

Mục lục

Contents

Mục lục.....	1
CHƯƠNG I : NGUYÊN LÝ HOẠT ĐỘNG CỦA HỆ THỐNG VI ĐIỀU KHIỂN.....	3
I. Xây dựng mục tiêu và sơ đồ khối của hệ thống dùng vi điều khiển Pic.....	3
1. Mục Tiêu:.....	3
2. Sơ đồ khối của hệ thống dùng vi điều khiển Pic.....	3
II. Nguyên lí hoạt động của hệ thống.....	4
1. Hệ thống.....	4
2. Nguyên lí hoạt động	5
CHƯƠNG II : TÍNH CHỌN LINH KIỆN TRONG HỆ THỐNG	9
I. Giới thiệu các linh kiện có trong hệ thống.....	9
A. Vi điều khiển PIC 16F877A.....	9
B. Mạch cầu H.....	23
C. Hiển thị LCD	26
D. Đối tượng điều khiển: Động cơ DC	29
II. Tính toán linh kiện trong hệ thống.....	30
A. Tính toán khối điều chế độ rộng xung PWM	30
B. Chọn diode cho mạch công suất.....	30
C. Chọn điện trở.....	31
CHƯƠNG III : MÔ PHỎNG HỆ THỐNG	32
I. Thiết kế mạch nguyên lý.....	32
A. Khối nguồn	33
B. Khối mạch công suất.....	34
C. Khối mạch phím.....	36
D. Khối mạch hiển thị.....	38
E. Khối mạch điều khiển.....	41
II. Mô phỏng bằng phần mềm ứng dụng Proteus.....	41
1/ Lưu đồ thuật toán	41
2/ Chạy mô phỏng bằng phần mềm Proteus.....	43

CHƯƠNG 4: CHẾ TẠO MẠCH THỰC TẾ.....	45
A. THIẾT KẾ MẠCH IN	45
B. LẮP ĐẶT THIẾT BỊ VÀ HOÀN THIỆN MẠCH	46
C. CHẠY MẠCH VÀ ĐÁNH GIÁ KẾT QUẢ.....	46
PHỤ LỤC.....	47
A. CODE CHƯƠNG TRÌNH BẰNG NGÔN NGỮ C QUA PHẦN MỀM CCS.....	47
B. CHƯƠNG TRÌNH HỢP NGỮ THIẾT LẬP BAN ĐẦU.....	55

CHƯƠNG I

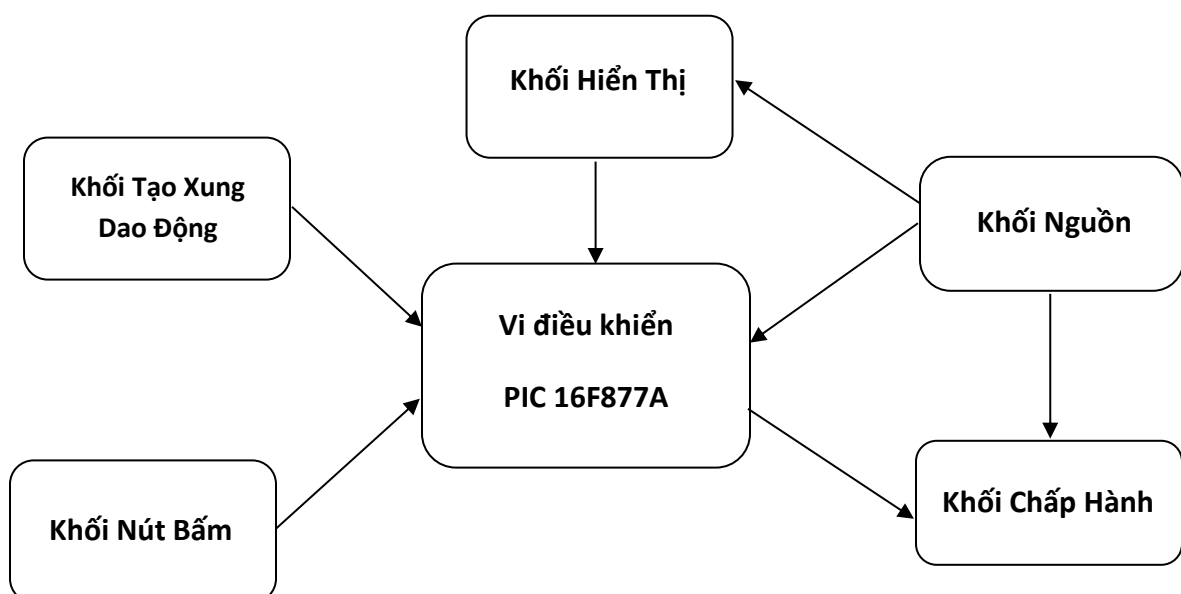
NGUYÊN LÝ HOẠT ĐỘNG CỦA HỆ THỐNG VI ĐIỀU KHIỂN

I. Xây dựng mục tiêu và sơ đồ khối của hệ thống dùng vi điều khiển Pic

1. Mục Tiêu:

- Dùng vi điều khiển Pic 16F877A là vi điều khiển trung tâm để điều khiển tốc độ động cơ DC.
- Lập trình C cho Pic 16F877A điều khiển tốc độ cho động cơ DC bằng phương pháp điều biến độ rộng xung (PWM).
- Mức tốc độ được biểu hiện qua hiển thị LCD 16x2.
- Có nút bấm điều khiển chọn 3 mức tốc độ thể hiện bằng 3 LED màu xanh, đỏ, vàng.
- Chương trình vi điều khiển có sử dụng timer, ngắt.

2. Sơ đồ khối của hệ thống dùng vi điều khiển Pic



Chức năng của từng khối :

- Vi điều khiển PIC16F877A: Là vi điều khiển chính sử dụng để lập trình cho hệ thống hoạt động , gửi và nhận tín hiệu từ nút bấm, sau đó xuất ra LCD và cơ cấu chấp hành.
- Khối hiển thị: Sử dụng LCD 16x2 (2 dòng và 16 kí tự) để hiển thị mức tốc độ.
- Khối nguồn: - Adapter 12V/2A: Biến đổi nguồn điện từ mạng điện thành nguồn phù hợp với hệ thống.
- Mạch nạp: Nạp chương trình cho vi điều khiển, kèm nguồn 5V cấp cho vi điều khiển.
- Khối tạo xung dao động: Sử dụng thạch anh tạo xung dao động ổn định cấp cho hệ thống.
- Khối nút bấm: Sử dụng nút bấm cơ học để thao tác điều khiển hoạt động của hệ thống.
- Khối cơ cấu chấp hành: Động cơ DC 12V quay theo 3 mức tốc độ.

II. Nguyên lí hoạt động của hệ thống

1. Hệ thống

- Sử dụng Pic 16F877A là vi điều khiển trung tâm. Dùng chương trình CCS lập trình C và biên dịch chương trình thành ngôn ngữ hexacode.
- Xây dựng khối bàn phím gồm 5 phím để cho phép hoạt động, nhập mức tốc độ và reset để điều khiển động cơ DC:
 - 1 phím để bật/tắt chế độ cho phép nhập mức tốc độ.
 - 3 phím (Mức 1, mức 2, mức 3) tương ứng với 3 led xanh, đỏ, vàng, để nhập mức tốc độ lần lượt với chậm, vừa, nhanh.
 - 1 phím reset để ngắt động cơ (bắt buộc dừng)

- Hiện thị mức tốc độ dùng màn hình LCD 16x2, lập trình ở chế độ 4 bit (sử dụng 4 chân để nhận dữ liệu từ Pic).
- Sử dụng mạch cầu H là IC L298N để điều khiển tốc độ động cơ.
- Sử dụng 2 kênh PWM (CCP1 và CCP2) của vi điều khiển Pic thay đổi giá trị áp trung bình đặt vào module L298 để điều khiển động cơ.
- Đối tượng điều khiển là động cơ DC 12V.
- Ngoài ra trên mạch còn có nguồn (POWER) cấp điện từ adapter cho mạch và 1 phím RESET cho pic 16F877A.
- Để cấp nguồn cho mạch ta dùng adapter AC/DC (220V/12V) và khối nguồn sử dụng IC 7805 để ổn áp điện áp 5V cung cấp cho mạch công suất và Pic.

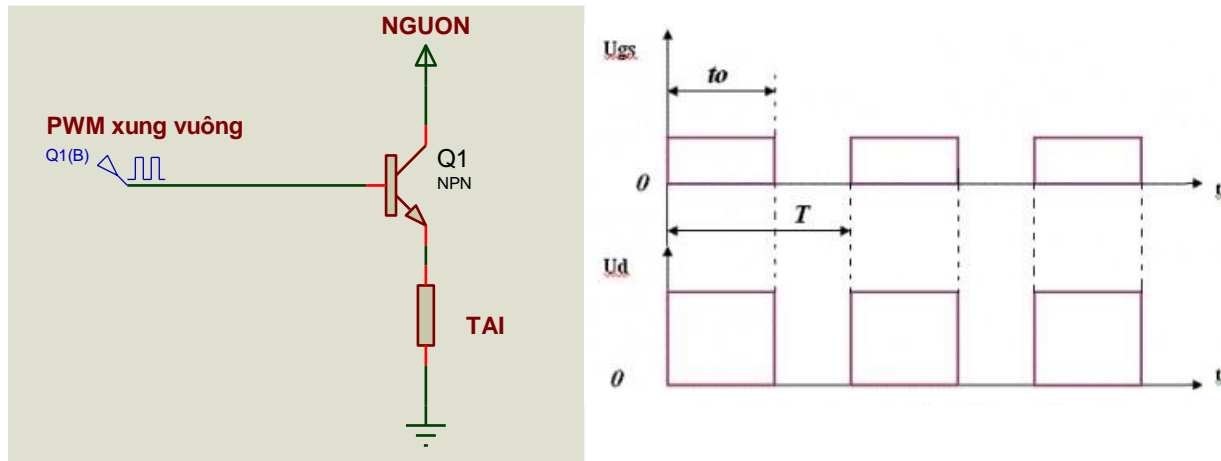
2. Nguyên lý hoạt động

- Để điều khiển tốc độ động cơ DC người ta có thể dùng nhiều phương pháp khác nhau trong đó có một phương pháp quan trọng và thông dụng là phương pháp điều chế độ rộng xung(PWM), có nghĩa là thay đổi độ rộng xung kích để điều khiển linh kiện đóng ngắt (SCR hay Transistor), từ đó điều khiển tốc độ động cơ. Bộ PWM có thể tạo ra từ các linh kiện điện tử.

+ Nguyên lý của PWM:

Đây là phương pháp được thực hiện theo nguyên tắc đóng ngắt nguồn có tải một cách có chu kì theo luật điều chỉnh thời gian đóng ngắt. Phần tử thực hiện nhiệm vụ đóng cắt là các van bán dẫn.

Trong khoảng thời gian 0 - to ta cho van Q1 mở toàn bộ điện áp nguồn U_d được đưa ra tải. Cũng trong khoảng thời gian từ t_0 đến T cho van Q1 khóa, cắt nguồn cung cấp cho tải. Vì vậy với thời gian t_0 thay đổi từ 0 cho đến T ta sẽ cung cấp toàn bộ , một phần hay khóa hoàn toàn điện áp cung cấp cho tải.



Hình 1: Sơ đồ nguyên lý dùng PWM điều khiển điện áp tải (trái)

Sơ đồ xung van điều khiển và đầu ra (phải)

Công thức tính giá trị trung bình của điện áp ra tải là:

$$U_d = U_{\max} \cdot (t_o/T) \text{ hay } U_d = U_{\max} \cdot D$$

Trong đó U_d : là điện áp trung bình ra tải.

U_{\max} : là điện áp nguồn.

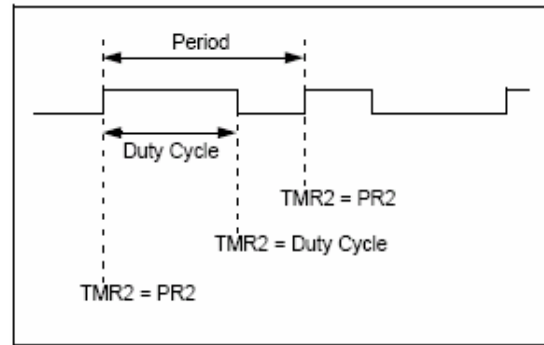
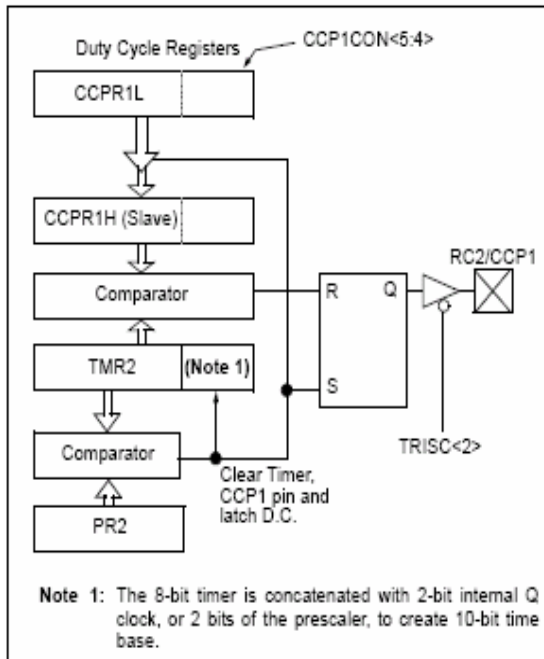
t_o : là thời gian xung ở sườn dương (van khóa mở)

T : thời gian cả thời gian xung sườn dương và sườn âm.

$D = t_o/T$: hệ số điều chỉnh hay PWM được tính bằng %

- Với yêu cầu dùng PIC16F877A điều khiển động cơ quay thuận và thay đổi tốc độ, ta sử dụng bộ điều chế độ rộng xung (PWM) tích hợp sẵn bên trong PIC với 2 ngõ ra xung tại hai chân CCP1(17) và CCP2 (16). Tại các chân này khi hoạt động sẽ xuất chuỗi xung vuông, độ rộng điều chỉnh được dễ dàng. Xung ra này dùng để tạo tín hiệu đóng ngắt Transistor trong mạch động lực, với độ rộng xác định sẽ tạo ra một điện áp trung bình xác định.

⇒ Thay đổi độ rộng xung sẽ thay đổi điện áp trung bình và do đó thay đổi được tốc độ động cơ.



Sơ đồ khối CCP (PWMmode)(trái)

Các tham số của PWM (phải)

Khối PWM gồm có 2 mạch so sánh: mạch so sánh 2 dữ liệu 8 bit nằm bên dưới và mạch so sánh 2 dữ liệu 10 bit nằm bên trên.

Mạch so sánh 8 bit sẽ so sánh giá trị đếm của Timer2 với giá trị thanh ghi PR2 (Period Register), giá trị trong Timer2 tăng từ giá trị đặt trước cho đến khi bằng giá trị của PR2 thì mạch so sánh sẽ kích flip flop RS làm ngõ ra RC2/CCP1 lên mức 1. Đồng thời nạp giá trị 10 bit từ thanh ghi CCPR1L sang thanh ghi CCPR1H.

Timer 2 bị reset và bắt đầu đếm lại cho đến khi giá trị của Timer2 bằng giá trị của CCPR2H thì mạch so sánh sẽ reset flip flop RS làm ngõ ra RC2/CCP1 về mức 0.

⇒ Quá trình này lặp lại liên tục để tạo ra dạng sóng PWM liên tục.

Chu kỳ không thay đổi, muốn thay đổi thời gian xung ở mức 1 thì ta thay đổi hệ số chu kỳ (Duty Cycle). Khi hệ số chu kỳ thay đổi thì điện áp hay dòng trung bình thay đổi

Hệ số chu kỳ càng lớn thì dòng trung bình càng lớn, điều chỉnh hệ số chu kỳ sẽ làm thay đổi tốc độ động cơ.

Cách thiết lập chế độ PWM cho PIC 16F877A

- Khi hoạt động ở chế độ PWM (Pulse Width Modulation _ khối điều chế độ rộng xung), tín hiệu sau khi điều chế sẽ được đưa ra các pin của khối CCP (cần ấn định các pin này là output).
 - Để sử dụng chức năng điều chế này trước tiên ta cần tiến hành các bước cài đặt sau:
 - Thiết lập thời gian của 1 chu kỳ của xung điều chế cho PWM (period) bằng cách đưa giá trị thích hợp vào thanh ghi PR2.
 - Thiết lập độ rộng xung cần điều chế (duty cycle) bằng cách đưa giá trị vào thanh ghi CCPRxL và các bit CCP1CON<5:4>.
 - Điều khiển các pin của CCP là output bằng cách clear các bit tương ứng trong thanh ghi TRISC.
 - Thiết lập giá trị bộ chia tần số prescaler của Timer2 và cho phép Timer2 hoạt động bằng cách đưa giá trị thích hợp vào thanh ghi T2CON.
 - Cho phép CCP hoạt động ở chế độ PWM.
- Khi giá trị thanh ghi PR2 bằng giá trị thanh ghi TMR2 thì quá trình sau xảy ra:
- Thanh ghi TMR2 tự động được xóa.
 - Pin của khối CCP được set.
 - Giá trị thanh ghi CCPR1L (chứa giá trị ấn định độ rộng xung điều chế duty cycle) được đưa vào thanh ghi CCPRxH.

CHƯƠNG II

TÍNH CHỌN LINH KIỆN TRONG HỆ THỐNG

I. Giới thiệu các linh kiện có trong hệ thống

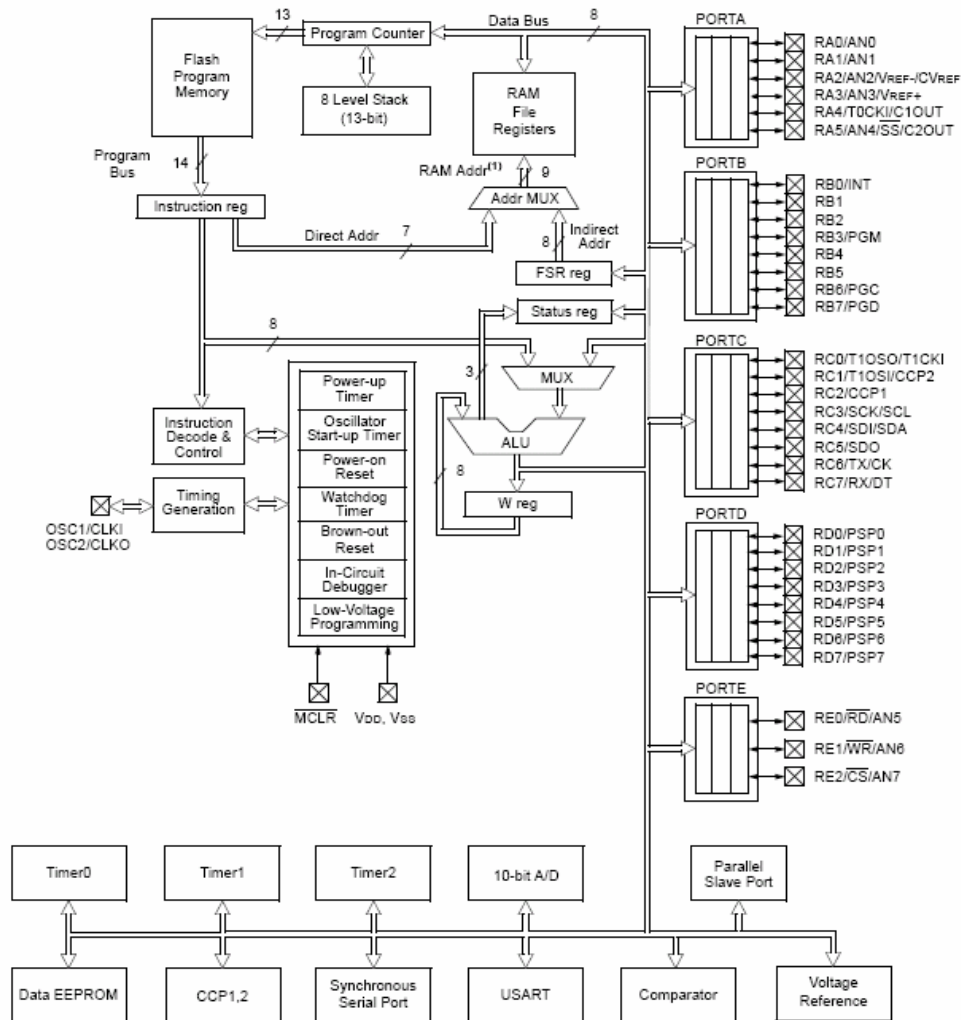
A. Vi điều khiển PIC 16F877A

1. Khái quát về vi điều khiển PIC16F877A

a/ Khái quát:

- PIC là tên viết tắt của “Programmable Intelligent computer” do hãng General Instrument đặt tên cho con vi điều khiển đầu tiên của họ. Hãng Microchip tiếp tục phát triển sản phẩm này và cho đến hàng đã tạo ra gần 100 loại sản phẩm khác nhau.
- PIC16F887A là dòng PIC khá phổ biến, khá đầy đủ tính năng phục vụ cho hầu hết tất cả các ứng dụng thực tế. Đây là dòng PIC khá dễ cho người mới làm quen với PIC có thể học tập và tạo nền tảng về họ vi điều khiển PIC của mình.
- PIC 16F877A thuộc họ vi điều khiển 16Fxxx có các đặc tính sau:
 - Ngôn ngữ lập trình đơn giản với 35 lệnh có độ dài 14 bit.
 - Tất cả các câu lệnh thực hiện trong 1 chu kỳ lệnh ngoại trừ 1 số câu lệnh rẽ nhánh thực hiện trong 2 chu kỳ lệnh. Chu kỳ lệnh bằng 4 lần chu kỳ dao động của thạch anh.
 - Bộ nhớ chương trình Flash 8Kx14 words, với khả năng ghi xoá khoảng 100 ngàn lần.
 - Bộ nhớ Ram 368x8bytes.
 - Bộ nhớ EPROM 256x8 bytes.
 - Khả năng ngắt (lên tới 14 nguồn cả ngắt trong và ngắt ngoài).

- Ngăn nhớ Stack được chia làm 8 mức.
- Truy cập bộ nhớ bằng địa chỉ trực tiếp hoặc gián tiếp.
- Dải điện thế hoạt động rộng: 2.0V đến 5.5V.
- Nguồn sử dụng 25mA.
- Công suất tiêu thụ thấp: <0.6mA với 5V, 4MHz , 20uA với nguồn 3V, 32 kHz.
- Có 3 timer: timer0, 8 bit chức năng định thời và bộ đếm với hệ số tỷ lệ trước. Timer1, 16 bit chức năng bộ định thời, bộ đếm với hệ số tỷ lệ trước, kích hoạt chế độ Sleep. Timer2, 8 bit chức năng định thời và bộ đếm với hệ số tỷ lệ trước và sau.
- Có 2 kênh Capture/ so sánh điện áp (Compare)/điều chế độ rộng xung PWM 10 bit / (CCP).
- Có 8 kênh chuyển đổi ADC 10 bit.
- Cổng truyền thông nối tiếp SSP với SPI phương thức chủ và I²C (chủ/phụ). Bộ truyền nhận thông tin đồng bộ, dị bộ (USART/SCL) có khả năng phát hiện 9 bit địa chỉ.
- Cổng phụ song song (PSP) với 8 bit mở rộng, với RD, WR và CS điều khiển.
- Do thời gian làm đồ án có hạn nên chúng em chỉ tập trung tìm hiểu các tính năng của PIC 16F877A có liên quan đến đề tài, dưới đây là 1 vài tính năng của PIC 16F877A được ứng dụng trong đồ án như:
 - Tổ chức bộ nhớ của PIC 16F877A.
 - Chức năng của các Port I/O.
 - Chức năng và cách thiết lập các tham số của Timer2.
 - Chức năng và cách thiết lập bộ điều chế độ rộng xung PWM.



Hình 2: Sơ đồ nguyên lí PIC

b. Sơ đồ nguyên lí của PIC16F877a

Cấu tạo của vi điều khiển có thể chia làm 2 phần cơ bản như sau:

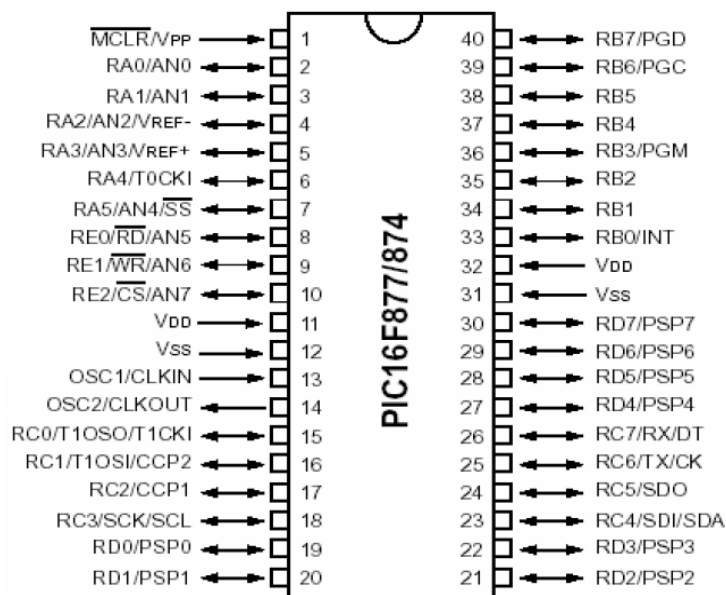
- **Phần lõi:** gồm bộ điều khiển trung tâm có chức năng chạy chương trình (gồm các câu lệnh) đã được nạp vào trong bộ nhớ chương trình (program memory) trước đó.
- **Phần ngoại vi:** gồm có các timer, bộ biến đổi tương tự số ADC và các modun khác. Phần lõi của vi điều khiển chịu trách nhiệm chạy chương trình trong vi điều khiển và quản lý toàn bộ các hoạt động khác bao gồm hoạt động của ngoại vi.

Vi điều khiển chạy chương trình gồm các lệnh trong bộ nhớ chương trình, địa chỉ của lệnh nằm trong thanh ghi bộ đếm chương trình PC, lúc khởi động $PC=0$, sau khi thực hiện một lệnh $PC=PC+1$ do đó vi điều khiển chạy lệnh kế tiếp trong chương trình. Lệnh vi điều khiển trong bộ nhớ thực ra đã được mã hóa mỗi lệnh thành 14 bit.

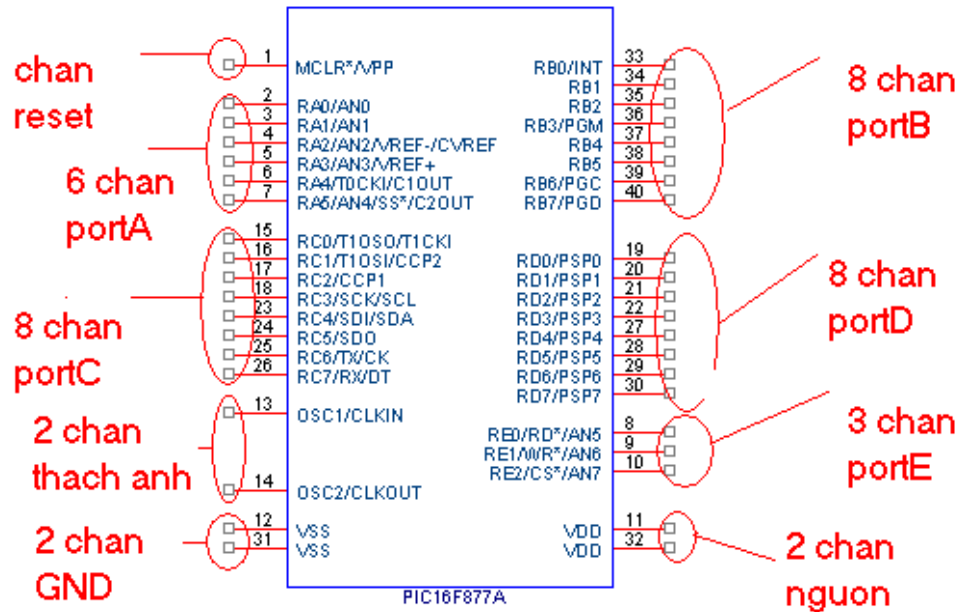
Quá trình thực hiện một lệnh gồm các bước:

- Lệnh trong bộ nhớ chương trình được đưa vào thanh ghi lệnh (địa chỉ của lệnh nằm trong thanh ghi PC).
- Sau đó lệnh đưa vào bộ giải mã và điều khiển để giải mã lệnh. Trên cơ sở đó, vi điều khiển biết lệnh đó là lệnh gì, thao tác với dữ liệu nào, phép thao tác v.v.v Nếu lệnh thao tác với dữ liệu chứa trong các thanh ghi trong RAM, bộ điều khiển điều khiển đọc dữ liệu trong RAM đưa vào bộ xử lý số học và logic ALU, các phép toán sẽ được thực hiện qua trung gian là thanh ghi làm việc W
- Quá trình sẽ kết thúc khi kết quả trả dữ liệu về cho chương trình, tiếp theo PC tăng lên 1 đơn vị, vi điều khiển nhảy đến lệnh kế tiếp, tiếp tục 1 chu kỳ thực hiện.

c. Sơ đồ chân của PIC16F877A



Hình 3: Sơ đồ chân của PIC

d. Sơ đồ nguyên lý các port của PIC16F877A

Hình 4: Sơ đồ nguyên lý các Port của PIC 16F877A

e. Nhận xét:

Từ sơ đồ chân và sơ đồ nguyên lý ở trên, ta rút ra các nhận xét ban đầu như sau :

- PIC16F877A có tất cả 40 chân
- 40 chân trên được chia thành 5 PORT, 2 chân cấp nguồn, 2 chân GND, 2 chân thạch anh và một chân dùng để RESET vi điều khiển.
- 5 port của PIC16F877A bao gồm :
 - + PORT B: 8 chân
 - + PORT D: 8 chân
 - + PORT C: 8 chân
 - + PORT A: 6 chân
 - + PORT E: 3 chân

2. Tổ chức bộ nhớ:

Cấu trúc bộ nhớ của vi điều khiển PIC16F877A bao gồm bộ nhớ chương trình (Program memory) và bộ nhớ dữ liệu (Data Memory).

a. Bộ nhớ chương trình:

Bộ nhớ chương trình của vi điều khiển PIC16F877A là bộ nhớ flash, dung lượng bộ nhớ 8K word (1 word = 14 bit) và được phân thành nhiều trang (từ page0 đến page 3).

Như vậy bộ nhớ chương trình có khả năng chứa được $8 \times 1024 = 8192$ lệnh (vì một lệnh sau khi mã hóa sẽ có dung lượng 1 word (14 bit)).

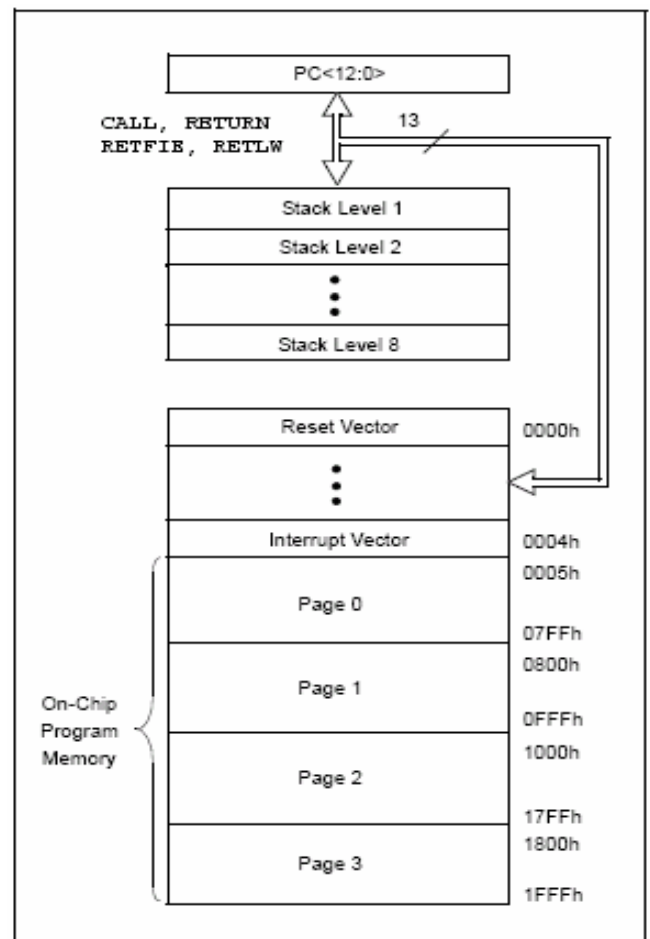
Để mã hóa được địa chỉ của 8K word bộ nhớ

chương trình, bộ đếm chương trình có dung lượng 13 bit (PC<12:0>).

Khi vi điều khiển được reset, bộ đếm chương trình sẽ chỉ đến địa chỉ 0000h (Reset vector). Khi có ngắt xảy ra, bộ đếm chương trình sẽ chỉ đến địa chỉ 0004h (Interrupt vector). Bộ nhớ chương trình không bao gồm bộ nhớ stack và không được địa chỉ hóa bởi bộ đếm chương trình.

b. Bộ nhớ dữ liệu:

- Bộ nhớ dữ liệu của PIC16F877A được chia thành 4 bank. Mỗi bank có dung lượng 128 byte.
- Nếu như 2 bank bộ nhớ dữ liệu của 8051 phân chia riêng biệt : 128 byte đầu tiên thuộc bank1 là vùng Ram nội chỉ để chứa dữ liệu, 128 byte còn lại thuộc bank 2 là cùng các thanh ghi có chức năng đặc biệt SFR mà người dùng không được



Hình 5: Cấu trúc bộ nhớ chương trình PIC 16F877A

chứa dữ liệu khác, còn 4 bank bộ nhớ dữ liệu của PIC16F877A được tổ chức theo cách khác.

- Mỗi bank của bộ nhớ dữ liệu PIC16F877A bao gồm cả các thanh ghi có chức năng đặc biệt SFR nằm ở các ô nhớ địa chỉ thấp và các thanh ghi mục đích dùng chung GPR nằm ở vùng địa chỉ còn lại của mỗi bank thanh ghi. Vùng ô nhớ các thanh ghi mục đích dùng chung này chính là nơi người dùng sẽ lưu dữ liệu trong quá trình viết chương trình. Tất cả các biến dữ liệu nên được khai báo chứa trong vùng địa chỉ này.
- Trong cấu trúc bộ nhớ dữ liệu của PIC16F877A, các thanh ghi SFR nào mà thường xuyên được sử dụng (như thanh ghi STATUS) sẽ được đặt ở tất cả các bank để thuận tiện trong việc truy xuất. Sở dĩ như vậy là vì, để truy xuất một thanh ghi nào đó trong bộ nhớ của 16F877A ta cần phải khai báo đúng bank chứa thanh ghi đó, việc đặt các thanh ghi sử dụng thường xuyên giúp ta thuận tiện hơn rất nhiều trong quá trình truy xuất, làm giảm lệnh chương trình.

Dựa trên sơ đồ 4 bank bộ nhớ dữ liệu PIC16F877A ta rút ra các nhận xét như sau :


- Bank 0 gồm các ô nhớ có địa chỉ từ 00h đến 7Fh, trong đó các thanh ghi dùng chung để chứa dữ liệu của người dùng địa chỉ từ 20h đến 7Fh. Các thanh ghi PORTA, PORTB, PORTC, PORTD, PORTE đều chứa ở bank 0, do đó để truy xuất dữ liệu các thanh ghi này ta phải chuyển đến bank0. Ngoài ra một vài các thanh ghi thông dụng khác (sẽ giới thiệu sau) cũng chứa ở bank0

- Bank 1 gồm các ô nhớ có địa chỉ từ 80h đến FFh. Các thanh ghi dùng chung có địa chỉ từ A0h đến EFh. Các thanh ghi TRISA, TRISB, TRISC, TRISD, TRISE cũng được chứa ở bank 1

- Tương tự ta có thể suy ra các nhận xét cho bank 2 và bank 3 dựa trên sơ đồ trên.

Cũng quan sát trên sơ đồ, ta nhận thấy thanh ghi STATUS, FSR... có mặt trên cả 4 bank. Một điều quan trọng cần nhắc lại trong việc truy xuất dữ liệu của PIC16F877A là : phải khai báo đúng bank chứa thanh ghi đó. Nếu thanh ghi nào mà 4 bank đều chứa thì không cần phải chuyển bank.

File Address		File Address		File Address		File Address	
Indirect addr. ^(*)	00h	Indirect addr. ^(*)	80h	Indirect addr. ^(*)	100h	Indirect addr. ^(*)	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h		105h		185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
PORTC	07h	TRISC	87h		107h		187h
PORTD ⁽¹⁾	08h	TRISD ⁽¹⁾	88h		108h		188h
PORTE ⁽¹⁾	09h	TRISE ⁽¹⁾	89h		109h		189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDATA	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh	Reserved ⁽²⁾	18Eh
TMR1H	0Fh		8Fh	EEADRH	10Fh	Reserved ⁽²⁾	18Fh
T1CON	10h		90h		110h		190h
TMR2	11h	SSPCON2	91h		111h		191h
T2CON	12h	PR2	92h		112h		192h
SSPBUF	13h	SSPADD	93h		113h		193h
SSPCON	14h	SSPSTAT	94h		114h		194h
CCPR1L	15h		95h		115h		195h
CCPR1H	16h		96h		116h		196h
CCP1CON	17h		97h	General Purpose Register 16 Bytes	117h	General Purpose Register 16 Bytes	197h
RCSTA	18h	TXSTA	98h		118h		198h
TXREG	19h	SPBRG	99h		119h		199h
RCREG	1Ah		9Ah		11Ah		19Ah
CCPR2L	1Bh		9Bh		11Bh		19Bh
CCPR2H	1Ch	CMCON	9Ch		11Ch		19Ch
CCP2CON	1Dh	CVRCON	9Dh		11Dh		19Dh
ADRESH	1Eh	ADRESL	9Eh		11Eh		19Eh
ADCON0	1Fh	ADCON1	9Fh		11Fh		19Fh
	20h		A0h		120h		1A0h
General Purpose Register 96 Bytes		General Purpose Register 80 Bytes		General Purpose Register 80 Bytes		General Purpose Register 80 Bytes	
			EFh		16Fh		1EFh
		accesses 70h-7Fh	F0h	accesses 70h-7Fh	170h	accesses 70h - 7Fh	1F0h
			FFh		17Fh		1FFh
Bank 0	7Fh	Bank 1		Bank 2		Bank 3	

 Unimplemented data memory locations, read as '0'.
 * Not a physical register.

Note 1: These registers are not implemented on the PIC16F876A.
Note 2: These registers are reserved; maintain these registers clear.

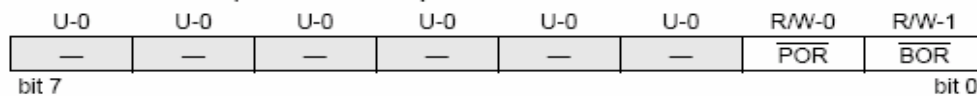
Hình 6: Cấu trúc bộ nhớ dữ liệu của PIC 16F877A

c. Thanh ghi chức năng đặc biệt SFR: (Special Function Register)

- Đây là các thanh ghi được sử dụng bởi CPU hoặc được dùng để thiết lập và điều khiển các khối chức năng được tích hợp bên trong vi điều khiển. Có thể phân thành ghi SFR làm hai loại: thanh ghi SFR liên quan đến các chức năng bên trong (CPU) và thanh ghi SRF dùng để thiết lập và điều khiển các khối chức năng bên ngoài (ví dụ như ADC, PWM, ...).
- Một số thanh ghi chức năng đặc biệt:
 - **Thanh ghi STATUS** (03h, 83h, 103h, 183h): thanh ghi chứa kết quả thực hiện phép toán của khối ALU, trạng thái reset và các bit chọn bank cần truy xuất trong bộ nhớ dữ liệu.
 - **Thanh ghi OPTION_REG** (81h, 181h): thanh ghi này cho phép đọc và ghi, cho phép điều khiển chức năng pull-up của các chân trong PORTB, xác lập các tham số về xung tác động, cạnh tác động của ngắt ngoại vi và bộ đếm Timer0.
 - **Thanh ghi INTCON** (0Bh, 8Bh, 10Bh, 18Bh): thanh ghi cho phép đọc và ghi, chứa các bit điều khiển và các cờ hiệu khi timer0 bị tràn, ngắt ngoại vi RB0/INT và ngắt interrupt-on-change tại các chân của PORTB.
 - **Thanh ghi PIE1** (8Ch): chứa các bit điều khiển chi tiết các ngắt của các khối chức năng ngoại vi.
 - **Thanh ghi PIR1** (0Ch) chứa cờ ngắt của các khối chức năng ngoại vi, các ngắt này được cho phép bởi các bit điều khiển chứa trong thanh ghi PIE1.
 - **Thanh ghi PIE2** (8Dh): chứa các bit điều khiển các ngắt của các khối chức năng CCP2, SSP bus, ngắt của bộ so sánh và ngắt ghi vào bộ nhớ EEPROM.
 - **Thanh ghi PCON** (8Eh): chứa các cờ hiệu cho biết trạng thái các chế độ reset của vi điều khiển.

d. Thanh ghi mục đích chung GPR: (General Purpose Register)

Các thanh ghi này có thể được truy xuất trực tiếp hoặc gián tiếp thông qua thanh ghi FSG (File Select Register). Đây là các thanh ghi dữ liệu thông thường, người sử dụng có thể tùy theo mục đích chương trình mà có thể dùng các thanh ghi này để chứa các biến số, hằng số, kết quả hoặc các tham số phục vụ cho chương trình.



Hình 7: Cấu trúc thanh ghi chức năng chung của PIC 16F877A

e. Stack

Stack không nằm trong bộ nhớ chương trình hay bộ nhớ dữ liệu mà là một vùng nhớ đặc biệt không cho phép đọc hay ghi. Khi lệnh CALL được thực hiện hay khi một ngắt xảy ra làm chương trình bị rẽ nhánh, giá trị của bộ đếm chương trình PC tự động được vi điều khiển cất vào trong stack. Khi một trong các lệnh RETURN, RETLW hay RETFIE được thực thi, giá trị PC sẽ tự động được lấy ra từ trong stack, vi điều khiển sẽ thực hiện tiếp chương trình theo đúng qui trình định trước.

Bộ nhớ Stack trong vi điều khiển PIC họ 16F87xx có khả năng chứa được 8 địa chỉ và hoạt động theo cơ chế xoay vòng. Nghĩa là giá trị cất vào bộ nhớ Stack lần thứ 9 sẽ ghi đè lên giá trị cất vào Stack lần đầu tiên và giá trị cất vào bộ nhớ Stack lần thứ 10 sẽ ghi đè lên giá trị cất vào Stack lần thứ 2.

Cần chú ý là không có cờ hiệu nào cho biết trạng thái stack, do đó ta không biết được khi nào stack tràn. Bên cạnh đó tập lệnh của vi điều khiển dòng PIC cũng không có lệnh POP hay PUSH, các thao tác với bộ nhớ stack sẽ hoàn toàn được điều khiển bởi CPU.

f. khái quát về chức năng của các port trong vi điều khiển PIC16F877A**• PORTA**

- PORTA (RPA) bao gồm 6 I/O pin. Đây là các chân “hai chiều” (bidirectional pin), nghĩa là có thể xuất và nhập được. Chức năng I/O này được điều khiển bởi thanh ghi TRISA (địa chỉ 85h).

Muốn xác lập chức năng của một chân trong PORTA là input, ta “set” bit điều khiển tương ứng với chân đó trong thanh ghi TRISA và ngược lại, muốn xác lập chức năng của một chân trong PORTA là output, ta “clear” bit điều khiển tương ứng với chân đó trong thanh ghi TRISA.

Thao tác này hoàn toàn tương tự đối với các PORT và các thanh ghi điều khiển tương ứng TRIS (đối với PORTA là TRISA, đối với PORTB là TRISB, đối với PORTC là TRISC, đối với PORTD là TRISD và đối với PORTE là TRISE).

Ngoài ra, PORTA còn có các chức năng quan trọng sau :

- Ngõ vào Analog của bộ ADC : thực hiện chức năng chuyển từ Analog sang Digital
- Ngõ vào điện thế so sánh
- Ngõ vào xung Clock của Timer0 trong kiến trúc phần cứng : thực hiện các nhiệm vụ đếm xung thông qua Timer0...
- Ngõ vào của bộ giao tiếp MSSP (Master Synchronous Serial Port)

- Các thanh ghi SFR liên quan đến PORTA bao gồm:

PORTA (địa chỉ 05h) : chứa giá trị các pin trong PORTA.

TRISA (địa chỉ 85h) : điều khiển xuất nhập.

CMCON (địa chỉ 9Ch) : thanh ghi điều khiển bộ so sánh.

CVRCON (địa chỉ 9Dh) : thanh ghi điều khiển bộ so sánh điện áp.

ADCON1 (địa chỉ 9Fh) : thanh ghi điều khiển bộ ADC.

- **PORTB**

- PORTB (RPB) gồm 8 pin I/O. Thanh ghi điều khiển xuất nhập tương ứng là TRISB.
- Bên cạnh đó một số chân của PORTB còn được sử dụng trong quá trình nạp chương trình cho vi điều khiển với các chế độ nạp khác nhau. PORTB còn liên quan đến ngắt ngoại vi và bộ Timer0. PORTB còn được tích hợp chức năng điện trở kéo lên được điều khiển bởi chương trình.
- Các thanh ghi SFR liên quan đến PORTB bao gồm:
 - PORTB (địa chỉ 06h, 106h) : Chứa giá trị các pin trong PORTB
 - TRISB (địa chỉ 86h, 186h) : Điều khiển xuất nhập
 - OPTION_REG (địa chỉ 81h, 181h): điều khiển ngắt ngoại vi và bộ Timer0.

- **PORTC**

PORTC có 8 chân và cũng thực hiện được 2 chức năng input và output dưới sự điều khiển của thanh ghi TRISC tương tự như hai thanh ghi trên.

Ngoài ra PORTC còn có các chức năng quan trọng sau :

- Ngõ vào xung clock cho Timer1 trong kiến trúc phần cứng
- Bộ PWM thực hiện chức năng điều xung lập trình được tần số, duty cycle: sử dụng trong điều khiển tốc độ và vị trí của động cơ v.v....
- Tích hợp các bộ giao tiếp nối tiếp I2C, SPI, SSP, USART

- **PORTD**

- PORTD có 8 chân. Thanh ghi TRISD điều khiển 2 chức năng input và output của PORTD tương tự như trên. PORTD cũng là cổng xuất dữ liệu của chuẩn giao tiếp song song PSP (Parallel Slave Port).

- Các thanh ghi liên quan đến PORTD bao gồm:
 - + Thanh ghi PORTD: Chứa giá trị các pin trong PORTD.
 - + Thanh ghi TRISD: Điều khiển xuất nhập.
 - + Thanh ghi TRISE: Điều khiển xuất nhập PORTE và chuẩn giao tiếp PSP.

- **PORTE**

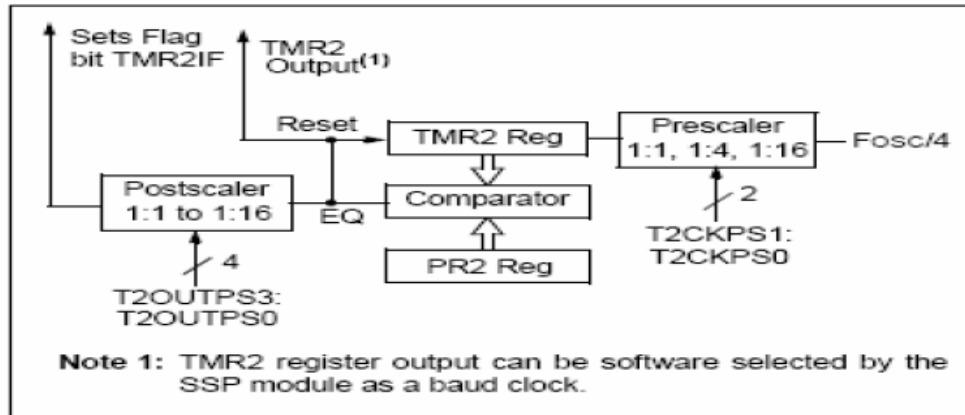
- PORTE có 3 chân, thanh ghi điều khiển xuất nhập tương ứng là TRISE. Các chân của PORTE có ngõ vào analog. Bên cạnh đó PORTE còn là các chân điều khiển của chuẩn giao tiếp PSP.

- Các thanh ghi liên quan đến PORTE bao gồm:
 - + PORTE: chứa giá trị các chân trong PORTE.
 - + TRISE: điều khiển xuất nhập và xác lập các thông số cho chuẩn giao tiếp PSP.
 - + ADCON1: thanh ghi điều khiển khởi ADC.

3. Các vấn đề về Timer

PIC16F877A có tất cả 3 timer : timer0 (8 bit), timer1 (16 bit) và timer2 (8 bit). Nhưng theo yêu cầu của đề tài nên em chỉ dùng Timer 2 để điều khiển tốc độ động cơ DC.

1/ Timer2: là bộ định thời 8 bit bao gồm một bộ tiền định (prescaler), một bộ hậu định (Postscaler) và một thanh ghi chu kỳ viết tắt là PR2. Việc kết hợp timer2 với 2 bộ định tỉ lệ cho phép nó hoạt động như một bộ định thời 16 bit. Module timer2 cung cấp thời gian hoạt động cho chế độ điều biến xung PWM nếu module CCP được chọn.



Hình 8: Sơ đồ khối Timer2

b/ Hoạt động của bộ Timer2

- Timer2 được dùng chủ yếu ở phần điều chế xung của bộ CCP, thanh ghi TMR2 có khả năng đọc và viết, nó có thể xóa bằng việc reset lại thiết bị. Đầu vào của xung có thể chọn các tỷ số sau; 1:1; 1:4 hoặc 1:16 việc lựa chọn các tỷ số này có thể điều khiển bằng các bit sau T2CKPS1 và bit T2CKPS0.
- Bộ Timer2 có 1 thanh ghi 8 bit PR2 . Timer 2 tăng từ giá trị 00h cho đến khớp với PR2 và tiếp theo nó sẽ reset lại giá trị 00h và lệnh kế tiếp thực hiện. Thanh ghi PR2 là một thanh ghi có khả năng đọc và khả năng viết.

c/ Thanh ghi T2CON: điều khiển hoạt động của Timer2

T2CON: TIMER2 CONTROL REGISTER (ADDRESS 12h)

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
bit 7							bit 0

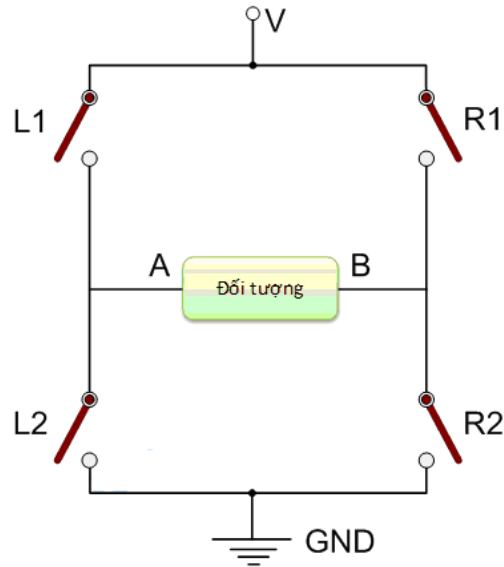
Hình 9: Cấu trúc thanh ghi T2CON điều khiển hoạt động của Timer2

- bit 7 không sử dụng
- bit 6-3 TOUTPS3: TOUTPS0 bit lựa chọn hệ số đầu ra Timer 2
 - 0000=1:1
 - 0001=1:2
 - 0010=1:3
 - ...
 - 1111=1:16
- bit 2 TMR2ON bit bật tắt hoạt động Timer 2
 - 1 = enable
 - 0 = disable
- bit 1- 0 T2CKPS1:T2CKPS0 chọn hệ chia đầu vào
 - 00 = 1:1
 - 01 = 1:4
 - 1x = 1:16

B. Mạch cầu H

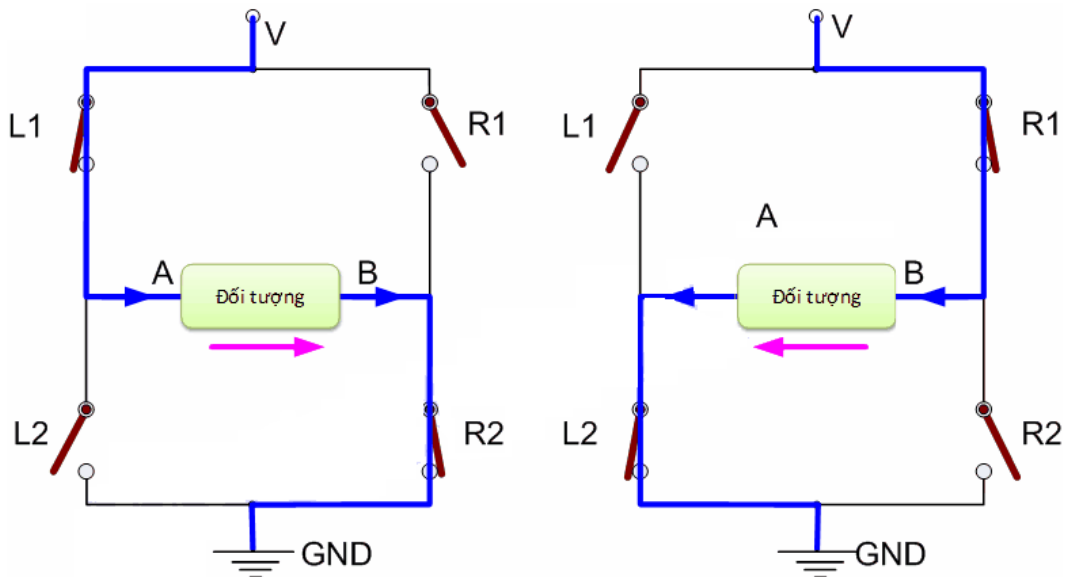
1/ Công dụng và nguyên lí hoạt động:

- Mạch cầu H là một mạch điện giúp đảo chiều dòng điện qua một đối tượng. Đối tượng là động cơ DC mà chúng ta cần điều khiển. Mục đích điều khiển là cho phép dòng điện qua đối tượng theo chiều A đến B hoặc B đến A. Từ đó giúp đổi chiều quay của động cơ.
- Hiện nay, ngoài loại mạch cầu H được thiết kế từ các linh kiện rời như: BJT công suất, Mosfet, ... Còn có các loại mạch cầu H được tích hợp thành các IC như: L293D và L298D. Do đối tượng điều khiển trong đề tài này là động cơ DC có điện áp 12V và công suất nhỏ nên chúng em dùng mạch cầu H đảo chiều động cơ là IC L298.



Hình 10: Mạch cầu H

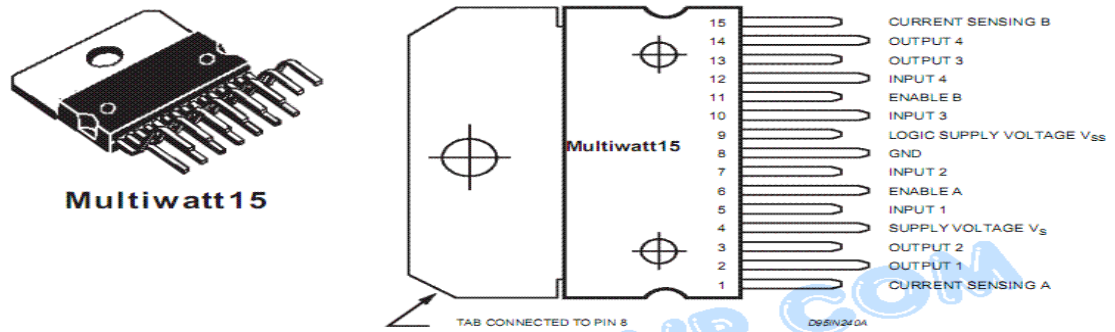
Khảo sát hoạt động của mạch cầu H



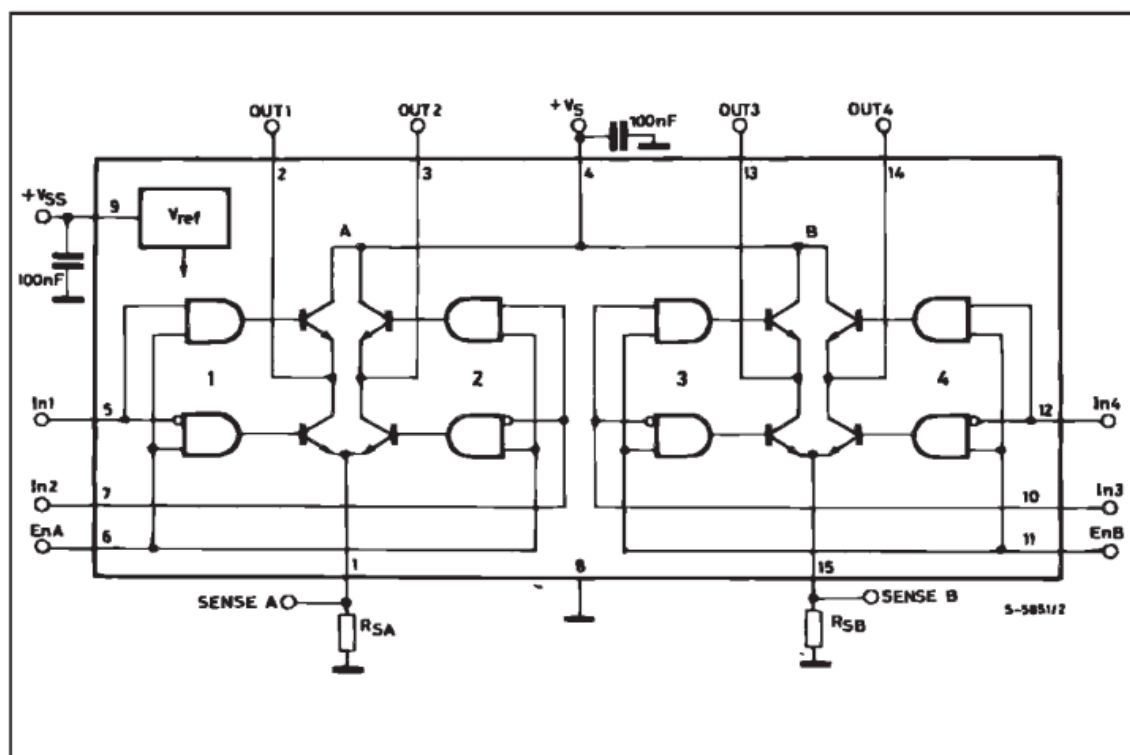
Hình 11: Nguyên lý hoạt động của mạch cầu H

2/ Mạch cầu H L298D

L298D là một chip tổng hợp 2 mạch trong gói 15 chân. L298D có điện áp danh nghĩa cao (lớn hơn 50V) và dòng điện danh nghĩa lớn hơn 2A nên rất thích hợp cho các ứng dụng công suất nhỏ như các động cơ DC loại vừa và nhỏ.



Hình 12: Sơ đồ chân của IC L298D (phải)



Hình 13: Sơ đồ nguyên lí của IC L298D

Có 2 mạch cầu H trên mỗi chip L298D nên có thể điều khiển 2 đối tượng riêng với 1 chip này. Mỗi mạch cầu H bao gồm 1 đường nguồn V_S (thật ra là đường chung cho 2 mạch cầu), một chân current sensing (cảm biến dòng) ở phần cuối của mạch cầu H, chân này không được nối đất mà bỏ trống để cho người dùng nối 1 điện trở nhỏ gọi là sensing resistor.

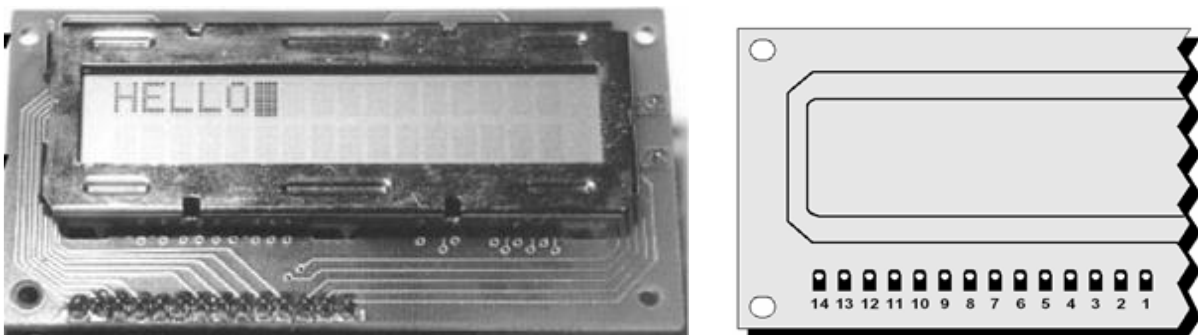
Bằng cách đo điện áp rơi trên điện trở này chúng ta có thể tính được dòng qua điện trở, cũng là dòng qua động cơ, mục đích của việc này là để xác định dòng quá tải. Nếu việc đo lường là không cần thiết thì ta có thể nối chân này với GND. Động cơ sẽ được nối với 2 chân OUT1, OUT2 hoặc OUT3, OUT4. Chân EN (ENA và ENB) cho phép mạch cầu hoạt động, khi chân này được kéo lên mức cao.

L298D không chỉ được dùng để đảo chiều động cơ mà còn điều khiển vận tốc động cơ bằng PWM. Trong thực tế, công suất thực mà L298D có thể tải nhỏ hơn giá trị danh nghĩa của nó ($U = 50V$, $I = 2A$).

Để tăng dòng tải của chip lên gấp đôi, chúng ta có thể nối hai mạch cầu H song song với nhau (các chân có chức năng như nhau của 2 mạch cầu được nối chung).

C. Hiển thị LCD

Ngày nay, thiết bị hiển thị LCD (Liquid Crystal Display) được sử dụng trong rất nhiều các ứng dụng của vi điều khiển. LCD có rất nhiều ưu điểm so với các dạng hiển thị khác: có khả năng hiển thị kí tự đa dạng, trực quan (chữ, số và kí tự đồ họa), dễ dàng đưa vào mạch ứng dụng theo nhiều giao thức giao tiếp khác nhau, tốn rất ít tài nguyên hệ thống và giá thành rẻ ...



Hình 14: LCD và sơ đồ chân

1. Chức năng các chân

Bảng Chức năng các chân của LCD

Chân số	Tên	Chức năng
1	V_{SS}	Chân nối đất cho LCD, khi thiết kế mạch ta nối chân này với GND của mạch điều khiển
2	V_{DD}	Chân cấp nguồn cho LCD, khi thiết kế mạch ta nối chân này với $V_{CC}=5V$ của mạch điều khiển
3	V_{ee}	Chân này dùng để điều chỉnh độ tương phản của LCD.
4	RS	Chân chọn thanh ghi (Register select). Nối chân RS với logic “0” (GND) hoặc logic “1” (V_{CC}) để chọn thanh ghi. + Logic “0”: Bus DB0-DB7 sẽ nối với thanh ghi lệnh IR của LCD (ở chế độ “ghi” - write) hoặc nối với bộ đếm địa chỉ của LCD (ở chế độ “đọc” - read) . + Logic “1”: Bus DB0-DB7 sẽ nối với thanh ghi dữ liệu DR bên trong LCD
5	R/W	Chân chọn chế độ đọc/ghi (Read/Write). Nối chân R/W với logic “0” để LCD hoạt động ở chế độ ghi, hoặc nối với logic “1” để LCD ở chế độ đọc.
6	E	Chân cho phép (Enable). Sau khi các tín hiệu được đặt lên bus DB0-DB7, các lệnh chỉ được chấp nhận khi có 1 xung cho phép của chân E. + Ở chế độ ghi: Dữ liệu ở bus sẽ được LCD chuyển vào (chấp nhận) thanh ghi bên trong khi phát hiện một xung (high-to-low transition) của tín hiệu chân E. + Ở chế độ đọc: Dữ liệu sẽ được LCD xuất ra DB0-DB7 khi phát hiện cạnh lên (low-to-high transition) ở chân E và được LCD giữ ở bus đến khi nào chân E xuống mức thấp.
7-14	DB0-DB7	Tám đường của bus dữ liệu dùng để trao đổi thông tin với MPU. Có 2 chế độ sử dụng 8 đường bus này : + Chế độ 8 bit: Dữ liệu được truyền trên cả 8 đường, với bit MSB là bit DB7. + Chế độ 4 bit: Dữ liệu được truyền trên 4 đường từ DB4 tới DB7, bit MSB là DB7.

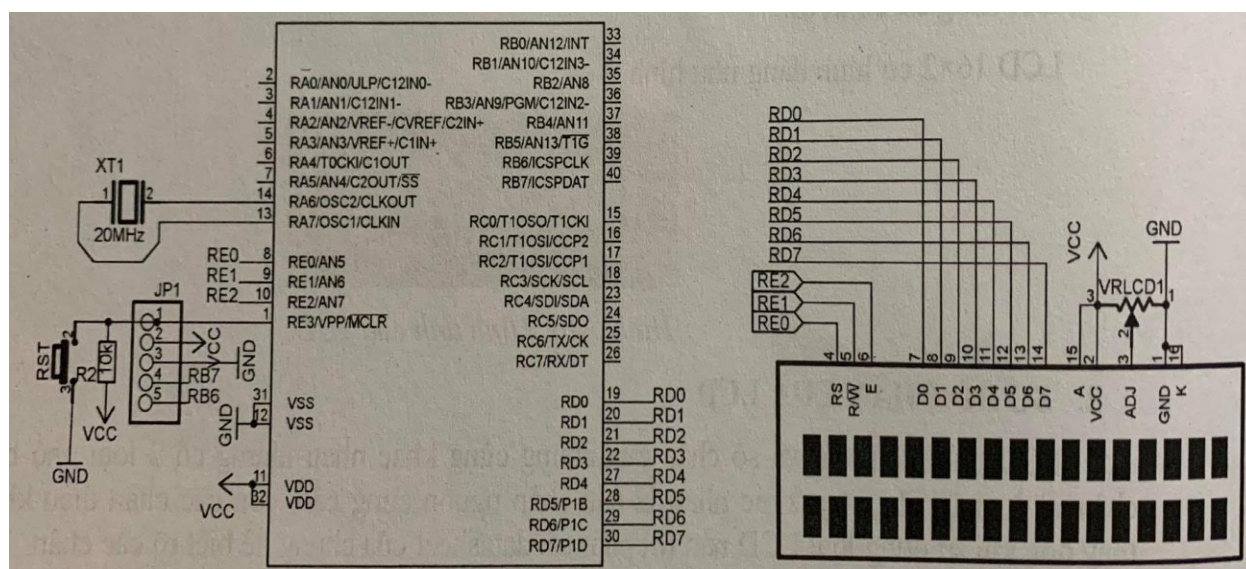
* Ghi chú: Ở chế độ “đọc”, nghĩa là MPU sẽ đọc thông tin từ LCD thông qua các chân DBx. Còn khi ở chế độ “ghi”, nghĩa là MPU xuất thông tin điều khiển cho LCD thông qua các chân DBx.

2. Đặc tính điện của các chân giao tiếp:

LCD sẽ bị hỏng nghiêm trọng, hoặc hoạt động sai lệch nếu bạn vi phạm khoảng đặc tính điện sau đây:

Chân cấp nguồn (Vcc-GND)	Min:-0.3V , Max+7V
Các chân ngõ vào (DBx, E, ...)	Min:-0.3V , Max:(Vcc+0.3V)
Nhiệt độ hoạt động	Min:-30C , Max:+75C
Nhiệt độ bảo quản	Min:-55C , Max:+125C

3 Sơ đồ mạch giao tiếp vi điều khiển và PIC



(Cắt từ sách “Vi điều khiển PIC”, tác giả Nguyễn Đình Phú)

D. Đối tượng điều khiển: Động cơ DC

Đây là động cơ được sử dụng trong đề tài:



Hình 15: Động cơ DC

- Các thông số của động cơ như sau:
 - + Điện áp DC cấp cho động cơ: 12VDC
 - + Tốc độ tối đa 2000 vòng/phú
 - + Điện cảm $L=102\text{mH}$
- Động cơ có tất cả 2 dây lấy nguồn cung cấp 12 V cho động cơ
- Phương pháp điều khiển: Thay đổi tốc độ động cơ bằng cách thay đổi áp cấp vào cho động cơ.

II. Tính toán linh kiện trong hệ thống

A. Tính toán khối điều chế độ rộng xung PWM

- Giá trị 1 chu kì (period) của xung điều chế được tính bằng công thức:

$$\text{PWM period} = [(PR2) + 1] * 4 * T_{osc} * (\text{giá trị bộ chia tần số của TMR2}).$$

- Độ rộng của xung điều chế hay còn gọi là hệ số chu kỳ (duty cycle) được thiết lập giá trị trong thanh ghi 10 bit được tính theo công thức:

$$\text{PWM duty cycle} = (CCPRxL:CCPxCON<5:4>) * T_{osc} * (\text{giá trị bộ chia tần số TMR2})$$

Trong đó: T_{osc} là chu kỳ của thạch anh, PR2 có giá trị 8 bit từ 0 – 255, giá trị bộ chia tần số TMR2 chọn giữa 1,4 và 16. Ta chọn giá trị lớn nhất là 16, tính được PR2.

- Vì giá trị hệ số chu kỳ là 10 bit nên có thể thay đổi từ 0 – 1023 tạo ra 1024 cấp giá trị điều khiển. Ở đây em chọn 4 cấp là 0, 400, 700 và 1000

B. Chọn diode cho mạch công suất

- Khi tải hoạt động và ngừng lại, thì do tải mang tính cảm nên trong đó sẽ lưu trữ 1 dòng điện, vì vậy khi ngừng hoạt động tải cảm này sẽ sinh ra 1 dòng ngược, dòng ngược này nguy hiểm cho các Transistor nên cần phải dập đi.
- Ta sử dụng 4 Diode 1N4007 đó để dập xung do tải gây ra (tải cảm thường là motor - động cơ).

1N4007 Electrical Characteristics		
Parameter	Values	Units
Forward voltage (V_F) at 1.0A	1.1	V
Reverse current at 25°C	5	μA
Total capacitance at 1.0 MHz	15	pF
Maximum full load reverse current at 75 °C	30	μA
Average rectified forward current ($I_{F(AV)}$)	1	A
Peak repetitive reverse voltage	1000	V

Hình: Đặc tính điện của diode 1N4007

C. Chọn điện trở

- R5 nối với chân MCLR: Điện áp input là 5V, dòng cho phép trên mỗi chân I/O của vi điều khiển thường chỉ nằm trong khoảng 10-20mA

$$I = V / R$$

$$I = 5V / 10000 \text{ Ohms}$$

$$I = 0.0005A \text{ (0.5mA)}$$

Như vậy khi nhấn nút thì dòng 0.5mA này sẽ đi xuống mass và chân input sẽ được nối mass, đảm bảo an toàn cho vi điều khiển.

Giá trị điện trở này trong các mạch số thường là 4k7 hoặc 10k. Đề tài này em chọn 10k. Tương tự cho R1, R2, R3, R4

- R7, R8, R9 nối với đèn led và 3 chân thuộc PORTE: Mỗi led cần điện áp khoảng 2.4 – 5V và dòng khoảng 15-20 mA để hoạt động. Khi nối với nguồn 5V thì sẽ cần 1 điện trở khoảng 330-1000 ohm để hạn dòng.

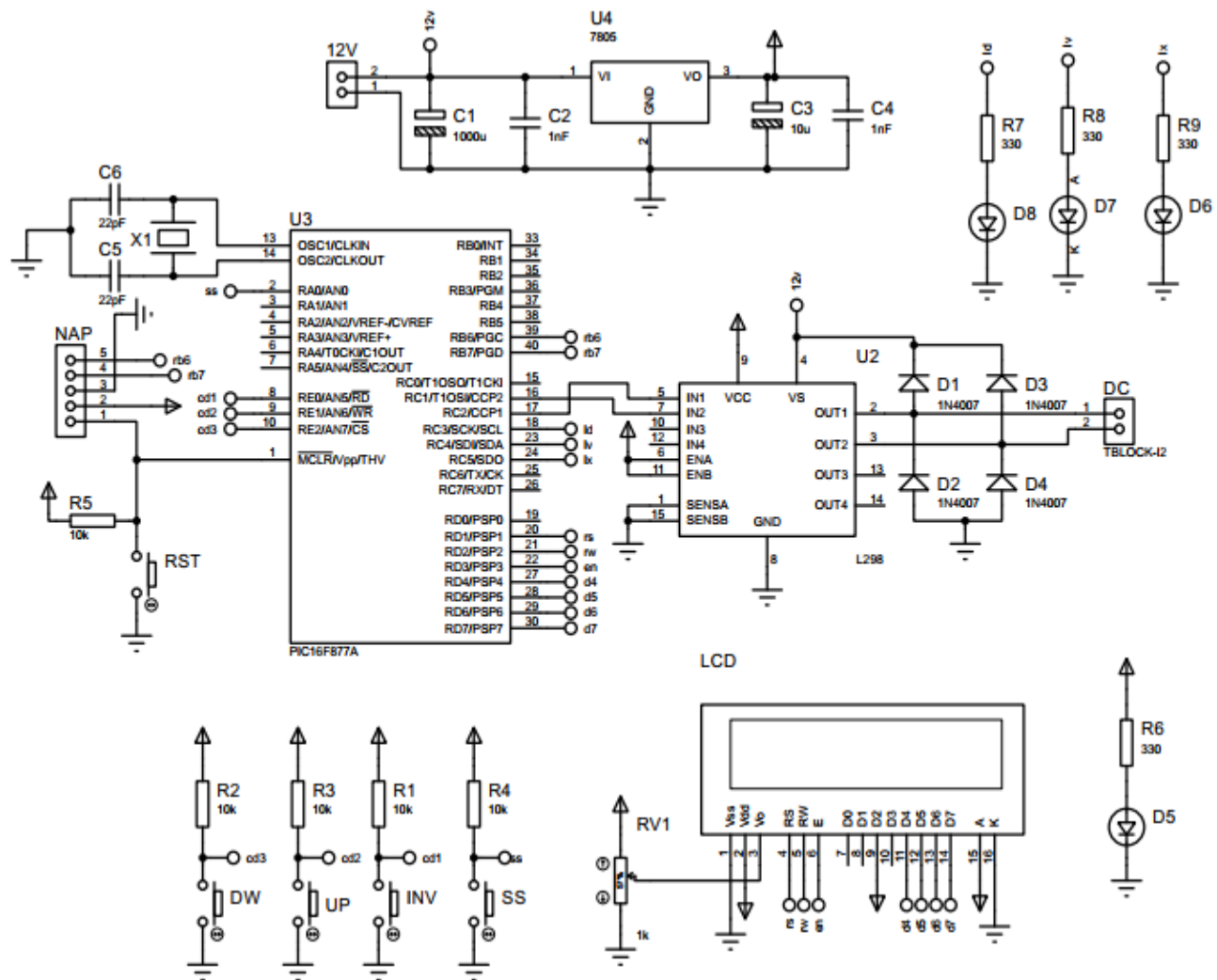
Giả sử led dùng điện áp 2.4V và dòng 15mA thì điện trở cần là $\frac{5-2.4}{0.015} = 180$

Nhưng để đảm bảo cho PIC không bị tổn hao điện áp khi nối với led quá nhiều thì em chọn điện trở = 330 ohm để hạn dòng

CHƯƠNG III

MÔ PHỎNG HỆ THỐNG

I. Thiết kế mạch nguyên lý

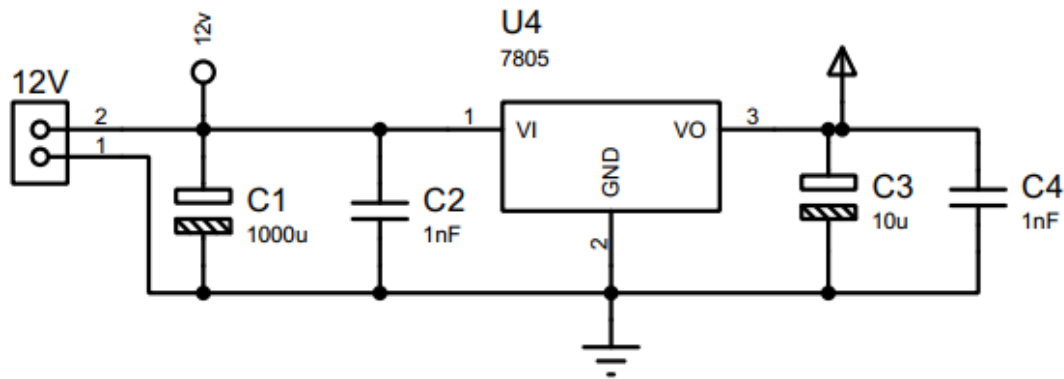


Hình 16: Mạch nguyên lý hệ thống điều khiển tốc độ động cơ 1 chiều

(Chi tiết các phần sẽ được giải thích qua các mục bên dưới)

A. Khối nguồn

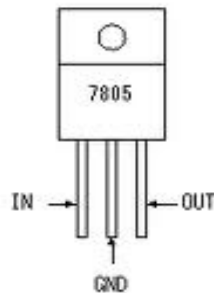
Mạch lấy nguồn xoay chiều qua adapter AC/DC 220VAC/12VDC, và được ổn áp nhờ IC 7805. Sơ đồ nguyên lý mạch:



Hình 17: Khối mạch ổn áp

Chức năng của các phần tử trong mạch:

- IC 7805: chức năng ổn áp điện áp 5V

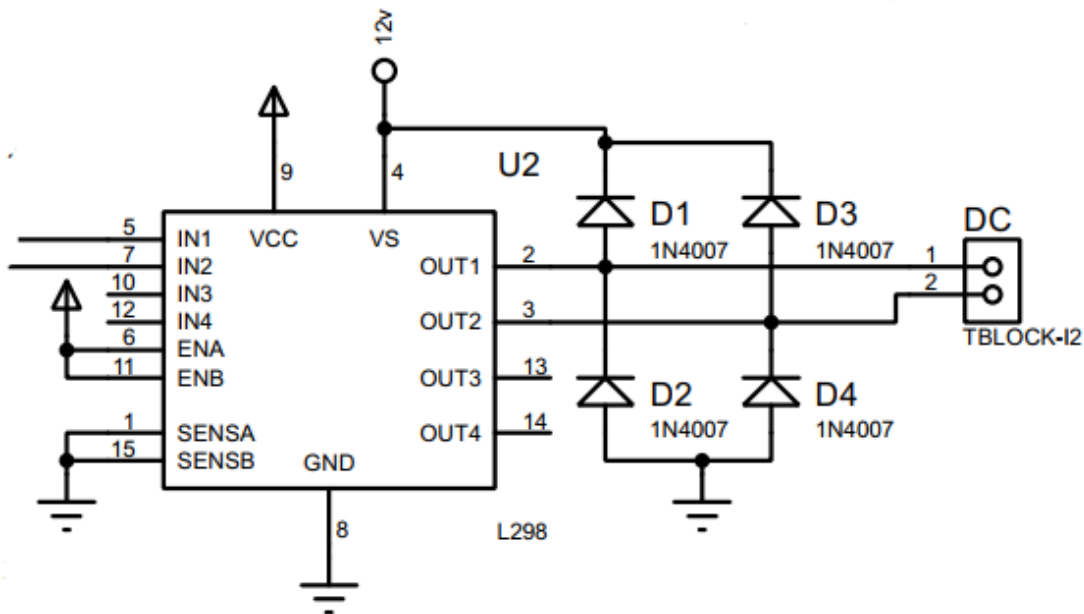


Hình 18: IC 7805

- C1 tụ hóa (có phân cực) ổn áp ngõ vào, điện dung của tụ này càng lớn thì điện áp vào IC 7805 càng phẳng.
- C2 và C4 tụ giấy (không phân cực) là hai tụ lọc nhiễu tần số cao ở ngõ vào và ngõ ra.
- C3 tụ hóa có tác dụng dập dao động tự kích khi sử dụng IC ổn áp dòng 78xx.

B. Khối mạch công suất

Mạch công suất sử dụng IC cầu H L298, với 2 kênh A và B, mỗi kênh với điện áp định mức 50V và dòng định mức cho tải là 2A. Khi đấu song song 2 kênh ta được dòng cấp cho tải lên đến 4A (gấp đôi). Điện áp điều khiển 5V.



Cầu Diode dùng để chống dòng điện ngược, do tải động cơ có tính chất cảm kháng. Nguồn cấp cho động cơ 12V. Sử dụng IC cầu H này, không những dùng để đảo chiều động cơ mà còn điều khiển tốc độ động cơ bằng phương pháp băm xung (PWM).

Chức năng các chân sử dụng cho khối công suất:

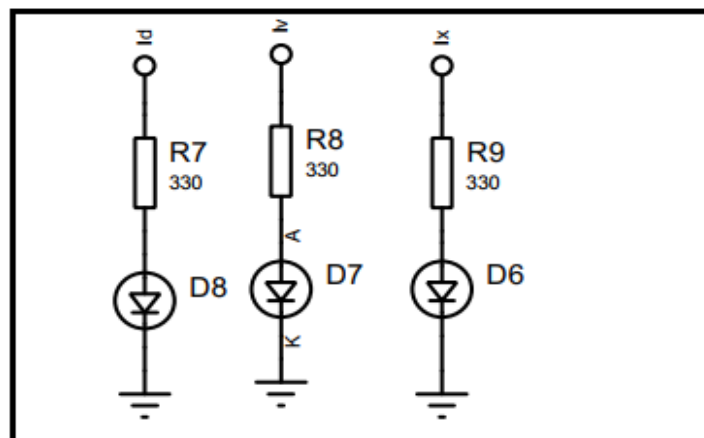
- Ta sử dụng 2 chân RC1(CCP2) và RC2 (CCP1) của Port C để xuất tín hiệu PWM điều khiển động cơ.
- Chân RC1/T1OSI/CCP2 (16) : có 3 chức năng
 RC1: xuất/ nhập số - bit thứ 1 của port C.
 T1OSI: ngõ vào của bộ dao động Timer 1.
 CCP2: ngõ vào capture2, ngõ ra compare2, ngõ ra PWM2

- Chân RC2/P1A/CCP1 (17): có 3 chức năng
 RC2: xuất/nhập số - bit thứ 2 của port C.
 P1A: ngõ ra PWM
 CCP1: ngõ vào capture1, ngõ ra compare2, ngõ ra PWM1

Các chức năng của thành phần trong mạch công suất:

- Cầu Diode (D1,D2,D3,D4) dùng để chống dòng điện ngược, do tải động cơ có tính chất cảm cấp cho động cơ 12V
- Sử dụng IC cầu H này, không những dùng để đảo chiều động cơ mà còn điều khiển tốc độ động cơ bằng phương pháp băm xung (PWM).

Chức năng các chân sử dụng cho 3 LED thể hiện tốc độ động cơ :



- ta dùng 3 chân RC3(ld), RC4(lv), RC5(lx) để điều khiển 3 led hiển thị tốc độ động cơ
- Chân RC3/SCK/SCL (18) (ld) : có 3 chức năng
 RC3: xuất/nhập số - bit thứ 3 của port C.
 SCK: ngõ vào xung clock nối tiếp đồng bộ/ngõ ra của chế độ SPI.
 SCL: ngõ vào xung clock nối tiếp đồng bộ/ngõ ra của chế độ I² C
- Chân RC4/SDI/SDA (23) (lv) : có 3 chức năng
 RC4: xuất/nhập số - bit thứ 4 của port C.
 SDI: ngõ vào dữ liệu trong truyền dữ liệu kiểu SPI

SDA: xuất/nhập dữ liệu I² C

- Chân RC5/SDO (24) (1x) : có 2 chức năng

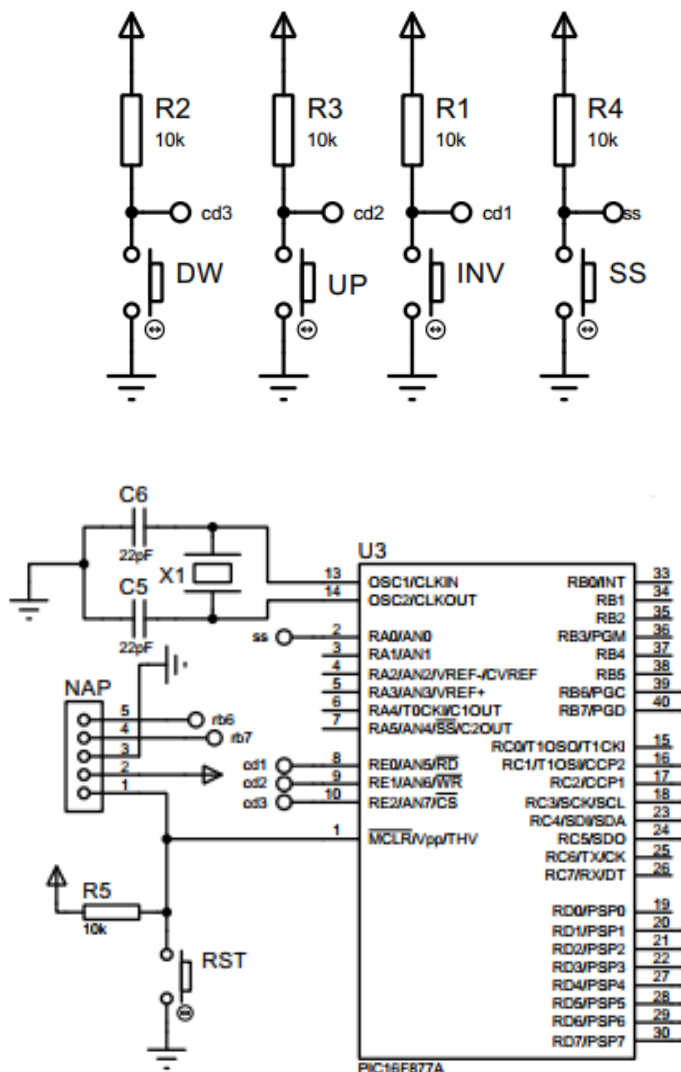
RC5: xuất/nhập số - bit thứ 5 của port C.

SDO: ngõ xuất dữ liệu trong truyền dữ liệu kiểu SPI.

Các chức năng của thành phần trong 3 LED thể hiện tốc độ động cơ

- 3 điện trở (R7,R8,R9) là 3 điện trở hạn dòng bảo vệ LED
- D6, D7, D8 lần lượt là led xanh, led đỏ, led vàng

C. Khối mạch phím



1. Chân MCLR/Vpp/THV:

- MCLR :
 - Là chân reset ở mức thấp
 - Phục vụ cho việc reset pic về địa chỉ 0000h
- Vpp :
 - Khi lập trình cho Pic thì đóng vai trò là ngõ vào nhận điện áp lập trình

2. Chân RA0/AN0 (tích cực logic 0): Nối với nút “ in1-SS ”, cho phép bật/tắt chống trôi nút nhấn

- RA0 :
 - Xuất/Nhập số : Nếu = 0 - Output ; = 1 – Input
- AN0 :
 - Ngõ vào tương tự của kênh thứ 0

3. Chân RE0/AN5/RD (tích cực logic 0): Nối với nút “ in2-INV ”, cho phép động cơ chạy với cấp độ 1

- RE0 :
 - Xuất/Nhập số : Nếu = 0 – Output , = 1 – Input
- AN5 :
 - Ngõ vào tương tự của kênh thứ 5
- RD :
 - Điều khiển đọc port tứ song song , đọc dữ liệu song song từ các ngoại vi

4. Chân RE1/WR/AN6 : Nối với nút “ in3-UP ”, cho phép động cơ chạy với cấp độ 2

- RE1 :
 - Xuất/Nhập số : Nếu = 0 – Output , = 1 – Input
- AN6 :
 - Ngõ vào tương tự kênh thứ 6
- WR :
 - Cho phép ghi dữ liệu song song từ các ngoại vi

5. Chân RE2/CS/AN7 (tích cực logic 0): Nối với nút “ in4-DW ”, cho phép động cơ chạy với cấp độ 3

- RE2 :
 - Xuất/Nhập số : Nếu = 0 – Output , = 1 – Input
- AN7 :
 - Ngõ vào tương tự của kênh thứ 7
- CS :

- Chip chọn lựa điều khiển port tứ song song từ các ngoại vi

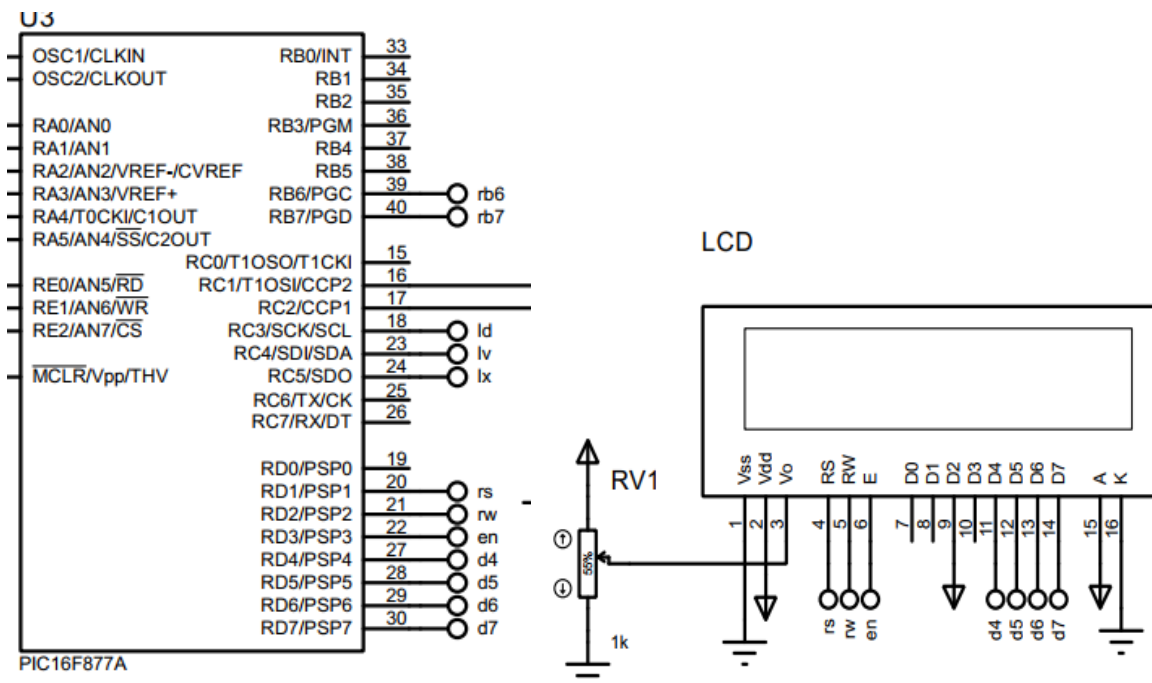
6. Chân RB6/PGC :

- RB6 :
 - Xuất/Nhập số : Nếu = 0 – Output , = 1 – Input
 - Có tính phát động ngắt theo sự thay đổi trạng thái trên chân này. Nó còn có thể lập trình để dùng chân này phát xung nhịp (serial clock) dùng cho truyền bit dạng nối tiếp
- PGC :
 - Mạch gỡ rối và xung clock lập trình ICSP

7. Chân RB7/PGD:

- RB7 :
 - Xuất/Nhập số : Nếu = 0 – Output , = 1 – Input
 - Phát động ngắt theo sự thay đổi trạng thái trên chân này. Nó còn có thể lập trình để dùng chân này trao đổi dữ liệu (serial data) dùng cho truyền bit dạng nối tiếp
- PGD :
 - Mạch gỡ rối dữ liệu lập trình ICSP

D. Khối mạch hiển thị



Mạch hiển thị bao gồm màn hình LCD giao tiếp với PIC qua Port D với giao thức 4 bit, ngoài ra còn có biến trở để điều chỉnh độ sáng của LCD.

1. *Chân RD1/PSP1(20) nối với RS(4) của LCD:*

RD1: xuất/ nhập số, bit thứ 1 của PORT D.

2. *Chân RD2/PSP2(21) nối với RW(5) của LCD:*

RD2: xuất/ nhập số, bit thứ 2 của PORT D.

3. *Chân RD3/PSP3(22) nối với E(6) của LCD:*

RD3: xuất/ nhập số, bit thứ 3 của PORT D.

4. *Chân RD4/PSP4(27) nối với D4(11) của LCD:*

RD4: xuất/ nhập số, bit thứ 4 của PORT D.

5. *Chân RD5/P1B(28) nối với D5(12) của LCD:*

- RD5: xuất/ nhập số, bit thứ 5 của PORT D.
- P1B: ngõ ra PWM.

6. *Chân RD6/P1C(29) nối với D6(13) của LCD:*

- RD6: xuất/ nhập số, bit thứ 6 của PORT D.
- P1C: ngõ ra PWM.

7. *Chân RD7/P1D(30) nối với D6(14) của LCD:*

- RD7: xuất/ nhập số, bit thứ 7 của PORT D.
- P1D: ngõ ra tăng cường CPP1

Với :

RS: là trở thanh ghi (bao gồm IR thanh ghi lệnh và RD thanh ghi số liệu)

RW: read – write

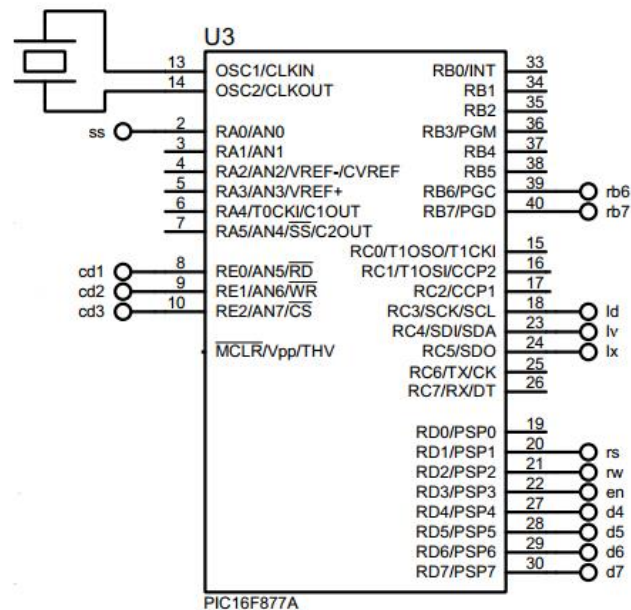
E: cho phép VĐK read or write.

Chú ý: Khi VXL viết lệnh đến IR hoặc viết mã số liệu đến DR (thì kiểm tra bít D7 để biết LCD có bận hay không). VXL thường xuyên kiểm tra LCD có bận hay không để gửi lệnh hoặc mã số liệu.

❖ Đề viết hoặc đọc lên LCD

- Đưa RS:
 - Mức 0 lựa chọn thanh ghi kênh IR (cài đặt thông số)
 - Mức 1 lựa chọn thanh ghi số liệu DR.
- Đọc LCD: chân RW, R ở mức cao 1, W ở mức thấp 0; viết LCD thì ngược lại.
- Chân E ở mức 1.

E. Khối mạch điều khiển

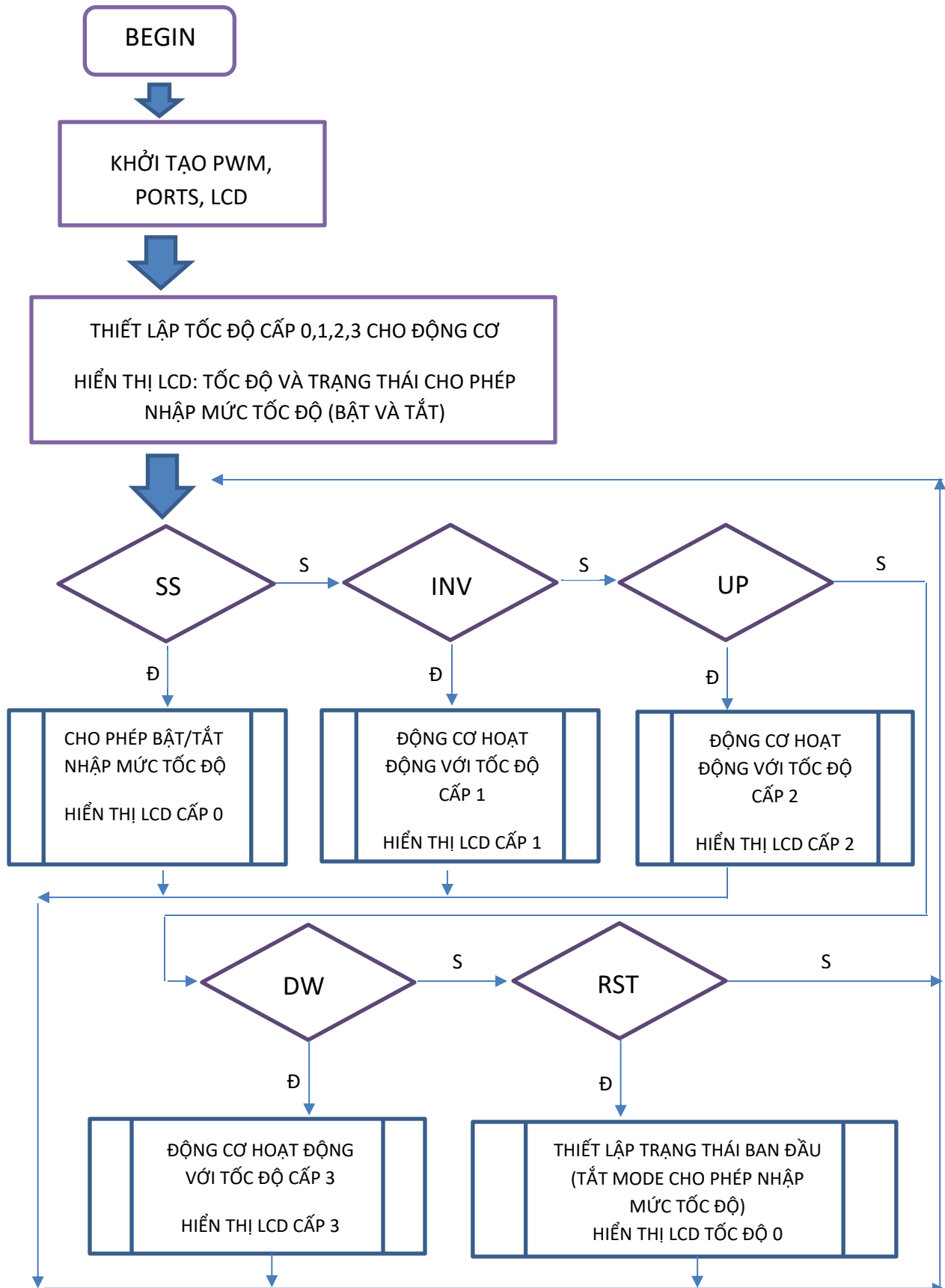


Vi điều khiển trung tâm là PIC 16F877A. Với chức năng của các port như sau:

- Ta sử dụng 2 chân RC1(CCP2) và RC2 (CCP1) của Port C để xuất tín hiệu PWM điều khiển động cơ.
- Port D (trừ chân RD0) gửi tín hiệu đến khối hiển thị LCD. Ba chân từ RD1 đến RD3 nối với 3 chân điều khiển của LCD. Bốn chân từ RD4 đến RD7 nối với 4 bit cao của các chân nhận dữ liệu của LCD.
- Thạch anh dùng trong mạch có giá trị 20MHz.

II. Mô phỏng bằng phần mềm ứng dụng Proteus

1/ Lưu đồ thuật toán

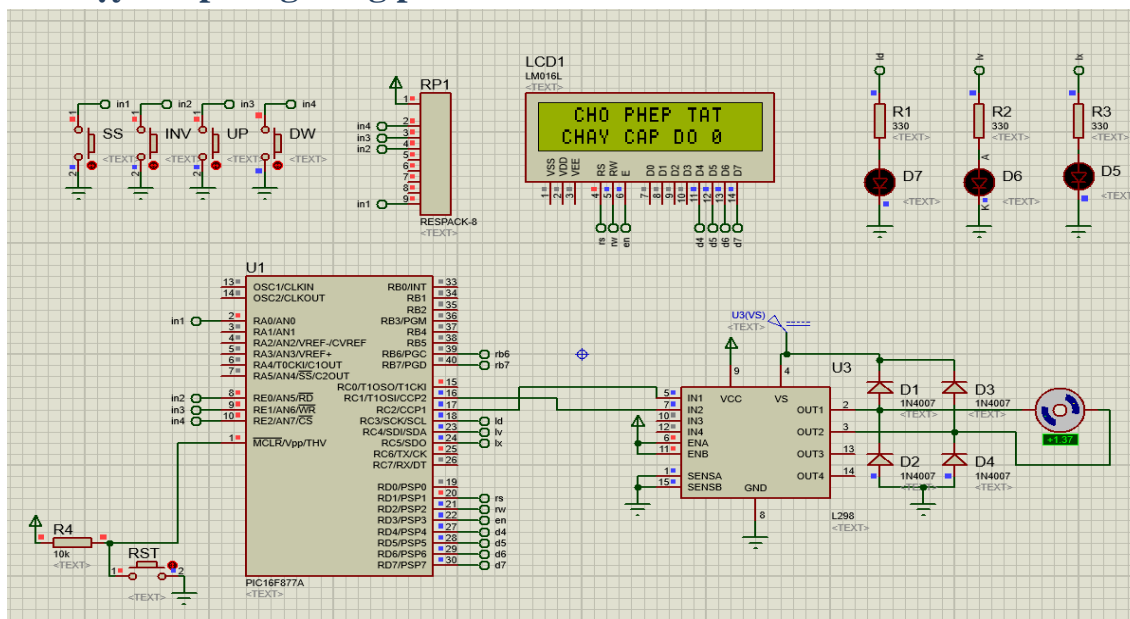


Giải thích lưu đồ thuật toán:

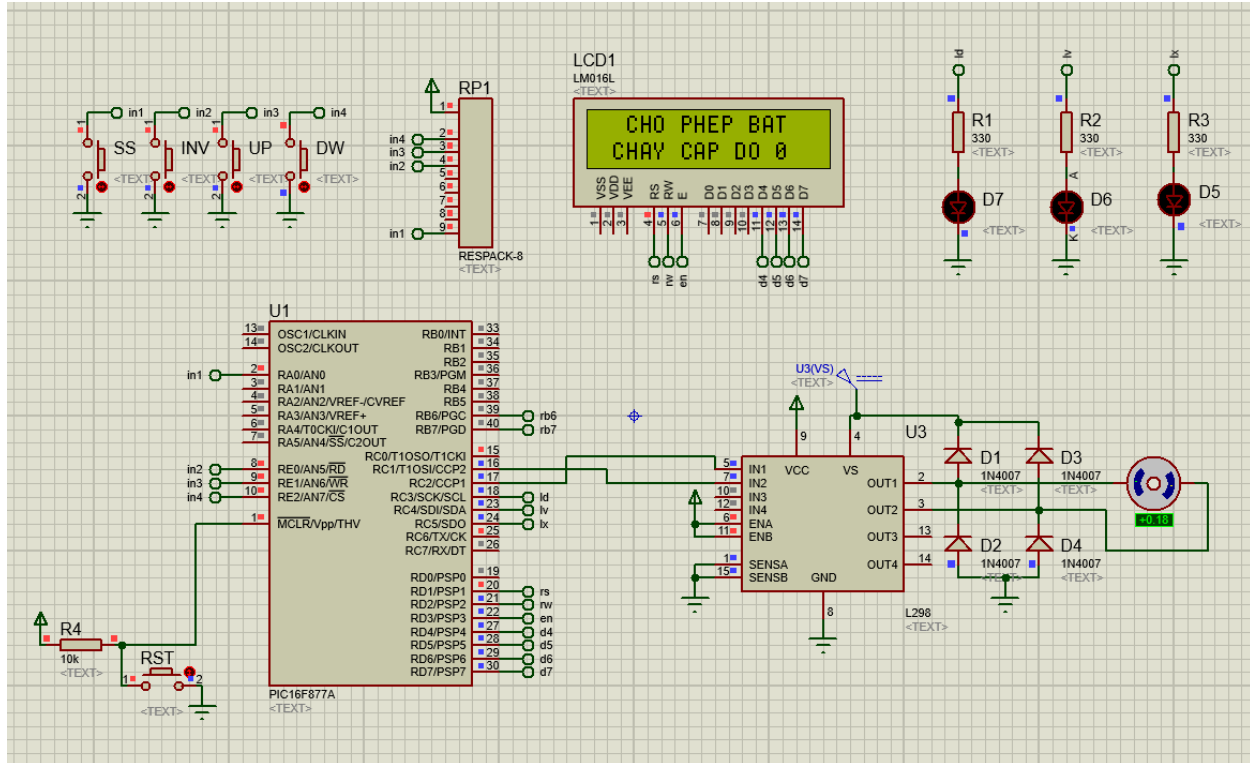
❖ Chương trình chính thực hiện các yêu cầu:

- Khởi tạo các Port, PWM, Timer2 và tốc độ ban đầu là 0
- Khởi tạo LCD, hiển thị thông tin và trạng thái tốc độ
- Nhấn SS: Cho phép motor sẵn sàng hoạt động và hiển thị cấp độ ban đầu là 0, kiêm vai trò dừng động cơ khi đang hoạt động.
- Vòng lặp while thực hiện thao tác nhấn phím INV, UP, DW
- Nếu **đúng** thì chương trình sẽ tiếp tục thao tác kiểm tra và thực hiện lệnh như đã lập trình. Nếu **sai** sẽ quay trở lại thiết lập ban đầu
- Chương trình con phím INV kiểm tra nếu nhấn thì động cơ sẽ quay theo mức tốc độ 1 là 184 vòng/phút, hiển thị LCD Cấp 1
- Chương trình con phím UP kiểm tra nếu nhấn thì động cơ sẽ quay theo mức tốc độ 2 là 323 vòng/phút, hiển thị LCD Cấp 2
- Chương trình con phím INV kiểm tra nếu nhấn thì động cơ sẽ quay theo mức tốc độ 3 là 461 vòng/phút, hiển thị LCD Cấp 3
- Chương trình con phím RS kiểm tra nếu nhấn thì động cơ sẽ dừng hoàn toàn, sinh ra hãm động năng.

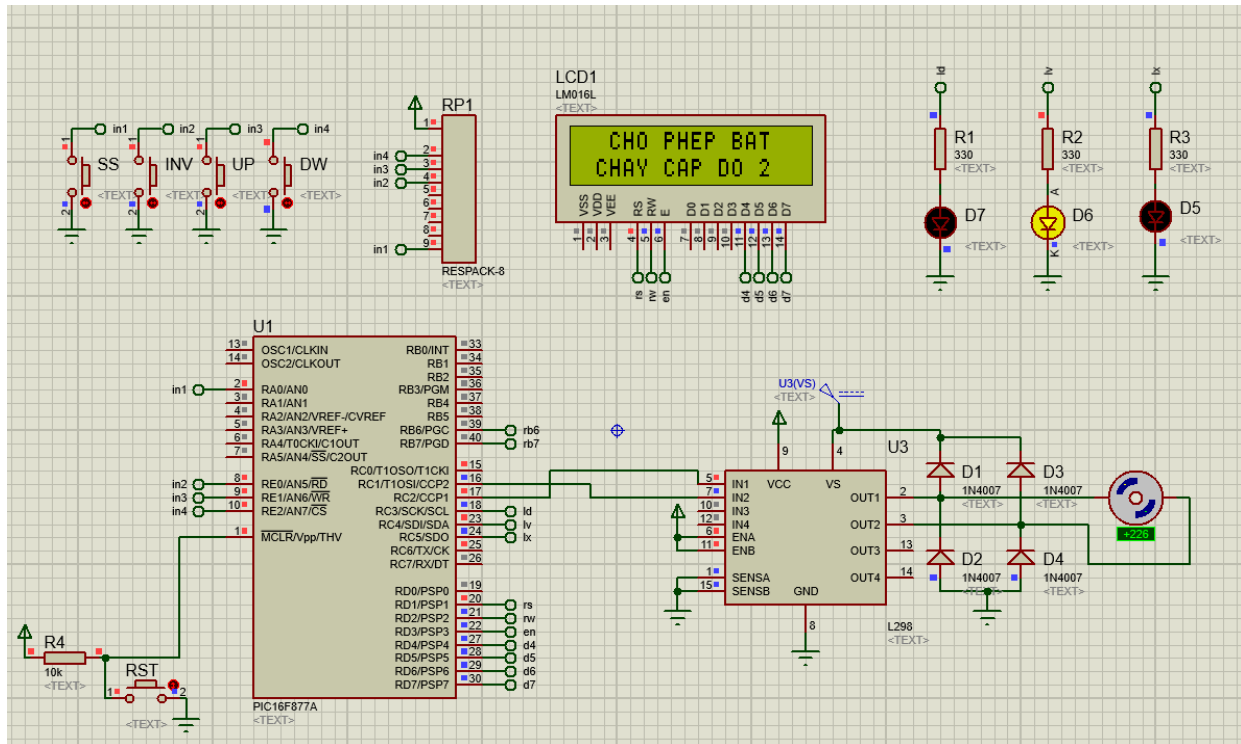
2/ Chạy mô phỏng bằng phần mềm Proteus



Hình 18: Khi mạch chưa cho phép bật động cơ



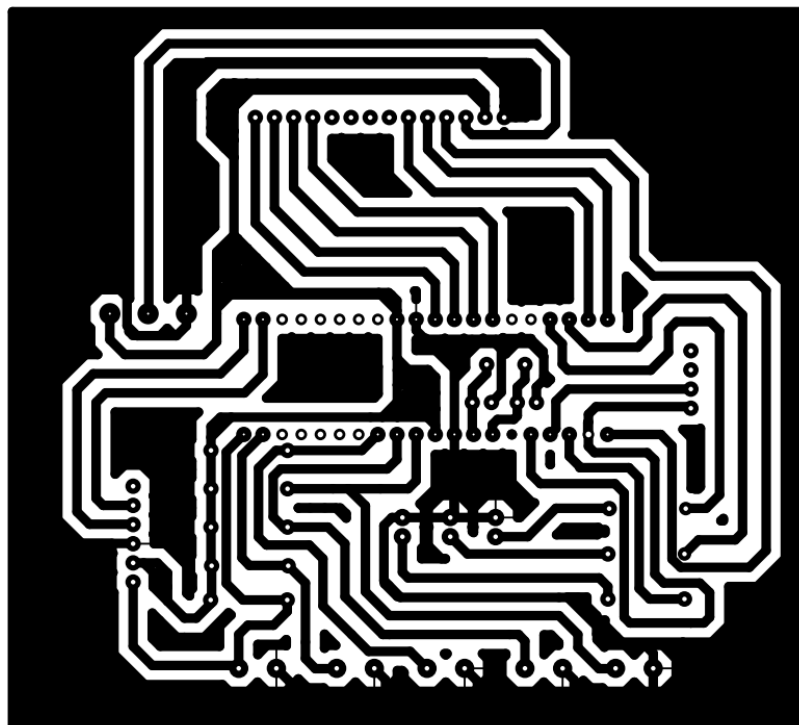
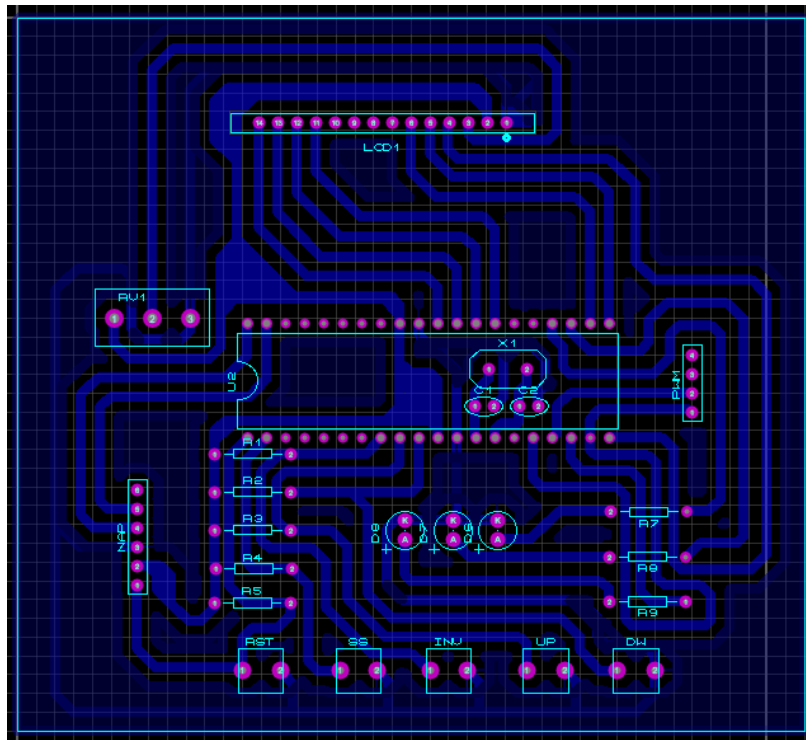
Hình 19: Khi ta nhấn nút SS cho phép kích hoạt mức tốc độ điều khiển



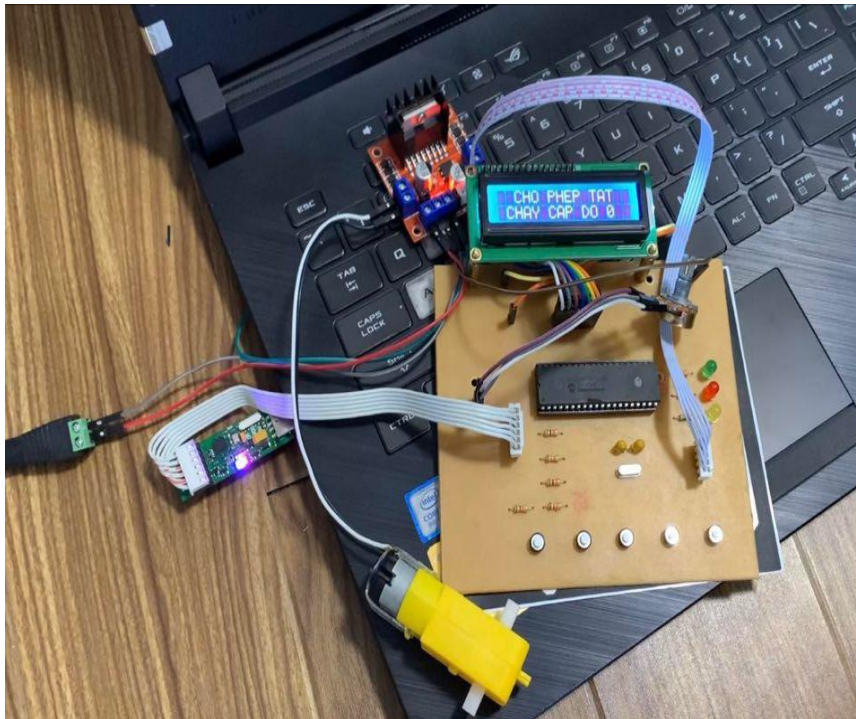
Hình 20: Khi ta bật động cơ quay với cấp độ 2 từ phím INV

CHƯƠNG 4: CHẾ TẠO MẠCH THỰC TẾ

A. THIẾT KẾ MẠCH IN



B. LẮP ĐẶT THIẾT BỊ VÀ HOÀN THIỆN MẠCH



C. CHẠY MẠCH VÀ ĐÁNH GIÁ KẾT QUẢ

- ❖ Sau khi cho hệ thống điều khiển tốc độ động cơ hoạt động tuần tự các nút mức tốc độ, các phím dừng. Ta đo điện áp ở 1 số chân linh kiện và nhận được các kết quả đúng như mong muốn:
 - Ở các chân PIC, do sử dụng nguồn riêng từ mạch nạp và 1 phần đầu ra 5V của module L298 nên điện áp ổn định ở 5V và không có hiện tượng sụt áp.
 - Động cơ nối với cổng vào 12V của module L298, được cấp bởi nguồn 220V qua adapter 12V/1A, điện áp ổn định.
 - LCD và LED hoạt động tốt, điều chỉnh biến trở sao cho LCD hiện rõ kết quả nhất, thu được như trên hình.
 - Dễ dàng điều chỉnh và cắm dây, đã thử nghiệm nhiều lần và không xảy ra sự cố.

PHỤ LỤC

A. CODE CHƯƠNG TRÌNH BẰNG NGÔN NGỮ C QUA PHẦN MỀM CCS

***Những từ sau // là giải nghĩa**

`#include <16f877a.h>` // Khai báo chỉ định đường dẫn 16f877a.h cho trình biên dịch

`#FUSES HS` // Khai báo dùng thạch anh trên 4M

`#use delay(crystal=20m)` // thạch anh ngoài 20MHz

`#define LCD_ENABLE_PIN PIN_d3` // Dùng để khai báo chân E của LCD nối với chân d3 của VĐK

`#define LCD_RS_PIN PIN_d1` // Dùng khai báo chân RS (chân điều khiển lựa chọn thanh ghi) của LCD nối với chân d1 của VĐK

`#define LCD_RW_PIN PIN_d2` // Dùng khai báo chân RW (chân điều khiển quá trình đọc và ghi) của LCD nối với chân d2 của VĐK

`#define LCD_DATA4 PIN_D4` // Dùng khai báo trao đổi dữ liệu giữa chân D4 LCD và chân d4 của VĐK

`#define LCD_DATA5 PIN_D5` // Dùng khai báo trao đổi dữ liệu giữa chân D5 LCD và chân d5 của VĐK

`#define LCD_DATA6 PIN_D6` // Dùng khai báo trao đổi dữ liệu giữa chân D6 LCD và chân d6 của VĐK

`#define LCD_DATA7 PIN_D7` // Dùng khai báo trao đổi dữ liệu giữa chân D7 LCD và chân d7 của VĐK

```
#include <lcd.c>          // Khai báo chỉ định đường dẫn lcd.c cho
trình biên dịch

signed int32  PWM_DUTY;

int1  tt_onoff;

#define  ld      pin_c3  //Dùng khai báo chân đèn D8 nối với
chân RC3 của VĐK

#define  lv      pin_c4  //Dùng khai báo chân đèn D7 nối với
chân RC4 của VĐK

#define  lx      pin_c5  //Dùng khai báo chân đèn D6 nối với
chân RC5 của VĐK


#define  ss      pin_a0  // Dùng khai báo chân nút SS nối với
chân RA0 của vđk

#define  cd1     pin_e0  // dùng khai báo chân nút INV nối với
chân RE0 của VĐK

#define  cd2     pin_e1  // dùng khai báo chân nút UP nối với
chân RE1 của VĐK

#define  cd3     pin_e2  // dùng khai báo chân nút DW nối với
chân RA2 của VĐK


void nn_ss()      //Hàm Khai báo nút nhấn SS
{
    if(input(ss)==0)    // so sánh nếu tín hiệu vào nút nhấn là mức thấp
    {
        delay_ms(20);    // Chống đôi nút nhấn , tạm dừng 20 mili giây
    }
}
```



```
if(input(ss)==0)    // so sánh nếu tín hiệu vào nút nhấn là mức thấp
{
    tt_onoff=~tt_onoff; // đảo trạng thái onoff

    if(tt_onoff==1)    //so sánh nếu tín hiệu nút nhấn onoff là mức
cao
    {
        lcd_gotoxy(1,1);    //Hàm trở tới vị trí cần hiển thị vị trí
cột 1 hàng 2
        lcd_putc(" CHO PHEP BAT "); // hàm gửi chuỗi ký tự lệnh
lên LCD
        PWM_DUTY=0;    //Gán chu kỳ xung là 0
        output_low(pin_c1); // Thiết lập mức 0v cho chân RC1 của
VĐK
        SETUP_CCP2(CCP_OFF); // Vô hiệu hóa tính năng CCP2
        SETUP_CCP1(CCP_PWM); //Thiết lập chế độ hoạt động
của ccp1
        SET_PWM1_DUTY(PWM_DUTY); //Đặt 0 mức cao
        output_low(ld); // Thiết lập mức 0v cho chân ld của led
        output_low(lx); // Thiết lập mức 0v cho chân lx của led
        output_low(lv); // Thiết lập mức 0v cho chân lv của led
    }
    else
    {
```

```

        lcd_gotoxy(1,1);      //Hàm trở tới vị trí cần hiển thị vị trí
cột 1 hàng 2
        lcd_putc(" CHO PHEP TAT "); // hàm gửi chuỗi ký tự lệnh
lên LCD

        lcd_gotoxy(1,2);      //Hàm trở tới vị trí cần hiển thị vị trí
cột 2 hàng 2
        lcd_putc(" CHAY CAP DO 0 "); // hàm gửi chuỗi ký tự lệnh
lên LCD

        PWM_DUTY=0; // Gán PWM_DUTY = 0

        SETUP_CCP2(CCP_off);   // Vô hiệu hóa tính năng CCP2
        SETUP_CCP1(CCP_OFF);   // Vô hiệu hóa tính năng CCP1
        output_low(pin_c2);    // Thiết lập mức 0v cho chân RC2 của
VĐK

        output_low(pin_c1);    // Thiết lập mức 0v cho chân RC1 của
VĐK

        output_low(ld); // Thiết lập mức 0v cho chân ld của led
        output_low(lx); // Thiết lập mức 0v cho chân lx của led
        output_low(lv); // Thiết lập mức 0v cho chân lv của led
    }

    while(input(ss)==0); // Chờ nhả phím SS khi tín hiệu ở nút
nhấn SS là mức thấp

    }

}

}

```

```

void nn_cd1() //Hàm Khai báo nút nhấn cd1
{
    if(input(cd1)==0) // so sánh nếu tín hiệu ở nút cd1 là mức thấp
    {
        delay_ms(20); // Chống đỗi nút nhấn , tạm dừng 20 mili giây
        if(input(cd1)==0) // so sánh nếu tín hiệu ở nút cd1 là mức thấp
        {
            if(tt_onoff==1) // so sánh nếu tín hiệu ở nút onoff là mức cao
            {
                PWM_DUTY=400; //gán chu kỳ xung là 400
                SET_PWM1_DUTY(PWM_DUTY); //Đặt
                lcd_gotoxy(1,2); //Hàm trở tới vị trí cần hiển thị vị trí
hàng 2 cột 2
                lcd_putc(" CHAY CAP DO 1 "); // hàm gửi chuỗi ký tự lệnh
lên LCD
                output_low(ld); //Thiết lập mức 0v cho chân ld của led
                output_high(lx); //Thiết lập mức 5v cho chân lx của led
                output_low(lv); // Thiết lập mức 0v cho chân lv của led
            }
            while(input(cd1)==0); // Chờ nhả phím cd1 khi tín hiệu ở nút
nhấn cd1 là mức thấp
        }
    }
}

```

```

void nn_cd2()    //Hàm Khai báo nút nhấn cd2
{
    if(input(cd2)==0)    // so sánh nếu tín hiệu ở nút cd2 là mức thấp
    {
        delay_ms(20);    // Chống đôi nút nhấn , tạm dừng 20 mili giây
        if(input(cd2)==0)    // so sánh nếu tín hiệu ở nút cd2 là mức thấp
        {
            if(tt_onoff==1)    // so sánh nếu tín hiệu ở nút onoff là mức cao
            {
                PWM_DUTY=700;    //gán chu kỳ xung là 700
                SET_PWM1_DUTY(PWM_DUTY);    //Đặt
                lcd_gotoxy(1,2);    //Hàm trở tới vị trí cần hiển thị vị trí cột 2
                lcd_putc(" CHAY CAP DO 2 ");    // hàm gửi chuỗi ký tự lệnh
                output_low(ld);    //Thiết lập mức 0v cho chân ld của led
                output_low(lx);    //Thiết lập mức 0v cho chân lx của led
                output_high(lv);    //Thiết lập mức 5v cho chân lv của led
            }
            while(input(cd2)==0);    // Chờ nhả phím cd2 khi tín hiệu ở nút
            nhấn cd2 là mức thấp
        }
    }
}

```

hàng 2

lên LCD

```

    }

void nn_cd3() //Hàm Khai báo nút nhấn cd3
{
    if(input(cd3)==0) // so sánh nếu tín hiệu ở nút cd3 là mức thấp
    {
        delay_ms(20); // Chống đơ nút nhấn , tạm dừng 20 giây
        if(input(cd3)==0) // so sánh nếu tín hiệu ở nút cd3 là mức thấp
        {
            if(tt_onoff==1) // so sánh nếu tín hiệu ở nút onoff là mức cao
            {
                PWM_DUTY=1000; //gán chu kỳ xung là 1000
                SET_PWM1_DUTY(PWM_DUTY); //Đặt
                lcd_gotoxy(1,2); //Hàm trở tới vị trí cần hiển thị vị trí
hàng 2 cột 2
                lcd_putc(" CHAY CAP DO 3 "); // hàm gửi chuỗi ký tự
lệnh lên LCD

                output_high(ld); //Thiết lập mức 5v cho chân ld của led
                output_low(lx); //Thiết lập mức 0v cho chân lx của led
                output_low(lv); //Thiết lập mức 0v cho chân lv của led
            }

            while(input(cd3)==0); // Chờ nhả phím cd3 khi tín hiệu ở nút
nhấn cd3 là mức thấp
        }
    }
}

```

```

    }
}

void main()
{
    set_tris_c(0x80);    // 1000 0000

    set_tris_e(0xff);    // tắt cả các chân trong port E là input
    set_tris_a(0xff);    // tắt cả các chân trong port A là input

    lcd_init();

    output_low(pin_c2);  // Thiết lập mức 0v cho chân RC2 của VĐK
    output_low(pin_c1);  // Thiết lập mức 0v cho chân RC1 của VĐK

    SETUP_TIMER_2(T2_DIV_BY_16,249,1);    //Khởi tạo của
timer 2 (dùng riêng cho PWM )

    SETUP_CCP1(CCP_off);    // Vô hiệu hóa tính năng CCP1

    PWM_DUTY=0;    //Gán chu kỳ xung là 0%

    output_low(ld);    //Thiết lập mức 0v cho chân ld của led
    output_low(lx);    //Thiết lập mức 0v cho chân lx của led
    output_low(lv);    //Thiết lập mức 0v cho chân lv của led

    SET_PWM1_DUTY(PWM_DUTY);    //Đặt 0 mức cao

    lcd_gotoxy(1,1);    //Hàm trở tới vị trí cần hiển thị vị trí hàng 2
cột 1

    lcd_putc(" CHO PHEP TAT ");    // hàm gửi chuỗi ký tự lệnh lên
LCD

```

```
        lcd_gotoxy(1,2);    //Hàm trở tới vị trí cần hiển thị vị trí hàng 2
cột 2

        lcd_putc(" CHAY CAP DO 0 "); // hàm gửi chuỗi ký tự lệnh lên
LCD

while(TRUE) // vòng lặp vô hạn
{
    nn_ss();
    nn_cd1();
    nn_cd2();
    nn_cd3();
}
}
```

B. CHƯƠNG TRÌNH HỢP NGỮ THIẾT LẬP BAN ĐẦU

```
PROCESSOR PIC16F877A

#include <PIC16F877A.INC>

__CONFIG_HS_OSC & _WDT_OFF & _PWRTE_OFF

#define E PORTD,3
#define RS PORTD,1
#define RW PORTD,2
```

```
#DEFINE    LD PORTC,3
```

```
#DEFINE    LV PORTC,4
```

```
#DEFINE    LX PORTC,5
```

```
#DEFINE    SS PORTA,0
```

```
#DEFINE    CD1 PORTE,0
```

```
#DEFINE    CD2 PORTE,1
```

```
#DEFINE    CD2 PORTE,2
```

```
ORG 0X00
```

```
GOTO MAIN
```

```
MAIN
```

```
CLRF TRISB
```

```
CLRF TRISE
```

```
CLRF TRISA
```

```
BSF T2CON,1
```

```
BSF T2CON,2
```