

**ĐẠI HỌC QUỐC GIA TP HỒ CHÍ MINH  
TRƯỜNG ĐẠI HỌC BÁCH KHOA  
KHOA ĐIỆN – ĐIỆN TỬ**



**LUẬN VĂN TỐT NGHIỆP ĐẠI HỌC**

**ĐIỀU KHIỂN ĐỘNG CƠ KHÔNG ĐỒNG BỘ 3 PHA  
SỬ DỤNG VI ĐIỀU KHIỂN PIC18F4431  
THEO PHƯƠNG PHÁP VECTOR KHÔNG GIAN**

**SVTH : NGUYỄN HUỖNH QUANG  
MSSV : 40202088  
CBHD : TS. PHAN QUỐC DŨNG  
BỘ MÔN : CUNG CẤP ĐIỆN**

**TP Hồ Chí Minh, 01/2007**



**NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN**

-----  
-----  
-----  
-----  
-----  
-----  
-----  
-----  
-----  
-----  
-----  
-----  
-----  
-----  
-----  
-----

Tp Hồ Chí Minh, tháng 1 năm 2007  
Giáo viên phản biện

# LỜI CẢM ƠN !

*Tôi xin gửi lời cảm ơn chân thành nhất đến quý Thầy Cô trường Đại Học Bách Khoa Tp. Hồ Chí Minh, những người đã truyền đạt cho tôi những kiến thức và kinh nghiệm quý báu trong suốt thời gian tôi học tập tại trường.*

*Tôi xin trân trọng gửi lời cảm ơn đến tất cả các Thầy, Cô Khoa Điện - Điện Tử : thầy Lê Minh Phương, thầy Phan Quốc Dũng và thầy Trần Thanh Vũ..... đã tận tình hướng dẫn, giúp đỡ, tạo mọi điều kiện thuận lợi để tôi hoàn thành tốt luận văn tốt nghiệp này.*

*Tôi xin gửi lời cảm ơn đến tất cả những người bạn, những người anh em ( Lê Trung Nam, Võ Văn Vũ, Tiết Vĩnh Phúc.....) những người đã cùng gắn bó, cùng học tập và giúp đỡ tôi trong những năm qua cũng như trong suốt quá trình thực hiện luận văn tốt nghiệp.*

*Cuối cùng, tôi cảm ơn gia đình, những người thân, người yêu (Đ.T.T.N) và đặc biệt là thân mẫu đã cho tôi những điều kiện tốt nhất để học tập trong suốt thời gian dài.*

*Tp. Hồ Chí Minh, tháng 1 năm 2007*

# MỤC LỤC

<b>CHƯƠNG 1:</b>	<b>2</b>
<b>GIỚI THIỆU VỀ ĐỘNG CƠ KĐB VÀ PHƯƠNG PHÁP ĐIỀU KHIỂN</b>	<b>2</b>
1.1> TỔNG QUAN VỀ ĐỘNG CƠ ĐỒNG BỘ:.....	2
1.1.1) Giới thiệu:.....	2
1.1.2) Cấu tạo:.....	2
1.1.3) Ứng dụng:.....	3
1.2> CÁC PHƯƠNG PHÁP ĐIỀU KHIỂN ĐỘNG CƠ KĐB:.....	4
<b>CHƯƠNG 2:</b>	<b>5</b>
<b>GIỚI THIỆU VỀ BIẾN TẦN NGUỒN ÁP ĐIỀU KHIỂN <math>V/f=const</math></b>	<b>5</b>
2.1> BIẾN TẦN NGUỒN ÁP:.....	5
2.2> PHƯƠNG PHÁP ĐIỀU KHIỂN $V/f$ :.....	5
2.2.1) Phương pháp $E/f$ .....	5
2.2.2) Phương pháp $V/f$ .....	6
2.3> PHƯƠNG PHÁP ĐIỀU CHẾ SIN PWM:.....	7
2.3.1) Giới thiệu:.....	7
2.3.2) Các công thức tính toán:.....	9
2.3> PHƯƠNG PHÁP ĐIỀU CHẾ VECTOR KHÔNG GIAN ( SVM).....	10
2.3.1) giới thiệu chung:.....	10
2.3.2) Sơ đồ sắp xếp các vector $V_0 \rightarrow V_7$ trên trục $V_a; V_b; V_c$ .....	11
2.3.2) Giới thiệu vector $V_s$ :.....	13
2.3.3) Cách tính toán thời gian để tạo ra vector $\vec{V}_s$ :.....	15
2.4> KỸ THUẬT ĐIỀU CHẾ VECTOR KHÔNG GIAN:.....	16
2.4.1) Giải đồ đóng ngắt các khóa để tạo ra Vector $V_s$ trong từng sector:.....	16
2.4.2) Sơ đồ tóm tắt của quá trình điều chế :.....	19
2.4.3) Tính toán góc update của vector $V_s$ theo phương pháp điều khiển $V/f$ :.....	20
<b>CHƯƠNG 3:</b>	<b>22</b>
<b>GIỚI THIỆU VỀ PIC<sup>®</sup> Microcontrollers (MCUs)</b>	<b>22</b>
3.1>TỔNG QUAN:.....	22
3.1.1> Những đặc điểm nổi bật PIC18F4431:.....	24
3.1.2> Những đặc điểm chính:.....	25
3.2>TÓM TẮT TRÚC PHẦN CỨNG:.....	26
3.2.1> Sơ đồ chân MCU PIC18F4431 :.....	26
3.2.3) Chức năng của từng chân:.....	28
3.3> CÁC MODULE CƠ BẢN:.....	32
3.3.1> Power control PWM module :.....	32
3.3.2> Analog to digital converter module (A/D):.....	48
<b>CHƯƠNG 4 :</b>	<b>51</b>
<b>THIẾT KẾ PHẦN CỨNG</b>	<b>51</b>
4.1> YÊU CẦU CƠ BẢN :.....	51
4.2> SƠ ĐỒ KHỐI CỦA HỆ THỐNG :.....	52
4.3> MẠCH ĐỘNG LỰC :.....	53
4.3.1) Bộ chỉnh lưu:.....	53
4.3.2) Bộ nghịch lưu:.....	54
4.3.3) Mạch lái ( driver) & cách ly:.....	55
4.2> MẠCH ĐIỀU KHIỂN:.....	59

4.2.1) Sơ đồ khối mạch điều khiển: .....	59
4.2.2) Các tín hiệu vào của mạch điều khiển: .....	59
4.2.3) Tín hiệu đầu ra của mạch điều khiển: .....	59
<b>CHƯƠNG 5:</b>	<b>60</b>
<b>LẬP TRÌNH</b>	<b>60</b>
5.1> GIẢI THUẬT LẬP TRÌNH : .....	60
5.1.1) Chương trình chính: .....	60
5.1.2) Chương trình ngắt: .....	61
5.2> GIẢI THÍCH GIẢI THUẬT : .....	62
5.2.1) Chương trình chính: .....	62
5.2.2) Chương trình ngắt : .....	62
<b>CHƯƠNG 6:</b>	<b>64</b>
<b>KẾT QUẢ ĐẠT ĐƯỢC</b>	<b>64</b>
6.1> PHẦN CỨNG:.....	64
6.1.1> Mạch động lực: .....	64
6.1.2> Mạch điều khiển:.....	65
6.2> PHẦN MỀM GIAO TIẾP VỚI NGƯỜI SỬ DỤNG:.....	66
6.2.2) Mô tả: .....	67
6.3> DẠNG SÓNG ĐIỆN ÁP NGỒ RA:.....	67
6.4> HƯỚNG PHÁT TRIỂN:.....	68
6.4.1) Khắc phục những khuyết điểm hiện tại: .....	68
<b>CHƯƠNG 7:</b>	<b>69</b>
<b>TÀI LIỆU THAM KHẢO</b>	<b>69</b>
<b>CHƯƠNG 8:</b>	<b>70</b>
<b>PHỤ LỤC</b>	<b>70</b>
8.1> SƠ ĐỒ MẠCH (VẼ TRÊN ORCAD):.....	70
8.1.1) Sơ đồ mạch cách ly .....	70
8.1.2) Sơ đồ mạch lái: .....	72
8.1.3) Sơ đồ mạch nghịch lưu : .....	73
8.1.4) Sơ đồ mạch điều khiển : .....	74
8.2> CHƯƠNG TRÌNH VIẾT CHO PIC18F4431 : .....	76
8.3> CODE PHẦN MỀM GIAO TIẾP NGƯỜI SỬ DỤNG:.....	102

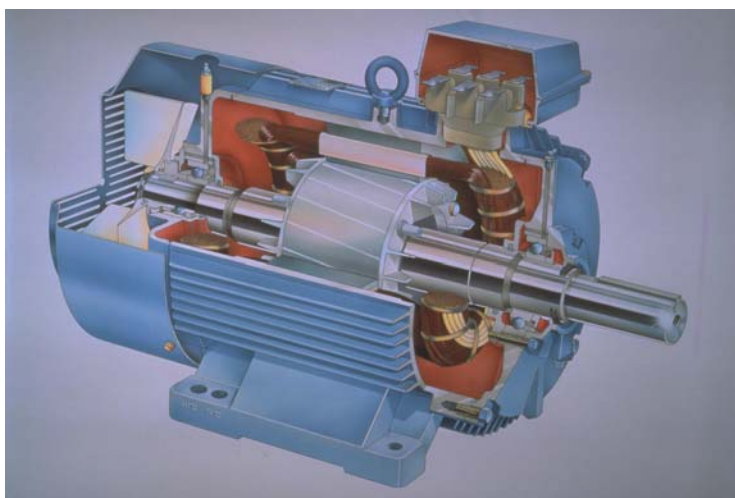
**CHƯƠNG 1:  
GIỚI THIỆU VỀ ĐỘNG CƠ KĐB VÀ PHƯƠNG PHÁP ĐIỀU KHIỂN**

**1.1> TỔNG QUAN VỀ ĐỘNG CƠ ĐỒNG BỘ:**

**1.1.1) Giới thiệu:**

Động cơ điện không đồng bộ ba pha (AC Induction Motor) được sử dụng rất phổ biến ngày nay với vai trò cung cấp sức kéo trong hầu hết các hệ thống máy công nghiệp. Công suất của các động cơ không đồng bộ có thể đạt đến 500 kW (tương đương 670 hp) và được thiết kế tuân theo quy chuẩn cụ thể nên có thể thay đổi dễ dàng các nhà cung cấp.

**1.1.2) Cấu tạo:**



Hình 1.1: Cấu tạo bên trong động cơ KĐB

**1.1.2a) Phần tĩnh:** Stator có cấu tạo gồm vỏ máy, lõi sắt và dây quấn

**+ Vỏ máy:**

Vỏ máy có tác dụng cố định lõi sắt và dây quấn, không dùng để làm mạch dẫn từ. Thường vỏ máy được làm bằng gang. Đối với máy có công suất tương đối lớn ( 1000kW ) thường dùng thép tấm hàn lại làm thành vỏ máy. Tùy theo cách làm nguội máy mà dạng vỏ cũng khác nhau.

**+ lõi sắt:**

Lõi sắt là phần dẫn từ. Vì từ trường đi qua lõi sắt là từ trường quay nên để giảm tổn hao: lõi sắt được làm bằng những lá thép kỹ thuật điện ép lại.

**+ Dây quấn:**

Dây quấn stator được đặt vào các rãnh của lõi sắt và được cách điện tốt với lõi sắt.

### 1.1.2b) Phân quay ( roto):

Rotor có 2 loại chính : rotor kiểu dây quấn và rotor kiểu lồng sóc.

#### + rotor kiểu dây quấn:

Rôto có dây quấn giống như dây quấn của stator. Dây quấn 3 pha của rôto thường đấu hình sao còn ba đầu kia được nối vào vành trượt thường làm bằng đồng đặt cố định ở một đầu trục và thông qua chổi than có thể đấu với mạch điện bên ngoài. Đặc điểm là có thể thông qua chổi than đưa điện trở phụ hay suất điện động phụ vào mạch điện rôto để cải thiện tính năng mở máy, điều chỉnh tốc độ hoặc cải thiện hệ số công suất của máy. Khi máy làm việc bình thường dây quấn rotor được nối ngắn mạch. Nhược điểm so với động cơ rotor lồng sóc là giá thành cao, khó sử dụng ở môi trường khắc nghiệt, dễ cháy nổ ...

#### + rotor kiểu lồng sóc:

Kết cấu loại dây quấn này rất khác với dây quấn stator. Trong mỗi rãnh của lõi sắt rotor đặt vào thanh dẫn bằng đồng hay nhôm dài ra khỏi lõi sắt và được nối tắt lại ở hai đầu bằng hai vành ngắn mạch bằng đồng hay nhôm làm thành một cái lồng mà người ta quen gọi là lồng sóc

### 1.1.2c) Khe hở không khí:

Vì rotor là một khối tròn nên khe hở đều. Khe hở trong máy điện không đồng bộ rất nhỏ để hạn chế dòng điện từ hóa lấy từ lưới và như vậy mới có thể làm cho hệ số công suất của máy cao hơn.

### 1.1.3) Ứng dụng:

Máy điện không đồng bộ là loại máy điện xoay chiều chủ yếu dùng làm động cơ điện( đặc biệt là loại rotor lồng sóc) có nhiều ưu điểm hơn so với động cơ DC. Do kết cấu đơn giản, làm việc chắc chắn, hiệu suất cao, giá thành hạ nên động cơ không đồng bộ là loại máy được dùng rộng rãi trong công nghiệp, nông nghiệp , đời sống hằng ngày.

Trong công nghiệp, động cơ không đồng bộ thường được dùng làm nguồn động lực cho các máy cán thép loại vừa và nhỏ, cho các máy công cụ ở các nhà máy công nghiệp nhẹ . . .

Trong nông nghiệp, được dùng làm máy bơm hay máy gia công nông sản phẩm. ...

Trong đời sống hằng ngày, động cơ không đồng bộ ngày càng chiếm một vị trí quan trọng với nhiều ứng dụng như: quạt gió, động cơ trong tủ lạnh, máy quay đĩa, . .

Tóm lại, cùng với sự phát triển của nền sản xuất điện khí hóa và tự động hóa, phạm vi ứng dụng của động cơ không đồng bộ ngày càng rộng rãi.



### **1.2> CÁC PHƯƠNG PHÁP ĐIỀU KHIỂN ĐỘNG CƠ KĐB:**

So với máy điện DC, việc điều khiển máy điện xoay chiều gặp rất nhiều khó khăn bởi vì các thông số của máy điện xoay chiều là các thông số biến đổi theo thời gian, cũng như bản chất phức tạp về mặt cấu trúc máy của động cơ điện xoay chiều so với máy điện một chiều.

Các phương pháp điều khiển phổ biến:

- Điều khiển điện áp stator
- Điều khiển điện trở rôto
- Điều khiển tần số
- Điều khiển công suất trượt rôto

## **CHƯƠNG 2: GIỚI THIỆU VỀ BIẾN TẦN NGUỒN ÁP ĐIỀU KHIỂN $V/f=const$**

### **2.1> BIẾN TẦN NGUỒN ÁP:**

Được sử dụng hầu hết trong các biến tần hiện nay. Tốc độ của động cơ không đồng bộ tỉ lệ trực tiếp với tần số nguồn cung cấp. Do đó, nếu thay đổi tần số của nguồn cung cấp cho động cơ thì cũng sẽ thay đổi được tốc độ đồng bộ, và tương ứng là tốc độ của động cơ.

Tuy nhiên, nếu chỉ thay đổi tần số mà vẫn giữ nguyên biên độ nguồn áp cấp cho động cơ sẽ làm cho mạch từ của động cơ bị bão hòa. Điều này dẫn đến dòng từ hóa tăng, méo dạng điện áp và dòng điện cung cấp cho động cơ gây ra tổn hao lõi từ, tổn hao đồng trong dây quấn Stator. Ngược lại, nếu từ thông giảm dưới định mức sẽ làm giảm moment của động cơ.

Vì vậy, khi giảm tần số nguồn cung cấp cho động cơ nhỏ hơn tần số định mức thường đi đôi với giảm điện áp cung cấp cho động cơ. Và khi động cơ hoạt động với tần số định mức thì điện áp động cơ được giữ không đổi và bằng định mức do giới hạn của cách điện của Stator cũng như của điện áp nguồn cung cấp, moment của động cơ sẽ bị giảm.

### **2.2> PHƯƠNG PHÁP ĐIỀU KHIỂN $V/f$ :**

#### **2.2.1) Phương pháp $E/f$**

Ta có công thức sau:

$$a = \frac{f}{f_{dm}} \quad (2.1)$$

+ Với  $f$ : tần số hoạt động của động cơ,

+  $f_{dm}$ : tần số định mức của động cơ.

Giả sử động cơ hoạt động dưới tần số định mức ( $a < 1$ ). Từ thông động cơ được giữ ở giá trị không đổi. Do từ thông của động cơ phụ thuộc vào dòng từ hóa của động cơ, nên từ thông được giữ không đổi khi dòng từ hóa được giữ không đổi tại mọi điểm làm việc của động cơ.

Ta có phương trình tính dòng từ hóa tại điểm làm việc định mức như sau:

$$I_m = \frac{E_{dm}}{f_{dm}} \cdot \frac{1}{2\pi L_m} \quad (2.2)$$

+ Với  $L_m$  là điện cảm mạch từ hóa

Tại tần số làm việc  $f$ :

$$I_m = \frac{E}{a \cdot f_{dm}} \cdot \frac{1}{2\pi L_m} \quad (2.3)$$

Từ 2 phương trình trên suy ra điều kiện để dòng điện từ hóa không đổi:

$$\frac{E}{a} = E_{đm} \Rightarrow \frac{E}{f} = \frac{E_{đm}}{f_{đm}} = const \quad (2.4)$$

Như vậy từ thông động cơ được giữ không đổi khi tỉ lệ  $E/f$  được giữ không đổi ( $E/f = const$ ).

### 2.2.2) Phương pháp $V/f$

Tuy nhiên trong thực tế, việc giữ từ thông không đổi đòi hỏi mạch điều khiển rất phức tạp. Nếu bỏ qua sụt áp trên điện trở và điện kháng tản mạch stator, ta có thể xem như  $U \approx E$ . Khi đó nguyên tắc điều khiển  $E/f=const$  được thay bằng phương pháp  $V/f=const$ .

Trong phương pháp  $V/f=const$  (gọi ngắn là  $V/f$ ), như đã trình bày ở trên thì tỉ số  $V/f$  được giữ không đổi và bằng giá trị tỉ số này ở định mức.

Ta có công thức moment định mức ứng với sơ đồ đơn giản của động cơ:

$$M = \frac{3}{\omega_{đb}} \cdot \left[ \frac{V_{đm}^2 \cdot \frac{R_2'}{s}}{\left( R_1 + \frac{R_2'}{s} \right)^2 + (X_1 + X_2')^2} \right] \quad (2.5)$$

Và moment cực đại ở chế độ định mức:

$$M_{\max} = \frac{3}{2 \cdot \omega_{đb}} \cdot \left[ \frac{V_{đm}^2}{R_1 \pm \sqrt{R_1^2 + (X_1 + X_2')^2}} \right] \quad (2.6)$$

Khi thay các giá trị định mức bằng giá trị đó nhân với tỉ số  $a$  ( $a\omega_{đm}$ ,  $aV_{đm}$ ,  $aX$ ), Ta có được công thức moment của động cơ ở tần số  $f$  khác định mức:

$$M = \frac{3}{\omega_{đb}} \cdot \left[ \frac{V_{đm}^2 \cdot \frac{R_2'}{a \cdot s}}{\left( \frac{R_1}{a} + \frac{R_2'}{as} \right)^2 + (X_1 + X_2')^2} \right]; (a < 1) \quad (2.7)$$

Và moment cực đại ở tần số  $f$  khác định mức:

$$M_{max} = \frac{3}{2 \cdot \omega_{db}} \left[ \frac{V_{đm}^2}{\frac{R_1}{a} \pm \sqrt{\left(\frac{R_1}{a}\right)^2 + (X_1 + X_2')^2}} \right], a < 1 \quad (2.8)$$

Dựa theo công thức trên ta thấy, các giá trị  $X_1$  và  $X_2'$  phụ thuộc vào tần số, trong khi  $R_1$  lại là hằng số. Như vậy, khi hoạt động ở tần số cao, giá trị  $(X_1 + X_2') \gg R_1/a$ , sụt áp trên  $R_1$  rất nhỏ nên giá trị E suy giảm rất ít dẫn đến từ thông được giữ gần như không đổi. Moment cực đại của động cơ gần như không đổi.

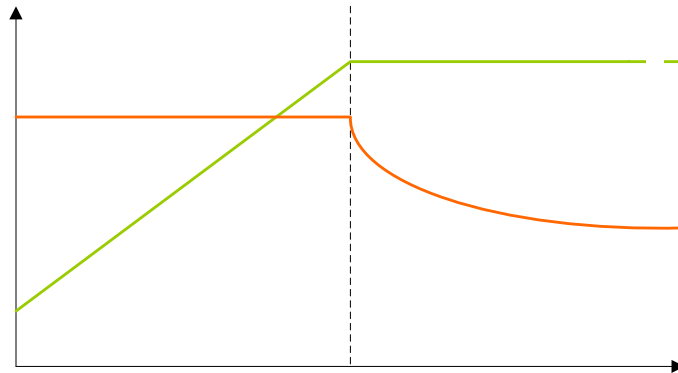
Tuy nhiên, khi hoạt động ở tần số thấp thì giá trị điện trở  $R_1/a$  sẽ tương đối lớn so với giá trị của  $(X_1 + X_2')$ , dẫn đến sụt áp nhiều ở điện trở stator khi moment tải lớn. Điều này làm cho E bị giảm và dẫn đến suy giảm từ thông và moment cực đại.

Để bù lại sự suy giảm từ thông ở tần số thấp. Ta sẽ cung cấp thêm cho động cơ một điện áp  $U_0$  để cung cấp cho động cơ từ thông định mức khi  $f=0$ . Từ đó ta có quan hệ như sau:

$$U = U_0 + K \cdot f$$

Với K là một hằng số được chọn sao cho giá trị U cấp cho động cơ bằng  $U_{đm}$  tại  $f=f_{đm}$ .

Khi  $a > 1$  ( $f > f_{đm}$ ), Điện áp được giữ không đổi và bằng định mức. Khi đó động cơ hoạt động ở chế độ suy giảm từ thông.



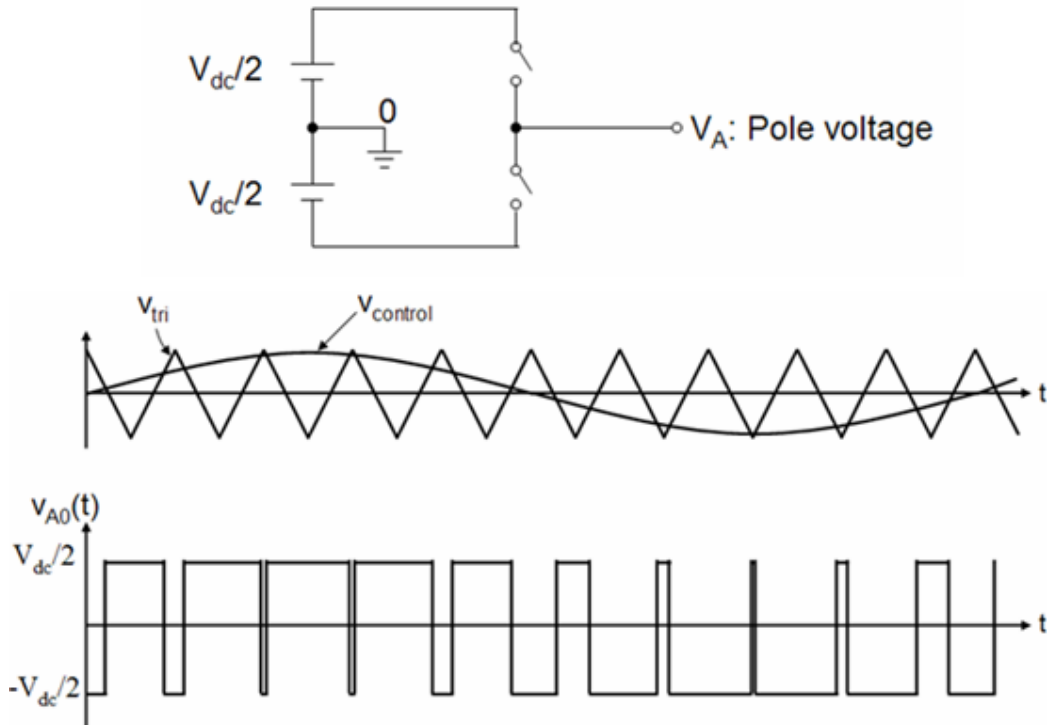
Hình 2.1: đồ thị biểu diễn mối quan hệ giữa moment và điện áp theo tần số trong phương pháp điều khiển  $V/f=const$ .

## 2.3> PHƯƠNG PHÁP ĐIỀU CHẾ SIN PWM:

### 2.3.1) Giới thiệu:

Để tạo ra một điện áp xoay chiều bằng phương pháp SIN PWM, ta sử dụng một tín hiệu xung tam giác tần số cao đem so sánh với một điện áp sin chuẩn có tần

số f. Nếu đem xung điều khiển này cấp cho một bộ biến tần một pha thì đó ngõ ra sẽ thu được một dạng điện áp dạng điều rộng xung có tần số bằng với tần số nguồn sin mẫu và biên độ hài bậc nhất phụ thuộc vào nguồn điện một chiều cung cấp và tỉ số giữa biên độ sóng sin mẫu và sóng mang. Tần số sóng mang phải lớn hơn tần số của sóng sin mẫu. Sau đây là hình vẽ miêu tả nguyên lý của phương pháp điều rộng xung sin một pha:



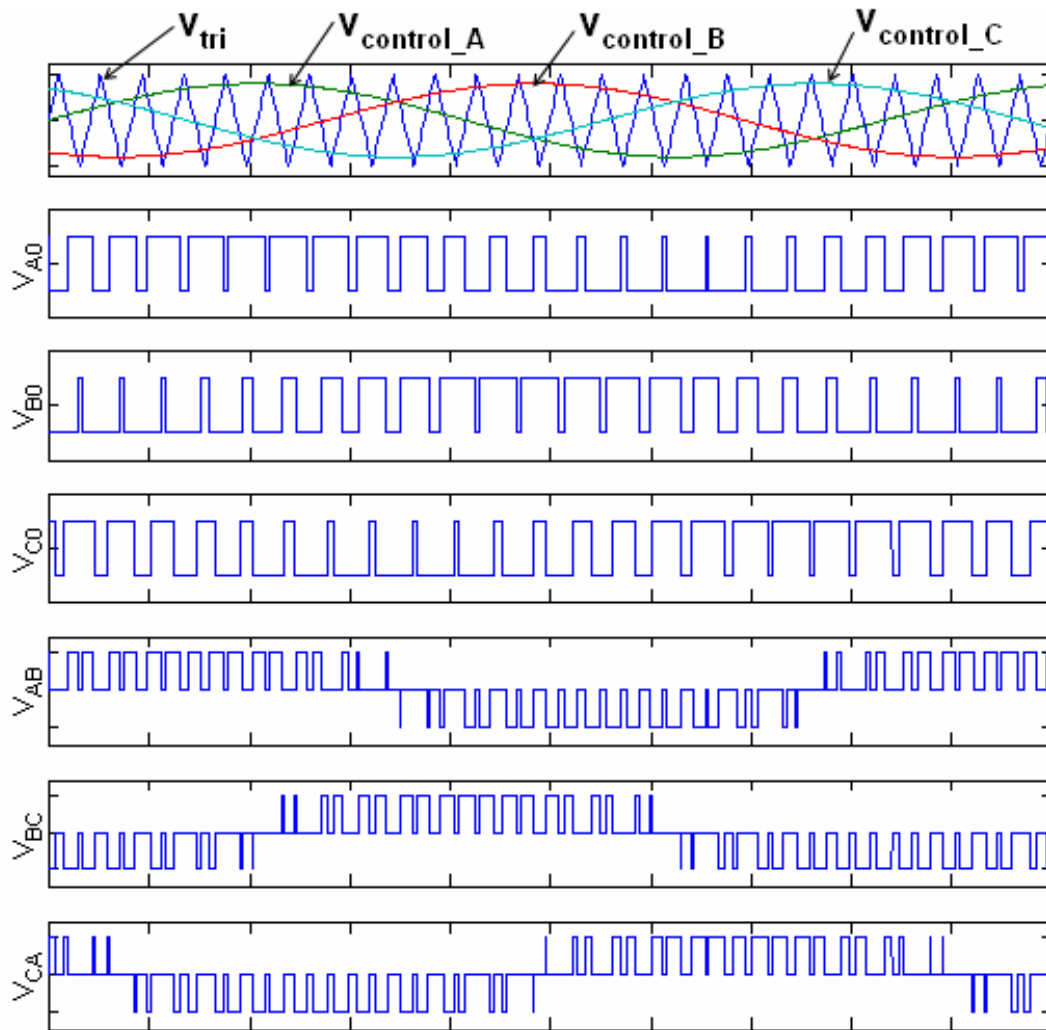
Hình 2.2: nguyên lý của phương pháp điều rộng xung SIN một pha

Khi:

$$V_{control} > V_{tri} \text{ thì } V_{AO} = \frac{V_{dc}}{2} \quad (2.9)$$

$$V_{control} < V_{tri} \text{ thì } V_{AO} = -\frac{V_{dc}}{2}$$

Như vậy, để tạo ra nguồn điện 3 pha dạng điều rộng xung, ta cần có nguồn sin 3 pha mẫu và giản đồ kích đóng của 3 pha sẽ được biểu diễn như hình vẽ dưới đây:



Hình 2.3: nguyên lý của phương pháp điều rộng SIN 3 pha và dạng sóng điện áp ngõ ra

### 2.3.2) Các công thức tính toán:

Ta cần tính được biên độ hài bậc nhất của điện áp ngõ ra từ tỉ số biên độ giữa sóng mang và sóng tam giác

Ta có công thức sau tính biên độ của hài bậc nhất:

$$U_{SIN(1)} = ma \cdot \frac{U_{DC}}{2} \quad (2.10)$$

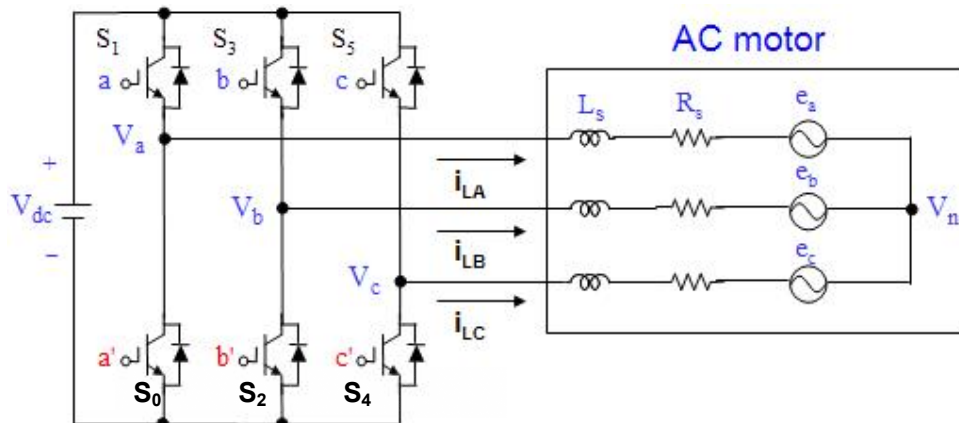
Trong đó ma là tỉ số giữa biên độ sóng sin mẫu và biên độ sóng mang – còn gọi là tỉ số điều biên.

$$ma = \frac{U_{SINsmp}}{U_{carry}} \quad (2.11)$$

**2.3> PHƯƠNG PHÁP ĐIỀU CHẾ VECTOR KHÔNG GIAN ( SVM)**

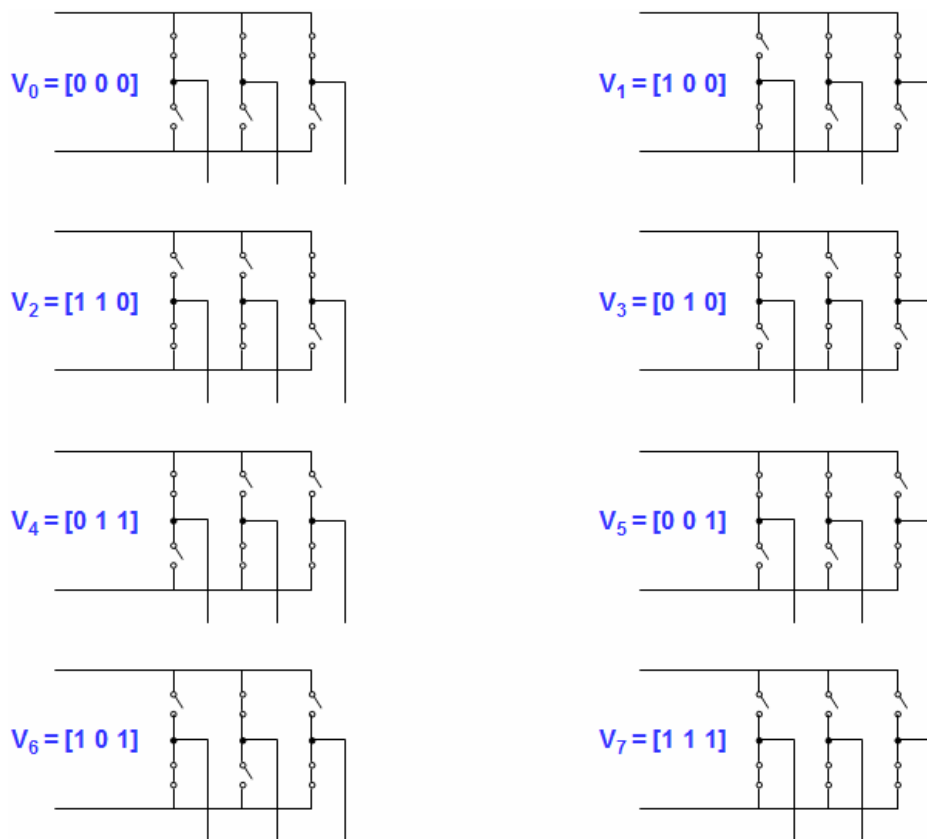
**2.3.1) giới thiệu chung:**

Sau đây là sơ đồ nguyên lý của bộ biến tần sử dụng 6 khóa transistor công suất :



Hình 2.4: Sơ nguyên lý đồ bộ nghịch lưu 3 pha

Đối với phương pháp điều chế xung vector không gian, bộ nghịch lưu được xem như là một khối duy nhất với 8 trạng thái đóng ngắt riêng biệt từ 0 đến 7.



Hình 2.5: Trạng thái đóng ngắt các khóa bộ nghịch lưu

Bảng tóm tắt :

Vector điện áp	Trạng thái của các khóa			Điện áp pha			Điện áp dây		
	Q1	Q3	Q5	V <sub>an</sub>	V <sub>bn</sub>	V <sub>cn</sub>	V <sub>ab</sub>	V <sub>bc</sub>	V <sub>ca</sub>
<b>V0</b>	0	0	0	0	0	0	0	0	0
<b>V1</b>	1	0	0	2/3	-1/3	-1/3	1	0	-1
<b>V2</b>	1	1	0	1/3	1/3	-2/3	0	1	-1
<b>V3</b>	0	1	0	-1/3	2/3	-1/3	-1	1	0
<b>V4</b>	0	1	1	-2/3	1/3	1/3	-1	0	1
<b>V5</b>	0	0	1	-1/3	-1/3	2/3	0	-1	1
<b>V6</b>	1	0	1	1/3	-2/3	1/3	1	-1	0
<b>V7</b>	1	1	1	0	0	0	0	0	0

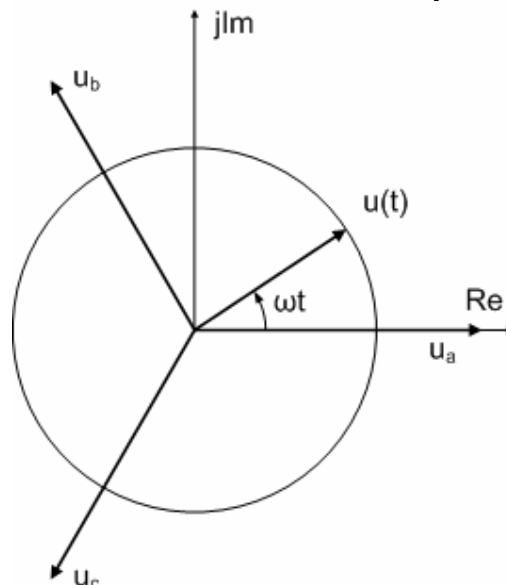
Ghi chú: độ lớn điện áp phải nhân với VDC

**2.3.2) Sơ đồ sắp xếp các vector V0 -> V7 trên trục Va; Vb; Vc**

Đối với nguồn áp ba pha cân bằng, ta luôn có phương trình sau:

$$u_a(t) + u_b(t) + u_c(t) = 0 \quad (2.12)$$

Và bất kỳ ba hàm số nào thỏa mãn phương trình trên đều có thể chuyển sang hệ tọa độ 2 chiều vuông góc. Ta có thể biểu diễn phương trình trên dưới dạng 3 vector gồm:  $[u_a \ 0 \ 0]^T$  trùng với trục x, vector  $[0 \ u_b \ 0]^T$  lệch một góc  $120^\circ$  và vector  $[0 \ 0 \ u_c]^T$  lệch một góc  $240^\circ$  so với trục x như hình sau đây.



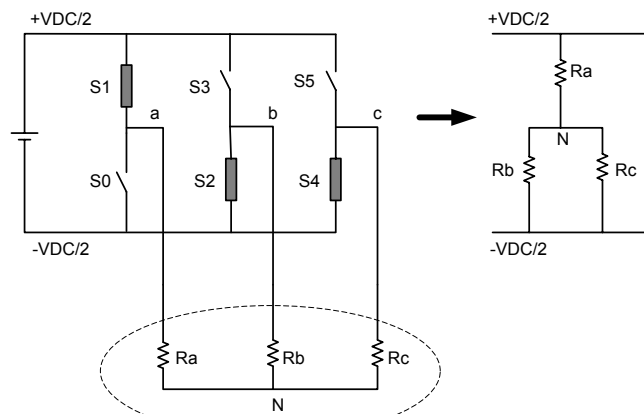
Hình 2.6: Biểu diễn vector không gian trong hệ tọa độ x-y



Từ đó ta xây dựng được phương trình của vector không gian trong hệ tọa độ phức như sau

$$u(t) = \frac{2}{3} \left( u_a + u_b \cdot e^{j(2/3)\pi} + u_c \cdot e^{-j(2/3)\pi} \right) \quad (2.13)$$

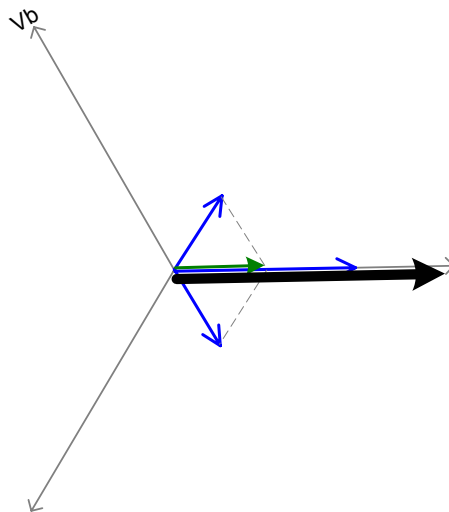
+ Ta xét trường hợp bộ nghịch lưu ở trạng thái đầu V1 :



Hình 2.7: Bộ nghịch lưu ở trạng thái V1

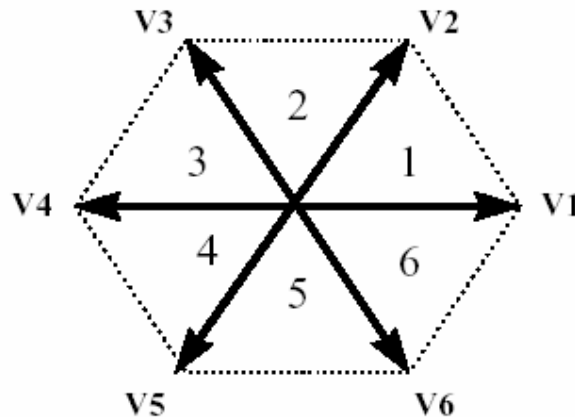
Ta có:  $R_a \approx R_b \approx R_c \Rightarrow V_a = 2/3 V_{dc} ; V_b = V_c = -1/3 V_{dc}$

Xét trên hệ tọa độ  $\alpha - \beta$  : trong đó  $\vec{V}_s = \vec{V}_1 = K * (\vec{V}_a + \vec{V}_b + \vec{V}_c)$ ;  $K=2/3$  là hệ số biên hình



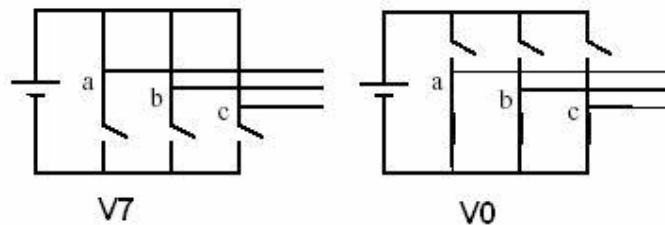
Hình 2.8: Vector điện áp V1 trên tọa độ  $\alpha - \beta$

+ Tương tự như vậy với các vector  $V_2 \rightarrow V_6$ , ta có giản đồ sau:



Hình 2.9: Vector điện áp  $V_1 \rightarrow V_6$  trên giản đồ  $\alpha - \beta$

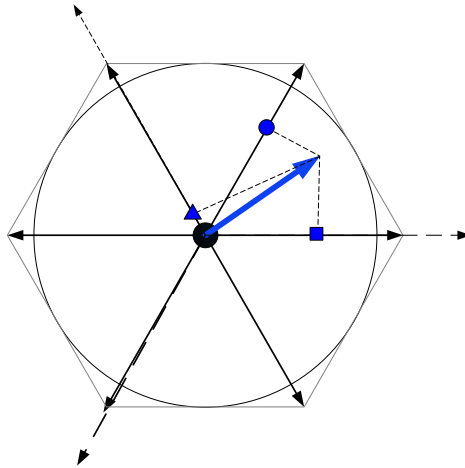
+ Ngoài ra, chúng ta còn 2 trường hợp đặc biệt là vector  $V_0 = V_7 = 0$



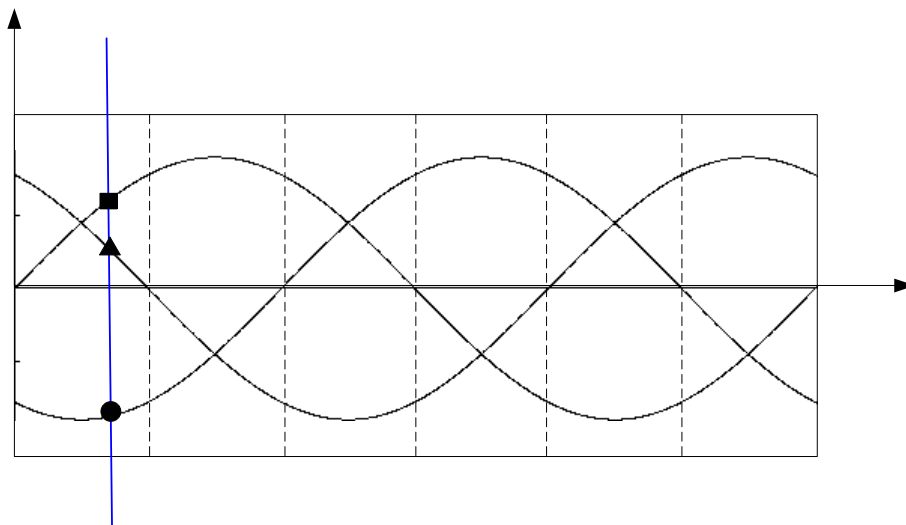
Hình 2.10 :  $V_7$  &  $V_0$

### 2.3.2) Giới thiệu vector $V_s$ :

Ý tưởng của việc điều chế vector không gian là tạo nên sự dịch chuyển liên tục của *vector không gian tương đương* của vector điện áp bộ nghịch lưu trên quỹ đạo đường tròn, tương tự như trường hợp của vector không gian của đại lượng 3 pha hình sin tạo được. Với sự dịch chuyển của đầu đạnh của vector không gian trên quỹ đạo tròn các sóng hài bậc cao được loại bỏ và biên độ áp ra trở nên tuyến tính. Vector tương đương ở đây chính là vector trung bình trong thời gian một chu kỳ lấy mẫu  $T_s$  của quá trình điều khiển bộ nghịch lưu áp



Hình 2.11: Vector  $V_s$  trên hệ trục  $\alpha - \beta$

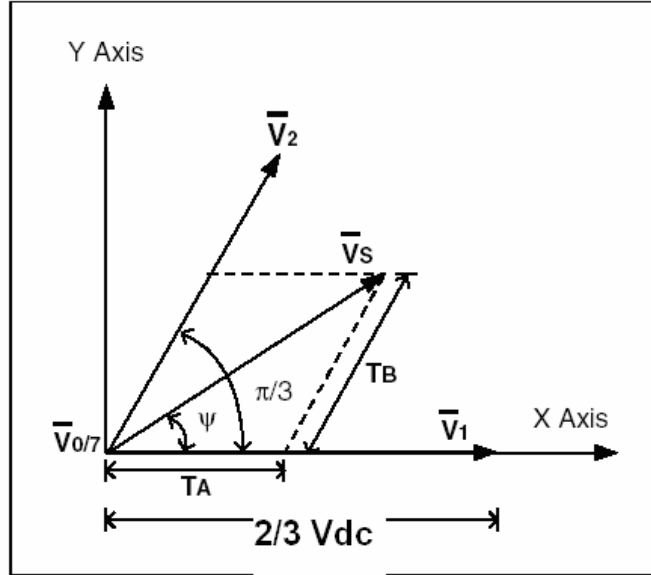


Hình 2.12: Điện áp 3 pha ngõ ra trong miền thời gian tương ứng Hình 2.11

Vector  $\vec{V}_s$  liên quan đến các trạng thái khóa transistor trong bộ biến tần nguồn áp VSI ( Voltage Source Inverter). Trong phương pháp SVM thì VSI được đóng ngắt ở tần số rất lớn ( $F_{PWM}$ ).  $F_{PWM}$  quyết định thời gian lấy mẫu  $T_s$  cho vector  $\vec{V}_s$  ( $T_s=1/F_{PWM}$ )

Có rất nhiều cách đóng ngắt các khóa BJT để tạo ra vector  $\vec{V}_s$  từ các vector  $\vec{V}_0; \vec{V}_1; \vec{V}_2; \vec{V}_3; \vec{V}_4; \vec{V}_5; \vec{V}_6; \vec{V}_7$ .

**2.3.3) Cách tính toán thời gian để tạo ra vector  $\vec{V}_s$  :**



Hình 2.13:  $V_s$  ở sector 1

Xét góc 1 phần sáu đầu tiên của hình lục giác được tạo bởi đỉnh của ba vector  $\vec{V}_0$ ;  $\vec{V}_1$ ;  $\vec{V}_2$ . Giả sử trong khoảng thời gian  $T_s$ , ta cho tác dụng vector  $\vec{V}_1$  trong khoảng thời gian  $T_A$ , vector  $\vec{V}_2$  trong khoảng thời gian  $T_B$ ; vector  $\vec{V}_0$  trong khoảng thời gian còn lại trong chu kỳ lấy mẫu ( $T_s - T_A - T_B$ ). Vector tương đương được tính bằng vector trung bình của chuỗi tác động liên tiếp trên:

$$\vec{V}_s = \left( \frac{T_A}{T_s} \vec{V}_1 \right) + \left( \frac{T_B}{T_s} \vec{V}_2 \right) + \left( \frac{T_{0/7}}{T_s} \vec{V}_{0/7} \right) \quad (2.14)$$

$$T_s = T_A + T_B + T_{0/7} \quad (2.15)$$

Ta có tỉ lệ biên độ được định nghĩa như sau :

$$m = \frac{V_s}{\frac{2}{3} V_{dc}} \quad (2.16)$$

+ trong đó  $V_s$  điện áp (pha) ngõ ra của bộ biến tần ( $V_a, V_b, V_c$ )

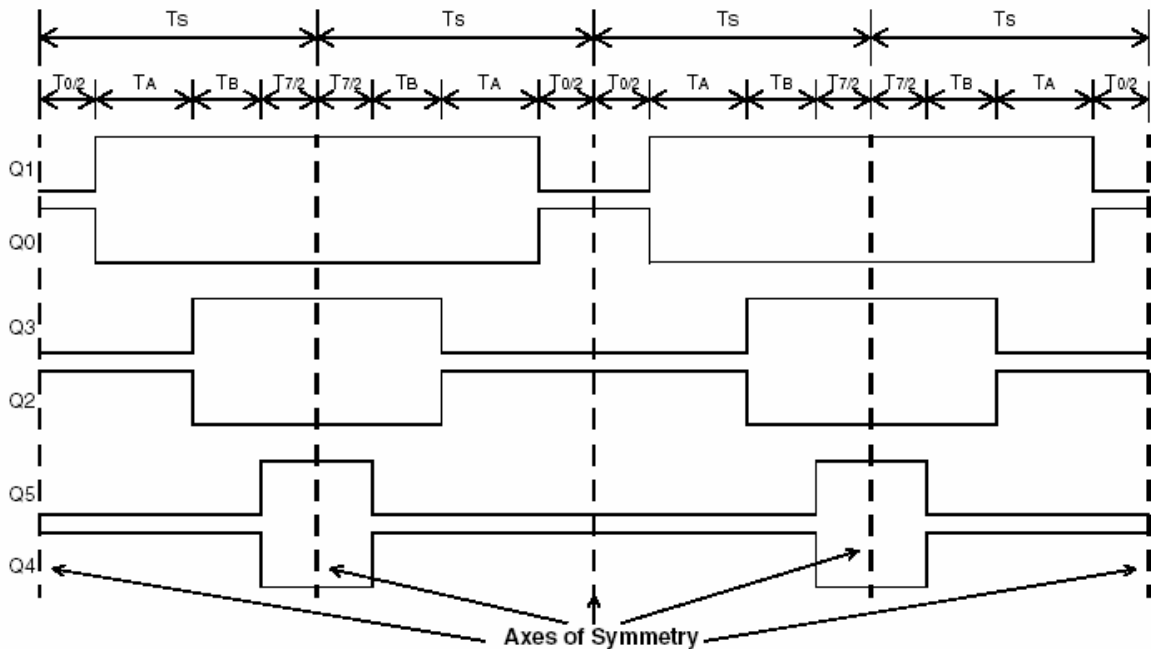
Chiếu phương trình (2.14) lên trục X - Y ; sử dụng thêm phương trình (2.16) và tỉ số m (2.15)

$$\begin{cases} T_1 = T_s \cdot \frac{2}{\sqrt{3}} m \cdot \sin(\pi/3 - \Psi) \\ T_2 = T_s \cdot \frac{2}{\sqrt{3}} m \cdot \sin(\Psi) \\ T_{0-7} = T_s - T_1 - T_2 \end{cases} \quad (2.17)$$

=> Như vậy trong khoảng thời gian lấy mẫu  $T_s$ , thời gian tồn tại của các trạng thái  $T_A$ ;  $T_B$ ;  $T_{0/7}$  dựa vào tỉ số  $m$  và góc pha  $\Psi$  của vector  $V_s$  ( hay nói cách khác là dựa vào **độ lớn** và **vị trí** của vector  $V_s$  trong không gian)

**2.4> KỸ THUẬT ĐIỀU CHẾ VECTOR KHÔNG GIAN:**

Thông thường, một trong những tiêu chuẩn để lựa chọn giải đồ đóng kích linh kiện là sao cho giảm thiểu tối đa số lần chuyển mạch của linh kiện => giảm tổn hao trong quá trình đóng ngắt chúng. Số lần chuyển mạch sẽ ít nếu ta thực hiện trình tự điều khiển sau:

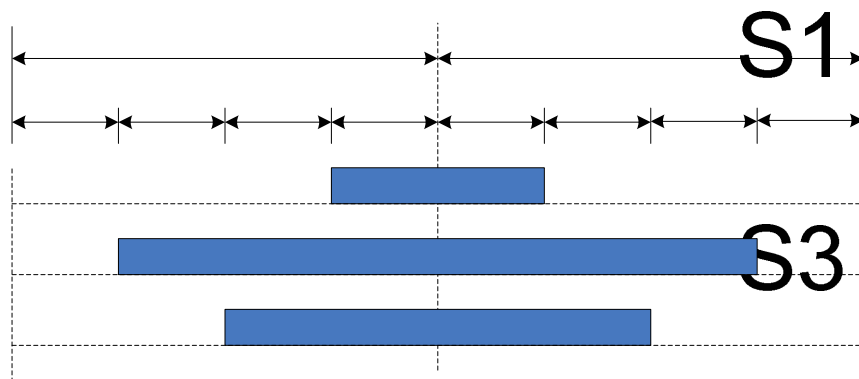
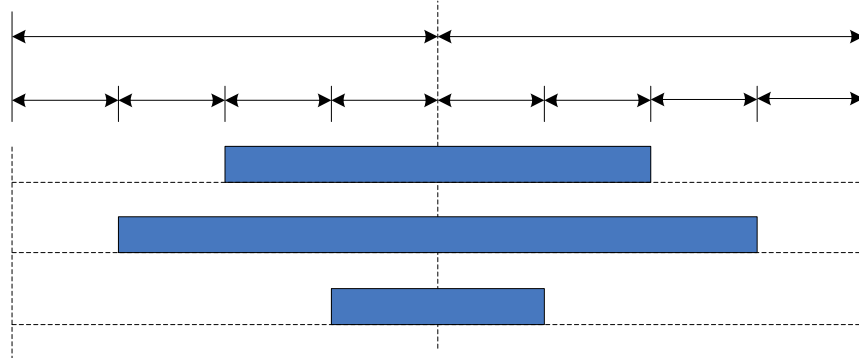
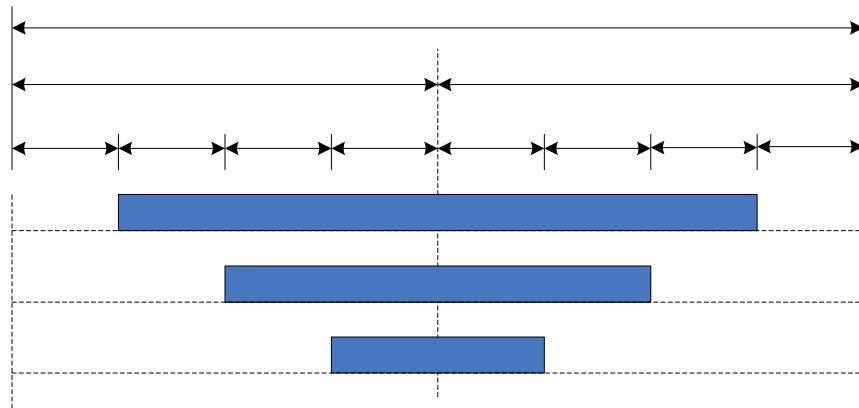


Hình 2.14: Giải đồ đóng ngắt linh kiện

**2.4.1) Giải đồ đóng ngắt các khóa để tạo ra Vector  $V_s$  trong từng sector:**

Các khóa công suất trong từng nhánh đóng ngắt đối nghịch nhau. Để đơn giản hóa sơ đồ, ta chỉ vẽ trạng thái của 3 khóa công suất phía trên. Ba khóa còn lại có trạng thái đối nghịch với 3 khóa trên theo từng cặp như sau :

- + S0 – S1
- + S2 – S3
- + S4 – S5



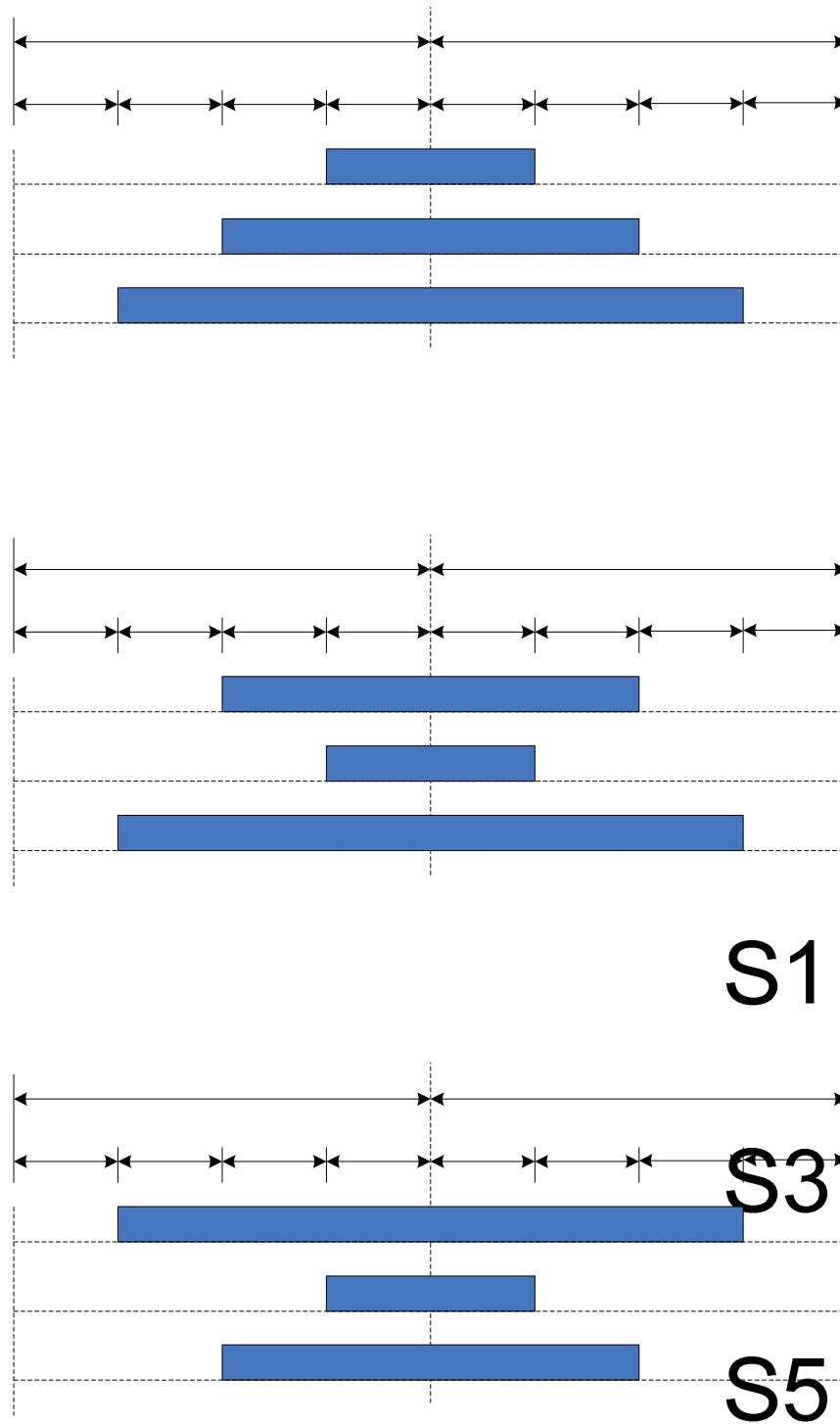
To/

S1

S3

S5

V0

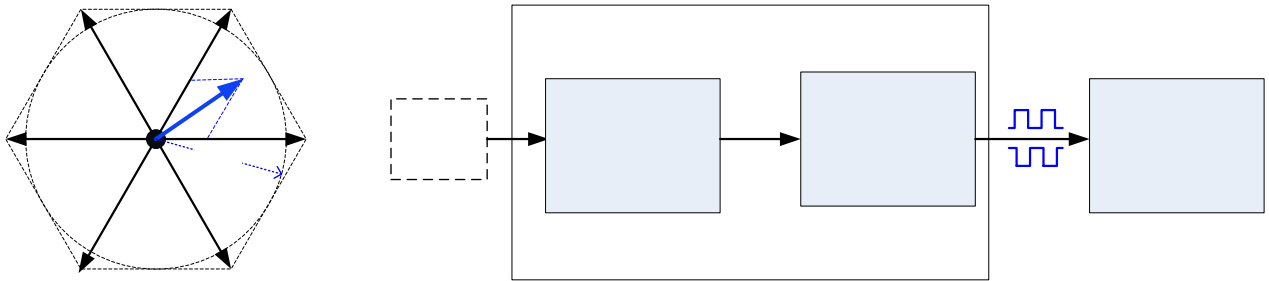


To/

Hình 2.15 : Giảm đồ đồng ngắt các khóa khi  $V_s$  ở sector 1- $\rightarrow$  6

V0

**2.4.2) Sơ đồ tóm tắt của quá trình điều chế :**



Hình 2.15: Sơ đồ tóm tắt của quá trình điều chế

Như vậy vector trung bình ( $V_s$ ) được điều khiển theo quỹ đạo đường tròn. Chiều quay có thể thuận hay nghịch theo chiều kim đồng hồ. Đường tròn nội tiếp hình lục giác là quỹ đạo của vector ko gian lớn nhất mà phương pháp điều chế vector không gian của bộ nghịch lưu áp hai bậc có thể đạt được trong phạm vi điều khiển tuyến tính. Bán kính đường tròn này chính bằng biên độ thành phần cơ bản điện áp (pha) tải

Hay 
$$|\vec{V}_s| = |\vec{V}_A| = |\vec{V}_B| = |\vec{V}_C| = \frac{V_{dc}}{\sqrt{3}}$$

$$\begin{cases} T_1 = T_s \cdot \frac{2}{\sqrt{3}} m \cdot \sin(\pi/3 - \Psi) \\ T_2 = T_s \cdot \frac{2}{\sqrt{3}} m \cdot \sin(\Psi) \\ T_{0-7} = T_s - T_1 - T_2 \end{cases}$$

Trong đó:

- +  $m = \frac{V_s}{\frac{2}{3}V_{dc}}$  là tỉ số điều biên
- +  $T_s$  là chu kỳ điều rộng xung
- +  $\Psi$  là góc lệch giữa  $V_A$  và  $V_B$

$$\begin{matrix} 2 & V_s \\ & 1 \\ & (2.18) \end{matrix}$$

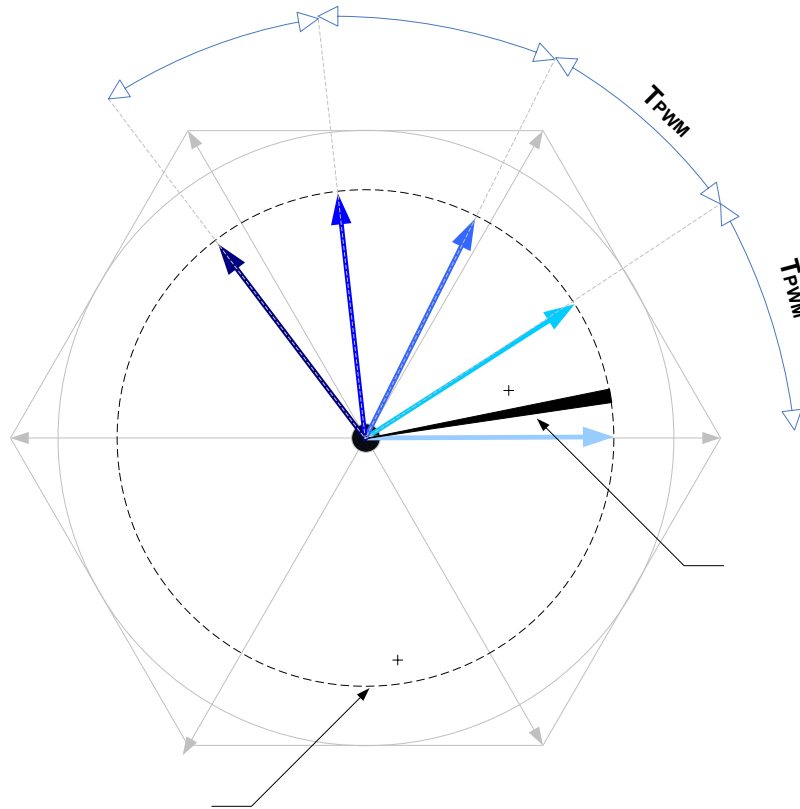
$$\begin{matrix} (2/3) \cdot V_{dc} \\ 6 \\ 5 \end{matrix}$$

V5

V6



**2.4.3) Tính toán góc update của vector  $V_s$  theo phương pháp điều khiển  $V/f$ :**



Hình 2.16: góc update của vector  $V_s$

- 1) Đầu tiên ta chia các sector (mỗi sector  $60^\circ$ ), thành  $n$  phần bằng nhau:  
 => Góc chia nhỏ nhất trong 1 sector:

$$\alpha_{\min} = \frac{60}{n} \text{ (độ)} \quad (2.19)$$

- 2) Tại tần số đặt  $f \Rightarrow (T=1/f)$ :

Vector  $V_s$  quay  $360^\circ$  trong thời gian  $T$   
 Vector  $V_s$  quay  $?$  trong thời gian  $T_{PWM}$

$$\Rightarrow \alpha' = \frac{T_{PWM}}{T} 360^\circ \text{ (độ) : góc update của vector } V_s \quad (2.20)$$

- 3) Xây dựng  $\alpha' = K * \alpha_{\min} = update\_angle$  ( $K$  là số nguyên)

$$\Rightarrow \frac{T_{PWM}}{T} 360^\circ = K * \frac{60}{n}$$

$$\Rightarrow K = \frac{T_{PWM}}{T} * 360 * \frac{n}{60}$$

Mà  $T=1/f$

$$\Rightarrow K = \frac{1}{f_{PWM}} * 360 * \frac{n}{60} * f$$

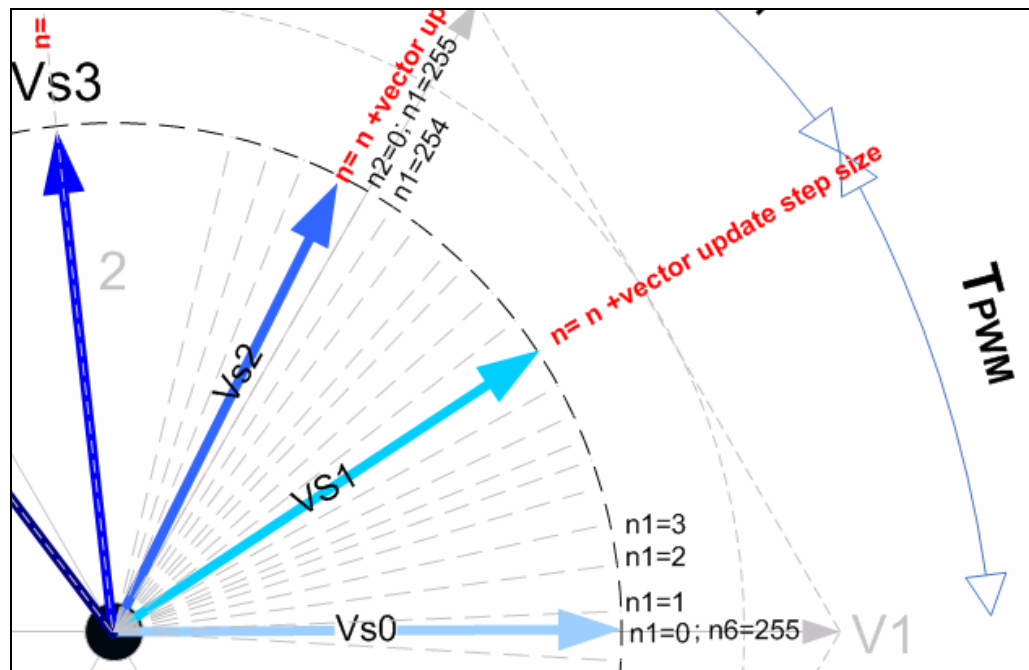
Ta chọn  $T_{PWM}= 5 \text{ KHz}$  ;  $n=512$  giá trị trong 1 sector

$$\Rightarrow K = \frac{1}{5000} * 360 * \frac{512}{60} * f$$

$$\Rightarrow K = 0.6144 f = \text{step size} \quad (2.21)$$

Ta có tần số  $f$  đặt thay đổi từ  $0 \rightarrow 60 \text{ Hz}$

$$\Rightarrow K = (0 \rightarrow 36.684)$$



Hình 2.17: Update vector  $V_s$  with stepsize

Vậy góc của  $V_s$  được tính bởi công thức sau :

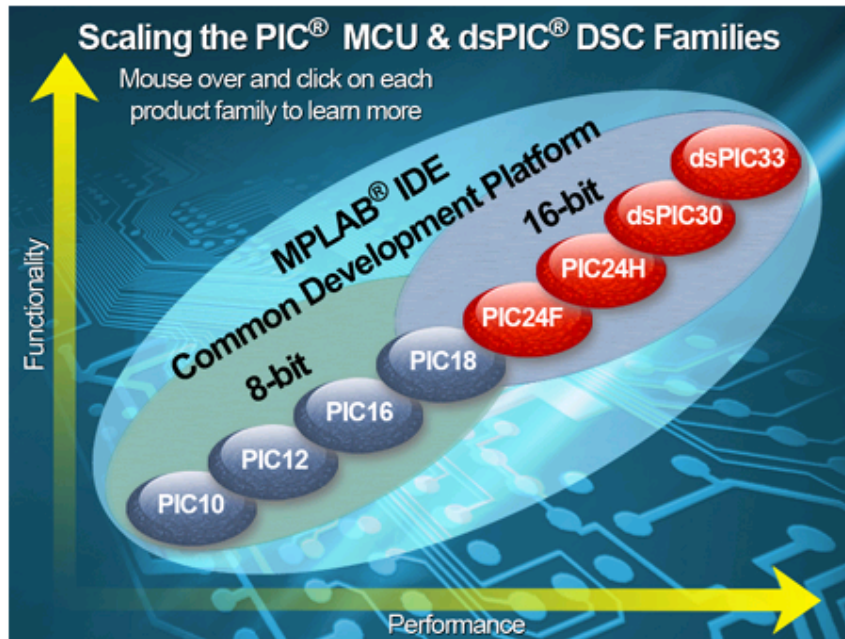
Vector update step size = DEGREE\_CONSTANT x required Motor Speed (Hz)


$$\text{Vector angle} = \text{Vector angle} + \text{Update\_angle}$$

$$(2.22)$$

### CHƯƠNG 3: GIỚI THIỆU VỀ PIC<sup>®</sup> Microcontrollers (MCUs)

#### 3.1>TỔNG QUAN:



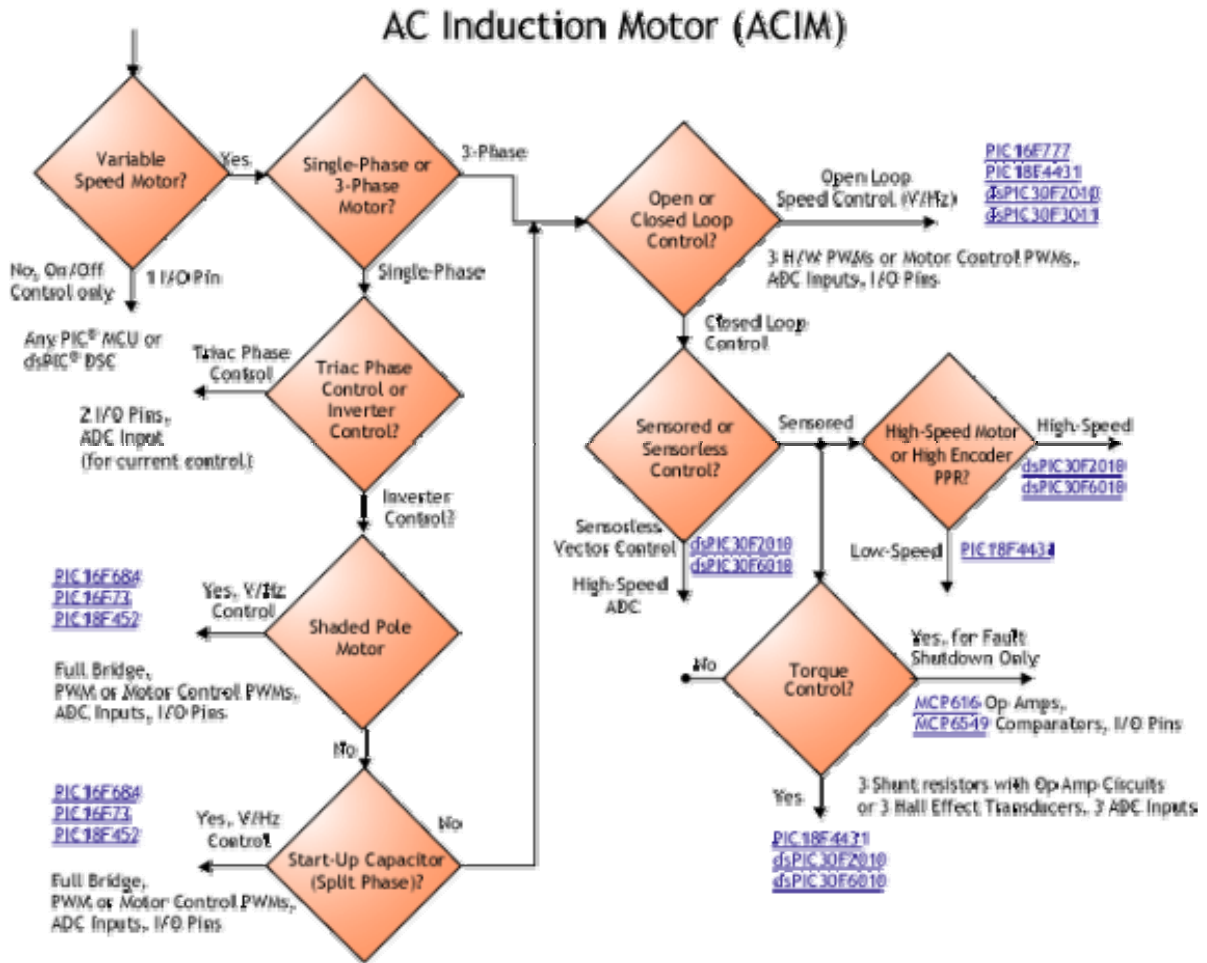
Họ vi điều khiển PIC và dsPIC do hãng  MICROCHIP chế tạo và sản xuất với công nghệ hiện đại, phù hợp cho các ứng dụng đơn giản cho đến phức tạp. Đặc biệt ngoài ngôn ngữ lập trình assembler như các MCU khác, người dùng có thể lập trình PIC trên ngôn ngữ C quen thuộc thông qua các phần mềm hỗ trợ ( PIC18C ; CCS C ; .....)

Gồm các họ như sau:

- 8 bit:
  - + PIC10
  - + PIC12
  - + PIC16
  - + PIC18
- 16 bit:
  - + PIC24F
  - + PIC24H
  - + dsPIC30
  - + dsPIC33

## CHƯƠNG 3: GIỚI THIỆU VỀ PIC<sup>®</sup> Microcontrollers (MCUs)

Tùy theo các ứng dụng cụ thể mà người dùng có thể chọn ra Chip phù hợp (theo hướng dẫn của nhà sản xuất tại trang chủ của microchip). Trong đó PIC18F4431 là IC chuyên dùng để điều khiển động cơ 3 pha theo đề nghị của của Microchip



### 3.1.1> Những đặc điểm nổi bật PIC18F4431:

- **14 bit Power Control PWM module:**
  - + Có đến 4 kênh ( mỗi kênh gồm 1 cặp xung đối nghịch)
  - + Thời gian dead time linh hoạt
  - + update từng duty cycle => ngõ ra PWM đáp ứng nhanh
  - +.....
- **Motion Feedback Module:**
  - + Có 3 kênh capture độc lập:
    - các chế độ hoạt động linh hoạt cho việc đo đặc độ rung xung
    - Module hỗ trợ Hall Sensor
    - Special event trigger cho các module khác
  - + Quadrature Encoder interface:
    - 2 pha vào và 1 ngõ vào index từ encoder
    - hỗ trợ đo đặc vận tốc
- **High speed, 200Ksps 10-bit A/D Converter:**
  - + Có 9 kênh A/D
  - + 2 kênh lấy mẫu tức thời
  - + Lấy mẫu liên tục: 1 ; 2 hay 4 kênh được lựa chọn
  - + .....
- **Flexible Oscillator Structure:**
  - + 4 chế độ thạch anh ( hỗ trợ đến 40 MHz)
  - + 2 nguồn xung lock ngoài lên đến 40 MHz
  - + Chế độ thạch anh nội :
    - Có 8 tần số người dùng có thể lựa chọn : từ 31Khz -> 8 MHz
    - OSCTUNE có thể bù cho sự lệch tần số (?)
  - +.....
- **Peripheral Highlights:**
  - + Chịu dòng cao : sink/source ( 25mA/25ma)
  - + 3 nguồn ngắt ngoài
  - + 2 module Capture / Compare / PWM (CCP)
    - Capture 16 bit, độ phân giải tối đa 6.25 ns (  $T_{CY}/6$ )
    - Compare 16 bit, độ phân giải tối đa 100 ns (  $T_{CY}$ )
    - PWM output: độ phân giải từ 1 -> 10 bit
  - + Module USART:
    - Hỗ trợ RS-485, RS-232 và LIN1.2
    - Auto weak-up on start bit
    - Auto-Bound detect
  - + RS-232 sử dụng khối dao động nội ( ko cần thạch anh ngoài)

### 3.1.2> Những đặc điểm chính:

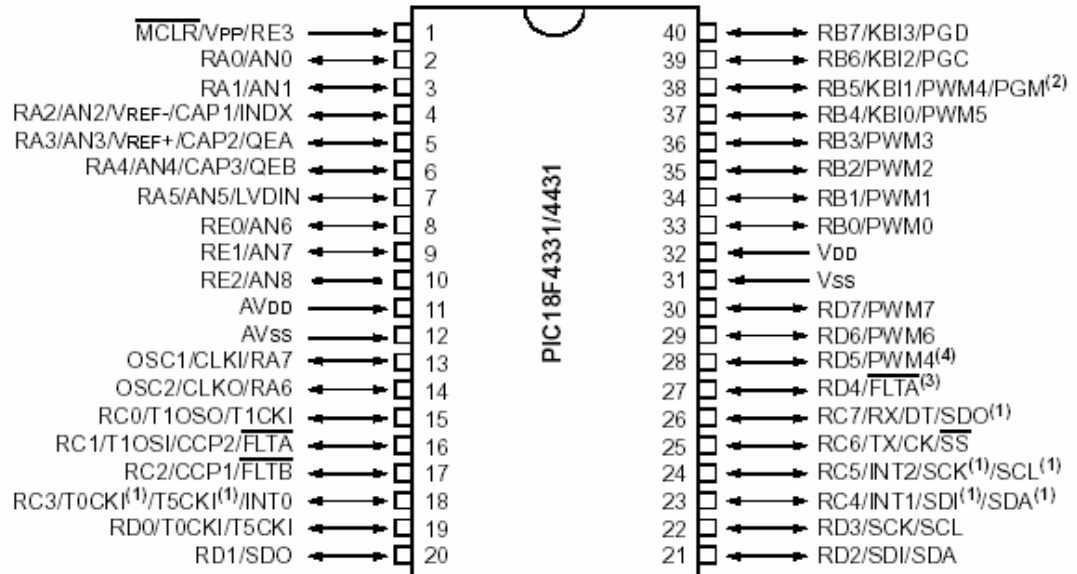
- + Là CPU sử dụng tập lệnh RISC và có tốc độ xử lý cao , công suất thấp nhờ sử dụng công nghệ CMOS FLASH/EEPROM.
- + Tập lệnh có 75 lệnh .
- + Một chu kỳ lệnh bằng 4 chu kỳ xung . Sử dụng bộ dao động 40 Mhz thì chu kỳ lệnh là 0,1 us .
- + Tần số bộ dao động cho phép tới 40Mhz.
- + **8K x 14** word bộ nhớ FLASH lập trình.
- + **768** byte bộ nhớ RAM , trong đó bộ nhớ EEPROM lên đến **256 byte**.
- + Trang bị tới 34 ngắt với 8 cấp độ ngắt
- + 5 port I / O.
- + Trang bị 3 bộ định thời: 2 bộ 8 bit, 1 bộ 16 bit.
- + 2 module Capture/Compare/PWM.
- + Bộ chuyển đổi 10 bit ADC với tốc độ 5-10us.
- + Cổng serial đồng bộ với chế độ SPI(Master) và I2C (Master/Slave) thực hiện bằng phần cứng .
- + Chế độ chuyển nhận đồng bộ/bất đồng bộ với 9 bit địa chỉ kiểm tra.
- + Cổng song song (PSP) 8bit .
- + Các chế độ định địa chỉ:trực tiếp , gián tiếp , và tương đối.
- + Cho phép đọc/ghi bộ nhớ chương trình .
- + Có chế độ bảo vệ mã lập trình .
- + Chế độ SLEEP(tạm nghỉ) để tiết kiệm điện năng .
- + Cho phép chọn lựa chế độ dao động ( nội , ngoại ).
- + 2 chân cho phép gỡ rối hoạt động của vi điều khiển.
- + Lập trình thông qua cổng serial với điện thế chỉ 5 V.
- + Tầm điện thế hoạt động rộng: từ 2 đến 5.5V. Dòng cấp khoảng 25mA.
- + Được sản xuất với nhiều loại khác nhau cho cùng 1 mã vi điều khiển , tùy thuộc vào số tính năng được trang bị thêm . Các kiểu đế cắm:PDIP(40 chân), PLCC và QFP (cùng 44 chân).

## CHƯƠNG 3: GIỚI THIỆU VỀ PIC® Microcontrollers (MCUs)

### 3.2> TÓM TẮT TRÚC PHẦN CỨNG:

#### 3.2.1> Sơ đồ chân MCU PIC18F4431 :

40-Pin PDIP



**Note 1:** RC3 is the alternate pin for T0CKI/T5CKI; RC4 is the alternate pin for SDI/SDA; RC5 is the alternate pin for SCK/SCL.

**2:** Low-voltage programming must be enabled.

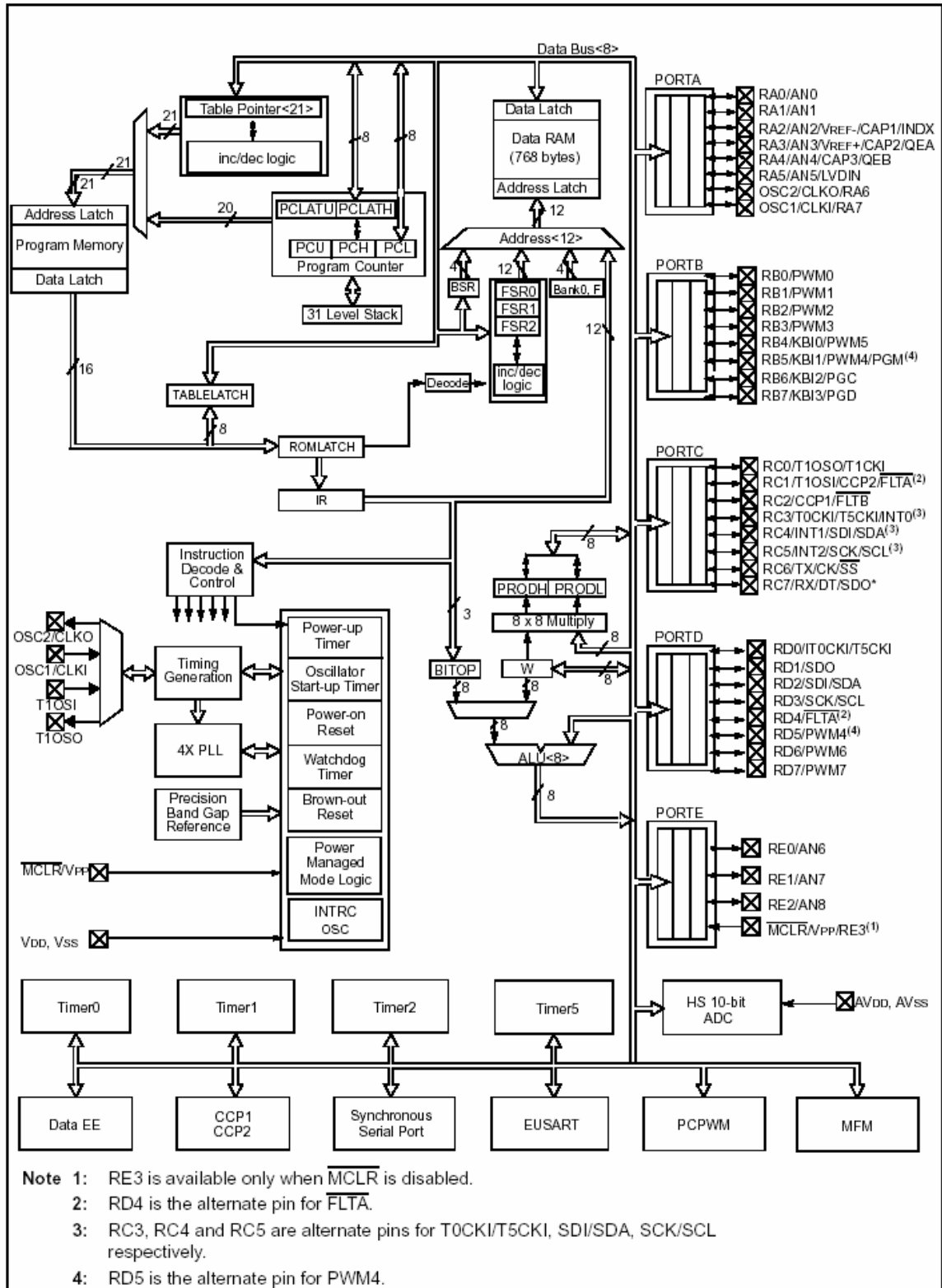
**3:** RD4 is the alternate pin for FLTA.

**4:** RD5 is the alternate pin for PWM4.

#### 3.2.2> Sơ đồ các khối chức năng :

# CHƯƠNG 3: GIỚI THIỆU VỀ PIC® Microcontrollers (MCUs)

FIGURE 1-2: PIC18F4331/4431 BLOCK DIAGRAM





### 2.2.3) Chức năng của từng chân:

#### a)\_PORT A:

- + Là port I/O . Có tất cả 6 chân, từ RA0 đến RA5. Trong đó RA2 và RA3 có thể dùng tiếp nhận điện áp Vref+ và Vref-.
- + RA4 còn là ngõ vào xung clock cho Timer0. RA5 có thể làm chân chọn slave cho port serial đồng bộ.

#### b)\_PORT B:

- + Là port I/O ,có thể được lập trình bởi phần mềm để làm chức năng kéo lên cho tất cả ngõ vào.
- + RB0 có thể làm chân ngắt ngoài.
- + RB3 có thể làm ngõ vào lập trình điện thế thấp.
- + Các chân còn lại có thể làm ngõ vào ngắt trên chân,lập trình với xung và dữ liệu serial.

#### c)\_PORT C:

- + Là port I/O, có 8 chân:
- + RC0 dùng làm ngõ ra bộ dao động Timer1 hoặc ngõ vào xung timer1.
- + RC1 ,RC2 có cùng 3 chức năng: làm ngõ ra PWM / chân Compare( so sánh) / chân capture (lấy mẫu).RC1 còn là ngõ vào bộ dao động Timer1.
- + RC3 là ngõ vào xung tuần tự đồng bộ/ hoặc ra (với chế độ SPI và I2C).
- + RC4 làm chân nhận data (chế độ SPI) hay data I/O (chế độ I2C).
- + RC5 có thể xuất data SPI ( chế độ SPI).
- + RC6 có thể làm chân phát bất đồng bộ (USART) hoặc xung đồng bộ.

#### d)\_PORT D:

- + Là port I/O ,có thể làm port slave song song khi giao tiếp với 1 bus vi xử lý.

#### e)\_PORT E:

- \_Port I/O này thường dùng điều khiển chọn/đọc/ghi cho port slave song song.

#### f)\_ Các chân khác:

- + Chân 13(OSC1/CLKIN) tiếp nhận xung ngoài cho bộ dao động thạch anh bên trong.
- + Chân 14(OSC2/CLKOUT) làm ngõ ra bộ dao động thạch anh.Ở chế độ RC,chân này có tần số bằng  $\frac{1}{4}$  của OSC1.
- + Chân 1 : làm ngõ vào reset .
- + Chân 12, 31 là nối đất Vss.Chân 11, 32 là chân cấp nguồn Vdd.







### 3.3> CÁC MODULE CƠ BẢN:

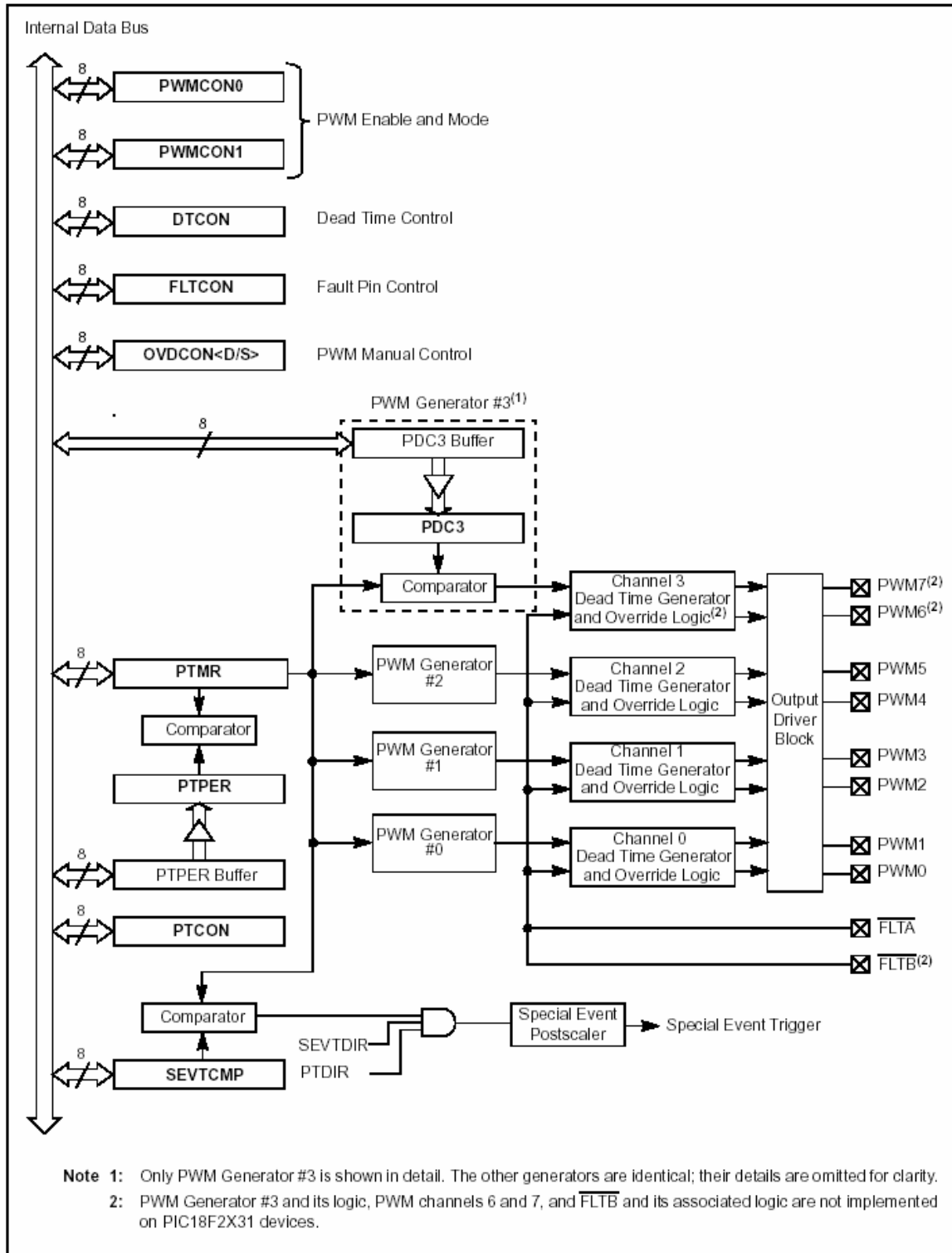
#### 3.3.1> Power control PWM module :

Power Control PWM module đơn giản là tạo ra nhiều xung đồng bộ có độ rộng thay đổi được ( PWM : Pulse Width Modulation ). Các ngõ ra PWM ứng dụng trong điều khiển động cơ và các ứng dụng chuyển đổi công suất . Module PWM này hỗ trợ điều khiển các ứng dụng sau :

- + Động cơ KĐB 1 pha và 3 pha
  - + Switched Reluctance Motor
  - + Động cơ DC không chổi than
  - + UPS ( Uninterruptible Power Suppliers)
  - + Multiple DC Brush motor
- 
- **Các thông số cơ bản của module PWM:**
    - + Có 8 ngõ I/O PWM với 4 duty cycle khác nhau
    - + Độ phân giải 14 bit dựa trên PWM periode
    - + Thời gian dead time có thể lập trình ( ứng dụng trong trường PWM đối nghịch => chống trùng dẫn )
    - + Ngắt hỗ trợ update không đối( asymmetrical update ) xứng trong chế độ canh giữa ( center aligned mode)

## ▪ Sơ đồ khối của module PWM

**FIGURE 17-1: POWER CONTROL PWM MODULE BLOCK DIAGRAM**



## CHƯƠNG 3: GIỚI THIỆU VỀ PIC® Microcontrollers (MCUs)

FIGURE 17-2: PWM MODULE BLOCK DIAGRAM, ONE OUTPUT PAIR, COMPLEMENTARY MODE

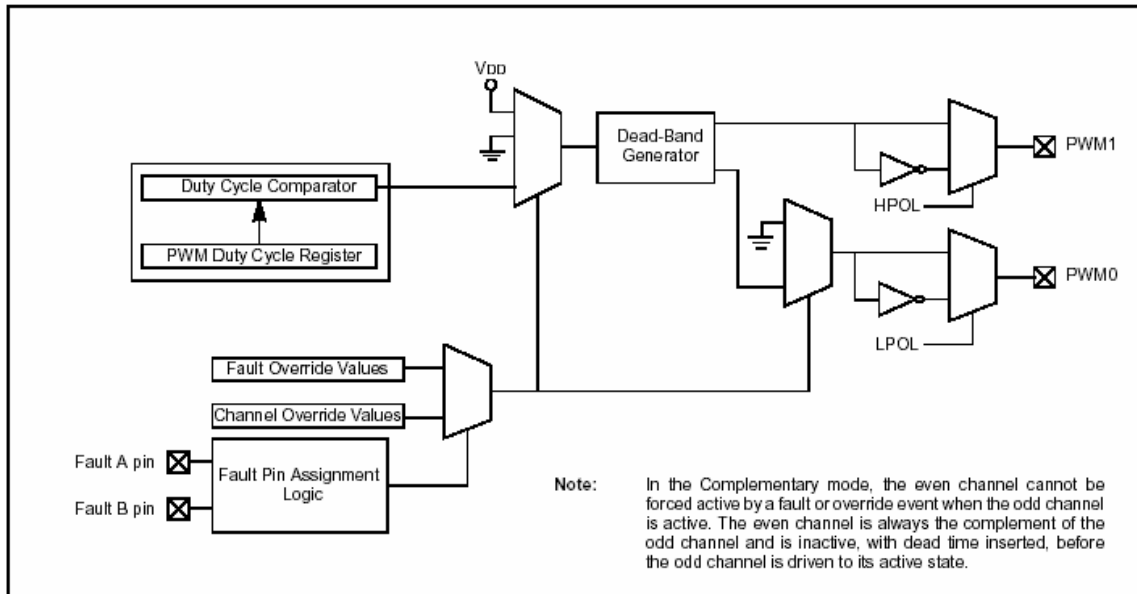
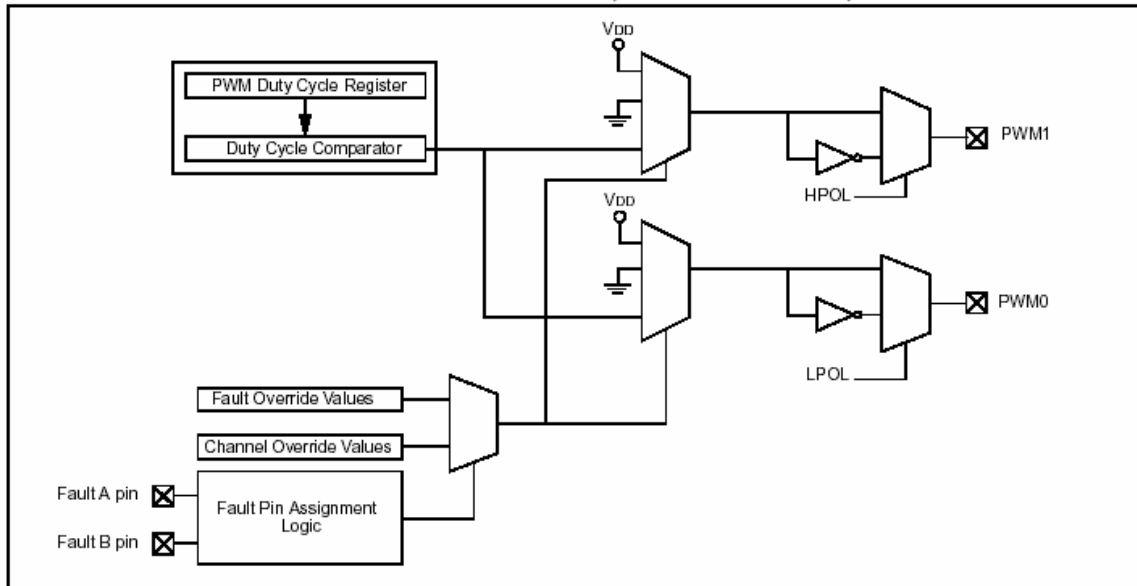


FIGURE 17-3: PWM MODULE BLOCK DIAGRAM, ONE OUTPUT PAIR, INDEPENDENT MODE



Trong module PWM có 4 bộ tạo duty cycle riêng biệt, chúng được đánh số từ 0 -> 3. Module này có 8 ngõ ra, được đánh số từ 0->7. Trong chế độ đối nghịch các pin chẵn – pin lẻ là 1 cặp. VD: PWM0 sẽ đối nghịch với PWM1; PWM2 sẽ đối nghịch với PWM3; ....

---

## CHƯƠNG 3: GIỚI THIỆU VỀ PIC<sup>®</sup> Microcontrollers (MCUs)

---

Bộ tạo dead time sẽ chèn 1 khoản “ off” giữa lúc xung PWM của pin này đang cạnh xuống và xung PWM của chân đối nghịch đang ở cạnh lên ( trong 1 cặp chân đối nghịch). Điều này ngăn chặn trùng dẫn => các khóa công suất được bảo vệ

### 3.3.1a) Các thanh ghi điều khiển:

Hoạt động của module PWM được điều khiển thông qua 22 thanh ghi khác nhau. 8 trong số đó được dùng để điều chỉnh các thông số của module:

- + PWM timer control register 0 ( **PTCON0**)
- + PWM timer control register 1 ( **PTCON1**)
- + PWM control register 0 ( **PWCON0**)
- + PWM control register 1 ( **PWCON1**)
- + Dead time control register (**DTCON**)
- + Output override register(**OVDCOND**)
- + Output state register (**OVDCONS**)
- + Fault configuration register (**FLTCONFIG**)

7 cặp ( 14 thanh ghi) còn lại : hiệu chỉnh thông số đặc biệt:

- + PWM time base registers (**PTMRH** and **PTMRL**)
- + PWM periode registers (**PTPERH** and **PTPERL**)
- + PWM special event compare register ( **SEVTCMPH** and **SEVTCMPL**)
- + PWM duty cycle #0 register ( **PDC0H** and **PDC0L**)
- + PWM duty cycle #1 register ( **PDC1H** and **PDC1L**)
- + PWM duty cycle #2 register ( **PDC2H** and **PDC2L**)
- + PWM duty cycle #3 register ( **PDC3H** and **PDC3L**)

Những cặp thanh ghi trên đều double buffers

### 3.3.1b) Các module chức năng:

PWM module hỗ trợ nhiều chế độ hoạt động phù hợp cho yêu cầu điều khiển động cơ. PWM module được tổng hợp từ các khối chức năng sau:

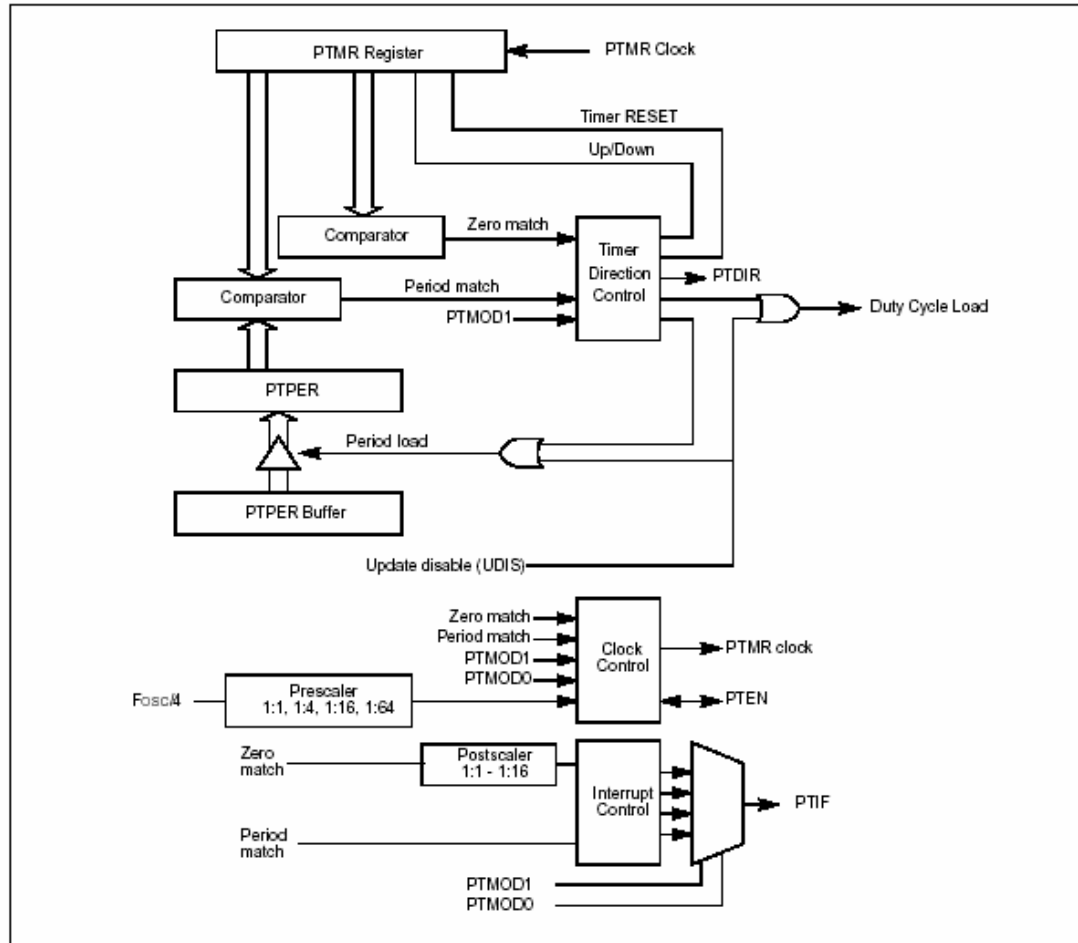
- + PWM Time Base
- + PWM Time Base Interrupts
- + PWM Period
- + PWM Duty Cycle
- + Dead Time Generators
- + PWM Output Overrides
- + PWM Fault Inputs
- + PWM Special Event Trigger



### 3.3.1c) PWM Time Base:

PWM time base được cung cấp 12 bit timer với chức năng prescaler and postcaler. Sơ đồ khối đơn giản của PWM time base được trình bày trong hình 17-4. PWM time base được hiệu chỉnh thông qua 2 thanh ghi PTCON0 và PTCON1. Time base được enabled hay disabled bởi set hay clear bit PTEN trong thanh ghi PTCON1. Chú ý, cặp thanh ghi PTMR (PTMRH:PTMRL) sẽ không bị clear khi bit PTEN bị clear trong phần mềm !!!

FIGURE 17-4: PWM TIME BASE BLOCK DIAGRAM



PWM time base có 4 chế độ hoạt động như sau

- + Free running mode => edge aligned PWM
- + Single shot mode => center aligned PWM
- + Continous Up/Down count mode => support electronically commtated motors
- + Continous Up/Down count mode with interrupts for double updates

4 chế độ trên được lựa chọn thông qua bit PTMOD1:PTMOD0 trong thanh ghi PTCON0.

## CHƯƠNG 3: GIỚI THIỆU VỀ PIC® Microcontrollers (MCUs)

**REGISTER 17-1: PTCON0: PWM TIMER CONTROL REGISTER 0**

	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	PTOPS3	PTOPS2	PTOPS1	PTOPS0	PTCKPS1	PTCKPS0	PTMOD1	PTMOD0
bit 7								bit 0

- bit 7-4 **PTOPS3:PTOPS0:** PWM Time Base Output Postscale Select bits  
 0000 =1:1 Postscale  
 0001 =1:2 Postscale  
 .  
 .  
 1111 =1:16 Postscale
- bit 3-2 **PTCKPS1:PTCKPS0:** PWM Time Base Input Clock Prescale Select bits  
 00 =PWM time base input clock is Fosc/4 (1:1 prescale)  
 01 =PWM time base input clock is Fosc/16 (1:4 prescale)  
 10 =PWM time base input clock is Fosc/64 (1:16 prescale)  
 11 =PWM time base input clock is Fosc/256 (1:64 prescale)
- bit 1-0 **PTMOD1:PTMOD0:** PWM Time Base Mode Select bits  
 11 =PWM time base operates in a Continuous Up/Down mode with interrupts for double PWM updates.  
 10 =PWM time base operates in a Continuous Up/Down Counting mode.  
 01 =PWM time base configured for Single-shot mode.  
 00 =PWM time base operates in a Free Running mode.

<b>Legend:</b>			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = bit is set	'0' = bit is cleared	x = bit is unknown

**REGISTER 17-2: PTCON1: PWM TIMER CONTROL REGISTER 1**

	R/W-0	R-0	U-0	U-0	U-0	U-0	U-0	U-0
	PTEN	PTDIR	—	—	—	—	—	—
bit 7								bit 0

- bit 7 **PTEN:** PWM Time Base Timer Enable bit  
 1 = PWM time base is ON  
 0 = PWM time base is OFF
- bit 6 **PTDIR:** PWM Time Base Count Direction Status bit  
 1 = PWM time base counts down.  
 0 = PWM time base counts up.
- bit 5-0 **Unimplemented:** Read as '0'.

<b>Legend:</b>			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = bit is set	'0' = bit is cleared	x = bit is unknown

## CHƯƠNG 3: GIỚI THIỆU VỀ PIC® Microcontrollers (MCUs)

**REGISTER 17-3: PWMCON0: PWM CONTROL REGISTER 0**

	U-0	R/W-1 <sup>(1)</sup>	R/W-1 <sup>(1)</sup>	R/W-1 <sup>(1)</sup>	R/W-0	R/W-0	R/W-0	R/W-0
bit 7	—	PWMEN2	PWMEN1	PWMEN0	PMOD3 <sup>(3)</sup>	PMOD2	PMOD1	PMOD0
								bit 0

- bit 7 **Unimplemented:** Read as '0'.
- bit 6-4 **PWMEN2:PWMEN0:** PWM Module Enable bits<sup>(1)</sup>  
 111 =All odd PWM I/O pins enabled for PWM output<sup>(2)</sup>.  
 110 =PWM1, PWM3 pins enabled for PWM output.  
 101 =All PWM I/O pins enabled for PWM output<sup>(2)</sup>.  
 100 =PWM0, PWM1, PWM2, PWM3, PWM4 and PWM5 pins enabled for PWM output.  
 011 =PWM0, PWM1, PWM2 and PWM3 I/O pins enabled for PWM output.  
 010 =PWM0 and PWM1 pins enabled for PWM output.  
 001 =PWM1 pin is enabled for PWM output.  
 000 =PWM module disabled. All PWM I/O pins are general purpose I/O.
- bit 3-0 **PMOD3:PMOD0:** PWM Output Pair Mode bits  
For PMOD0:  
 1 = PWM I/O pin pair (PWM0, PWM1) is in the Independent mode.  
 0 = PWM I/O pin pair (PWM0, PWM1) is in the Complementary mode.  
For PMOD1:  
 1 = PWM I/O pin pair (PWM2, PWM3) is in the Independent mode.  
 0 = PWM I/O pin pair (PWM2, PWM3) is in the Complementary mode.  
For PMOD2:  
 1 = PWM I/O pin pair (PWM4, PWM5) is in the Independent mode.  
 0 = PWM I/O pin pair (PWM4, PWM5) is in the Complementary mode.  
For PMOD3<sup>(3)</sup>:  
 1 = PWM I/O pin pair (PWM6, PWM7) is in the Independent mode.  
 0 = PWM I/O pin pair (PWM6, PWM7) is in the Complementary mode.

- Note 1:** Reset condition of PWMEN bits depends on PWMPIN device configuration bit.
- 2:** When PWMEN2:PWMEN0 = 101, PWM[5:0] outputs are enabled for PIC18F2X31 devices; PWM[7:0] outputs are enabled for PIC18F4X31 devices. When PWMEN2:PWMEN0 = 111, PWM outputs 1, 3 and 5 are enabled in PIC18F2X31 devices; PWM outputs 1, 3, 5 and 7 are enabled in PIC18F4X31 devices.
- 3:** Unimplemented in PIC18F2X31 devices; maintain these bits clear.

<b>Legend:</b>			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = bit is set	'0' = bit is cleared	x = bit is unknown

### 3.3.1d) PWM Time Base Interrupts:

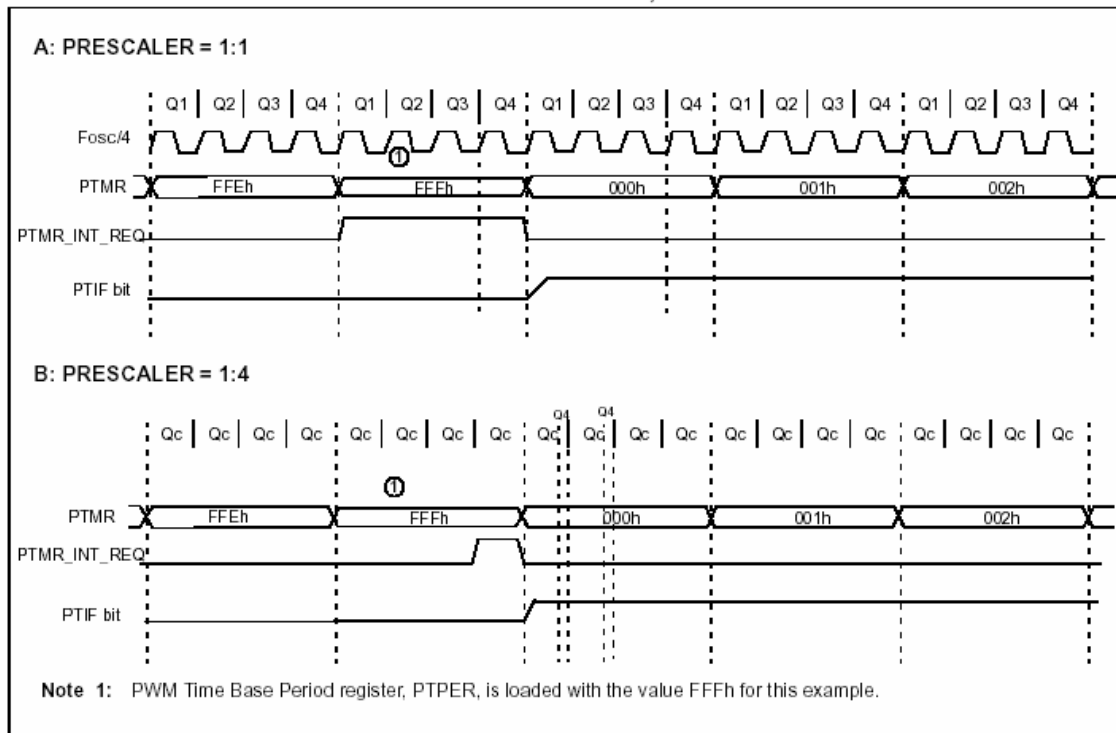
PWM timer tạo ra interrupts dựa trên chế độ hoạt động được lựa chọn bởi những bit PTMOD<1:0> và những bit postscaler<3:0>

#### ▪ Interrupts trong chế độ FREE RUNNING:

PWM time base ở chế độ time base ( PTMOD<1:0>=00 ), sự kiện interrupts xảy ra khi giá trị trong thanh ghi PTPER bằng giá trị của thanh ghi PTMR. Giá trị của thanh ghi PTMR sẽ được đưa về zero ngay xung clock sau đó.

Sử dụng postscaler lớn hơn 1:1 sẽ giảm tần số của các sự kiện interrupts .

FIGURE 17-5: PWM TIME BASE INTERRUPT TIMING, FREE RUNNING MODE

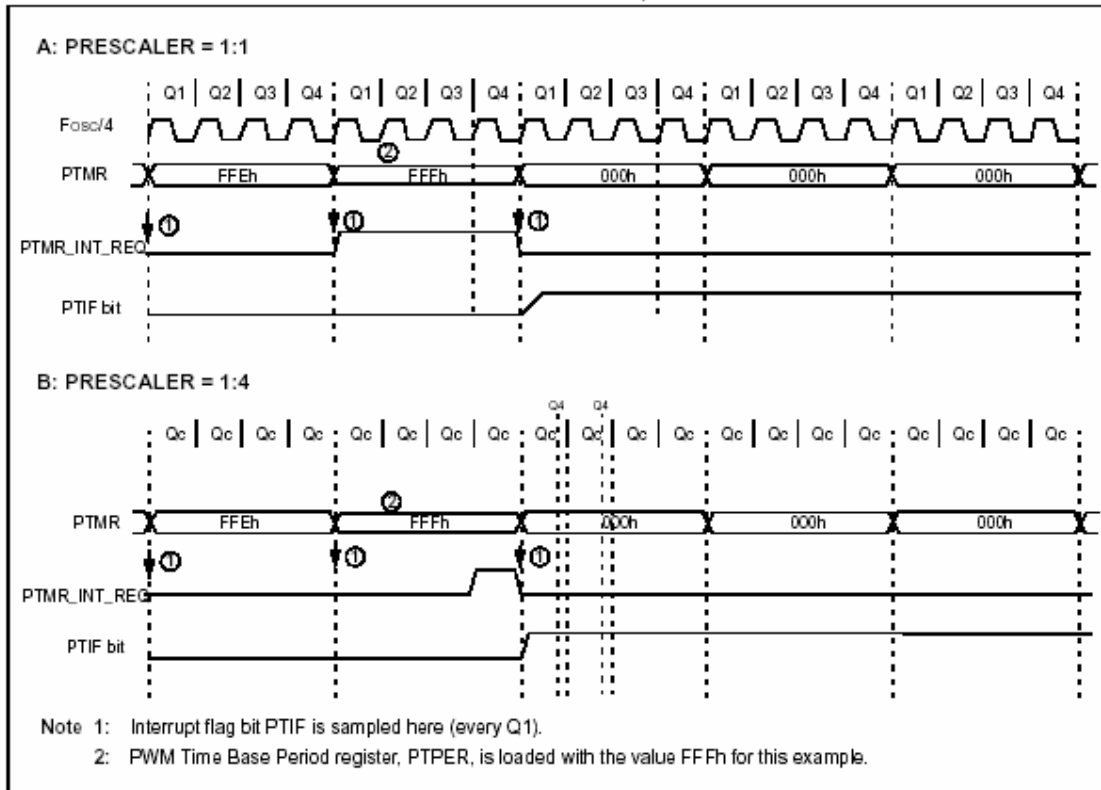


### ▪ Interrupts trong chế độ SINGLE SHOT:

Khi bit  $PTMOD<1:0>=01$  => PWM time base ở chế độ single shot. Sự kiện interrupts xảy ra khi giá trị trong thanh ghi PTPER bằng giá trị của thanh ghi PTMR. Giá trị của thanh ghi PTMR sẽ được đưa về zero ngay xung clock sau đó.

Những bit postscaler ko có tác dụng gì khi timer ở chế độ này.

FIGURE 17-6: PWM TIME BASE INTERRUPT TIMING, SINGLE SHOT MODE



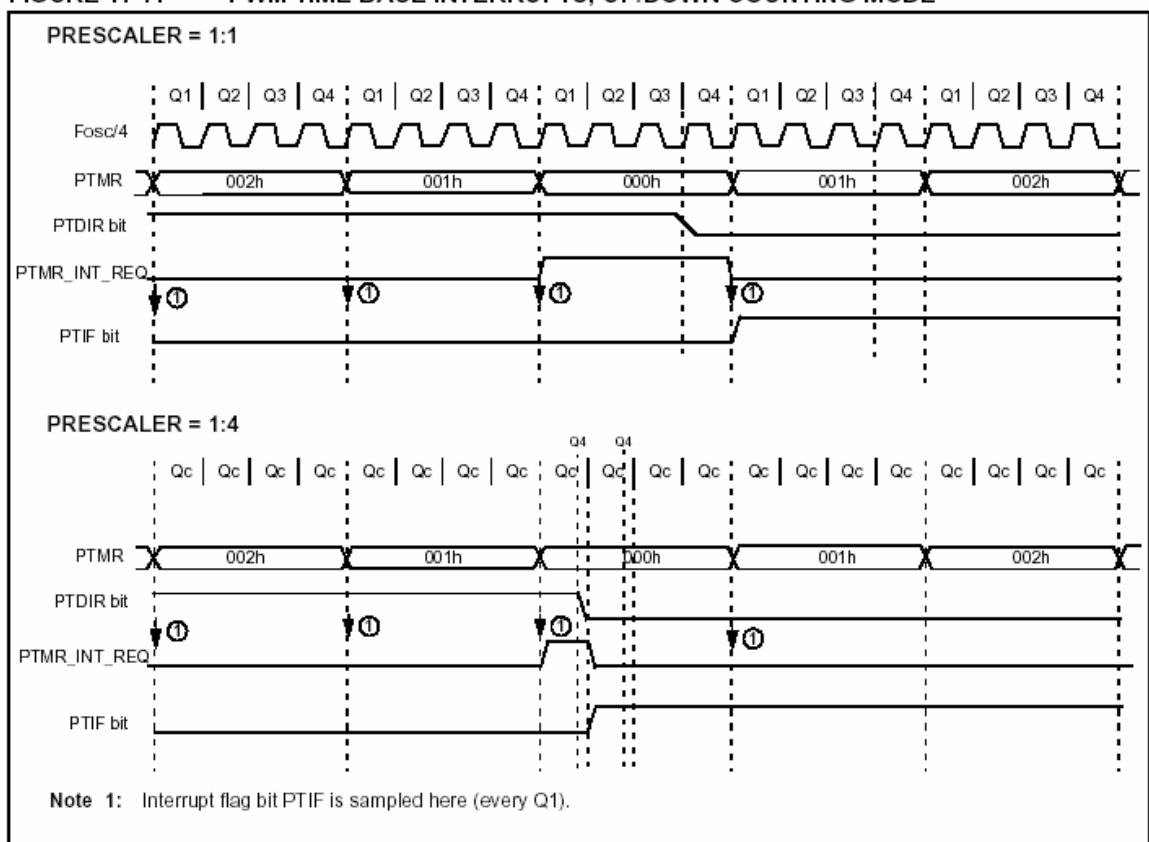
## CHƯƠNG 3: GIỚI THIỆU VỀ PIC® Microcontrollers (MCUs)

### ▪ Interrupts trong chế độ COUNTINOUS UP/DOWN COUTING:

Khi bit  $PTMOD<1:0>=10 \Rightarrow$  PWM time base ở chế độ countinuous up/down counting. Sự kiện interrupts xảy ra khi giá trị trong thanh ghi PTMR bằng zero, và PWM time base bắt đầu đếm lên.

Những bit lựa chọn postscaler có thể sử dụng trong chế độ này của timer để làm giảm tần số của sự kiện interrupts.

FIGURE 17-7: PWM TIME BASE INTERRUPTS, UP/DOWN COUNTING MODE



## CHƯƠNG 3: GIỚI THIỆU VỀ PIC<sup>®</sup> Microcontrollers (MCUs)

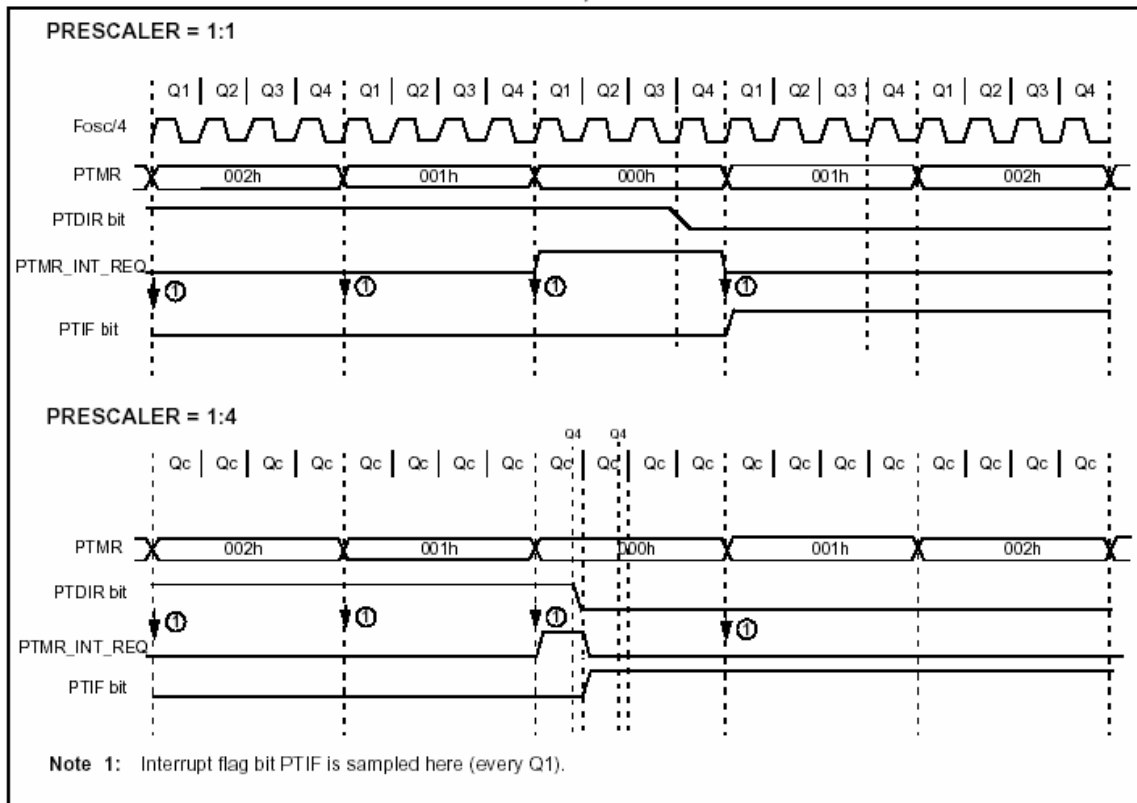
### ▪ Interrupts trong chế độ DOUBLE UPDATE:

Chế độ này chỉ có trong Up/Down Counting mode (  $PTMOD<1:0>=11$  ). Sự kiện interrupts xảy ra mỗi khi giá trị thanh ghi PTMR tương đương với zero hay khi giá trị thanh ghi PTMR trùng với giá trị thanh ghi PTPER.

Chế độ double update cung cấp cho người dùng thêm 2 chức năng trong chế độ center-align mode:

- + Bandwidth có độ lớn gấp đôi vì PWM duty cycle được update 2 lần trong mỗi chu kỳ (periode)
- + Có thể tạo ra được dạng sóng PWM center-align không đối xứng, điều này rất hữu dụng trong việc hạn chế tối đa sự méo dạng của dạng sóng ngõ ra trong 1 số ứng dụng điều khiển động cơ

FIGURE 17-7: PWM TIME BASE INTERRUPTS, UP/DOWN COUNTING MODE



### 3.3.1e) PWM Period :

PWM periode được định nghĩa bởi cặp thanh ghi PTPER ( PTPERH và PTPERL). PWM periode có độ phân giải 12 bit. PTPER là cặp thanh ghi double buffered sử dụng để set chế độ đếm của PWM time base.

Nội dung của PTPER buffer được nạp vào thanh ghi PTPER ở các thời điểm sau:

- + Free running mode và Single shot modes: thanh ghi PTMR được đưa về zero sau khi trùng giá trị với thanh ghi PTPER
- + Up/down counting mode: khi PTMR bằng zero. Giá trị được lưu trong PTPER buffer tự động nạp vào thanh ghi PTPER khi PWM time base được disabled ( PTEN=0)

FIGURE 17-9: PWM PERIOD BUFFER UPDATES IN FREE RUNNING COUNT MODE

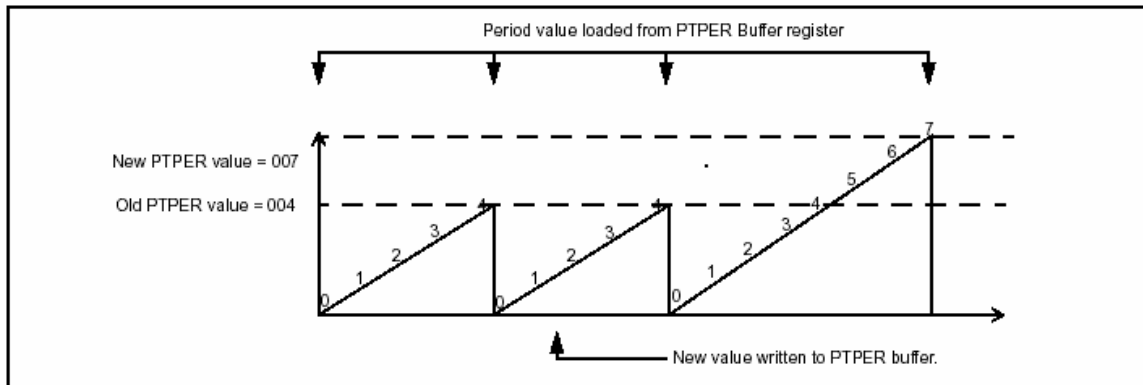
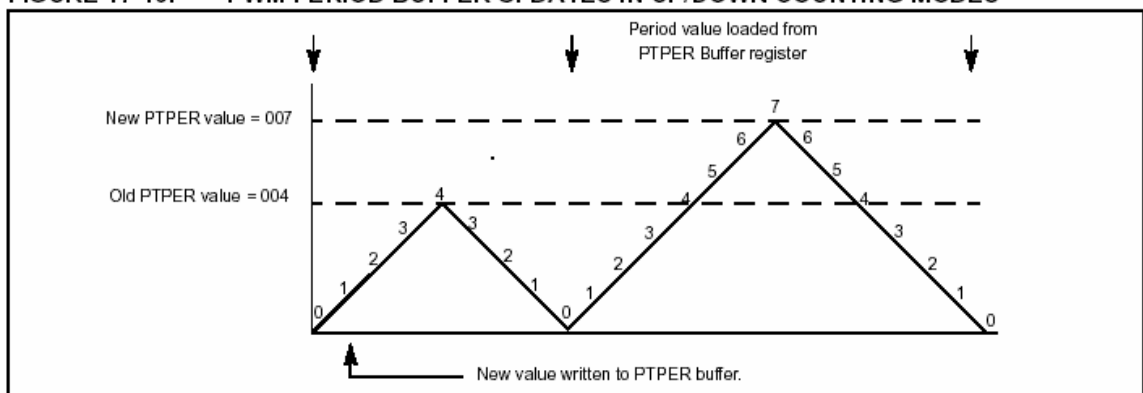


FIGURE 17-10: PWM PERIOD BUFFER UPDATES IN UP/DOWN COUNTING MODES





### 3.3.1f) PWM duty cycle:

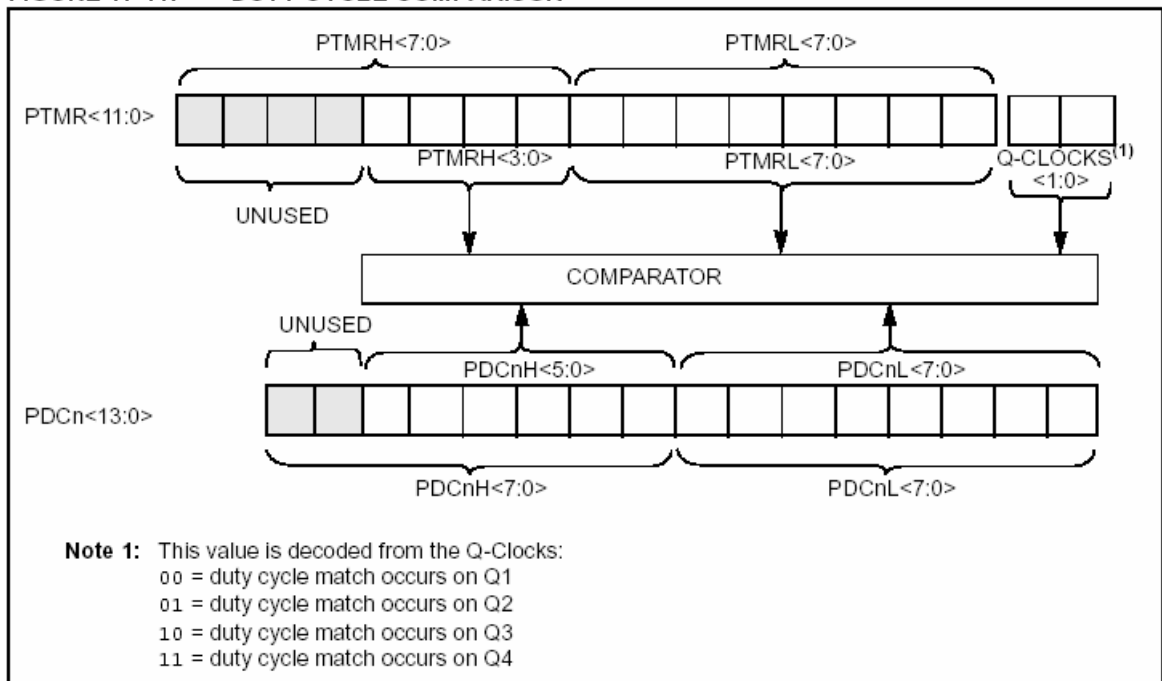
PWM duty cycle được xác định bởi các thanh ghi PDCx ( PDCxH và PDCxL). Có tổng cộng 4 cặp thanh ghi PWM duty cycle cho 4 cặp xung PWM.

- + PDC0 (PDC0L và PDC0H)
- + PDC1 (PDC1L và PDC1H)
- + PDC2 (PDC2L và PDC2H)
- + PDC3 (PDC3L và PDC3H)

Giá trị trong mỗi thanh ghi xác định khoản thời gian mà ngõ ra PWM đó tích cực.

Trong chế độ Edge-aligned, PWM periode bắt đầu tại Q1 và kết thúc khi thanh ghi duty cycle trùng với giá trị PTMR.

**FIGURE 17-11: DUTY CYCLE COMPARISON**

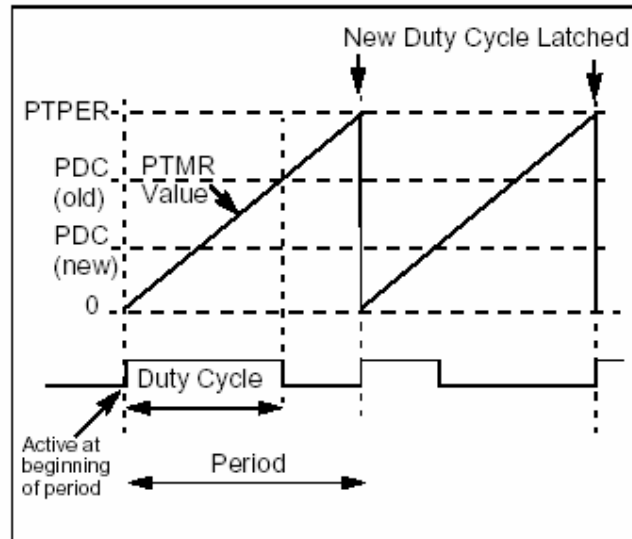


#### ▪ Duty cycle register buffer:

4 thanh ghi PWM duty cycle đều được double buffered. Mỗi duty cycle block, đều có thanh ghi duty cycle buffer mà có thể truy xuất bởi người dùng. Thanh ghi duty cycle buffer thứ hai sẽ giữ giá trị so sánh với PWM periode hiện tại.

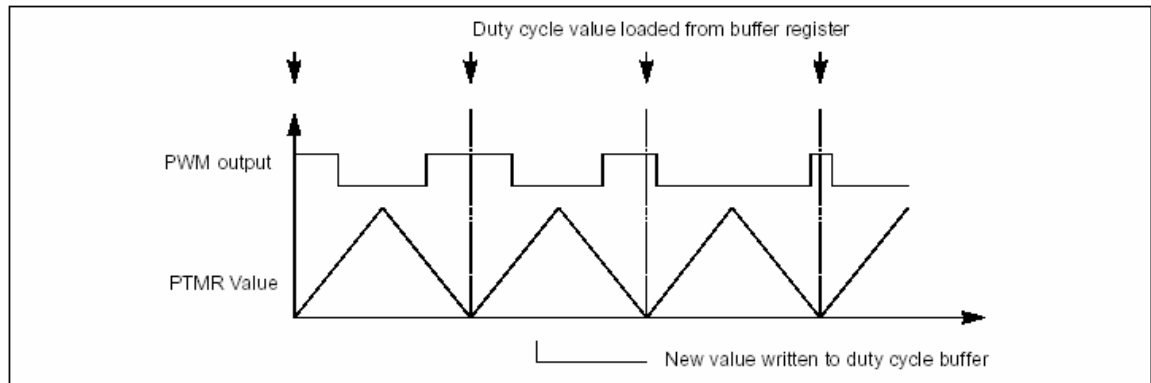
Trong chế độ edge-aligned PWM output, giá trị duty cycle mới sẽ được update mỗi khi giá trị thanh ghi PTMR và PTPER trùng nhau. Sau đó PTMR sẽ được reset như trong hình 17-12. Nội dung của duty cycle buffer sẽ tự động cập nhật vào thanh ghi duty cycle khi PWM time base bị disable ( PTEN=0)

FIGURE 17-12: EDGE-ALIGNED PWM



Khi PWM time base ở chế độ Up/Down counting, giá trị duty cycle mới sẽ được update khi giá trị thanh ghi PTMR bằng zero và PWM time base bắt đầu đếm lên. Nội dung của duty cycle buffer sẽ tự động cập nhật vào thanh ghi duty cycle khi PWM time base bị disable (PTEN=0). Hình 17-13 trình bày giản đồ thời gian khi duty cycle được update ở chế độ Up/Down counting. Trong chế độ này PWM periode phải được sẵn sàng để nạp và tính toán trước PWM duty cycle mới trước khi các thay đổi có hiệu lực.

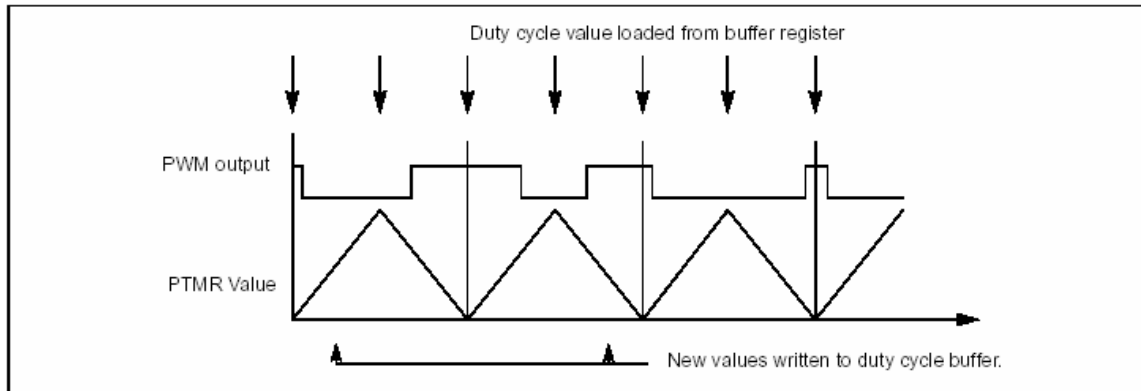
FIGURE 17-13: DUTY CYCLE UPDATE TIMES IN UP/DOWN COUNTING MODE



Khi PWM time base ở chế độ Up/Down counting với double update mode, giá trị duty cycle mới sẽ được update khi giá trị thanh ghi PTMR bằng zero và khi giá trị hai thanh ghi PTMR và PTPER trùng nhau. Nội dung của duty cycle buffer sẽ tự động được nạp vào thanh ghi duty cycle khi một trong hai điều kiện trên xảy ra.

## CHƯƠNG 3: GIỚI THIỆU VỀ PIC<sup>®</sup> Microcontrollers (MCUs)

FIGURE 17-14: DUTY CYCLE UPDATE TIMES IN UP/DOWN COUNTING MODE WITH DOUBLE UPDATES

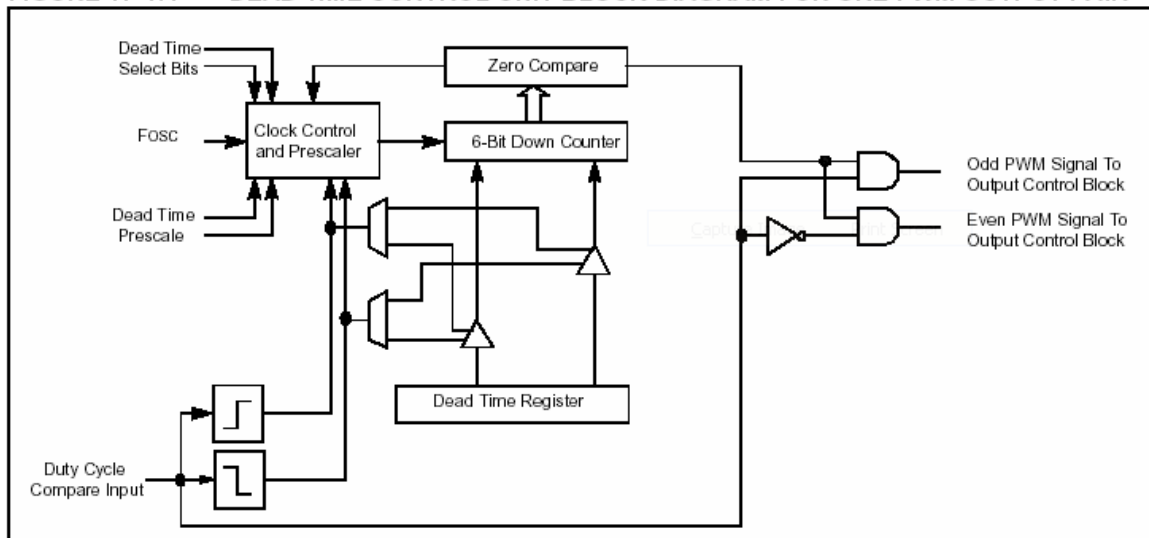


### 3.3.1g) Bộ tạo thời gian dead time:

Trong bộ biến tần, khi các xung PWM ở chế độ đối nghịch để điều khiển các khóa công suất phía cao; phía thấp trong cùng 1 nhánh, phải chèn 1 khoản thời gian dead time. Khoản thời gian dead time đó làm cho ngõ ra PWM đối nghịch đều ở trạng thái không tác động trong 1 khoản thời gian ngắn=> tránh trùng dẫn khi khóa này đang ON, khóa kia đang OFF

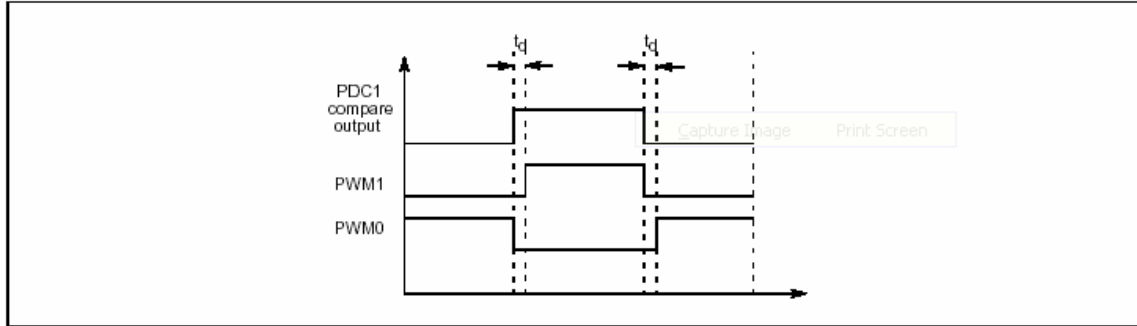
Mỗi cặp xung PWM đối nghịch đều có một counter 6 bit đếm xuống, để chèn khoản dead time vào xung PWM. Mỗi bộ tạo dead time có bộ phát hiện cạnh lên và cạnh xuống được kết nối với bộ so sánh duty cycle. Dead time được nạp vào timer khi phát hiện PWM ở cạnh lên hay cạnh xuống. Tùy vào xung PWM đang ở cạnh lên hay cạnh xuống, mà 1 khoản thời gian chuyển tiếp được làm trễ cho đến khi timer đếm về zero.

FIGURE 17-17: DEAD TIME CONTROL UNIT BLOCK DIAGRAM FOR ONE PWM OUTPUT PAIR



## CHƯƠNG 3: GIỚI THIỆU VỀ PIC® Microcontrollers (MCUs)

**FIGURE 17-18: DEAD TIME INSERTION FOR COMPLEMENTARY PWM**



### ▪ Thanh ghi DTCON:

**REGISTER 17-5: DTCON – DEAD TIME CONTROL REGISTER**

	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DTPS1	DTPS0	DT5	DT4	DT3	DT2	DT1	DT0
bit 7								bit 0

bit 7-6 **DTPS1:DTPS0:** Dead Time Unit A Prescale Select bits

11 = Clock source for Dead Time Unit is  $F_{osc}/16$ .

10 = Clock source for Dead Time Unit is  $F_{osc}/8$ .

01 = Clock source for Dead Time Unit is  $F_{osc}/4$ .

00 = Clock source for Dead Time Unit is  $F_{osc}/2$ .

bit 5-0 **DT5:DT0:** Unsigned 6-bit dead time value bits for Dead Time Unit.

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = bit is set

'0' = bit is cleared

x = bit is unknown

## CHƯƠNG 3: GIỚI THIỆU VỀ PIC® Microcontrollers (MCUs)

Bảng tóm tắt các thanh ghi có liên quan của POWER CONTROL PWM MODULE :

**TABLE 17-6: REGISTERS ASSOCIATED WITH THE POWER CONTROL PWM MODULE**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBFIF	0000 000x	0000 000u
IPR3	—	—	—	PTIP	IC3DRIP	IC2QEIP	IC1IP	TMR5IP	---1 1111	---1 1111
PIE3	—	—	—	PTIE	IC3DRIE	IC2QEIE	IC1IE	TMR5IE	---0 0000	---0 0000
PIR3	—	—	—	PTIF	IC3DRIF	IC2QEIF	IC1IF	TMR5IF	---0 0000	---0 0000
PTCON0	PTOPS3	PTOPS2	PTOPS1	PTOPS0	PTCKPS1	PTCKPS0	PTMOD1	PTMOD0	0000 0000	0000 0000
PTCON1	PTEN	PTDIR	—	—	—	—	—	—	00-- ----	00-- ----
PTMRL <sup>(1)</sup>	PWM Time Base (lower 8 bits)								0000 0000	0000 0000
PTMRH <sup>(1)</sup>	—	—	—	—	PWM Time Base (upper 4 bits)				---- 0000	---- 0000
PTPERL <sup>(1)</sup>	PWM Time Base Period (lower 8 bits)								1111 1111	1111 1111
PTPERH <sup>(1)</sup>	—	—	—	—	PWM Time Base Period (upper 4 bits)				---- 1111	---- 1111
SEVTCMPL <sup>(1)</sup>	PWM Special Event Compare (lower 8 bits)								0000 0000	0000 0000
SEVTCMPH <sup>(1)</sup>	—	—	—	—	PWM Special Event Compare (upper 4 bits)				---- 0000	---- 0000
PWMCON0	—	PWMEN2	PWMEN1	PWMEN0	PMOD3 <sup>(2)</sup>	PMOD2	PMOD1	PMOD0	-101 0000	-101 0000
PWMCON1	SEVOPS3	SEVOPS2	SEVOPS1	SEVOPS0	SEVTDIR	—	UDIS	OSYNC	0000 0-00	0000 0-00
DTCON	DTPS1	DTPS0	Dead Time A Value register						0000 0000	0000 0000
FLTCONFIG	BRFEN	FLTBS <sup>(2)</sup>	FLTBMOD <sup>(2)</sup>	FLT BEN <sup>(2)</sup>	FLTCON	FLTAS	FLTAMOD	FLTAEN	0000 0000	0000 0000
OVDCONCOND	POVD7 <sup>(2)</sup>	POVD6 <sup>(2)</sup>	POVD5	POVD4	POVD3	POVD2	POVD1	POVD0	1111 1111	1111 1111
OVDCONCONS	POUT7 <sup>(2)</sup>	POUT6 <sup>(2)</sup>	POUT5	POUT4	POUT3	POUT2	POUT1	POUT0	0000 0000	0000 0000
PDC0L <sup>(1)</sup>	PWM Duty Cycle #0L register (lower 8 bits)								--00 0000	--00 0000
PDC0H <sup>(1)</sup>	—	—	PWM Duty Cycle #0H register (upper 6 bits)						0000 0000	0000 0000
PDC1L <sup>(1)</sup>	PWM Duty Cycle #1L register (lower 8 bits)								0000 0000	0000 0000
PDC1H <sup>(1)</sup>	—	—	PWM Duty Cycle #1H register (upper 6 bits)						--00 0000	--00 0000
PDC2L <sup>(1)</sup>	PWM Duty Cycle #2L register (Lower 8 bits)								0000 0000	0000 0000
PDC2H <sup>(1)</sup>	—	—	PWM Duty Cycle #2H register (Upper 6 bits)						--00 0000	--00 0000
PDC3L <sup>(1,2)</sup>	PWM Duty Cycle #3L register (Lower 8 bits)								0000 0000	0000 0000
PDC3H <sup>(1,2)</sup>	—	—	PWM Duty Cycle #3H register (Upper 6 bits)						--00 0000	--00 0000

Legend: — = Unimplemented, u = Unchanged. Shaded cells are not used with the power control PWM.

Note 1: Double-buffered register pairs. Refer to text for explanation of how these registers are read and written to.

2: Unimplemented in PIC18F2X31 devices; maintain these bits clear. Reset values shown are for PIC18F4X31 devices.

### 3.3.2> Analog to digital converter module (A/D):

Bộ A/D có 5 ngõ vào cho PIC 28 chân và 8 cho các PIC khác . Tín hiệu analog được lấy mẫu và giữ bởi tụ điện , sau đó đưa vào bộ chuyển đổi . Bộ này tạo ra 1 kết quả số tương ứng . Giá trị này là 1 số 10 bit.

Bộ A /D có ngõ vào so sánh áp cao và thấp ,và có thể lựa chọn thông qua kết hợp Vdd , Vss , RA2 hay RA3. Bộ A/D có điểm đặc biệt là có thể hoạt động trong khi vi điều khiển ở trạng thái SLEEP . Để làm được điều này , xung clock A/D phải được nhận từ bộ dao động RC nội của bộ A/D.

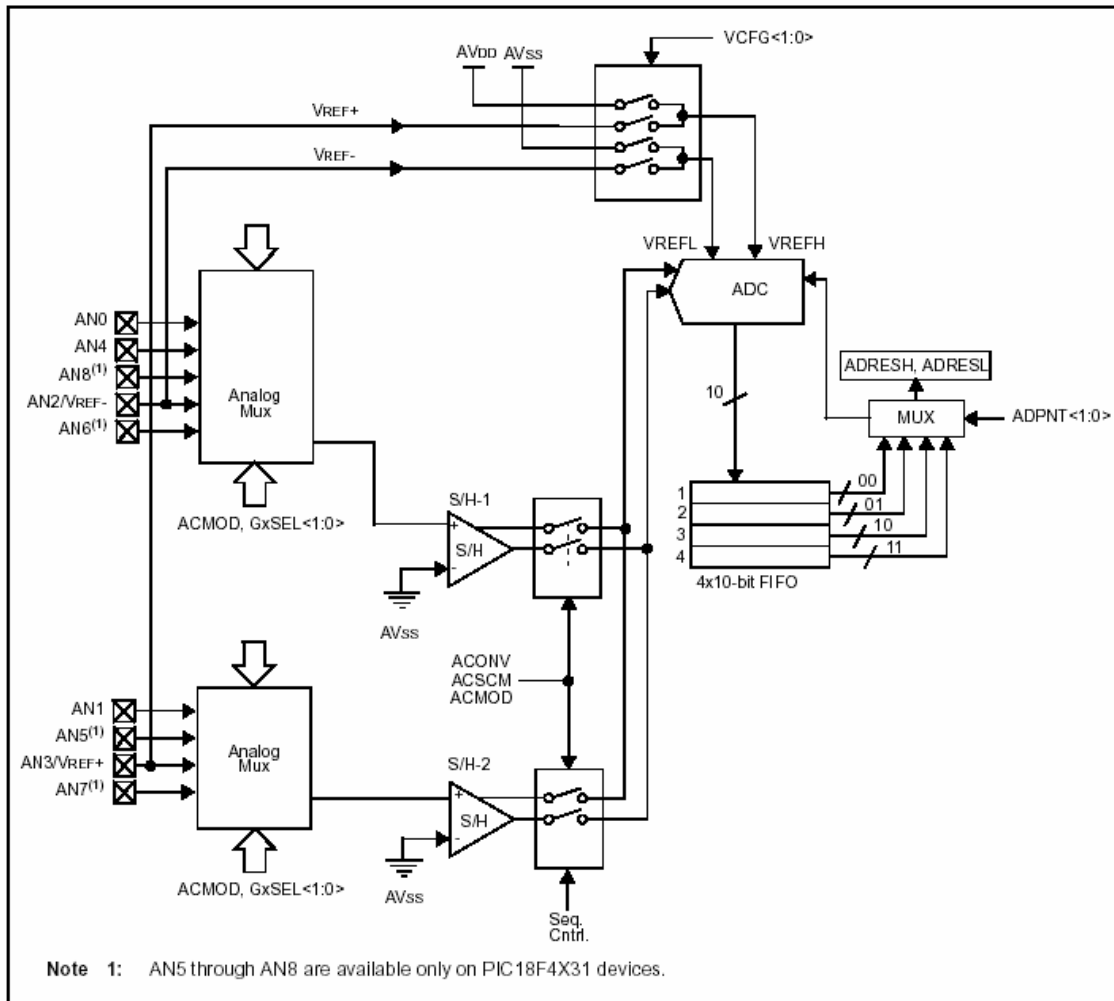
## CHƯƠNG 3: GIỚI THIỆU VỀ PIC® Microcontrollers (MCUs)

Module A/D có 9 thanh ghi :

- + A/D Result High Register (ADRESH)
- + A/D Result Low Register (ADRESL)
- + A/D Control Register0 (ADCON0)
- + A/D Control Register1 (ADCON1)
- + A/D Control Register2 (ADCON2)
- + A/D Control Register3 (ADCON3)
- + A/D channel Select Register (ADCHS)
- + Analog I/O Select Register 0 ( ANSEL0)
- + Analog I/O Select Register 1 ( ANSEL1)

Sơ đồ khối bộ A/D :

FIGURE 20-1: A/D BLOCK DIAGRAM

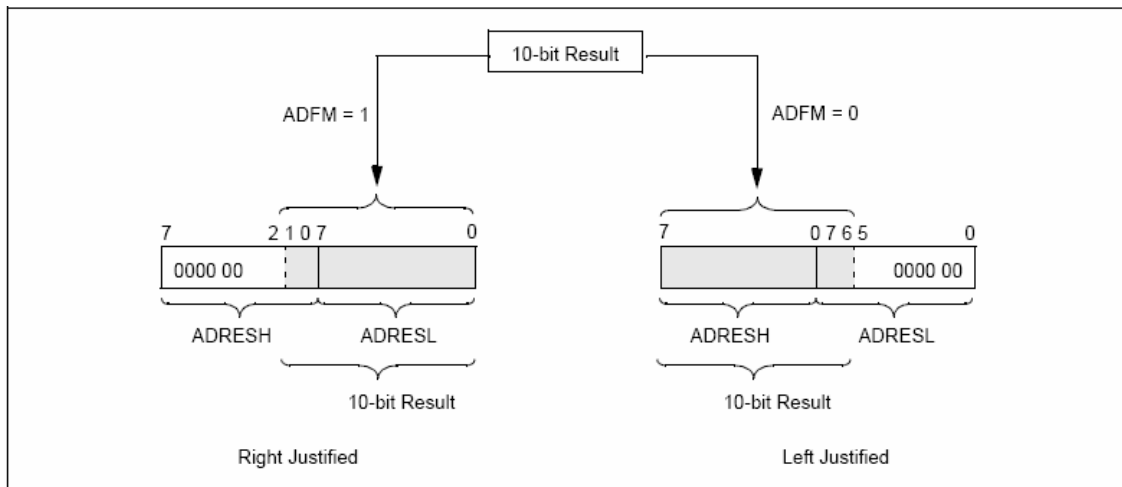


## CHƯƠNG 3: GIỚI THIỆU VỀ PIC<sup>®</sup> Microcontrollers (MCUs)

Các bước sau để làm việc với bộ A/D :

- 1\_Thiết lập bộ A/D :
  - + Thiết lập các chân analog / so sánh áp và I/O số ( ADCON1 ) .
  - + Chọn kênh ngõ vào A/D (ADCON0).
  - + Chọn xung clock bộ A/D ( ADCON0).
  - + Kích hoạt A/D ( ADCON0 ) .
- 2\_Thiết lập ngắt A/D nếu sử dụng
  - + xoá bit ADIF.
  - + Set bit ADIE.
  - + set bit PEIE
  - + set bit GIE
- 3\_Chờ thời gian đáp ứng cần thiết.
- 4\_Bắt đầu chuyển đổi : set bit ADCON0<2>.
- 5\_Chờ chuyển đổi A/D hoàn thành bằng cách hỏi vòng bit ADCON0<2> có bị xoá chưa hay chờ ngắt A/D
- 6\_Đọc kết quả từ cặp thanh ghi ADRESH : ADRESL , xoá bit ADIF nếu cần .
- 7\_Lặp lại từ bước 1 hay 2 nếu có yêu cầu. Thời gian chuyển đổi A/D mỗi bit gọi là  $T_{AD}$  .

Một khoảng chờ tối thiểu  $2T_{AD}$  được yêu cầu trước khi lần đáp ứng kế tiếp bắt đầu.



Các thanh ghi ADRESH : ADRESL chứa 10 bit kết quả của chuyển đổi A/D . Khi sự chuyển đổi A/D hoàn tất , kết quả đưa vào cặp thanh ghi này , bit ADCON0 <2> bị xoá và cờ ngắt ADIF được set. Cặp thanh ghi này rộng 16 bit . Do đó nếu bit ADFM =1 :lấy 10 bit bên phải và ADFM = 0 thì lấy 10 bit bên trái , các bit còn lại bằng 0. Nếu A/D bị vô hiệu , các thanh ghi này có thể dùng như 2 thanh ghi đa mục đích

**CHƯƠNG 4 :  
THIẾT KẾ PHẦN CỨNG****4.1> YÊU CẦU CƠ BẢN :**

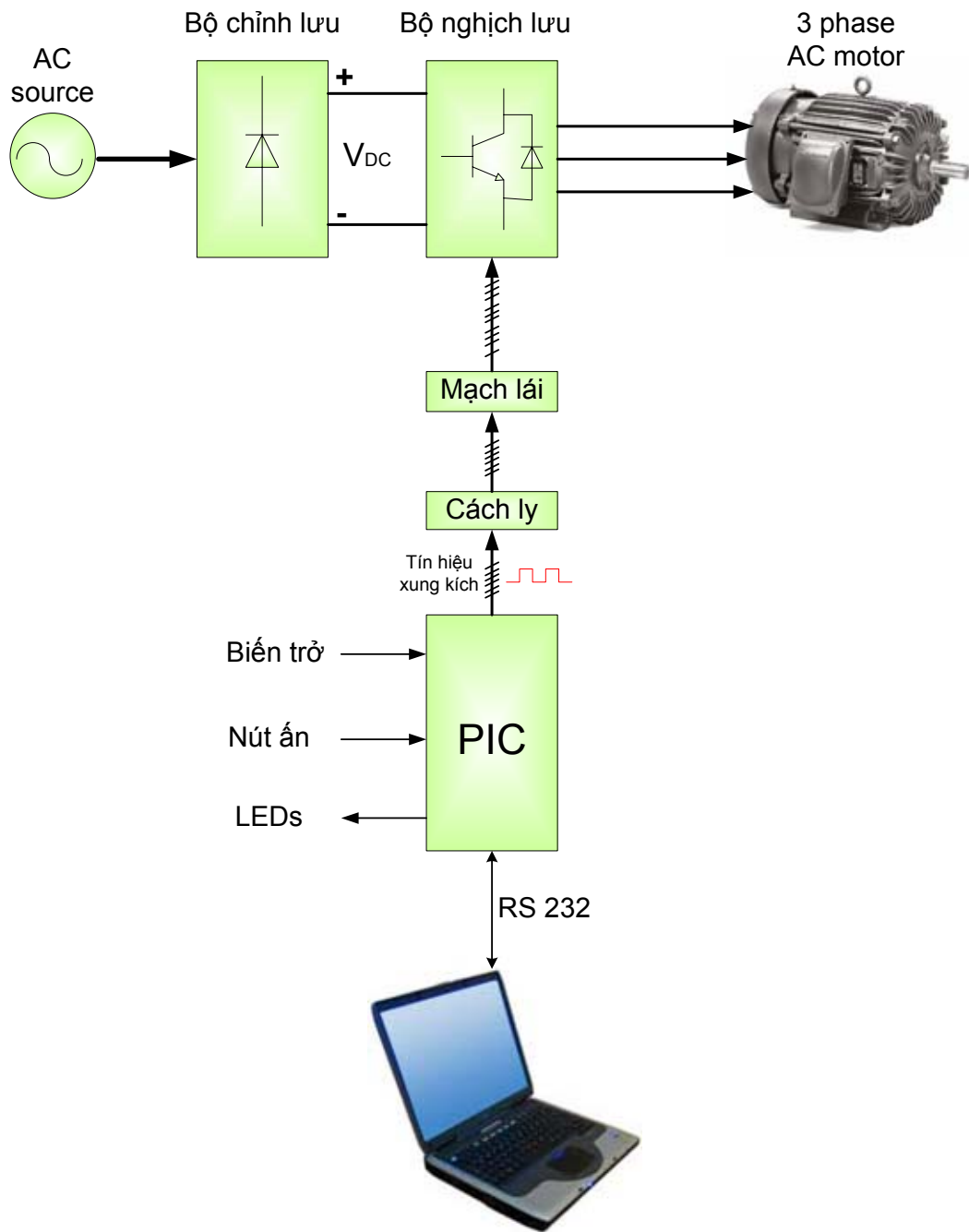
*“Thiết kế bộ biến tần truyền thống ( 6 khóa) ba pha điều khiển động cơ KĐB 1.5 kW “*

Thông số tiêu biểu của động cơ 1.5 kW ( 2 HP) ở tần số 50 Hz như sau :

	Các thông số	Đơn vị	Động cơ đấu $\Delta$ / sao
<b>P<sub>đm</sub></b>	Công suất định mức	(KW)	1.5
<b>V<sub>đm</sub></b>	Điện áp định mức	(Vac)	380/220
<b>I<sub>đm</sub></b>	Dòng điện định mức	(A)	5.9/3.4
<i>cos<math>\varphi</math></i>	Hệ số công suất		0.81
<b>RPM</b>	Vận tốc	( vòng /phút)	1420



4.2> SƠ ĐỒ KHỐI CỦA HỆ THỐNG :



Hình 4.1: Sơ đồ khối của hệ thống

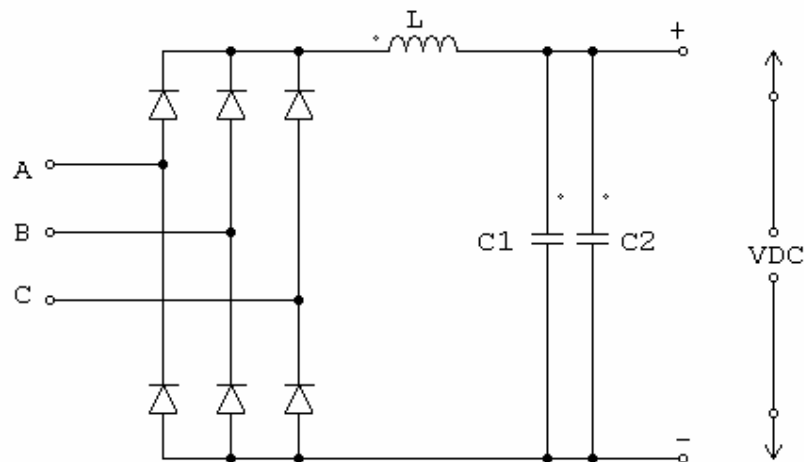
**4.3> MẠCH ĐỘNG LỰC :**

**4.3.1) Bộ chỉnh lưu:**

Yêu cầu:

- Điện áp  $V_{DC}$  đầu ra của bộ chỉnh lưu:
  - + Trong phương pháp SVPWM thì :  $|\vec{V}_A| = |\vec{V}_B| = |\vec{V}_C| = \frac{V_{DC}}{\sqrt{3}}$
  - + Để động cơ vận hành ở chế độ định mức (  $Y$  ) => trị biên độ  $V_{pha} = (380 * \sqrt{2}) / \sqrt{3}$
  - $\Rightarrow V_{DC} \approx \sqrt{3} * |\vec{V}_{pha}| \approx 540(V)$
  - + Để động cơ vận hành ở chế độ định mức (  $\Delta$  ) trị biên độ  $V_{pha} = (220 * \sqrt{2}) / \sqrt{3}$
  - $\Rightarrow V_{DC} \approx \sqrt{3} * |\vec{V}_{pha}| \approx 311(V)$
- Trị tức thời của VDC được nắn tương đối phẳng
- Gọn nhẹ , giá thành rẻ

=> Ta sử dụng phương pháp chỉnh lưu cầu với 6 diode ( có thể chỉnh lưu 1 pha , hay 3 pha )



Trị trung bình điện áp đầu ra khi chỉnh lưu cầu 3 pha (không điều khiển):

$$V_{DC} = \frac{3\sqrt{6} * V_{pha}}{\pi} \cos \alpha \approx \mathbf{515 (V)} \approx V_{DC} \text{ yêu cầu ( ĐC chế độ đấu sao)}$$

+  $V_{pha}$  : trị hiệu dụng áp pha nguồn (220 VAC)

+  $\alpha = 0$  : bộ chỉnh lưu không điều khiển

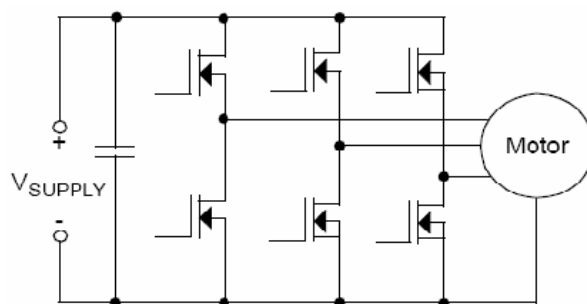
### ✓ Ghi chú:

Trong điều kiện thực tế, nếu chỉ có nguồn 1 pha để thực hiện chỉnh lưu thì điện áp VDC sau chỉnh lưu :

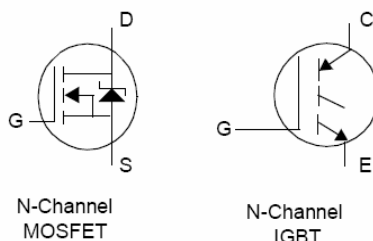
$$V_{DC} = \frac{2\sqrt{2} * V_{pha}}{\pi} \cos \alpha \approx 200(V) \Rightarrow \text{Động cơ sẽ không thể vận hành hết định}$$

mức cả hai chế độ

### 4.3.2) Bộ nghịch lưu:



Có hai lựa chọn chính cho việc sử dụng khoá đóng cắt công suất trong điều khiển động cơ đó là MOSFET và IGBT. Cả hai loại MOSFET và IGBT đều là linh kiện được điều khiển bằng điện áp, nghĩa là việc dẫn và ngưng dẫn của linh kiện được điều khiển bằng một nguồn điện áp nối với cực gate của linh kiện thay vì là dòng điện trong các bộ nghịch lưu sử dụng transistor như trước đây. Vì vậy cách sử dụng loại linh kiện này làm cho việc điều khiển trở nên dễ dàng hơn.



Thông thường MOSFET được sử dụng với các ứng dụng đòi hỏi tốc độ cao, tuy nhiên MOSFET không có khả năng chịu dòng điện cao. Trong khi đó IGBT thích hợp với các ứng dụng ở tốc độ thấp, tuy nhiên IGBT có khả năng chịu được dòng điện cao. Vì vậy tùy vào đặc điểm của ứng dụng mà có sự lựa chọn linh kiện phù hợp.

Các yêu cầu chính đặt ra cho linh kiện sử dụng làm bộ nghịch lưu :

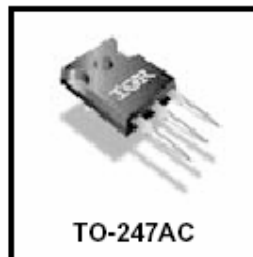
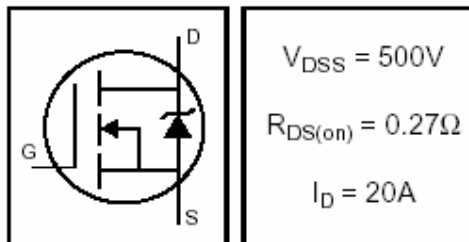
- Điện áp VDS ( Mosfet) hay VCE ( IGBT) > V<sub>DC</sub>
- Dòng điện qua linh kiện > dòng định mức của động cơ ≈ 10A ở nhiệt độ hoạt động
- Chịu được tần số đóng ngắt cao

▪ ...

=> **IRFP460P** được lựa chọn : thỏa mãn các yếu tố trên, có thể mua dễ dàng và giá thành rẻ !

### IRFP460P

HEXFET® Power MOSFET



#### 4.3.3) Mạch lái ( driver ) & cách ly:

##### a) Mạch lái :

Có hai phương án chính để lái MOSFET hay IGBT :

- + Biến áp xung
- + IC lái

Trong các phương án có biến áp xung, trường hợp xung điều khiển có cạnh tác động kéo dài hoặc tần số thấp, biến áp xung sớm đạt trạng thái bão hòa và ngõ ra của nó không phù hợp yêu cầu điều khiển. Do đó ta nên sử dụng loại high voltage bootstrap driver ICs.

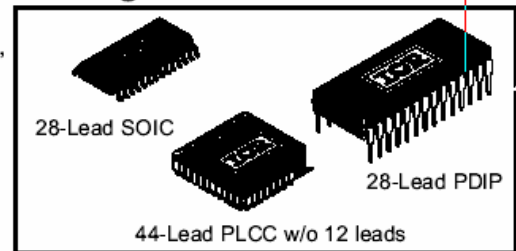
Trong đó : **IR2136** là loại IC chuyên dụng để lái MOSFET và IGBT của hãng IR - International Rectifier. IC này có 3 kênh output độc lập (mỗi kênh gồm high side and low side) dùng cho các ứng dụng 3 pha.

**Features**

- Floating channel designed for bootstrap operation Fully operational to +600V  
Tolerant to negative transient voltage - dV/dt immune
- Gate drive supply range from 10 to 20V (IR2136/IR21368), 11.5 to 20V (IR21362) or 12 to 20V (IR21363/IR21365/IR21366/IR21367)
- Undervoltage lockout for all channels
- Over-current shutdown turns off all six drivers
- Independent 3 half-bridge drivers
- Matched propagation delay for all channels
- Cross-conduction prevention logic
- Lowside outputs out of phase with inputs. High side outputs out of phase (IR2136/IR21363/IR21365/IR21366/IR21367/IR21368) or in phase (IR21362) with inputs.
- 3.3V logic compatible
- Lower di/dt gate driver for better noise immunity
- Externally programmable delay for automatic fault clear
- Also available LEAD-FREE

**3-PHASE BRIDGE DRIVER**

**Packages**

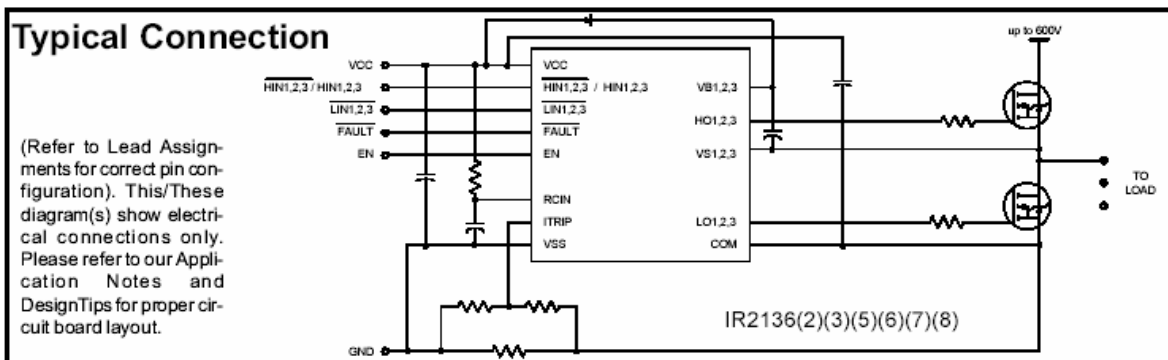


**Feature Comparison: IR2136/IR21362/IR21363/IR21365/IR21366/IR21367/IR21368**

Part	IR2136	IR21362	IR21363	IR21365	IR21366	IR21367	IR21368
Input Logic	HIN, LIN	HIN/LIN	HIN, LIN	HIN, LIN	HIN, LIN	HIN, LIN	HIN, LIN
Ton (typ.)	400ns	400ns	400ns	400ns	250ns	250ns	400ns
Toff (typ.)	380ns	380ns	380ns	380ns	180ns	180ns	380ns
V <sub>IH</sub> (typ.)	2.7V	2.7V	2.7V	2.7V	2.0V	2.0V	2.0V
V <sub>IL</sub> (typ.)	1.7V	1.7V	1.7V	1.7V	1.3V	1.3V	1.3V
V <sub>trip+</sub>	0.46V	0.46V	0.46V	4.3V	0.46V	4.3V	4.3V
UV CC/BS+	8.9V	10.4V	11.2V	11.2V	11.2V	11.2V	8.9V
UV CC/BS-	8.2V	9.4V	11.0V	11.0V	11.0V	11.0V	8.2V

**Description**

The IR2136/IR21362/IR21363/IR21365/IR21366/IR21367/IR21368(J&S) are high voltage, high speed power MOSFET and IGBT drivers with three independent high and low side referenced output channels for 3-phase applications. Proprietary HVIC technology enables ruggedized monolithic construction. Logic inputs are compatible with CMOS or LSTTL outputs, down to 3.3V logic. A current trip function which terminates all six outputs can be derived from an external current sense resistor. An enable function is available to terminate all six outputs simultaneously. An open-drain FAULT signal is provided to indicate that an overcurrent or undervoltage shutdown has occurred. Overcurrent fault conditions are cleared automatically after a delay programmed externally via an RC network connected to the RCIN input. The output drivers feature a high pulse current buffer stage designed for minimum driver cross-conduction. Propagation delays are matched to simplify use in high frequency applications. The floating channel can be used to drive N-channel power MOSFETs or IGBTs in the high side configuration which operates up to 600 volts.



### b) mạch cách ly:

Các mạch phát ra tín hiệu để điều khiển mạch công suất dùng bán dẫn phải được cách ly về điện. Điều này có thể thực hiện bằng opto hoặc bằng biến áp xung.

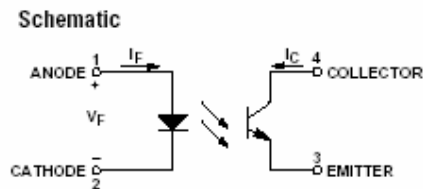
#### + Biến áp xung :

Gồm một cuộn dây sơ cấp và có thể nhiều cuộn thứ cấp. Với nhiều cuộn dây phía thứ cấp, ta có thể kích đóng nhiều transistor mắc nối tiếp hoặc song song.

Biến áp xung cần có cảm kháng tản nhỏ và đáp ứng nhanh. Trong trường hợp xung điều khiển có cạnh tác động kéo dài hoặc tần số thấp, biến áp xung sớm đạt trạng thái bão hòa và ngõ ra của nó không phù hợp yêu cầu điều khiển.

#### + Opto :

Gồm nguồn phát tia hồng ngoại dùng diode (IR - LED) và mạch thu dùng phototransistor. Do đó thỏa mãn yêu cầu cách ly về điện, đồng thời đáp ứng của opto tốt hơn máy biến áp xung.



=> ta lựa chọn phương án dùng OPTO. Yêu cầu đặt ra đối với opto là phải chịu được tần số đóng ngắt khá cao (>5KHz) mà điện áp xung ngõ ra ko bị méo dạng. Trong đó, **HCPL-2630** là optocouplers của hãng fairchild có tần số đóng ngắt lên thỏa mãn yêu cầu trên.



July 2005

**Single-channel: 6N137, HCPL-2601, HCPL-2611**  
**Dual-Channel: HCPL-2630, HCPL-2631**  
**High Speed-10 MBit/s Logic Gate Optocouplers**

**Features**

- Very high speed-10 MBit/s
- Superior CMR-10 kV/μs
- Double working voltage-480V
- Fan-out of 8 over -40°C to +85°C
- Logic gate output
- Strobable output
- Wired OR-open collector
- U.L. recognized (File # E90700)

**Applications**

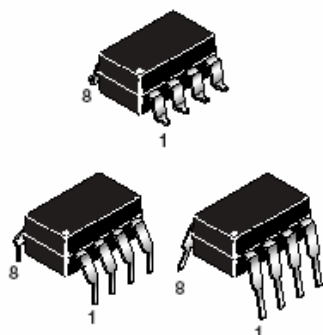
- Ground loop elimination
- LSTTL to TTL, LSTTL or 5-volt CMOS
- Line receiver, data transmission
- Data multiplexing
- Switching power supplies
- Pulse transformer replacement
- Computer-peripheral Interface

**Description**

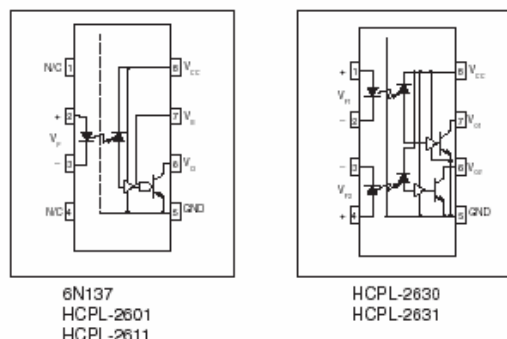
The 6N137, HCPL-2601/2611 single-channel and HCPL-2630/2631 dual-channel optocouplers consist of a 850 nm AlGaAs LED, optically coupled to a very high speed integrated photo-detector logic gate with a strobable output. This output features an open collector, thereby permitting wired OR outputs. The coupled parameters are guaranteed over the temperature range of -40°C to +85°C. A maximum input signal of 5 mA will provide a minimum output sink current of 13mA (fan out of 8).

An internal noise shield provides superior common mode rejection of typically 10kV/μs. The HCPL-2601 and HCPL-2631 has a minimum CMR of 5 kV/μs. The HCPL-2611 has a minimum CMR of 10 kV/μs.

**Package**



**Schematic**



**Truth Table (Positive Logic)**

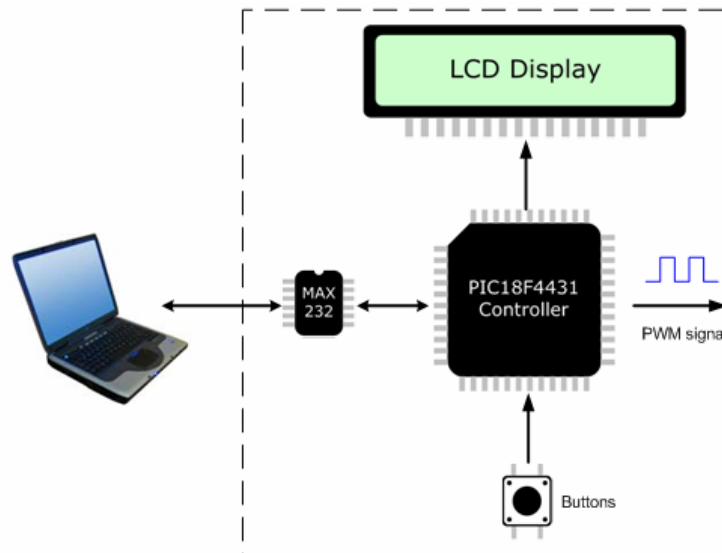
Input	Enable	Output
H	H	L
L	H	H
H	L	H
L	L	H
H	NC	L
L	NC	H

A 0.1μF bypass capacitor must be connected between pins 8 and 5. (See note 1)

Single-channel: 6N137, HCPL-2601, HCPL-2611 Dual-Channel: HCPL-2630, HCPL-2631 High Speed-10 MBit/s Logic Gate Optocouplers

### 4.2> MẠCH ĐIỀU KHIỂN:

#### 4.2.1) Sơ đồ khối mạch điều khiển:



#### 4.2.2) Các tín hiệu vào của mạch điều khiển:

- Nút ấn điều khiển động cơ:
  - + RUN
  - + STOP
  - + F/R
  - + Biến trở điều chỉnh tốc độ
- Nút ấn điều khiển LCD:
  - + MODE
  - + UP
  - + DOWN
  - + LEFT
  - + RIGHT
  - + SELECT
- Tín hiệu hồi tiếp: (\*)
  - + Dòng điện của động cơ
  - + Điện áp động cơ
  - + Tốc độ động cơ
  - + Nhiệt độ của khóa BJT
- Tín hiệu điều khiển từ PC

#### 4.2.3) Tín hiệu đầu ra của mạch điều khiển:

- + 6 xung PWM điều khiển bộ nghịch lưu
- + Hiện thị trạng thái hoạt động của mạch thông qua đèn LED
- + Hiện thị các thông số điều khiển bằng LCD
- + Xuất tín hiệu cho PC

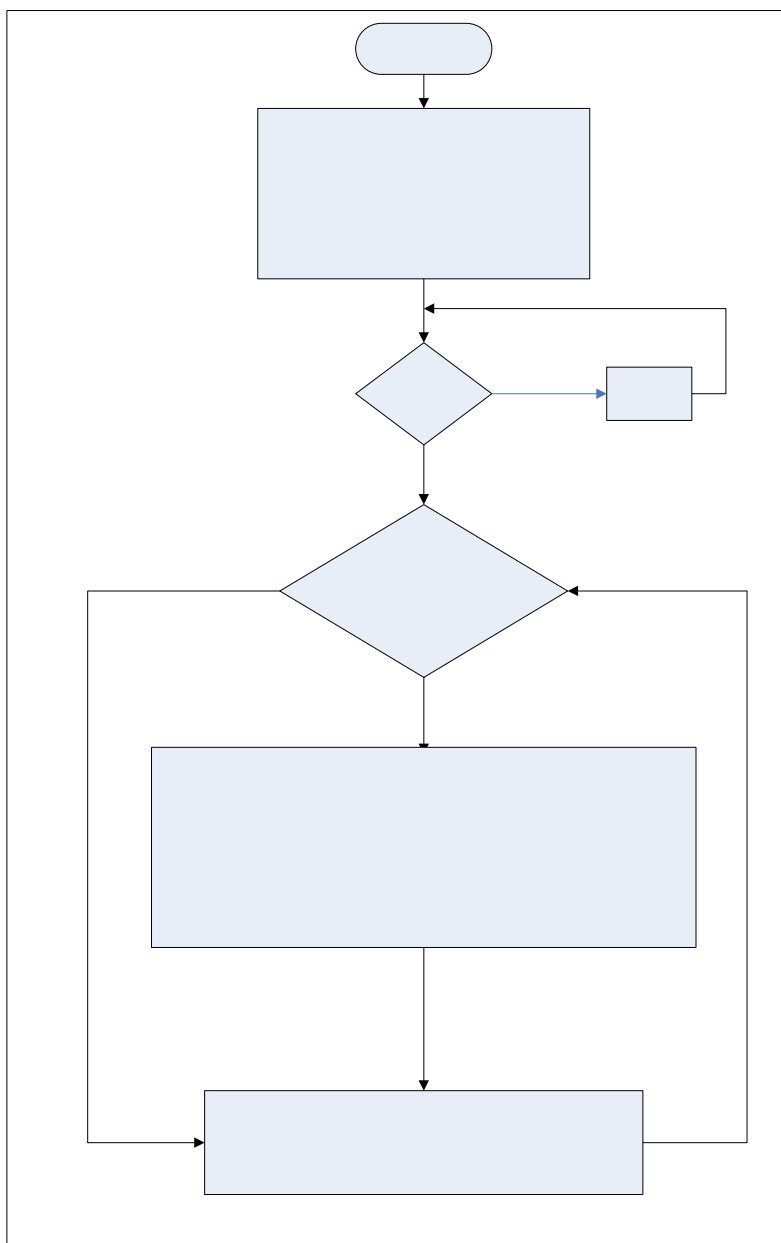
Ghi chú: (\*) => sẽ phát triển sau



CHƯƠNG 5:  
LẬP TRÌNH

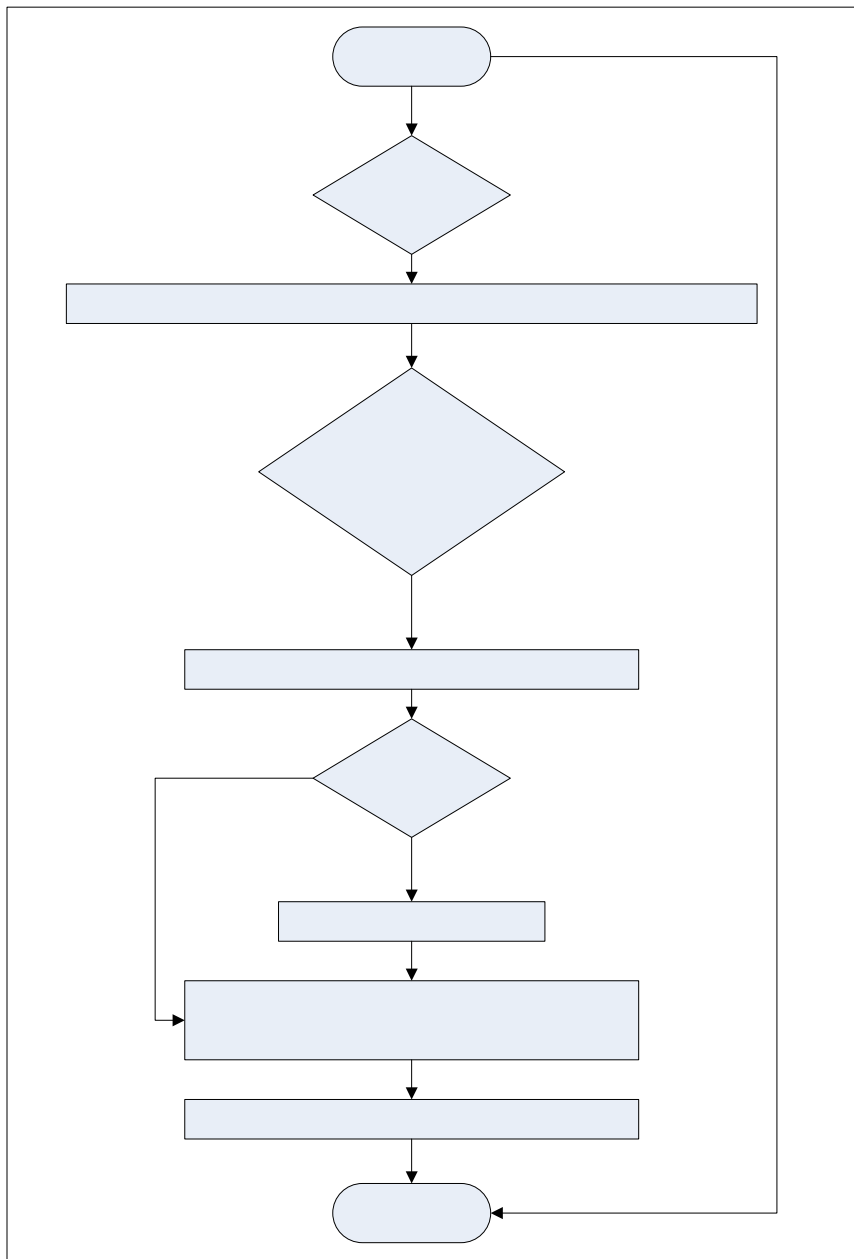
5.1> GIẢI THUẬT LẬP TRÌNH :

5.1.1) Chương trình chính:



MAI

5.1.2) Chương trình ngắt:



PWM int

Ghi chú: **PTIF**

+ interrupt flag bit

+ biến này được set lên 1 khi giá trị PTMR=0 và đếm lên ( trong chế độ center aligned )

### 5.2> GIẢI THÍCH GIẢI THUẬT :

#### 5.2.1) Chương trình chính:

- (1) Chương trình bắt đầu khi cấp nguồn cho PIC
- (2) Xác lập các thông số ban đầu :
  - + I/O pin
  - + A/D module
  - + Timer
  - + Power Contrl PWM module
  - + Interrupts event
- (3) Xử lý nút ấn RUN
- (4) Trạng thái IDLE: hiển thị LED báo trạng thái idle, đồng thời qua lại phần (3) kiểm tra xem nút RUN có được ấn hay không
- (5) Đọc giá trị f yêu cầu từ biến trở (mode 1) ; LCD (mode 2) hoặc PC (mode 3)
- (6) Khi tần số f yêu cầu thay đổi: tính toán các biến số **Vref**, **stepsize**. Hai thông số này dùng để update các giá trị về độ lớn và bước nhảy của vector Vs khi chương trình ngắt PWM xảy ra. Vref dùng để tính toán tỉ số điều biên  $m = Vref/Vdc$ . Stepsize xác định góc update của vector Vs
- (7) Kiểm tra xem button nào được ấn ( STOP , F/R.....) => xử lý button được ấn
  - + STOP button: => set các duty cycle về zero => qua lại vị trí (4) : IDLE
  - + F/R button: => gọi hàm RAM\_DOWN giảm tốc động cơ về zero => đảo chiều quay vector Vs => gọi hàm RAM\_UP tăng tốc động cơ đến tần số đặt
  - + .....

#### 5.2.2) Chương trình ngắt :

- (1) Khi cò ngắt được set lên 1, sao lưu trạng thái của vi điều khiển
- (2) Góc của vector Vs = giá trị góc ban đầu + góc update ( bước nhảy). Độ lớn của vector Vs được xác định trên tần số đặt (=> tỉ số điều biên m)
- (3) Có tổng cộng 6 sector. Mỗi sector 60 độ được chia thành 512 phần bằng nhau. Khi vector Vs quét hết sector hiện tại ( stepsize > 512), chuyển sang Vs sector mới => (4)
- (4) Vs chuyển sang sector mới và reset giá trị stepsize. ( stepsize = stepsize – 512 ). Hình ... trình bày cụ thể vấn đề này.

## CHƯƠNG 5: LẬP TRÌNH

---

- (5) Và (6) reset giá trị của sector khi Vs qua hết 1 vòng.
- (7) xác định thời gian TA, TB, T0/2 và (Ts - T0/2)
- (8) Nạp các giá trị trên vào thanh ghi PWM duty cycle

### CHƯƠNG 6: KẾT QUẢ ĐẠT ĐƯỢC

#### 6.1> PHẦN CỨNG:

##### 6.1.1> Mạch động lực:



Hình 6.1: Mạch động lực

#### + Ưu điểm:

Mạch động lực vận hành ổn định động cơ 2 HP ( đấu  $\Delta$  ; không tải ) ở tất cả các chế độ điều khiển thông thường( RUN, STOP, đảo chiều, thay đổi tốc độ.....).

#### + Nhược điểm:

- Nhiệt độ các khóa công suất khá cao ( 70-80 ° C)
- Chưa có khâu hồi tiếp dòng ,hồi tiếp tốc độ, hồi tiếp nhiệt độ khóa công suất

## CHƯƠNG 6: KẾT QUẢ ĐẠT ĐƯỢC

---

### + Giải pháp khắc phục:

- Nhiệt độ các khóa công suất khá cao => thay thế các khóa công suất bằng loại chất lượng tốt, đáp ứng tốt hơn .
- Phát triển thêm khâu hồi tiếp dòng => ngăn chặn quá dòng động cơ
- Phát triển khâu hồi tiếp tốc độ => điều khiển vòng kín động cơ
- Phát triển khâu hồi tiếp nhiệt độ của khóa công suất => ngăn chặn hiện tượng quá nhiệt

### 6.1.2> Mạch điều khiển:



Hình 6.2: Mạch điều khiển

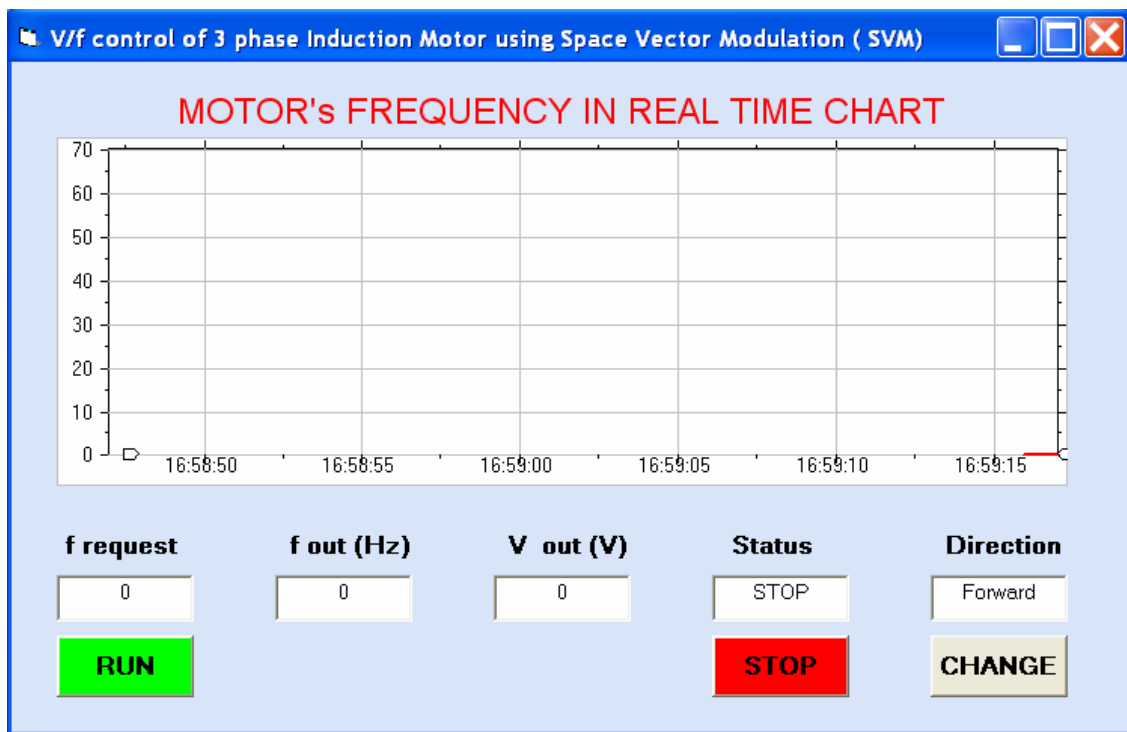
### Ưu điểm:

Mạch điều khiển có khả năng đáp ứng các yêu cầu điều khiển động cơ trong thực tế:

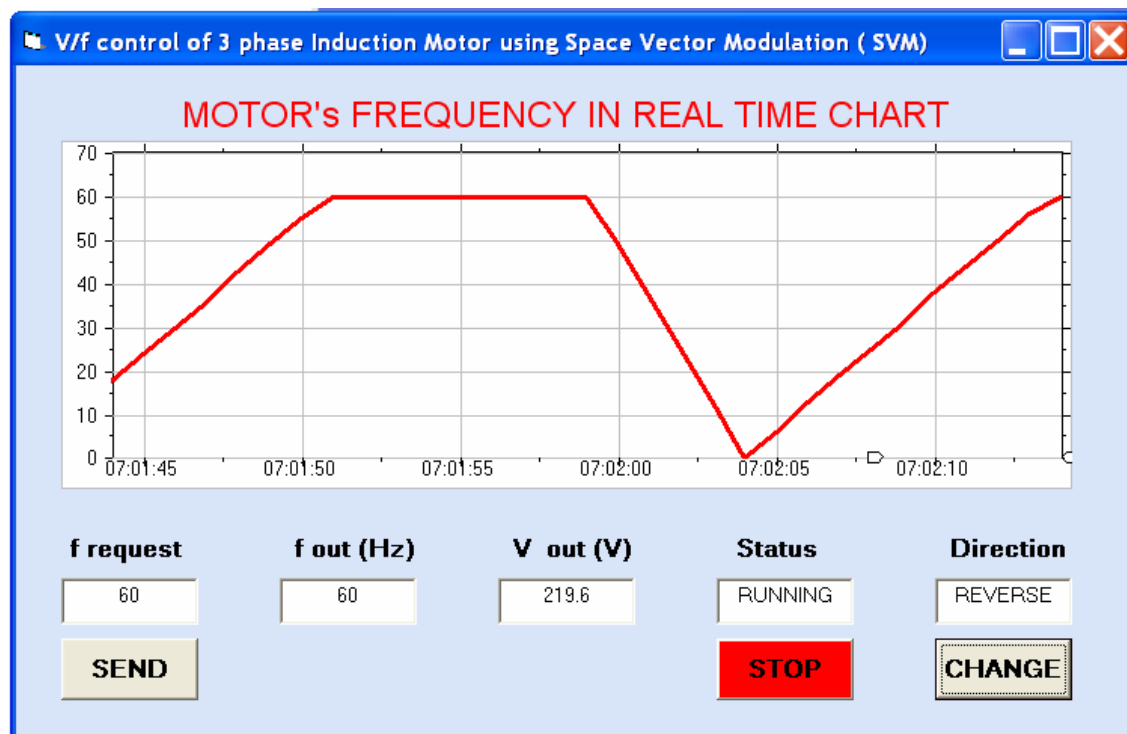
- + Các buttons điều khiển động cơ: RUN, STOP, đảo chiều, biến trở hiệu chỉnh tốc độ.....
- + Các buttons điều khiển LCD: set các thông số cài đặt (thời gian tăng tốc, giảm tốc.....)
- + LCD : hiển thị trạng thái hoạt động của động cơ
- + giao tiếp với PC: nhận giá trị tốc độ đặt từ PC, hiển thị trạng thái hoạt động của motor lên máy tính

## CHƯƠNG 6: KẾT QUẢ ĐẠT ĐƯỢC

### 6.2> PHẦN MỀM GIAO TIẾP VỚI NGƯỜI SỬ DỤNG:



Hình 6.3: phần mềm điều khiển



Hình 6.4: phần mềm điều khiển ( lúc động cơ hoạt động)

## CHƯƠNG 6: KẾT QUẢ ĐẠT ĐƯỢC

---

### 6.2.2) Mô tả:

Phần mềm điều khiển được viết trên ngôn ngữ Visual Basic 6.0

Phần mềm giao tiếp với vi xử lý PIC18F thông qua cổng COM ( chuẩn RS232)

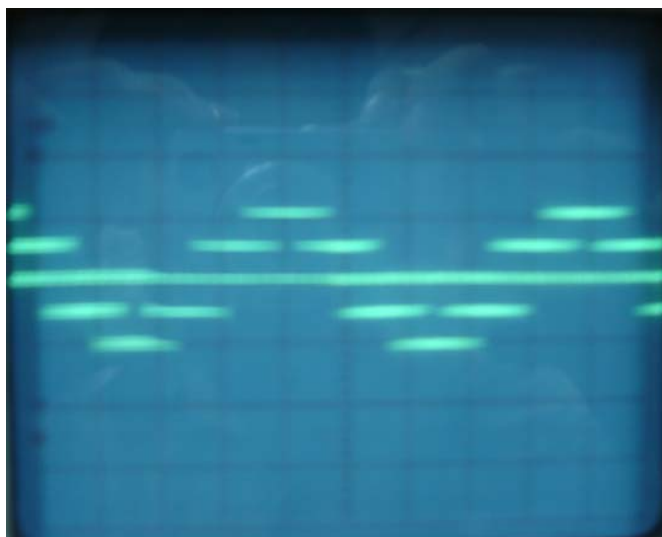
Các nút điều khiển:

- + **RUN / SEND**: Khởi động động cơ / gửi tần số yêu cầu đến vi xử lý
- + **STOP**: dừng động cơ
- + **CHANGE**: đảo chiều động cơ

Các ô hiển thị và nhập liệu:

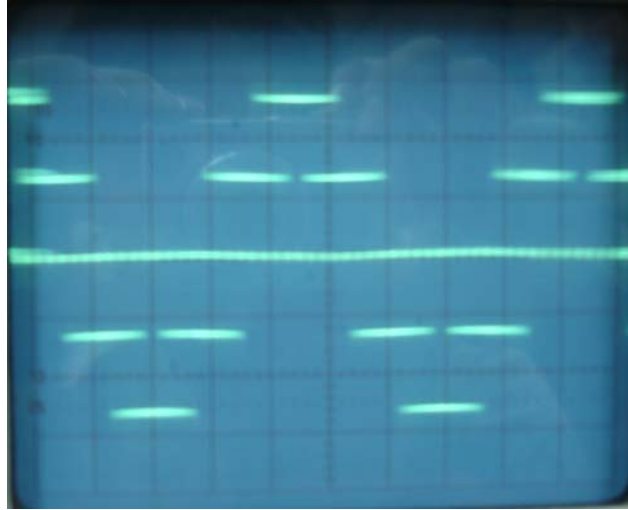
- + **f request**: ô nhập liệu tần số từ bàn phím
- + **f out**: hiển thị giá trị tần số ngõ ra
- + **V out**: hiển thị dạng điện áp ngõ ra (  $V/f = \text{const}$  )
- + **Status**: hiển thị trạng thái động cơ ( RUNNING , STOP... )
- + **Direction**: hiển thị chiều quay của động cơ ( thuận ; nghịch )

### 6.3> DẠNG SÓNG ĐIỆN ÁP NGÕ RA:



Hình 6.5: điện áp pha ngõ ra ( tải R)





Hình 6.6: điện áp pha ngõ ra ( tải R)



Hình 6.7: điện áp dây ngõ ra ( tải động cơ)

### 6.4> HƯỚNG PHÁT TRIỂN:

#### 6.4.1) Khắc phục những khuyết điểm hiện tại:

- + Phần cứng: đã đề cập tại phần 6.1.1 trang 63
- + Phần mềm ( giao tiếp người sử dụng và PIC18F):  
Phát triển thêm phần cài đặt các thông số ( PID cho điều khiển vòng kín)
- + Phương pháp điều khiển:  
Điều khiển vòng kín

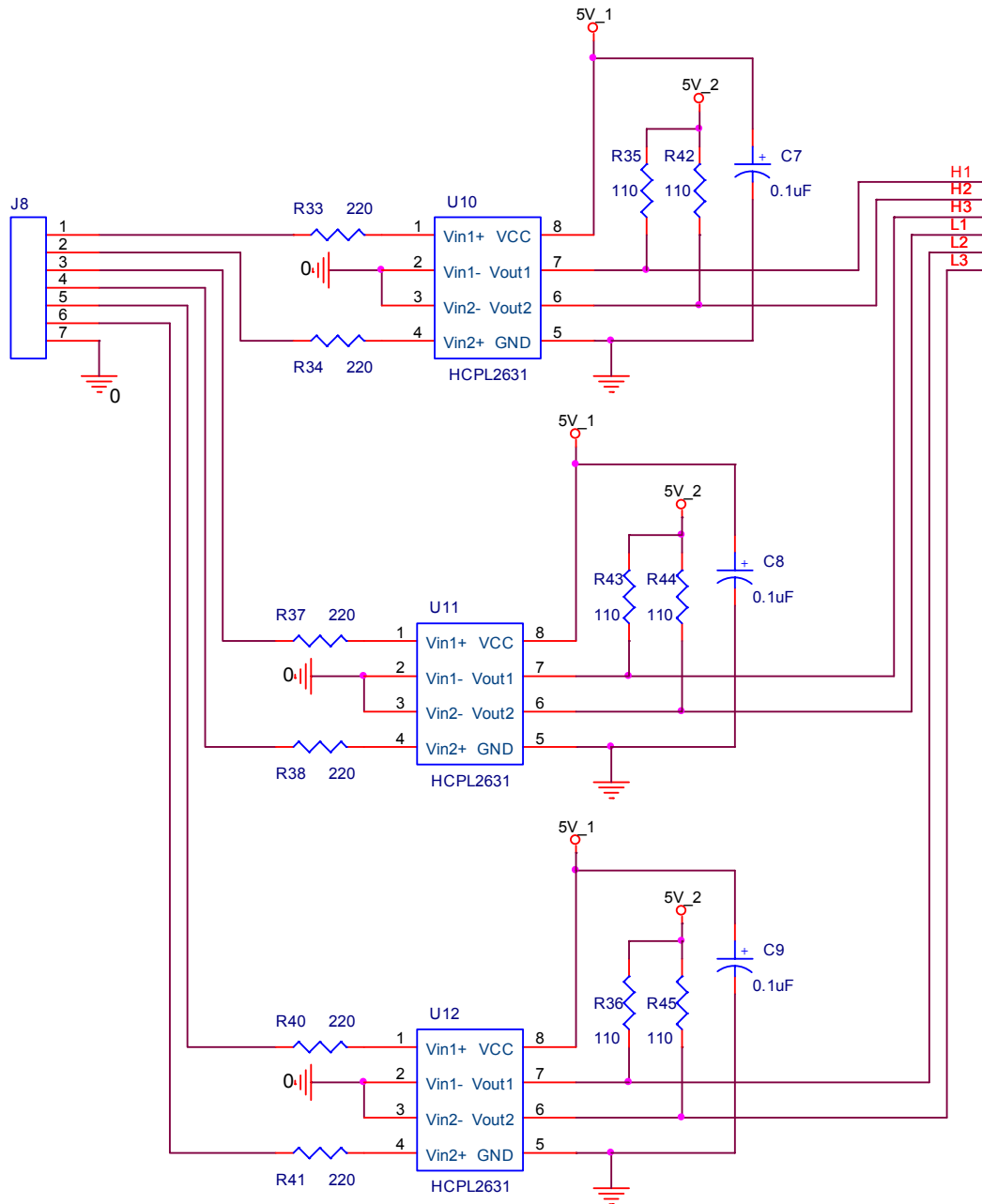
CHƯƠNG 7:  
TÀI LIỆU THAM KHẢO

- [1] Ts. Phan Quốc Dũng , *Truyền Động Điện*
- [2] Ts. Nguyễn Văn Nhò, *Điện tử công suất 1*
- [3] Jon Buroughs, *AN900: Controlling 3 phase induction motors using the PIC18F4431*, Microchip Technology Inc
- [4] Rakesh Parekh, *AN955:V/f Control of 3 phase induction motor using space vector modulation*, Microchip Technology Inc
- [5] Prof. Ali Keyhani, *Pulse-Width Modulation (PWM) Techniques – lecture 25*, Department of Electrical and Computer Engineering The Ohio State University
- [6] **PIC18F4431 datasheet**
- [7] **CCSC User Manual**
- [8] **Flex LCD Driver** – Administrator of CCS Forum
- [9] .....

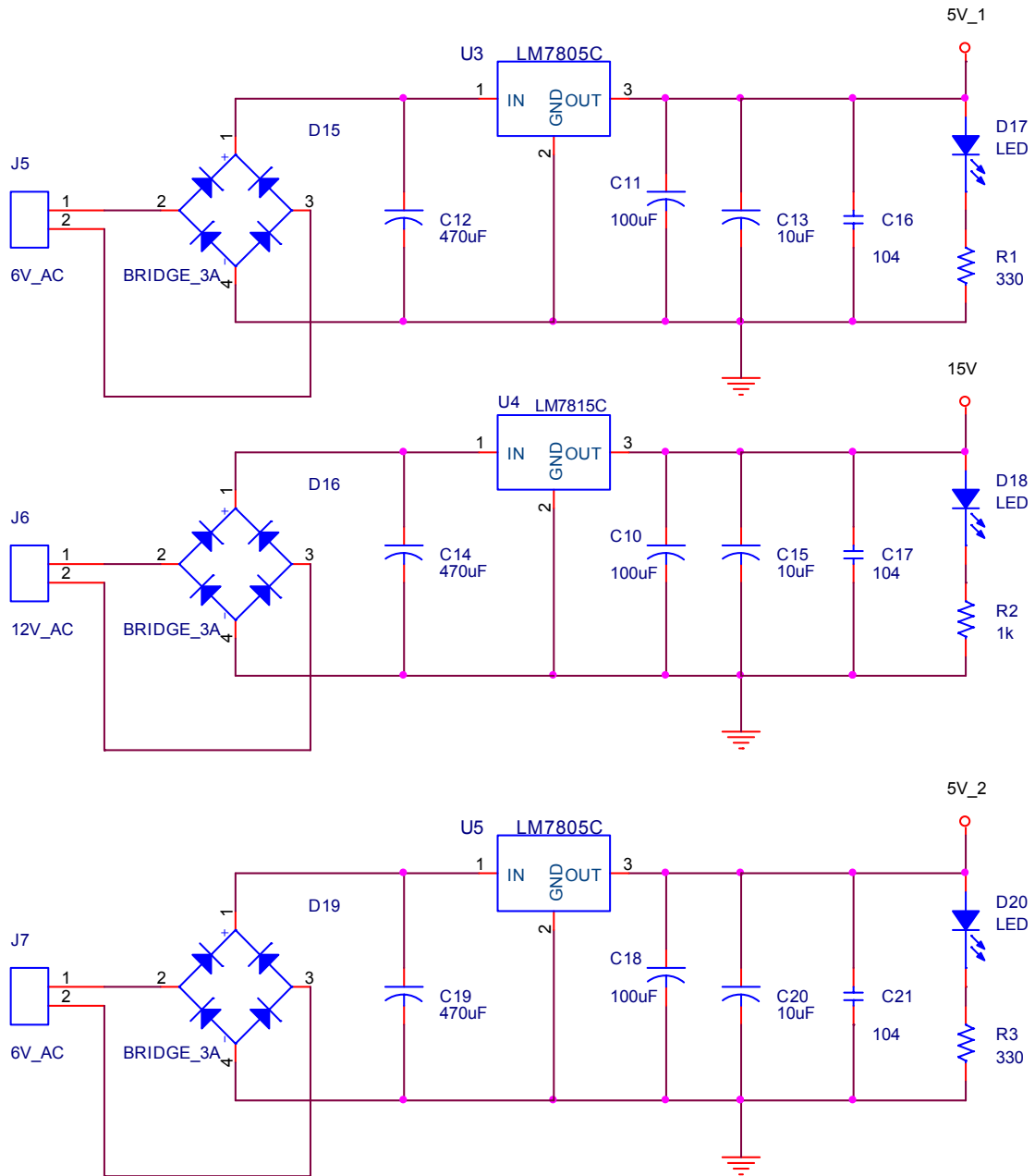
## CHƯƠNG 8: PHỤ LỤC

### 8.1> SƠ ĐỒ MẠCH (VẼ TRÊN ORCAD):

#### 8.1.1) Sơ đồ mạch cách ly

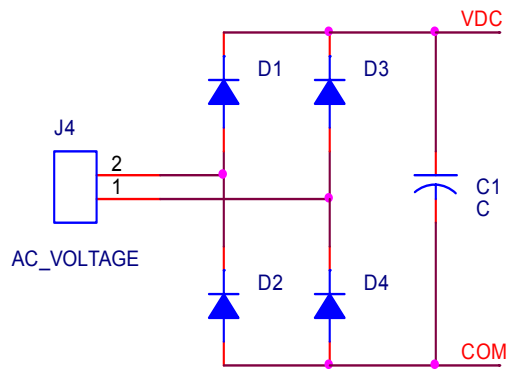
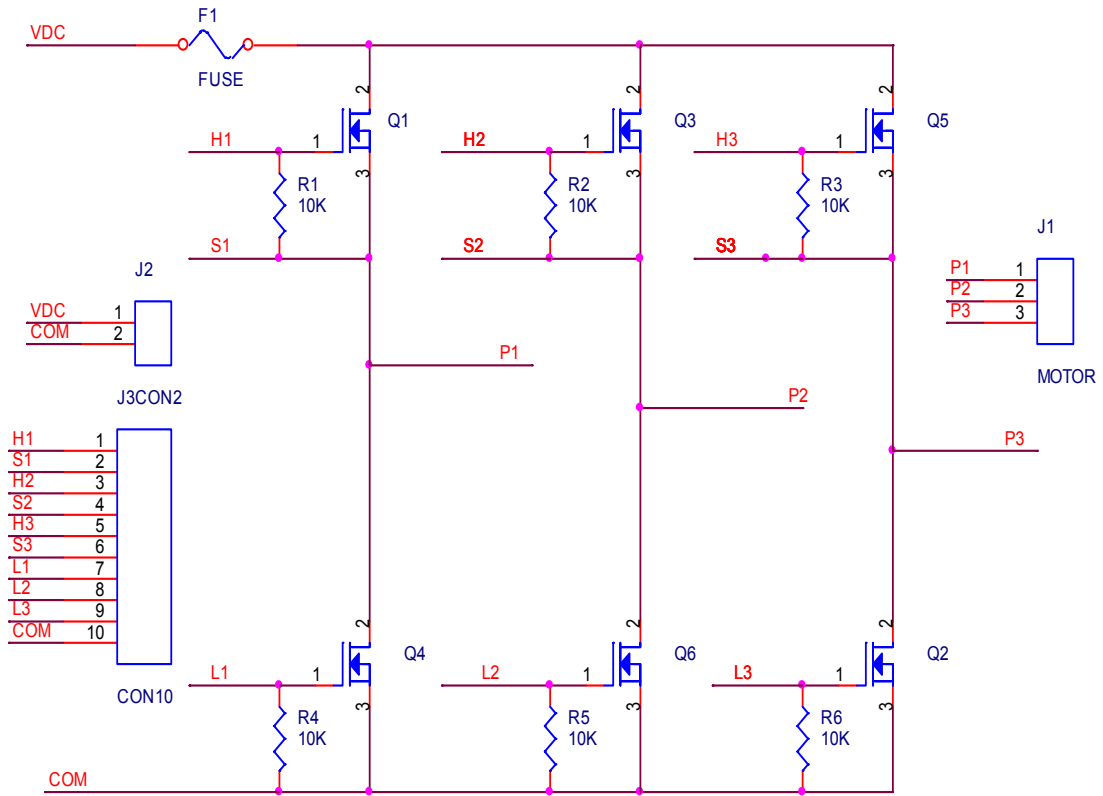


# CHƯƠNG 8: PHỤ LỤC



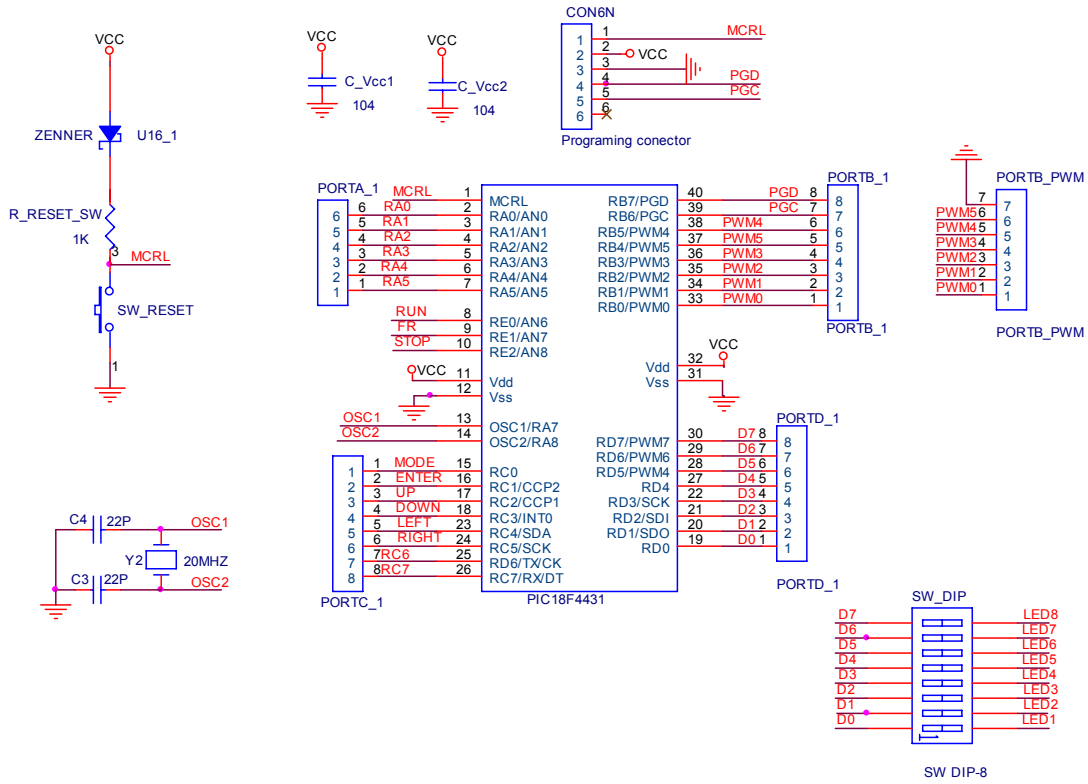


8.1.3) Sơ đồ mạch nghịch lưu :



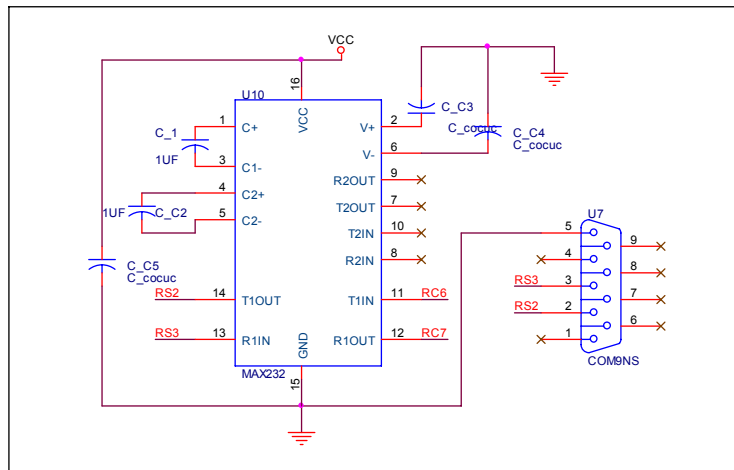
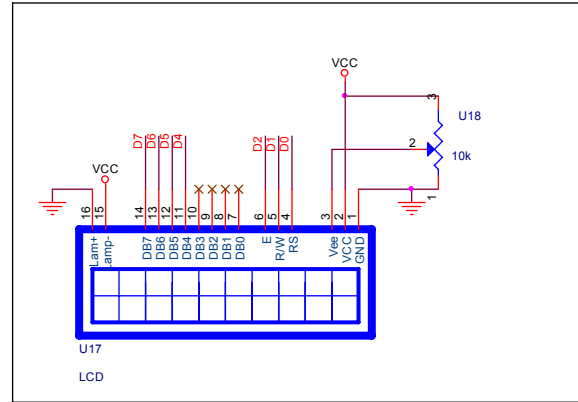
8.1.4) Sơ đồ mạch điều khiển :

8.1.4.a) Phần chính:

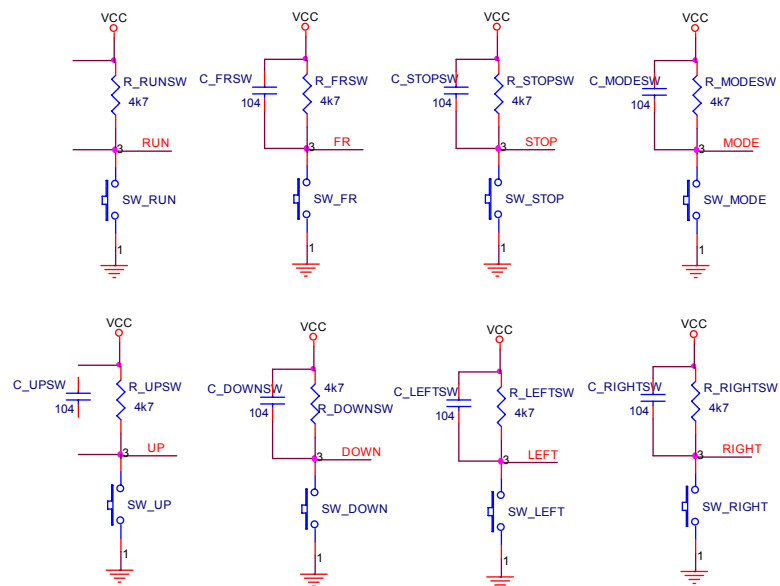


8.1.4.b) Phần hiển thị và giao tiếp máy tính nút ấn:

# CHƯƠNG 8: PHỤ LỤC



## BUTTON





**8.2> CHƯƠNG TRÌNH VIẾT CHO PIC18F4431 :**

Chương trình sau đây được viết trên ngôn ngữ CCS

```
////////////////////////////////////
// MODE 1: R_VAR => dieu khien = RUN , FR, STOP button va` R_VAR
//   MODE 2: AUTO
//       1) nhap gia tri f1 (freq1)
//       2) nhap gia tri f2 (freq2)
//       3) nhap gia tri T ramp_up
//       4) nhap gia tri T ramp_down
//       =>dieu khien = RUN , FR, STOP button va` mode 2 toc do ( thong
button //thay doi toc do= "^" key)
//   MODE 3: PC control
////////////////////////////////////

#include<18F4431.h>           //Header file in project manager of MPLAB
#define * =16 adc=8 HIGH_INTS=TRUE
#define fuses HS,NOWDT,NOPROTECT,PUT,NOBROWNOUT,NOLVP
#define use delay (clock=20000000) //use delay function
#define use rs232(baud=9600,xmit=PIN_C6,rcv=PIN_C7)

#include<flex_LCD.C>        // Other files in project manager of MPLAB

//PTPER*4*sqrt(3)*SIN {data[0] -> data[511] }
int16 const data[512]={
0, 7, 14, 21, 28, 35, 43, 50, 57, 64,
71, 78, 85, 92, 99, 106, 114, 121, 128, 135,
142, 149, 156, 163, 170, 177, 184, 192, 199, 206,
213, 220, 227, 234, 241, 248, 255, 262, 269, 277,
284, 291, 298, 305, 312, 319, 326, 333, 340, 347,
354, 361, 368, 376, 383, 390, 397, 404, 411, 418,
425, 432, 439, 446, 453, 460, 467, 474, 481, 488,
495, 502, 509, 516, 523, 530, 537, 544, 551, 558,
565, 572, 579, 586, 593, 600, 607, 614, 621, 628,
635, 642, 649, 656, 663, 670, 677, 684, 691, 698,
705, 712, 719, 726, 733, 740, 747, 754, 760, 767,
774, 781, 788, 795, 802, 809, 816, 823, 830, 836,
843, 850, 857, 864, 871, 878, 885, 891, 898, 905,
912, 919, 926, 933, 939, 946, 953, 960, 967, 973,
980, 987, 994, 1001,1007,1014,1021,1028,1035,1041,
1048,1055,1062,1068,1075,1082,1089,1095,1102,1109,
1116,1122,1129,1136,1142,1149,1156,1163,1169,1176,
1183,1189,1196,1203,1209,1216,1223,1229,1236,1242,
1249,1256,1262,1269,1275,1282,1289,1295,1302,1308,
1315,1322,1328,1335,1341,1348,1354,1361,1367,1374,
1380,1387,1393,1400,1406,1413,1419,1426,1432,1439,
1445,1452,1458,1465,1471,1477,1484,1490,1497,1503,
1509,1516,1522,1529,1535,1541,1548,1554,1560,1567,
```

## CHƯƠNG 8: PHỤ LỤC

---

```
1573,1579,1586,1592,1598,1605,1611,1617,1623,1630,
1636,1642,1648,1655,1661,1667,1673,1680,1686,1692,
1698,1704,1710,1717,1723,1729,1735,1741,1747,1754,
1760,1766,1772,1778,1784,1790,1796,1802,1808,1814,
1820,1826,1832,1839,1845,1851,1857,1863,1868,1874,
1880,1886,1892,1898,1904,1910,1916,1922,1928,1934,
1940,1946,1951,1957,1963,1969,1975,1981,1986,1992,
1998,2004,2010,2015,2021,2027,2033,2038,2044,2050,
2056,2061,2067,2073,2078,2084,2090,2095,2101,2107,
2112,2118,2124,2129,2135,2140,2146,2151,2157,2163,
2168,2174,2179,2185,2190,2196,2201,2207,2212,2218,
2223,2228,2234,2239,2245,2250,2256,2261,2266,2272,
2277,2282,2288,2293,2298,2304,2309,2314,2320,2325,
2330,2335,2341,2346,2351,2356,2361,2367,2372,2377,
2382,2387,2392,2398,2403,2408,2413,2418,2423,2428,
2433,2438,2443,2448,2453,2458,2463,2468,2473,2478,
2483,2488,2493,2498,2503,2508,2513,2518,2522,2527,
2532,2537,2542,2547,2551,2556,2561,2566,2571,2575,
2580, 2585,2589,2594,2599,2604,2608,2613,2618,2622,
2627,2631,2636,2641,2645,2650,2654,2659,2664,2668,
2673,2677,2682,2686,2691,2695,2699,2704,2708,2713,
2717,2722,2726,2730,2735,2739,2743,2748,2752,2756,
2761,2765,2769,2773,2778,2782,2786,2790,2795,2799,
2803,2807,2811,2815,2820,2824,2828,2832,2836,2840,
2844,2848,2852,2856,2860,2864,2868,2872,2876,2880,
2884,2888,2892,2896,2900,2903,2907,2911,2915,2919,
2923,2927,2930,2934,2938,2942,2945,2949,2953,2957,
2960,2964,2968,2971,2975,2978,2982,2986,2989,2993,
2996,3000    };
```

```
#define RUN          PIN_C0          //all BUTTON is active LOW
#define FR          PIN_E0
#define STOP        PIN_E1

#define MENU        PIN_E2          // back to previous level in MENU
#define OK          PIN_C1
#define UP          PIN_C2
#define DOWN        PIN_C3
#define BACK        PIN_C4
#define NEXT        PIN_C5
```

```
//-----caculation variable
```

```
float          f_float=0,temp_float=0;
```

```
signed long    Vs_angle;          //long=int16
long           update_angle;
```

```
long           M,Vref,Vdc=311;
long           TS=2000,TA,TB,Tz;    //TS=PTPER*4
```

## CHƯƠNG 8: PHỤ LỤC

```

// Fpwm=5Khz=> PTPER=500;
// real TS is PTPER*0.2uS, when Fosc=20M )
//Tz= T0/2
unsigned int  temp_int=0,sector,adc,count_timer1_interrupt;
int          f,f_req;

int1         first_run_flag,direction_flag;

//----- variable in MODEs
int          mode_select=1;
int          return_2_mode_select;
int          f1,f2;
int
    T_ramp_up=10,T_ramp_down=5,T_ramp_up_ms=50,T_ramp_down_ms=25;
//default value
int          eeprom_check;
int1         f_select=0;          //as defaultf_select=0 => f1 ; f_select=1 => f2( use in
mode2)

//-----TEMP variable
int32        count=0,interrupt_number=0;

int1         disable_update_freq=0;  //1= active

#INT_PWM_TB_HIGH //It will generate code to save and restore the machine state,
and will clear the interrupt flag
void PWM_INTERRUPT() //calculating base on "Vref" and "stepsize"
{
interrupt_number=interrupt_number+1;
    TB=data[V_s_angle]; //data=PTPER*4*sqrt(3)*SIN ;
// at 1st RUN: n=0
    TA=data[511-V_s_angle];
    M=Vref*16/Vdc; // mutiply 16=> will shift right 4 bit later
    TA=TA*M;
    TB=TB*M;
    TA=(TA>>4)&0x0FFF; //4TA
    TB=(TB>>4)&0x0FFF; //4TB
    Tz=(TS-TA-TB)/2; //TS=4TS

    if(direction_flag==1) //FORWARD direction//
    {
        V_s_angle=V_s_angle+update_angle;
        if(V_s_angle>511)
        {
            V_s_angle=V_s_angle-511;
            sector=sector+1;
            if(sector>6) //sector (1)->(6)
            {
                sector=1;
            }
        }
    }
}
}
```

```
    else //REVERSE direction//
    { Vs_angle=Vs_angle-update_angle; //n<0 => overflow to value (2^16)-
"negative value" if n is unsigned int
    if(Vs_angle<0)
    { Vs_angle=Vs_angle+511;
      sector=sector-1;
      if(sector==0) //sector (1)->(6)
      { sector=6;
      }
    }
  }

  switch (sector)
  {
    case 1: set_power_pwm0_duty(TS-Tz);
            set_power_pwm2_duty(Tz+TB);
            set_power_pwm4_duty(Tz);
            break;
    case 2: set_power_pwm0_duty(TA+Tz);
            set_power_pwm2_duty(TS-Tz);
            set_power_pwm4_duty(Tz);
            break;
    case 3: set_power_pwm0_duty(Tz);
            set_power_pwm2_duty(TS-Tz);
            set_power_pwm4_duty(Tz+TB);
            break;
    case 4: set_power_pwm0_duty(Tz);
            set_power_pwm2_duty(Tz+TA);
            set_power_pwm4_duty(TS-Tz);
            break;
    case 5: set_power_pwm0_duty(Tz+TB);
            set_power_pwm2_duty(Tz);
            set_power_pwm4_duty(TS-Tz);
            break;
    case 6: set_power_pwm0_duty(TS-Tz);
            set_power_pwm2_duty(Tz);
            set_power_pwm4_duty(Tz+TA);
            break;
  }
}

#INT_TIMER1
void READ_AD_RESULT() //With an internal clock at 20mhz and with the
T1_DIV_BY_8 mode, the timer will increment every 1.6us. It will overflow every
104.8576ms.
{
    if(count_timer1_interupt==10) // 1s
    {
        adc=read_adc();
    }
}
```

---

## CHƯƠNG 8: PHỤ LỤC

---

```
        f_req=adc/4;                //required motor speed in Hz {adc=0-
255 => f_req=1-60 (Hz)}
        if(f_req>60)
            {f_req=60;}

        count_timer1_interupt=1;

    }

    if(disable_update_freq==1)
    {count_timer1_interupt=9;}
    else
    {count_timer1_interupt=count_timer1_interupt+1;}
}

void PORTS_INIT()    //1
{
    set_tris_a(0b00000001);        //RA0 as input ( AD converter)
    set_tris_b(0b11000000);        //PWM0=>PWM5 as output
                                    //PWM6,PWM7 as input
    set_tris_c(0b11111111);        //portC as BUTTON input
    set_tris_d(0b00000000);        //portD as output for display led
}

void PWM_MODULE_INIT()    //2
{
    setup_power_pwm_pins(PWM_COMPLEMENTARY,PWM_COMPLEMENTAR
Y,PWM_COMPLEMENTARY,PWM_OFF );
                                    //module 0(PWM0,PWM1) = COMPLEMENTARY
                                    //module 1(PWM2,PWM3) = COMPLEMENTARY
                                    //module 2(PWM4,PWM5) = COMPLEMENTARY
                                    //module 3(PWM6,PWM7) = OFF
    setup_power_pwm(PWM_CLOCK_DIV_4|PWM_UP_DOWN|PWM_DEAD_CL
OCK_DIV_4,1,0,500,0,1,10);
    PWM_DEAD_CLOCK_DIV_4,          // 1) mode:PWM_CLOCK_DIV_4; PWM_UP_DOWN;
                                    // 2) postscale:1
                                    // 3) time_base:=> first value of timebase
                                    // 4) period:chu ky` xung 6 PWM =500 =>200uS
                                    // 5) compare:0
                                    // 6) compare_postscale:1
                                    // 7) dead_time:10 => Tdeatime=10*0.2=2uS
    set_power_pwm0_duty(0);
    set_power_pwm2_duty(0);
    set_power_pwm4_duty(0);
}

void INTERRUPTS_INIT() //3        //INT_PWM will be enable after run button is
pressed !
```

```
{
    enable_interrupts(INT_TIMER1);
    enable_interrupts(GLOBAL);
}

void ADC_INIT()          //4
{
    setup_adc_ports(sAN0);          //AN0 as analog INPUT
    setup_adc(ADC_CLOCK_DIV_32);
    set_adc_channel(0);
    delay_us(10);
}

void TIMER_INIT()       //5
{
    setup_timer_1(T1_INTERNAL|T1_DIV_BY_8);
    set_timer1(62500);          //All timers count up. When a timer
    reaches the maximum value it will flip over to 0 and continue counting (254, 255, 0, 1,
    2...)
                                //62500*1.6 us =0.1s
}

void PARAs_CAL()
{
    f_float=f;

    temp_float=f_float*3;
    Vref=temp_float;          //Vref(Vphase; motor in deltal mode) at f
    frequency to maintain  $V/f=cont=(220*\sqrt{2}/\sqrt{3})/60=3$ 

    temp_float=0.6144*f_float;          //0.6144=Tpwm*360*n/60 ; n=512
    update_angle=temp_float;          //stepsize is INTERGER after this line
}

void RAM_DOWN_SPEED()
{
    while(f>f_req)
    {
        f=f-1;
        PARAs_CAL();
        delay_ms(T_ramp_down_ms);          // 0.05s/Hz
        if(f<f_req)
        {
            f=f_req;          //f=f_req when ram speed finished !
        }

        switch(mode_select)
        {
            case 1:
                lcd_gotoxy(4,0);
                printf(lcd_putc,"%2.0d",f_req);
            }
        }
    }
}
```

```
        lcd_gotoxy(10,0);
        printf(lcd_putc,"%2.0d",f);
        break;
    case 2:
        lcd_gotoxy(10,0);
        printf(lcd_putc,"%2.0d",f);
        break;
    case 3:
        fputc(f); //send value of f for PC

        lcd_gotoxy(4,0);
        printf(lcd_putc,"%2.0d",f_req);
        lcd_gotoxy(10,0);
        printf(lcd_putc,"%2.0d",f);
        break;
} //end switch
}
}
void RAM_UP_SPEED()
{
    while(f<f_req)
    {
        f=f+1;
        PARAs_CAL();
        delay_ms(T_ramp_up_ms); // 0.1s/Hz
        if(f>f_req)
        {
            f=f_req; //f=f_req when ram speed finished !
        }
        switch(mode_select)
        {
            case 1:
                lcd_gotoxy(4,0);
                printf(lcd_putc,"%2.0d",f_req);
                lcd_gotoxy(10,0);
                printf(lcd_putc,"%2.0d",f);
                break;
            case 2:
                lcd_gotoxy(10,0);
                printf(lcd_putc,"%2.0d",f);
                break;
            case 3:
                fputc(f); //send value of f for PC

                lcd_gotoxy(4,0);
                printf(lcd_putc,"%2.0d",f_req);
                lcd_gotoxy(10,0);
                printf(lcd_putc,"%2.0d",f);
                break;
        } //end switch
    }
}
```

```
    }
}
void default_value_in_EEPROM() //use in MODE2: AUTO ; P18F has 256 bytes
eeprom which address from 0x00 -> 0xFF
{
    write_eeprom(0X00,100); //temp_eeprom for checking at 1st reading
    write_eeprom(0X10,30); //f1
    write_eeprom(0X20,60); //f2
    write_eeprom(0X03,6); //T_ramp_up
    write_eeprom(0x04,3); //T_ramp_down
}
////////////////////////////////////

void MAIN ()
{
    PORTS_INIT(); //1
    lcd_init(); // this subroutine in flex_LCD.C file
    PWM_MODULE_INIT(); //2
    INTERRUPTS_INIT(); //3
    ADC_INIT(); //4
    TIMER_INIT(); //5

MODE_SELECT:
    return_2_mode_select=0; //return_2_mode_select=0 as default
    ;return_2_mode_select=1 when mode button is pressed
    switch (mode_select)
    {
        case 1:
            lcd_gotoxy(1,1);
            printf(lcd_putc,"< M1:Read_AD >");
            lcd_gotoxy(1,0);
            printf(lcd_putc," ok");

            while( 1)
            {
                if(!input(OK))
                {
                    output_bit(PIN_D3,1);
                    delay_ms(200);
                    output_bit(PIN_D3,0);
                    goto MODE_R_VAR;
                }
                if(!input(NEXT))
                {
                    output_bit(PIN_D3,1);
                    delay_ms(200);
                    output_bit(PIN_D3,0);
                    mode_select=2;
                    goto MODE_SELECT;
                }
            }
        }
    }
}
```



```
        if(!input(BACK))
        {
            output_bit(PIN_D3,1);
            delay_ms(200);
            output_bit(PIN_D3,0);
            mode_select=3;
            goto MODE_SELECT;
        }
    }
    break;
case 2:
    lcd_gotoxy(1,1);
    printf(lcd_putc,"< M2:Set freq >");
    lcd_gotoxy(1,0);
    printf(lcd_putc,"          ok");
    while( 1)
    {
        if(!input(OK))
        {
            output_bit(PIN_D3,1);
            delay_ms(200);
            output_bit(PIN_D3,0);
            goto MODE_AUTO;
        }
        if(!input(NEXT))
        {
            output_bit(PIN_D3,1);
            delay_ms(200);
            output_bit(PIN_D3,0);
            mode_select=3;
            goto MODE_SELECT;
        }
        if(!input(BACK))
        {
            output_bit(PIN_D3,1);
            delay_ms(200);
            output_bit(PIN_D3,0);
            mode_select=1;
            goto MODE_SELECT;
        }
    }
    break;
case 3:
    lcd_gotoxy(1,1);
    printf(lcd_putc,"< M3:COMPUTER >");
    lcd_gotoxy(1,0);
    printf(lcd_putc,"          ok");
    while( 1)
    {
        if(!input(OK))
        {
            output_bit(PIN_D3,1);
            delay_ms(200);
```

```
        output_bit(PIN_D3,0);
        goto MODE_COMPUTER;
    }
    if(!input(NEXT))
    {
        output_bit(PIN_D3,1);
        delay_ms(200);
        output_bit(PIN_D3,0);
        mode_select=1;
        goto MODE_SELECT;
    }
    if(!input(BACK))
    {
        output_bit(PIN_D3,1);
        delay_ms(200);
        output_bit(PIN_D3,0);
        mode_select=2;
        goto MODE_SELECT;
    }
}
break;

}

//=====
//                                MODE_R_VAR
//=====
MODE_R_VAR:
    lcd_gotoxy(1,1);
    printf(lcd_putc,"T ramp up :?? s"); //clear screen
    lcd_gotoxy(1,0);
    printf(lcd_putc,"          "); //clear screen
T_RAMP_UP_MODE1:
    while(1)
    {
        if(!input(OK)) // OK button is pressed?
        {
            output_bit(PIN_D3,1);
            delay_ms(200);
            output_bit(PIN_D3,0);

            goto NEXT_MODE1;
        }
        if(!input(UP)) // UP button is pressed?
        {
            output_bit(PIN_D3,1);
            delay_ms(200);
            output_bit(PIN_D3,0);
            T_ramp_up=T_ramp_up+1;
            if(T_ramp_up>20)
            {T_ramp_up=5;}
        }
        if(!input(DOWN)) // DOWN button is
pressed?
```

## CHƯƠNG 8: PHỤ LỤC

---

```

    {
        output_bit(PIN_D3,1);
        delay_ms(200);
        output_bit(PIN_D3,0);
        T_ramp_up=T_ramp_up-1;
        if(T_ramp_up<5) //6Hz is
minimum frequency for motor can RUN
        {T_ramp_up=20;}
    }

    lcd_gotoxy(13,1); //print new value of f1
    printf(lcd_putc,"%2.0d",T_ramp_up);

    if(!input(NEXT)) // OK button is pressed?
    {
        output_bit(PIN_D3,1);
        delay_ms(500);
        output_bit(PIN_D3,0);

        T_ramp_up_ms=T_ramp_up*5;
        goto T_RAMP_DOWN_MODE1;
    }

    if(!input(MENU))
    {
        output_bit(PIN_D3,1);
        delay_ms(200);
        output_bit(PIN_D3,0);
        goto MODE_SELECT; //return to MAIN menu (motor is
running => user press stop BUTTON => want to return to main menu)
    }
} //end while T ramp up mode1:

T_RAMP_DOWN_MODE1:
    lcd_gotoxy(1,0);
    printf(lcd_putc,"T ramp down:?? s");
while(1)
{

    if(!input(UP)) // UP button is pressed?
    {
        output_bit(PIN_D3,1);
        delay_ms(200);
        output_bit(PIN_D3,0);
        T_ramp_down=T_ramp_down+1;
        if(T_ramp_down>20)
        {T_ramp_down=5;}
    }

    if(!input(DOWN)) // DOWN button is
pressed?
    {
        output_bit(PIN_D3,1);
        delay_ms(200);
        output_bit(PIN_D3,0);
    }
}

```

```

        T_ramp_down=T_ramp_down-1;
        if(T_ramp_down<3)                                //6Hz is
minimum frequency for motor can RUN
        {T_ramp_down=20;}
    }

    lcd_gotoxy(13,0);                                    //print new value of f1
    printf(lcd_putc,"%2.0d",T_ramp_down);

    if(!input(NEXT))                                     // OK button is pressed?
    {
        output_bit(PIN_D3,1);
        delay_ms(200);
        output_bit(PIN_D3,0);

        T_ramp_down_ms=T_ramp_down*5;
        goto NEXT_MODE1;
    }

    if(!input(MENU))
    {
        output_bit(PIN_D3,1);
        delay_ms(200);
        output_bit(PIN_D3,0);
        goto MODE_SELECT;                                //return to MAIN menu (motor is
running => user press stop BUTTON => want to return to main menu)
    }
} //end while T ramp down mode1:
NEXT_MODE1:
MODE_R_VAR_return_from_stop_button:
    lcd_gotoxy(1,1);
    printf(lcd_putc,"M1 freq READY "); //clear screen
    lcd_gotoxy(1,0);
    printf(lcd_putc,"Rv ?? 2 RUN "); // ?? wil be cleard when value update

//----- DEFAULT VALUE -----
    first_run_flag=1;
    direction_flag=1;

    f=0,f_req=0;
    Vs_angle=0;                                         //default value for 1st Vs; direction=1
    update_angle=0;
    sector=1;

    count_timer1_interupt=10; // get 1st value of A/D
//-----

while (1) //MAIN of MODE 1

```

```
{
  lcd_gotoxy(4,0);
  printf(lcd_putc,"%2.0d",f_req); //int_timer1 is enable as default to read AD
result =>f_req

  if(return_2_mode_select==1) //return_2_mode_select=0 as default
;return_2_mode_select=1 when mode button is pressed
  {
    goto MODE_SELECT; //return to MODE select
  }
  if(!input(MENU))
  {
    output_bit(PIN_D3,1);
    delay_ms(200);
    output_bit(PIN_D3,0);
    goto MODE_SELECT; //return to MAIN menu (motor is
running => user press stop BUTTON => want to return to main menu)
  }
  //RUN Button is pressed ? -----
  if(!input(RUN))
  {
    output_bit(PIN_D3,1);
    delay_ms(200);
    output_bit(PIN_D3,0);
    enable_interrupts(INT_PWM);//int_PWM must be enable after RUN
button is pressed to prevent HIGH CURRENT ( don't know why it is, just seen it in
testing if int_PWM enable b4 run button is pressed !!!)

    lcd_gotoxy(1,1);
    printf(lcd_putc," "); //clear screen
    lcd_gotoxy(1,0);
    printf(lcd_putc," ");
    lcd_gotoxy(1,1);
    printf(lcd_putc,"M1 freq fo DIR"); //clear screen
    lcd_gotoxy(1,0);
    printf(lcd_putc,"AD ?? ?? ?"); // ?? will be cleared when value update

    switch(direction_flag) // direction display
    {
      case 1:
        lcd_gotoxy(15,0);
        printf(lcd_putc,"F");
        break;
      case 0:
        lcd_gotoxy(15,0);
        printf(lcd_putc,"R");
        break;
    }

    if(first_run_flag==1) //RAM UP SPEED at 1st RUN
    {
      RAM_UP_SPEED();
    }
  }
}
```

---

## CHƯƠNG 8: PHỤ LỤC

---

```

                                first_run_flag=0;    //disable RUN button when motor is
RUNNING
                                }
                                } //end if(!PIN_E0)
                                //-----END of "RUN Button is pressed ?"
                                while (first_run_flag==0)
                                {
                                RAM_UP_SPEED();
                                RAM_DOWN_SPEED();

                                if(!input(FR))//FR Button is pressed ? -----
                                -----
                                {
                                    output_bit(PIN_D3,1);
                                    delay_ms(200);
                                    output_bit(PIN_D3,0);
                                    disable_interrups(INT_TIMER1);    //stop reading A/D

                                    temp_int=f_req;    //save current f_req
                                    f_req=0;
                                    RAM_DOWN_SPEED();
                                    direction_flag=direction_flag+1; //complement
                                direction_flag=direction_flag+1
                                    switch(direction_flag)    //change direction display
                                    {   case 1:
                                        lcd_gotoxy(15,0);
                                        printf(lcd_putc,"F");
                                        break;
                                        case 0:
                                        lcd_gotoxy(15,0);
                                        printf(lcd_putc,"R");
                                        break;
                                    }

                                    f_req=temp_int;    //restore f_req
                                    RAM_UP_SPEED();

                                    enable_interrups(INT_TIMER1);    //enable reading A/D
                                }//-----END of "FR Button is pressed
                                ?"

                                if(!input(STOP))
                                {
                                    output_bit(PIN_D3,1);
                                    delay_ms(200);
                                    output_bit(PIN_D3,0);

                                STOP_MOTOR_MODE1:    //lable for MODE
                                button is pressed => stop motor
                                    //disable_interrups(INT_TIMER1);    //stop reading A/D
                                    disable_update_freq=1;

```

## CHƯƠNG 8: PHỤ LỤC

---

```
        f_req=0;
        RAM_DOWN_SPEED();

        first_run_flag=1;           //prepare for RAM_UP if
RUN button is pressed next time

        //enable_interrupts(INT_TIMER1);   //enable reading A/D
        disable_update_freq=0;
        disable_interrupts(INT_PWMTB);    // it'll enable later when
run button is pressed

        goto MODE_R_VAR_return_from_stop_button;    //return to
current mode

    }
    if(!input(MENU))
    {
        output_bit(PIN_D3,1);
        delay_ms(200);
        output_bit(PIN_D3,0);
        return_2_mode_select=1;
        goto STOP_MOTOR_MODE1;
    }
} //end while(first_run=0)

} //while(1)

//=====
//                END of MODE R_VAR                //
//=====

//=====
//                MODE AUTO                //
//=====

MODE_AUTO:
lcd_gotoxy(1,1);
printf(lcd_putc,"T ramp up :?? s"); //clear screen
lcd_gotoxy(1,0);
printf(lcd_putc,"                "); //clear screen
T_RAMP_UP_MODE2:
while(1)
{
    if(!input(OK)) // OK button is pressed?
    {
        output_bit(PIN_D3,1);
        delay_ms(200);
        output_bit(PIN_D3,0);

        goto NEXT_MODE2;
    }
}
```

---

## CHƯƠNG 8: PHỤ LỤC

---

```
        if(!input(UP))                                // UP button is pressed?
        {
            output_bit(PIN_D3,1);
            delay_ms(200);
            output_bit(PIN_D3,0);
            T_ramp_up=T_ramp_up+1;
            if(T_ramp_up>20)
                {T_ramp_up=5;}
        }
pressed? if(!input(DOWN))                            // DOWN button is
        {
            output_bit(PIN_D3,1);
            delay_ms(200);
            output_bit(PIN_D3,0);
            T_ramp_up=T_ramp_up-1;
            if(T_ramp_up<5)                          //6Hz is
                {T_ramp_up=20;}                      minimum frequency for motor can RUN
        }

        lcd_gotoxy(13,1);                            //print new value of f1
        printf(lcd_putc,"%2.0d",T_ramp_up);

        if(!input(NEXT))                             // OK button is pressed?
        {
            output_bit(PIN_D3,1);
            delay_ms(500);
            output_bit(PIN_D3,0);

            T_ramp_up_ms=T_ramp_up*5;
            goto T_RAMP_DOWN_MODE3;
        }

        if(!input(MENU))
        {
            output_bit(PIN_D3,1);
            delay_ms(200);
            output_bit(PIN_D3,0);
            goto MODE_SELECT;                          //return to MAIN menu (motor is
running => user press stop BUTTON => want to return to main menu)
        }
    }//end while T ramp up mode3:

T_RAMP_DOWN_MODE2:
    lcd_gotoxy(1,0);
    printf(lcd_putc,"T ramp down:?? s");
while(1)
    {

        if(!input(UP))                                // UP button is pressed?
        {
            output_bit(PIN_D3,1);
            delay_ms(200);
```



```
        output_bit(PIN_D3,0);
        T_ramp_down=T_ramp_down+1;
        if(T_ramp_down>20)
            {T_ramp_down=5;}
    }
    if(!input(DOWN))                // DOWN button is
pressed?
    {
        output_bit(PIN_D3,1);
        delay_ms(200);
        output_bit(PIN_D3,0);
        T_ramp_down=T_ramp_down-1;
        if(T_ramp_down<3)           //6Hz is
minimum frequency for motor can RUN
            {T_ramp_down=20;}
    }

    lcd_gotoxy(13,0);                //print new value of f1
    printf(lcd_putc,"%2.0d",T_ramp_down);

    if(!input(NEXT))                // OK button is pressed?
    {
        output_bit(PIN_D3,1);
        delay_ms(200);
        output_bit(PIN_D3,0);

        T_ramp_down_ms=T_ramp_down*5;
        goto NEXT_MODE2 ;
    }

    if(!input(MENU))
    {
        output_bit(PIN_D3,1);
        delay_ms(200);
        output_bit(PIN_D3,0);
        goto MODE_SELECT;           //return to MAIN menu (motor is
running => user press stop BUTTON => want to return to main menu)
    }
} //end while T ramp down mode2:

NEXT_MODE2:
MODE_AUTO_return_from_stop_button:

if(return_2_mode_select==1)        //return_2_mode_select=0 as default
;return_2_mode_select=1 when mode button is pressed
    {
        goto MODE_SELECT;         //return to MODE select
    }

disable_interruptions(INT_TIMER1); //disable reading AD from R_VAR
```

```

lcd_gotoxy(1,1);
printf(lcd_putc,"          "); //clear screen
lcd_gotoxy(1,0);
printf(lcd_putc,"          ");
lcd_gotoxy(1,1);
printf(lcd_putc,"M2 f1    ->"); //clear screen
lcd_gotoxy(1,0);
printf(lcd_putc,"Au ??      "); // ?? will be cleared when value update

eeprom_check=read_eeprom(0x00);
if(eeprom_check!=100) //100 is default set for
eeprom_check
{   default_value_in_EEPROM(); //load default value
}

f1_select:
//f1=read_eeprom(0x10); //read f_req2 value in eeprom
f1=30;

while(1)
{
    if(!input(OK)) // OK button is pressed?
    {
        output_bit(PIN_D3,1);
        delay_ms(200);
        output_bit(PIN_D3,0);

        goto NEXT_MODE2;
    }
    if(!input(UP)) // UP button is pressed?
    {
        output_bit(PIN_D3,1);
        delay_ms(200);
        output_bit(PIN_D3,0);

        f1=f1+1;
        if(f1>60)
        {f1=6;}
    }
    if(!input(DOWN)) // DOWN button is
pressed?
    {
        output_bit(PIN_D3,1);
        delay_ms(200);
        output_bit(PIN_D3,0);

        f1=f1-1;
        if(f1<6) //6Hz is minimum
frequency for motor can RUN
        {f1=60;}
    }
}

```

## CHƯƠNG 8: PHỤ LỤC

---

```
lcd_gotoxy(4,0); //print new value of f1
printf(lcd_putc,"%d",f1);

if(!input(NEXT)) // OK button is pressed?
{
    output_bit(PIN_D3,1);
    delay_ms(200);
    output_bit(PIN_D3,0);

    write_eeprom(0X10,f1); //save the value of f1
in eeprom for using when POWER ON next time
    goto f2_SELECT;

}

if(!input(MENU))
{
    output_bit(PIN_D3,1);
    delay_ms(200);
    output_bit(PIN_D3,0);
    goto MODE_SELECT; //return to MAIN menu (motor is
running => user press stop BUTTON => want to return to main menu)
}
} //end while f1_select:

f2_SELECT:
lcd_gotoxy(1,1);
printf(lcd_putc," "); //clear screen
lcd_gotoxy(1,0);
printf(lcd_putc," ");
lcd_gotoxy(1,1);
printf(lcd_putc,"M2 f2 "); //clear screen
lcd_gotoxy(1,0);
printf(lcd_putc,"Au ?? ok");// ?? will be cleared when value update

//f2=read_eeprom(0x20); //read frep2 value in eeprom
f2=60;

while(1)
{
    if(!input(UP)) // UP button is pressed?
    {
        output_bit(PIN_D3,1);
        delay_ms(200);
        output_bit(PIN_D3,0);

        f2=f2+1;
        if(f2>60)
        {f2=6;}
    }
    if(!input(DOWN)) // DOWN button is
pressed?
```

---

## CHƯƠNG 8: PHỤ LỤC

---

```
        {
            output_bit(PIN_D3,1);
            delay_ms(200);
            output_bit(PIN_D3,0);

            f2=f2-1;
            if(f2<6) //6Hz is minimum frequency for
motor can RUN
                {f2=60;}
        }

        lcd_gotoxy(4,0); //print new value of f1
        printf(lcd_putc,"%d",f2);

        if(!input(OK)) // OK button is pressed?
        {
            output_bit(PIN_D3,1);
            delay_ms(200);
            output_bit(PIN_D3,0);

            write_eeprom(0X20,f2);
            goto NEXT_DEFAULT_VALUE_MODE3;
        }
        if(!input(MENU))
        {
            output_bit(PIN_D3,1);
            delay_ms(200);
            output_bit(PIN_D3,0);

            goto MODE_SELECT; //return to MAIN menu (motor is
running => user press stop BUTTON => want to return to main menu)
        }
    } //end while f2_select:
```

NEXT\_DEFAULT\_VALUE\_MODE3:

//----- DEFAULT VALUE -----

```
    first_run_flag=1;
    direction_flag=1;
```

```
    f=0,f_req=0;
    Vs_angle=0; //default value for 1st Vs; direction=1
    update_angle=0;
    sector=1;
```

```
    //count_timer1_interupt=10; => different from MODE1: we don't need to
interupt timer to get AD result
```

//-----

```
        lcd_gotoxy(1,1); // AVAILABLE to RUN
screen IN mode 2 AUTO
        printf(lcd_putc," ");
        lcd_gotoxy(1,0);
        printf(lcd_putc," ");
```

```
    lcd_gotoxy(1,1);
    printf(lcd_putc,"M1 f1 f2 READY ");
    lcd_gotoxy(1,0);
    printf(lcd_putc,"Au ?? ?? 2 RUN "); // ?? will be cleared when value update
    lcd_gotoxy(4,0);
    printf(lcd_putc,"%2.0d",f1);
    lcd_gotoxy(7,0); //print value of f1
    printf(lcd_putc,"%2.0d",f2);

while (1) //MAIN of MODE 2
{
    if(!input(MENU))
    {
        output_bit(PIN_D3,1);
        delay_ms(200);
        output_bit(PIN_D3,0);
        goto MODE_SELECT; //return to MAIN menu (motor is
running => user press stop BUTTON => want to return to main menu)
    }

    //RUN Button is pressed ? -----
    if(!input(RUN))
    {
        output_bit(PIN_D3,1);
        delay_ms(200);
        output_bit(PIN_D3,0);
        enable_interrupts(INT_PWM); //int_PWM must be enable after
RUN button is pressed to prevent HIGH CURRENT ( don't know why it is, just seen it
in testing if int_PWM enable b4 run button is pressed !!!)

        lcd_gotoxy(1,1);
        printf(lcd_putc," "); //clear screen
        lcd_gotoxy(1,0);
        printf(lcd_putc," ");
        lcd_gotoxy(1,1);
        printf(lcd_putc,"M2 f1 f2 fo DIR"); //clear screen
        lcd_gotoxy(1,0);
        printf(lcd_putc,"Au ?? ?? ?? ? "); // ?? will be cleared when value
update

        lcd_gotoxy(4,0); //print value of f1
        printf(lcd_putc,"%2.0d",f1);

        lcd_gotoxy(7,0); //print value of f1
        printf(lcd_putc,"%2.0d",f2);

        switch(direction_flag) // direction display
        {
            case 1:
                lcd_gotoxy(15,0);

```

---

## CHƯƠNG 8: PHỤ LỤC

---

```
                printf(lcd_putc,"F");
                break;
            case 0:
                lcd_gotoxy(15,0);
                printf(lcd_putc,"R");
                break;
        }

    if(first_run_flag==1) //RAM UP SPEED at 1st RUN
    {
        f_req=f1;          //as default of MODE2
        RAM_UP_SPEED();
        first_run_flag=0;  //disable RUN button when motor is
RUNNING
    }
} //-----END of "RUN Button is pressed ?"

while (first_run_flag==0)
{
    if(!input(FR))//FR Button is pressed ? -----
    -----
    {
        output_bit(PIN_D3,1);
        delay_ms(200);
        output_bit(PIN_D3,0);
        temp_int=f_req;          //save current f_req
        f_req=0;
        RAM_DOWN_SPEED();
        direction_flag=direction_flag+1; //complement
direction_flag=direction_flag+1
        switch(direction_flag)    //change direction display
        {
            case 1:
                lcd_gotoxy(15,0);
                printf(lcd_putc,"F");
                break;
            case 0:
                lcd_gotoxy(15,0);
                printf(lcd_putc,"R");
                break;
        }

        f_req=temp_int;          //restore f_req
        RAM_UP_SPEED();
    }
} //-----END of "FR Button is pressed
?"

if(!input(STOP))
{
    output_bit(PIN_D3,1);
```

---

## CHƯƠNG 8: PHỤ LỤC

---

```

        delay_ms(200);
        output_bit(PIN_D3,0);
STOP_MOTOR_MODE2:                                //lable for
MODE button is pressed => stop motor
        disable_interrups(INT_TIMER1);           //stop reading A/D
        f_req=0;
        RAM_DOWN_SPEED();

        first_run_flag=1;                         //prepare for
RAM_UP if RUN button is pressed next time

        disable_interrups(INT_PWMTB);           // it'll enable later
when run button is pressed
        enable_interrups(INT_TIMER1);           //enable reading A/D
        goto MODE_AUTO_return_from_stop_button;
//return to current mode

    }

    if(!input(MENU))
    {
        output_bit(PIN_D3,1);
        delay_ms(200);
        output_bit(PIN_D3,0);
        return_2_mode_select=1;
        goto STOP_MOTOR_MODE2;
    }

    if(!input(NEXT))                             //change freq = f2
(f1) as muplti speed mode ( mode2)
    {
        output_bit(PIN_D3,1);
        delay_ms(200);
        output_bit(PIN_D3,0);
        f_select=f_select+1;                     //complement bit
f_select

        if(f_select==0)
        { f_req=f1;}
        else
        { f_req=f2;}

        RAM_UP_SPEED();                         // RAM_SPEED
to reach the new request frequency
        RAM_DOWN_SPEED();

    }
} //end while(first_run=0)

} //while(1)
//=====
```

## CHƯƠNG 8: PHỤ LỤC

---

```
//          MODE COMPUTER          //
//=====

MODE_COMPUTER:

disable_interrupts(INT_TIMER1);      //disable reading AD from R_VAR
disable_interrupts(INT_PWMTB);

    lcd_gotoxy(1,1);
    printf(lcd_putc,"T ramp up :?? s"); //clear screen
    lcd_gotoxy(1,0);
    printf(lcd_putc,"          "); //clear screen
T_RAMP_UP_MODE3:
    while(1)
    {
        if(!input(OK))                // OK button is pressed?
        {
            output_bit(PIN_D3,1);
            delay_ms(200);
            output_bit(PIN_D3,0);

            goto NEXT_MODE3;
        }
        if(!input(UP))                // UP button is pressed?
        {
            output_bit(PIN_D3,1);
            delay_ms(200);
            output_bit(PIN_D3,0);
            T_ramp_up=T_ramp_up+1;
            if(T_ramp_up>20)
            {T_ramp_up=5;}
        }
        if(!input(DOWN))              // DOWN button is
pressed?
        {
            output_bit(PIN_D3,1);
            delay_ms(200);
            output_bit(PIN_D3,0);
            T_ramp_up=T_ramp_up-1;
            if(T_ramp_up<5)            //6Hz is
minimum frequency for motor can RUN
            {T_ramp_up=20;}
        }

        lcd_gotoxy(13,1);             //print new value of f1
        printf(lcd_putc,"%2.0d",T_ramp_up);

        if(!input(NEXT))              // OK button is pressed?
        {
            output_bit(PIN_D3,1);
            delay_ms(500);
            output_bit(PIN_D3,0);
        }
    }
}
```



```
        T_ramp_up_ms=T_ramp_up*5;
        goto T_RAMP_DOWN_MODE3;
    }

    if(!input(MENU))
    {
        output_bit(PIN_D3,1);
        delay_ms(200);
        output_bit(PIN_D3,0);
        goto MODE_SELECT; //return to MAIN menu (motor is
running => user press stop BUTTON => want to return to main menu)
    }
} //end while T ramp up mode3:

T_RAMP_DOWN_MODE3:
    lcd_gotoxy(1,0);
    printf(lcd_putc,"T ramp down:?? s");
while(1)
{
    if(!input(UP)) // UP button is pressed?
    {
        output_bit(PIN_D3,1);
        delay_ms(200);
        output_bit(PIN_D3,0);
        T_ramp_down=T_ramp_down+1;
        if(T_ramp_down>20)
        {T_ramp_down=5;}
    }
    if(!input(DOWN)) // DOWN button is
pressed?
    {
        output_bit(PIN_D3,1);
        delay_ms(200);
        output_bit(PIN_D3,0);
        T_ramp_down=T_ramp_down-1;
        if(T_ramp_down<3) //6Hz is
minimum frequency for motor can RUN
        {T_ramp_down=20;}
    }

    lcd_gotoxy(13,0); //print new value of f1
    printf(lcd_putc,"%2.0d",T_ramp_down);

    if(!input(NEXT)) // OK button is pressed?
    {
        output_bit(PIN_D3,1);
        delay_ms(200);
        output_bit(PIN_D3,0);

        T_ramp_down_ms=T_ramp_down*5;
        goto NEXT_MODE3 ;
    }
}
```

```
        if(!input(MENU))
        {
            output_bit(PIN_D3,1);
            delay_ms(200);
            output_bit(PIN_D3,0);
            goto MODE_SELECT;          //return to MAIN menu (motor is
running => user press stop BUTTON => want to return to main menu)
        }
    } //end while T ramp down mode3:

NEXT_MODE3:
//----- DEFAULT VALUE -----
    first_run_flag=1;
    direction_flag=1;

    f=0,f_req=0;
    Vs_angle=0;          //default value for 1st Vs; direction=1
    update_angle=0;
    sector=1;
//-----

    lcd_gotoxy(1,1);
    printf(lcd_putc,"M3 freq fo DIR"); //clear screen
    lcd_gotoxy(1,0);
    printf(lcd_putc,"PC ??  ??  ? "); // ?? wil be cleard when value update

loop_mode3:
    f_req=fgetc();

    if(first_run_flag==1)
    {
        enable_interrupts(INT_PWM_TB);
        first_run_flag=0;
    }
    if(f_req==0) //stop button is pressed
    {
        RAM_DOWN_SPEED();
        first_run_flag=1;
        disable_interrupts(INT_PWM_TB);
    }

    if(f_req==70) //Change direction button is pressed
    {
        temp_int=f;          //save current f_out =
f_req !!!
        f_req=0;
        RAM_DOWN_SPEED();
    }
}
```

```
        direction_flag=direction_flag+1; //complement
direction_flag=direction_flag+1

        switch(direction_flag)           //change direction display
        {
            case 1:
                lcd_gotoxy(15,0);
                printf(lcd_putc,"F");
                break;
            case 0:
                lcd_gotoxy(15,0);
                printf(lcd_putc,"R");
                break;
        }
        f_req=temp_int;                  //restore f_req
        RAM_UP_SPEED();

    }

    RAM_UP_SPEED();                    //ram up speed at 1st RUN and then .....
    RAM_DOWN_SPEED();

    goto loop_mode3;

} //main
```

### 8.3> CODE PHẦN MỀM GIAO TIẾP NGƯỜI SỬ DỤNG:

Option Explicit

```
Dim Y           As Double           'variable in chart drawing
Dim Xx          As Double
Dim i           As Double
Dim dir_flag    As Integer
```

```
Dim strtemp     As String 'variable ONCOMM event
Dim strdata     As String
Dim datavu      As String
Dim j           As String
```

```
Dim intdigvu    As Integer
Dim digdata     As Integer
```

```
Private Sub Change_direction_button_Click()
```

```
If (dir_flag = 0) Then
    Text_direction = "FORWARD"
```

```
    dir_flag = 1
Else
    Text_direction = "REVERSE"
    dir_flag = 0

End If

    MSComm1.Output = Chr(70) 'send the request 70 as Change_direction_code to
PIC

End Sub

Private Sub RUN_SEND_button_Click()

j = txt_f_request.Text 'send the request value of frequency to PIC
If (j > 60) Then
    MsgBox ("Frequency must be in range from 0 to 60 Hz")
    txt_f_request = ""
    txt_f_request.SetFocus
Else
    MSComm1.Output = Chr(j)
End If

If (RUN_SEND_button.Caption = "RUN") Then

    RUN_SEND_button.Caption = "SEND" 'Change caption of RUN button
    RUN_SEND_button.BackColor = &H8000000F

    Text_motor_status = "RUNNING"
End If

End Sub
Private Sub STOP_button_Click()

MSComm1.Output = Chr(0) 'send the request value of frequency(=0) to PIC

If (RUN_SEND_button.Caption = "SEND") Then
    RUN_SEND_button.Caption = "RUN" 'Change caption
    RUN_SEND_button.BackColor = &HFF00&

    Text_motor_status = "STOP"
End If

End Sub

Private Sub Form_Load()

dir_flag = 1
```

```
'Dong Serial Port neu no mo
  If frmMain.MSComm1.PortOpen = True Then
    frmMain.MSComm1.PortOpen = False
  End If
'Cau hinh lai Serial Port
  frmMain.MSComm1.RThreshold = 1      'Khi nhan 1 ki tu don se phat sinh su
kien CommEvent
  frmMain.MSComm1.CommPort = 1      'Dung PORT1
  frmMain.MSComm1.InputLen = 0      'Doc toan bo buffer
  frmMain.MSComm1.Settings = "9600,n,8,1"
  frmMain.MSComm1.PortOpen = True   'Mo cong
'Form hien giua man hinh
  frmMain.Move (Screen.Width - frmMain.Width) / 2, (Screen.Height -
frmMain.Height) / 2
```

.....

```
' Chart SETTING

Strip1.CursorColor = RGB(255, 0, 0)
'Left = (Main.Width - Width) / 2
'Top = (Main.Height - Height) / 2

Xx = Now

For i = 0 To Strip1.Variables - 1
  Strip1.VariableID = i
  '.5 seconds
  Strip1.VariableDeltaX = 1 / 24 / 60 / 60 / 2 '.5 seconds interval
  Strip1.VariableLastX = Xx 'Set LastX to current time
Next

Strip1.XTicMode = 1 'Set X Mode to Date/Time Display
'30 seconds
Strip1.XSpan = 1 / 24 / 60 / 60 * 30 '30 seconds of display on plot

End Sub
```

```
Private Sub MSComm1_OnComm()

  With frmMain.MSComm1
    Select Case .CommEvent
      Case comEvReceive
        'Nhan du lieu tu vi dieu khien
        strtemp = .Input
        strdata = Left(strtemp, 1)
        datavu = Right(strtemp, 1)
    End Select
  End With
```

```
        digdata = Asc(strdata)
        intdigvu = Asc(datavu)

        'txtFreg = digdata
        'txt_f_out = intdigvu    'xuat du lieu ra o txt upload cua

        txt_f_out = digdata
        txt_u_out = digdata * 3.66 ' V/f=const
    End Select
End With
End Sub

Private Sub Timer1_Timer()

    Strip1.AddXY 0, Now, Y

    Y = digdata    'data will be printed in chart

End Sub

Private Sub Form_Unload(Cancel As Integer)
    MSComm1.Output = Chr(0)    'send stop signal for PIC to stop motor

    MSComm1.PortOpen = False    'Dong cong
End Sub
```