วิธีการออกแบบโดยใช้การควบคุมสัญญาณพัลส์วิดธ์มอดูเลชั่นแบบเวกเตอร์แรงดัน สำหรับควบคุมมอเตอร์ชนิดเหนี่ยวนำ 3 เฟส

ALGORITHM DESIGN APPROACH THE METHOD OF SPACE-VECTOR PWM FOR A THREE PHASE INDUCTION MOTER CONTROL

โดย

นายกฤษฎา กรดเต็ม

นายอาจณรงค์ พิพัฒน์ไพศาล

นายอำนวย ประมงค์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2551

วิธีการออกแบบโดยใช้การควบคุมสัญญาณพัลส์วิดธ์มอดูเลชั่นแบบเวกเตอร์แรงดัน สำหรับควบคุมมอเตอร์ชนิดเหนี่ยวนำ 3 เฟส

ALGORITHM DESIGN APPROACH THE METHOD OF SPACE-VECTOR PWM FOR A THREE PHASE INDUCTION MOTER CONTROL

โดย

นายกฤษฎา	กรดเต็ม	เลขประจำตัว 49015225
นายอาจณรงค์	พิพัฒน์ไพศาล	เลขประจำตัว 49015265
นายอำนวย	ประมงค์	เลขประจำตัว 49015267

อาจารย์ที่ปรึกษา

อาจารย์อิทธิภูมิ บุญพิคำ

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาอิเล็กทรอนิกส์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2551

ปริญญานิพนธ์ปีการศึกษา : 2551

ภาควิชา : อิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง : วิธีการออกแบบโดยใช้การควบคุมสัญญาณพัลส์วิดธ์มอดูเลชั่นแบบเวกเตอร์แรงดันสำหรับควบคุมมอเตอร์ชนิดเหนี่ยวนำ 3 เฟส

ผู้จัดทำ

1. นายกฤษฎา กรดเต็ม
2. นายอาจณรงค์ พิพัฒน์ไพศาล
3. นายอำนวย ประมงค์

..... **อาจารย์ที่ปรึกษา**

(.....อิทธิพล ประสงค์.....)

วิธีการออกแบบโดยใช้การควบคุมสัญญาณพัลส์วิดธ์มอดูเลชั่นแบบเวกเตอร์แรงดันสำหรับควบคุมมอเตอร์ชนิดเหนี่ยวนำ 3 เฟส

นาย กฤษฎา กรดเต็ม รหัส 49015225
นาย อาจณรงค์ พิพัฒน์ไพศาล รหัส 49015265
นาย อำนวย ประมงค์ รหัส 49015267
ดร.อิทธิภูมิ บุญพิคำ อาจารย์ที่ปรึกษา
ปีการศึกษา 2551

บทคัดย่อ

เสนอวิธีการออกแบบวิธีทางสเปซเวกเตอร์พัลส์วิดธ์มอดูเลชั่น สำหรับควบคุมมอเตอร์ชนิดเหนี่ยวนำ 3 เฟส เริ่มจากการวิเคราะห์ทางคณิตศาสตร์ ประมวลผลโดยโปรแกม MATLAB พร้อมด้วย GUI (การเชื่อมแผนภาพกับผู้ใช้) ซึ่งยังเป็นการศึกษาวิธีการควบคุมทางเวกเตอร์และสามารถเห็นทั้งเวกเตอร์และเรื่องของเวลาได้ โดยตัวแปรทั้งสองที่กล่าวนี้คือขนาดและเวลาของเวกเตอร์ที่ทำการควบคุมมอเตอร์ที่สำคัญ หลังจากนั้นจะทำการส่งผ่านระหัสและรูปแบบการควบคุมไปยังไมโครคอนโทรลเลอร์ที่ได้รับการโปรแกรมโดยโปรแกรมภาษาซี และเพื่อยืนยันผลการจำลองที่ได้ได้แสดงผลการทดสอบขับและควบคุมมอเตอร์เหนี่ยวนำ 3 เฟส กำลัง 1 แรงม้า ไว้ด้วย

คำสำคัญ อินเวอร์เตอร์ สเปซเวกเตอร์

ALGORITHM DESIGN APPROACH THE METHOD OF SPACE-VECTOR PWM FOR A THREE PHASE INDUCTION MOTER CONTROL

Mr. Kridsada Krodtem ID.49015225
Mr. Ardnarong Pipatpaisarn ID.49015265
Mr. Amnuai Pramong ID.49015267
Dr. Ittibhoom Boonpikum Advisor
Educational Year 2008

Abstract

Algorithm Design to Approach the Method of Space-vector (SV) pulse width modulation (PWM) for a Three Phase Induction Motor Control is dealing with an algorithm of space vector PWM for three phase induction motor control. Starting with Mathematics analysis, then MATLAB simulation with GUI (graphic user interface), we understand the method of vector control and can see both vector and timing. Two variables, as magnitude and timing of an on-off vector are important. After simulation by MATLAB, we transfer code and algorithm to a micro controller implementation (dspic30f4011) programmed by C language. To verify the result of simulations, tests of driving and controlling a three phase induction motor, 1 horse power will be performed.

Keyword Inverter SpaceVector

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลงด้วยดี โดยได้รับความช่วยเหลือจากอาจารย์ที่ปรึกษาปริญญานิพนธ์ อาจารย์ ดร.อิทธิภูมิ บุญพิคำ ที่ได้กรุณาให้คำปรึกษาในการแก้ไขปัญหาต่าง ๆ ระหว่างการทำวิจัย รวมทั้งการตรวจสอบและแก้ไขข้อบกพร่อง จนปริญญานิพนธ์ฉบับนี้สำเร็จลงได้ด้วยดี ผู้วิจัยขอกราบขอบพระคุณมา ณ ที่นี้

ขอกราบขอบพระคุณ อาจารย์สาทร ทองถึง ที่ได้กรุณาให้คำปรึกษาในการแก้ไขปัญหาต่างๆ ระหว่างการทำวิจัย จนปริญญานิพนธ์ฉบับนี้สำเร็จลงด้วยดี

ขอกราบขอบพระคุณบริษัท ไพรมัส จำกัด ที่ได้ช่วยเหลือทางด้านฮาร์ดแวร์ ในการทำวิจัยตลอดจนพี่ๆ ในแผนกวิจัยทุกคน ที่ได้ให้คำปรึกษาในการแก้ไขปัญหาต่างๆ และเป็นกำลังใจในระหว่างการทำวิจัย จนปริญญานิพนธ์ฉบับนี้สำเร็จลงด้วยดี

ท้ายสุดนี้ผู้วิจัยขอกราบขอบพระคุณ บิดา-มารดา ที่คอยช่วยเหลือ และเป็นกำลังใจ และให้ความสนับสนุนในด้านปัจจัยในการดำเนินการทำปริญญานิพนธ์ฉบับนี้

นายอาจณรงค์ พิพัฒน์ไพศาล
นายอำนวย ประมงค์
นายกฤษฎา กรดเต็ม

สารบัญ

	หน้า
บทคัดย่อภาษาไทย	ก
บทคัดย่อภาษาอังกฤษ	ข
กิตติกรรมประกาศ	ค
สารบัญภาพ	ง
สารบัญตาราง	ช
บทที่ 1 บทนำ	1
1.1 ความเป็นมาของการทำปริญญานิพนธ์หัวข้อนี้	1
1.2 โครงสร้างของอินเวอร์เตอร์	1
1.3 หลักการทำงานของอินเวอร์เตอร์อย่างง่าย	2
1.4 วัตถุประสงค์	3
1.5 ขอบเขตของงานวิจัย	3
1.6 ขั้นตอนและวิธีดำเนินการ	3
1.7 ประโยชน์ที่คาดว่าจะได้รับ	3
บทที่ 2 หลักการสัญญาณพีดับบลิวเอ็มด้วยวิธีสเปซเวกเตอร์แรงดัน	4
2.1 หลักการสร้างสัญญาณ PWM โดยวิธีสเปซเวกเตอร์	4
บทที่ 3 วิธีการออกแบบ Space Vector PWM	11
3.1 ศึกษากระบวนการทำงานพื้นฐานของ PWM ด้วย MATLAB	11
3.2 การออกแบบชุดคำสั่ง (Software) บนไมโครคอนโทรลเลอร์	15
3.3 การออกแบบชุดวงจร (Hardware)	23
บทที่ 4 ผลการทดสอบการสร้างสัญญาณ PWM	25
4.1 ผลการทดสอบการสร้างสัญญาณ PWM	25
4.2 ผลการทดสอบวัดแรงดันระหว่างสายและแรงดันเฟส	28
บทที่ 5 สรุปผลและข้อเสนอแนะ	31
5.1 สรุปผลการทดสอบ	31
5.2 ข้อเสนอแนะในการปรับปรุงระบบในอนาคต	31

เอกสารอ้างอิง 32
ภาคผนวก ก. ไมโครคอนโทรลเลอร์ dsPIC30F4011 33
ภาคผนวก ข. Intelligent Power Module (IPM) เบอร์ PS11034 61
ภาคผนวก ค. ชุดคำสั่งภาษาซี (Software) 66
ภาคผนวก ง. โฟร์ชาร์ตการทำงานของโปรแกรม 79

สารบัญรูปภาพ

รูปที่ หน้า

1.1 โครงสร้างของอินเวอร์เตอร์ 1
1.2 วิธีสร้างไฟสลับอย่างง่าย 2
1.3 คลื่นไฟสลับ 2
2.1 อินเวอร์เตอร์สามเฟส 4
2.2 ความสัมพันธ์ระหว่างแกน U, V, W กับแกนอ้างอิง d-q 6
2.3 ความสัมพันธ์ระหว่างแกน U, V, W กับแกนอ้างอิง d-q ในเซกเตอร์ที่ 1 6
2.4 แสดงการคำนวณเวกเตอร์เพื่อหาค่าคาบเวลาการ เปิด-ปิด ของสวิทช์ในเซกเตอร์ที่ 1 7
2.5 แสดงเวลาการเวลาการ เปิด-ปิด ของสวิทช์ในเซกเตอร์ที่ 1 8
2.6 ความสัมพันธ์ระหว่างแกน U, V, W กับแกนอ้างอิง d-q ในเซกเตอร์ที่ 2 9
2. 7 แสดงการคำนวณเวกเตอร์เพื่อหาค่าคาบเวลาการ เปิด-ปิด ของสวิทช์ในเซกเตอร์ที่ 2 9
2.8 แสดงเวลาการเวลาการ เปิด-ปิด ของสวิทช์ในเซกเตอร์ที่2 10
3.1 สัญญาณ PWM 11
3.2 การทำงานของเวกเตอร์ในเซกเตอร์ที่ 1 13
3.3 ช่วงเวลาการเปิด-ปิด สวิทช์ของเฟส U และเฟส U+V ในเซกเตอร์ที่ 1 13
3.4 แสดงรูปสัญญาณ Sampling คงที่ กับการเปลี่ยนแปลงค่าความถี่สัญญาณอ้างอิง 15
3.5 แสดง Flow Chart ของโปรแกรมหลัก 16
3.6 แสดง Flow Chart การกำหนดค่าเริ่มต้นให้ MCU 17
3.7 แสดง Flow Chart ของการคำนวณค่าความถี่ของแต่ละเซกเตอร์ 18
3.8 แสดงความสัมพันธ์ของค่า PDC เพื่อนำไปหาขนาด duty cycle ในโมดูล PWM 20
3.9 แสดง Flow Chart การทำงานของ Interrupt Timer1 20
3.10 แสดงไดอะแกรมของสเปซเวกเตอร์ 21
3.11 แสดงคาบเวลา Ta, Tb และ T0 ในเซกเตอร์ที่ 1 21
3.12 ระบบอินเวอร์เตอร์สำหรับขับมอเตอร์ไฟฟ้าเหนี่ยวนำ 23
3.13 วงจรวงจรเรตติไฟล์เออร์ 23
3.14 วงจรลดกระแสกระชากขณะสตาร์ท 24
3.15 วงจรอินเวอร์เตอร์ 24

สารบัญรูปภาพ (ต่อ)

รูปที่ หน้า

4.1 สัญญาณ PWM จากไมโครคอนโทรลเลอร์ ทั้ง 6 ขา 25
4.2 สัญญาณ PWM จากไมโครคอนโทรลเลอร์ ทั้ง 6 ขา 26
4.3 รูปแบบการวัดสัญญาณ PWM แบบอนาล็อก 27
4.4 สัญญาณ PWM หลังผ่านวงจรกรองความถี่ต่ำผ่าน 27
4.5 แสดงการวัดแรงดันระหว่างสาย 28
4.6 สัญญาณแรงดันระหว่างสาย 28
4.7 แสดงการวัดแรงดันเฟส 29
4.8 สัญญาณการวัดแรงดันเฟส U 29
4.9 สัญญาณการวัดแรงดันเฟส V 30
4.10 สัญญาณการวัดแรงดันเฟส W 30

สารบัญตาราง

ตารางที่	หน้า
2.1 แสดงค่าแรงดันระหว่างสาย และค่าแรงดันระหว่างเฟส	5

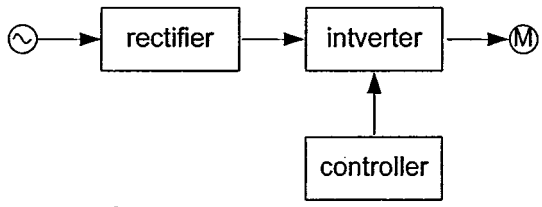
บทที่ 1
บทนำ

1.1 ความเป็นมาของการทำปริญญานิพนธ์หัวข้อนี้

การควบคุมการขับเคลื่อนมอเตอร์เหนี่ยวนำในงานอุตสาหกรรม โดยใช้อินเวอร์เตอร์เป็นเครื่องควบคุมความเร็ว อินเวอร์เตอร์นั้นถูกนำไปใช้ในงานอุตสาหกรรมหลายด้าน เช่นเครื่องบดโลหะ เครื่องม้วนโลหะ ขับเคลื่อนรถราง ขับเคลื่อนสายพาน และได้มีการนำมาประยุกต์ใช้งานกับเครื่องใช้ไฟฟ้าภายในบ้าน เช่นเครื่องปรับอากาศ และ ตู้เย็น เป็นต้น เพื่อลดการใช้พลังงานในเครื่องใช้ไฟฟ้าต่างๆ โดยมีวิธีการควบคุมแบบที่ง่ายที่สุดคือ การควบคุมแบบการเปลี่ยนอัตราส่วนแรงดันต่อความถี่คงที่ (Scalar Control : Constant V/F) ซึ่งเป็นที่นิยมใช้ในงานที่ไม่ต้องการผลตอบสนองของแรงบิดที่รวดเร็วมาก การควบคุมความเร็วมอเตอร์เหนี่ยวนำโดยวิธีนี้ใช้การปรับความถี่เพื่อเปลี่ยนค่าความเร็วซิงโครนัส (Synchronous Speed) ปัจจุบันมีการพัฒนาให้อินเวอร์เตอร์มีการใช้งานได้มีประสิทธิภาพมากยิ่งขึ้น เช่นลดฮาร์มอนิกส์ดิสทรอชั่นของกระแส (Harmonic Distortion) ลดการสูญเสียจากการสวิตชิ่งและเพิ่มแรงดันเอาต์พุตให้กับอินเวอร์เตอร์ การควบคุมอินเวอร์เตอร์อีกแบบหนึ่งคือการสร้างสัญญาณพัลซ์วิธด์มอดูเลชั่น (Pulse Width Modulation: PWM) เพื่อไปควบความเร็วรอบของมอเตอร์ ซึ่งในการวิจัยนี้จะศึกษาการปรับความกว้างพัลส์โดยวิธีการควบคุมเวกเตอร์สเปซพัลส์วิธด์มอดูเลชั่น (Space Vector PWM: SVM) เป็นตัวควบคุมอินเวอร์เตอร์ ซึ่งวิธีนี้จะทำให้ได้แรงดันเอาท์พุทกระแสสลับมีความใกล้เคียงรูปซายน์มาก และมีค่าแรงสูงสุดเท่ากับค่าแรงดันดีซีลิ้ง (DC Link)

1.2 โครงสร้างของอินเวอร์เตอร์

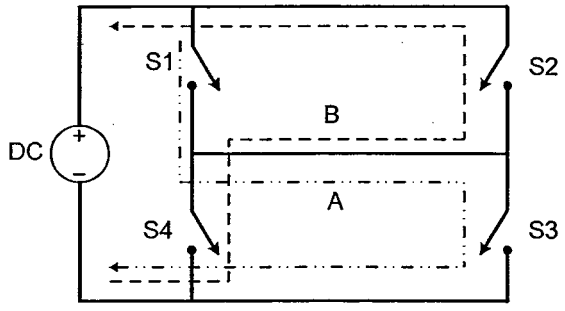
รูปที่ 1.1 แสดงโครงสร้างของอินเวอร์เตอร์ อินพุตของอินเวอร์เตอร์เป็นไฟสลับ (AC) จากแหล่งจ่ายไฟ (50 Hz) ไฟสลับนี้จะถูกแปลงเป็นไฟตรง (DC) โดยวงจรบริดจ์เร็กติไฟเออร์จากนั้นไฟกระแสตรงจะถูกแปลงเป็นไฟกระแสสลับ ที่สามารถแปรแรงดันและความถี่ได้โดยวงจรอินเวอร์เตอร์ (Inverter) และยังมีวงจรควบคุมการทำงานด้วย



รูปที่ 1.1 โครงสร้างของอินเวอร์เตอร์

1.3 หลักการทำงานของอินเวอร์เตอร์อย่างง่าย

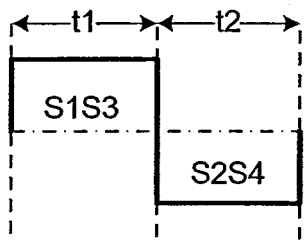
วิธีสร้างไฟสลับจากไฟตรง



รูปที่ 1.2 วิธีสร้างไฟสลับอย่างง่าย

พิจารณารูปที่ 1.2 วงจรสร้างไฟสลับเฟสเดียว (1-phase Full Bridge Inverter) อธิบายหลักการเปลี่ยนไฟตรงเป็นไฟสลับดังนี้ สวิตซ์ 4 ตัว S1 S2 S3 S4 ต่ออยู่ระหว่างแหล่งจ่ายไฟตรงและโหลด เมื่อสวิตซ์ S1 และ S3 ปิด จะมีกระแสวิ่งผ่านสวิทช์และโหลดตามทิศทาง A เมื่อสวิตซ์ S2 และ S4 ปิด จะมีกระแสวิ่งผ่านสวิทช์และโหลดตามทิศทาง B และตรงข้ามกับ A ผลที่ได้คือไฟสลับ

ถ้าควบคุมเวลาที่ปิดเปิดสวิตซ์ S1-S4 ได้ ก็สามารถควบคุมความถี่ของไฟสลับได้ดังรูป 1.3



รูปที่ 1.3 คลื่นไฟสลับ

ความถี่ที่ได้สามารถคำนวณได้ดังนี้

$$\text{ความถี่ } f = \frac{1}{t1+t2}$$

1.4 วัตถุประสงค์

เพื่อศึกษา ค้นคว้า ออกแบบ การสร้างสัญญาณควบคุมพัลส์โดยวิธีการเวกเตอร์สเปช เพื่อที่สามารถนำไปควบคุมอินเวอร์เตอร์ได้

1.5 ขอบเขตของงานวิจัย

ศึกษาวิธีการสร้างสัญญาณควบคุมพัลส์โดยวิธีการเวกเตอร์สเปช เพื่อไปควบคุมอินเวอร์เตอร์ และศึกษาการควบคุมผ่านไมโครคอนโทรลเลอร์ โดยใช้ตัวประมวลผลสัญญาณดิจิตอล (Digital Signal Processing: DSP) ภายในตัวไมโครคอนโทรลเลอร์ เบอร์ dsPIC30F4011

1.6 ขั้นตอนและวิธีดำเนินการ

1.6.1 ค้นคว้าทฤษฎี การควบคุมอินเวอร์เตอร์โดยหลักการของสเปซเวกเตอร์พัลส์วิดธ์มอดูเลชั่น

1.6.2 จำลองการทำงานของเวกเตอร์สเปช โดยโปรแกรม MATLAB เพื่อศึกษาขนาดและเวลาของเวกเตอร์

1.6.3 ศึกษาการเขียนโปรแกรมของตัวไมโครคอนโทรลเลอร์ เบอร์ dsPIC30F4011

1.6.4 นำหลักการและทฤษฎีของเวกเตอร์แรงดันที่ได้จากการจำลองการทำงานมาปรับใช้กับตัวประมวลผลสัญญาณดิจิตอลภายในไมโครคอนโทรลเลอร์

1.6.5 สร้างภาคควบคุมของอินเวอร์เตอร์และทดสอบการทำงาน

1.6.6 สร้างส่วนของภาคกำลังของอินเวอร์เตอร์

1.6.7 ทดสอบระบบอินเวอร์เตอร์รวม

1.6.8 รวบรวมข้อมูล ประเมินผล และสรุปผล

1.6.9 เขียนและพิมพ์ปริญญานิพนธ์

1.7 ประโยชน์ที่คาดว่าจะได้รับ

1.7.1 ทำให้รู้ถึงหลักการ การทำงานของอินเวอร์เตอร์

1.7.2 ทำให้รู้ถึงหลักการ การสร้างสัญญาณพัลส์โดยวิธีเวกเตอร์สเปช

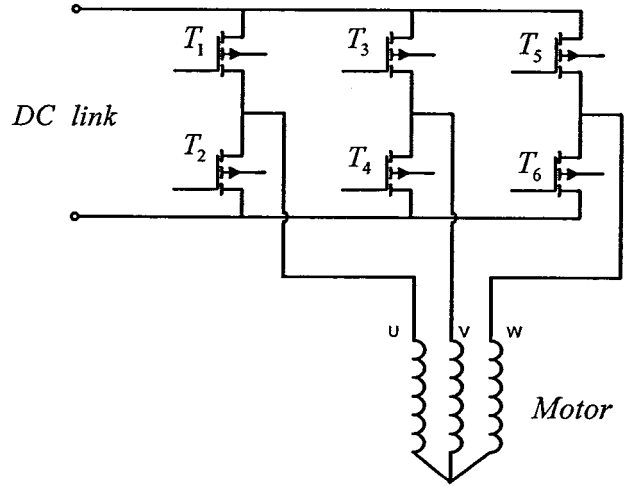
1.7.3 ทำให้รู้ถึงหลักการเขียนโปรแกรมภาษาซี ลงบนตัวไมโครคอนโทรลเลอร์

บทที่ 2
หลักการสัญญาณพีดับบลิวเอ็มด้วยวิธีสเปซเวกเตอร์แรงดัน และทฤษฎีที่เกี่ยวข้อง

ในบทนี้จะกล่าวถึงวิธีการมอดดูเลตความกว้างพัลส์ (PWM) โดยวิธีสเปซเวกเตอร์มีคุณสมบัติที่ดีคือ ลดฮาร์โมนิกส์ และกำลังไฟฟ้าสูญเสียเนื่องจากการสวิทช์ของอุปกรณ์ ในวงจรอินเวอร์เตอร์ได้ดีกว่าการมอดดูเลตความกว้างแบบอื่นรวมทั้งกล่าวถึงการนำหลักการสเปซเวกเตอร์ไปประยุกต์ใช้กับฟังชั่นการมอดดูเลตแบบต่อเนื่องและกล่าวถึงผลงานที่ถูกนำเสนอไว้ในส่วนท้าย

2.1 หลักการสร้างสัญญาณ PWM โดยวิธีสเปซเวกเตอร์

อินเวอร์เตอร์สามเฟสมีอุปกรณ์สวิทช์อิเล็กทรอนิกส์อยู่ทั้งหมด 6 ตัว ใน 3 กิ่ง ดังแสดงในรูปที่ 2.1 โดยในแต่ละกิ่งจะให้ค่าสถานะของการสวิทช์เป็น 1 เมื่อสวิทช์ทำงานในตอนนี้จะกล่าวถึงเพียงสวิทช์ตัวบนเท่านั้น ส่วนสวิทช์ตัวล่างจะอยู่ในสถานะตรงกันข้ามกับตัวบนเสมอ



รูปที่ 2.1 อินเวอร์เตอร์สามเฟส

จากความสัมพันธ์ระหว่างค่าการเปลี่ยนแปลงเวกเตอร์ (u,v,w) และเวกเตอร์แรงดันระหว่างสาย (V_{uv}, V_{vw}, V_{wu}) สามารถเขียนเป็นสมการได้ดังนี้

$$\begin{bmatrix} V_{uv} \\ V_{vw} \\ V_{wu} \end{bmatrix} = DClink \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \qquad (2.1)$$

จากความสัมพันธ์ระหว่างค่าการเปลี่ยนแปลงเวกเตอร์ (u,v,w) และเวกเตอร์แรงดันเฟส (V_u, V_v, V_w) สามารถเขียนเป็นสมการได้ดังนี้

$$\begin{bmatrix} V_{un} \\ V_{un} \\ V_{wu} \end{bmatrix} = \frac{DC\ link}{3} \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \qquad (2.2)$$

จากรูปที่ 2.1 ในการเปิด-ปิด สวิทช์ชุดบนจะมีความเป็นไปได้ถึง 8 สภาวะ คือ [0,0,0] , [0,0,1] , [0,1,0] , [0,1,1] , [1,0,0] , [1,0,1] , [1,1,0] และ [1,1,1] ซึ่งสวิทช์ชุดล่างจะทำงานตรงข้ามกับสวิทช์ชุดบนเสมอ โดยจะมีสองสภาวะที่ไม่มีแรงดันเอาท์พุทจ่ายให้โหลด คือ [0,0,0] และ [1,1,1] ส่วนสภาวะที่เหลือจะมีแรงดันเอาท์พุทจ่ายให้โหลด จากสมการที่ (2.1) และ (2.2) เราสามารถหาค่าแรงดันระหว่างสาย และค่าแรงดันระหว่างเฟส ในรูปของแรงดัน ดี ซี ลิ้ง (DC Link : Vdc) ได้ ซึ่งแสดงให้เห็นในตารางที่ 2.1

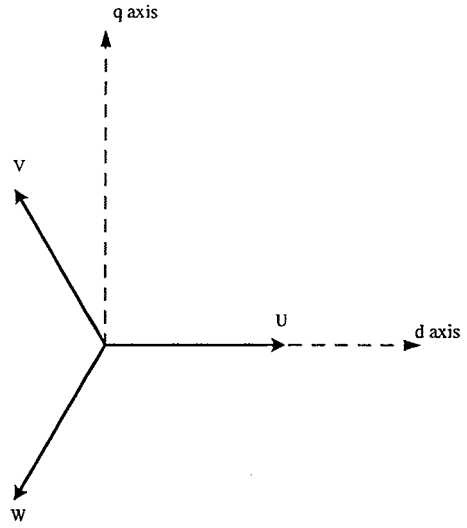
ตารางที่ 2.1 แสดงค่าแรงดันระหว่างสาย และค่าแรงดันระหว่างเฟส

Voltage	switching vector			line to neutral vector			line to line voltage		
vector	u	v	w	Vun	Vvn	Vwn	Vuv	Vvw	Vwu
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

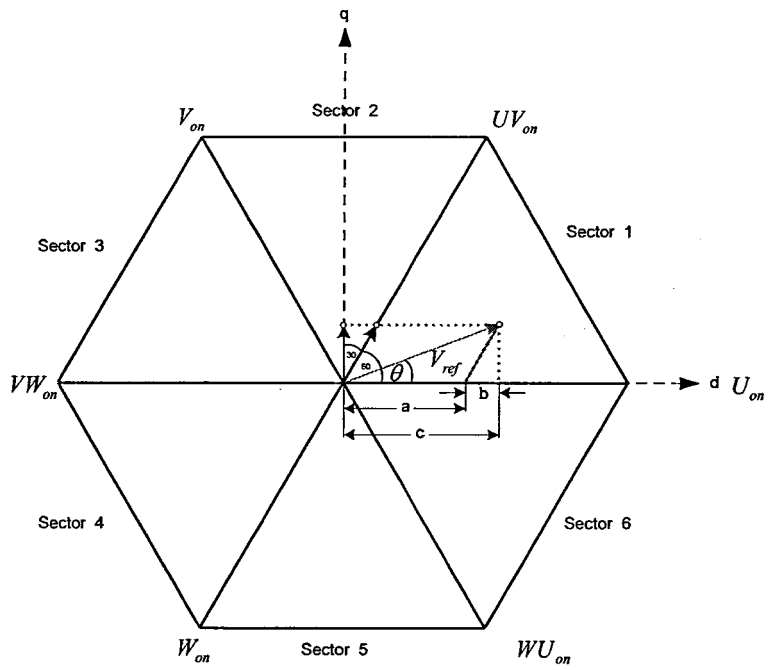
<u>หมายเหตุ</u> ค่าในตารางต้องคูณด้วยค่า Vdc

2.2 เครื่องมือสำหรับการหาค่า Space Vector PWM

เครื่องมือสำหรับการหาค่า Space Vector PWM คือการใช้แกนอ้างอิง d-q ในการหาค่า โดยแกนนอนจะเป็นแกน (d) และแกนตั้งจะเป็นแกน (q) ดังรูปที่ 2.42



รูปที่ 2.2 ความสัมพันธ์ระหว่างแกน U, V, W กับแกนอ้างอิง d-q

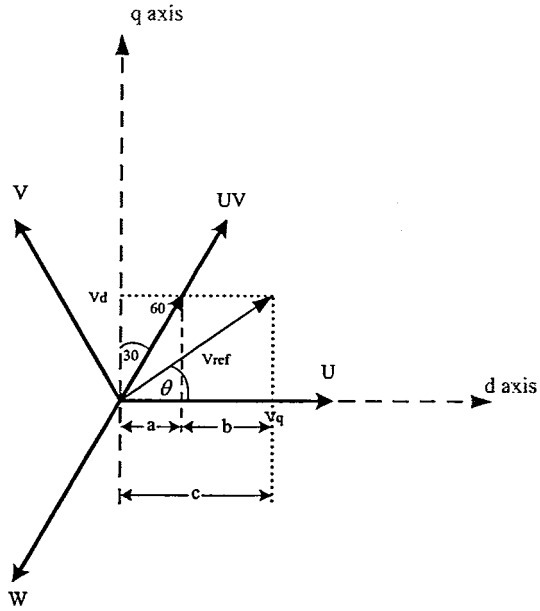


รูปที่ 2.3 ความสัมพันธ์ระหว่างแกน U, V, W กับแกนอ้างอิง d-q ในเซกเตอร์ที่ 1

จากรูปที่ 2.3 ในการแปลงเวกเตอร์ (U, V, W) ไปยังแกนอ้างอิง d-q สามารถนำไปสร้างรูปหกเหลี่ยมได้ โดยที่แต่ละแกนมีมุมห่างกัน 60 องศา ซึ่งจากรูปที่ 2.3 นั้นจะเห็นได้ว่าจะมีเวกเตอร์ที่ไม่เป็นศูนย์อยู่ที่แกนทั้งหก และ เวกเตอร์ที่เป็นศูนย์อยู่ที่จุดกำเนิด และยังมีพารามิเตอร์ที่สำคัญอีกตัวหนึ่งคือการหาค่าแรงดันอ้างอิง (V_{ref}) ซึ่งในการหาค่าแรงดันอ้างอิง นั้นเราจะใช้รูปแบบการสวิทช์ทั้ง 8 แบบ เป็นเครื่องมือในการหาค่า ซึ่งในการหาค่า Space Vector PWM จะสามารถทำได้ดังนี้

- หาค่า Vd, Vq, Vref, และ ค่ามุม (θ)
- หาค่าระยะเวลาของ t_0, t_1 และ t_2
- หาค่าเวลาที่จะใช้ในการเปิด-ปิด สวิทช์ทั้ง 6 ตัว

เซกเตอร์ที่ 1



รูปที่ 2.4 แสดงการคำนวณเวกเตอร์เพื่อหาค่าคาบเวลาการ เปิด-ปิด ของสวิทช์ในเซกเตอร์ที่ 1

จากรูปที่ 2.4 สามารถคำนวณได้ดังนี้

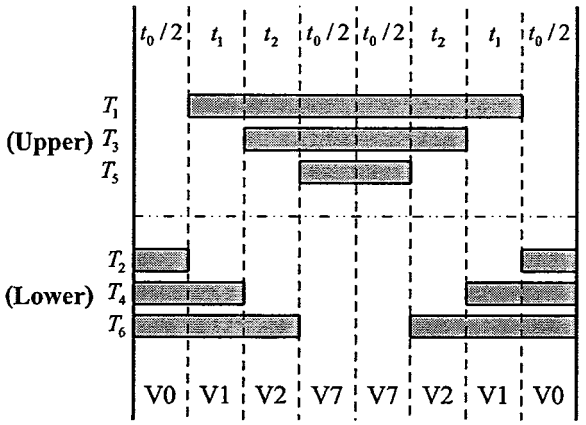
$$(UV)_{on} = \frac{V_{ref} \sin\theta}{\sin 60} \tag{2.3}$$

$$b = U_{on} = V_{ref} \cos\theta - (UV)_{on} \cos\theta \tag{2.4}$$

$$a = c - b \tag{2.5}$$

$$c = V_{ref} \cos\theta \tag{2.6}$$

$$a = (UV)_{on} \cos 60 \tag{2.7}$$



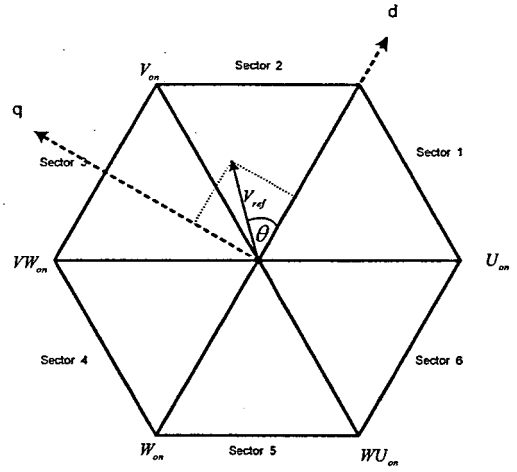
รูปที่ 2.5 แสดงเวลาการ เปิด-ปิด ของสวิทช์ในเซกเตอร์ที่ 1

ในเซกเตอร์ที่ 1 เลือกพิจารณาเฉพาะด้าน (Upper) จะมีการทำงานอยู่สองสภาวะคือ V1 (100), V2 (110) ในช่วงเวลา t_1 นั้น T_1 จะนำกระแสเพียงตัวเดียว และ ในช่วงเวลา t_2 มี T_1 , T_3 ที่จะนำกระแส

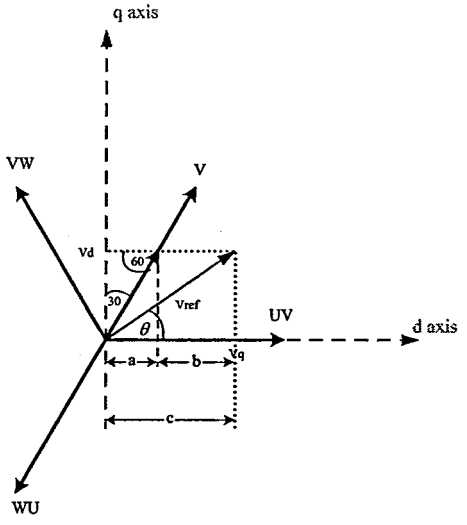
ส่วนในช่วงเวลา $t_0/2$ จะอยู่ในสภาวะ V0 (000) และV7 (111) จะไม่มีการนำกระแสใน 2 สภาวะนี้

เซกเตอร์ที่ 2

ในเซกเตอร์ที่ 2 นี้จะมีการคำนวณที่คล้ายกันกับเซกเตอร์ที่ 1 เพียงแต่เลื่อนแกนอ้างอิง d-q ไปอยู่ในเซกเตอร์ที่ 2 ดังรูปที่2.6 และสามารถหาค่าเวลาการ เปิด-ปิด สวิทช์แต่ละตัวได้ดังรูปที่2.7



รูปที่ 2.6 ความสัมพันธ์ระหว่างแกน U, V, W กับแกนอ้างอิง d-q ในเซกเตอร์ที่ 2

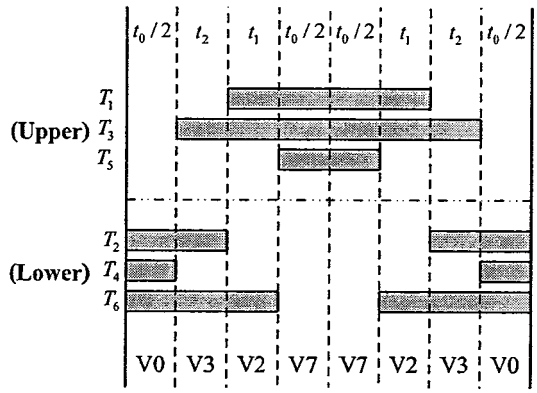


รูปที่ 2.7 แสดงการคำนวณเวกเตอร์เพื่อหาค่าคาบเวลาการ เปิด-ปิด ของสวิทช์ในเซกเตอร์ที่ 2

จากรูปที่ 2.7 สามารถคำนวณได้ดังนี้

$$(V)_{on} = \frac{V_{ref} \sin\theta}{\sin 60} \tag{2.8}$$

$$b = UV_{on} = V_{ref} \cos\theta - (UV)_{on} \cos\theta \tag{2.9}$$



รูปที่ 2.8 แสดงเวลาการเวลาการ เปิด-ปิด ของสวิทช์ในเซกเตอร์ที่2

ในเซกเตอร์ที่ 2 เลือกพิจารณาเฉพาะด้าน (Upper) จะมีการทำงานอยู่สองสภาวะคือ V2 (110), V3 (010) ในช่วงเวลา t_1 จะนำกระแสสองตัวคือ T_1 , T_3 และ ในช่วงเวลา t_2 จะมี T_3 นำกระแสเพียงตัวเดียว ส่วนในช่วงเวลา $t_0/2$ จะอยู่ในสภาวะ V0 (000) และV7 (111) ในเซกเตอร์อื่นๆ ก็ใช้หลักการเดียวกันในการคำนวณหาค่าการเวลา เปิด-ปิด ของสวิทช์ดังกล่าว

จากสมการทฤษฎีดังกล่าวสามารถสรุปได้ว่าตัวแปรที่สำคัญของ สเปซเวกเตอร์คือ มุม θ และขนาดของ V_{ref} ซึ่งในแต่ละเซกเตอร์นั้นจะให้ค่าการ เปิด-ปิด ของสวิทช์ที่แต่ละตัวที่ต่างกัน

บทที่ 3

วิธีการออกแบบ SPACE VECTOR PWM

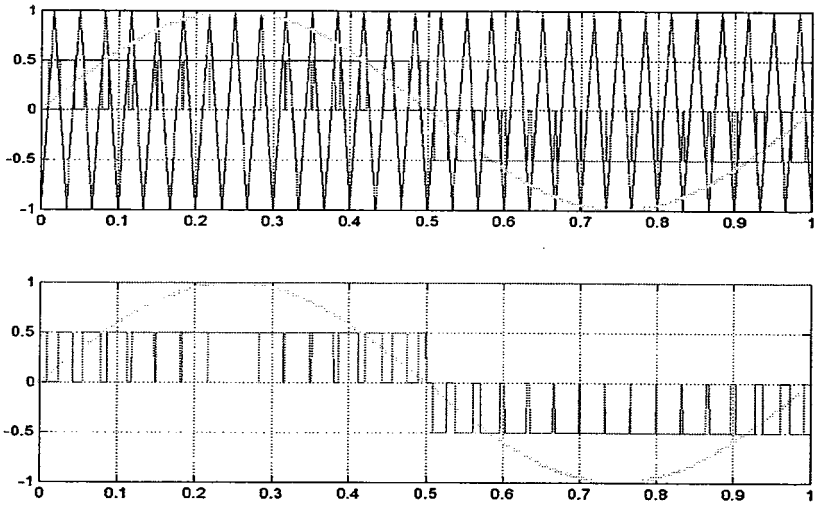
3.1 ศึกษากระบวนการทำงานพื้นฐานของ PWM ด้วย MATLAB

MATLAB เป็นโปรแกรมที่เหมาะสมกับการพัฒนาแบบจำลอง และการจำลองแบบ (simulation) กระบวนการทางวิศวกรรม เมื่อเปรียบเทียบกับการใช้ภาษาอื่นๆเช่น C หรือ FORTRAN จะพบว่า โปรแกรมMATLAB มี built-in function หลากหลายและเรียกใช้งานได้ง่ายกว่า ไม่ว่าจะเป็นฟังก์ชันการคำนวณ เชิงตัวเลขการแสดงผล และการติดต่อกับผู้ใช้ (user interface) โปรแกรม MATLAB สามารถเรียกโปรแกรมที่มีอยู่แล้วทั้งภาษา C และ FORTRAN มาใช้งานได้ อีกทั้งยังเป็นภาษาที่มีโครงสร้างเป็นโมดูล ทำให้สามารถนำโปรแกรมย่อยที่เคยพัฒนามาใช้ใหม่ได้

3.1.1 หลักการสร้างสัญญาณ PWM

ในการสร้างสัญญาณ PWM นั้น เราจะใช้สัญญาณ 2 สัญญาณมาเปรียบเทียบกัน โดยจะมีสัญญาณที่มาเปรียบเทียบ (สัญญาณสามเหลี่ยม) และสัญญาณอ้างอิง (สัญญาณซายน์) โดยจะเปรียบเทียบว่าถ้าสัญญาณซายน์ มากกว่า สัญญาณสามเหลี่ยม ก็จะให้สัญญาณ PWM มีค่าเป็น 1 และถ้าสัญญาณซายน์มีค่าน้อยกว่าสัญญาณสามเหลี่ยม ก็จะให้สัญญาณ PWM มีค่าเป็น 0

ในที่นี้เราได้จำลองการสร้างสัญญาณ PWM ขึ้นมาโดยใช้โปรแกรม MATLAB ซึ่งจะได้แสดงดังรูปที่ 3.1



รูปที่ 3.1 สัญญาณ PWM

ชุดคำสั่งในโปรแกรม MATLAB ที่ใช้ในการสร้างสัญญาณ PWM

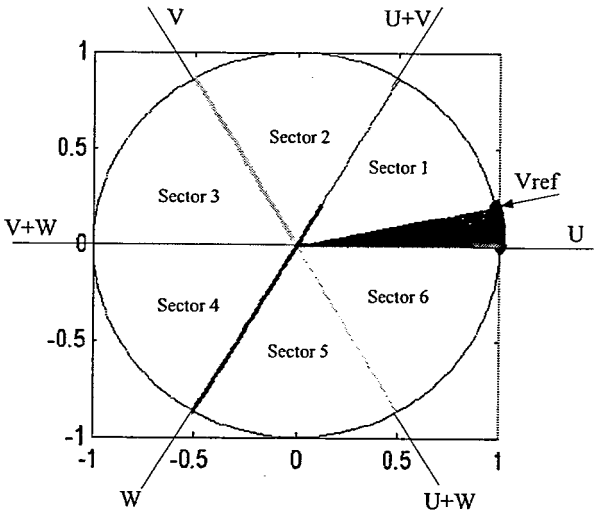
```
t = 0:0.0001:1;                                % time step

fsaw = 30;                                     % saw tooth frequency
saw_wave = sawtooth(2*pi*fsaw*t,0.5);
subplot(211);plot(t,saw_wave,'b-');hold on;grid on

A = 1;                                         % Amplitude of sine
fsin = 1;                                      % sine wave frequency
sine_wave = A*sin(2*pi*fsin*t);
subplot(211);plot(t,sine_wave,'y.-');hold on

pwm1 = zeros(1,length(saw_wave));

for d = 1:round(length(saw_wave)/2)              % positive sine
    if(sine_wave(d) > saw_wave(d))
       pwm1(d) = 0.5;
   elseif(sine_wave(d) < saw_wave(d))
       pwm1(d) = 0;

   else
   end
 end

 for d = round(length(saw_wave)/2):length(saw_wave)    % negative sine
    if(sine_wave(d) < saw_wave(d))
       pwm1(d) = -0.5;
   elseif(sine_wave(d) > saw_wave(d))
       pwm1(d) = 0;

   else
   end
 end

subplot(211);plot(t,pwm1,'r-')
subplot(212);plot(t,sine_wave,'y.-');hold on
subplot(212);plot(t,pwm1,'r-');grid on
axis([0 1 -1 1]);
```

3.1.2 การทำงานในเซกเตอร์ที่ 1

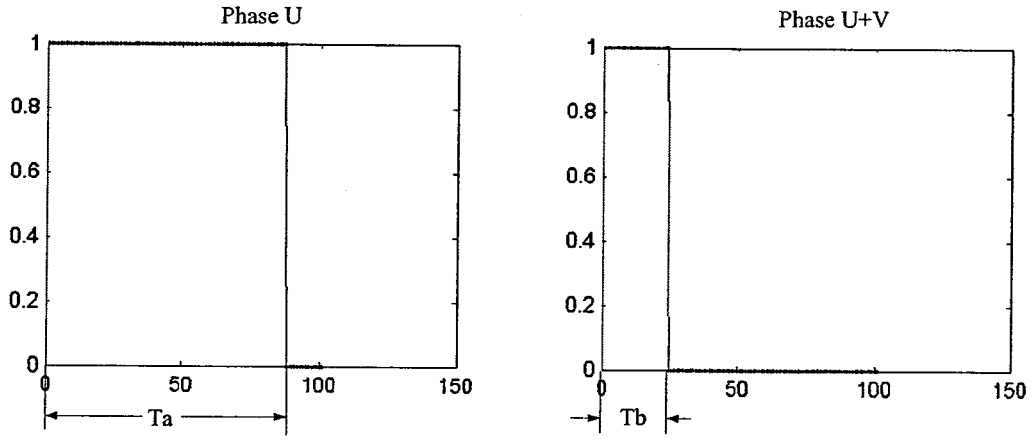
ในการศึกษาการทำงานของเวกเตอร์ในเซกเตอร์ที่ 1 นั้นสามารถดูได้จากรูปที่ 3.2 ซึ่งจะแสดงการกวาดไปของเวกเตอร์ (Vref) ที่เริ่มจาก 0 องศา จนถึง 60 องศา

และในรูปที่ 3.3 จะแสดงการช่วงเวลาการเปิด-ปิด ของสวิทช์ของเฟส U และเฟส U+V ในเซกเตอร์ที่ 1 โดยเมื่อ Vref มีค่ามุมที่เพิ่มขึ้น ค่าเวลาการเปิดสวิทช์ของเฟส U จะมีค่าค่อยๆ ลดลง และในเฟส U+V จะมีค่าค่อยๆ เพิ่มขึ้น สวนทางกับเฟส U

ส่วนในเซกเตอร์อื่น ๆ ก็จะคล้ายกับเซกเตอร์ที่ 1 แต่จะเปลี่ยนเฟสการเปิด-ปิด สวิทช์แทน



รูปที่ 3.2 การทำงานของเวกเตอร์ในเซกเตอร์ที่ 1



รูปที่ 3.3 ช่วงเวลาการเปิด-ปิด สวิทช์ของเฟส U และเฟส U+V ในเซกเตอร์ที่ 1

ชุดคำสั่งในโปรแกรม MATLAB ที่ใช้ในการสร้างสัญญาณ PWM

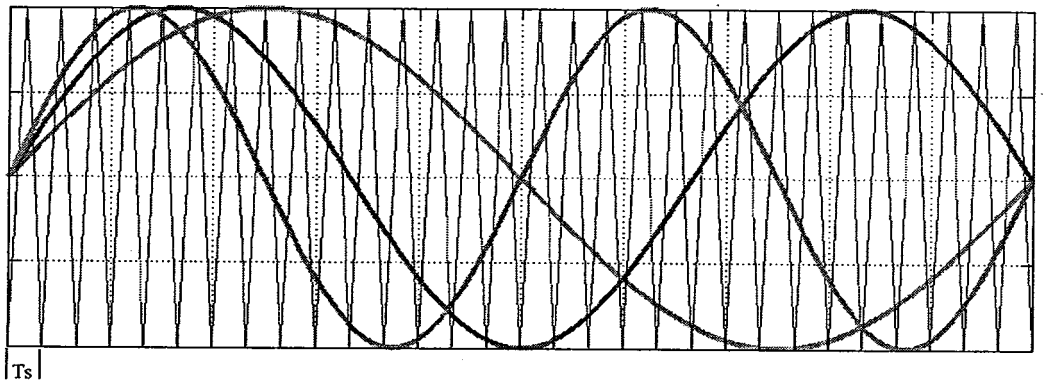
```
%referrence frame three phase 120 (2*pi/3)
mag = [0:.01:1];
ph = [exp(j*0) exp(j*2*pi/3) exp(j*4*pi/3)];
step =[0:2*pi/1000:2*pi];
cir = exp(j*step);

for k = 1:168
subplot(221);plot(real(cir),imag(cir),'k');hold on

plot(mag*real(ph(1)),mag*imag(ph(1)),'r.' )
plot(mag*real(ph(2)),mag*imag(ph(2)),'g.' )
plot(mag*real(ph(3)),mag*imag(ph(3)),'b.' )

plot(-mag*real(ph(1)),-mag*imag(ph(1)),'r' )
plot(-mag*real(ph(2)),-mag*imag(ph(2)),'g' )
plot(-mag*real(ph(3)),-mag*imag(ph(3)),'b' )


    v  = 1;
    step_v = 0:.01:v;

    a = step_v*exp(j*step(k));plot(real(a),imag(a),'k')
    plot(real(v*exp(j*step(k))),imag(v*exp(j*step(k))),'kx')
    plot(real(v*exp(j*step(k))),imag(v*exp(j*step(k))),'ko')

    d  = v*sin(step(k));            %plot(0,d,'kx')
    q1 = v*cos(step(k));            %plot(q1,0,'kx')

    u1 = v*sin(step(k))/cos(pi/6);  %plot(u1,0,'kx') % magnitude of phase w
    q2 = u1*cos(pi/3);              %plot(q2,0,'kx')
    u0 = q1 - q2;                   %plot(u0,0,'kx') % magnitude of phase u


m_u1 = find(mag<=u1);
for n = 1:length(m_u1)
     plot(-mag(n)*real(ph(3)),-mag(n)*imag(ph(3)),'b.' )
end

m_u0 = find(mag<=u0);
for n = 1:length(m_u0)
    plot(mag(n),0,'k.');hold on
end
%----- pwm on off signal ------
pwmu0 = [ones(1,length(m_u0)) zeros(1,(length(mag)-length(m_u0)))];
pwmu1 = [ones(1,length(m_u1)) zeros(1,(length(mag)-length(m_u1)))];
subplot(222);plot(pwmu0,'.-');xlabel('phase U')
subplot(224);plot(pwmu1,'.-');xlabel('phase U + phase V')
    pause;
    hold off;
end
```

3.2 การออกแบบชุดคำสั่ง (Software) บนไมโครคอนโทรลเลอร์

ในปริญญานิพนธ์ฉบับนี้จะใช้ไมโครคอนโทรลเลอร์เบอร์ dsPIC30F4011 ของบริษัท Microchip ซึ่งเป็นตัวประมวลผลสัญญาณดิจิตอลขนาด 16 บิต และยังมีโมดูลพีดับเบิ้ลยูเอ็ม (PWM Module) ที่ใช้ในการควบคุมมอเตอร์โดยเฉพาะ

3.2.1 การใช้งานฟังก์ชั่นอินเทอร์รัปไทม์เมอร์ (Timer Interrupt)

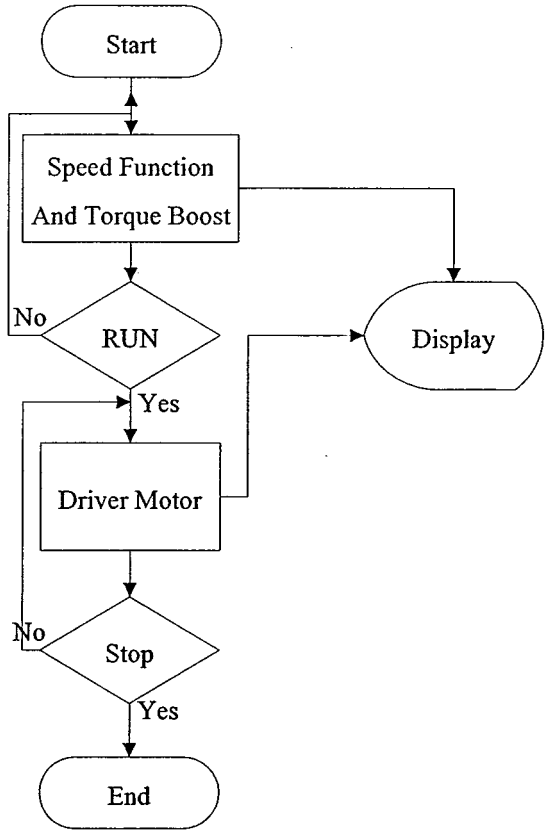
ในตัวไมโครคอนโทรลเลอร์นี้จะมีโมดูลไทม์เมอร์ (Timer module) อยู่ภายในถึง 5 โมดูล ซึ่งเรามาสารถใช้งานได้ทั้ง 16 บิต และ 32 บิต แต่ในปริญญานิพนธ์ฉบับนี้จะใช้ 16 บิต โดยจะกำหนดให้ทำการ Interrupt ทุกๆ 66 ไมโครวินาที เพื่อให้ได้ความถี่ในการสวิทช์ซึ่งเท่ากับ 15 KHz ซึ่งความถี่นี้จะเป็นความถี่คงที่ (Sampling signal) แต่สิ่งที่เปลี่ยนแปลงคือ ขนาด และ ความถี่ ของสัญญาณอ้างอิง (Sine signal)ที่นำมามอดูเลต (modulate) ดังแสดงในรูปที่ 3.4



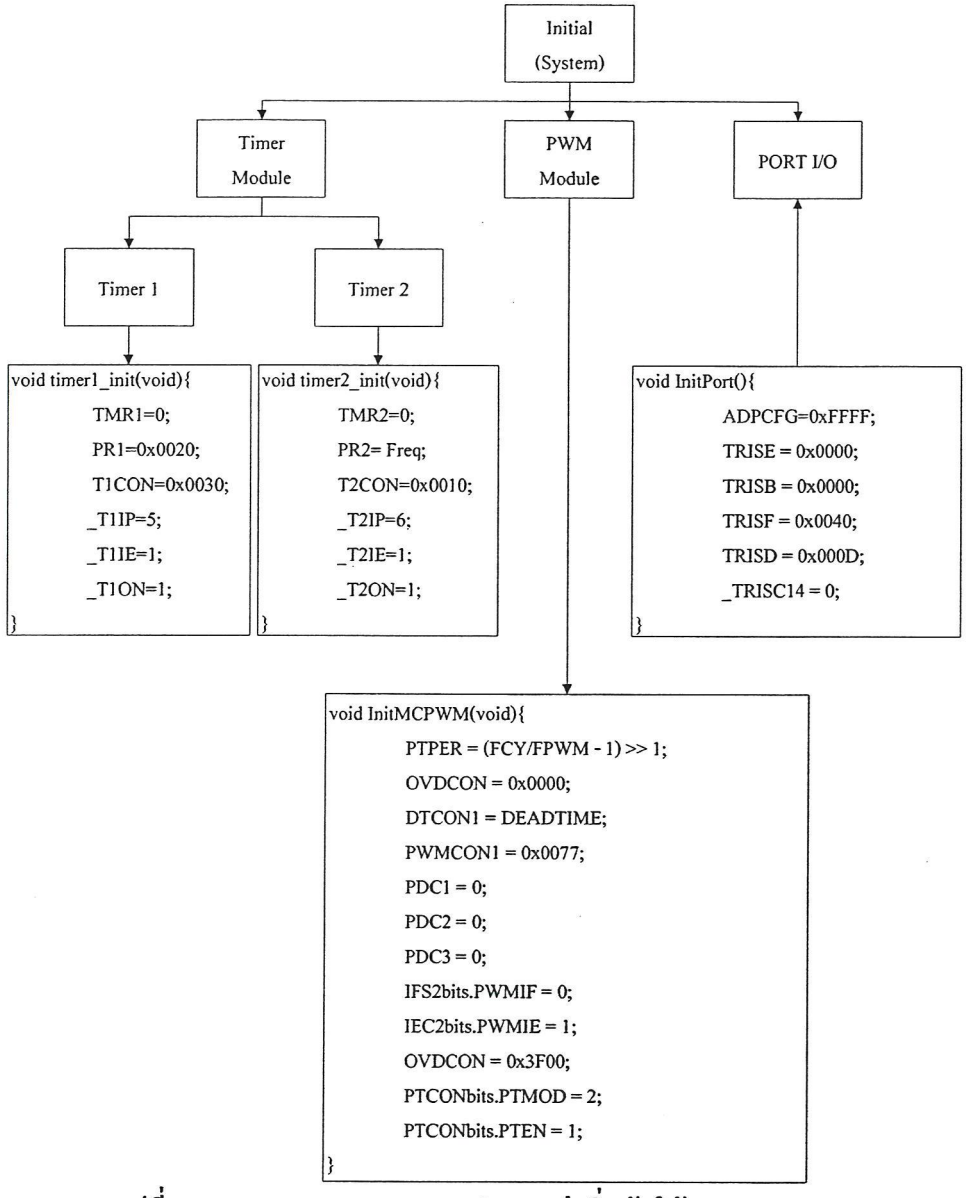
รูปที่ 3.4 แสดงรูปสัญญาณ Sampling คงที่ กับการเปลี่ยนแปลงค่าความถี่สัญญาณอ้างอิง

เราจะต้องสร้าง ความถี่ ของ สัญญาณ sine ให้เข้าจังหวะ (synchronize) กับจังหวะการ interrupt

3.2.2 แผนผังลำดับการทำงานของไมโครคอนโทรลเลอร์ (Flow chart)



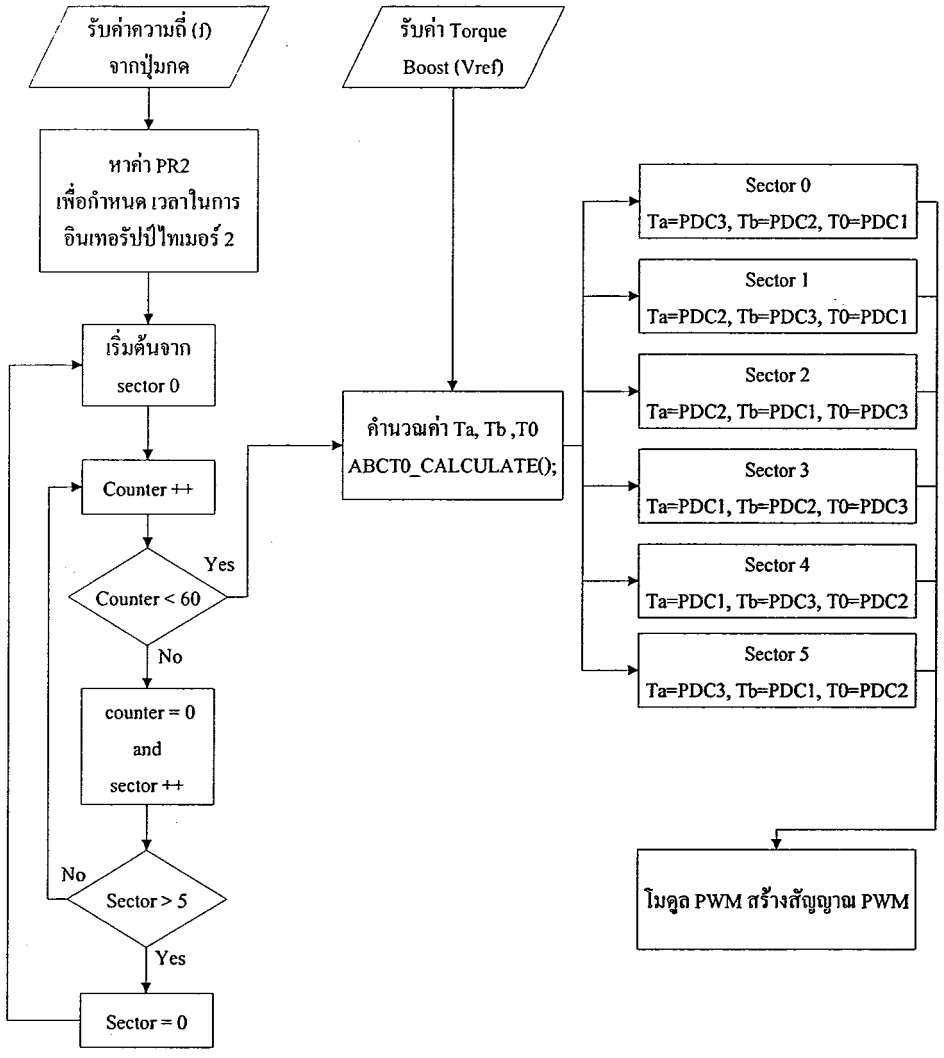
รูปที่ 3.5 แสดง Flow Chart ของโปรแกรมหลัก



รูปที่ 3.6 แสดง Flow Chart การกำหนดค่าเริ่มต้นให้ MCU

จากรูปที่ 3.6 เราจะทำการกำหนดค่าเริ่มต้นให้กับตัวไมโครคอนโทรลเลอร์ โดยเริ่มจากการกำหนดค่าให้โมดูลไทเมอร์1 ซึ่งจะมีการอินเทอร์รัทุกๆ 555.55 ไมโครวินาที และในโมดูลไทเมอร์2 จะมีการอินเทอร์รัปเปลี่ยนแปลงตามค่าของ PR2 ซึ่งจะเป็นตัวกำหนดความถี่ซายน์

ส่วนในการกำหนดค่าเริ่มต้นในโมดูล PWM นั้นจะกำหนดค่า deadtime ไว้ที่ประมาณ 3 ไมโครวินาที



รูปที่ 3.7 แสดง Flow Chart ของการคำนวณค่าความถี่ของแต่ละเซกเตอร์

เริ่มต้นจากการรับค่าความถี่ที่จากปุ่มกด จากนั้นก็จะคำนวณหาค่า PR2 เพื่อใช้ในการกำหนดค่าเวลาในการอินเทอร์รัปไทเมอร์2 และเมื่อ Timer2 ทำการอินเทอร์รัป ก็จะเพิ่มค่า counter ขึ้น 1 แล้วนำค่าไปคำนวณค่าเวลา Ta, Tb และ T0 โดยเริ่มต้นโปรแกรมจะอยู่ที่เซกเตอร์1 และนำค่า Ta, Tb, และ T0 ไปใส่ค่าในตัวแปร PDC เพื่อไปกำหนดค่าความพัลส์ต่อไป หลังจากนั้นก็จะไปเพิ่มค่า counter ทีละ 1 โดยในแต่ละครั้งที่เพิ่มค่า counter นั้นก็จะนำค่าไปคำนวณเพื่อหาค่าความพัลส์ทุกครั้ง และเมื่อค่า counter มีค่ามากกว่า 60 แล้วจะทำการรีเซตค่า counter เป็น 0 และจะเพิ่มค่า Sector ขึ้นอีก 1 และเมื่อค่า sector มีค่ามากกว่า 5 ก็จะทำการรีเซตค่า Sector เป็น 0

จากรูปที่ 3.7 จะเป็นการแสดงการคำนวณค่าที่จะใช้ในการกำหนดการอินเทอร์รัปให้โมดูลไทเมอร์2 เพื่อนำไปเปลี่ยนค่าความถี่สัญญาณซายน์ โดยเริ่มจากการรับค่าความถี่เข้ามาจากสวิทช์ปุ่มกด ก็จะมาคำนวณคาบเวลา

$$T = \frac{1}{F} \tag{3.1}$$

และนำไปหาค่า dt และเนื่องจากค่ามุมที่ใช้ในการอัพเดทนั้นเพิ่มขึ้นครั้งละ 1 องศา ดังนั้นจะต้องนำค่าคาบเวลา หารด้วย 360

$$dt = \frac{T}{360} \tag{3.2}$$

หลังจากนั้นก็จะคำนวณค่า PR2 เพื่อใช้เป็นตัวกำหนดค่าความถี่ของซายน์ จากความถี่ Fcy=14745600 Hz และในโมดูลไทเมอร์2 ได้กำหนดค่า prescale ไว้ที่ 1:8 ดังนั้นค่าความถี่จะต้องถูกหารด้วย 8 แล้วจึงนำไปคูณกับค่า dt

$$PR2 = \frac{Fcy}{8} \times dt \tag{3.3}$$

หลังจากที่กำหนดค่าให้ไทเมอร์2 อินเทอร์รัปแล้ว ต่อมาก็จะคำนวณค่า คาบเวลาในการสวิทช์ ของแต่ละเฟส โดยในขั้นตอนแรกเริ่มจากเซกเตอร์0 ก็จะทำการทำซ้ำไปจนถึง 60 ครั้ง ครั้งละ 1 องศา และจะเปลี่ยนมาเป็นเซกเตอร์1 และในเซกเตอร์1 ก็จะทำซ้ำไป 60 ครั้งเหมือนกันทุกเซกเตอร์

และในการทำซ้ำแต่ละครั้งนั้นก็จะทำการหาค่าคาบเวลา Ta และ Tb ไปด้วย จากสมการ

$$Tb = \frac{Vref \sin(\theta)}{\sin 60} \tag{3.4}$$

$$Ta = Vref \cos(\theta) - Tb \cos 60 \tag{3.5}$$

โดยจะกำหนดค่าคงที่ Ka และ Kb เพื่อช่วยให้การคำนวณเร็วขึ้น

$$Kb = \frac{Vref}{\sin 60 \times 10000} \tag{3.6}$$

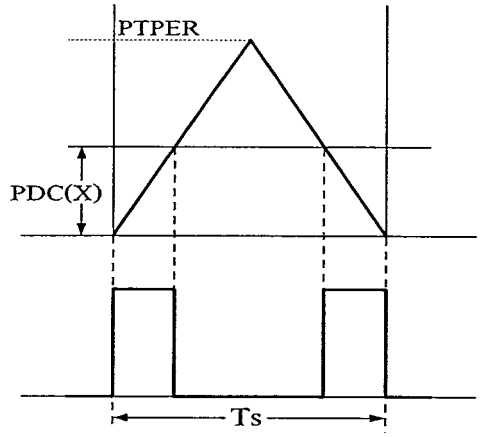
เนื่องจากค่า $\sin(\theta)$ นั้น จะสร้างเป็นตารางซายน์ 0 -60 ขึ้นมาและค่าในตารางจะถูกคูณด้วย 10000 เพื่อทำให้เป็นจำนวนเต็ม ดังนั้นในการนำค่าจากตารางซายน์มาใช้จะต้องการออกด้วย 10000

$$Ka = \frac{Tb * \cos 60}{Vref} \tag{3.7}$$

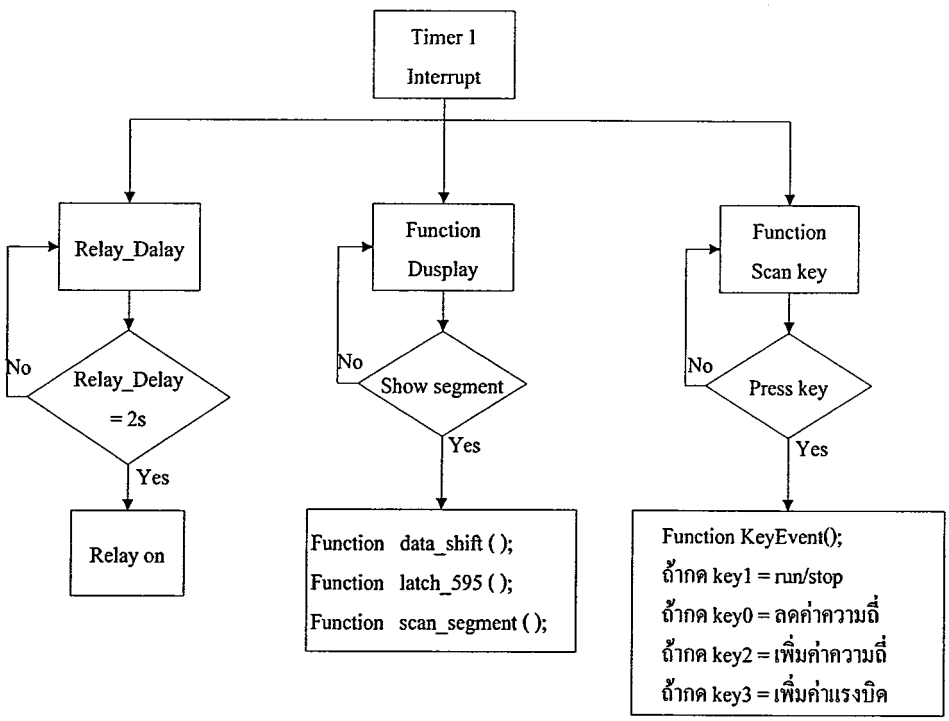
ดังนั้นค่า Ta และ Tb ใหม่จะได้เป็นดังนี้

$$Tb = Kb \times \sin(\theta) \tag{3.8}$$

$$Ta = Vref \times (\cos(\theta) - Ka) \tag{3.9}$$

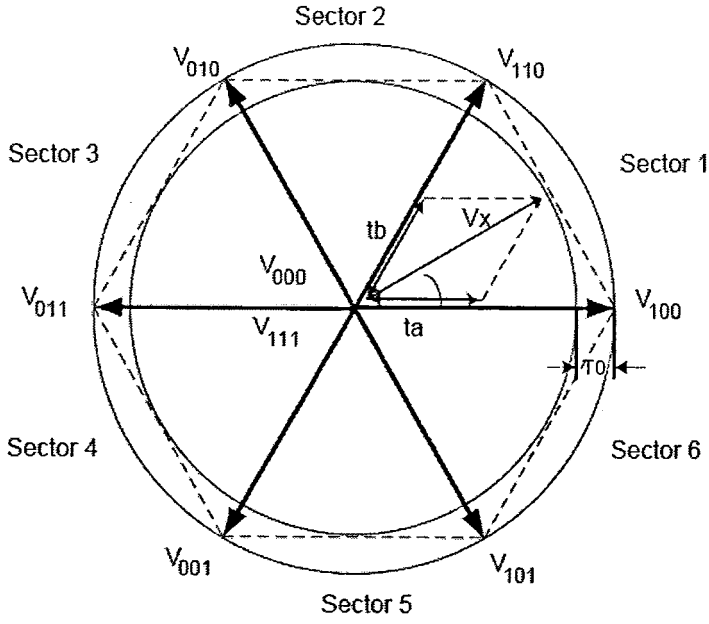


รูปที่ 3.8 แสดงความสัมพันธ์ของค่า PDC เพื่อนำไปหาขนาด duty cycle ในโมดูล PWM

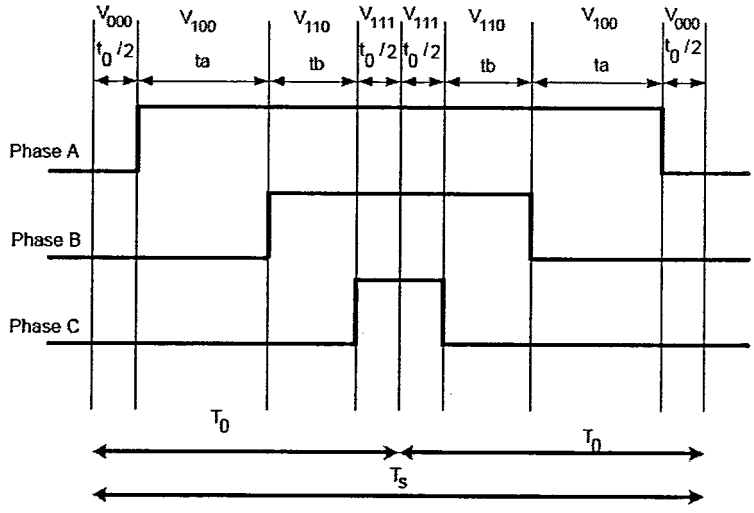


รูปที่ 3.9 แสดง Flow Chart การทำงานของ Interrupt Timer1

ในส่วนของการอินเทอร์รัปไทเมอร์1 นั้นจะเป็นใช้เป็นฐานเวลาในการจัดการระบบรวมของไมโครคอนโทรลเลอร์ โดยเริ่มแรกจะเป็นการหน่วงเวลาประมาณ 2 วินาที เพื่อให้รีเลย์ทำงาน และจะเป็นตัวควบคุมการแสดงค่าตัวเลขผ่านทาง 7-Segment และควบคุมการรับค่าของปุ่มกด



รูปที่ 3.10 แสดงไดอะแกรมของสเปซเวกเตอร์



รูปที่ 3.11 แสดงคาบเวลา Ta, Tb และ T0 ในเซกเตอร์ที่ 1

<u>ตัวอย่างการคำนวณ 1</u>

ถ้าต้องการสร้างสัญญาณซายน์ที่มีความถี่ 10 เฮิร์ท และมีความถี่ในการ แซมปลิ้ง เท่ากับ 15 กิโลเฮิร์ท

จากสมการที่ 3.1 จะได้

$$\text{T(update)} = \frac{1}{F} = \frac{1}{10} = \text{0.1 วินาที}$$

จากสมการที่ 3.2 จะได้

$$\text{dt} = \frac{T}{360} = \text{0.27777 มิลลิวนาที}$$

จากสมการที่ 3.3 จะได้

$$\text{PR2} = \frac{Fcy}{8} \times dt = 512$$

<u>ตัวอย่างการคำนวณ 2</u>

ถ้าต้องการสร้างสัญญาณซายน์ที่มีความถี่ 60 เฮิร์ท และมีความถี่ในการ แซมปลิ้ง เท่ากับ 15 กิโลเฮิร์ท

จากสมการที่ 3.1 จะได้

$$\text{T(update)} = \frac{1}{F} = \frac{1}{60} = \text{16.667 มิลลิวินาที}$$

จากสมการที่ 3.2 จะได้

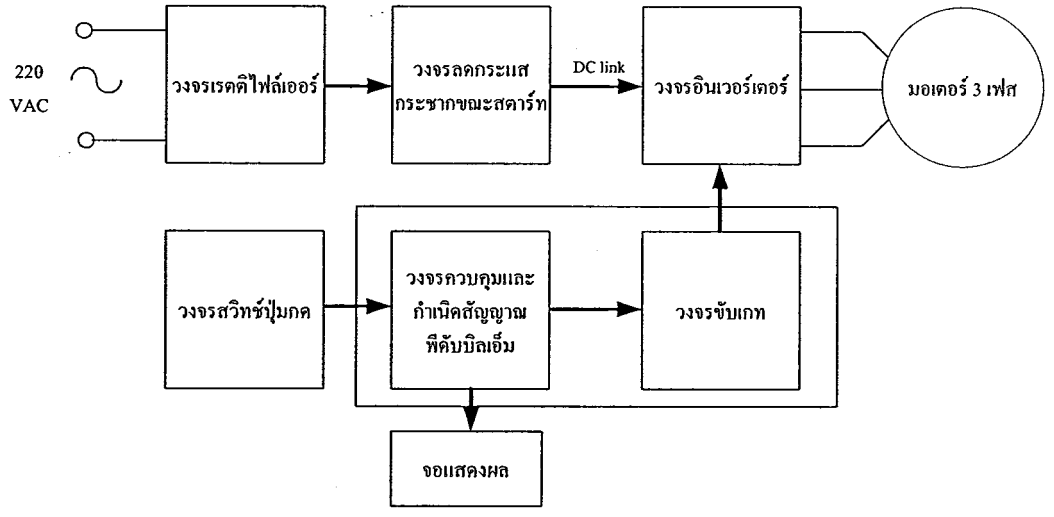
$$\text{dt} = \frac{T}{360} = \text{46.296 ไมโครวินาที}$$

จากสมการที่ 3.3 จะได้

$$\text{PR2} = \frac{Fcy}{8} \times dt = 85$$

3.3 การออกแบบชุดวงจร (Hardware)

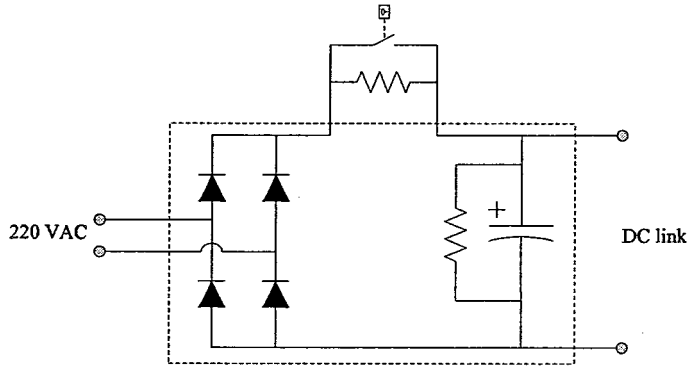
ชุดวงจรทั้งหมดของอินเวอร์เตอร์ที่ใช้ในปริญญานิพนธ์ฉบับนี้แสดงดังรูปที่ 3.12



รูปที่ 3.12 ระบบอินเวอร์เตอร์สำหรับขับมอเตอร์ไฟฟ้าเหนี่ยวนำ

3.3.1 วงจรเรตติไฟล์เออร์

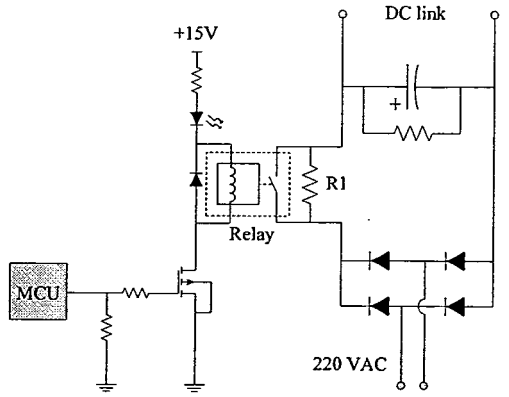
วงจรเรตติไฟล์เออร์นั้นประกอบไปด้วยไดโอด และ คาปาซิเตอร์ ดังรูปที่ 3.13 ซึ่งตัวไดโอดนั้นจะเป็นวงจรรวมอยู่ใน IPM (Intelligent Power Module) เบอร์ PS11034 ของบริษัท Mitsubishi ซึ่งสามารถทนแรงดันไบอัสย้อนกลับได้สูงสุด 800 V และสามารถทนกระแสไบอัสตรงได้สูงสุด 15 A



รูปที่ 3.13 วงจรเรตติไฟล์เออร์

3.3.2 วงจรลดกระแสกระชากขณะสตาร์ท

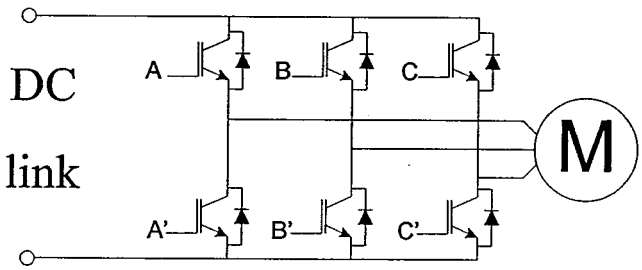
เมื่อเริ่มจ่ายแรงดันให้อินเวอร์เตอร์ คาปาซิเตอร์ในวงจรเชื่อมโยงกระแสตรง (DC link) จะทำการอัดประจุไฟฟ้า ทำให้เกิดกระแสกระชากที่มีค่าสูง จึงต้องต่อตัวต้านทาน R1 อนุกรมไว้ และเพื่อไม่ให้เกิดความสูญเสียที่ตัวต้านทาน R1 นั้น จึงต้องต่อรีเลย์ไว้ด้วย ซึ่งเมื่อเริ่มจ่ายแรงดันให้อินเวอร์เตอร์เป็นเวลาประมาณ 2 วินาที ไมโครคอนโทรลเลอร์ก็จะสั่งให้รีเลย์ทำงาน ซึ่งจะทำให้กระแสไม่ไหลผ่านความต้านทาน R1 แต่จะไหลผ่านรีเลย์แทน ดังรูปที่ 3.14



รูปที่ 3.14 วงจรลดกระแสกระชากขณะสตาร์ท

3.3.3 วงจรอินเวอร์เตอร์

วงจรอินเวอร์เตอร์ประกอบด้วยไอจีบีที (IGBT) จำนวน 6 ตัว ดังรูปที่ 3.15 โดยแต่ละตัวจะมีไดโอดต่อคร่อมขั้ว C-E อยู่เพื่อเป็นตัวคายพลังงานที่ตกค้างอยู่ในมอเตอร์ หลังจากที่มอเตอร์หยุดทำงาน ถ้าไม่มีไดโอดตัวนี้ พลังงานที่สะสมอยู่ในขดลวดในมอเตอร์ก็จะมาตกคร่อมที่ตัว IGBT ซึ่งอาจทำให้ IGBT เสียหายได้



รูปที่ 3.15 วงจรอินเวอร์เตอร์

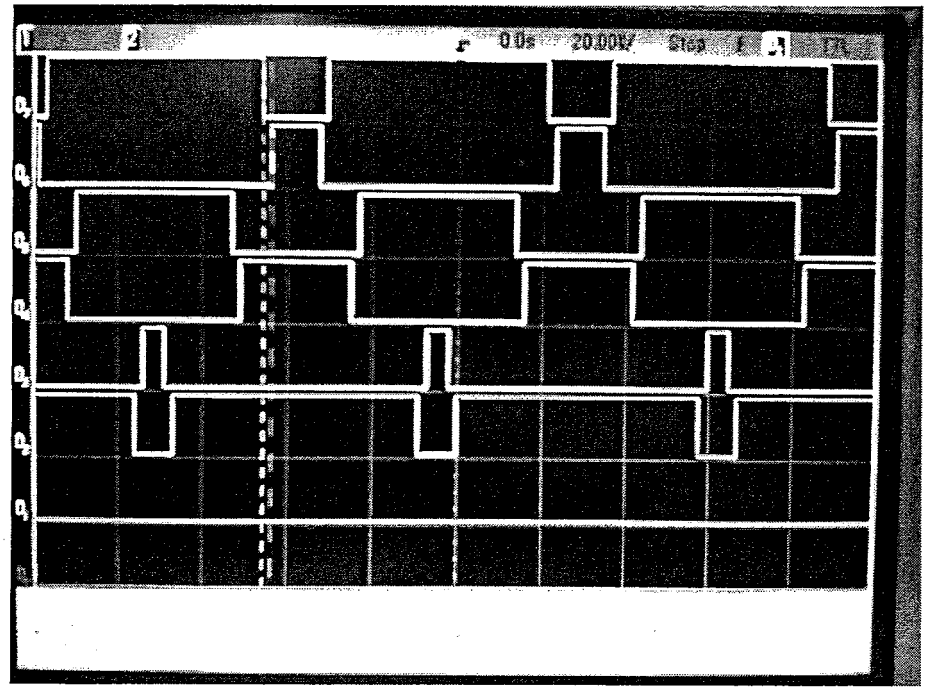
ในโมดูล IGBT นั้นสามารถทนแรงดันได้สูงสุดเท่ากับ 600 โวลต์ และสามารถกระแสได้สูงสุดเท่ากับ 15 แอมป์

บทที่ 4
ผลการทดสอบการสร้างสัญญาณ PWM

4.1 ผลการทดสอบการสร้างสัญญาณ PWM

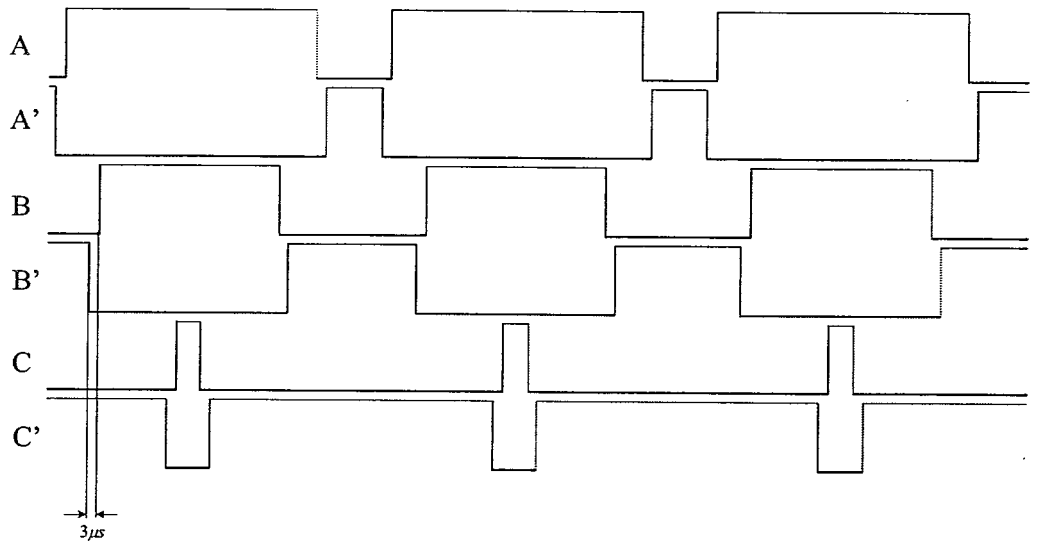
4.1.1 สัญญาณ PWM ก่อนเข้าผ่านวงจรกรองความถี่ต่ำผ่าน

การทดสอบการสร้างสัญญาณ PWM จะทำการวัดสัญญาณพัลส์สวิทช์ ที่ขาไมโครคอนโทรลเลอร์ โดยเครื่องมือที่ใช้วัด คือ ออสซิลโลสโคป ยี่ห้อ Agilent รุ่น MSO-6032A ซึ่งสามารถวัดสัญญาณอนาล็อกได้ 2 ช่อง และสามารถวัดสัญญาณดิจิตอลได้ 16 ช่อง ในการวัดสัญญาณ PWM จะเลือกการวัดแบบดิจิตอล โดยจะใช้สายโพรบแบบดิจิตอลโดยเฉพาะ ซึ่งผลการวัดสัญญาณ PWM แสดงในรูปที่ 4.1



รูปที่ 4.1 สัญญาณ PWM จากไมโครคอนโทรลเลอร์ ทั้ง 6 ขา

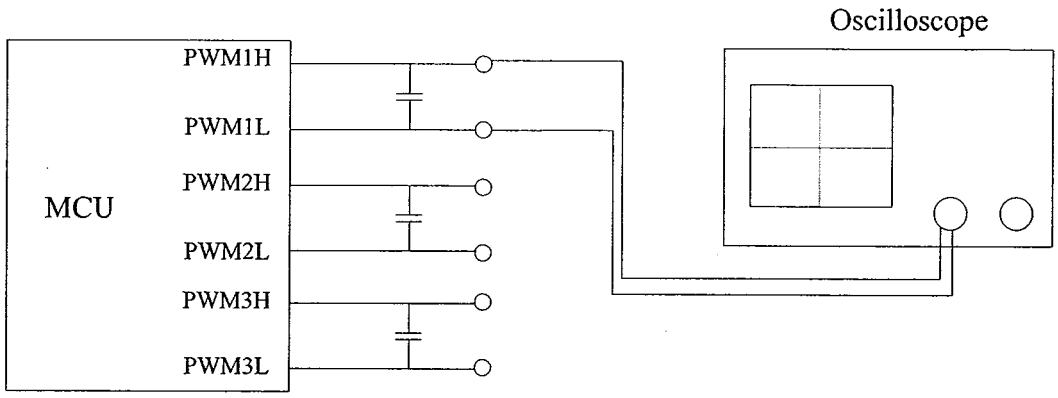
ซึ่งจากรูปสัญญาณ PWM ที่วัดได้นั้น มี Dead Time เท่ากับ 3 ไมโครวินาที ซึ่งตรงกับที่ออกแบบไว้ในบทที่ 3



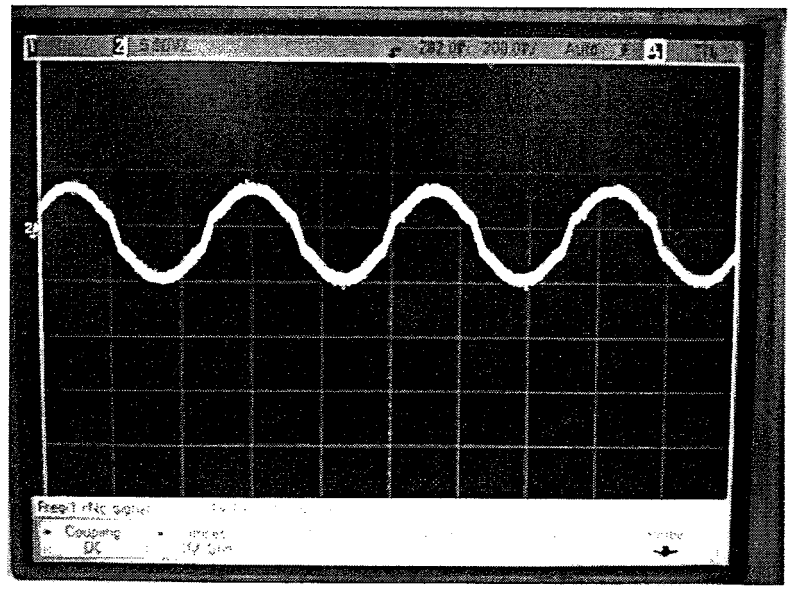
รูปที่ 4.2 สัญญาณ PWM จากไมโครคอนโทรลเลอร์ ทั้ง 6 ขา

4.1.2 สัญญาณ PWM หลังผ่านวงจรกรองความถี่ต่ำผ่าน (Low Pass Filter)

ในการวัดสัญญาณ PWM โดยผ่านวงจรกรองความถี่ต่ำผ่าน จะใช้การวัดสัญญาณแบบอนาล็อก โดยจะใช้สายโพรบธรรมดาในการวัด โดยรูปแบบในการวัดนั้นจะใช้คาปาซิเตอร์ ต่อคร่อมระหว่างขาที่ไปขับเกทตัวบนกับตัวล่าง ดังแสดงดังรูปที่ 4.3



รูปที่ 4.3 รูปแบบการวัดสัญญาณ PWM แบบอนาล็อก



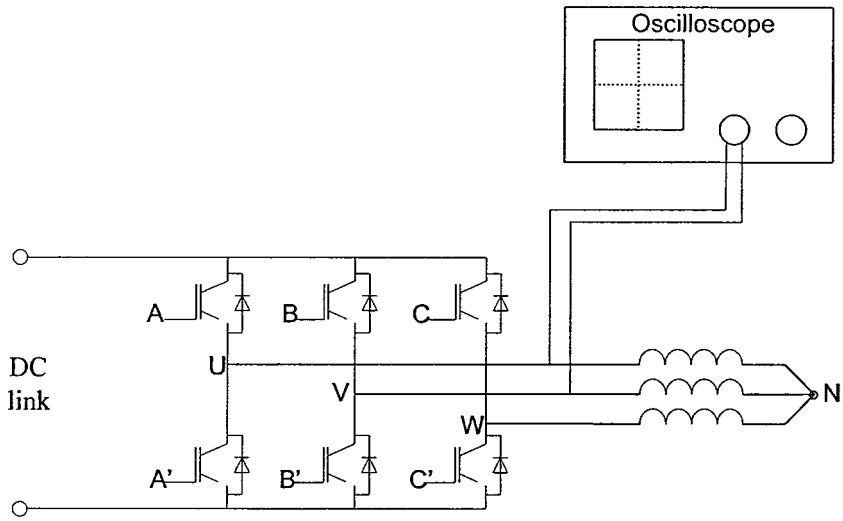
รูปที่ 4.4 สัญญาณ PWM หลังผ่านวงจรกรองความถี่ต่ำผ่าน

สัญญาณที่ได้หลังจากผ่านวงจรกรองความถี่ต่ำผ่านแล้ว จะมีลักษณะคล้ายเคลื่อน สัญญาณซานย์

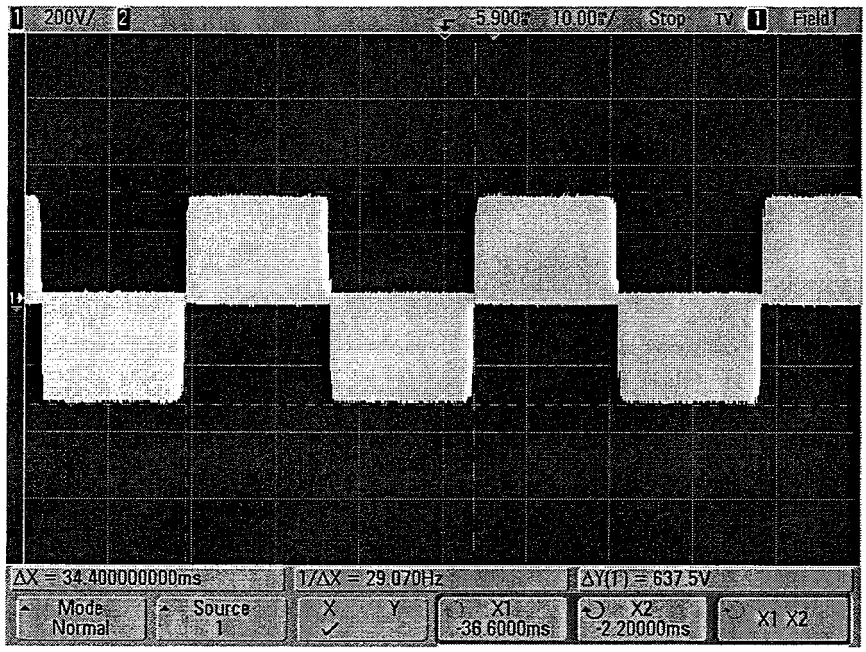
4.2 ผลการทดสอบวัดแรงดันระหว่างสายและแรงดันเฟส

4.2.1 ผลการวัดแรงดันระหว่างสาย

ในการวัดแรงดันระหว่างสาย จะทำการวัดคร่อมขั้วเอาต์พุตของอินเวอร์เตอร์ระหว่างขั้ว U กับ V ดังรูปที่ 4.5



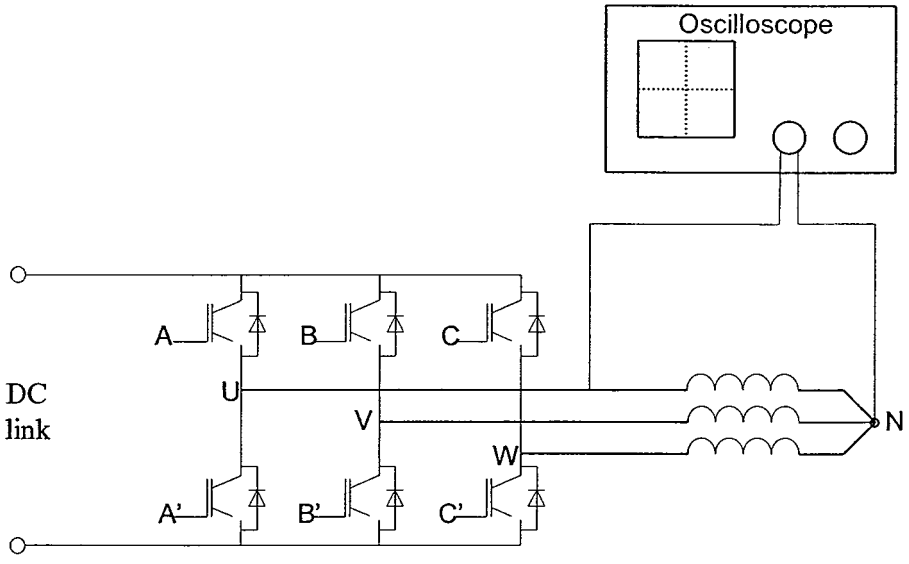
รูปที่ 4.5 แสดงการวัดแรงดันระหว่างสาย



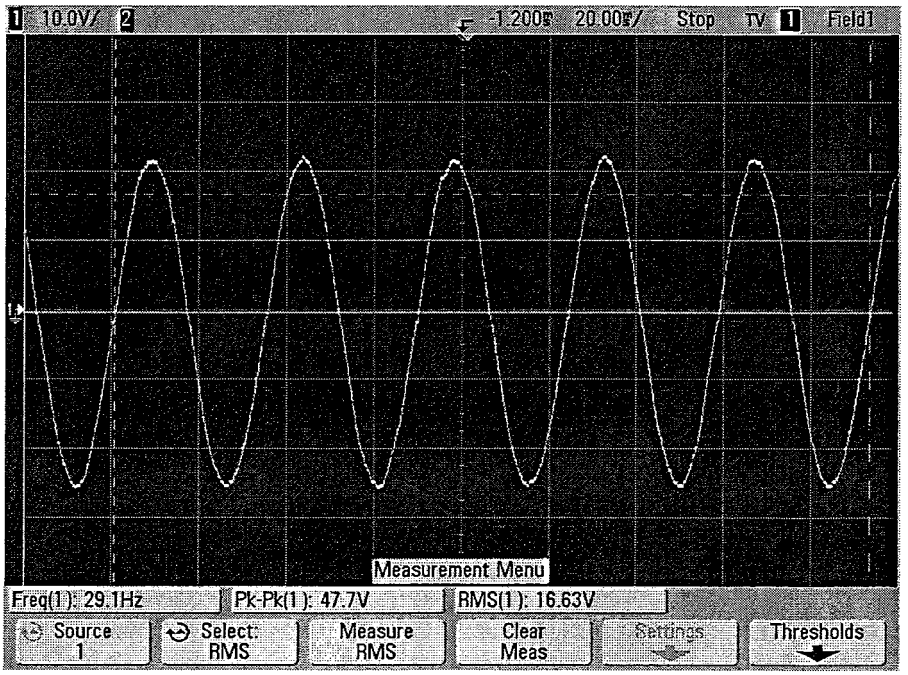
รูปที่ 4.6 สัญญาณแรงดันระหว่างสาย

4.2.2 ผลการวัดแรงดันเฟส

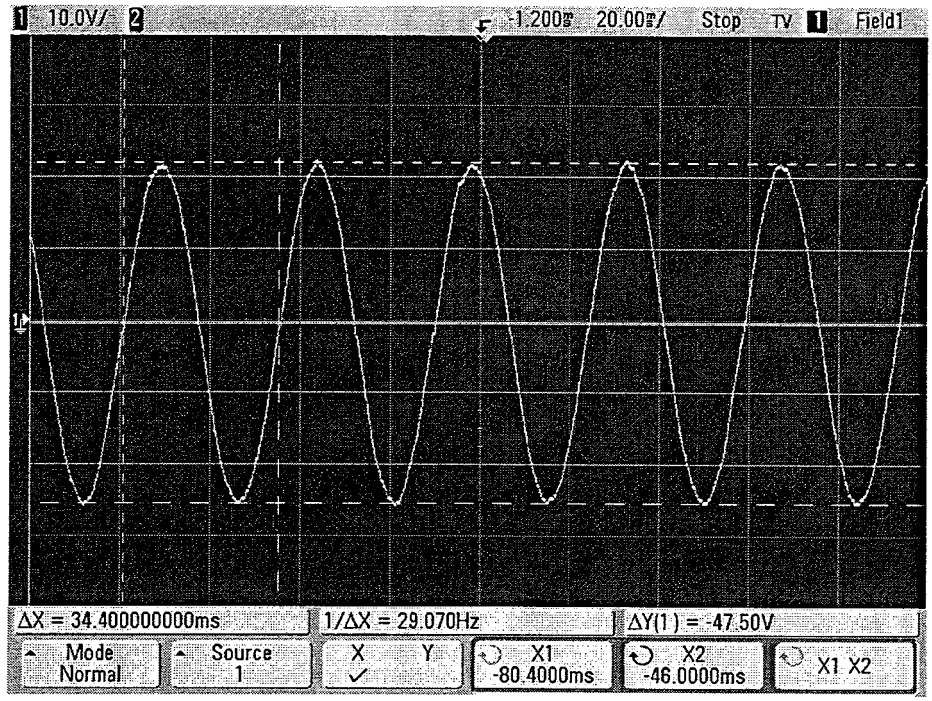
ในการวัดแรงดันเฟสนั้น จะวัดโดยใช้ออสซิลโลสโคปวัดสัญญาณระหว่างขั้ว U กับ ขั้ว N ที่มอเตอร์ ดังรูปที่ 4.7



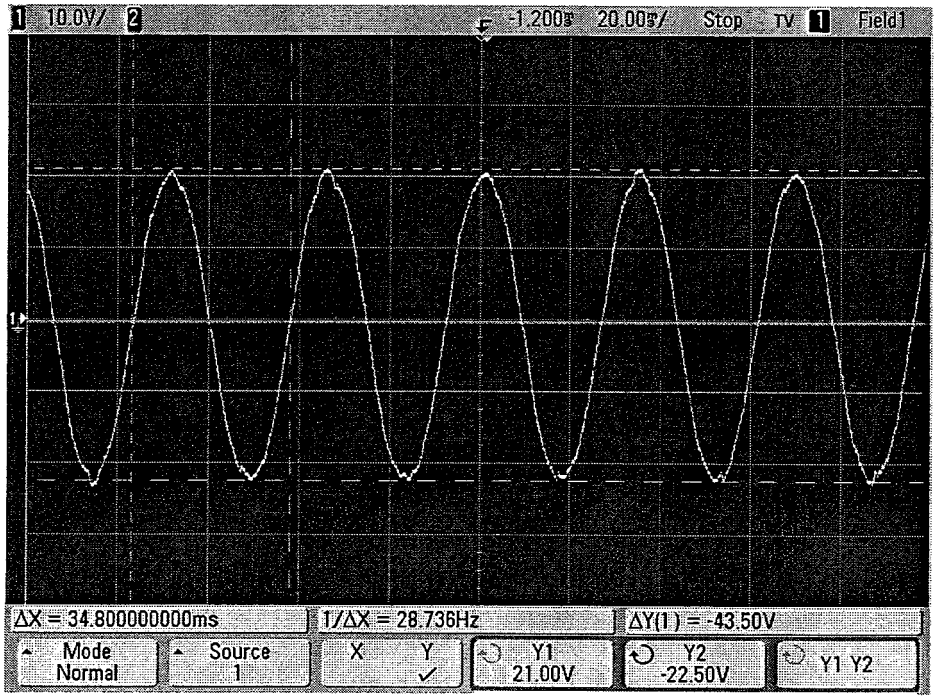
รูปที่ 4.7 แสดงการวัดแรงดันเฟส



รูปที่ 4.8 สัญญาณการวัดแรงดันเฟส U



รูปที่ 4.9 สัญญาณการวัดแรงดันเฟส V



รูปที่ 4.10 สัญญาณการวัดแรงดันเฟส W

บทที่ 5
สรุปผลและข้อเสนอแนะ

5.1 สรุปผลการทดสอบ

ปริญญานิพนธ์นี้ได้ศึกษาวิธีการออกแบบอัลกอลิธึม ในการออกแบบระบบอินเวอร์เตอร์ โดยใช้วิธีการแบบสเปซเวกเตอร์แรงดัน เพื่อนำมาควบคุมความเร็วมอเตอร์สามเฟส

ในปริญญานิพนธ์นี้ ได้นำเทคนิคที่ได้จากการค้นคว้า มาประยุกต์ใช้งานกับตัวไมโครคอนโทรลเลอร์ที่ใช้สร้างสัญญาณ PWM ซึ่งภายในตัวไมโครคอนโทรลเลอร์ จะมีโมดูลตัวประมวลผลสัญญาณดิจิตอล และโมดูลตัวสร้างสัญญาณ PWM ที่มีความสามารถในการควบคุมมอเตอร์โดยเฉพาะ โดยชุดวงจรอินเวอร์เตอร์ได้ใช้ โมดูล IGBT ที่มีพิกัดแรงดัน 600 โวลต์ และพิกัดกระแส 15 แอมป์ ใช้เป็นตัวขับเคลื่อนมอเตอร์

จากผลการทดลองการขับเคลื่อนระบบอินเวอร์เตอร์ สามารถปรับความถี่ได้ตั้งแต่ 1-60 เฮิร์ท ตามที่ได้ออกแบบไว้ โดยหน้าจอจะแสดงความถี่ของมอเตอร์ด้วย และสามารถปรับแรงบิดเพิ่มได้อย่างอิสระ โดยที่ความถี่เท่าเดิม แต่ความร้อนของอินเวอร์เตอร์ค่อนข้างสูง จึงต้องทดสอบในระยะเวลาไม่นานมาก

5.2 ข้อเสนอแนะในการปรับปรุงระบบในอนาคต

5.2.1 เนื่องจากการออกแบบและการสร้างอินเวอร์เตอร์มีข้อบกพร่องอยู่ ซึ่งยังไม่มีวงจรป้องกัน จึงอาจทำให้อินเวอร์เตอร์เสียหายได้

5.2.2 ควรปรับปรุงให้ระบบมีเสถียรภาพมากกว่านี้

5.2.3 การควบคุมอินเวอร์เตอร์แบบสามระดับโดยวิธีสเปซเวกเตอร์พัลส์วิดธ์มอดูเลชั่นนี้ ไม่ได้นำไปทดสอบกับการทำงานแบบควบคุมแรงบิดโดยตรง (Direct Torque Control) จึงควรที่จะมีการศึกษาเพื่อนำวิธีการนี้ไปทดสอบกับการทำงานในแบบดังกล่าว

5.2.4 เนื่องจากอินเวอร์เตอร์ที่ออกแบบไว้มีความร้อนสูง จึงต้องปรับปรุงระบบอินเวอร์เตอร์ให้มีความร้อนน้อยลง

เอกสารอ้างอิง

1. D.Grahame Holmes Thomas A. Lipo, " Pulse Width Modulation For Power Converter Principle & Practice " , WILEY INTERSCIENCE, ISBN. 0-471-20814-0
2. J.O.P. Pinto,B.K. Bose, L.E.B. da Silva, and M.P. Kazmierkowski. " A neural network base space vector PWM controller for voltage-fed inverter induction motor drive." IEEE Trans.Ind.Applicent.vol.36,(Nov./Dec.2000):1628-36
3. Microchip,"dsPIC30F4011 [Motor Control PWM] – Ref.Manual ", Revision E
4. Microchip,"dsPIC30F4011[Timer] ",Revision D
5. Microchip,"dsPIC30F4011 Reference Manual ",Revision D, June 2004

ภาคผนวก ก.

ไมโครคอนโทรลเลอร์ dsPIC30F4011

คุณสมบัติที่สำคัญของไมโครคอนโทรลเลอร์

dsPIC30F4011/4012 Enhanced Flash 16-Bit Digital Signal Controller

Note: This data sheet summarizes features of this group of dsPIC30F devices and is not intended to be a complete reference source. For more information on the CPU, peripherals, register descriptions and general device functionality, refer to the "*dsPIC30F Family Reference Manual*" (DS70046). For more information on the device instruction set and programming, refer to the "*dsPIC30F/33F Programmer's Reference Manual*" (DS70157).

High-Performance, Modified RISC CPU:

- Modified Harvard architecture
- C compiler optimized instruction set architecture with flexible addressing modes
- 83 base instructions
- 24-bit wide instructions, 16-bit wide data path
- 48 Kbytes on-chip Flash program space (16K instruction words)
- 2 Kbytes of on-chip data RAM
- 1 Kbyte of nonvolatile data EEPROM
- Up to 30 MIPS operation:
  - DC to 40 MHz external clock input
  - 4 MHz-10 MHz oscillator input with PLL active (4x, 8x, 16x)
- 30 interrupt sources:
  - 3 external interrupt sources
  - 8 user-selectable priority levels for each interrupt source
  - 4 processor trap sources
- 16 x 16-bit working register array

DSP Engine Features:

- Dual data fetch
- Accumulator write-back for DSP operations
- Modulo and Bit-Reversed Addressing modes
- Two, 40-bit wide accumulators with optional saturation logic
- 17-bit x 17-bit single-cycle hardware fractional/integer multiplier
- All DSP instructions are single cycle
- ±16-bit, single-cycle shift

Peripheral Features:

- High-current sink/source I/O pins: 25 mA/25 mA
- Timer module with programmable prescaler:
  - Five 16-bit timers/counters; optionally pair 16-bit timers into 32-bit timer modules
- 16-bit Capture input functions
- 16-bit Compare/PWM output functions
- 3-wire SPI modules (supports 4 Frame modes)
- I^2C™ module supports Multi-Master/Slave mode and 7-bit/10-bit addressing
- 2 UART modules with FIFO Buffers
- 1 CAN module, 2.0B compliant

Motor Control PWM Module Features:

- 6 PWM output channels:
  - Complementary or Independent Output modes
  - Edge and Center-Aligned modes
- 3 duty cycle generators
- Dedicated time base
- Programmable output polarity
- Dead-time control for Complementary mode
- Manual output control
- Trigger for A/D conversions

Quadrature Encoder Interface Module Features:

- Phase A, Phase B and Index Pulse input
- 16-bit up/down position counter
- Count direction status
- Position Measurement (x2 and x4) mode
- Programmable digital noise filters on inputs
- Alternate 16-Bit Timer/Counter mode
- Interrupt on position counter rollover/underflow

คุณสมบัติที่สำคัญของไมโครคอนโทรลเลอร์ (ต่อ)

dsPIC30F4011/4012

Analog Features:

- 10-Bit Analog-to-Digital Converter (A/D) with 4 S/H inputs:
  - 1 Msps conversion rate
  - 9 input channels
  - Conversion available during Sleep and Idle
- Programmable Brown-out Reset

Special Digital Signal Controller Features:

- Enhanced Flash program memory:
  - 10,000 erase/write cycle (min.) for industrial temperature range, 100K (typical)
- Data EEPROM memory:
  - 100,000 erase/write cycle (min.) for industrial temperature range, 1M (typical)
- Self-reprogrammable under software control
- Power-on Reset (POR), Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)

Special Digital Signal Controller Features (Cont.):

- Flexible Watchdog Timer (WDT) with on-chip, low-power RC oscillator for reliable operation
- Fail-Safe Clock Monitor operation detects clock failure and switches to on-chip, low-power RC oscillator
- Programmable code protection
- In-Circuit Serial Programming™ (ICSP™)
- Selectable Power Management modes:
  - Sleep, Idle and Alternate Clock modes

CMOS Technology:

- Low-power, high-speed Flash technology
- Wide operating voltage range (2.5V to 5.5V)
- Industrial and Extended temperature ranges
- Low-power consumption

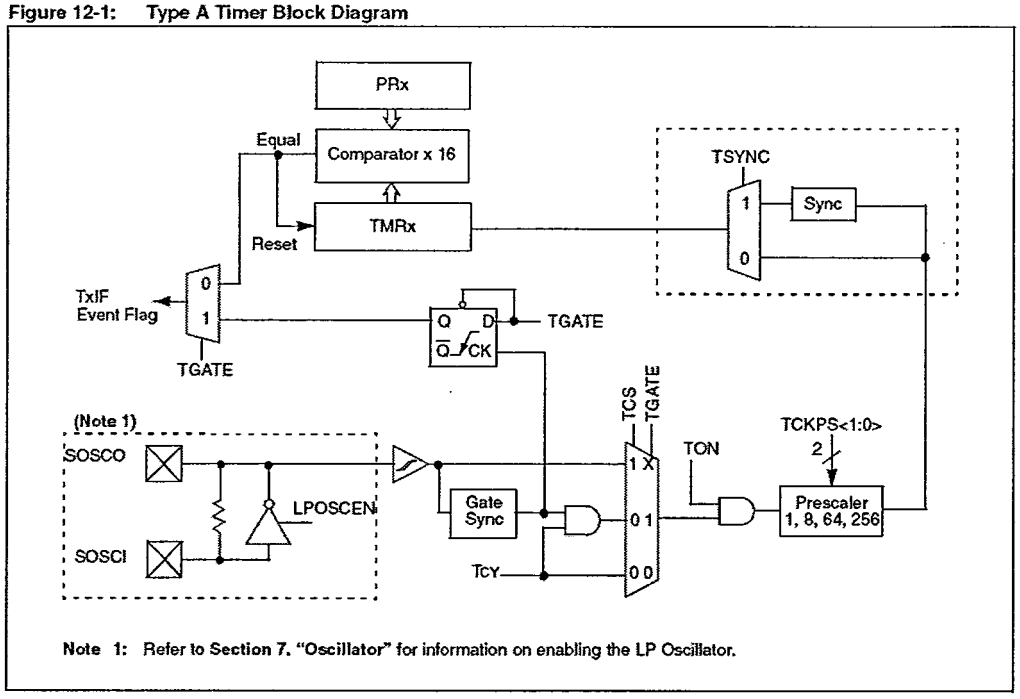
การใช้งานโมดูลไทม์เมอร์ (Timer Module) type A

At least one Type A timer is available on most dsPIC30F devices. For most dsPIC30F devices, Timer1 is a Type A timer. A Type A timer has the following unique features over other types:

- can be operated from the device Low Power 32 kHz Oscillator
- can be operated in an Asynchronous mode from an external clock source

In particular, the unique features of a Type A timer allow it to be used for Real-Time Clock (RTC) applications. A block diagram of the Type A timer is shown in Figure 12-1.

Figure 12-1: Type A Timer Block Diagram



Note 1: Refer to **Section 7. "Oscillator"** for information on enabling the LP Oscillator.

Register 12-1: TxCON: Type A Time Base Register

Upper Byte:							
R/W-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
TON	—	TSIDL	—	—	—	—	—
bit 15							bit 8

Lower Byte:							
U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	U-0
—	TGATE	TCKPS<1:0>		—	TSYNC	TCS	—
bit 7							bit 0

bit 15 **TON:** Timer On Control bit
1 = Starts the timer
0 = Stops the timer

bit 14 **Unimplemented:** Read as '0'

bit 13 **TSIDL:** Stop in Idle Mode bit
1 = Discontinue timer operation when device enters Idle mode
0 = Continue timer operation in Idle mode

bit 12-7 **Unimplemented:** Read as '0'

bit 6 **TGATE:** Timer Gated Time Accumulation Enable bit
1 = Gated time accumulation enabled
0 = Gated time accumulation disabled
(TCS must be set to '0' when TGATE = 1. Reads as '0' if TCS = 1)

bit 5-4 **TCKPS<1:0>:** Timer Input Clock Prescale Select bits
11 = 1:256 prescale value
10 = 1:64 prescale value
01 = 1:8 prescale value
00 = 1:1 prescale value

bit 3 **Unimplemented:** Read as '0'

bit 2 **TSYNC:** Timer External Clock Input Synchronization Select bit
When TCS = 1:
1 = Synchronize external clock input
0 = Do not synchronize external clock input
When TCS = 0:
This bit is ignored. Read as '0'. Timer1 uses the internal clock when TCS = 0.

bit 1 **TCS:** Timer Clock Source Select bit
1 = External clock from pin TxCK
0 = Internal clock (Fosc/4)

bit 0 **Unimplemented:** Read as '0'

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

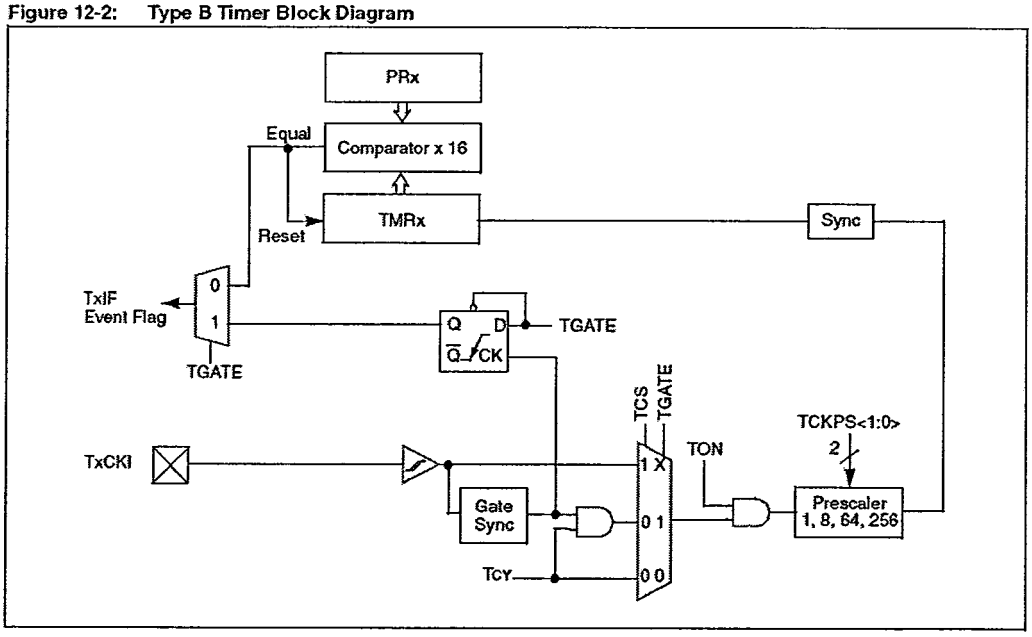
การใช้งานโมดูลไทม์เมอร์ (Timer Module) type B

Timer2 and Timer4, if present, are Type B timers on most dsPIC30F devices. A Type B timer has the following unique features over other types of timers:

- A Type B timer can be concatenated with a Type C timer to form a 32-bit timer. The TxCON register for a Type B timer has the T32 control bits to enable the 32-bit timer function.
- The clock synchronization for a Type B timer is performed after the prescale logic.

A block diagram of the Type B timer is shown in Figure 12-2.

Figure 12-2: Type B Timer Block Diagram



Register 12-2: TxCON: Type B Time Base Register

Upper Byte:							
R/W-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
TON	—	TSIDL	—	—	—	—	—
bit 15							bit 8

Lower Byte:							
U-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	U-0
—	TGATE	TCKPS<1:0>		T32	—	TCS	—
bit 7							bit 0

bit 15 **TON:** Timer On bit
When T32 = 1 (in 32-bit Timer mode):
1 = Starts 32-bit TMRx:TMRy timer pair
0 = Stops 32-bit TMRx:TMRy timer pair
When T32 = 0 (in 16-bit Timer mode):
1 = Starts 16-bit timer
0 = Stops 16-bit timer

bit 14 **Unimplemented:** Read as '0'

bit 13 **TSIDL:** Stop in Idle Mode bit
1 = Discontinue timer operation when device enters Idle mode
0 = Continue timer operation in Idle mode

bit 12-7 **Unimplemented:** Read as '0'

bit 6 **TGATE:** Timer Gated Time Accumulation Enable bit
1 = Timer gated time accumulation enabled
0 = Timer gated time accumulation disabled
(TCS must be set to logic '0' when TGATE = 1)

bit 5-4 **TCKPS<1:0>:** Timer Input Clock Prescale Select bits
11 = 1:256 prescale value
10 = 1:64 prescale value
01 = 1:8 prescale value
00 = 1:1 prescale value

bit 3 **T32:** 32-bit Timer Mode Select bits
1 = TMRx and TMRy form a 32-bit timer
0 = TMRx and TMRy form separate 16-bit timer

bit 2 **Unimplemented:** Read as '0'

bit 1 **TCS:** Timer Clock Source Select bit
1 = External clock from pin TxCK
0 = Internal clock (Fosc/4)

bit 0 **Unimplemented:** Read as '0'

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

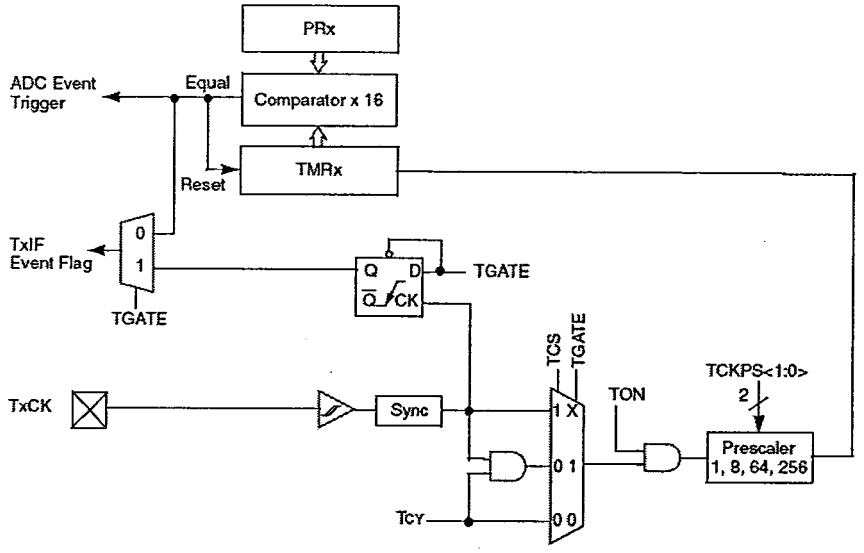
การใช้งานโมดูลไทม์เมอร์ (Timer Module) type C

Timer3 and Timer5 are Type C timers on most dsPIC30F devices. A Type C timer has the following unique features over other types of timers:

- A Type C timer can be concatenated with a Type B timer to form a 32-bit timer.
- On a given device, at least one Type C timer has the ability to trigger an A/D conversion.

A block diagram of the Type C timer is shown in Figure 12-3.

Figure 12-3: Type C Timer Block Diagram



Note: In certain variants of the dsPIC30F family, the TxCK pin may not be available. Refer to the device data sheet for the I/O pin details. In such cases, the timer must use the system clock (FOSC/4) as its input clock, unless it is configured for 32-bit operation.

Register 12-3: TxCON: Type C Time Base Register

Upper Byte:							
R/W-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
TON	—	TSIDL	—	—	—	—	—
bit 15							bit 8

Lower Byte:							
U-0	R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0	U-0
—	TGATE	TCKPS<1:0>		—	—	TCS	—
bit 7							bit 0

bit 15 **TON:** Timer On bit
1 = Starts 16-bit TMRx
0 = Stops 16-bit TMRx

bit 14 **Unimplemented:** Read as '0'

bit 13 **TSIDL:** Stop in Idle Mode bit
1 = Discontinue module operation when device enters Idle mode
0 = Continue module operation in Idle mode

bit 12-7 **Unimplemented:** Read as '0'

bit 6 **TGATE:** Timer Gated Time Accumulation Enable bit
1 = Timer gated time accumulation enabled
0 = Timer gated time accumulation disabled (Read as '0' if TCS = 1)
(TCS must be set to logic '0' when TGATE = 1)

bit 5-4 **TCKPS<1:0>:** Timer Input Clock Prescale Select bits
11 = 1:256 prescale value
10 = 1:64 prescale value
01 = 1:8 prescale value
00 = 1:1 prescale value

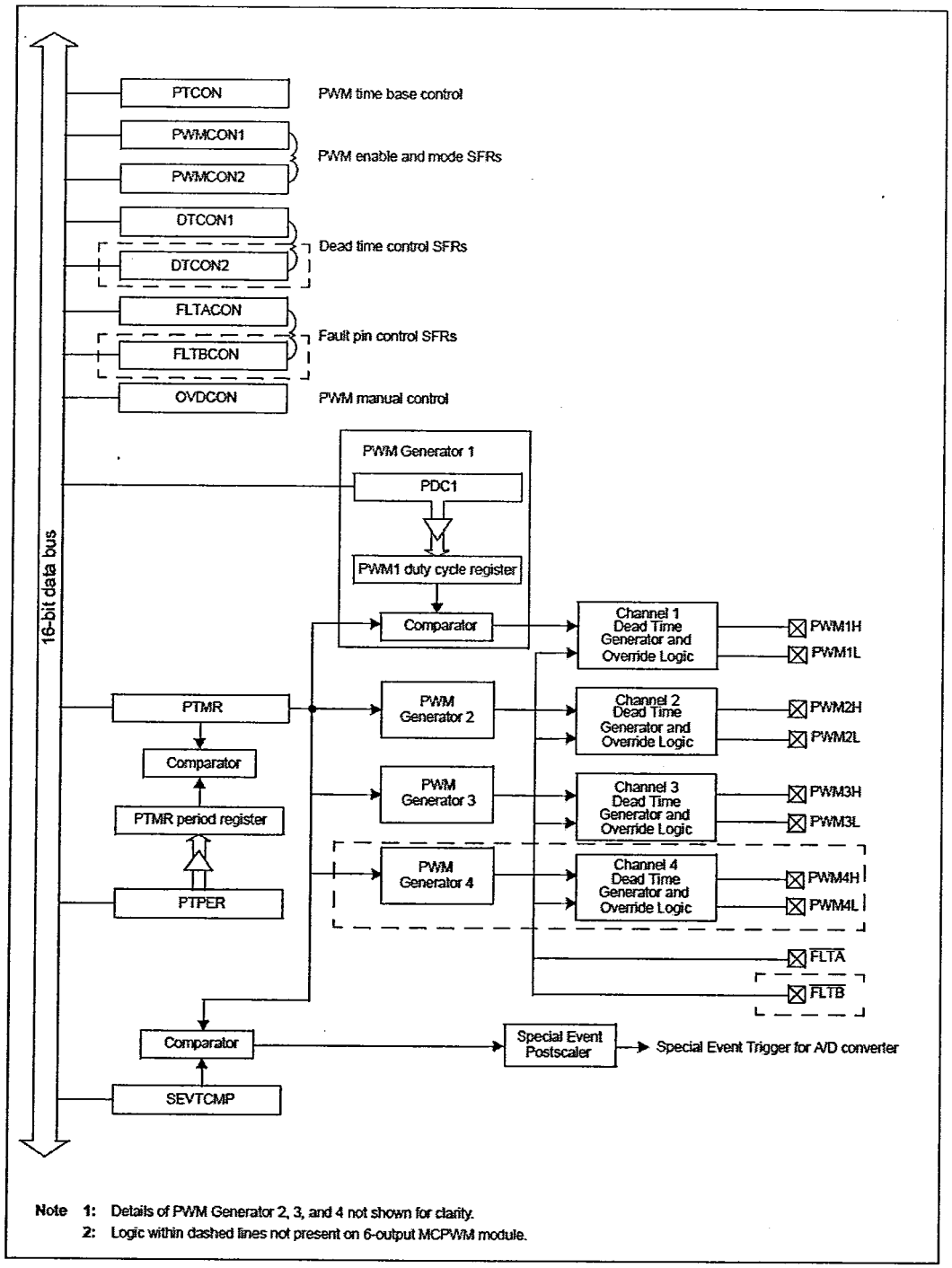
bit 3-2 **Unimplemented:** Read as '0'

bit 1 **TCS:** Timer Clock Source Select bit
1 = External clock from pin TxCK
0 = Internal clock (FOSC/4)

bit 0 **Unimplemented:** Read as '0'

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

การใช้งานพัลส์วิดธ์มอดูเลชั่น (PWM)



Note 1: Details of PWM Generator 2, 3, and 4 not shown for clarity.
2: Logic within dashed lines not present on 6-output MCPWM module.

Register 15-1: PTCON: PWM Time Base Control Register

Upper Byte:							
R/W-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
PTEN	—	PTSIDL	—	—	—	—	—
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTOPS<3:0>				PTCKPS<1:0>		PTMOD<1:0>	
bit 7							bit 0

bit 15 **PTEN:** PWM Time Base Timer Enable bit
1 = PWM time base is ON
0 = PWM time base is OFF

bit 14 **Unimplemented:** Read as '0'

bit 13 **PTSIDL:** PWM Time Base Stop in Idle Mode bit
1 = PWM time base halts in CPU Idle mode
0 = PWM time base runs in CPU Idle mode

bit 12-8 **Unimplemented:** Read as '0'

bit 7-4 **PTOPS<3:0>:** PWM Time Base Output Postscale Select bits
1111 = 1:16 Postscale
•
•
0001 = 1:2 Postscale
0000 = 1:1 Postscale

bit 3-2 **PTCKPS<1:0>:** PWM Time Base Input Clock Prescale Select bits
11 = PWM time base input clock period is 64 TCY (1:64 prescale)
10 = PWM time base input clock period is 16 TCY (1:16 prescale)
01 = PWM time base input clock period is 4 TCY (1:4 prescale)
00 = PWM time base input clock period is TCY (1:1 prescale)

bit 1-0 **PTMOD<1:0>:** PWM Time Base Mode Select bits
11 = PWM time base operates in a continuous up/down mode with interrupts for double PWM updates
10 = PWM time base operates in a continuous up/down counting mode
01 = PWM time base operates in single event mode
00 = PWM time base operates in a free running mode

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

Register 15-2: PTMR: PWM Time Base Register

Upper Byte:							
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTDIR	PTMR <14:8>						
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTMR <7:0>							
bit 7							bit 0

bit 15 **PTDIR:** PWM Time Base Count Direction Status bit (Read Only)
1 = PWM time base is counting down
0 = PWM time base is counting up

bit 14-0 **PTMR <14:0>:** PWM Time Base Register Count Value

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

Register 15-3: PTPER: PWM Time Base Period Register

Upper Byte:							
U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	PTPER <14:8>						
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTPER <7:0>							
bit 7							bit 0

bit 15 **Unimplemented:** Read as '0'

bit 14-0 **PTPER<14:0>:** PWM Time Base Period Value bits

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

Register 15-4: SEVTCMP: Special Event Compare Register

Upper Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SEVTDIR	SEVTCMP <14:8>						
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SEVTCMP <7:0>							
bit 7							bit 0

bit 15 **SEVTDIR:** Special Event Trigger Time Base Direction bit[1]
1 = A special event trigger will occur when the PWM time base is counting downwards.
0 = A special event trigger will occur when the PWM time base is counting upwards.

bit 14-0 **SEVTCMP <14:0>:** Special Event Compare Value bit[2]

Note 1: SEVTDIR is compared with PTDIR (PTMR<15>) to generate the special event trigger.
2: SEVTCMP<14:0> is compared with PTMR<14:0> to generate the special event trigger.

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

Register 15-5: PWMCON1: PWM Control Register 1

Upper Byte:							
U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	PMOD4	PMOD3	PMOD2	PMOD1
bit 15							bit 8

Lower Byte:							
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
PEN4H	PEN3H	PEN2H	PEN1H	PEN4L	PEN3L	PEN2L	PEN1L
bit 7							bit 0

bit 15-12 **Unimplemented:** Read as '0'

bit 11-8 **PMOD4:PMOD1:** PWM I/O Pair Mode bits
1 = PWM I/O pin pair is in the independent output mode
0 = PWM I/O pin pair is in the complementary output mode

bit 7-4 **PEN4H-PEN1H:** PWMxH I/O Enable bits[1]
1 = PWMxH pin is enabled for PWM output
0 = PWMxH pin disabled. I/O pin becomes general purpose I/O

bit 3-0 **PEN4L-PEN1L:** PWMxL I/O Enable bits[1]
1 = PWMxL pin is enabled for PWM output
0 = PWMxL pin disabled. I/O pin becomes general purpose I/O

Note 1: Reset condition of the PENxH and PENxL bits depend on the value of the PWM/PIN device configuration bit in the FBORPOR Device Configuration Register.

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

Register 15-6: PWMCON2: PWM Control Register 2

Upper Byte:							
U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	SEVOPS<3:0>			
bit 15							bit 8

Lower Byte:							
U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
—	—	—	—	—	IUE	OSYNC	UDIS
bit 7							bit 0

bit 15-12 **Unimplemented:** Read as '0'

bit 11-8 **SEVOPS<3:0>:** PWM Special Event Trigger Output Postscale Select bits
1111 = 1:16 Postscale
•
•
0001 = 1:2 Postscale
0000 = 1:1 Postscale

bit 7-2 **Unimplemented:** Read as '0'

bit 2 **IUE:** Immediate Update Enable bit(1)
1 = Updates to the active PDC registers are immediate
0 = Updates to the active PDC registers are synchronized to the PWM time base

bit 1 **OSYNC:** Output Override Synchronization bit
1 = Output overrides via the OVDCON register are synchronized to the PWM time base
0 = Output overrides via the OVDCON register occur on next TCY boundary

bit 0 **UDIS:** PWM Update Disable bit
1 = Updates from duty cycle and period buffer registers are disabled
0 = Updates from duty cycle and period buffer registers are enabled

Note 1: IUE bit is not implemented on the dsPIC30F6010 device.

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

Register 15-7: DTCON1: Dead Time Control Register 1

Upper Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DTBPS<1:0>		DTB<5:0>					
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DTAPS<1:0>		DTA<5:0>					
bit 7							bit 0

bit 15-14 **DTBPS<1:0>:** Dead Time Unit B Prescale Select bits
11 = Clock period for Dead Time Unit B is 8 TCY
10 = Clock period for Dead Time Unit B is 4 TCY
01 = Clock period for Dead Time Unit B is 2 TCY
00 = Clock period for Dead Time Unit B is TCY

bit 13-8 **DTB<5:0>:** Unsigned 6-bit Dead Time Value bits for Dead Time Unit B

bit 7-6 **DTAPS<1:0>:** Dead Time Unit A Prescale Select bits
11 = Clock period for Dead Time Unit A is 8 TCY
10 = Clock period for Dead Time Unit A is 4 TCY
01 = Clock period for Dead Time Unit A is 2 TCY
00 = Clock period for Dead Time Unit A is TCY

bit 5-0 **DTA<5:0>:** Unsigned 6-bit Dead Time Value bits for Dead Time Unit A

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

Register 15-8: DTCON2: Dead Time Control Register 2

Upper Byte:							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DTS4A	DTS4I	DTS3A	DTS3I	DTS2A	DTS2I	DTS1A	DTS1I
bit 7							bit 0

bit 15-8 **Unimplemented:** Read as '0'

bit 7 **DTS4A:** Dead Time Select bit for PWM4 Signal Going Active
1 = Dead time provided from Unit B
0 = Dead time provided from Unit A

bit 6 **DTS4I:** Dead Time Select bit for PWM4 Signal Going Inactive
1 = Dead time provided from Unit B
0 = Dead time provided from Unit A

bit 5 **DTS3A:** Dead Time Select bit for PWM3 Signal Going Active
1 = Dead time provided from Unit B
0 = Dead time provided from Unit A

bit 4 **DTS3I:** Dead Time Select bit for PWM3 Signal Going Inactive
1 = Dead time provided from Unit B
0 = Dead time provided from Unit A

bit 3 **DTS2A:** Dead Time Select bit for PWM2 Signal Going Active
1 = Dead time provided from Unit B
0 = Dead time provided from Unit A

bit 2 **DTS2I:** Dead Time Select bit for PWM2 Signal Going Inactive
1 = Dead time provided from Unit B
0 = Dead time provided from Unit A

bit 1 **DTS1A:** Dead Time Select bit for PWM1 Signal Going Active
1 = Dead time provided from Unit B
0 = Dead time provided from Unit A

bit 0 **DTS1I:** Dead Time Select bit for PWM1 Signal Going Inactive
1 = Dead time provided from Unit B
0 = Dead time provided from Unit A

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

Register 15-9: FLTACON: Fault A Control Register

Upper Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
FAOV4H	FAOV4L	FAOV3H	FAOV3L	FAOV2H	FAOV2L	FAOV1H	FAOV1L
bit 15							bit 8

Lower Byte:							
R/W-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
FLTAM	—	—	—	FAEN4	FAEN3	FAEN2	FAEN1
bit 7							bit 0

bit 15-8 **FAOV4H-FAOV1L:** Fault Input A PWM Override Value bits
1 = The PWM output pin is driven ACTIVE on an external fault input event
0 = The PWM output pin is driven INACTIVE on an external fault input event

bit 7 **FLTAM:** Fault A Mode bit
1 = The Fault A input pin functions in the cycle-by-cycle mode
0 = The Fault A input pin latches all control pins to the programmed states in FLTACON<15:8>

bit 6-4 **Unimplemented:** Read as '0'

bit 3 **FAEN4:** Fault Input A Enable bit
1 = PWM4H/PWM4L pin pair is controlled by Fault Input A
0 = PWM4H/PWM4L pin pair is not controlled by Fault Input A

bit 2 **FAEN3:** Fault Input A Enable bit
1 = PWM3H/PWM3L pin pair is controlled by Fault Input A
0 = PWM3H/PWM3L pin pair is not controlled by Fault Input A

bit 1 **FAEN2:** Fault Input A Enable bit
1 = PWM2H/PWM2L pin pair is controlled by Fault Input A
0 = PWM2H/PWM2L pin pair is not controlled by Fault Input A

bit 0 **FAEN1:** Fault Input A Enable bit
1 = PWM1H/PWM1L pin pair is controlled by Fault Input A
0 = PWM1H/PWM1L pin pair is not controlled by Fault Input A

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

Register 15-10: FLTBCON: Fault B Control Register

Upper Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
FBOV4H	FBOV4L	FBOV3H	FBOV3L	FBOV2H	FBOV2L	FBOV1H	FBOV1L
bit 15							bit 8

Lower Byte:							
R/W-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
FLTBM	—	—	—	FBEN4	FBEN3	FBEN2	FBEN1
bit 7							bit 0

bit 15-8 **FBOV4H-FBOV1L:** Fault Input B PWM Override Value bits
1 = The PWM output pin is driven ACTIVE on an external fault input event
0 = The PWM output pin is driven INACTIVE on an external fault input event

bit 7 **FLTBM:** Fault B Mode bit
1 = The Fault B input pin functions in the cycle-by-cycle mode
0 = The Fault B input pin latches all control pins to the programmed states in FLTBCON<15:8>

bit 6-4 **Unimplemented:** Read as '0'

bit 3 **FAEN4:** Fault Input B Enable bit(1)
1 = PWM4H/PWM4L pin pair is controlled by Fault Input B
0 = PWM4H/PWM4L pin pair is not controlled by Fault Input B

bit 2 **FAEN3:** Fault Input B Enable bit(1)
1 = PWM3H/PWM3L pin pair is controlled by Fault Input B
0 = PWM3H/PWM3L pin pair is not controlled by Fault Input B

bit 1 **FAEN2:** Fault Input B Enable bit(1)
1 = PWM2H/PWM2L pin pair is controlled by Fault Input B
0 = PWM2H/PWM2L pin pair is not controlled by Fault Input B

bit 0 **FAEN1:** Fault Input B Enable bit(1)
1 = PWM1H/PWM1L pin pair is controlled by Fault Input B
0 = PWM1H/PWM1L pin pair is not controlled by Fault Input B

Note 1: Fault pin A has priority over Fault pin B, if enabled.

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

Register 15-11: OVDCON: Override Control Register

Upper Byte:							
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
POVD4H	POVD4L	POVD3H	POVD3L	POVD2H	POVD2L	POVD1H	POVD1L
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
POUT4H	POUT4L	POUT3H	POUT3L	POUT2H	POUT2L	POUT1H	POUT1L
bit 7							bit 0

bit 15-8 **POVD4H-POVD1L:** PWM Output Override bits
1 = Output on PWMxx I/O pin is controlled by the PWM generator
0 = Output on PWMxx I/O pin is controlled by the value in the corresponding POUTxx bit

bit 7-0 **POUT4H-POUT1L:** PWM Manual Output bits
1 = PWMxx I/O pin is driven ACTIVE when the corresponding POVDxx bit is cleared
0 = PWMxx I/O pin is driven INACTIVE when the corresponding POVDxx bit is cleared

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

Register 15-12: PDC1: PWM Duty Cycle Register 1

Upper Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PWM Duty Cycle 1 bits 15-8							
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PWM Duty Cycle 1 bits 7-0							
bit 7							bit 0

bit 15-0 **PDC1<15:0>:** PWM Duty Cycle 1 Value bits

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

Register 15-13: PDC2: PWM Duty Cycle Register 2

Upper Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PWM Duty Cycle 2 bits 15-8							
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PWM Duty Cycle 2 bits 7-0							
bit 7							bit 0

bit 15-0 **PDC2<15:0>:** PWM Duty Cycle 2 Value bits

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

Register 15-14: PDC3: PWM Duty Cycle Register 3

Upper Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PWM Duty Cycle 3 bits 15-8							
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PWM Duty Cycle 3 bits 7-0							
bit 7							bit 0

bit 15-0 **PDC3<15:0>:** PWM Duty Cycle 3 Value bits

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

Register 15-15: PDC4: PWM Duty Cycle Register 4

Upper Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PWM Duty Cycle 4 bits 15-8							
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PWM Duty Cycle 4 bits 7-0							
bit 7							bit 0

bit 15-0 **PDC4<15:0>:** PWM Duty Cycle 4 Value bits

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

Register 15-16: FBORPOR: BOR AND POR Device Configuration Register

Upper Byte:							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

Middle Byte:							
U-0	U-0	U-0	U-0	U-0	R/P	R/P	R/P
—	—	—	—	—	PWMPIN	HPOL	LPOL
bit 15							bit 8

Lower Byte:							
R/P	U-0	R/P	R/P	U-0	U-0	R/P	R/P
BOREN	—	BORV<1:0>		—	—	FPWRT<1:0>	
bit 7							bit 0

bit 10 **PWMPIN:** MPWM Drivers Initialization bit
1 = Pin state at reset controlled by I/O Port (PWMCON1<7:0> = 0x00)
0 = Pin state at reset controlled by module (PWMCON1<7:0> = 0xFF)

bit 9 **HPOL:** MCPWM High Side Drivers (PWMxH) Polarity bit
1 = Output signal on PWMxH pins has active high polarity
0 = Output signal on PWMxH pins has active low polarity

bit 8 **LPOL:** MCPWM Low Side Drivers (PWMxL) Polarity bit
1 = Output signal on PWMxL pins has active high polarity
0 = Output signal on PWMxL pins has active low polarity

Note: See **Section 24. "Device Configuration"** for information about other configuration bits on this register.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown
P = Programmable configuration bit			

15.3 PWM Time Base

The PWM time base is provided by a 15-bit timer with a prescaler and postscaler (see Figure 15-2). The 15 bits of the time base are accessible via the PTMR register. PTMR<15> is a read-only status bit, PTDIR, that indicates the present count direction of the PWM time base. If the PTDIR status bit is cleared, PTMR is counting upwards. If PTDIR is set, PTMR is counting downwards.

The time base is enabled/disabled by setting/clearing the PTEN bit (PTCON<15>). PTMR is not cleared when the PTEN bit is cleared in software.

Figure 15-2: PWM Time Base Block Diagram

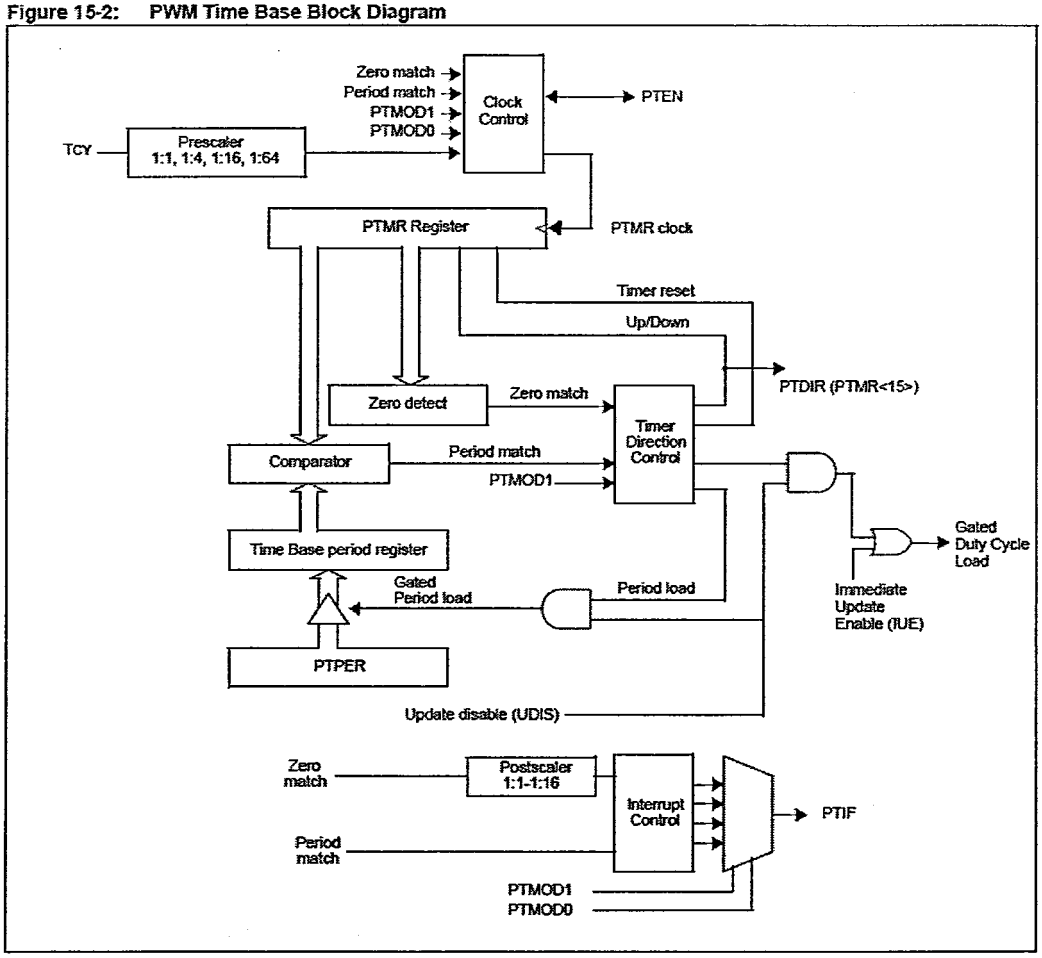
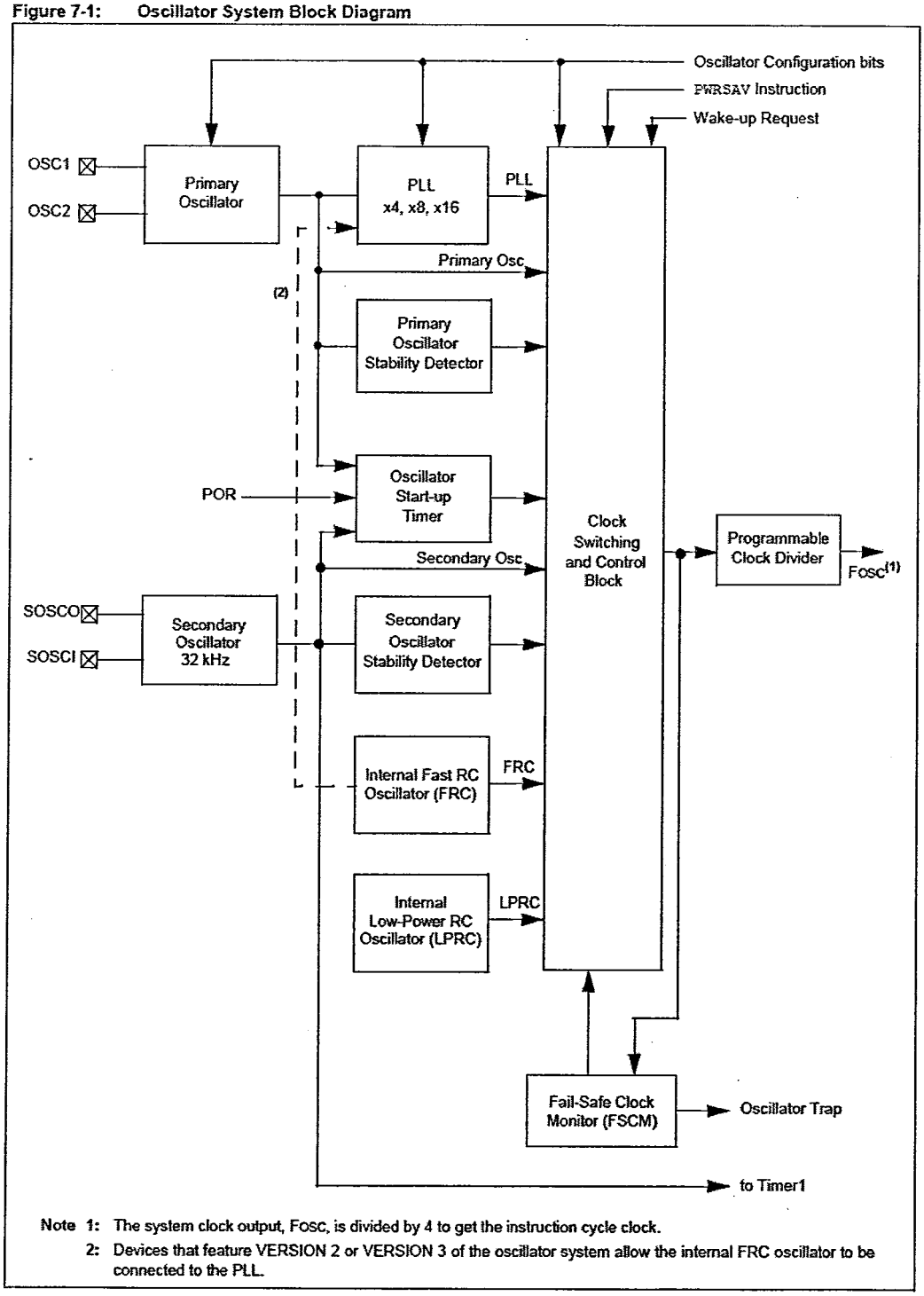


Figure 7-1: Oscillator System Block Diagram



Note 1: The system clock output, Fosc, is divided by 4 to get the instruction cycle clock.

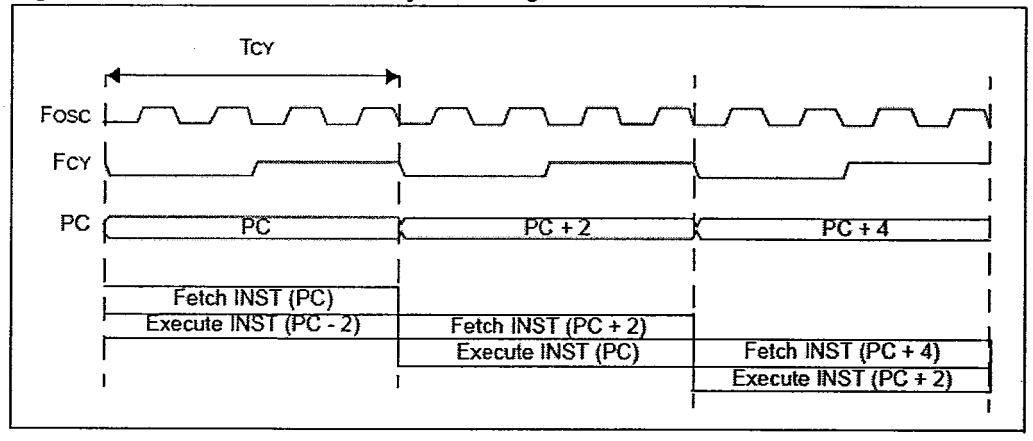
2: Devices that feature VERSION 2 or VERSION 3 of the oscillator system allow the internal FRC oscillator to be connected to the PLL.

Referring to Figure 7-1, the system clock source can be provided by one of four sources. These sources are the Primary oscillator, Secondary oscillator, Internal Fast RC (FRC) oscillator or the Low-Power RC (LPRC) oscillator. The primary oscillator source has the option of using the internal PLL. The frequency of the selected clock source can optionally be reduced by the programmable postscaler (clock divider). The output from the programmable postscaler becomes the system clock source, FOSC.

The system clock source is divided by four to produce the internal instruction cycle clock, FCY. In this document, the instruction cycle clock is also denoted by FOSC/4. The timing diagram in Figure 7-2 shows the relationship between the system clock source and instruction execution.

The internal instruction cycle clock, FCY, can be provided on the OSC2 I/O pin for some operating modes of the primary oscillator (see **Section 7.3 "Oscillator Configuration"**).

Figure 7-2: Clock/Instruction Cycle Timing



Equation 7-1: MIPS and Source Oscillator Frequency Relationship

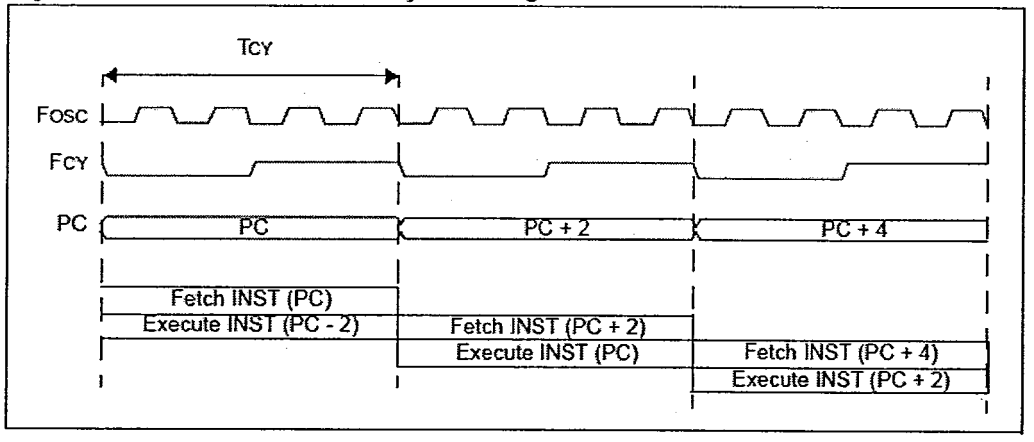
$$FCY = \frac{FOSC}{4} = \left(\frac{\text{SOURCE OSCILLATOR FREQUENCY} * \text{PLL MULTIPLIER}}{\text{PROGRAMMABLE POSTSCALER} * 4}\right)$$

Referring to Figure 7-1, the system clock source can be provided by one of four sources. These sources are the Primary oscillator, Secondary oscillator, Internal Fast RC (FRC) oscillator or the Low-Power RC (LPRC) oscillator. The primary oscillator source has the option of using the internal PLL. The frequency of the selected clock source can optionally be reduced by the programmable postscaler (clock divider). The output from the programmable postscaler becomes the system clock source, FOSC.

The system clock source is divided by four to produce the internal instruction cycle clock, FCY. In this document, the instruction cycle clock is also denoted by FOSC/4. The timing diagram in Figure 7-2 shows the relationship between the system clock source and instruction execution.

The internal instruction cycle clock, FCY, can be provided on the OSC2 I/O pin for some operating modes of the primary oscillator (see **Section 7.3 "Oscillator Configuration"**).

Figure 7-2: Clock/Instruction Cycle Timing



Equation 7-1: MIPS and Source Oscillator Frequency Relationship

$$FCY = \frac{FOSC}{4} = \left(\frac{\text{SOURCE OSCILLATOR FREQUENCY} * \text{PLL MULTIPLIER}}{\text{PROGRAMMABLE POSTSCALER} * 4}\right)$$

Register 7-1: FOSC: Oscillator Configuration Register for Oscillator System VERSION 1

Upper Byte:							
U	U	U	U	U	U	U	U
—	—	—	—	—	—	—	—
bit 23							bit 16

Middle Byte:							
R/P	R/P	U	U	U	U	R/P	R/P
FCKSM<1:0>		—	—	—	—	FOS<1:0>	
bit 15							bit 8

Lower Byte:							
U	U	U	U	R/P	R/P	R/P	R/P
—	—	—	—	FPR<3:0>			
bit 7							bit 0

bit 23-16 **Unimplemented:** Read as '0'

bit 15-14 **FCKSM<1:0>:** Clock Switching Mode Selection Fuses bits
1x = Clock switching is disabled, Fail-Safe Clock Monitor is disabled
01 = Clock switching is enabled, Fail-Safe Clock Monitor is disabled
00 = Clock switching is enabled, Fail-Safe Clock Monitor is enabled

bit 13-10 **Unimplemented:** Read as '0'

bit 9-8 **FOS<1:0>:** Oscillator Source Selection on POR bits
11 = Primary oscillator (Primary Oscillator mode selected by FPR<3:0>)
10 = Internal low-power RC oscillator
01 = Internal fast RC oscillator
00 = Low-power 32 kHz oscillator (Timer1 oscillator)

bit 7-4 **Unimplemented:** Read as '0'

bit 3-0 **FPR<3:0>:** Oscillator Selection within Primary Group bits (see Table 7-2)

Legend:		
R = Readable bit	P = Programmable bit	U = Unimplemented bit

Register 7-2: FOSC: Oscillator Configuration Register for Oscillator System VERSION 2

Upper Byte:							
U	U	U	U	U	U	U	U
—	—	—	—	—	—	—	—
bit 23							bit 16

Middle Byte:							
R/P	R/P	U	U	U	U	R/P	R/P
FCKSM<1:0>		—	—	—	—	FOS<1:0>	
bit 15							bit 8

Lower Byte:							
U	U	U	U	R/P	R/P	R/P	R/P
—	—	—	—	FPR<3:0>			
bit 7							bit 0

bit 23-16 **Unimplemented:** Read as '0'

bit 15-14 **FCKSM<1:0>:** Clock Switching Mode Selection Fuses bits
1x = Clock switching is disabled, Fail-Safe Clock Monitor is disabled
01 = Clock switching is enabled, Fail-Safe Clock Monitor is disabled
00 = Clock switching is enabled, Fail-Safe Clock Monitor is enabled

bit 13-10 **Unimplemented:** Read as '0'

bit 9-8 **FOS<1:0>:** Oscillator Source Selection on POR bits
11 = Primary oscillator (Primary Oscillator mode selected by FPR<3:0>)
10 = Internal low-power rc oscillator
01 = Internal fast RC oscillator
00 = Low-power 32 khz oscillator (Timer1 oscillator)

bit 7-4 **Unimplemented:** Read as '0'

bit 3-0 **FPR<3:0>:** Oscillator Mode Selection within Primary Group bits (see Table 7-3)

Legend:		
R = Readable bit	P = Programmable bit	U = Unimplemented bit

Register 7-3: FOSC: Oscillator Configuration Register for Oscillator System VERSION 3

Upper Byte:							
U	U	U	U	U	U	U	U
—	—	—	—	—	—	—	—
bit 23							bit 16

Middle Byte:							
R/P	R/P	U	U	U	R/P	R/P	R/P
FCKSM<1:0>		—	—	—	FOS<2:0>		
bit 15							bit 8

Lower Byte:							
U	U	U	R/P	R/P	R/P	R/P	R/P
—	—	—	FPR<4:0>				
bit 7							bit 0

bit 23-16 **Unimplemented:** Read as '0'

bit 15-14 **FCKSM<1:0>:** Clock Switching and Monitor Selection Configuration bits
1x = Clock switching is disabled, Fail-Safe Clock Monitor is disabled
01 = Clock switching is enabled, Fail-Safe Clock Monitor is disabled
00 = Clock switching is enabled, Fail-Safe Clock Monitor is enabled

bit 13-11 **Unimplemented:** Read as '0'

bit 10-8 **FOS<2:0>:** Oscillator Group Selection on POR bit
111 = PLL Oscillator; PLL source selected by FPR<4:0> bits
011 = EXT: External Oscillator; OSC1/OSC2 pins; external oscillator configuration selected by FPR<4:0> bits
010 = LPRC: Internal Low-Power RC
001 = FRC: Internal Fast RC
000 = LPOSC: Low-Power Crystal Oscillator; SOSCI/SOSCO pins

bit 7-4 **Unimplemented:** Read as '0'

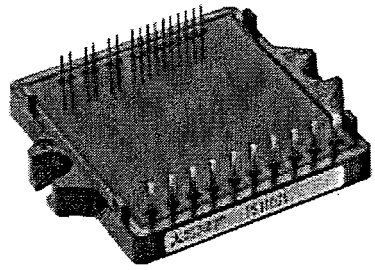
bit 3-0 **FPR<4:0>:** Oscillator Selection within Primary Group bits (see Table 7-4)

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'

ภาคผนวก ข.

Intelligent Power Module (IPM) เบอร์ PS11034

PS11034



INTEGRATED FUNCTIONS AND FEATURES

- Converter bridge for 3 phase AC-to-DC power conversion.
- 3 phase IGBT inverter bridge configured by the latest 3rd. generation IGBT and diode technology.
- Inverter output current capability Io (Note 1):

Type Name	Motor Rating	Io (100%)	Io (150%; 60sec)
PS11034	0.75 kW/200V AC	5.0Arms	7.5Arms

(Note 1) : The inverter output current is assumed to be sinusoidal and the peak current value of each of the above loading cases is defined as : $I_{OP} = Io \times \sqrt{2}$, Tc < 100°C

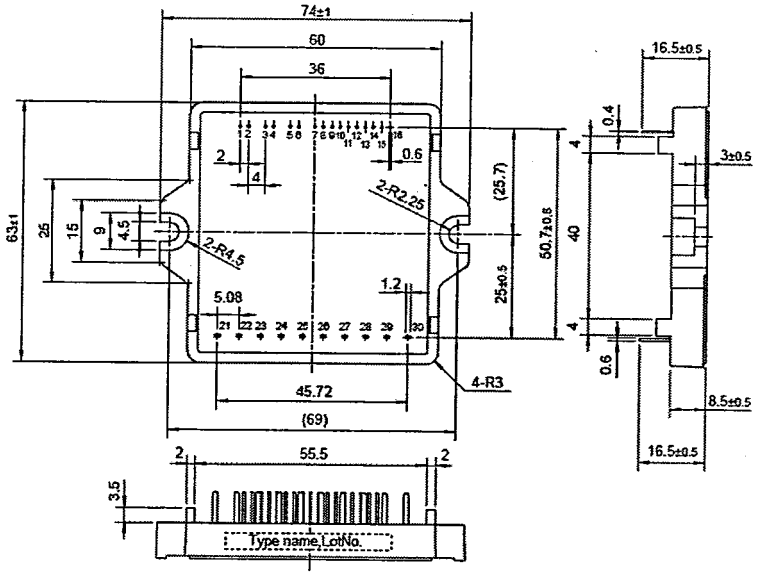
INTEGRATED DRIVE, PROTECTION AND SYSTEM CONTROL FUNCTIONS:

- P-Side IGBTs : Drive circuit, high-level-shift circuit, bootstrap circuit supply scheme for Single Control-Power-Source drive, and under voltage (UV) protection.
- N-Side IGBTs : Drive circuit, DC-Link current sense and amplifier circuits for overcurrent protection, control-supply under-voltage protection (UV), and fault output (Fo) signaling circuit.
- Fault Output : N-side IGBT short circuit (SC), over-current (OC), and control supply under-voltage (UV).
- Inverter Analog Current Sense : N-Side IGBT DC-Link Current Sense.
- Input Interface : 5V CMOS/TTL compatible, Schmitt Trigger input, and Arm-Shoot-Through interlock protective function.

APPLICATION

Acoustic noise-less 0.75kW/200V AC Class 3 phase inverters, motor control applications, and motors with built-in small size inverter package

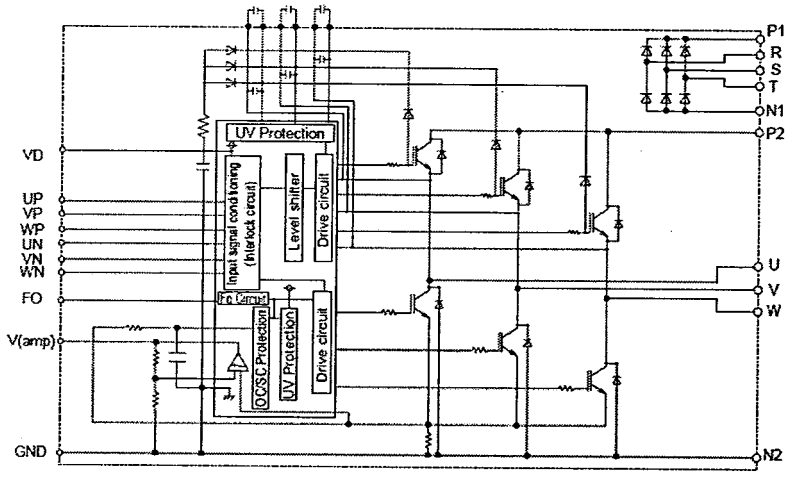
PACKAGE OUTLINES



Terminals Assignment :

1. CBU+	21. P1
2. CBU–	22. R
3. CBV+	23. S
4. CBV–	24. T
5. CBW+	25. N1
6. CBW–	26. P2
7. VD	27. U
8. UP	28. V
9. VP	29. W
10. WP	30. N2
11. UN	
12. VN	
13. WN	
14. FO	
15. Vamp	
16. GND	

INTERNAL FUNCTIONS BLOCK DIAGRAM



MAXIMUM RATINGS (T_j = 25°C)

INVERTER PART

Symbol	Item	Condition	Ratings	Unit
V_{CC}	Supply voltage	Applied between P2-N2	450	V
$V_{CC(surge)}$	Supply voltage (surge)	Applied between P2-N2, Surge-value	500	V
V_P or V_N	Each output IGBT collector-emitter static voltage	Applied between P2-U.V.W, U.V.W-N2	600	V
$V_{P(S)}$ or $V_{N(S)}$	Each output IGBT collector-emitter switching voltage	Applied between P2-U.V.W, U.V.W-N2	600	V
±Ic(±Icp)	Each output IGBT collector current	Tc = 25°C, "( )" means Ic peak value	±15 (±30)	A

CONVERTER PART

Symbol	Item	Condition	Ratings	Unit
V_{RRM}	Repetitive peak reverse voltage		800	V
Ea	Recommended AC input voltage		220	Vrms
Io	DC output current	3φ rectifying circuit	15	A
I_{FSM}	Surge (non-repetitive) forward current	1 cycle at 60Hz, peak value non-repetitive	150	A
I^2t	I^2t for fusing	Value for one cycle of surge current	93	A^2s

CONTROL PART

Symbol	Item	Ratings	Unit
V_D, V_{DB}	Supply voltage	−0.5 ~ 20	V
V_{CIN}	Input signal voltage	−0.5 ~ +7.5	V
V_{FO}	Fault output supply voltage	−0.5 ~ +7.5	V
I_{FO}	Fault output current	15	mA
I_{amp}	DC-Link IGBT current signal Amp output current	1	mA

TOTAL SYSTEM

Symbol	Item	Condition	Ratings	Unit
Tj	Junction temperature	(Note 2)	−20 ~ +125	°C
Tstg	Storage temperature	—	−40 ~ +125	°C
Tc	Module case operating temperature	(Fig. 3)	−20 ~ +100	°C
V_{ISO}	Isolation voltage	60 Hz sinusoidal AC applied between all terminals and the base plate for 1 minute.	2500	Vrms
—	Mounting torque	Mounting screw: M4	0.98 ~ 1.47	N·m

(Note 2) : The indicated values are specified considering the safe operation of all the parts within the ASIPM. The max. ratings for the ASIPM power chips (IGBT & FWDI) is Tj < 150.

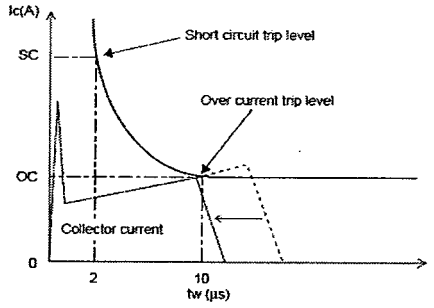
THERMAL RESISTANCE

Symbol	Item	Condition	Ratings			Unit
			Min.	Typ.	Max.	
Rth(jc)Q	Junction to case Thermal Resistance	Inverter IGBT (1/6)	—	—	2.8	°C/W
Rth(jc)F		Inverter FWDi (1/6)	—	—	3.9	°C/W
Rth(jc)FR		Converter Di (1/6)	—	—	4.8	°C/W
Rth(cf)	Contact Thermal Resistance	Case to fin thermal, grease applied (1 Module)	—	—	0.074	°C/W

ELECTRICAL CHARACTERISTICS (T_j = 25°C, V_D = 15V, V_{DB} = 15V unless otherwise noted)

Symbol	Item	Condition	Ratings			Unit
			Min.	Typ.	Max.	
$V_{CE(sat)}$	Collector-emitter saturation voltage	Tj = 25°C, Input = ON, Ic = 15A, V_D = V_{DB} = 15V (Shunt voltage drop not included)	—	—	2.9	V
V_{EC}	FWDi forward voltage	Tj = 25°C, −Ic = 15A	—	—	2.9	V
V_{FR}	Converter diode voltage	Tj = 25°C, I_{FR} = 10A	—	—	1.5	V
I_{RRM}	Converter diode reverse current	V_R = V_{RRM}, Tj = 125°C	—	—	8	mA
ton	Switching times	1/2 Bridge inductive, Input = 5V ↔ 0V Vcc = 300V, Ic = 15A, Tj = 125°C V_D = 15V, V_{DB} = 15V Note: ton, toff include delay time of the internal control circuit.	0.3	0.6	1.5	μs
tc(on)			—	0.5	1.0	μs
toff			—	1.6	2.5	μs
tc(off)			—	0.5	1.3	μs
trr	FWDi reverse recovery time		—	0.12	—	μs
Short circuit endurance (Output, Arm, and Load Short Circuit Modes)		@Vcc ≤ 400V, Input = 5V → 0V (One-Shot) −20°C ≤ Tj (start) ≤ 125°C, 13.5V ≤ V_D = V_{DB} ≤ 16.5V	• No destruction • Fo output by protection operation			
Switching SOA		@Vcc ≤ 400V, Input = 5V ↔ 0V, Tj ≤ 125°C Ic < OC trip level, 13.5V ≤ V_D = V_{DB} ≤ 16.5V	• No destruction • No protecting operation • No Fo output			

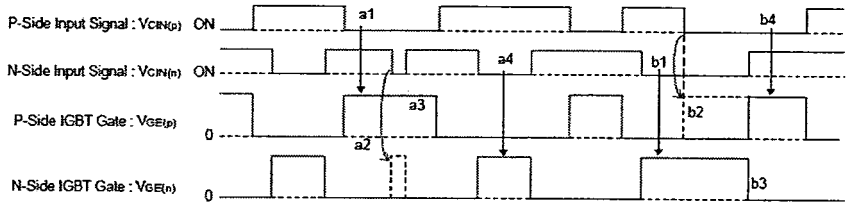
CURRENT ABNORMALITY PROTECTIVE FUNCTIONS



(Fig. 5)

Protection is achieved by monitoring and filtering the N-side DC-Bus current. The over-current protection is activated (after allowing a filtering time of 10 µs) when the line current reaches 250% of the rated load-current Io (rms). Similarly, the short circuit protection is activated (after allowing a filtering time of 2 µs) when the line current reaches twice the rated collector-current (Ic). When a current trip-level is exceeded (OC or SC), all the N-side IGBTs are intercepted (turned OFF) and a fault-signal is output. After the fault-signal output duration (1.8 ms - typ.), the interception is Reset at the following OFF input signal. However, since the fault may be repetitive, it is recommended to stop the system after the fault-signal is received and check the fault. The trip-level settings described above are summarized in the following figure:

ARM-SHOOT-THROUGH INTER-LOCK PROTECTIVE FUNCTION



(Fig. 6)

Description:

(1) During the ON-State of either of the upper-arm or the lower-arm IGBT, the inter-lock protection circuit blocks any erroneous ON pulses (resulting from input noise) from triggering the other arm IGBT and thus it prevents the arm-shoot-through situation.

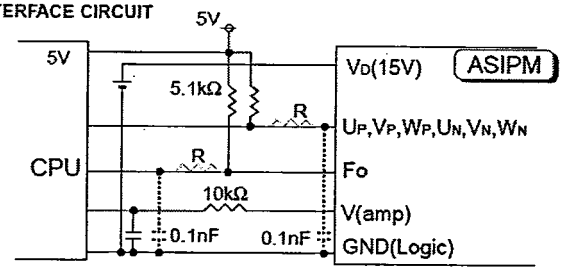
(2) When two ON-signals are received for both the upper and the lower arms, the signal received first will be passed to the IGBT and the second signal will be blocked. The second signal will be passed to its corresponding IGBT immediately after the first signal is OFF.

Note: This protective function provides no fault signaling output. The Dead-Time has to be set using the micro-controller (CPU).

Operation:

a1. P-side normal ON-signal ⇒ P-side IGBT gate turns ON.
a2. N-side erroneous ON-signal ⇒ N-side IGBT gate remains OFF.
a3. While P-side ON-signal remains ⇒ P-side IGBT gate remains ON.
a4. N-side normal ON-signal ⇒ N-side IGBT gate turns ON.
b1. N-side normal ON-signal ⇒ N-side IGBT gate turns ON.
b2. Simultaneous ON-signals ⇒ P-side IGBT gate remains OFF.
b3. N-side receives OFF-signal ⇒ N-side IGBT gate turns OFF.
b4. Immediately after (b3) ⇒ P-side IGBT gate turns ON.

RECOMMENDED I/O INTERFACE CIRCUIT



(Fig. 7)

ภาคผนวก ค.
ชุดคำสั่งภาษาซี (Software)

Main.c

```
#include"p30f4011.h"
#include"Var.c"
#include"Def.c"
#include"Const.c"
#include"init.c"
#include"function.c"
#include"Interrupt.c"

int main(void){
        Nop();
        InitPort();
        timer2_init();
        timer1_init();
        InitMCPWM();

        if(relay_delay=3600){RELAY = 1;}   // relay_delay=2s


        while(1){

                if(tmr1.istask0){
                        scan_segment();
                        tmr1.istask0=0;
                }
                if(tmr1.istask1){
                        SCANKEY();
                        key_event();
                        tmr1.istask1=0;
                 }
                Nop();
        };
        return 0;
}
```

Const.c

```
// file def: constant variable

#define FCY 14745600                                    // Fosc/ 4
#define FPWM 15000
#define DEADTIME  (unsigned int)(0.000003*FCY)          //~3 us

unsigned int sine[61] = {
0    , 175,  349,  523,  698,  872,  1045,  1219,  1392,  1564, \
1736,  1908,  2079,  2250,  2419,  2588,  2756,  2924,  3090,  3256, \
3420,  3584,  3746,  3907,  4067,  4226,  4384,  4540,  4695,  4848, \
5000,  5150,  5299,  5446,  5592,  5736,  5878,  6018,  6157,  6293, \
6428,  6561,  6691,  6820,  6947,  7071,  7193,  7314,  7431,  7547, \
7660,  7771,  7880,  7986,  8090,  8192,  8290,  8387,  8480,  8572, \
8660
};      // the value of sine function factor by 10000

unsigned int cose[61] ={
10000, 9998,  9994,  9986,  9976, 9962, 9945, 9925, 9903, 9877,\
9848,  9816,  9781,  9744,  9703, 9659, 9613, 9563, 9511, 9455,\
9397,  9336,  9272,  9205,  9135, 9063, 8988, 8910, 8829, 8746,\
8660,  8572,  8480,  8387,  8290, 8192, 8090, 7986, 7880, 7771,\
7660,  7547,  7431,  7314,  7193, 7071, 6947, 6820, 6691, 6561,\
6428,  6293,  6157,  6018,  5878, 5736, 5592, 5446, 5299, 5150,\
5000
};      // the value of sine function factor by 10000

unsigned int Freq[60] = {
5120, 2560, 1707, 1280, 1024, 853,  731,  641,  569,  512,\
465,  427,  394,  366,  341,  320,  301,  284,  269,  256,\
244,  232,  223,  213,  205,  197,  190,  183,  177,  171,\
165,  160,  155,  150,  146,  142,  138,  135,  131,  128,\
125,  122,  119,  116,  114,  111,  109,  107,  104,  102,\
100,  98,   97,   95,   93,   91,   90,   88,   87,   85
};
```

Def.c

```
// File des. :  definition pin layout

#define SHIFT_CLK   _RB4
#define SER_DATA    _RB2
#define LATCH_CLK  _RB3
#define COM0 _RD1
#define COM1 _RE8
#define COM2 _RC14
#define COM3 _RB5
#define RELAY _RF5

#define U PDC3
#define V PDC2
#define W PDC1

#define _T2ON       T2CONbits.TON
#define _T1ON    T1CONbits.TON
struct{
        unsigned SW0 :2;
        unsigned SW1 :1;
        unsigned SW2 :2;
        unsigned SW3 :2;
        unsigned SW_status0 :1;
        unsigned SW_status1 :1;
        unsigned SW_status2 :1;
        unsigned SW_status3 :1;
        unsigned message0 :1;
        unsigned message1 :1;
        unsigned message2 :1;
        unsigned message3 :1;
}SW;
#define SCANKEY()\
{\
        if((_RF6==0)&(SW.SW_status0==0))\
                {\
                 SW.SW0++;\
                 SW.SW_status0++;\
                 SW.message0 = 1;\
                }\
        if((_RD0==0)&(SW.SW_status1==0))\
                {\
                 SW.SW1++;\
                 SW.SW_status1++;\
                 SW.message1 = 1;\
                }\
```

Def.c

```
        if((_RD2==0)&(SW.SW_status2==0))\
                {\
                 SW.SW2++;\
                 SW.SW_status2++;\
                 SW.message2 = 1;\
                }\
        if((_RD3==0)&(SW.SW_status3==0))\
                {\
                 SW.SW3++;\
                 SW.SW_status3++;\
                 SW.message3 = 1;\
                }\
        if(_RF6==1)\
                {\
                 SW.SW_status0=0;\
                }\
        if(_RD0==1)\
                {\
                 SW.SW_status1=0;\
                }\
        if(_RD2==1)\
                {\
                 SW.SW_status2=0;\
                }\
        if(_RD3==1)\
                {\
                 SW.SW_status3=0;\
                }\
}
unsigned char num;
#define numX(a)\
{\
                if(a==1)\
                {\
                        num = 0xF9;\
                }\
                else if(a==2)\
                {\
                        num= 0xA4;\
                }\
                else if(a==3)\
                {\
                        num= 0xB0;\
                }\
```

Def.c

```
                else if(a==4)\
                {\
                        num = 0x99;\
                }\
                else if(a==5)\
                {\
                        num = 0x92;\
                }\
                else if(a==6)\
                {\
                        num= 0x82;\
                }\
                else if(a==7)\
                {\
                        num = 0xF8;\
                }\
                else if(a==8)\
                {\
                        num= 0x80;\
                }\
                else if(a==9)\
                {\
                        num = 0x90;\
                }\
                else if(a==0)\
                {\
                        num = 0xC0;\
                }\
                else{}\
}

#define loadsegment(a,b,c,d)\
{\
        segment[0] = a;\
        segment[1] = b;\
        segment[2] = c;\
        segment[3] = d;\
}
```

Function.c.

```
void data_shift(unsigned char *dptr){
        unsigned char i;
        for(i=0x80;i>0;i=i/2){
                SER_DATA=(i&*dptr) ? 1 : 0;          // serial data
                SHIFT_CLK=1;                         // shift clock high
                SHIFT_CLK=0;
        }
}
void latch_595(unsigned char *dptr){
        LATCH_CLK=0;
        SHIFT_CLK=0;
        data_shift(dptr);
        LATCH_CLK = 1;
        LATCH_CLK = 0;
}

void scan_segment(void){
        COM0=1;
        COM1=1;
        COM2=1;
        COM3=1;
        latch_595(&segment[count.counter2bit]);
        switch(count.counter2bit){
                case 0: COM0=0;  break;
                case 1: COM1=0;  break;
                case 2: COM2=0;  break;
                case 3: COM3=0;  break;
        }
    count.counter2bit++;
}

unsigned char Xnum(unsigned char a){
        numX(a);
        return num;
}
```

Function.c

```
void key_event(void){
        if(SW.message0){                        // key down
                Temp--;
                if(Temp>=0){
                        PR2=Freq[Temp];
                        segment[0] = 0xFF;
                        segment[1] = 0xFF;
                        segment[2] = Xnum((Temp+1)/10);
                        segment[3] = Xnum((Temp+1)%10);
                }
                SW.message0 = 0;
        }

        if(SW.message1){                        // run/stop
                if(SW.SW1==0){
                        loadsegment( 0x92,0x87,0xA3,0x8C);
                        PTCONbits.PTEN  = 0;}
                else{loadsegment( 0xAF,0xE3,0xAB,0xFF);
                        PTCONbits.PTEN = 1;}
                        SW.message1 = 0;
        }

        if(SW.message2){
                Temp++;                         // key up
                if(Temp<61){
                        PR2=Freq[Temp];
                        segment[0] = 0xFF;
                        segment[1] = 0xFF;
                        segment[2] = Xnum((Temp+1)/10);
                        segment[3] = Xnum((Temp+1)%10);
                }
                SW.message2=0;
        }

        if(SW.message3){                        // increase Torque
                Vref=Vref+10;                   // increase step by = 10%
                if(Vref>100){Vref=100;}
                Kb=(Vref/100)/8660.2540;        // Kb=Vref/(sine60*10000)
                Ka=(Temp_TB*0.5)/(Vref/100);    // Ka=(TB*cos60)/Vref
                SW.message3=0;
        }
}
```

Init.c

```
// init system file

_FOSC(CSW_FSCM_OFF & XT_PLL8);
_FWDT(WDT_OFF);
_FBORPOR(PBOR_ON & BORV_27 & PWRT_OFF & MCLR_DIS);

void timer1_init(void){                    // MULTI TASKING LOOP
        TMR1=0;
        PR1=0x0020;
        T1CON=0x0030;
        _T1IP=5;
        _T1IE=1;
        _T1ON=1;
}

void timer2_init(void){
        TMR2=0;
        PR2= 512;
        T2CON=0x0010;
        _T2IP=6;
        _T2IE=1;
        _T2ON=1;
}

void InitPort(){
         ADPCFG=0xFFFF;
         TRISE = 0x0000;
         TRISB = 0xFF00;
         TRISF = 0x0040;
         TRISD = 0x000D;
         _TRISC14 = 0;

}
```

Init.c

```
void InitMCPWM(void){
        PTPER = (FCY/FPWM - 1) >> 1;
        OVDCON = 0x0000;                    // Disable all PWM outputs.
        DTCON1 = DEADTIME;                  // ~3 us of dead
        PWMCON1 = 0x0077;                   // Enable PWM output pins and enable
                                            // complementary mode
        PDC1 = 0;                           // 0 Volts on Phase A.
        PDC2 = 0;                           // 0 Volts on Phase B.
        PDC3 = 0;                           // 0 Volts on Phase C.
        IFS2bits.PWMIF = 0;                 // Clear PWM Interrupt flag
        IEC2bits.PWMIE = 1;                 // Enable PWM Interrupts
        OVDCON = 0x3F00;                    // PWM outputs are controller by
                                            // PWM module
        PTCONbits.PTMOD = 3;                // Center aligned PWM operation
        PTCONbits.PTEN = 1;                 // Start PWM
}
```

Interrupt.c

```
void sector0(void);
void sector1(void);
void sector2(void);
void sector3(void);
void sector4(void);
void sector5(void);
void (*PDC_UPDATE[])(void) = {          // PDC1 = W , PDC2 = V , PDC3 = U
        sector0,
        sector1,
        sector2,
        sector3,
        sector4,
        sector5
};
void sector0(void){                                 // U  , U+V

        U = TA;         V = TB;         W = T0;
}
void sector1(void){                                 // U+V , V

        V = TA;         U = TB;         W = T0;
}
void sector2(void){                                 // V, V+W

        U = T0;         V = TA;         W = TB;
}
void sector3(void){                                 // V+W , W

        U = T0;         V = TB;         W = TA;
}
void sector4(void){                                 // W ,W+U

        V = T0;         W = TA;         U = TB;
}
void sector5(void){                                 // W+U , U

        V =T0;          W = TB;         U = TA;
}

void _ISR _PWMInterrupt(void){

        IFS2bits.PWMIF = 0;
}
```

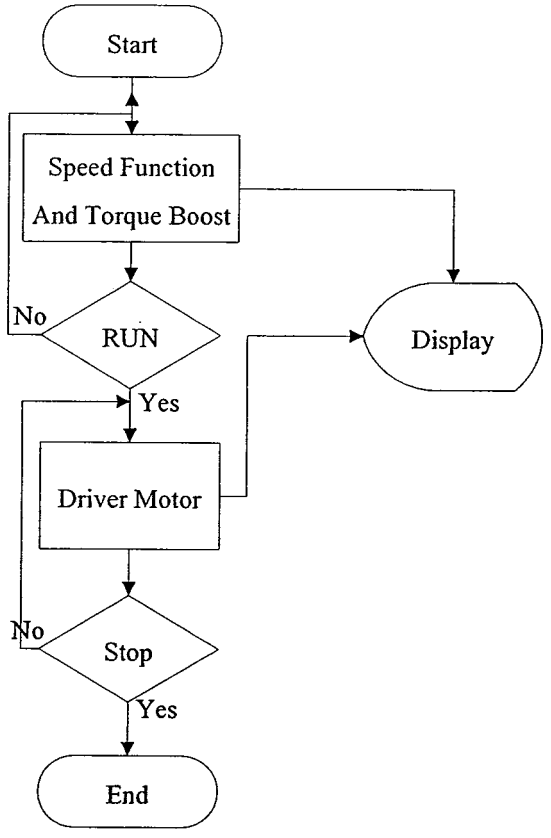
Interrupt.c

```
void ABCT0_CALCULATE(void){
        T = Kb*sine[counter];
        TB = (int)(T*490);
        Temp_TB =  (int)(T*490);

        T = (Vref/100)*(cose[counter]-Ka);
        TA = (int)(T*490);

        T0 =    93;                     // T0 from define
}

void _ISR _T2Interrupt(void){
        _T2IF= 0;
        ABCT0_CALCULATE();
        PDC_UPDATE[sector]();
        if(counter>60){                 // counter for update angle 0- 60 deg
                counter = 0;
                if(sector>6){           // sector update for 1- 6
                        sector = 0;
                }
                sector++;
        }
        counter++;
}

void _ISR _T1Interrupt(void){           // timing  base
        _T1IF = 0;
        relay_delay++;
        if(tmr1.mtmcount==0){
                tmr1.istask0=1;
        }
        else if(tmr1.mtmcount==1){
                if(tmr1.ctask1==0){
                        tmr1.istask1=1;
                }
                tmr1.ctask1++;
        }
        else
                Nop();
                tmr1.mtmcount++;
}
```

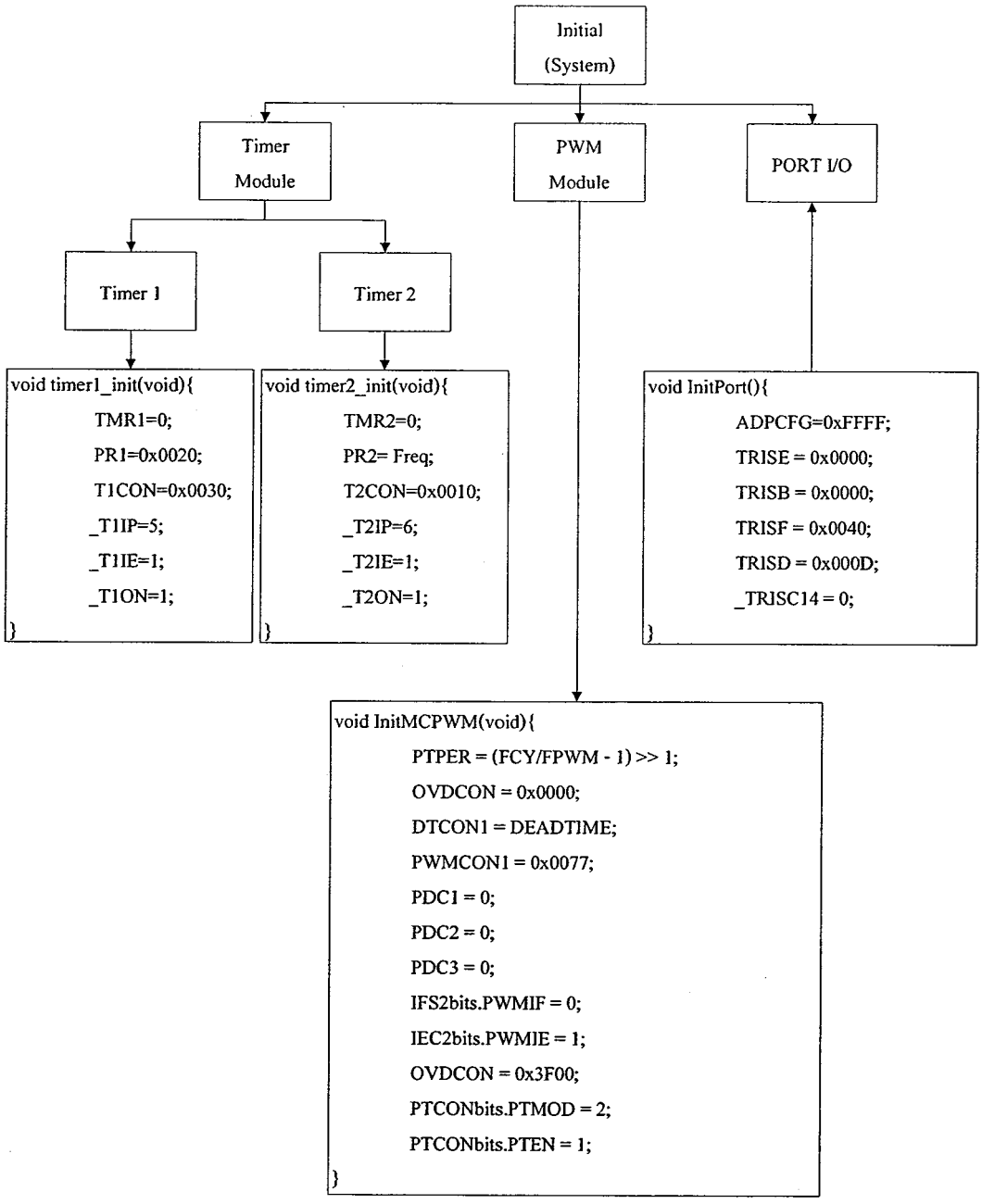
Var.c

```
// file def: Variable declaration

//------ variable for  update space vector
unsigned char   counter;
unsigned char   sector;
unsigned char   Vref=30;
double T;
unsigned int TA,TB,TC,T0,Temp_TB;
double Ka,Kb;
unsigned int relay_delay;
unsigned char Temp=9;

struct{
        unsigned index    :2;
}quadrent;

// SCAN KEY  PARAMETER AND DISPLAY
struct{
        unsigned counter2bit :2;
}count;

unsigned char segment[4];

// MULTI TASKING TASKING PARAMETER
struct{
        unsigned mtmcount  :2;
        unsigned istask0       :1;
        unsigned istask1       :1;
        unsigned istask2       :1;
        unsigned istask3       :1;
        unsigned ctask1        :1;
        unsigned ctask2        :2;
        unsigned ctask3        :3;
}tmr1;
```

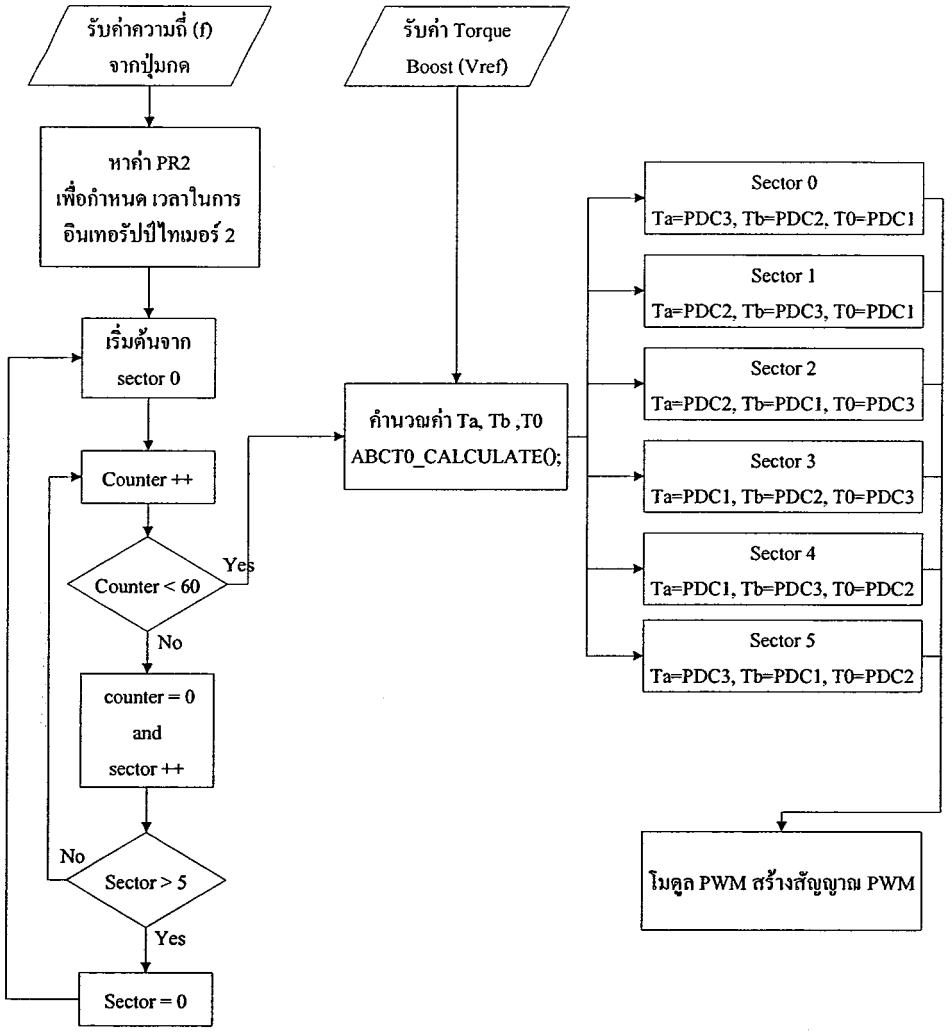
ภาคผนวก ง.
โฟร์ชาร์ตการทำงานของโปรแกรม



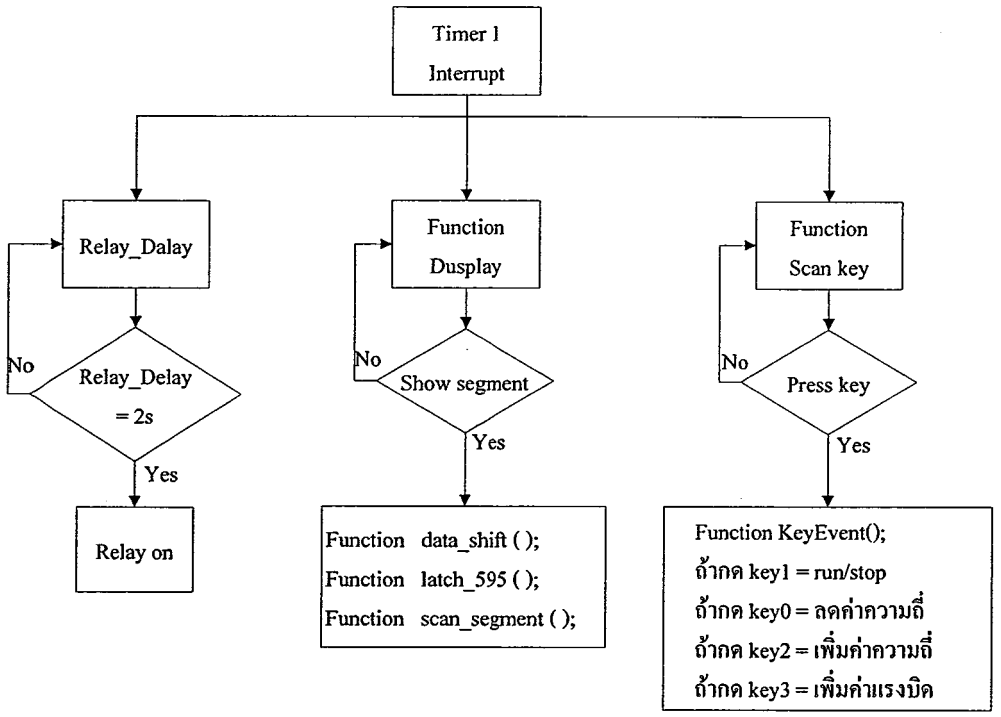
โฟร์ชาร์ตการทำงานของโปรแกรมหลัก



โฟร์ชาร์ตการกำหนดค่าเริ่มต้นให้โปรแกรม



โฟร์ชาร์ตการคำนวณค่า Ta, Tb, และ T0 ของแต่ละเซกเตอร์



โฟร์ชาร์ตการทำงานของระบบรวม โดยใช้ฐานเวลาของการอินเทอร์รัปไทเมอร์ 1