**GIỚI THIỆU**

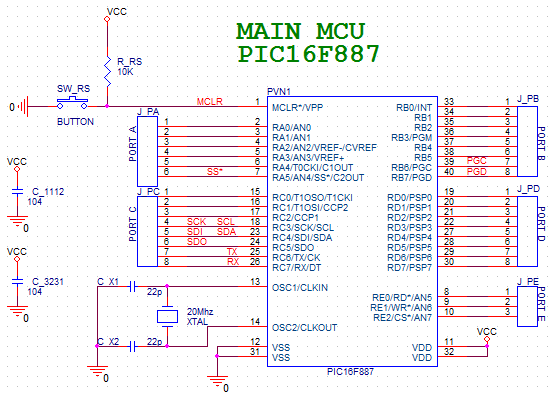
**Mục lục**

1. **Mạch nguyên lý của KIT PIC16F887**
2. **Main MCU 4**
3. **LED module 4**
4. **Module nút nhấn 5**
5. **Cổng nạp 5**
6. **Khối cấp nguồn 5**
7. **Giao tiếp I2C 6**
8. **Giao tiếp SPI 6**
9. **Giao tiếp UART 6**
10. **Module ADC 6**
11. **Khối hiển thị (LCD) 10**
12. **Mạch lái động cơ (L298N) 11**
13. **Tổng quan về vi điều khiển PIC16F887**
14. **Giới thiệu chung 12**
15. **Sơ đồ khối bên trong 12**
16. **Sơ đồ chân 13**
17. **Chức năng các chân 13**
18. **Cổng nạp 15**
19. **Hoạt động ngắt 16**
20. **Nguồn xung nhịp cho chip 16**
21. **Giới thiệu về LCD**
22. **Tổng quát về LCD HD44780 17**
23. **Chức năng các chân 17**
24. **Vùng hiển thị (DDRAM) và tập lệnh 19**
25. **Kết luận 20**
26. **PHỤ LỤC**
27. **Lập trình và nạp chương trình cho PIC16F887 21**
28. **Áp dụng viết chương trình điều khiển tốc độ động cơ 22**

**Nhận xét của Giáo Viên hướng dẫn**

**………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………**

1. **Mạch nguyên lý của KIT**
   1. **Main MCU**

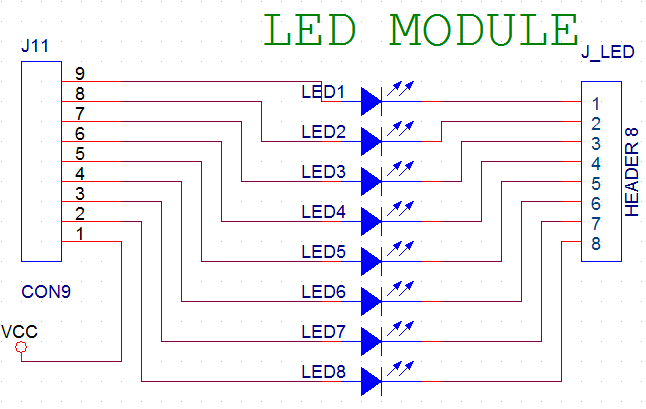


* + - * + Pic 16F887:Chip vi điều khiển 8-bit của Microchip điều khiển các khối module còn lại.
        + Thạch anh 20Mhz + 2 tụ 22p: tạo nguồn xung nhịp cho chip. Tụ 22p là tụ gốm.
        + Các port A-E: Dùng header hàn vào, để có thể linh động trong việc viết chương trình cũng như kết nối phần cứng.
        + SW\_RS(Button): nút nhấn reset. Đây là loại nút nhấn - nhả, có 2 chân
        + Tụ C\_1112 và C\_3231: 2 tụ gốm có mã số là 104, dùng để lọc nhiễu nguồncho IC.

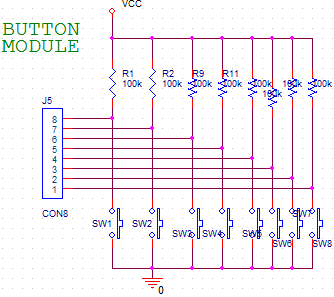
Tụ C\_1112 phải gắn gần 2 chân số 11, 12 của PIC.

Tụ C\_3231 phải gắn gần 2 chân số 32, 31 của PIC.

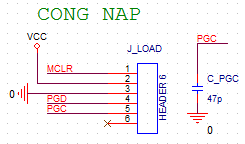
* 1. **LED Module**



* + - * + Gồm 8 LED, 1 điên trở thanh 330ohm để hạn dòng cho LED, nối lên nguồn dương (+5V)
        + J\_LED là header 8 dùng để cắm dây nối sang các porttín hiệu của PIC (Port A, B, C, …)
        + Với sơ đồ mắc như thế này thì để LED sáng ta chỉ cần cho các chân tương ứng với header J\_LED xuống mức 0
  1. **Module nút nhấn**

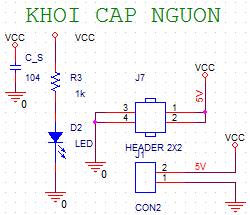


* + - * + 8 nút nhấn nối với điện trở 100k lên nguồn dương.
        + J5 là header 8, dùng để cắm dây nối với các chân tín hiệu của PIC.
        + Bình thường khi chưa nhấn nút, các chân của header J5 sẽ được nối lến nguồn (5V – mức 1), khi nhấn nút thì chân ứng với nút được nhấn sẽ nối xuống GND (0V), tức có sự chuyển mức điện áp từ 1 sang 0,do đó ta sẽ sử dụng ngắt PORTB để phát hiện ra sự chuyến mức điện áp này (cạnh lên hoặc cạnh xuống tùy vào việc cấu hình PIC), từ đó biết được nút nào được nhấn.
  1. **Cổng nạp**

Gồm 1 header 6 và 1 tụ 47p để ổn định cho chân PGC.

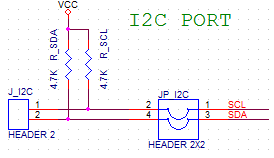
Cổng này dùng để nạp chương trình cho chip 16F887.

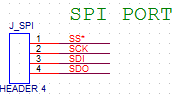
* 1. **Khối cấp nguồn**

Điện trở R3 1k nối tiếp với LED D2 để báo nguồn

Tụ pi C\_S 104 để lọc nhiễu nguồn

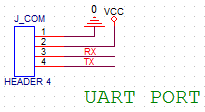
Để đưa nguồn 5V từ mạch nguồn sang KIT, ta có thể sử dụng header hoặc JACK\_DC

* 1. **Giao tiếp I2C**
     + - * Dùng để giao tiếp với các thiết bị, cảm biến, … theo chuẩn giao tiếp I2C.
         * Gồm 2 điện trở 4.7k treo lên nguồn dương.
         * J\_I2C là header 2 để nối với thiết bị bên ngoài.
         * JP\_I2C: là header 2x2 nối với con PIC. Khi cần dùngport I2C thì dùng jumper nối chân 1 với 2, 3 với 4.
  2. **Giao tiếp SPI**

Cổng giao tiếp với thiết bị, cảm biến bên ngoài theo chuẩn SPI.

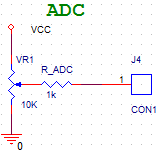
J\_SPI là header 4.

* 1. **UART Port**

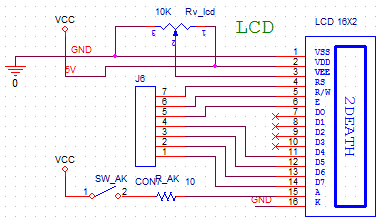


Là header 4, dùng để kết nối với mạch giao tiếp cổng COM bên ngoài.

* 1. **ADC module**

VR1 là biến trở 10k để điều chỉnh mức điện áp cho module ADC

* 1. **Khối hiển thị LCD**



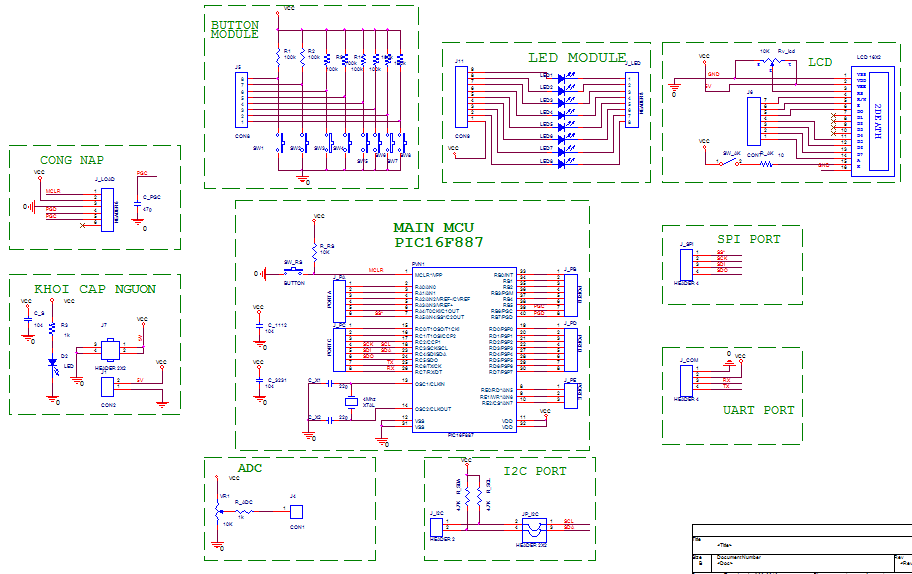
* + - * + LCD gồm 16 chân, chân VSS nối GND, VDD nối nguồn 5V, VEE nối với biến trở Rv\_lcd 10k để điều chỉnh độ sáng của LCD.
        + 3 đường điều khiển RS,R/W,E
        + 8 đường dữ liệu D0-D7, ở đây ta chỉ sử dụng chế độ 4 đường dữ liệu cho đơn giản
        + Chân A, K được nối với công tắc SW\_AK để bật đèn bên trong LCD

Figure 1. Sơ đồ nguyên lý PIC KIT

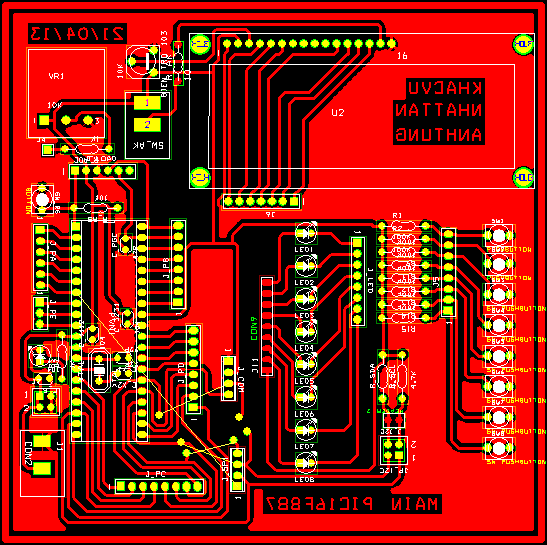
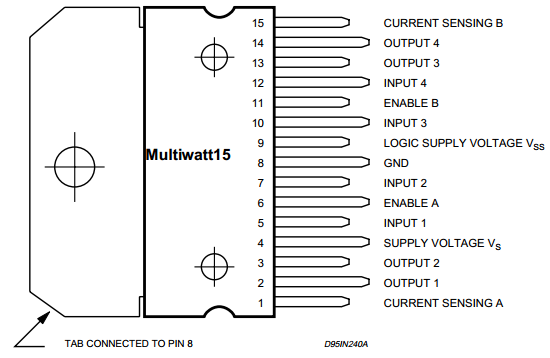
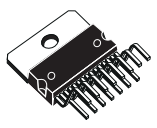


Figure 2. Layout PIC KIT

* 1. **Mạch lái động cơ dùng chip L298N**
* L298N là một chip tích hợp 2 mạch cầu H
* L298N có điện áp danh nghĩa cao (lớn  nhất 50V) và dòng điện danh nghĩa lớn hơn 2A, thích hợp cho các các ứng dụng công suất nhỏ như các động cơ DC loại nhỏ và vừa.





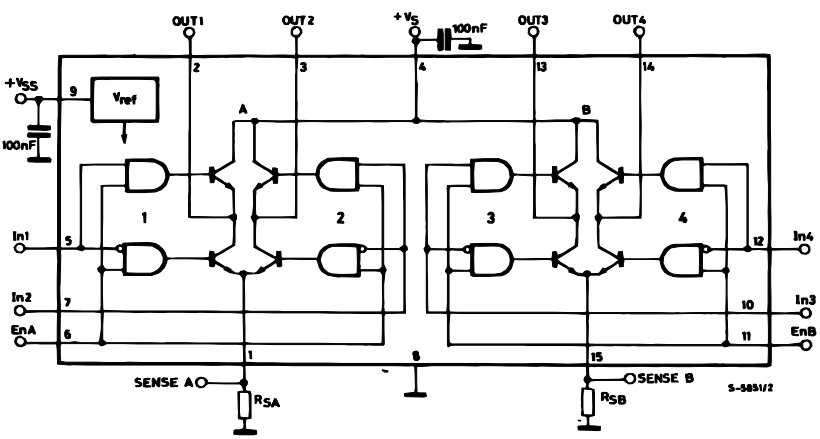
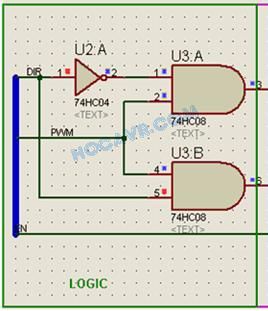
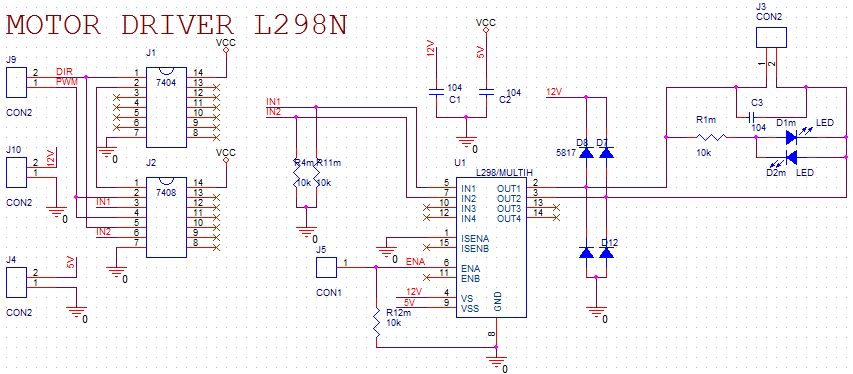


Figure 3. Mô hình thật và cấu trúc bên trong chip

* Động cơ sẽ được nối với 2 đường **OUT1, OUT2** (hoặc **OUT3, OUT4**). Một chân **En** (EnA và EnB cho 2 mạch cầu) cho phép mạch cầu hoạt động, khi chân En được kéo lên mức cao, mạch cầu sẵn sang hoạt động. Hai đường kích cho mỗi cầu H gọi là **In1** và **In2** (hoặc **In3, In4**).
* Để motor hoạt động chúng ta phải kéo 1 trong 2 đường kích này lên cao trong khi đường kia giữ ở mức thấp
  + ví dụ In1=1, In2=0. Khi đảo mức kích của 2 đường In, động cơ sẽ đảo chiều quay.
* Tuy nhiên, do L298D không chỉ được dùng đề đảo chiều động cơ mà còn điều khiển vận tốc động cơ bằng PWM, các đường **In** cần được “tổ hợp lại” bằng các cổng Logic
* Ngoài ra, trên chip L298D còn có các đường Vss cấp điện áp cho phần logic (5V) và GND chung cho cả logic và motor.
* Trong thực tế, công suất thực mà L298N có thể tải nhỏ hơn so với giá trị danh nghĩa của nó (V=50V, I=2A). Để tăng dòng điện tải của chip lên gấp đôi, chúng ta có thể nối 2 mạch cầu H song song với nhau (các chân có chức năng như nhau của 2 mạch cầu được nối chung).
* Thông thường, khi thiết kế một mạch driver cho motor người ta thường dành 3 đường điều khiển:
  + **PWM** : điều khiển vận tốc,
  + **DIR :** điều khiển hướng và
  + **EN :** cho phép mạch hoạt động.
* Chip L298N đã có sẵn đường En nhưng 2 đường điều khiển In1 và In2 không thật sự chức năng như chúng ta mong muốn. Vì thế, chúng ta sẽ thiết kế một mạch logic phụ với 2 ngõ vào là PWM và DIR trong khi 2 ngõ ra là 2 đường điều khiển In1 và In2. Bảng chân trị của mạch logic cần thiết kế được trình bày trong bảng 1.  
  *Bảng 1. Bảng chân trị của mạch logic cho driver L298D*

|  |  |  |  |
| --- | --- | --- | --- |
| PWM | DIR | In1 | In2 |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

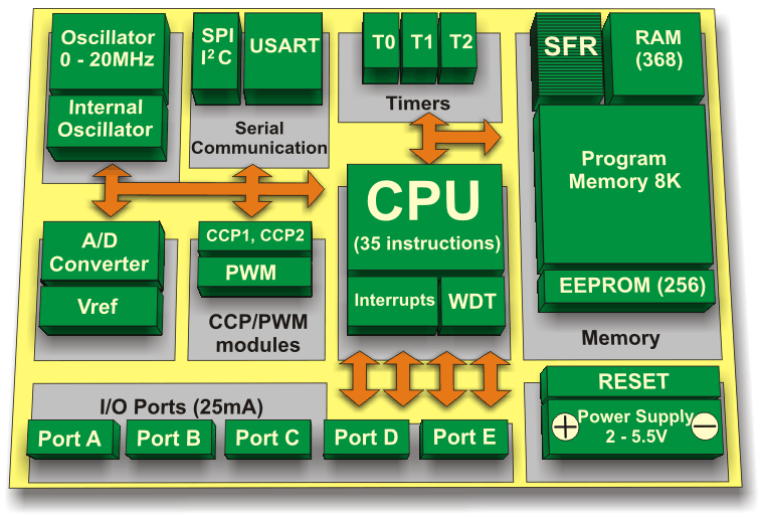
* Từ bảng chân trị này, chúng ta có thể viết hàm bool cho 2 ngõ In1 và In2:  
  ***In1=PWM.NOT(DIR)  
        In2=PWM.DIR***  
        Mạch logic vì thế sẽ có dạng như sau
* Với mạch logic này, đường **DIR** có chức năng đảo chiều động cơ trong khi đường **PWM** điều khiển vận tốc động cơ bằng tín hiệu PWM.
* Tóm lại, sơ đồ nguyên lý mạch lái động cơ như sau:
  + - * 1. J4 cấp nguồn 5V
        2. J10 cấp nguồn 12V cho L298N
        3. J9 là 2 chân DIR và PWM lấy từ PIC16F887
        4. J3 là header2 nối với 2 cực của động cơ
        5. 2 LED D1m vaf D2m để biết động cơ quay theo chiều nào
        6. Tụ C1 và C2 lọc nhiễu nguồn

1. **Tổng quan về vi điều khiển PIC16F887**
   1. **Giới thiệu chung**
      * Vi điều khiển 8-bit của Microchip, thuộc dòng Low-power.
      * Kiến trúc ***Harvard*** *(vs. Von Neumann),* tập lệnh ***RISC*** *(****R****educed* ***I****nstructions* ***S****et* ***C****omputer) (vs. Complexed Instructions Set Computer)*
      * Thạch anh gắn ngoài tối đa 20MHz.
      * Tầm điện áp hoạt động 2.0V- 5.5V.
      * 5 port, 35 chân xuất nhập (I/O pins)
      * Có đầy đủ các chức năng cần thiết của Vi điều khiển 8-bit: Timer (3 bộ), ADC (14 kênh ADC 10-bit), USART, SPI, I2C, PWM, Compare, …
      * 8K ROM memory in FLASH technology: Chip can be reprogrammed up to 100.000 times
      * 256 bytes EEPROM memory: Data can be written more than 1.000.000 times
      * 368 bytes RAM memory
      * A/D converter:

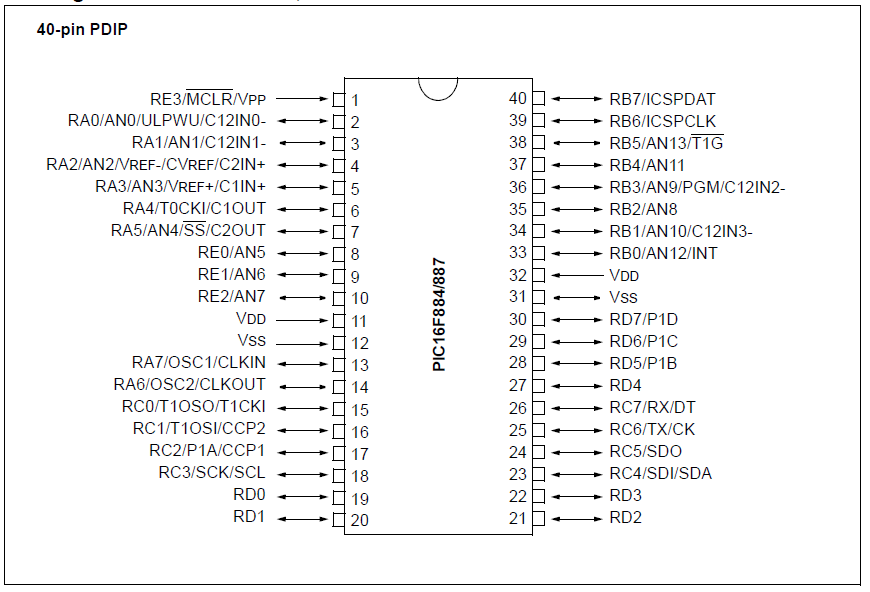
14-kênh

10-bit độ phân giải

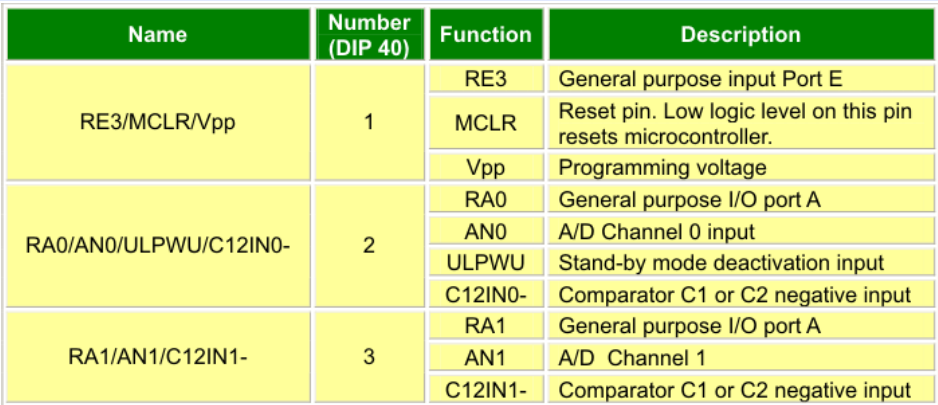
* 3 independent timers/counters
* Watch-dog timer
* Analogue comparator module with
  + - Two analogue comparators
    - Fixed voltage reference (0.6V)
    - Programmable on-chip voltage reference
* PWM output steering control
* Enhanced USART module: hỗ trợ RS-485, RS-232 and LIN2.0 , Auto-Baud
* Master Synchronous Serial Port (MSSP): hỗ trợ giao tiếp SPI và I2C
  1. ***Sơ đồ khối bên trong***

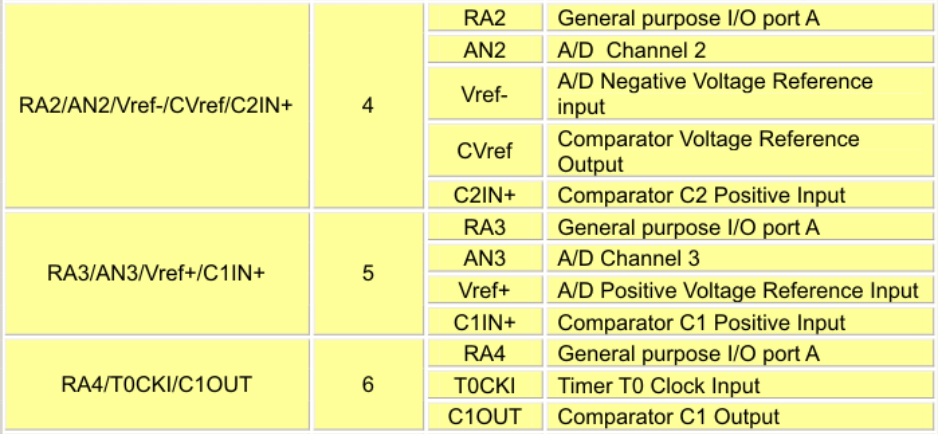


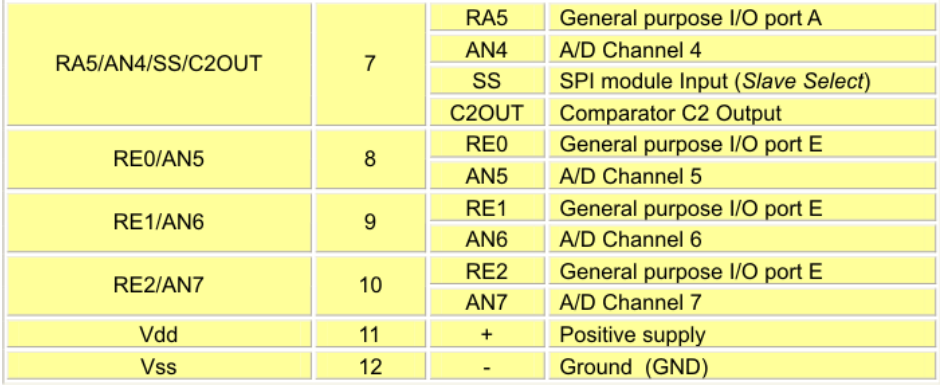
* 1. ***Sơ đồ chân***

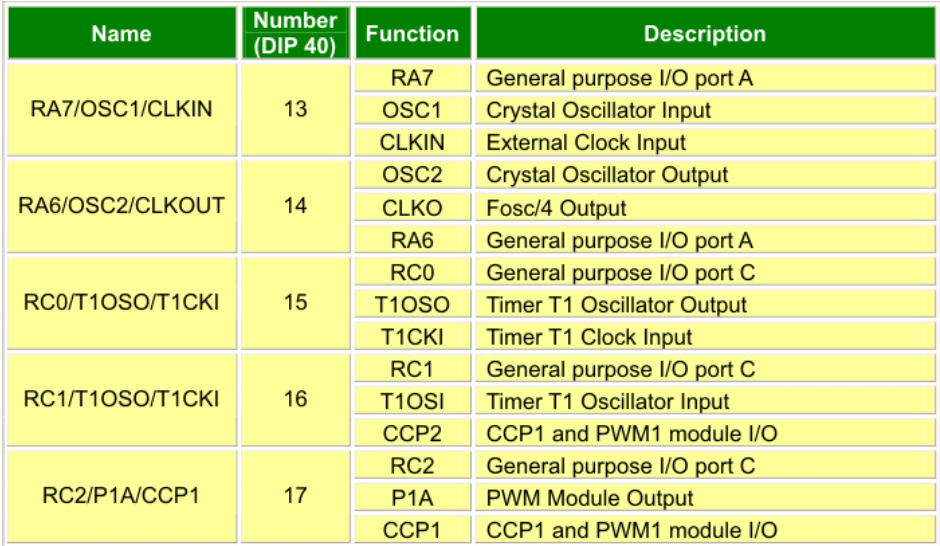
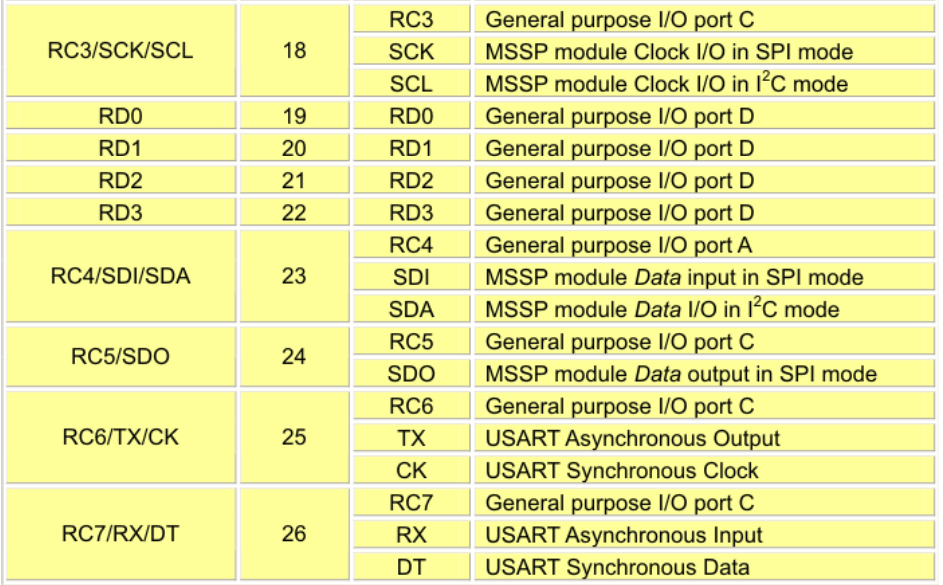
******

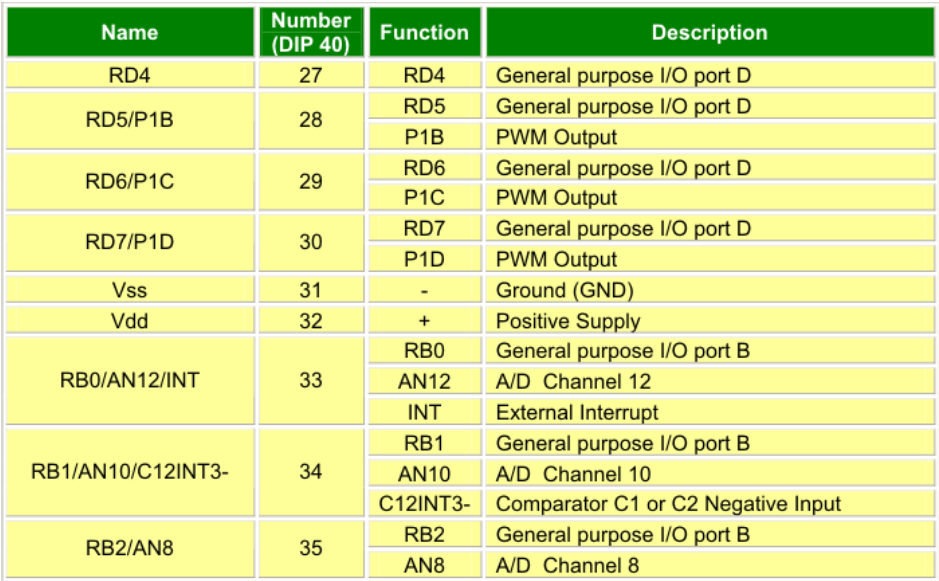
* 1. ***Chức năng các chân***

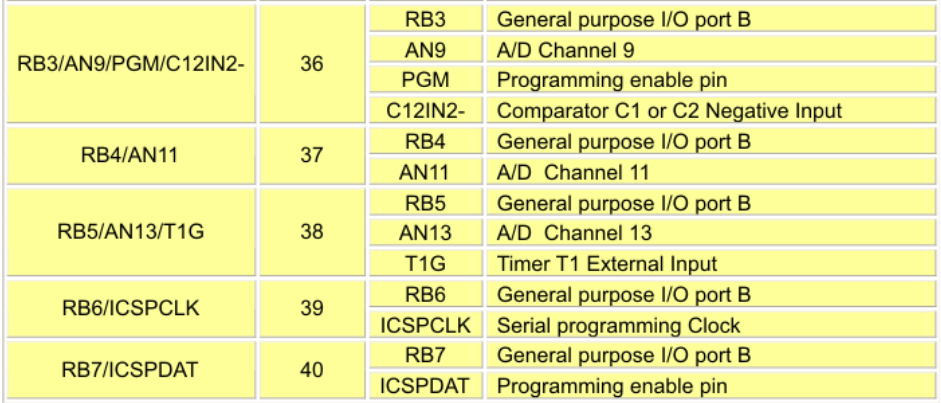




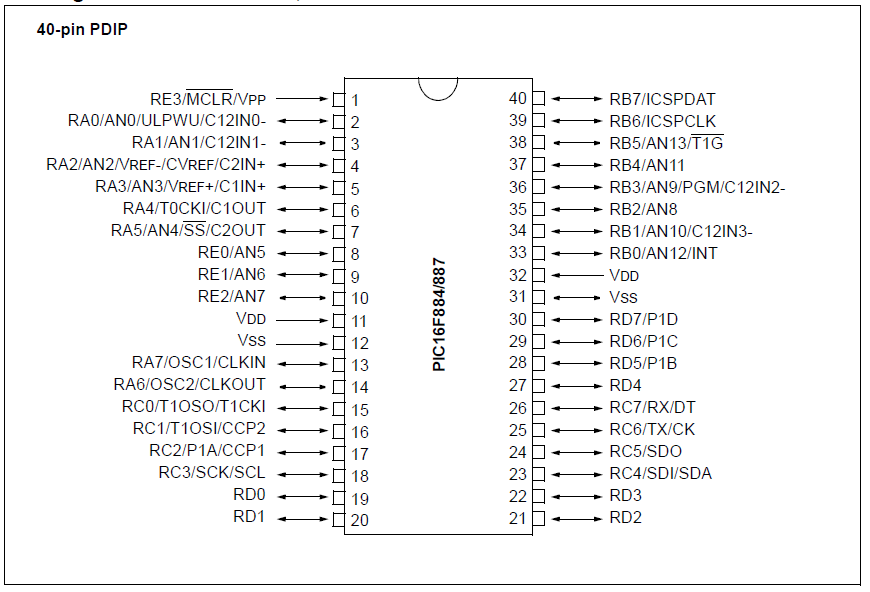




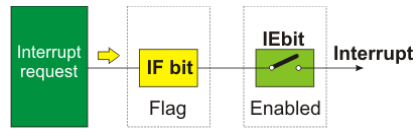
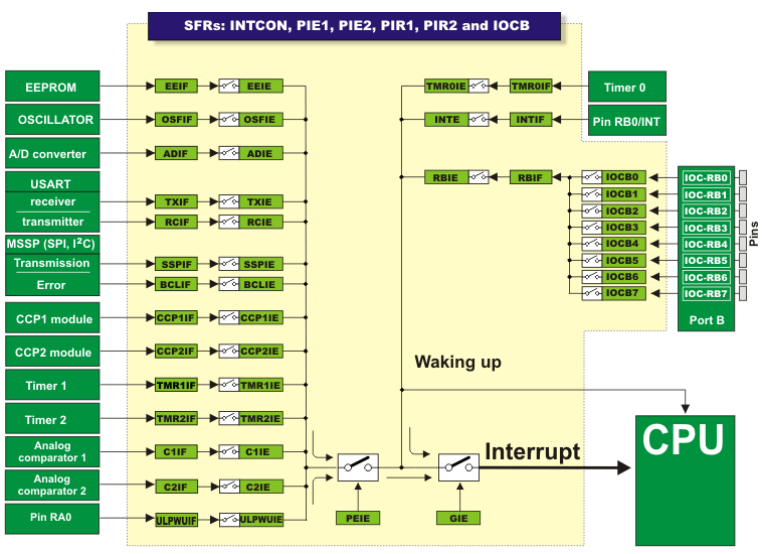




* 1. ***Cổng nạp chuẩn ICSPTM*** *(In-Circuit Serial ProgrammingTM):*



* 1. **Hoạt động ngắt:**



* 1. ***Các nguồn xung nhịp cho chip:***
     + - Bộ dao động RC nội *(RC internal oscillator)*
       - Bộ dao động tạo bởi thạch anh (và tụ) gắn ngoài *(External Crystal and Ceramic Oscillator)*
       - Bộ RC gắn ngoài.
       - Xung clock từ một nguồn khác.

**III.Giới thiệu về LCD**

Ngày nay, thiết bị hiển thị LCD (Liquid Crystal Display) được sử dụng trong rất nhiều các ứng dụng của VĐK. LCD có rất nhiều ưu điểm so với các dạng hiển thị khác: Nó 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 rẽ …

1. **Tổng Quát Về LCD HD44780**

* Hình dáng và kích thước: Có rất nhiều loại LCD với nhiều hình dáng và kích thước [](http://www.dientumaytinh.com/2012/03/gioi-thieu-co-ban-ve-lcd-16x2.html)khác nhau

Figure 4. Hình dáng của loại LCD thông dụng

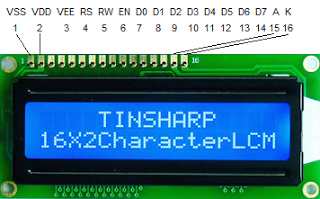
* Khi sản xuất LCD, nhà sản xuất đã tích hợp chíp điều khiển (HD44780) bên trong lớp [](http://www.dientumaytinh.com/2012/03/gioi-thieu-co-ban-ve-lcd-16x2.html)vỏ và chỉ đưa các chân giao tiếp cần thiết:

Figure 5. Sơ đồ chân của LCD

1. **Chức năng các chân :**

|  |  |  |
| --- | --- | --- |
| **Chân** | **Ký hiệu** | **Mô tả** |
| 1 | Vss | Chân nối đất cho LCD, khi thiết kế mạch ta nối chân này với GND của mạch điều khiển |
| 2 | VDD | Chân cấp nguồn cho LCD, khi thiết kế mạch ta nối chân này với VCC=5V của mạch điều khiển |
| 3 | VEE | 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” (VCC) 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 nó 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 |
| 15 | - | Nguồn dương cho đèn nền |
| 16 | - | GND cho đèn nền |

\* *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.

* + - Chíp HD44780 có 2 thanh ghi 8 bit quan trọng : Thanh ghi lệnh IR (Instructor Register) và thanh ghi dữ liệu DR (Data Register)
    - Bằng cách điều khiển chân RS và R/W chúng ta có thể chuyển qua lại giữ 2 thanh ghi này khi giao tiếp với MPU. Bảng sau đây tóm tắt lại các thiết lập đối với hai chân RS và R/W theo mục đích giao tiếp.

|  |  |  |
| --- | --- | --- |
| **RS** | **R/W** | **Chức năng** |
| **0** | **0** | **Ghi vào thanh ghi IR để ra lệnh cho LCD** |
| **0** | **1** | **Đọc cờ bận ở DB7 và giá trị của bộ đếm địa chỉ ở DB0-DB6** |
| **1** | **0** | **Ghi vào thanh ghi DR** |
| **1** | **1** | **Đọc dữ liệu từ DR** |

* Khi thực hiện các hoạt động bên trong chíp, mạch nội bên trong cần một khoảng thời gian để hoàn tất. Khi đang thực thi các hoạt động bên trong chip như thế, LCD bỏ qua mọi giao tiếp với bên ngoài và bật cờ BF (thông qua chân DB7 khi có thiết lập RS=0, R/W=1) lên để báo cho MPU biết nó đang “bận”. Dĩ nhiên, khi xong việc, nó sẽ đặt cờ BF lại mức 0.

**3.Vùng RAM hiển thị DDRAM : (Display Data RAM)**

* Đây là vùng RAM dùng để hiển thị, nghĩa là ứng với một địa chỉ của RAM là một ô kí tự trên màn hình và khi bạn ghi vào vùng RAM này một mã 8 bit, LCD sẽ hiển thị tại vị trí tương ứng trên màn hình một kí tự có mã 8 bit mà bạn đã cung cấp. Hình sau đây sẽ trình bày rõ hơn mối liên hệ này

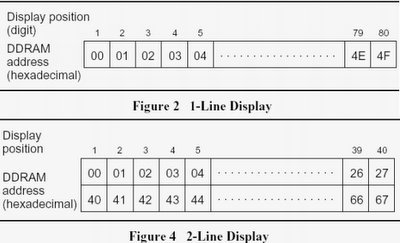
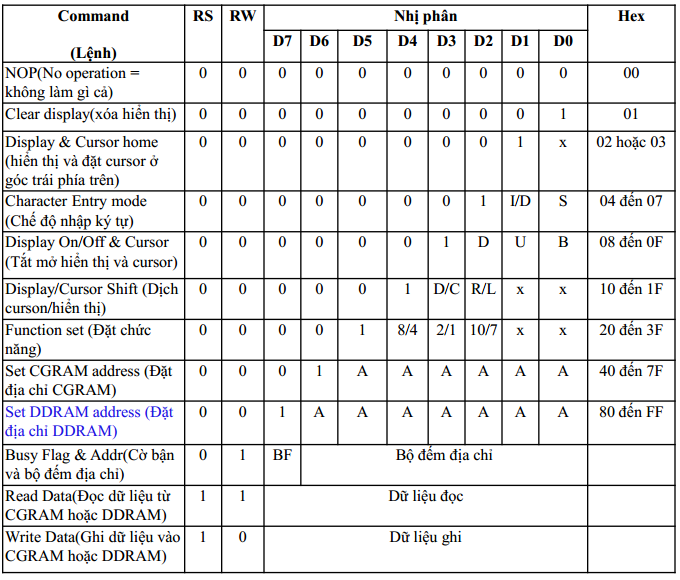
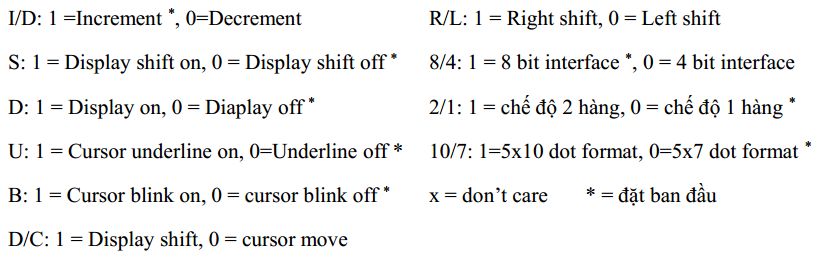
[](http://www.dientumaytinh.com/2012/03/gioi-thieu-co-ban-ve-lcd-16x2.html)

Figure 6. Mối liên hệ giữa địa chỉ của DDRAM và vị trí hiển thị của LCD

* Vùng RAM này có 80x8 bit nhớ, nghĩa là chứa được 80 kí tự mã 8 bit. Những vùng RAM còn lại không dùng cho hiển thị có thể dùng như vùng RAM đa mục đích.

Lưu ý là để truy cập vào DDRAM, ta phải cung cấp địa chỉ cho AC theo mã HEX

* ***Bảng 4 : Tập lệnh của LCD***

**IV.Kết luận**

* 1. **Ưu điểm**
  2. **Khuyết điểm**
  3. **Hướng phát triển**

1. **PHỤ LỤC**
2. **Lập trình và nạp chương trình cho PIC16F887**

* *Dùng phần mêm* ***MPLAB*** *của* ***MicroChip*** *trình dịch* ***HiTech PIC,*** *viết chương trình bằng* ***ngôn ngữ C*** *để lập trình cho* ***PIC16F887.***
* *Để nạp chương trình xuống* ***PIC,*** *ta cần có mạch nạp + chương trình nạp, ở đây dùng chương trình* ***PICkit2***



* *Cấu trúc một chương trình như sau:*

#include <htc.h>

\_\_CONFIG(XT & WDTDIS & PWRTEN & MCLREN & UNPROTECT & SWBOREN & IESODIS & FCMDIS & LVPDIS & DEBUGDIS); *//1st config. Word*

\_\_CONFIG(BORV21); *//2st config. Word*

#define \_XTAL\_FREQ 4000000 *//tần số thạch anh 4Mhz*

#define HANG\_SO 123

***//Khai báo biến toàn cục***

**int** a,b=1,c=3;

**char** bien\_char;

***//Khai báo các chương trình con***

voidinit\_port(void)

{

***//chương trình con ở đây***

}

chartinh\_toan(char a, int b)

{

*//………*

}

***/\* chương trình con cũng có thể được viết sau hàm main(), nhưng trước main() phải có khai báo***

***void init\_port(void);***

***char tinh\_toan(char a, int b); \*/***

***//Chương trình chính***

void main(void)

{

*// Phần khởi tạo*

*//vòng lặp vô tận*

while(1)

{

*//thực hiện lệnh*

};

}

void interrupt isr()

{

*// Chương trình phục vụ ngắt*

}

1. **Áp dụng để viết chương trình điều khiển tốc độ động cơ**

Ta sẽ viết một chương trình điều khiển tốc độ động cơ đơn giản

Ngõ vào là giá trị lấy từ biến trở 10k. Ta nối ngõ ra của biến trở với chân số 2 (AN0) và dùng module ADC của vi điều khiển, như đã giới thiệu, module ADC có độ phân giải 10 bit từ 0-1023 tương ứng với điện áp của biến trở từ 0-5V, chu kỳ đọc ADC là 20ms (khoảng thời gian này được định thì bởi Timer 1)

Để điều khiển được tốc độ động cơ, ta dùng module PWM của vi điều khiển để điều chế độ rộng xung, với chu kỳ xung khoảng Tpwm = 40us, duty cycle có giá trị từ 0-100% tương ứng với giá trị đọc vào từ module ADC, xuất xung này ra đường CCP1(chân số 17) kèm với tín hiệu chiều quay RC3 (chân số 18) và tín hiệu cho phép EN (chân số 16) vào mạch lái động cơ.

Đọc tốc độ động cơ đang quay, ta dùng module Timer0 ở chế độ Counter để đếm xung từ Encoder, xung ra của Encoder nối với chân số 6 (T0CKI), đếm xung trong khoảng thời gian là Ts = 10ms (khoảng thời gian này được định thì bởi Timer 1) ta được n xung, từ giá trị này ta có thể dễ dàng suy ra tốc độ của động cơ (vòng/phút) theo công thức:

Các chân 23 đến 26 là 4 đường data xuất ra LCD, chân 8,9,10 là 3 đường điều khiển LCD, để hiển thị LCD một cách đơn giản ta chỉ cần copy 2 file lcd.h và lcd.c sửa lại cho phù hợp với sơ đồ phần cứng và dùng các lệnh đã được viết sẵn trong 2 file này.

Chân số 35,36,37 (RB2,RB3,RB4) được nối với các nút nhấn với chức năng tương ứng DIR (chỉnh chiều quay), START (bắt đầu cho động cơ chạy), STOP (dừng)

Sau đây là code cho chương trình (phần giải thích được chú thích ngay sau các dòng lệnh của chương trình)

