

**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
BA PHA THEO PHƯƠNG PHÁP SINPWM,
SỬ DỤNG VI ĐIỀU KHIỂN dsPIC30F6010**

**SVTH : LÊ TRUNG NAM
CBHD : TS. LÊ MINH PHƯƠNG
MSSV : 40201632
BỘ MÔN : ĐIỆN - ĐIỆN TỬ**

TP Hồ Chí Minh, 01/2007

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

Tp Hồ Chí Minh, tháng 1 năm 2007

Giáo viên hướng dẫn

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 dìu dắt tôi tận tình, 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 đặc biệt là thầy Lê Minh Phương, thầy Phan Quốc Dũng, 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 cảm ơn gia đình tôi, những người thân đã cho tôi những điều kiện tốt nhất để học tập trong suốt thời gian dài.

Ngoài ra tôi xin gửi lời cảm ơn đến bạn gái tôi(H.T.T), đến tất cả những người bạn của tôi, 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.

Tp. Hồ Chí Minh, tháng 1 năm 2007

MỤC LỤC

CHƯƠNG 1: GIỚI THIỆU VỀ ĐỘNG CƠ KHÔNG ĐỒNG BỘ	1
1.1. Tổng quan về máy điện không đồng bộ	2
1.1.1 Nguyên lý làm việc:.....	2
1.1.2 Cấu tạo	3
1.2 Ứng dụng của động cơ không đồng bộ.....	4
1.3 Khả năng dùng động cơ xoay chiều thay thế máy điện một chiều:	5
1.4 Kết luận:	6
CHƯƠNG 2: LÝ THUYẾT VÀ PHƯƠNG PHÁP ĐIỀU KHIỂN	7
2.1 Giới thiệu về biến tần nguồn áp điều khiển theo phương pháp V/f.....	8
2.2.1 Phương pháp E/f.....	8
2.2.2 Phương pháp V/f	8
2.3 Các phương pháp thông dụng trong điều khiển động cơ không đồng bộ:.....	10
2.3.1 Phương pháp điều rộng xung SINPWM.....	10
2.3.1.1 Các công thức tính toán.....	12
2.3.1.2 Cách thức điều khiển.....	13
2.3.1.3 Quy trình tính toán:.....	14
2.3.1.4 Hiệu quả của phương pháp điều khiển :.....	15
2.3.2 Phương pháp điều chế vector không gian (Space Vector):.....	17
2.3.2.1 Thành lập vector không gian:	17
2.3.2.2 Tính toán thời gian đóng ngắt:	20
2.3.2.3 Phân bố các trạng thái đóng ngắt:.....	22
2.3.2.4 Kỹ thuật thực hiện điều chế vector không gian:.....	22
2.3.2.5 Giảm đồ đóng ngắt các khóa để tạo ra Vector Vs trong từng sector:	22
CHƯƠNG 3 : CẤU TẠO VÀ CÁC THÔNG SỐ PHẦN CỨNG	25
3.1 Sơ đồ khối của mạch điều khiển động cơ:	27
3.2 Giới thiệu chi tiết các khối điều khiển:.....	27
3.2.1 Mạch lái	27
3.2.2 Mạch cách ly	31
3.2.3 Mạch MOSFETs.....	31
3.2.4 Mạch chỉnh lưu.....	33
3.2.4.1 Bộ chỉnh lưu:	33
3.2.4.2 Phương pháp chỉnh lưu :	33
CHƯƠNG 4 : SƠ ĐỒ CẤU TẠO MẠCH ĐIỀU KHIỂN	34
4.1 Sơ đồ mạch cách ly	35
4.2 Sơ đồ mạch lái.....	37
4.3 Sơ đồ mạch động lực	38
4.4 Sơ đồ mạch điều khiển	39
4.4.1 Khối điều khiển	39
4.4.2 Khối giao tiếp máy tính	40
4.4.3 Khối hiển thị.....	40
4.4.4 Khối nút bấm.....	41
CHƯƠNG 5: GIỚI THIỆU VỀ DSPIC 6010.....	42
5.1 Tổng quan về vi điều khiển dsPIC30F6010	43
5.2 Các đặc điểm đặc biệt ở họ MCU dsPic-6010:.....	44
5.3 Giới thiệu khái quát về cấu trúc phần cứng:	45

5.4 Khái quát về các thanh ghi làm việc	50
5.4.1 Các thanh ghi điều khiển :	50
5.4.2 Thanh ghi TRIS:	50
5.4.3 Thanh ghi PORT:	51
5.4.4 Thanh ghi LAT:	51
5.5 Giới thiệu về các module cơ bản.....	52
5.5.1 Module Timer :	52
5.5.1.1 Module Timer 1	52
5.5.1.2 Timer2/3 module:	54
5.5.1.3 Timer4/5 module :	57
5.5.2 Module AD:	59
5.5.2.1 Giải thích hoạt động.....	60
5.5.2.2 Quá trình hoạt động của module ADC được tóm tắt như các bước sau:	60
5.5.2.3 Các sự kiện kích chuyển đổi:.....	61
5.5.2.4 Tác động reset.....	61
5.5.2.5 Định dạng kiểu dữ liệu trong module A/D	61
5.5.3 Module PWM:	62
5.5.3.1 Các đặc điểm của module PWM	62
5.5.3.2 Giải thích hoạt động của module PWM	63
5.5.3.3 Các bộ đếm tỉ lệ trong module PWM:	67
5.5.3.4 Các thanh ghi làm việc trong module PWM	68
5.6 GIỚI THIỆU VỀ TẬP LỆNH CỦA MCU DSPIC-6010	70
CHƯƠNG 6: SƠ ĐỒ KHỐI VÀ GIẢI THUẬT ĐIỀU KHIỂN.....	75
6.1 Sơ đồ khối chương trình :	76
6.2 Sơ đồ giải thuật chương trình :	77
CHƯƠNG 7 : KẾT QUẢ ĐẠT ĐƯỢC	80
7.1 Phần cứng:	81
7.1.1 Mạch động lực:	81
7.1.2 Mạch điều khiển	82
7.2 Phần mềm:	83
7.3 Dạng sóng điện áp ngõ ra:	83
PHỤ LỤC.....	85
TÀI LIỆU THAM KHẢO.....	111
TÀI LIỆU THAM KHẢO TRONG NƯỚC.....	111
TÀI LIỆU THAM KHẢO NƯỚC NGOÀI.....	111
WEBSITE THAM KHẢO	111

DANH SÁCH HÌNH VẼ

Hình1.1: Nguyên lý hoạt động của động cơ.....	2
Hình1.2: Lá thép kỹ thuật điện	3
Hình 2.1: Quan hệ giữa moment và điện áp theo tần số	10
Hình 2.2: Nguyên lý của phương pháp điều rộng sin	11
Hình 2.3 : Sơ đồ dạng điện áp trên các pha.....	12
Hình 2.4: Quá trình hoạt động của bộ điều khiển.....	13
Hình 2.5: Sơ đồ kết nối các khóa trong bộ nghịch lưu	16
Hình 2.6 : Sơ đồ bộ biến tần nghịch lưu áp 6 khóa (MOSFETs hoặc IGBTs).....	17
Hình 2.7: Biểu diễn vector không gian trong hệ tọa độ x-y	17
Hình 2.8: Các vector không gian từ 1 đến 6.....	19
Hình 2.9: Trạng thái đóng-ngắt của các khóa.....	19
Hình 2.10: Vector không gian Vs trong vùng 1	20
Hình 2.11: Vector không gian Vs trong vùng bất kỳ	21
Hình 2.12: Giảm đồ đồng cắt linh kiện	22
Hình 2.13: Vector Vs trong các vùng từ 0-6	24
Hình 3.1: Sơ đồ khối mạch điều khiển.....	27
Hình 3.2: Ví dụ sơ đồ điều khiển mosfet	28
Hình 3.3: Sơ đồ khối của IC lái mosfet.....	29
Hình 3.4: IC IR2136	29
Hình 3.5: Sơ đồ kết nối IR2136.....	30
Hình 3.6: Sơ đồ khối của opto	31
Hình 3.7: Sơ đồ khối của MOSFET và IGBT	32
Hình 3.8: IRFP460P.....	33
Hình 4.1 : Sơ đồ mạch cách ly.....	36
Hình 4.2 : Sơ đồ mạch lái mosfet	37
Hình 4.3 : Sơ đồ mạch động lực	38
Hình 4.4 : Sơ đồ khối điều khiển chính	39
Hình 4.5 : Sơ đồ khối giao tiếp máy tính	40
Hình 4.6 : Sơ đồ khối hiển thị	40
Hình 4.7 : Sơ đồ khối nút bấm.....	41
Hình 5.1 : Các họ vi điều khiển PIC và dsPIC	43
Hình 5.2: Sơ đồ ứng dụng các họ vi điều khiển	43
Hình 5.3: Sơ đồ chân dsPIC30F6010.....	45
Hình 5.4: Sơ đồ tổ chức bên trong MCU dsPIC6010	46
Hình 5.5: Sơ đồ tổ chức bộ nhớ bên trong MCU dsPIC6010	49
Hình 5.6:Sơ đồ cấu tạo bên trong một I/O	50
Hình 5.7: Sơ đồ cấu tạo tổng quan của các I/O Port trong MCU	51
Hình 5.8: Sơ đồ cấu tạo của bộ 16-bit Timer1	53
Hình 5.9: Sơ đồ cấu tạo của bộ 32-bit Timer2/3	56
Hình 5.10: Sơ đồ cấu tạo của bộ 16-bit Timer2 (Timer loại B)	56
Hình 5.11: Sơ đồ cấu tạo của bộ 16-bit Timer3 (Timer loại C)	57
Hình 5.12: Sơ đồ cấu tạo của bộ 32-bit Timer4/5	58
Hình 5.13: Sơ đồ cấu tạo của bộ 16-bit Timer4 (Timer loại B)	58
Hình 5.14: Sơ đồ cấu tạo của bộ 16-bit Timer5 (Timer loại C)	59
Hình 5.15: Sơ đồ cấu tạo bên trong module A/D	60
Hình 5.16: Sơ đồ cấu tạo bên trong module PWM.....	63

Hình 5.17 : Cập nhật giá trị PWM trong chế độ tự do.....	64
Hình 5.18 : Cập nhật giá trị PWM trong chế độ đếm lên xuống	65
Hình 5.19 : Cập nhật giá trị PWM trong chế độ cập nhật kép	65
Hình 5.20: Tín hiệu PWM trong chế độ hoạt động hỗ trợ	66
Hình 5.21: Xung PWM dạng Edge Aligned.....	66
Hình 5.22: Xung PWM dạng Center Aligned	67
Hình 5.23: Bộ đếm tỉ lệ trong module PWM	67
Hình 7.1 : Mạch động lực.....	81
Hình 7.2: Mạch điều khiển.....	82
Hình 7.3: Giao diện giao tiếp máy tính.....	83
Hình 7.4: Dạng điện áp pha ngõ ra	83
Hình 7.5 : Dạng điện áp dây ngõ ra	84

DANH SÁCH BẢNG BIỂU

Bảng 2.1: Giá trị điện áp các trạng thái đóng ngắt và vector không gian tương ứng	20
Bảng 3.1: Thông số động cơ.....	26
Bảng 3.2 : Định nghĩa các chân trong IR2136	31
Bảng 5.1 : Thiết lập tần số hoạt động	44
Bảng 5.2: Mô tả chức năng, tính chất các I/O trong MCU	49
Bảng 5.3: Trình bày sơ đồ các thanh ghi điều khiển TIMER1	53
Bảng 5.4: Trình bày các thanh ghi điều khiển Timer2/3	55
Bảng 5.5: Trình bày các thanh ghi điều khiển Timer4/5	57
Bảng 5.6: Định dạng kiểu lưu trữ kết quả.....	62
Bảng 5.7: Bảng thanh ghi điều khiển module AD.....	62
Bảng 5.8 : Bảng thanh ghi điều khiển module PWM	69
Bảng 5.9: Bảng tập lệnh MCU 6010.....	74

DANH MỤC TỪ VIẾT TẮT, TÊN NƯỚC NGOÀI

ACIM	AC Induction Motor	Động cơ dùng nguồn xoay chiều
AD	Analog To Digital	Tuần tự sang số
ADC	Analog To Digital Conversion	Bộ chuyển đổi tuần tự sang số
ĐCKĐB		Động cơ không đồng bộ
Fcy		Tần số hoạt động trong vi điều khiển
Fosc		Tần số thạch anh
I/O	Input/Output	Ngõ vào, ngõ ra
MCU	Micro Controller Unit	Vi điều khiển
MODULE		Khối
PWM	Pulse Width Modulation	Điều rộng xung
SINPWM	Sin Pulse Width Modulation	Điều rộng xung sin
TIMER		Bộ định thì

TÓM TẮT LUẬN VĂN

MỤC ĐÍCH LUẬN VĂN:

- Tìm hiểu và thiết kế bộ biến tần truyền thống (6 khóa) ba pha điều khiển ĐCKĐB theo phương pháp V/f và điều chế SINPWM .
- Khảo sát nguyên tắc đóng cắt các khóa bán dẫn trong bộ nghịch lưu .
- Kiểm tra, đánh giá dạng sóng điện áp ngõ ra.
- Nguyên cứu giải thuật và viết chương trình điều khiển.

PHƯƠNG PHÁP NGHIÊN CỨU

- Tham khảo và tổng hợp tài liệu trong và ngoài nước.
- Tiến hành thực nghiệm trên mô hình thực tế.
- Theo dõi, đánh giá, nhận xét các thông số thực nghiệm.
- Xử lý số liệu, tính toán, và viết báo cáo.

THỜI GIAN THỰC HIỆN

Thời gian thực hiện luận văn: 3/9/2006 – 30/12/2006.

ĐỊA ĐIỂM THỰC HIỆN

Nghiên cứu này được thực hiện bằng các mô hình ở qui mô phòng thí nghiệm Điện tử công suất đặt tại trường Đại học Bách Khoa TP Hồ Chí Minh.

Ý NGHĨA KHOA HỌC VÀ THỰC TIỄN CỦA NGHIÊN CỨU

- Đề xuất mô hình biến tần điều khiển động cơ không đồng bộ ba pha dùng trong các hệ thống truyền động với giá thành thấp, đáp ứng được các yêu cầu cơ bản của thực tế.
- Do hạn chế về mặt thời gian, điều kiện kinh tế nên trong phạm vi luận văn tốt nghiệp này chỉ dừng lại ở điều khiển vòng hở động cơ không đồng bộ ba pha và hi vọng đề tài sẽ được tiếp tục phát triển trong tương lai .

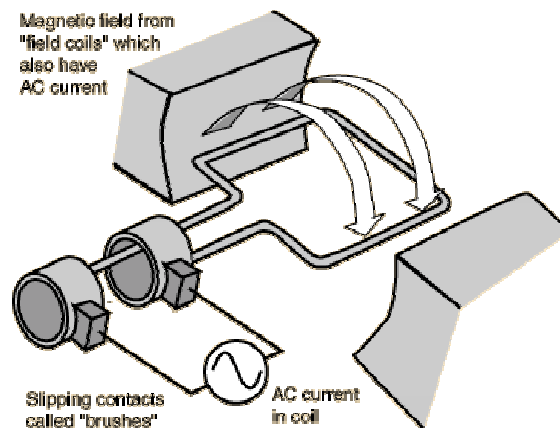
CHƯƠNG 1

GIỚI THIỆU VỀ ĐỘNG CƠ KHÔNG ĐỒNG BỘ

CHƯƠNG 1: GIỚI THIỆU VỀ ĐỘNG CƠ KHÔNG ĐỒNG BỘ

1.1. Tổng quan về máy điện không đồng bộ

1.1.1 Nguyên lý làm việc:



Hình 1.1: Nguyên lý hoạt động của động cơ

Khi nam châm điện quay (tốc độ n_1 vòng/ phút) làm đường sức từ quay cắt qua các cạnh của khung dây cảm ứng gây nên sức điện động E trên khung dây. Sức điện động E sinh ra dòng điện I chạy trong khung dây. Vì dòng điện I nằm trong từ trường nên khi từ trường quay làm tác động lên khung dây một lực điện từ F . Lực điện từ này làm khung dây chuyển động với tốc độ n vòng/ phút.

Vì $n < n_1$ nên gọi là không đồng bộ.

ĐCKĐB ba pha có dây quấn ba pha phía stator, Roto của ĐCKĐB là một bộ dây quấn ba pha có cùng số cực trên lõi thép của Roto.

Khi Stator được cung cấp bởi nguồn ba pha cân bằng với tần số f , từ trường quay với tốc độ ω_{db} sẽ được tạo ra. Quan hệ giữa từ trường quay và tần số f của nguồn ba pha là :

$$\omega_{db} = \frac{2\pi f}{p} = \frac{\omega_1}{p} \text{ (rad / s)}$$

Trong đó :

p - số đôi cực

ω_1 - tần số góc của nguồn ba pha cung cấp cho động cơ: $\omega_1 = 2\pi f$

Nếu tốc độ quay của roto là ω , độ sai lệch giữa tốc độ từ trường quay stator và roto là:

$$\omega_{sl} = \omega_{db} - \omega = s \cdot \omega_{db}$$

Trong đó ω_{sl} gọi là tốc độ trượt

Thông số s gọi là độ trượt, ta có:

$$s = \frac{\omega_{db} - \omega}{\omega_{db}}$$

Vì có tốc độ tương đối giữa roto và từ trường quay stator , điện áp cảm ứng ba pha sẽ được sinh ra trong roto .Tần số của điện áp này sẽ tỉ lệ với độ trượt theo công thức :

$$\omega_r = s * \omega_1 (\text{rad} / \text{s})$$

Moment động cơ sinh ra:

$$M = -\frac{\pi}{2} p^2 \phi_m F_m \sin \delta_r$$

Trong đó :

ϕ_m : từ thông trên một cực (Wb).

F_m : giá trị đỉnh của sức từ động roto.

δ_r : góc lệch pha giữa sức từ động roto và sức từ động khe hở không khí.

1.1.2 Cấu tạo

a) Phần tĩnh (Stator)

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. Khi đường kính ngoài lõi sắt nhỏ hơn 90 mm thì dùng cả tấm tròn ép lại. Khi đường kính ngoài lớn hơn thì dùng những tấm hình rẻ quạt (hình 2) ghép lại.



Hình 1.2: Lá thép kỹ thuật điện

*** Dây quấn**

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

b) Phần quay (roto)

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

Rotor 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 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.

c) 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. 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 đời sống hàng ngày, động cơ không đồng bộ ngày càng chiếm một vị trí quan trọng với nhiều ứng dụng trong công nghiệp, nông nghiệp và trong đời sống hàng ngày.

1.2 Ứng dụng của động cơ không đồng bộ

Ngày nay, các hệ thống truyền động điện được sử dụng rất rộng rãi trong các thiết bị hoặc dây chuyền sản xuất công nghiệp, trong giao thông vận tải, trong các thiết bị điện dân dụng, ... Ước tính có khoảng 50% điện năng sản xuất ra được tiêu thụ bởi các hệ thống truyền động điện.

Hệ truyền động điện có thể hoạt động với tốc độ không đổi hoặc với tốc độ thay đổi được. Hiện nay khoảng 75 – 80% các hệ truyền động là loại hoạt động với tốc độ không đổi. Với các hệ thống này, tốc độ của động cơ hầu như không cần điều khiển trừ các quá trình khởi động và hãm. Phần còn lại, là các hệ thống có thể điều chỉnh được tốc độ để phối hợp đặc tính động cơ và đặc tính tải theo yêu cầu. Với sự phát triển mạnh mẽ của kỹ thuật bán dẫn công suất lớn và kỹ thuật vi xử lý, các hệ điều tốc sử dụng kỹ thuật điện tử ngày càng được sử dụng rộng rãi và là công cụ không thể thiếu trong quá trình tự động hóa.

Động cơ không đồng bộ có nhiều ưu điểm như: kết cấu đơn giản, làm việc chắc chắn, hiệu suất cao, giá thành hạ, có khả năng làm việc trong môi trường độc hại hoặc nơi có khả năng cháy nổ cao. Vì những ưu điểm này nên động cơ không đồng bộ được ứng dụng rất rộng

rải trong các ngành kinh tế quốc dân với công suất từ vài chục đến hàng nghìn kW. 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.

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.

Cho nên việc tách riêng điều khiển giữa moment và từ thông để có thể điều khiển độc lập đòi hỏi một hệ thống có thể tính toán cực nhanh và chính xác trong việc qui đổi các giá trị xoay chiều về các biến đơn giản . Vì vậy, cho đến gần đây, phần lớn động cơ xoay chiều làm việc với các ứng dụng có tốc độ không đổi do các phương pháp điều khiển trước đây dùng cho máy điện thường đắt và có hiệu suất kém. Động cơ không đồng bộ cũng không tránh khỏi nhược điểm này.

1.3 Khả năng dùng động cơ xoay chiều thay thế máy điện một chiều:

Những khó khăn trong việc ứng dụng động cơ xoay chiều chính là làm thế nào để có thể dễ dàng điều khiển được tốc độ của nó như việc điều khiển của động cơ DC. Vì vậy, một ý tưởng về việc biến đổi một máy điện xoay chiều thành một máy điện một chiều trên phương diện điều khiển đã ra đời. Đây chính là điều khiển vector. Điều khiển vector sẽ cho phép điều khiển từ thông và moment hoàn toàn độc lập với nhau thông qua điều khiển giá trị tức thời của dòng (động cơ tiếp dòng) hoặc giá trị tức thời của áp (động cơ tiếp áp).

Điều khiển vector cho phép tạo ra những phản ứng nhanh và chính xác của cả từ thông và moment trong cả quá trình quá độ cũng như quá trình xác lập của máy điện xoay chiều giống như máy điện một chiều. Cùng với sự phát triển của kỹ thuật bán dẫn và những bộ vi xử lý có tốc độ nhanh và giá thành hạ, việc ứng dụng của điều khiển vector ngày càng được sử dụng rộng rãi trong nhiều hệ truyền động và đã trở thành một tiêu chuẩn công nghiệp.

Với sự phát triển nhanh chóng, ngành công nghiệp tự động luôn đòi hỏi sự cải tiến thường xuyên của các loại hệ truyền động khác nhau. Những yêu cầu cải tiến cốt yếu là tăng độ tin cậy, giảm khả năng tiêu thụ điện năng, giảm thiểu chi phí bảo dưỡng, tăng độ chính xác và tăng khả năng điều khiển phức tạp. Vì vậy, những hệ truyền động với động cơ điện một chiều đang dần thay thế bởi những hệ truyền động động cơ xoay chiều sử dụng điều khiển vector. Bởi vì, lý do chính để sử dụng rộng rãi động cơ điện một chiều trước kia là khả năng điều khiển độc lập từ thông và moment lực đã nêu cũng như cấu trúc hệ truyền động khá đơn giản. Tuy nhiên, chi phí mua và bảo trì động cơ cao, đặc biệt khi số lượng máy điện phải dùng lớn. Trong khi đó, các ứng dụng thực tế của lý thuyết điều khiển vector đã được thực hiện từ những năm 70 với các mạch điều khiển liên tục. Nhưng các mạch liên tục không thể đáp ứng

được sự đòi hỏi phải chuyển đổi tức thời của hệ quy chiếu quay do điều này đòi hỏi một khối lượng tính toán trong một thời gian ngắn.

Sự phát triển của những mạch vi xử lý đã làm thay đổi việc ứng dụng của lý thuyết điều khiển vector. Khả năng tối ưu trong điều khiển quá độ của điều khiển vector là nền móng cho sự phát triển rộng rãi của các hệ truyền động xoay chiều (vì giá thành của động cơ xoay chiều rất rẻ hơn so với động cơ một chiều).

Ngoài những phát triển trong điều khiển vector, một sự phát triển đáng chú ý khác chính là việc ứng dụng mạng neural (neural networks) và logic mờ (fuzzy logic) vào điều khiển vector đang là những đề tài nghiên cứu mới trong nghiên cứu hệ truyền động. Hai kỹ thuật điều khiển mới này sẽ tạo nên những cải tiến vượt bậc cho hệ truyền động của máy điện xoay chiều trong một tương lai gần. Triển vọng ứng dụng rộng rãi của hai kỹ thuật này phụ thuộc vào sự phát triển của bộ vi xử lý bán dẫn (semiconductor microprocessor).

Với sự phát triển mạnh của các bộ biến đổi điện tử công suất, một lý thuyết điều khiển máy điện xoay chiều khác hẳn với điều khiển vector đã ra đời. Lý thuyết điều khiển trực tiếp moment lực (Direct Torque Control hay viết tắt là DTC) do giáo sư Noguchi Takahashi đưa ra vào cuối năm 80. Tuy nhiên, kỹ thuật điều khiển moment trực tiếp vẫn chưa phải hoàn hảo và cần phải nghiên cứu thêm.

1.4 Kết luận:

Với sự phát triển mạnh mẽ của kỹ thuật bán dẫn công suất cao và kỹ thuật vi xử lý, hiện nay các bộ điều khiển ĐCKĐB đã được chế tạo với đáp ứng tốt hơn, giá thành rẻ hơn các bộ điều khiển động cơ DC. Do đó, ĐCKĐB có thể thay thế được động cơ DC trong rất nhiều ứng dụng. Dự kiến trong tương lai gần, ĐCKĐB sẽ được sử dụng rộng rãi trên hầu hết các bộ truyền động điều khiển tốc độ.

CHƯƠNG 2

LÝ THUYẾT VÀ PHƯƠNG PHÁP ĐIỀU KHIỂN

CHƯƠNG 2: LÝ THUYẾT VÀ PHƯƠNG PHÁP ĐIỀU KHIỂN**2.1 Giới thiệu về biến tần nguồn áp điều khiển theo phương pháp V/f**

Được sử dụng hầu hết trong các biến tần hiện nay. Tốc độ của ĐCKĐB 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 sẽ làm giảm khả năng mang tải của động cơ.

Vì vậy, khi giảm tần số nguồn cung cấp cho động cơ nhỏ hơn tần số định mức thường đòi hỏi phải giảm điện áp V cung cấp cho động cơ sao cho từ thông trong khe hở không khí được giữ không đổi. Khi động cơ làm việc với tần số cung cấp lớn hơn tần số định mức, thường giữ điện áp cung cấp không đổi và bằng định mức, do giới hạn về cách điện stator hoặc điện áp nguồn.

2.2 Phương pháp điều khiển V/f**2.2.1 Phương pháp E/f**

Ta có công thức sau:

$$a = \frac{f}{f_{dm}}$$

Với f - tần số làm việc 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:

$$I_m = \frac{E_{dm}}{f_{dm}} \cdot \frac{1}{2\pi L_m}$$

Với L_m là điện cảm mạch từ hóa.

Tại tần số làm việc f :

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

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

$$\frac{E}{a} = E_{dm} \Rightarrow \frac{E}{f} = \frac{E_{dm}}{f_{dm}} = \text{const}$$

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

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 $E/f = \text{const}$ được thay bằng phương pháp $V/f = \text{const}$.

CHƯƠNG 2: LÝ THUYẾT VÀ PHƯƠNG PHÁP ĐIỀU KHIỂN

Trong phương pháp $V/f=\text{const}$ (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. Cần lưu ý là khi moment tải tăng, dòng động cơ tăng làm gia tăng sụt áp trên điện trở Stator dẫn đến E giảm, có nghĩa là từ thông động cơ giảm. Do đó động cơ không hoàn toàn làm việc ở chế độ từ thông không đổi.

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

$$M = \frac{3}{\omega_{db}} \cdot \left[\frac{V_{dm}^2 \cdot \frac{R'_2}{s}}{\left(R_1 + \frac{R'_2}{s}\right)^2 + (X_1 + X'_2)^2} \right]$$

Và moment cực đại ở chế độ định mức:

$$M_{\max} = \frac{3}{2 \cdot \omega_{db}} \cdot \left[\frac{V_{dm}^2}{R_1 \pm \sqrt{R_1^2 + (X_1 + X'_2)^2}} \right]$$

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

$$M = \frac{3}{\omega_{db}} \cdot \left[\frac{V_{dm}^2 \cdot \frac{R'_2}{a \cdot s}}{\left(\frac{R_1}{a} + \frac{R'_2}{as}\right)^2 + (X_1 + X'_2)^2} \right], a < 1$$

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

$$M_{\max} = \frac{3}{2 \cdot \omega_{db}} \cdot \left[\frac{V_{dm}^2}{\frac{R_1}{a} \pm \sqrt{\left(\frac{R_1}{a}\right)^2 + (X_1 + X'_2)^2}} \right], a < 1$$

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_{dm} tại $f=f_{dm}$.

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

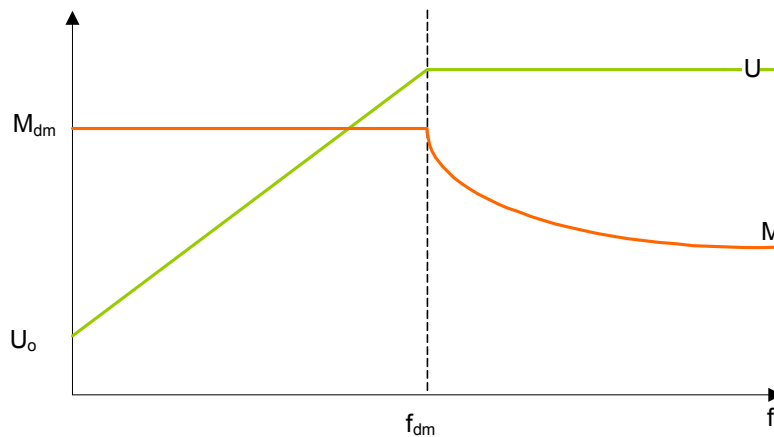
Khi đó moment và moment cực đại của động cơ tại tần số f cung cấp sẽ là:

$$M = \frac{3}{\omega_{db}} \cdot \left[\frac{V_{dm}^2 \cdot \frac{R_2'}{a \cdot s}}{\left(R_1 + \frac{R_2'}{s} \right)^2 + a^2 (X_1 + X_2')^2} \right], a > 1.$$

Và moment cực đại ở tần số f :

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

Sau đây là đồ thị biểu diễn mối quan hệ giữa moment và điện áp theo tần số trong phương pháp điều khiển $V/f = \text{const}$.



Hình 2.1: Quan hệ giữa moment và điện áp theo tần số

2.3 Các phương pháp thông dụng trong điều khiển động cơ không đồng bộ:

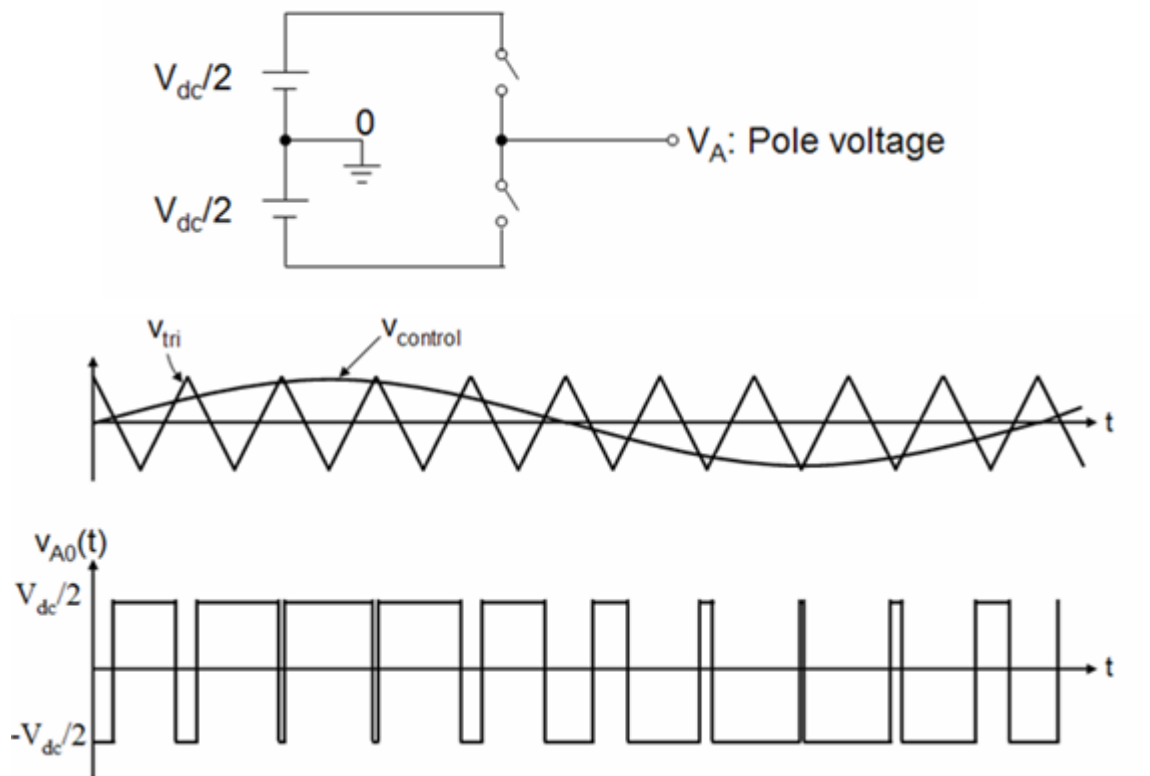
Có nhiều phương pháp để điều khiển bộ nghịch lưu áp để tạo ra điện áp có biên độ và tần số mong muốn cung cấp cho động cơ. Trong nội dung này chúng ta khái quát hai phương pháp đó là:

Phương pháp điều rộng xung (SinPWM).

Phương pháp điều chế vector không gian (Space Vector).

2.3.1 Phương pháp điều rộng xung SINEPWM

Để tạo ra một điện áp xoay chiều bằng phương pháp SINEPWM, 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 sin một pha:

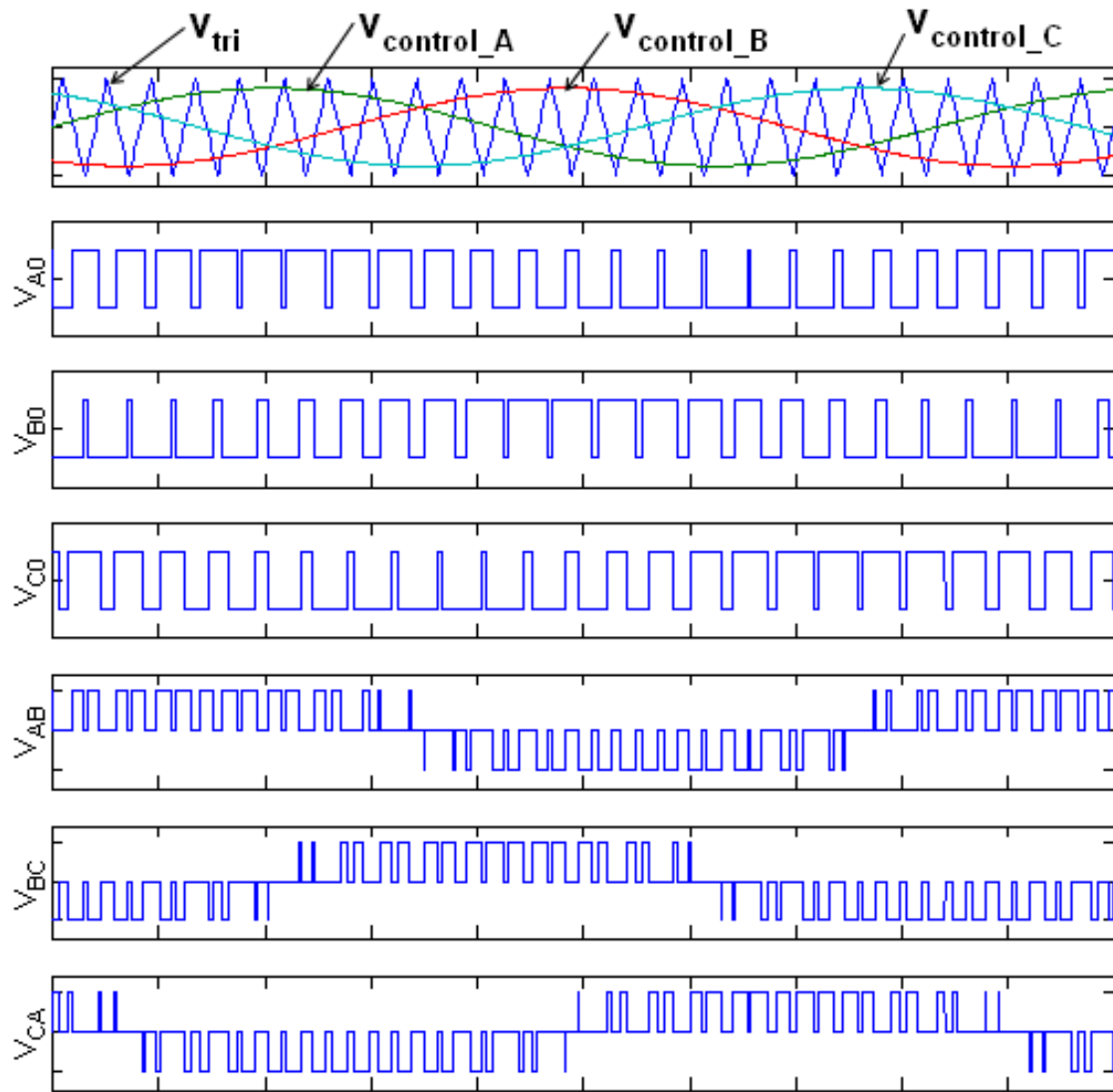


Hình 2.2: Nguyên lý của phương pháp điều rộng xung

Khi: $V_{control} > V_{tri}$ thì $V_{AO} = \frac{V_{dc}}{2}$

$V_{control} < V_{tri}$ 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 : Sơ đồ dạng điện áp trên các pha

2.3.1.1 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:

$$U_1 = ma \cdot \frac{U_{DC}}{2} \quad (1)$$

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

$$ma = \frac{U_{dk}}{U_{carry}} \quad (2)$$

2.3.1.2 Cách thức điều khiển

Sau khi đã nói về phương pháp điều khiển $V/f=\text{const}$ và phương pháp điều khiển bộ nghịch lưu áp theo phương pháp điều rộng xung SINPWM, ta có thể đưa ra một thuật toán điều khiển động cơ theo một tần số đặt cho trước như sau.

Do động cơ được điều khiển vòng hở nên không thể đo đạc được tốc độ thực của động cơ, nên ta hiểu tần số đặt ở đây là tần số nguồn sin điều rộng xung cấp cho động cơ.

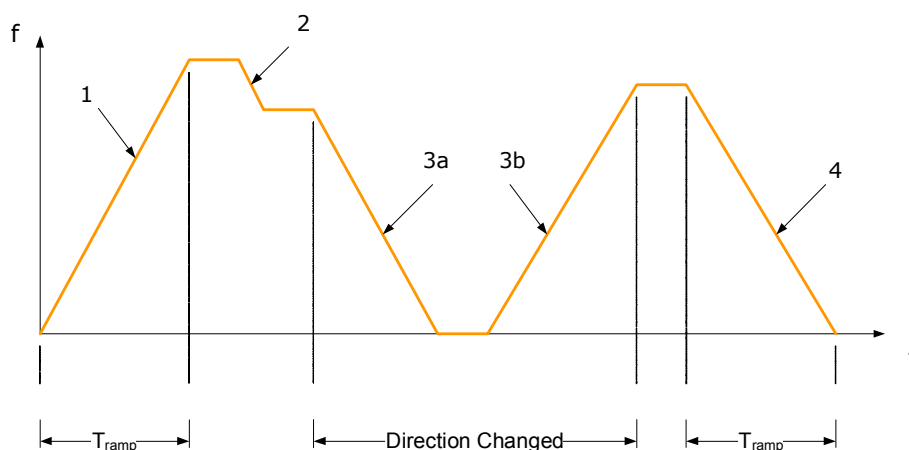
Trong trường hợp ta muốn cho động cơ đang ở trạng thái đứng yên chuyển sang chạy ở tần số đặt thì phải thông qua một quá trình khởi động mềm tránh cho động cơ khởi động lập tức đến tốc độ đặt, gây ra dòng điện khởi động lớn làm hỏng động cơ. Tần số nguồn cung cấp sẽ tăng từ giá trị 0 (đứng yên) đến giá trị đặt (tương ứng với biên độ tăng từ V_0 đến $V_f=V_0+K.f_{\text{req}}$). Thời gian khởi động này có thay đổi theo công suất của từng động cơ. Đối với động cơ công suất lớn thì thời gian khởi động lâu hơn so với động cơ công suất nhỏ. Thời gian khởi động của động cơ thông thường được chọn từ 5 đến 10 giây.

Sau khi tần số nguồn đã đạt đến giá trị yêu cầu lúc đầu thì sẽ giữ nguyên giá trị đó. Trong quá trình động cơ đang chạy ổn định mà có một nhu cầu thay đổi tần số thì cũng có một quá trình chuyển tần số từng bước thay vì nhảy ngay lập tức đến giá trị tần số yêu cầu mới.

Khi muốn thay đổi chiều của động cơ cần phải đưa động cơ về tần số đủ nhỏ rồi sau đó mới thực hiện việc đổi chiều quay (thay đổi thứ tự pha nguồn cấp cho động cơ) – tránh hiện tượng moment xoắn có thể làm gãy trục động cơ và tăng dòng đột ngột.

Khi muốn dừng động cơ thì phải hạ tần số từ giá trị hiện tại về giá trị 0. Thời gian hãm này phụ thuộc vào quán tính quay của động cơ. Khi muốn hãm nhanh có thể dùng các phương pháp hãm như phương pháp hãm động năng (Dynamic Breaking) – có dùng điện trở thắng.

Như vậy có thể hình dung quá trình hoạt động của bộ điều khiển như sau:



Hình 2.4: Quá trình hoạt động của bộ điều khiển

Đoạn 1 ứng với khởi động động cơ – tần số tăng từ 0 đến giá trị đặt sau khoảng thời gian khởi động (T_{ramp}).

Đoạn 2 ứng với việc thay đổi tần số khi động cơ đang chạy ổn định.

Đoạn 3 ứng với việc đổi chiều động cơ – được chia làm hai giai đoạn. Đoạn 3a ứng với giảm tần số về 0. Cuối đoạn 3a sẽ tiến hành đảo thứ tự pha nguồn cung cấp cho động cơ. Đoạn 3b ứng với tăng tần số lên đến giá trị mới (Có thay đổi tần số đặt trong lúc đổi chiều nên giá trị tần số sau khi đổi chiều không bằng giá trị cũ).

Đoạn 4 ứng với ngừng động cơ. Tần số cấp cho động cơ được giảm dần từ giá trị đặt về 0 sau khoảng thời gian dừng (T_{ramp}).

2.3.1.3 Quy trình tính toán:

Tần số sóng mang trong MCU 6010 được tạo ra theo công thức sau:

$$PTPER = \frac{f_{cy}}{2f_{pwm}} - 1$$

Trong đó PTPER là giá trị cần nạp vào thanh ghi PTPER để có được tần số sóng mang mong muốn

$$f_{cy} = \frac{f_{osc}}{4}$$

với f_{osc} là tần số của thạch anh đưa vào vi điều khiển

Trong phần này với tần số thạch anh đưa vào vi điều khiển là 10MHz, cộng với sử dụng chế độ nhân tần số PLL=8, ta có tần số thực đưa vào vi điều khiển là 20MHz, thời gian tính toán của một chu kì lệnh là 0.05 micro giây

Ứng với các giá trị của tần số tính toán trên, để tạo ra một sóng mang có tần số là 5Khz, giá trị cần nạp vào thanh ghi PTPER là 1999.

Sóng điều khiển (U_{dk}) được tạo ra bằng cách lập một bảng sin các giá trị từ 0 đến 2π tương ứng cho một chu kì của sóng điều khiển dạng Sin. Theo đã biết, sóng điều khiển mang thông tin về độ lớn trị hiệu dụng và tần số sóng hài cơ bản của điện áp ngõ ra, vì vậy khi biên độ và tần số của sóng điều khiển thay đổi thì ta có biên độ và tần số của điện áp ngõ ra cũng thay đổi theo.

Tần số của sóng điều khiển thay đổi tùy thuộc vào tần số di chuyển của con trở trong bảng sin. Nếu tần số của sóng điều khiển càng lớn thì số bước nhảy của con trở di chuyển trong bảng sin trong một chu kì sóng điều khiển càng ít và ngược lại. Quan hệ giữa số bước nhảy của con trở trong bảng sin và tần số của sóng điều khiển được xác định theo công thức sau:

$$K = \frac{\alpha}{\alpha_{min}}$$

Trong đó α_{min} là độ phân giải của bảng sin (với bảng sin gồm 720 giá trị thì độ phân giải của bảng sin là 0.5 độ/giá trị)

$$\alpha = \frac{T_{pwm} * 360}{T_{Udk}}$$

là góc nhảy của con trở trong bảng sin sau một chu kì PWM

Từ công thức (1),(2) ta có :

$$U_t = \frac{U_{dk}}{U_{carry}} * \frac{U_{dc}}{2} \quad (3)$$

Trong đó $U_{dc} = 440V$ ứng với trường hợp tỉ số điều chế $ma=1$ và động cơ hoạt động ở chế độ định mức.

Khi động cơ hoạt động ở chế độ định mức ta có:

$$\frac{U_{dm}}{f_{dm}} = 3.667 \quad (U_{dm} = 220V, f_{dm} = 60Hz) \quad (4)$$

Từ (3),(4) ta có:

$$f_{req} = 0.03 * U_{dk}$$

Giá trị của tần số đặt vào động cơ từ biến trở được thông qua bộ chuyển đổi ADC 10 bits theo công thức :

$$f_{req} = \text{ADC_Result} * 60 / 1024$$

Dựa vào tần số yêu cầu đầu vào ta có thể tính được biên độ của sóng điều khiển để giữ cho tỉ số V/f bằng hằng số.

Thời gian tăng tốc và giảm tốc của động cơ được tính toán dựa vào chu kì PWM, kể từ khi có sự thay đổi tần số đặt, sau mỗi chu kì PWM, giá trị tần số hiện tại sẽ cộng thêm vào hoặc trừ ra một giá trị cho đến khi nào bằng với giá trị của tần số đặt mới. Giá trị cộng vào hoặc trừ ra được tính toán theo công thức sau:

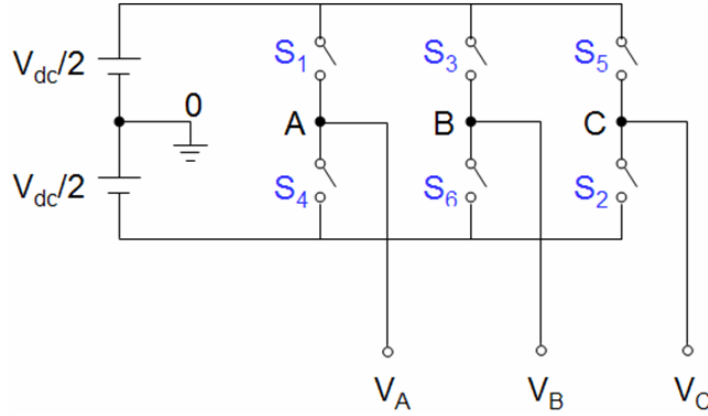
$$\Delta = (60 * T_{PWM}) / t$$

Trong đó $t(s)$ là thời gian tăng tốc hoặc giảm tốc của động cơ

Để đảm bảo sự chuyển mạch diễn ra đúng, tại mỗi thời điểm trên một nhánh chỉ có một khoá bán dẫn trong trạng thái dẫn, một khoảng thời gian nghỉ (dead time) cần được thêm vào khoảng giữa hai khoá, với tần số thạch anh đưa vào vi điều khiển là 10Mhz, tần số sóng mang là 5Khz, khoảng thời gian nghỉ được phép từ 1 đến 25 micro giây, ở đây khoảng thời gian nghỉ được chọn là 2 micro giây.

2.3.1.4 Hiệu quả của phương pháp điều khiển :

Đối với phương pháp điều chế SINPWM, tại mỗi thời điểm mà một trong hai khoá trên cùng một nhánh ở trạng thái ON thì biểu thức điện áp giữa mỗi pha và điểm trung tính ảo (O) có dạng như sau:



Hình 2.5: Sơ đồ kết nối các khóa trong bộ nghịch lưu

$$V_{AO} = \frac{V_{DC}}{2} * (m * \sin(\theta))$$

$$V_{BO} = \frac{V_{DC}}{2} * (m * \sin(\theta + \frac{2\pi}{3}))$$

$$V_{CO} = \frac{V_{DC}}{2} * (m * \sin(\theta + \frac{4\pi}{3}))$$

Điện áp giữa hai pha được tính toán như sau:

$$V_{AB} = V_{AO} - V_{BO} = \frac{\sqrt{3}}{2} * V_{DC} * m * \sin(\theta + \frac{\pi}{6})$$

$$V_{BC} = \frac{\sqrt{3}}{2} * V_{DC} * m * \sin(\theta + \frac{5\pi}{6})$$

$$V_{AC} = \frac{\sqrt{3}}{2} * V_{DC} * m * \sin(\theta + \frac{3\pi}{2})$$

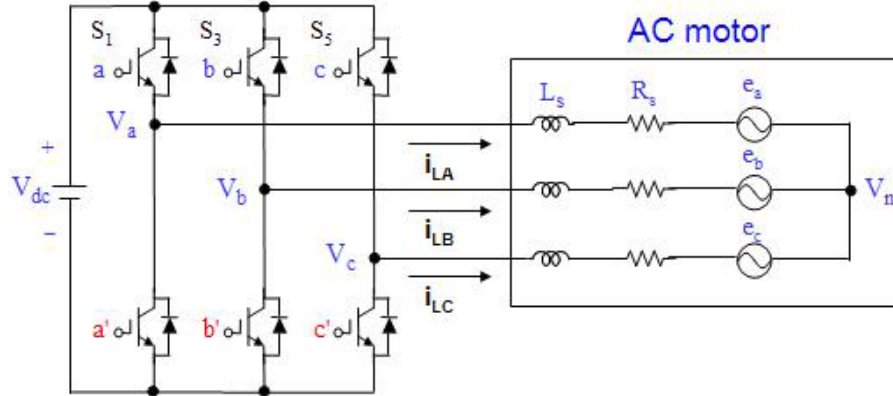
Từ công thức trên ta thấy giá trị điện áp lớn nhất giữa hai pha đạt được trong vùng tuyến tính khi $m=1$

$$\text{Giá trị điện áp lớn nhất là } V_{line_to_line_max} = \frac{\sqrt{3}}{2} * V_{DC}$$

Vậy đối với phương pháp này, điện áp do bộ chỉnh lưu cung cấp chỉ được sử dụng tối đa là 86.67% trong vùng điều khiển tuyến tính.

2.3.2 Phương pháp điều chế vector không gian (Space Vector):

Phương pháp điều chế xung vector không gian (SVM - Space Vector Modulation) khác với các phương pháp điều chế xung khác (PWM - Pulse Width Modulation). Với các phương pháp PWM khác, bộ nghịch lưu được xem như là ba bộ biến đổi kéo-dẩy riêng biệt với ba điện áp pha độc lập với nhau.



Hình 2.6 : Sơ đồ bộ biến tần nghịch lưu áp 6 khóa (MOSFETs hoặc IGBTs)

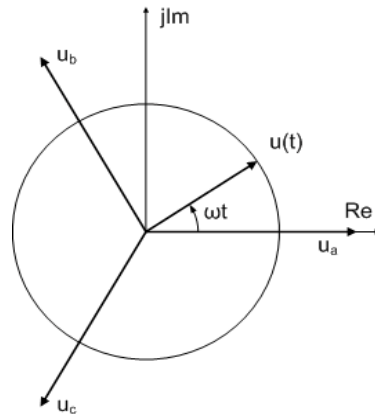
Đối với phương pháp điều chế xung vector không gian, bộ nghịch lưu được xem như là một khối duy nhất với 8 trạng thái đóng ngắt riêng biệt từ 0 đến 7.

2.3.2.1 Thành lập vector không gian:

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

$$u_a(t) + u_b(t) + u_c(t) = 0 \quad (2.1)$$

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° và vector $[0 \ 0 \ u_c]^T$ lệch một góc 240° so với trục x như hình sau đây.



Hình 2.7: 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:

$$u(t) = \frac{2}{3} (u_a + u_b \cdot e^{j(2/3)\pi} + u_c \cdot e^{-j(2/3)\pi}) \quad (2.2)$$

Trong đó $2/3$ là hệ số biến hình. Phân tích $u(t)$ trong phương trình trên thành phần thực và phần ảo.

$$u(t) = u_x + ju_y \quad (2.3)$$

Ta xây dựng được công thức chuyển đổi hệ tọa độ từ ba pha abc sang hệ tọa độ phức x-y bằng cách cân bằng phần thực và phần ảo trong phương trình (2.2) ta có :

$$\begin{aligned} u(t) &= \frac{2}{3} [u_a + u_b (\cos(2\pi/3) + j \sin(2\pi/3)) + u_c (\cos(-2\pi/3) + j \sin(-2\pi/3))] \\ &\Rightarrow \begin{cases} u_x = \frac{2}{3} [u_a + u_b \cos(2\pi/3) + u_c \cos(-2\pi/3)] \\ u_y = \frac{2}{3} [u_b \sin(2\pi/3) - u_c \sin(-2\pi/3)] \end{cases} \\ &\Rightarrow \begin{pmatrix} u_x \\ u_y \end{pmatrix} = \frac{2}{3} \begin{pmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{pmatrix} \begin{pmatrix} u_a \\ u_b \\ u_c \end{pmatrix} \end{aligned} \quad (2.4)$$

Tiếp theo hình thành tọa độ quay α - β bằng cách cho hệ tọa độ x-y quay với vận tốc góc ωt . Ta có công thức chuyển đổi hệ tọa độ như sau

$$\begin{pmatrix} u_\alpha \\ u_\beta \end{pmatrix} = \begin{pmatrix} \cos(\omega t) & \cos\left(\frac{\pi}{2} + \omega t\right) \\ \sin(\omega t) & \sin\left(\frac{\pi}{2} + \omega t\right) \end{pmatrix} \begin{pmatrix} u_x \\ u_y \end{pmatrix} = \begin{pmatrix} \cos(\omega t) & -\sin(\omega t) \\ \sin(\omega t) & \cos(\omega t) \end{pmatrix} \begin{pmatrix} u_x \\ u_y \end{pmatrix} \quad (2.5)$$

Nguồn áp ba pha tạo ra là cân bằng và sin nên ta có thể viết lại phương trình điện áp pha như sau:

$$\begin{aligned} u_a &= V_m \sin(\omega t) \\ u_b &= V_m \sin(\omega t - 2\pi/3) \\ u_c &= V_m \sin(\omega t + 2\pi/3) \end{aligned} \quad (2.6)$$

Từ phương trình (2.5) ta xây dựng được phương trình sau:

$$u(t) = V_r e^{j\theta} = V_r e^{j\omega t} \quad (2.7)$$

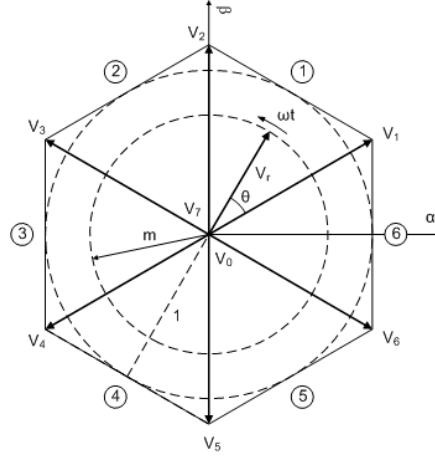
Thể hiện vector không gian có biên độ V_r quay với vận tốc góc ωt quanh gốc tọa độ 0. Phương trình điện áp dây như sau theo phương trình (2.4) như sau:

$$\begin{pmatrix} V_{L\alpha} \\ V_{L\beta} \end{pmatrix} = \frac{2}{3} \sqrt{\frac{3}{2}} V_s \begin{pmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{pmatrix} \begin{pmatrix} q_1 \\ q_3 \\ q_5 \end{pmatrix} \quad (2.8)$$

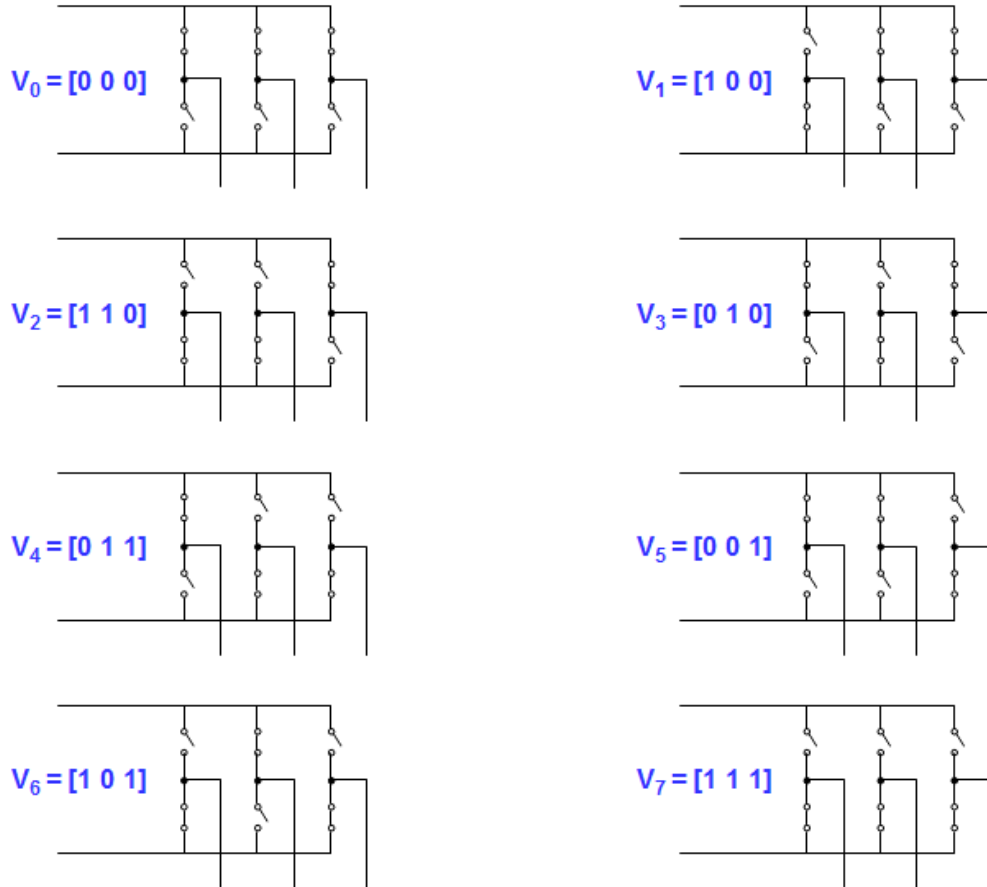
Trong đó $\sqrt{2}$ để chuyển từ giá trị biên độ thành giá trị hiệu dụng, $\sqrt{3}$ để chuyển giá trị điện áp pha thành điện áp dây. Vector điện áp dây sẽ sớm pha hơn vector điện áp pha một góc $\pi/6$. Nếu lồng ghép các trạng thái có thể có của q_1 , q_3 và q_5 vào phương trình (2.8), ta thu được phương trình điện áp dây (trị biên độ) theo các trạng thái của các khóa.

$$\vec{V}_n = \frac{\sqrt{2} \times \sqrt{2}}{\sqrt{3}} e^{j(2n-1)\pi/6} = \frac{2}{\sqrt{3}} \left[\cos\left(\frac{(2n-1)\pi}{6}\right) + j \sin\left(\frac{(2n-1)\pi}{6}\right) \right] \quad (2.9)$$

Với $n = 0, 1, 2..6$, ta thành lập được 6 vector không gian $V_1 - V_6$ và 2 vector 0 là V_0 và V_7 như hình sau



Hình 2.8: Các vector không gian từ 1 đến 6



Hình 2.9: Trạng thái đóng-ngắt của các khóa

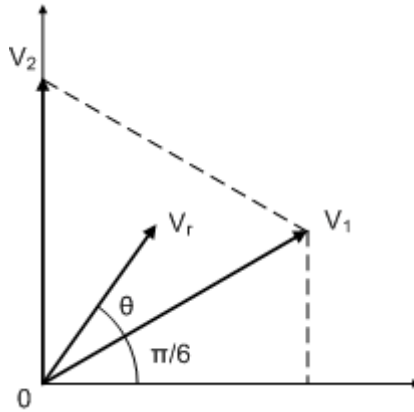
Bảng 2.1: Giá trị điện áp các trạng thái đóng ngắt và vector không gian tương ứng

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

Ghi chú: độ lớn điện áp phải nhân với V_{dc}

2.3.2.2 Tính toán thời gian đóng ngắt:

Xét trường hợp vector V_r nằm trong vùng 1 như hình sau



Hình 2.10: Vector không gian V_s trong vùng 1

Giả sử tần số điều rộng xung f_{PWM} đủ cao để trong suốt chu kỳ điều rộng xung T_s, vector V_s không thay đổi vị trí. Nhờ đó, ta có thể phân tích V_s theo các vector V₁, V₂, và V₀ hoặc V₇ như phương trình sau

$$\begin{aligned} V_r \times T_s &= V_1 \times T_1 + V_2 \times T_2 + V_{0-7} \times T_{0-7} \\ T_s &= T_1 + T_2 + T_{0-7} \end{aligned} \quad (2.10)$$

Với T_s là chu kỳ điều rộng xung

T_n là thời gian duy trì ở trạng thái V_n

Chuyển sang hệ tọa độ vuông góc, ta có phương trình sau - suy ra từ phương trình (2.7) và (2.9)

$$T_s m \begin{pmatrix} \cos\left(\frac{\pi}{6} + \theta\right) \\ \sin\left(\frac{\pi}{6} + \theta\right) \end{pmatrix} = T_1 \frac{2}{\sqrt{3}} \begin{pmatrix} \cos\left(\frac{\pi}{6}\right) \\ \sin\left(\frac{\pi}{6}\right) \end{pmatrix} + T_2 \frac{2}{\sqrt{3}} \begin{pmatrix} \cos\left(\frac{\pi}{2}\right) \\ \sin\left(\frac{\pi}{2}\right) \end{pmatrix} + T_{0-7} \cdot 0$$

Cân bằng phần thực và phần ảo, ta có

$$\begin{cases} T_s m \cos\left(\frac{\pi}{6} + \theta\right) = T_1 \frac{2}{\sqrt{3}} \cos\left(\frac{\pi}{6}\right) + T_1 \frac{2}{\sqrt{3}} \cos\left(\frac{\pi}{2}\right) \\ T_s m \sin\left(\frac{\pi}{6} + \theta\right) = T_1 \frac{2}{\sqrt{3}} \sin\left(\frac{\pi}{6}\right) + T_1 \frac{2}{\sqrt{3}} \sin\left(\frac{\pi}{2}\right) \end{cases}$$

Giải phương trình trên để tìm T_1 và T_2

$$\Rightarrow T_1 = T_s m \frac{\sqrt{3}}{2} \frac{\cos\left(\frac{\pi}{6} + \theta\right)}{\cos\left(\frac{\pi}{6}\right)} = T_s m \frac{\sqrt{3}}{2} \frac{\cos\left(\frac{\pi}{6} + \theta\right)}{\frac{\sqrt{3}}{2}}$$

$$\Leftrightarrow T_1 = T_s m \cos\left(\frac{\pi}{6} + \theta\right) = T_s m \cos\left(\frac{\pi}{2} - \left(\frac{\pi}{3} - \theta\right)\right) = T_s m \sin\left(\frac{\pi}{3} - \theta\right)$$

$$\Rightarrow T_2 = T_s m \frac{\sqrt{3}}{2} \sin\left(\frac{\pi}{6} + \theta\right) - T_s m \cos\left(\frac{\pi}{6} + \theta\right) \sin\left(\frac{\pi}{6}\right)$$

$$\Leftrightarrow T_2 = T_s m \left[\sin\left(\frac{\pi}{6} + \theta\right) \cos\left(\frac{\pi}{6}\right) - \cos\left(\frac{\pi}{6} + \theta\right) \sin\left(\frac{\pi}{6}\right) \right]$$

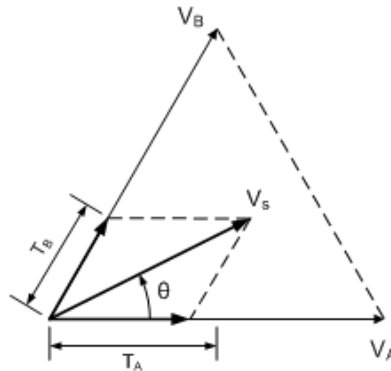
$$\Leftrightarrow T_2 = T_s m \sin\left(\frac{\pi}{6} + \theta - \frac{\pi}{6}\right) = T_s m \sin(\theta)$$

Suy ra

$$\begin{cases} T_1 = T_s m \sin(\pi/3 - \theta) \\ T_2 = T_s m \sin(\theta) \\ T_{0-7} = T_s - T_1 - T_2 \end{cases} \quad (2.11)$$

Trong đó : m là tỉ số điều biên
 T_s là chu kỳ điều rộng xung
 θ là góc lệch giữa V_r và V_n .

Ta nhận thấy việc giải phương trình 2-10 để tìm T_1 , T_2 và T_s không phụ thuộc vào hai vector giới hạn của vùng đó



Hình 2.11: Vector không gian V_s trong vùng bất kỳ

Dựa trên kết quả phương trình 2-11, ta xây dựng công thức tổng quát như trong phương trình (2.12) sau đây:

$$\begin{cases} T_A = T_s m \sin(\pi/3 - \theta) \\ T_B = T_s m \sin(\theta) \\ T_{0-7} = T_s - T_A - T_B \end{cases} \quad (2.12)$$

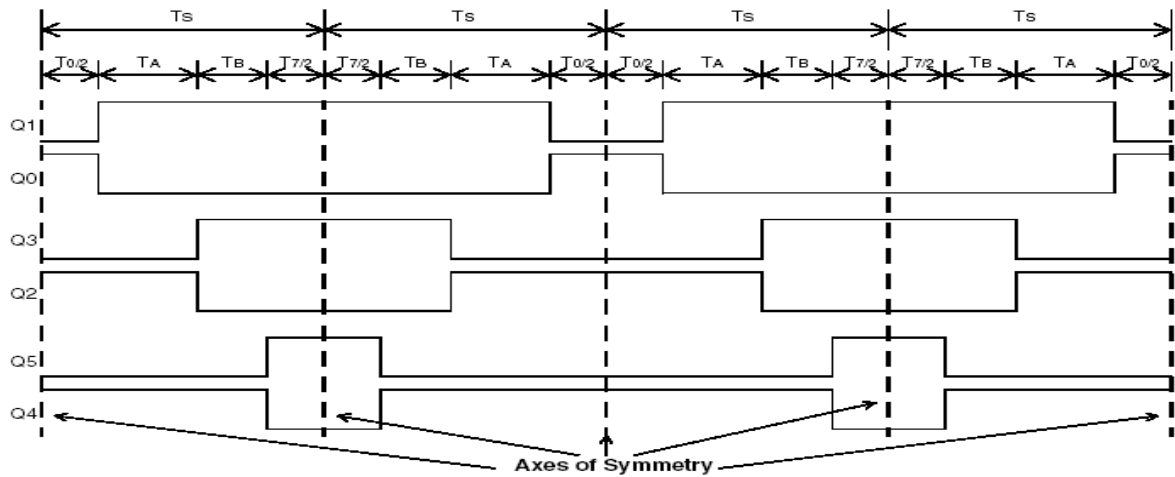
2.3.2.3 Phân bố các trạng thái đóng ngắt:

Vấn xét trường hợp vector V_s nằm trong vùng 1, với kết quả từ phương trình 2-11:

$$\begin{cases} T_1 = T_s m \sin(\pi/3 - \theta) \\ T_2 = T_s m \sin(\theta) \\ T_{0-7} = T_s - T_1 - T_2 \end{cases}$$

2.3.2.4 Kỹ thuật thực hiện điều chế vector không gian:

Thông thường, một trong những tiêu chuẩn để lựa chọn giản đồ đóng ngắt 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:

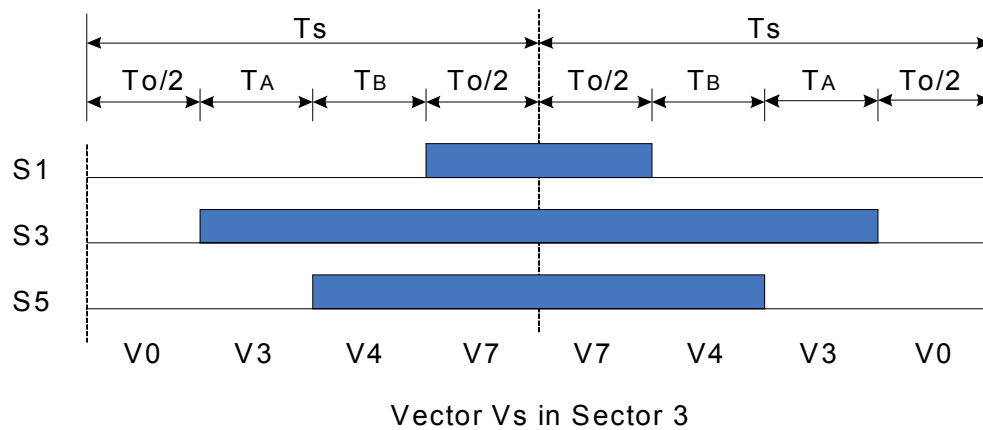
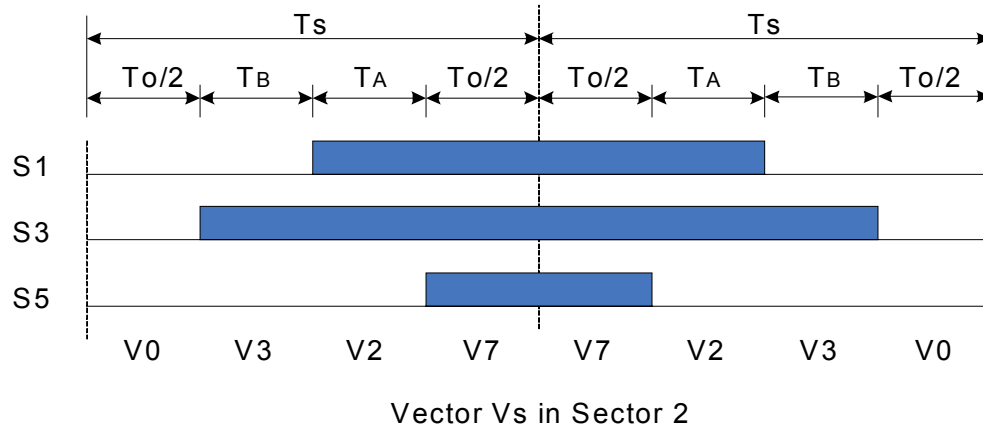
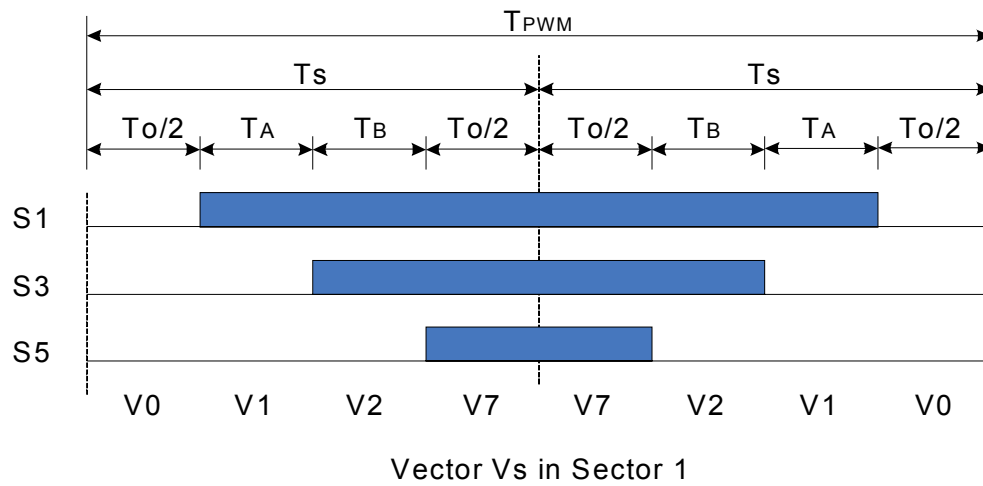


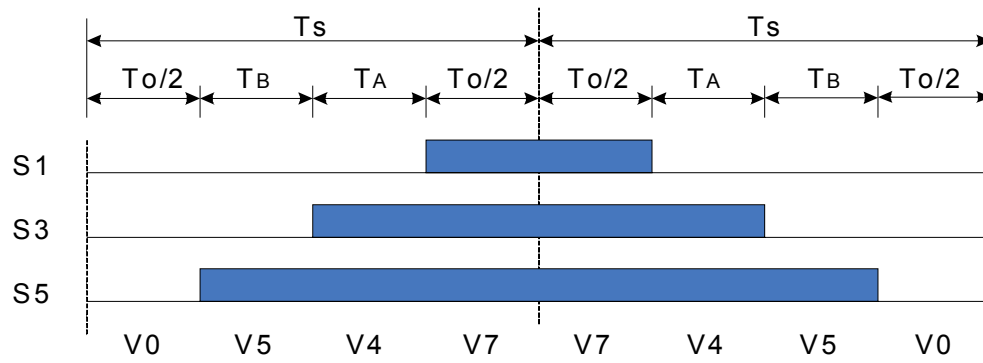
Hình 2.12: Giản đồ đóng ngắt linh kiện

2.3.2.5 Giản đồ đóng ngắt các khóa để tạo ra Vector V_s trong từng sector:

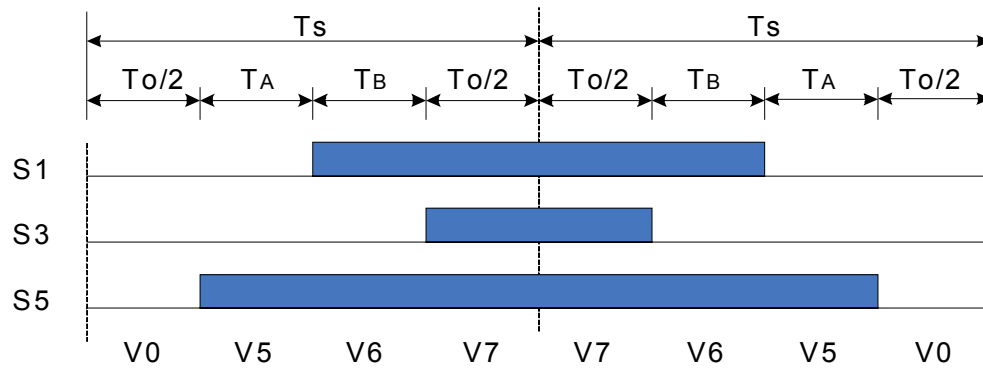
Các khóa công suất trong từng nhánh đóng ngắt đối nghịch nhau. Để đơn giản hóa sơ đồ, ta chỉ vẽ trạng thái của 3 khóa công suất phía trên. Ba khóa còn lại có trạng thái đối nghịch với 3 khóa trên theo từng cặp như sau :

- + S0 – S1
- + S2 – S3
- + S4 – S5

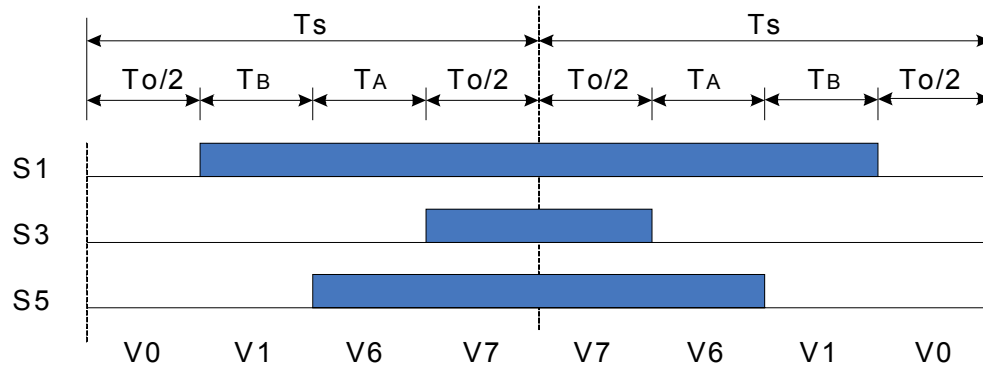




Vector V_s in Sector 4



Vector V_s in Sector 5



Vector V_s in Sector 6

Hình 2.13: Vector V_s trong các vùng từ 0-6

CHƯƠNG 3

CẤU TẠO VÀ CÁC THÔNG SỐ PHẦN CỨNG

CHƯƠNG 3 : CẤU TẠO VÀ CÁC THÔNG SỐ PHẦN CỨNG**YÊU CẦU ĐẶT RA :**

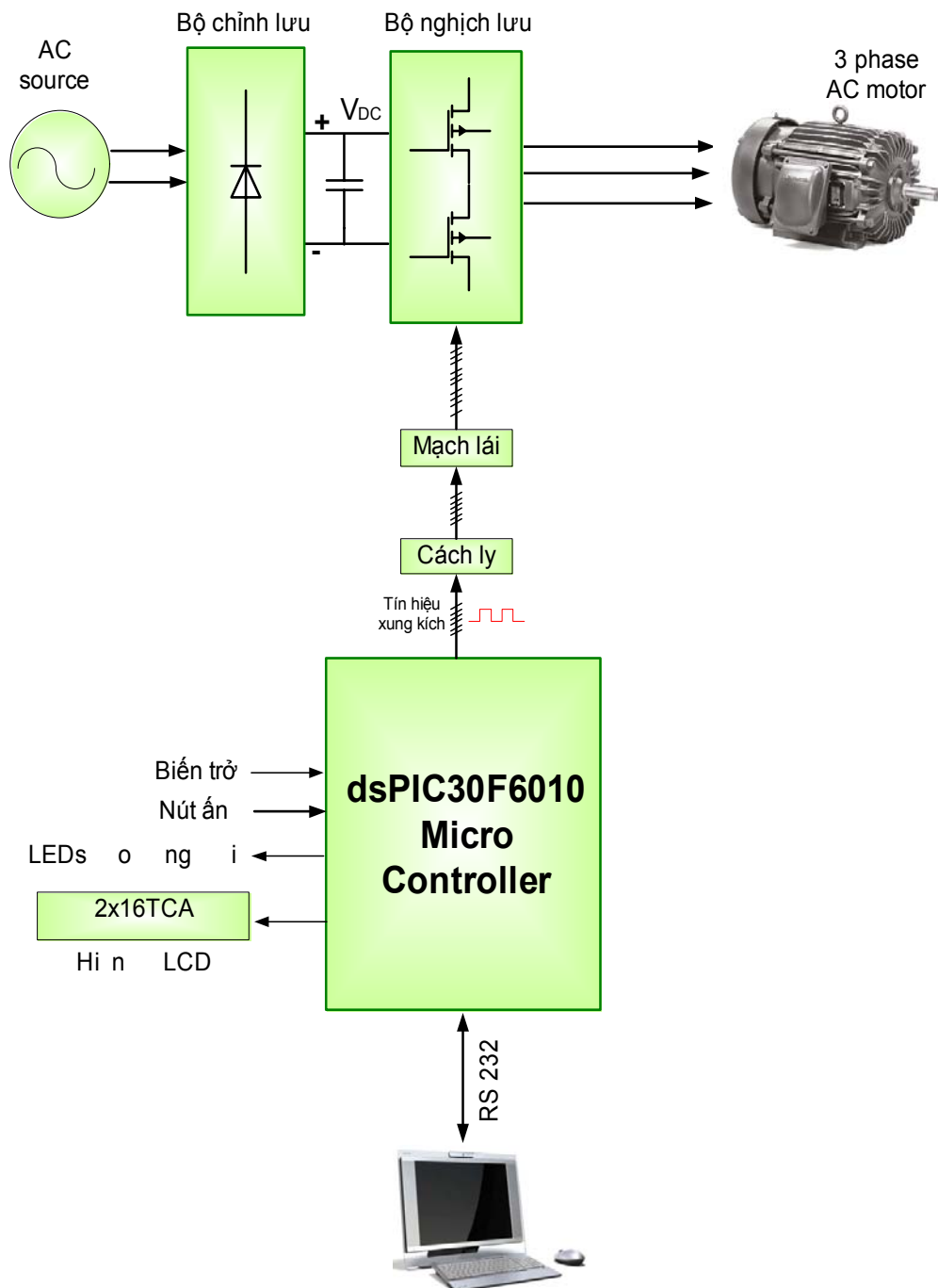
“Thiết kế bộ biến tần truyền thống (6 khóa) ba pha điều khiển động cơ KĐB theo phương pháp V/f và điều chế SINPWM)”

Thông số động cơ như sau :

	Các thông số	Đơn vị	Động cơ đầu sao	Động cơ đầu tam giác
P_{dm}	Công suất định mức	(KW)	1.5 (2HP)	1.5(2HP)
V_{dm}	Điện áp định mức	(VAC)	380	220
I_{dm}	Dòng điện định mức	(A)	10	10
η	Hiệu suất	%		
$\cos\varphi$	Hệ số công suất		0.8	0.8
s	Độ trượt	%		
M	Moment			
RPM	Vận tốc	(vòng /phút)		

Bảng 3.1: Thông số động cơ

3.1 Sơ đồ khối của mạch điều khiển động cơ:



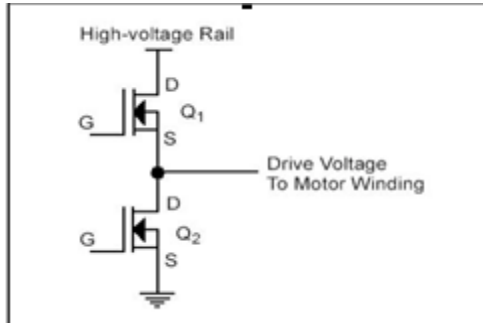
Hình 3.1: Sơ đồ khối mạch điều khiển

3.2 Giới thiệu chi tiết các khối điều khiển:

3.2.1 Mạch lái

Có hai sự lựa chọn cho các khóa đóng ngắt công suất để điều khiển động cơ là MOSFET và IGBT vì khả năng chịu dòng và áp cao.

Nói chung , những loại động cơ mà sử dụng các khóa đóng ngắt (MOSFET , IGBT) để điều khiển thì đều cần dùng đến mạch lái (gate drive scheme). Có 2 phần cơ bản trong việc điều khiển các đóng ngắt các linh kiện công suất là: điều khiển phía cao (high side – Q1) và phía thấp (low side Q2).



Hình 3.2: Ví dụ sơ đồ điều khiển mosfet

Trong ví dụ trên Q1 và Q2 luôn ở trạng thái làm việc đối nghịch nhau. Khi Q1 ở trạng thái ON thì Q2 ở trạng thái OFF và ngược lại.

Khi Q1 đang ở trạng thái OFF chuyển sang trạng thái ON => chân S (MOSFET) hay chân E (IGBT) của Q1 chuyển từ ground sang điện áp cao (high voltage rail). Do đó muốn kích Q1 tiếp tục ON thì phải tạo điện áp kích V_{GS1} có giá trị $V_{GS1} = V_{SQ1} + \Delta V$. Trong khi đó tín hiệu ra của vi xử lý điều khiển đóng ngắt các khóa chỉ có giá trị điện áp +5V (so với ground). Nên cần phải có mạch lái để tạo trôi áp và cách ly trong việc đóng ngắt phía cao Q1.

Tuy nhiên đối với Q2 thì chân S được nối ground , do đó điện áp kích V_{GS2} chỉ cần có giá trị ΔV . Do đó việc đóng ngắt khóa low side (Q2) được điều khiển dễ dàng hơn .

Ghi chú:

ΔV : giá trị điện áp cần thiết để kích Q1 hay Q2 dẫn. Đối với MOSFET và IGBT ΔV có giá trị từ 10 đến 15 (V).

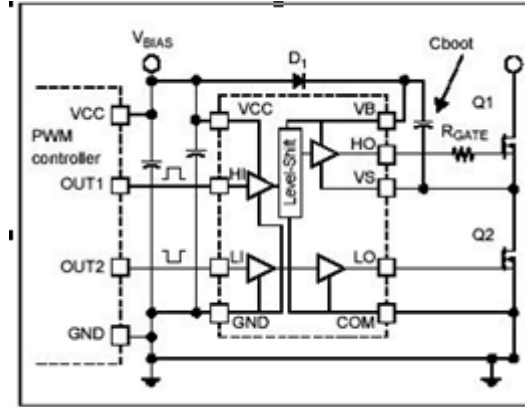
Sơ đồ mạch lái để điều khiển đóng ngắt MOSFET hay IGBT (Q1)

Có 3 dạng sơ đồ cơ bản như sau:

- 1-Single ended or double ended gate drive transformer.
- 2-Floating bias voltages and opto – isolater drive.
- 3-High voltage bootstrap driver ICs.

Trong các phương án (1),(2) sử dụng biến áp xung trong thiết kế mạch lái mosfet , 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 trong phần này đề cập đến phương án sử dụng loại *High Voltage Bootstrap Driver ICs*.



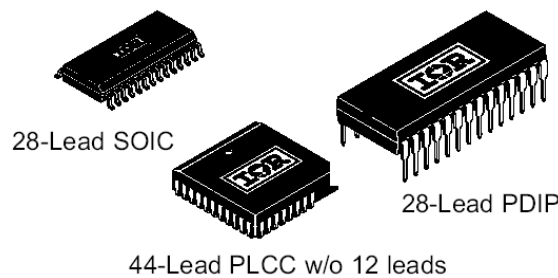
Hình 3.3: Sơ đồ khối của IC lái mosfet

Hình 11 đưa ra một giải pháp để điều khiển kích đóng ngắt phía cao Q1, và hơn thế nữa nó không đòi hỏi người dùng cần phải có kiến thức về máy biến áp. Những ICs loại này sử dụng mạch dịch mức (level shifting circuitry) bằng tụ C “bootstrap” để lái phía cao.

Trong suốt thời gian ON của Q2 chân S của Q1 có điện thế là ground. Điều này cho phép tụ C_{boot} được nạp (thông qua diode D1) đến giá trị V_{BIAS} . Khi Q2 được kích OFF và Q1 được kích ON thì điện áp chân S của Q1 bắt đầu tăng lên. Tụ C_{boot} lúc này đóng vai trò của nguồn phân cực, cung cấp dòng để lái phía cao Q1.

Nhược điểm mạch lái loại này là có thời gian delay giữa tín hiệu input và tín hiệu đóng ngắt các khóa bán dẫn. Thời gian trễ hoàn từ 500ns \rightarrow 1us. Nó có thể là vấn đề khi tiến hành các ứng dụng hoạt động ở tần số cao (nhưng tần số hoạt động của động cơ $< 60\text{Hz}$).

Giới thiệu về IC IR2136 (High voltage bootstrap driver ICs)



Hình 3.4: IC IR2136

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.

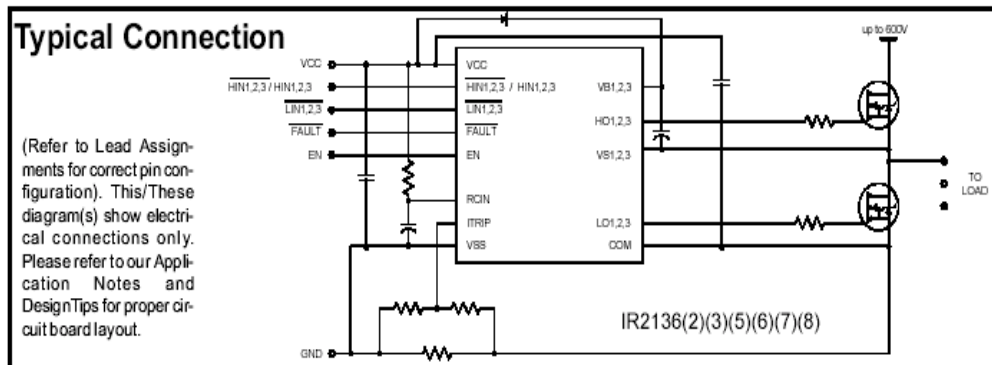
Các thông số:

- Các kênh trôi áp thiết kế cho chế độ bootstrap có thể lên đến +600V.
- Chống dV/dt (dV/dt immune)
- Điện áp kích công từ 10V – 20V.
- Undervoltage lockout for all channels.

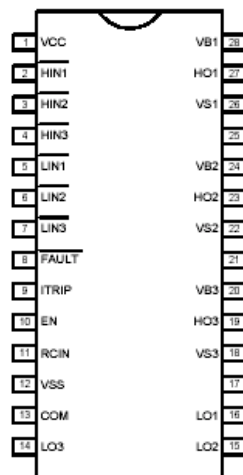
CHƯƠNG 3: CẤU TẠO VÀ CÁC THÔNG SỐ PHẦN CỨNG

- Chống quá dòng: sáu tín hiệu driver outputs sẽ bị tắt khi quá dòng xảy ra (Thông qua chân ITRIP của IR).
- Logic inputs tương thích với CMOS hay LSTTL outputs, có thể xuống đến 3.3 V
- Giảm di/dt cho các tín hiệu lái cổng, do đó chống nhiễu tốt hơn.
- Có thể điều chỉnh thời gian delay cho chế độ tự động xóa lỗi (automatically fault clear), thông qua chân FAULT của IR.

Sơ đồ kết nối do hãng sản xuất IR cung cấp



Sơ đồ chân của IR2136



28 Lead PDIP

IR2136/IR2136(5)(6)(7)(8)

Hình 3.5: Sơ đồ kết nối IR2136

Định nghĩa các chân của IR2136

VCC	Nguồn cung cấp 15VDC
VSS	Ground
HIN1,2,3	Logic input cho phía gate driver outputs (HO1,2,3), tích cực mức thấp
LIN1,2,3	Logic input cho phía gate driver outputs (LO1,2,3), tích cực mức thấp
FAULT	Phát hiện quá dòng (I_{TRIP}) hay low side undervoltage lockout xảy ra

EN	Logic input cho phép chức năng I/O .
ITRIP	Analog input . Khi hoạt động, I_{TRIP} khóa các ngõ ra và kích hoạt chân FAULT và RCIN. Khi I_{TRIP} trở về trạng thái bình thường (inactive), FAULT vẫn tích cực set thời gian T_{FLTCLR} , sau đó tự động inactive (open drain high impedance).
RCIN	Đặt thời gian FAULT CLEAR delay
COM	Low side gate return
VB1,2,3	High side floating supply
HO1,2,3	High side gate driver output
VS1,2,3	High voltage floating supply returns
LO1,2,3	Low side gate driver output

Bảng 3.2 : Định nghĩa các chân trong IR2136**3.2.2 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.

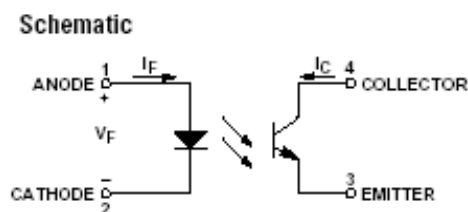
+ Biến áp xung :

Gồm một cuộn dây sơ cấp và có thể nhiều cuộn thứ cấp. Với nhiều cuộn dây phía thứ cấp, ta có thể kích động 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.

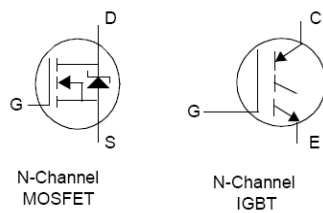
**Hình 3.6: Sơ đồ khối của opto**

=> ta lựa chọn phương án dùng opto. Yêu cầu đặt ra đối với opto là phải chịu được tần số đóng ngắt khá cao ($\approx 10\text{KHz}$). Trong đó, **HCPL-2631** là optocouplers của hãng Fairchild có đáp ứng tần số lên đến 10MHz

3.2.3 Mạch MOSFETs**Các loại linh kiện thường được sử dụng trong 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.



Hình 3.7: Sơ đồ khối của MOSFET và IGBT

Việc đóng cắt linh kiện cũng sẽ gây nên tổn hao công suất, công thức xác định tổn hao công suất được trình bày như sau:

MOSFET

$$P_{\text{LOSS}} = I_{\text{rms}}^2 * R_{\text{DS-ON}}$$

where:

$R_{\text{DS-ON}}$ = drain-to-source on-state resistance

I_{rms} = drain-to-source rms current

IGBT

$$P_{\text{LOSS}} = I_{\text{ave}} * V_{\text{CE-SAT}}$$

$V_{\text{CE-SAT}}$ = collector-to-emitter saturation voltage

I_{ave} = collector-to-emitter average current

Đặc điểm, ứng dụng:

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

IGBT là linh kiện có tần số đóng cắt giới hạn thấp hơn so với MOSFET, vì vậy dẫn đến tổn thất công suất do đóng cắt linh kiện sẽ cao hơn đối với ở MOSFET có tần số đóng cắt cao hơn. Các kỹ thuật sử dụng IGBT trong điều khiển đã được sớm áp dụng cách đây hơn 10 năm. Có rất nhiều thay đổi cải thiện linh kiện với các ứng dụng khác nhau, nhiều công ty đã sản xuất ra nhiều dòng IGBT, một số được chế tạo thích hợp với các ứng dụng ở tốc độ thấp và điện áp VCE-SAT nhỏ, dẫn tới tổn hao sẽ nhỏ. Một số khác được sản xuất phù hợp với các ứng dụng đòi hỏi tốc độ cao (60kHz đến 150 kHz) và có tổn thất công suất thấp hơn nhưng có VCE-SAT cao hơn. Khoảng 5 năm trở lại đây nhiều cải tiến trong việc sản xuất MOSFET có thể chấp nhận tần số đóng cắt cao hơn với RDS-ON nhỏ (khoảng vài miliohm) làm cho tổn thất công suất được giảm đi rất nhiều. Vì vậy ngày nay, đa số các bộ nghịch lưu thường sử dụng MOSFET hơn là IGBT như trước kia.

Dựa vào các đặc điểm nêu trên, khi lựa chọn linh kiện ta cần xem xét đến khả năng giới hạn của linh kiện. Trong phạm vi thực hiện của đồ án môn học hai, ta chỉ quan tâm đến các thông số hoạt động của động cơ để lựa chọn cho phù hợp.

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

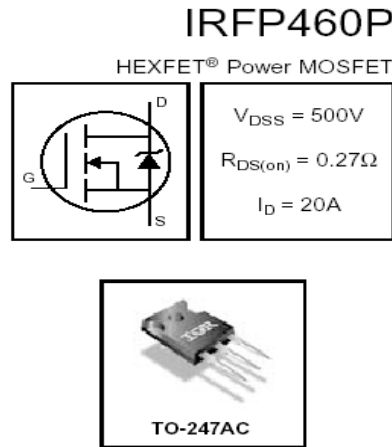
Điện áp VDS (Mosfet) hay VCE (IGBT) $\gg V_{\text{DC}} / 2$

Dòng điện qua linh kiện lớn hơn dòng định mức của động cơ $\approx 10\text{A}$ ở nhiệt độ hoạt động

Chịu được tần số đóng ngắt cao

...

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



Hình 3.8: IRFP460P

3.2.4 Mạch chỉnh lưu

3.2.4.1 Bộ chỉnh lưu:

Yêu cầu:

- Điện áp trung bình (V_{DC}) đầu ra của bộ chỉnh lưu:
 - +Trong phương pháp SINEPWM thì : $V_{OA} = \frac{V_{DC}}{2}$
 - +Động cơ vận hành ở chế độ định mức ($V_{AO} = 220V$)
 $\Rightarrow V_{DC} = 2 * V_{AO} = 2 * 220 = 440V$
- Trị tức thời của VDC được nắn tương đối phẳng
- Gọn nhẹ , giá thành rẻ

3.2.4.2 Phương pháp chỉnh lưu :

Ta sử dụng phương pháp chỉnh lưu cầu với 6 diode

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

$$V_{DC} = \frac{3\sqrt{6} * V_{pha}}{\pi} \approx 515 (V)$$

+Với giá trị này của V_{DC} thì động cơ có thể vận hành ở định mức

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

✓ 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 :

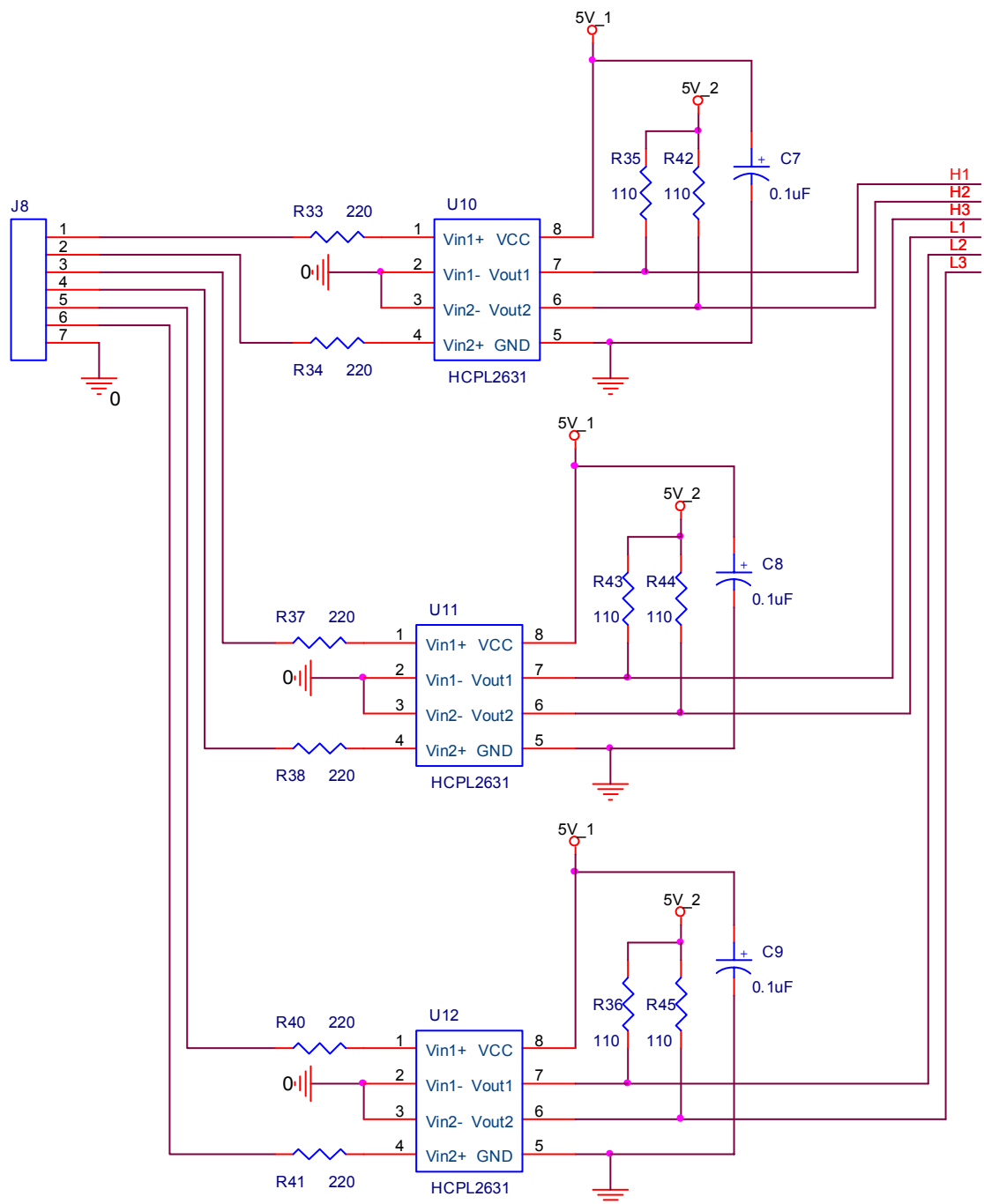
$$V_{DC} = \frac{2\sqrt{2} * V_{pha}}{\pi} \approx 200(V) \Rightarrow \text{Động cơ sẽ không thể vận hành hết định mức}$$

CHƯƠNG 4

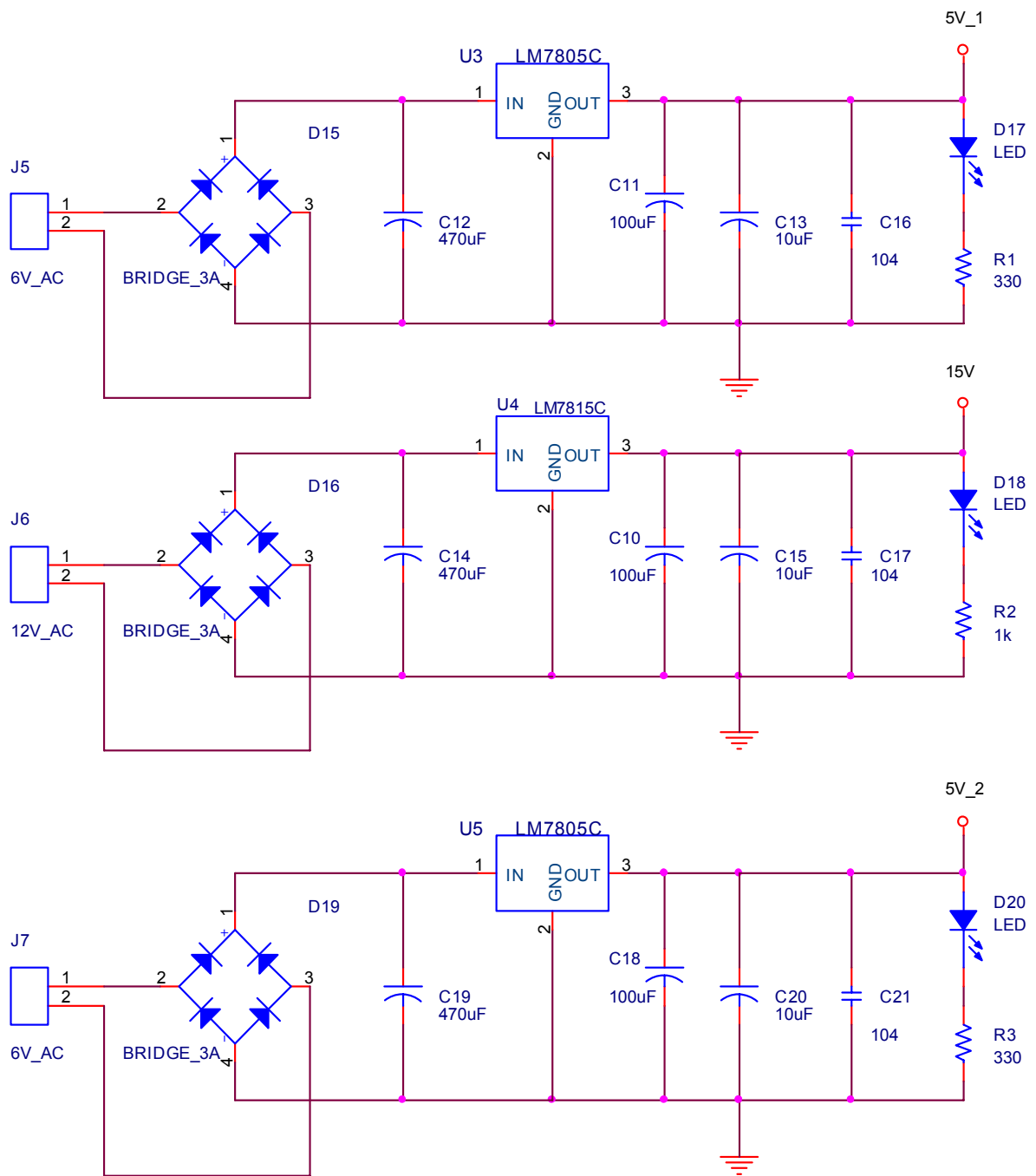
SƠ ĐỒ CẤU TẠO MẠCH ĐIỀU KHIỂN

CHƯƠNG 4 : SƠ ĐỒ CẤU TẠO MẠCH ĐIỀU KHIỂN

4.1 Sơ đồ mạch cách ly



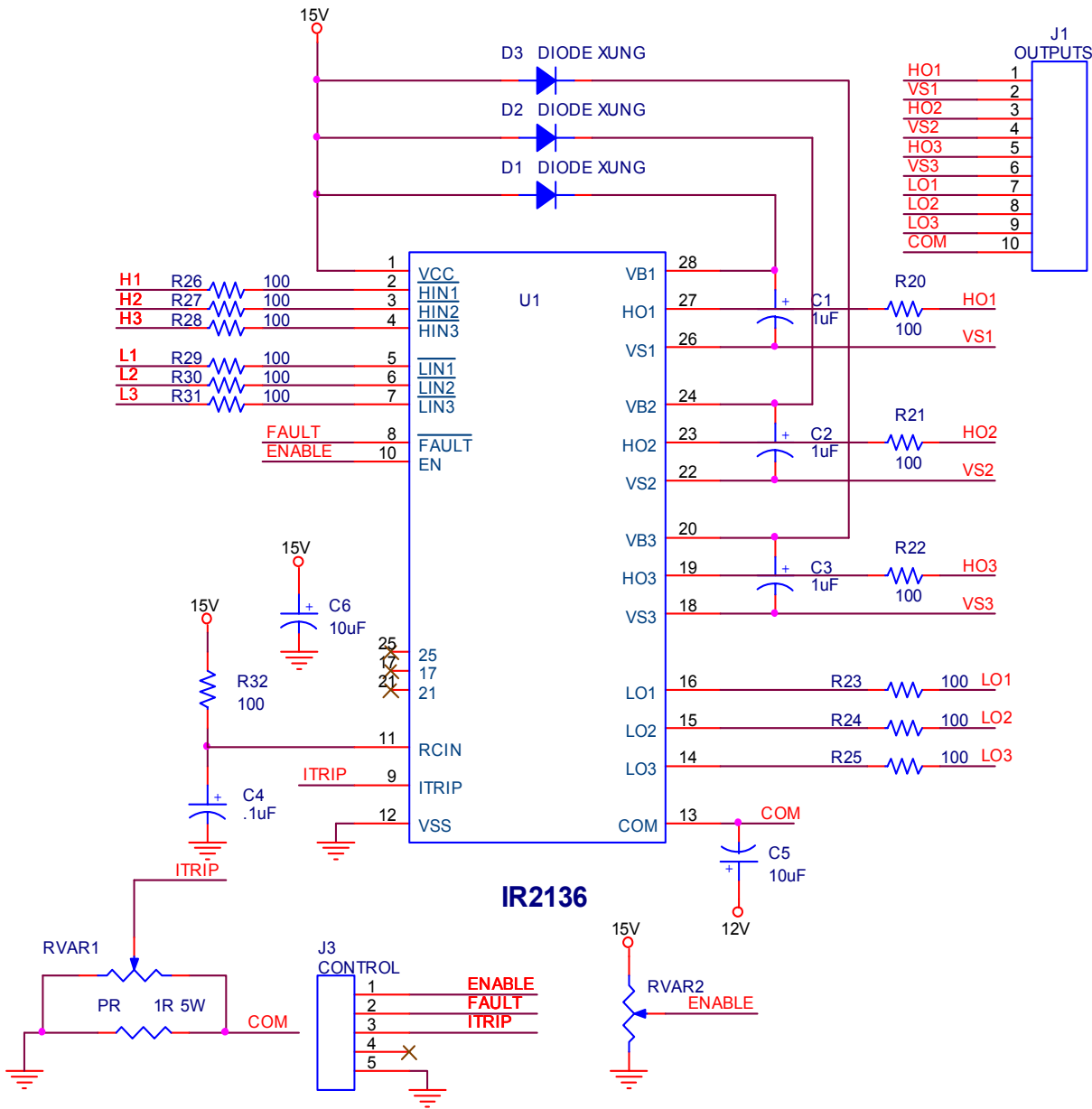
CHƯƠNG 4: SƠ ĐỒ CẤU TẠO MẠCH ĐIỀU KHIỂN



Hình 4.1 : Sơ đồ mạch cách ly

CHƯƠNG 4: SƠ ĐỒ CẤU TẠO MẠCH ĐIỀU KHIỂN

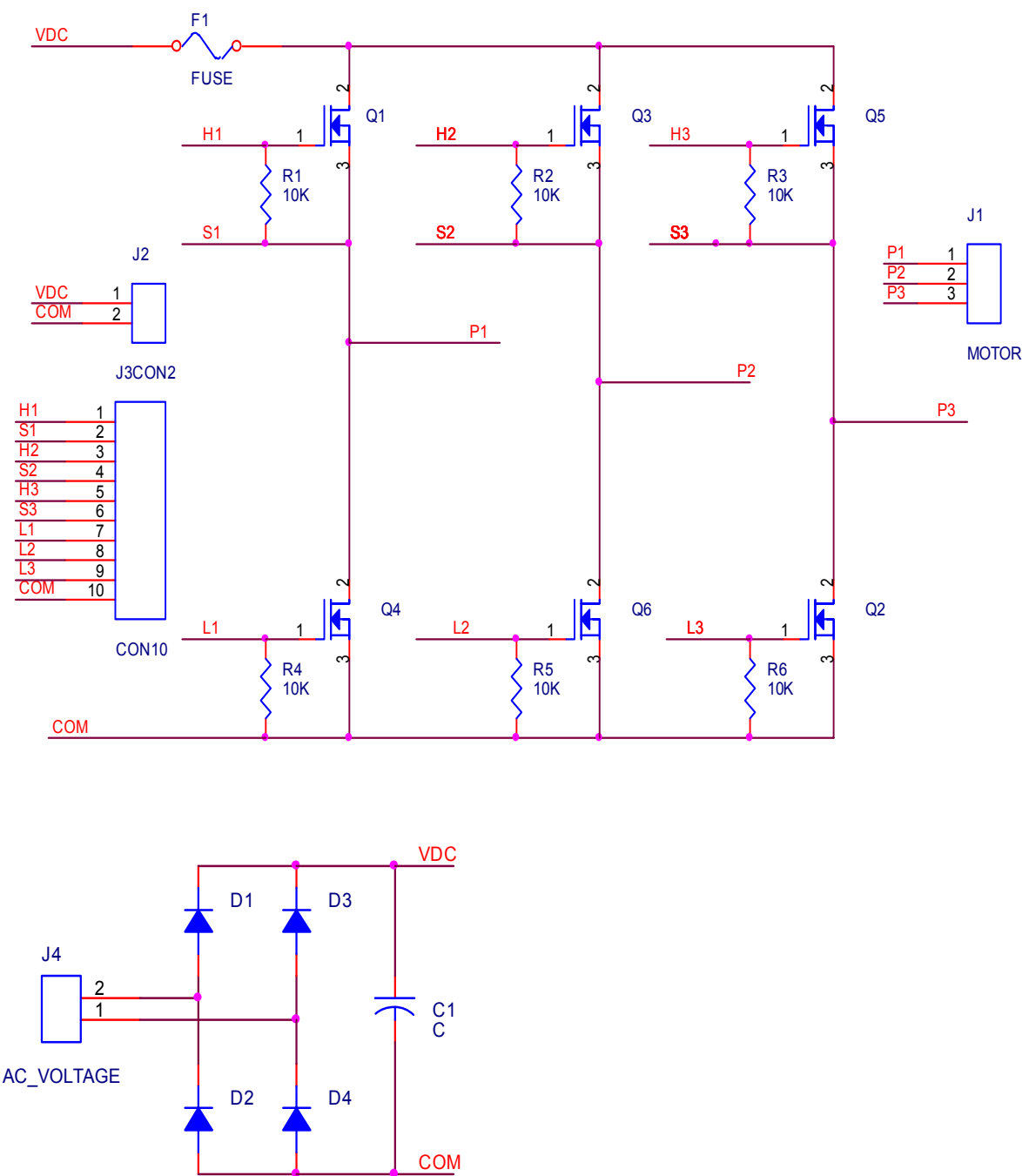
4.2 Sơ đồ mạch lái



Hình 4.2 : Sơ đồ mạch lái mosfet

CHƯƠNG 4: SƠ ĐỒ CẤU TẠO MẠCH ĐIỀU KHIỂN

4.3 Sơ đồ mạch động lực

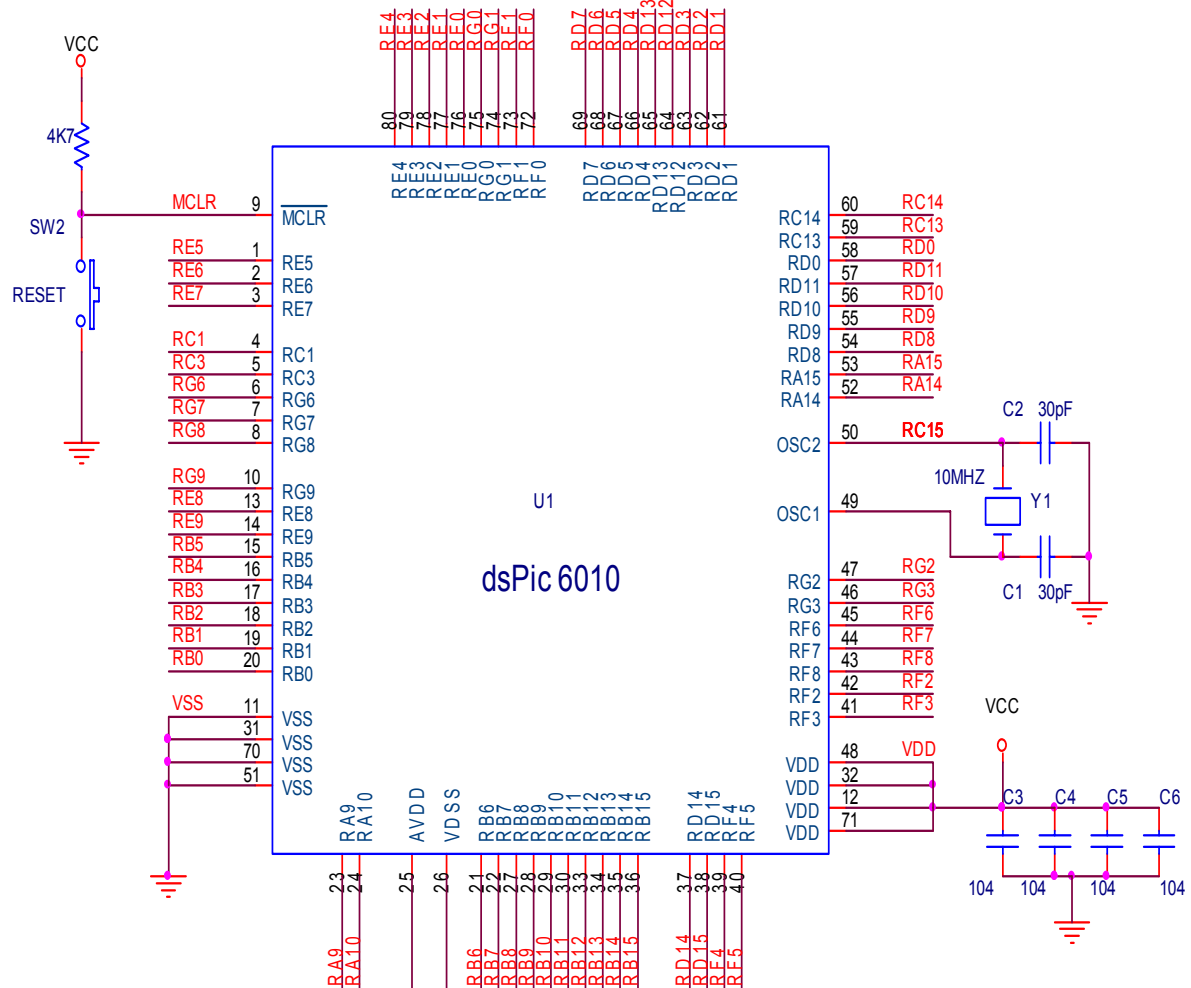


Hình 4.3 : Sơ đồ mạch động lực

CHƯƠNG 4: SƠ ĐỒ CẤU TẠO MẠCH ĐIỀU KHIỂN

4.4 Sơ đồ mạch điều khiển

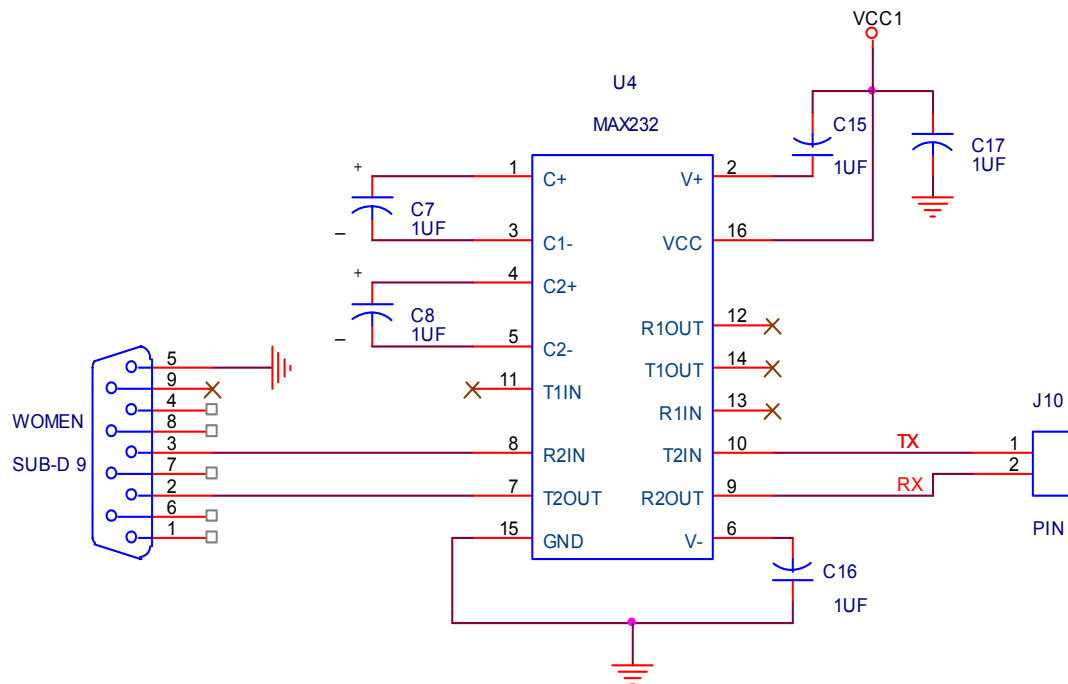
4.4.1 Khối điều khiển



Hình 4.4 : Sơ đồ khối điều khiển chính

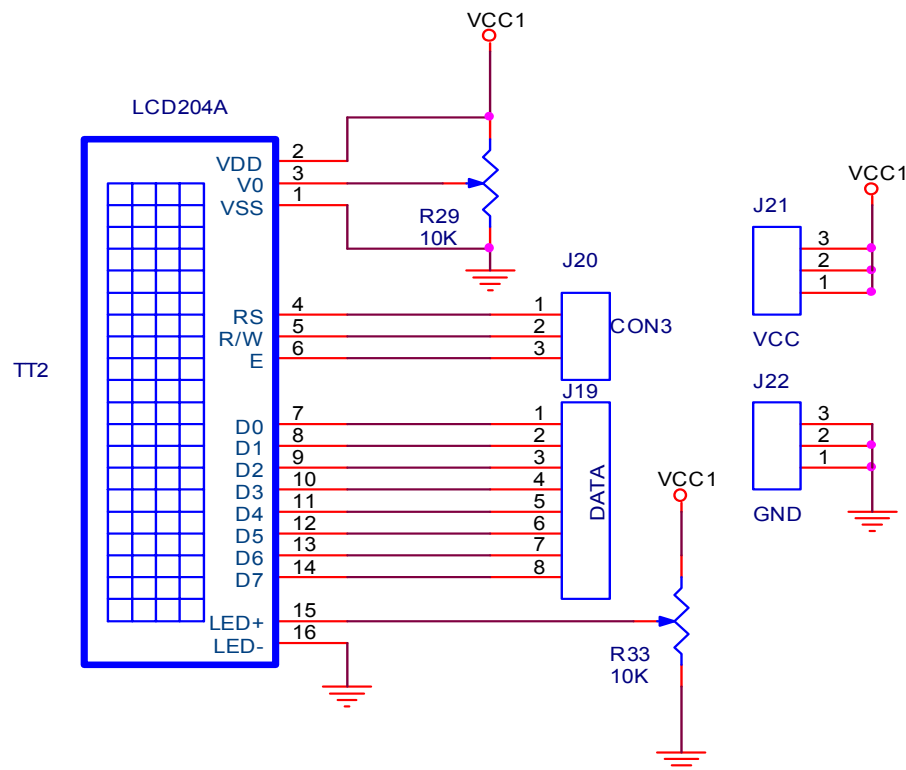
CHƯƠNG 4: SƠ ĐỒ CẤU TẠO MẠCH ĐIỀU KHIỂN

4.4.2 Khối giao tiếp máy tính



Hình 4.5 : Sơ đồ khối giao tiếp máy tính

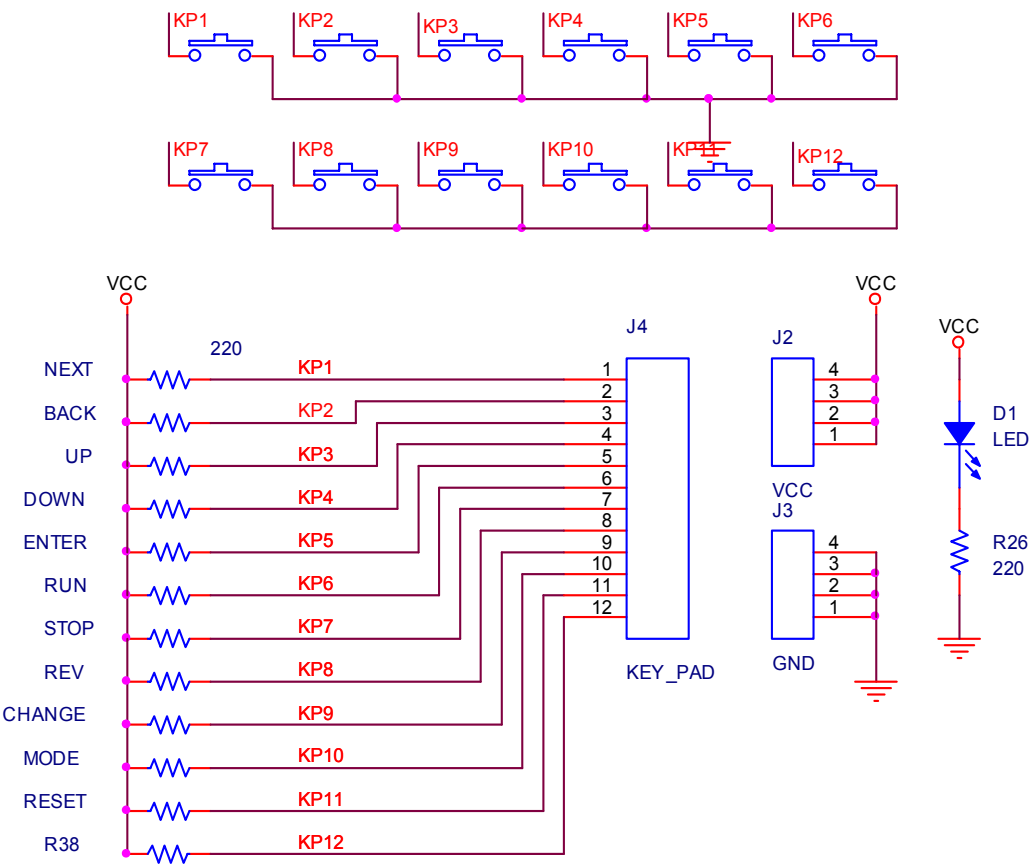
4.4.3 Khối hiển thị



Hình 4.6 : Sơ đồ khối hiển thị

CHƯƠNG 4: SƠ ĐỒ CẤU TẠO MẠCH ĐIỀU KHIỂN

4.4.4 Khối nút bấm




Hình 4.7 : Sơ đồ khối nút bấm

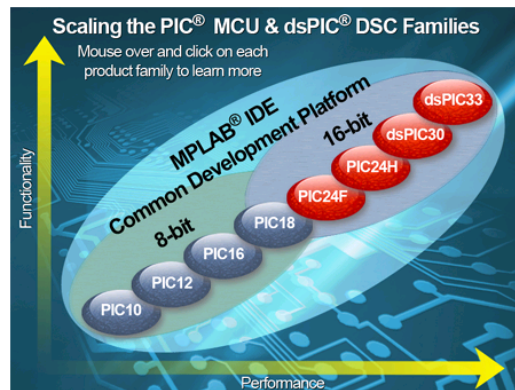
CHƯƠNG 5

GIỚI THIỆU VỀ dsPIC6010

CHƯƠNG 5: GIỚI THIỆU VỀ DSPIC 6010

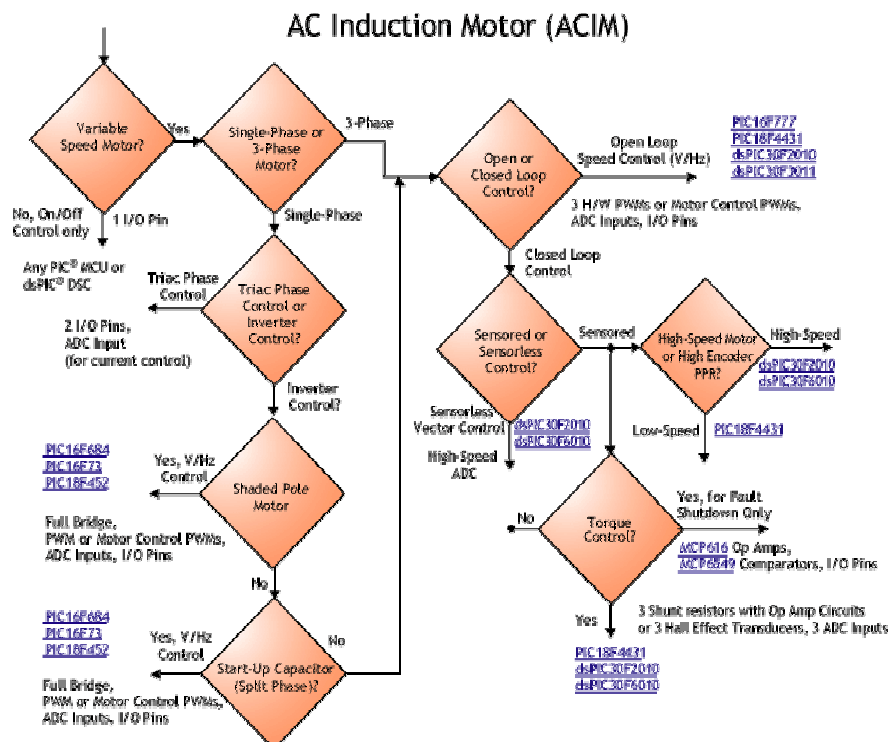
5.1 Tổng quan về vi điều khiển dsPIC30F6010

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 trên ngôn ngữ quen thuộc như C (C18, C30, CCSC, MIKO_C, HI-TECH PICC), Pascal (MIKO_PASCAL) thông qua các phần mềm hỗ trợ. Gồm các họ như sau:



Hình 5.1 : Các họ vi điều khiển PIC và dsPIC

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 các loại IC chuyên dùng để điều khiển động cơ 3 pha theo đề nghị của của Microchip được trình bày theo sơ đồ sau



Hình 5.2: Sơ đồ ứng dụng các họ vi điều khiển

Trong nội dung luận văn này, chúng ta sử dụng MCU dsPIC30F6010 cho việc điều khiển động cơ không đồng bộ ba pha với các đặc điểm được trình bày như sau:

- MCU dsPic 6010 được thiết kế dựa trên kiến trúc RISC, hoạt động ở tầm điện áp rộng từ 2.5-5.5V, công suất thấp, có tốc độ xử lý cao do sử dụng công nghệ CMOS, đáp ứng được yêu cầu làm việc đối với khối lượng tính toán tính toán lớn, đáp ứng nhanh và yêu cầu độ chính xác cao

- Tập lệnh của MCU 6010 gồm 84 lệnh

- Có bộ nhớ RAM là 8K, bộ nhớ EPPROM là 4K, đặc biệt có bộ nhớ Plash_Program lên đến 144K, phù hợp với các chương trình đòi hỏi có bộ nhớ chương trình lớn

- Hoạt động chế độ nguồn dao động ngoài (External Clock) tối đa là 40Mhz

- Hoạt động ở chế độ dao động thạch anh tần số từ 4Hhz-10Mhz với các cấp độ nhân

PLL(Phase Locked Loop) 4x, 8x, 16x, tùy theo các thiết lập chương trình mà dao động thực sự đưa vào trong MCU có thể lên đến 120Mhz

PLL có thể được thiết lập thông qua các bits FPR<3:0>, tần số dao động thạch anh đưa vào và tần số dao động thực sự của MCU được tóm tắt trong bảng 1.1

FIN	PLL Multiplier	FOUT
4 MHz-10 MHz	x4	16 MHz-40 MHz
4 MHz-10 MHz	x8	32 MHz-80 MHz
4 MHz-7.5 MHz	x16	64 MHz-120 MHz

Bảng 5.1 : Thiết lập tần số hoạt động

- MCU 6010 có 44 nguồn Interrup, mỗi nguồn ngắt có 8 cấp độ ngắt ưu tiên

- Có 16x16 bits mảng thanh ghi làm việc

- Gồm có 7 port I/O (A,B,C,D,E,F,G)

- 5 bộ Timer/Counter 16bits, trong đó có thể ghép lại với nhau thành các bộ Timer/Counter 32 bits

- Được tích hợp các module Compare/ Capture, có 8 kênh PWM được sử dụng trong kỹ thuật điều khiển động cơ

- Module I2C hỗ trợ chế độ Multi- Master/Slave và 7 bits/10 bits xác định địa chỉ

- Ngoài ra còn tích hợp các chuẩn giao tiếp CAN, UART và 3-wire SPI

- 16 kênh 10 bits - Analog to Digital Converter (10bits - AD) tốc độ cao (500Ksps), có khả năng chuyển đổi trong lúc MCU ở trạng thái Sleep, Idle

5.2 Các đặc điểm đặc biệt ở họ MCU dsPic-6010:

- Cho phép lập trình lại nhiều lần, số lần nạp xóa đối với bộ nhớ chương trình là từ 10.000(min) – 100.000(max) lần đối với các dòng chip sử dụng trong công nghiệp

- Chế độ Fail-safe clock monitor cho phép MCU kiểm soát phát hiện ra khi nào dao động bên ngoài từ nguồn dao động gặp sự cố, MCU sẽ tự động chuyển sang chế độ dao động nội trong MCU

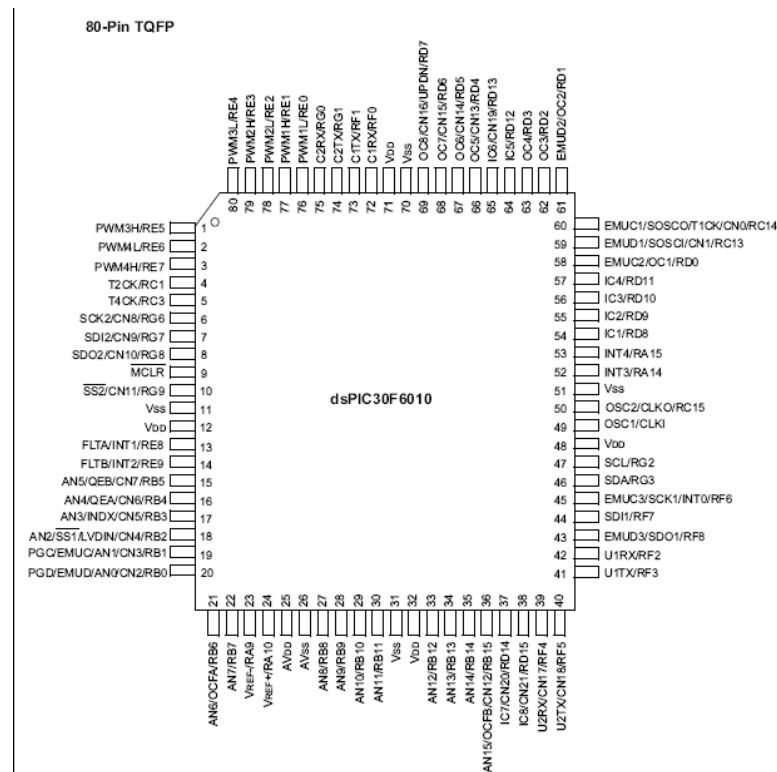
- Chế độ bảo vệ code bên trong MCU

- Hỗ trợ In-circuit serial programming (ICSP) tạo thuận lợi cho người lập trình khả năng lập trình trực tiếp trên phần cứng, không phải tháo lắp IC như các loại MCU trước đây

- Chế độ quản lý, tiết kiệm năng lượng : Sleep, idle, alternate clock mode

5.3 Giới thiệu khái quát về cấu trúc phần cứng:

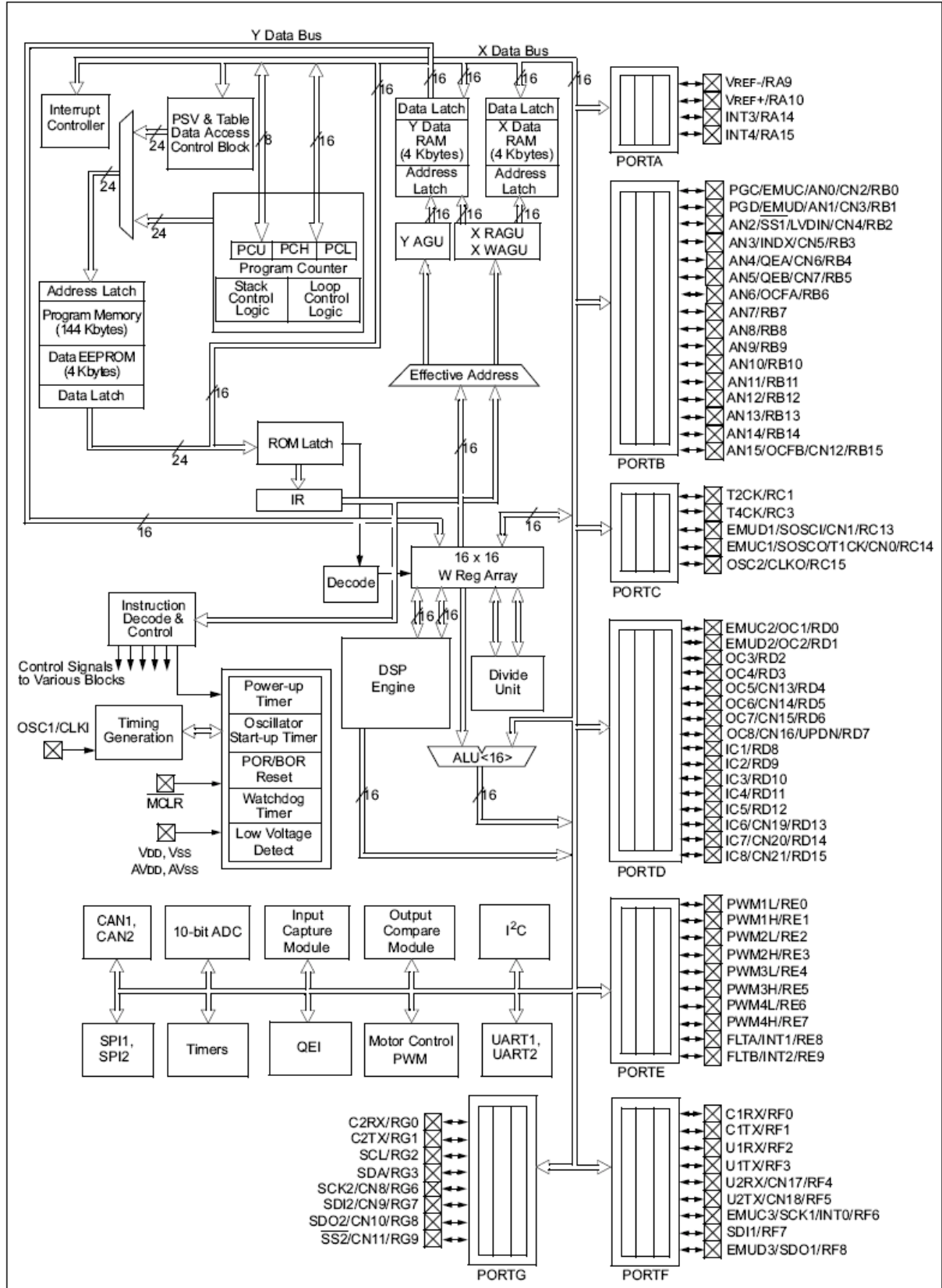
Sơ đồ chân MCU dsPIC6010 được trình bày trong hình 2.1



Hình 5.3: Sơ đồ chân dsPIC30F6010

CHƯƠNG 5: GIỚI THIỆU VỀ dsPIC6010

Sơ đồ tổ chức bên trong MCU dsPIC6010 được trình bày như hình 31



Hình 5.4: Sơ đồ tổ chức bên trong MCU dsPIC6010

CHƯƠNG 5: GIỚI THIỆU VỀ dsPIC6010

Mô tả chức năng, tính chất các I/O trong MCU

Pin Name	Pin Type	Buffer Type	Description
AN0-AN15	I	Analog	Analog input channels. AN0 and AN1 are also used for device programming data and clock inputs, respectively.
AVdd	P	P	Positive supply for analog module.
AVss	P	P	Ground reference for analog module.
CLKI CLKO	I O	ST/CMOS —	External clock source input. Always associated with OSC1 pin function. Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. Optionally functions as CLKO in RC and EC modes. Always associated with OSC2 pin function.
CN0-CN23	I	ST	Input change notification inputs. Can be software programmed for internal weak pull-ups on all inputs.
COFS	I/O	ST	Data Converter Interface frame synchronization pin.
CSCK	I/O	ST	Data Converter Interface serial clock input/output pin.
CSDI	I	ST	Data Converter Interface serial data input pin.
CSDO	O	—	Data Converter Interface serial data output pin.
C1RX C1TX C2RX C2TX	I O I O	ST — ST —	CAN1 bus receive pin. CAN1 bus transmit pin. CAN2 bus receive pin. CAN2 bus transmit pin.
EMUD EMUC EMUD1 EMUC1 EMUD2 EMUC2 EMUD3 EMUC3	I/O I/O I/O I/O I/O I/O I/O I/O	ST ST ST ST ST ST ST ST	ICD Primary Communication Channel data input/output pin. ICD Primary Communication Channel clock input/output pin. ICD Secondary Communication Channel data input/output pin. ICD Secondary Communication Channel clock input/output pin. ICD Tertiary Communication Channel data input/output pin. ICD Tertiary Communication Channel clock input/output pin. ICD Quaternary Communication Channel data input/output pin. ICD Quaternary Communication Channel clock input/output pin.
IC1-IC8	I	ST	Capture inputs 1 through 8.
INDX QEA QEB UPDN	I I I O	ST ST ST CMOS	Quadrature Encoder Index Pulse input. Quadrature Encoder Phase A input in QEI mode. Auxiliary Timer External Clock/Gate input in Timer mode. Quadrature Encoder Phase A input in QEI mode. Auxiliary Timer External Clock/Gate input in Timer mode. Position Up/Down Counter Direction State.
INT0 INT1 INT2 INT3 INT4	I I I I I	ST ST ST ST ST	External interrupt 0. External interrupt 1. External interrupt 2. External interrupt 3. External interrupt 4.
LVDIN	I	Analog	Low Voltage Detect Reference Voltage input pin.

Legend: CMOS = CMOS compatible input or output Analog = Analog input
ST = Schmitt Trigger input with CMOS levels O = Output
I = Input P = Power

CHƯƠNG 5: GIỚI THIỆU VỀ dsPIC6010

Pin Name	Pin Type	Buffer Type	Description
FLTA	I	ST	PWM Fault A input.
FLTB	I	ST	PWM Fault B input.
PWM1L	O	—	PWM 1 Low output.
PWM1H	O	—	PWM 1 High output.
PWM2L	O	—	PWM 2 Low output.
PWM2H	O	—	PWM 2 High output.
PWM3L	O	—	PWM 3 Low output.
PWM3H	O	—	PWM 3 High output.
PWM4L	O	—	PWM 4 Low output.
PWM4H	O	—	PWM 4 High output.
MCLR	I/P	ST	Master Clear (Reset) input or programming voltage input. This pin is an active low Reset to the device.
OCFA	I	ST	Compare Fault A input (for Compare channels 1, 2, 3 and 4).
OCFB	I	ST	Compare Fault B input (for Compare channels 5, 6, 7 and 8).
OC1-OC8	O	—	Compare outputs 1 through 8.
OSC1	I	ST/CMOS	Oscillator crystal input. ST buffer when configured in RC mode; CMOS otherwise.
OSC2	I/O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. Optionally functions as CLK0 in RC and EC modes.
PGD	I/O	ST	In-Circuit Serial Programming data input/output pin.
PGC	I	ST	In-Circuit Serial Programming clock input pin.
RA9-RA10	I/O	ST	PORTA is a bi-directional I/O port.
RA14-RA15	I/O	ST	
RB0-RB15	I/O	ST	PORTB is a bi-directional I/O port.
RC1	I/O	ST	PORTC is a bi-directional I/O port.
RC3	I/O	ST	
RC13-RC15	I/O	ST	
RD0-RD15	I/O	ST	PORTD is a bi-directional I/O port.
RE0-RE9	I/O	ST	PORTE is a bi-directional I/O port.
RF0-RF8	I/O	ST	PORTF is a bi-directional I/O port.
RG0-RG3	I/O	ST	PORTG is a bi-directional I/O port.
RG6-RG9	I/O	ST	
SCK1	I/O	ST	Synchronous serial clock input/output for SPI™ #1.
SDI1	I	ST	SPI #1 Data In.
SDO1	O	—	SPI #1 Data Out.
SS1	I	ST	SPI #1 Slave Synchronization.
SCK2	I/O	ST	Synchronous serial clock input/output for SPI #2.
SDI2	I	ST	SPI #2 Data In.
SDO2	O	—	SPI #2 Data Out.
SS2	I	ST	SPI #2 Slave Synchronization.
SCL	I/O	ST	Synchronous serial clock input/output for I ² C.
SDA	I/O	ST	Synchronous serial data input/output for I ² C.
SOSCO	O	—	32 kHz low power oscillator crystal output.
SOSCI	I	ST/CMOS	32 kHz low power oscillator crystal input. ST buffer when configured in RC mode; CMOS otherwise.

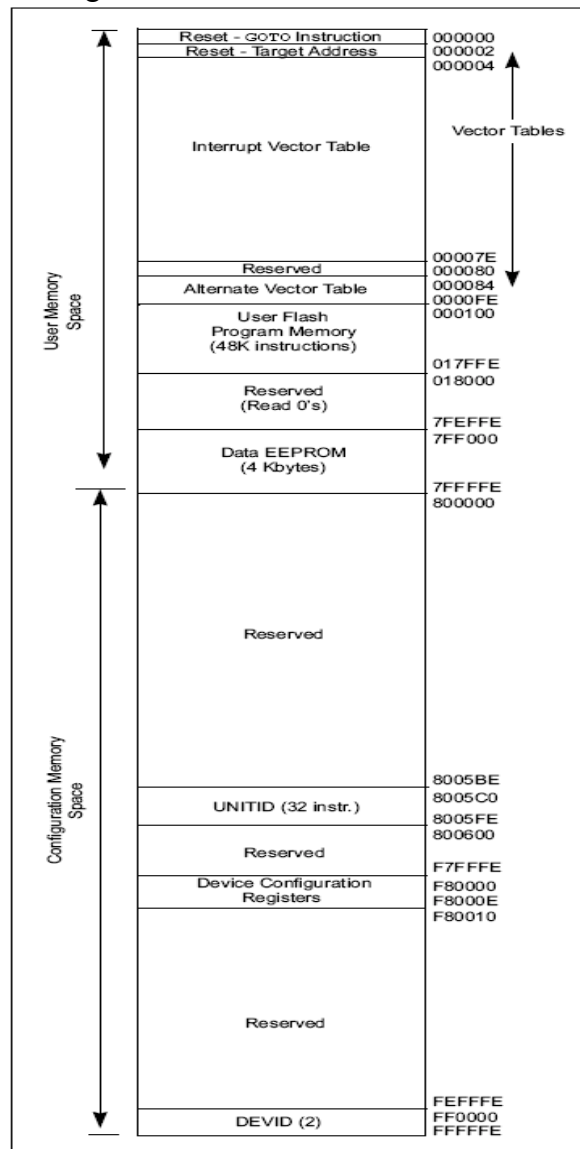
Legend: CMOS = CMOS compatible input or output Analog = Analog input
ST = Schmitt Trigger input with CMOS levels O = Output
I = Input P = Power

CHƯƠNG 5: GIỚI THIỆU VỀ dsPIC6010

Pin Name	Pin Type	Buffer Type	Description
T1CK	I	ST	Timer1 external clock input.
T2CK	I	ST	Timer2 external clock input.
T3CK	I	ST	Timer3 external clock input.
T4CK	I	ST	Timer4 external clock input.
T5CK	I	ST	Timer5 external clock input.
U1RX	I	ST	UART1 Receive.
U1TX	O	—	UART1 Transmit.
U1ARX	I	ST	UART1 Alternate Receive.
U1ATX	O	—	UART1 Alternate Transmit.
U2RX	I	ST	UART2 Receive.
U2TX	O	—	UART2 Transmit.
VDD	P	—	Positive supply for logic and I/O pins.
VSS	P	—	Ground reference for logic and I/O pins.
VREF+	I	Analog	Analog Voltage Reference (High) input.
VREF-	I	Analog	Analog Voltage Reference (Low) input.

Bảng 5.2: Mô tả chức năng, tính chất các I/O trong MCU

Sơ đồ tổ chức bộ nhớ bên trong MCU dsPIC6010:

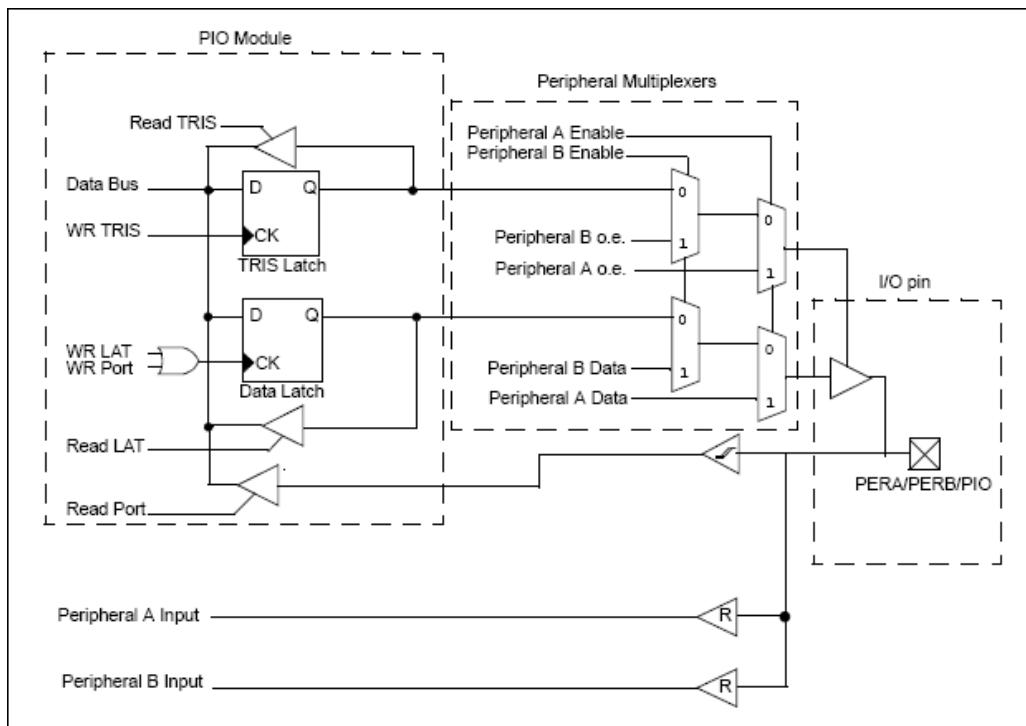


Hình 5.5: Sơ đồ tổ chức bộ nhớ bên trong MCU dsPIC6010

5.4 Khái quát về các thanh ghi làm việc

Tất cả các chân I/O trên MCU (ngoại trừ các chân VDD, VSS, AVDD, AVSS, MCLR, OSC1/CLK1) có thể vừa đóng vai trò là chức năng tổng quát (General Purpose) vừa có đóng vai trò sử dụng đặc biệt. Các chức năng tổng quát cho phép dsPIC30F giám sát và điều khiển các thiết bị khác. Hầu hết các I/O đều được kết hợp nhiều chức năng riêng biệt khác nhau. Sự kết hợp các chức năng này tùy thuộc vào đặc điểm trên loại MCU xác định. Và khi một I/O được sử dụng với chức năng chuyên biệt thì I/O đó có thể sẽ không thể được sử dụng với chức năng tổng quát.

Sơ đồ cấu tạo bên trong thể hiện khả năng kết hợp nhiều chức năng trong cùng một I/O được thể hiện như trong hình



Hình 5.6:Sơ đồ cấu tạo bên trong một I/O

5.4.1 Các thanh ghi điều khiển :

Tất cả các I/O Port trong MCU thuộc họ 12F, 16F, 18F, 30F ... đều có ba thanh ghi trực tiếp liên quan đến phương thức hoạt động của các Port , các thanh ghi đó là TRISx, PORTx, LATx, trong đó x là tên tương ứng của các Port trong MCU. Mỗi I/O pin đều có một bit tham chiếu tương ứng trong ba thanh ghi trên

5.4.2 Thanh ghi TRIS:

Các bits điều khiển trong thanh ghi TRIS xác định trạng thái hoạt động của các I/O là input hay output. Nếu bit TRIS của một I/O là 1 thì I/O đó sẽ đóng vai trò như là một ngõ input, ngược lại nếu bit TRIS của một I/O là 0 thì I/O đó sẽ đóng vai trò như là một ngõ output. Điều cần lưu ý là tất cả các I/O sẽ đóng vai trò là input ngay sau khi MCU bị Reset.

5.4.3 Thanh ghi PORT:

Dữ liệu trên một I/O được truy xuất thông qua thanh ghi PORT, sự kiện đọc thanh ghi PORTx sẽ đọc giá trị của của I/O tương ứng và sự kiện ghi vào thanh ghi PORTx sẽ ghi giá trị vào Port chốt dữ liệu

Một số lệnh như BSET và BCLR là các lệnh cho phép Read-Modify-Write dữ liệu trên các Port. Việc ghi vào một Port nghĩa là các I/O của Port đó sẽ được đọc vào, giá trị đó sẽ được hiệu chỉnh lại, sau đó được ghi vào Port chốt dữ liệu. Một điều cần chú ý là khi các lệnh Read-Modify-Write sử dụng trên một thanh ghi PORTx thì các I/O có liên quan của Port đó phải được cấu hình như là ngõ input. Nếu một I/O được cấu hình như là ngõ input bị chuyển sang cấu hình là output trong khi thực hiện các lệnh Read-Modify-Write thì sẽ dẫn đến những kết quả không mong muốn trên I/O đó.

5.4.4 Thanh ghi LAT:

Thanh ghi LATx liên quan đến một I/O pin hạn chế các sự cố có thể xảy ra đối với các lệnh Read-Modify-Write. Việc đọc thanh ghi LAT sẽ trả về giá trị được cất giữ trong Port chốt ngõ ra (Port output latches), thay vì giá trị trên chân I/O port. Lệnh Read-Modify-Write trên thanh ghi LAT, liên quan đến một I/O, tránh khả năng ghi giá trị của chân input vào Port chốt. Và trình tự ghi vào thanh ghi LATx cũng tương tự như trên.

Sự khác nhau giữa thanh ghi PORT và LAT có thể được tóm tắt như sau:

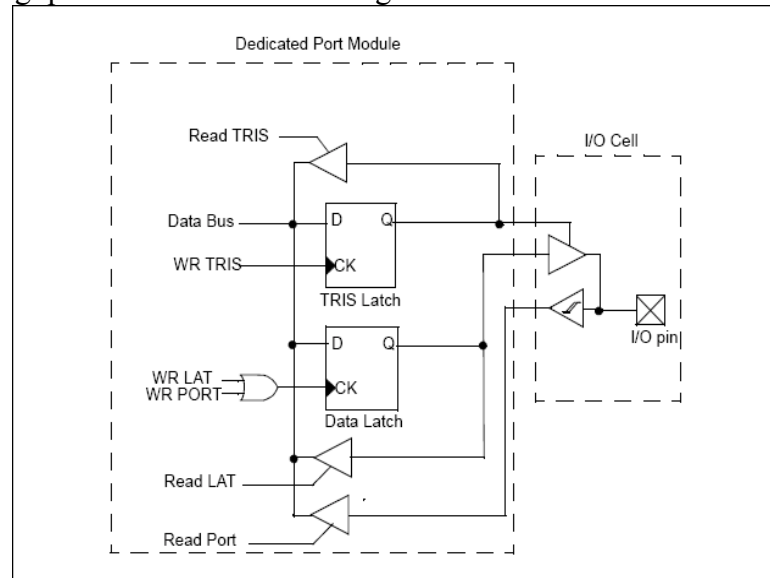
Việc ghi vào thanh ghi PORTx sẽ ghi giá trị dữ liệu vào Port chốt (Port latch)

Việc ghi vào thanh ghi LATx sẽ ghi giá trị dữ liệu vào Port chốt

Việc đọc từ thanh ghi PORTx sẽ đọc giá trị dữ liệu trực tiếp trên chân I/O

Việc đọc từ thanh ghi LATx sẽ đọc giá trị dữ liệu được cất giữ trong Port chốt.

Sơ đồ cấu tạo tổng quan của các I/O Port trong MCU:



Hình 5.7: Sơ đồ cấu tạo tổng quan của các I/O Port trong MCU

5.5 Giới thiệu về các module cơ bản

5.5.1 Module Timer :

MCU dsPIC 6010 cung cấp 5 module timer 16 bits, mỗi module Timer/Counter 16 bit đều có các thanh ghi chỉ đọc(Reable/Writeable Register):

-TMRx: là thanh ghi 16 bit, dùng để lưu giá trị hiện tại của Timer tương ứng.

-PRx: là thanh ghi 16 bit, dùng để nạp giá trị đếm cho Timer.

-TxCON: là thanh ghi điều khiển 16 bit, dùng để thiết lập các thông số điều khiển, chế độ hoạt động của Timer.

Mỗi Module Timer đều có các Bit liên quan dùng cho phục vụ ngắt:

Interrupt Enable Control bit(TxIE): dùng để kích hoạt hoặc ngưng kích hoạt Timer.

Interrupt Flag Status bit(TxIF): Dùng để báo khi tràn bộ đếm.

Interrupt Priority Control bit(TxIP<2:0>): dùng để đặt các mức độ ưu tiên cho ngắt Timer.

Các timer được phân loại thành các loại sau:

5.5.1.1 Module Timer 1

Timer 1 module là một timer 16 bits , có thể được sử dụng làm bộ đếm thời gian cho thời gian thực (Real Time Clock), hoặc có thể được vận hành như là bộ định thì nội bộ (interval timer), bộ đếm (counter), bộ 16-bits Timer1 có các chế độ như sau:

-Chế độ định thì 16 bits (16-bits Timer): ở chế độ này bộ định thì sẽ tăng giá trị sau mỗi chu kỳ máy cho đến khi bằng với một giá trị được nạp trước vào thanh ghi PR1, sau đó sẽ reset trở về giá trị 0 và tiếp tục đếm. Khi CPU đi vào trạng thái nghỉ, Timer1 sẽ ngưng việc tăng giá trị trừ khi bit TSIDL(T1CON<13>)=0

-Chế độ bộ đếm đồng bộ 16 bits (16-bits Synchronous Counter): ở chế độ này Timer1 sẽ tăng giá trị ở mỗi cạnh lên của tín hiệu xung clock từ bên ngoài đưa vào MCU và được đồng bộ hoá với xung clock bên trong MCU. Timer1 sẽ đếm lên đến một giá trị được nạp trước vào thanh ghi PR1, sau đó sẽ reset trở về giá trị 0 và tiếp tục đếm. Khi CPU đi vào trạng thái nghỉ, Timer1 sẽ ngưng việc tăng giá trị trừ khi bit TSIDL(T1CON<13>)=0

-Chế độ bộ đếm bất đồng bộ 16 bits (16-bits Asynchronous Counter): ở chế độ này Timer1 sẽ tăng giá trị ở mỗi cạnh lên của tín hiệu xung clock từ bên ngoài đưa vào MCU .Timer1 sẽ đếm lên đến một giá trị được nạp trước vào thanh ghi PR1, sau đó sẽ reset trở về giá trị 0 và tiếp tục đếm. Khi CPU đi vào trạng thái nghỉ, Timer1 sẽ ngưng việc tăng giá trị trừ khi bit TSIDL(T1CON<13>)=0

Các đặc điểm của module Timer1:

Hoạt động của Timer ở trạng thái ngủ:

Timer1 vẫn hoạt trong lúc CPU trong trạng thái nghỉ nếu:

- Timer được kích hoạt (TON=1)
- Nguồn xung clock cho Timer được chọn là nguồn ngoài (TCS=1)
- Bit TSYNC (T1CON<2>) được đặt ở mức logic 0 mà được định nghĩa là nguồn xung clock ngoài bất đồng bộ

Nếu cả 3 điều kiện trên được thỏa mãn thì Timer1 sẽ đếm lên đến giá trị trong thanh ghi PR1 và sau đó được reset về giá trị 0x0000.

16-bit TIMER có khả năng tạo ra ngắt . Khi giá trị của Timer bằng với giá trị trong thanh ghi PR1 thì bit T1IF sẽ được tác động và tạo ra ngắt. Và nếu được kích hoạt , bit T1IF phải được xóa bằng phần mềm. Cờ ngắt Timer T1IF được đặt trong thanh ghi điều khiển IFS0 .

16-bits Timer1 có thể được đặt ở chế độ Gated Timer Accumulation. Ở chế độ này cho phép xung klok nội làm tăng giá trị của Timer1 khi cộng tín hiệu vào (T1CK) được đặt lên mức tích cực (high). Bit điều khiển TGATE(T1CON<6>) phải được set lên (TON=1). TIMER1 phải được kích hoạt (TON=1), và nguồn xung clock của Timer phải là chế độ nguồn nội (TCS=0).

Xung đưa vào (là Fosc/4 hoặc là xung clock ngoài) vào 16 bit TIMER , có thể được tỉ lệ 1:1,1:8,1:64,1:256 bởi việc thiết lập các bit điều khiển TCKPS<1:0>(T1CON<5:4>). Bộ đếm tỉ lệ (Prescaler Counter) sẽ được xoá khi xảy ra các trường hợp sau:

- Tác động ghi vào thanh ghi TMR1
- Tác động xoá bit TON(T1CON<15>
- Các tác động Reset MCU như là POR (Power on reset), BOR(Brown out Reset)

TMR1 sẽ không bị xoá khi T1CON được ghi vào. Nó sẽ bị xoá bằng việc ghi vào thanh ghi TMR1

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State
TMR1	0100	Timer 1 Register																xxxx xxxx xxxx xxxx
PR1	0102	Period Register 1																1111 1111 1111 1111
T1CON	0104	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TKCP51	TKCP50	—	TSYNC	TCS	—	0000 0000 0000 0000

[illegible]

53

5.5.1.2 Timer2/3 module:

Module Timer 2/3 là một 32-bit Timer và có thể được thiết lập thành hai Timer 16 bits riêng biệt là Timer2 và Timer3, trong đó Timer2 thuộc loại Timer B(typeB) và Timer 3 thuộc loại Timer C (type C), Timer2/3 có các chế độ như sau:

-*Chế độ hai Timer 16 bit độc lập (Two Independent 16 bit Timer(Timer2 và Timer3))* :với tất cả các chế độ hoạt động như các Timer 16 bit khác (trừ chế độ đếm bất đồng bộ (Asynchronous Counter))

-*Chế độ một Timer 32 bit (Single 32-bit Timer)*:

Khi hoạt động trong chế độ này, bit điều khiển T32 phải được set trong thanh ghi T3CON. Khi hai timer (Timer 2 và Timer 3) được cấu hình hoạt động như một Timer 32 bit , thanh ghi T3CON được bỏ qua, chỉ có thanh ghi T2CON đóng vai trò là thanh ghi điều khiển. Sự kiện ngắt Timer xảy ra với cờ ngắt T3IF. Timer 2 đóng vai trò là 16 bit thấp, và Timer 3 đóng vai trò là 16 bit cao trong Timer 32 bit. Timer 32 bit sẽ tăng giá trị cho đến khi bằng với giá trị nạp vào 2 thanh ghi PR3:PR2 , sau đó được reset về 0 và tiếp tục đếm lên trong lần xung clock tiếp theo . Giá trị tối đa nạp vào cặp thanh ghi PR3:PR2 là 0xFFFFFFFF. Nếu được kích hoạt, một sự kiện ngắt sẽ tạo ra khi bộ đếm có giá trị bằng với giá trị trong cặp thanh ghi PR3:PR2 .

Các bước thiết lập module Timer 32 bit:

TON(T2CON<15>)=1

T32(T2CON<3>)=1

TCKPS<1:0>(T2CON<5:4>) được sử dụng để cấu hình cho bộ đếm tỉ lệ (Prescaler)của Timer 2

Cấp thanh ghi TMR3:TMR2 chứa giá trị của Timer 32 bit

Cấp thanh ghi PR3:PR2 chứa giá trị dùng để so sánh với cặp thanh ghi TMR3:TMR2

T3IE(IEC0<7>) dùng để kích hoạt khả năng tạo sự kiện ngắt Timer

T3IF(IFS0<7>) được sử dụng như cờ báo ngắt

T3IP<2:0>(IPC<14:12>) dùng để thiết đặt cấp độ ngắt ưu tiên cho Timer 32 bit

-*Chế độ bộ đếm đồng bộ 32 bit (Single 32-bit Counter)*

Các đặc điểm của module Timer2/3

Sự kiện kích chuyển đổi AD:

Khi giá trị giữa hai thanh ghi 32 bit TMR3/TMR2 và thanh ghi kết hợp PR3/PR2 bằng nhau thì một tín hiệu kích ADC sẽ được tạo ra bởi Timer3

Timer Gate Operation:

32-bit Timer có thể được đặt ở chế độ Gated Time Accumulation, Ở chế độ này cho phép xung clock nội làm tăng giá trị của Timer khi cổng tín hiệu vào (T2CK) được đặt lên mức cao.Bit điều khiển TGATE(T2CON<6>) phải được set để kích hoạt trong chế độ này. Và khi đang hoạt động trong chế độ này thì Timer2 sẽ là nguồn xung clock . Việc set bit TGATE sẽ được bỏ qua cho Timer3. Timer phải được kích hoạt (TON=1)và nguồn xung clock phải là nguồn nội (TCS=0)

Cạnh xuống của tín hiệu từ bên ngoài sẽ kết thúc việc đếm, người lập trình phải reset timer để bắt đầu đếm từ giá trị 0.

CHƯƠNG 5: GIỚI THIỆU VỀ dsPIC6010

Cấu hình cho bộ đếm tỉ lệ:

Xung đưa vào (là $F_{osc}/4$ hoặc là xung clock ngoài) vào TIMER , có thể được tỉ lệ 1:1,1:8,1:64,1:256 bởi việc thiết lập các bit điều khiển TCKPS<1:0>(T2CON<5:4>) và (T3CON<5:4>).Đối với Timer 32 bit , Timer2 sẽ đóng vai trò là nguồn xung clock, và việc lựa chọn các tỉ lệ sẽ không được áp dụng trên Timer3. Bộ đếm tỉ lệ sẽ bị xóa khi xảy ra các trường hợp sau:

-Tác động ghi vào thanh ghi TMR2/TMR3.

-Tác động xóa một trong hai bit TON(T2CON<15>) hoặc T3CON<15> về 0.

-Các tác động reset MCU như POR, BOR.

Tuy nhiên , nếu Timer chưa được kích hoạt thì bộ tỉ lệ Timer2 không thể được reset khi xung clock Prescaler được giữ

TMR2/TMR3 sẽ không bị xóa khi T2CON/T3CON được ghi vào.

Hoạt động của Timer trong trạng thái idle và trạng thái ngủ:

Khi CPU đang trong trạng thái ngủ (sleep mode) thì timer sẽ không hoạt động vì nguồn xung clock nội bị vô hiệu hoá

Sự kiện ngắt Timer:

32-bit TIMER có khả năng tạo ra ngắt khi giá trị của Timer bằng với một giá trị định trước được nạp vào thanh ghi 32 bit PR2/PR3 hoặc khi phát hiện được có cạnh xuống từ cổng tín hiệu bên ngoài.Bit T3IF (IFS0<7>) sẽ được tác động và tạo ra ngắt.Trong chế độ này, T3IF được sử dụng như là nguồn tạo ngắt và bit T3IF phải được xóa bằng phần mềm. Việc kích hoạt hoạt động của ngắt được thực hiện thông qua bit T3IE<ICE0<7>)(Timer Interrupt Enable)

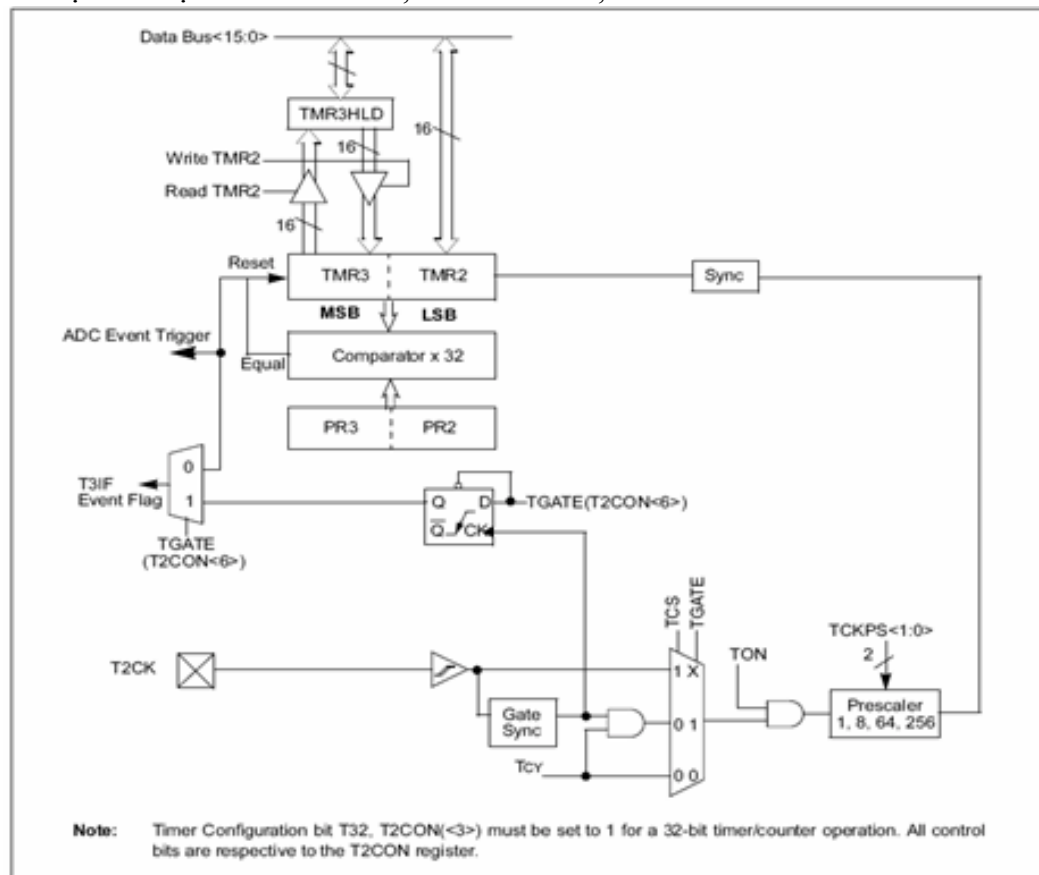
Bảng 5.4: Trình bày các thanh ghi điều khiển Timer2/3

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State
TMR2	0106	Timer2 Register																uuuu uuuu uuuu uuuu
TMR3HLD	0108	Timer3 Holding Register (For 32-bit timer operations only)																uuuu uuuu uuuu uuuu
TMR3	010A	Timer3 Register																uuuu uuuu uuuu uuuu
PR2	010C	Period Register 2																1111 1111 1111 1111
PR3	010E	Period Register 3																1111 1111 1111 1111
T2CON	0110	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	T32	—	TCS	—	0000 0000 0000 0000
T3CON	0112	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	—	—	TCS	—	0000 0000 0000 0000

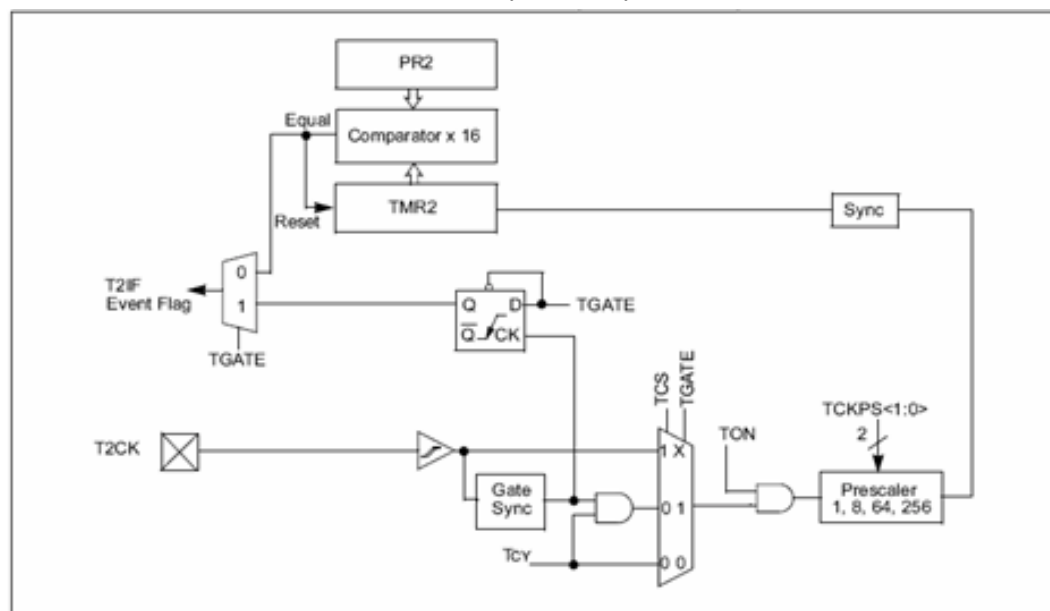
Legend: u = uninitialized bit

CHƯƠNG 5: GIỚI THIỆU VỀ dsPIC6010

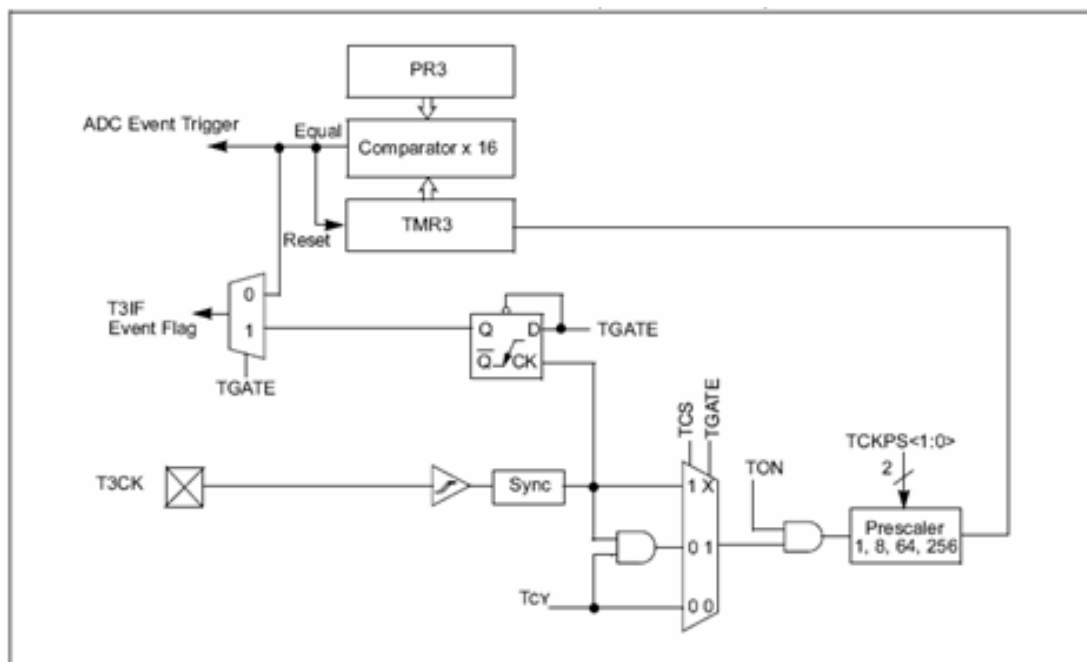
Sơ đồ cấu tạo của bộ 32-bit Timer2/3, 16-bit Timer2, 16-bit Timer3 như các hình vẽ sau:



Hình 5.9: Sơ đồ cấu tạo của bộ 32-bit Timer2/3



Hình 5.10: Sơ đồ cấu tạo của bộ 16-bit Timer2 (Timer loại B)



Hình 5.11: Sơ đồ cấu tạo của bộ 16-bit Timer3 (Timer loại C)

5.5.1.3 Timer4/5 module :

Module Timer 4/5 là một Timer 32 bit tổng quát và có thể phân thành hai module Timer 16 bit là Timer4, Timer5

Timer 4/5 hoạt động tương tự như Timer 2/3, Tuy nhiên có một số khác biệt sau:

- Timer4/5 không hỗ trợ đặt điểm ADC Even Trigger như ở Timer2/3
- Timer4/5 không được sử dụng bởi các module ngoại vi khác như Input Capture, Outpy Compare

Hoạt động của Timer4/5 được xác định bởi việc thiết lập các bit trong các thanh ghi đặt biệt như T4CON, T5CON

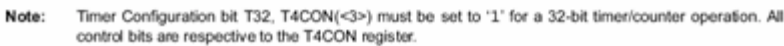
Trong chế độ Timer4/5, Timer4 đóng vai trò là 16 bit thấp (LS Word), Timer5 đóng vai trò là 16 bit cao (MS Word)

Điều cần chú ý là khi hoạt động ở chế độ Timer 32 bit, thanh ghi T5CON được bỏ qua, chỉ có thanh ghi T4CON là được sử dụng cho việc cấu hình hoạt động của Timer. Xung clock của Timer4 và cổng vào được sử dụng cho module 32 bit Timer, nhưng sự kiện ngắt Timer4/5 là do cờ ngắt T5IF quyết định, và chức năng ngắt sẽ được kích hoạt thông qua bit T5IE

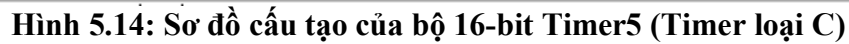
Bảng 5.5: Trình bày các thanh ghi điều khiển Timer4/5

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State
TMR4	0114	Timer 4 Register																uuuu uuuu uuuu uuuu
TMR5HLD	0116	Timer 5 Holding Register (For 32-bit operations only)																uuuu uuuu uuuu uuuu
TMR5	0118	Timer 5 Register																uuuu uuuu uuuu uuuu
PR4	011A	Period Register 4																1111 1111 1111 1111
PR5	011C	Period Register 5																1111 1111 1111 1111
T4CON	011E	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	T45	—	TCS	—	0000 0000 0000 0000
T5CON	0120	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	—	—	TCS	—	0000 0000 0000 0000

Legend: — = unimplemented bit



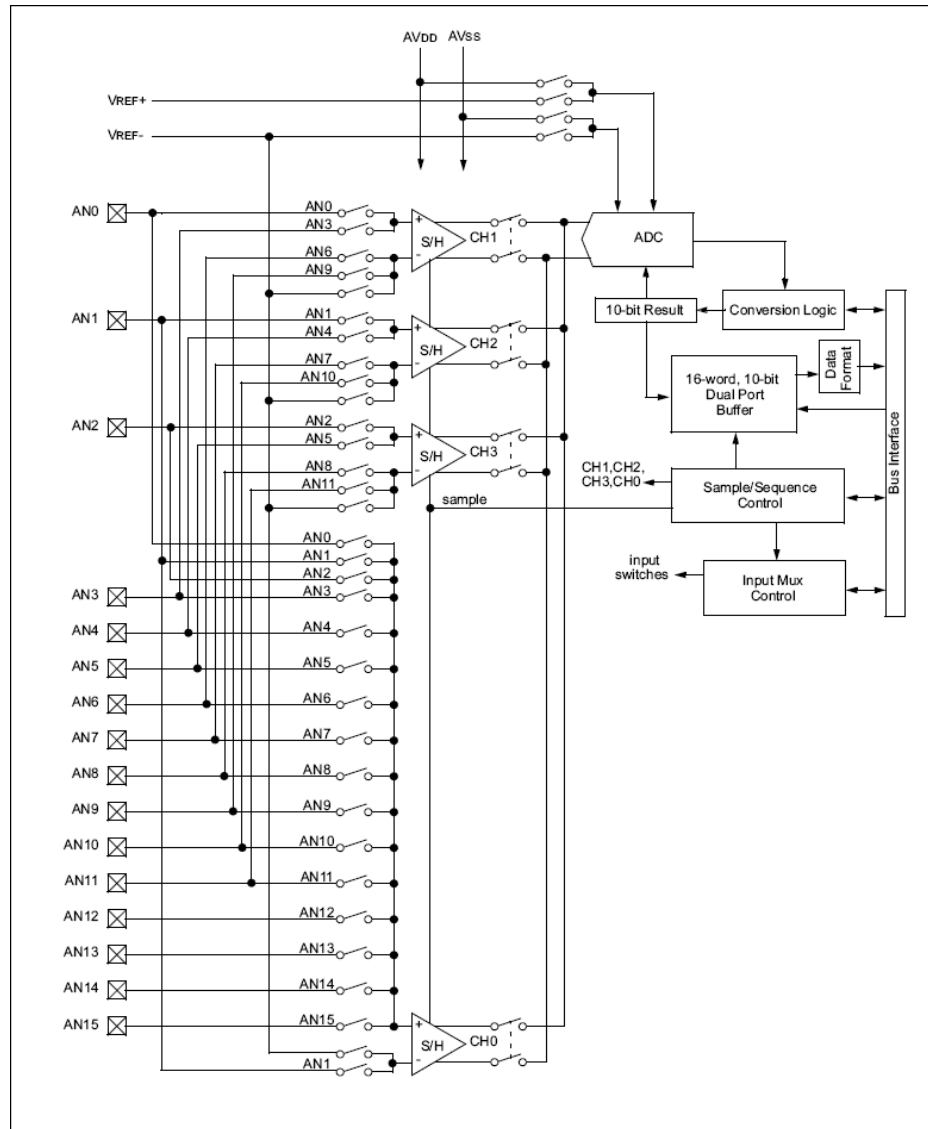




Bộ chuyển đổi 10 bit AD tốc độ cao cho phép chuyển đổi tín hiệu vào tuần tự sang 10 bit tín hiệu số (10-bit high-speed analog to digital converter). Module AD này dựa trên kiến trúc Successive Approximation Register (SAR) và cung cấp khả năng lấy mẫu tối đa 500kps. Module AD có 16 ngõ vào analog . Điện áp tham chiếu cho module A/D được lựa chọn bằng phần mềm ,có thể là chính nguồn cung cấp cho MCU (AVDD/AVSS) hoặc mức điện áp trên các chân (VREF+/VREF-) . Bộ chuyển đổi AD có khả năng hoạt động khi CPU đang trong trạng thái ngủ (Sleep Mode)

- A/D Control Register1(ADCON1)
- A/D Control Register2(ADCON2)
- A/D Control Register3(ADCON3)
- A/D Input Select Register (ADCHS)
- A/D Port Configuration Register(ADPCFG)
- A/D Input Scan Selection Register(ADCSSL)

Điều cần lưu ý là các bit SSRC<2:0>, ASAM, SIMSAM, SMPI<3:0> cũng như các thanh ghi ADCON3, ADCSSL không thể được ghi vào trong khi bit ADON=1, Điều này sẽ ảnh hưởng đến kết quả chuyển đổi.



Hình 5.15: Sơ đồ cấu tạo bên trong module A/D

5.5.2.1 Giải thích hoạt động

Module bao gồm bộ nhớ đệm kép 16-word (16-word dual port read-only buffer), đây là bộ nhớ chỉ đọc dùng để chứa giá trị của việc chuyển đổi AD. Nội dung của 16 thanh ghi trong bộ nhớ đệm (từ ADCBUF0 đến ADCBUFF) không thể được ghi vào bởi người sử dụng. Sau khi Module ADC được thiết lập xong, việc thu thập mẫu được bắt đầu bằng việc set bit SAMP. Một số tác động như là các bit lập trình (Programmable bit), sự kiện ngoài (external events), tràn timer (timer time-out), sẽ kết thúc việc thu thập mẫu và bắt đầu quá trình chuyển đổi. Khi sự chuyển đổi hoàn tất, kết quả sẽ được đưa vào các thanh ghi trong bộ nhớ đệm từ ADCBUF0...ADCBUFF. Cờ báo ADIF và bit DONE sẽ được set sau một số lần lấy mẫu được xác định bởi bit SMPI.

5.5.2.2 Quá trình hoạt động của module ADC được tóm tắt như các bước sau:

1. Thiết lập cho module ADC

- Thiết lập các chân analog, điện áp tham chiếu, và các digital I/O
- Lựa chọn kênh vào
- Lựa chọn xung clock dùng cho việc chuyển đổi

- Lựa chọn tác động kích cho việc chuyển đổi
- Kích hoạt quá trình chuyển đổi
- 2.Thiết lập các ngắt AD (nếu cần thiết):
- Xóa bit ADIF
- Chọn các ưu tiên ngắt
- 3.Bắt đầu việc lấy mẫu
- 4.Chờ cho quá trình lấy mẫu thành công
- 5.Việc thu thập mẫu kết thúc, bắt đầu quá trình chuyển đổi
- 6.Chờ cho việc chuyển đổi hoàn tất bằng cách:
- Chờ cho đến khi ngắt AD xảy ra
- 7. Đọc giá trị từ bộ nhớ đệm , xóa cờ ADIF nếu cần thiết

5.5.2.3 Các sự kiện kích chuyển đổi:

Các kích chuyển đổi sẽ kết thúc quá trình thu thập mẫu và bắt đầu quá trình chuyển đổi. Việc lựa chọn các nguồn kích chuyển đổi được qui định bởi bit SSRC<2:0>, Bit SSRC<2:0> cung cấp cho người dùng 5 lựa chọn cho việc kích chuyển đổi. Khi SSRC<2:0>=000, việc kích chuyển đổi được điều khiển bằng phần mềm. Việc xóa bit SAMP sẽ tạo nên một kích chuyển đổi. Khi bit SSRC<2:0> =111, (Chế độ tự động) , việc kích chuyển đổi được điều khiển bằng xung clock của module A/D. Bit SAMC sẽ lựa chọn số lượng xung clock giữa các lần thu thập mẫu và chuyển đổi. Điều này tạo khả năng chuyển đổi nhanh nhất ở chế độ đa kênh (Multiple Channels) . Số lượng xung clock giữa quá trình chuyển đổi và thu thập mẫu phải luôn luôn ít nhất là 1 chu kỳ xung clock. Các nguồn kích chuyển đổi khác sẽ là từ các module Timer, module PWM, và các nguồn ngắt ngoài (external interrupt). Điều cần chú ý là để module A/D có thể hoạt động được ở tốc độ tối đa , chế độ chuyển đổi tự động nên được lựa chọn SSRC<2:0>=111, và số xung clock giữa quá trình chuyển đổi và thu thập mẫu nên được cài đặt là 1 TAD (SAMC=00001). Việc cấu hình như thế sẽ được thời gian tổng cộng bao gồm việc thu thập mẫu và chuyển đổi là 13 TAD

5.5.2.4 Tác động reset

Các tác động reset CPU sẽ làm tắt cả các thanh ghi trở về trạng thái reset ban đầu. Điều này làm cho module A/D sẽ bị tắt, và tất cả các quá trình chuyển đổi và thu thập mẫu bị kết thúc. Giá trị trong các thanh ghi ADCBUF sẽ không được cập nhật. Các thanh ghi kết quả sẽ bao gồm các dạng dữ liệu không xác định khi CPU hoạt động trở lại

5.5.2.5 Định dạng kiểu dữ liệu trong module A/D

Kết quả chuyển đổi từ module A/D có chiều dài 10 bit. bộ nhớ đệm RAM cũng có chiều dài 10 bit. 10 bit dữ liệu này có thể được đọc ở 1 trong 4 kiểu định dạng được trình bày như hình 33. Bit FORM<1:0> sẽ qui định kiểu định dạng của kết quả chuyển đổi thu được . Và các kết quả này được chuyển sang kiểu kết quả dạng 16 bit trên bus dữ liệu. Việc ghi dữ liệu vào luôn luôn sẽ là kiểu định dạng canh phải (Right Justify)

RAM Contents:	<table><tr><td>d09</td><td>d08</td><td>d07</td><td>d06</td><td>d05</td><td>d04</td><td>d03</td><td>d02</td><td>d01</td><td>d00</td></tr></table>										d09	d08	d07	d06	d05	d04	d03	d02	d01	d00			
d09	d08	d07	d06	d05	d04	d03	d02	d01	d00														
Read to Bus:																							
Signed Fractional (1.15)	<table><tr><td>$\overline{\text{d09}}$</td></tr></table>	$\overline{\text{d09}}$	d08	d07	d06	d05	d04	d03	d02	d01	d00	0	0	0	0	0	0						
$\overline{\text{d09}}$																							
Fractional (1.15)	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00	0	0	0	0	0	0							
Signed Integer	<table><tr><td>$\overline{\text{d09}}$</td></tr></table>	$\overline{\text{d09}}$	<table><tr><td>$\overline{\text{d09}}$</td></tr></table>	$\overline{\text{d09}}$	<table><tr><td>$\overline{\text{d09}}$</td></tr></table>	$\overline{\text{d09}}$	<table><tr><td>$\overline{\text{d09}}$</td></tr></table>	$\overline{\text{d09}}$	<table><tr><td>$\overline{\text{d09}}$</td></tr></table>	$\overline{\text{d09}}$	<table><tr><td>$\overline{\text{d09}}$</td></tr></table>	$\overline{\text{d09}}$	<table><tr><td>$\overline{\text{d09}}$</td></tr></table>	$\overline{\text{d09}}$	d08	d07	d06	d05	d04	d03	d02	d01	d00
$\overline{\text{d09}}$																							
$\overline{\text{d09}}$																							
$\overline{\text{d09}}$																							
$\overline{\text{d09}}$																							
$\overline{\text{d09}}$																							
$\overline{\text{d09}}$																							
$\overline{\text{d09}}$																							
Integer	0	0	0	0	0	0	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00							

Bảng 5.6: Định dạng kiểu lưu trữ kết quả

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State					
ADCBUF0	0280	—	—	—	—	—	—						ADC Data Buffer 0				0000	0000	0000	0000			
ADCBUF1	0282	—	—	—	—	—	—						ADC Data Buffer 1				0000	0000	0000	0000			
ADCBUF2	0284	—	—	—	—	—	—						ADC Data Buffer 2				0000	0000	0000	0000			
ADCBUF3	0286	—	—	—	—	—	—						ADC Data Buffer 3				0000	0000	0000	0000			
ADCBUF4	0288	—	—	—	—	—	—						ADC Data Buffer 4				0000	0000	0000	0000			
ADCBUF5	028A	—	—	—	—	—	—						ADC Data Buffer 5				0000	0000	0000	0000			
ADCBUF6	028C	—	—	—	—	—	—						ADC Data Buffer 6				0000	0000	0000	0000			
ADCBUF7	028E	—	—	—	—	—	—						ADC Data Buffer 7				0000	0000	0000	0000			
ADCBUF8	0290	—	—	—	—	—	—						ADC Data Buffer 8				0000	0000	0000	0000			
ADCBUF9	0292	—	—	—	—	—	—						ADC Data Buffer 9				0000	0000	0000	0000			
ADCBUFA	0294	—	—	—	—	—	—						ADC Data Buffer 10				0000	0000	0000	0000			
ADCBUFB	0296	—	—	—	—	—	—						ADC Data Buffer 11				0000	0000	0000	0000			
ADCBUFC	0298	—	—	—	—	—	—						ADC Data Buffer 12				0000	0000	0000	0000			
ADCBUFD	029A	—	—	—	—	—	—						ADC Data Buffer 13				0000	0000	0000	0000			
ADCBUFE	029C	—	—	—	—	—	—						ADC Data Buffer 14				0000	0000	0000	0000			
ADCBUFF	029E	—	—	—	—	—	—						ADC Data Buffer 15				0000	0000	0000	0000			
ADCON1	02A0	ADON	—	—	ADSIDL	—	—	—	FORM<1:0>		SSRC<2:0>			—	SIMSAM	ASAM	SAMP	DONE	0000	0000	0000	0000	
ADCON2	02A2	VCFG<2:0>				—	—	CSCNA	CHPS<1:0>		BUFS	—	SMPI<3:0>					BUFM	ALTS	0000	0000	0000	0000
ADCON3	02A4	—	—	—	SAMC<4:0>				ADRC	—	ADCS<5:0>									0000	0000	0000	0000
ADCHS	02A6	CH123NB<1:0>			CH123SB	CH0NB	CH0SB<3:0>					CH123NA<1:0>		CH123SA	CH0NA	CH0SA<3:0>				0000	0000	0000	0000
ADPCFG	02A8	PCFG15	PCFG14	PCFG13	PCFG12	PCFG11	PCFG10	PCFG9	PCFG8	PCFG7	PCFG6	PCFG5	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0	0000	0000	0000	0000		
ADCSSL	02AA	CSSL15	CSSL14	CSSL13	CSSL12	CSSL11	CSSL10	CSSL9	CSSL8	CSSL7	CSSL6	CSSL5	CSSL4	CSSL3	CSSL2	CSSL1	CSSL0	0000	0000	0000	0000		

Bảng 5.7: Bảng thanh ghi điều khiển module AD

5.5.3 Module PWM:

Module PWM được sử dụng để tạo ra các tín hiệu xung đồng bộ có khả năng điều chỉnh được độ rộng (Synchronized Pulse Width Modulated) . Được ứng dụng trong các mục đích điều khiển chuyển động và điều khiển công suất

Module PWM có các ứng dụng phổ biến sau:

Sử dụng phổ biến trong điều khiển động cơ xoay chiều 3 pha (Three Phase AC Induction Motor)

Sử dụng trong các thiết bị dùng để lưu trữ điện năng dùng để cung cấp năng lượng khi mất điện (Uninterruptable Power Supply)

Sử dụng trong điều khiển động cơ một chiều không chổi than (Brushless DC Motor)

5.5.3.1 Các đặc điểm của module PWM

Có 8 ngõ tín hiệu ra PWM với 4 bộ tạo chu kì PWM

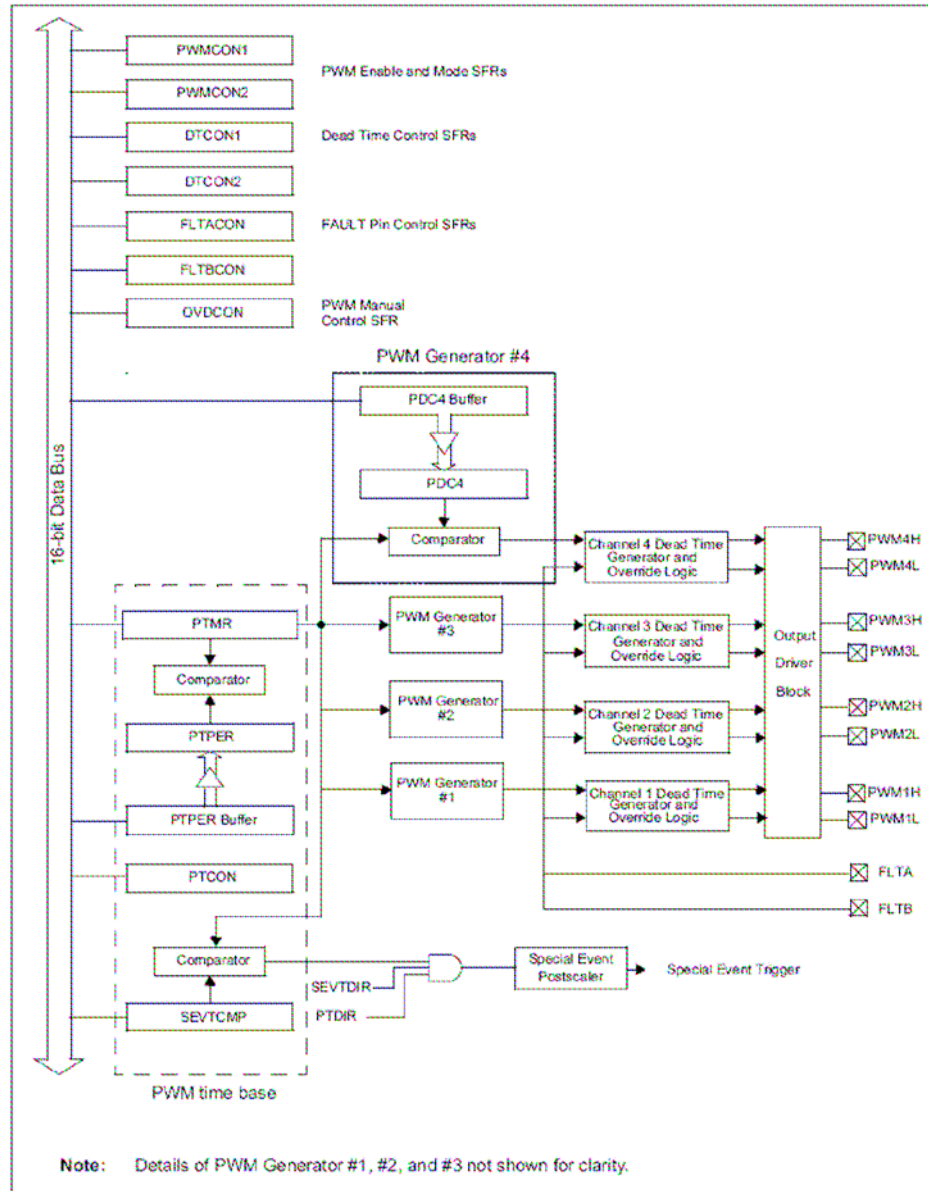
Có độ phân giải lên đến 16 bit

Có khả năng thay đổi tần số tín hiệu PWM khi module đang hoạt động

Có các chế độ canh giữa ,canh cạnh (Edgle and Center Aligned output)

Có thể vận hành ở chế độ độc lập, nghĩa là tín hiệu ở mỗi kênh PWM sẽ hoàn toàn độc lập với nhau

Sơ đồ cấu tạo



Hình 5.16: Sơ đồ cấu tạo bên trong module PWM

5.5.3.2 Giải thích hoạt động của module PWM

Module PWM có thể được cấu hình để hoạt động ở 4 chế độ vận hành khác nhau gồm:

- Free Running Mode
- Single Shot Mode
- Continuous Up/Down Counting Mode
- Double Update Mode

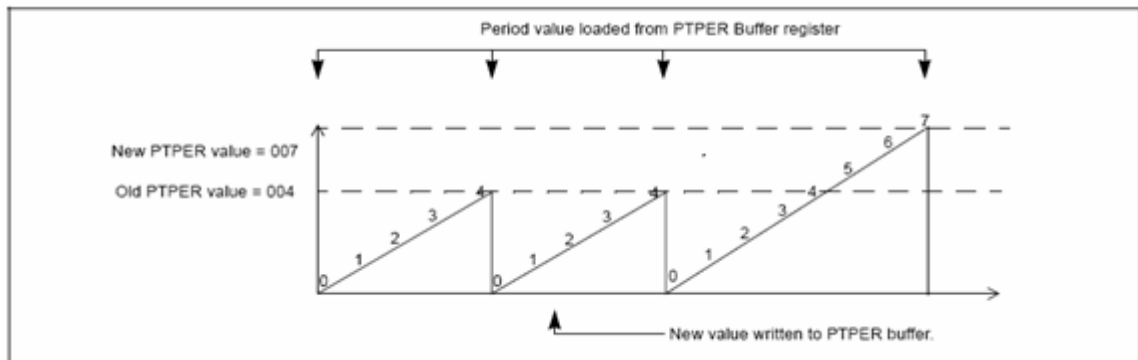
Bốn chế độ hoạt động này được lựa chọn bởi bit PTMOD<1:0> trong thanh ghi PTCN. Các sự kiện ngắt được tạo ra bởi bộ đếm thời gian PWM phụ thuộc vào bit (PTMOD<1:0>) và bit Postscaler (PTOPS<3:0>) trong thanh ghi PTCN.

Các chế độ hoạt động của module PWM:

Chế độ tự do (Free Running Mode)

Trong chế độ Free Running bộ đếm thời gian trong module PWM(PWM time base) sẽ đếm lên cho đến khi nào bằng với giá trị trong thanh ghi PTPER. Thanh ghi PTMR sẽ reset vào lần xung clock kế tiếp và bộ đếm thời gian sẽ tiếp tục đếm lên nếu bit PTEN vẫn còn được set

Trong khi bộ đếm thời gian của module PWM trong chế độ Free Running ($PTMOD<1:0>=00$), một sự kiện ngắt sẽ được tạo ra mỗi lần giá trị của bộ đếm trùng với giá trị trong thanh ghi PTPER và thanh ghi PTMR sẽ được reset về 0. Bit lựa chọn Postscaler nên được chọn trong chế độ này để giảm bớt số lần sự kiện ngắt xảy ra



Hình 5.17 : Cập nhật giá trị PWM trong chế độ tự do

Chế độ đơn (Single Shot Mode)

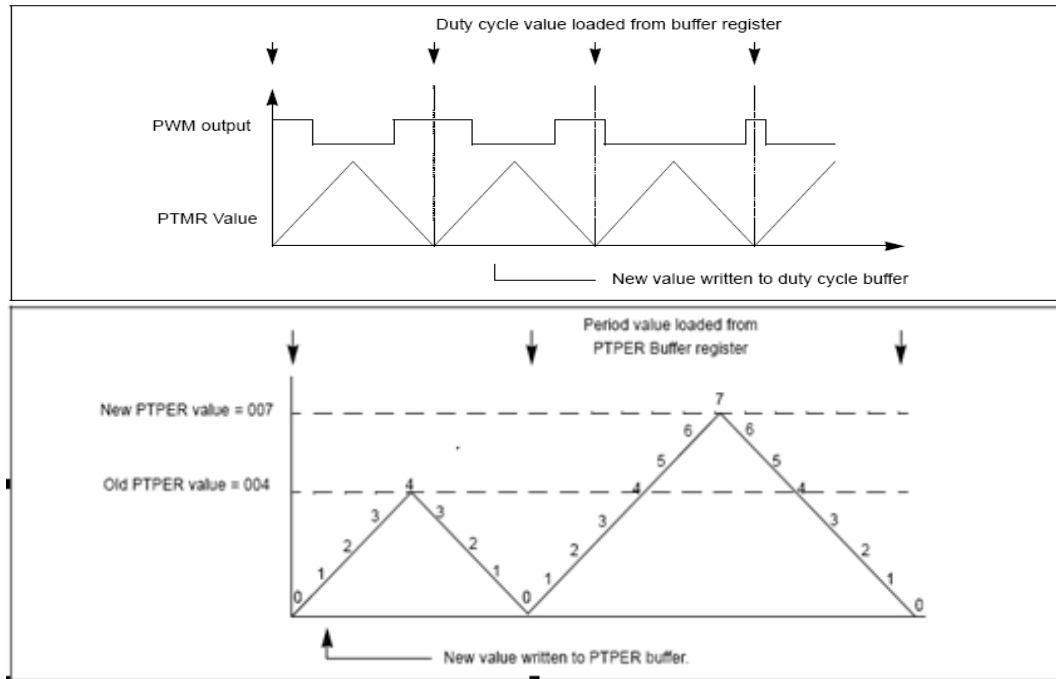
Trong chế độ Single Shot , bộ đếm thời gian của module PWM sẽ đếm lên khi bit PTEN được set. Khi giá trị trong thanh ghi PTMR bằng với giá trị trong thanh ghi PTPER, thanh ghi PTMR sẽ được reset reset tron lần xung clock kế tiếp, và thanh ghi PTEN sẽ bị xóa bởi phản cứng để tạm dừng lại bộ đếm thời gian.

Trong khi bộ đếm thời gian của module PWM trong chế độ Single Shot ($PTMOD<1:0>=01$), một sự kiện ngắt sẽ được tạo ra mỗi lần giá trị của bộ đếm trùng với giá trị trong thanh ghi PTPER và thanh ghi PTMR sẽ được reset về 0 , bit PTEN cũng sẽ được reset. Bit lựa chọn Postscaler không có tác dụng trong chế độ này

Chế độ đếm lên xuống (Continous Up/Down Counting Mode)

Trong chế độ Continous Up/Down Counting bộ đếm thời gian trong module PWM(PWM time base) sẽ đếm lên cho đến khi nào bằng với giá trị trong thanh ghi PTPER. Sau đó Timer sẽ bắt đầu đếm xuống trong lần xung clock tiếp theo. Bit PTDIR trong thanh ghi PTCN cho biết Timer đang đếm lên hay đếm xuống. Bit PTDIR sẽ được set khi timer bắt đầu đếm xuống.

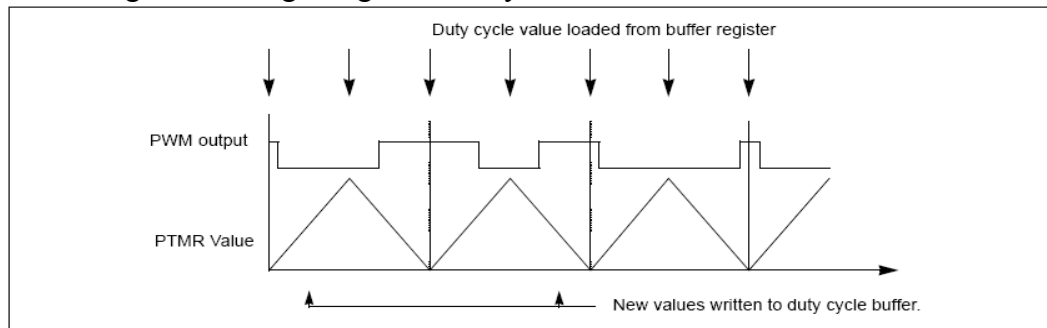
Trong chế độ này ($PTMOD<1:0>=10$) một sự kiện ngắt sẽ xảy ra mỗi khi giá trị của thanh ghi PTMR bằng 0 và bộ đếm thời gian PWM bắt đầu đếm lên. Bit lựa chọn Postscaler nên được chọn trong chế độ này để giảm bớt số lần sự kiện ngắt xảy ra



Hình 5.18 : Cập nhật giá trị PWM trong chế độ đếm lên xuống

Chế độ cập nhật kép (Double Update Mode)

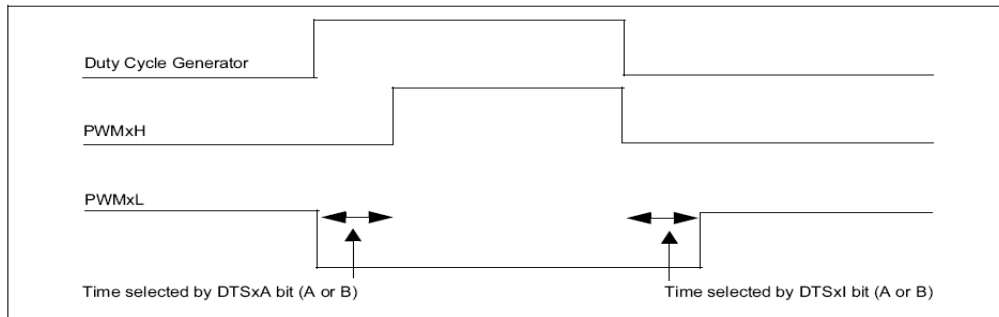
Trong chế độ Double Update (PTMOD<1:0>=11) một sự kiện ngắt sẽ xảy ra mỗi khi giá trị trong thanh ghi PTMR bằng 0, và mỗi khi bằng với giá trị trong thanh ghi PTPER. Trong chế độ này chu kỳ PWM sẽ được cập nhật 2 lần trong một chu kỳ. Bit lựa chọn Postscaler không có tác dụng trong chế độ này



Hình 5.19 : Cập nhật giá trị PWM trong chế độ cập nhật kép

Chế độ hoạt động hỗ trợ (Complementary PWM Operation)

Trong chế độ hoạt động hỗ trợ (Complementary mode), mỗi cặp tín hiệu PWM thu được từ một tín hiệu PWM hỗ trợ (Complementary PWM signal). Khoảng thời gian nghỉ (Dead Time) có thể được lựa chọn để đưa vào trong quá trình đóng ngắt các khoá, khi cả hai tín hiệu có cùng trạng thái tích cực trong một thời gian ngắn



Hình 5.20: Tín hiệu PWM trong chế độ hoạt động hỗ trợ

Trong chế độ hoạt động hỗ trợ này, các thanh ghi so sánh được phân chia như sau:

Thanh ghi PDC1 điều khiển tín hiệu PWM1H/PWM1L

Thanh ghi PDC2 điều khiển tín hiệu PWM2H/PWM2L

Thanh ghi PDC3 điều khiển tín hiệu PWM3H/PWM3L

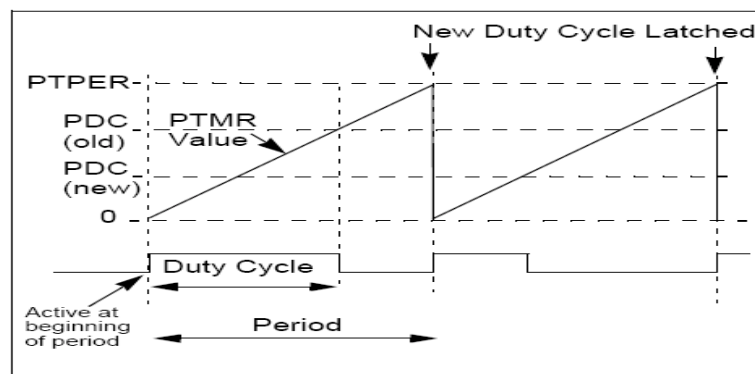
Thanh ghi PDC4 điều khiển tín hiệu PWM4H/PWM4L

Xung PWM dạng Edge Aligned

Tín hiệu Center Aligned PWM được tạo ra bởi module PWM khi bộ đếm thời gian PWM được cấu hình hoạt động ở chế độ Free Running hoặc Single Shot

Đối với tín hiệu Edgle PWM, có thời gian (Period) được xác định bởi giá trị trong thanh ghi PTPER và có chu kỳ (Duty cycle) được xác định bởi thanh ghi PDCx tương ứng . Tín hiệu PWM được chuyển sang tích cực vào thời điểm bắt đầu của chu kỳ (PTMR=0) và chuyển sang không tích cực khi giá trị trong thanh ghi PDCx bằng với giá trị trong thanh ghi PTMR. Nếu giá trị trong thanh ghi PDCx tương ứng bằng 0 , thì tín hiệu ra trên chân PWM tương ứng sẽ không tích cực trong suốt toàn bộ chu kỳ PWM. Tín hiệu ra trên chân PWM tương ứng sẽ tích cực trong suốt toàn bộ chu kỳ PWM nếu giá trị trong thanh ghi PDCx lớn hơn giá trị được lưu trong thanh ghi PTPER

Quá trình hoạt động được thể hiện trong hình



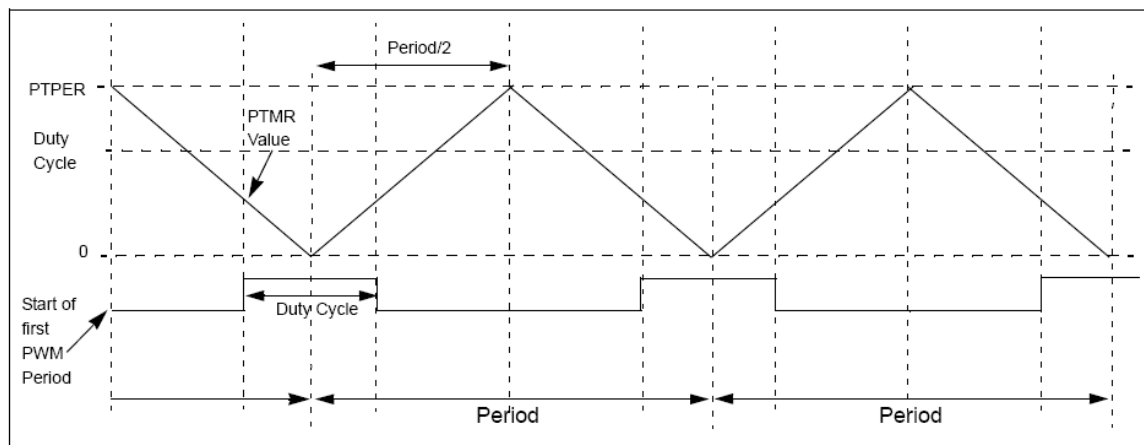
Hình 5.21: Xung PWM dạng Edge Aligned

Xung PWM dạng Center Aligned

Tín hiệu Center Aligned PWM được tạo ra bởi module PWM khi bộ đếm thời gian PWM được cấu hình hoạt động ở chế độ Up/Down Counting

Tín hiệu PWM (PWM compare output) được chuyển sang trạng thái tích cực khi giá trị trong thanh ghi PTMR bằng với giá trị trong thanh ghi PTPER và bộ đếm thời gian PWM đang đếm xuống . Tín hiệu PWM được chuyển sang trạng thái không tích cực khi bộ đếm đang đếm lên và giá trị trong thanh ghi PTMR bằng với giá trị trong thanh ghi PTPER

Quá trình hoạt động được thể hiện trong hình



Hình 5.22: Xung PWM dạng Center Aligned

5.5.3.3 Các bộ đếm tỉ lệ trong module PWM:

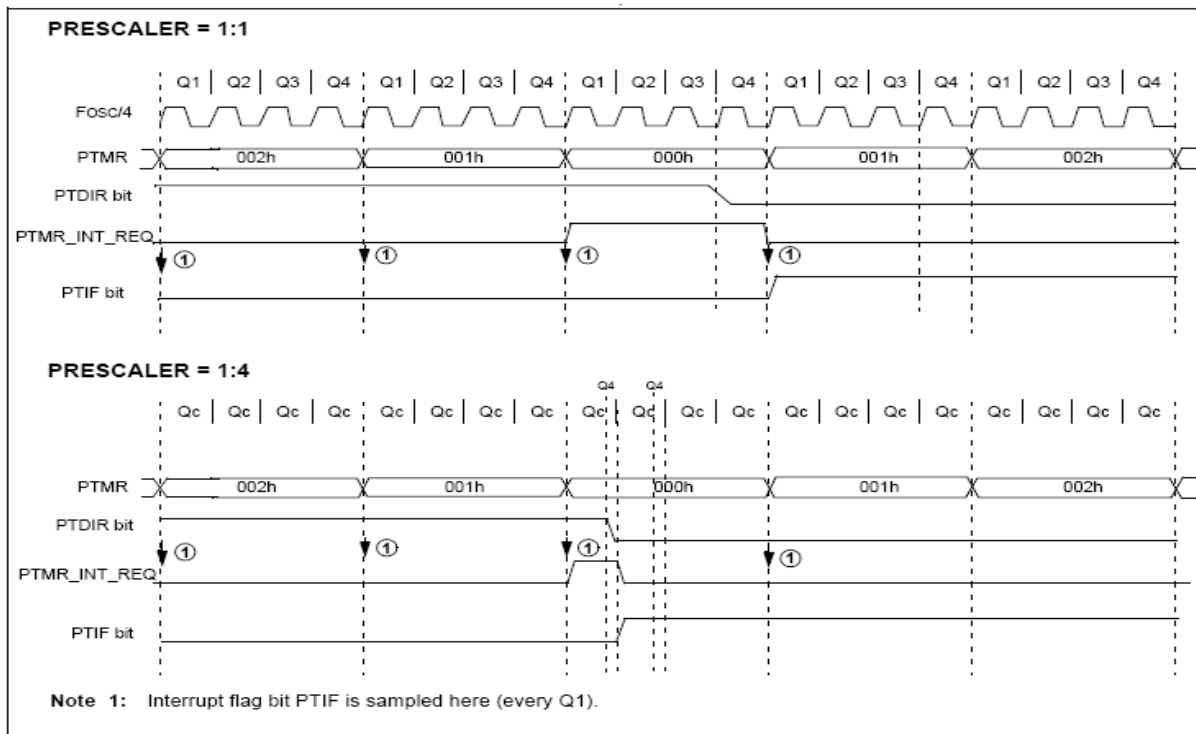
PWM Timer Base Prescaler

Xung clock đưa vào thanh ghi PTMR ($F_{OSC}/4$) được tỉ lệ 1:1, 1:4, 1:16 hoặc 1:64, được lựa chọn bởi các bit điều khiển PTCKPS<1:0> trong thanh ghi PTCN. Việc tỉ lệ sẽ bị xoá khi xảy ra các trường hợp sau:

Ghi vào thanh ghi PTMR

Ghi vào thanh ghi PTCN

Các reset CPU



Hình 5.23: Bộ đếm tỉ lệ trong module PWM

PWM Timer Base Postscaler

Sự trùng lặp giữa thanh ghi PTPER và thanh ghi PTMR có thể được lựa chọn theo các tỉ lệ từ 1:1 đến 1:16 thông qua 4-bit postscaler để tạo ra tín hiệu ngắt. Việc tỉ lệ này được sử dụng trong trường hợp không cần thay đổi duty cycle của xung PWM ở mỗi chu kì PWM. Bộ đếm postscaler sẽ bị xóa bởi các tác động sau:

Ghi vào thanh ghi PTMR

Ghi vào thanh ghi PTCN

Các reset CPU

Thanh ghi PTMR sẽ không bị xóa khi thanh ghi PTCN được ghi vào

5.5.3.4 Các thanh ghi làm việc trong module PWM

Thanh ghi PTPER (PWM Period)

PTPER là một thanh ghi 15 bit và được sử dụng để cài đặt việc đếm thời gian cho module PWM. PTPER là một thanh ghi đếm kép. Nội dung trong thanh ghi đếm PTPER được nạp vào thanh ghi PTPER như sau:

Ở chế độ Free Running và Single Shot : Khi thanh ghi PTMR được reset về 0 sau khi bằng giá trị trong thanh ghi PTPER

Chu kì PWM trong chế độ Free Running được tính bởi công thức sau:

$$PTPER = \frac{F_{cy}}{F_{PWM} * (PTMR Prescaler)} - 1$$

Ví dụ:

FCY = 20 MHz

FPWM = 20,000 Hz

PTMR Prescaler = 1:1

PTPER = 20000000 / (1 * 20000) - 1 = 999

Ở chế độ Up/Down Counting : Khi thanh ghi PTMR bằng 0

Giá trị được lưu giữ trong bộ đếm PTPER sẽ tự động được nạp vào thanh ghi PTPER khi bộ đếm thời gian PWM bị vô hiệu hoá (PTEN=0)

Chu kì PWM trong chế độ Up/Down Counting được tính bởi công thức sau:

$$PTPER = \frac{F_{cy}}{F_{PWM} * (PTMR Prescaler) * 2} - 1$$

Ví dụ:

FCY = 20 MHz

FPWM = 20,000 Hz

PTMR Prescaler = 1:1

PTPER = 20000000 / (1 * 2 * 20000) - 1 = 499

Các thanh ghi so sánh: (PWM Duty Cycle Comparison Units)

Module PWM có 4 thanh ghi 16 bit (PDC1, PDC2, PDC3, PDC4(Duty cycle register)) được dùng để xác định chu kì của module này

Giá trị trong mỗi thanh ghi định nghĩa khoảng thời gian mà tín hiệu PWM (PWM output) ở trạng thái tích cực. Bit ở vị trí thấp nhất (LSB) cho biết bắt đầu xuất hiện cạnh của PWM.

Các thanh ghi đệm (Duty Cycle Register Buffer):

4 thanh ghi PDCx là các bộ đệm kép cho phép cập nhật tín hiệu PWM. Trong mỗi chu kỳ, có một thanh ghi đệm được truy cập bởi người dùng và thanh ghi còn lại lưu trữ giá trị so sánh thực tế sử dụng trong chu kì PWM hiện tại

CHƯƠNG 5: GIỚI THIỆU VỀ dsPIC6010

Đối với tín hiệu Edge Aligned PWM, giá trị chu kỳ mới sẽ được cập nhật mỗi khi giá trị trong thanh ghi PTMR bằng với giá trị trong thanh ghi PTPER và thanh ghi PTMR được reset. Nội dung trong thanh ghi đếm sẽ tự động nạp vào thanh ghi dùng để so sánh khi bộ đếm thời gian PWM bị vô hiệu hóa (PTEN=0) và bit UDIS trong thanh ghi PWMCON2 sẽ bị xóa. Trong chế độ Up/Down Counting, giá trị chu kỳ mới sẽ được cập nhật khi giá trị trong thanh ghi PTMR bằng 0, và bộ đếm thời gian bắt đầu đếm lên. Nội dung trong thanh ghi đếm sẽ tự động được nạp vào thanh ghi dùng để so sánh khi bộ đếm thời gian PWM bị vô hiệu hóa (PTEN=0).

Trong chế độ Up/Down Counting với đặc điểm cập nhật hai lần (Double Update), PWM, giá trị chu kỳ mới sẽ được cập nhật mỗi khi giá trị trong thanh ghi PTMR bằng với giá trị trong thanh ghi PTPER và khi giá trị trong thanh ghi PTMR bằng 0. Nội dung trong thanh ghi đếm sẽ tự động được nạp vào thanh ghi dùng để so sánh khi bộ đếm thời gian PWM bị vô hiệu hóa (PTEN=0).

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State
PTCON	01C0	PTEN	---	PTSIDL	---	---	---	---	---	PTOPS<3:0>				PTCKPS<1:0>		PTMOD<1:0>		0000 0000 0000 0000
PTMR	01C2	PTDIR	PWM Timer Count Value															0000 0000 0000 0000
PTPER	01C4	---	PWM Time Base Period Register															0000 0000 0000 0000
SEVTCMP	01C8	SEVTDIR	PWM Special Event Compare Register															0000 0000 0000 0000
PWMCON1	01C8	---	---	---	---	PTMOD4	PTMOD3	PTMOD2	PTMOD1	PENH	PEN3H	PEN2H	PEN1H	PENIL	PEN3L	PEN2L	PEN1L	0000 0000 1111 1111
PWMCON2	01CA	---	---	---	---	SEVOPS<3:0>				---	---	---	---	---	---	OSYNC	UDIS	0000 0000 0000 0000
DTCON1	01CC	DTBPS<1:0>		Dead Time B Value						DTAPS<1:0>		Dead Time A Value						0000 0000 0000 0000
DTCON2	01CE	---	---	---	---	---	---	---	---	DTS4A	DTS4I	DTS3A	DTS3I	DTS2A	DTS2I	DTS1A	DTS1I	0000 0000 0000 0000
FLTAON	01D0	FAOV4H	FAOV4L	FAOV3H	FAOV3L	FAOV2H	FAOV2L	FAOV1H	FAOV1L	FLTAM	---	---	---	FAEN1	FAEN3	FAEN2	FAEN1	0000 0000 0000 0000
FLTBON	01D2	FBOV4H	FBOV4L	FBOV3H	FBOV3L	FBOV2H	FBOV2L	FBOV1H	FBOV1L	FLTBM	---	---	---	FBEN4	FBEN3	FBEN2	FBEN1	0000 0000 0000 0000
OVDCON	01D4	POVD4H	POVD4L	POVD3H	POVD3L	POVD2H	POVD2L	POVD1H	POVD1L	POUT4H	POUT4L	POUT3H	POUT3L	POUT2H	POUT2L	POUT1H	POUT1L	1111 1111 0000 0000
PDC1	01D8	PWM Duty Cycle #1 Register															0000 0000 0000 0000	
PDC2	01D8	PWM Duty Cycle #2 Register															0000 0000 0000 0000	
PDC3	01DA	PWM Duty Cycle #3 Register															0000 0000 0000 0000	
PDC4	01DC	PWM Duty Cycle #4 Register															0000 0000 0000 0000	

Legend: u = uninitialized bit

Bảng 5.8 : Bảng thanh ghi điều khiển module PWM

5.6 GIỚI THIỆU VỀ TẬP LỆNH CỦA MCU DSPIC-6010

Base Instr #	Assembly Mnemonic	Assembly Syntax	Description	# of words	# of cycles	Status Flags Affected
1	ADD	ADD Acc	Add Accumulators	1	1	OA,OB,SA,SB
		ADD f	$f = f + WREG$	1	1	C,DC,N,OV,Z
		ADD f,WREG	$WREG = f + WREG$	1	1	C,DC,N,OV,Z
		ADD #lit10,Wn	$Wd = lit10 + Wd$	1	1	C,DC,N,OV,Z
		ADD Wb,Ws,Wd	$Wd = Wb + Ws$	1	1	C,DC,N,OV,Z
		ADD Wb,#lit5,Wd	$Wd = Wb + lit5$	1	1	C,DC,N,OV,Z
2	ADDC	ADDC Wso,#lit4,Acc	16-bit Signed Add to Accumulator	1	1	OA,OB,SA,SB
		ADDC f	$f = f + WREG + (C)$	1	1	C,DC,N,OV,Z
		ADDC f,WREG	$WREG = f + WREG + (C)$	1	1	C,DC,N,OV,Z
		ADDC #lit10,Wn	$Wd = lit10 + Wd + (C)$	1	1	C,DC,N,OV,Z
		ADDC Wb,Ws,Wd	$Wd = Wb + Ws + (C)$	1	1	C,DC,N,OV,Z
		ADDC Wb,#lit5,Wd	$Wd = Wb + lit5 + (C)$	1	1	C,DC,N,OV,Z
3	AND	AND f	$f = f .AND. WREG$	1	1	N,Z
		AND f,WREG	$WREG = f .AND. WREG$	1	1	N,Z
		AND #lit10,Wn	$Wd = lit10 .AND. Wd$	1	1	N,Z
		AND Wb,Ws,Wd	$Wd = Wb .AND. Ws$	1	1	N,Z
		AND Wb,#lit5,Wd	$Wd = Wb .AND. lit5$	1	1	N,Z
4	ASR	ASR f	$f = \text{Arithmetic Right Shift } f$	1	1	C,N,OV,Z
		ASR f,WREG	$WREG = \text{Arithmetic Right Shift } f$	1	1	C,N,OV,Z
		ASR Ws,Wd	$Wd = \text{Arithmetic Right Shift } Ws$	1	1	C,N,OV,Z
		ASR Wb,Wns,Wnd	$Wnd = \text{Arithmetic Right Shift } Wb \text{ by } Wns$	1	1	N,Z
		ASR Wb,#lit5,Wnd	$Wnd = \text{Arithmetic Right Shift } Wb \text{ by } lit5$	1	1	N,Z
5	BCLR	BCLR f,#bit4	Bit Clear f	1	1	None
		BCLR Ws,#bit4	Bit Clear Ws	1	1	None
6	BRA	BRA C,Expr	Branch if Carry	1	1 (2)	None
		BRA GE,Expr	Branch if greater than or equal	1	1 (2)	None
		BRA GEU,Expr	Branch if unsigned greater than or equal	1	1 (2)	None
		BRA GT,Expr	Branch if greater than	1	1 (2)	None
		BRA GTU,Expr	Branch if unsigned greater than	1	1 (2)	None
		BRA LE,Expr	Branch if less than or equal	1	1 (2)	None
		BRA LEU,Expr	Branch if unsigned less than or equal	1	1 (2)	None
		BRA LT,Expr	Branch if less than	1	1 (2)	None
		BRA LTU,Expr	Branch if unsigned less than	1	1 (2)	None
		BRA N,Expr	Branch if Negative	1	1 (2)	None
		BRA NC,Expr	Branch if Not Carry	1	1 (2)	None
		BRA NN,Expr	Branch if Not Negative	1	1 (2)	None
		BRA NOV,Expr	Branch if Not Overflow	1	1 (2)	None
		BRA NZ,Expr	Branch if Not Zero	1	1 (2)	None
		BRA OA,Expr	Branch if accumulator A overflow	1	1 (2)	None
		BRA OB,Expr	Branch if accumulator B overflow	1	1 (2)	None
		BRA OV,Expr	Branch if Overflow	1	1 (2)	None
		BRA SA,Expr	Branch if accumulator A saturated	1	1 (2)	None
		BRA SB,Expr	Branch if accumulator B saturated	1	1 (2)	None
		BRA Expr	Branch Unconditionally	1	2	None
		BRA Z,Expr	Branch if Zero	1	1 (2)	None
		BRA Wn	Computed Branch	1	2	None
7	BSET	BSET f,#bit4	Bit Set f	1	1	None
		BSET Ws,#bit4	Bit Set Ws	1	1	None
8	BSW	BSW.C Ws,Wb	Write C bit to Ws<Wb>	1	1	None
		BSW.Z Ws,Wb	Write Z bit to Ws<Wb>	1	1	None
9	BTG	BTG f,#bit4	Bit Toggle f	1	1	None
		BTG Ws,#bit4	Bit Toggle Ws	1	1	None

CHƯƠNG 5: GIỚI THIỆU VỀ dsPIC6010

Base Instr #	Assembly Mnemonic	Assembly Syntax	Description	# of words	# of cycles	Status Flags Affected
10	BTSC	BTSC f,#bit4	Bit Test f, Skip if Clear	1	1 (2 or 3)	None
		BTSC Ws,#bit4	Bit Test Ws, Skip if Clear	1	1 (2 or 3)	None
11	BTSS	BTSS f,#bit4	Bit Test f, Skip if Set	1	1 (2 or 3)	None
		BTSS Ws,#bit4	Bit Test Ws, Skip if Set	1	1 (2 or 3)	None
12	BTST	BTST f,#bit4	Bit Test f	1	1	Z
		BTST.C Ws,#bit4	Bit Test Ws to C	1	1	C
		BTST.Z Ws,#bit4	Bit Test Ws to Z	1	1	Z
		BTST.C Ws,Wb	Bit Test Ws<Wb> to C	1	1	C
		BTST.Z Ws,Wb	Bit Test Ws<Wb> to Z	1	1	Z
13	BTSTS	BTSTS f,#bit4	Bit Test then Set f	1	1	Z
		BTSTS.C Ws,#bit4	Bit Test Ws to C, then Set	1	1	C
		BTSTS.Z Ws,#bit4	Bit Test Ws to Z, then Set	1	1	Z
14	CALL	CALL lit23	Call subroutine	2	2	None
		CALL Wn	Call indirect subroutine	1	2	None
15	CLR	CLR f	f = 0x0000	1	1	None
		CLR WREG	WREG = 0x0000	1	1	None
		CLR Ws	Ws = 0x0000	1	1	None
		CLR Acc,Wx,Wxd,Wy,Wyd,AWB	Clear Accumulator	1	1	OA,OB,SA,SB
16	CLRWDT	CLRWDT	Clear Watchdog Timer	1	1	WDTO,Sleep
17	COM	COM f	f = \bar{f}	1	1	N,Z
		COM f,WREG	WREG = \bar{f}	1	1	N,Z
		COM Ws,Wd	Wd = Ws	1	1	N,Z
18	CP	CP f	Compare f with WREG	1	1	C,DC,N,OV,Z
		CP Wb,#lit5	Compare Wb with lit5	1	1	C,DC,N,OV,Z
		CP Wb,Ws	Compare Wb with Ws (Wb - Ws)	1	1	C,DC,N,OV,Z
19	CP0	CP0 f	Compare f with 0x0000	1	1	C,DC,N,OV,Z
		CP0 Ws	Compare Ws with 0x0000	1	1	C,DC,N,OV,Z
20	CP1	CP1 f	Compare f with 0xFFFF	1	1	C,DC,N,OV,Z
		CP1 Ws	Compare Ws with 0xFFFF	1	1	C,DC,N,OV,Z
21	CPB	CPB f	Compare f with WREG, with Borrow	1	1	C,DC,N,OV,Z
		CPB Wb,#lit5	Compare Wb with lit5, with Borrow	1	1	C,DC,N,OV,Z
		CPB Wb,Ws	Compare Wb with Ws, with Borrow (Wb - Ws - \bar{C})	1	1	C,DC,N,OV,Z
22	CPSEQ	CPSEQ Wb, Wn	Compare Wb with Wn, skip if =	1	1 (2 or 3)	None
23	CPSGT	CPSGT Wb, Wn	Compare Wb with Wn, skip if >	1	1 (2 or 3)	None
24	CPSLT	CPSLT Wb, Wn	Compare Wb with Wn, skip if <	1	1 (2 or 3)	None
25	CPSNE	CPSNE Wb, Wn	Compare Wb with Wn, skip if \neq	1	1 (2 or 3)	None
26	DAW	DAW Wn	Wn = decimal adjust Wn	1	1	C
27	DEC	DEC f	f = f - 1	1	1	C,DC,N,OV,Z
		DEC f,WREG	WREG = f - 1	1	1	C,DC,N,OV,Z
		DEC Ws,Wd	Wd = Ws - 1	1	1	C,DC,N,OV,Z
28	DEC2	DEC2 f	f = f - 2	1	1	C,DC,N,OV,Z
		DEC2 f,WREG	WREG = f - 2	1	1	C,DC,N,OV,Z
		DEC2 Ws,Wd	Wd = Ws - 2	1	1	C,DC,N,OV,Z
29	DISI	DISI #lit14	Disable Interrupts for k instruction cycles	1	1	None

CHƯƠNG 5: GIỚI THIỆU VỀ dsPIC6010

Base Instr #	Assembly Mnemonic	Assembly Syntax	Description	# of words	# of cycles	Status Flags Affected
30	DIV	DIV.S Wm,Wn	Signed 16/16-bit Integer Divide	1	18	N,Z,C, OV
		DIV.SD Wm,Wn	Signed 32/16-bit Integer Divide	1	18	N,Z,C, OV
		DIV.U Wm,Wn	Unsigned 16/16-bit Integer Divide	1	18	N,Z,C, OV
		DIV.UD Wm,Wn	Unsigned 32/16-bit Integer Divide	1	18	N,Z,C, OV
31	DIVF	DIVF Wm,Wn	Signed 16/16-bit Fractional Divide	1	18	N,Z,C, OV
32	DO	DO #lit14,Expr	Do code to PC+Expr, lit14+1 times	2	2	None
		DO Wn,Expr	Do code to PC+Expr, (Wn)+1 times	2	2	None
33	ED	ED Wm*Wm,Acc,Wx,Wy,Wxd	Euclidean Distance (no accumulate)	1	1	OA,OB,OAB, SA,SB,SAB
34	EDAC	EDAC Wm*Wm,Acc,Wx,Wy,Wxd	Euclidean Distance	1	1	OA,OB,OAB, SA,SB,SAB
35	EXCH	EXCH Wns,Wnd	Swap Wns with Wnd	1	1	None
36	FBCL	FBCL Ws,Wnd	Find Bit Change from Left (MSb) Side	1	1	C
37	FF1L	FF1L Ws,Wnd	Find First One from Left (MSb) Side	1	1	C
38	FF1R	FF1R Ws,Wnd	Find First One from Right (LSb) Side	1	1	C
39	GOTO	GOTO Expr	Go to address	2	2	None
		GOTO Wn	Go to indirect	1	2	None
40	INC	INC f	$f = f + 1$	1	1	C,DC,N,OV,Z
		INC f,WREG	$WREG = f + 1$	1	1	C,DC,N,OV,Z
		INC Ws,Wd	$Wd = Ws + 1$	1	1	C,DC,N,OV,Z
41	INC2	INC2 f	$f = f + 2$	1	1	C,DC,N,OV,Z
		INC2 f,WREG	$WREG = f + 2$	1	1	C,DC,N,OV,Z
		INC2 Ws,Wd	$Wd = Ws + 2$	1	1	C,DC,N,OV,Z
42	IOR	IOR f	$f = f .IOR. WREG$	1	1	N,Z
		IOR f,WREG	$WREG = f .IOR. WREG$	1	1	N,Z
		IOR #lit10,Wn	$Wd = lit10 .IOR. Wd$	1	1	N,Z
		IOR Wb,Ws,Wd	$Wd = Wb .IOR. Ws$	1	1	N,Z
		IOR Wb,#lit5,Wd	$Wd = Wb .IOR. lit5$	1	1	N,Z
43	LAC	LAC Wso,#Slit4,Acc	Load Accumulator	1	1	OA,OB,OAB, SA,SB,SAB
44	LNK	LNK #lit14	Link frame pointer	1	1	None
45	LSR	LSR f	$f = \text{Logical Right Shift } f$	1	1	C,N,OV,Z
		LSR f,WREG	$WREG = \text{Logical Right Shift } f$	1	1	C,N,OV,Z
		LSR Ws,Wd	$Wd = \text{Logical Right Shift } Ws$	1	1	C,N,OV,Z
		LSR Wb,Wns,Wnd	$Wnd = \text{Logical Right Shift } Wb \text{ by } Wns$	1	1	N,Z
		LSR Wb,#lit5,Wnd	$Wnd = \text{Logical Right Shift } Wb \text{ by } lit5$	1	1	N,Z
46	MAC	MAC Wm*Wn,Acc,Wx,Wxd,Wy,Wyd, AWB	Multiply and Accumulate	1	1	OA,OB,OAB, SA,SB,SAB
		MAC Wm*Wm,Acc,Wx,Wxd,Wy,Wyd	Square and Accumulate	1	1	OA,OB,OAB, SA,SB,SAB
47	MOV	MOV f,Wn	Move f to Wn	1	1	None
		MOV f	Move f to f	1	1	N,Z
		MOV f,WREG	Move f to WREG	1	1	N,Z
		MOV #lit16,Wn	Move 16-bit literal to Wn	1	1	None
		MOV.b #lit8,Wn	Move 8-bit literal to Wn	1	1	None
		MOV Wn,f	Move Wn to f	1	1	None
		MOV Wso,Wdo	Move Ws to Wd	1	1	None
		MOV WREG,f	Move WREG to f	1	1	N,Z
		MOV.D Wns,Wd	Move Double from W(ns):W(ns+1) to Wd	1	2	None
48	MOV.SAC	MOV.SAC Acc,Wx,Wxd,Wy,Wyd,AWB	Pre-fetch and store accumulator	1	1	None
		MOV.SAC Wm*Wn,Acc,Wx,Wxd,Wy,Wyd	Multiply Wm by Wn to Accumulator	1	1	OA,OB,OAB, SA,SB,SAB
49	MPY	MPY Wm*Wm,Acc,Wx,Wxd,Wy,Wyd	Square Wm to Accumulator	1	1	OA,OB,OAB, SA,SB,SAB
		MPY Wm*Wn,Acc,Wx,Wxd,Wy,Wyd	-(Multiply Wm by Wn) to Accumulator	1	1	None
50	MPY.N	MPY.N Wm*Wn,Acc,Wx,Wxd,Wy,Wyd	-(Multiply Wm by Wn) to Accumulator	1	1	None
51	MSC	MSC Wm*Wm,Acc,Wx,Wxd,Wy,Wyd, AWB	Multiply and Subtract from Accumulator	1	1	OA,OB,OAB, SA,SB,SAB

CHƯƠNG 5: GIỚI THIỆU VỀ dsPIC6010

Base Instr #	Assembly Mnemonic	Assembly Syntax	Description	# of words	# of cycle s	Status Flags Affected
52	MUL	MUL.SS Wb,Ws,Wnd	{Wnd+1, Wnd} = signed(Wb) * signed(Ws)	1	1	None
		MUL.SU Wb,Ws,Wnd	{Wnd+1, Wnd} = signed(Wb) * unsigned(Ws)	1	1	None
		MUL.US Wb,Ws,Wnd	{Wnd+1, Wnd} = unsigned(Wb) * signed(Ws)	1	1	None
		MUL.UU Wb,Ws,Wnd	{Wnd+1, Wnd} = unsigned(Wb) * unsigned(Ws)	1	1	None
		MUL.SU Wb,#lit5,Wnd	{Wnd+1, Wnd} = signed(Wb) * unsigned(lit5)	1	1	None
		MUL.UU Wb,#lit5,Wnd	{Wnd+1, Wnd} = unsigned(Wb) * unsigned(lit5)	1	1	None
		MUL f	W3:W2 = f * WREG	1	1	None
53	NEG	NEG Acc	Negate Accumulator	1	1	OA,OB,OAB, SA,SB,SAB
		NEG f	$f = \overline{f} + 1$	1	1	C,DC,N,OV,Z
		NEG f,WREG	WREG = $\overline{f} + 1$	1	1	C,DC,N,OV,Z
		NEG Ws,Wd	$Wd = \overline{Ws} + 1$	1	1	C,DC,N,OV,Z
54	NOP	NOP	No Operation	1	1	None
		NOPR	No Operation	1	1	None
55	POP	POP f	Pop f from top-of-stack (TOS)	1	1	None
		POP Wdo	Pop from top-of-stack (TOS) to Wdo	1	1	None
		POP.D Wnd	Pop from top-of-stack (TOS) to W(nd):W(nd+1)	1	2	None
		POP.S	Pop Shadow Registers	1	1	All
56	PUSH	PUSH f	Push f to top-of-stack (TOS)	1	1	None
		PUSH Wso	Push Wso to top-of-stack (TOS)	1	1	None
		PUSH.D Wns	Push W(ns):W(ns+1) to top-of-stack (TOS)	1	2	None
		PUSH.S	Push Shadow Registers	1	1	None
57	PWRSV	PWRSV #lit1	Go into Sleep or Idle mode	1	1	WDTO,Sleep
58	RCALL	RCALL Expr	Relative Call	1	2	None
		RCALL Wn	Computed Call	1	2	None
59	REPEAT	REPEAT #lit14	Repeat Next Instruction lit14+1 times	1	1	None
		REPEAT Wn	Repeat Next Instruction (Wn)+1 times	1	1	None
60	RESET	RESET	Software device Reset	1	1	None
61	RETFIE	RETFIE	Return from interrupt	1	3 (2)	None
62	RETLW	RETLW #lit10,Wn	Return with literal in Wn	1	3 (2)	None
63	RETURN	RETURN	Return from Subroutine	1	3 (2)	None
64	RLC	RLC f	f = Rotate Left through Carry f	1	1	C,N,Z
		RLC f,WREG	WREG = Rotate Left through Carry f	1	1	C,N,Z
		RLC Ws,Wd	Wd = Rotate Left through Carry Ws	1	1	C,N,Z
65	RLNC	RLNC f	f = Rotate Left (No Carry) f	1	1	N,Z
		RLNC f,WREG	WREG = Rotate Left (No Carry) f	1	1	N,Z
		RLNC Ws,Wd	Wd = Rotate Left (No Carry) Ws	1	1	N,Z
66	RRC	RRC f	f = Rotate Right through Carry f	1	1	C,N,Z
		RRC f,WREG	WREG = Rotate Right through Carry f	1	1	C,N,Z
		RRC Ws,Wd	Wd = Rotate Right through Carry Ws	1	1	C,N,Z
67	RRNC	RRNC f	f = Rotate Right (No Carry) f	1	1	N,Z
		RRNC f,WREG	WREG = Rotate Right (No Carry) f	1	1	N,Z
		RRNC Ws,Wd	Wd = Rotate Right (No Carry) Ws	1	1	N,Z
68	SAC	SAC Acc,#Slit4,Wdo	Store Accumulator	1	1	None
		SAC.R Acc,#Slit4,Wdo	Store Rounded Accumulator	1	1	None
69	SE	SE Ws,Wnd	Wnd = sign extended Ws	1	1	C,N,Z
70	SETM	SETM f	f = 0xFFFF	1	1	None
		SETM WREG	WREG = 0xFFFF	1	1	None
		SETM Ws	Ws = 0xFFFF	1	1	None
71	SFTAC	SFTAC Acc,Wn	Arithmetic Shift Accumulator by (Wn)	1	1	OA,OB,OAB, SA,SB,SAB
		SFTAC Acc,#Slit6	Arithmetic Shift Accumulator by Slit6	1	1	OA,OB,OAB, SA,SB,SAB

CHƯƠNG 5: GIỚI THIỆU VỀ dsPIC6010

Base Instr #	Assembly Mnemonic	Assembly Syntax	Description	# of words	# of cycles	Status Flags Affected
72	SL	SL f	f = Left Shift f	1	1	C,N,OV,Z
		SL f,WREG	WREG = Left Shift f	1	1	C,N,OV,Z
		SL Ws,Wd	Wd = Left Shift Ws	1	1	C,N,OV,Z
		SL Wb,Wns,Wnd	Wnd = Left Shift Wb by Wns	1	1	N,Z
		SL Wb,#lit5,Wnd	Wnd = Left Shift Wb by lit5	1	1	N,Z
73	SUB	SUB Acc	Subtract Accumulators	1	1	OA,OB,OAB,SA,SB,SAB
		SUB f	f = f - WREG	1	1	C,DC,N,OV,Z
		SUB f,WREG	WREG = f - WREG	1	1	C,DC,N,OV,Z
		SUB #lit10,Wn	Wn = Wn - lit10	1	1	C,DC,N,OV,Z
		SUB Wb,Ws,Wd	Wd = Wb - Ws	1	1	C,DC,N,OV,Z
74	SUBB	SUBB f	f = f - WREG - (C)	1	1	C,DC,N,OV,Z
		SUBB f,WREG	WREG = f - WREG - (C)	1	1	C,DC,N,OV,Z
		SUBB #lit10,Wn	Wn = Wn - lit10 - (C)	1	1	C,DC,N,OV,Z
		SUBB Wb,Ws,Wd	Wd = Wb - Ws - (C)	1	1	C,DC,N,OV,Z
		SUBB Wb,#lit5,Wd	Wd = Wb - lit5 - (C)	1	1	C,DC,N,OV,Z
75	SUBR	SUBR f	f = WREG - f	1	1	C,DC,N,OV,Z
		SUBR f,WREG	WREG = WREG - f	1	1	C,DC,N,OV,Z
		SUBR Wb,Ws,Wd	Wd = Ws - Wb	1	1	C,DC,N,OV,Z
		SUBR Wb,#lit5,Wd	Wd = lit5 - Wb	1	1	C,DC,N,OV,Z
76	SUBBR	SUBBR f	f = WREG - f - (C)	1	1	C,DC,N,OV,Z
		SUBBR f,WREG	WREG = WREG - f - (C)	1	1	C,DC,N,OV,Z
		SUBBR Wb,Ws,Wd	Wd = Ws - Wb - (C)	1	1	C,DC,N,OV,Z
		SUBBR Wb,#lit5,Wd	Wd = lit5 - Wb - (C)	1	1	C,DC,N,OV,Z
77	SWAP	SWAP.b Wn	Wn = nibble swap Wn	1	1	None
		SWAP Wn	Wn = byte swap Wn	1	1	None
78	TBLRDH	TBLRDH Ws,Wd	Read Prog<23:16> to Wd<7:0>	1	2	None
79	TBLRDL	TBLRDL Ws,Wd	Read Prog<15:0> to Wd	1	2	None
80	TBLWTH	TBLWTH Ws,Wd	Write Ws<7:0> to Prog<23:16>	1	2	None
81	TBLWTL	TBLWTL Ws,Wd	Write Ws to Prog<15:0>	1	2	None
82	ULNK	ULNK	Unlink frame pointer	1	1	None
83	XOR	XOR f	f = f .XOR. WREG	1	1	N,Z
		XOR f,WREG	WREG = f .XOR. WREG	1	1	N,Z
		XOR #lit10,Wn	Wd = lit10 .XOR. Wd	1	1	N,Z
		XOR Wb,Ws,Wd	Wd = Wb .XOR. Ws	1	1	N,Z
		XOR Wb,#lit5,Wd	Wd = Wb .XOR. lit5	1	1	N,Z
84	ZE	ZE Ws,Wnd	Wnd = Zero-Extend Ws	1	1	C,Z,N

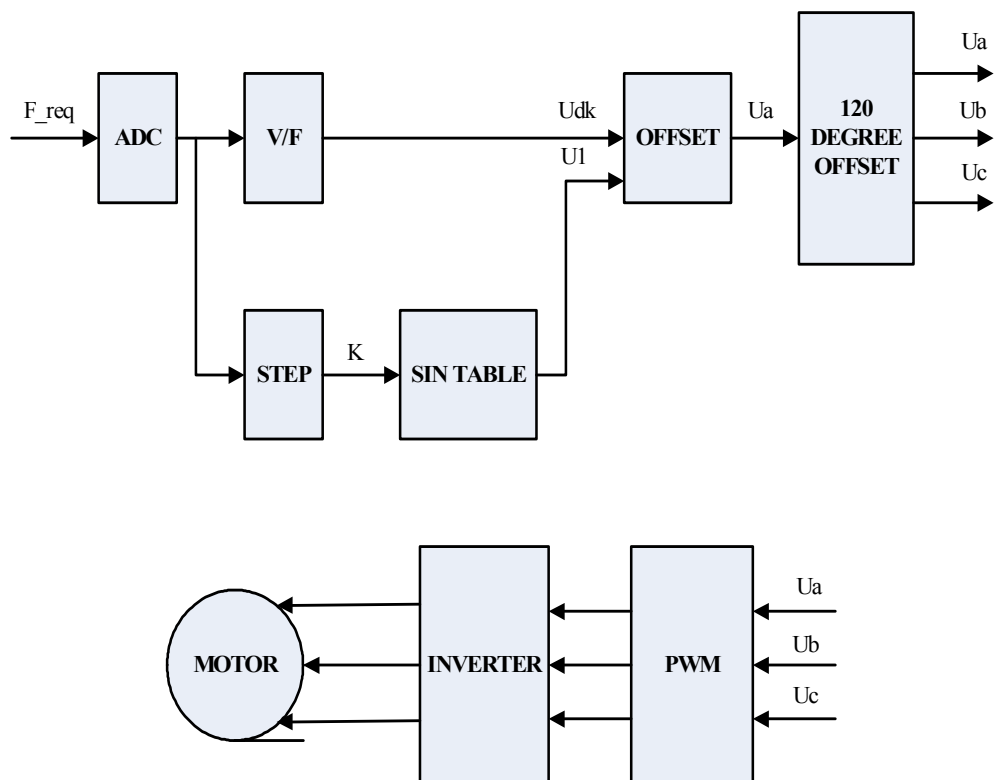
Bảng 5.9: Bảng tập lệnh MCU 6010

CHƯƠNG 6

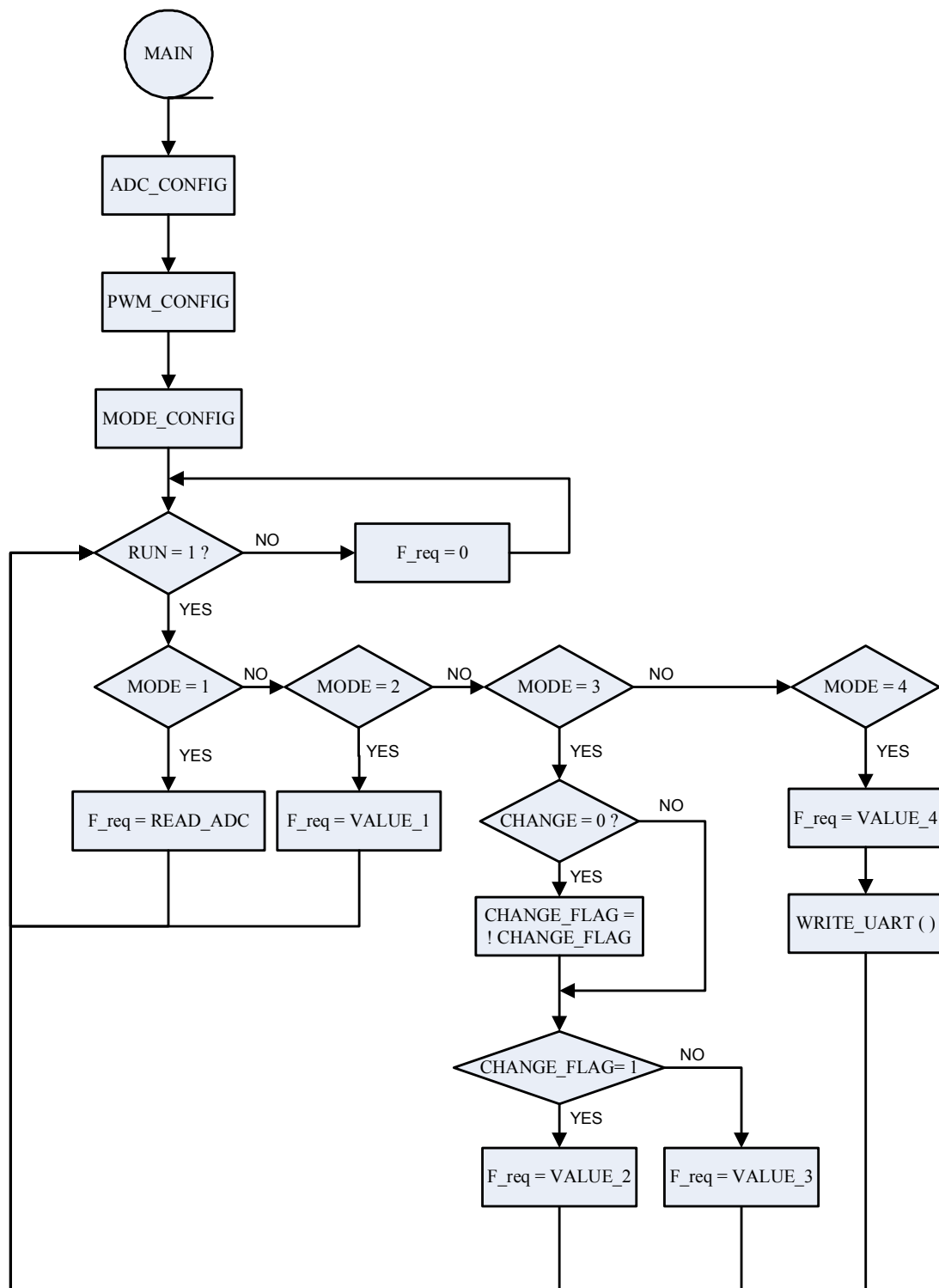
SƠ ĐỒ KHỐI VÀ GIẢI THUẬT ĐIỀU KHIỂN

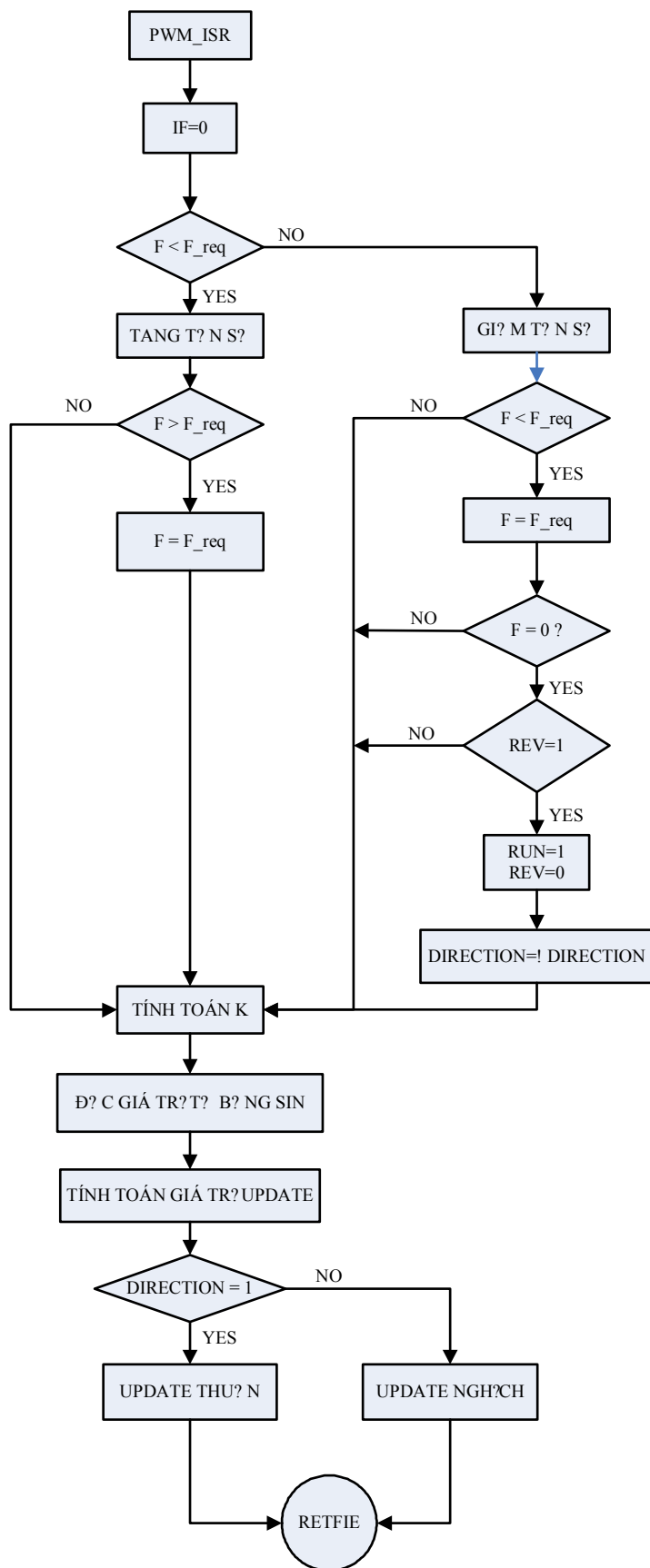
CHƯƠNG 6: SƠ ĐỒ KHỐI VÀ GIẢI THUẬT ĐIỀU KHIỂN

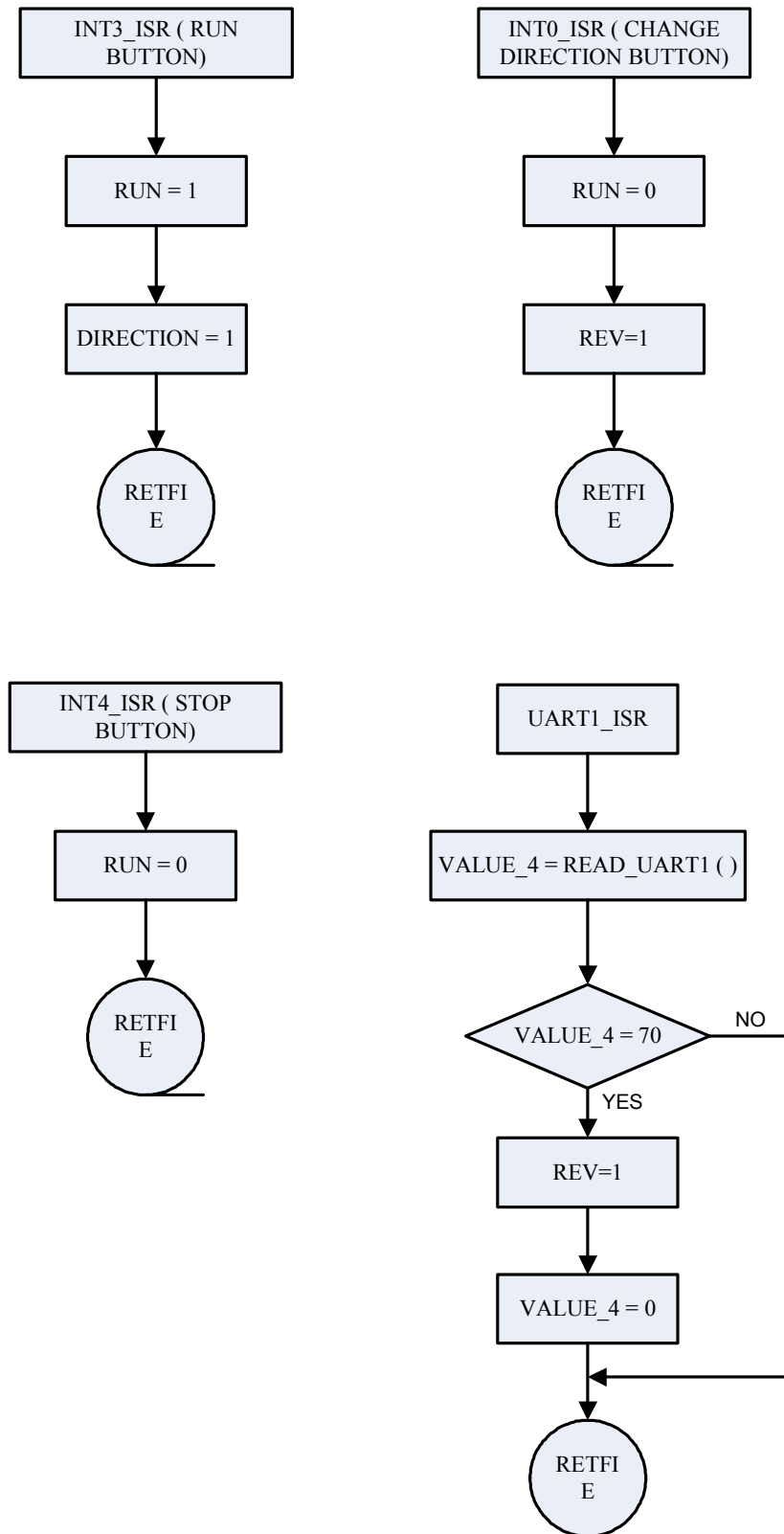
6.1 Sơ đồ khối chương trình :



6.2 Sơ đồ giải thuật chương trình :







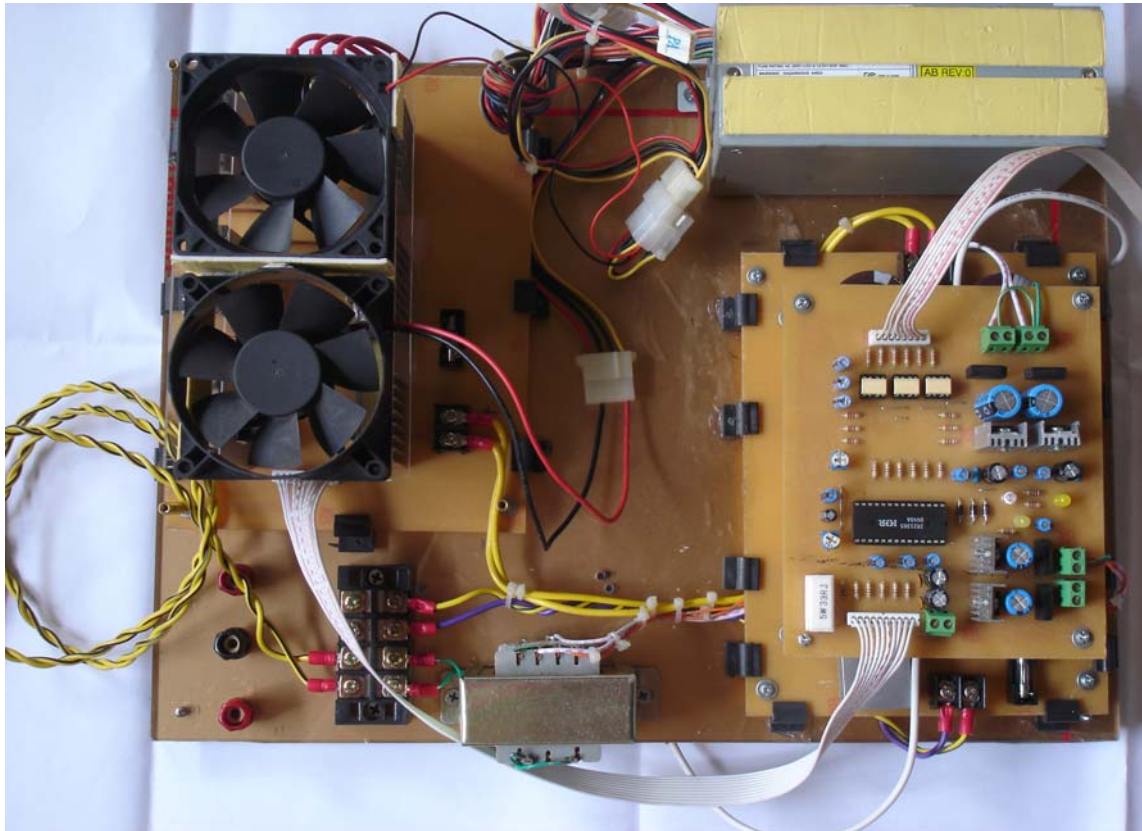
CHƯƠNG 7

KẾT QUẢ ĐẠT ĐƯỢC

CHƯƠNG 7 : KẾT QUẢ ĐẠT ĐƯỢC

7.1 Phần cứng:

7.1.1 Mạch động lực:



Hình 7.1 : Mạch động lực

Mạch động lực gồm (chỉnh lưu 1 pha không điều khiển, bộ nghịch lưu 6 khoá , mạch lái mosfet, nguồn cung cấp 5V,12V,) hoạt động ổn định

Mạch vận hành động cơ 2 HP (đấu tam giác, vận hành ở chế độ không tải) ở tất cả các chế độ điều khiển thông thường(RUN, STOP, đảo chiều, thay đổi tốc độ đặt).

+ Khuyết điểm:

- Nguồn AC(nguồn 1 pha) cung cấp không đủ yêu cầu dẫn đến động cơ không thể vận hành định mức

- Nhiệt độ các khóa công suất khá cao (50-60 ° C)

- Chưa có khâu hồi tiếp dòng ,hồi tiếp tốc độ,hồi tiếp điện áp DC

....

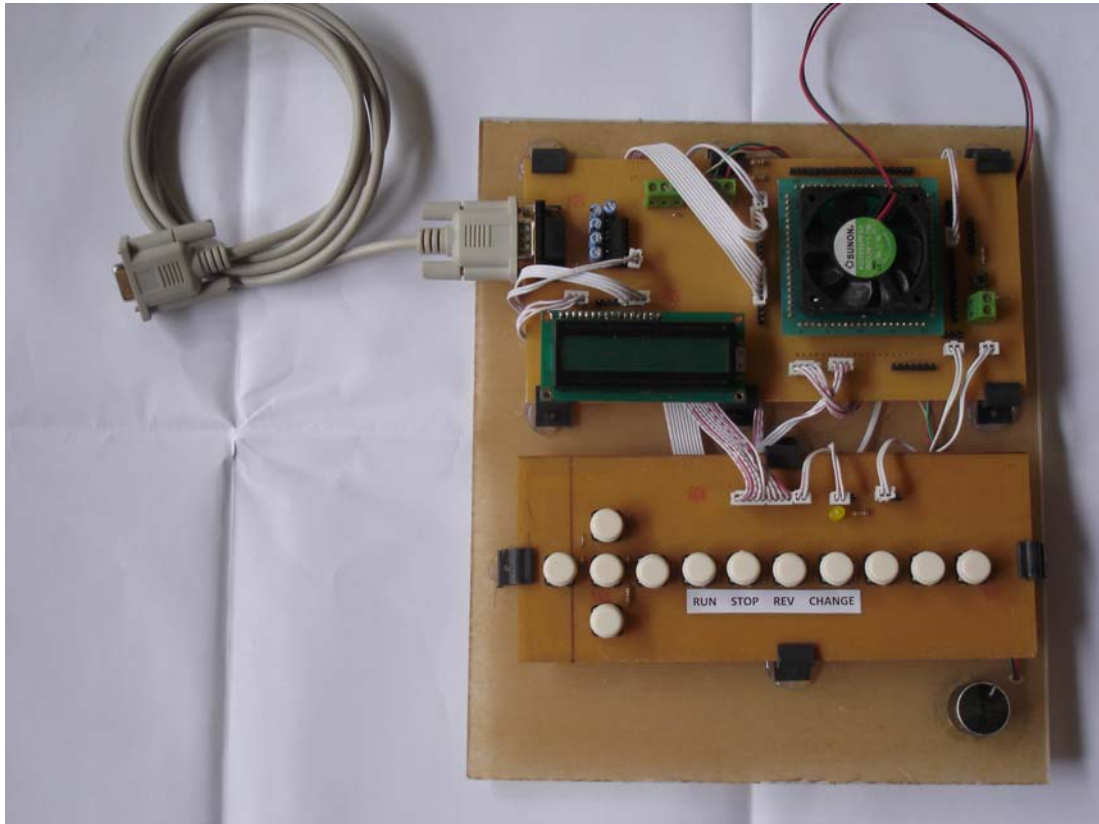
+ Giải pháp khắc phục:

Sử dụng bộ chỉnh lưu cầu 3 pha

- Nhiệt độ các khóa công suất khá cao cần thay thế các khóa công suất bằng loại chất lượng cao , đáp ứng tốt hơn .

- Xây dựng giải thuật điều khiển vòng kín (Hồi tiếp tốc độ, khâu hiệu chỉnh PID)

7.1.2 Mạch điều khiển

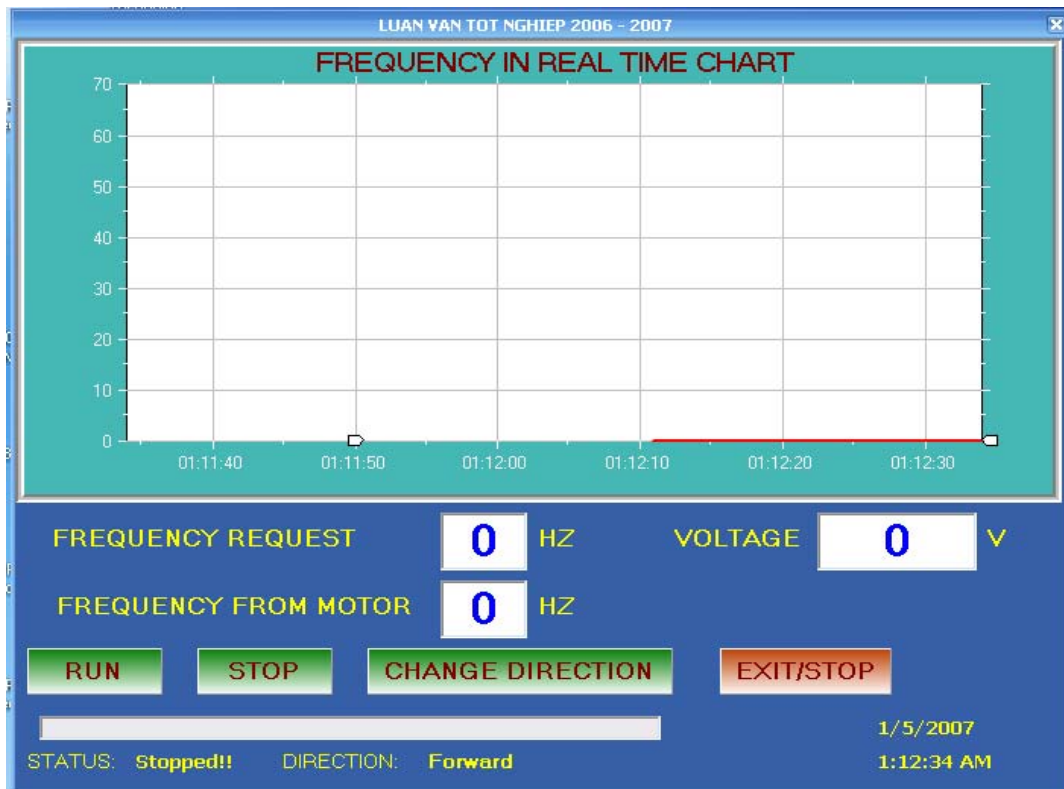


Hình 7.2: Mạch điều khiển

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

- + Các nút bấm điều khiển động cơ:
NEXT(tới),BACK(lùi),UP(lên),DOWN(xuống),ENTER(xác nhận),RUN(chạy),
STOP(dừng), REV(đảo chiều),CHANGE(thay đổi tốc độ), MENU(quay về menu
chính),RESET(reset phần mềm điều khiển),biến trở hiệu chỉnh tốc độ.....
- + Các nút bấm điều khiển LCD: cài đặt các thông số (thời gian tăng tốc, giảm tốc, cài
đặt mode, cài đặt tốc độ, di chuyển giữa các menu,)
- + LCD : hiển thị trạng thái hoạt động của động cơ (tần số, chiều quay, trạng thái hoạt
động, menu)
- + Giao tiếp với máy tính: nhận giá trị tốc độ đặt từ máy tính, hiển thị trạng thái hoạt
động của động cơ lên máy tính, vẽ giản đồ trạng thái thay đổi tần số của động cơ

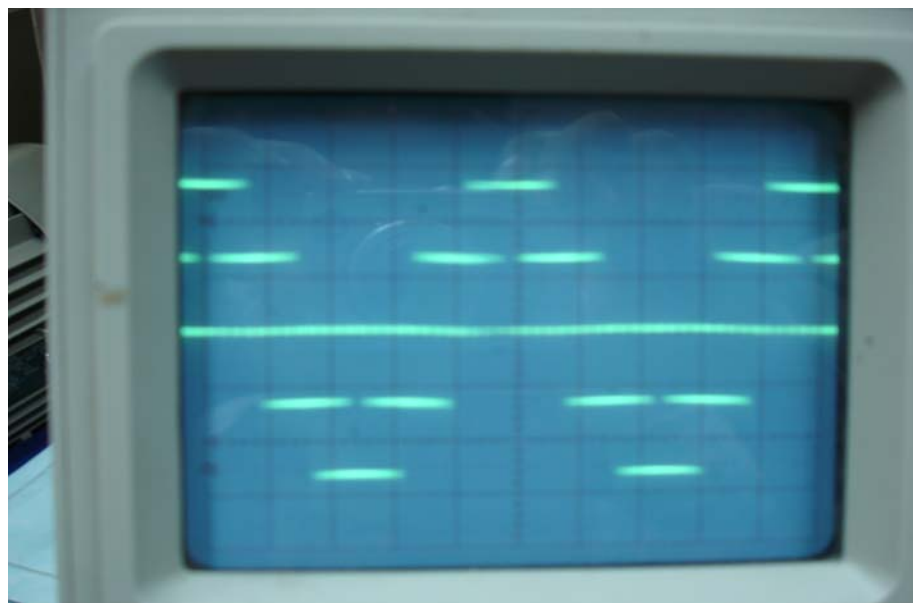
7.2 Phần mềm:



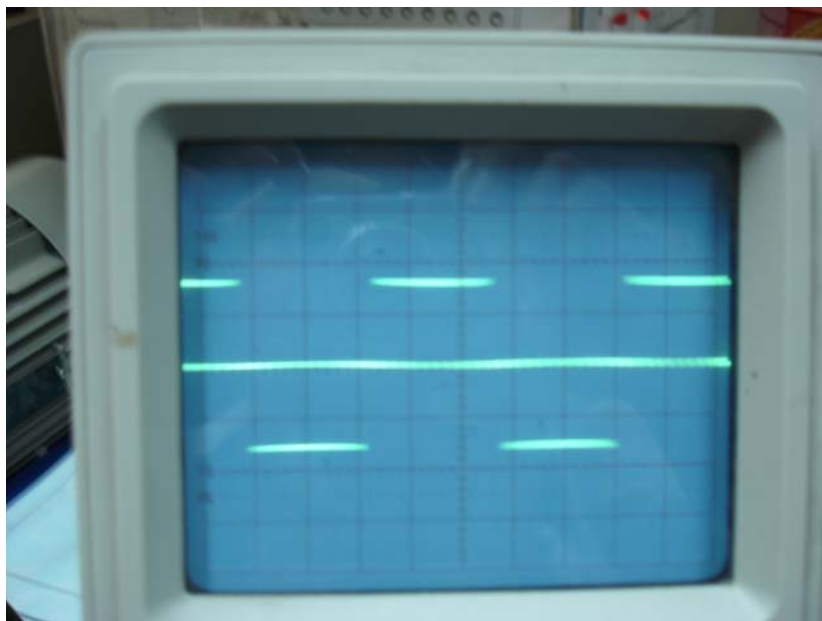
Hình 7.3: Giao diện giao tiếp máy tính

Tương tự như chế độ điều khiển tại mạch điều khiển. Chúng ta có thể điều khiển các chế độ hoạt động của motor trực tiếp trên máy tính. Đồng thời có thể quan sát được thông số ngõ ra theo thời gian thực (Tần số , điện áp tính toán)

7.3 Dạng sóng điện áp ngõ ra:



Hình 7.4: Dạng điện áp pha ngõ ra



Hình 7.5 : Dạng điện áp dây ngõ ra

PHỤ LỤC

PHỤ LỤC

CODE TRONG CHƯƠNG TRÌNH ĐIỀU KHIỂN

```
/******  
* Project:          Dieu Khiem Dong Co Khong Dong Bo          *  
* Dependencies:     Header (p30f6010.h) files                  *  
* Processor:        dsPIC30F6010                               *  
* Compiler:         MPLAB® C30 v2.02.00 or higher              *  
* IDE:              MPLAB® IDE v7.50.00 or later                *  
*****/  
#include <p30f6010.h>  
  
#include<ports.h>  
#include<adc10.h>  
#include<timer.h>  
#include<pwm.h>  
#include<math.h>  
#include<uart.h>  
#include "xlcd4bit.h"  
  
/*=====*/  
_FOSC(CSW_FSCM_OFF & XT_PLL8);  
_FWDTP(WDT_OFF);  
_FBORPOR(PBOR_ON & BORV_20 & PWRT_64 & MCLR_EN);  
_FGS(CODE_PROT_OFF);  
/*===== VARIABLE DEFINITION =====*/  
float    Float_k, temp1;  
float    Voltage_Value,ADC_Value;  
int       Voltage_Value_A,Voltage_Value_B,Voltage_Value_C;  
int       j,k,a,b,c,ADC_Result,Int_k,AD;  
float     F_req;  
int       First_Run,Run,Rev_Status,Stop;  
int       Direction,Menu_Flag,Reset;  
float     Temp_1, Temp_2, Temp_Mode_1,Temp_Value,Temp_Mode3_1,Temp_Mode3_2;  
int       Step_up,Step_down,t,ADC;  
float     up,down;  
int       PC_Value,PC_Value_Temp;  
int       int_F_req;  
float     delta;  
int       Mode;  
int       Change_Speed;  
  
//===== Button Define=====//  
#define NEXT          PORTDbits.RD8  
#define ENTER         PORTDbits.RD9  
#define UP            PORTDbits.RD10  
#define BACK          PORTDbits.RD11  
#define DOWN         PORTFbits.RF8  
#define CHANGE        PORTFbits.RF7  
  
float const SinValue[720]={0.0 , 0.00873 ,0.01745 ,0.02618 ,0.03490 ,0.04362 ,0.05234 ,0.06105 ,0.06976  
,0.07846 ,0.08716 ,  
0.09585 ,0.10453 ,0.11320 ,0.12187 ,0.13053 ,0.13917 ,0.14781 ,0.15643 ,0.16505 ,0.17365 ,0.18224 ,0.19081  
,0.19937 ,  
0.20791 ,0.21644 ,0.22495 ,0.23345 ,0.24192 ,0.25038 ,0.25882 ,0.26724 ,0.27564 ,0.28402 ,0.29237 ,0.30071  
,0.30902 ,  
0.31730 ,0.32557 ,0.33381 ,0.34202 ,0.35021 ,0.35837 ,0.36650 ,0.37461 ,0.38268 ,0.39073 ,0.39875 ,0.40674  
,0.41469 ,
```


0.42262 ,0.43051 ,0.43837 ,0.44620 ,0.45399 ,0.46175 ,0.46947 ,0.47716 ,0.48481 ,0.49242 ,0.50000 ,0.50754
 ,0.51504 ,
 0.52250 ,0.52992 ,0.53730 ,0.54464 ,0.55194 ,0.55919 ,0.56641 ,0.57358 ,0.58070 ,0.58779 ,0.59482 ,0.60182
 ,0.60876 ,
 0.61566 ,0.62251 ,0.62932 ,0.63608 ,0.64279 ,0.64945 ,0.65606 ,0.66262 ,0.66913 ,0.67559 ,0.68200 ,0.68835
 ,0.69466 ,
 0.70091 ,0.70711 ,0.71325 ,0.71934 ,0.72537 ,0.73135 ,0.73728 ,0.74314 ,0.74896 ,0.75471 ,0.76041 ,0.76604
 ,0.77162 ,
 0.77715 ,0.78261 ,0.78801 ,0.79335 ,0.79864 ,0.80386 ,0.80902 ,0.81412 ,0.81915 ,0.82413 ,0.82904 ,0.83389
 ,0.83867 ,
 0.84339 ,0.84805 ,0.85264 ,0.85717 ,0.86163 ,0.86603 ,0.87036 ,0.87462 ,0.87882 ,0.88295 ,0.88701 ,0.89101
 ,0.89493 ,
 0.89879 ,0.90259 ,0.90631 ,0.90996 ,0.91355 ,0.91706 ,0.92050 ,0.92388 ,0.92718 ,0.93042 ,0.93358 ,0.93667
 ,0.93969 ,
 0.94264 ,0.94552 ,0.94832 ,0.95106 ,0.95372 ,0.95630 ,0.95882 ,0.96126 ,0.96363 ,0.96593 ,0.96815 ,0.97030
 ,0.97237 ,
 0.97437 ,0.97630 ,0.97815 ,0.97992 ,0.98163 ,0.98325 ,0.98481 ,0.98629 ,0.98769 ,0.98902 ,0.99027 ,0.99144
 ,0.99255 ,
 0.99357 ,0.99452 ,0.99540 ,0.99619 ,0.99692 ,0.99756 ,0.99813 ,0.99863 ,0.99905 ,0.99939 ,0.99966 ,0.99985
 ,0.99996 ,
 1.00000 ,0.99996 ,0.99985 ,0.99966 ,0.99939 ,0.99905 ,0.99863 ,0.99813 ,0.99756 ,0.99692 ,0.99619 ,0.99540
 ,0.99452 ,
 0.99357 ,0.99255 ,0.99144 ,0.99027 ,0.98902 ,0.98769 ,0.98629 ,0.98481 ,0.98325 ,0.98163 ,0.97992 ,0.97815
 ,0.97630 ,
 0.97437 ,0.97237 ,0.97030 ,0.96815 ,0.96593 ,0.96363 ,0.96126 ,0.95882 ,0.95630 ,0.95372 ,0.95106 ,0.94832
 ,0.94552 ,
 0.94264 ,0.93969 ,0.93667 ,0.93358 ,0.93042 ,0.92718 ,0.92388 ,0.92050 ,0.91706 ,0.91355 ,0.90996 ,0.90631
 ,0.90259 ,
 0.89879 ,0.89493 ,0.89101 ,0.88701 ,0.88295 ,0.87882 ,0.87462 ,0.87036 ,0.86603 ,0.86163 ,0.85717 ,0.85264
 ,0.84805 ,
 0.84339 ,0.83867 ,0.83389 ,0.82904 ,0.82413 ,0.81915 ,0.81412 ,0.80902 ,0.80386 ,0.79864 ,0.79335 ,0.78801
 ,0.78261 ,
 0.77715 ,0.77162 ,0.76604 ,0.76041 ,0.75471 ,0.74896 ,0.74314 ,0.73728 ,0.73135 ,0.72537 ,0.71934 ,0.71325
 ,0.70711 ,
 0.70091 ,0.69466 ,0.68835 ,0.68200 ,0.67559 ,0.66913 ,0.66262 ,0.65606 ,0.64945 ,0.64279 ,0.63608 ,0.62932
 ,0.62251 ,
 0.61566 ,0.60876 ,0.60182 ,0.59482 ,0.58779 ,0.58070 ,0.57358 ,0.56641 ,0.55919 ,0.55194 ,0.54464 ,0.53730
 ,0.52992 ,
 0.52250 ,0.51504 ,0.50754 ,0.50000 ,0.49242 ,0.48481 ,0.47716 ,0.46947 ,0.46175 ,0.45399 ,0.44620 ,0.43837
 ,0.43051 ,
 0.42262 ,0.41469 ,0.40674 ,0.39875 ,0.39073 ,0.38268 ,0.37461 ,0.36650 ,0.35837 ,0.35021 ,0.34202 ,0.33381
 ,0.32557 ,
 0.31730 ,0.30902 ,0.30071 ,0.29237 ,0.28402 ,0.27564 ,0.26724 ,0.25882 ,0.25038 ,0.24192 ,0.23345 ,0.22495
 ,0.21644 ,
 0.20791 ,0.19937 ,0.19081 ,0.18224 ,0.17365 ,0.16505 ,0.15643 ,0.14781 ,0.13917 ,0.13053 ,0.12187 ,0.11320
 ,0.10453 ,
 0.09585 ,0.08716 ,0.07846 ,0.06976 ,0.06105 ,0.05234 ,0.04362 ,0.03490 ,0.02618 ,0.01745 ,0.00873 ,0.00000 ,
 -0.00873 ,
 -0.01745 , -0.02618 , -0.03490 , -0.04362 , -0.05234 , -0.06105 , -0.06976 , -0.07846 , -0.08716 , -0.09585 , -0.10453 ,
 -0.11320 ,
 -0.12187 , -0.13053 , -0.13917 , -0.14781 , -0.15643 , -0.16505 , -0.17365 , -0.18224 , -0.19081 , -0.19937 , -0.20791 ,
 -0.21644 ,
 -0.22495 , -0.23345 , -0.24192 , -0.25038 , -0.25882 , -0.26724 , -0.27564 , -0.28402 , -0.29237 , -0.30071 , -0.30902 ,
 -0.31730 ,
 -0.32557 , -0.33381 , -0.34202 , -0.35021 , -0.35837 , -0.36650 , -0.37461 , -0.38268 , -0.39073 , -0.39875 , -0.40674 ,
 -0.41469 ,
 -0.42262 , -0.43051 , -0.43837 , -0.44620 , -0.45399 , -0.46175 , -0.46947 , -0.47716 , -0.48481 , -0.49242 , -0.50000 ,
 -0.50754 ,

-0.51504,-0.52250,-0.52992,-0.53730,-0.54464,-0.55194,-0.55919,-0.56641,-0.57358,-0.58070,-0.58779,-0.59482,
-0.60182,-0.60876,-0.61566,-0.62251,-0.62932,-0.63608,-0.64279,-0.64945,-0.65606,-0.66262,-0.66913,-0.67559,
-0.68200,-0.68835,-0.69466,-0.70091,-0.70711,-0.71325,-0.71934,-0.72537,-0.73135,-0.73728,-0.74314,-0.74896,
-0.75471,-0.76041,-0.76604,-0.77162,-0.77715,-0.78261,-0.78801,-0.79335,-0.79864,-0.80386,-0.80902,-0.81412,
-0.81915,-0.82413,-0.82904,-0.83389,-0.83867,-0.84339,-0.84805,-0.85264,-0.85717,-0.86163,-0.86603,-0.87036,
-0.87462,-0.87882,-0.88295,-0.88701,-0.89101,-0.89493,-0.89879,-0.90259,-0.90631,-0.90996,-0.91355,-0.91706,
-0.92050,-0.92388,-0.92718,-0.93042,-0.93358,-0.93667,-0.93969,-0.94264,-0.94552,-0.94832,-0.95106,-0.95372,
-0.95630,-0.95882,-0.96126,-0.96363,-0.96593,-0.96815,-0.97030,-0.97237,-0.97437,-0.97630,-0.97815,-0.97992,
-0.98163,-0.98325,-0.98481,-0.98629,-0.98769,-0.98902,-0.99027,-0.99144,-0.99255,-0.99357,-0.99452,-0.99540,
-0.99619,-0.99692,-0.99756,-0.99813,-0.99863,-0.99905,-0.99939,-0.99966,-0.99985,-0.99996,-1.00000,-0.99996,
-0.99985,-0.99966,-0.99939,-0.99905,-0.99863,-0.99813,-0.99756,-0.99692,-0.99619,-0.99540,-0.99452,-0.99357,
-0.99255,-0.99144,-0.99027,-0.98902,-0.98769,-0.98629,-0.98481,-0.98325,-0.98163,-0.97992,-0.97815,-0.97630,
-0.97437,-0.97237,-0.97030,-0.96815,-0.96593,-0.96363,-0.96126,-0.95882,-0.95630,-0.95372,-0.95106,-0.94832,
-0.94552,-0.94264,-0.93969,-0.93667,-0.93358,-0.93042,-0.92718,-0.92388,-0.92050,-0.91706,-0.91355,-0.90996,
-0.90631,-0.90259,-0.89879,-0.89493,-0.89101,-0.88701,-0.88295,-0.87882,-0.87462,-0.87036,-0.86603,-0.86163,
-0.85717,-0.85264,-0.84805,-0.84339,-0.83867,-0.83389,-0.82904,-0.82413,-0.81915,-0.81412,-0.80902,-0.80386,
-0.79864,-0.79335,-0.78801,-0.78261,-0.77715,-0.77162,-0.76604,-0.76041,-0.75471,-0.74896,-0.74314,-0.73728,
-0.73135,-0.72537,-0.71934,-0.71325,-0.70711,-0.70091,-0.69466,-0.68835,-0.68200,-0.67559,-0.66913,-0.66262,
-0.65606,-0.64945,-0.64279,-0.63608,-0.62932,-0.62251,-0.61566,-0.60876,-0.60182,-0.59482,-0.58779,-0.58070,
-0.57358,-0.56641,-0.55919,-0.55194,-0.54464,-0.53730,-0.52992,-0.52250,-0.51504,-0.50754,-0.50000,-0.49242,
-0.48481,-0.47716,-0.46947,-0.46175,-0.45399,-0.44620,-0.43837,-0.43051,-0.42262,-0.41469,-0.40674,-0.39875,
-0.39073,-0.38268,-0.37461,-0.36650,-0.35837,-0.35021,-0.34202,-0.33381,-0.32557,-0.31730,-0.30902,-0.30071,
-0.29237,-0.28402,-0.27564,-0.26724,-0.25882,-0.25038,-0.24192,-0.23345,-0.22495,-0.21644,-0.20791,-0.19937,
-0.19081,-0.18224,-0.17365,-0.16505,-0.15643,-0.14781,-0.13917,-0.13053,-0.12187,-0.11320,-0.10453,-0.09585,
-0.08716,-0.07846,-0.06976,-0.06105,-0.05234,-0.04362,-0.03490,-0.02618,-0.01745,-0.00873 };

/*===== ADC Module Setup=====*/

void ADCSetup(void)

{

unsigned int config1, config2, config3, configport, configscan;

unsigned int channel;

ConfigIntADC10(ADC_INT_DISABLE);

channel = ADC_CH0_POS_SAMPLEA_AN6&

ADC_CH0_NEG_SAMPLEA_NVREF;

SetChanADC10(channel);

```
configport = ENABLE_AN6_ANA;
//Auto conversion trigger, auto sampling
config3 = ADC_SAMPLE_TIME_6&
          ADC_CONV_CLK_SYSTEM&
          ADC_CONV_CLK_6Tcy;
config2 = ADC_VREF_AVDD_AVSS&
          ADC_SCAN_OFF&
          ADC_CONVERT_CH0&
          ADC_SAMPLES_PER_INT_2&
          ADC_ALT_BUF_OFF&
          ADC_ALT_INPUT_OFF;
config1 = ADC_MODULE_ON&
          ADC_IDLE_CONTINUE&
          ADC_FORMAT_INTG&
          ADC_CLK_AUTO&
          ADC_AUTO_SAMPLING_ON&
          ADC_SAMPLE_SIMULTANEOUS;
configscan = SCAN_NONE;
OpenADC10(config1, config2, config3, configport, configscan);
}
/*=====PWM Module Setup=====*/
void PWMSetup(void)
{
    unsigned int config1, config2, config3;
    unsigned int period, sptime;
    unsigned int DeadTime_Config;

    //Setup deadtime for 2us
    DTCON1bits.DTBPS = 0;
    DTCON1bits.DTAPS = 0;
    DTCON1bits.DTB = 40;           //DT=DeadTime/(Prescaler*Tcy)
    DTCON1bits.DTA = 40;
    DeadTime_Config = PWM_DTS3A_UA&PWM_DTS3I_UB&
                     PWM_DTS2A_UA&PWM_DTS2I_UB&
                     PWM_DTS1A_UA&PWM_DTS1I_UB;
    SetMCPWMDeadTimeAssignment(DeadTime_Config);
    ConfigIntMCPWM(PWM_INT_EN & PWM_INT_PR7);
    period = 1999;                //1999 Fpwm=5KHz
    sptime = 0;
    config1 = PWM_EN&                //enable
             PWM_IDLE_CON&
             PWM_OP_SCALE1&        //output post scaler
             PWM_IPCLK_SCALE1&    //input prescaler
             PWM_MOD_UPDN;        //mode of operation
    config2 = PWM_MOD1_COMP&
             PWM_MOD2_COMP&
             PWM_MOD3_COMP&
             PWM_PEN1H&
             PWM_PEN2H&
             PWM_PEN3H&
             PWM_PEN1L&
             PWM_PEN2L&
             PWM_PEN3L;
    config3 = PWM_SEVOPS1&          //Special event post scaler
             PWM_OSYNC_PWM&        //output Override synchronization
             PWM_UEN;              //PWM update enable/disable

    OpenMCPWM(period, sptime, config1, config2, config3);
    DisableIntFLTA;
```

```
    DisableIntFLTB;
    PDC1=0;
    PDC2=0;
    PDC3=0;
}
void Uart_Setup(void)
{
    unsigned int baudvalue,U1MODEvalue,U1STAvalue;
    CloseUART1();
    ConfigIntUART1(UART_RX_INT_EN & UART_RX_INT_PR6 &
        UART_TX_INT_DIS & UART_TX_INT_PR1);
    baudvalue = 129;
    U1MODEvalue =UART_EN & UART_IDLE_CON &
        UART_DIS_WAKE & UART_DIS_LOOPBACK &
        UART_DIS_ABAUD & UART_NO_PAR_8BIT &
        UART_1STOPBIT;
    U1STAvalue = UART_INT_TX &
        UART_TX_PIN_NORMAL &
        UART_TX_ENABLE &
        UART_INT_RX_CHAR &
        UART_ADR_DETECT_DIS &
        UART_RX_OVERRUN_CLEAR;
    OpenUART1(U1MODEvalue, U1STAvalue, baudvalue);
}
//=====Delay Routine=====//
void Delay_Cycle(unsigned long i_cycle)
{
    unsigned long i;
    for(i=0;i<i_cycle;i++)
        asm("clrwdt");
}
//=====LCD Routine=====//
void ClearLCD(void)                // Clear LCD
{
    XlcdCursor(1,1);
    printf(" ");
    XlcdCursor(2,1);
    printf(" ");
}
void ClearLCD_1(void)              // Clear LCD at first line
{
    XlcdCursor(1,1);
    printf(" ");
}
void ClearLCD_2(void)              // Clear LCD at second line
{
    XlcdCursor(2,1);
    printf(" ");
}
//=====External Interrupt =====//
void Ext_Interrupt (void)
{
    ConfigINT3(FALLING_EDGE_INT & EXT_INT_ENABLE & EXT_INT_PRI_6);
    ConfigINT4(FALLING_EDGE_INT & EXT_INT_ENABLE & EXT_INT_PRI_5);
    ConfigINT0(FALLING_EDGE_INT & EXT_INT_ENABLE & EXT_INT_PRI_4);
    ConfigINT1(FALLING_EDGE_INT & EXT_INT_ENABLE & EXT_INT_PRI_3);
    ConfigINT2(FALLING_EDGE_INT & EXT_INT_ENABLE & EXT_INT_PRI_3);
}
//===== Update PDC Routine =====//
```

```
void Update_PDC_FW (void)
{
    a=a+k;
    if(a>719)
    {
        a=(a-720);
        Voltage_Value_A = Voltage_Value * SinValue[a];
    }
    else
    {
        Voltage_Value_A = Voltage_Value * SinValue[a];
    }

    j=a;
    b=j+240;
    if(b>719)
    {
        b=(b-720);
        Voltage_Value_B = Voltage_Value * SinValue[b];
    }
    else
    {
        Voltage_Value_B = Voltage_Value * SinValue[b];
    }

    c=j+480;
    if(c>719)
    {
        c=(c-720);
        Voltage_Value_C = Voltage_Value * SinValue[c];
    }
    else
    {
        Voltage_Value_C = Voltage_Value * SinValue[c];
    }

    PDC1 = 1999 + Voltage_Value_A;
    PDC2 = 1999 + Voltage_Value_B;
    PDC3 = 1999 + Voltage_Value_C;
}
```

```
void Update_PDC_REV (void)
{
    a=a+k;
    if(a>719)
    {
        a=(a-720);
        Voltage_Value_B= Voltage_Value * SinValue[a];
    }
    else
    {
        Voltage_Value_B = Voltage_Value * SinValue[a];
    }

    j=a;
    b=j+240;
    if(b>719)
    {
        b=(b-720);
        Voltage_Value_A = Voltage_Value * SinValue[b];
    }
}
```

```
        }
    else
    {
        Voltage_Value_A = Voltage_Value * SinValue[b];
    }

    c=j+480;
    if(c>719)
    {
        c=(c-720);
        Voltage_Value_C = Voltage_Value * SinValue[c];
    }
    else
    {
        Voltage_Value_C = Voltage_Value * SinValue[c];
    }
    PDC1 = 1999 + Voltage_Value_A;
    PDC2 = 1999 + Voltage_Value_B;
    PDC3 = 1999 + Voltage_Value_C;
}

//===================================================== Calculate Sub_routine =====//
void Step_Calculate(void)
{
    Float_k=0.144*F_req;
    Int_k=Float_k;
    temp1=Float_k-Int_k;
    if(temp1>=0.5)
    {
        k=Float_k+0.5;
    }
    else
    {
        k=Int_k;
    }
}

void Voltage_Calculate(void)
{
    Voltage_Value= F_req/0.03;

    if(    Voltage_Value >= 1900.05)        //Limit modulation amplitude at 95% to avoid
    {
        Voltage_Value=1900.05;            // dead_time induced distortion in PWM modulation
    }
}

//===================================================== Interrupt Sub_routine =====//

void __attribute__((__interrupt__)) _PWMInterrupt(void)
{
    IFS2bits.PWMIF=0;

    ADC_Value=ADC_Result*0.05859375;        //ADC_Value=((ADC_Result*60)/1024);
    if((F_req+0.000005)>ADC_Value)
    {
        F_req=F_req-down;                //SPEED DOWN
        if(F_req<=ADC_Value)
    }
```

```
        {
            F_req=ADC_Value;
            if(Rev_Status==1&&F_req==0)
            {
                Run=1; //Prepair For Speed Up, Load Value From ADC
                Direction=!Direction; //Change Direction
                Rev_Status=0;
                PC_Value=PC_Value_Temp;
            }
        }
    }
    else
    {
        F_req=F_req+up; //SPEED UP
        if(F_req>=ADC_Value)
        {
            F_req=ADC_Value;
        }
    }
    switch(Direction)
    {
        case 0: //UPDATE FOR REV
            Step_Calculate();
            Voltage_Calculate();
            Update_PDC_REV();
            break;
        case 1: //UPDATE FOR FW
            Step_Calculate();
            Voltage_Calculate();
            Update_PDC_FW();
            break;
    }
}

//===== UART1 Interrupt Sub-Routine =====//
void __attribute__((__interrupt__)) _U1RXInterrupt(void)
{
    IFS0bits.U1RXIF = 0;
    PC_Value = ReadUART1();
    if(PC_Value==70)
    {
        PC_Value=0;
        Rev_Status=1;
    }
    else
    {
        PC_Value_Temp=PC_Value;
    }
}

//===== External Interrupt Sub-Routine =====//
void __attribute__((__interrupt__)) _INT3Interrupt(void)
{
    CloseINT3();
    Run=1; //RUN
    First_Run=1;
    Direction=1;
    EnableINT3;
}

void __attribute__((__interrupt__)) _INT4Interrupt(void)
```

```

{
    CloseINT4();
    Run=0;                      //STOP
    Stop=1;
    First_Run=0;
    EnableINT4;
}
void __attribute__((__interrupt__)) _INT0Interrupt(void)
{
    CloseINT0();                // Change Direction
    Run=0;
    First_Run=0;
    Rev_Status=1;
    EnableINT0;
}
void __attribute__((__interrupt__)) _INT1Interrupt(void)
{
    CloseINT1();                // Menu
    Run=0;
    First_Run=0;
    Menu_Flag=1;
    EnableINT1;
}
void __attribute__((__interrupt__)) _INT2Interrupt(void)
{
    CloseINT2();                // Reset
    Run=0;
    First_Run=0;
    Reset=1;
    EnableINT2;
}
//===== MAIN CODE =====//
int main(void)
{
    Main:
        TRISA=0xF000;
        F_req=0;
        a=0;
        c=0;
        b=0;
        Menu_Flag=0;
        Direction=1;           //FW
        Reset=0;
        Run=0;
        Mode=1;
        Change_Speed=0;
        Step_up=10;
        Step_down=10;
        up=0.0;
        down=0.0;
        PC_Value=0;
        PC_Value_Temp=0;
        Temp_Value=0;
        Temp_Mode3_1=0;
        Temp_Mode3_2=0;

        ADCSetup();
        XlcdInit();             //SETUP 4BIT LCD
        XlcdWriteCmd(DON&CURSOR_OFF&BLINK_OFF);

```



```

    Ext_Interrupt();

    TRISFbits.TRISF8 = 1;
    TRISFbits.TRISF7 = 1;
    TRISDbits.TRISD8 = 1;
    TRISDbits.TRISD9 = 1;
    TRISDbits.TRISD10 = 1;
    TRISDbits.TRISD11 = 1;

//=====WELCOME=====//
    ClearLCD();
    XlcdCursor(1,8);
    printf("LC");
    Delay_Cycle(300000);
    XlcdCursor(1,7);
    printf("LLCO");
    Delay_Cycle(300000);
    XlcdCursor(1,6);
    printf("ELLCOM");
    Delay_Cycle(300000);
    XlcdCursor(1,5);
    printf("WELLCOME");
    Delay_Cycle(300000);
    XlcdCursor(1,4);
    printf("*WELLCOME*");
    Delay_Cycle(300000);
    XlcdCursor(1,3);
    printf("***WELLCOME***");
    Delay_Cycle(300000);
    XlcdCursor(1,2);
    printf("****WELLCOME****");
    Delay_Cycle(300000);
    XlcdCursor(1,1);
    printf("*****WELLCOME*****");
    Delay_Cycle(800000);

    XlcdCursor(2,1);
    printf("READY TO RUN");

    while(ENTER)
    {
        XlcdCursor(2,1);
        printf("READY TO RUN");
        Delay_Cycle(300000);
        ClearLCD_2();
        Delay_Cycle(300000);
    }
    Delay_Cycle(1000000);
LCD_Loop:
    ClearLCD();
    XlcdCursor(1,1);
    printf("Speed_Up:  s");
    XlcdCursor(2,1);
    printf("Speed_Down:  s");
    while(ENTER)
    {
        if(BACK==0)
        {
            Delay_Cycle(800000);

```

```
        goto LCD_Loop;
    }
    if(NEXT==0)
    {
        Delay_Cycle(800000);
        goto MODE_STATUS;
    }
    if(UP==0)
    {
        Delay_Cycle(150000);
        if(UP==0)
        {
            Step_up=Step_up+1;
            if(Step_up>20)
            {
                Step_up=3;
            }
            Delay_Cycle(100000);
        }
    }
    if(DOWN==0)
    {
        Delay_Cycle(150000);
        if(DOWN==0)
        {
            Step_up=Step_up-1;
            if(Step_up<3)
            {
                Step_up=20;
            }
            Delay_Cycle(100000);
        }
    }
    XlcdCursor(1,12);
    printf("%.2.0d",Step_up);
}
Delay_Cycle(700000);          //Waitting for ENTER go back to 1
while(ENTER)
{
    if(BACK==0)
    {
        Delay_Cycle(800000);
        goto LCD_Loop;
    }
    if(NEXT==0)
    {
        Delay_Cycle(800000);
        goto MODE_STATUS;
    }
    if(UP==0)
    {
        Delay_Cycle(150000);
        if(UP==0)
        {
            Step_down=Step_down+1;
            if(Step_down>20)
            {
                Step_down=3;
            }
        }
    }
}
```

```
                Delay_Cycle(100000);
            }
        }
        if(DOWN==0)
        {
            Delay_Cycle(150000);
            if(DOWN==0)
            {
                Step_down=Step_down-1;
                if(Step_down<3)
                {
                    Step_down=20;
                }
                Delay_Cycle(150000);
            }
        }
        XlcdCursor(2,12);
        printf("%2.0d",Step_down);
    }
    Delay_Cycle(300000);

    ClearLCD();
    XlcdCursor(1,1);
    printf(" IN_PUT DATA. ");
    XlcdCursor(2,1);
    printf("Loading");
    Delay_Cycle(300000);
    XlcdCursor(2,8);
    printf("..");
    Delay_Cycle(300000);
    XlcdCursor(2,10);
    printf("..");
    Delay_Cycle(300000);
    XlcdCursor(2,12);
    printf("..");
    Delay_Cycle(300000);
    XlcdCursor(2,14);
    printf("...");
    Delay_Cycle(300000);
    XlcdCursor(2,1);
    Delay_Cycle(1000000);

//Calculate Speed_up and Speed_Down time for the motor

    up=0.012/Step_up;                // Time_up=(60*Tpwm)/Tup
    down=0.012/Step_down;            // Time_down=(60*Tpwm)/Tdown

//===== MODE CHECKING =====//
MODE_STATUS:
    ClearLCD();
    switch (Mode)
    {
        case 1:                //Manual Mode is chosen
            ClearLCD();
            XlcdCursor(1,1);
            printf("< M1_MANUAL >");
            XlcdCursor(2,14);
            printf("OK");
```

```
while(1)
{
    if(BACK==0)
    {
        Delay_Cycle(800000);
        goto LCD_Loop;
    }
    if(ENTER==0)
    {
        Delay_Cycle(50000);
        if(ENTER==0)
        {
            XlcdCursor(2,14);
            printf("RUN");
            Delay_Cycle(300000);
            goto LOAD_PARAMETER;
        }
    }
    if(UP==0)
    {
        Delay_Cycle(50000);
        if(UP==0)
        {
            Mode=2;
            Delay_Cycle(300000);
            goto MODE_STATUS;
        }
    }
    if(DOWN==0)
    {
        Delay_Cycle(50000);
        if(DOWN==0)
        {
            Mode=4;
            Delay_Cycle(300000);
            goto MODE_STATUS;
        }
    }
}
break;
case 2: // AUTO_1 is chosen
ClearLCD();
XlcdCursor(1,1);
printf("< M2_AUTO_1 >");
XlcdCursor(2,14);
printf("OK");
while(1)
{
    if(BACK==0)
    {
        Delay_Cycle(800000);
        goto LCD_Loop;
    }
    if(ENTER==0)
    {
        Delay_Cycle(50000);
        if(ENTER==0)
        {
            Delay_Cycle(300000);
```

```
                                goto LOAD_PARAMETER;
                                }
                                }
                                if(UP==0)
                                {
                                    Delay_Cycle(50000);
                                    if(UP==0)
                                    {
                                        Mode=3;
                                        Delay_Cycle(300000);
                                        goto MODE_STATUS;
                                    }
                                }
                                }
                                if(DOWN==0)
                                {
                                    Delay_Cycle(50000);
                                    if(DOWN==0)
                                    {
                                        Mode=1;
                                        Delay_Cycle(300000);
                                        goto MODE_STATUS;
                                    }
                                }
                                }
                                break;
case 3:                                // AUTO_2 is chosen
    ClearLCD();
    XlcdCursor(1,1);
    printf("< M3_AUTO_2 >");
    XlcdCursor(2,14);
    printf("OK");
    while( 1)
    {
        if(BACK==0)
        {
            Delay_Cycle(800000);
            goto LCD_Loop;
        }
        if(ENTER==0)
        {
            Delay_Cycle(50000);
            if(ENTER==0)
            {
                Delay_Cycle(300000);
                goto LOAD_PARAMETER;
            }
        }
        if(UP==0)
        {
            Delay_Cycle(50000);
            if(UP==0)
            {
                Mode=4;
                Delay_Cycle(250000);
                goto MODE_STATUS;
            }
        }
        if(DOWN==0)
        {
```

```

        Delay_Cycle(50000);
        if(DOWN==0)
        {
            Mode=2;
            Delay_Cycle(300000);
            goto MODE_STATUS;
        }
    }
}
break;
case 4: // PC_Control is chosen
    ClearLCD();
    XlcdCursor(1,1);
    printf("< M4_PC_CON >");
    XlcdCursor(2,14);
    printf("OK");
    while( 1)
    {
        if(BACK==0)
        {
            Delay_Cycle(800000);
            goto LCD_Loop;
        }
        if(ENTER==0)
        {
            Delay_Cycle(50000);
            if(ENTER==0)
            {
                Delay_Cycle(300000);
                goto LOAD_PARAMETER;
            }
        }
        if(UP==0)
        {
            Delay_Cycle(50000);
            if(UP==0)
            {
                Mode=1;
                Delay_Cycle(250000);
                goto MODE_STATUS;
            }
        }
        if(DOWN==0)
        {
            Delay_Cycle(50000);
            if(DOWN==0)
            {
                Mode=3;
                Delay_Cycle(300000);
                goto MODE_STATUS;
            }
        }
    }
}
break;
} //End Of Switch

//===================================================== LOAD PARAMETER =====//
LOAD_PARAMETER:
    ClearLCD();
    if(Mode==3)

```

```

{
    Temp_1=Temp_Mode3_1;           //Restore Temp_1 Value
    Temp_2=Temp_Mode3_2;           //Restore Temp_2 Value
    XlcdCursor(1,1);
    printf("  AUTO_2  ");
    XlcdCursor(2,1);
    printf("F1:  F2:  ");

Mode3_Loop_1:
    while(ENTER)
    {
        if(BACK==0)
        {
            Delay_Cycle(800000);
            goto MODE_STATUS;
        }
        if(UP==0)
        {
            Delay_Cycle(150000);
            if(UP==0)
            {
                Temp_1=Temp_1+1;
                if(Temp_1>60)    // Limit frequency under 60Hz
                {
                    Temp_1=0;
                }
                Delay_Cycle(100000);
            }
        }
        if(DOWN==0)
        {
            Delay_Cycle(150000);
            if(DOWN==0)
            {
                Temp_1=Temp_1-1;
                if(Temp_1<0)
                {
                    Temp_1=60;
                }
                Delay_Cycle(100000);
            }
        }
        XlcdCursor(2,4);
        printf("%.2.0lf",Temp_1);
    } //End of while
    Temp_Mode3_1=Temp_1;           // Save Temp_1 Value
    Delay_Cycle(600000);           // Waiting for Enter back to 1

Mode3_Loop_2:
    while(ENTER)
    {
        if(BACK==0)
        {
            Delay_Cycle(800000);
            XlcdCursor(2,13);
            printf("  ");
            goto Mode3_Loop_1;
        }
        if(UP==0)
        {
            Delay_Cycle(150000);

```

```
        if(UP==0)
        {
            Temp_2=Temp_2+1;
            if(Temp_2>60)
            {
                Temp_2=0;
            }
            Delay_Cycle(100000);
        }
    }
    if(DOWN==0)
    {
        Delay_Cycle(150000);
        if(DOWN==0)
        {
            Temp_2=Temp_2-1;
            if(Temp_2<0)
            {
                Temp_2=60;
            }
            Delay_Cycle(100000);
        }
    }
    XlcdCursor(2,13);
    printf("%2.0lf",Temp_2);
} //End of While
Temp_Mode3_2=Temp_2; //Save Temp_2 Value
ClearLCD_2();
XlcdCursor(2,1);
printf("Loading");
Delay_Cycle(300000);
XlcdCursor(2,8);
printf("..");
Delay_Cycle(300000);
XlcdCursor(2,10);
printf("..");
Delay_Cycle(300000);
XlcdCursor(2,12);
printf("..");
Delay_Cycle(300000);
XlcdCursor(2,14);
printf("...");
Delay_Cycle(300000);
XlcdCursor(2,1);
printf("      RUN");
while(!Run)
{
    if(BACK==0)
    {
        Delay_Cycle(800000);
        ClearLCD_2();
        XlcdCursor(2,1);
        printf("F1:  F2:  ");
        XlcdCursor(2,4);
        printf("%2.0lf",Temp_1);
        goto Mode3_Loop_1;
    }
}
ClearLCD();
```



```
        XlcdCursor(1,1);
        printf("AUTO_2");
        XlcdCursor(1,10);
        printf("F1:%2.0lf",Temp_1);
        XlcdCursor(2,1);
        printf("F:");
        XlcdCursor(2,10);
        printf("F2:%2.0lf",Temp_2);

        Temp_1=Temp_1/0.05859375;
        Temp_2=Temp_2/0.05859375;
        goto loop1;
    }//end of if mode 3
    if(Mode==1)
    {
        ClearLCD();
        XlcdCursor(1,1);
        printf("  MANUAL  ");
        XlcdCursor(2,14);
        printf("RUN");
        XlcdCursor(2,1);
        printf("AD:");
        while(!Run)
        {
            AD=ReadADC10(0);
            ADC=AD*0.05859375;
            XlcdCursor(2,4);
            printf("%2.0d",ADC);          //Print Value from Variable Resistor
            if(BACK==0)
            {
                Delay_Cycle(800000);
                goto MODE_STATUS;
            }
        }
        ClearLCD();
        XlcdCursor(1,1);
        printf("M1  RUNNING...");
        XlcdCursor(2,1);
        printf("F:");
        goto loop1;
    }//end if mode 1
    if(Mode==2)
    {
        ClearLCD();
        //Temp_Mode_1=0;
        XlcdCursor(1,1);
        printf("  AUTO_1  ");
        XlcdCursor(2,1);
        printf("F_Ref:");

Mode2_Loop:

        Temp_Mode_1=Temp_Value;    //Restore Temp_Mode_1 Value
        while(ENTER)
        {
            if(BACK==0)
            {
                Delay_Cycle(800000);
                goto MODE_STATUS;
            }
            if(UP==0)
```

```
{
    Delay_Cycle(150000);
    if(UP==0)
    {
        Temp_Mode_1=Temp_Mode_1+1;
        if(Temp_Mode_1>60)
        {
            Temp_Mode_1=0;
        }
        Delay_Cycle(100000);
    }
}
if(DOWN==0)
{
    Delay_Cycle(150000);
    if(DOWN==0)
    {
        Temp_Mode_1=Temp_Mode_1-0.5;
        if(Temp_Mode_1<0)
        {
            Temp_Mode_1=60;
        }
        Delay_Cycle(100000);
    }
}
//Temp_Value=Temp_Mode_1;
XlcdCursor(2,7);
printf("%2.0lf ", Temp_Mode_1);
} //end while
Temp_Value=Temp_Mode_1; //Save Temp_Mode_1 Value
ClearLCD_2();
XlcdCursor(2,1);
printf("Loading");
Delay_Cycle(300000);
XlcdCursor(2,8);
printf("..");
Delay_Cycle(300000);
XlcdCursor(2,10);
printf("..");
Delay_Cycle(300000);
XlcdCursor(2,12);
printf("..");
Delay_Cycle(300000);
XlcdCursor(2,14);
printf("...");
Delay_Cycle(300000);
ClearLCD_2();
XlcdCursor(2,1);
printf("F:%2.0lf",Temp_Mode_1);

XlcdCursor(2,14);
printf("RUN");

Temp_Mode_1=Temp_Mode_1/0.05859375;

while(!Run)
{
    XlcdCursor(2,14); //Blink "RUN" character till it is pressed
    printf("RUN");
```

```
        Delay_Cycle(200000);
        XlcdCursor(2,14);
        printf(" ");
        Delay_Cycle(200000);
        if(BACK==0)
        {
            Delay_Cycle(500000);
            ClearLCD_2();
            XlcdCursor(2,1);
            printf("F_Ref:");
            XlcdCursor(2,7);
            printf("%2.0lf",Temp_Mode_1);
            goto Mode2_Loop;
        }
    }
    XlcdCursor(2,14);
    printf(" ");
    XlcdCursor(1,1);
    printf("M2  RUNNING...");
    XlcdCursor(2,9);
    printf("F_Req:%2.0lf",Temp_Value);
    goto loop1;
} //end if mode 2
if(Mode==4)
{
    ClearLCD();
    XlcdCursor(1,1);
    printf(" PC_CONTROL ");
    XlcdCursor(2,14);
    printf("RUN");

    while(!Run)
    {
        if(BACK==0)
        {
            Delay_Cycle(700000);
            goto MODE_STATUS;
        }
    }
    ClearLCD_2();
    XlcdCursor(2,1);
    printf("M2  RUNNING");
    PC_Value=0;
    Uart_Setup();
    goto loop1;
} //end if mode 4

loop1:
PWMSetup();
while(1)
{
    switch(Run)
    {
        //Waiting for Run Button Is Pressed
        case 1:
            //Run Button is not pressed
            switch(Mode)
            {
                case 1:
                    ADC_Result=ReadADC10(0);
                    XlcdCursor(2,3);
```

```
        ADC_Result=ReadADC10(0);
        printf("%2.0lf", F_req);
break;
case 2:
    ADC_Result=Temp_Mode_1;
    XlcdCursor(2,3);
    printf("%2.0lf", F_req);
break;
case 3:
    if(CHANGE==0)
    {
        Delay_Cycle(70000);
        if(CHANGE==0);
        {
            Change_Speed=!Change_Speed;
            Delay_Cycle(70000);
        }
    }
    if(Change_Speed==1)
    {
        ADC_Result=Temp_2;
    }
    else
    {
        ADC_Result=Temp_1;
    }
    XlcdCursor(2,3);
    printf("%2.0lf", F_req);
break;
case 4:
    ADC_Result=PC_Value/0.05859375;
    int_F_req=F_req;
    delta=F_req-int_F_req;
    if(delta>0.5)
    {
        int_F_req=int_F_req+1;
    }
    XlcdCursor(2,1);
    printf("F:%2.0lf", F_req);
    WriteUART1(int_F_req);
break;
}
break;
case 0: //Run Button is not pressed
    ADC_Result=0;
    XlcdCursor(2,3);
    printf("%2.0lf", F_req);
    if(F_req==0 && Stop==1)
    {
        CloseMCPWM();// Disable PWM module
        Stop=0;
        Change_Speed=0;
        Delay_Cycle(800000);
        goto LOAD_PARAMETER;
    }
    if(F_req==0 && Menu_Flag==1)
    {
        CloseMCPWM(); // Disable PWM module
        Menu_Flag=0; // Clear Menu_Flag
    }
}
```

```

        Change_Speed=0;
        Delay_Cycle(1000000);
        PC_Value=0;
        PC_Value_Temp=0;
        Direction=1;
        goto LCD_Loop; //Return to main menu
    }
    if(F_req==0 && Reset==1)
    {
        CloseMCPWM();// Disable PWM module
        Reset=0;           // Clear Menu_Flag
        Change_Speed=0;
        ClearLCD();
        XlcdCursor(1,1);
        printf("RESET .");
        Delay_Cycle(400000);
        XlcdCursor(1,8);
        printf("..");
        Delay_Cycle(400000);
        printf("..");
        Delay_Cycle(400000);
        XlcdCursor(1,12);
        printf("..");
        Delay_Cycle(400000);
        XlcdCursor(1,14);
        printf("...");
        Delay_Cycle(1000000);
        goto Main;       //Return to main menu
    }
    break;
}
}
}
}

```

CODE TRONG CHƯƠNG TRÌNH GIAO TIẾP MÁY TÍNH

```

Dim Y As Double           'variable in chart drawing
Public Running As Integer
Public dir As Integer
Public StsRun As Byte
Dim chuoichay As String
Dim Voltage_Temp As Integer
Dim strtemp As String 'variable ONCOMM event
Dim strdata As String
Dim datavu As String
Dim intdigvu As Integer
Dim digdata As Integer

```

```

=====
Private Sub Change_Button_Click()
MSComm1.Output = Chr(70)
txt_f_request.SetFocus
If dir = 0 Then
    dir = 1
Else
    dir = 0
End If
End Sub

```

```

=====
Private Sub Exit_Button_Click()

```

PHỤ LỤC

```
MSComm1.Output = Chr(0)      'send stop signal for PIC to stop motor
MSComm1.PortOpen = False     'Dong cong
End
End Sub
```

```
Private Sub Form_Load()
    chuoi chay = "LUAN VAN TOT NGHIEP 2006 - 2007"
"
'Close Serial Port if it have been already opened
    If frmMain.MSComm1.PortOpen = True Then
        frmMain.MSComm1.PortOpen = False
    End If
    frmMain.MSComm1.PortOpen = True
    MSComm1.Output = Chr(0)
    frmMain.MSComm1.PortOpen = False
' 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
    Timer2.Enabled = True
    Timer2.Interval = 1000
    Label8.Caption = Date
```

```
' 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
    '.1 seconds
    Strip1.VariableDeltaX = 1 / 24 / 60 / 60 / 10      '.1 seconds interval
    Strip1.VariableLastX = Xx 'Set LastX to current time
Next
Strip1.XTicMode = 1 'Set X Mode to Date/Time Display
'60 seconds
Strip1.XSpan = 1 / 24 / 60 / 60 * 60                  '60 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)
                digdata = Asc(strdata)
                intdigvu = Asc(datavu)
                txt_f_out = digdata
```

PHỤ LỤC

```
        Voltage_Temp = digdata * 3.667
        Voltage = Voltage_Temp
    End Select
End With
End Sub

=====
Private Sub ProgressBar1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)

End Sub

=====
Private Sub Timer1_Timer()
    Strip1.AddXY 0, Now, Y
    Y = digdata
End Sub

=====
Private Sub cmdStart_Click()
    j = txt_f_request.Text
    If (j > 60) Then
        MsgBox ("Frequency must be in range from 0 to 60 Hz")
        txt_f_request = ""
        txt_f_request.SetFocus
    Else
        'Status = "Motor Is Running ....."
        StsRun = 1
        MSComm1.Output = Chr(j)
        txt_f_request.SetFocus
    End If
End Sub

=====
Private Sub Cmd_STOP_Click()
    MSComm1.Output = Chr(0)
    Status = "Stopping"
    StsRun = 0
    txt_f_request.SetFocus
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

=====
Private Sub Timer2_Timer()
    Label7.Caption = Time
End Sub

=====
Private Sub Timer3_Timer()
    If StsRun = 1 Then
        Running = Running + 5
        If (Running > 100) Then
            Running = RunProgressBar.Min
        End If
        RunProgressBar.Value = Running + RunProgressBar.Min
    Else
        If txt_f_out = 0 Then
            RunProgressBar.Value = 0
        Else
            Running = Running + 5
            If (Running > 100) Then
                Running = RunProgressBar.Min
            End If
        End If
    End If
End Sub
```

PHỤ LỤC

```
End If
RunProgressBar.Value = Running + RunProgressBar.Min
End If
End If
End Sub
```

```
Private Sub Timer4_Timer()
If (StsRun = 1) Then
    Status = "Running"
End If
If (txt_f_out = 0) Then
    Status = "Stopped!!"
End If
If dir = 0 Then
    Direction = "Forward"
Else
    Direction = "Reverse"
End If
End Sub
```

```
Private Sub Timer5_Timer()
Dim chuoi1, chuoi2 As String
chuoi1 = Left(chuoichay, 1)
chuoi2 = Right(chuoichay, Len(chuoichay) - 1)
frmMain.Caption = chuoi2 + chuoi1
chuoichay = chuoi2 + chuoi1
End Sub
```


TÀI LIỆU THAM KHẢO

TÀI LIỆU THAM KHẢO TRONG NƯỚC

- [1] TS. Phan Quốc Dũng – Tô Hữu Phước (2003). *Truyền Động Điện*. Nhà xuất bản Đại học Quốc gia TP.Hồ Chí Minh
- [2] TS. Nguyễn Văn Nhờ (2003). *Cơ Sở Truyền Động Điện*. Nhà xuất bản Đại học Quốc gia TP.Hồ Chí Minh
- [3] TS. Nguyễn Văn Nhờ (2003). *Điện Tử Công Suất 1*. Nhà xuất bản Đại học Quốc gia TP.Hồ Chí Minh

TÀI LIỆU THAM KHẢO NƯỚC NGOÀI

- [4] *16bit_Language_Tools_Libraries*. Microchip Technology Inc
- [5] Prof. Ali Keyhani, *Pulse-Width Modulation (PWM) Techniques – lecture 25*, Department of Electrical and Computer EngineeringThe Ohio State University
- [6] *dsPic® Language Tools Getting Started*. Microchip Technology Inc
- [7] *MPLAB® C30 _ C Compiler User's Guide* . Microchip Technology Inc
- [8] *dsPIC30F6010 Data Sheet High-Performance Digital Signal Controllers* . Microchip Technology Inc
- [9] *dsPIC30F Family Reference Manual*. Microchip Technology Inc
- [10] *An Introduction To AC Induction Motor Control Using The dsPIC30F MCU*. Microchip Technology Inc

WEBSITE THAM KHẢO

<http://www.microchip.com>