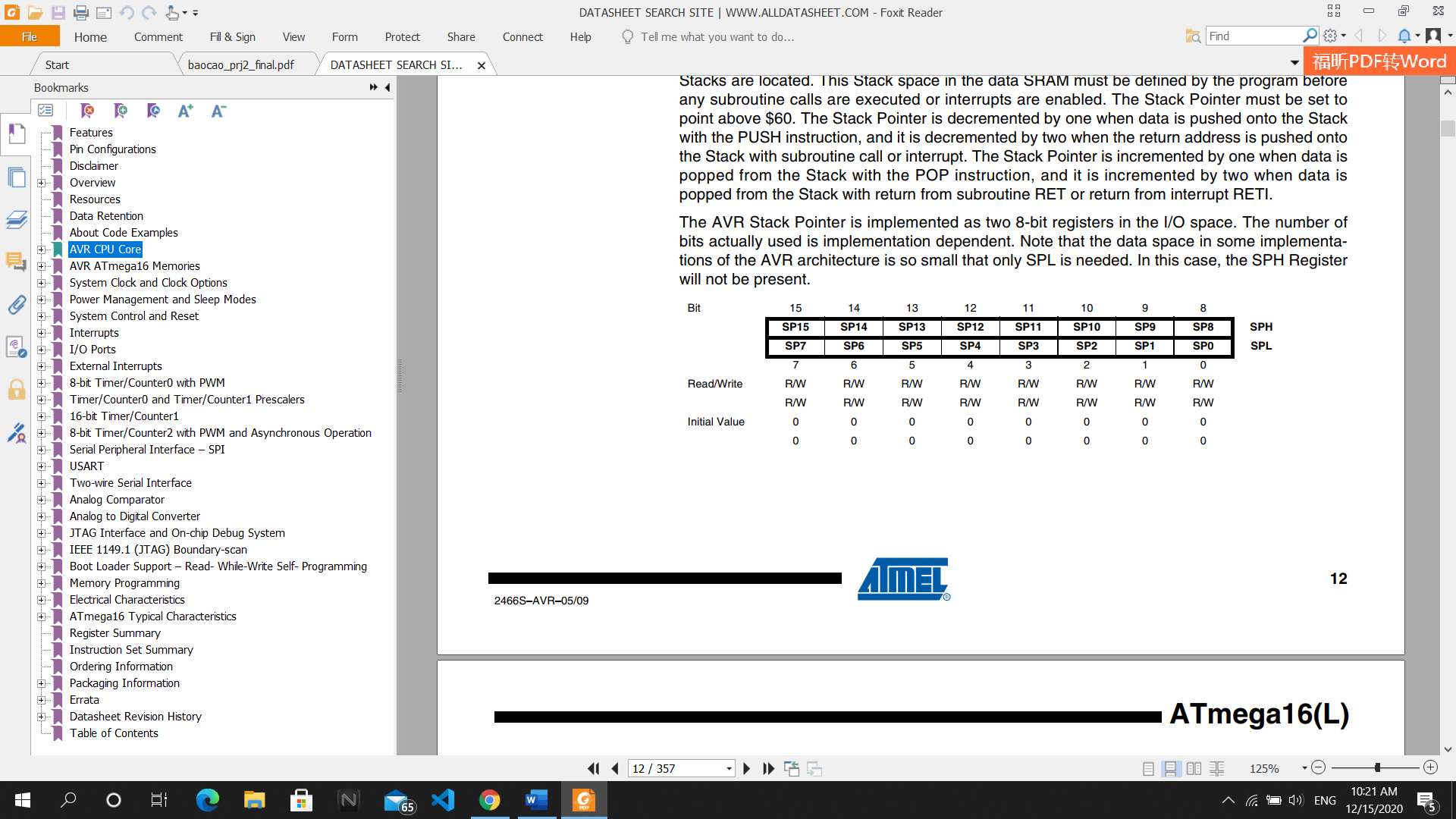
**Các thanh ghi chức năng chung**



Hình 1. 3 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.

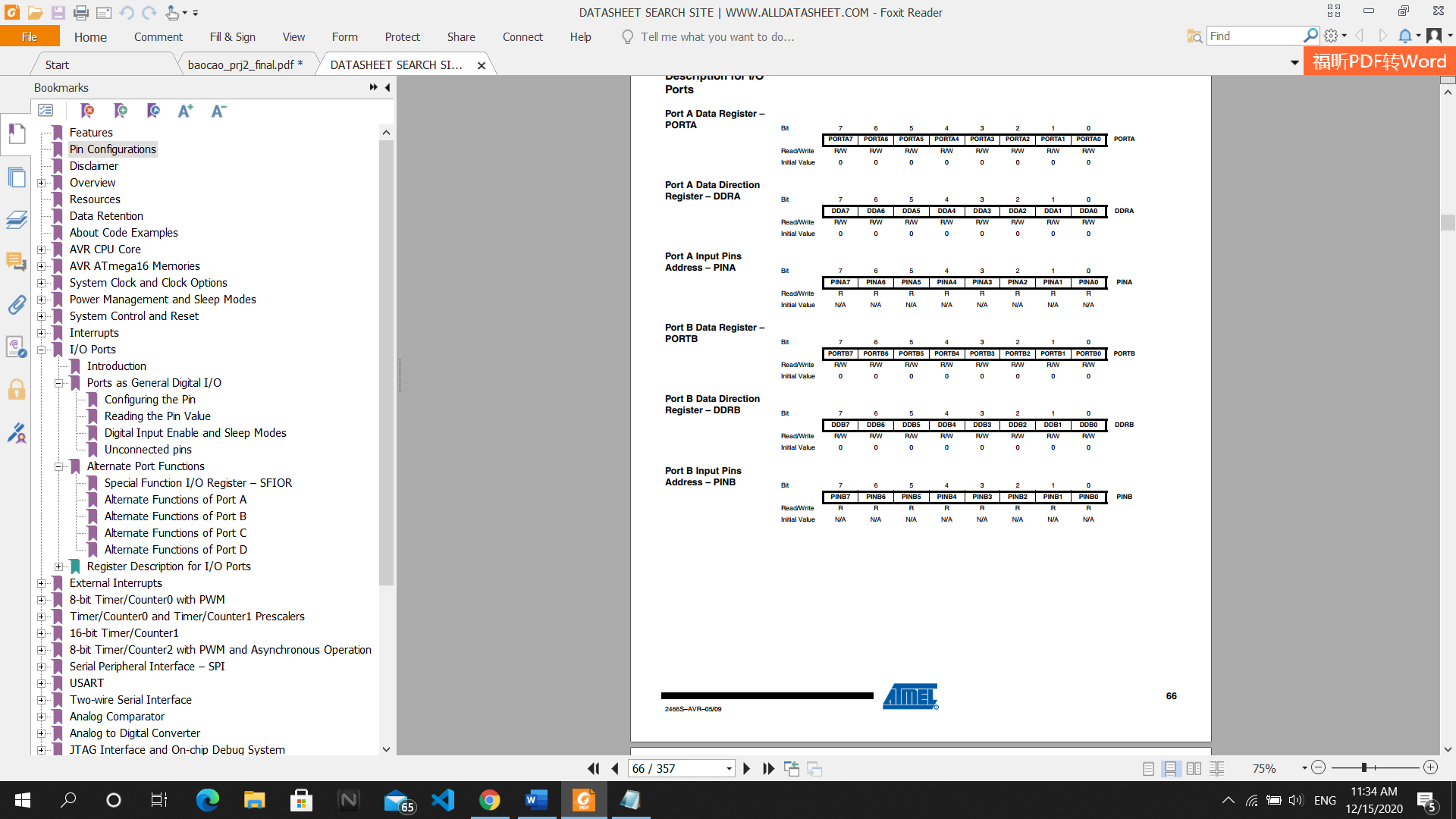
Khi chương trình phục vụ ngắt hoặc chương trình con thì con trỏ PC được lưu vào ngăn xếp trong khi con trỏ ngăn xếp bị giảm hai vị trí. Và con trỏ ngăn xếp sẽ giảm 1 khi thực hiện lệnh PUSH. Ngược lại thực hiện lệnh POP thì con trỏ ngăn xếp sẽ tăng 1 và khi thực hiện lệnh RET hoặc RETI thì con trỏ ngăn 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

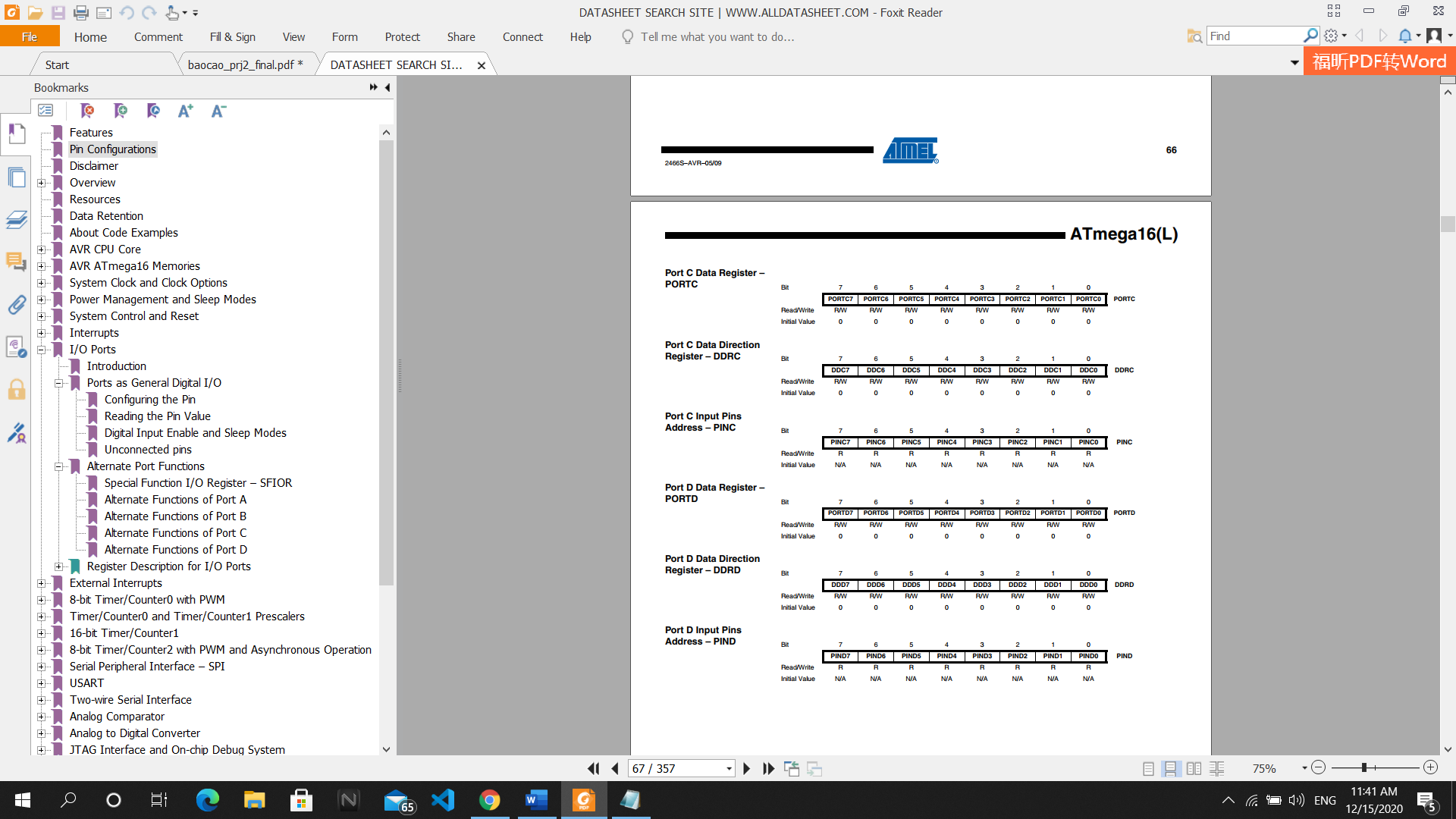
### Các cổng I/O (xuất nhập)



Hình 1. 4 Cấu hình chân Atmega16

**Các thanh ghi mô tả các cổng I/O:**



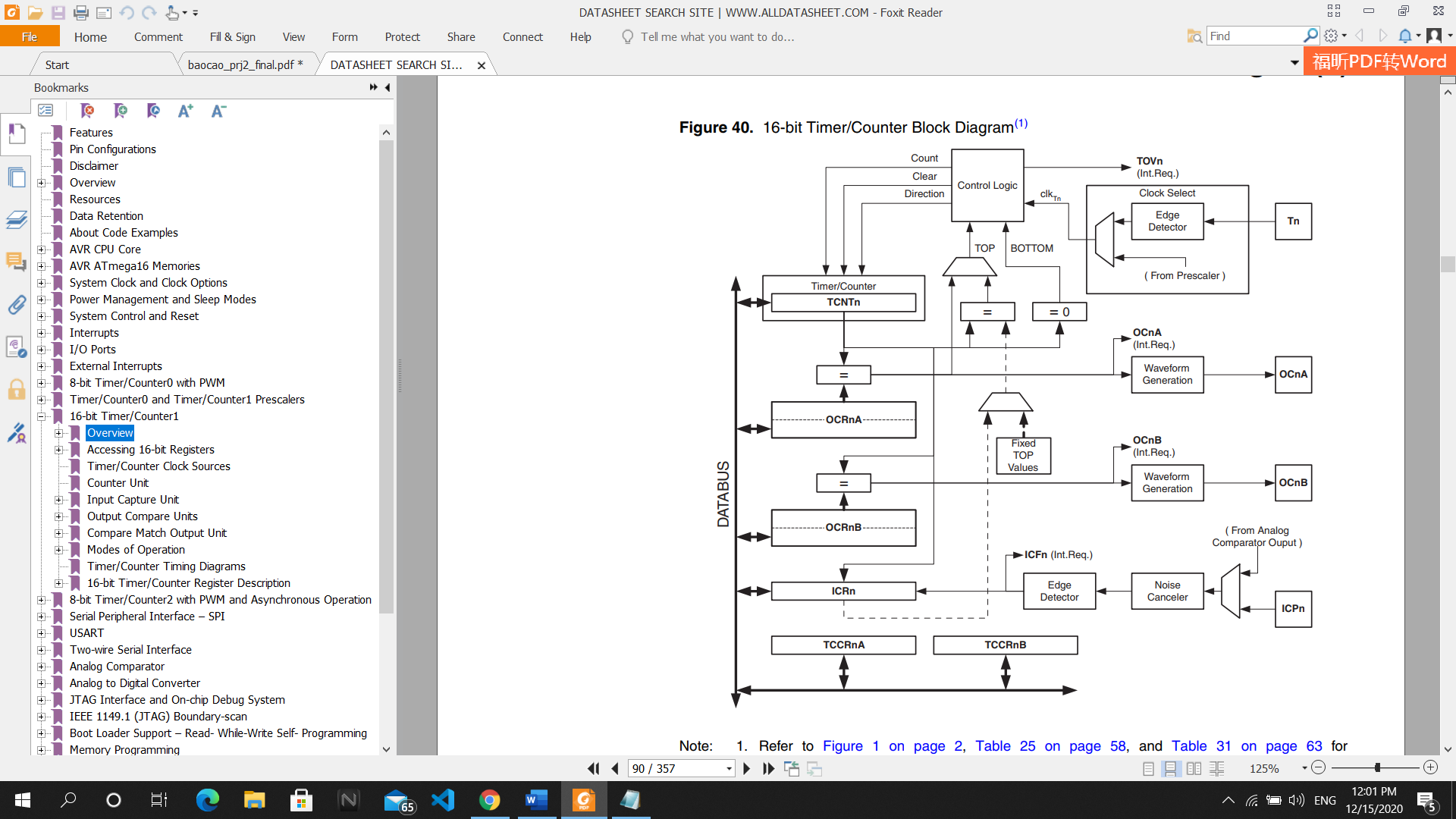


**Chức năng các chân:**

* 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

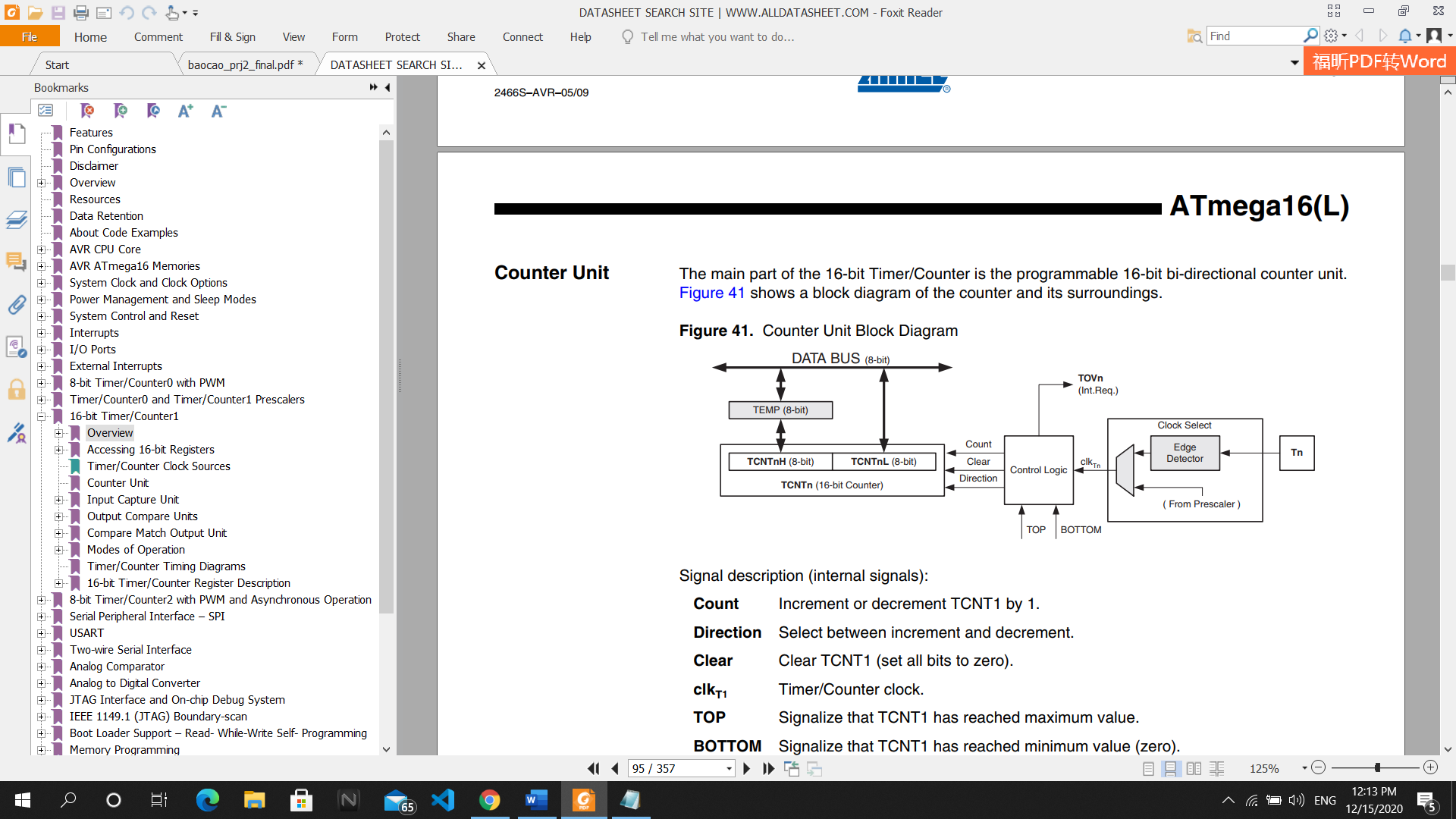
Atmega cung cấp cho lập trình viên 3 bộ timer/counter đó là: timer/counter0 8-bit, timer/counter1 16-bit, timer/counter 8-bit. Trong đề tài này chúng em sử dụng timer/counter1



Hình 1. 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. 6 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

## PWM(****Pulse Width Modulation)- điều chế độ rộng xung****

### Tổng quan về PWM

Phương pháp điều xung PWM (Pulse Width Modulation) là phương pháp điều chỉnh điện áp ra tải, hay nói cách khác, là phương pháp điều chế dựa trên sự thay đổi độ rộng của chuỗi xung vuông, dẫn đến sự thay đổi điện áp ra.

Các PWM khi biến đổi thì có cùng 1 tần số và khác nhau về độ rộng của sườn âm và sườn dương

Ảnh có chứa bàn phím

Mô tả được tạo tự động

Hình 1. 7 Đồ thị dạng xung điều chế PWM

**Ứng dụng của PWM trong điều khiển**

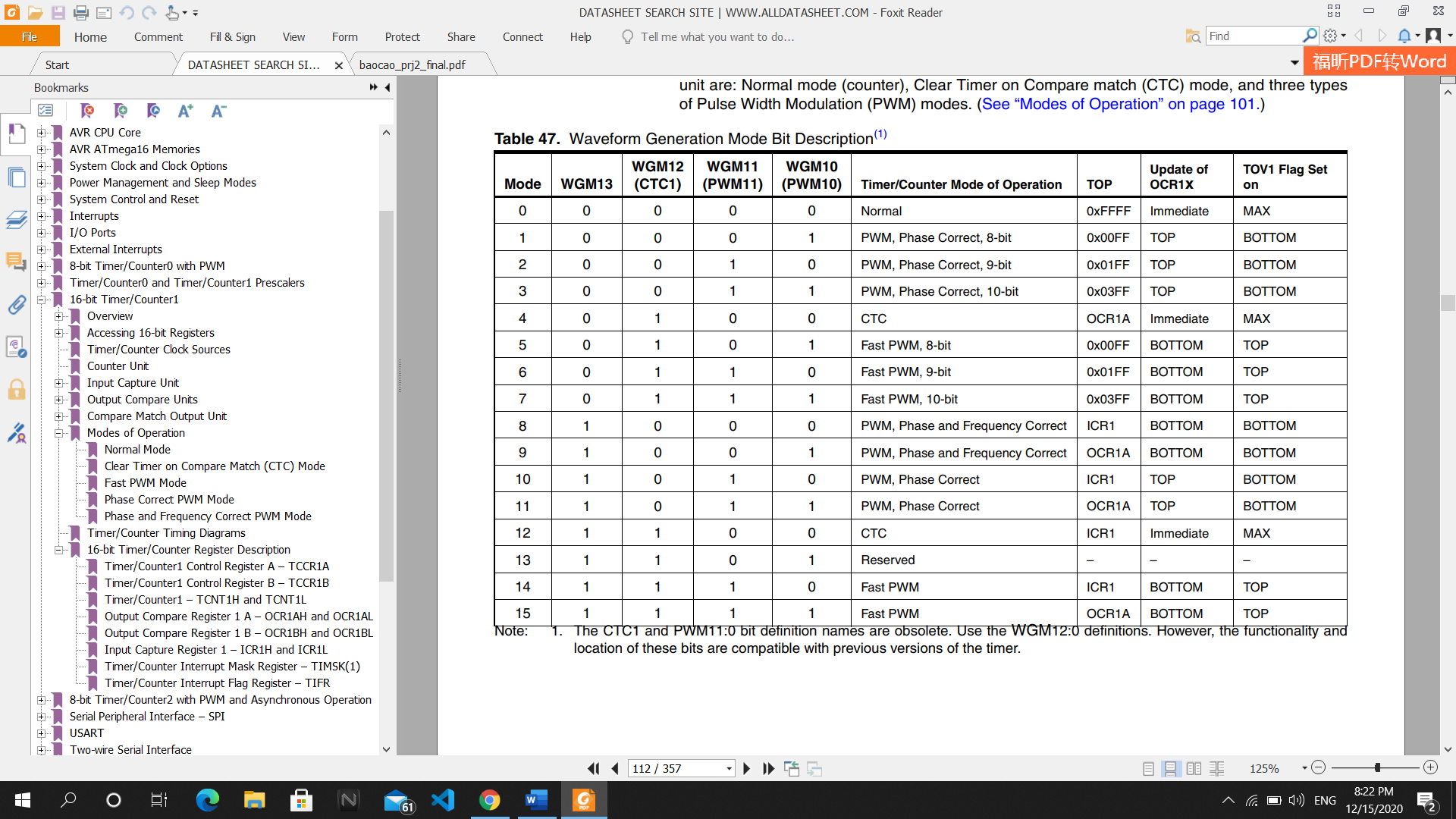
PWM được ứng dụng nhiều trong điều khiển. Điển hình nhất mà chúng ta thường hay gặp là điều khiển động cơ và các bộ xung áp, điều áp… Sử dụng PWM điều khiển độ nhanh chậm của động cơ hay cao hơn nữa, nó còn được dùng để điều khiển sự ổn định tốc độ động cơ.

Ngoài lĩnh vực điều khiển hay ổn định tải thì PWM còn tham gia và điều chế các mạch nguồn như : boot, buck, nghịch lưu 1 pha và 3 pha…

PWM còn gặp nhiều trong thực tế ở các mạch điện điều khiển. Điều đặc biệt là PWM chuyên dùng để điều khiển các phần tử điện tử công suất có đường đặc tính là tuyến tính khi có sẵn 1 nguồn 1 chiều cố định .Như vậy PWM được ứng dụng rất nhiều trong các thiết bị điện- điện tử. PWM cũng chính là nhân tố mà các đội Robocon sử dụng để điều khiển động cơ hay ổn định tốc độ động cơ.

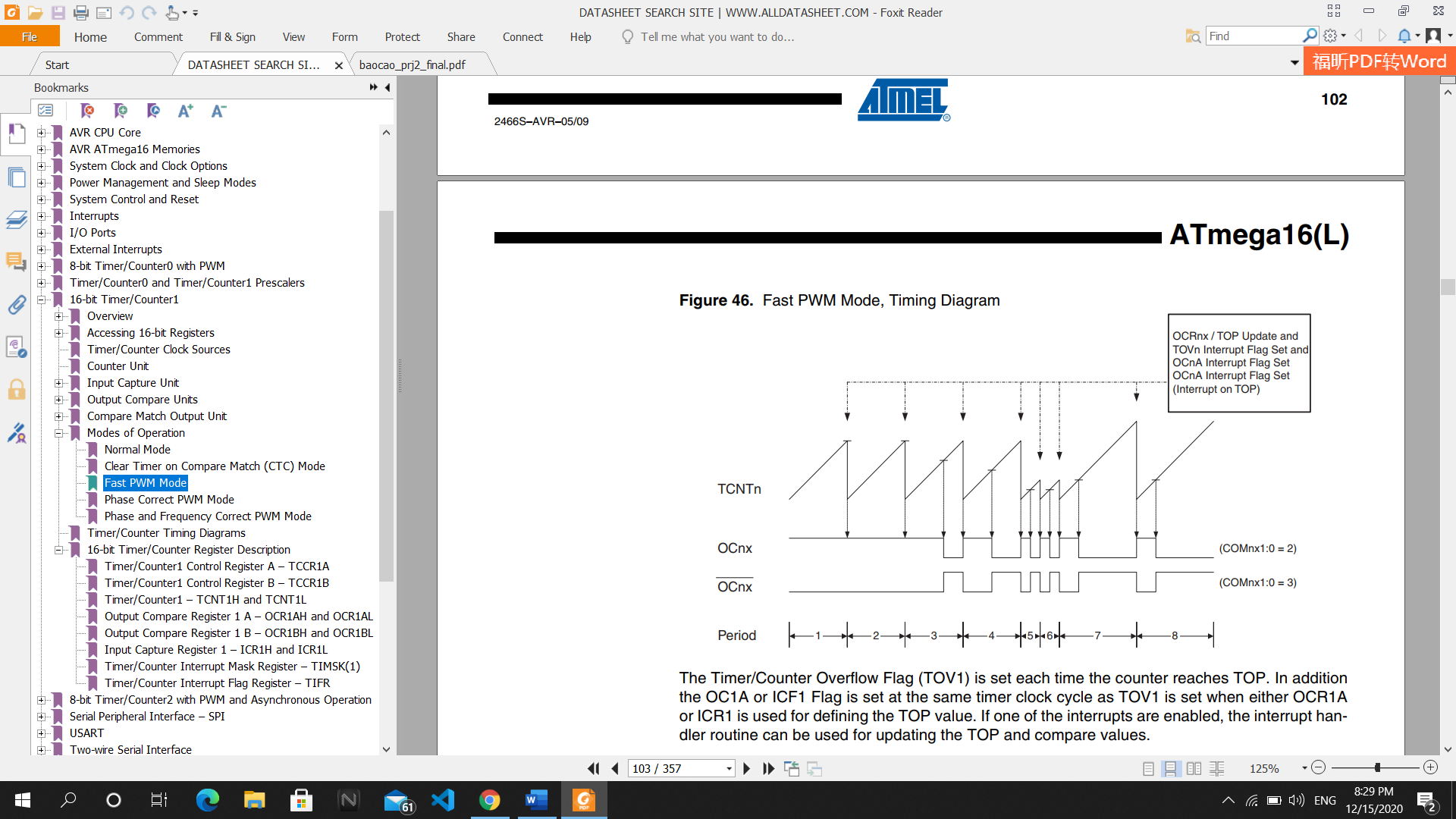
### Điều chế xung PWM bằng chế độ Fast PWM của vi điều khiển Atmega16

Ở đề tài này chúng em dùng bộ timer/counter1 để điều chế xung PWM cụ thể là mode 14 đó là Fast PWM với giá trị TOP được ghi lên thanh ghi ICR1



Hình 1. Các chế độ hoạt động của timer/counter1

**Fast PWM:**



Hình 1. 9 Mô tả hoạt động Fast PWM

Fast PWM khác với PWM ở chỗ Fast PWM là kiểu 1 sườn dốc(single-slope). Thanh ghi TCNTn có thể đếm từ BOTTOM(0x0000) đến TOP(0xFFFF) sau đó cờ tràn được bật TCNTn được đặt về BOTTOM hoặc giá trị TOP có thể được đặt bởi ICR1. Mỗi khi đến giá trị so sánh giá trị thanh ghi OCnx sẽ đảo trạng thái.

Bộ Timer1 của Atmega16 cung cấp cho chúng ta 2 kênh tạo xung điều rộng PWM:

* Kênh A: tín hiệu ra ở OC1A (PORTB.1)
* Kênh B: tín hiệu ra ở OC1B (PORTB.2)

Trong đồ án này, sử dụng chế độ Fast PWM, mode 14 (Kênh A).

* Thiết lập dạng tín hiệu PWM ra trên thanh ghi TCCR1A .
* Khi hoạt động, ban đầu chân OC1A ở mức cao (tùy vào dạng tín hiệu PWM ra chọn ở trên), TCNT1 tăng giá trị từ 0 cho đến khi bằng giá trị trên thanh ghi OCR1A26thì chân OC1A được xóa về 0. TCNT1 vẫn tiếp tục tăng đến khi bằng giá trị trong thanh ghi ICR1 thì TCNT1 reset về 0 và chân OC1A được kéo lên mức cao.

Thiết lập:

* Giá trị của ICR1: là chu kỳ xung.
* Giá trị của OCR1A: là thời gian xung ở mức cao.

## Các phần mềm sử dụng

### Trình biên dịch CodevisionAVR

CodeVisionAVR là một trình biên dịch chéo C, môi trường phát triền tích hợp và  
tạo chương trình tự động được thiết kế cho họ AVR của Atmel

Graphical user interface, text, application

Description automatically generated

Hình 1. 10 Giao diện của CodevisionAVR

* Ưu điểm: CodevisionAVR cung cấp nhiều thư viện khởi tạo cho người lập trình, giúp lập trình viên dễ dàng trong việc chạy code và debug
* Nhược điểm: sau khi khởi tạo thì khá nhiều phần rườm rà, việc sử dụng thư viện dẫn đến người lập trình không hiểu hết chức năng của vi điều khiển gây khó khăn trong debug mạch thực tế

### Phần mềm mô phỏng Proteus

Proteus giúp người dùng mô phỏng mạch, kiểm tra code để sau đó dựa vào kết quả mô phỏng mà ta đi thiết kế mạch thực tế

A picture containing diagram

Description automatically generated

Hình 1. 11 Giao diện Proteus

### Phần mềm vẽ mạch Altium

Altium là một công cụ mạnh mẽ hỗ trợ vẽ mạch nguyên lý, mạch in

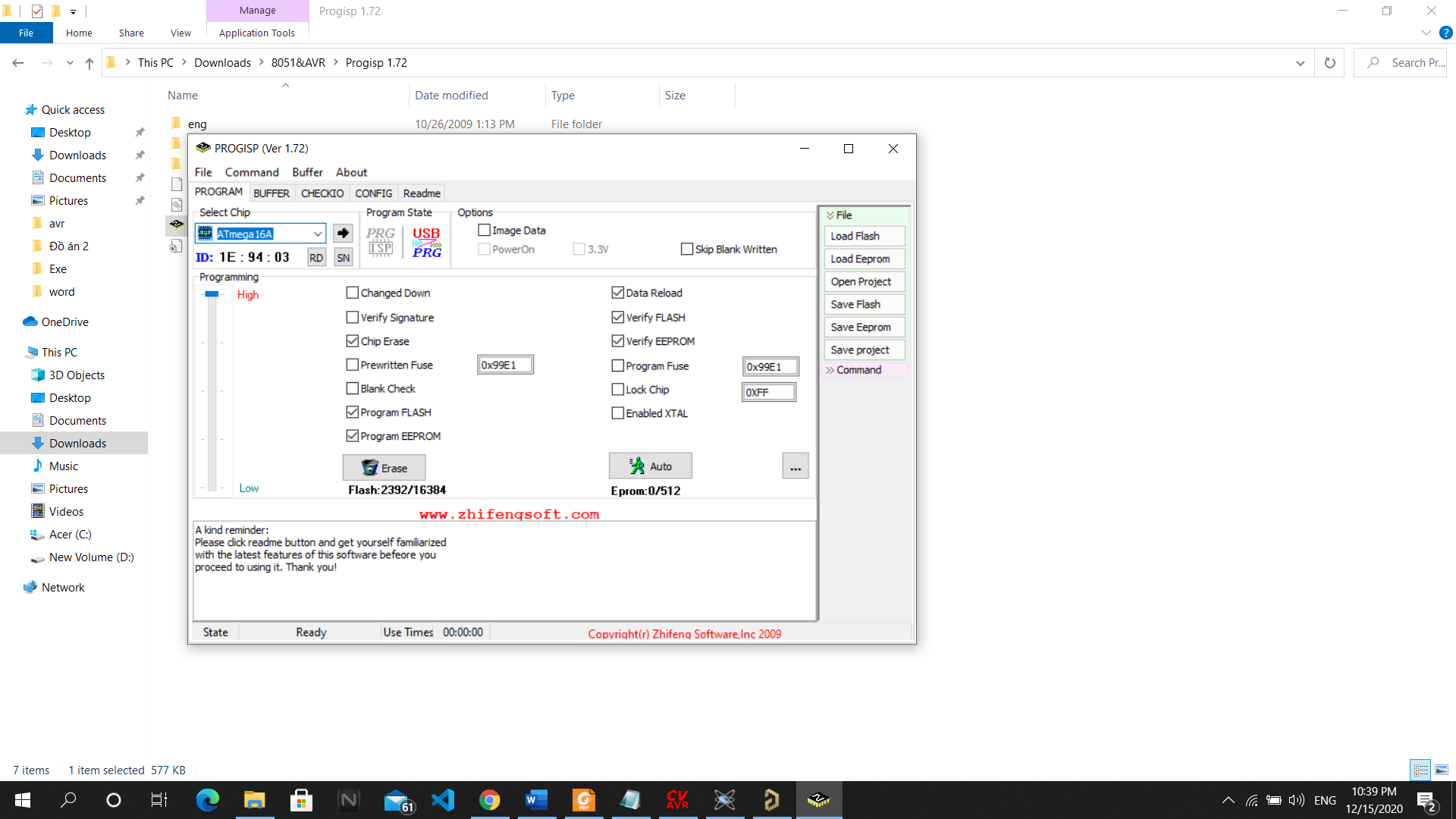
A screenshot of a computer

Description automatically generated

Hình 1. 12 Giao diện Altium Designer 18

### Phần mềm nạp code ProgISP

Phần mềm giúp người lập trình nạp code viết từ trình biên dịch CodevisionAVR đã chạy mô phỏng ở proteus xuống kit nạp Atmega16. Lưu ý cần cấu hình Fulse Bit lại để PORTC được sử dụng đúng mục đích do chức năng JTAG của vi điều khiển đã cấu hình PORTC



Hình 1. 13 Giao diện ProgISP 1.72

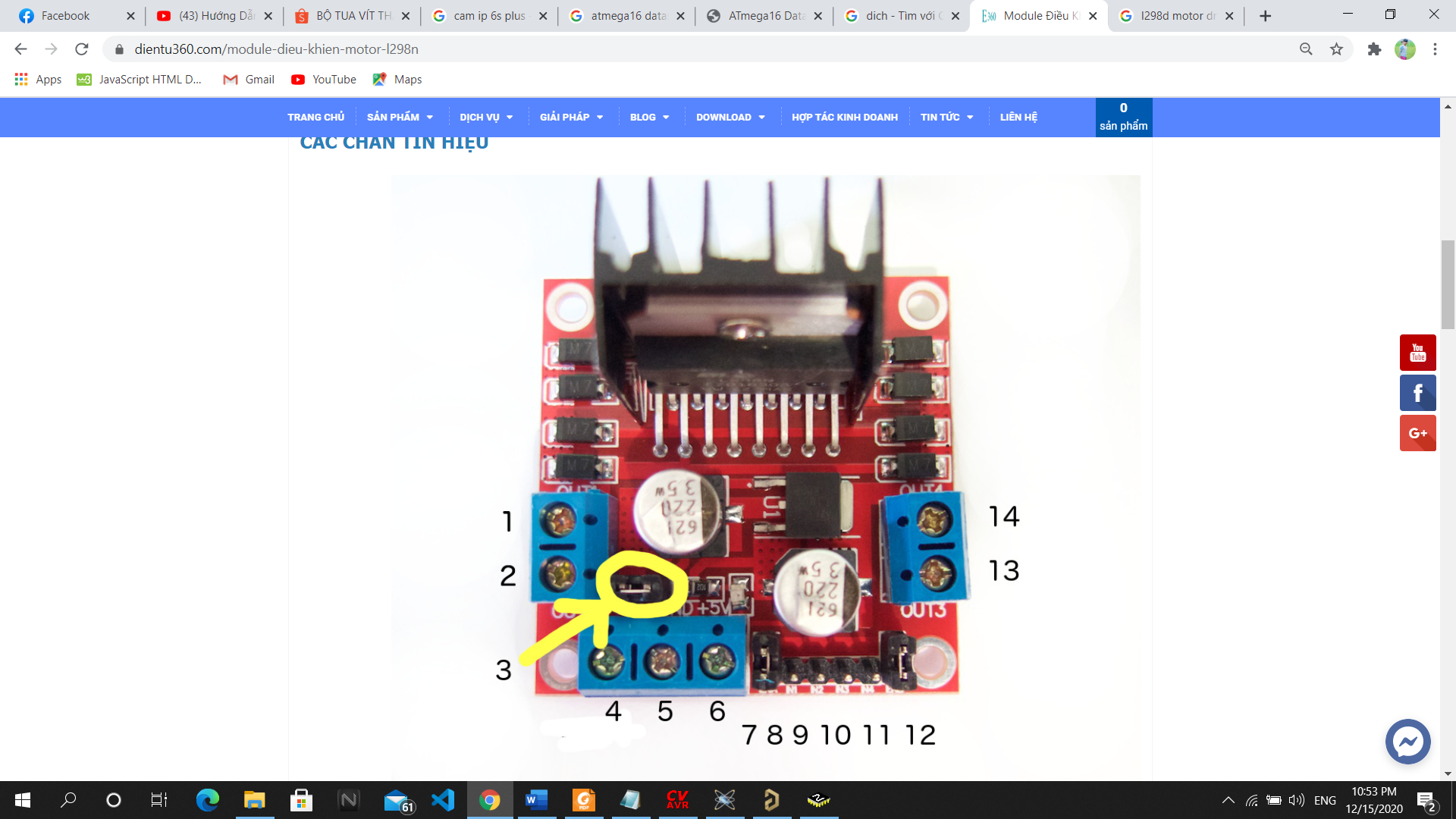
## Các linh kiện được sử dụng

### Module điều khiển động cơ L298N

**Thông số kĩ thuật:**

* Driver: L298N tích hợp hai mạch cầu H
* Điện áp điều khiển : +5V ~ +12 V
* Dòng tối đa cho mỗi cầu H là :2A
* Điện áp của tín hiệu điều khiển : +5 V ~ +7 V
* Dòng của tín hiệu điều khiển : 0 ~ 36Ma
* Công suất hao phí : 20W (khi nhiệt độ T = 75 °C)
* Nhiệt độ bảo quản : -25°C ~ +130

**Chức năng các chân:**



Hình 1. Module L298N

1. DC motor 1 “+” hoặc stepper motor A+

2. DC motor 1 “-” hoặc stepper motor A-

3. 12V jumper – tháo jumper ra nếu sử dụng nguồn trên 12V. Jumper này dùng để cấp nguồn cho IC ổn áp tạo ra nguồn 5V nếu nguồn trên 12V sẽ làm cháy IC Nguồn

4. Cắm dây nguồn cung cấp điện áp cho motor vào đây từ 6V đến 35V.

5. Cắm chân GND của nguồn vào đây

6. Ngõ ra nguồn 5V, nếu jumper đầu vào không rút ra.

7. Chân Enable của Motor 1, chân này dùng để cấp xung PWM cho motor nếu dùng VDK thì rút jumper ra và cắm chân PWM vào đây. Giữ nguyên khi dùng với động cơ bước

8. IN1

9. IN2

10.IN3

11.IN4

12. Chân Enable của Motor 2, chân này dùng để cấp xung PWM cho motor nếu dùng VDK thì rút jumper ra và cắm chân PWM vào đây. Giữ nguyên khi dùng với động cơ bước

13. DC motor 2 “+” hoặc stepper motor B+

14. DC motor 2 “-” hoặc stepper motor B-

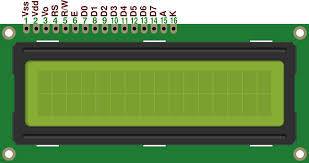
### Động cơ DC 12V



Hình 1. 15 Động cơ 365 12VDC

### Màn hình hiển thị LCD

Đây là loại màn hình phổ biến dùng chip HD7470, hiển thị 16 kí tự 2 hàng



Hình 1. 16 LCD 16x2

* Vss: cấp GND
* Vdd: cấp +5V
* Vo: chỉnh độ đậm nhạt
* RS, R/W, E: các chân điều khiển
* D0-D7: chân dữ liệu
* A,K(đèn nền): cấp +5V và GND

# CHƯƠNG 2: THIẾT KẾ MẠCH

2.1. Sơ đồ khối

LCD

Nguồn

vi điều khiển

PWM

Nút nhấn

Hình 2. Sơ đồ khối

Sơ đồ khối gồm 4 khối chính:

* khối nguồn: nguồn vào cung cấp cho cả vi điều khiển và động cơ
* khối input: các nút nhấn gửi tín hiệu đầu vào
* khối xử lí: vi điều khiển atmega xử lí đầu vào, tính toán các thông số sau đó gửi ra output
* khối output: LCD hiển thị các thông số và động cơ DC được điều khiển bởi các chân output và tín hiệu PWM

2.2. Sơ đồ nguyên lí mạch kit trên Altium

Do đã có mạch kit nên việc chạy thử mạch và kết nối ngoại vị trở nên đơn giản hơn. Vì vậy ta không cần phải thiết kế mạch riêng cho đề tài mà chỉ cần kết nối ngoại vi vào để sử dụng.



Hình 2. Sơ đồ nguyên lí mạch kit

2.3. Mạch in



Hình 2. 3 Mạch in PCB

# CHƯƠNG 3: YÊU CẦU MẠCH

3.1. Yêu cầu chức năng

Mạch có thể điều khiển được động cơ 1 chiều 12V, có thể đảo chiều quay động cơ, tăng giảm tốc độ quay, cho phép khởi động, dừng khi người dùng bấm nút. Khi nút dừng lại được kích hoạt thì mọi thao tác với các nút khác đều không được thực hiện và tất cả các thông số như: phần trăm xung PWM, chiều quay, trang thái bật tắt, động cơ đều phải được hiển thị ra LCD.

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. Ứng dụng

Từ yêu cầu thì mạch được ứng dụng trong điều khiển động cơ quay trong các thiết bị công nghiệp đến các thiết bị điện tử nhỏ như là: điều khiển động cơ bước, ứng dụng trong xe tự hành, hay tản nhiệt cho các thiết bị máy móc tỏa nhiệt

# CHƯƠNG 4: KẾT QUẢ THỰC HIỆN

4.1. Kết quả mô phỏng

Kết quả mô phỏng chạy đúng với yêu cầu đề ra

Chart

Description automatically generated with medium confidence

Hình 4. 1 Kết quả mô phỏng bằng phần mềm Proteus

4.2. Kết quả thực tế trên mạch

Tuy không thể đạt thông số như yêu cầu và mạch mô phỏng nhưng mạch thực tế chạy khá tốt và đạt yêu cầu đề ra

Ảnh có chứa thiết bị điện tử

Mô tả được tạo tự động

Hình 4. 2 Kết quả thực tế

# KẾT LUẬN

Sau một kỳ tìm hiểu chúng em đã cơ bản hoàn thành đề tài “Thiết kế hệ thống điều khiển động cơ một chiều”. Trong quá trình xây dựng và hoàn thành sản phẩm, em đã tích lũy được kinh nghiệm thiết kế, làm và hàn mạch. Trong quá trình thực hiện đề tài, do còn nhiều hạn chế về mặt thời gian, kiến thức cũng như công cụ và kinh nghiệm nên đề tài vẫn còn những thiếu sót nhất định, mong được thầy và các bạn đánh giá, góp ý một cách khách quan để chúng em có thể rút ra kinh nghiệm và hoàn thiện đề tài hơn nữa.

Để hoàn thành bài báo cáo này, chúng em xin chân thành gửi lời cảm ơn tới sự quan tâm và hướng dẫn tận tình của thầy!

# DANH MỤC TÀI LIỆU THAM KHẢO

[1] <https://www.keil.com/dd/docs/datashts/atmel/atmega16_ds.pdf>

[2] <https://viettech.edu.vn/news/dieu-che-pwm-la-gi/>

[3] <https://www.youtube.com/watch?v=ItNSLtxjmrk>

[4] <https://www.homemade-circuits.com/dc-motor-speed-controller-circuits/>

# PHỤ LỤC

### Phụ lục 1: Source code

#include <mega16.h>

#include <delay.h>

#include <alcd.h>

#include <stdio.h>

#define Tang PINB.0

#define Giam PINB.1

#define Dao PINB.2

#define Stop PINB.3

unsigned long encoder=0;

unsigned char countT0=0,countT2=0 ;

unsigned int speed;

unsigned int temp=0, temp1=0;

unsigned char phantram=0;

char lcd[20];

interrupt [EXT\_INT0] void ext\_int0\_isr(void)

{

encoder++;

}

interrupt [TIM0\_OVF] void timer0\_ovf\_isr(void)

{

TCCR0=(0<<CS02) | (0<<CS01) | (0<<CS00);

if(countT0 >= 20)

{

countT0=0;

//speed=(float)((float)(encoder\*1000)/(200\*640))\*10;

speed=((float)encoder/128)\*10\*60;

encoder=0;

}

countT0++;

TCCR0=(1<<CS02) | (0<<CS01) | (1<<CS00);

TCNT0=0x06;

}

interrupt [TIM2\_OVF] void timer2\_ovf\_isr(void)

{

TCCR2=(0<<CS22) | (0<<CS21) | (0<<CS20);

if(countT2 >=5)

{

countT2=0;

if(!Tang)

{

delay\_ms(50);

if(!Tang)

{

while(!Tang);

temp+=800;

if(temp>=7999)temp=7999;

}

}

else if(!Giam)

{

if(temp>=800)

{

delay\_ms(50);

if(!Giam)

{

while(!Giam);

temp-=800;

if(temp<=0) temp=0;

}

}

else temp=0;

}

else if(!Dao)

{

delay\_ms(50);

if(!Dao)

{

while(!Dao);

PORTD.3=!PORTD.3;

}

}

else if(!Stop)

{

delay\_ms(50);

if(!Stop)

{

while(!Stop);

if(temp !=0){

temp1=temp;

temp=0; }

else{

temp= temp1;}

}

}

if(PORTD.3==0)

{

OCR1AH= temp >>8 ;

OCR1AL= temp && 0xFF ;

}

else

{

unsigned int Temp=7999-temp;

OCR1AH=Temp>>8 ;

OCR1AL=Temp && 0xFF ;

}

}

countT2++;

TCNT2=0x00;

TCCR2=(1<<CS22) | (1<<CS21) | (1<<CS20);

}

void main(void)

{

DDRA=(0<<DDA7) | (0<<DDA6) | (0<<DDA5) | (0<<DDA4) | (0<<DDA3) | (0<<DDA2) | (0<<DDA1) | (0<<DDA0);

PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) | (0<<PORTA4) | (0<<PORTA3) | (0<<PORTA2) | (0<<PORTA1) | (0<<PORTA0);

DDRB=(0<<DDB7) | (0<<DDB6) | (0<<DDB5) | (0<<DDB4) | (0<<DDB3) | (0<<DDB2) | (0<<DDB1) | (0<<DDB0);

PORTB=(0<<PORTB7) | (0<<PORTB6) | (0<<PORTB5) | (0<<PORTB4) | (1<<PORTB3) | (1<<PORTB2) | (1<<PORTB1) | (1<<PORTB0);

DDRC=(0<<DDC7) | (0<<DDC6) | (0<<DDC5) | (0<<DDC4) | (0<<DDC3) | (0<<DDC2) | (0<<DDC1) | (0<<DDC0);

PORTC=(0<<PORTC7) | (0<<PORTC6) | (1<<PORTC5) | (0<<PORTC4) | (1<<PORTC3) | (0<<PORTC2) | (0<<PORTC1) | (0<<PORTC0);

DDRD=(0<<DDD7) | (0<<DDD6) | (1<<DDD5) | (0<<DDD4) | (1<<DDD3) | (0<<DDD2) | (0<<DDD1) | (0<<DDD0);

PORTD= 0x20;

TCCR0=(0<<WGM00) | (0<<COM01) | (0<<COM00) | (0<<WGM01) | (1<<CS02) | (0<<CS01) | (1<<CS00);

TCNT0=0x06;

OCR0=0x00;

TCCR1A=(1<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) | (0<<COM1B0) | (1<<WGM11) | (0<<WGM10);

TCCR1B=(0<<ICNC1) | (0<<ICES1) | (1<<WGM13) | (1<<WGM12) | (0<<CS12) | (0<<CS11) | (1<<CS10);

TCNT1H=0x00;

TCNT1L=0x00;

ICR1H=0x1F;

ICR1L=0x3F;

OCR1AH=0x00;

OCR1AL=0x00;

OCR1BH=0x00;

OCR1BL=0x00;

ASSR=0<<AS2;

TCCR2=(0<<PWM2) | (0<<COM21) | (0<<COM20) | (0<<CTC2) | (1<<CS22) | (1<<CS21) | (1<<CS20);

TCNT2=0x06;

OCR2=0x00;

TIMSK=(0<<OCIE2) | (1<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) | (0<<OCIE1B) | (0<<TOIE1) | (0<<OCIE0) | (1<<TOIE0);

GICR|=(0<<INT1) | (1<<INT0) | (0<<INT2);

MCUCR=(0<<ISC11) | (0<<ISC10) | (1<<ISC01) | (0<<ISC00);

MCUCSR=(0<<ISC2);

GIFR=(0<<INTF1) | (1<<INTF0) | (0<<INTF2);

lcd\_init(16);

#asm("sei")

lcd\_clear();

lcd\_gotoxy(0,0);

lcd\_puts("Xung: ");

lcd\_gotoxy(0,1);

lcd\_puts("Toc do: ");

while (1)

{

lcd\_gotoxy(8,1);

sprintf(lcd,"%d.%d V/s ",speed/10,speed%10);

lcd\_puts(lcd);

lcd\_gotoxy(6,0);

phantram = ((float)temp/7999) \*100;

sprintf(lcd,"%d %% ",phantram);

lcd\_puts(lcd);

delay\_ms(100);

}

}