การควบคุมมอเตอร์กระแสตรงไร้แปรงถ่านในย่านกำลังไฟฟ้าคงที่ด้วยหลักการชดเชยมุมเฟสก้าวหน้าด้วยการประยุกต์ใช้บอร์ดประมวลผลสัญญาณดิจิตอล dsPIC30F2010

THE CONSTANT POWER MODE CONTROL OF BRUSTLESS DC MOTOR BY PHASE ADVANCED ANGLE COMMUTATION TECHNIQUE USING dsPIC30F2010 BOARD

สำเริง เต็มราม

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาวิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า
คณะวิศวกรรมศาสตร์
มหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรี
ปีการศึกษา 2556

ลิขสิทธิ์ของมหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรี

การควบคุมมอเตอร์กระแสตรงไร้แปรงถ่านในย่านกำลังไฟฟ้าคงที่ด้วยหลักการชดเชยมุมเฟสก้าวหน้าด้วยการประยุกต์ใช้บอร์ดประมวลผลสัญญาณดิจิตอล dsPIC30F2010

สำเริง เต็มราม

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาวิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า
คณะวิศวกรรมศาสตร์
มหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรี
ปีการศึกษา 2556

หัวข้อวิทยานิพนธ์	การควบคุมมอเตอร์กระแสตรงไร้แปรงถ่านในย่านกำลังไฟฟ้าคงที่ด้วยหลักการชดเชยมุมเฟสก้าวหน้าด้วยการประยุกต์ใช้บอร์ดประมวลผลสัญญาณดิจิตอล dsPIC30F2010
ชื่อ - นามสกุล	นายสำเริง เต็มราม
สาขาวิชา	วิศวกรรมไฟฟ้า
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์วันชัย ทรัพย์สิงห์, Ph.D.
ปีการศึกษา	2556

บทคัดย่อ

งานวิจัยนี้มีวัตถุประสงค์ เพื่อควบคุมการขับเคลื่อนมอเตอร์กระแสตรงไร้แปรงถ่านซึ่งเป็นมอเตอร์ความเร็วสูงที่ใช้กันอย่างแพร่หลายในงานอุตสาหกรรม ในส่วนขยายความเร็วและแรงบิดในย่านการทำงานกำลังไฟฟ้าคงที่ด้วยหลักการชดเชยมุมเฟสก้าวหน้าระหว่างกระแสขาเข้ากับแรงดันต้านกลับของมอเตอร์ โดยศึกษาผลการตอบสนองของด้านความเร็วและแรงบิดมอเตอร์ขณะมีการชดเชยมุมเฟสที่ค่าต่างๆ เพื่อเป็นแนวทางในการออกแบบชุดควบคุมการขับเคลื่อนมอเตอร์กระแสตรงไร้แปรงถ่านให้ได้ประสิทธิภาพสูงสุด

งานวิจัยแบ่งออกเป็นสองส่วนคือ ในส่วนแรกเป็นการสร้างแบบจำลองของมอเตอร์กระแสตรงไร้แปรงถ่านด้วยโปรแกรม MATLAB/Simulink เพื่อศึกษาพฤติกรรมความเร็วและแรงบิดของมอเตอร์ทั้งขณะไม่มีการชดเชยมุมเฟส และขณะมีการชดเชยมุมเฟสก้าวหน้าที่ค่าต่างๆ ส่วนที่สองเป็นการประยุกต์ใช้บอร์ดประมวลผลสัญญาณดิจิตอลรุ่น dsPIC30F2010 ในการควบคุมการขับเคลื่อนมอเตอร์กระแสตรงไร้แปรงถ่านขนาดพิกัดกำลังไฟฟ้า 500 วัตต์ แรงดัน 48 VDC พร้อมทั้งแสดงผลการทดสอบโดยใช้โปรแกรม Graphic User Interface เพื่อนำผลที่ได้จากชุดทดสอบเชิงปฏิบัติเปรียบเทียบกับผลที่ได้กับผลที่ได้จากแบบจำลอง

ผลที่ได้จากชุดทดสอบการควบคุมการขับเคลื่อนมอเตอร์ไฟฟ้ากระแสตรงไร้แปรงถ่านเชิงปฏิบัติพบว่า เมื่อโหลดมีพิกัดที่ 0.28 นิวตัน-เมตร และมอเตอร์จะความเร็วที่ 300 รอบ/นาที เมื่อไม่มีการชดเชยมุมเฟส และเมื่อมีการปรับค่ามุมเฟสก้าวหน้าเป็น 5°, 10° และ 15° มอเตอร์จะมีความเร็วที่ 412, 442 และ 451 รอบ/นาที ตามลำดับ ซึ่งจะเห็นได้ว่ามอเตอร์กระแสตรงไร้แปรงถ่านสามารถเพิ่มค่าความเร็วขึ้นได้ด้วยหลักการชดเชยมุมเฟสก้าวหน้าในย่านกำลังไฟฟ้าคงที่

คำสำคัญ : การชดเชยมุมเฟสก้าวหน้า มอเตอร์ไฟฟ้ากระแสตรงไร้แปรงถ่าน

Thesis Title	The Constant Power Mode Control of Brushless DC Motor by Phase Advanced Angle Commutation Technique using dsPIC30F2010 Board
Name – Surname	Mr. Samraeng Temram
Program	Electrical Engineering
Thesis Advisor	Assistant Professor Wanchai Subsingha, Ph.D.
Academic Year	2013

ABSTRACT

This research is purposed to control a Brushless DC (BLDC) motor, which is a high speed motor and widely uses in various industrial applications. It concentrates in increasing motor speed and torque in constant power operating using a Phase Advanced Angle Compensation (PAAC) technique. This PAAC technique is to control phase angle between input current and back EMF of such motor. This leads into the studying of the motor speed response and its toque in various values of the phase advanced angle in order to design a BLDC drive control for a better performance.

This thesis is separated into two steps. Firstly, a BLDC motor is modeled and simulated using MATLAB/Simulink program in order to investigate a motor speed and its torque both with and without the compensation of such phase angle in PAAC technique. Secondly, a control of BLDC motor of 500 Watt, 48 VDC power rating is built using the PAAC technique. However, the control is implemented by dsPIC30F2010 DSP board and Graphic User Interfacing program in order to display the experimental results.

From the experimental results of the BLDC motor control, it shows that when the motor load is at 0.28 N-m motor is at 300 rpm. when the control has no compensation of such PAAC angle. And at the same motor load, when the phase advanced angle is at 5°, 10° and 15°, motor speed is about 412, 442 and 451 rpm., respectively. It means that the BLDC motor control that using PAAC technique can raise the motor speed in a constant power operating mode.

Keywords: phase advance angle commutation, brushless DC motor

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงลงไปด้วยความช่วยเหลือเป็นอย่างดีจากผู้ช่วยศาสตราจารย์ ดร.วันชัย ทรัพย์สิงห์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ดร.สุรินทร์ แหงมงาม ดร.ณฐภัทร พันธ์คง กรรมการสอบ และรองศาสตราจารย์เสถียร ธัญญศรีรัตน์ ผู้ทรงคุณวุฒิ ที่กรุณาให้คำแนะนำและให้คำปรึกษาตลอดจนให้ความช่วยเหลือแก้ไขข้อบกพร่องต่างๆ เพื่อให้วิทยานิพนธ์ฉบับนี้มีความสมบูรณ์ ผู้วิจัยขอขอบคุณเป็นอย่างสูงไว้ ณ โอกาสนี้

ขอขอบคุณคณาจารย์ทุกท่านของมหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรี คณะวิศวกรรมศาสตร์ สาขาวิชาวิศวกรรมศาสตร์อุตสาหการ ทุกท่านที่ได้อบรมสั่งสอนและประสิทธิ์ประสาทวิชาต่างๆให้ตลอดระยะเวลาที่ได้ศึกษา

สำเริง เต็มราม

สารบัญ

หน้า

สารบัญ (ต่อ)

สารบัญตาราง

ตารางที่ หน้า

สารบัญตาราง (ต่อ)

ตารางที่ หน้า

สารบัญรูป

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

รูปที่ หน้า

สารบัญรูป (ต่อ)

รูปที่ หน้า

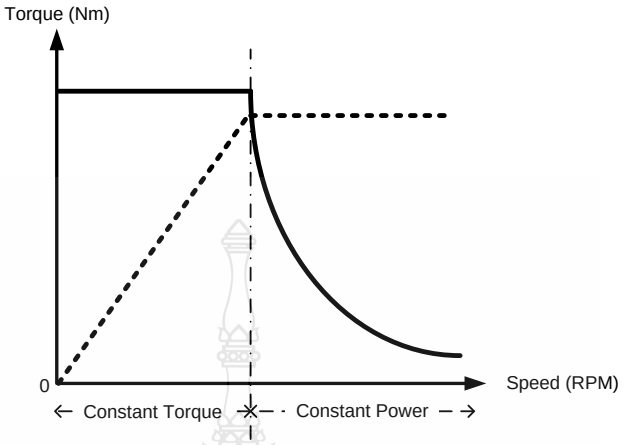
บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ปัจจุบันการพัฒนาเทคโนโลยีชุดควบคุมการขับเคลื่อนเครื่องกลไฟฟ้า ให้ได้ประสิทธิภาพสูงสุดนับได้ว่ามีความจำเป็นอย่างสูงในภาวะที่โลกผจญกับวิกฤติพลังงาน ในภาคอุตสาหกรรมการพยายามลดต้นทุนด้านพลังงานในการผลิตสินค้าถือเป็นความจำเป็นอย่างยิ่ง ทั้งนี้เครื่องกลไฟฟ้าเป็นส่วนหนึ่งของเครื่องจักรในการผลิตในกระบวนการผลิต ดังนั้นการเลือกใช้เครื่องกลไฟฟ้าที่มีประสิทธิภาพสูงเป็นปัจจัยที่สำคัญในการลดต้นทุนในการผลิตเช่นกัน มอเตอร์ไฟฟ้ากระแสตรงแบบไร้แปรงถ่าน (Brushless DC Motor) เป็นที่นิยมในปัจจุบัน เนื่องจากมีคุณลักษณะข้อดีหลายประการเมื่อเทียบกับการใช้มอเตอร์ไฟฟ้าชนิดอื่น เช่น ประสิทธิภาพการใช้งานซึ่งจะให้ค่าอัตราแรงบิด/ขนาดพิกัดมอเตอร์ (Torque/Size) ค่อนข้างสูง และจากการรบกวนสนามแม่เหล็ก (Electromagnetic emissions) ขณะใช้งานก็มีค่าต่ำ การสนองตอบต่อการเปลี่ยนแปลงความเร็วค่อนข้างดี มีอายุการใช้งานยาวนาน และไม่เกิดประกายไฟขณะทำงานจากการ Commutate ทางกล

อย่างไรก็ตาม การขับเคลื่อนในรถไฟฟ้าปัจจุบันได้ให้ความสนใจต่อการประยุกต์ใช้มอเตอร์ไฟฟ้ากระแสตรงแบบไร้แปรงถ่าน (Brushless DC Motor) หรือ BLDC Motor กันมากขึ้น รวมไปถึงระบบขับเคลื่อนทางไฟฟ้าในรถยนต์ที่ทำงานร่วมกับเครื่องยนต์แบบสันดาปด้วยเช่นกัน การออกแบบวิธีการขับเคลื่อน BLDC Motor ต้องอาศัยเทคนิคการควบคุมกระบวนการการไหลของกระแสไฟฟ้าเพื่อควบคุมการหมุน (Electronic Commutation) ของมอเตอร์ ซึ่งเป็นการควบคุมการสร้างสนามแม่เหล็กหมุนในขดลวดสเตเตอร์ (Stator) ให้สัมพันธ์กับตำแหน่งในส่วนหมุน (Rotor) โดยการตรวจจับตำแหน่งในขดลวด Rotor หลักการตรวจจับตำแหน่งของ Rotor ในปัจจุบันมี 2 แบบคือ การใช้ Hall Sensor ในการตรวจจับการเปลี่ยนแปลงของสนามแม่เหล็ก และการตรวจจับจุดกำเนิดของแรงดันไฟฟ้าต้านกลับ (Back EMF) ในชุดขดลวด Stator ด้วยหลักการ Zero Crossing detect จุดสำคัญในการออกแบบ Electronic Commutation ใน BLDC Motor นั่นคือจะต้องครอบคลุมการทำงานของมอเตอร์ในทุกย่านการทำงานได้อย่างมีประสิทธิภาพ ทั้งนี้ คุณสมบัติย่านการทำงานของ BLDC Motor จะมี 2 ย่านการทำงาน [1] [2] ซึ่งเหมือนกันกับคุณสมบัติย่านการทำงานของมอเตอร์ไฟฟ้ากระแสตรงแบบใช้แปรงถ่าน (Brush DC Motor) ทั่วไปทุกประการ คือ ประกอบด้วย ย่านการ

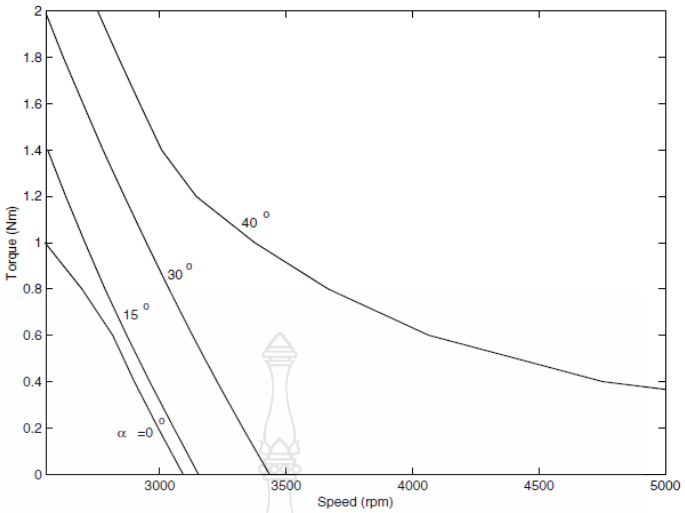
ทำงานแบบแรงบิดคงที่ (Constant Torque) และย่านการทำงานแบบกำลังไฟฟ้าคงที่ (Constant Power) ดังแสดงในรูปที่ 1.1



รูปที่ 1.1 คุณลักษณะของความเร็ว-แรงบิดในทั้งสองย่านการทำงานของ BLDC Motor

ในย่านการทำงานแบบแรงบิดคงที่ ความเร็วของมอเตอร์จะมีค่าไม่เกินกว่าความเร็วพิกัด (Rated Speed) ของมอเตอร์นั้น ซึ่งค่าแรงบิดของมอเตอร์จะแปรผันโดยตรงกับกระแสที่ไหลในชุดขดลวดสเตเตอร์ เนื่องจากสนามแม่เหล็กที่เกิดขึ้นในส่วนตัวหมุนของมอเตอร์จะมีค่าคงที่ อย่างไรก็ตาม เนาองจากส่วนตัวหมุน (Rotor) ของ BLDC Motor เป็นแม่เหล็กถาวรและมีความหนาแน่นของสนามแม่เหล็กคงที่ตลอดเวลาใช้งาน จะทำให้เป็นปัญหายุ่งยากในการควบคุมมอเตอร์ขณะทำงานสูงกว่าความเร็วพิกัดในย่านการทำงานแบบกำลังไฟฟ้าคงที่ (Constant Power)

การแก้ปัญหาของการทำงาน BLDC Motor ในย่านทำงานแบบกำลังไฟฟ้าคงที่นั้น สามารถทำได้โดยการควบคุมกระแสสเตเตอร์ในแต่ละเฟสให้เกิดขึ้นก่อนแรงดันต้านกลับ (Back EMF) ในแต่ละเฟสของมอเตอร์นั้น ทั้งนี้ความสัมพันธ์ระหว่างค่าแรงบิดค่ากับความเร็วมอเตอร์ในย่านการทำงานแบบกำลังคงที่จะไม่เป็นเชิงเส้น ดังแสดงในรูปที่ 1.1 ดังนั้นการหาค่ามุมเฟสก้าวหน้า(Advanced Phase Angle) ที่เหมาะสม เพื่อนำไปใช้ในการนำกระแสล่วงหน้าของกระแสสเตเตอร์ในแต่ละเฟสในแต่ละย่านความเร็ว ดังแสดงในรูปที่ 1.2 จึงมีความสำคัญต่อการนำไปประยุกต์ใช้ควบคุมการทำงานของ BLDC Motor ในย่านกำลังคงที่



รูปที่ 1.2 ความสัมพันธ์ระหว่างค่าแรงบิดกับความเร็วที่ค่ามุมเฟสก้าวหน้าต่างๆ [1]

จากการสืบค้นงานวิจัยที่เกี่ยวข้องกับหลักการหาค่ามุมเฟสก้าวหน้า(Advanced Phase Angle) และการประยุกต์ใช้ในย่านการทำงานแบบกำลังคงที่ของ BLDC Motor นั้นที่สำคัญมีอยู่ 2 วิธี

วิธีแรก [3] เป็นการประยุกต์ใช้วงจรที่สร้างขึ้นสำเร็จรูปเพื่อใช้ปรับให้สัญญาณขาออกของวงจรมีเฟสนำหน้าสัญญาณขาเข้าของวงจรได้ตามที่กำหนดในวงจรนั้นๆ ซึ่งจากการประยุกต์ใช้จะทำให้กระแสสเตเตอร์ของมอเตอร์มีค่ามุมเฟสนำหน้าแรงดันต้านกลับ (Back EMF) ของมอเตอร์ในแต่ละเฟสได้ตามต้องการ ซึ่งจะเห็นว่าการใช้วงจรสำเร็จรูปดังกล่าวมีข้อดีที่สามารถประยุกต์ได้โดยง่ายไม่ซับซ้อนแต่ผลตอบสนองแรงบิดที่ได้มีความคลาดคลื่นจากค่าที่ต้องการ เนื่องจากในย่านการทำงานแบบกำลังคงที่นี้ความสัมพันธ์ระหว่างแรงบิดกับความเร็วไม่เป็นเชิงเส้น

วิธีที่สอง [4] เป็นการใช้ตัวตรวจจับระยะทางซึ่งในงานวิจัยนี้จะมีลักษณะเป็นวงกลม โดยระบบควบคุมจะเป็นแบบที่สามารถตั้งโปรแกรมการทำงานได้ (Programmable Rotary Encoder)เพื่อสร้างสัญญาณอ้างอิงตำแหน่งการหมุนของมอเตอร์ และนำไปใช้เป็นตัวกำหนดมุมเฟสล่วงหน้า (Advanced Phase Angle) ของกระแสสเตเตอร์ตามที่ได้กล่าวมาข้างต้น ซึ่งข้อดีของวิธีนี้ก็คือลดขั้นตอนการคำนวณของตัวประเมินผล แต่มีข้อเสียคือไม่สามารถเปลี่ยนแปลงมุมเฟสก้าวหน้าในขณะที่มอเตอร์กำลังทำงานได้ จึงไม่มีความยืดหยุ่นพอในการนำไปใช้งานเชิงปฏิบัติ

จากงานวิจัยที่ได้กล่าวมาแล้ว ผู้วิจัยจึงมีแนวคิดในการแก้ปัญหาโดยการประยุกต์ใช้บอร์ดไมโครคอนโทรลเลอร์ในการสร้างตัวควบคุมการชดเชยมุมเฟสก้าวหน้า (Advanced Phase Angle) โดยสามารถกำหนดค่ามุมเฟสก้าวหน้าที่เปลี่ยนแปลงไปตามค่าความเร็วจริงขณะที่มอเตอร์ทำงานอยู่

โดยมีจุดประสงค์หลักคือ เพื่อศึกษาการทำงานของมอเตอร์ในย่านกำลังคงที่ในขณะที่มีค่าการชดเชยมุมเฟสก้าวหน้าที่ค่าต่างๆ

1.2 ความสำคัญของปัญหา

เนื่องจากความสัมพันธ์ระหว่างความเร็ว และแรงบิด ของมอเตอร์ไฟฟ้ากระแสตรง แบบไร้แปรงถ่าน (BLDC Motor) ในย่านการทำงานแบบกำลังคงที่ (Constant Power Mode) ไม่เป็นเชิงเส้น ดังแสดงในรูปที่ 1.1 ตลอดจนค่ามุมที่ได้จากการตรวจจับจาก Hall Sensor ในระบบควบคุมมีสัญญาณรบกวน จึงทำให้การพัฒนาโปรแกรมเพื่อควบคุมการทำงานในบอร์ดไมโครคอนโทรลเลอร์ เพื่อใช้ในการหาค่าชดเชยมุมเฟสก้าวหน้า (Advanced Phase Commutation) นั้นอาจมีความยุ่งยากพอสมควร และอาจเป็นผลให้ผลทดสอบในย่านทำงานแบบกำลังคงที่ของมอเตอร์เชิงปฏิบัติ จากชุดทดสอบการทำงานของ BLDC Motor โดยการประยุกต์ใช้บอร์ดไมโครคอนโทรลเลอร์ของวิทยานิพนธ์นี้ได้ ซึ่งเป็นผลให้ค่าแรงบิดขาออกไม่ตรงตามที่ต้องการในทุกค่าความเร็วที่ทดสอบ

อย่างไรก็ตาม การแสดงผลการวิเคราะห์จากการทดสอบมีความจำเป็นอย่างยิ่งต่อการศึกษาการทำงานของมอเตอร์ทั้งสองย่านการทำงาน เพื่อสามารถนำไปใช้ในการวิเคราะห์และพัฒนาระบบควบคุม BLDC Motor ต่อไป ดังนั้นในโครงงานนี้จึงได้พัฒนาการแสดงผลวิเคราะห์การทดสอบแบบ General User Interface (GUI) โดยการประยุกต์ใช้โปรแกรม Visual Basic

1.3 วัตถุประสงค์ของงานวิจัย

- ศึกษาหลักการทำงานของ BLDC Motor
- ศึกษาพฤติกรรมการทำงานของ BLDC Motor ในย่าน Constant Torque และ Constant Power
- ศึกษาหลักการชดเชยมุมเฟสก้าวหน้า (Advanced Phase Angle Commutation) ใน BLDC Motor
- ออกแบบสร้างระบบจำลองการชดเชยมุมเฟสก้าวหน้าใน BLDC Motor
- ออกแบบสร้างชุดทดสอบและระบบควบคุมการทำงานของ BLDC Motor
- เปรียบเทียบการทำงานของ BLDC Motor ทั้งขณะไม่มีการชดเชยมุมเฟสก้าวหน้าและมีการชดเชยมุมเฟสก้าวหน้า

1.4 ขอบเขตงานวิจัย

- ออกแบบและสร้างระบบจำลองการควบคุม BLDC Motor ด้วยหลักการชดเชยมุมเฟสก้าวหน้า (Advanced Phase Angle Commutation)
- ออกแบบและสร้างชุดทดลองการทำงานของมอเตอร์ไฟฟ้ากระแสตรงแบบไร้แปรงถ่าน (BLDC Motor) ขนาดพิกัดกำลังไฟฟ้า 500 W. พร้อมระบบควบคุมโดยการประยุกต์ใช้บอร์ดประมวลผลสัญญาณรุ่น dsPIC33F2010
- วิเคราะห์ผลการทำงานของระบบจากชุดทดลองเปรียบเทียบกับผลที่ได้จากการจำลองระบบ

1.5 ประโยชน์ที่คาดว่าจะได้รับ

- เป็นแนวทางในการนำไปสร้างชุดขับเคลื่อนมอเตอร์ไฟฟ้ากระแสตรงแบบไร้แปรงถ่านให้เหมาะสมกับการนำไปใช้งานเชิงปฏิบัติ
- เป็นแนวทางในการพัฒนาระบบควบคุมการขับเคลื่อนมอเตอร์ไฟฟ้ากระแสตรงแบบไร้แปรงถ่านในอนาคต

บทที่ 2
เอกสารงานวิจัยและทฤษฎีที่เกี่ยวข้อง

2.1 บทนำ

มอเตอร์ไฟฟ้ากระแสตรงแบบไร้แปรงถ่าน (Brushless DC Motor) เกิดขึ้นในปี ค.ศ 1962 เมื่อ T.G. Wilson และ P.H. Trickey [5] ได้สร้างมอเตอร์ไฟฟ้ากระแสตรงโดยใช้อุปกรณ์ทางอิเล็กทรอนิกส์เป็นชุดควบคุมการไหลของกระแสในขดลวดสเตเตอร์ ซึ่งเรียกว่า Electronic Commutation ซึ่งถูกพัฒนาให้เป็นเครื่องกลไฟฟ้าทีมีแรงบิดสูงและ ตอบสนองต่อการใช้งานเฉพาะทางได้อย่างดีเยี่ยม เช่น เครื่องเล่นเทป อุปกรณ์อ่านข้อมูลในระบบคอมพิวเตอร์ ระบบควบคุมตำแหน่งในงานอุตสาหกรรมแบบต่างๆ รวมทั้งการประยุกต์ใช้งานในเทคโนโลยีอากาศยานด้วย แต่ในช่วงเวลานั้น Brushless DC Motor มีพิกัดกำลังไฟฟ้าไม่เกิน 5 HP. เท่านั้น แต่หลังจากการค้นพบแม่เหล็กถาวรแบบกำลังสูง เช่น Neodymium (NdFeB) พร้อมทั้งความก้าวหน้าทางเทคโนโลยีทางด้านอิเล็กทรอนิกส์กำลังที่อุปกรณ์สวิตซ์อิเล็กทรอนิกส์กำลังมีพิกัดกำลังไฟฟ้าสูงขึ้น และตอบสนองต่อความถี่สวิตซ์ได้สูงขึ้นมาก ในราวๆ กลางทศวรรศที่ 80 (คศ. 1980) จึงทำให้สามารถสร้าง Brushless DC Motor ที่มีพิกัดกำลังไฟฟ้าสูงขึ้นได้ โดย Brushless DC motor ขนาดพิกัดกำลังไฟฟ้าที่ 50 HP. ถูกสร้างขึ้นโดย Robert E. Lordo จากบริษัท Industrial Corporation และหลังจากนั้นจึงได้เริ่มมีการผลิต Brushless DC Motor ที่พิกัดกำไฟฟ้าต่างๆ ตั้งแต่ 0.5 HP. ถึง 300 HP. ตั้งแต่ปี 1980 จนถึงปัจจุบัน

เนื่องจาก Brushless DC Motor มีคุณลักษณะการทำงานที่ใกล้เคียงกับ Brush DC Motor ซึ่งเป็นมอเตอร์ไฟฟ้ากระแสตรงแบบใช้แปรงถ่านที่คุ้นเคยและใช้งานกันมานานแล้ว รวมทั้ง Brushless DC Motor มีประสิทธิภาพการทำงานค่อนข้างสูง (มากกว่า 90 %) มีคุณสมบัติทางกายภาพที่เหมาะสมต่อการใช้งานได้ในหลายประเภท มีค่าแรงบิดเฉื่อย (Moment of Inertia) ที่ค่อนข้างต่ำ เป็นผลให้มีอัตราเร่งความเร็วได้สูง และตอบสนองต่อการเปลี่ยนแปลงความเร็วได้ดีกว่า อีกทั้งประสิทธิภาพของชุดควบคุมการขับเคลื่อนในปัจจุบันที่สูงถึงประมาณ 97 % จึงทำให้ Brushless DC Motor เป็นที่นิยมใช้ในงานอุตสาหกรรมอย่างกว้างขวางเช่นกัน

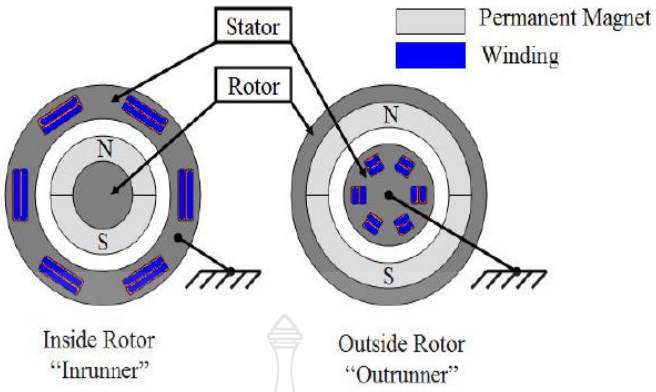
คุณลักษณะเปรียบเทียบระหว่าง Brushless DC Motor กับ Brushed DC Motor ขนาดพิกัดกำลังไฟฟ้า 150 HP. ขณะทำงานที่ 80 % ของพิกัด แสดงดังในตารางที่ 2.1

ตารางที่ 2.1 คุณสมบัติของ Brushless DC Motor และ Brushed Motor ที่ 80 % ของพิกัด 150 HP.

คุณลักษณะ	Brushed DC Motor	Brushless DC Motor	หน่วย
TORQUE OUTPUT	360	360	Lbs-ft
SPEED	1400	1400	RPM
POWER OUTPUT	71616	71616	WATT
MOTOR VOLTAGE	400	206	VDC/VAC
MORTOR CURRENT	196	206	ADC/AAC
POWER TO MOTOR	79400	75845	WATT
MOTOR EFFICIENCY	90.2	94.4	%
CONTROLLER LOSS	1007	1562	WATT
INPUT POWER	80407	77407	WATT
AC LINE CURRENT	160	102	A.
POWER FACTOR	0.63	0.96	
KVR PER KW	1.59	1.05	

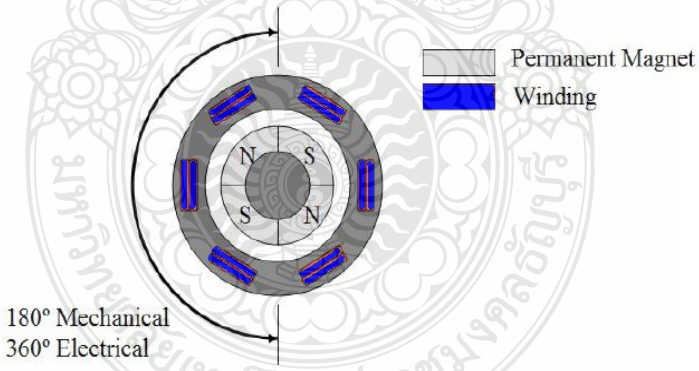
2.2 พื้นฐานเกี่ยวกับมอเตอร์ไฟฟ้ากระแสตรงไร้แปรงถ่าน

มอเตอร์ไฟฟ้ากระแสตรงแบบไร้แปรงถ่าน (Brushless DC Motor หรือ BLDC Motor) มีองค์ประกอบทางกายภาพเหมือนกับมอเตอร์เหนี่ยวนำไฟฟ้ากระแสสลับทั่วไป คือ ประกอบด้วยส่วนขดลวดอยู่กับที่ หรือสเตเตอร์ (Stator) และส่วนขดลวดหมุน หรือ โรเตอร์ (Rotor) โดยส่วน Stator จะเป็นที่รองรับขดลวดสเตเตอร์ (จำนวน1 ชุดต่อเฟส) และส่วน Rotor จะเป็นแม่เหล็กถาวร ทั้งนี้ Brushless DC Motor ที่ใช้อยู่ในปัจจุบัน จะมีทั้งแบบตัวหมุน (ติดตั้งแม่เหล็กถาวร) อยู่ด้านในและแบบตัวหมุนอยู่ด้านนอก ดังแสดงในรูปที่ 2.1 ตามลำดับ



รูปที่ 2.1 โครงสร้าง BLDC Motor ที่มีการวางตำแหน่งตัวหมุนกับขดลวดสเตเตอร์[6]

Brushless DC Motor ที่มีใช้อยู่ในปัจจุบันมีตั้งแต่ 1 เฟส 2 เฟส หรือ 3 เฟส ขึ้นอยู่กับจำนวนของชุดขดลวด Stator ซึ่งที่เป็นที่นิยมกันมากที่สุดเป็นแบบ Brushless DC Motor ชนิด 3 เฟส ทั้งนี้จำนวนขั้วแม่เหล็กของมอเตอร์จะขึ้นอยู่กับจำนวนขั้วแม่เหล็กถาวรที่ Rotor ซึ่งต้องมีอย่างน้อย 2 ขั้วแม่เหล็ก ประกอบด้วยขั้วเหนือและขั้วใต้ ส่วนมอเตอร์ที่มีจำนวนขั้วแม่เหล็กถาวรมากกว่า 2 ขั้วนั้นจะมีผลต่อความสัมพันธ์ระหว่างองศาทางกลกับองศาทางไฟฟ้าของตัวหมุนเท่านั้น ดังแสดงในรูปที่ 2.2



รูปที่ 2.2 ความสัมพันธ์ระหว่างองศาทางกลกับองศาทางไฟฟ้าของมอเตอร์ 4 ขั้วแม่เหล็ก [6]

ความสัมพันธ์ระหว่างองศาทางกลกับองศาทางไฟฟ้าแสดงให้เห็นตามสมการที่ 2.1 และ 2.2

$$\theta_e = \frac{P}{2}.\theta_m \qquad (2.1)$$

$$\omega_e = P\omega_m \qquad (2.2)$$

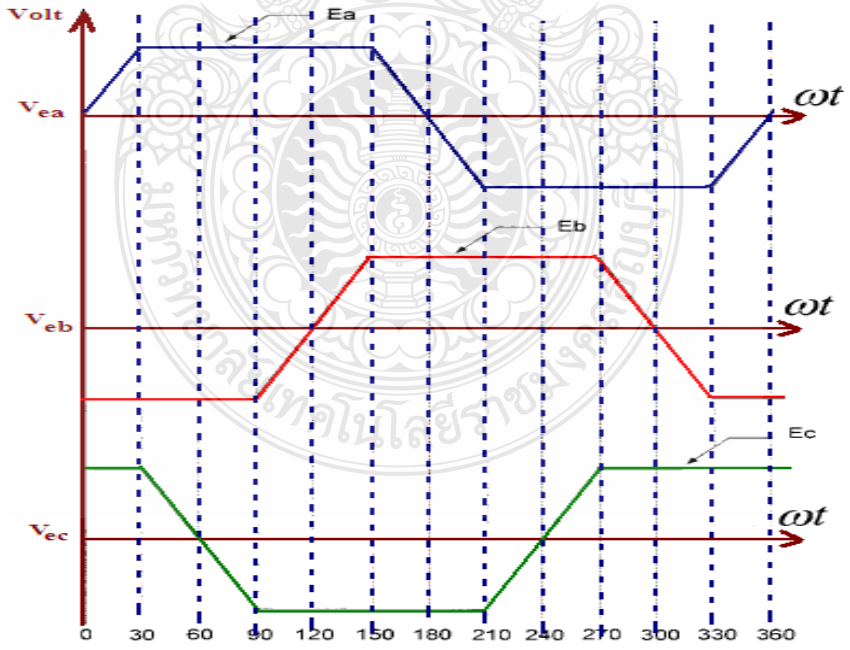
โดย θ_e องศาทางไฟฟ้า ω_e ความเร็วเชิงมุมทางไฟฟ้า

θ_m องศาทางกล ω_m ความเร็วเชิงมุมทางกล

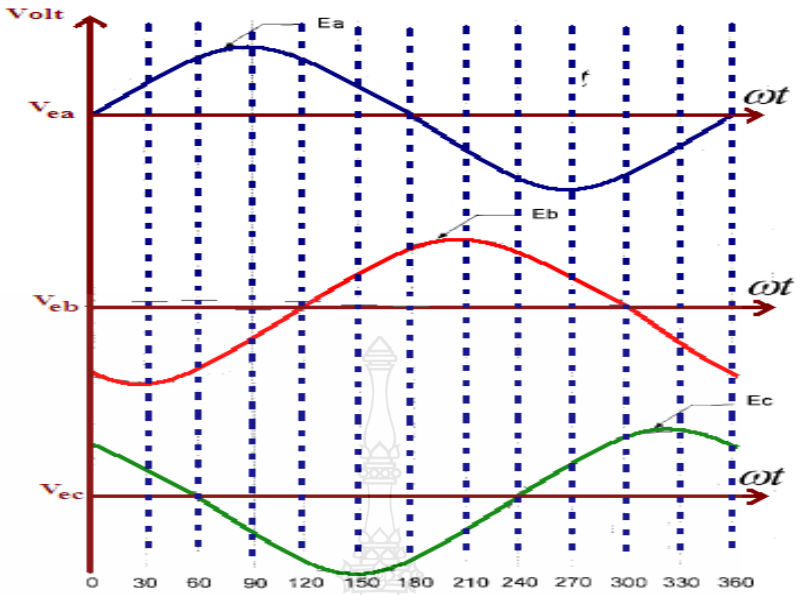
P จำนวน pole

2.3 แรงดันต้านกลับ (Back EMF) ในมอเตอร์ไฟฟ้ากระแสตรงไร้แปรงถ่าน [7]

มอเตอร์ไฟฟ้ากระแสตรงไร้แปรงถ่านยังสามารถแบ่งออกได้ตามลักษณะของแรงดันต้านกลับ (Back EMF) ที่เกิดขึ้นในมอเตอร์นี้ได้เป็น 2 แบบคือ แบบที่ Back EMF มีลักษณะเป็น Trapezoidal Back EMF (สัญญาณรูปสี่เหลี่ยมคางหมู) และ Sinusoidal Back EMF (สัญญาณรูปซายน์) ซึ่งเป็นการเรียกตามลักษณะของรูปคลื่นแรงดัน Back EMF ทั้งนี้ความแตกต่างของรูปคลื่นแรงดัน Back EMF มาจากลักษณะการพันขดลวด Stator ลักษณะของรูปคลื่นแรงดัน Back EMF แบบ Trapezoidal และ Sinusoidal แสดงดังรูปที่ 2.3 และรูปที่ 2.4 ตามลำดับ



รูปที่ 2.3 รูปคลื่นแรงดันรูปคลื่นสี่เหลี่ยมคางหมู



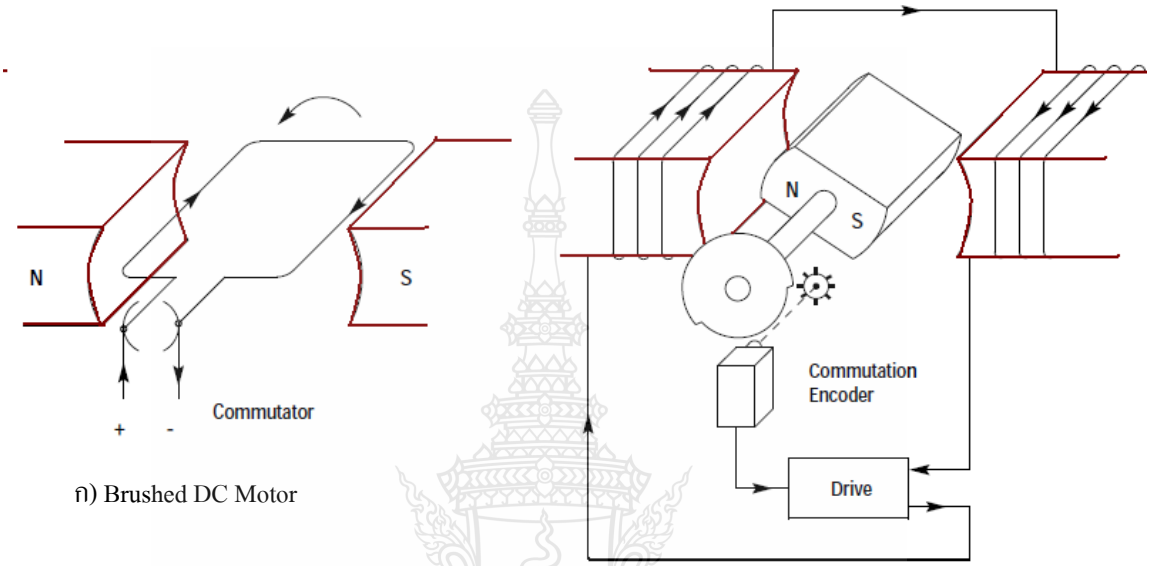
รูปที่ 2.4 รูปคลื่นแรงดันรูปคลื่นซายน์

จากความแตกต่างของรูปคลื่นแรงดัน Back EMF ดังกล่าว จะส่งผลถึงการออกแบบชุดขับเคลื่อนที่แตกต่างกันเช่นกัน ซึ่งทั้งนี้จะต้องออกแบบระบบควบคุมมอเตอร์ให้มีการจ่ายกระแสสเตเตอร์ให้มีรูปร่างใกล้เคียงกับแรงดัน Back EMF มากที่สุด เนื่องจากจะส่งผลโดยตรงต่อประสิทธิภาพของแรงบิดขาออกของมอเตอร์ ทั้งนี้การสร้างวงจรควบคุมสำหรับ BLDC Motor แบบ Sinusoidal Back EMF จะมีความยุ่งยากมากกว่า การสร้างวงจรควบคุมสำหรับ BLDC Motor แบบ Trapezoidal Back EMF จึงทำให้ BLDC Motor แบบ Trapezoidal Back EMF มีการนำไปประยุกต์ใช้ในงานทั่วไปมากกว่าแบบ Sinusoidal Back EMF

2.4 กระบวนการคอมมิวเตชั่น [8]

โดยทั่วไป Brushed DC Motor จะใช้แปรงถ่านทำหน้าที่ควบคุมการไหลของกระแสในขดลวดส่วนหมุน เพื่อสร้างสนามแม่เหล็กหมุนระหว่างขดลวดทั้งสอง ซึ่งเป็นการควบคุมทิศทางกระแสแบบทางกล (Mechanical Commutation) ปัญหาที่เกิดขึ้น คือการเกิดประกายไฟที่แปรงถ่าน ซึ่งจะไม่เหมาะสมกับการนำไปใช้งานที่ต้องหลีกเลี่ยงการเกิดประกายไฟ และเกิดการสึกหรอจากการเสียดสีที่แปรงถ่านนั้น จึงต้องมีช่วงเวลาในการซ่อมบำรุงที่แน่นอน

Brushless DC Motor ซึ่งเป็นวิวัฒนาการของ brushed DC Motor ได้เปลี่ยนวิธีการควบคุมการไหลของกระแสในขดลวด จากวิธีการคอมมิวเตชั่นทางกล (Mechanical Commutation) เป็นวิธีการคอมมิวเตชั่นทางอิเล็กทรอนิกส์ (Electronic commutation) ซึ่งจะไม่มีการสัมผัสทางกลเนื่องจากไม่ใช้แปรงถ่านในส่วน Rotor นั่นเอง



ก) Brushed DC Motor

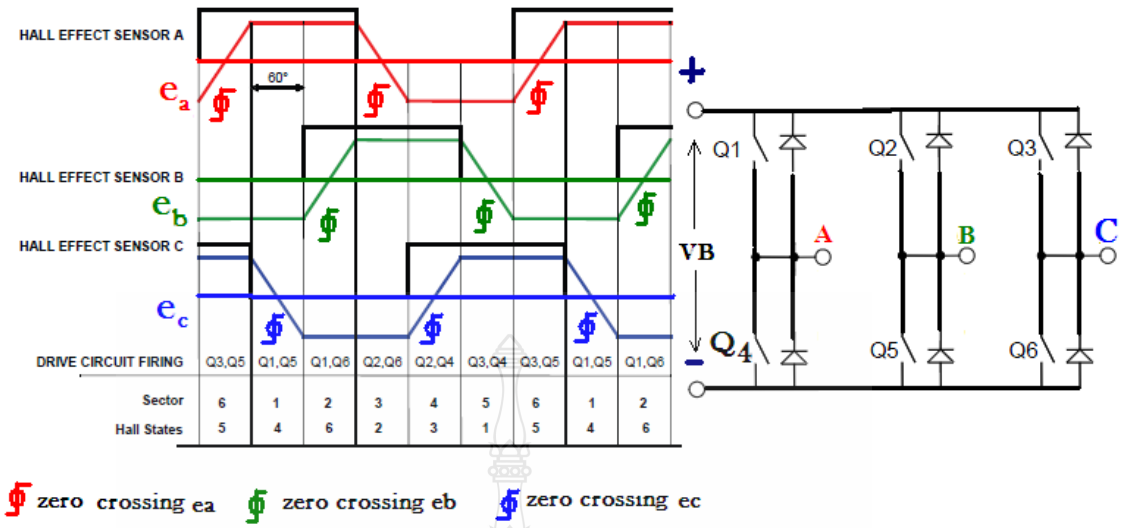
ข) Brushless DC Motor

รูปที่ 2.5 โครงสร้างของ Brushed DC Motor และ Brushless DC Motor

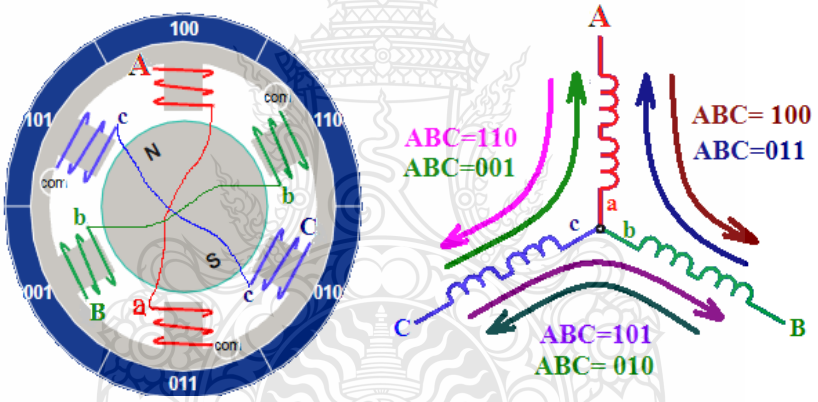
จากรูปที่ 2.5 จะเห็นว่าการจ่ายกระแสให้กับชุดขดลวด Stator ทั้งในแบบ Brushed DC Motor และแบบ Brushless DC Motor จะมีความสัมพันธ์กับตำแหน่งของตัวหมุน (Rotor) เสมอ

ตัวตรวจจับตำแหน่งของตัวหมุน(rotor) สำหรับ Brushless DC Motor มีหลายรูปแบบ แต่ที่นิยมใช้ในปัจจุบัน แบบแรกจะเป็นตัวตรวจจับแบบ Hall Sensor ซึ่งเป็นตัวตรวจจับสัญญาณจากการเปลี่ยนแปลงสนามแม่เหล็ก และแบบที่สอง จะเป็นการตรวจจับหาตำแหน่งของตัวหมุนจากจุดที่แรงดัน Back EMF ตกลงสู่ค่าศูนย์ (Zero crossing of Back EMF)

รูปที่ 2.6 และรูปที่ 2.7 แสดงสัญญาณจาก Hall Sensor และรูปคลื่นแรงดัน Back EMF สำหรับใช้ในการคอมมิวเตชั่นกระแสไหลในขดลวดสเตเตอร์ของ Brushless DC Motor ชนิด 3 เฟส



รูปที่ 2.6 สัญญาณจาก Hall Sensor และรูปคลื่น Back EMF ใน BLDC Motor ชนิด 3 Phase

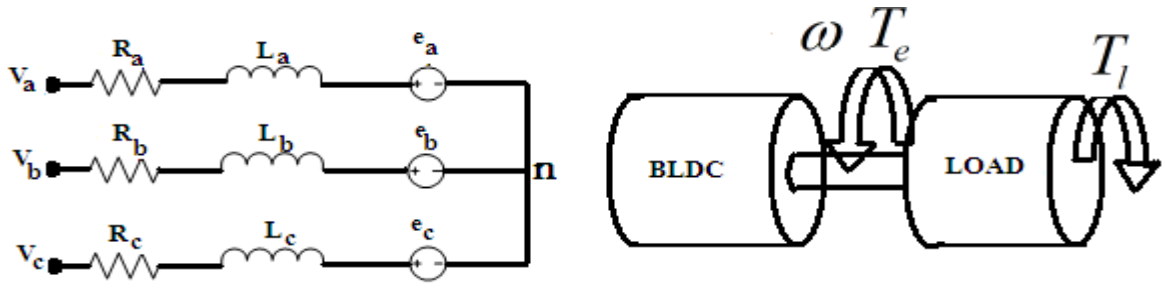


รูปที่ 2.7 ตำแหน่งติดตั้งของ Hall Sensor และทิศทางกระแสไหลในขดลวดเสเตเตอร์ใน 1 ไซเคิล

2.5 แบบจำลองทางคณิตศาสตร์ของมอเตอร์ไฟฟ้ากระแสตรงไร้แปรงถ่าน [9]

โดยทั่วไป เราสามารถใช้สมการทางคณิตศาสตร์ในการอธิบายการเปลี่ยนแปลงกำลังไฟฟ้าเป็นกำลังงานกล (Power developed) ตามหลักพลศาสตร์ ในมอเตอร์ไฟฟ้ากระแสตรงแบบไร้แปรงถ่านได้เช่นเดียวกับในมอเตอร์ชนิดอื่นๆ

รูปที่ 2.8 แสดงวงจรสมมูลทางไฟฟ้าและแบบจำลองทางกลของมอเตอร์ไฟฟ้ากระแสตรงแบบไร้แปรงถ่าน (Brushless DC Motor)



รูปที่ 2.8 วงจรสมมูลทางไฟฟ้าและแบบจำลองทางกลของ BLDC Motor

จากวงจรสมมูลของ BLDC Motor สามารถเขียนอยู่ในรูปสมการทางคณิตศาสตร์ได้

$$V_a = Ri_a + L\frac{di_a}{dt} + e_a \tag{2.3}$$

$$V_b = Ri_b + L\frac{di_b}{dt} + e_b \tag{2.4}$$

$$V_c = Ri_c + L\frac{di_c}{dt} + e_c \tag{2.5}$$

โดย

L	เป็นค่าความเหนี่ยวนำของขดลวดสเตเตอร์ (Stator coil 's inductance) [H]
R	เป็นค่าความต้านทานของขดลวดสเตเตอร์ (Stator coil's resistance) [Ω]
V_{an}, V_{bn}, V_{cn}	เป็นแรงดันตกคร่อมชุดขดลวดสเตเตอร์ [V]
i_a , i_b , i_c	เป็นกระแสไหลในขดลวดสเตเตอร์ (Motor input current) [A]
e_a , e_b , e_c	เป็นแรงดัน Back EMF ของขดลวดสเตเตอร์ในแต่ละเฟส [V]

ใน Brushless DC Motor ชนิด 3 เฟส แรงดัน Back EMF จะสัมพันธ์กับตำแหน่งในส่วนหมุน (Rotor) ซึ่งแรงดัน Back EMFในแต่ละเฟสจะทำมุมกัน 120^o ไฟฟ้า ดังแสดงในสมการที่ (2.4) (2.5) และ (2.6) ตามลำดับ

$$e_a = K_w f(\theta_e)\omega \tag{2.6}$$

$$e_b = K_w f(\theta_e - 2\pi/3)\omega \tag{2.7}$$

$$e_c = K_w f(\theta_e + 2\pi/3)\omega \tag{2.8}$$

โดย K_w ค่าคงที่ของแรงดันต้านกลับ (Back EMF Constant of each phase) [v/rad. s^{-1}]

θ_e ค่ามุมทางไฟฟ้าของตัวหมุน(electric rotor angle) [$^o el$]

ω ความเร็วของตัวหมุน(rotor speed) [rad. s^{-1}]

แรงบิดทางไฟฟ้าที่ได้เป็นการรวมแรงกันในแต่ละเฟสซึ่งอธิบายได้ตามสมการที่ (2.7)

$$T_e = \frac{e_a i_a + e_b i_b + e_c i_c}{\omega} \tag{2.9}$$

โดย T_e เป็นแรงบิดรวมทางไฟฟ้า (Total Electromagnetic Torque) [N.m]

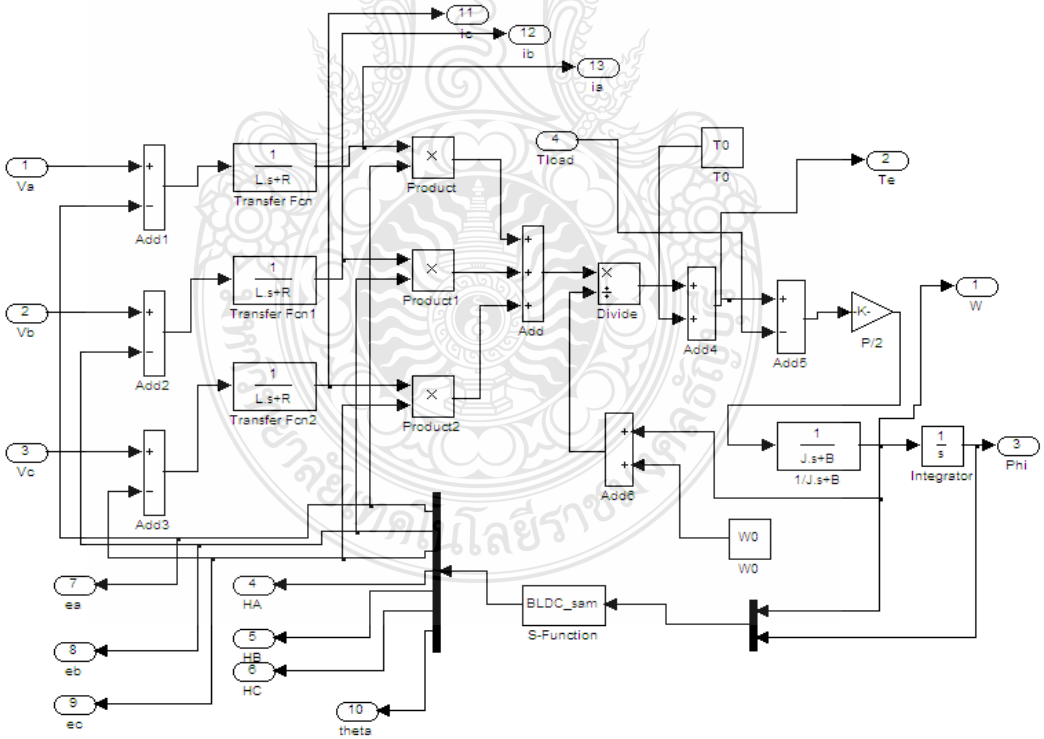
ในส่วนของสมการทางกลอธิบายให้เห็นได้ตามสมการที่ (2.8)

$$T_e - T_l = J\frac{d\omega}{dt} + B\omega \tag{2.10}$$

โดย T_l เป็นแรงบิดที่โหลด (Load Torque) [N.m]

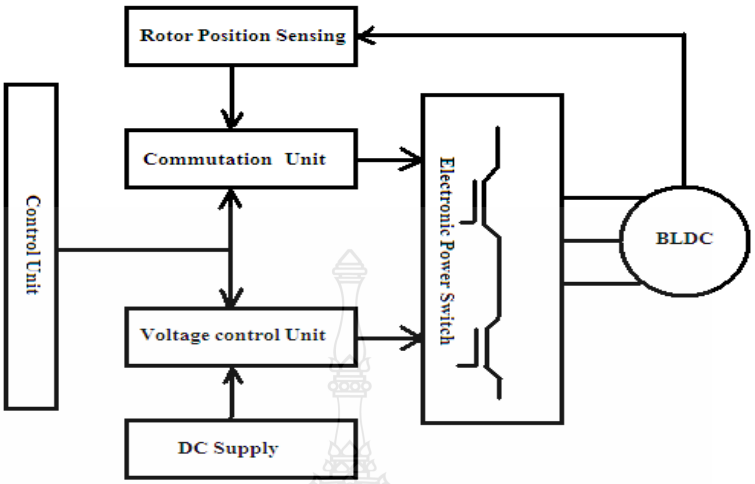
B ค่าแรงเสียดทานความหนืดคงที่ (Viscous friction constant)

ทั้งนี้ จากสมการที่ (2.3) ถึง (2.8) นำมาเขียนเป็นระบบจำลองทางคณิตศาสตร์ได้ดังนี้



รูปที่ 2.9 แสดงระบบจำลองทางคณิตศาสตร์ของ Brushless DC Motor

2.6 การขับเคลื่อนมอเตอร์ไฟฟ้ากระแสตรงไร้แปรงถ่าน



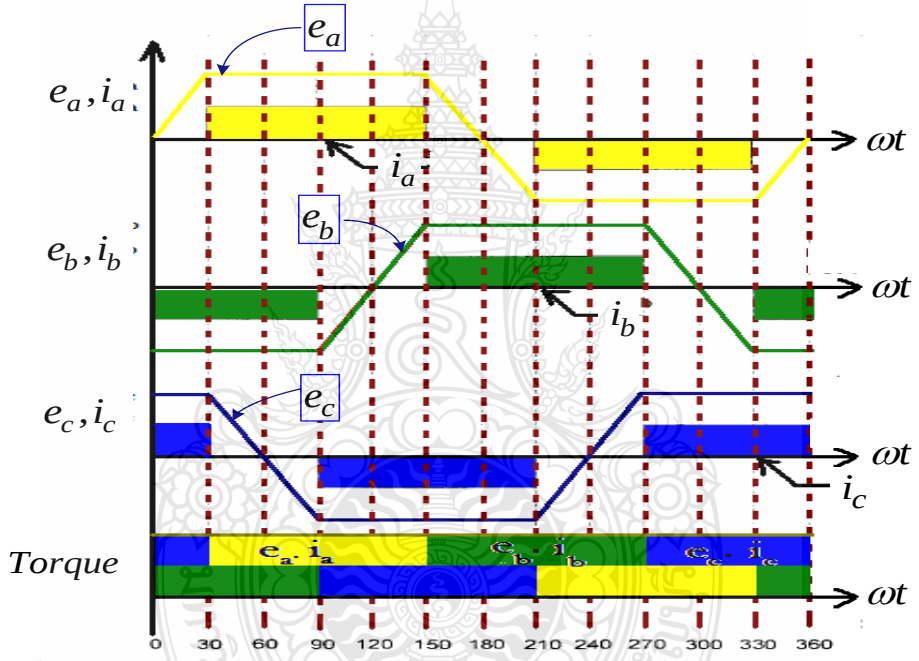
รูปที่ 2.10 องค์ประกอบระบบขับเคลื่อนของ Brushless DC Motor แบบลูปเปิด

จากรูปที่ 2.10 ระบบควบคุมการขับเคลื่อน Brushless DC Motor โดยทั่วไปแบบแบบลูปเปิดประกอบด้วย

- ส่วนตรวจจับตำแหน่งของส่วนหมุน ซึ่งอาจใช้ตัวตรวจจับสัญญาณแบบ Hall Sensor หรือตรวจจับค้นหาตำแหน่งเริ่มต้นของแรงดัน Back EMF แต่ละเฟส ด้วยหลักการ Zero Crossing
- ส่วนควบคุมการไหลของกระแสในขดลวดสเตเตอร์แต่ละเฟส (Commutation Unit) โดยใช้สัญญาณจากส่วนตรวจจับตำแหน่งส่วนหมุนเป็นตัวกำหนดจุดเริ่มต้นการนำกระแสแต่ละเฟส
- ส่วนควบคุมขนาดแรงดันที่จ่ายให้กับมอเตอร์ (Voltage Control Unit) เพื่อคุมค่าความเร็วและแรงบิดที่เกิดขึ้นในมอเตอร์นั้น
- ส่วนควบคุมระบบรวม (Control Unit) เนื่องจากส่วนควบคุมแต่ละส่วนจะต้องทำงานสัมพันธ์ซึ่งกันละกัน
- แหล่งจ่ายแรงดันไฟตรงให้กับระบบทั้งหมด

2.6.1 หลักการควบคุมการไหลกระแสของชุดขดลวดสเตเตอร์

กระแสไหลในแต่ละชุดขดลวดสเตเตอร์ของ Brushless DC Motor ในแต่ละคาบเวลาจะต้องสัมพันธ์กับตำแหน่งของตัวหมุน (Rotor) ดังแสดงในรูปที่ 2.11 ซึ่งจะเห็นจะสัมพันธ์กับจุดเริ่มต้นของแรงดัน Back EMF ที่เกิดขึ้นในแต่ละเฟสเช่นกัน ทั้งนี้ระบบควบคุมอาจใช้สัญญาณจากตัวตรวจจับแบบ Hall sensor เป็นตัวตรวจจับตำแหน่ง รูปแบบของการควบคุมการไหลของกระแสของชุดขดลวดสเตเตอร์ใน Brushless DC Motor ชนิดที่ให้สัญญาณแรงดัน Back EMF เป็นรูปสี่เหลี่ยมคางหมู (Trapezoidal Back EMF) ในอุดมคติ ซึ่งจะทำให้ลักษณะของรูปคลื่นกระแสในแต่ละเฟสเป็นแบบ รูปคลื่นสี่เหลี่ยม (Square Wave) ซึ่งจุดเริ่มต้นของการไหลของกระแสในแต่ละเฟสจะเป็นเวลาของจุดอิ่มตัวของแรงดัน Back EMF ในเฟสนั้นๆ เช่นกัน ดังแสดงในรูปที่ 2.11

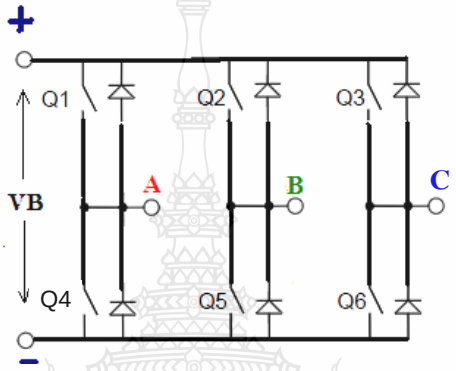


รูปที่ 2.11 รูปคลื่นของแรงดัน Back EMF กระแสไหลในขดลวดสเตเตอร์ และแรงบิดที่เกิดขึ้นในแต่ละเฟส ของ Brushless DC Motor

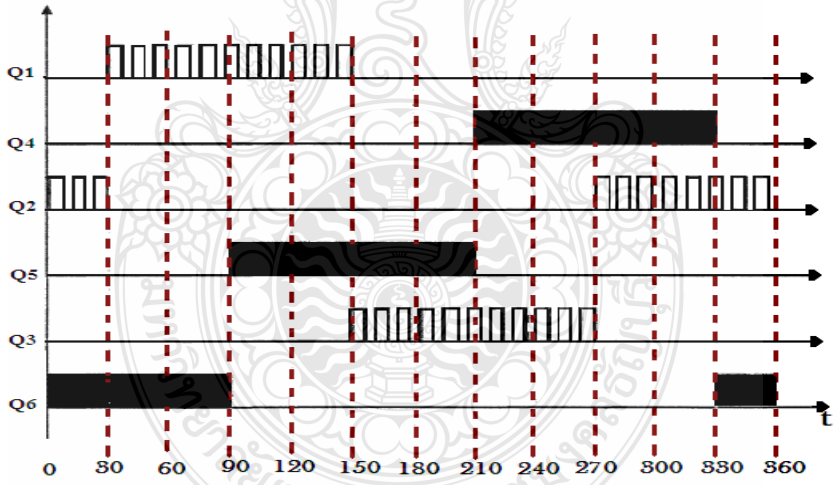
2.6.2 ส่วนควบคุมคุมแรงดันมอเตอร์ (Voltage Control Unit)

นอกจากการควบคุมการไหลของกระแสที่จ่ายให้กับขดลวดสเตเตอร์ในแต่ละเฟส ดังแสดงในรูปที่ 2.11 แล้ว ยังจำเป็นจะต้องมีการควบคุมแรงดันที่จ่ายให้กับมอเตอร์อีกด้วย เพื่อให้สามารถควบคุมความเร็วและแรงบิดที่เกิดขึ้นให้เหมาะสมในแต่ละสภาวะการทำงานของมอเตอร์ได้ จากรูปที่ 2.12 ก). แสดงรูปวงจรแปลงกำลังไฟฟ้า (แบบวงจรอินเวอร์เตอร์) ที่ใช้ในการควบคุมแรงดันและ

กระแสที่ป้อนให้กับขดลวดสเตเตอร์ทั้ง 3 เฟสของ Brushless DC Motor ส่วนรูปแบบการควบคุมการทำงานของสวิตซ์อิเล็กทรอนิกส์กำลังทั้ง 6 ตัว (Q1 …. Q6) เพื่อให้ควบคุมแรงดันและกระแสที่จ่ายให้กับขดลวดมอเตอร์ในแต่ละชุด จะเป็นไปตามรูปที่ 2.12 ข). ทั้งนี้จะเห็นว่าในแต่ละจังหวะการทำงาน สวิตซ์อิเล็กทรอนิกส์กำลังจะทำงานพร้อมกันครั้งละสองตัว โดยสวิตซ์อิเล็กทรอนิกส์กำลังตัวหนึ่งใน 2 ตัวที่ทำงานพร้อมกัน จะถูกนำกระแสด้วยสัญญาณแบบมอดูเลตตามความกว้างพัลส์ (Pulse Width Modulation; PWM) เพื่อทำหน้าที่ควบคุมขนาดแรงดันเฉลี่ยที่จ่ายให้กับขดลวดมอเตอร์ได้



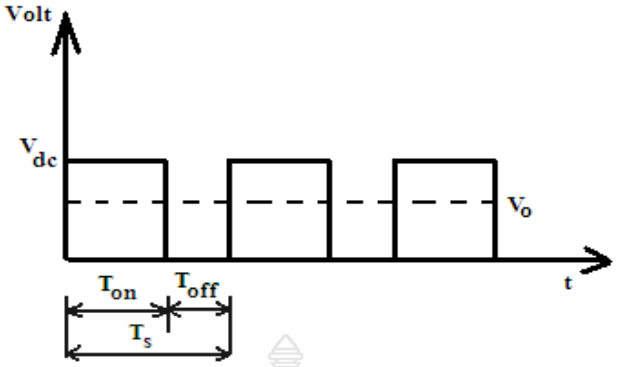
[ก]โครงสร้างวงจรอินเวอร์เตอร์ที่ใช้ในการควบคุม Brushless DC Motor



[ข] สัญญาณควบคุมสวิตซ์อิเล็กทรอนิกส์กำลัง (Q1 – Q6) เพื่อควบคุม Brushless DC Motor

รูปที่ 2.12 วงจรอินเวอร์เตอร์และสัญญาณควบคุม

ทั้งนี้หลักการมอดูเลตตามความกว้างพัลส์ (Pulse Width Modulation หรือ PWM) สามารถใช้ควบคุมแรงดันเฉลี่ยไฟตรงขาออกได้ โดยควบคุมอัตราการปิด/เปิดของสวิตซ์อิเล็กทรอนิกส์กำลังในแต่ละคาบเวลา ดังแสดงในรูปที่ 2.13



รูปที่ 2.13 ค่าเวลาในการเปิด/ปิดสวิตซ์ตามหลักการ PWM

ทั้งนี้ $$D = \frac{Ton}{T_s} \tag{2.9}$$

$$V_o = \frac{1}{T_S}[\int_0^{T_{on}} V_{dc} d(t) + \int_{T_{on}}^{T_S} 0\, d(t) = \frac{T_{ON}}{T_S}.V_{DC} \tag{2.10}$$

หรือ $$V_O = D.V_{DC} \tag{2.11}$$

2.7 การเพิ่มความเร็วและแรงบิดในย่านกำลังคงที่ใน Brushless DC Motor

ในย่านการทำงานแบบกำลังคงที่ใน Brushless DC Motor ดังแสดงไว้แล้วในรูปที่ 1.1 นั้น สามารถปรับค่าความเร็วและค่าแรงบิดที่เกิดขึ้นในย่านนี้ได้ โดยใช้การปรับค่าความต่างของมุมเฟสระหว่างแรงดันและกระแสสเตเตอร์ในแต่ละเฟส ซึ่งเรียกว่า หลักการชดเชยมุมเฟสก้าวหน้า (Advanced Phase Angle Commutation, α_a) [1] ทั้งนี้โดยการควบคุมกระแสสเตเตอร์ให้ทำมุมนำหน้าแรงดันต้านกลับ(Back EMF) ในแต่ละเฟส ดังแสดงในรูปที่ 2.15

ค่ามุมเฟสก้าวหน้าที่ใช้นี้ เพื่อใช้ในการชดเชยสนามแม่เหล็กซึ่งในสภาวะปกติจะเกิดขึ้นจากกรณีกระแสล้าหลังแรงดันนั่นเอง ทั้งนี้ค่ามุมเฟสก้าวหน้าจะมีค่าเปลี่ยนแปลงตามค่าความเร็ว [10] ดังสมการที่ 2.12

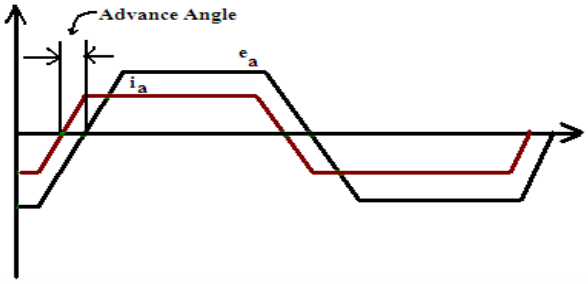
$$\alpha_a = \omega_r \frac{L_s I}{V_{dc}};\ \omega_r = 2pn\pi \tag{2.12}$$

เมื่อ L_s เป็นค่าความเหนี่ยวนำของชุดขดลวดสเตเตอร์แต่ละเฟส

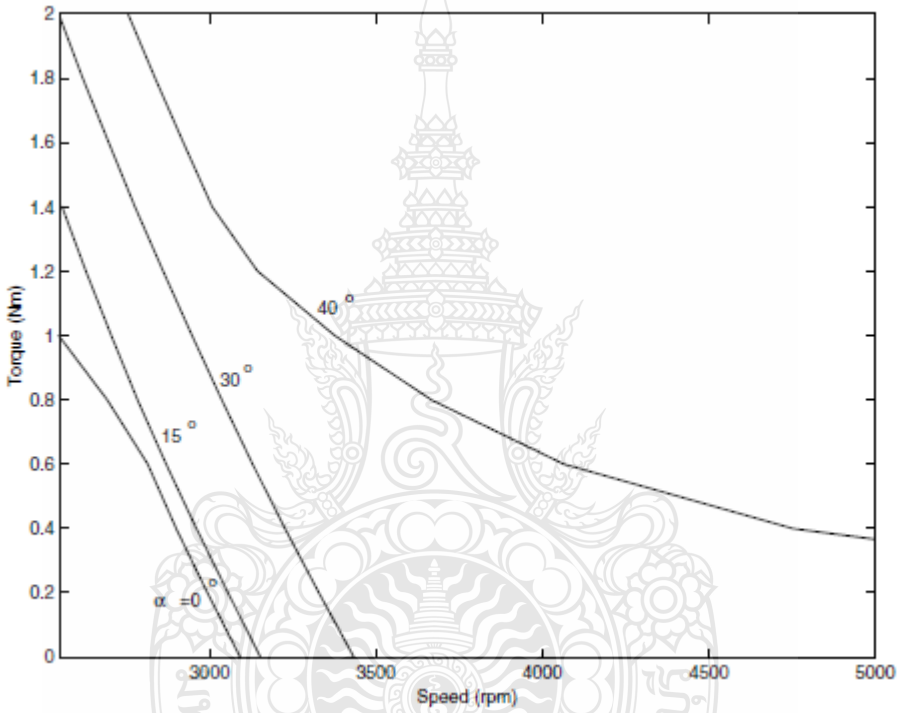
p เป็นจำนวนขั้วแม่เหล็กของมอเตอร์

n เป็นค่าความเร็วมอเตอร์ (รอบ/วินาที หรือ rps.)

V_{dc} เป็นขนาดแรงดันไฟฟ้าที่จ่ายให้กับมอเตอร์ (Volt)



รูปที่ 2.14 การชดเชยมุมเฟสก้าวหน้า (Advanced Phase Angle)



รูปที่ 2.15 ความสัมพันธ์ระหว่างความเร็วและแรงบิดมอเตอร์ที่มุมเฟสก้าวหน้า 0,15,30 และ 40 องศา

2.8 การออกแบบระบบควบคุมมอเตอร์ไฟฟ้ากระแสตรงไร้แปรงถ่านในย่านกำลังคงที่ด้วยหลักการชดเชยมุมเฟสก้าวหน้า

การออกแบบชุดควบคุมการขับเคลื่อนมอเตอร์ไฟฟ้ากระแสตรงไร้แปรงถ่าน (Brushless DC Motor Drive Unit) เพื่อใช้ในงานวิจัยชิ้นนี้มีหน่วยควบคุมหลักๆ 4 ส่วนด้วยกัน คือ

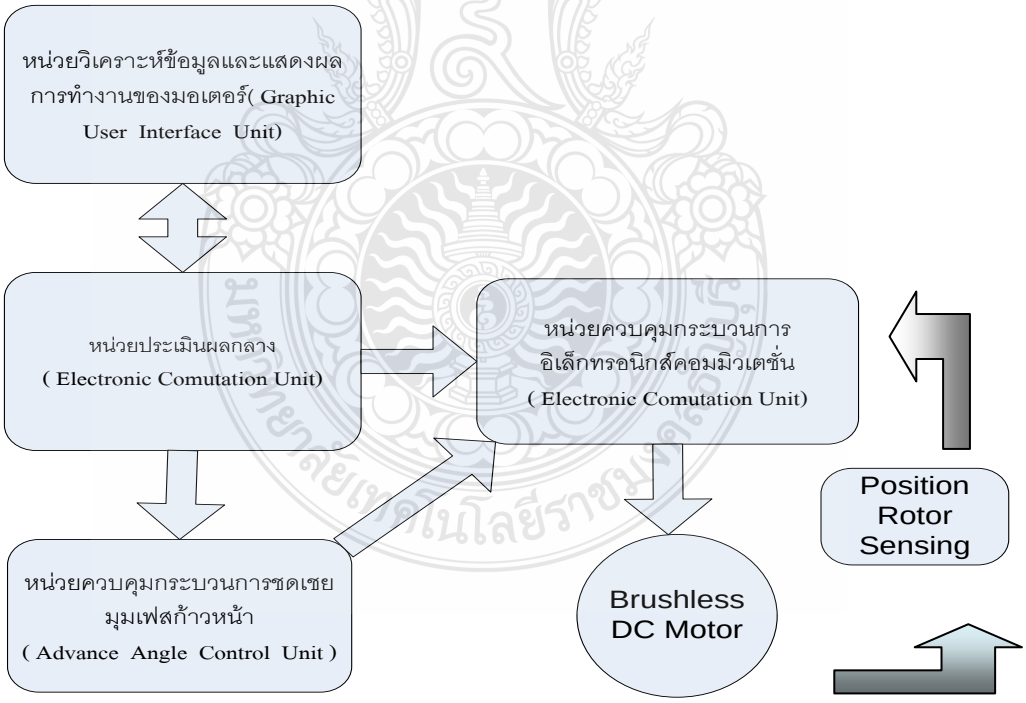
1 หน่วยควบคุมกระบวนการ อิเล็กทรอนิกส์คอมมิวเตชั่น (Electronic Comutation Unit) ทำหน้าที่ในการควบคุมกระบวนการจ่ายกระแสในชุดขดลวดอยู่กับที่ของมอเตอร์ในแต่ละจังหวะนั้นจะต้องสัมพันธ์กับตำแหน่งของตัวหมุน (rotor) ตามข้อ 2.6.1

2 หน่วยควบคุมกระบวนการชดเชยมุมเฟสก้าวหน้า (Advance Angle Control Unit) เป็นหน่วยรับคำสั่งการเปลี่ยนค่าการชดเชยมุมเฟสก้าวหน้า ระหว่างกระแสกับแรงดันต้านกลับในชุดขดลวด Stator (ตามข้อ 2.7)จากหน่วยประเมินผลกลาง

3 หน่วยประเมินผลกลาง (Central Processing Unit) เป็นหน่วยที่คอยกับกับการทำงานของระบบทั้งหมดเข้าด้วยกัน เช่น คอยตรวจสอบสถานการณ์ทำงานของหน่วยควบคุมกระบวนการคอมมิวเตชั่น กำหนดค่าการชดเชยมุมเฟสก้าวหน้า และ เป็นตัวกลางในการประสานงานกับหน่วยวิเคราะห์ข้อมูลเป็นต้น

4 หน่วยวิเคราะห์ข้อมูลและแสดงผลการทำงานของมอเตอร์ (Graphic User Interface **Unit)** เป็นหน่วยที่ทำหน้าที่สั่งการและ รับค่าข้อมูลต่าง ๆ ของระบบเช่น ค่ากระแสโหลด (Load Current) แรงดันโหลด (Load Voltage) ความเร็วของมอเตอร์ (Motor Speed) และ แรงบิดโหลด (Load Torque) เป็นต้น และ นำค่าต่าง ๆ ในข้างต้น แสดงผลการทำงานของระบบในลักษณะที่เป็นรูปภาพเพื่อให้สะดวกในการศึกษาและการวิเคราะห์ข้อมูล

ทั้ง 4 ส่วนดังกล่าวในข้างต้นแสดงได้ตามรูปที่ 2.16



รูปที่ 2.16 ระบบควบคุมมอเตอร์ไฟฟ้ากระแสตรงไร้แปรงถ่านในย่านกำลังคงที่ด้วยหลักการชดเชยมุมเฟสก้าวหน้า

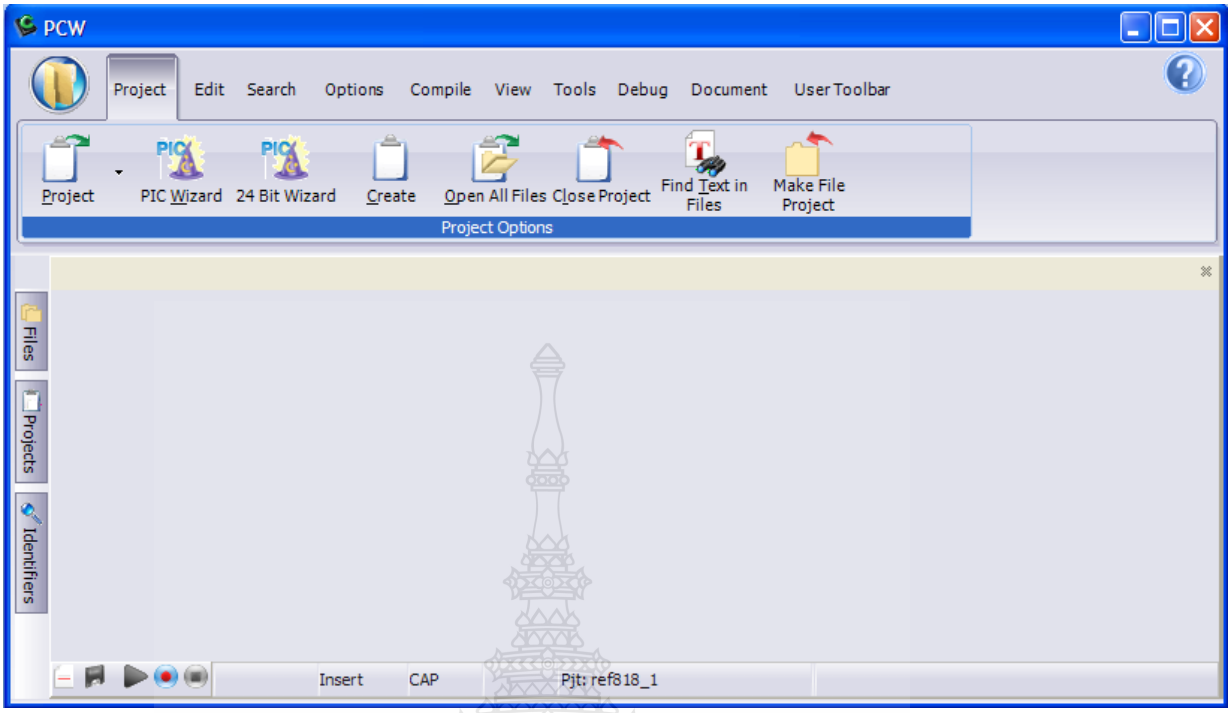
ในแต่ละส่วนของหน่วยควบคุมนั้นได้ ใช้เครื่องมือหลักๆ ในการสร้างที่ สำคัญๆ 2 ส่วนด้วยกันคือ

1 ส่วนของโปรแกรม(Software) โปรแกรม PROGRAM PIC _ C COMPILER ใช้ในการพัฒนาโปรแกรมในส่วนของ หน่วยควบคุมกระบวนการ อิเล็กทรอนิกส์คอมมิวเตชั่น(Electronic Comutation Unit) หน่วยควบคุมกระบวนการชดเชยมุมเฟสก้าวหน้า (Advance Angle Control Unit) และ หน่วยประเมินผลกลาง (Central Processing Unit) และ ใช้โปรแกรม Visual Basic ในการพัฒนาหน่วยวิเคราะห์ข้อมูลและแสดงผลการทำงานของมอเตอร์(Graphic User Interface Unit)

2 ส่วนของ แผงวงจรควบคุม เช่น แผงวงจรหน่วยควบคุมกระบวนการ อิเล็กทรอนิกส์คอมมิวเตชั่น (Electronic Comutation Circuit Board) แผงวงจรหน่วยควบคุมกระบวนการชดเชยมุมเฟสก้าวหน้าและหน่วยประเมินผล ซึ่งระบบต่าง ๆ ดังกล่าว Digital Signal Controller DSPIC30F2010เป็นตัวประเมินผลหลัก

2.8.1 การใช้ PROGRAM PIC _ C COMPILER V 4.084 [11]

PIC_C COMPILER เป็นเครื่องมืออีกตัวหนึ่งที่นำมาใช้ในวิทยานิพนธ์ เพื่อ ใช้เป็นตัวกลางในการแปลงคำสั่ง (Translators) ระหว่างภาษามนุษย์กับภาษาเครื่อง (Machine Language)ซึ่งในเวอร์ชั่น 4.084 ได้เพิ่มความสามารถและสิ่งอำนวยความสะดวกให้กับผู้ใช้งานอย่างมาก โดยสามารถใช้งานได้กับ pic microcontroller ได้ตั้งแต่ PIC12 , PIC16 PIC18 , PIC24 ไปจนถึง DSPIC และในส่วนของ user interface ก็หน้าตาดีมีการแบ่งออกเป็นหมวดหมู่ ทำให้ง่ายต่อการใช้งานเป็นอย่างมาก หน้าต่างของโปรแกรมแสดงดังรูปที่ 2.18



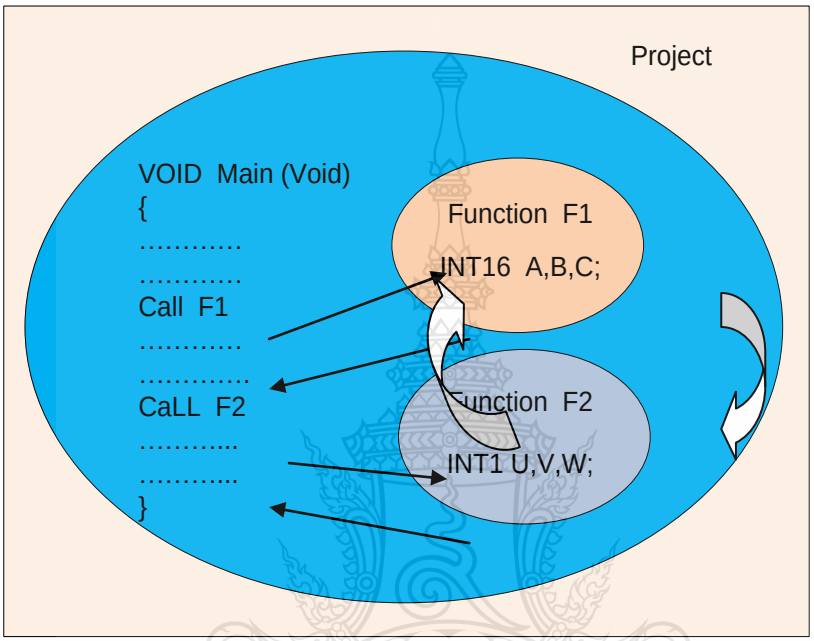
รูปที่ 2.17 User Interface Of PIC C V4.08

จากรูปที่ 2.17 แสดงถึงการแบ่งกลุ่มเมนูเป็นกลุ่มดังนี้

1) TAB project เป็นเมนูเริ่มต้นในการสร้างงาน หน้าที่หลักของเมนู project คือนำ source file (*.c) ซึ่งในแต่ละโปรเจ็กอาจจะประกอบด้วยหลาย source file แต่ต้องมี 1 source file ที่มีฟังชั่น (main) อยู่ใน project เนื่องจากงานที่สร้างขึ้นทั้งหมดจะถูกบัญชาการด้วยฟังชั่น (main)นั่นเอง
2) TAB Edit ใช้ในการแก้ไขปรับปรุงการเขียนโปรแกรมเป็นหลัก
3) TAB Search ใช้อำนวยความสะดวกในการค้นหาและ แทนที่คำใน project
4) ใช้ในการกำหนดคุณลักษณะต่าง ๆ ของโปรเจ็ก เช่น Editor Tollbar และ Project เป็นต้น
5) Tab Compile ใช้ในการตรวจสอบไวยากรณ์และแปรงโปรเจ็กเป็น HEX File
6) Tab view ใช้ในการเยี่ยมชมองค์ประกอบต่างต่าง ๆ ของโปรเจ็ก เช่น datasheet cpu , CPU Register การใช้ Preprocesser ต่าง ๆ เป็นต้น

ในการใช้งานโปรแกรม PIC _ C COMPILER จำเป็นต้องเข้าใจลักษณะโครงสร้างการทำงานของโปรแกรม รูปที่ 2.19 แสดงโครงสร้างการทำงานของ PIC _ C COMPILER โดยฟังชั่น (main) เป็นฟังชั่นหลักในการบริหารจัดการระบบงานใน project ทั้งหมด นั่นคือว่า sub function

ต่างๆ ที่อยู่ใน project จะถูกเรียกใช้งานโดย function main และระหว่าง sub function ก็สามารถเรียกใช้ซึ่งกันและกันได้ ในส่วนของตัวแปรถ้าประกาศไว้นอกฟังชั่นให้ถือว่าเป็นตัวแปรแบบ global (ยึดพื้นที่หน่วยความจำอย่างถาวร) ส่วนตัวแปรที่ประกาศในฟังชั่นจะเป็นตัวแปรแบบ private (ใช้พื้นที่หน่วยความจำเมื่อมีการเรียกใช้ฟังชั่นเท่านั้น)



รูปที่ 2.18 Structure pic c compiler

นอกจากการทำความเข้าใจกับโครงสร้างของโปรแกรม PIC _ C COMPILER แล้วยังมีสิ่งต่างๆ ที่จำเป็นดังนี้

- Pre-Processor
- Data Definition
- Function Definition
- Operators
- ส่วนควบคุมการทำงานของโปรแกรม (Process Control)
- Built-in-function
- การสร้าง Project
- Pre-Processor

Pre-processor ใช้สำหรับเตรียมความพร้อมให้กับ compiler ให้ทำความรู้จักกับองค์ประกอบต่างๆ ของ Project เช่น cpu (#include<30F2010.h>) การกำหนดฟังชั่นการทำงานให้กับ cpu(#Fuse NOWDT, NOPUT, PROTECT) และกำหนด clock speed ให้กับ cpu (#use delay(clock=20000000)) เป็นต้น

- Data Definition

การพัฒนาโปรแกรมในเรื่องใด ๆ ก็ตามการประกาศตัวแปรปันสิ่งที่หลีกเลี่ยงได้ยาก เนื่องจากงานด้านพัฒนาโปรแกรมเป็นการนำอินพุทข้อมูลที่มีการเปลี่ยนแปรงตามสภาวะต่างมาประเมินผลเพื่อให้ได้ค่าเอาท์พุทตามที่เราต้องการนั่นเอง ชนิดและวิธีการประกาศตัวแปร

ตารางที่ 2.2 ชนิดและวิธีการประใช้ตัวแปร

type	Number bit	value	signint
int	1	0 to 1	n/a
Int8	8	0 to 255	-128 to 127
Int16	16	0 to 65535	-32768 to 32767
Int32	32	0 to 4294967295	-2147483648 to 2147483647
Float32	32	-1.5×10^{45} to 3.4×10^{38}	

Constance Data

A =123; ฐานสิบ

B = O123; ฐานแปด

C = 0X0F; ฐานสิหก

D = 0B00110011; ฐานสอง

- Function Definition

รูปแบบการสร้างและการเรียกใช้ sub function มีดังนี้

- ประกาศฟังชั่น prototype ให้ compiler ทราบทุกครั้งก่อนการเรียกใช้ฟังชั่น
- กำหนดรูปแบบการรับและส่งค่าของฟังชั่นให้ถูกต้อง

ตัวอย่างโปรแกรมที่ประกอบด้วยฟังชั่นย่อยแบบต่าง ๆ

```
#include <30F2010.h>
#device adc=8
#Fuses  HS    #Fuses  BROWNOUT
#Fuses  NOMCLR
#Fuses  PUT4
#Fuses  NOWDT
#use delay(clock=10000000,restart_wdt)
#use rs232(UART1,baud=19200,parity=N,bits=8,)
void f1(void);
void f2(int8 data);
int16 f3(int8 a,int8 b);
void main(){
  int16 x;
   while(true)
  {
   f1();    f2(10);    f3(10,20);
  }
}
void f1(void)
{  output_b(0xff); }
 void f2(int8 data)
{ output_b(data);}
int16 f3(int8 a,int8 b)
{  return a+b;}
```

Function prototype

Call Function

- Operators

Operator ในโปรแกรม PIC C Compiler แบ่งออกได้เป็น 4 กลุ่มด้วยกันคือ

1 กลุ่มที่ใช้ดำเนินการทางคณิตศาสตร์(Arithmetic Operator)

เป็น Operator ใช้ในการคำนวณค่าทางคณิตศาสตร์

ตารางที่ 2.3 ตัวดำเนินการทางคณิตศาสตร์

+	Addition Operator
+=	Addition assignment operator, x += y ,is the same as x = x + y
++	Increment
--	Decrement
-=	Subtraction assignment operator, x -= y ,is the same as x = x - y
/	Division Operator
/=	Division assignment operator, x /= y ,is the same as x = x / y
*	Multiplication Operator
*=	Multiplication assignment operator, x *= y ,is the same as x = x * y

2 กลุ่มที่ใช้ดำเนินการทางลอจิก(Logic Operator)

เป็น Operator ที่ใช้ดำเนินการต่าง ๆ ทางลอจิก ดังแสดงในตารางที่ 2.4

ตารางที่ 2.4 Operator ที่ใช้ดำเนินการต่าง ๆ ทางลอจิก

&	Bitwise
&=	Bitwise assignment operator X &= y, is the same as x = x & y
^	Bitwise exclusive Opertor
|	Bitwise inclusive OR Operator
|=	Bitwise inclusive OR assignment Operator
&&	Logic AND Operator
||	Logic OR Operator
!	Logic Negation Operator
~	One Complement Operator

กลุ่มที่ใช้ดำเนินการเปรียบเทียบข้อมูล(Relational Operator)

ใช้สำหรับเปรียบเทียบข้อมูลซึ่งผลที่ได้จะเป็นค่า จริง(True) หรือ เท็จ(False)

ตารางที่ 2.5 กลุ่มที่ใช้ดำเนินการเปรียบเทียบข้อมูล(Relational Operator)

<==	Left shif assignment operator, x < == y , , is the same as x = x< y
<	Less than Operator
<<	Left Shif Operator
>>	Right Shif Operator
>>=	Right Shif assignment operator,x >>= y, is the same as x = x>> y
==	Equality
!=	Inequality

กลุ่มที่ใช้ดำเนินการ การเข้าถึงข้อมูลในหน่วยความจำข้อมูล(Memory Excess Operator)

เป็น Operator ที่ใช้ดำเนินการในการเข้าถึงข้อมูลในหน่วยความจำ

ตารางที่ 2.6 กลุ่มที่ใช้ดำเนินการ การเข้าถึงข้อมูลในหน่วยความจำ

*	Indirection Operator
&	Address Operator

- ส่วนควบคุมการทำงานของโปรแกรม (Process Control)

ส่วนที่ใช้ควบคุมลำดับการทำงานของโปรแกรมประกอบด้วยกลุ่มคำสั่ง 3 กลุ่มด้วยกันคือ กลุ่มคำสั่ง Branching , Iteration และ Condition

กลุ่มคำสั่ง Branching

เป็นกลุ่มคำสั่งที่ใช้ในการกระโดดข้ามการทำงานจากคำสั่งหนึ่งไปยังอีกคำสั่งหนึ่ง

ตารางที่ 2.7 กลุ่มคำสั่ง Branching

คำสั่ง	รูปแบบการใช้
Goto Label;	Goto loop;
Label : statement;	Loop :a++;
break	Break;
continue	Continue;

กลุ่มคำสั่ง Iteration

เป็นคำสั่งที่มีลักษณะการทำซ้ำ บ่อยครั้งที่โปรแกรมจำเป็นต้องทำงานที่ซ้ำๆ กันภายใต้เงื่อนไขใดๆ

ตารางที่ 2.8 กลุ่มคำสั่ง Iteration

คำสั่ง	รูปแบบการใช้
While(Expr){statement;}	While(x<10){x++;}
Do {statement;}while(Expr)	Do{x++;}while(x<10)
For(expr1;expr2;expr3){statment}	For(x=0;x<10;x++){printf("x");}

กลุ่มคำสั่ง Condition

ใช้กำหนดให้กลุ่มคำสั่งทำงานภายใต้เงื่อนไขที่กำหนดเป็นจริง

ตารางที่ 2.9 กลุ่มคำสั่ง Condition

คำสั่ง	รูปแบบการใช้
If(expr){statment}	If(x==10){x=0;}
If(expr) {statment} Else if(expr) {statement} Else{statment}	If(x==1) {y=1;} Else if(x==2) {y=3;} Else{y=4;}
Switch(expr) Case expr1:{statement} Case expr2:{statement} Default {statement}	Switch(x) Case 0:{y=2;break;} Case 1:{y=4;break;} Default {y=0;break;}

- Built-in-function

Built in function นับได้ว่าเป็นสิ่งที่ช่วยอำนวยความสะดวกกับผู้ใช้โปรแกรม pic c compiler เป็นอย่างมากโดยที่ผู้ใช้แค่ทำความเข้าใจกับการส่งผ่านค่าตัวแปรและ การ return ค่าของฟังชั่น

เท่านั้น ตัวอย่างที่ใช้บ่อยๆ เช่น Delay_ms (100) Delay_us (200) เป็นฟังชั้นที่ใช้สำหรับการหน่วงเวลาซึ่งมีความจำเป็นต้องใช้เกือบทุกโปรแกรม

- การสร้าง Project

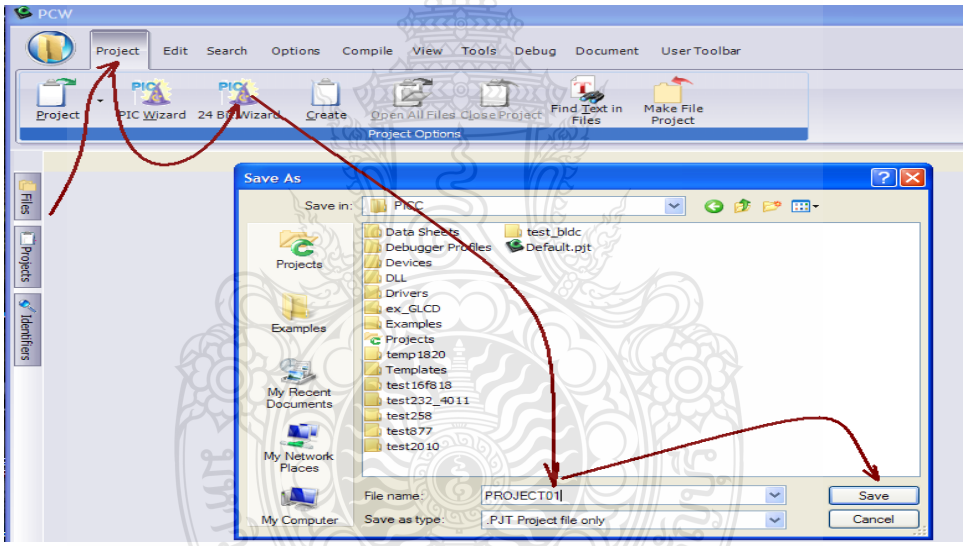
การสร้าง Project มีทางเลือกได้ 2 ทางดังนี้คือ

1) ใช้ Project Wizard

2) ใช้ Manual create Project

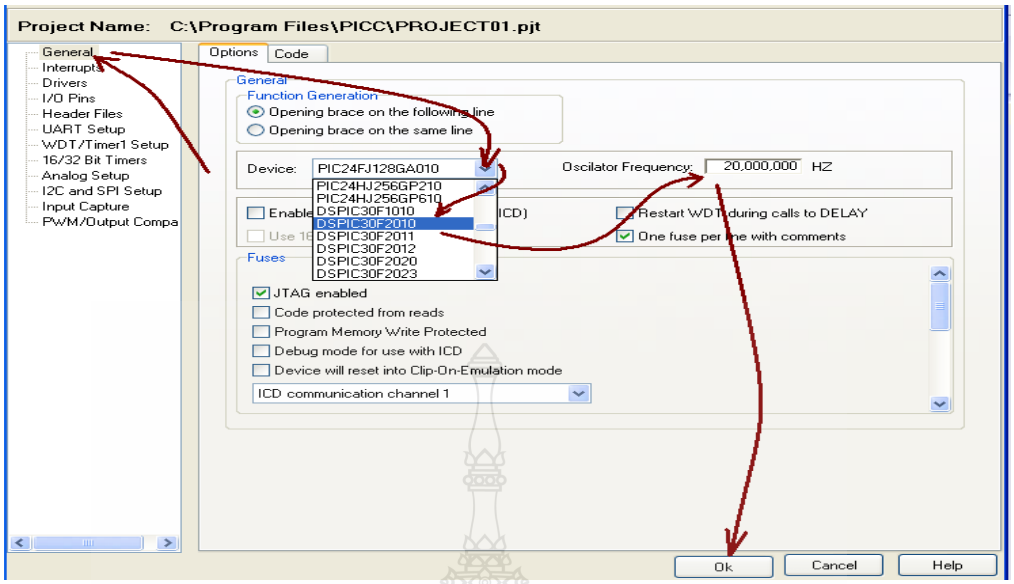
Project Wizard

นับได้ว่ามีประโยชน์ต่อผู้เริ่มต้นใช้โปรแกรมเป็นอย่างมาก เนื่องจาก pic microcontroller มี register ค่อนข้างมากทำให้ยากต่อการ config ในการนำไปใช้งานในแต่ละอย่าง และ project wizard จะช่วยนำทางมือใหม่สู่เป้าหมายได้ ขั้นตอนการใช้ Project wizard แสดงดังรูปที่ 2.20 และ 2.21



รูปที่ 2.19 เริ่ม project wizard Step 1

- เลือก Tab Project → Pic Wizard สำหรับ(8 bit databus) or 24 bit Wizard(16 bit data bus) → project name → save



รูปที่ 2.20 Step 2 register config

Pic c compiler จะแบ่งกลุ่มส่วนที่ผู้ใช้ต้องกำหนดค่าเริ่มต้นให้กับ compiler ไว้ใน list box ทางด้านซ้ายมือ เช่น ส่วนของ general,Interrupt,Driver,I/O Pin เป็นต้นซึ่งการกำหนดค่าในแต่ละส่วนนั้นจำเป็นต้องทำความเข้าใจโครงสร้างและฟังชั่นการทำงานอย่างละเอียด หลังจากที่เราเข้าไปกำหนดส่วนต่างๆ เรียบร้อยแล้ว click ok โปรแกรมจะสร้าง source code ส่วนหนึ่งมาใส่ไว้ในฟังชั่นหลักเพื่อกำหนดการทำงานเริ่มต้นให้กับ CPU เพื่อพร้อมที่จะใช้งานตามความต้องการต่อไป

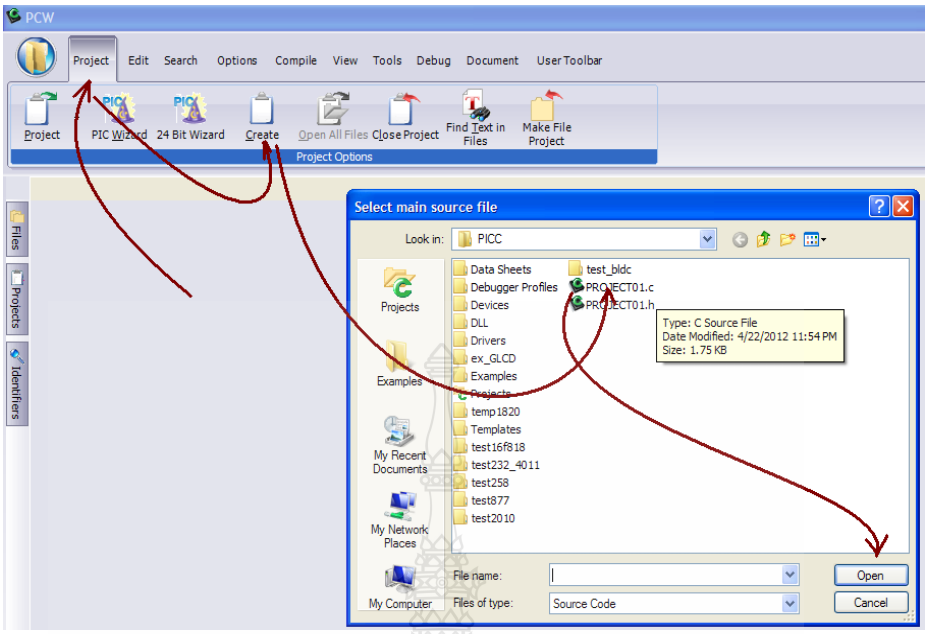
ตัวอย่าง code ได้จากการ wizard

```
//  #include "C:\Program Files\PICC\PROJECT01.h"

#include <30F2010.h>
#FUSES NOWDT                    //No Watch Dog Timer
#FUSES HS                       //High speed Osc (> 4mhz for PCM/PCH) (>10mhz for PCD)
#FUSES PR                       //Primary Oscillator
#FUSES NOCKSFSM                 //Clock Switching is disabled, fail Safe clock monitor is disabled
#FUSES WPSB16                   //Watch Dog Timer PreScalar B 1:16
#FUSES WPSA512                  //Watch Dog Timer PreScalar A 1:512
#FUSES PUT64                    //Power On Reset Timer value 64ms
```

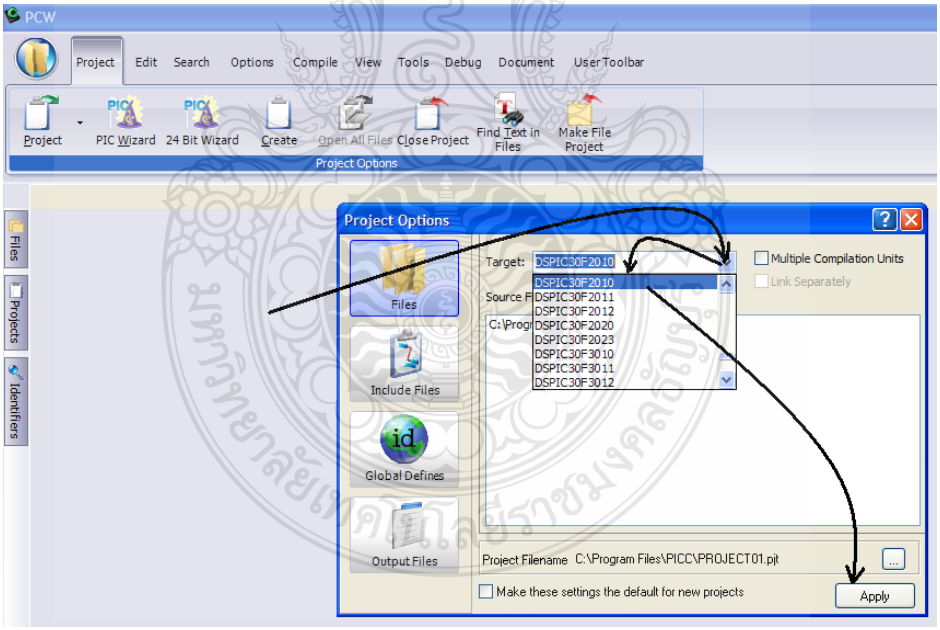
```
#FUSES NOBROWNOUT          //No brownout reset
#FUSES BORV47              //Brownout reset at 4.7V
#FUSES LPOL_HIGH           //Low-Side Transistors Polarity is Active-High (PWM 0,2,4 and 6)
                           //PWM module low side output pins have active high output polar
#FUSES HPOL_HIGH           //High-Side Transistors Polarity is Active-High (PWM 1,3,5 and 7)
                           //PWM module high side output pins have active high output
                           polarity
#FUSES NOPWMPIN            //PWM outputs drive active state upon Reset
#FUSES MCLR                //Master Clear pin enabled
#FUSES NOPROTECT           //Code not protected from reading
#FUSES NOCOE               //Device will reset into operational mode
#FUSES ICS0                //ICD communication channel 0
#FUSES RESERVED            //Used to set the reserved FUSE bits
#use delay (clock=10000000)
#use rs232 (UART1,baud=9600,parity=N,bits=8)
                           // TODO: USER CODE!!}
```

Manual Create Project

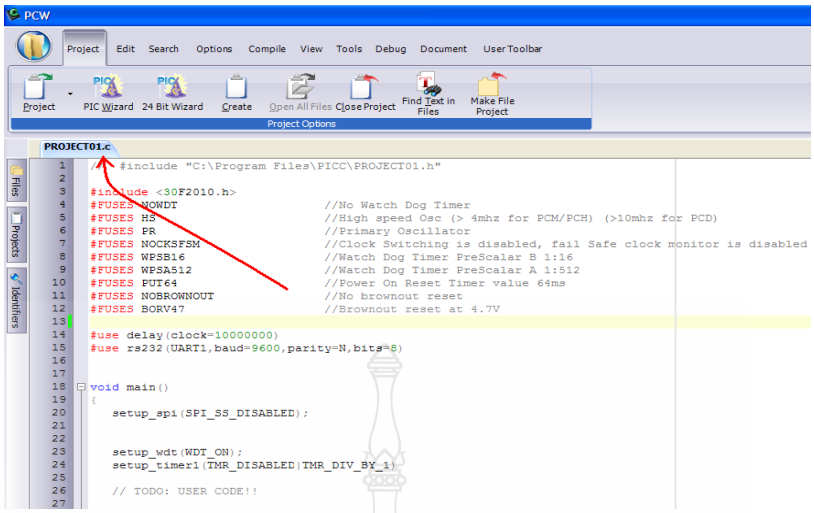
เป็นวิธีการนำ source file ที่ถูกสร้างขึ้นมาแล้ว มาประกอบรวมเป็น project อย่างไรก็ตาม จะต้องมี (function main) อยู่ใน source file ด้วยเนื่องจากกฎเกณฑ์ของภาษาซีที่ project ใดๆ ก็ตาม จะต้องใช้ (function main) ในการบริหารจัดการกับ project ทั้งหมด ขั้นตอนการ manual create project แสดงให้เห็นดังรูปที่ 2.22 2.23 และ 2.24 ตามลำดับ



รูปที่ 2.21 step 1 เส้นทางสู่ การเลือก source file

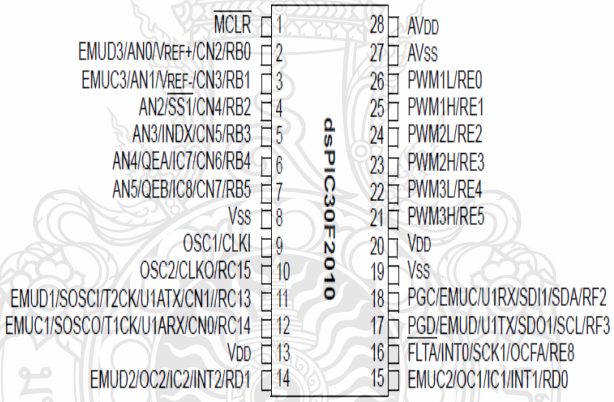


รูปที่ 2.22 เลือก Device

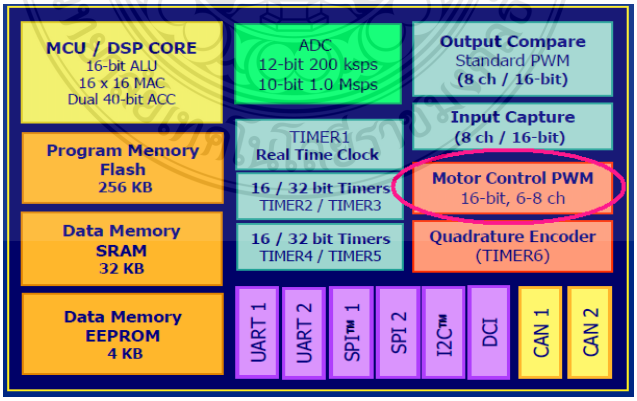


รูปที่ 2.23 การ manual create project เสร็จสมบูรณ์

2.8.2 สถาปัตยกรรมและการใช้โมดูล Power Control PWM ใน DSPIC30F2010 [12]



[ก]



[ข]

รูปที่ 2.24 ก) ลักษณะทางกายภาพของ DSPIC30F2010 ข)องค์ประกอบภายใน DSPIC30F2010

องค์ประกอบภายในของ DSPIC30F2010 ได้ออกแบบมาเพื่อรองรับกับระบบควบคุม โดยเฉพาะซึ่งองค์ประกอบดังกล่าวพอจะแบ่งออกเป็นกลุ่มๆ คือ

- กลุ่มที่ใช้ในการจัดเก็บลำดับคำสั่ง หน่วยประเมินผลผลและการบริหารจัดการข้อมูล เช่น MCU/DSP CORE,Program Memory,Data Memor และ Data Memory EEPROM เป็นต้น

- กลุ่มที่ใช้จัดการกับเรื่องของจำนวนนับและเวลา เช่น Timer1,Timer2,Timer3,Output Capture และ Input Capture เป็นต้น

- กลุ่มที่ใช้สำหรับเปลี่ยน สัญญานอนาล็อกเป็นดิจิตอล คือ ADC

- กลุ่มที่ใช้ในการติดต่อสื่อสารกับอุปกรณ์ภายนอก เช่น UART1,UART2,SPI1,SPI2,I2C, CAN1 และ CAN2 เป็นต้น

- กลุ่มที่ใช้ในการออกควบคุมการทำงานของมอเตอร์ เช่น Motor Control PWM และ Quadrature Encoder ซึ่งเป็นส่วนสำคัญในการนำมาใช้ประโยชน์ในงานวิจัย ซึ่งจะอธิบายรายละเอียดใน ลำดับถัดไป

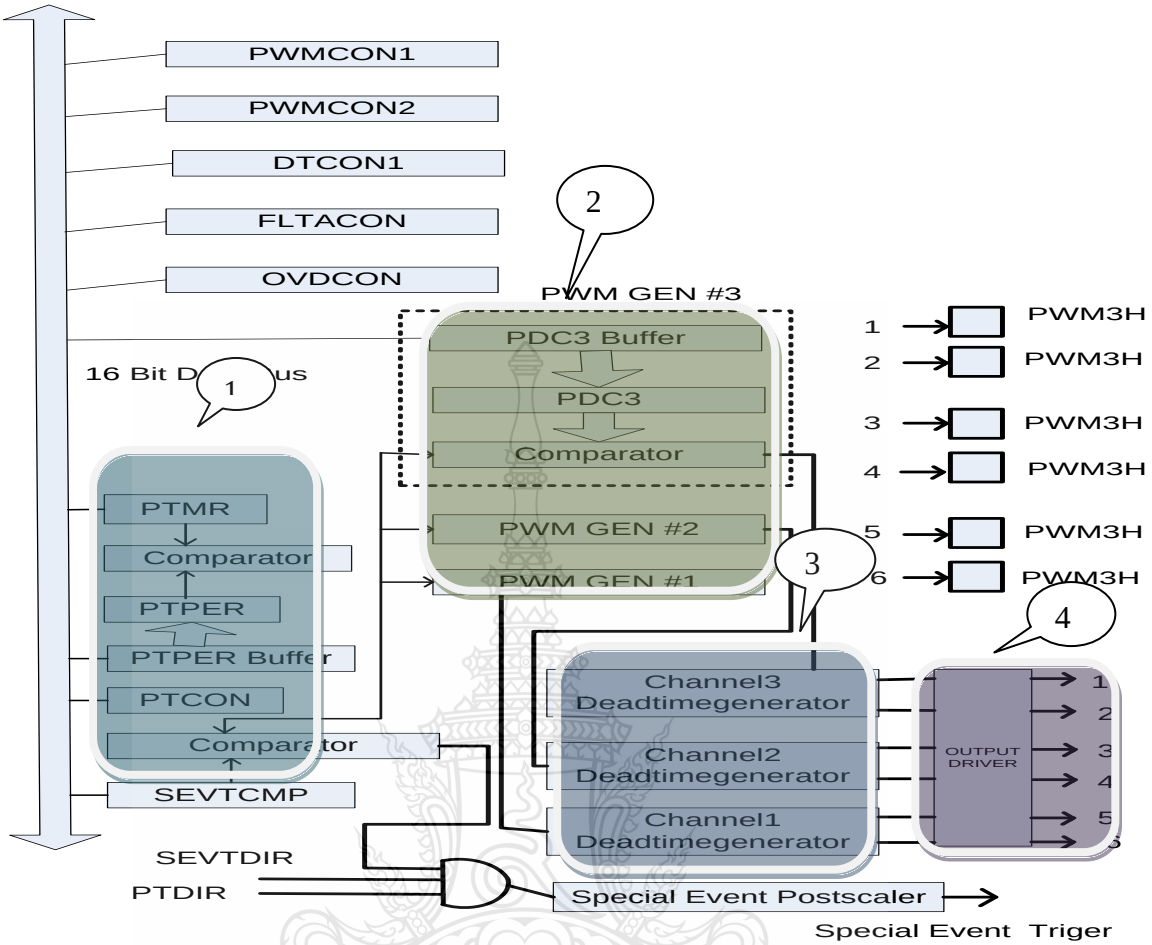
สำหรับในวิทยานิพนธ์นี้จะมุ่งไปที่การใช้โมดูล Motor Control PWM เป็นหลักซึ่งการที่จะใช้โมดูลดังกล่าว ได้อย่างมีประสิทธิภาพนั้นจำเป็นอย่างยิ่งที่จะต้องเข้าใจภาพรวมการทำงานของโมดูลให้มากที่สุด จากรูปที่ 2.26 จะเห็นได้ว่ามีการแบ่งการควบคุมออกเป็น 4 ส่วนด้วยกันคือ

1 ส่วนของการกำหนดความถี่ของสัญญาณ PWM ประกอบด้วย REGISTER PTMR, PTPER,PTCON เป็นต้น

2 ส่วนของการกำหนด DUTY CYCLE คือ PDC1,PDC2 และ PDC3

3 ส่วนของการกำหนดค่า DEAD TIME และ DISABLE OR ENABLE สัญญาณ PWM

4 ส่วนของ BUFFER AND DRIVE สัญญาณ PWM



รูปที่ 2.25 Block Diagram of Motor Control PWM

โดยส่วนสำคัญในการเรียนรู้เพื่อสร้างสัญญาณ PWM แบ่งออกได้ดังนี้คือ

- หน้าที่และการใช้งาน Register
- การเปลี่ยนแปลง Duty Cycle ของสัญญาณ PWM แบบ Free running mode
- การเปลี่ยนแปลง Duty Cycle ของสัญญาณ PWM แบบ Up/Down Counting Modes
- การกำหนดค่า DEAD TIME ให้กับคู่สัญญาณ PWM ที่เป็น Complementary
- การใช้ INPUT FAULT PINS ในการป้องกันความผิดปกติในภาคขยายกำลังไฟฟ้า

Register PWMCON1

Register PWMCON1 ใช้กำหนดโหมดการทำงานของคู่สัญญาณ PWM_{XH} PWM_{Xl} และใช้เป็นประตูปิดเปิดของสัญญาณ PWM

ตารางที่ 2.10 Register PWMCON1 Upper Byte (bit8 – bit15)

U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
-	-	-	-	PMOD4	PMOD3	PMOD2	PMOD1
Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8

ตารางที่ 2.11 Register PWMCON1 Lower Byte (Bit0 – Bit7)

U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
PEN4H	PEN3H	PEN2H	PEN1H	PEN4L	PEN3L	PEN2L	PEN1L
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

BIT 15-12 Unimplement : read as 0

BIT 11-18 PMOD4:PMOD1: PWM I/O Pair Mode Bit

1 = PWM I/O pin pair is in the independent mode

0 = PWM I/O pin pair is in the complementary mode

BIT 7 – 4 PEN4H – PEN4L: PWM I/O Enable Bit

1 = PWMxH pin is enable for PWM output

0 = PWMxH pin disable I/O pin become general perpose I/O

BIT 3 – 0 PEN4L-PEN1L: PWMxL I/O Enable bit

1 = PWMxL pin is enable for PWM output

0 = PWMxL pin disable I/O pin become general perpose I/O

Register PWMCON2

Register PWMCON2 ใช้กำหนดการทำงานของสัญญาณ PWM ออกเป็น 3 กลุ่มด้วยกัน คือ

1 ใช้แบ่งช่วงเวลาการสร้างสัญญาณ PWM ในหนึ่งคาบเวลา

2 ใช้ควบคุมการอนุญาตการเปลี่ยนแปลงค่า Duty Cycle ของสัญญาณ PWM

3 ใช้ควบคุมความสัมพันธ์ ระหว่างหนึ่งคาบเวลากับสัญญาณ PWM

ตารางที่ 2.12 Register PWMCON2 Upper Byte (bit8 – bit15)

U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
-	-	-	-	SEVOPS<3:0>			
Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8

ตารางที่ 2.13 Register PWMCON2 Lower Byte (bit0 – bit7)

U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
-	-	-	-	-	IUE	OSYNC	UDIS
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

BIT 15 – 12 Unimplemented read as '0'

BIT 11 – 8 SEVOPS<3:0>:PWM Special Event trigger output Post scale Select Bit

1111 = 1 : 16 post scale

0001 = 1 : 2 post scale

0000 = 1 : 1 post scale

BIT 7 – 2 Unimplemented : Read as '0'

BIT 2 IUE: Immediate Update Enable bit

1 = Update to the active PDC register immediate

0 = Update to the active PDC registers are synchronize to the PWM time base

BIT 1 OSYNC : Output Override Synchronization bit

1 = Output override via the OVDCON register are synchronized to the PWM time base

0 = Output override 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 disable

***Register DTCON1* :**

Register DTCON1 มีหน้าที่ในการกำหนดค่า Dead Time ให้กับคู่สัญญาณ PWM_{XH}, PWM_{XL}

ตารางที่ 2.14 Register **DTCON1** Upper Byte (bit8 – bit15)

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>					
Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8

ตารางที่ 2.15 Register **DTCON1** Lower Byte (bit0 – bit7)

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>					
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

BIT 15-14 DTPS<1:0> Dead Time Unit B Prescale select bit

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 for Dead Time Unit B

BIT 7 – 6 <1:0> Dead Time Unit A Prescale select bit

BIT 5 – 0 : Unsigned 6 – bit Dead Time value for Dead Time Unit A

Register FLTACON :

Register FLTACON ใช้กำหนดการหยุดการทำงานของสัญญาณ PWM กรณีเกิดความผิดปกติอย่างใดอย่างหนึ่ง เช่น กรณี Over Load เป็นต้น

ตารางที่ 2.16 Register FLTACON Upper Byte (bit8 – bit15)

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
Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8

ตารางที่ 2.17 Register FLTACON Lower Byte (bit0 – bit7)

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
Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit1	Bit0

BIT 15 – 8 FAOV4H – FAOV1L : Fault input A PWM Override Value

1 = The PWM Output pin is driven ACTIVE on an external input

0 = The PWM Output pin is driven INACTIVE on an external input

BIT 7 FLTAM : Fault a Mode bit

1 = The fault A input pin Function in the cycle by cycle Mode

0 = The fault A input pin latches all control pins to the program states in FLTA<15:8>

BIT 6 – 4 Unimplement read as '0'

BIT 3 FAEN: 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 FAEN: 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 FAEN: 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 FAEN: 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

Register FLTBCON :

Register FLTBCON ทำหน้าที่เหมือนกันกับ Register FLTACON (กรณีไมโครคอนโทรลเลอร์มีขารับสัญญาณอินพุท Fault 2 ช่อง)

ตารางที่ 2.18 Register FLTBCON Upper Byte (bit8 – bit15)

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
Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8

ตารางที่ 2.19 Register FLTBCON Lower Byte (bit0 – bit7)

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
Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit1	Bit0

BIT 15 – 8 FBOV4H – FBOV1L : Fault input A PWM Override Value

1 = The PWM Output pin is driven ACTIVE on an external input

0 = The PWM Output pin is driven INACTIVE on an external input

BIT 7 FLTBM : Fault a Mode bit

1 = The fault A input pin Function in the cycle by cycle Mode

0 = The fault A input pin latches all control pins to the program states in FLTA<15:8>

BIT 6 – 4 Unimplement read as '0'

BIT 3 FBEN: Fault input a enable bit

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 FBEN: Fault input a enable bit

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 FBEN: Fault input a enable bit

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 FBEN: Fault input a enable bit

1 = PWM1H/PWM1L Pin pair is controlled by fault input B

0 = PWM1H/PWM1L Pin pair is not controlled by fault input B

Register PTCON :

Register PTCON ใช้กำหนดฐานเวลาให้กับสัญญาณ PWM และเป็นประตูปิดเปิดสัญญาณนาฬิกาให้กับกระบวนการสร้างสัญญาณ PWM

ตารางที่ 2.20 Register PTCON Uper Byte (bit8 – bit15)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTEN	-	PTSIDL	-	-	-	-	-
Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8

ตารางที่ 2.21 Register PTCON Lower Byte (bit0 – bit7)

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		PTMOD	
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

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 is ‘0’

BIT 13 PTSIDL : PWM Time Base is off in idle mode bit

1 = PWM Time Base halts in CPU Idle Mod

0 = PWM Time Base Run in CPU Idle Mod

BIT 12 – 8 Unimplemented read is ‘0’

BIT 7 – 4 PTOPS<3:0> PWM Time Base Output Postscale Select bits

1111 = 1 : 16 postscale

↕

0001 = 1 : 2 postscale

00 = 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 16Tcy (1 : 16 prescale)

01 = PWM Time Base input clock period is 4 Tcy (1 : 4 prescale)

00 = PWM Time Base input clock period is 1 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 update

01 = PWM Time Base operates in a continuous up/down counting mode

10 = PWM Time Base operates in single event mode

00 = PWM Time Base operates in a free running mode

Register PTMR

Register PTMR ใช้เป็นตัวนับค่าและเก็บค่าของสัญญาณนาฬิกาในหนึ่งรอบการทำงาน (Period)ของสัญญาณ PWM

ตารางที่ 2.22 Register PTMR Uper Byte(bit8 – bit15)

R/W-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>						
Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8

ตารางที่ 2.23 Register PTMR Lwer Byte(bit0 – bit7)

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:>							
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

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

Register PTPER

Register PTPER ใช้กำหนดค่าความถี่(Time Period)ให้กับสัญญาณ PWM

ตารางที่ 2.24 Register PTPER Uper Byte(bit8 – bit15)

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<14:8>						
Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8

ตารางที่ 2.25 Register PTPER Lower Byte(bit0 – bit7)

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:>							
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

BIT 15 Unimplemented read as '0'

BIT 14 – 0 PTPER<14:0> PWM Time Base Period value bits

Register OVDCON :

Register OVDCON ใช้กำหนดรูปแบบการส่งสัญญาณ PWM ออกทาง Output

ตารางที่ 2.26 Register PTPER Uper Byte(bit8 – bit15)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
POVD4H	POVD4L	POVD3H	POVD3L	POVD2H	POVD2L	POVD1H	POVD1L
Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8

ตารางที่ 2.27 Register PTPER Lower Byte(bit0 – bit7)

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
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

BIT 15 – 8 POVD4H – POVD1L: PWM Output Override bits

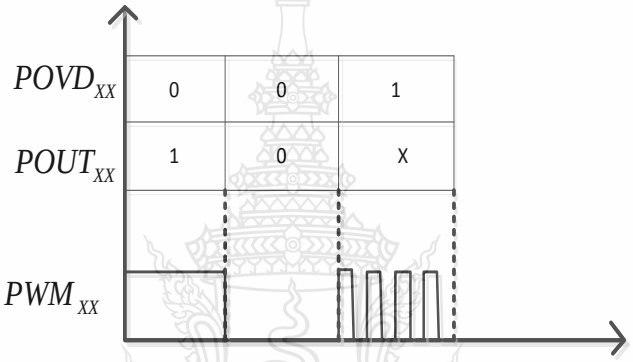
1 = Output on PWMxx I/O pin is controlled by the PWM generator

0 = Outpu: 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



รูปที่ 2.26 ตารางแสดงความสัมพันธ์ระหว่าง POVDxx และ POUTxx

Register PDC1 PDC2 PDC3

Register PDC1 PDC2 PDC3 ใช้กำหนดค่า Duty Cycleให้แต่ละคู่สัญญาณ PWM

ตารางที่ 2.28 Register PDC1 PDC2 PDC3 Uper Byte(bit8 – bit15)

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<15:8>							
Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8

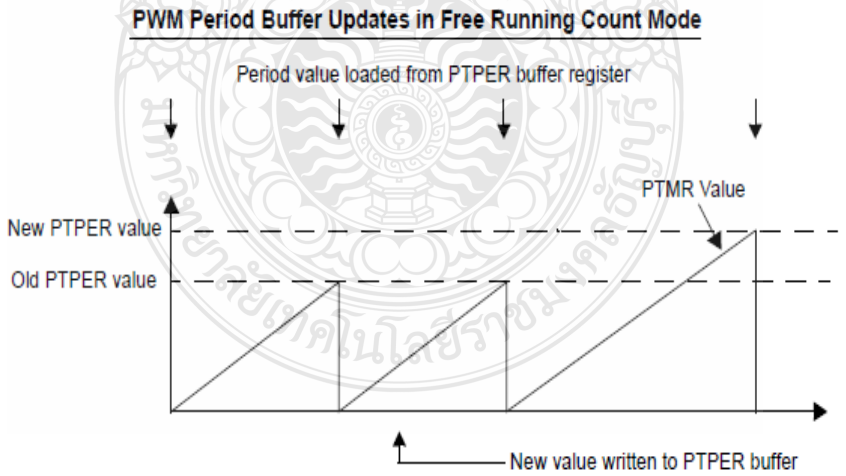
ตารางที่ 2.29 Register PDC1 PDC2 PDC3 Lower Byte(bit8 – bit15)

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<7:0>							
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

ตารางที่ 2.30 ADDRESS ของ REGISTER ที่เกี่ยวข้องกับ MOTOR CONTROL PWM MODULE

REGISTER	ADDRESS
PTCON	0X01C0
PTMR	0X01C2
PTPER	0X01C4
PWMCON1	0X01C8
PWMCON2	0X01CA
DTCON	0X1CC
FLTA	0X1D0
FLTB	-
PDC1	0X01D6
PDC2	0X01D8
PDC3	0X01DA

การกำหนดคาบเวลาสัญญาณ PWM ที่นิยมใช้กันมีอยู่ 2 แบบ คือ Free running และ Up/Down Counting Modes ดังแสดงดังรูปที่ 2.28 และ 2.30 ตามลำดับ



รูปที่ 2.27 การเปลี่ยนแปลงคาบเวลาของสัญญาณ PWM ที่ขึ้นอยู่กับค่าใน Register PTER ในโหมด Free running

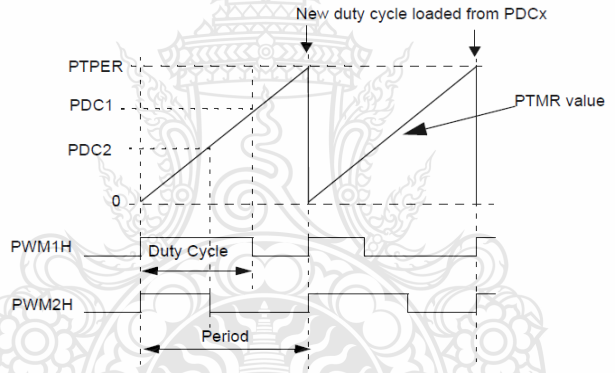
การคำนวณคาบเวลาของสัญญาณ PWM ในโหมด Free Running mode เป็นไปตามสมการที่ 2.17

$$PTPER = \frac{Fcy}{F_{pwm}.(PTMR_{PRESCALE})} - 1 \qquad (2.17)$$

ตัวอย่าง การคำนวณค่า PTPER ในโหมด Free running

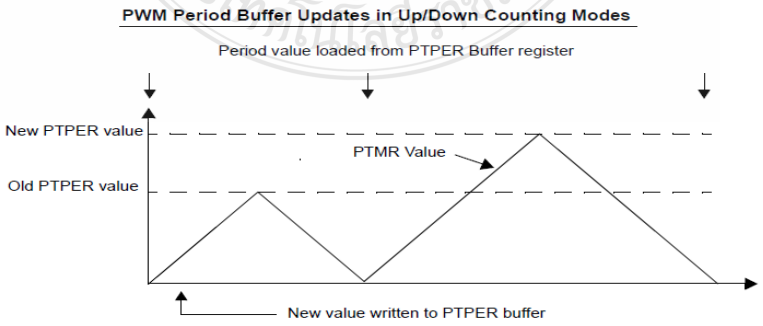
$$F_{CY} = 20\,MHZ$$
$$F_{PWM} = 20,000\,HZ$$
$$PTMR_{PRESCALE} = 1:1$$
$$PTPER = \frac{20,000,000}{20,000.1} - 1$$
$$= 1000 - 1$$
$$= 999$$

การเปลี่ยนแปลง DUTYCYCLE ของสัญญาณ PWM ในโหมด Free Running mode



รูปที่ 2.28 ความสัมพันธ์ของ REGISTER PTPER,PTMR,PDC ในการกำหนด DUTY CYCLE ของสัญญาณ PWM

การเปลี่ยนแปลง DUTYCYCLE ของสัญญาณ PWM ในโหมด Up/Down Counting Modes



รูปที่ 2.29 ความถี่ หรือ คาบเวลาของสัญญาณ PWM ที่ขึ้นอยู่กับค่าใน Register PTER

$$PTPER = \frac{Fcy}{F_{pwm}.(PTMR_{PRESCALE}).2} - 1 \qquad (2.18)$$

ตัวอย่าง การคำนวณค่า PTPER ในโหมด Up/Down Counting

$$F_{CY} = 20\,MHZ$$

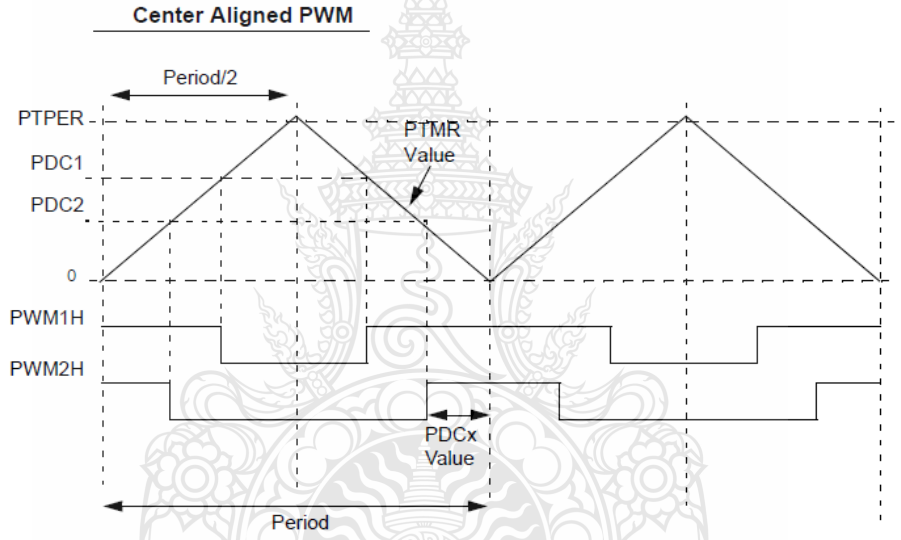
$$F_{PWM} = 20,000\,HZ$$

$$PTMR_{PRESCALE} = 1:1$$

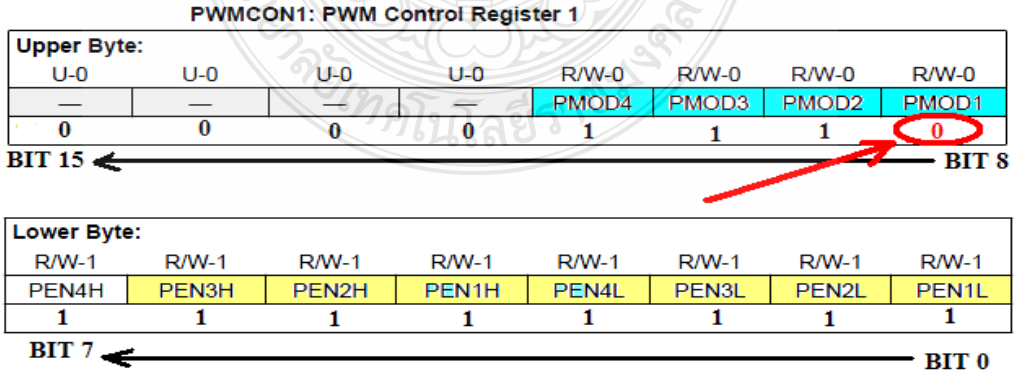
$$PTPER = \frac{20,000,000}{20,000.1.2} - 1$$

$$= 500 - 1$$

$$= 499$$



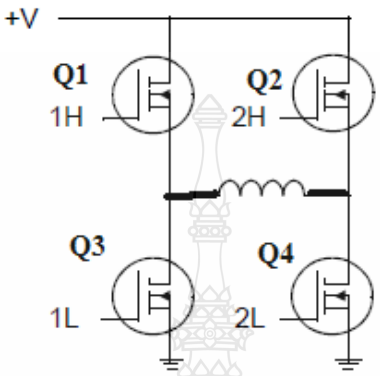
รูปที่ 2.30 ความสัมพันธ์ของ Register PTPER,PTMR,PDC ในการกำหนด Duty Cycle



รูปที่ 2.31 PMOD1 = 0 PWM1 ทำงานเป็น Complementary

การกำหนดค่า DEAD TIME ให้กับคู่สัญญาณ PWM ที่เป็น Complementary

คู่สัญญาณ PWM ที่เป็น Complementary จำเป็นที่จะต้องพิจารณาในเรื่องของค่า DEAD TIME เนื่องจากข้อจำกัดทางเวลา TURN OFF ของอุปกรณ์ ELECTRONIC SWITCHING ที่อยู่ในวงจรดังรูปที่ 2.33



รูปที่ 2.32 วงจร Switching ที่จำเป็นต้องมีค่า Dead Time

ตัวอย่างโปรแกรมในการกำหนดค่า DEAD TIME

```
void main(void)
{
 ptper=0x007D;   ptmr=0x1ff;  PWMCON1=0X00FF;   PWMCON2=0X0002;
 PTCON=0X8000;
 OVDCON=0XFF00;
 PDC1=PDC2=PDC3=100;
 DTCON1=0X00FF;
  while(true)  {  }}
```

กำหนดค่า DEAD

การคำนวณค่า Dead Time

$$DT = \frac{Dead\ Time}{\text{Pr}escale\ value.T_{cy}} \qquad (2.19)$$

โดยค่าของ DT กำหนดโดย DTA<5:0> หรือ DTB<5:0> ใน register DTCON

ตัวอย่าง การคำนวณค่า DEAD TIME จากโปรแกรม DTCON1=0X00FF;ดังแสดงในรูปที่ 2.23

DTCON1: Dead Time Control Register 1

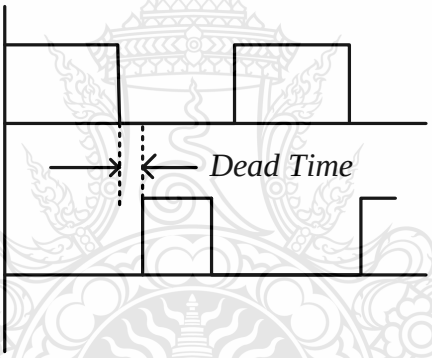
Upper Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
0	0	0	0	0	0	0	0

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
1	1	1	1	1	1	1	1

รูปที่ 2.33 ค่าใน Register DTCON1

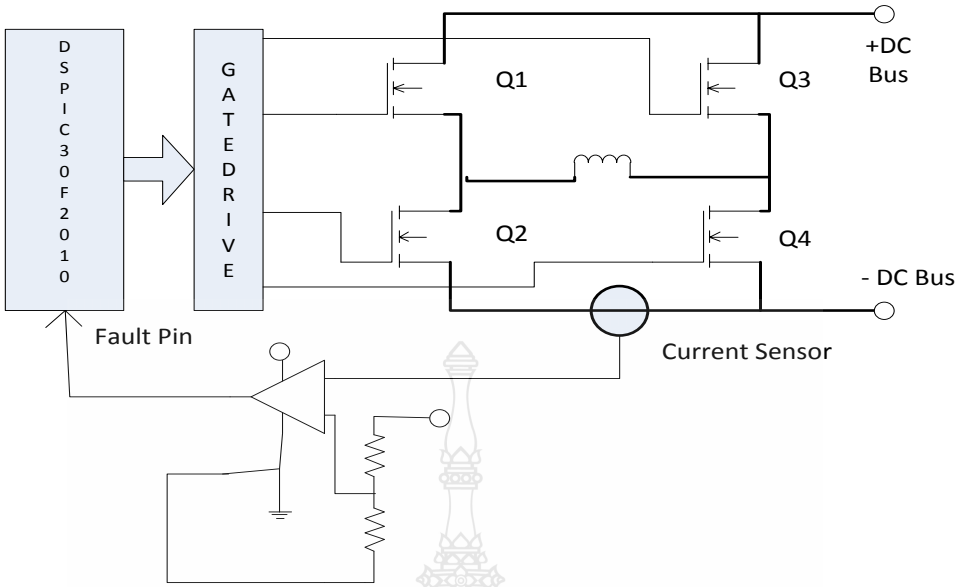
DT = 111111b = 64d ; Prescale = 8 ; TCY = 20000000/4 = 5000000

Dead Time = (1/5000000) x 8 x 64 = 102 usec



รูปที่ 2.34 ค่า Dead Time

การใช้ INPUT FAULT PINS ในการป้องกันความผิดปกติในภาคขยายกำลังไฟฟ้า INPUT FAULT PINS เป็นที่ใช้รองรับการพัฒนาโปรแกรมในการตรวจสอบความผิดปกติในส่วนของภาคขยายกำลัง เช่น กระแสเกินพิกัด หรือ เกิดการลัดวงจร เป็นต้น



รูปที่ 2.35 การใช้งาน FAULT PIN

ตัวอย่างโปรแกรม การใช้งาน FAULT PIN

```
#include <30f2010.h>
#device adc=8
#Fuses XT_PLL16
#Fuses  BORV27
#fuses  PUT64
#fuses BROWNOUT
#FUSES MCLR
#use delay(clock=160000000,restart_wdt)
#use rs232(UART1,baud=19200,parity=N,bits=8,)
int16 PTCON;        #locate PTCON = 0x1C0
INT16 PTMR;        #LOCATE  PTMR = 0X1C2
INT16 PTPER;       #LOCATE PTPER = 0X1C4
INT16 SEVTCPP;   #LOCATE SEVTCPP = 0X1C6
INT16 PWMCON1; #locate PWMCON1 = 0x1c8
INT16 PWMCON2;   #locate PWMCON2 = 0x1ca
```

```
INT16 DTCON1; #LOCATE DTCON1 = 0X1CC
INT16 FLTACON;
#LOCATE FLTACON = 0X1D0
INT16 OVDCON;  #LOCATE OVDCON  =  0X1D4
INT16 PDC1;   # INT16 PDC2;
#LOCATE PDC2 = 0X1D8
INT16 PDC3;   #LOCATE PDC3 = 0X1DA
void main(void)
{
 ptper=0x0080;
 PWMCON2=0X0F00;   PWMCON1=0X00FF;
 PWMCON2=0X0F00;   PTCON=0X8000;
 OVDCON=0XFF00;  PDC1=PDC2=PDC3=0;
 FLTACON=0x000F;
 while(true)
 {  restart_wdt();
 }}
```

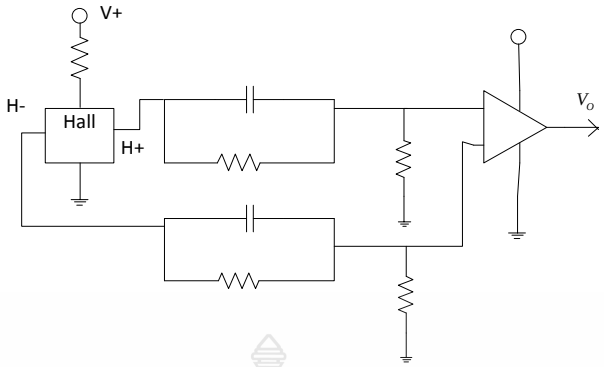
ประกาศตัวแปรเพื่อการเข้าถึงข้อมูลใน Register

กำหนดการทำงานให้กับ Input Fault Pin(FLTA)

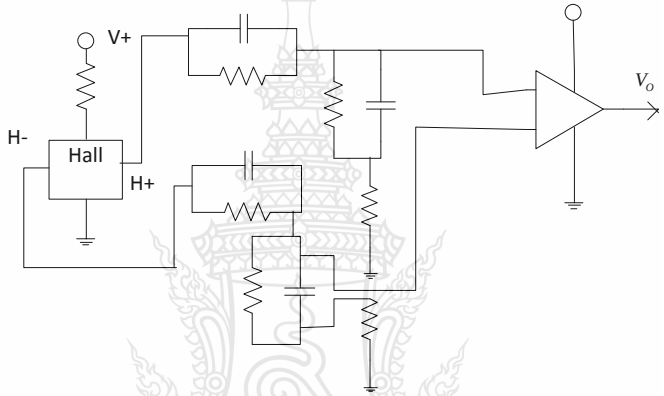
2.9 เอกสารงานวิจัยที่เกี่ยวข้อง

[1] Chun-Lung Chiu□, Yie-Tone Chen_, Yu-Hsiang Shen_, and Ruey-Hsun Liang **"An Accurate Automatic Phase Advance Adjustment of Brushless DC Motor"** Department of Electrical Engineering, National Yunlin University of Science and Technology

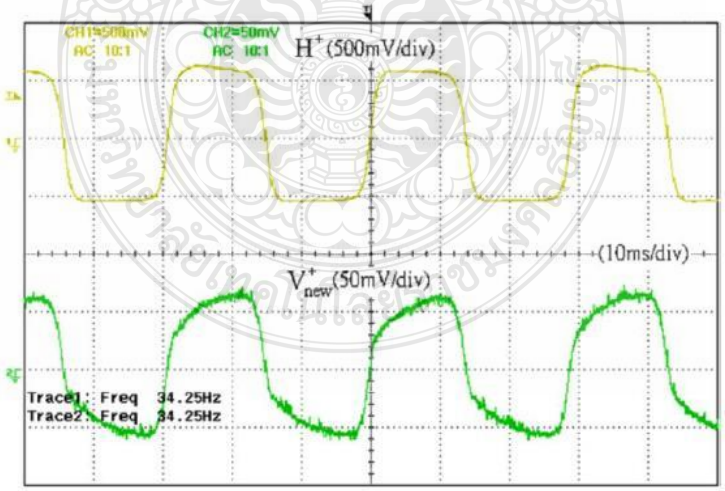
นำเสนอการเพิ่มประสิทธิภาพ และ แรงบิดของ Brushless DC Motor โดยการสร้างวงจรส่วนชดเชยมุมเฟสก้าวหน้า (Phase Advance Circuit)โดยทำการแยกฮาร์มอนิกจากสัญญาณของ ตัวตรวจจับการเปลี่ยนแปลงของสนามแม่เหล็ก (Hall Sensor Signal) เพื่อให้สัญญาณเอาท์พุทของ Hall Sensor ที่ได้มีการชดเชยมุมเฟสก้าวหน้าที่เหมาะสมในทุกๆ ย่านความเร็วของมอเตอร์



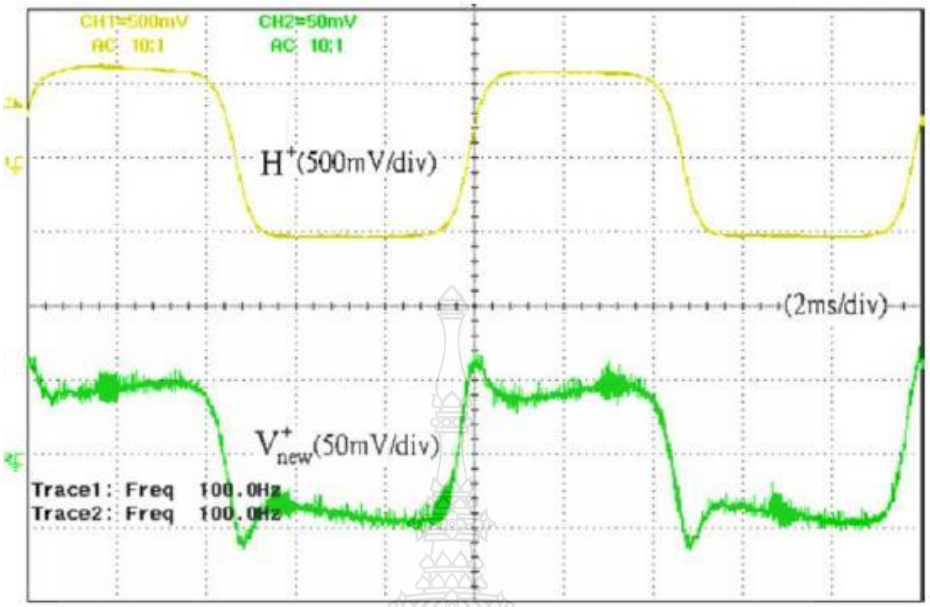
รูปที่ 2.36 วงจรชดเชยมุมเฟสก้าวหน้าโดยทั่วไป (Conventional phase advance circuit)



รูปที่ 2.37 วงจรชดเชยมุมเฟสก้าวหน้าที่นำเสนอ (Proposed phase advance circuit)



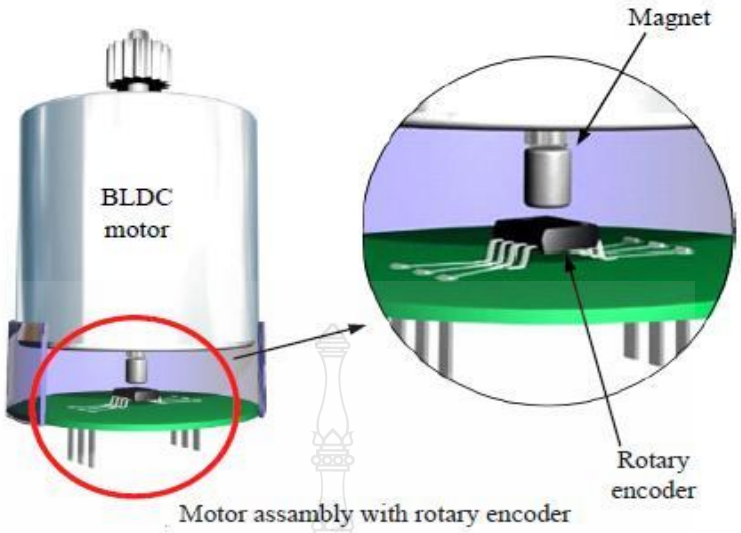
รูปที่ 2.38 ผลการทดลองสัญญาณเอาท์พุทของ Hall Sensor ที่ได้จากวงจรชดเชยมุมเฟสก้าวหน้าทั่วไป



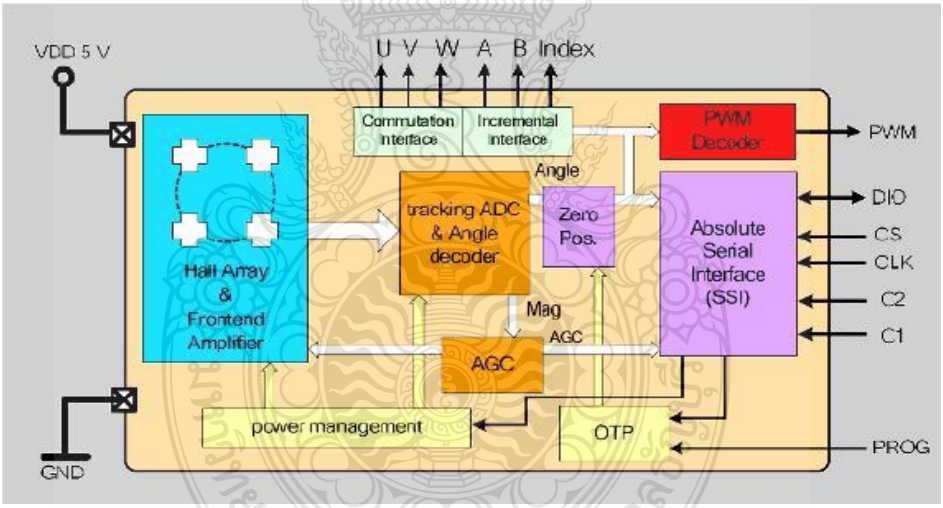
รูปที่ 2.39 ผลการทดลองสัญญาณเอาท์พุทของ Hall Sensor ที่ได้จากวงจรชดเชยมุมเฟสก้าวหน้าที่นำเสนอ

[2] RÓBERT ISTVÁN LŐRINCZ, MIHAI EMANUEL BASCH, DAVID CRISTEA,IVAN BOGDANOV, VIRGIL TIPONUT “Hardware Implementation of Field-Weakening BLDC Motor Control” Electronics and Telecommunications “Politehnica” University of TimiŞoara

ได้นำเสนอวิธีการขับมอเตอร์กระแสตรงไร้แปรงถ่านแบบมีตัวป้อนกลับตำแหน่งของแกนหมุน (Sensored BLDC) แบบมีการชดเชยมุมเฟสก้าวหน้า (Phase advance angle or Field weakening) โดยใช้ตัวบอกตำแหน่งแกนหมุนของมอเตอร์ที่สามารถเปลี่ยนแปลงจุดอ้างอิงตำแหน่งศูนย์องศาของแกนมอเตอร์ได้ (Programmable Rotary Encoder) นั่นหมายถึงว่าตัว Rotary Encoder สามารถปรับมุมเฟสให้กับตัวคอนโทรลเลอร์ ในการขับมอเตอร์แบบมีการชดเชยมุมเฟสก้าวหน้า หรือ ล้าหลัง (Advance angle Or Restart angle) ก็ได้



รูปที่ 2.40 มอเตอร์กระแตรงไร้แปรงถ่านใช้ Programmable rotary encoder บอกตำแหน่งของแกนหมุน



รูปที่ 2.41 Block diagram programmable rotary encoder

Programmable rotary encoder จะสร้างสัญญาณบอกตำแหน่งของแกนหมุน เป็น U V W ซึ่งก็คือ H_A H_B และ H_c เมื่อเทียบกับการใช้ Hall sensor บอกตำแหน่งแกนหมุนนั้นเอง

<u>ตัวอย่าง</u>

ถ้าตำแหน่งศูนย์องศาของแกนหมุนสอดคล้องกับ $\alpha_0 = 50^o$ ดังนั้นค่าของมุมที่อ่านได้จาก Rotary encoder (α_e) จะต้องลบด้วย 50^o อธิบายได้ด้วยสมการ 2.20

$$\alpha_r = \alpha_e - \alpha_0 \tag{2.21}$$

โดยให้

α_r ค่าองศาทางกลของแกนหมุนปัจจุบัน

α_e ค่าองศาที่อ่านได้จาก Encoder

ดังนั้นเมื่อพิจารณาจำนวนคู่ขั้วแม่เหล็ก(Pole pair)แล้วจะได้ค่าการชดเชยมุมเฟสล่วงเป็น

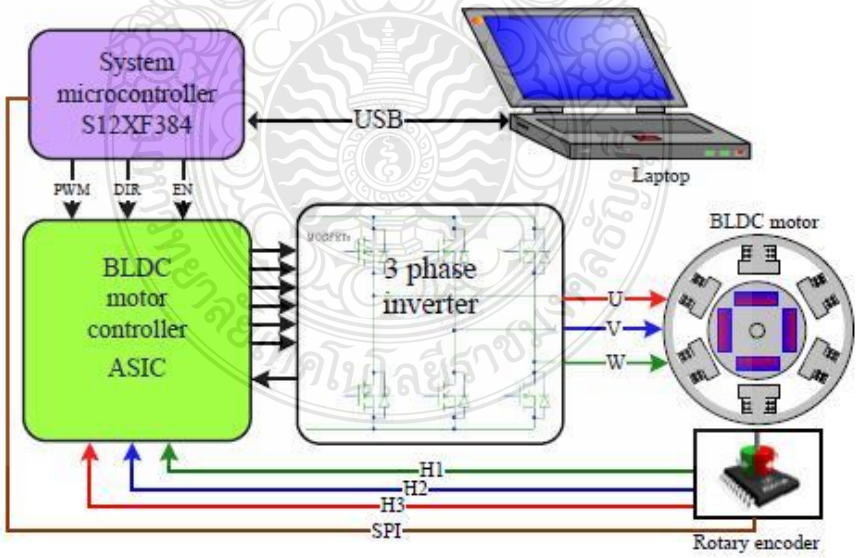
$$\alpha_r = \alpha_e - (\alpha_0 - \frac{\alpha_{advance}}{polepair}) \tag{2.22}$$

โดยให้

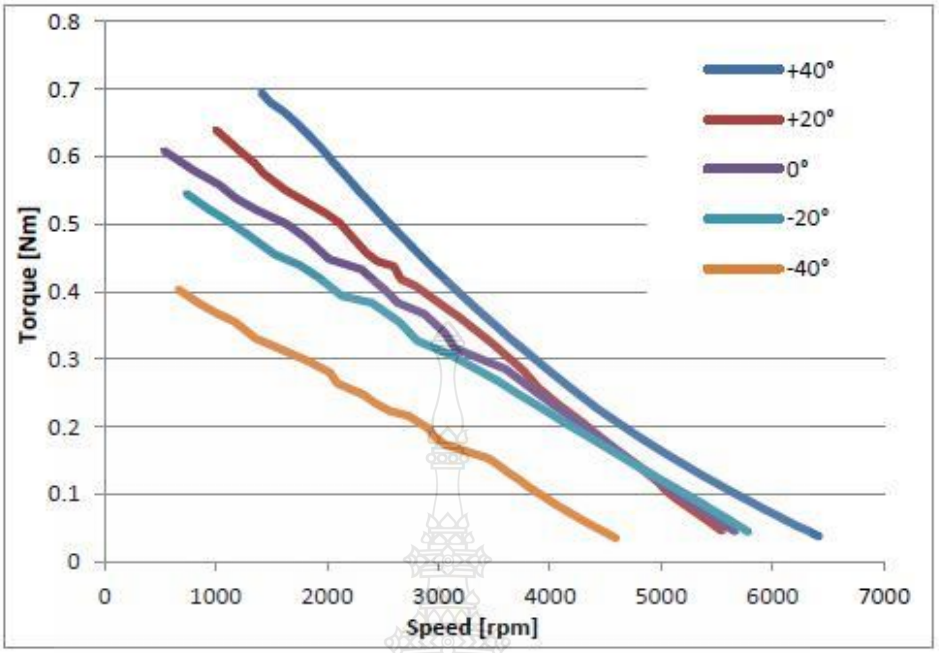
$\alpha_{advance}$ เป็นค่าชดเชยมุมเฟสล่วงหน้า

ดังนั้นเราจะได้มุม α_0 ใหม่ ที่มีการบวกค่าของมุมเฟสก้าวหน้า เป็น $\alpha_0^{'}$

$$\alpha_0^{'} = \alpha_0 + \frac{\alpha_{advance}}{Polepair} \tag{2.23}$$



รูปที่ 2.42 ระบบชดเชยมุมเฟสก้าวหน้าโดยใช้ Programmable Rotary Encoder



รูปที่ 2.43 ผลการทดสอบระบบชดเชยมุมเฟสโดยใช้ Programmable Rotary Encoder

[3] S.K. Safi, PhD, P.P. Acarnley, PhD, CEng, FIEE , A.G. Jack, PhD, CEng, MIEE "Analysis and simulation of the high-speed torque performance of brushless DC motor drives "

ได้นำเสนอแบบจำลองเพื่อใช้ในการทำนาย สมรรถนะของชุดขับเคลื่อนมอเตอร์กระแสตรงไร้แปรงถ่าน ในย่านการทำงานที่ให้ แรงบิด และ ความเร็ว สูง โดยชุดขับเป็นอินเวอร์เตอร์แบบ PWM ในโหมด 120 และ 180 องศา ด้วยวิธีการ Advance Commutation Angle ซึ่งเป็นองค์ประกอบที่มีความสำคัญมากต่อสมรรถนะทางด้าน ความเร็ว และ แรงบิด ของมอเตอร์ โดยมีการนำเสนอการวิเคราะห์ค่ามุมเฟสก้าวหน้าที่เหมาะสมในแต่ละย่านความเร็ว ด้วยวิธีการลองผิด ลองถูก โดยอธิบายภายใต้กระบวนการจำลอง

บทที่ 3
ขั้นตอนการดำเนินงานวิจัย

3.1 บทนำ

การดำเนินงานในการศึกษาพฤติกรรมของมอเตอร์กระแตรงไร้แปรงถ่านในย่านกำลังคงที่ด้วยหลักการชดเชยมุมเฟสก้าวหน้า 3 ขั้นตอนหลัก ๆ ด้วยกันคือ

1 ศึกษาทฤษฎีที่เกี่ยวข้อง ทั้งทางด้านกายภาพ และ ทางด้านคณิตศาสตร์ของมอเตอร์กระแตรงไร้แปรงถ่าน

2 ใช้โปรแกรม MatLab/Simulink ในการสร้างแบบจำลองทางคณิตศาสตร์เพื่อศึกษาพฤติกรรมต่าง ๆ ของมอเตอร์และนำข้อมูลที่ได้ มาศึกษาหลักการและความเป็นไปได้ในการออกแบบระบบควบคุมให้มอเตอร์สามารถทำงานในย่านกำลังคงที่ได้

3 ศึกษาแนวทางการสร้างระบบควบคุมที่สามารถใช้ทดสอบแสดงผลการทำงานของมอเตอร์ให้สอดคล้องกับผลที่ได้จากการจำลองด้วยโปรแกรม MatLab/Simulink ซึ่งการศึกษาข้อมูลในขั้นตอนนี้จะประกอบไปด้วยสองส่วนหลักๆ ด้วยกันคือ ทางด้านการออกแบบแผงวงจรควบคุม (Hardware) และ ทางด้านการพัฒนาโปรแกรมระบบควบคุม รวมไปถึง พัฒนาโปรแกรมที่ใช้ติดต่อกับคอมพิวเตอร์ (Graphic User Interface)เพื่อความสะดวกในการศึกษาและการวิเคราะห์ข้อมูล เพื่อนำไปเปรียบเทียบกับผลที่ได้จากการจำลองด้วยโปรแกรม MathLab Simulink ต่อไป

รูปที่ 3.1 ขั้นตอนการดำเนินงานวิจัย

3.2 การสร้างระบบจำลองทางคณิตศาสตร์เพื่อศึกษาพฤติกรรมของมอเตอร์กระแสตรงไร้แปรงถ่าน (Brushless DC Motor)ในย่านกำลังคงที่ด้วยหลักการชดเชยมุมเฟสก้าวหน้า [13] [14]

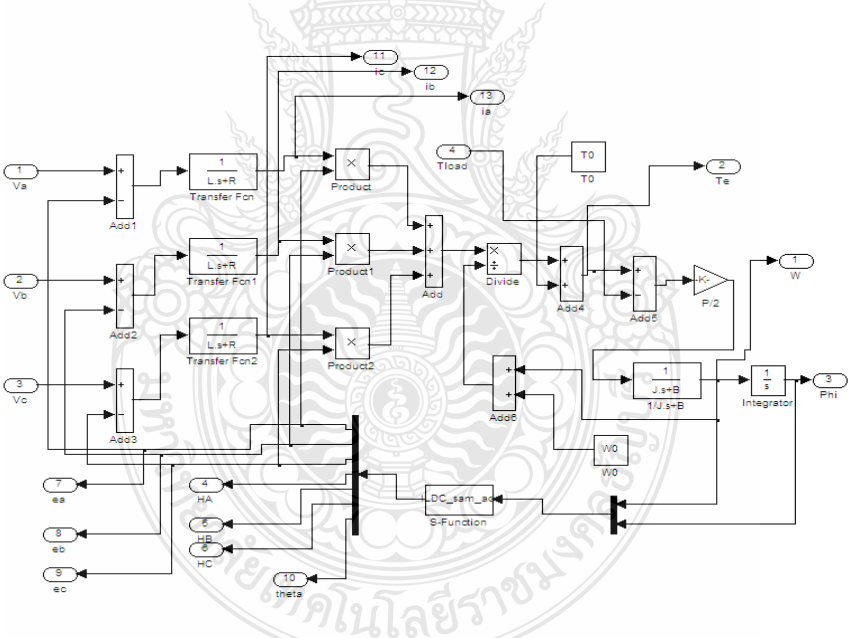
ระบบจำลองทางคณิตศาสตร์ของ BLDC Motor สามารถแบ่งออกเป็น 4 ส่วนได้ดังนี้

3.2.1 แบบจำลองทางคณิตศาสตร์มอเตอร์กระแสตรงไร้แปรงถ่าน(Breushless DC Motor Mathematical Model)

แบบจำลองทางคณิตศาสตร์ของมอเตอร์กระแสตรงไร้แปรงถ่าน(Breushless DC Motor Math Model) ประกอบด้วย

1. ส่วนที่รับสัญญาณทางเข้า(Input Signal)คือ แรงที่ป้อนให้กับมอเตอร์(V_a V_b V_C) และแรงบิดโหลด (Load Torque)

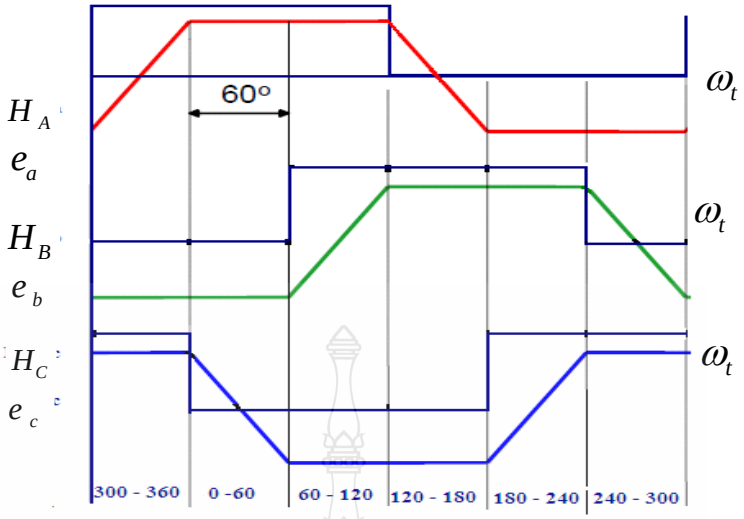
2. ส่วนที่ส่งสัญญาณออกภายนอก(Output Signal) ได้แก่ แรงดันต้านกลับ(e_a e_b e_c) ความเร็วทางไฟฟ้าของสนามแม่เหล็กหมุน(ω_e)ความเร็วทางกล(ω)สัญญาณ Hall Efect (H_A H_B H_C) และแรงบิดทางไฟฟ้า (T_e)



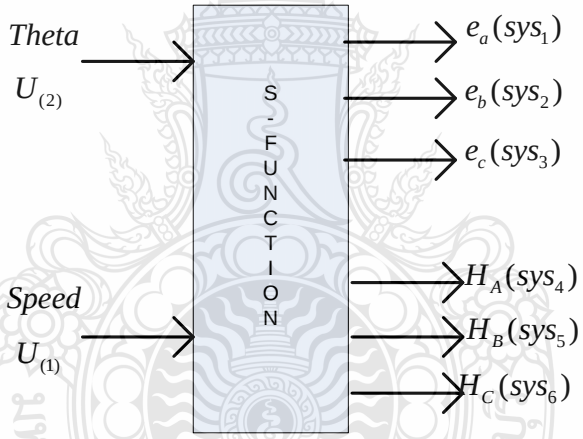
รูปที่ 3.2 แบบจำลองมอเตอร์ไฟฟ้ากระแสตรงไร้แปรงถ่าน

3.2.2 สร้าง S-Function แรงดันต้านกลับ (Back EMF) และสัญญาณ Hall sensor

S-Function ทำหน้าที่ในการรับค่าตำแหน่งส่วนหมุนของมอเตอร์ (Rotor Position) และความเร็วของมอเตอร์ (Motor Speed) ในการประเมินผล สร้างสัญญาณแรงดันต้านกลับ (e_a e_b e_c) และสัญญาณ Hall Sensor (H_A H_B H_C)



[ก] กราฟแรงดันต้านกลับ และ สัญญาณ Hall



[ข] อินพุท เอาท์พุทของ S – Function

รูปที่ 3.3 S – Function แรงดันต้านกลับ และ สัญญาณ Hall Sensor

รูปแบบของโปรแกรมใน S – Function ในส่วนของแรงดันต้านกลับ (รายละเอียดของคำสั่งใน S – Function ดูได้จากภาคผนวก ง)

```
Sys (1) = LA*N*BM*Rl*Rr*u (1);
sys (2) = LB*N*BM*Rl*Rr*u (1);
sys (3) = LC*N*BM*Rl*Rr*u (1);
```

โดยให้

Sys (1) เป็น แรงดันต้านกลับของเฟส A

Sys (2) เป็น แรงดันต้านกลับของเฟส B

Sys (3) เป็น แรงดันต้านกลับของเฟส C

N*BM*Rl*Rr เป็นค่าคงที่ของแรงดันต้านกลับ (K_w)

LA,LB,LC เป็นค่าตัวคูนที่มุม θ ใด ๆ ของเฟส A,B และ C

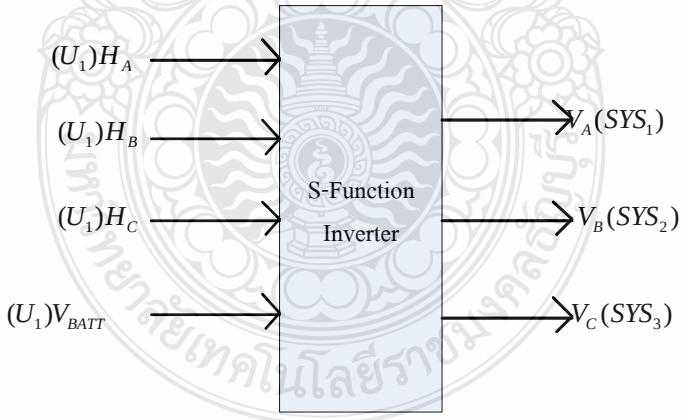
(U1) เป็นความเร็วมอเตอร์

ตัวอย่างโปรแกรมใน s-function กำหนดค่า Hall Sensor ที่ 0 – 60 องศา

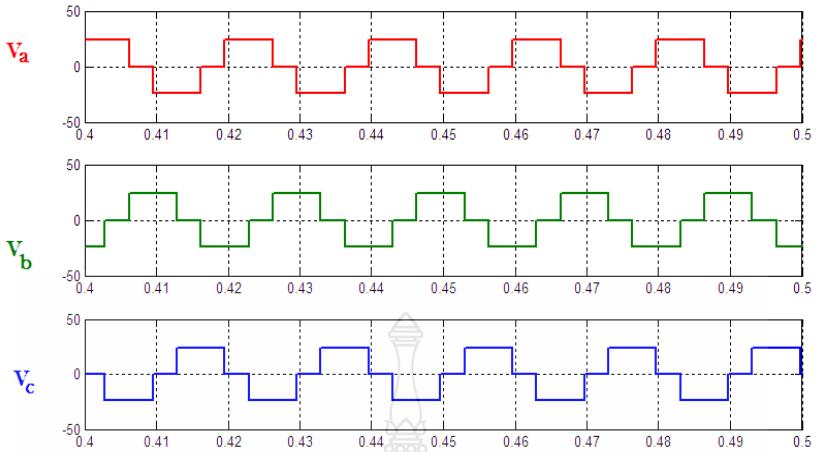
```
sys(4) = 5;      HALL A
sys(5) = 0;      HALL B
sys(6) = 0;      HALL C
```

3.2.3 การสร้างS-Function Inverter

โดย S-Function ทำหน้าที่ในการรับสัญญาณสัญญาณจาก Hall Efect (H_A H_B H_C) และ ค่าแรงดันที่จ่ายให้กับมอเตอร์ (V_{batt}) เพื่อประเมินผลสร้างแรงดันที่จ่ายให้กับมอเตอร์ (V_A V_B V_C)



[ก] S-Function Inverter



[ข] เอาท์พุท S- Function Va,Vb,Vc

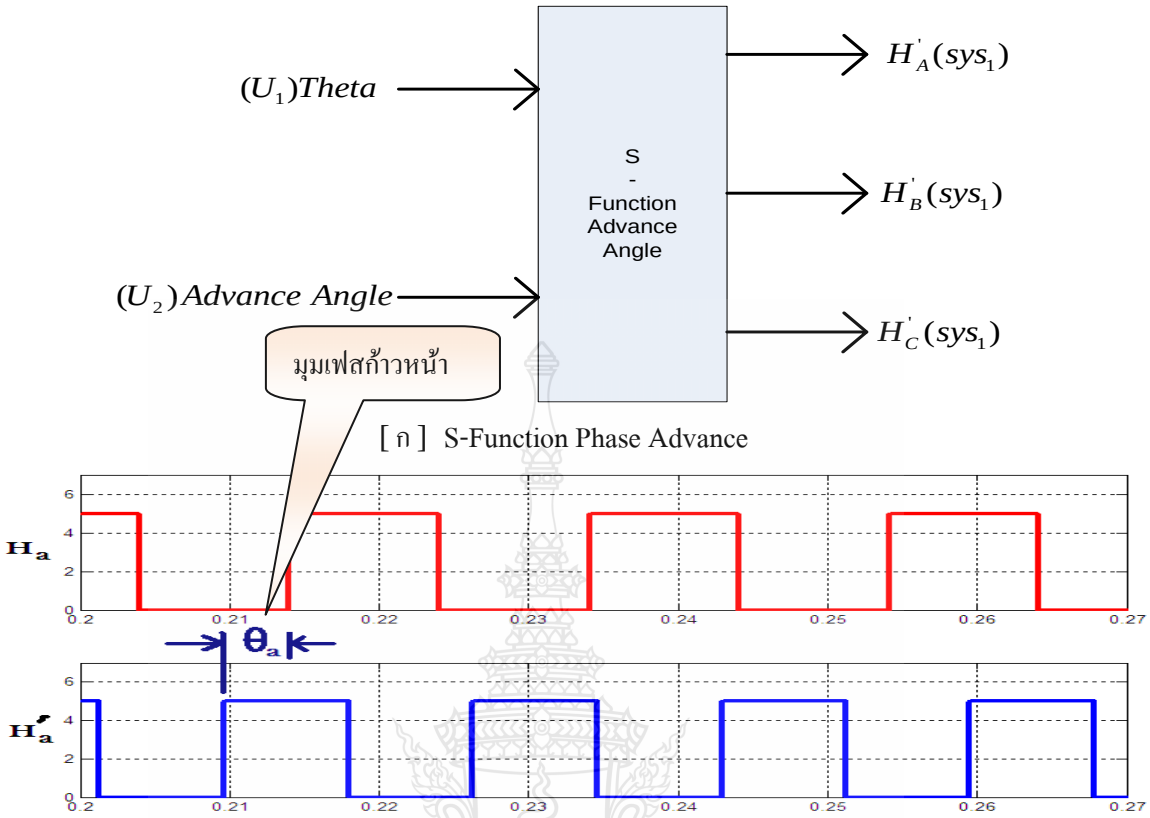
รูปที่ 3.4 S – Function Inverter

ตัวอย่างโปรแกรมใน S-Function บางส่วนในการสร้าง V_A V_B V_C ที่ Ha = 0;Hb = 0;Hc = 0;

```
if(ha>0 & hb==0 & hc==0)%100
 if(vb > vmax)
 vb=vmax;
 end
 sys(1)=vb;
 sys(2)=-vb;
 sys(3)=0;
  end
```

3.2.4 การสร้าง S-Function ส่วนชดเชยมุมเฟสก้าวหน้า

โดย S-Function ทำหน้าที่ในการรับค่าตำแหน่งส่วนหมุนของมอเตอร์ (Rotor Position) และ ค่ามุม Advance Angle เพื่อประเมินผลสร้างสัญญาณ $H_A^{'}$ $H_B^{'}$ $H_C^{'}$

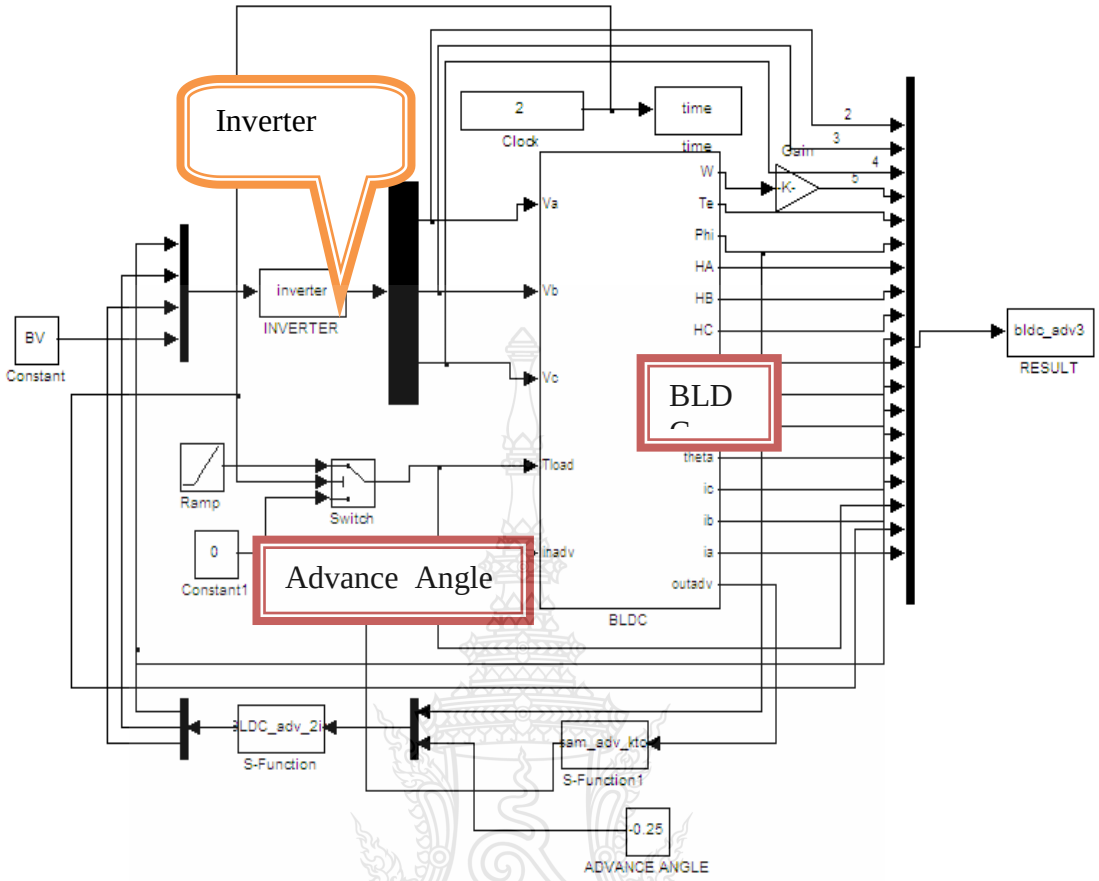


[ก] S-Function Phase Advance

[ข] สัญญาณ Hall Sensor H_A เทียบกับ $H_A^{'}$

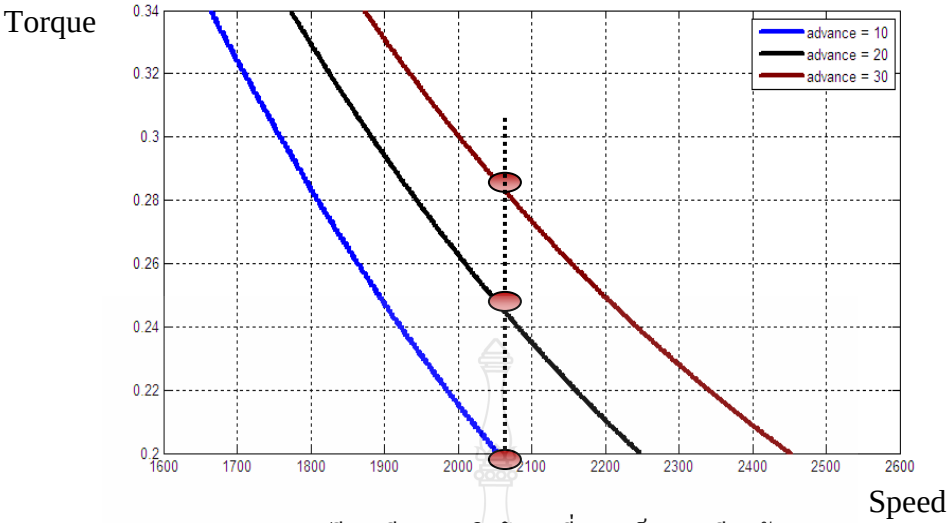
รูปที่ 3.5 S-Function Phase Advance

ขั้นตอนถัดไปหลังจากทำการสร้างแบบจำลอง ทั้ง 4 ส่วนในข้างต้น คือ แบบจำลองมอเตอร์กระแสตรงไร้แปรงถ่าน (Breushless DC Motor Math Model) แบบจำลองอินเวอร์เตอร์ (S – Function Inverter) แบบจำลอง Back EMF (S-Function Back EMF & Hall sensor) และแบบจำลอง ตัวชดเชยมุมเฟสก้าวหน้า (S – Function Advance Angle) ตามลำดับ มาประกอบเป็นระบบจำลองทางคณิตศาสตร์เพื่อศึกษาพฤติกรรมของมอเตอร์กระแตรงไร้แปรงถ่าน (Brushless DC Motor) ในย่านกำลังคงที่ด้วยหลักการชดเชยมุมเฟสก้าวหน้า ดังแสดงดังรูปที่ 3.6

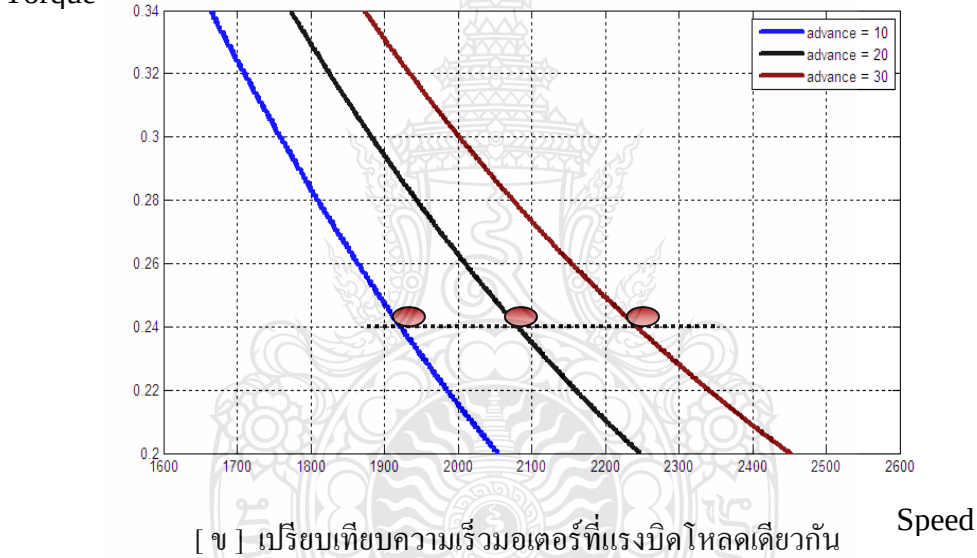


รูปที่ 3.6 แบบจำลองมอเตอร์กระแสตรงไร้แปรงถ่านที่มีการชดเชยมุมเฟสก้าวหน้า

จำลองการทำงานของมอเตอร์กระแสตรงไร้แปรงถ่านโดยอินพุทค่ามุมเฟสก้าวหน้าที่ 10 20 และ 30 องศาตามลำดับ โดยมอเตอร์ มีพารามิเตอร์ที่สำคัญดังนี้ แรงดันที่จ่ายให้กับมอเตอร์ (VB) = 24 VDC ความต้านทานของชุดขดลวดมอเตอร์ (R) = 1.5 Ω ค่าความเหนี่ยวนำของชุดขดลวดมอเตอร์ (L) = 1.92 mH จำนวนขั้วแม่เหล็ก (P) = 48 Pole ค่าโมเมนต์ความเฉื่อยที่เพลามอเตอร์ (J) = $1X10^{-4}$ kgm^2 ค่าสัมประสิทธิ์ความหน่วงที่เพลา (B) = $1X10^{-4}$ จำนวนขั้วแม่เหล็ก (P) = 4, ค่าคงที่ (Kw) = 0.0545 และ แรงบิดโหลด (TL) = 0 – 0.5 Nm ผลการจำลองที่ได้แสดงดังรูปที่ 3.7



[ก] เปรียบเทียบแรงบิดโหลดที่ความเร็วรอบเดียวกัน



[ข] เปรียบเทียบความเร็วมอเตอร์ที่แรงบิดโหลดเดียวกัน

รูปที่ 3.7 ผลที่ได้จากการจำลองการชดเชยมุมเฟสก้าวหน้า

3.3 ชุดทดสอบระบบชดเชยมุมเฟสก้าวหน้ามอเตอร์ไฟฟ้ากระแสตรงไร้แปรงถ่านพิกัด 500 W 48 volt

หลังจากที่ได้ศึกษาความเป็นไปได้จากแบบจำลองจนได้ข้อสรุปถึงความเป็นไปได้แล้วก็ทำการออกแบบสร้างแผงควบคุมตามองค์ประกอบต่าง ๆ ในแบบจำลองดัง รูปที่3.8 และ 3.9 ซึ่งมีองค์ประกอบหลักดังนี้

1 หน่วยชดเชยมุมเฟสก้าวหน้า (Advance Angle Operating System)

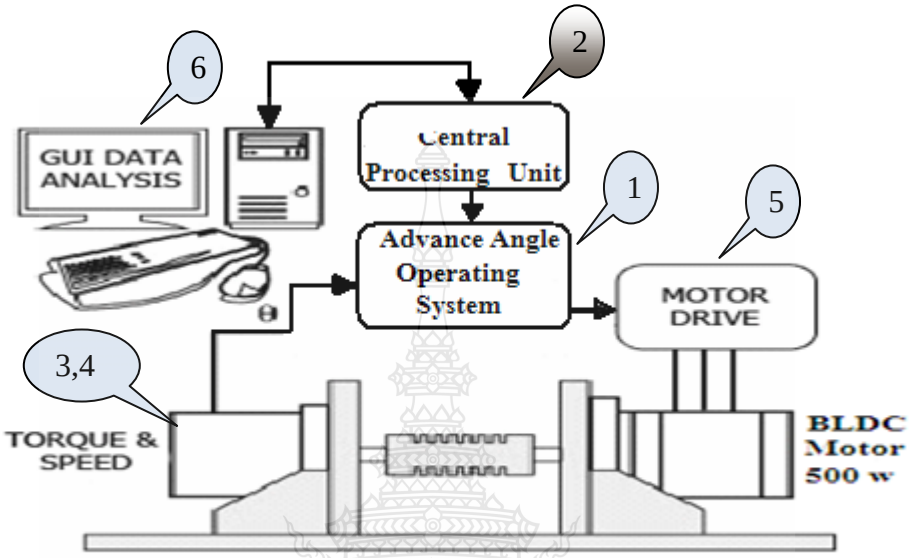
2 หน่วยประเมินผลกลาง (Central Processing Unit)

3 หน่วยวัดความเร็วมอเตอร์ (Speed Sensor Unit)

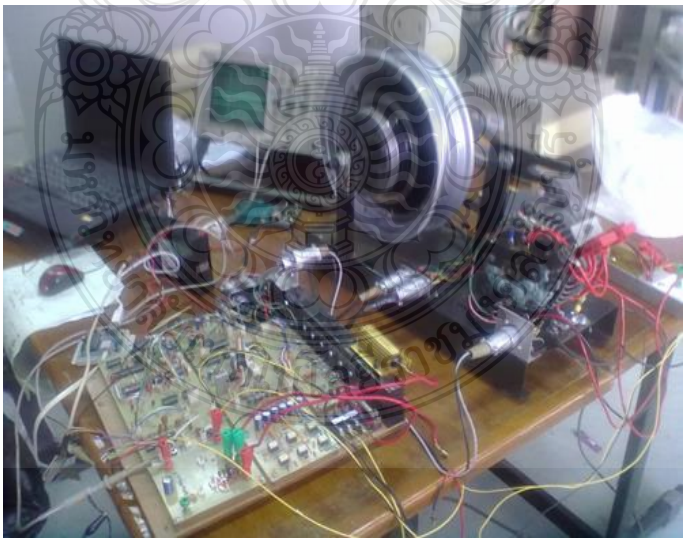
4 หน่วยปรับแรงบิดโหลด (Torque Control Unit)

5 หน่วยควบคุมการขับเคลื่อนมอเตอร์ (Motor Drive Unit)

6 หน่วยติดต่อสื่อสารกับคอมพิวเตอร์(GUI Data Analysis)



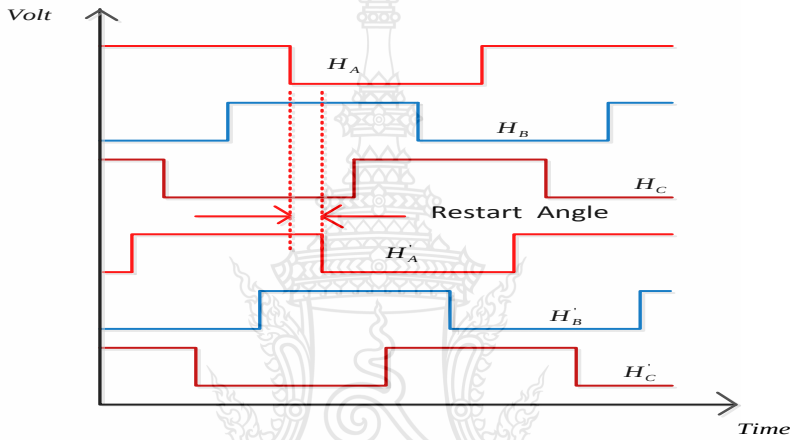
รูปที่ 3.8 องค์ประกอบชุดทดสอบมอเตอร์ไฟฟ้ากระแสตรงไร้แปรงถ่านพิกัด 500 W 48 volt



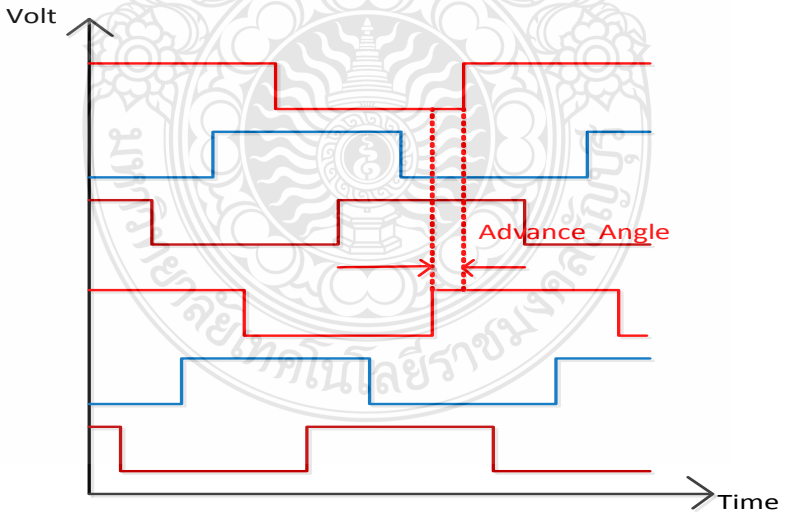
รูปที่ 3.9 ชุดทดสอบมอเตอร์ไฟฟ้ากระแสตรงไร้แปรงถ่านพิกัด 500 W 48 VDC

3.3.1 หน่วยประเมินผลระบบชดเชยมุมเฟสก้าวหน้า

ระบบปฏิบัติการ การชดเชยมุมเฟสก้าวหน้า มีหน้าที่ในการปรับแต่งสัญญาณอินพุท ที่ได้จาก Hall Sensor H_A H_B แล H_C แล้วส่งออกทางเอาท์พุทเป็น $H_A^{'}$ $H_B^{'}$ และ $H_C^{'}$ โดยการทำให้เกิดการต่างเฟสกันระหว่างคู่สัญญาณ H_A กับ $H_A^{'}$ H_B กับ $H_B^{'}$ และ H_C กับ $H_C^{'}$ ตามลำดับ โดยรับคำสั่งมุมต่างเฟสจากคอมพิวเตอร์ (GUI Data Analysis) ผ่านหน่วยประเมินผลกลาง (Central Processing Unit) โดยลักษณะการต่างเฟสของสัญญาณมีสองรูปแบบคือ $H_A^{'}$ ล้าหลัง และ $H_A^{'}$ นำหน้า H_A ดังแสดงในรูปที่ 3.10 [ก] [ข]

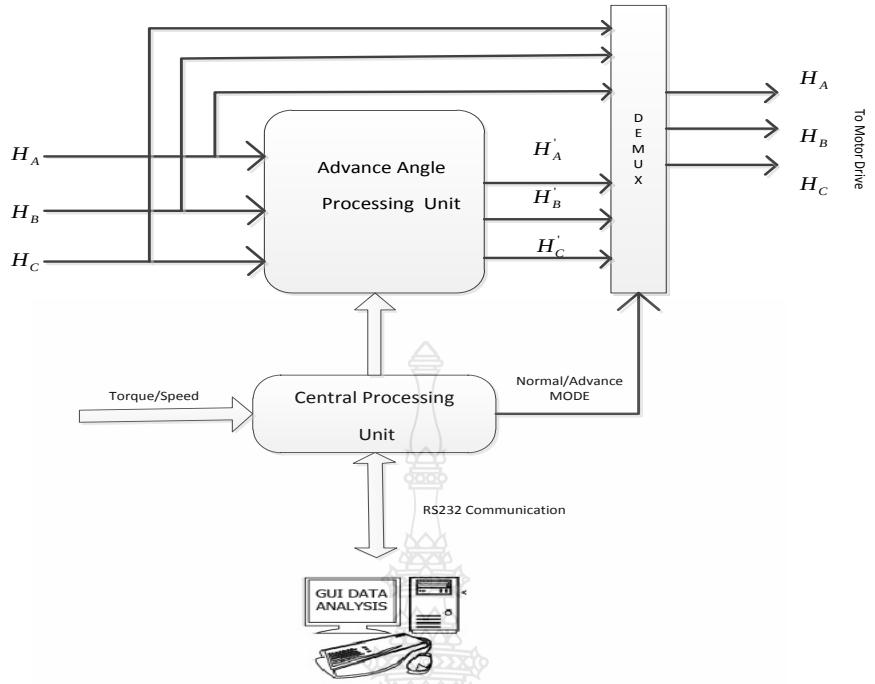


[ก]

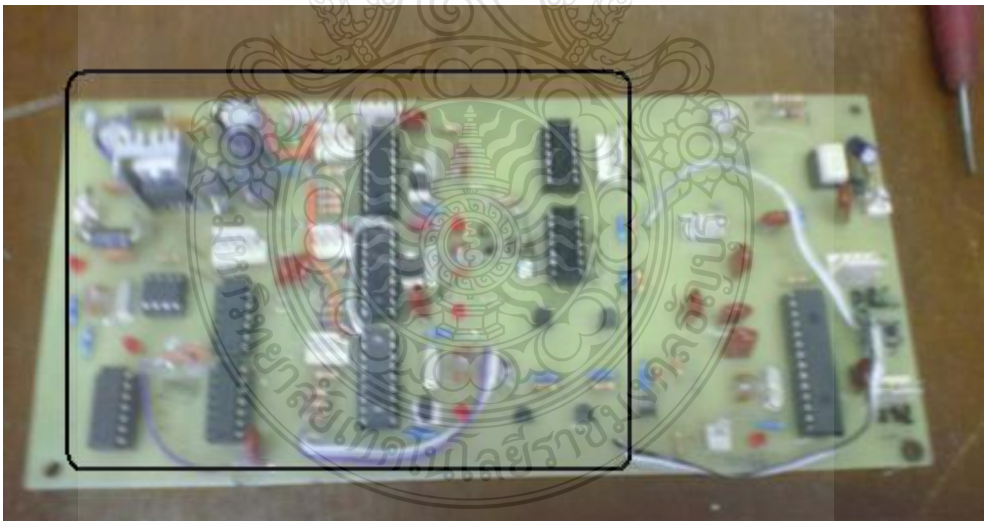


[ข]

รูปที่ 3.10 ก) $H_A^{'}$ ล้าหลัง H_A ข) $H_A^{'}$ นำหน้า H_A

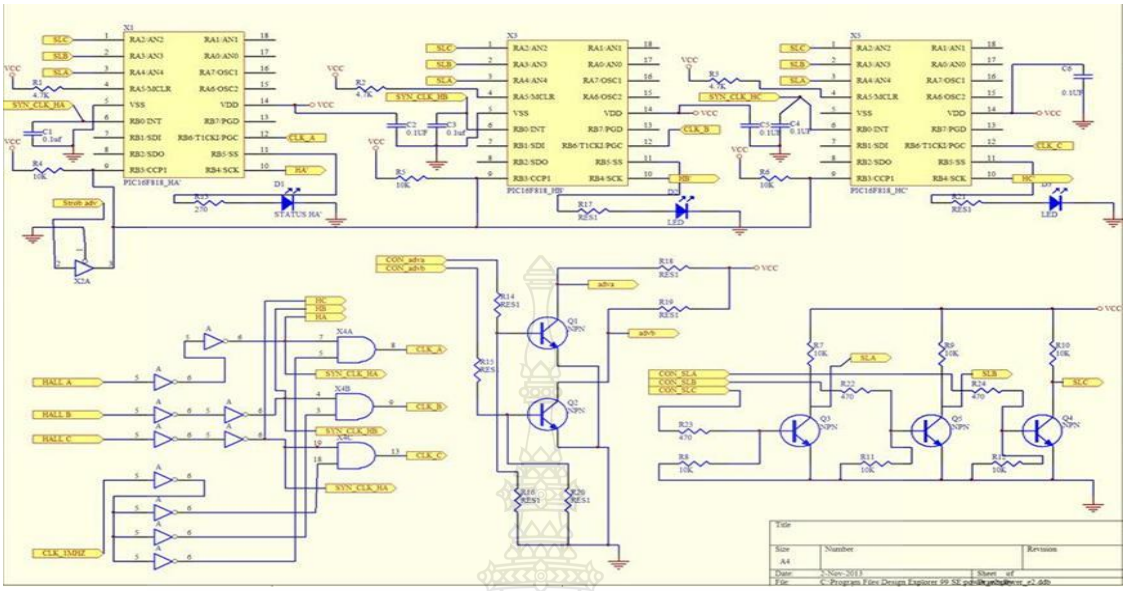


รูปที่ 3.11 องค์ประกอบการทำงานของหน่วยเชยมุมเฟสก้าวหน้า (Advance Angle Operating System)



รูปที่ 3.12 แผงวงจรหน่วยการชดเชยมุมเฟสก้าวหน้า

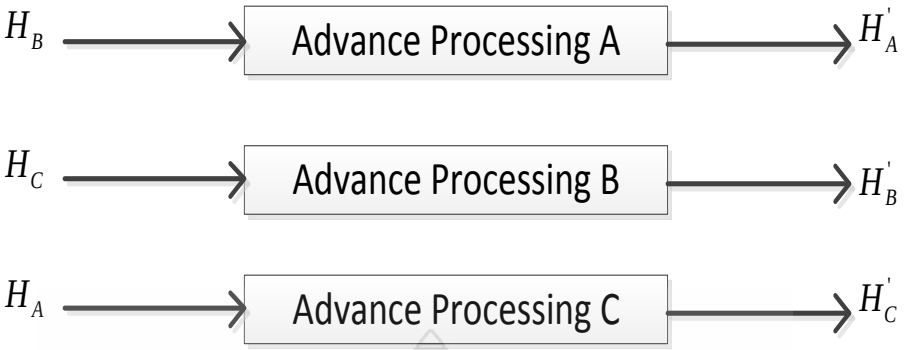
ในส่วนของวงจรของหน่วยชดเชยมุมเฟสก้าวหน้า แสดงดังรูปที่ 3.13



รูปที่ 3.13 วงจรระบบปฏิบัติการ การชดเชยมุมเฟสก้าวหน้า (Advance Angle Operating System Circuit)

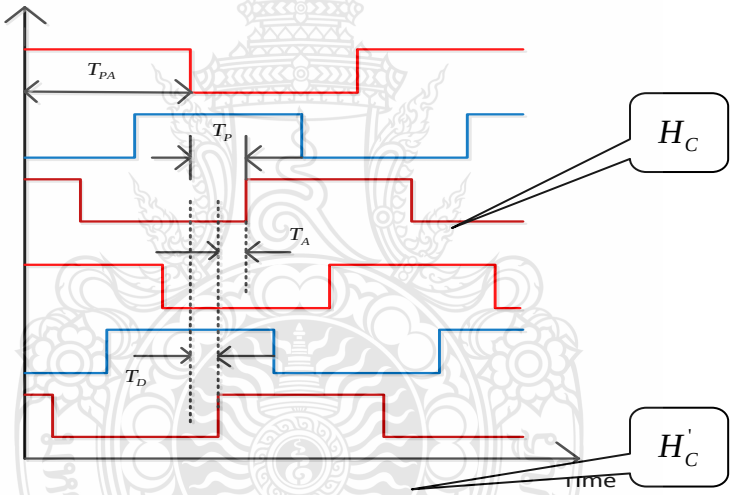
แนวคิดและการออกแบบสร้างมุมต่างเฟสของคู่สัญญาณ H_A H_B H_C ***กับ*** $H_A^{'}$ $H_B^{'}$, $H_C^{'}$

การสร้างคู่สัญญาณดังกล่าวใช้แนวคิดอ่านค่าและจดจำลักษณะของสัญญาณที่เกิดขึ้นในอดีตทีมีรูปแบบเป็นวัตจักร (Time Period) แล้วทำการสร้างสัญญาณขึ้นในอนาคตให้มีลักษณะเหมือนกับสัญญาณที่เกิดขึ้นในอดีตเพียงแต่เราสามารถเลื่อนตำแหน่งการเกิดและการดับของสัญญาณให้แตกต่างไปจากสัญญาณเดิมได้ จากหลักการดังกล่าวที่กล่าวมาข้างต้น จะเห็นได้ว่าเราไม่สามารถสร้างสัญญาณใหม่จากสัญญาณของตัวมันเองได้ เช่น เราไม่สามารถ สร้างสัญญาณ $H_A^{'}$ จาก H_A ได้ ตามภาพที่ 3.9 จะเห็นได้ว่าสัญญาณ H_A H_B และ H_C จะมีค่าเวลาในหนึ่งวัตจักร(Time Period) เท่ากัน แต่งต่างกันตรงเวลาการเกิดและการดับดังนั้นเราจะใช้หลักการที่กล่าวมาในข้างต้นโดยใช้ H_A สร้าง $H_C^{'}$, H_C สร้าง $H_B^{'}$ และ H_B สร้าง $H_A^{'}$ ตามลำดับ ดังรูปที่ 3.14



รูปที่ 3.14 คู่สัญญาณการสร้าง $H_A^{'}$ $H_B^{'}$ และ $H_C^{'}$

ซึ่งตัวอย่างการประเมินผลการชดเชยมุมเฟสก้าวหน้าของ $H_C^{'}$ (Advance Processing Unit C)แสดงให้เห็นได้ดังรูปที่ 3.15



รูปที่ 3.15 การใช้สัญญาณ H_A สร้าง $H_C^{'}$

ค่าเวลาต่าง ๆ ตามรูปที่ 3.15 สามารถอธิบายได้ดังนี้

$$T_P = T_{PA}/3 \quad (3.1)$$

$$T_A = T_P - T_D \quad (3.2)$$

โดยให้

T_A ค่าชดเชยมุมเฟสล่วงหน้า

T_P มุมต่างเฟสระหว่าง H_A กับ H_c

T_d ค่าหน่วงเวลาการเกิด $H_c^{'}$

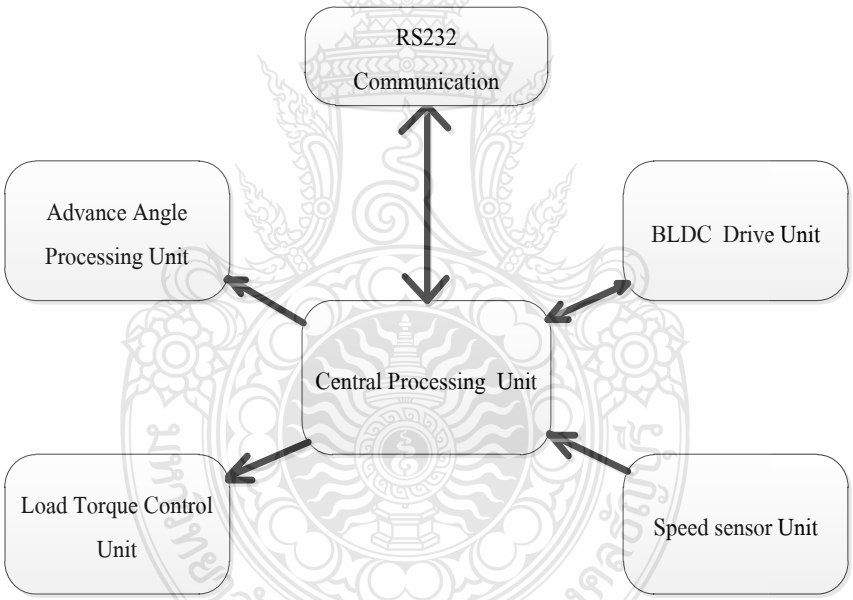
T_{PA} ครึ่งคาบเวลาทางไฟฟ้า

3.3.2 หน่วยประเมินผลกลาง(Central Processing Unit)

หน่วยประเมินผลกลางทำหน้าที่กับกับดูแลระบบย่อยทั้งหมดได้แก่

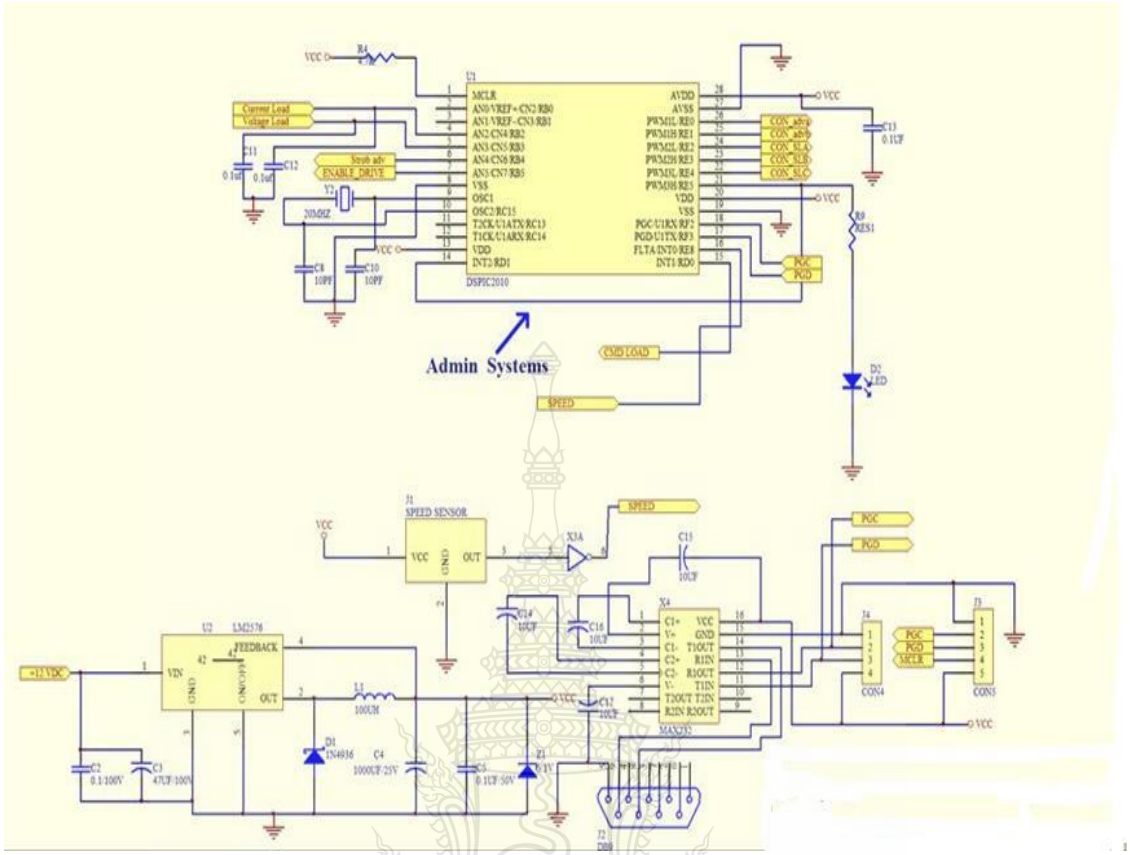
1. กับกับการทำงานชุดควบคุมการขับเคลื่อน (BLDC Motor Drive Unit)
2. กับกับการทำงานชุดประเมินผลการชดเชยมุมเฟสก้าวหน้า (Advance Angle Process Unit)
3. กับกับการทำงานตัวควบคุมแรงบิดโหลด (Load Torque Control Unit)
4. วัดความเร็ว(Speed Sensor Unit)
5. แรงบิดโหลด(Load Torque Unit)
6. ติดต่อสื่อสารกับคอมพิวเตอร์ผ่านโปรแกรม GUI Data Analysis

ซึ่งสามารถเขียนเป็น Block Diagram ได้ดังนี้

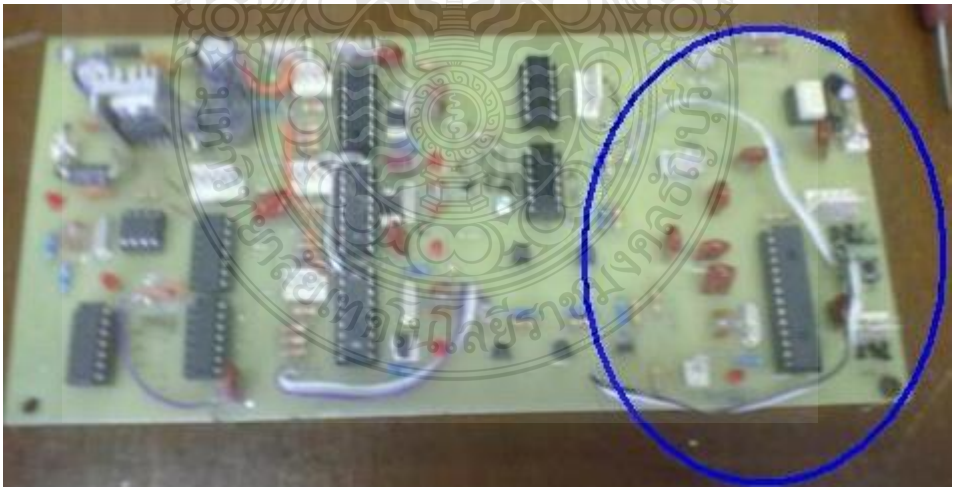


รูปที่ 3.16 Block diagram ของหน่วยประเมินผลกลาง

โดยวงจรหน่วยประเมินผลกลางแสดงดังรูปที่ 3.17



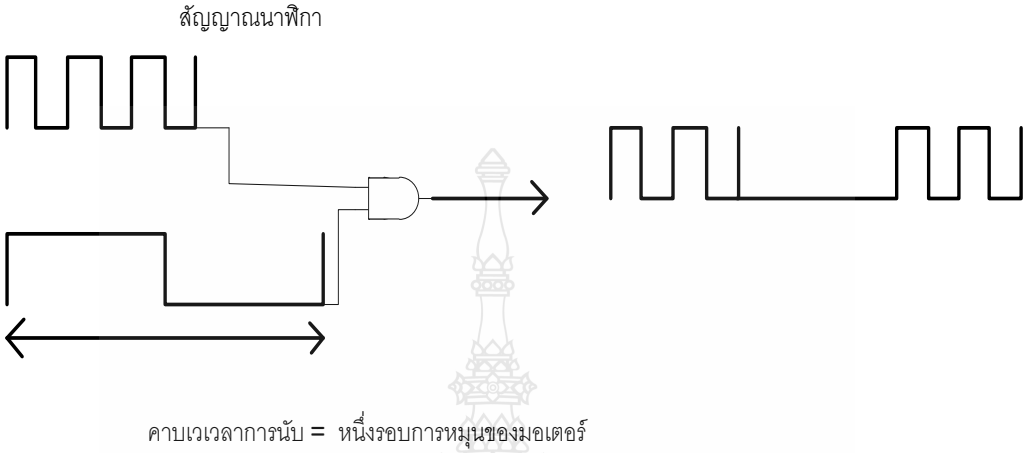
รูปที่ 3.17 รายละเอียดวงจรหน่วยประเมินผลกลาง (Central Processing Unit)



รูปที่ 3.18 แผงวงจรหน่วยประเมินผลกลาง (Central Processing Unit)

3.3.3 หน่วยวัดความเร็วมอเตอร์ (Speed Sensor Unit)

หลักการตรวจจับความเร็วในระบบดิจิตอลใช้หลักการนับสัญญาณนาฬิกาที่มีความเที่ยงตรงสูงต่อหนึ่งหน่วยเวลา(Clock/Time Unit) อธิบายได้ตามรูปที่3.19



รูปที่ 3.19 หลักการวัดความเร็วระบบดิจิตอล

ตัวอย่าง

ถ้าให้สัญญาณนาฬิกามีความถี่เท่ากับ 1000000 Hz

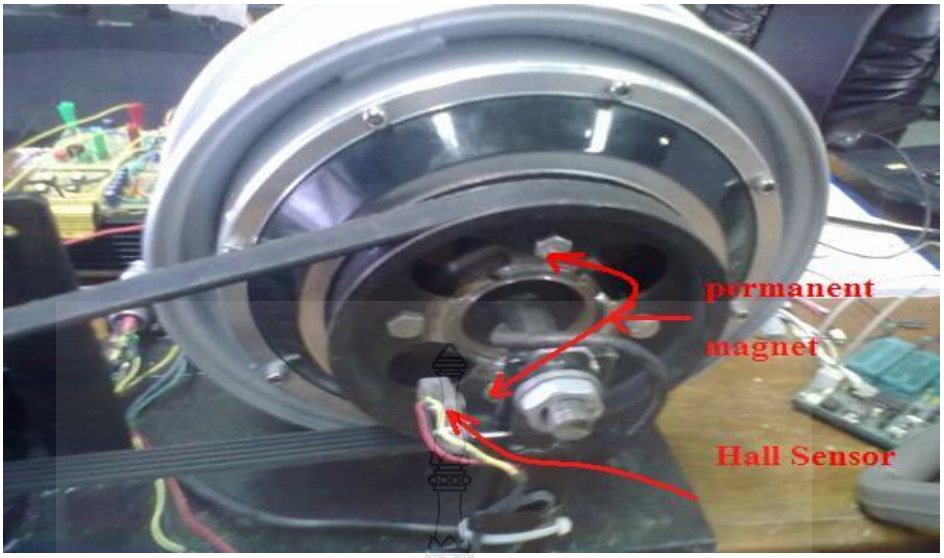
หนึ่งคาบเวลาของสัญญาณนาฬิกา เท่ากับ 1/1000000 = 0.000001 sec

ถ้าหนึ่งรอบการหมุนของมอเตอร์นับสัญญาณนาฬิกาได้ 1000000

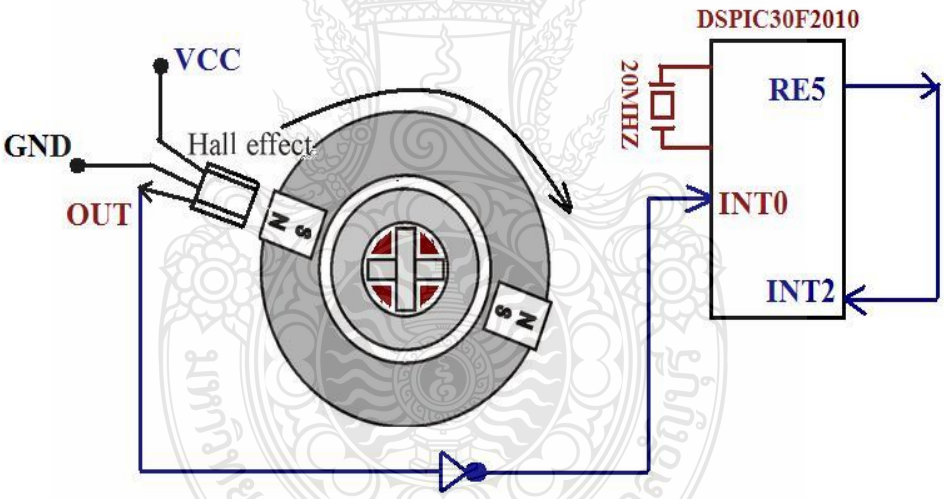
นั่นหมายถึงว่ามอเตอร์ใช้เวลาการหมุนต่อหนึ่งรอบ

= (1000000 x 2 0.000001) = 2 sec

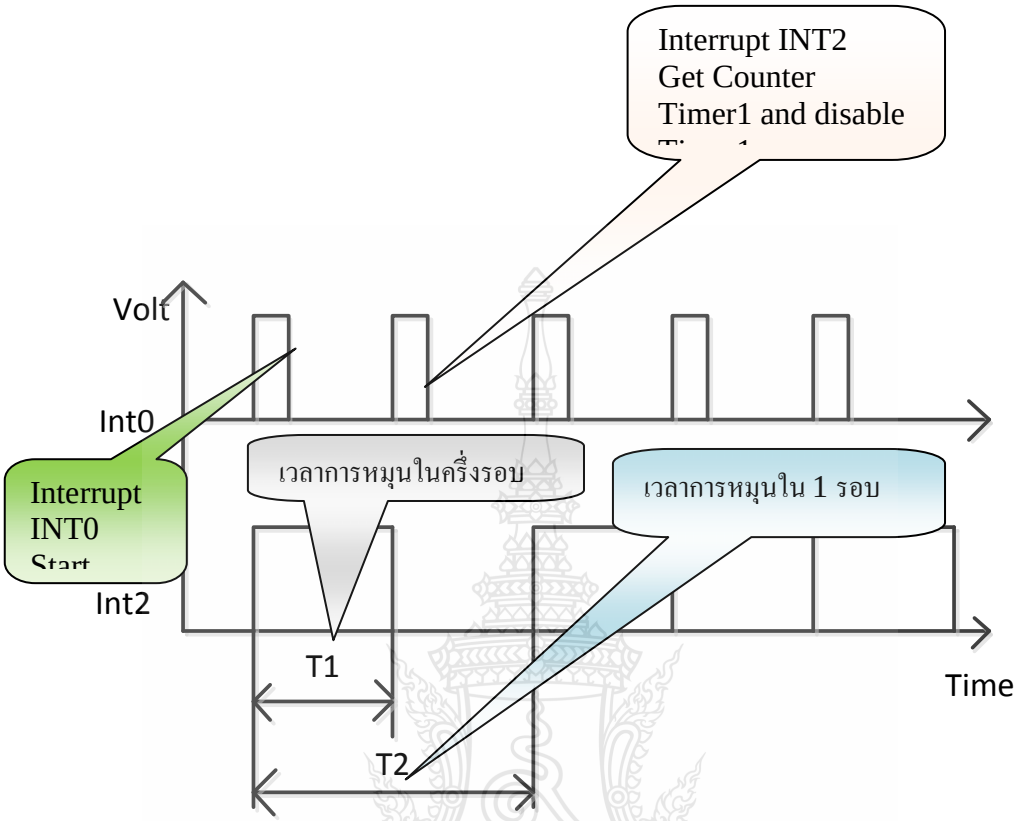
= 2 x 60 = 120 RPM



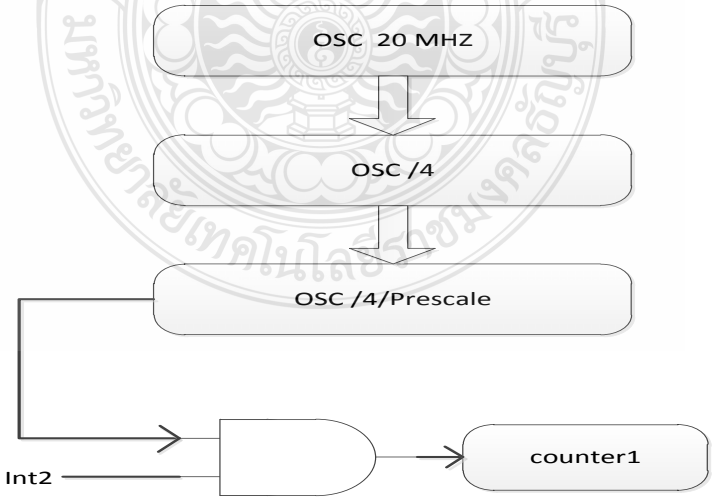
รูปที่ 3.20 ตำแหน่งติดตั้งแม่เหล็กถาวรและ Hall Sensor เพื่อตรวจจับความเร็วรอบมอเตอร์



รูปที่ 3.21 วิธีการตรวจจับความเร็วมอเตอร์



รูปที่ 3.22 สัญญาณอินพุทที่ขา INT0 , INT2



รูปที่ 3.23 การนับสัญญาณนาฬิกาของ Counter 1

จากรูปที่ 3.19 และ 3.20 สามารถหาความเร็วรอบของมอเตอร์ได้ดังนี้

XTAL = 20 MHZ

OSC/4 = 5MHZ

Prescale = 64

เพราะฉะนั้นความถี่ของสัญญาณนาฬิกาที่ส่งเข้า Counter 1 = 5000000/64 = 78125 HZ

ตัวอย่างการคำนวณความเร็วรอบมอเตอร์

สมมุติค่าที่ได้จาก counter 1 = 10000

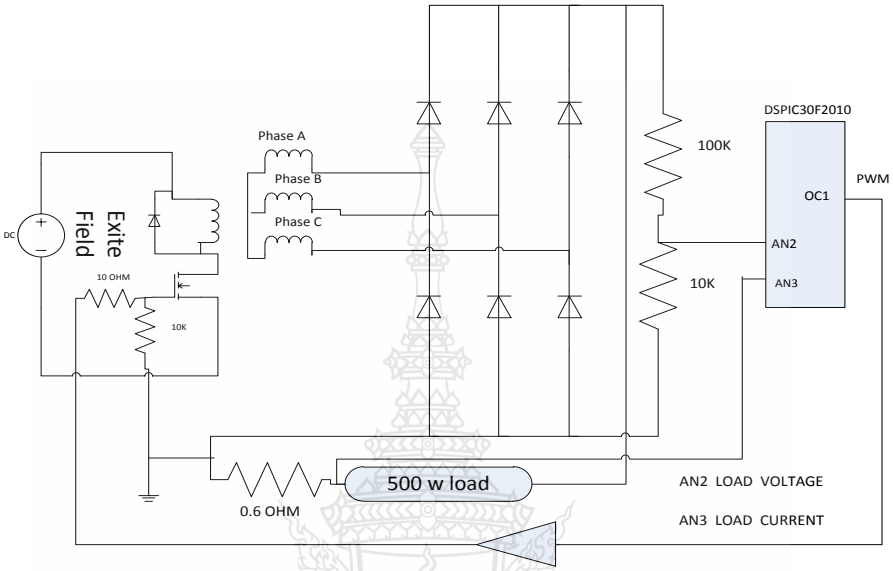
T1 (เวลาที่ใช้ในครึ่งรอบ) = 10000 x 64 x(1/5000000) = 0.128 sec

T2 (เวลาที่ใช้ในหนึ่งรอบ) = 0.128 x 2 = 0.256 sec

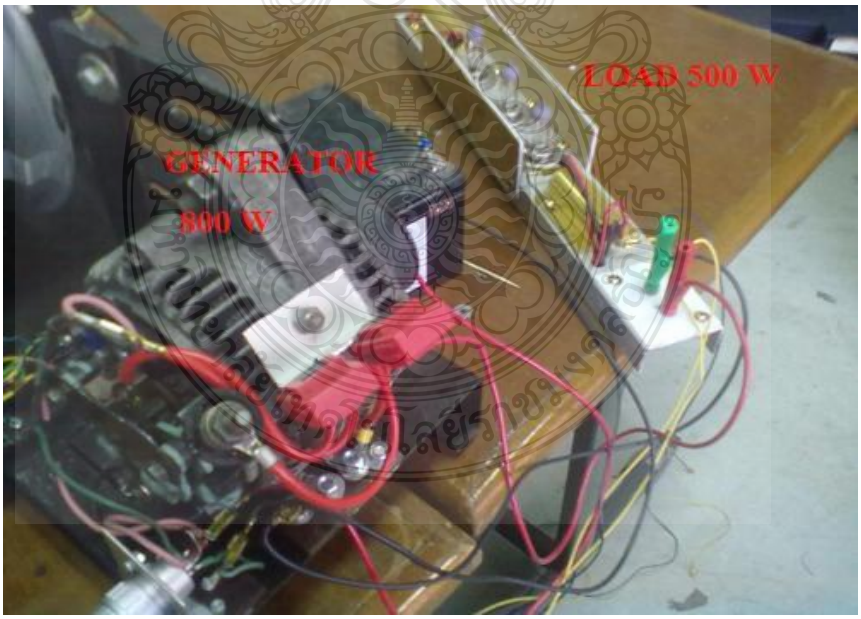
ดังนั้นจะได้ความเร็วรอบมอเตอร์ = 60/0.256 = 234 RPM ข้างล่างเป็น คำสั่งในส่วนของโปรแกรมบริการอินเตอร์รับ INT0 และ INT2 ในการอ่านค่าความเร็วรอบมอเตอร์

```
#int_EXT0
void  EXT0_isr(void)
{
output_toggle(pin_e5); if(input(pin_d1));
{
set_timer1(0);flg_t1=0;setup_timer1(TMR_INTERNAL|TMR_DIV_BY_64);}}
#int_EXT2
void  EXT2_isr(void)
{
GET_T2=get_timer1();
flg_ext2=1;setup_timer1(TMR_DISABLED);
}
```

3.3.4 หน่วยควบคุมแรงบิดโหลด (Load Torque Control)



รูปที่ 3.24 วงจรโดยสังเขปของหน่วยความคุมแรงบิดโหลด



รูปที่ 3.25 ชุดจำลองแรงบิดโหลด

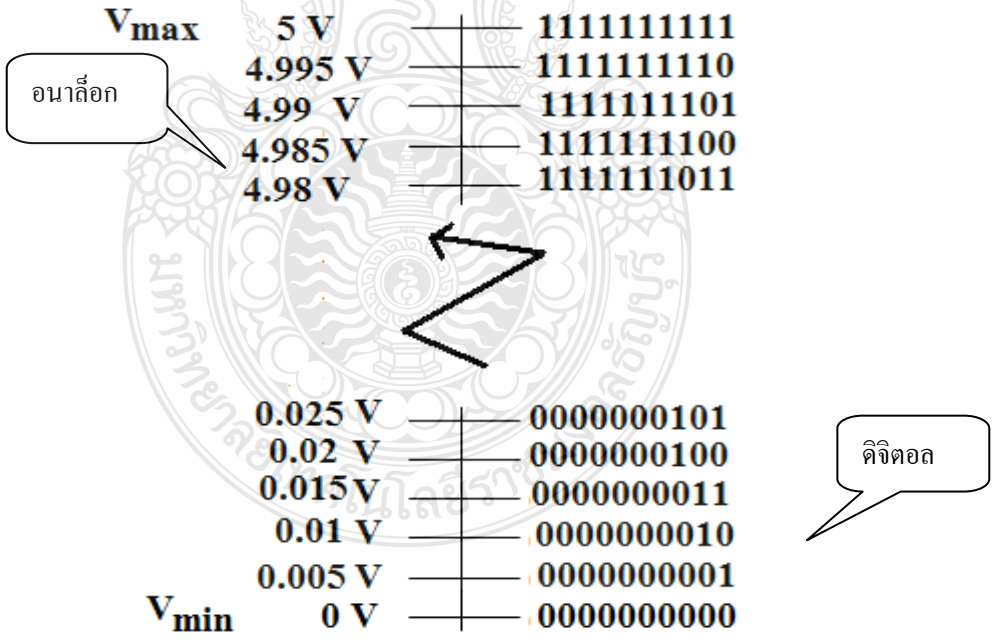
หลักการควบคุมแรงบิดโหลด (Load Torque Control Concept)

จากรูปที่ 3.24 และ 3.25 เราใช้เครื่องกำเนิดไฟฟ้าขนาด 800 W เป็นโหลดให้กับมอเตอร์ โดยให้เครื่องกำเนิดรับภาระโหลดสูงสุด 500 W การเปลี่ยนแปลงภาระโหลดทำได้โดยการปรับกระแสในขดลวด Exit Field ซึ่งถูกสั่งการจากโปรแกรม GUI Data Analysis ดังนั้นการหาแรงบิดโหลดจะได้มาจาก การนำกระแสที่ไหล และ แรงดันที่ตกคร่อมโหลด และความเร็วของมอเตอร์ซึ่งเป็นไปตามสมการข้างล่าง

$$P = E.I \qquad (3.3)$$

$$T = P/\omega \qquad (3.4)$$

โดยการคำนวณแรงบิดโหลด (Load Torque Calculation) จากรูปที่ 3.18 ใช้ AN2 อ่านค่าแรงดันโหลด และ AN3 อ่านค่ากระแสโหลด โดยกำหนดให้ระดับความละเอียดในการแปลงสัญญาณจากอนาล็อกเป็นดิจิตอลขนาด 10 บิตขอบเขตการอ่านค่าแรงดันระหว่าง VSS – VCC (0 – 5VDC) แสดงดังรูปที่ 3.26



รูปที่ 3.26 อัตราส่วนระหว่างค่าของอนาล็อกกับดิจิตอลที่ความละเอียด 10 บิต

ตัวอย่าง

ที่ความเร็วรอบของมอเตอร์เท่ากับ 350 RPM

ค่าใน AN2 = 1111111110 จะได้แรงดันที่ AN2 = 4.995 V และ จากวงจรรูป 3.18 จะได้แรงดัน ตกคร่อมโหลด (Vload) = 4.995 x 10 = 49.95 V

ค่าใน AN3 = 1111111011 จะได้แรงดันที่ AN2 = 4.98 V และจากรูป 3.18 จะได้ค่ากระแสที่ไหลผ่านโหลด(Iload) = 4.98/0.6 = 8.3 Amp

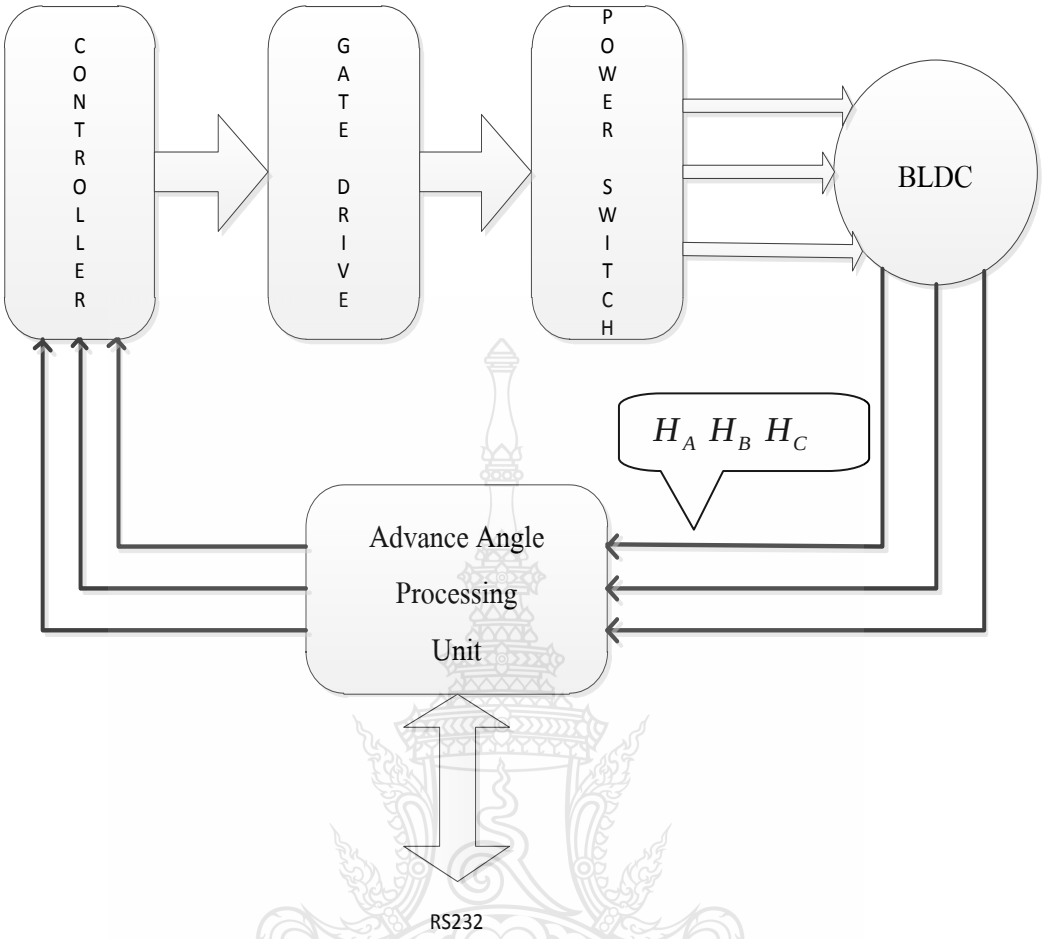
จากค่าดังกล่าวข้างต้น จะได้

P = E.I = 49.95 X 8.3 = 414.63 WATT

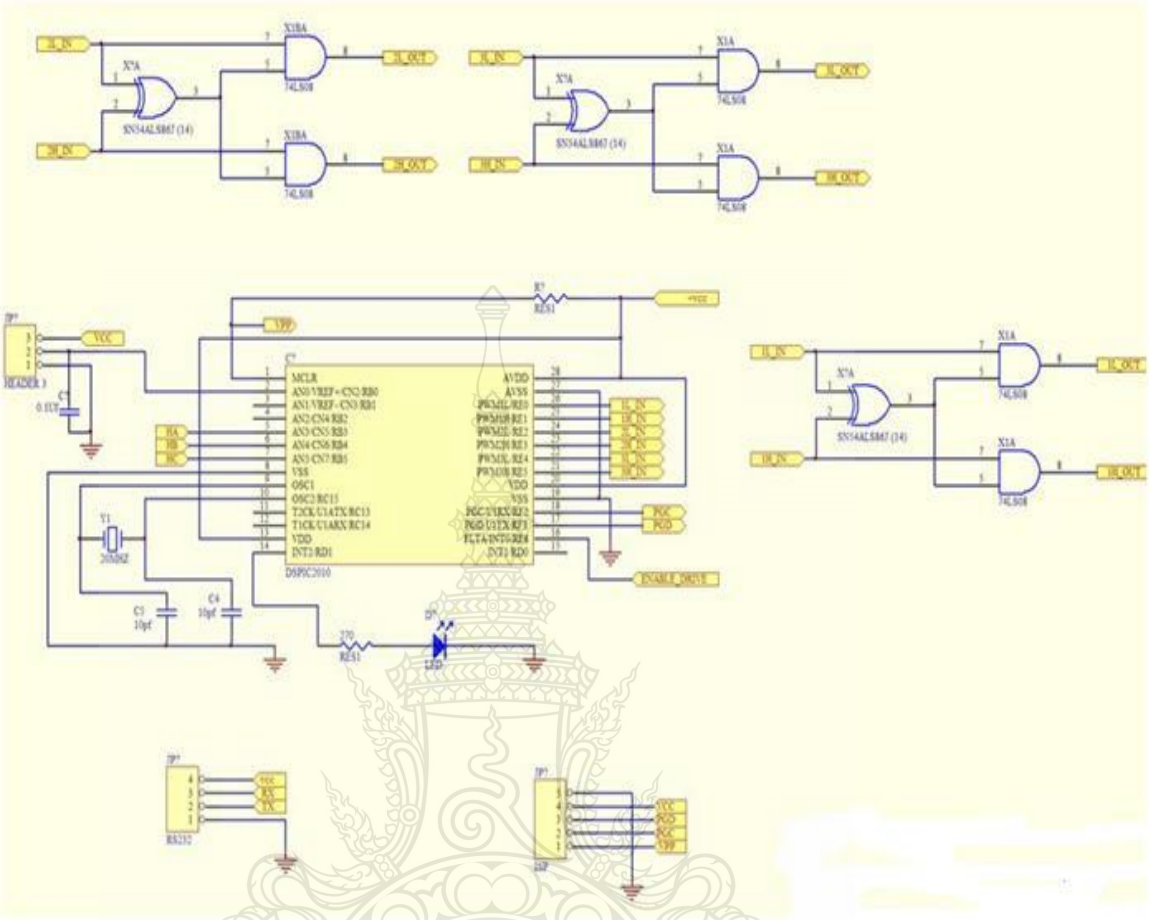
$T = P / \omega$ = 414.63/350 = 1.18 N.m

3.3.5 ชุดความคุมการขับเคลื่อน (BLDC Motor Drive)

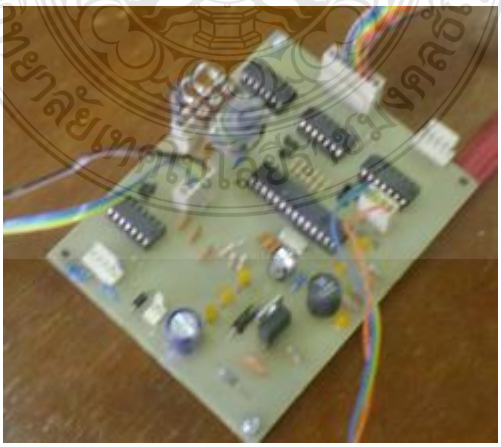
เป็นส่วนควบคุมการทำงานขงมอเตอร์ให้เป็นไปตามในหัวข้อที่ 2.6.1 ซึ่งมีส่วนประกอบหลักๆ 4 ส่วนด้วยกันคือ หน่วยประเมินผลสัญญาณควบคุม (Controller Unit) หน่วยขยายสัญญาณจุดชนวน (Gate Drive Unit) หน่วยของอิเล็กทรอนิกส์สวิทช์ (Power r Electronic Switch) และส่วนรับและขยายสัญญาณตำแหน่งแกนหมุนของมอเตอร์ (Rotor Position Sensing Unit)



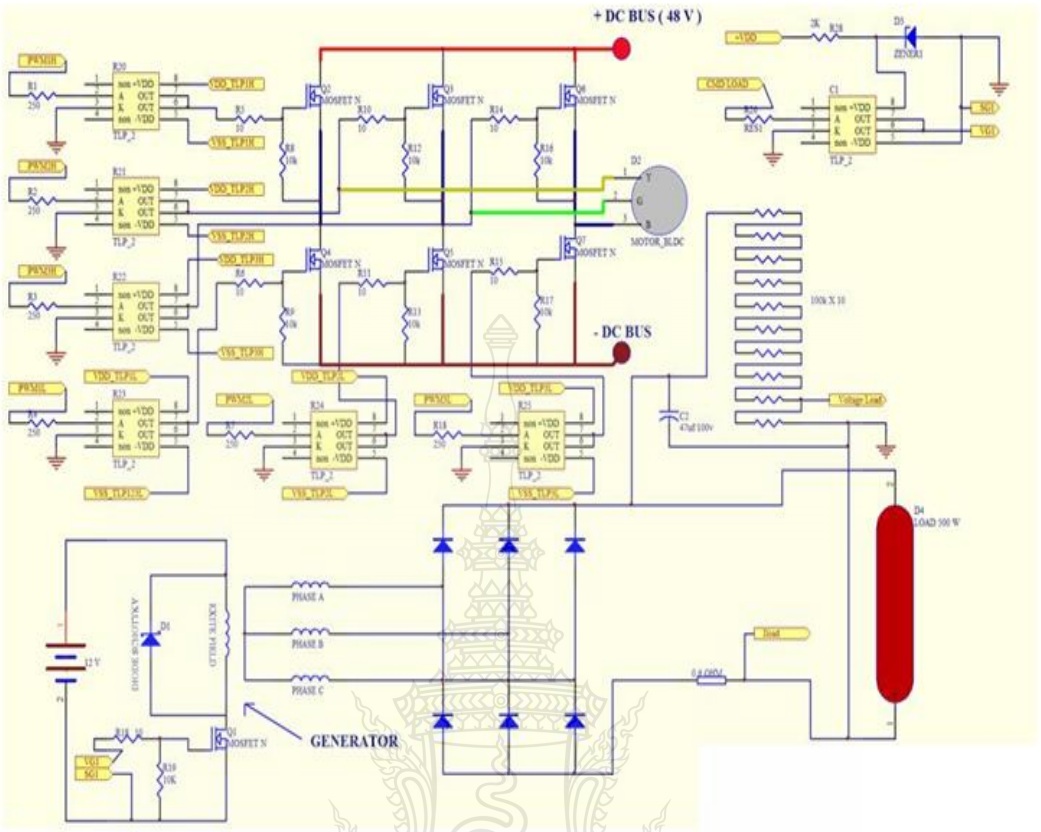
รูปที่ 3.27 ไดอะแกรมชุดควบคุมการขับเคลื่อน BLDC Motor



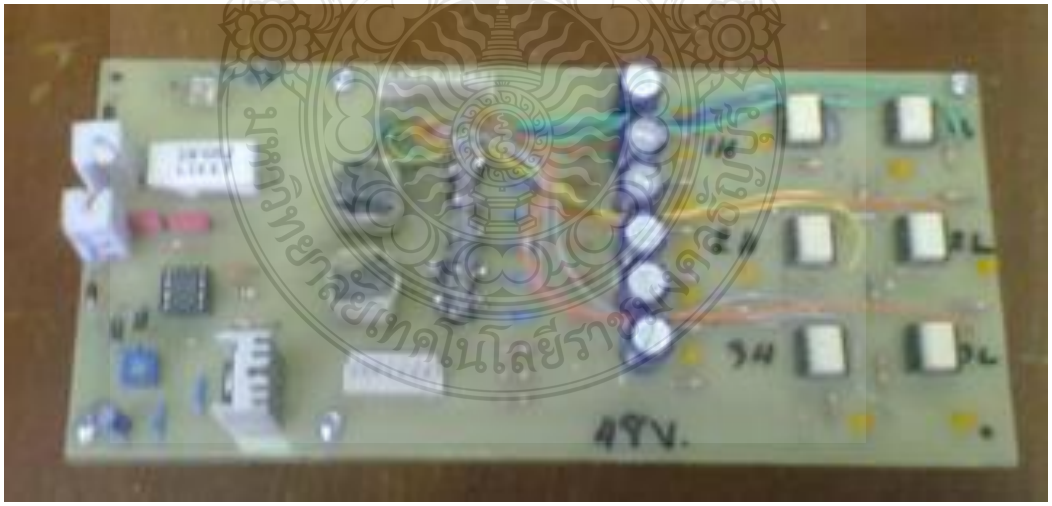
รูปที่ 3.28 วงจรชุด Controller



รูปที่ 3.29 แผงวงจรชุด Controller



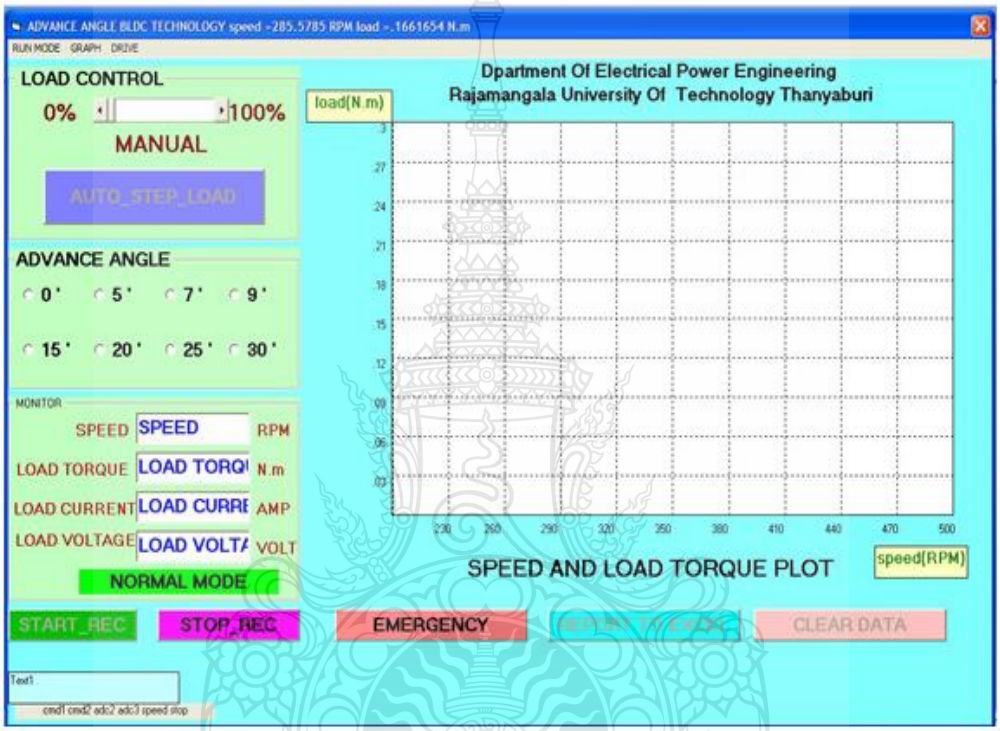
รูปที่ 3.30 วงจรชุด Gate Drive และ Power Switch



รูปที่ 3.31 แผงวงจรชุด Gate Drive

3.3.6 GUI And Data Analysis [15] [16]

GUI And Data Analysis พัฒนาขึ้นจากโปรแกรม Visual Basic เพื่อใช้อำนวยความสะดวกในการศึกษาคุณลักษณะการทำงานของมอเตอร์ในโหมดแรงบิดคงที่และกำลังคงที่ โดยเฉพาะในโหมดกำลังคงที่ ทั้งขณะที่มีและไม่มีการชดเชยมุมเฟสก้าวหน้า ซึ่งลักษณะของหน้าต่างที่ใช้อำนวยความสะดวกในการใช้โปรแกรม แสดงให้เห็นดังรูปที่ 3.31

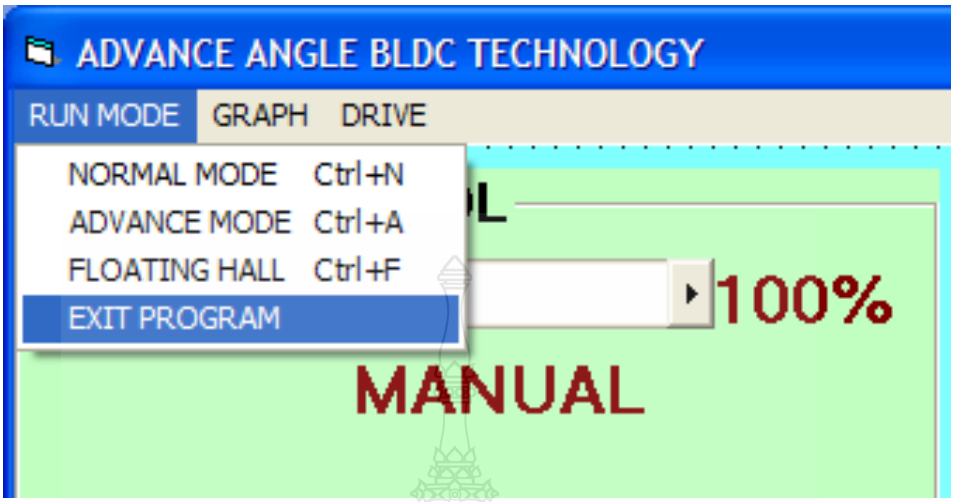


รูปที่ 3.32 หน้าต่างโปรแกรม GUI And Data Analysis

การใช้งานโปรแกรม GUI And Data Analysis แบ่งตามปุ่มต่าง ๆ และ เมนูย่อย ได้ดังต่อไปนี้

- เมนู RUN MODE
- เมนู GRAPH
- เมนู DRIVE
- Frame Load Control
- Frame ADVANCE ANGLE
- Frame MONITOR
- Command Button

เมนู RUN MODE



รูปที่ 3.33 เมนู RUN MODE

NORMAL MODE (Ctrl + N)

ให้สัญญาณ HALL A HALL B และ HALL C ไม่ผ่าน(By Pass) ชุดควบคุมการชดเชยมุมเฟสก้าวหน้าตามรูปที่ 3.11

ADVANCE MODE (Ctrl + A)

ให้สัญญาณ HALL A HALL B และ HALL C ผ่าน ชุดควบคุมการชดเชยมุมเฟสก้าวหน้าตามรูปที่ 3.7

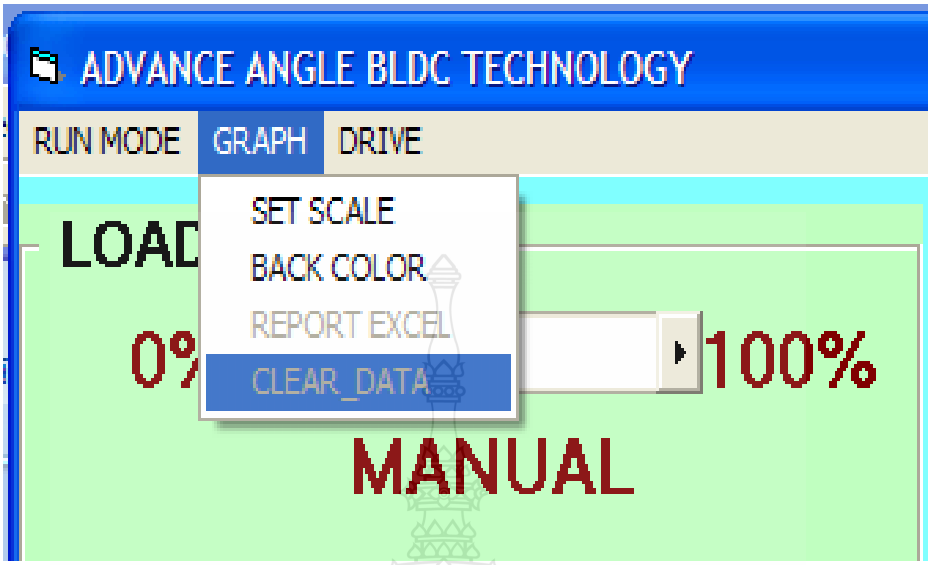
FLOATING HALL (Ctrl + F)

ตัดสัญญาณ HALL A HALL B และ HALL C ผ่านไปยังชุดควบคุมการขับเคลื่อน BLDC Motor (Disable DEMUX block ตามรูปที่ 3.11

EXIT PROGRAM

ออกจากการทำงานของโปรแกรม GUI And Data Analysis

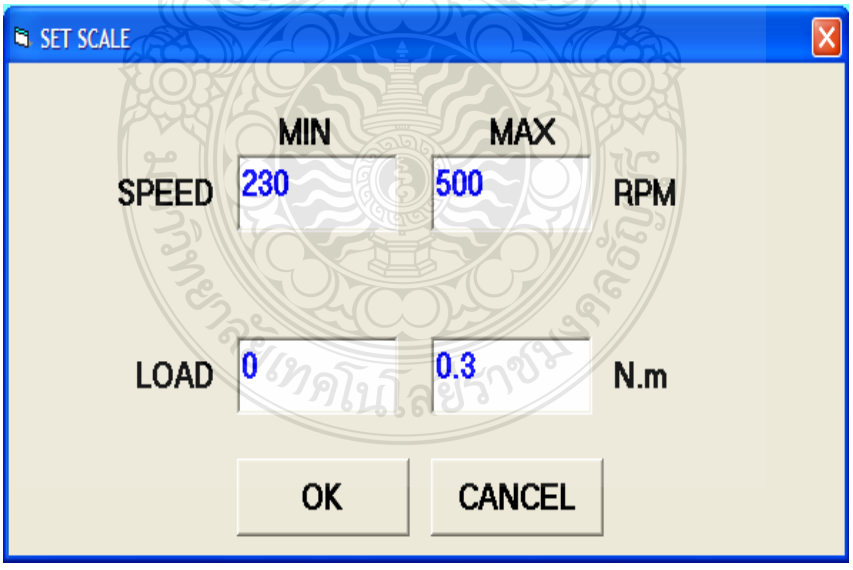
เมนู GRAPH



รูปที่ 3.34 เมนู GRAPH

SET SCALE

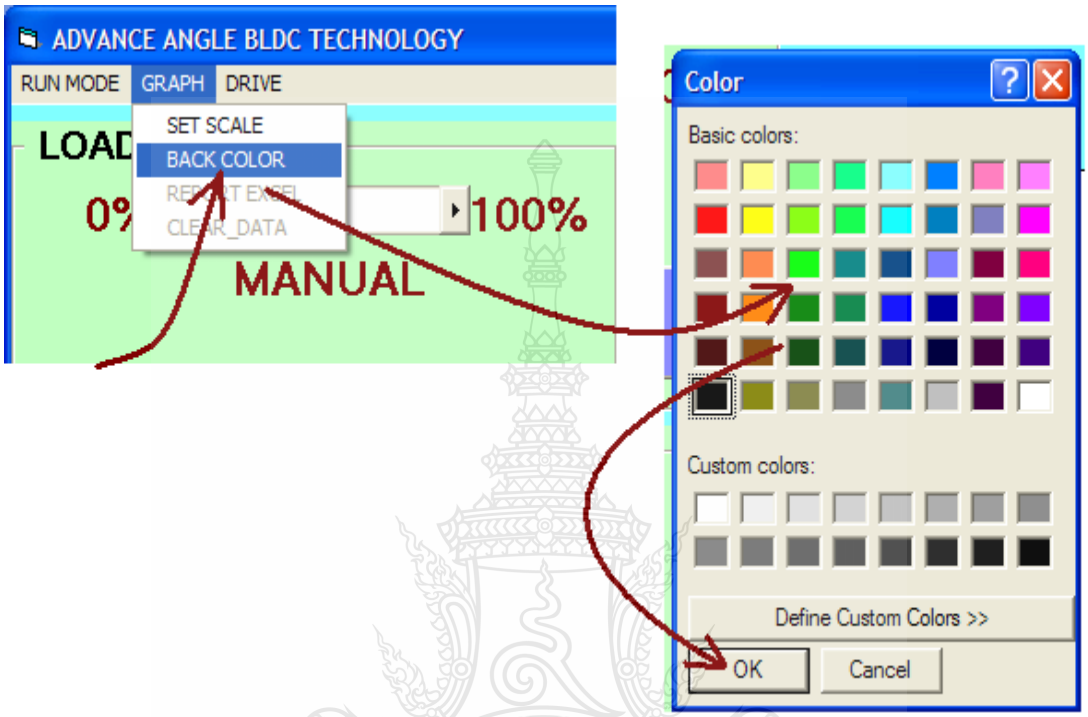
ใช้ปรับสเกลของกราฟให้เหมาะสมกับขนาด แรงบิด และ ความเร็วมอเตอร์



รูปที่ 3.35 หน้าต่างใช้ปรับสเกล แรงบิด และ ความเร็ว มอเตอร์

BACK COLOR

ใช้เปลี่ยนสีพื้นของกราฟ



รูปที่ 3.36 การใช้เมนู BACK COLOR

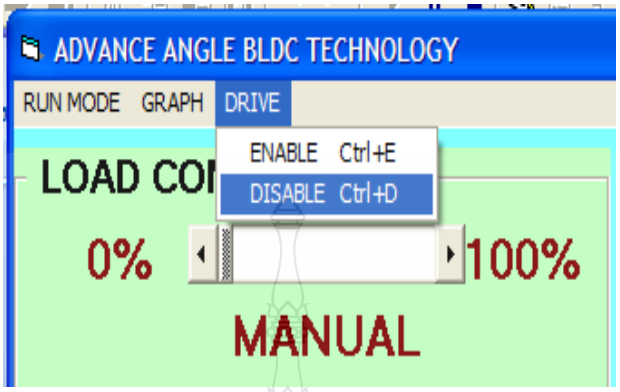
REPORT EXCEL

ใช้ส่งข้อมูลจากกราฟไปยังโปรแกรม EXCEL

CLEAR DATA

ใช้ลบข้อมูลการทดลองก่อนหน้าออกจากหน่วยความจำ (Memory Clear)

เมนู DRIVE



รูปที่ 3.37 เมนู DRIVE

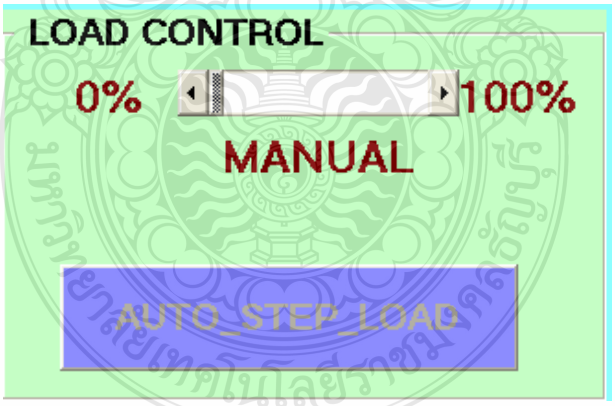
ENABLE (Ctrl + E)

อนุญาตการทำงานของชุดขับเคลื่อน BLDC MOTOR

DISABLE (Ctrl + D)

ไม่อนุญาตการทำงานของชุดขับเคลื่อน BLDC MOTOR

Fram Load Control



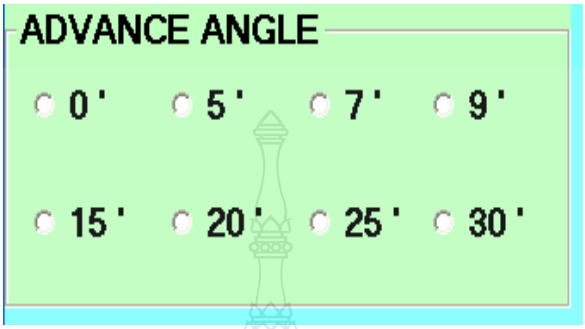
รูปที่ 3.38 Frame Load Control

Frame LOAD CONTROL มีหน้าที่ในการกำหนดการทำงานของโปรแกรม 2 อย่าง คือ

- Hscroll Bar ใช้เปลี่ยนแปลงแรงบิดโหลด (Load Torque) แบบ Manual
- ปุ่ม AUTO_STEP_LOAD ใช้ปรับแรงบิดโหลด (Load Torque) แบบอัตโนมัติ และมีการ Plot Graph ความสัมพันธ์ระหว่างแรงบิดและความเร็วมอเตอร์ ใช้งานคู่กับปุ่ม START_REC

Frame ADVANCE ANGLE

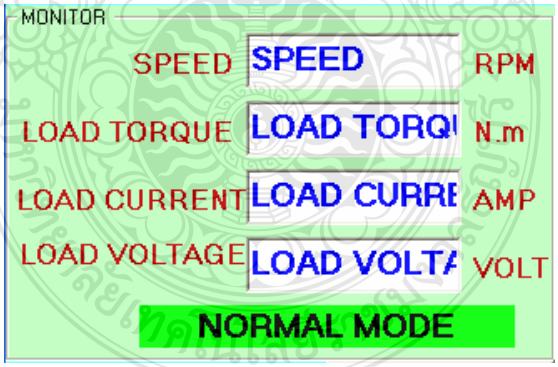
ใช้ปรับค่ามุมชดเชยมุมเฟสล่วงหน้าให้กับหน่วย ประเมินผลการชดเชยมุมเฟสล่วงหน้า ดังแสดงให้เห็นตามรูปที่ 3.38



รูปที่ 3.39 Frame Advance Angle

Fram MONITOR

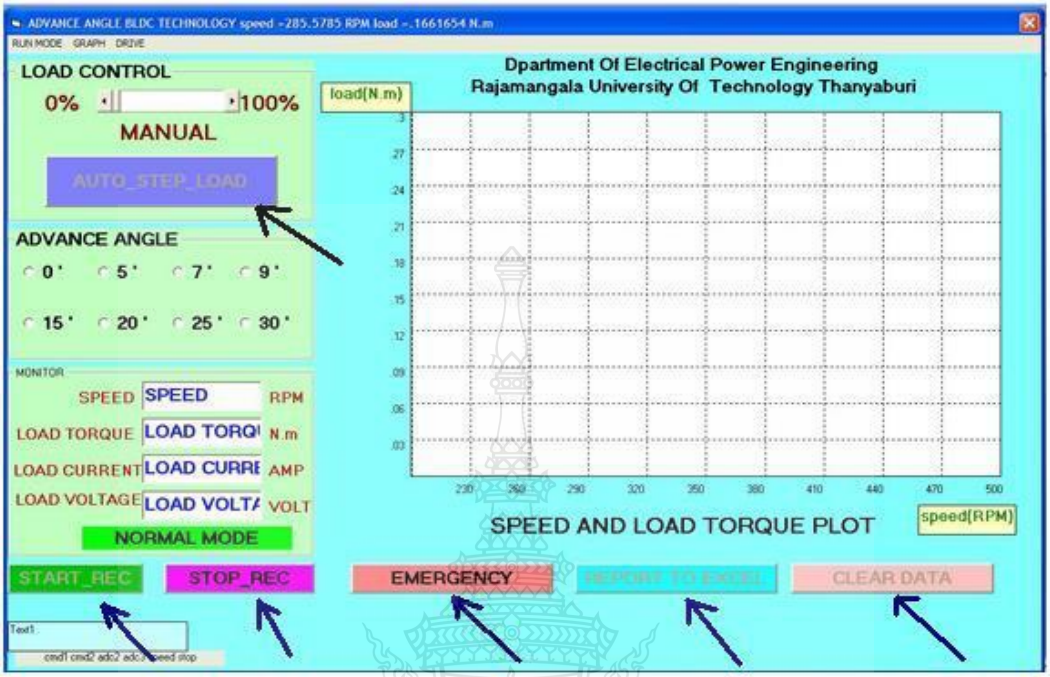
Fram MONITOR ใช้แสดงค่าต่าง ๆ ดังแสดงในรูป 3.31 เช่น ความเร็วมอเตอร์ แรงบิดโหลด กระแสโหลด แรงดันโหลด และ โหมดการทำงานของชุดขับเคลื่อน BLDC MOTOR ตามลำดับ ดังแสดงให้เห็นตามรูปที่ 3.39



รูปที่ 3.40 Frame Monitor

Command Button

Command Buttom ใช้สำหรับการเข้าถึงคำสั่งของโปรแกรมซึ่งมีความสะดวกและคล่องตัวกว่าการเข้าถึงคำสั่งโดยใช้เมนูบาร์ โดยเฉพาะคำสั่งที่ต้องใช้บ่อย ๆ ดังแสดงตำแหน่งลูกษรชี้ตามรูปที่ 3.40



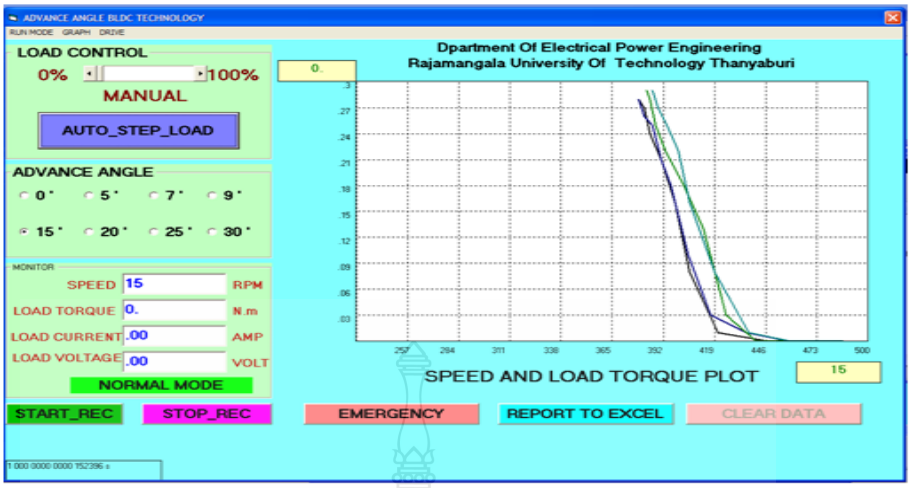
รูปที่ 3.41 Command Button

ปุ่ม AUTO_STEP_LOAD จะทำงานร่วมกับปุ่ม START_REC ในการปรับแรงบิดโหลดเป็นขั้นบันไดแบบอัตโนมัติ และมีการบันทึกกราฟแสดงความสัมพันธ์ระหว่างค่าแรงบิดโหลดและความเร็วมอเตอร์ โดย รายละเอียดลำดับการทำงานเป็นดังนี้

1 เข้าเมนู GRAPH ---→ SET SCALE

2 Click ปุ่ม START_REC

3 Click ปุ่ม AUTO_STEP_LOAD



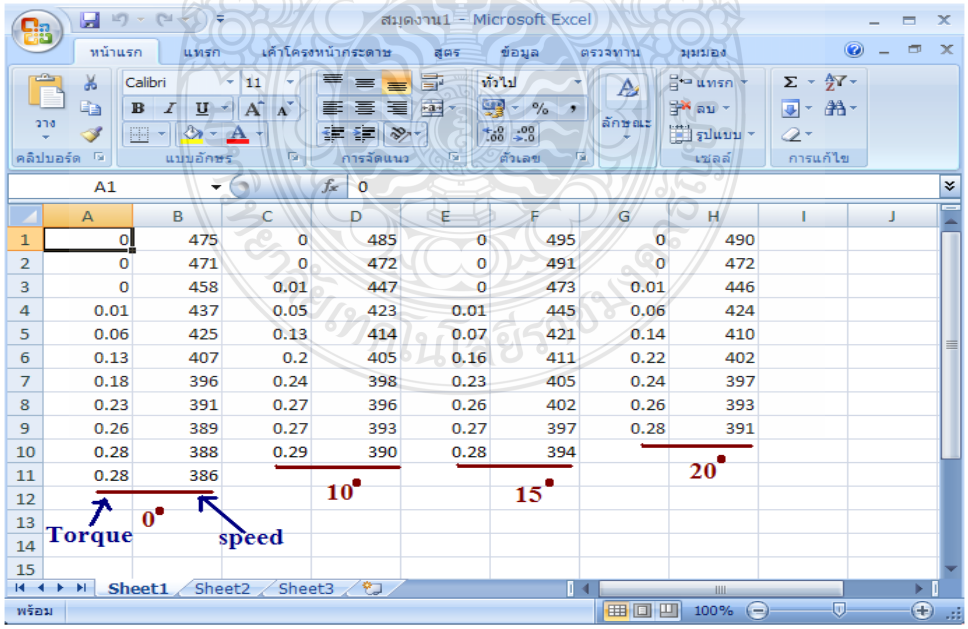
รูปที่ 3.42 กราฟแสดงแรงบิดและความเร็วมอเตอร์ที่มีการชดเชยมุมเฟสก้าวหน้าแตกต่างกัน

ปุ่ม EMERGENCY ใช้กรณีต้องการหยุดการทำนของชุดขับเคลื่อน BLDC MOTOR

ปุ่ม STOP_REC ใช้กรณีต้องการหยุดการบันทึกกราฟและลบหน่วยความจำข้อมูล

ปุ่ม REPORT TO EXCEL ใช้ส่งข้อมูลที่บันทึกในกราฟไปยังโปรแกรม EXCEL เพื่อความสะดวกในการจัดการกับข้อมูลต่อไป

ปุ่ม CLEAR DATA ใช้ลบข้อมูลในหน่วยความจำกรณีต้องการบันทึกข้อมูลชุดใหม่



	A	B	C	D	E	F	G	H
1	0	475	0	485	0	495	0	490
2	0	471	0	472	0	491	0	472
3	0	458	0.01	447	0	473	0.01	446
4	0.01	437	0.05	423	0.01	445	0.06	424
5	0.06	425	0.13	414	0.07	421	0.14	410
6	0.13	407	0.2	405	0.16	411	0.22	402
7	0.18	396	0.24	398	0.23	405	0.24	397
8	0.23	391	0.27	396	0.26	402	0.26	393
9	0.26	389	0.27	393	0.27	397	0.28	391
10	0.28	388	0.29	390	0.28	394		
11	0.28	386						

รูปที่ 3.43 นำค่าแรงบิดและความเร็วมอเตอร์จากกราฟเขียนลงโปรแกรม Excel

บทที่ 4

ผลการทดลอง

ในวิทยานิพนธ์ฉบับนี้จะมุ่งเน้นการวิเคราะห์การทำงานของ BLDC Motor ในย่านกำลังคงที่โดยใช้วิธีการการชดเชยมุมเฟสก้าวหน้า (Advance Angle) โดยการสร้างชุดทดสอบการขับเคลื่อน BLDC Motorที่สามารถเปลี่ยนแปลงค่ามุมเฟสก้าวหน้า และ สามารถแสดงผลความสัมพันธ์ระหว่างแรงบิดและความเร็วมอเตอร์ เพื่อนำไปเปรียบเทียบกับค่าที่ได้จากการจำลองด้วยโปรแกรม MatLab/Simulink โดยการนำเสนอจะแบ่งออกเป็น 4 ส่วน คือ

1 การวัดสัญญาณมุมต่างเฟสของ Hall sensor จาก BLDC Motor H_A H_B H_C

2 การวัดสัญญาณมุมต่างเฟสของ Hall sensor จาก BLDC Motor H_A H_B H_C เทียบกับสัญญาณที่ผ่านหน่วยชดเชยมุมเฟสก้าวหน้า ($H_A^{'}$ $H_B^{'}$ $H_C^{'}$) ที่มุมเฟสก้าวหน้าต่างๆ

3 วิเคราะห์สัญญาณแรงดันตกคร่อมขดลวดมอเตอร์ที่ความเร็วต่างๆ

4 การเปรียบเทียบผลที่ได้จากการทดลองกับผลที่ได้จากการจำลองระบบ

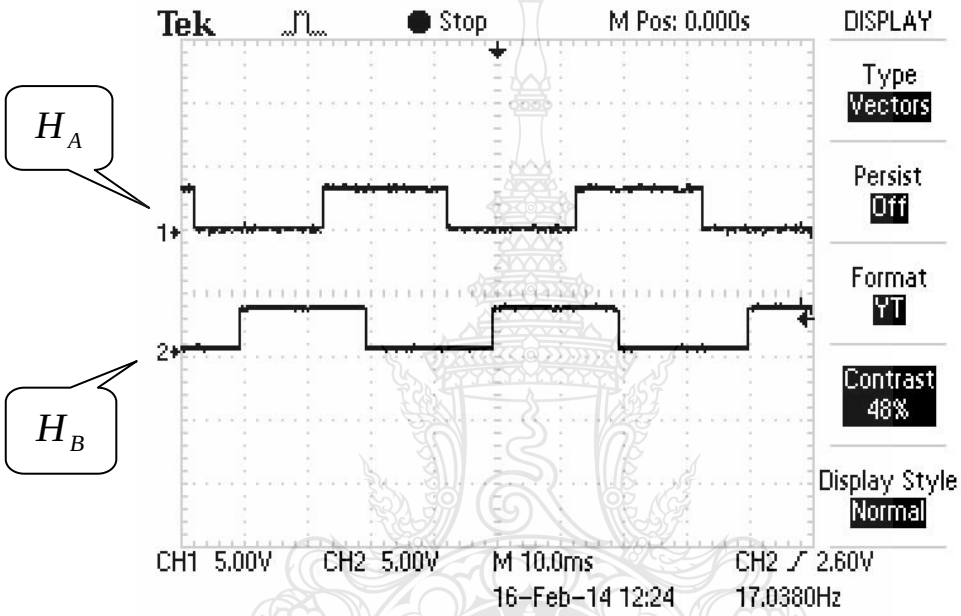
ซึ่งชุดทดสอบในงานวิทยานิพนธ์นี้แสดงดังรูปที่ 4.1

รูปที่ 4.1 ชุดทดสอบ BLDC Motor

4.1 การวัดสัญญาณมุมต่างเฟสของ Hall sensor จาก BLDC Motor (H_A H_B H_C)

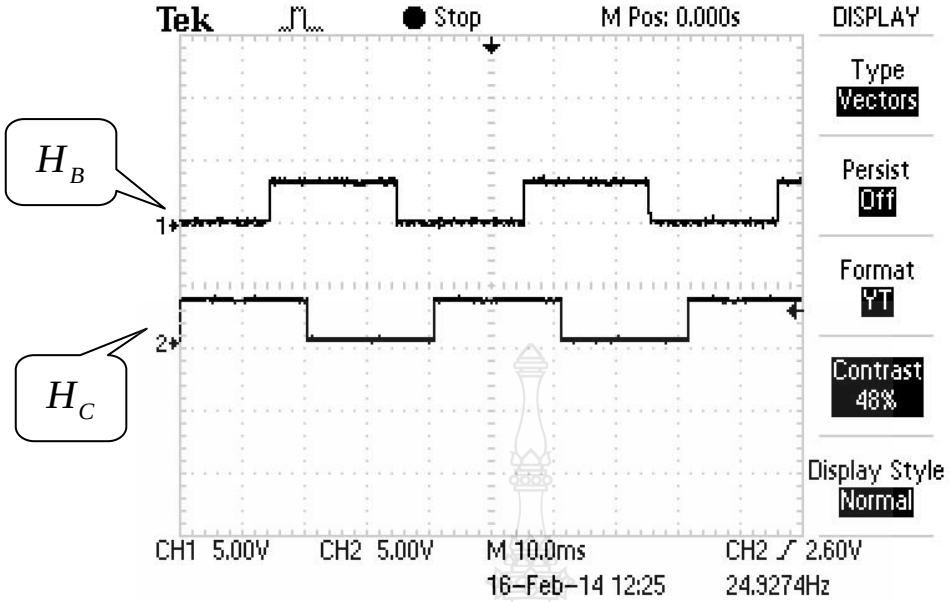
การวัดสัญญาณมุมต่างเฟสของ Hall sensor จาก BLDC Motor เพื่อตรวจสอบมุมต่างเฟส ระหว่าง H_A กับ H_B H_B กับ H_C และ H_C กับ H_A ก่อนส่งเข้าหน่วยชดเชยมุมเฟสก้าวหน้า

4.1.1 วัดสัญญาณมุมต่างเฟสของสัญญาณ H_A กับ H_B



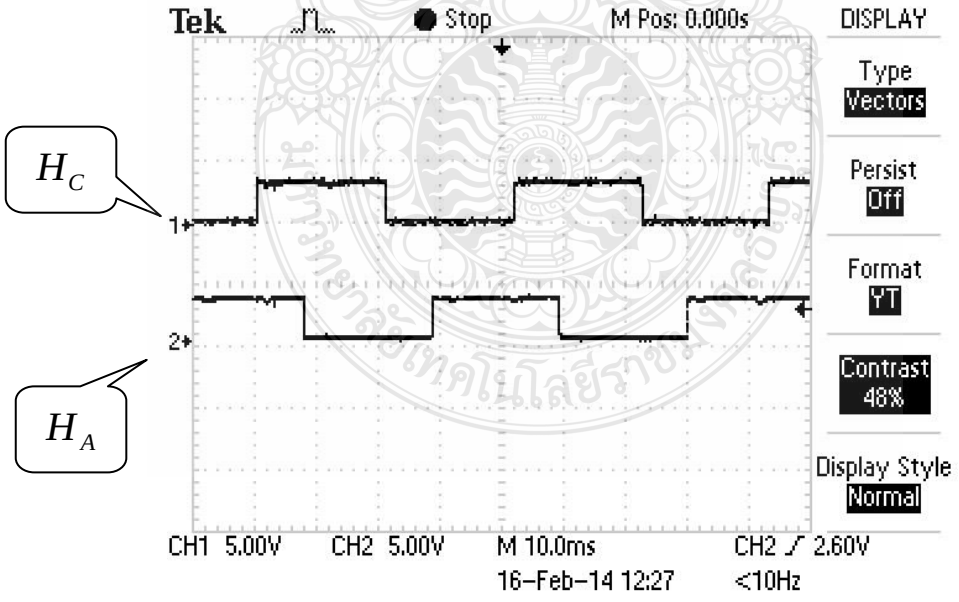
รูปที่ 4.2 สัญญาณ H_A กับ H_B

4.1.2 วัดสัญญาณมุมต่างเฟสของสัญญาณ H_B กับ H_C



รูปที่ 4.3 สัญญาณ H_B กับ H_C

4.1.3 วัดสัญญาณมุมต่างเฟสของสัญญาณ H_C กับ H_A

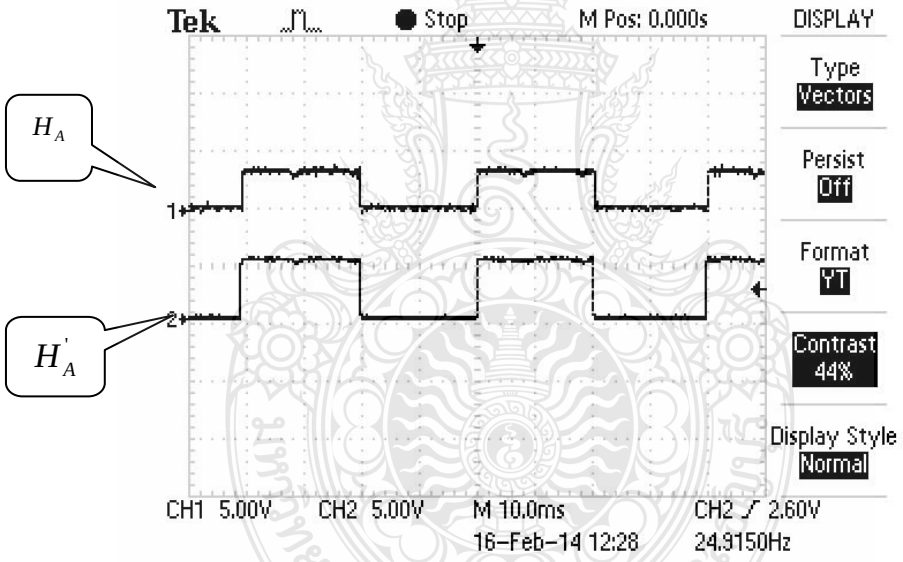


รูปที่ 4.4 สัญญาณ H_C กับ H_A

การตรวจสอบสัญญาณของ Hall Sensor ที่ออกจากมอเตอร์ก่อนส่งเข้าหน่วยควบคุมการชดเชยมุมเฟสก้าวหน้า เพื่อเช็คมุมต่างเฟสระหว่าง H_A กับ H_B H_B กับ H_C และ H_C กับ H_A ตามลำดับ ผลที่ได้แสดงดังรูปที่ 4.2 4.3 และ 4.4 ซึ่งผลที่ได้เป็นไปตาม [12] และ ทำการตรวจสอบสัญญาณที่ผ่านหน่วยควบคุมการชดเชยมุมเฟสก้าวหน้าที่มุมชดเชยเป็นศูนย์ (Phase Advance = 0) และทำการวัดค่าความต่างเฟสของ H_A กับ $H_B^{'}$ H_B กับ $H_C^{'}$ และ H_C กับ $H_A^{'}$ ดังแสดงในรูปที่ 4.5 4.6 และ 4.7 ผลที่ได้ก็เป็นไปตาม [12] เช่นกัน

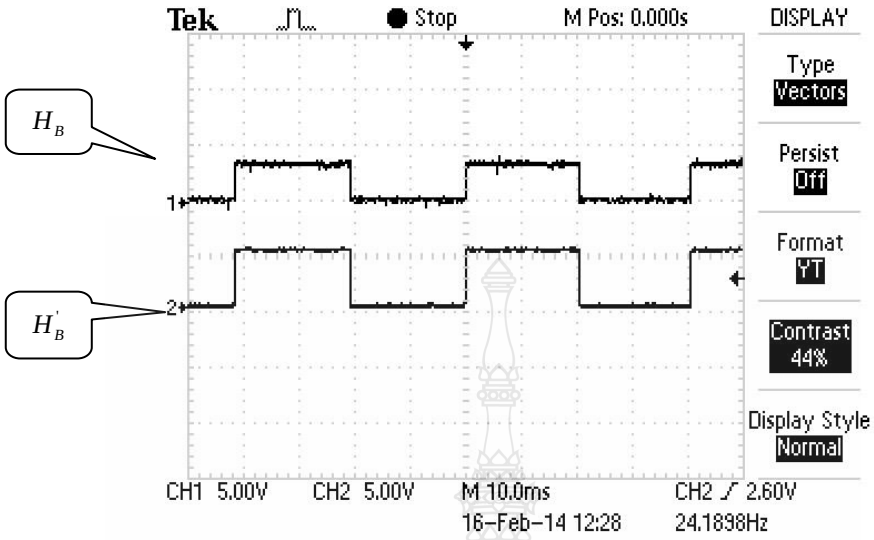
4.2 การวัดสัญญาณมุมต่างเฟสของ Hall sensor จาก BLDC Motor H_A H_B H_C เทียบกับสัญญาณที่ผ่านหน่วยชดเชยมุมเฟสก้าวหน้า ($H_A^{'}$ $H_B^{'}$ $H_C^{'}$)ที่มุมเฟสก้าวหน้าต่าง ๆ

4.2.1 วัดสัญญาณมุมต่างเฟสของสัญญาณ H_A กับ $H_A^{'}$ ที่มุมเฟสก้าวหน้าเป็น 0 องศา



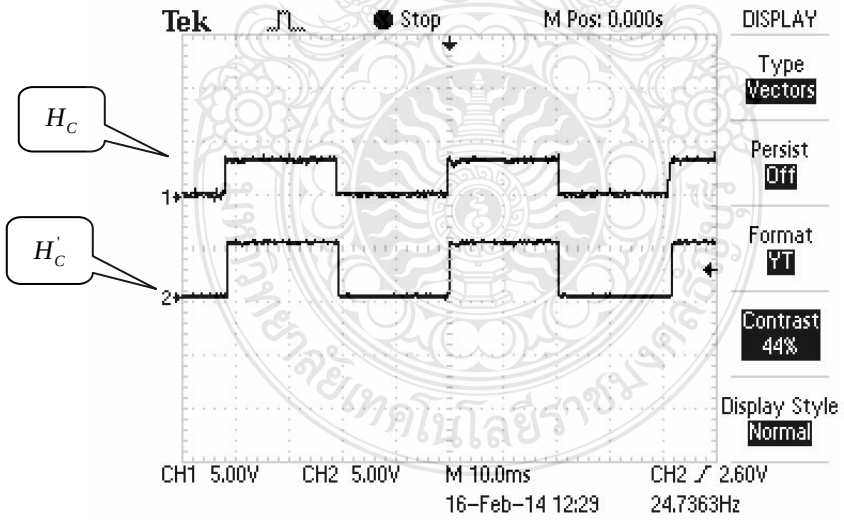
รูปที่ 4.5 สัญญาณ H_A กับ $H_A^{'}$ ที่มุมเฟสก้าวหน้า 0 องศา

4.2.2 วัดสัญญาณมุมต่างเฟสของสัญญาณ H_B กับ $H_B^{'}$ ที่มุมเฟสก้าวหน้าเป็น 0 องศา



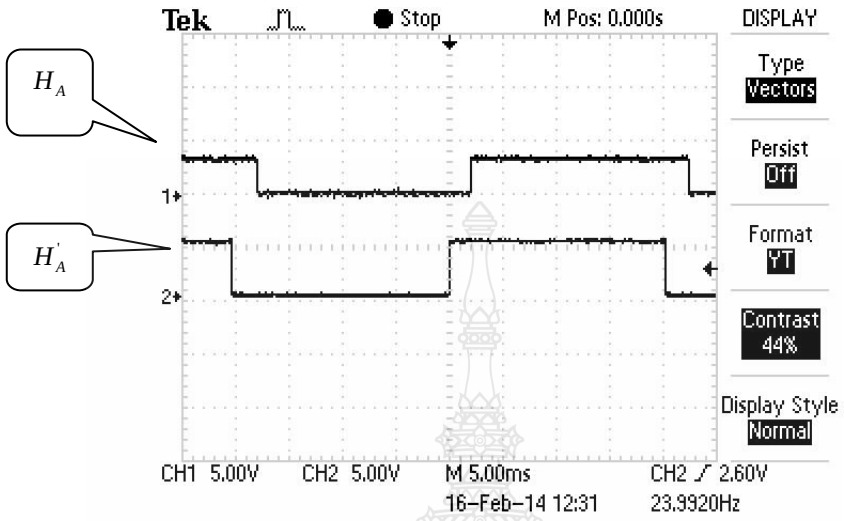
รูปที่ 4.6 สัญญาณ H_B กับ $H_B^{'}$ ที่มุมเฟสก้าวหน้า 0 องศา

4.2.3 วัดสัญญาณมุมต่างเฟสของสัญญาณ H_C กับ $H_C^{'}$ ที่มุมเฟสก้าวหน้าเป็น 0 องศา



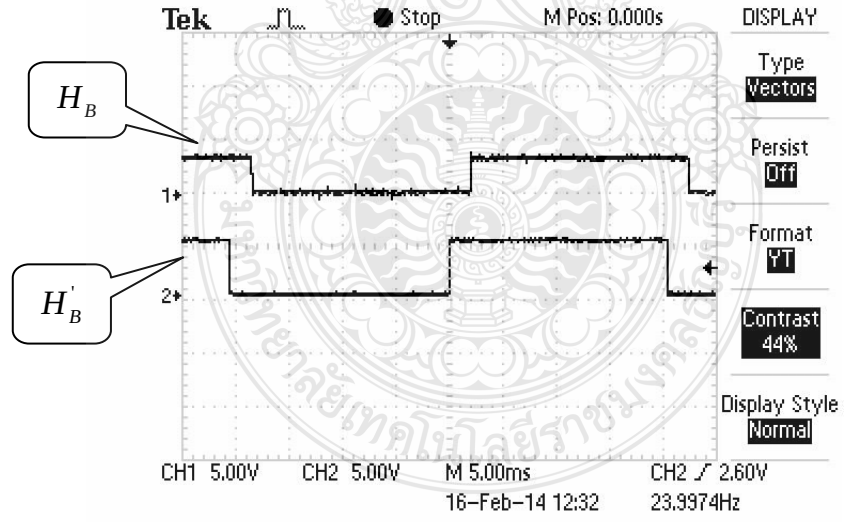
รูปที่ 4.7 สัญญาณ H_C กับ $H_C^{'}$ ที่มุมเฟสก้าวหน้า 0 องศา

4.2.4 วัดสัญญาณมุมต่างเฟสของสัญญาณ H_A กับ $H_A^{'}$ ที่มุมเฟสก้าวหน้า 15 องศา



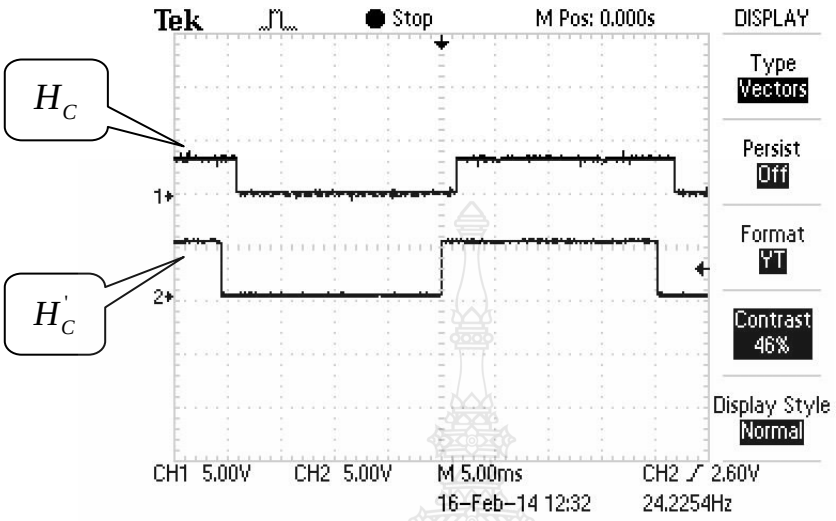
รูปที่ 4.8 สัญญาณ H_A กับ $H_A^{'}$ ที่มุมเฟสก้าวหน้า 15 องศา

4.2.5 วัดสัญญาณมุมต่างเฟสของสัญญาณ H_B กับ $H_B^{'}$ ที่มุมเฟสก้าวหน้า 15 องศา



รูปที่ 4.9 สัญญาณ H_B กับ $H_B^{'}$ ที่มุมเฟสก้าวหน้า 15 องศา

4.2.6 วัดสัญญาณมุมต่างเฟสของสัญญาณ H_C กับ $H_C^{'}$ ที่มุมเฟสก้าวหน้า 15 องศา



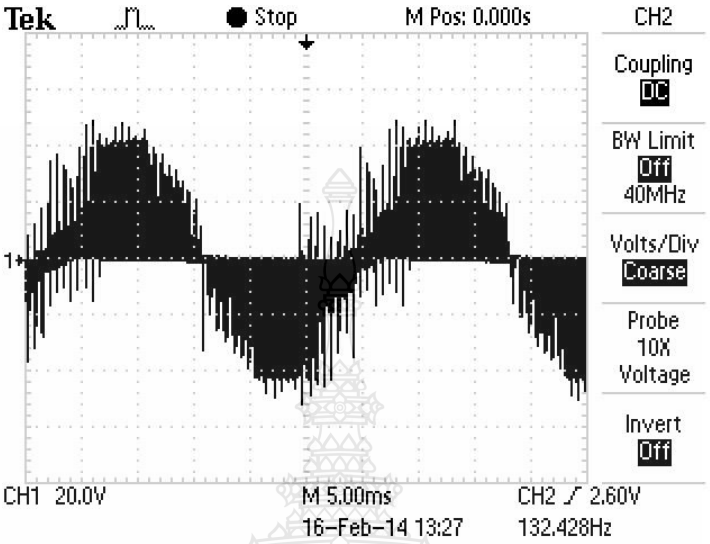
รูปที่ 4.10 สัญญาณ H_C กับ $H_C^{'}$ ที่มุมเฟสก้าวหน้า 15 องศา (Advance Angle $=15^{o}$)

ผลการตรวจสอบสัญญาณของ Hall Sensor ที่ออกจากมอเตอร์เทียบกับสัญญาณที่ผ่านหน่วยควบคุมการชดเชยมุมเฟสก้าวหน้า (H_A กับ $H_A^{'}$ H_B กับ $H_B^{'}$ และ H_C กับ $H_C^{'}$) โดยที่ค่ามุมชดเชย 0^{o} แสดงดังรูปที่ 4.5 4.6 และ 4.7 จะเห็นได้ว่าคู่สัญญาณดังกล่าวจะอินเฟสกัน

ส่วนที่ค่ามุมชดเชย 15^{o} ดังแสดงดังรูปที่ 4.8 4.9 และ 4.10 จะเห็นได้ว่าสัญญาณ $H_A^{'}$ จะมีเฟสนำหน้า H_A $H_B^{'}$ มีเฟสนำหน้า H_B และสัญญาณ $H_C^{'}$ มีเฟสนำหน้า H_C ตามลำดับ ซึ่งเมื่อนำค่าเวลาของเฟสที่นำหน้าคิดเป็น องศา จะได้ค่าอยู่ที่ $15^{o} \pm 15\ \%$

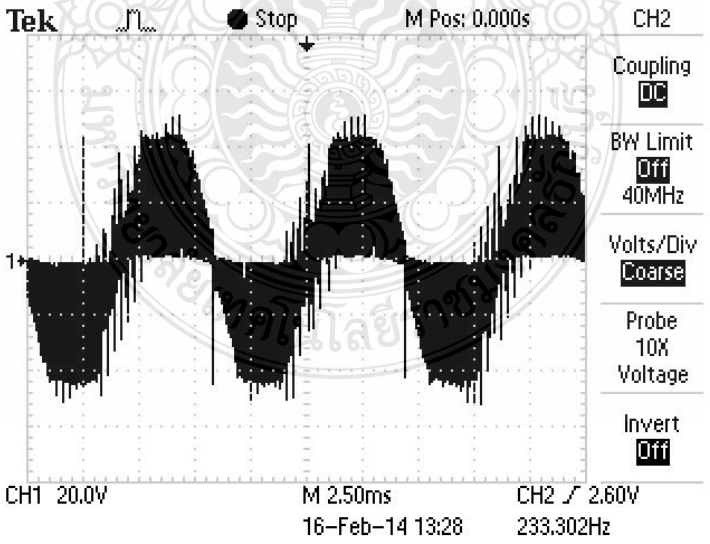
4.3 วิเคราะห์สัญญาณแรงดันตกคร่อมขดลวดมอเตอร์ที่ความเร็วต่าง ๆ

4.3.1 วัดสัญญาณแรงดันตกคร่อมขดลวด Phase A B ที่ความเร็วรอบมอเตอร์ 109 RPM



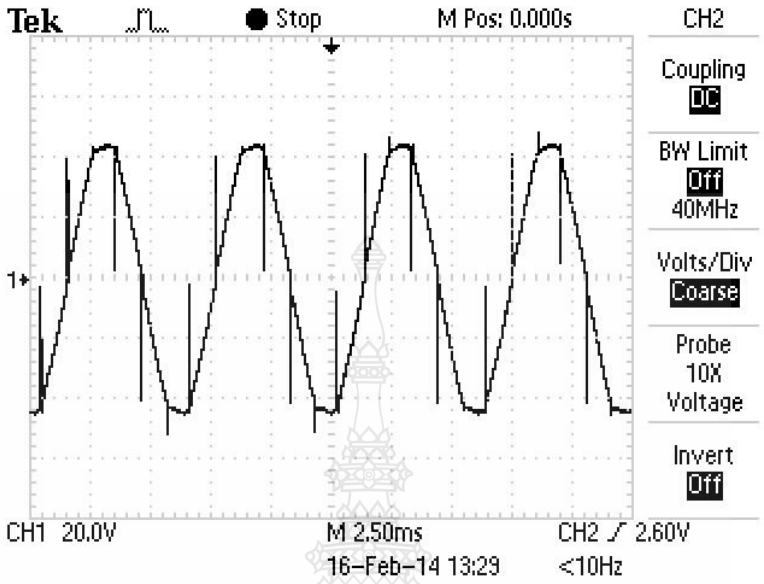
รูปที่ 4.11 แรงดันตกคร่อมชุดขดลวด V_{AB} ของมอเตอร์ที่ความเร็วรอบ 109 RPM

4.3.2 วัดสัญญาณแรงดันตกคร่อมขดลวด Phase A B ที่ความเร็วรอบมอเตอร์ 389 RPM



รูปที่ 4.12 แรงดันตกคร่อมชุดขดลวด V_{AB} ของมอเตอร์ที่ความเร็วรอบ 300 RPM

4.3.3 วัดสัญญาณแรงดันตกคร่อมขดลวด Phase A B ที่ความเร็วรอบมอเตอร์ 484 RPM



รูปที่ 4.13 แรงดันตกคร่อมชุดขดลวด V_{AB} ของมอเตอร์ที่ความเร็วรอบ 428 RPM

จากการวัดสัญญาณแรงดันตกคร่อมขดลวดมอเตอร์ที่ความเร็ว 109 RPM 389 RPM และ 484 RPM ดังแสดงตามรูปที่ 4.14 4.15 และ 4.16 ตามลำดับ จะเห็นได้ว่าแรงดันตกคร่อมขดลวดมอเตอร์ มีลักษณะรูปคลื่นไฟฟ้ากระแสสลับ ที่ถูกผสมด้วยสัญญาณ PWM ซึ่งพอจะสังเกตเห็นได้ว่าความเร็วของมอเตอร์จะเป็นสัดส่วนโดยตรง กับ ค่า Duty Cycle ของสัญญาณ PWM ดูได้จากรูปที่ 4.16 เป็นช่วงที่ค่า Duty Cycle 100 % เป็นช่วงที่มอเตอร์ให้ความเร็วสูงสุด

4.4 การเปรียบเทียบผลที่ได้จากการทดลองกับผลที่ได้จากการจำลองระบบ

4.4.1 ทดสอบการทำงานของมอเตอร์จากชุดทดสอบที่สร้างขึ้น

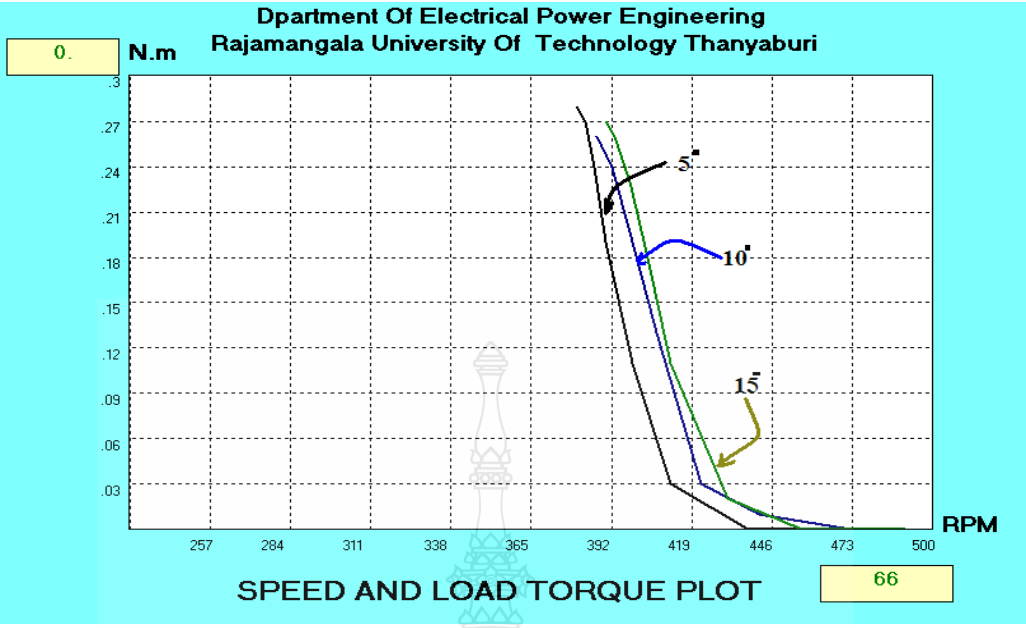
โดยพารามิเตอร์ของมอเตอร์ตามตารางที่ 4.1

ตารางที่ 4.1 พารามิเตอร์ของมอเตอร์ที่ใช้ในการทดสอบ

คุณลักษณะมอเตอร์	ขนาด	หน่วย
พิกัดแรงดันที่จ่ายให้กับมอเตอร์	48	volt
ความต้านทานของชุดขดลวดมอเตอร์(R)	1.5	Ohm
ค่าความเหนี่ยวนำของชุดขดลวดมอเตอร์ (L)	1.92	mH
ค่าโมเมนต์ความเฉื่อยที่เพลามอเตอร์(J)	$1 x 10^{-4}$	kgm^2
ค่าสัมประสิทธิ์ความหน่วงที่เพลา(B)	$1 x 10^{-4}$	-
จำนวนขั้วแม่เหล็ก (P)	46	Pole
ค่าคงที่ (Kw)	0.0445	-

ตารางที่ 4.2 ผลการทดลองจากชุดทดสอบที่สร้างขึ้น

step	5^o		10^o		15^o	
	torque	speed	torque	speed	torque	speed
1	0	460	0	480	0	483
2	0	438	0	472	0	456
3	0.05	412	0.01	442	0.02	451
4	0.11	399	0.03	422	0.11	412
5	0.19	390	0.13	407	0.18	404
6	0.24	386	0.19	399	0.23	398
7	0.27	383	0.24	392	0.26	393
8	0.28	380	0.26	387	0.27	390

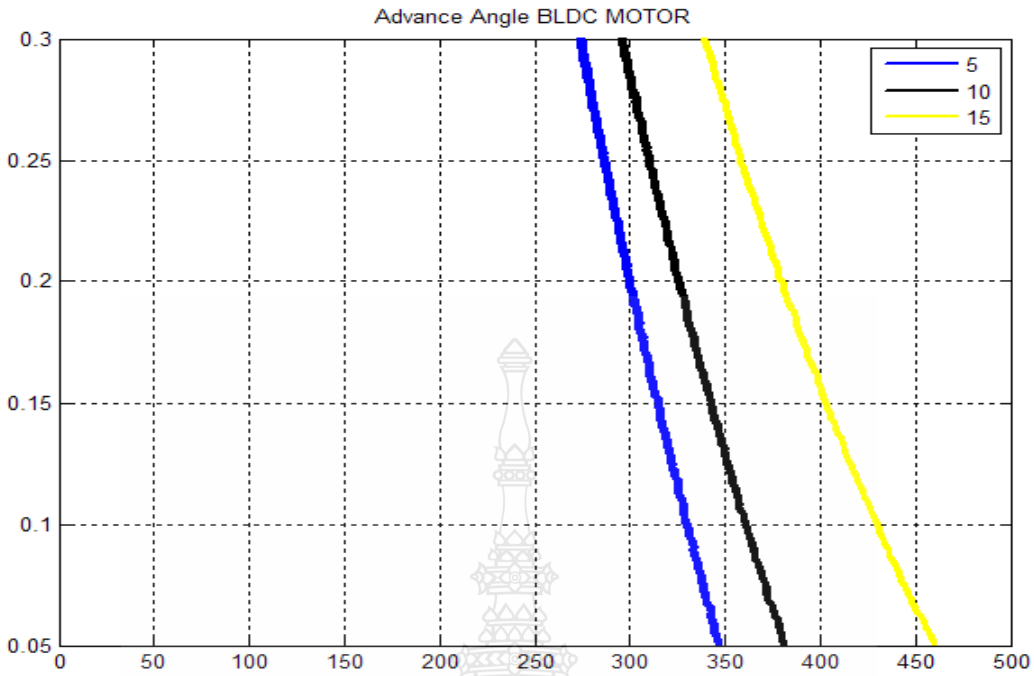


รูปที่ 4.14 กราฟแรงบิดโหลดกับความเร็วมอเตอร์ที่ได้โปรแกรม GUI And Data Analysis

4.2.2 การจำลองการทำงานของมอเตอร์ด้วยโปรแกรม Matlab/Simulink โดยใช้พารามิเตอร์ของมอเตอร์ตามตารางที่ 4.1

ตารางที่ 4.3 ผลที่ได้จากการจำลอง

step	5^o		10^o		15^o	
	torque	speed	torque	speed	torque	speed
1	0.05	346	0.05	378	0.05	458
2	0.08	334	0.08	364	0.08	435
3	0.12	322	0.12	350	0.12	414
4	0.16	311	0.16	333	0.16	395
5	0.2	300	0.2	324	0.2	379
6	0.23	290	0.23	314	0.23	364
7	0.26	282	0.26	305	0.26	351
8	0.3	273	0.3	299	0.3	339



รูปที่ 4.15 กราฟแรงบิดโหลดกับความเร็วมอเตอร์จากโปรแกรม MatLab/Simulink

จากการเปรียบเทียบผลที่ได้ระหว่างชุดทดสอบที่สร้างขึ้น กับ ผลที่ได้จากการจำลอง ซึ่งดูได้จากตารางที่ 4.2 และ 4.3 ซึ่งเป็นการเปรียบเทียบค่าของ แรงบิดและความเร็วมอเตอร์ที่มีการชดเชยมุมเฟสก้าวหน้า ที่ 5^{o} 10^{o} และ 15^{o} ตามลำดับ จะเห็นได้ว่าผลจากชุดทดสอบและจากการจำลองมีค่าเอนเอียงไปในทิศทางเดียวกัน

บทที่ 5
สรุปผลการวิจัยและข้อเสนอแนะ

วิทยานิพนธ์นี้ได้ออกแบบระบบควบคุม และ ทดสอบ การทำงาน BLDC Motor ในย่านการทำงานกำลังคงที่ ซึ่งระบบที่สร้างขึ้นประกอบด้วย 2 ส่วน หลัก ๆ

1 แผงวงจรควบคุม(Hardware) ประกอบด้วย

- แผงวงจรชุดควบคุมการขับเลื่อน BLDC Motor
- แผงวงจรชุดควบคุมการปรับมุมเฟสก้าวหน้า และ ติดต่อสื่อสารกับ คอมพิวเตอร์

2 ส่วนของโปรแกรม(Soft Ware) ประกอบด้วย

- ระบบปฏิบัติการในชุดขับเคลื่อน (BLDC Motor Drive Systems) และระบบปฏิบัติการ การชดเชยมุมเฟสก้าวหน้า ซึ่งพัฒนาขึ้นจาก PIC C Compiler
- ระบบปฏิบัติการ GUI And Data Analysis พัฒนาจากโปรแกรม Visual Basic

5.1 สรุปผลการวิจัย

ผลการทดสอบการทำงานของ BLDC Motor ในย่านกำลังคงที่ด้วยชุดควบคุมที่สร้างขึ้นดังกล่าวข้างต้น ดูจากกราฟ รูปที่ 4.14,4.15 และ ตารางที่ 4.2 และ 4.3 ซึ่งได้มาจากทดสอบด้วยชุดควบคุมที่สร้างขึ้น กับ ค่าที่ได้จากการจำลองด้วยโปรแกรม MatLab/simulink ผลที่ได้มีค่าโน้มเอียงไปในทิศทางเดียวกันดังแสดงในตาราง 5.1

ตารางที่ 5.1 เปรียบเทียบผลที่ได้จากชุดควบคุมที่สร้างขึ้นเทียบกับผลที่ได้จากการจำลอง

ลำดับขั้นการเพิ่มแรงบิดโหลด	ค่าแรงบิดและความเร็วมอเตอร์จากชุดควบคุมที่สร้างขึ้น		ค่าแรงบิดและความเร็วมอเตอร์จากการจำลองด้วย MATHLAB/Simulink	
	Advance 15		Advance 15	
1	0.23	398	0.23	364
2	0.26	393	0.26	351
3	0.27	390	0.3	339

5.2 ข้อเสนอแนะและแนวทางการพัฒนา

วิทยานิพนธ์นี้เป็นแนวทางเบื้องต้นในการศึกษาและพัฒนาสร้างชุดควบคุมการขับเคลื่อน BLDC Motor ให้เหมาะสมกับการนำไปใช้งานในทางปฏิบัติ เพื่อให้ได้ประสิทธิภาพการทำงานของมอเตอร์สูงสุด และ เป็นแนวทางเบื้องต้น ในการสร้างชุดทดสอบมอเตอร์ เพื่อใช้ในห้องปฏิบัติการ และ เป็นแนวทางสำหรับผู้ที่มีความสนใจงานทางด้านเทคโนโลยีการขับเคลื่อน ต่อไป

รายการอ้างอิง

[1] Hamid A. Toliyat , Tilak Gopalarathnam "Brushless DC Machines/Electronics" Texas A&M University

[2] BinhMinh Nguyen, Minh C. Ta "Phase Advance Approach to Expand the Speed Range of Brushless DC Motor "

[3] Chun-Lung Chiu, Yie-Tone Chen, Yu-Hsiang Shen, and Ruey-Hsun Liang "An Accurate Automatic Phase Advance Adjustment of Brushless DC Motor "

[4] RÓBERT ISTVÁN LŐRINCZ, MIHAI EMANUEL BASCH, DAVID CRISTEA, IVAN BOGDANOV, VIRGIL TIPONUT " Hardware Implementation of Field-Weakening BLDC Motor Control"

[5] EDWARD C. LEE ." APPLICATION OF BRUSHLESS DC DRIVES IN BLOW MOLDING" POWERTEC INDUSTRIAL CORPORATION

[6] Shane W. Colton "Design and Prototyping Methods for Brushless Motors and Motor Control"

[7] ANAND SATHYAN "DIGITAL PWM CONTROL OF BRUSH-LESS DC (BLDC)" DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING Illinois Institute of Technology

[8] Microchip Technology Inc "DSPIC Digital Signal Controller Workshop"

Hamid A. Toliyat Texas A&M UniversityTilak Gopalarathnam Texas A&M University "**AC** Machines Controlled as DC Machines (Brushless DC Machines/Electronics)"

[9] B. Tibor, V. Fedak and F. Durovsky, "Modeling And simulation of the BLDC Motor in MATHLAB GUI", 2011

[10] Ion Boldea "Electric Drive" S.A. Nasar 1998

[11] Custom Computer Services, Inc "PCD C Compiler Reference Manual" April 2009

[12] Microchip Technology Inc **"**High-Performance, 16-bit Digital Signal Controllers"

[13] Devendra Rai "BRUSHLESS DC MOTOR SIMULINK SIMULATOR" Department of Electronics and Communication Engineering,National Institute of Technology Karnataka

[14] The MathWorks, Inc. “ MATHLAB “COPYRIGHT 1984–2013

[15] ธนพล ฉันจรัสวิชัย การพัฒนาโปรแกรมด้วย Visual Basic V4.0 พ.ศ. 2541

[16] กิตติ ภัคดีวัฒนะกุล , จำลอง ครูอุตสาหะ Visual Basic 5 ฉบับโปรแกรมเมอร์ พ.ศ. 2541

ภาคผนวก

ภาคผนวก ก

Source Code Advance Angle Processing และ Central Processing Unit

Source Code Advance angle processing And Central Processing Unit

Source Code Advance angle processing

```
#include <16F818.h>

#device adc=8

#FUSES NOWDT                    //No Watch Dog Timer

#FUSES HS                       //High speed Osc (> 4mhz for PCM/PCH) (>10mhz for PCD)

#FUSES NOPUT                    //No Power Up Timer

#FUSES MCLR                     //Master Clear pin enabled

#FUSES NOBROWNOUT               //No brownout reset

#FUSES NOLVP                    //No low voltage prgming, B3(PIC16) or B5(PIC18) used for I/O

#FUSES NOCPD                    //No EE protection

#FUSES NOWRT                    //Program memory   not write protected

#FUSES NODEBUG                  //No Debug mode for ICD

#FUSES NOPROTECT                //Code not protected from reading

#use delay(clock=20000000)   #use rs232 (baud=9600,parity=N,xmit=PIN_A3,rcv=PIN_A2,bits=8)

struct sel_adv{

int  NON :2 ;    int  data  :3;

int  non1   :3;
```

```
}select_adv;

#locate select_adv=0x005

int8 t0con;

#locate t0con= 0x081

#bit t0se = t0con.4          #BIT t0cs = t0con.5       #bit psa  = t0con.3

int8 trisb;                   #locate trisb = 0x086

#bit trisb7 = trisb.7        #bit trisb6 = trisb.6       #bit trisb5 = trisb.5

#bit trisb4 = trisb.4        #bit trisb3 = trisb.3       #bit trisb2 = trisb.2

#BIT trisb1  =trisb.1        #BIT trisb0  =trisb.0

int8 portb;                  #locate portb = 0x006

#bit b7 = portb.7           #bit b6 = portb.6       #bit b5 = portb.5

#bit b4 = portb.4           #bit b3 = portb.3       #bit b2 = portb.2

#BIT b1 = portb.1

#BIT b0 = portb.0

#priority EXT,RTCC

int16 get_t1;     int16 loop_check;     int16 triger_time;     int1 flg_t0;

int16 advance_t1;     int8 k;     int8 get_angle;     float adv_angle;     int1 flg_ext;

void advance(void);     void ini(void); void select_angle(void);
```

```
#int_CCP1

void  CCP1_isr(void)

{select_angle();}

#int_EXT

void  EXT_isr(void)

{ get_t1=get_timer1();    set_timer1(0);    flg_ext=1;}

void main()

{

setup_adc_ports(NO_ANALOGS);

setup_adc(ADC_OFF);

setup_spi(SPI_SS_DISABLED);

setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);

setup_timer_1(T1_EXTERNAL|T1_DIV_BY_1);

setup_timer_2(T2_DISABLED,0,1);

enable_interrupts(INT_EXT);

enable_interrupts(INT_CCP1);

setup_ccp1(CCP_CAPTURE_fE);

enable_interrupts(GLOBAL);
```

```
ext_int_edge( h_TO_1 );

get_t1=loop_check=0;

flg_ext=0;flg_t0=0;

set_timer1(0);

//--------initial i/o port-------

trisb1 = trisb4 =trisb5 = 0;

trisb0 = 1;t0se=0;t0cs =1;psa=1;

b1=b4=b5=0;k=0;

ini();

while(true)

{ triger_time=get_t1/3;

advance_t1= triger_time*adv_angle;

triger_time-=advance_t1;

if(flg_ext)

{ delay_us(triger_time);

b4=1;

b5=1;

delay_us(get_t1);
```

```
b4=0;b5=0;flg_ext=0; }    }  }

void ini(void)

{

while(k<=10)

{  b5=~b5;     delay_ms(100);   k++; }

b5=0; }

void select_angle(void)

{  get_angle=select_adv.data;

switch (get_angle) {

case 0:

adv_angle= 0;

break;

case 1:// advance 5 degree

adv_angle= 0.025;

break;

case 2:// advance 7 degree

adv_angle= 0.075;

break;
```

```
case 3:// advance 9 degree

adv_angle= 0.135;

break;

case 4:// advance 15 degree

adv_angle= 0.27;

break;

case 5:/// advance 20 degree

adv_angle= 0.33;

break;

case 6:// advance 25 degree

adv_angle= 0.422; break;

case 7:// advance 30 degree

adv_angle= 0.455; break;

default:

break; }}
```

Source Code Central Processing Unit

```
//master operating system of advance angle brushless dc motor

//detail of system
```

```
// 1 control advance angle  by pin_e2  pin_e3  pin_e4

// 2 control running mode (advance or normal) pin_e0 pin_e1

//   2.1 normal pin_e0 = low pin_e1 = high

//   2.2 advance pin_e0 = high pin_e1 = low

#include <30F2010.h>

#device adc=10

#FUSES NOWDT                 //No Watch Dog Timer

#FUSES HS                    //High speed Osc (> 4mhz for PCM/PCH) (>10mhz for PCD)

#FUSES PR                    //Primary Oscillator

#FUSES NOCKSFSM              //Clock Switching is disabled, fail Safe clock monitor is disabled

#FUSES WPSB16                //Watch Dog Timer PreScalar B 1:16

#FUSES WPSA512               //Watch Dog Timer PreScalar A 1:512

#FUSES PUT64                 //Power On Reset Timer value 64ms

#FUSES NOBROWNOUT            //No brownout reset

#FUSES BORV47                //Brownout reset at 4.7V

#FUSES LPOL_HIGH             //Low-Side Transistors Polarity is Active-High (PWM 0,2,4 and 6)

//PWM module low side output pins have active high output polar

#FUSES HPOL_HIGH             //High-Side Transistors Polarity is Active-High (PWM 1,3,5 and 7)
```

```
//PWM module high side output pins have active high output polarity

#FUSES NOPWMPIN          //PWM outputs drive active state upon Reset

#FUSES MCLR              //Master Clear pin enabled

#FUSES NOPROTECT         //Code not protected from reading

#FUSES NOWRT             //Program memory not write protected

#FUSES NODEBUG          //No Debug mode for ICD

#FUSES NOCOE             //Device will reset into operational mode

#FUSES ICSP1             //ICD uses PGC1/PGD1 pins

#FUSES RESERVED          //Used to set the reserved FUSE bits

#use delay(clock=20000000)      #use rs232(UART1,baud=19200,parity=N,bits=8)

#priority RDA,EXT0,EXT2,TIMER1

int16 trise;

#locate trise   = 0x02d8     #bit    tris_e2 = trise.2

#bit    tris_e3 = trise.3        #bit    tris_e4 = trise.4

int16 Porte; #locate Porte = 0x02da

#bit   e2    = Porte.2    #bit   e3    = Porte.3    #bit   e4    = Porte.4

int16 late;      #locate late = 0x2dc

#bit late_2  = late.2    #bit late_3  = late.3     #bit late_4  = late.4
```

```
int16 trisb;     #locate trisb = 0x02c6

#bit    tris_b4 = trisb.4    #bit    tris_b5 = trisb.5

int16 portb;    #locate portb = 0x02c8

#bit    b4    = portb.4     #bit    b5    = portb.5

int16 latb;     #locate latb = 0x02ca

#bit    latb_4 = latb.4     #bit    latb_5 = latb.5

void get_speed(void);

void recive_data(void);

byte n;       byte datacom[3];

int1 flg_rda,flg_com;   int32 k;

int1 flg_ext0,flg_ext2,flg_t1;    int16 sec;

int32 GET_T2;      int32 get_over;    int32 load1;

int16 pwm;      int8 cmd1,cmd2;    int16 adc0,adc1,adc2,adc3;

#int_EXT0

void  EXT0_isr(void)

{  output_toggle(pin_e5);

if(input(pin_d1))

{ set_timer1(0);flg_t1=0;setup_timer1(TMR_INTERNAL|TMR_DIV_BY_64);     }  }
```

```
#int_EXT2

void  EXT2_isr(void)

{ GET_T2=get_timer1();  flg_ext2=1;setup_timer1(TMR_DISABLED);}

#int_TIMER1

void  TIMER1_isr(void)

{  flg_t1=1; }

#int_RDA

void  RDA_isr(void)

{  datacom[n]=getc();      n++;

if(n>1)

{  flg_com=1;

//disable_interrupts(INT_RDA);

}

}

void main()

{

setup_spi(SPI_SS_DISABLED);   setup_wdt(WDT_OFF);

SETUP_ADC_PORTS(sAN0|sAN1|sAN2|sAN3|VSS_VDD );
```

```
SETUP_ADC(ADC_CLOCK_INTERNAL);

setup_compare(1,COMPARE_PWM | COMPARE_TIMER3 );

setup_timer3(TMR_INTERNAL | TMR_DIV_BY_1,1000 );

setup_timer1(TMR_INTERNAL|TMR_DIV_BY_8);

SETUP_TIMER2(TMR_DISABLED);

set_timer2(0);

ext_int_edge( 0, L_TO_H);  ext_int_edge( 2, H_TO_L); enable_interrupts(INT_EXT0);

enable_interrupts(INT_EXT2); enable_interrupts(INT_TIMER1); enable_interrupts(INT_RDA);

enable_interrupts(INTR_GLOBAL);

k=0;   flg_ext0=0; output_low(pin_e0); output_high(pin_e1);

output_HIGH(pin_d1);   set_timer1(1000); sec=0;flg_rda=0;

set_pwm_duty(1,0);   n=0;

tris_e2=tris_e3=tris_e4=0;   e2=e3=e4=0;   tris_b4=0;tris_b5=0;

b4=1; b5=1;     k=0;get_over=0;   flg_ext2=0;

e2=1;e3=1;e4=1;  late_2=1;late_3=1;late_4=1;

WHILE(true)

{

//---------get an0
```

```
set_adc_channel(0);

adc0 = read_adc();

//---------get an1

set_adc_channel(1);

adc1 = read_adc();

//---------get an2 current

set_adc_channel(2);

adc2 = read_adc();

//---------get an3 voltage

set_adc_channel(3);

adc3 = read_adc();

if(flg_ext2)

{  get_over=k*65535;

GET_T2 += get_over;

//printf("speed = %lu \r\n",GET_T2);

flg_ext2=0;get_over=0;k=0;  }

if(flg_t1)

{   k++;
```

```
flg_t1=0;

}

if(flg_com)

{  cmd1 =  datacom[0];cmd2=datacom[1] ;

switch (cmd1) {

case 1:// check data communication from comport

printf("%u %03u %04lu %04lu %06lu ",datacom[0],datacom[1],adc2,adc3,GET_T2);

printf("s\r\n");

flg_com=0;n=0;

break;

case 2:// command load torque

load1 = datacom[1];

load1*=1000;

load1/=255;

pwm=load1;

printf("%u %03u %04lu %04lu %06lu ",datacom[0],datacom[1],adc2,adc3,GET_T2);

printf("s\r\n");

set_pwm_duty(1,pwm);
```

```
flg_com=0;n=0;

break;

case 3:// command adjust advance angle

switch (cmd2){

case 0:

e2=1;e3=1;e4=1;

late_2=1;late_3=1;late_4=1;

b4=0;delay_ms(1);b4=1;

printf("%u %03u %04lu %04lu %06lu ",datacom[0],datacom[1],adc2,adc3,GET_T2);

printf("s\r\n");

flg_com=0;n=0;

break;

case 5:

e2=1;e3=1;e4=0;

late_2=1;late_3=1;late_4=0;

b4=0;delay_ms(1);b4=1;

printf("%u %03u %04lu %04lu %06lu ",datacom[0],datacom[1],adc2,adc3,GET_T2);

printf("s\r\n");
```

```
flg_com=0;n=0;

break;

case 7:

e2=1;e3=0;e4=1;

late_2=1;late_3=0;late_4=1;

b4=0;delay_ms(1);b4=1;

printf("%u %03u %04lu %04lu %06lu ",datacom[0],datacom[1],adc2,adc3,GET_T2);

printf("s\r\n");

flg_com=0;n=0;

break;

case 9:

e2=1;e3=0;e4=0;

late_2=1;late_3=0;late_4=0;

b4=0;delay_ms(1);b4=1;

printf("%u %03u %04lu %04lu %06lu ",datacom[0],datacom[1],adc2,adc3,GET_T2);

printf("s\r\n");

flg_com=0;n=0;

break;
```

```
case 15:

e2=0;e3=1;e4=1;

late_2=0;late_3=1;late_4=1;

b4=0;delay_ms(1);b4=1;

printf("%u %03u %04lu %04lu %06lu ",datacom[0],datacom[1],adc2,adc3,GET_T2);

printf("s\r\n");

flg_com=0;n=0;

break;

case 20:

e2=0;e3=1;e4=0;

late_2=0;late_3=1;late_4=0;

b4=0;delay_ms(1);b4=1;

printf("%u %03u %04lu %04lu %06lu ",datacom[0],datacom[1],adc2,adc3,GET_T2);

printf("s\r\n");

flg_com=0;n=0;

break;

case 25:

e2=0;e3=0;e4=1;
```

```
late_2=0;late_3=0;late_4=1;

b4=0;delay_ms(1);b4=1;

printf("%u %03u %04lu %04lu %06lu ",datacom[0],datacom[1],adc2,adc3,GET_T2);

printf("s\r\n");

flg_com=0;n=0;

break;

case 30:

e2=0;e3=0;e4=0;

late_2=0;late_3=0;late_4=0;

b4=0;delay_ms(1);b4=1;

printf("%u %03u %04lu %04lu %06lu ",datacom[0],datacom[1],adc2,adc3,GET_T2);

printf("s\r\n");

flg_com=0;n=0;

break;

default:

break;

}

break;
```

```
case 4:// command select drive mode (advance or normal)

switch (cmd2){

case 0 :// normal mode

// floating hall sensor

output_high(pin_e0);

output_high(pin_e1);

delay_ms(10);

output_low(pin_e0);

output_high(pin_e1);

printf("%u %03u %04lu %04lu %06lu ,datacom[0],datacom[1],adc2,adc3,GET_T2);

printf("s\r\n");

 flg_com=0;n=0;

break;

case 1 :// advance mode

output_high(pin_e0);

output_high(pin_e1);

delay_ms(10);

output_high(pin_e0);
```

```
output_low(pin_e1);

printf("%u %03u %04lu %04lu %06lu ,datacom[0],datacom[1],adc2,adc3,GET_T2);

printf("s\r\n");

flg_com=0;n=0;

break;

case 2 :// floating hall sensor

output_low(pin_e0);

output_low(pin_e1);

printf("%u %03u %04lu %04lu %06lu ,datacom[0],datacom[1],adc2,adc3,GET_T2);

printf("s\r\n");

flg_com=0;n=0;

break;

default:

break;

}

 break;

case 5: // command enable or disable drive

switch (cmd2){
```

```
case 0://disable drive

b5=0;

latb_5=0;

 printf("%u %03u %04lu %04lu %06lu ,datacom[0],datacom[1],adc2,adc3,GET_T2);

printf("s\r\n");

flg_com=0;n=0;

break;

case 1:// enable drive

b5=1;

latb_5=1;

printf("%u %03u %04lu %04lu %06lu ,datacom[0],datacom[1],adc2,adc3,GET_T2);

printf("s\r\n");

flg_com=0;n=0;

break;

default:

break;

}

break;
```

```
case 6:// get speed ,torque

printf("%u %03u %04lu %04lu %06lu ",datacom[0],datacom[1],adc2,adc3,GET_T2);

printf("s\r\n");

flg_com=0;n=0;

break;

default :

break;

}



}}}
```

ภาคผนวก ข

Source Code BLDC Drive

```
//Hall effect PIN_B3-->Blue:PIN_b4-->Green:PIN_b5-->Yellow

#include <30F2010.h>

#device adc=10

#Fuses  HS

//#Fuses  BORV20

#Fuses  BROWNOUT

#Fuses  noMCLR

#Fuses  PUT4

#Fuses  NOWDT

#fuses WPSB10

#use delay(clock=20000000,restart_wdt)

#use rs232(UART1,baud=19200,parity=N,bits=8,)

//#include <math.h>

//#FUSES HPOL_LOW //High.Side Transistors Polarity is Active.High (PWM 1,3,5 and 7)

//#FUSES LPOL_HIGH //Low.Side Transistors Polarity is Active.Low (PWM 0,2,4 and 6)

#priority TIMER1,EXT1

int16 trisB;
```

```
#locate trisB=0x02C6    #bit    trisb3=trisb.3

#bit    trisb4=trisb.4      #bit    trisb5=trisb.5

struct sensor{

INT NON  :3;

int  data :3;

}hall_data;

#locate hall_data=0x02C8

int16 PTCON;

#locate PTCON = 0x1C0     #bit PTEN     = PTCON.15

#bit PTSIDL   = PTCON.13  #bit PTOPS3 = PTCON.7

#bit PTOPS2 = PTCON.6      #bit PTOPS1 = PTCON.5

#bit PTOPS0 = PTCON.4      #bit PTCKPS1 = PTCON.3

#bit PTCKPS0 = PTCON.2     #BIT PTMOD1  =PTCON.1

#BIT PTMOD0  =PTCON.0

INT16 PTMR;

#LOCATE  PTMR = 0X1C2

INT16 PTPER;
```

```
#LOCATE PTPER = 0X1C4

INT16 SEVTCMP;

#LOCATE SEVTCMP = 0X1C6

INT16 PWMCON1;

#locate PWMCON1 = 0x1c8

#bit PMOD3 = PWMCON1.10

#bit PMOD2 = PWMCON1.9

#bit PMOD1 = PWMCON1.8

#bit PEN3H = PWMCON1.6

#bit PEN2H = PWMCON1.5

#bit PEN1H = PWMCON1.4

#bit PEN3L = PWMCON1.2

#bit PEN2L = PWMCON1.1

#bit PEN1L = PWMCON1.0

INT16 PWMCON2;

#locate PWMCON2 = 0x1ca

#bit SEVOPS3 = PWMCON2.11
```

```
#bit SEVOPS2 = PWMCON2.10

#bit SEVOPS1 = PWMCON2.9

#bit SEVOPS0 = PWMCON2.8

#bit OSYNC   = PWMCON2.1

#bit UDIS    = PWMCON2.0

INT16 DTCON1;

#LOCATE DTCON1 = 0X1CC

#BIT DTAPS1 = DTCON1.7

#BIT DTAPS0 = DTCON1.6

#BIT DTIME5 = DTCON1.5

#BIT DTIME4 = DTCON1.4

#BIT DTIME3 = DTCON1.3

#BIT DTIME2 = DTCON1.2

#BIT DTIME1 = DTCON1.1

#BIT DTIME0 = DTCON1.0

int16 ifs2;

#locate ifs2=0x088
```

```
#bit   fltaif = ifs2.11

INT16 FLTACON;

#LOCATE FLTACON = 0X1D0

#BIT    FAOV3H  = FLTACON.13

#BIT    FAOV3L  = FLTACON.12

#BIT    FAOV2H  = FLTACON.11

#BIT    FAOV2L  = FLTACON.10

#BIT    FAOV1H  = FLTACON.9

#BIT    FAOV1L  = FLTACON.8

#BIT    FLTAM   = FLTACON.7

#BIT    FAEN3   = FLTACON.2

#BIT    FAEN2   = FLTACON.1

#BIT    FAEN1   = FLTACON.0

INT16 OVDCON;

#LOCATE OVDCON  =  0X1D4

#BIT    POVD3H  =  OVDCON.13

#BIT    POVD3L  =  OVDCON.12
```

```
#BIT    POVD2H  =  OVDCON.11

#BIT    POVD2L  =  OVDCON.10

#BIT    POVD1H  =  OVDCON.9

#BIT    POVD1L  =  OVDCON.8

#BIT    POUT4H  =  OVDCON.7

#BIT    POUT3H  = OVDCON.5

#BIT    POUT3L  = OVDCON.4

#BIT    POUT2H  = OVDCON.3

#BIT    POUT2L  = OVDCON.2

#BIT    POUT1H  = OVDCON.1

#BIT    POUT1L  = OVDCON.0

  INT16 PDC1;

  #LOCATE PDC1 = 0X1D6

  INT16 PDC2;

  #LOCATE PDC2 = 0X1D8

  INT16 PDC3;

  #LOCATE PDC3 = 0X1DA
```

```
INT16 CONST TABLE_FW[]={0x0000,0x2001,0x0810,0x0801,0x0204,0x2004,0x0210};

INT16 CONST TABLE_RW[]={0x0000,0x0210,0x2004,0x0204,0x0801,0x0810,0x2001};

INT8 INDEX,TEMPINDEX;

int1 flg_int,flg_t4,FLG_RDA;

int16 n;

INT32 GET_T23;

int16 duty;

INT8 GET_DUTY;

int16 loop;

#int_RDA

void  RDA_isr(void)

{

GET_DUTY=GETC();

FLG_RDA=1;

}

#int_EXT1

void  EXT1_isr(void)
```

```
{
flg_int=1;
GET_T23=get_timer23();
set_timer23(0);
}
#int_TIMER1
void  TIMER1_isr(void)
{set_timer1(500);
set_adc_channel( 0 ); duty = read_adc();pdc1= pdc2= pdc3=duty;
flg_t4=1;
clear_interrupt(int_timer1);
}
void main(void)
{ setup_wdt (WDT_OFF);
flg_t4=0;
//trisb=0x000f;
trisb3=trisb4=trisb5=1;
```

```
ptper=0x0080;

SEVTCMP=ptper;

ptmr=0x0000;

//===============PMOD3(PWM3 MODE) PMDO2(PWM2MODE) AND
PMOD1(PWM1MODE) FOR SELECT COMPLEMENTARY OR Independent mode

PMOD3=1;//PWM3

PMOD2=1;//PWM2

PMOD1=1;// PWM1

pten=1;      // ENABLE INPUT CLOCK FOR PWM

PTSIDL=0;

//===========PTMOD0 AND PTMOD1 FOR SELECT MODE PWM

ptmod0=0;    // 00 Free Running mode

ptmod1=0;    // 01  Single-shot mode

             // 10  Continuous Up/Down Counting mode.

             // 11   Continuous Up/Down mode with interrupts for double PWM

//============PTCKPS0 AND PTCKPS1 BIT FOR PTMR PRESCALE

ptckps0=1;   // 00 (1:1 prescale)
```

```
ptckps1=0;   // 01 (1:4 prescale)

                // 10 (1:16 prescale)

               // 11 (1:64 prescale)

//============PENxh PENxl for enable pwmoutput===============

pen3h=1;    //     pen2h=1;    // pen1h=1;    //  pen3l=1;    //

pen2l=1;    //  pen1l=1;    //

//=========================================

pdc1=0x120; // pdc2=0x120; //  pdc3=0x120; //

//========================

OSYNC=1;UDIS=0;

POVD1H =1; POVD1L =1; POVD2H =1; POVD2L =1;

POVD3H =1;   povd3l =0;   Pout3L =1;

//========================

PWMCON2=0X0F00; INDEX=0;

SETUP_ADC_PORTS(sAN0|VSS_VDD );

SETUP_ADC(ADC_CLOCK_INTERNAL);

enable_interrupts(INTR_GLOBAL);
```

```
ENable_interrupts(INT_TIMER1);

disable_interrupts(INT_RDA);

set_timer1(500);

//=====================SET TIMER23 32 BIT=======

SETUP_TIMER2(TMR_EXTERNAL |TMR_DIV_BY_1|TMR_32_BIT);

set_timer23(0);

setup_timer1(TMR_INTERNAL|TMR_DIV_BY_8);

disable_interrupts(INT_EXT1);

ext_int_edge( 1, H_TO_L);

pdc1= pdc2= pdc3= 20;

flg_int=0;FLG_RDA=0;

n=0;duty=200;

OVDCON=TABLE_FW[0];

FLTACON=0XFF87;

WHILE(N<5)

{

OUTPUT_TOGGLE(PIN_d1);
```

```
DELAY_MS(1000);
N++;
 }
 OUTPUT_HIGH(PIN_d1);loop=0;
 while(true)
 {
 INDEX=hall_data.data;
 while(INDEX==hall_data.data)
{
OVDCON=TABLE_FW[INDEX];
 n++;
 if(n>30000)
 {
 output_toggle(PIN_d1);
 n=0;
 } }}}
```

ภาคผนวก ค

Source Code Visual Basic GUI And Data Anlysis

```
Private Sub Form_Load()
On Error Resume Next
 MSComm1.PortOpen = True
'---------- ini timer-----
 Timer1.Enabled = False
 get_speed_torque.Enabled = False
 cmd_advance.Enabled = False
 cmd_load.Enabled = False
 cmd_drive_adv.Enabled = False
 cmd_enable_drive.Enabled = False
 cmd_disable_drive.Enabled = False
 cmd_floatint_hall.Enabled = False
 cmd_drive_normal.Enabled = False
 step_pwm.Enabled = False
 st_rec.Enabled = False
 t_normal.Enabled = True
 t_advance.Enabled = False
```

```
start_rec.Enabled = False

auto_step_load.Enabled = False

MANUAL_LOAD.Enabled = False

AUTO_LOAD.Enabled = False

c_data = 0

CLEAR_DATA.Enabled = False

CLR.Enabled = False

REPORT_EXCEL.Enabled = False

report.Enabled = False

'--------------initial variable--------

ref_datax = 0

ref_datay = 0

num_data = 0

step_load = 0

max_load = 0.3

min_load = 0

num_plot = 0
```

```
    max_speed = 500
    min_speed = 200
    scalex1 = 0
    scaley2 = 0
    dx = 0
    dy = 0
    record = False
    toggle = 0
    avr_value = 0
    avr_value2 = 0
    avr_value3 = 0
    value_x = 0
    value_y = 0
    load_data = 0
End Sub
Private Sub Label25_Click()
End Sub
```

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, x As Single, Y As Single)
BLDC2013.Caption = "ADVANCE ANGLE BLDC TECHNOLOGY"
End Sub
Private Sub Form_Unload(Cancel As Integer)
End
End Sub


Private Sub MANUAL_LOAD_Timer()
 '------------code command -------
  If c_data = 0 Then
        MSComm1.Output = Chr$(2)
      c_data = c_data + 1
   End If
'------------data command---------
 If c_data = 1 Then
   MSComm1.Output = Chr$(PWM_.Value)
    c_data = c_data + 1
```

```
  End If
'------------------ stop command---------
 If c_data = 2 Then
   MSComm1.Output = Chr$(0)
   c_data = 0
   MANUAL_LOAD.Enabled = False
   'Exit Sub
 End If
End Sub


Private Sub max_t_Click()
input_var.Show
End Sub
Private Sub MSComm1_OnComm()
Dim finish$
Dim offset$
Dim datain$, A$, b$, c$, d$, G$, h$
```

```
Dim loop_check

Dim rpm$

Dim count$

Dim real_speed

   Do

   buffer$ = buffer$ & MSComm1.Input

   loop_check = loop_check + 1

   If loop_check > 100000 Then

   Exit Sub

   End If

   Loop Until InStr(buffer$, "s")

    datain$ = buffer$

    Text1 = datain$

    A$ = Mid$(datain$, 1, 1)

    If Val(A$) < 1 And Val(A$) > 6 Then GoTo lb7

    If Val(A$) = 1 Then GoTo lb1

    If Val(A$) = 2 Then GoTo lb2
```

```
If Val(A$) = 3 Then GoTo lb3

If Val(A$) = 4 Then GoTo lb4

If Val(A$) = 5 Then GoTo lb5

If Val(A$) = 6 Then GoTo lb6

lb1:

b$ = Mid$(datain$, 17, 6)

'SHOW_SPEED.Text = b$

If Val(b$) > 1000 Then

real_speed = Val(b$) * 64 * 0.0000002 'time/0.5rev (64 is div_clk 64 and 0.0000002 is 1 T of clock)use t1

real_speed = real_speed * 2 ' time/rev

If real_speed > 0 Then real_speed = 60 / real_speed   ' rpm

End If

If avr_value3 <= 9 Then

sum_rpm = sum_rpm + real_speed

avr_value3 = avr_value3 + 1

Else
```

```
sum_rpm = sum_rpm / 10

real_speed = sum_rpm

rpm = real_speed

SHOW_SPEED.Text = Format$(real_speed, "###")

avr_value3 = 0

sum_rpm = 0

 power = real_current * real_voltage

If rpm > 0 Then torque = power / rpm

show_loadtorque.Text = Format$(torque, "0.##")

End If

'------------------------display current

c$ = Mid$(datain$, 7, 4)

current = Val(c$) * 5

current = current / 1024

current = current * 3

If avr_value <= 9 Then
```

```
sum_current = sum_current + current

avr_value = avr_value + 1

Else

sum_current = sum_current / 10

current = sum_current

current = current / 0.6

real_current = current

show_loadcurrent.Text = Format$(current, "##.00")

avr_value = 0

sum_current = 0

End If

'------------------------- display voltage

 d$ = Mid$(datain$, 12, 4)

 voltage = Val(d$) * 5

 voltage = voltage / 1024

 voltage = voltage * 10

If avr_value2 <= 9 Then
```

```
      sum_voltage = sum_voltage + voltage
      avr_value2 = avr_value2 + 1
    Else
      sum_voltage = sum_voltage / 10
      voltage = sum_voltage
      real_voltage = voltage
      show_loadvoltage.Text = Format$(voltage, "##.00")
      avr_value2 = 0
      sum_voltage = 0
    End If
    '------------------------display load torque
lb2:


lb3:


lb4:
```

```
lb5:


lb6:


lb7:

End Sub

Private Sub PWM_LOAD_Change()

PWM_LOAD.Enabled = True

End Sub

Private Sub Option1_Click()

adv_angle = 0

cmd_advance.Enabled = True

End Sub

Private Sub Option2_Click()

adv_angle = 5

cmd_advance.Enabled = True

End Sub
```

```
Private Sub Option3_Click()

adv_angle = 7

cmd_advance.Enabled = True

End Sub

Private Sub Option4_Click()

adv_angle = 9

cmd_advance.Enabled = True

End Sub

Private Sub Option5_Click()

adv_angle = 15

cmd_advance.Enabled = True

End Sub

Private Sub Option6_Click()

adv_angle = 20

cmd_advance.Enabled = True

End Sub

Private Sub Option7_Click()
```

```
adv_angle = 25

cmd_advance.Enabled = True

End Sub

Private Sub Option8_Click()

adv_angle = 30

cmd_advance.Enabled = True

End Sub

Private Sub Picture1_MouseMove(Button As Integer, Shift As Integer, x As Single, Y As
Single)

BLDC2013.Caption = "ADVANCE ANGLE BLDC TECHNOLOGY" & " " & "speed =" & x
& " " & "RPM" & " " & "load =" & Y & " " & "N.m"

End Sub

Private Sub Picture1_Paint()

divy = (max_load - min_load) / 10

divx = (max_speed - min_speed) / 10

'----------------------------------DISPLAY SCALE LOAD---------------

lbt1.Caption = min_load + (divy * 1)
```

```
lbt2.Caption = min_load + (divy * 2)

lbt3.Caption = min_load + (divy * 3)

lbt4.Caption = min_load + (divy * 4)

lbt5.Caption = min_load + (divy * 5)

lbt6.Caption = min_load + (divy * 6)

lbt7.Caption = min_load + (divy * 7)

lbt8.Caption = min_load + (divy * 8)

lbt9.Caption = min_load + (divy * 9)

lbt10.Caption = min_load + (divy * 10)

'---------------------------------DISPLAY SCALE E_SPEED------------------

lbsp1.Caption = min_speed + (divx * 1)

lbsp2.Caption = min_speed + (divx * 2)

lbsp3.Caption = min_speed + (divx * 3)

lbsp4.Caption = min_speed + (divx * 4)

lbsp5.Caption = min_speed + (divx * 5)

lbsp6.Caption = min_speed + (divx * 6)

lbsp7.Caption = min_speed + (divx * 7)
```

```
lbsp8.Caption = min_speed + (divx * 8)

lbsp9.Caption = min_speed + (divx * 9)

lbsp10.Caption = min_speed + (divx * 10)

'--------------------------------SET SCALE--------------------------------------

Picture1.Cls

Picture1.DrawWidth = 2

Picture1.DrawStyle = 0

scalex1 = min_speed

scaley1 = max_load

scalex2 = max_speed

scaley2 = min_load

Picture1.Scale (scalex1, scaley1)-(scalex2, scaley2)

'---------------------------------draw gride scale x-------------------------

Picture1.DrawWidth = 1

Picture1.DrawStyle = 2

For x = min_speed To max_speed Step divx

Picture1.Line (x, 0)-(x, max_load)
```

```
Next x

'---------------------------------draw gride scale y--------------------------

For x = min_load To max_load Step divy

Picture1.Line (0, x)-(max_speed, x)

Next x

'Me.Circle (1500, 1500), 150, vbBlack

End Sub

Private Sub PWM__Change()

MANUAL_LOAD.Enabled = True

End Sub

Private Sub report_Click()

Dim MYOBJECT As Object

Dim k

Dim j

Dim t$

Dim S$

Dim A, b As Integer, c
```

```
Dim colum As String

Dim index

index = 65

colum = Chr$(index)

Set MYOBJECT = CreateObject("Excel.Application")

MYOBJECT.workbooks.Add

'MsgBox (W2$)

For j = 0 To ref_datax - 1

'----------------------------

If j = 0 Then

For k = 0 To save_numdata(j)

colum = Chr$(index)

t$ = colum & k + 1

colum = Chr$(index + 1)

S$ = colum & k + 1

MYOBJECT.range(S$).Value = save_datax(j, k)

MYOBJECT.range(t$).Value = save_datay(j, k)
```

```
Next k

End If

'------------------------------

If j = 1 Then

For k = 0 To save_numdata(j)

colum = Chr$(index + 2)

t$ = colum & k + 1

colum = Chr$(index + 3)

S$ = colum & k + 1

MYOBJECT.range(S$).Value = save_datax(j, k)

MYOBJECT.range(t$).Value = save_datay(j, k)

Next k

End If

'-------------------------------------

If j = 2 Then

For k = 0 To save_numdata(j)

colum = Chr$(index + 4)
```

```
t$ = colum & k + 1

colum = Chr$(index + 5)

S$ = colum & k + 1

MYOBJECT.range(S$).Value = save_datax(j, k)

MYOBJECT.range(t$).Value = save_dataY(j, k)

Next k

End If

'----------------------------------------

If j = 3 Then

For k = 0 To save_numdata(j)

colum = Chr$(index + 6)

t$ = colum & k + 1

colum = Chr$(index + 7)

S$ = colum & k + 1

MYOBJECT.range(S$).Value = save_datax(j, k)

MYOBJECT.range(t$).Value = save_dataY(j, k)

Next k
```

```
End If

Next j



MYOBJECT.Visible = True

'------------------------ENABLE COMMAND----------

 CLEAR_DATA.Enabled = True

 CLR.Enabled = True

 CLEAR_DATA.Enabled = True

 CLR.Enabled = True

 REPORT_EXCEL.Enabled = False

 report.Enabled = False

End Sub

Private Sub REPORT_EXCEL_Click()

Dim MYOBJECT As Object

Dim k

Dim j

Dim t$
```

```
Dim S$

Dim A, b As Integer, c

Dim colum As String

Dim index

index = 65

colum = Chr$(index)

Set MYOBJECT = CreateObject("Excel.Application")

MYOBJECT.workbooks.Add

For j = 0 To ref_datax - 1

'----------------------------

If j = 0 Then

For k = 0 To save_numdata(j)

colum = Chr$(index)

t$ = colum & k + 1

colum = Chr$(index + 1)

S$ = colum & k + 1

MYOBJECT.range(S$).Value = save_datax(j, k)
```

```
MYOBJECT.range(t$).Value = save_datay(j, k)

Next k

End If

If j = 1 Then

For k = 0 To save_numdata(j)

colum = Chr$(index + 2)

t$ = colum & k + 1

colum = Chr$(index + 3)

S$ = colum & k + 1

MYOBJECT.range(S$).Value = save_datax(j, k)

MYOBJECT.range(t$).Value = save_datay(j, k)

Next k

End If

'-------------------------------------

If j = 2 Then

For k = 0 To save_numdata(j)

colum = Chr$(index + 4)
```

```
t$ = colum & k + 1

colum = Chr$(index + 5)

S$ = colum & k + 1

MYOBJECT.range(S$).Value = save_datax(j, k)

MYOBJECT.range(t$).Value = save_dataY(j, k)

Next k

End If

'----------------------------------------

If j = 3 Then

For k = 0 To save_numdata(j)

colum = Chr$(index + 6)

t$ = colum & k + 1

colum = Chr$(index + 7)

S$ = colum & k + 1

MYOBJECT.range(S$).Value = save_datax(j, k)

MYOBJECT.range(t$).Value = save_dataY(j, k)

Next k
```

```
End If

Next j

MYOBJECT.Visible = True

End Sub

Private Sub run_advance_Click()

cmd_drive_adv.Enabled = True

t_advance.Enabled = True

t_normal.Enabled = False

End Sub



Private Sub run_normal_Click()

cmd_drive_normal.Enabled = True

t_normal.Enabled = True

t_advance.Enabled = False

End Sub

Private Sub scal_rpm_Change()

 If record = True Then
```

```
 x1 = Val(scal_rpm.Text)

 y1 = Val(scal_t.Text)

If (x1 < value_x) Then

 save_datax(ref_datax, num_data) = x1

 save_datay(ref_datay, num_data) = y1

 num_data = num_data + 1

Picture1.Line (value_x, value_y)-(x1, y1), QBColor(num_plot)

End If

  value_x = x1

  value_y = y1

 End If

End Sub



Private Sub scal_t_Change()

End Sub

Private Sub SCALE_Click()

scal_rpm.Text = ""
```

```
input_var.Show

st_rec.Enabled = False

End Sub

Private Sub show_loadtorque_Change()

scal_t.Text = show_loadtorque.Text

End Sub

Private Sub SHOW_SPEED_Change()

scal_rpm.Text = SHOW_SPEED.Text

End Sub

Private Sub st_rec_Timer()

If toggle = 0 Then

Timer1.Enabled = True

toggle = toggle + 1

Else

Timer1.Enabled = False

toggle = 0

End If
```

```
End Sub

Private Sub start_rec_Click()

show_loadtorque.Text = "0.00"

SHOW_SPEED.Text = "000"

show_loadcurrent.Text = "0.00"

show_loadvoltage.Text = "0.00"

st_rec.Enabled = True

avr_value = 0

avr_value2 = 0

avr_value3 = 0

sum_current = 0

sum_voltage = 0

sum_rpm = 0

value_x = 0

value_y = 0

x1 = Val(scal_rpm.Text)

y1 = Val(scal_t.Text)
```

```
save_datax(ref_datax, num_data) = x1

save_datay(ref_datay, num_data) = y1

record = True

Picture1.DrawWidth = 2

Picture1.DrawStyle = 0

'----------------------DISABLE COMMAND-----

 CLEAR_DATA.Enabled = False

 CLR.Enabled = False

 REPORT_EXCEL.Enabled = False

End Sub

Private Sub step_pwm_Timer()

step_load = step_load + 15

PWM_.Value = step_load

If step_load > 240 Then

step_pwm.Enabled = False

record = False

save_numdata(ref_datax) = num_data
```

```
ref_datax = ref_datax + 1

ref_datay = ref_datax

num_data = 0

num_plot = num_plot + 1

PWM_.Value = 0

'---------------ENABLE COMMAND------

 CLEAR_DATA.Enabled = False

 CLR.Enabled = False

 REPORT_EXCEL.Enabled = True

 report.Enabled = True

End If

End Sub



Private Sub t_advance_Timer()

If display_mode = 0 Then

displaymode.BackColor = QBColor(12)

display_mode = display_mode + 1
```

```
displaymode.Caption = "ADVANCE MODE"

Else

displaymode.BackColor = QBColor(7)

display_mode = 0

displaymode.Caption = ""

End If

End Sub

Private Sub t_normal_Timer()

If display_mode = 0 Then

displaymode.BackColor = QBColor(10)

display_mode = display_mode + 1

displaymode.Caption = "NORMAL MODE"

Else

displaymode.BackColor = QBColor(7)

display_mode = 0

displaymode.Caption = ""

End If
```

```
End Sub

Private Sub Text2_Change()

code = Val(Text2.Text)

End Sub

Private Sub Text3_Change()

PWM = Val(Text3.Text)

End Sub

Private Sub Timer1_Timer()

'------------code command -------

  If c_data = 0 Then

     MSComm1.Output = Chr$(1)

      c_data = c_data + 1

   End If

'------------data command---------

 If c_data = 1 Then

   MSComm1.Output = Chr$(0)

    c_data = c_data + 1
```

```
  End If
'------------------ stop command---------
  If c_data = 2 Then
    MSComm1.Output = Chr$(0)
    c_data = 0
    Timer1.Enabled = False
    'Exit Sub
  End If
End Sub
```

PUBLIC VARIABLE

```
Public max_load
Public min_load
Public max_speed
Public min_speed
Public scalex1
Public scalex2
Public scaley1
```

Public scaley2

Public divx

Public divy

Public dx

Public dy

Public toggle

Public current

Public voltage

Public avr_value

Public avr_value2

Public avr_value3

Public sum_rpm

Public sum_current

Public sum_voltage

Public real_current

Public real_voltage

Public torque

```
Public power

Public rpm

Public step_load

Public record As Boolean

Public x1

Public y1

Public value_x

Public value_y

Public num_plot

Public save_datax(5, 500)

Public save_datay(5, 500)

Public ref_datax

Public ref_datay

Public num_data

Public load_data

Public save_numdata(5)
```

ภาคผนวก ง

Source Code Mathlab (S –Function)

[1] S- Function Inverter

```
function [sys,x0,str,ts] = timestwo(t,x,u,flag,vmax)

% Dispatch the flag. The switch function controls the calls to

% S-function routines at each simulation stage.

%INPUT VECTOR

%u(1) Ha

%u(2) Hb

%u(3) Hc

%u(4) Vbatt

% output vector

%sys(1) Va

%sys(2) Vb

%sys(3) Vc

switch flag

case 0

[sys,x0,str,ts] = mdlInitializeSizes(t,x,u,flag,vmax); % Initialization
```

```
case 1,

sys = mdlderivatives(t,x,u,flag,vmax);

case 3

sys = mdlOutputs(t,x,u,vmax); % Calculate outputs

case { 1, 2, 4, 9 }

sys = []; % Unused flags

otherwise

error(['Unhandled flag = ',num2str(flag)]); % Error handling

end;

% End of function timestwo.

%=============================================================

% Function mdlInitializeSizes initializes the states, sample

% times, state ordering strings (str), and sizes structure.

%=============================================================

function [sys,x0,str,ts] = mdlInitializeSizes(t,x,u,flag,vmax)

% Call function simsizes to create the sizes structure.

sizes = simsizes;
```

```
% Load the sizes structure with the initialization information.

sizes.NumContStates= 0;

sizes.NumDiscStates= 0;

sizes.NumOutputs= 3;

sizes.NumInputs= 4;

sizes.DirFeedthrough=1;

sizes.NumSampleTimes=1;

% Load the sys vector with the sizes information.

sys = simsizes(sizes);

x0 = []; % No continuous states

str = []; % No state ordering

ts = [-1 0]; % Inherited sample time

% End of mdlInitializeSizes.

%=============================================================

% Function mdlOutputs performs the calculations.

%=============================================================

function sys = mdlOutputs(t,x,u,vmax)
```

```
%a=2;b=3;c=4;

ha=u(1);% = a*u(1);

hb=u(2);%=b*u(2);

hc=u(3);%=c*u(3);

vb=u(4);

if(vb > vmax)

vb=vmax;

end

%sys = rccvmaxt(t,x,u);

% End of mdlOutputs.

if(ha>0 & hb==0 & hc==0)%100

if(vb > vmax)

vb=vmax;

end

sys(1)=vb;

sys(2)=-vb;

sys(3)=0;
```

```
end

if(ha>0 & hb>0 & hc==0)%110

if(vb > vmax)

vb=vmax;

end

sys(1)=vb;

sys(2)=0;

sys(3)=-vb;

end

if(ha==0 & hb>0 & hc==0)%010

if(vb > vmax)

vb=vmax;

end

sys(1)=0;

sys(2)=vb;

sys(3)=-vb;

end
```

```
if(ha==0 & hb>0 & hc>0)%011

if(vb > vmax)

vb=vmax;

end

sys(1)=-vb;

sys(2)=vb;

sys(3)=0;

end

if(ha==0 & hb==0 & hc>0)%001

if(vb > vmax)

vb=vmax;

end

sys(1)=-vb;

sys(2)=0;

sys(3)=vb;

end

if (ha>0 & hb==0 & hc>0)%101
```

```
if(vb > vmax)

vb=vmax;

end

sys(1)=0;

sys(2)=-vb;

sys(3)=vb;

end
```

[2] S- Function Back EMF And Hall sensor

```
function [sys,x0,str,ts] = bldc(t,x,u,flag,R,L,J,P,kw)

% BLDC An example M-file S-function for defining a system of

% The expected input vector is:

% (1): Wr

%(2): theta

%OUTPUT VECTOR GENERATED BY THE SYSTEM:

%(1): Emf_u

%(2): Emf_v
```

```
%(3):  Emf_w

%(4):  Ha

%(5):  Hb

%(6):  Hc

%(7):  theta

% Dispatch the flag.

switch flag,

case 0,

[sys,x0,str,ts]=mdlInitializeSizes(R,L,J,P,kw); % Initialization

case 1,

sys = mdlDerivatives(t,x,u,R,L,J,P,kw); % Calculate derivatives

case 2,   %model update

sys = mdlUpdate(t,x,u,R,L,J,P,kw); % update the model

case 3,

sys = mdlOutputs(t,x,u,R,L,J,P,kw); % Calculate outputs

case 4, % Unused flags

sys = [];
```

```
case 'reset'

% name = 'mybldc_mdl2/mybldc/my state-space';

%LocalResetController(name);

case 9,

% sys=mdlTerminate(t,x,u);

%%%%%%%%%%%%%%%%%%%%

% Unexpected flags %

%%%%%%%%%%%%%%%%%%%%

otherwise

error(['Unhandled flag = ',num2str(flag)]);

end

% End of csfunc.

%=============================================================

% mdlInitializeSizes

% Return the sizes, initial conditions, and sample times for the

% S-function.

%=============================================================
```

```
function [sys,x0,str,ts] = mdlInitializeSizes(R,L,J,P,kw)

% Call simsizes for a sizes structure, fill it in and convert it

% to a sizes array.

sizes = simsizes;

sizes.NumContStates  = 0;

sizes.NumDiscStates  = 0;

sizes.NumOutputs     = 7;

sizes.NumInputs      = 2;

sizes.DirFeedthrough = 1;     % Matrix D is nonempty.

sizes.NumSampleTimes = 1;

sys = simsizes(sizes);

% Initialize the initial conditions.

x0 = [];

% str is an empty matrix.

%

str = [];

%
```

```
% Initialize the array of sample times; in this example the sample

% time is continuous, so set ts to 0 and its offset to 0.

%

ts = [0 0];

% End of mdlInitializeSizes.

%==============================================================

% mdlDerivatives

% Return the derivatives for the continuous states.

% Nothing needs to be done here.

%==============================================================

function sys = mdlDerivatives(t,x,u,R,L,J,P,kw)

sys=[];

%==============================================================

% mdlUpdate

% Update the model parameters.

% Nothing needs to be done here

%==============================================================
```

```
function sys = mdlUpdate(t,x,u,R,L,J,P,kw); % update the model

sys=[];

%==============================================================

% mdlOutput

% calculate the model outputs and update other blocks ( Adaptive coefficients)

%==============================================================

function sys = mdlOutputs(t,x,u,R,L,J,P,kw)

theta = rem (u(2),2*pi);                % make theta betweeen 2*pi and -2*pi

sys(7)=theta;

if  (theta>=0 & theta<pi/6)        % 0 TO  30 DEGREE

LA= 6*theta/pi;

LB=-1;

LC=1;

sys(1)= LA*kw*u(1);     % Back emf for phase U

sys(2)=LB*kw*u(1);      %Back emf for phase V

sys(3)=LC*kw*u(1);      %Back emf for phase W

%-------------------------------------------------12
```

```
sys(4)= 5;                  %HALL SENSOR A

sys(5)=0;                   %HALL SENSOR B

sys(6)=0;                   %HALL SENSOR C

end;

if (theta>=pi/6 & theta<pi/3)       % 30 TO  60 DEGREE

LA=1;

LB=-1;

LC=-6*(theta-(2*pi/6))/pi;

sys(1)= LA*kw*u(1);     % Back emf for phase U

sys(2)=LB*kw*u(1);      %Back emf for phase V

sys(3)=LC*kw*u(1); %Back emf for phase W

sys(4)= 5;                  %HALL SENSOR A

sys(5)=0;                   %HALL SENSOR B

sys(6)=0;                   %HALL SENSOR C

end;

if (theta>=pi/3 & theta<pi/2)       % 60 TO  90 DEGREE

LA=1;
```

```
LB=-1;

LC=-6*(theta-(2*pi/6))/pi;

sys(1)= LA*kw*u(1);     % Back emf for phase U

sys(2)=LB*kw*u(1);      %Back emf for phase V

sys(3)=LC*kw*u(1); %Back emf for phase W

sys(4)= 5;                    %HALL SENSOR A

sys(5)=5;                    %HALL SENSOR B

sys(6)=0;                    %HALL SENSOR C

end;

if (theta>=pi/2 & theta<2*pi/3)      % 90 TO  120 DEGREE

LA=1;

LB=(  theta-(4*pi/6) )*6/pi;

LC=-1;

sys(1)= LA*kw*u(1);     % Back emf for phase U

sys(2)=LB*kw*u(1);      %Back emf for phase V

sys(3)=LC*kw*u(1); %Back emf for phase W

sys(4)= 5;                    %HALL SENSOR A
```

```
sys(5)=5;                    %HALL SENSOR B

sys(6)=0;                    %HALL SENSOR C

end;

if(theta>=2*pi/3 & theta<5*pi/6)    % 120 TO  150 DEGREE

LA=1;

LB=(  theta-(2*pi/3) )*6/pi;

LC=-1;

sys(1)= LA*kw*u(1);     % Back emf for phase U

sys(2)=LB*kw*u(1);      %Back emf for phase V

sys(3)=LC*kw*u(1);      %Back emf for phase W

sys(4)= 0;                   %HALL SENSOR A

sys(5)=5;                    %HALL SENSOR B

sys(6)=0;                    %HALL SENSOR C

end;

if (theta>=5*pi/6 & theta<pi)     % 150 TO  180 DEGREE

LA=(pi-theta)*6/pi;

LB=1;
```

```
LC=-1;
sys(1)= LA*kw*u(1);     % Back emf for phase U
sys(2)=LB*kw*u(1);      %Back emf for phase V
sys(3)=LC*kw*u(1); %Back emf for phase W
sys(4)=0;                   %HALL SENSOR A
sys(5)=5;                   %HALL SENSOR B
sys(6)=0;                   %HALL SENSOR C
%------------------------------------------- 7
end;
if(theta>=pi & theta<7*pi/6)    % 180 TO  210 DEGREE
 LA=(pi-theta)*6/pi;
LB=1;
LC=-1;
sys(1)= LA*kw*u(1);      % Back emf for phase U
sys(2)=LB*kw*u(1);       %Back emf for phase V
sys(3)=LC*kw*u(1);       %Back emf for phase W
sys(4)= 0;                  %HALL SENSOR A
```

```
sys(5)=5;                    %HALL SENSOR B

sys(6)=5;                    %HALL SENSOR C

%-----------------------------------------------6

end;

if(theta>=7*pi/6 & theta<4*pi/3)  % 210 TO  240 DEGREE

LA= -1;

LB=1;

LC=( theta-(4*pi/3))*6/pi;

sys(1)= LA*kw*u(1);     % Back emf for phase U

sys(2)=LB*kw*u(1);      %Back emf for phase V

sys(3)=LC*kw*u(1); %Back emf for phase W

sys(4)= 0;                   %HALL SENSOR A

sys(5)=5;                    %HALL SENSOR B

sys(6)=5;                    %HALL SENSOR C

%-----------------------------------------------5

end;

if (theta>=4*pi/3 & theta<3*pi/2)   % 240 TO  270 DEGREE
```

```
LA= -1;

LB=1;

LC=( theta -(4*pi/3))*6/pi;

sys(1)= LA*kw*u(1);     % Back emf for phase U

sys(2)=LB*kw*u(1);      %Back emf for phase V

sys(3)=LC*kw*u(1); %Back emf for phase W

 sys(4)= 0;                    %HALL SENSOR A

 sys(5)=0;                     %HALL SENSOR B

 sys(6)=5;                     %HALL SENSOR C

%-----------------------------------------------4

end;

if(theta>=3*pi/2 & theta<5*pi/3)   % 270 TO  300 DEGREE

 LA= -1;

 LB=((5*pi/3)-theta)*6/pi;

 LC=1;

 sys(1)= LA*kw*u(1);     % Back emf for phase U

 sys(2)=LB*kw*u(1);      %Back emf for phase V
```

```
sys(3)=LC*kw*u(1); %Back emf for phase W

sys(4)= 0;                  %HALL SENSOR A

sys(5)=0;                   %HALL SENSOR B

sys(6)=5;                   %HALL SENSOR C

%-----------------------------------------------3

end;

if(theta>=5*pi/3 & theta<11*pi/6)  % 300 TO  330 DEGREE

LA= -1;

LB=((5*pi/3)-theta)*6/pi;

LC=1;

sys(1)= LA*kw*u(1);     % Back emf for phase U

sys(2)=LB*kw*u(1);      %Back emf for phase V

sys(3)=LC*kw*u(1);      %Back emf for phase W

sys(4)= 5;                   %HALL SENSOR A

sys(5)=0;                   %HALL SENSOR B

sys(6)=5;                   %HALL SENSOR C

%-------------------------------------------------
```

```
end;

if(theta>=11*pi/6 & theta<2*pi)   % 330 TO  360 DEGREE

LA= 6*(theta-2*pi)/pi;

LB=-1;

LC=1;

sys(1)= LA*kw*u(1);     % Back emf for phase U

sys(2)= LB*kw*u(1);     %Back emf for phase V

sys(3)= LC*kw*u(1);     %Back emf for phase W

sys(4)= 5;                    %HALL SENSOR A

sys(5)=0;                     %HALL SENSOR B

sys(6)=5;                     %HALL SENSOR C

%-------------------------------------------

end;

%%%-------------CALCULATION FOR%%%SYS(4)<0-------------------------------------tt

if  (theta>=-2*pi  & theta<-11*pi/6) %-360 to -330

LA= (theta+2*pi)*6/pi;

LB=-1;
```

```
LC=1;

sys(1)= LA*kw*u(1);     % Back emf for phase U

sys(2)=LB*kw*u(1);      %Back emf for phase V

sys(3)=LC*kw*u(1);      %Back emf for phase W

sys(4)= 5;                  %HALL SENSOR A

sys(5)= 0;                  %HALL SENSOR B

sys(6)= 5;                  %HALL SENSOR C

%-------------------------------------------------

end;

if (theta>=-11*pi/6 & theta<-5*pi/3)%-330 to -300

 LA=1;

LB=-1;

LC=-(theta+(10*pi/6))*6/pi;

sys(1)= LA*kw*u(1);     % Back emf for phase U

sys(2)=LB*kw*u(1);      %Back emf for phase V

sys(3)=LC*kw*u(1);      %Back emf for phase W

sys(4)= 5;                  %HALL SENSOR A
```

```
sys(5)= 0;                  %HALL SENSOR B

sys(6)= 5;                  %HALL SENSOR C

%-------------------------------------------------

end;

if (theta>=-5*pi/3 & theta<-3*pi/2)%-300 to -270

LA=1;

LB=-1;

LC=-(theta+(10*pi/6))*6/pi;

sys(1)= LA*kw*u(1);     % Back emf for phase U

sys(2)=LB*kw*u(1);      %Back emf for phase V

sys(3)=LC*kw*u(1); %Back emf for phase W

sys(4)= 0;                  %HALL SENSOR A

sys(5)=0;                   %HALL SENSOR B

sys(6)=5;                   %HALL SENSOR C

%-------------------------------------------------

end;

if (theta>=-3*pi/2 & theta<-4*pi/3)%-270 to -240
```

```
LA=1;

LB=(  theta+(4*pi/3) )*6/pi;

LC=-1;

sys(1)= LA*kw*u(1);     % Back emf for phase U

sys(2)=LB*kw*u(1);      %Back emf for phase V

sys(3)=LC*kw*u(1); %Back emf for phase W

sys(4)= 0;                    %HALL SENSOR A

sys(5)=0;                     %HALL SENSOR B

sys(6)=5;                     %HALL SENSOR C

%-----------------------------------------------4

end;

if(theta>=-4*pi/3 & theta<-7*pi/6)%-240 to -210

LA=1;

LB=(  theta+(4*pi/3) )*6/pi;

LC=-1;

sys(1)= LA*kw*u(1);     % Back emf for phase U

sys(2)=LB*kw*u(1);      %Back emf for phase V
```

```
sys(3)=LC*kw*u(1);      %Back emf for phase W

sys(4)= 0;                    %HALL SENSOR A

sys(5)=5;                    %HALL SENSOR B

sys(6)=5;                    %HALL SENSOR C

end;

if (theta>=-7*pi/6 & theta<-pi)%-210 to -180

LA=-(theta+pi)*6/pi;

LB=1;

LC=-1;

sys(1)= LA*kw*u(1);     % Back emf for phase U

sys(2)=LB*kw*u(1);      %Back emf for phase V

sys(3)=LC*kw*u(1); %Back emf for phase W

sys(4)= 0;                    %HALL SENSOR A

sys(5)=5;                    %HALL SENSOR B

sys(6)=5;                    %HALL SENSOR C

%----------------------------------------------6

end;
```

```
if(theta>=-pi & theta<-5*pi/6)%-180 to -150
 LA=-(theta+pi)*6/pi;
LB=1;
LC=-1;
sys(1)= LA*kw*u(1);     % Back emf for phase U
sys(2)=LB*kw*u(1);      %Back emf for phase V
sys(3)=LC*kw*u(1); %Back emf for phase W
%---------------------------------------------- 7
sys(4)= 0;                    %HALL SENSOR A
sys(5)=5;                     %HALL SENSOR B
sys(6)=0;                     %HALL SENSOR C
end;
if(theta>=-5*pi/6 & theta<-4*pi/6)%-150 to -120
LA= -1;
LB=1;
LC=( theta+(4*pi/6))*6/pi;
sys(1)= LA*kw*u(1);     % Back emf for phase U
```

```
sys(2)=LB*kw*u(1);      %Back emf for phase V

sys(3)=LC*kw*u(1); %Back emf for phase W

sys(4)= 0;                    %HALL SENSOR A

sys(5)=5;                    %HALL SENSOR B

sys(6)=0;                    %HALL SENSOR C

%------------------------------------------------ 8

end;

if (theta>=-4*pi/6 & theta<-3*pi/6)%-120 to -90

LA= -1;

LB=1;

LC=( theta+(4*pi/6))*6/pi;

sys(1)= LA*kw*u(1);     % Back emf for phase U

sys(2)=LB*kw*u(1);      %Back emf for phase V

sys(3)=LC*kw*u(1);     %Back emf for phase W

sys(4)= 5;                    %HALL SENSOR A

sys(5)=5;                    %HALL SENSOR B

sys(6)=0;                    %HALL SENSOR C
```

```
end;

if(theta>=-3*pi/6 & theta<-2*pi/6)%-90 to -60

LA= -1;

LB=-(theta+(2*pi/6))*6/pi;

LC=1;

sys(1)= LA*kw*u(1);     % Back emf for phase U

sys(2)= LB*kw*u(1);     %Back emf for phase V

sys(3)= LC*kw*u(1);     %Back emf for phase W

sys(4)= 5;                    %HALL SENSOR A

sys(5)=5;                     %HALL SENSOR B

sys(6)=0;                     %HALL SENSOR C

end;

if(theta>=-2*pi/6 & theta<-pi/6)%-60 to -30

 LA= -1;

LB=-(theta+(2*pi/6))*6/pi;

LC=1;

sys(1)= LA*kw*u(1);     % Back emf for phase U
```

```
sys(2)=LB*kw*u(1);      %Back emf for phase V

sys(3)=LC*kw*u(1); %Back emf for phase W

sys(4)= 5;                    %HALL SENSOR A

sys(5)=0;                    %HALL SENSOR B

sys(6)=0;                    %HALL SENSOR C

%-------------------------------------------- 11

end;

if(theta>=-pi/6 & theta<0)%-30 to 0

LA= 6*theta/pi;

LB=-1;

LC=1;

sys(1)= LA*kw*u(1);     % Back emf for phase U

sys(2)=LB*kw*u(1);      %Back emf for phase V

sys(3)=LC*kw*u(1); %Back emf for phase W

sys(4)= 5;                    %HALL SENSOR A

sys(5)=0;                    %HALL SENSOR B

sys(6)=0;                    %HALL SENSOR C
```

```
end;
```

[2] S- Function Advance Angle

```
function [sys,x0,str,ts] = bldc(t,x,u,flag,angle)

% BLDC An example M-file S-function for defining a system of



% The expected input vector is:

% (1):  Wr

%(2):   theta

%OUTPUT VECTOR GENERATED BY THE SYSTEM:

%(1):  Emf_u

%(2):  Emf_v

%(3):  Emf_w

%(4):  Ha

%(5):  Hb

%(6):  Hc

%(7):  theta

% Dispatch the flag.
```

```
switch flag,

case 0,

[sys,x0,str,ts]=mdlInitializeSizes(angle); % Initialization

case 1,

sys = mdlDerivatives(t,x,u,angle); % Calculate derivatives

case 2,                                                    %model update

sys = mdlUpdate(t,x,u,angle); % update the model

case 3,

sys = mdlOutputs(t,x,u,angle); % Calculate outputs

case 4, % Unused flags

sys = [];

case 'reset'

% name = 'mybldc_mdl2/mybldc/my state-space';

%LocalResetController(name);

case 9,

% sys=mdlTerminate(t,x,u);

% Unexpected flags %
```

```
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
% End of csfunc.
%=============================================================
% mdlInitializeSizes
% Return the sizes, initial conditions, and sample times for the
% S-function.
function [sys,x0,str,ts] = mdlInitializeSizes(angle)
% Call simsizes for a sizes structure, fill it in and convert it
% to a sizes array.
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 3;
sizes.NumInputs      = 2;
sizes.DirFeedthrough = 1;     % Matrix D is nonempty.
```

```
sizes.NumSampleTimes = 1;

sys = simsizes(sizes);

% Initialize the initial conditions.

x0 = [];

% str is an empty matrix.

str = [];

% Initialize the array of sample times; in this example the sample

% time is continuous, so set ts to 0 and its offset to 0.

ts = [0 0];

% End of mdlInitializeSizes.

%=============================================================

% mdlDerivatives

% Return the derivatives for the continuous states.

% Nothing needs to be done here.

%=============================================================

function sys = mdlDerivatives(t,x,u,angle)

sys=[];
```

```
%==========================================================

% mdlUpdate

% Update the model parameters.

% Nothing needs to be done here

%==========================================================

 function sys = mdlUpdate(t,x,u,angle); % update the model

 sys=[];

% mdlOutput

% calculate the model outputs and update other blocks ( Adaptive coefficients)

function sys = mdlOutputs(t,x,u,angle)

u(1)=u(1) + u(2);    % + advance and - restard

theta = rem (u(1),2*pi);              % make theta betweeen 2*pi and -2*pi

%sys(7)=theta;

%sys(11)=theta;                        %output normalised angular position

if  (theta>=0 & theta<pi/6)       % 0 TO  30 DEGREE

sys(1)= 5;                    %HALL SENSOR A

sys(2)=0;                     %HALL SENSOR B
```

```
sys(3)=0;                   %HALL SENSOR C

end;

if (theta>=pi/6 & theta<pi/3)       % 30 TO  60 DEGREE

sys(1)= 5;                  %HALL SENSOR A

sys(2)=0;                   %HALL SENSOR B

sys(3)=0;                   %HALL SENSOR C

end;

if (theta>=pi/3 & theta<pi/2)      % 60 TO  90 DEGREE

sys(1)= 5;                   %HALL SENSOR A

sys(2)=5;                   %HALL SENSOR B

sys(3)=0;                   %HALL SENSOR C

end;

if (theta>=pi/2 & theta<2*pi/3)     % 90 TO  120 DEGREE

sys(1)= 5;                   %HALL SENSOR A

sys(2)=5;                   %HALL SENSOR B

sys(3)=0;                   %HALL SENSOR C

end;
```

```
if(theta>=2*pi/3 & theta<5*pi/6)    % 120 TO  150 DEGREE
 sys(1)= 0;                  %HALL SENSOR A
 sys(2)=5;                   %HALL SENSOR B
 sys(3)=0;                   %HALL SENSOR C
end;
if (theta>=5*pi/6 & theta<pi)     % 150 TO  180 DEGREE
 sys(1)= 0;                   %HALL SENSOR A
 sys(2)=5;                   %HALL SENSOR B
 sys(3)=0;                   %HALL SENSOR C
end;
if(theta>=pi & theta<7*pi/6)    % 180 TO  210 DEGREE
 sys(1)= 0;                  %HALL SENSOR A
 sys(2)=5;                   %HALL SENSOR B
 sys(3)=5;                   %HALL SENSOR C
end;
if(theta>=7*pi/6 & theta<4*pi/3)  % 210 TO  240 DEGREE
 sys(1)= 0;                   %HALL SENSOR A
```

```
sys(2)=5;                    %HALL SENSOR B

sys(3)=5;                    %HALL SENSOR C

end;

if (theta>=4*pi/3 & theta<3*pi/2)   % 240 TO  270 DEGREE

sys(1)= 0;                   %HALL SENSOR A

sys(2)=0;                    %HALL SENSOR B

sys(3)=5;                    %HALL SENSOR C

end;

if(theta>=3*pi/2 & theta<5*pi/3)   % 270 TO  300 DEGREE

sys(1)= 0;                   %HALL SENSOR A

sys(2)=0;                    %HALL SENSOR B

sys(3)=5;                    %HALL SENSOR C

%-------------------------------------------3

end;

if(theta>=5*pi/3 & theta<11*pi/6)  % 300 TO  330 DEGREE

sys(1)= 5;                   %HALL SENSOR A

sys(2)=0;                    %HALL SENSOR B
```

```
sys(3)=5;                    %HALL SENSOR C

end;

if(theta>=11*pi/6 & theta<2*pi)   % 330 TO  360 DEGREE

sys(1)= 5;                   %HALL SENSOR A

sys(2)=0;                    %HALL SENSOR B

sys(3)=5;                    %HALL SENSOR C

end;

%%%-------------CALCULATION FOR%%%SYS(4)<0---------------------------------------tt

if  (theta>=-2*pi  & theta<-11*pi/6) %-360 to -330

sys(1)= 5;                   %HALL SENSOR A

sys(2)= 0;                   %HALL SENSOR B

sys(3)= 5;                   %HALL SENSOR C

end;

if (theta>=-11*pi/6 & theta<-5*pi/3) %-330 to -300

sys(1)= 5;                   %HALL SENSOR A

sys(2)= 0;                    %HALL SENSOR B

sys(3)= 5;                   %HALL SENSOR C
```

```
end;

if (theta>=-5*pi/3 & theta<-3*pi/2)%-300 to -270

 sys(1)= 0;                %HALL SENSOR A

 sys(2)=0;                 %HALL SENSOR B

 sys(3)=5;                 %HALL SENSOR C

end;

if (theta>=-3*pi/2 & theta<-4*pi/3)%-270 to -240

 sys(1)= 0;                %HALL SENSOR A

 sys(2)=0;                 %HALL SENSOR B

 sys(3)=5;                 %HALL SENSOR C

end;

if(theta>=-4*pi/3 & theta<-7*pi/6)%-240 to -210

 sys(1)= 0;                %HALL SENSOR A

 sys(2)=5;                 %HALL SENSOR B

 sys(3)=5;                 %HALL SENSOR C

end;

if (theta>=-7*pi/6 & theta<-pi)%-210 to -180
```

```
sys(1)= 0;                  %HALL SENSOR A

sys(2)=5;                   %HALL SENSOR B

sys(3)=5;                   %HALL SENSOR C

end;

if(theta>=-pi & theta<-5*pi/6)%-180 to -150

%---------------------------------------------- 7

sys(1)= 0;                  %HALL SENSOR A

sys(2)=5;                   %HALL SENSOR B

sys(3)=0;                   %HALL SENSOR C

end;

if(theta>=-5*pi/6 & theta<-4*pi/6)%-150 to -120

sys(1)= 0;                  %HALL SENSOR A

sys(2)=5;                   %HALL SENSOR B

sys(3)=0;                   %HALL SENSOR C

end;

if (theta>=-4*pi/6 & theta<-3*pi/6)%-120 to -90

sys(1)= 5;                  %HALL SENSOR A
```

```
sys(2)=5;                  %HALL SENSOR B

sys(3)=0;                  %HALL SENSOR C

end;

f(theta>=-3*pi/6 & theta<-2*pi/6)%-90 to -60

sys(1)= 5;                  %HALL SENSOR A

sys(2)=5;                  %HALL SENSOR B

sys(3)=0;                  %HALL SENSOR C

%------------------------------------------------- 10

end;

if(theta>=-2*pi/6 & theta<-pi/6)%-60 to -30

sys(1)= 5;                  %HALL SENSOR A

sys(2)=0;                  %HALL SENSOR B

sys(3)=0;                  %HALL SENSOR C

end;

if(theta>=-pi/6 & theta<0)%-30 to 0

sys(1)= 5;                  %HALL SENSOR A

sys(2)=0;                  %HALL SENSOR B
```

```
sys(3)=0;                    %HALL SENSOR C

end;

Private Sub Form_Load()

On Error Resume Next

MSComm1.PortOpen = True

'---------- ini timer-----

Timer1.Enabled = False

get_speed_torque.Enabled = False

cmd_advance.Enabled = False

cmd_load.Enabled = False

cmd_drive_adv.Enabled = False

cmd_enable_drive.Enabled = False

cmd_disable_drive.Enabled = False

cmd_floatint_hall.Enabled = False

cmd_drive_normal.Enabled = False

step_pwm.Enabled = False

st_rec.Enabled = False
```

```
t_normal.Enabled = True

t_advance.Enabled = False

start_rec.Enabled = False

auto_step_load.Enabled = False

MANUAL_LOAD.Enabled = False

AUTO_LOAD.Enabled = False

c_data = 0

CLEAR_DATA.Enabled = False

CLR.Enabled = False

REPORT_EXCEL.Enabled = False

report.Enabled = False

'----------initial variable--------

ref_datax = 0

ref_datay = 0

num_data = 0

step_load = 0

max_load = 0.3
```

```
min_load = 0

num_plot = 0

max_speed = 500

min_speed = 200

scalex1 = 0

scaley2 = 0

dx = 0

dy = 0

record = False

toggle = 0

avr_value = 0

avr_value2 = 0

avr_value3 = 0

value_x = 0

value_y = 0

load_data = 0

End Sub
```

```
Private Sub Label25_Click()

End Sub

Private Sub Form_MouseMove (Button As Integer, Shift As Integer, x As Single,

Y As Single)

BLDC2013.Caption = "ADVANCE ANGLE BLDC TECHNOLOGY"

End Sub

Private Sub Form_Unload(Cancel As Integer)

End

End Sub

Private Sub MANUAL_LOAD_Timer()

'------------code command -------

If c_data = 0 Then

MSComm1.Output = Chr$(2)

c_data = c_data + 1

End If

'------------data command---------

If c_data = 1 Then
```

```
MSComm1.Output = Chr$(PWM_.Value)

c_data = c_data + 1

End If

'------------------ stop command---------

If c_data = 2 Then

MSComm1.Output = Chr$(0)

c_data = 0

MANUAL_LOAD.Enabled = False

'Exit Sub

End If

End Sub

Private Sub max_t_Click()

input_var.Show

End Sub

Private Sub MSComm1_OnComm()

Dim finish$

Dim offset$
```

```
Dim datain$, A$, b$, c$, d$, G$, h$

Dim loop_check

Dim rpm$

Dim count$

Dim real_speed

 Do

buffer$ = buffer$ & MSComm1.Input

loop_check = loop_check + 1

If loop_check > 100000 Then

Exit Sub

End If

Loop Until InStr(buffer$, "s")

datain$ = buffer$

Text1 = datain$

A$ = Mid$(datain$, 1, 1)

If Val(A$) < 1 And Val(A$) > 6 Then GoTo lb7

If Val(A$) = 1 Then GoTo lb1
```

```
If Val(A$) = 2 Then GoTo lb2

If Val(A$) = 3 Then GoTo lb3

If Val(A$) = 4 Then GoTo lb4

If Val(A$) = 5 Then GoTo lb5

If Val(A$) = 6 Then GoTo lb6

lb1:

b$ = Mid$(datain$, 17, 6)

'SHOW_SPEED.Text = b$

If Val(b$) > 1000 Then

real_speed = Val(b$) * 64 * 0.0000002 'time/0.5rev (64 is div_clk 64 and 0.0000002 is 1 T of clock)use t1

real_speed = real_speed * 2 ' time/rev

If real_speed > 0 Then real_speed = 60 / real_speed   ' rpm

End If

If avr_value3 <= 9 Then

sum_rpm = sum_rpm + real_speed

avr_value3 = avr_value3 + 1
```

```
Else



sum_rpm = sum_rpm / 10

real_speed = sum_rpm

rpm = real_speed

SHOW_SPEED.Text = Format$(real_speed, "###")

avr_value3 = 0

sum_rpm = 0

power = real_current * real_voltage

If rpm > 0 Then torque = power / rpm

show_loadtorque.Text = Format$(torque, "0.##")

End If

'------------------------display current

c$ = Mid$(datain$, 7, 4)

current = Val(c$) * 5

current = current / 1024

current = current * 3
```

```
If avr_value <= 9 Then

sum_current = sum_current + current

avr_value = avr_value + 1

Else

sum_current = sum_current / 10

current = sum_current

current = current / 0.6

real_current = current

show_loadcurrent.Text = Format$(current, "##.00")

avr_value = 0

sum_current = 0

End If

'------------------------ display voltage

d$ = Mid$(datain$, 12, 4)

voltage = Val(d$) * 5

voltage = voltage / 1024

voltage = voltage * 10
```

```
If avr_value2 <= 9 Then

sum_voltage = sum_voltage + voltage

avr_value2 = avr_value2 + 1

Else

sum_voltage = sum_voltage / 10

voltage = sum_voltage

real_voltage = voltage

show_loadvoltage.Text = Format$(voltage, "##.00")

avr_value2 = 0

sum_voltage = 0

 End If

'-------------------------display load torque

lb2:

lb3:

lb4:

lb5:
```

```
lb6:

lb7:

End Sub

Private Sub PWM_LOAD_Change()

PWM_LOAD.Enabled = True

End Sub

Private Sub Option1_Click()

adv_angle = 0

cmd_advance.Enabled = True

End Sub

Private Sub Option2_Click()

adv_angle = 5

cmd_advance.Enabled = True

End Sub

Private Sub Option3_Click()

adv_angle = 7

cmd_advance.Enabled = True
```

```
End Sub

Private Sub Option4_Click()

adv_angle = 9

cmd_advance.Enabled = True

End Sub

Private Sub Option5_Click()

adv_angle = 15

cmd_advance.Enabled = True

End Sub

Private Sub Option6_Click()

adv_angle = 20

cmd_advance.Enabled = True

End Sub

Private Sub Option7_Click()

adv_angle = 25

cmd_advance.Enabled = True

End Sub
```

```
Private Sub Option8_Click()

adv_angle = 30

cmd_advance.Enabled = True

End Sub

Private Sub Picture1_MouseMove(Button As Integer, Shift As Integer, x As Single, Y As
Single)

BLDC2013.Caption = "ADVANCE ANGLE BLDC TECHNOLOGY" & " " & "speed =" &
x & " " & "RPM" & " " & "load =" & Y & " " & "N.m"

End Sub

Private Sub Picture1_Paint()

divy = (max_load - min_load) / 10

divx = (max_speed - min_speed) / 10

'-----------------------------------DISPLAY SCALE LOAD----------------

lbt1.Caption = min_load + (divy * 1)

lbt2.Caption = min_load + (divy * 2)

lbt3.Caption = min_load + (divy * 3)

lbt4.Caption = min_load + (divy * 4)
```

```
lbt5.Caption = min_load + (divy * 5)

lbt6.Caption = min_load + (divy * 6)

lbt7.Caption = min_load + (divy * 7)

lbt8.Caption = min_load + (divy * 8)

lbt9.Caption = min_load + (divy * 9)

lbt10.Caption = min_load + (divy * 10)

'----------------------------------DISPLAY SCALE E_SPEED-------------------

lbsp1.Caption = min_speed + (divx * 1)

lbsp2.Caption = min_speed + (divx * 2)

lbsp3.Caption = min_speed + (divx * 3)

lbsp4.Caption = min_speed + (divx * 4)

lbsp5.Caption = min_speed + (divx * 5)

lbsp6.Caption = min_speed + (divx * 6)

lbsp7.Caption = min_speed + (divx * 7)

lbsp8.Caption = min_speed + (divx * 8)

lbsp9.Caption = min_speed + (divx * 9)

lbsp10.Caption = min_speed + (divx * 10)
```

```
'--------------------------------SET SCALE--------------------------------------

Picture1.Cls

Picture1.DrawWidth = 2

Picture1.DrawStyle = 0

scalex1 = min_speed

scaley1 = max_load

scalex2 = max_speed

scaley2 = min_load

Picture1.Scale (scalex1, scaley1)-(scalex2, scaley2)

'--------------------------------draw gride scale x-------------------------

Picture1.DrawWidth = 1

Picture1.DrawStyle = 2

For x = min_speed To max_speed Step divx

Picture1.Line (x, 0)-(x, max_load)

Next x

'--------------------------------draw gride scale y-------------------------

For x = min_load To max_load Step divy
```

```
Picture1.Line (0, x)-(max_speed, x)

Next x

'Me.Circle (1500, 1500), 150, vbBlack

End Sub

Private Sub PWM__Change()

MANUAL_LOAD.Enabled = True

End Sub

Private Sub report_Click()

Dim MYOBJECT As Object

Dim k

Dim j

Dim t$

Dim S$

Dim A, b As Integer, c

Dim colum As String

Dim index

index = 65
```

```
colum = Chr$(index)

Set MYOBJECT = CreateObject("Excel.Application")

MYOBJECT.workbooks.Add

'MsgBox (W2$)

For j = 0 To ref_datax - 1

'-----------------------------

If j = 0 Then

For k = 0 To save_numdata(j)

colum = Chr$(index)

t$ = colum & k + 1

colum = Chr$(index + 1)

S$ = colum & k + 1

MYOBJECT.range(S$).Value = save_datax(j, k)

MYOBJECT.range(t$).Value = save_datay(j, k)

Next k

End If

'-----------------------------
```

```
If j = 1 Then

For k = 0 To save_numdata(j)

colum = Chr$(index + 2)

t$ = colum & k + 1

colum = Chr$(index + 3)

S$ = colum & k + 1

MYOBJECT.range(S$).Value = save_datax(j, k)

MYOBJECT.range(t$).Value = save_datay(j, k)

Next k

End If

'-----------------------------------

If j = 2 Then

For k = 0 To save_numdata(j)

colum = Chr$(index + 4)

t$ = colum & k + 1

colum = Chr$(index + 5)

S$ = colum & k + 1
```

```
MYOBJECT.range(S$).Value = save_datax(j, k)

MYOBJECT.range(t$).Value = save_datay(j, k)

Next k

End If

'-----------------------------------------

If j = 3 Then

For k = 0 To save_numdata(j)

colum = Chr$(index + 6)

t$ = colum & k + 1

colum = Chr$(index + 7)

S$ = colum & k + 1

MYOBJECT.range(S$).Value = save_datax(j, k)

MYOBJECT.range(t$).Value = save_datay(j, k)

Next k

End If

Next j
```

```
MYOBJECT.Visible = True

'------------------------ENABLE COMMAND----------

CLEAR_DATA.Enabled = True

CLR.Enabled = True

CLEAR_DATA.Enabled = True

CLR.Enabled = True

REPORT_EXCEL.Enabled = False

 report.Enabled = False

End Sub

Private Sub REPORT_EXCEL_Click()

Dim MYOBJECT As Object

Dim k

Dim j

Dim t$

Dim S$

Dim A, b As Integer, c

Dim colum As String
```

```
Dim index

index = 65

colum = Chr$(index)

Set MYOBJECT = CreateObject("Excel.Application")

MYOBJECT.workbooks.Add

For j = 0 To ref_datax - 1

'------------------------------

If j = 0 Then

For k = 0 To save_numdata(j)

colum = Chr$(index)

t$ = colum & k + 1

colum = Chr$(index + 1)

S$ = colum & k + 1

MYOBJECT.range(S$).Value = save_datax(j, k)

MYOBJECT.range(t$).Value = save_datay(j, k)

Next k

End If
```

```
If j = 1 Then

For k = 0 To save_numdata(j)

colum = Chr$(index + 2)

t$ = colum & k + 1

colum = Chr$(index + 3)

S$ = colum & k + 1

MYOBJECT.range(S$).Value = save_datax(j, k)

MYOBJECT.range(t$).Value = save_datay(j, k)

Next k

End If

'--------------------------------------

If j = 2 Then

For k = 0 To save_numdata(j)

colum = Chr$(index + 4)

t$ = colum & k + 1

colum = Chr$(index + 5)

S$ = colum & k + 1
```

```
MYOBJECT.range(S$).Value = save_datax(j, k)

MYOBJECT.range(t$).Value = save_datay(j, k)

Next k

End If

'-----------------------------------------

If j = 3 Then

For k = 0 To save_numdata(j)

colum = Chr$(index + 6)

t$ = colum & k + 1

colum = Chr$(index + 7)

S$ = colum & k + 1

MYOBJECT.range(S$).Value = save_datax(j, k)

MYOBJECT.range(t$).Value = save_datay(j, k)

Next k

End If

Next j

MYOBJECT.Visible = True
```

```
End Sub

Private Sub run_advance_Click()

cmd_drive_adv.Enabled = True

t_advance.Enabled = True

t_normal.Enabled = False

End Sub



Private Sub run_normal_Click()

cmd_drive_normal.Enabled = True

t_normal.Enabled = True

t_advance.Enabled = False

End Sub

Private Sub scal_rpm_Change()

 If record = True Then

   x1 = Val(scal_rpm.Text)

   y1 = Val(scal_t.Text)

 If (x1 < value_x) Then
```

```
 save_datax(ref_datax, num_data) = x1

 save_datay(ref_datay, num_data) = y1

 num_data = num_data + 1

Picture1.Line (value_x, value_y)-(x1, y1), QBColor(num_plot)

End If

  value_x = x1

  value_y = y1

 End If

End Sub



Private Sub scal_t_Change()

End Sub

Private Sub SCALE_Click()

scal_rpm.Text = ""

input_var.Show

st_rec.Enabled = False

End Sub
```

```
Private Sub show_loadtorque_Change()

scal_t.Text = show_loadtorque.Text

End Sub

Private Sub SHOW_SPEED_Change()

scal_rpm.Text = SHOW_SPEED.Text

End Sub

Private Sub st_rec_Timer()

If toggle = 0 Then

Timer1.Enabled = True

toggle = toggle + 1

Else

Timer1.Enabled = False

toggle = 0

End If

End Sub

Private Sub start_rec_Click()

show_loadtorque.Text = "0.00"
```

```
SHOW_SPEED.Text = "000"

show_loadcurrent.Text = "0.00"

show_loadvoltage.Text = "0.00"

st_rec.Enabled = True

avr_value = 0

avr_value2 = 0

avr_value3 = 0

sum_current = 0

sum_voltage = 0

sum_rpm = 0

value_x = 0

value_y = 0

x1 = Val(scal_rpm.Text)

y1 = Val(scal_t.Text)

save_datax(ref_datax, num_data) = x1

save_datay(ref_datay, num_data) = y1

record = True
```

```
Picture1.DrawWidth = 2

Picture1.DrawStyle = 0

'-----------------------DISABLE COMMAND-----

 CLEAR_DATA.Enabled = False

 CLR.Enabled = False

 REPORT_EXCEL.Enabled = False

End Sub

Private Sub step_pwm_Timer()

step_load = step_load + 15

PWM_.Value = step_load

If step_load > 240 Then

step_pwm.Enabled = False

record = False

save_numdata(ref_datax) = num_data

ref_datax = ref_datax + 1

ref_datay = ref_datax

num_data = 0
```

```
num_plot = num_plot + 1

PWM_.Value = 0

'---------------ENABLE COMMAND------

 CLEAR_DATA.Enabled = False

 CLR.Enabled = False

 REPORT_EXCEL.Enabled = True

 report.Enabled = True

End If

End Sub



Private Sub t_advance_Timer()

If display_mode = 0 Then

displaymode.BackColor = QBColor(12)

display_mode = display_mode + 1

displaymode.Caption = "ADVANCE MODE"

Else

displaymode.BackColor = QBColor(7)
```

```
display_mode = 0

displaymode.Caption = ""

End If

End Sub

Private Sub t_normal_Timer()

If display_mode = 0 Then

displaymode.BackColor = QBColor(10)

display_mode = display_mode + 1

displaymode.Caption = "NORMAL MODE"

Else

displaymode.BackColor = QBColor(7)

display_mode = 0

displaymode.Caption = ""

End If

End Sub

Private Sub Text2_Change()

code = Val (Text2.Text)
```

```
End Sub

Private Sub Text3_Change()

PWM = Val(Text3.Text)

End Sub

Private Sub Timer1_Timer()

'------------code command -------

If c_data = 0 Then

MSComm1.Output = Chr$(1)

 c_data = c_data + 1

 End If

'------------data command---------

If c_data = 1 Then

 MSComm1.Output = Chr$(0)

 c_data = c_data + 1

End If

'------------------ stop command---------

If c_data = 2 Then
```

```
MSComm1.Output = Chr$(0)

c_data = 0

Timer1.Enabled = False

'Exit Sub

End If

End Sub
```

PUBLIC VARIABLE

```
Public max_load

Public min_load

Public max_speed

Public min_speed

Public scalex1

Public scalex2

Public scaley1

Public scaley2

Public divx
```

Public divy

Public dx

Public dy

Public toggle

Public current

Public voltage

Public avr_value

Public avr_value2

Public avr_value3

Public sum_rpm

Public sum_current

Public sum_voltage

Public real_current

Public real_voltage

Public torque

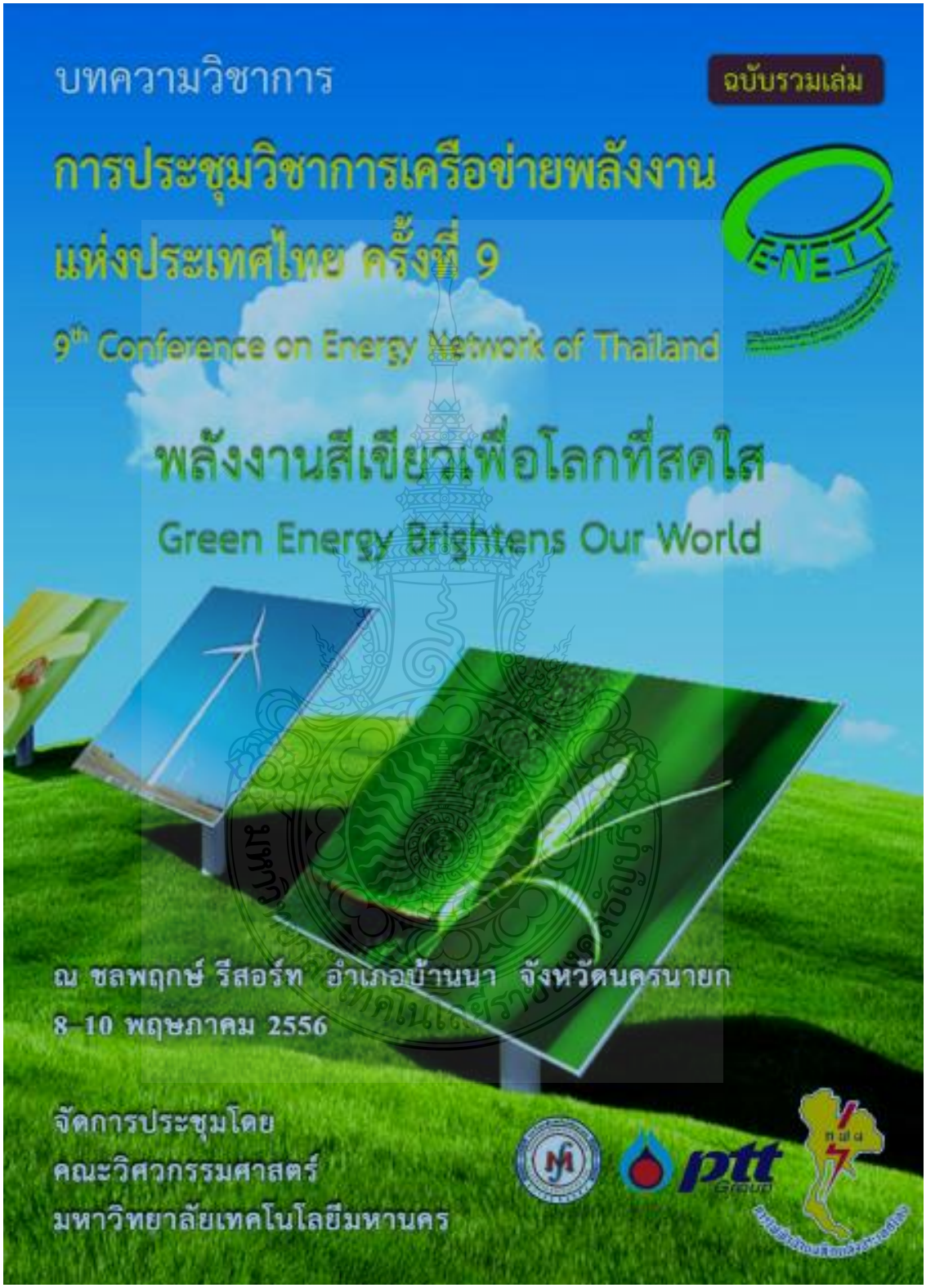
Public power

Public rpm

```
Public step_load

Public record As Boolean

Public x1

Public y1

Public value_x

Public value_y

Public num_plot

Public save_datax(5, 500)

Public save_datay(5, 500)

Public ref_datax

Public ref_datay

Public num_data

Public load_data

Public save_numdata(5)
```

ภาคผนวก จ

ผลงานตีพิมพ์เผยแพร่


บทความวิชาการ
ฉบับรวมเล่ม
การประชุมวิชาการเครือข่ายพลังงาน
แห่งประเทศไทย ครั้งที่ 9
9th Conference on Energy Network of Thailand
E-NET
พลังงานสีเขียวเพื่อโลกที่สดใส
Green Energy Brightens Our World
ณ ชลพฤกษ์ รีสอร์ท อำเภอบ้านนา จังหวัดนครนายก
8-10 พฤษภาคม 2556
จัดการประชุมโดย
คณะวิศวกรรมศาสตร์
มหาวิทยาลัยเทคโนโลยีมหานคร
ptt group

สารบัญ

ผู้ทรงคุณวุฒิพิจารณาบทความ

มหาวิทยาลัยเชียงใหม่
ศ.ดร. ทนงเกียรติ เกียรติศิริโรจน์
ผศ.ดร. กอดขวัญ นามสงวน
ผศ.ดร. ศิวะ อัจฉริยวิริยะ

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี
ศ.ดร. สมชาติ โสภณรณฤทธิ์
รศ.ดร. สักกมน เทพหัสดิน ณ อยุธยา

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ
ดร. เพ็ญญารัตน์ จินดา
ดร. อำนาจ บุญลอย
ดร. ฉัตรชัย นิมมล

มหาวิทยาลัยเทคโนโลยีมหานคร
รศ.ดร. สมิทธ์ เอี่ยมสอาด
รศ.ดร. ฐานิตย์ เมธิยานนท์
ผศ.ดร. นุภาพ แย้มไตรพัฒน์
ผศ.ดร. พรชัย นิเวศน์รังสรรค์
ผศ.ดร. วิชาญ คงเกียรติไพบูลย์
ผศ.ดร. สมชาย ศรีพัฒนพิพัฒน์
ผศ.ดร. ศุภเกียรติ ศรีพนมธนากร
ผศ.ดร. ขวัญจิต วงษ์ชารี
ผศ.ดร. สลิลทิพย์ สินธุสนธิชาติ
ผศ.ดร ประสาน สถิตย์เรืองศักดิ์
ดร. วาโย ช้างเจริญ
ดร. สมศักดิ์ เพ็ชร์กุล
ดร. วิไลลักษณ์ สระมูล
ดร. ฐิตะพล หุยะนันท์
อ. กิตติศักดิ์ ยงศิริ
อ. ขวัญชัย หนาแน่น
อ. ปุณยภัทร ภูมิภาค
อ. ไมตรี กระมุทพิจิตร

มหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรี
ดร. กฤษณ์ชนม์ ภูมิกิตติพิชญ์
ดร. สมชัย หิรัญวโรดม
ดร. สถาพร ทองวิค
ดร. สโรชา เจริญวัย
ดร. สรพงษ์ ภวสุปรีย์

มหาวิทยาลัยเทคโนโลยีราชมงคลรัตนโกสินทร์
ผศ. วิศิษฐ์ ลีลาผาติกุล

มหาวิทยาลัยเทคโนโลยีสุรนารี
ดร. ชโลธร ธรรมแท้
ดร. กีรติ สุลักษณ์
ดร. ธีระชาติ พรพิบูลย์

มหาวิทยาลัยนเรศวร
ผศ.ดร. สมชาย มณีวรรณ์
ดร. ยงยุทธ ชนบดีเฉลิมรุ่ง
ดร. สิริมาส เฮงรัศมี
ดร. สันต์ จันทร์สมศักดิ์
ดร. สุพรรณนิกา วัฒนา
ดร. ศรายุทธ วัยวุฒิ

มหาวิทยาลัยบูรพา
ดร. มัณฑนา รังสิโยภาส
ดร. วัชรินทร์ ดงบัง

มหาวิทยาลัยมหาสารคาม
ผศ.ดร.เจริญพร เลิศสถิตธนกร
ผศ.ดร. ณัฐพล ภูมิสะอาด
ดร. ธีรพัฒน์ ชมภูคำ
ดร. โสภา สุวแพทย์
ดร. นิดา ชัยมูล
ดร. มณีรัตน์ องค์วรรณดี

มหาวิทยาลัยมหิดล
ดร. รุ่ง กิตติพิชัย
ดร. วรศิษฐ์ ตรูทัศนวินท์
รศ. ศุภชัย นาทะพันธ์

ดร. ขวัญชัย จ้อยเจริญ
ดร. วันชัย ทรัพย์สิงห์
ดร. บุณย์ฤทธิ์ ประสาทแก้ว
ดร. วารุณี อริยะวิริย
ดร. ฉัตรชัย ศุภพิทักษ์สกุลนันท์
ดร. บุญยัง ปลั่งกลาง
ดร. อำนวย เรืองวารี

มหาวิทยาลัยราชภัฏบ้านสมเด็จเจ้าพระยา
ดร. โยธิน อิ่งกูล
ดร. ยิ่งรักษ์ อรรถเวชกุล

มหาวิทยาลัยราชภัฏวไลยอลงกรณ์
รศ.ดร. วัชระ เพิ่มชาติ

มหาวิทยาลัยรามคำแหง
ผศ.ดร. สมพร ธเนศวาณิชย์

มหาวิทยาลัยศรีนครินทรวิโรฒ
ดร. กิตติ สถาพรประสาธน์

มหาวิทยาลัยศรีปทุม
ดร. เทพฤทธิ์ ทองซุบ
ดร. กีรติ ชยะกุลศิริ
ดร. วิชชากร เฮงศรีธวัช
ดร. ชลธิศ เอี่ยมวรวุฒิกุล
ดร. วริสรา เลิศไพฑูรย์พันธ์
ดร. นิมิต บุญภิรมย์
อ. อภิรักษ์ สวัสดิ์กิจ
อ. เผชิญ จันทร์สา

มหาวิทยาลัยสงขลานครินทร์
ผศ.ดร. ชยุต นันทดุสิต

มหาวิทยาลัยอุบลราชธานี
ผศ.ดร. อำไพศักดิ์ ทีบุญมา

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
รศ.ดร. พงษ์เจต พรหมวงศ์
รศ.ดร. ชินรักษ์ เธียรพงษ์
รศ.ดร. จารุวัตร เจริญสุข
รศ.ดร. จำลอง ปราบแก้ว
ผศ.ดร. มนต์ศักดิ์ พิมสาร
ผศ.ดร. ณัฐวุฒิ เดไปวา
ดร. ดอน อิศรากร
ดร. วิทาดา เจษฎารัตนชัย
ดร. กุลนันทน์ เกียรติกิตติพงษ์

คณะกรรมการเครือข่ายพลังงานแห่งประเทศไทย

จุฬาลงกรณ์มหาวิทยาลัย
มหาวิทยาลัยเชียงใหม่
มหาวิทยาลัยทักษิณ
มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี
มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ
มหาวิทยาลัยเทคโนโลยีมหานคร
มหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรี
มหาวิทยาลัยเทคโนโลยีราชมงคลสุวรรณภูมิ
มหาวิทยาลัยเทคโนโลยีสุรนารี
มหาวิทยาลัยธรรมศาสตร์
มหาวิทยาลัยธุรกิจบัณฑิตย์
มหาวิทยาลัยนเรศวร
มหาวิทยาลัยบูรพา
มหาวิทยาลัยมหาสารคาม
มหาวิทยาลัยมหิดล
มหาวิทยาลัยแม่โจ้
มหาวิทยาลัยรังสิต
มหาวิทยาลัยราชภัฏนครศรีธรรมราช
มหาวิทยาลัยราชภัฏลำปาง

มหาวิทยาลัยราชภัฏเลย
มหาวิทยาลัยราชภัฏอุดรดิตถ์
มหาวิทยาลัยราชภัฏอุบลราชธานี
มหาวิทยาลัยศรีนครินทรวิโรฒ
มหาวิทยาลัยศรีปทุม
มหาวิทยาลัยศิลปากร

ระบบจำลองการขยายช่วงความเร็วและแรงบิดในมอเตอร์ไฟตรงไร้แปรงถ่านด้วยโปรแกรม MATLAB/Simulink

Modeling of the speed–torque extending in Brushless DC Motor using MATLAB/Simulink

สำเริง เต็มราม[๑] และ วันชัย ทรัพย์สิงห์[๒,*]

[๑] ภาควิชาไฟฟ้ากำลัง สำนักพัฒนาสมรรถนะครูและบุคลากรอาชีวศึกษา กรุงเทพมหานคร

[๒] ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรี ปทุมธานี ๑๒๑๑๐

*ผู้ติดต่อ: wanchai.s@en.rmutt.ac.th

บทคัดย่อ

บทความนี้นำเสนอการศึกษาและพัฒนาแบบจำลองทางคณิตศาสตร์ ด้วยโปรแกรม MATLAB/Simulink เพื่อวิเคราะห์วิธีการขับเคลื่อน มอเตอร์ไฟตรงไร้แปรงถ่าน (Brushless DC Motor) แบบมีการชดเชยเฟสแบบก้าวหน้า (Phase Advance) ระหว่างกระแสขาเข้ากับแรงดัน Back EMF ในแต่ละเฟสของมอเตอร์ เพื่อปรับปรุงแรงบิดและเพิ่มความเร็วให้มอเตอร์ในย่านการทำงานแบบกำลังคงที่ (Constant Power Region) ซึ่งเป็นองค์ประกอบที่สำคัญต่อการออกแบบชุดควบคุมการขับเคลื่อนมอเตอร์ไฟตรงไร้แปรงถ่าน ทั้งนี้ในบทความจะศึกษาผลการตอบสนองทางด้านแรงบิดและความเร็วของมอเตอร์ในระบบวงปิด ที่มีค่าการชดเชยมุมเฟสแตกต่างกัน เพื่อนำผลที่ได้ไปวิเคราะห์และสร้างชุด ควบคุมการขับเคลื่อนมอเตอร์ดังกล่าวให้เหมาะสมกับการนำไปใช้งานต่อไป

คำหลัก: การชดเชยมุมเฟสก้าวหน้า มอเตอร์ไฟตรงชนิดไร้แปรงถ่าน

Abstract

This paper presents the investigation and the development of a mathematical model of the Brushless Direct Current (BLDC) Motor using MATLAB / Simulink program. It concerns in Performance analysis of the BLDC Motor when it operates in a constant power region. This leads into an implemented of Phase Advanced Angle Compensation (PAAC) method in order to improve the torque/ speed characteristic when running in an over-rated speed range. Hence, this paper considers in studying of motor performances both without PAAC and without PAAC. Furthermore, it may useful for the implementation of the Phase Advance Angle Compensation method with the BLDC Motor in practice.

Keywords: Phase Advance Angle Compensation, Brushless DC Machine.

๑. บทนำ

ปัจจุบัน Brushless DC (BLDC) Motor เป็นที่สนใจในงานอุตสาหกรรมและมีการประยุกต์ใช้ในหลายๆ ด้านเช่น ในอุตสาหกรรมการบินและอวกาศ อุตสาหกรรมการแพทย์ ระบบควบคุมอัตโนมัติ และ อุตสาหกรรมยานยนต์ เป็นต้น โดยเฉพาะเป็นตัวขับในล้อ (In-Wheel Motor) ของรถยนต์ไฟฟ้าขนาดเล็กรุ่นใหม่ คุณสมบัติของ BLDC Motor มีจุดเด่นเมื่อเทียบกับมอเตอร์แบบเหนี่ยวนำ [๑] เช่น

- คุณลักษณะของความเร็ว-แรงบิดคล้าย DC Motor
- ตอบสนองต่อการเปลี่ยนแปลงความเร็วได้ดี
- ประสิทธิภาพในการทำงานสูง
- ตอบสนองการทำงานในย่านความเร็วสูงได้ดี
- มีค่าแรงบิดสูงกว่าเมื่อเทียบกับขนาดมอเตอร์

ภาควิชาวิศวกรรมเครื่องกล
และภาควิชาวิศวกรรมไฟฟ้ากำลัง

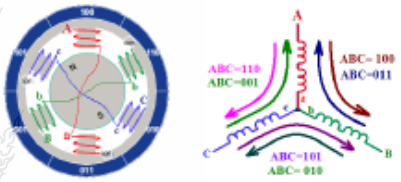
คณะวิศวกรรมศาสตร์
มหาวิทยาลัยเทคโนโลยีมหานคร

รหัสบทความ:
OT005

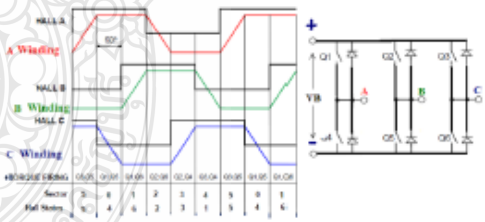
การประชุมวิชาการเครือข่ายพลังงานแห่งประเทศไทยครั้งที่ 9
8-10 พฤษภาคม 2556 จังหวัดนครนายก

นอกจากนี้ยังมีขนาดน้ำหนักต่อพิกัดกำลัง (Weight/Watt) อยู่ในเกณฑ์ต่ำ สามารถนำไปใช้ในส่วนที่ถูกจำกัดพื้นที่ได้ดีกว่า

แรงบิดของ BLDC Motor ขึ้นอยู่กับรูปคลื่นของแรงดัน Back EMF ในขดลวดสเตเตอร์ [๑] โดยทั่วไปแรงดัน Back EMF จะเป็นรูปคลื่นสี่เหลี่ยมคางหมู และกระแสจะเป็นรูปคลื่นสามเหลี่ยม ขณะทำงานในย่านแรงบิดคงที่ ส่วนในย่านกำลังไฟฟ้าคงที่ มอเตอร์จะทำงานที่ความเร็วสูงขึ้นในขณะที่แรงบิดอาจลดลงต่ำกว่าแรงบิดโหลด จึงจำเป็นต้องควบคุมให้แรงบิดมีค่าเหมาะสมต่อการใช้งานในย่านการทำงานนี้ เนื่องจาก BLDC Motor ไม่สามารถควบคุมกระแสขดลวดสนามได้เช่นเดียวกับ DC Motor ทั่วไป อย่างไรตามค่าของแรงบิดและกระแส จากการจำลองระบบในบทความนี้อาจมีผลตอบสนอง ต่างจากผลทดลองที่ได้ในเชิงปฏิบัติ เนื่องจากในทางปฏิบัติ แรงบิดและกระแสจะมี ripple เกิดขึ้นได้ในขณะทำงาน แต่การจำลองมีข้อดี ต่อการวิเคราะห์ระบบเพื่อควบคุมการทำงานให้สมบูรณ์ยิ่งขึ้น



รูปที่ ๒.๑ การควบคุมการไหลของกระแสสเตเตอร์



รูปที่ ๒.๒ การควบคุมสวิตช์อิเล็กทรอนิกส์ให้สอดคล้องตามรูปที่ ๒.๑

๒.๑ แบบจำลองทางคณิตศาสตร์ของ BLDC Motor

๒. ทฤษฎีพื้นฐาน

BLDC Motor ใช้แม่เหล็กถาวรแทนขดลวดสนามจึงมีคุณลักษณะทำงานคล้ายกับมอเตอร์ซิงโครนัส ซึ่งความถี่โรเตอร์จะเท่ากับความถี่สเตเตอร์ ควบคุมกระแสขาเข้าด้วยตำแหน่งโรเตอร์ ในปัจจุบัน BLDC Motor มีทั้งชนิด ๑ เฟส ๒ เฟสและ ๓ เฟส ขึ้นอยู่กับจำนวนชุดขดลวดสเตเตอร์ ที่นิยมใช้ เป็นแบบ ๓ เฟส

การตรวจจับตำแหน่งโรเตอร์ที่ใช้ในการควบคุมการไหลของกระแสสเตเตอร์ ทำได้โดยใช้ Hall sensor ตรวจจับการเปลี่ยนแปลงสนามแม่เหล็ก หรือตรวจจับหาจุดตัดแรงดันต้านกลับ (Back EMF Zero Crossing Detection) เพื่อควบคุมแรงปฏิกริยาจากสนามแม่เหล็กระหว่างโรเตอร์และสเตเตอร์ให้เกิดขึ้นที่มุมที่เหมาะสมต่อการทำงานในย่านนั้นๆ

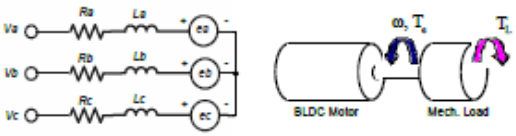
รูปที่ ๒.๑ และ ๒.๒ แสดงหลักการควบคุมการไหลของกระแสสเตเตอร์ด้วยตำแหน่งโรเตอร์โดยใช้ Hall sensor ตรวจจับการเปลี่ยนแปลงของสนามแม่เหล็ก

วงจรสมมูลของ BLDC Motor แสดงดังรูปที่ ๒.๓ ซึ่งสามารถเขียนสมการทางคณิตศาสตร์ได้ดังนี้ [๒]

$$V_a = Ri_a + L\frac{di_a}{dt} + e_a \qquad (๑)$$

$$V_b = Ri_b + L\frac{di_b}{dt} + e_b \qquad (๒)$$

$$V_c = Ri_c + L\frac{di_c}{dt} + e_c \qquad (๓)$$



รูปที่ ๒.๓ วงจรสมมูลและแบบจำลองทางกลของ BLDC Motor

โดย L เป็น armature self-inductance [H]
R เป็น armature resistance [Ω]
V_a, V_b, V_c เป็น terminal voltage [V]
i_a, i_b, i_c เป็น motor input current [A]
e_a, e_b, e_c เป็น motor back EMF [V]

ทั้งนี้ Back EMF ที่เกิดขึ้นในแต่ละเฟสจะเป็นดังสมการ

$$e_a = K_e f(\theta_e)\omega \quad (๔)$$

$$e_b = K_e f(\theta_e - 2\pi/3)\omega \quad (๕)$$

$$e_c = K_e f(\theta_e + 2\pi/3)\omega \quad (๖)$$

$$\theta_e = \frac{P}{2}.\theta_m \quad (๗)$$

สมการแรงบิดทางไฟฟ้าเป็นดังสมการ

$$T_e = \frac{e_a i_a + e_b i_b + e_c i_c}{\omega} \quad (๘)$$

และสมการแรงบิดทางกลเป็นไปตามสมการ

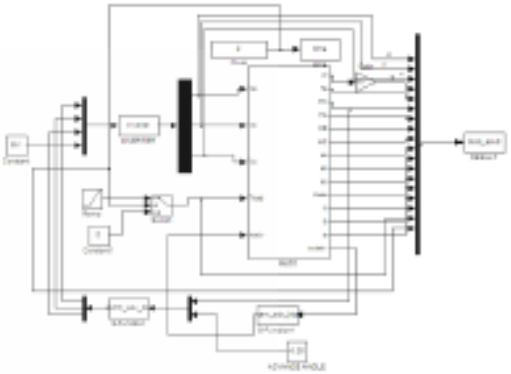
$$T_e - T_L = J\frac{d\omega}{dt} + B\omega \quad (๙)$$

โดย K_e เป็นค่าคงที่ [v/rad/s.]
θ_e เป็นมุมทางไฟฟ้าของโรเตอร์ [rad]
θ_m เป็นมุมทางกลของโรเตอร์ [rad]
ω เป็นความเร็วที่เพลามอเตอร์ [rad/s.]
P เป็น จำนวนขั้วแม่เหล็ก
T_e เป็น แรงบิดทางไฟฟ้า [N.m]
T_l เป็น แรงบิดทางกล [N.m]
J เป็นค่าโมเมนต์ความเฉื่อยที่เพลามอเตอร์
B เป็นค่าสัมประสิทธิ์ความหน่วงที่เพลา

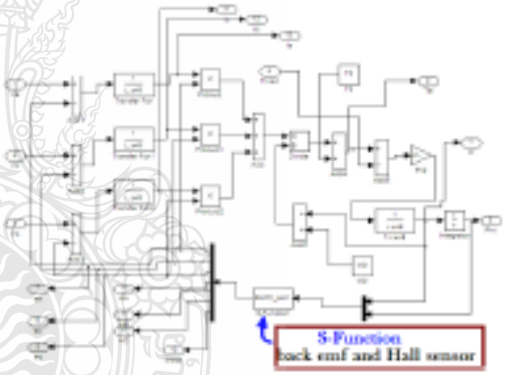
การจ่ายกระแสให้กับขดลวดสเตเตอร์ ในแต่ละไซเคิล จะถูกแบ่งออกเป็น ๖ step ดังแสดงในรูปที่ ๒.๒ ซึ่งแต่ละ step จะสัมพันธ์กับตำแหน่งโรเตอร์ตามลำดับ ของสัญญาณออกของ Hall sensor A B C ซึ่งฝังตัวอยู่ในสเตเตอร์ ผังเวลาการจ่ายกระแสให้กับขดลวดสเตเตอร์ทั้ง ๓ เฟสเขียนได้ดังสมการ [๓]

$$\left.\begin{array}{ll}
0^\circ < \theta_e \le 60^\circ & V_a = VB,\ V_b = -VB,\ V_c = 0 \\
60^\circ < \theta_e \le 120^\circ & V_a = VB,\ V_b = 0,\ V_c = -VB \\
120^\circ < \theta_e \le 180^\circ & V_a = 0,\ V_b = VB,\ V_c = -VB \\
180^\circ < \theta_e \le 240^\circ & V_a = -VB,\ V_b = VB,\ V_c = 0 \\
240^\circ < \theta_e \le 300^\circ & V_a = -VB,\ V_b = 0,\ V_c = VB \\
300^\circ < \theta_e \le 360^\circ & V_a = 0,\ V_b = -VB,\ V_c = VB
\end{array}\right\} \quad (10)$$

สมการ (๓) – (๙) ใช้เขียนเป็นระบบจำลองในบล็อกของ BLDC Motor ส่วนสมการ (๑๐) เป็น S-Function ในบล็อกของ Inverter ดังแสดงในรูปที่ ๒.๔ และ ๒.๕ ตามลำดับ



รูปที่ ๒.๔ ระบบจำลองการทำงานของ BLDC Motor



รูปที่ ๒.๕ โมเดลภายในบล็อก BLDC Motor

๒.๒ หลักการชิดเชยมุมเฟสล่วงหน้าใน BLDC Motor

เนื่องจากความเหนี่ยวนำในขดลวดมอเตอร์ทำให้กระแสเกิดการหน่วงตามค่าเวลา Time Constant และจะส่งผลให้กระแสในชุดขดลวดล้าหลังแรงดันด้านกลับ (Back EMF) ทั้งนี้จะแปรผันตามค่าความเร็วมอเตอร์ [๒] ค่ามุมต่างเฟสนี้จะส่งผลทางลบต่อค่าแรงบิดมอเตอร์ ในช่วงทำงานในย่านความเร็วสูงเกินพิกัด ดังนั้นการชุดให้กระแสมีมุมเฟสนำหน้าแรงดัน Back EMF ซึ่งเรียกหลักการนี้ว่า Phase advance angle เป็นวิธีหนึ่งที่ใช้ในการปรับปรุงค่าแรงบิดและควบคุมความเร็วให้กับมอเตอร์ในย่าน Constant power

๓.แบบจำลองและผลการจำลองระบบของการควบคุมมุมเฟสก้าวหน้า

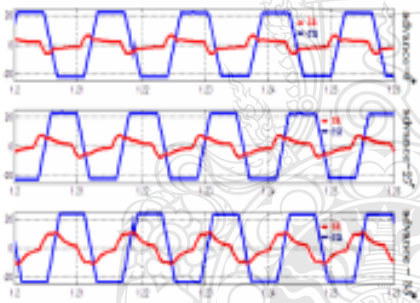
เพื่อให้เห็นสมรรถนะของการใช้หลักการ Phase Advance Angle ในการควบคุมแรงบิดและความเร็วทั้งสองย่านการทำงานของมอเตอร์ จึงแบ่งการทดสอบระบบจำลองเป็น ๒ รูปแบบดังนี้

๓.๑ การควบคุมความเร็วมอเตอร์แบบลูปเปิดที่มีการชิดเชยมุมเฟสก้าวหน้าต่างกัน

BLDC Motor ในระบบจำลองดังแสดงในรูปที่ ๒.๔ และ ๒.๕ มีพารามิเตอร์ที่สำคัญดังนี้

VB = ๒๔ V, R = ๑.๕ Ω , L = ๑.๘๖ mH,
J = ๑X๑๐$^{-๔}$ kgm^{๒}, B= ๑X๑๐$^{-๔}$, P = ๔,
Kw = ๐.๐๕๔๕ และ TL = ๐ – ๐.๕ Nm. ที่เวลา ๑ s.

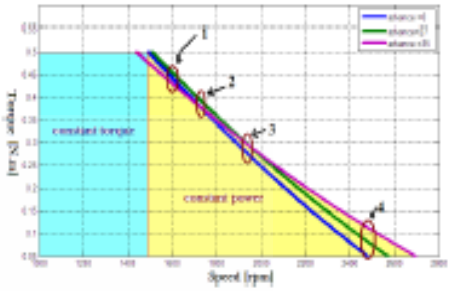
โดย มีการ ชิดเชยมุมเฟส ที่ ๐°, ๒๗° และ ๓๕° ตามลำดับ



รูปที่ ๓.๑ ผลจำลองของรูปคลื่นกระแสและแรงดันด้านกลับที่มีการชิดเชยมุมเฟสที่ ๐°, ๒๗° และ ๓๕° ตามลำดับ

Advance Angle Commutation(degree)	I_{peak} (Amp)
๐	๕
๒๗	๔
๓๕	๓.๑

ตาราง ๓.๑ ค่ากระแสสูงสุดที่จ่ายให้กับมอเตอร์ที่มีการชิดเชยมุมเฟสก้าวหน้า ๐ , ๒๗ และ ๓๕ องศาตามลำดับ



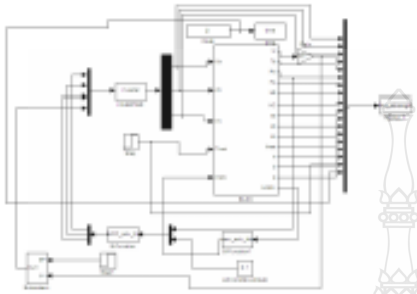
รูปที่๓.๒ ผลจำลองของแรงบิด-ความเร็วมอเตอร์ ที่มีการชิดเชยมุมเฟสที่ ๐°, ๒๗° และ ๓๕° ตามลำดับ

Speed	TORQUE		
	Advance Angle Commutation ๐	Advance Angle Commutation ๒๗	Advance Angle Commutation ๓๕
๑๖๐๐	๐.๑๔	๐.๑๖	๐.๑
๑๗๑๖	๐.๑๗	๐.๑	๐.๑๗
๑๗๕๕	๐.๑๓	๐.๑	๐.๑
๒๑๒๑	๐	๐.๐๘	๐.๑๓

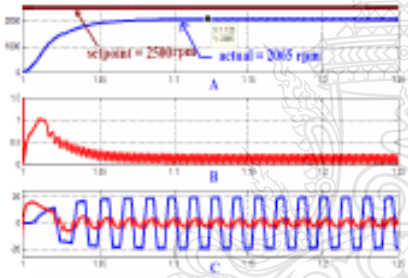
ตาราง ๓.๒ แสดงรายละเอียดความสัมพันธ์ระหว่างความเร็วและแรงบิดของมอเตอร์ที่มีการชิดเชยมุมเฟสที่ ๐°, ๒๗° และ ๓๕° ตามจุดที่ ๑-๔ ตามรูป ๓.๒

รูปที่ ๓.๑ แสดงผลการจำลอง ของรูปคลื่นแรงดัน และกระแสของ ระบบ ขณะ ที่มีการชิดเชยมุมเฟสก้าหน้า (Phase Advance Angle) ๐°, ๒๗° และ ๓๕° โดยให้ภาระโหลดของมอเตอร์เปลี่ยนแปลงจาก ๐ ถึง ๐.๕ Nm. ตามพิกัดของมอเตอร์ จะเห็นว่า กระแสมีนำหน้าแรงดันด้านกลับตามค่าของการชิดเชยมุมเฟสก้าวหน้าและค่าสูงสุดของกระแสที่จ่ายให้กับมอเตอร์ สูงขึ้นตามค่าของการชิดเชยมุมเฟสก้าวหน้า ดังแสดงในตารางที่ ๓.๑ ในขณะที่ความสัมพันธ์แรงบิด -ความเร็ว ขณะทำงานในย่าน Constant Power ค่าของการชิดเชยมุมเฟสก้าวหน้าจะเพิ่มขึ้นในลักษณะที่ไม่เป็นเชิงเส้น ดังแสดงในรูปที่ ๒.๓ และตาราง ๓.๒ ตามลำดับ จะเห็นได้ว่าจากจุดที่ ๒ ถึง จุดที่ ๓ ค่าของมุมเฟสก้าวหน้าควรจะเป็น ๒๗ องศาและจากจุดที่ ๓ ถึงจุดที่ ๔ ควรจะเป็น ๓๕ องศา เป็นต้น

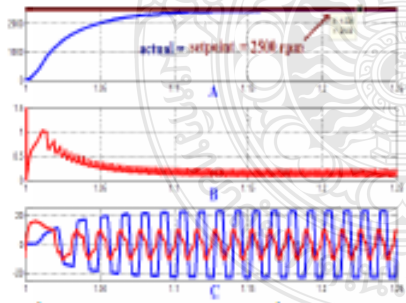
๓.๒ เปรียบเทียบการควบคุมความเร็วมอเตอร์แบบลูปปิด ที่มีและไม่มีการชิดเชยมุมเฟสล้ำหน้า



รูปที่ ๓.๓ Simulink Model การควบคุมมุมเฟสล้ำหน้าใน BLDCM แบบลูปปิด



รูปที่ ๓.๔ ความเร็ว แรงบิด และรูปคลื่นกระแส-แรงดัน BLDC Motor ขณะไม่มีการชิดเชยมุมเฟส



รูปที่ ๓.๕ ความเร็ว แรงบิด และรูปคลื่นกระแส-แรงดัน BLDC Motor ขณะมุมเฟสชิดเชยที่ ๓๕°

การจำลองในส่วนนี้เป็นการทำงานของ BLDC Motor ในย่านกำลังคงที่ ซึ่งมีความเร็วเกินพิกัด ทั้งนี้กำหนดให้โหลดทางกลมีค่าคงที่ ๐.๑ Nm.

PI Controller; K_p = ๐.๑, K_I = ๑๐๐

จากการจำลอง ขณะควบคุม ความเร็วของมอเตอร์แบบลูปปิด จะเห็นได้ว่าขณะไม่มีการชิดเชยมุมเฟส (๐°) มอเตอร์ไม่สามารถทำความเร็วเข้าหาความเร็วเป้าหมายได้ ดังแสดงในรูป ๓.๓[A] โดยค่าเฉลี่ยของ แรงบิดของมอเตอร์ที่ความเร็วคงที่ อยู่ที่ ๐.๒ Nmตามรูป ๓.๓[B] สำหรับส่วนของกระแสที่จ่ายให้กับมอเตอร์ จะเห็นได้ว่าอัตราการเพิ่มขึ้น (Rising)มีค่าน้อยเมื่อเทียบกับ รูปคลื่นของค่าของแรงดันด้านกลับ โดยค่ากระแสสูงสุดอยู่ที่ ๓ Amp ตามรูป ๓.๓[C]

เมื่อระบบมีการชิดเชยมุมเฟสที่ ๓๕° ขณะลูปปิดการควบคุมความเร็วทำให้อยู่เข้าสู่ค่าเป้าหมายได้ (ค่าผิดพลาดเป็นศูนย์)ตามรูป ๓.๕[A] โดยค่าแรงบิดเฉลี่ยอยู่ที่ ๐.๒ Nm ตามรูป ๓.๕[B] ในส่วนของกระแสที่จ่ายให้กับมอเตอร์จะเห็นได้ว่ามีอัตราการเพิ่มขึ้น(Rising)สูงกว่าเมื่อเทียบกับขณะที่ไม่มีการชิดเชยมุมเฟสล้ำหน้า ดังแสดงในรูป ๓.๕[C]

๔. สรุป

ผลที่ได้จากการทดสอบระบบจำลอง พบว่าการใช้หลักการชิดเชยมุมเฟสล้ำหน้าสามารถเพิ่มความเร็วและแรงบิดของมอเตอร์ในย่านกำลังคงที่ได้ดูได้จากผลการจำลองดังรูป ๓.๒ ซึ่งหลักการนี้สามารถนำไปประยุกต์ใช้เป็นแนวทางในการออกแบบชุดควบคุมการขับเคลื่อน BLDC Motor เพื่อให้ได้คุณสมบัติแรงบิด- ความเร็วที่เหมาะสม โดย เฉพาะในย่านความเร็วสูง ทั้งนี้สามารถนำไปประยุกต์ใช้งานเชิงปฏิบัติ ด้วยแผงวงจรประมวลสัญญาณดิจิตอล (DSP Board) เช่นกัน อย่างไรก็ตามบทความนี้ วิธีการ การชิดเชยมุมเฟสล้ำหน้า เป็นแบบคงที่ ที่ ๒๕ และ ๓๕ องศาตามลำดับ ตามรูป ๓.๒ จะเห็นได้ว่าในช่วงความเร็วจากตำแหน่งที่ ๑ – ๓ ควรใช้มุมเฟสล้ำหน้าที่ ๒๕ องศาและจากช่วงที่ ๓ – ๕ ควรใช้มุมเฟสล้ำหน้าที่ ๓๕ องศา เพื่อให้ได้แรงบิดมอเตอร์ที่สูงสุด ดังนั้นการปรับค่ามุมเฟสล้ำหน้าให้เหมาะสม ในแต่ละย่านความเร็ว ของมอเตอร์ ถือเป็นส่วน ที่สำคัญอย่างยิ่งที่นักวิจัยจะต้องทำการศึกษาต่อไป

๘. เอกสารอ้างอิง

[๑] B. Tibor, V. Fedak and F. Durovsky, "Modeling and simulation of the BLDC Motor in MATHLAB GUI", IEEE International Symposium on Industrial Electronics (ISIE) pp. ๑๔๐๓-๑๔๐๗, ๒๐๑๑

[๒] S.K. Safi ,P.P. Acarnley and A.G.Jack "Analysis and simulation of the high speed torque performance of Brushless DC Motor drives", IEE Proceedings on Electric Power Applications, Vol. ๑๔๒-๓, pp. ๑๙๑-๒๐๐, ๑๙๙๕.

[๓] I. Janpan, R. Chaisricharoen and P. Banyanant, "Control of the BLDC Motor in Combine Mode", Procedia Engineering ๓๒ (๒๐๑๒), pp. ๒๗๙-๒๘๕, Sci'Verse ScienceDirect, Elsevier Publication, ๒๐๑๒.

นายสำเริง เต็มราม ปัจจุบันเป็นนักศึกษาวิศวกรรมศาสตร์มหาบัณฑิต ภาควิชาวิศวกรรมไฟฟ้า มหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรี งานวิจัยที่สนใจระบบ Embedded Systems การประยุกต์ DSP Board ในการควบคุม BLDC Motor และระบบขับเคลื่อนในรถไฟฟ้าขนาดเล็ก

นายวันชัย ทรัพย์สิงห์ ปัจจุบันเป็นอาจารย์ประจำภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรี งานวิจัยที่สนใจ Power Electronics and Applications, Renewable Energy, FACTs และระบบขับเคลื่อนในรถไฟฟ้า


2013
EENET
5th ELECTRICAL ENGINEERING NETWORK
มหาวิทยาลัยเทคโนโลยีราชมงคลพระนคร
EENET
การประชุมวิชาการเครือข่ายวิศวกรรมไฟฟ้า
การประชุมวิชาการเครือข่ายวิศวกรรมไฟฟ้า
มหาวิทยาลัยเทคโนโลยีราชมงคล ครั้งที่ 5
บทความวิจัย
• ไฟฟ้ากำลัง
• อิเล็กทรอนิกส์กำลัง
• ไฟฟ้าสื่อสารและโทรคมนาคม
• ระบบควบคุมและการวัดคุม
• คอมพิวเตอร์และเทคโนโลยีสารสนเทศ
• พลังงานและการอนุรักษ์พลังงาน
• นวัตกรรมและสิ่งประดิษฐ์
• งานวิจัยที่เกี่ยวข้องกับวิศวกรรมไฟฟ้า
บทความวิชาการ
27-29 มีนาคม 2556 โรงแรมหัวหินแกรนด์ แอนด์ พลาซ่า จังหวัดประจวบคีรีขันธ์
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีราชมงคลพระนคร
EENET2013

คณะกรรมการอำนวยการจัดงาน EENET2013

1. รองศาสตราจารย์ดวงสุดา เตโชติรส อธิการบดีมหาวิทยาลัยเทคโนโลยีราชมงคลพระนคร	ประธานกรรมการ
2. ผู้ช่วยศาสตราจารย์สุภัทรา โกไศยกานนท์ รองอธิการบดีด้านวิชาการและวิเทศสัมพันธ์	กรรมการ
3. ผู้ช่วยศาสตราจารย์เฟื่องฟ้า เมฆเกรียงไกร รองอธิการบดีด้านวิจัยและบริการวิชาการ	กรรมการ
4. ผู้ช่วยศาสตราจารย์ฉัตรชัย เหียรหิรัญ รองอธิการบดีด้านเทคโนโลยีสารสนเทศและพัฒนากายภาพ	กรรมการ
5. ผู้ช่วยศาสตราจารย์ ดร.วัลลภ ภูผา คณบดีคณะวิศวกรรมศาสตร์	กรรมการและเลขานุการ

คณะกรรมการวิชาการเครือข่ายวิศวกรรมไฟฟ้า

1. รองศาสตราจารย์ ดร.โกศล โอฬารไพโรจน์ มหาวิทยาลัยเทคโนโลยีราชมงคล	ประธานกรรมการ
2. รองศาสตราจารย์ ดร.ธวัช เกิดชื่น มหาวิทยาลัยเทคโนโลยีราชมงคลอีสาน	กรรมการ
3. ผู้ช่วยศาสตราจารย์ ดร.เจนศักดิ์ เอกบูรณะวัฒน์ มหาวิทยาลัยเทคโนโลยีราชมงคลรัตนโกสินทร์	กรรมการ
4. ผู้ช่วยศาสตราจารย์วิสุทธิ์ พงศ์พฤกษธาตุ มหาวิทยาลัยเทคโนโลยีราชมงคลศรีวิชัย	กรรมการ
5. ผู้ช่วยศาสตราจารย์ ดร.ศักดิ์ระวี ระวีกุล มหาวิทยาลัยเทคโนโลยีราชมงคลอีสาน	กรรมการ
6. ผู้ช่วยศาสตราจารย์ ดร.กฤษณ์ชนม์ ภูมิกิตติพิชญ์ มหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรี	กรรมการ
7. ผู้ช่วยศาสตราจารย์ประสิทธิ์ นางทิน สถาบันเทคโนโลยีปทุมวัน	กรรมการ
8. อาจารย์ ดร.สุริยา แก้วอาษา มหาวิทยาลัยเทคโนโลยีราชมงคลอีสาน	กรรมการ
9. ผู้ช่วยศาสตราจารย์ ดร.อุเทน คำน่าน มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา	กรรมการ
10. อาจารย์ ดร.วุฒิวัฒน์ คงรัตนประเสริฐ มหาวิทยาลัยเทคโนโลยีราชมงคลกรุงเทพ	กรรมการ
11. อาจารย์ ดร.นัฐโชติ รักไทยเจริญชีพ มหาวิทยาลัยเทคโนโลยีราชมงคลพระนคร	กรรมการ
12. อาจารย์ประหยัด กองสุข มหาวิทยาลัยเทคโนโลยีราชมงคลตะวันออก	กรรมการ
13. ผู้ช่วยศาสตราจารย์ ดร.ประมุข อุณหเลขกะ มหาวิทยาลัยเทคโนโลยีราชมงคลสุวรรณภูมิ	กรรมการและเลขานุการ

คณะกรรมการดำเนินงานประชุมวิชาการ EENET2013 ประจำเครือข่าย

1. ผู้ช่วยศาสตราจารย์ ดร.วัลลภ ภูผา มหาวิทยาลัยเทคโนโลยีราชมงคลพระนคร	ประธานกรรมการ
2. อาจารย์ ดร.ณัฐพงศ์ พันธุนะ มหาวิทยาลัยเทคโนโลยีราชมงคลพระนคร	กรรมการ
3. ผู้ช่วยศาสตราจารย์วารุณี ศรีสงคราม มหาวิทยาลัยเทคโนโลยีราชมงคลสุวรรณภูมิ	กรรมการ
4. อาจารย์ ดร.นิธิโรจน์ พรสุวรรณเจริญ มหาวิทยาลัยเทคโนโลยีราชมงคลอีสาน	กรรมการ
5. อาจารย์ ดร.สายชล ชุดเจือจีน มหาวิทยาลัยเทคโนโลยีราชมงคลกรุงเทพ	กรรมการ
6. อาจารย์ ดร.ณฐภัทร พันธ์คง มหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรี	กรรมการ
7. อาจารย์ณรงค์ นันทกุศล มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา	กรรมการ
8. อาจารย์ ดร.วิวัฒน์ ทิพย์จร มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา	กรรมการ
9. อาจารย์พิทักษ์ บุญนุ่น มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา	กรรมการ
10. อาจารย์ณรงฤทธิ์ พิมพ์คำวงค์ มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา	กรรมการ
11. อาจารย์จิระยุทธ เนื่องรินทร์ มหาวิทยาลัยเทคโนโลยีราชมงคลอีสาน	กรรมการ
12. อาจารย์วิเชียร หทัยรัตน์ศิริ มหาวิทยาลัยเทคโนโลยีราชมงคลกรุงเทพ	กรรมการ
13. อาจารย์ภัทรพงศ์ อัญชันภาติ มหาวิทยาลัยเทคโนโลยีราชมงคลตะวันออก	กรรมการ

14. อาจารย์ ดร.ประสพโชค โห้ทองคำ มหาวิทยาลัยเทคโนโลยีราชมงคลรัตนโกสินทร์	กรรมการ
15. อาจารย์ปฏิวัติ บุญมา มหาวิทยาลัยเทคโนโลยีราชมงคลสุวรรณภูมิ	กรรมการ
16. อาจารย์ทศพร พรหมสิทธิ์ สถาบันเทคโนโลยีปทุมวัน	กรรมการ
17. อาจารย์ ดร.นัฐโชติ รักไทยเจริญชีพ มหาวิทยาลัยเทคโนโลยีราชมงคลพระนคร	กรรมการและเลขานุการ

คณะกรรมการดำเนินงานประชุมวิชาการ EENET2013 ด้านประชาสัมพันธ์

1. อาจารย์ ดร.ณัฐพงศ์ พันธุนะ	ประธานกรรมการ
2. อาจารย์ยุทธนา สรวลสรรค์	กรรมการ
3. ผู้ช่วยศาสตราจารย์สิทธิศักดิ์ วรดิษฐ์	กรรมการ
4. อาจารย์อภิชฎา ทองรักษ์	กรรมการ
5. อาจารย์ ดร.อดิศร ก้อนวัน	กรรมการ
6. นายสมยศ แสงจันทร์	กรรมการ
7. อาจารย์เกรียงไกร เหลืองอำพล	กรรมการและเลขานุการ

คณะกรรมการดำเนินงานประชุมวิชาการ EENET2013 ด้านบทความ

1. อาจารย์ ดร.นัฐโชติ รักไทยเจริญชีพ	ประธานกรรมการ
2. ผู้ช่วยศาสตราจารย์โกศล นิธิโสภา	กรรมการ
3. ผู้ช่วยศาสตราจารย์กมลทิพย์ วัฒกีกำธร	กรรมการ
4. อาจารย์ ดร.อดิศร ก้อนวัน	กรรมการ
5. อาจารย์เกรียงไกร เหลืองอำพล	กรรมการ
6. อาจารย์จตุรงค์ จตุรเชิดชัยสกุล	กรรมการ
7. อาจารย์พูนศรี วรรณการ	กรรมการ

8. อาจารย์พนา ดุสิตากร	กรรมการ
9. อาจารย์อดิศักดิ์ วิริยกรรม	กรรมการ
10. อาจารย์นิลมิต นิลาศ	กรรมการ
11. อาจารย์ธนะกิจ วัฒกีกำธร	กรรมการ
12. อาจารย์สาคร วุฒิพัฒนพันธุ์	กรรมการและเลขานุการ

คณะกรรมการดำเนินงานประชุมวิชาการ EENET2013 ด้านจัดทำเล่มบทความ

1. อาจารย์อรุณ ชลังสุทธิ์	ประธานกรรมการ
2. อาจารย์สมเกียรติ ทองแก้ว	กรรมการ
3. นายสมยศ แสงจันทร์	กรรมการ
4. อาจารย์มนัส บุญเทียรทอง	กรรมการ
5. อาจารย์เวทรินทร์ ธัญสิประเสริฐ	กรรมการ
6. อาจารย์ ดร.นัฐโชติ รักไทยเจริญชีพ	กรรมการและเลขานุการ

คณะกรรมการดำเนินงานประชุมวิชาการ EENET2013 ด้านสถานที่

1. อาจารย์ทง ลานธารทอง	ประธานกรรมการ
2. อาจารย์อดิศักดิ์ วิริยกรรม	กรรมการ
3. อาจารย์พูนศรี วรรณการ	กรรมการ
4. อาจารย์จตุรงค์ จตุรเชิดชัยสกุล	กรรมการ
5. อาจารย์สาคร วุฒิพัฒนพันธุ์	กรรมการ
6. อาจารย์พนา ดุสิตากร	กรรมการ
7. อาจารย์มนัส บุญเทียรทอง	กรรมการและเลขานุการ

คณะกรรมการดำเนินงานประชุมวิชาการ EENET2013 ด้านต้อนรับ

1. นางจุรีพร แสนสิ่ง	ประธานกรรมการ
2. นางอัจฉรา เหลือพงษ์	กรรมการ
3. อาจารย์อภิชฎา ทองรักษ์	กรรมการ
4. นางสาววรรณรัฐ ประจิมนอก	กรรมการ
5. นางนงเยาว์ เขียววิจิตร	กรรมการ
6. นางสาวนงนภัส สายแก้ว	กรรมการ
7. นางสาวสุภาภรณ์ ลาทุม	กรรมการและเลขานุการ

คณะกรรมการดำเนินงานประชุมวิชาการ EENET2013 ด้านงานทั่วไป

1. อาจารย์สมเกียรติ ทองแก้ว	ประธานกรรมการ
2. นายสมยศ แสงจันทร์	กรรมการ
3. นางสาวศิริกุล ภู่จำนงค์	กรรมการ
4. นายไพฑูรย์ อาราเบีย	กรรมการ
5. นายกิตติศักดิ์ กิ่งแก้ว	กรรมการ
6. นายปิยะพันธ์ บำเรอจิตร	กรรมการ
7. นางกัญญ์ชิสา ธัญสิประเสริฐ	กรรมการและเลขานุการ

คณะกรรมการดำเนินงานประชุมวิชาการ EENET2013 ด้านทะเบียน

1. ผู้ช่วยศาสตราจารย์ ดร.สมใจ เพียรประสิทธิ์	ประธานกรรมการ
2. นางวิไลวรรณ คำปาน	กรรมการ
3. นางนวพร พานอุป	กรรมการ
4. นางสาวเนตรนภา แสงเงิน	กรรมการ
5. นางสมศรี จันทร์ชนะ	กรรมการ
6. นางสาวแสงเดือน พุทธัง	กรรมการและเลขานุการ

คณะกรรมการดำเนินงานประชุมวิชาการ EENET2013 ด้านติดตามและประเมินผล

1. นางรัตนา เหล่าสินชัย	ประธานกรรมการ
2. นางกุสุมา ระงับภัย	กรรมการและเลขานุการ

สารบัญ

สาขาบทความงานวิจัยที่เกี่ยวข้องกับวิศวกรรมไฟฟ้า

สาขาบทความงานวิจัยที่เกี่ยวข้องกับวิศวกรรมไฟฟ้า

บทความวิจัย

การประชุมวิชาการเครือข่ายวิศวกรรมไฟฟ้า มหาวิทยาลัยเทคโนโลยีราชมงคล ครั้งที่ 5

Proceedings of The 5th Conference of Electrical Engineering Network of Rajamangala University of Technology (EENET 2013)

ระบบจำลองการชดเชยมุมเฟสก้าวหน้าในมอเตอร์ไฟตรงไร้แปรงถ่านด้วยโปรแกรม MATLAB/Simulink

Modeling of Phase Advance Angle Commutation in Brushless DC Motor using MATLAB/Simulink

สำเริง เต็มราม[1] และวันชัย ทรัพย์สิงห์[2]

[1]ภาควิชาไฟฟ้ากำลัง สำนักพัฒนาสมรรถนะครูและบุคลากรอาชีวศึกษา

ถนนรามอินทรา ก.ม. 5-6 แขวงท่าแร้ง เขตบางเขน กรุงเทพมหานคร โทรศัพท์ : 02-5093654-5 ต่อ 431 E-mail:elect_2510@hotmail.com

[2]ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรี

39 หมู่ 1 ถนนรังสิต-นครนายก ต.คลองหลวง อ.ธัญบุรี จ.ปทุมธานี 12110 โทรศัพท์ : 025-493420 E-mail: wanchai.s@en.rmutt.ac.th

บทคัดย่อ

บทความนี้นำเสนอการศึกษาและพัฒนาแบบจำลองทางคณิตศาสตร์ ด้วยโปรแกรม MATLAB/Simulink เพื่อวิเคราะห์สมรรถนะการทำงานของ มอเตอร์ไฟตรงไร้แปรงถ่าน (Brushless DC Motor) โดยพิจารณาในส่วนการชดเชยมุมเฟส แบบก้าวหน้า (Phase Advance Angle Compensation) ระหว่างกระแสขาเข้ากับแรงดัน Back EMF ในแต่ละเฟสของมอเตอร์ ในย่านการทำงานแบบ กำลังคงที่ (Constant Power Region) ที่ส่งผลตอบสนองโดยตรงต่อค่าแรงบิดและความเร็วขณะมอเตอร์ทำงานที่ความเร็วเกิ นพิกัด ซึ่งเป็นองค์ประกอบที่สำคัญต่อการออกแบบชุดควบคุมการขับเคลื่อนมอเตอร์ไฟตรงไร้แปรงถ่านนี้ ทั้งนี้ในบทความจะศึกษาผลการตอบสนองทางด้านแรงบิดและความเร็วของมอเตอร์ ในระบบวงเปิด ทั้งในขณะไม่มีการชดเชยมุมเฟส และขณะมีการชดเชยมุมเฟส เพื่อนำผลที่ได้ไปวิเคราะห์แ ละสร้างชุดควบคุมการขับเคลื่อนมอเตอร์ดังกล่าวให้เหมาะสมกับการนำไปใช้งานต่อไป

คำสำคัญ: การชดเชยมุมเฟสก้าวหน้า มอเตอร์ไฟตรงไร้แปรงถ่าน

Abstract

This paper presents the investigation and the development of a mathematical model of the Brushless Direct Current (BLDC) Motor using MATLAB / Simulink program. It concerns in Performance analysis of the BLDC Motor when it operates in a constant power region. This leads into an implemented of Phase Advanced Angle Compensation (PAAC) method in order to improve the torque/ speed characteristic when running in an over-rated speed range. Hence, this paper considers in studying of motor performances both without PAAC and without PAAC. Furthermore, it may useful for the implementation of the Phase Advance Angle Compensation method with the BLDC Motor in practice.

Keywords: Phase Advance Angle Commutation, Brushless DC Motor

1. บทนำ

ในปัจจุบันมอเตอร์ไฟตรงไร้แปรงถ่าน (BLDCM) กำลังเป็นที่สนใจในงานอุตสาหกรรมและมีการนำไปประยุกต์ใช้งานต่างๆหลายด้านเช่น ในอุตสาหกรรมการบินและอวกาศ อุตสาหกรรมการแพทย์ ระบบควบคุมอัตโนมัติ และ อุตสาหกรรมยานยนต์ เป็นต้น โดยเฉพาะเป็นตัวขับในล้อ (In-Wheel Motor) ของรถยนต์ไฟฟ้าขนาดเล็กรุ่นใหม่ ซึ่งเป็นที่สนใจในปัจจุบัน คุณสมบัติของมอเตอร์ไฟตรงไร้แปรงถ่านนี้มีจุดเด่นหลายด้านเมื่อเทียบกับมอเตอร์ไฟฟ้าแบบเหนี่ยวนำ [1] เช่น

- คุณลักษณะของความเร็ว-แรงบิดดีกว่า
- ตอบสนองต่อการเปลี่ยนแปลงความเร็วได้ดี
- มีประสิทธิภาพในการทำงานสูง
- อายุการใช้งานยาวนาน
- ไม่มีสัญญาณเสียงรบกวนขณะทำงาน
- ตอบสนองการทำงานในย่านความเร็วสูงได้ดี
- มีค่าแรงบิดสูงกว่าเมื่อเทียบกับขนาดของมอเตอร์

นอกจากนี้ยังมีขนาดน้ำหนักต่อพิกัดกำลัง (Weight / Watt) อยู่ในเกณฑ์ต่ำ สามารถนำไปใช้ในส่วนที่ถูกจำกัดพื้นที่ได้ดีกว่า

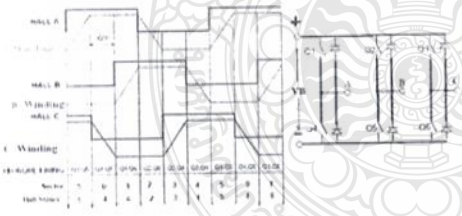
แรงบิดของ BLDC Motor ขึ้นอยู่กับรูปคลื่น Back EMF ในขดลวดสเตเตอร์ [1] ซึ่งโดยทั่วไปจะเป็นรูปคลื่นสี่เหลี่ยมคางหมู และกระแสจะเป็นรูปคลื่นสามเหลี่ยม ขณะทำงานในย่านแรงบิดคงที่ ส่วนในย่านกำลังไฟฟ้าคงที่ มอเตอร์จะทำงานที่ความเร็วสูง ขึ้นในขณะที่แรงบิดลดลง จึงจำเป็นต้องควบคุมให้แรงบิดมีค่าเหมาะสมต่อการใช้งาน เนื่องจากไม่สามารถควบคุมกระแสขดลวดสเนามได้เช่นเดียวกับมอเตอร์ไฟฟ้ากระแสตรงทั่วไป อย่างไรตามผลตอบสนองของแรงบิดและกระแสจากการจำลองระบบจะต่างจากผลทดลองที่ได้ทางปฏิบัติบ้าง เนื่องจากในทางปฏิบัติแรงบิดและกระแสจะมี ripple เกิดขึ้น แต่การจำลองระบบมีข้อดี สำหรับใช้ในการวิเคราะห์เพื่อควบคุมการทำงานระบบในอนาคต

2. ทฤษฎีพื้นฐาน

BLDC Motor อยู่ในกลุ่มของ Synchronous Motor ซึ่งความถี่ที่โรเตอร์จะเท่ากับที่สเตเตอร์ จึงไม่มีค่าสลิปเหมือนในมอเตอร์เหนี่ยวนำ เนื่องจากใช้แม่เหล็กถาวรแทนขดลวดสนาม และควบคุมการไหลของกระแสสเตเตอร์ด้วยตำแหน่งของโรเตอร์ BLDC Motor ในปัจจุบันมีทั้ง 1 เฟส 2 เฟสและ 3 เฟส ขึ้นอยู่กับจำนวนชุดขดลวดสเตเตอร์ แต่ที่นิยมใช้ทั่วไปเป็นแบบ 3 เฟส

ตำแหน่งของโรเตอร์ที่ใช้ในการควบคุมการไหลของกระแสสเตเตอร์ ปัจจุบันนิยมใช้ Hall sensor ในการตรวจจับการเปลี่ยนแปลงสนามแม่เหล็ก และการตรวจจับชุดคัดแรงดันด้านกลับ (Back EMF Zero Crossing Detection) ทั้งนี้เพื่อควบคุมให้แรงปฏิกิริยาจากสนามแม่เหล็กระหว่างโรเตอร์และสเตเตอร์ เกิดขึ้นที่มุมที่เหมาะสม รูปที่ 1.1 และ 1.2 แสดงหลักการควบคุมการไหลของกระแสสเตเตอร์ด้วยตำแหน่งของโรเตอร์โดยใช้ Hall sensor ตรวจจับการเปลี่ยนแปลงของสนามแม่เหล็ก

รูปที่ 1.1 การควบคุมการไหลของกระแสสเตเตอร์



รูปที่ 1.2 การควบคุมสวิตช์อิเล็กทรอนิกส์ให้สอดคล้องตามรูปที่ 1.1

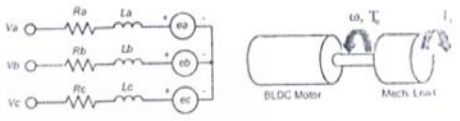
2.1 แบบจำลองทางคณิตศาสตร์ของ BLDC Motor

วงจรสมมูลของ BLDC Motor [2]แสดงดังรูปที่ 2.1 ซึ่งสามารถเขียนสมการทางคณิตศาสตร์ได้(1-3)

$$V_a = Ri_a + L\frac{di_a}{dt} + e_a \quad (1)$$

$$V_b = Ri_b + L\frac{di_b}{dt} + e_b \quad (2)$$

$$V_c = Ri_c + L\frac{di_c}{dt} + e_c \quad (3)$$



รูป 2.1 วงจรสมมูลและแบบจำลองทางกลของ BLDCM

โดย L เป็น armature self-inductance [H]
R เป็น armature resistance [Ω]
V_a, V_b, V_c เป็น terminal voltage [V]
i_a, i_b, i_c เป็น motor input current [A]
e_a, e_b, e_c เป็น motor back EMF [V]

แรงดัน Back EMF ที่เกิดขึ้นในแต่ละเฟสจะเป็นดังสมการ(4-6)

$$e_a = K_w f(\theta_e)\omega \quad (4)$$

$$e_b = K_w f(\theta_e - 2\pi/3)\omega \quad (5)$$

$$e_c = K_w f(\theta_e + 2\pi/3)\omega \quad (6)$$

องศาทางไฟฟ้าเทียบกับองศาทางกลเขียนได้ดังสมการ(7)

$$\theta_e = \frac{P}{2}.\theta_m \quad (7)$$

ในส่วนสมการแรงบิดทางไฟฟ้าและแรงบิดทางกลเป็นไปตามสมการ(8-9)

$$T_e = \frac{e_a i_a + e_b i_b + e_c i_c}{\omega} \quad (8)$$

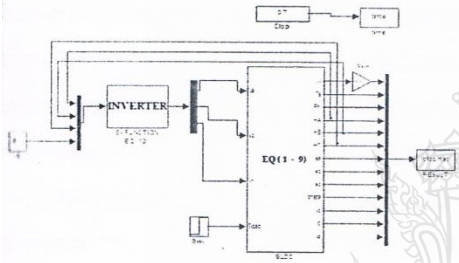
$$T_e - T_L = J\frac{d\omega}{dt} + B\omega \quad (9)$$

โดย K_w เป็นค่าคงที่ [v/rad/s.]
θ_e เป็นมุมทางไฟฟ้าของโรเตอร์ [°e]
θ_m เป็นมุมทางกลของโรเตอร์ [°]
ω เป็นความเร็วที่เพลามอเตอร์ [rad/s.]
P เป็น จำนวนขั้วแม่เหล็ก
T_e เป็น แรงบิดสนามแม่เหล็ก [N.m]
J เป็นค่าโมเมนต์ความเฉื่อยที่เพลามอเตอร์
B เป็นค่าสัมประสิทธิ์ความหน่วงที่เพลา

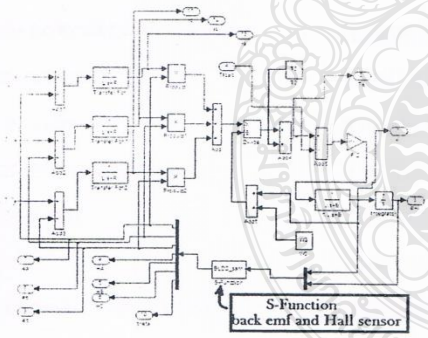
การจ่ายกระแสให้กับขดลวดสเตเตอร์ในแต่ละไซเคิล จะถูกแบ่งออกเป็น 6 step ดังแสดงในรูปที่ 1.2 ซึ่ง ในแต่ละ step จะสัมพันธ์กับกับตำแหน่งของ rotor ตามลำดับสัญญาณลอจิกของ Hall sensor A B C ซึ่งฝังตัวอยู่ในสเตเตอร์ ตามลำดับ[3] ทั้งนี้ผังเวลาการจ่ายกระแสให้กับขดลวดสเตเตอร์ทั้งสามเฟสเขียนได้ดังสมการ(10)

$$\left.\begin{array}{ll}
0^\circ < \theta_e \le 60^\circ & V_a = VB,\ V_b = -VB,\ V_c = 0 \\
60^\circ < \theta_e \le 120^\circ & V_a = VB,\ V_b = 0,\ V_c = -VB \\
120^\circ < \theta_e \le 180^\circ & V_a = 0,\ V_b = VB,\ V_c = -VB \\
180^\circ < \theta_e \le 240^\circ & V_a = -VB,\ V_b = VB,\ V_c = 0 \\
240^\circ < \theta_e \le 300^\circ & V_a = -VB,\ V_b = 0,\ V_c = VB \\
300^\circ < \theta_e \le 360^\circ & V_a = 0,\ V_b = -VB,\ V_c = VB
\end{array}\right\} \quad (10)$$

สมการ (1) – (9) เขียนเป็นระบบจำลองในบล็อก BLDCM ส่วนสมการ (10) เป็น S-Function ในบล็อก Inverter ดังแสดงในรูปที่ 2.2 และ 2.3 ตามลำดับ



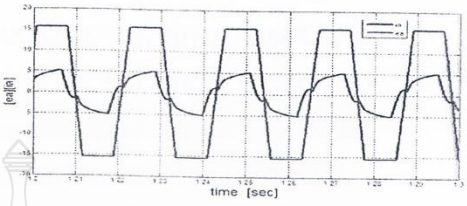
รูปที่ 2.2 ระบบจำลองการทำงานทั้งหมดของ BLDC Motor



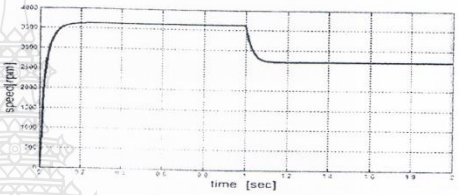
รูปที่ 2.3 โมเดลภายในบล็อก BLDCM

2.2 ผลการจำลองขณะไม่มีการชดเชยมุมเฟสก้าวหน้า

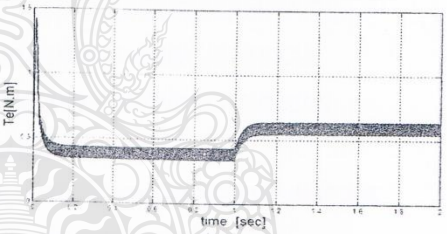
BLDC Motor ที่ใช้ในการจำลองระบบมีพารามิเตอร์ที่สำคัญดังนี้ VB = 24 V, R = 1.5 Ω, L = 5.72 mH, J = 2×10^{-4} kgm^2, B= 2×10^{-3}, P = 4, Kw = 0.0545 และ TL = 0.3 Nm. ที่เวลา 1 s.



รูปที่ 2.4 รูปคลื่นกระแสและ Back EMF ขณะไม่มีการชดเชยมุมเฟสก้าวหน้า



รูปที่ 2.5 กราฟความเร็วมอเตอร์ขณะไม่มีการชดเชยมุมเฟสก้าวหน้า



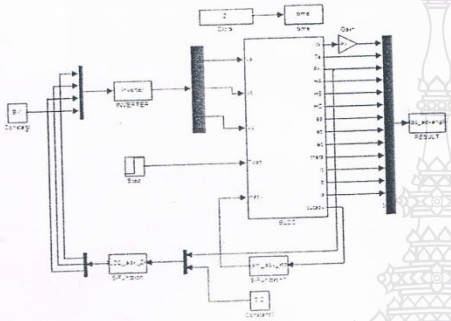
รูปที่ 2.6 กราฟแรงบิดมอเตอร์ขณะไม่มีการชดเชยมุมเฟสก้าวหน้า

การจำลองนี้กระทำขณะมอเตอร์ถูกควบคุมแบบ Open Loop ขณะทำงานในย่านความเร็วสูงกว่าพิกัด ซึ่ง ขณะไม่มีการชดเชยมุมเฟสก้าวหน้า (มุมนำกระแสนำหน้าเป็นศูนย์) ผลการจำลองจะเป็นดังแสดงในรูปที่ 2.4 – 2.6 ซึ่งจะเห็นว่ารูปคลื่นกระแสจะมีการเปลี่ยนแปลงช้ากว่าของแรงดันด้านกลับ (ดูจากจุดสูงสุดของแรงดันด้านกลับเทียบกับของกระแส) ความเร็วจะลดลงและแรงบิดจะเพิ่มขึ้นเมื่อเพิ่มโหลดที่ 1 sec.

2.3 หลักการชดเชยมุมเฟสก้าวหน้า

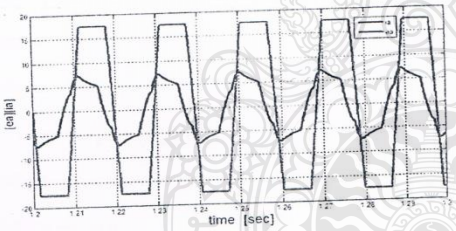
เนื่องจากความเหนี่ยวนำในขดลวดมอเตอร์ทำให้เกิดการหน่วงของกระแสตามเวลาค่าคงตัว (Time Constant) และจะส่งผลให้เกิดมุมต่างเฟสระหว่างกระแสและแรงดัน Back EMF ซึ่งจะแปรผันตามค่าความเร็วมอเตอร์ [2] ค่ามุมต่างเฟสนี้จะส่งผลทางลบต่อค่าแรงบิดมอเตอร์ด้วย ดังนั้นการจุดให้กระแสมีมุมเฟสนำหน้าแรงดัน Back EMF ซึ่งเรียก

หลักการนี้ว่า Phase advance angle เป็นวิธีหนึ่งในการปรับปรุงค่าแรงบิดของมอเตอร์ขณะทำงานที่ความเร็วสูง แบบจำลองของหลักการชดเชยมุมเฟสก้าวหน้า (Phase Advance Angle Commutation) แสดงดังรูปที่ 2.7

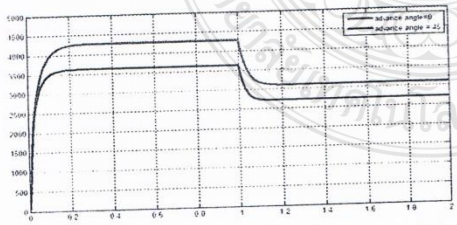


รูป 2.7 แสดง Simulink Model ตามหลักการชดเชยมุมเฟสก้าวหน้า

2.4 ผลการจำลองขณะมีการชดเชยมุมเฟสก้าวหน้า

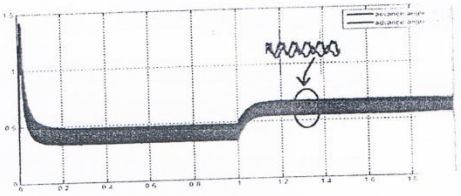


รูปที่ 2.8 รูปคลื่นกระแสและ Back EMF ที่มีการชดเชยมุมเฟสก้าวหน้า 42 องศา



รูปที่ 2.9 กราฟความเร็วมอเตอร์ที่มีและไม่มีการชดเชยมุมเฟสก้าวหน้า

จากผลการจำลองตามรูป 2.8 – รูปที่ 2.10 ขณะมีการชดเชยมุมเฟสก้าวหน้า จะเห็นว่ากระแสมีรูปคลื่น นำหน้าแรงดัน Back EMF รวมทั้งรูปคลื่นกระแสมีความหน่วงน้อยลง



รูปที่ 2.10 กราฟแรงบิดมอเตอร์ที่มีและไม่มีการชดเชยมุมเฟสก้าวหน้า

ซึ่งขณะมอเตอร์ได้รับ Load Torque 0.3 Nm ที่เวลา 1 sec นั้น ความเร็วมอเตอร์ลดลงและแรงบิดขับของมอเตอร์เพิ่มขึ้น เช่นเดียวกับขณะไม่มีการชดเชยมุมเฟส จากการพิจารณาจะเห็นว่าค่าแรงบิดขับของมอเตอร์สูงกว่าขณะไม่มีการชดเชยมุมเฟส และหลังจากมีการชดเชยมุมเฟส จะทำให้ความเร็วของมอเตอร์สูงขึ้นเช่นกัน ทั้งในขณะมอเตอร์ไม่มีโหลดและขณะมีโหลด

3. สรุป

ผลที่ได้จากการทดสอบระบบจำลอง พบว่า ที่มีการชดเชยมุมเฟสก้าวหน้า 42 องศา ทำให้ความเร็วและค่าเฉลี่ยของแรงบิดมอเตอร์ดีขึ้น ดังแสดงในรูป 2.9 และ 2.10 ตามลำดับ ผลที่ได้จากการจำลองดังกล่าวสามารถนำไปใช้เป็นแนวทางในการออกแบบชุดควบคุมการขับเคลื่อน BLDC Motor เพื่อให้ได้คุณสมบัติแรงบิด-ความเร็วที่เหมาะสม โดยจะต้องนำไปประยุกต์ใช้งานเชิงปฏิบัติด้วยแผงวงจรประมวลสัญญาณดิจิตอล (DSP Board) ต่อไปเช่นกัน รวมทั้งให้ความสนใจต่อการควบคุมแบบระบบปิด (Closed Loop System) อีกด้วย

เอกสารอ้างอิง

[1] B. Tibor, V. Fedak and F. Durovsky, "Modeling And simulation of the BLDC Motor in MATHLAB GUI", 2011

[2] S.K. Safi ,P.P. Acarnley and A.G.Jack "Analysis and simulation of the high speed torque performance of Brushless DC Motor" May 1995

[3] I. Janpan, R. Chaisricharoen and P. Banyanant, "Control of the BLDC Motor in Combine Mode", Procedia Engineering 32 (2012) pp. 279-285, Sci'Verse ScienceDirect, Elsevier Publication. 2012

นายสำเริง เต็มราม ปัจจุบันเป็นนักศึกษาวิศวกรรมศาสตร์มหาบัณฑิต ภาควิชาวิศวกรรมไฟฟ้า มหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรี งานวิจัยที่สนใจ ระบบฝังตัวกลฝังตัว (Embedded System) การประยุกต์ DSP Board ในการควบคุม BLDC Motor และระบบขับเคลื่อนในรถไฟฟ้าขนาดเล็ก

ประวัติผู้เขียน

ชื่อ – นามสกุล	นายสำเริง เต็มราม
วัน เดือน ปีเกิด	24 ตุลาคม 2510
ที่อยู่	บ้านเลขที่ 3 ตำบลวังพัฒนา อำเภอบางซ้าย
	จังหวัด พระนครศรีอยุธยา
การศึกษา	
พ.ศ. 2540	สำเร็จการศึกษาระดับวิศวกรรมศาสตร์บัณฑิต
	สาขาวิศวกรรมไฟฟ้า มหาวิทยาลัยเทคโนโลยีราชมงคลกรุงเทพ
ประสบการณ์การทำงาน	
2540 – ปัจจุบัน	ตำแหน่งนักทรัพยากรบุคคลชำนาญการ
	สำนักพัฒนาสมรรถนะครูและบุคลากรอาชีวศึกษา