การออกแบบตัวควบคุมพีไอโดยวิธีการหาค่าเหมาะสมที่สุดสำหรับเงื่อนไขไม่คอนแวกต์

DESIGN OF PI CONTROLLER BASED ON NON - CONVEX OPTIMIZATION

โดย

นาย พิศาล ชาวเสมา

นาย รัตนะ พลชนะ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมระบบควบคุม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2551

ปริญญานิพนธ์ปีการศึกษา 2551

ภาควิชาวิศวกรรมระบบควบคุม คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การออกแบบตัวควบคุมพีไอโดยวิธีการหาค่าเหมาะสมที่สุดสำหรับเงื่อนไขไม่คอนแวกต์

DESIGN OF PI CONTROLLER BASED ON NON - CONVEX OPTIMIZATION

ผู้จัดทำ นาย พิศาล ชาวเสมา 49015339

นาย รัตนะ พลชนะ 49015347

สังวาล บกสุวรรณ์ .....อาจารย์ที่ปรึกษา

(อาจารย์ สังวาล บกสุวรรณ)

การออกแบบตัวควบคุมพีไอโดยวิธีการหาค่าเหมาะสมที่สุดสำหรับเงื่อนไขไม่คอนแวกต์

โดย

นาย พิศาล ชาวเสมา 49015339

นาย รัตนะ พลชนะ 49015347

อาจารย์ที่ปรึกษา

อาจารย์ สังวาล บุกสุวรรณ

ปีการศึกษา 2550

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ นำเสนอทฤษฎีและการออกแบบตัวควบคุมพีไอโดยวิธีการหาค่าเหมาะสมที่สุดสำหรับเงื่อนไขไม่คอนแวกต์ โดยโครงสร้างของระบบประกอบด้วย รอก เกียร์ทด มอเตอร์กระแสตรง และวงจรอิเล็กทรอนิกส์ที่เกี่ยวข้อง จุดมุ่งหมายของโครงงานนี้คือการออกแบบค่าพารามิเตอร์ของตัวควบคุมแบบ พีไอ เพื่อควบคุมระบบที่สร้างขึ้นมาเพื่อทดลอง

ขั้นตอนดำเนินการ เริ่มจากศึกษาทฤษฎีและเขียนโปรแกรมคอมพิวเตอร์โดยใช้โปรแกรมคำนวณค่าทางคณิตศาสตร์เพื่อหาค่าพารามิเตอร์ของตัวควบคุม จากนั้นทำการประกอบโครงสร้างและเขียนโปรแกรมคอมพิวเตอร์โดยใช้ภาษาซีเพื่อควบคุมระบบว่าเป็นไปตามทฤษฎีหรือไม่ จากผลการทดลองพบว่าระบบควบคุมที่ได้ออกแบบนั้น สามารถควบคุมระบบได้ โดยมีค่าความผิดพลาดอยู่ในระดับที่ยอมรับได้

DESIGN OF PI CONTROLLER BASED ON NON - CONVEX OPTIMIZATION

By

Mr. Pisal Chawsema 49015339

Mr. Rattana Polchana 49015347

Advisor

Mr. Sungvan Bogsuwan

Academic Year 2009

ABSTRACT

This thesis represents theories and design of PI controller based on non – convex optimization. The system composes of structure, pulley, gears carries, DC motor and interfacing circuit. The goal of this project designs parameter value of PI controller. for control the system that establish to come to for experience

The project has been conducted as in the following step. First, studied the theory and a computer program written in MATLAB for seek parameter value of the controller. Second, the structure of control system is designed and a computer program written in software C is composed. Be in line with or not is the theory. Lastly, the experiments are conducted. The results show can be controlled to the set point with negligible error.

กิตติกรรมประกาศ

การจัดทำปริญญานิพนธ์ฉบับนี้ สามารถสำเร็จลุล่วงไปได้ด้วยดี เพราะได้รับความช่วยเหลือเป็นอย่างดี จาก อาจารย์สังวาล บกสุวรรณ ที่ได้กรุณาให้คำปรึกษาแนะนำที่ดีมาโดยตลอดตั้งแต่ในช่วงปิดเทอม รวมทั้งแนะนำหนังสือและสิ่งพิมพ์ที่มีประโยชน์ต่อโครงงานและความช่วยเหลืออื่นๆที่เป็นประโยชน์ต่อโครงงาน คณะผู้จัดทำรู้สึกซาบซึ้งและขอกราบขอบพระคุณเป็นอย่างสูง

ขอกราบขอบพระคุณเป็นอย่างสูง รศ.ดร.โยธิน เปรมปราณีรัชต์ , รศ.ดร.วรพงศ์ ตั้งศรีรัตน์ ผศ.ดร.ถาวร เบญจนราสุทธิ์ , อ.เทพจิตร์ เชยโภคา ที่ให้คำปรึกษา เสนอแนะและให้หยิบยืมอุปกรณ์ในการทำโครงงานและพี่เพชร พี่น้อย ที่ให้หยิบยืมเครื่องมือต่างๆในการทำโครงงาน

ขอบคุณเพื่อนๆภาคระบบควบคุมที่ให้คำปรึกษาและช่วยเสนอแนะข้อมูลในการทำงานและให้ยืมอุปกรณ์ที่ทางเราขาดหายไป ในการทำงาน ไม่อาจกล่าวขอบคุณเป็นรายบุคลได้ต้องขออภัยอย่างสูง

สุดท้ายนี้คณะผู้จัดทำขอกราบขอบพระคุณบิดา มารดา และครอบครัวที่คอยให้กำลังใจที่ดีเสมอมาและให้เงินสนับสนุนในเรื่องของงบประมาณที่ขาดเหลือ พี่วรรณที่คอยถามและให้กำลังใจที่ดีเสมอมา น้องสาวที่คอยถามความคืบหน้าในการทำงานและสุดท้ายขอบคุณตัวเองที่สู้และอดทนในการทำโครงงานจนเสร็จ

ผู้จัดทำ

นาย พิศาล ชาวเสมา

นาย รัตนะ พลชนะ

สารบัญ

สารบัญ(ต่อ)

สารบัญ(ต่อ)

สารบัญ(ต่อ)

หน้า

สารบัญรูป

สารบัญรูป(ต่อ)

สารบัญรูป(ต่อ)

บทที่ 1

บทนำ

1.1 กล่าวนำ

ในการออกแบบตัวควบคุมนั้นเป้าหมายสูงสุดคือการสร้างตัวควบคุมที่สามารถควบคุมระบบให้ทำงานได้อย่างมีประสิทธิภาพและเหมาะสม และสามารถทำงานได้ในสภาวะแวดล้อมจริงเพราะว่าสิ่งแวดล้อมที่รอบตัวเรานั้นมีองค์ประกอบหลายอย่างที่เป็นสาเหตุให้ระบบมีการเปลี่ยนแปลง เช่น อายุการใช้งานของอุปกรณ์และเครื่องมือต่างๆซึ่งเมื่อใช้ไปนานๆ แล้วประสิทธิภาพในการทำงานจะลดน้อยลง อุณหภูมิ สภาวะแวดล้อมเปลี่ยนแปลงไป มีการรบกวนจากภายนอก (Disturbance) ซึ่งเป็นสาเหตุให้การทำงานของระบบมีประสิทธิภาพลดลง คุณสมบัติของตัวควบคุมที่ดีนั้นจะต้องสามารถควบคุมให้ระบบควบคุมมีผลตอบสนองตามที่ได้ออกแบบไว้ มีเสถียรภาพและสามารถควบคุมระบบที่มีพารามิเตอร์ของกระบวนการ (Process) เปลี่ยนแปลงภายใต้ขอบเขต ที่กำหนดไว้ล่วงหน้าได้อย่างเหมาะสม คุณสมบัตินี้เรียกว่า คุณสมบัติความคงทน (Robustness) ของระบบควบคุม

สำหรับวิธีการที่จะใช้การออกแบบตัวควบคุมที่ดีนั้นนอกจากจะต้องเป็นวิธีที่สามารถทำให้ระบบควบคุมมีสมรรถนะที่ดีคือ สามารถควบคุมให้ระบบควบคุม มีผลตอบสนองตามที่ได้ออกแบบไว้ มีเสถียรภาพ และมีความคงทนต่อการเปลี่ยนแปลงพารามิเตอร์ของกระบวนการแล้ว วิธีการที่ดีที่จะนำมาใช้ในการออกแบบตัวควบคุมจะต้องเป็นวิธีการที่สามารถออกแบบได้ง่าย เช่น Karl J. Åström และ Sigurd Skogestad

1.2 วัตถุประสงค์ในการทำปริญญานิพนธ์

1. เพื่อนำเสนอวิธีการออกแบบตัวควบคุม พีไอ ซึ่งเป็นวิธีที่ง่าย และสามรถออกแบบได้อย่างรวดเร็ว แทนที่วิธีการเดิมที่มีอยู่

2. ระบบควบคุมที่ออกแบบด้วยวิธีการนี้จะต้องมีเสถียรภาพ มีสมรรถนะ สามารถควบคุมระบบที่มีพารามิเตอร์ของกระบวนการเปลี่ยนแปลงภายใต้ขอบเขตที่กำหนดไว้ล่วง หน้าได้อย่างเหมาะสม

3. ตัวควบคุมที่ออกแบบนี้สมารถควบคุมระบบที่มีการรบกวนจากภายนอกได้

1.3 ขอบเขตการศึกษา

1. ออกแบบโปรแกรมเพื่อคำควณหาค่าพารามิเตอร์ของตัวควบคุมด้วยโปรแกรมคำนวณค่าทางคณิตศาสตร์(Matlab)

2. พัฒนาโปรแกรมควบคุมที่ใช้ในการทำงานจริงและจำลองการทำงานของระบบ ด้วยโปรแกรมคอมพิวเตอร์

3. ศึกษาโครงสร้างของกระบวนการที่มีพารามิเตอร์ทุกตัวเปลี่ยนแปลง เนื่องจากการเปลี่ยนแปลงระดับของการควบคุม และระบบควบคุมที่มีการรบกวนจากภายนอก

1.4 รายละเอียดของปริญญานิพนธ์

เนื้อหาที่จะกล่าวในปริญญานิพนธ์ฉบับนี้ประกอบด้วย

บทที่ 1 บทนำ กล่าวถึงแนวคิดในการทำปริญญานิพนธ์ วัตถุประสงค์ในการทำและขอบเขตของปริญญานิพนธ์

บทที่ 2 ทฤษฎี กล่าวถึงทฤษฎีทางคณิตศาสตร์เกี่ยวกับตัวควบคุม พีไอ ซึ่งเป็นพื้นฐานที่นำไปสู่การออกแบบตัวควบคุมแบบ พีไอ

บทที่ 3 อุปกรณ์ กล่าวถึงอุปกรณ์ต่างๆที่เป็นส่วนประกอบโครงสร้างที่ใช้ในการทดลองศึกษาทฤษฎี

บทที่ 4 การออกแบบตัวควบคุมและผลการทดลอง กล่าวถึงแสดงผลการทดลองที่ออกแบบมาโดยใช้วิธีโปรแกรมคำนวณค่าทางคณิตศาสตร์(Matlab)และไมโครคอนโทรลเลอร์

บทที่ 5 บทวิจารณ์และสรุปผลการทดลอง กล่าวถึงการสรุปผลการทดลองการดำเนินงาน แนวทางการปรับปรุงและพัฒนาโครงงาน

บทที่ 2

ทฤษฎี

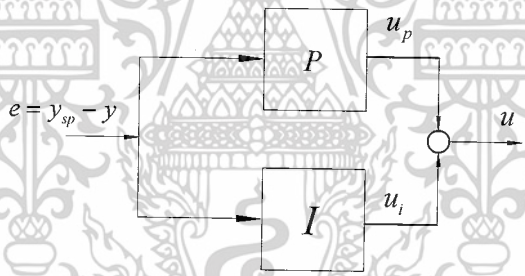
2.1 ตัวควบคุมแบบพีไอ (PI Controller)

เป็นตัวควบคุมที่รวมเอาคุณสมบัติของตัวควบคุมแบบ พี และแบบ ไอ เข้าด้วยกัน ดังแสดงในรูปที่ 2.1 โดยมีความสัมพันธ์ระหว่างสัญญาณจุดเข้า $e(t)$ และสัญญาณจุดออก $u(t)$ ดังนี้

$$u(t) = k_p e(t) + \frac{k_p}{T_i}\int_0^t e(t)dt \tag{2.1}$$

จากสมการจะได้ฟังก์ชั่นถ่ายโอนดังนี้

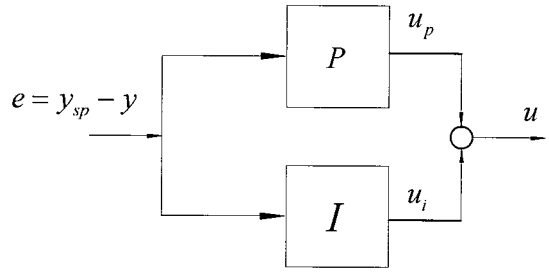
$$\frac{U(s)}{E(s)} = k_p\left(1 + \frac{1}{T_i s}\right) \tag{2.2}$$



รูปที่ 2.1 บล็อกไดอะแกรมของตัวควบคุมแบบ พีไอ

2.2 ตัวควบคุมพีไอสำหรับระบบหนึ่งสัญญาณเข้าหนึ่งสัญญาณออก (One Degree of Freedom Controller)

ตัวควบคุมพีไอเป็นตัวควบคุมที่นิยมกันมากในทางปฏิบัติ โดยเฉพาะการควบคุมกระบวนการและมีโครงสร้างในรูปที่ 2.2 รูปแบบนี้เรียกว่าตัวควบคุมพีไอชนิดขนานโดยที่มีสัญญาณเข้าเป็นสัญญาณค่าความผิดพลาด (e) ซึ่งเป็นผลต่างระหว่างสัญญาณอ้างอิง (y_{sp}) และสัญญาณขาออก (y) สัญญาณควบคุม (u) เป็นผลรวมของตัวควบคุม พี และ ไอ ตัวควบคุมชนิดนี้มีคุณสมบัติที่สำคัญ เช่น สามารถกำจัดค่าความผิดพลาดที่สถานะอยู่ตัวโดยอาศัยส่วนควบคุมไอ และสามารถคำนวณการเปลี่ยนแปลงของสัญญาณผิดพลาดเพื่อทำการชดเชยล่วงหน้า



รูปที่ 2.2 ตัวควบคุมพีไอชนิดขนาน

ตัวควบคุมพีไอในรูปที่ 2.2 สามารถเขียนฟังก์ชั่นถ่ายโอนได้ดังสมการ พบว่าพารามิเตอร์เพียงแค่สองตัวคือ k_p, k_i ที่ต้องออกแบบ

$$K(s) = k_p + \frac{k_i}{s} \tag{2.3}$$

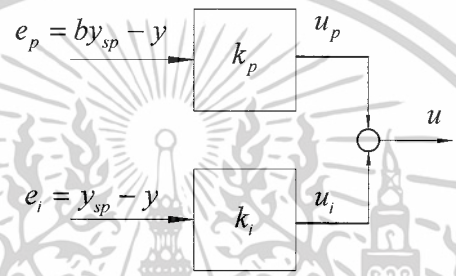
ระบบควบคุมป้อนกลับที่ใช้ตัวควบคุม พีไอ ชนิดขนานแสดงได้ในรูปที่ 2.3

รูปที่ 2.3 ระบบควบคุมป้อนกลับ พีไอ ชนิดขนาน

วัตถุประสงค์คือต้องการคำนวณค่า k_p, k_i ซึ่งทำให้ระบบควบคุมมีเสถียรภาพและมีสมรรถนะตามต้องการ โดยทั่วไปเราต้องมีความต้องการหลายอย่างเช่น ผลตอบสนองชั่วครู่ที่ดี สามารถกำจัดค่าความผิดพลาดที่สถานะอยู่ตัวและสามรถจำกัดสัญญาณรบกวนโหลดได้ในเวลาเดียวกัน ตัวควบคุมในสมการที่ 2.3 จะต้องตอบสนองความต้องการเหล่านี้โดยคำนวณจากสัญญาณความผิดพลาด (e)

2.3 ตัวควบคุมแบบพีไอถ่วงน้ำหนักสัญญาณอ้างอิง (Two Degree of Freedom Controller)

โครงสร้างของพีไอ แบบนี้มีความยืดหยุ่นสูงกว่าที่ผ่านมาโดยการจัดการผลตอบสนองต่อสัญญาณอ้างอิงและสัญญาณรบกวน โหลดแยกจากกันตัวควบคุมพีไอชนิดนี้มีลักษณะดังแสดงในรูปที่ 2.4



รูปที่ 2.4 โครงสร้างตัวควบคุมพีไอถ่วงน้ำหนักสัญญาณอ้างอิง

เราพบว่าสัญญาณควบคุมซึ่งเป็นผลรวมของสัญญาณออกของตัวควบคุมพีไอเขียนได้ดังนี้

$$u = k_p e_p + \frac{k_i}{s} e_i \tag{2.4}$$

สังเกตคือค่าความผิดพลาดของแต่ละตัวควบคุมมีนิยามที่แตกต่างกัน โดยค่าความผิดพลาดในส่วนของตัวควบคุมพีคือ

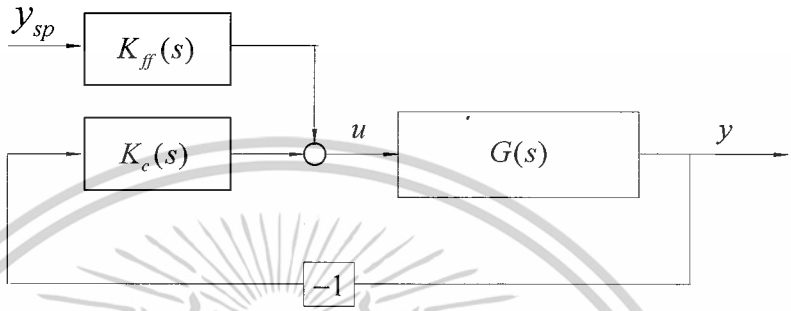
$$e_p = by_{sp} - y, \quad 0 \le b \le 1 \tag{2.5}$$

ค่าความผิดพลาดในส่วนตัวควบคุมไอจำเป็นจะต้องเป็นค่าที่แท้จริง

$$e_i = y_{sp} - y \tag{2.6}$$

เพื่อหลีกเลี่ยงความผิดพลาดที่สถานะอยู่ตัว ผลตอบสนองของระบบซึ่งใช้ตัวควบคุมพีไอที่มีค่าของ k_p, k_i ค่าเดียวกัน แต่ k_p จะมีค่าของ b ต่างกันจะให้ผลตอบสนองต่อสัญญาณรบกวนโหลดเหมือนกัน แต่ผลตอบสนองต่อสัญญาณอ้างอิงแตกต่างกัน ถ้ากำหนด b ให้เป็น 1 ผลตอบสนองจะ

มีค่าพุ่งเกินสูงสุดและถ้า b มีค่าเป็น 0 ให้ผลตอบสนองมีค่าพุ่งต่ำสุด ดังนั้นเราสามารถปรับจูนค่าของ b เพื่อปรับปรุงผลตอบสนองสัญญาณ อ้างอิงโดยไม่กระทบต่อผลตอบสนองสัญญาณรบกวนโหลด



รูปที่ 2.5 ระบบควบคุมป้อนกลับโดยใช้ตัวควบคุม พีไอ ถ่วงน้ำหนัก

สังเกตว่าจากรูปที่ 2.3 ซึ่งใช้ตัวควบคุม พีไอ สัญญาณควบคุม u ถูกคำนวณโดยอาศัยสัญญาณผิดพลาด e ไม่เป็นจริงสำหรับสัญญาณควบคุม u ที่นิยามโดยสมการที่ (2.4) และสัญญาณผิดพลาดถูกนิยามโดยสมการที่ (2.5) และ (2.6) ตามลำดับ ระบบควบคุมสำหรับระบบเชิงเส้นไม่เปลี่ยนแปลงตามเวลาหนึ่งสัญญาณเข้าหนึ่งสัญญาณออก (SISO Linear Time-Invariant System) จะได้ฟังก์ชั่นถ่ายโอนจากสัญญาณอ้างอิง y_{sp} ไปยังสัญญาณควบคุม u เขียนได้โดย

$$K_{ff}(s) = bk_p + \frac{k_i}{s} \tag{2.7}$$

และฟังก์ชั่นถ่ายโอนจากสัญญาณขาออก y ไปยังสัญญาณควบคุม u

$$K_c(s) = k_p + \frac{k_i}{s} \tag{2.8}$$

สังเกตว่าฟังก์ชั่นถ่ายโอนของตัวควบคุมทั้งสองแตกต่างกันและการออกแบบอิสระต่อกัน สำหรับการออกแบบตัวควบคุม $K_c(s)$ เป็นจุดประสงค์หลัก ส่วนตัวควบคุมออกแบบหลังจากได้รับตัวควบคุม $K_c(s)$ แล้วโดยการปรับจูนค่าของ b เพื่อปรับปรุงผลตอบสนองต่อสัญญาณอ้างอิง

2.4 ทฤษฎีการออกแบบตัวควบคุมชนิดพีไอ

สำหรับวิธีการที่จะใช้การออกแบบตัวควบคุมที่ดีนั้น นอกจากจะต้องเป็นวิธีที่สามารถทำให้ระบบควบคุมมีสมรรถนะที่ดีคือ สามารถควบคุมให้ระบบควบคุมมีผลตอบสนองตามที่ได้ออกแบบไว้ มีเสถียรภาพและมีความคงทนต่อการเปลี่ยนแปลงพารามิเตอร์ของกระบวนการโดยใช้ทฤษฎีของ Karl Johan Åström และ Sigurd Skogestad ในการหาค่าของตัวควบคุม

2.4.1 ทฤษฎีของคาร์ล เจ แอสสตอม (Karl J. Åström)

ในส่วนของทฤษฎีของ Karl J. Åström จะมีการหาค่าพารามิเตอร์มีอยู่ 2 วิธีคือ ตัวควบคุมพีไอสำหรับระบบหนึ่งสัญญาณเข้าหนึ่งสัญญาณออก (One Degree of Freedom Controller) และตัวควบคุมแบบพีไอถ่วงน้ำหนักสัญญาณอ้างอิง (Two Degree of Freedom Controller) ซึ่งจะมีสมการในการหาค่าของพารามิเตอร์ k_p และ k_i ดังนี้

2.4.1.1 Optimization Problem

$$\min \int_0^\infty e(t)dt \tag{2.9}$$

โดย

$$f(k,k_i,\omega) \geq R^2 \tag{2.10}$$

$$\therefore \quad R = \frac{1}{M_s}$$

ดังนั้น

$$f(k,k_i,\omega) = \left| C + (k + i\frac{k_i}{\omega})G(i\omega) \right|^2 \tag{2.11}$$

ซึ่ง

$$G(i\omega) = \alpha(\omega) + i\beta(\omega) = r(\omega)e^{i\varphi(\omega)} \tag{2.12}$$

เมื่อ

$$\beta(\omega) = r(\omega)\sin\varphi(\omega)$$
$$\alpha(\omega) = r(\omega)\cos\varphi(\omega) \tag{2.13}$$

โดยเราสามารถเขียนฟังก์ชันของสมการ (2.11) ใหม่ได้ดังนี้

$$f(k,k_i,\omega) = C^2 + 2C\alpha(\omega)k + 2C\alpha\frac{\beta(\omega)}{\omega}k_i + r^2(\omega)k^2 + \frac{r^2(\omega)}{\omega^2}k_i^2 \tag{2.14}$$

และสามารถหาค่าพารามิเตอร์ของ k, k_i จากสมการที่ (2.14) ได้ดังนี้

$$k = -C\frac{\alpha}{r^2} = -C\frac{1}{r}\cos\varphi,$$
$$k_i = \frac{\omega\beta C}{r^2} - \frac{\omega R}{r} = -\frac{\omega}{r}(C\sin\varphi + R) \tag{2.15}$$

ซึ่งเราสามารถหาค่าพารามิเตอร์ของ ω โดยใช้วิธีการหาแบบ Newton - Raphson Method ได้จากสมการที่ (2.17)

$$h(\omega) = 2R\left(\left(C\frac{\beta}{r} + R\right)\left(\frac{r'}{r} - \frac{1}{\omega}\right) - C\left(\frac{\beta}{r}\right)'\right) \tag{2.16}$$

$$= 2R\left((R + C\sin\varphi)\left(\frac{r'}{r} - \frac{1}{\omega}\right) - C\varphi'\cos\varphi\right) \tag{2.17}$$

$$= 0$$

ในการเลือกค่าของ ω เริ่มต้นนั้นเราสามารถหาได้ด้วยการเลือก ω เริ่มต้นดังสมการที่ (2.18)

$$\omega_{90} < \omega < \omega_{180-\arcsin(R/C)} \tag{2.18}$$

2.4.2 ทฤษฎีของไซกราส สโครแกสแตรน (Sigurd Skogestad)

จากทฤษฎีของ Skogestad มีอยู่สองวิธีที่จะเลือกการปรับจูนพารามิเตอร์ของ พีไอ แล้วแต่กรณี

- **Tight Control** คือการจูนค่าพารามิเตอร์ที่ต้องการให้ระบบที่ควบคุมมีความเร็วที่สุดที่จะยอมรับผลตอบสนองได้
- **Smooth Control** คือการจูนค่าพารามิเตอร์ให้ระบบมีความช้าที่ยอมรับได้เพื่อลดปัญหาจากสัญญาณรบกวน

ในทางปฏิบัติทางอุสาหกรรม จะเป็นระบบที่ที่คุณลักษณะจำเพาะที่มีอันดับสูง ซึ่งจะประกอบไปด้วยค่าโพล(Pole)ที่สำคัญและไม่สำคัญ โดยค่าโพล(Pole)ที่ไม่สำคัญจะแสดงผลตอบสนองทางเวลาของระบบวงปิดที่อยู่ในส่วนของเวลาที่มีช่วงเวลาไร้ผลตอบสนอง ดังนั้นในการปรับแต่งตัวควบคุมจากผลตอบสนองทางเวลายังคงประมาณหาแบบจำลองอันดับหนึ่งหรืออันดับสองที่หาได้จากระบบวงปิดแบบจำลองของระบบที่รวมเวลาหน่วง

2.5 แบบจำลองอันดับหนึ่งรวมช่วงเวลาหน่วง

$$G(s) = k\frac{e^{-\theta s}}{\tau_1 s + 1} \tag{2.19}$$

2.5.1 ตัวควบคุมแบบ พีไอ อธิบายโดยฟังก์ชันถ่ายโอน

$$K_c(s) = k_p(1 + \frac{1}{\tau_i s}) \tag{2.20}$$

2.5.1.1 การออกแบบหาตัวพารามิเตอร์ พีไอ โดยทฤษฎี Skogestad

$$k_p = \frac{1}{k}\frac{\tau_1}{\tau_c + \theta} \tag{2.21}$$

และ

$$\tau_i = \min(\tau_1, 4(\tau_c + \theta)) \tag{2.22}$$

τ_c = closed - loop time constant

2.6 การออกแบบ โดยไทร์คอนโทรล (Tight Control)

การปรับค่าพารามิเตอร์ของตัวควบคุมเป็นขั้นตอนสำคัญที่จะทำให้ผลตอบสนองของระบบวงปิดเป็นไปตามข้อกำหนดที่วางเอาไว้ โดยปกติระบบวงปิดจะต้องการให้มีผลตอบสนองที่เร็วและมีเสถียรภาพ ดังนั้นการออกแบบตัวควบคุมแบบโดยวิธีของ Skogestad จะเป็นทางเลือกอีกวิธีหนึ่งที่มีความรวดเร็วโดยมีวิธีการออกแบบดังนี้

วิธีในการออกแบบหาค่าพารามิเตอร์ k_p, τ_i เพื่อที่จะนำไปสมการที่ (2.21) และ (2.22)

$$k_p = \frac{1}{k} \frac{\tau_1}{\tau_{c,\min} + \theta} \tag{2.23}$$

$$\tau_i = \min(\tau_1, 4(\tau_{c,\min} + \theta)) \tag{2.24}$$

โดยสำหรับ Tight control จะกำหนดให้ค่าพารามิเตอร์ τ_c มีค่าเท่ากับค่าช่วงเวลาหน่วง

$$\tau_{c,\min} = \theta \tag{2.25}$$

และค่าพารามิเตอร์ของ k และ τ_1 สามารถนำค่ามาจากสมการที่ (2.19) นำมาแทนค่าในสมการที่ (2.21) และ (2.22) เราก็ทราบตัวพารามิเตอร์ ของตัวควบคุมแบบพีไอ

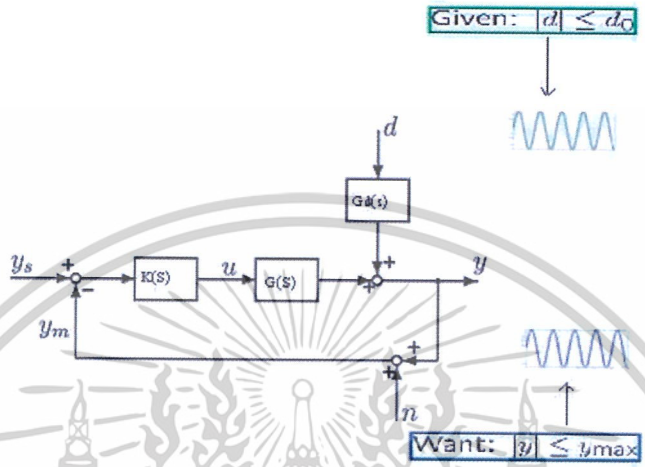
2.7 การออกแบบตัวควบคุม โดยสมูทคอนโทรล (Smooth Control)

การปรับค่าพารามิเตอร์ของตัวควบคุมเป็นขั้นตอนสำคัญที่ จะทำให้ผลตอบสนองของระบบวงปิดเป็นไปตามข้อกำหนดที่วางเอาไว้แล้วแต่บางกรณี โดยปกติระบบวงปิดจะต้องการให้มีผลตอบสนองที่เร็วและมีเสถียรภาพ แต่บางครั้งระบบอาจมีสัญญาณรบกวน เนื่องจากการวัดจึงต้องออกแบบตัวควบคุมไม่ให้มีความไวต่อสัญญาณรบกวนดังกล่าว Smooth control จึงเป็นตัวเลือกอีกวิธีหนึ่งที่ไม่มีความไวต่อสัญญาณรบกวน

ข้อดีของการออกแบบด้วยวิธี Smooth นี้

- ใช้พลังงาน Input น้อย
- สัญญาณรบกวนที่ได้มาการวัดมีค่าน้อย

- มีความคงทนต่อการเปลี่ยนแปลง
- เมื่อมีการเปลี่ยนแปลงของโหลดสามารถกำจัดได้รวดเร็ว



รูปที่ 2.6 ระบบวงปิดที่มีโหลด และสัญญาณรบกวน

วิธีในการออกแบบหาค่าพารามิเตอร์ k_p, τ_i เพื่อที่จะนำไปสมการที่ (2.21) และ (2.22) โดยวิธี Smoot

$$k_p = \frac{1}{k}\frac{\tau_1}{\tau_{c,\max} + \theta} \tag{2.26}$$

$$\tau_i = \min(\tau_1, 4(\tau_{c,\max} + \theta)) \tag{2.27}$$

โดยค่าของ

$$\tau_{c,\max} = \frac{\tau_1}{k}\frac{1}{k_{c,\min}} - \theta \tag{2.28}$$

และ

$$k_c \ge k_{c,\min} = \left|\frac{d_0}{y_{\max}}\right| \tag{2.29}$$

โดยมีเงื่อนไขอยู่ว่า

$$\theta < 0.5\frac{\tau_1}{k}\left|\frac{y_{\max}}{d_0}\right| \tag{2.30}$$

$|d_0|$: Input magnitude required for disturbance rejection

$|y_{\max}|$: Allowed output deviation

2.8 การทดลองและผลการทดลอง

2.8.1 ตัวอย่าง การออกแบบระบบตัวควบคุม พีไอ

$$G(s) = \frac{1}{(s+1)(0.5s+1)(0.2s+1)}$$

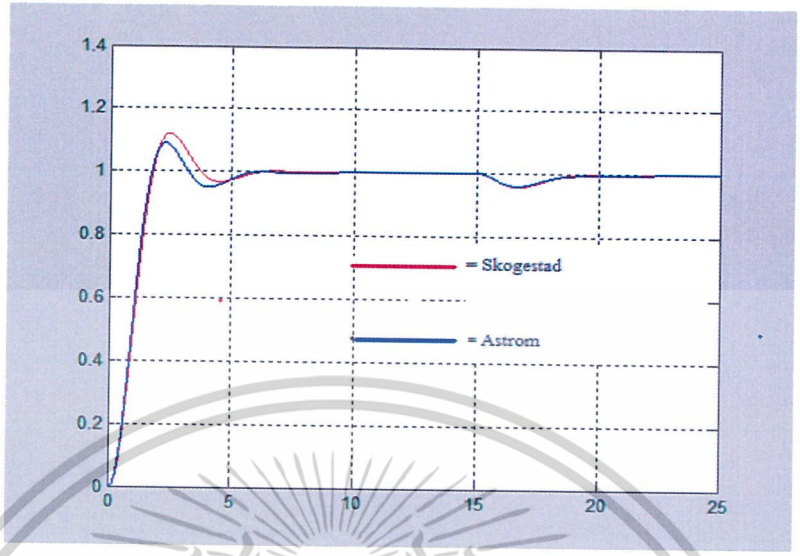
Astrom(MIGO)

Parameter Tight Controller $k_c = 1.5796,\ \tau_i = 1.514$

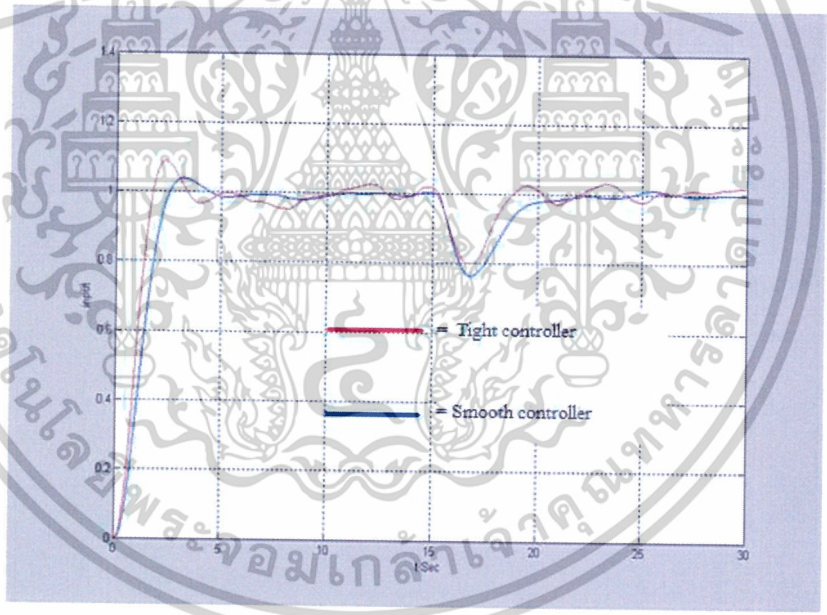
Skogestad

Parameter Tight Controller $k_c = 1.389,\ \tau_i = 1.25$

Parameter Smooth Controller $k_c = 1,\ \tau_i = 1.25$



รูปที่ 2.7 กราฟการเปรียบเทียบผลตอบสนองของ **Tight control**



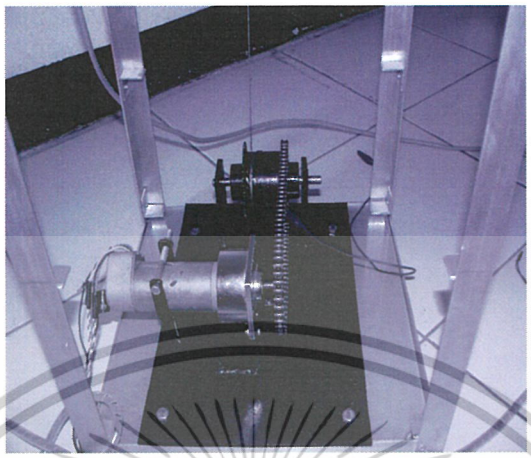
รูปที่ 2.8 กราฟการเปรียบเทียบผลตอบสนองของ **Tight control** กับ **Smooth control**

บทที่ 3
โครงสร้างและอุปกรณ์

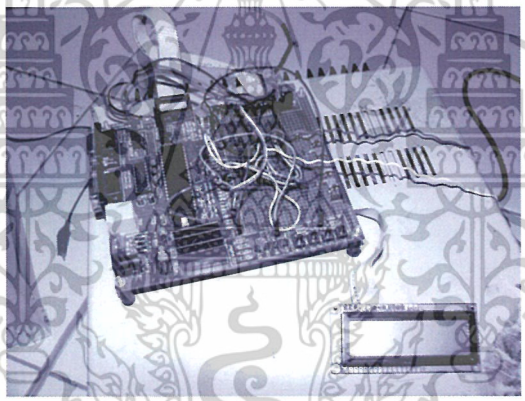
3.1 โครงสร้าง

ในโครงงานนี้ได้สร้างระบบระบบหนึ่งออกมาคือ อุปกรณ์ตรวจวัดความตึงของเชือก ซึ่งได้ใช้ทฤษฎีของ Karl Johan Åström ในการคำนวณหาค่าพารามิเตอร์ของตัวควบคุมแบบ พีไอ เมื่อทำการทดลอง ซึ่งตัวโครงสร้างทำมาจากสแตนเลสที่มีความคงทน แข็งแรงและมีน้ำหนักเบา และมีอุปกรณ์อื่นๆที่เข้ามาเกี่ยงข้องในการทำงานของระบบ เช่น ชุดไมโครคอนโทรลเลอร์ มอเตอร์กระแสตรง ลวดสลิง รอก ชุดเกียร์ทดรอบ และโหลดเซลล์ เป็นต้น ลักษณะโครงสร้างมีดังรูป

รูปที่ 3.1 โครงสร้างของระบบ



รูปที่ 3.2 ชุดมอเตอร์กระแสตรงและเกียร์ทดรอบ

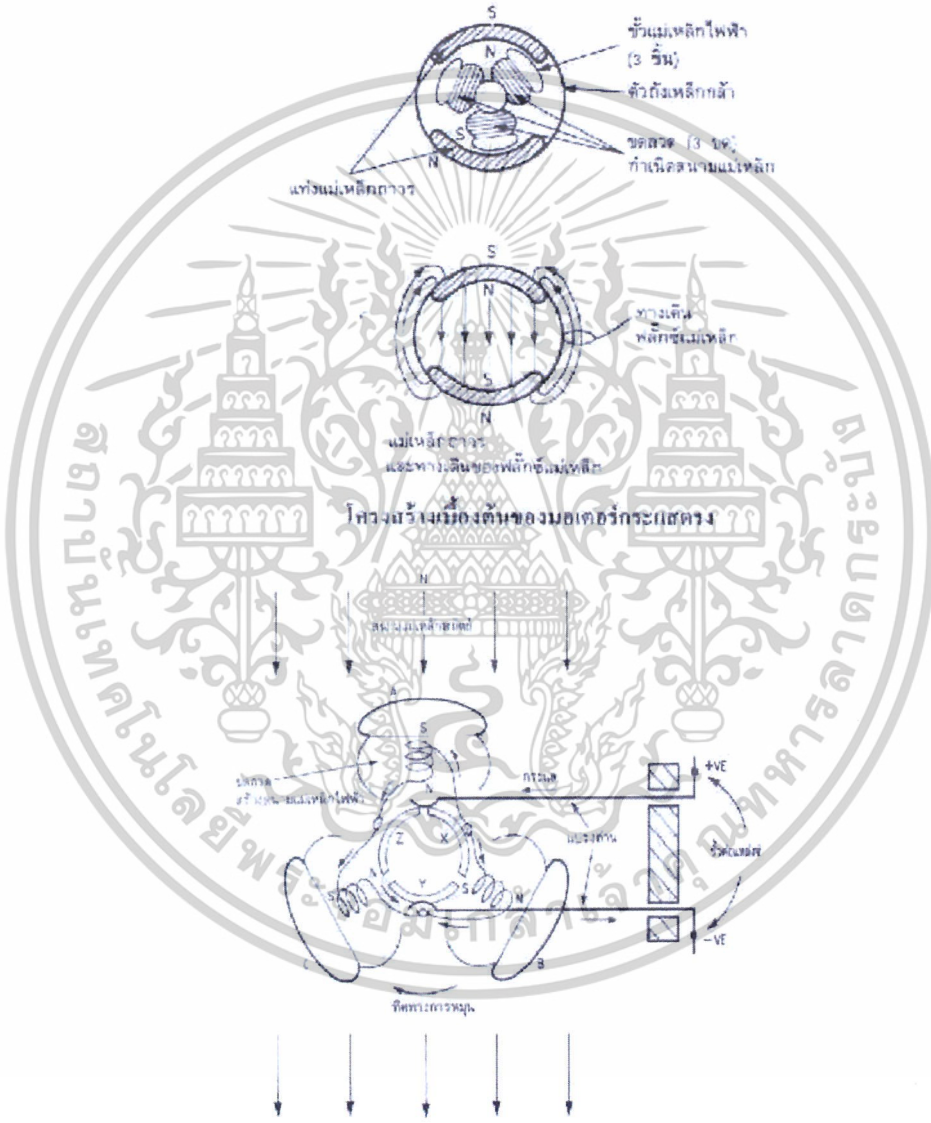


รูปที่ 3.3 ชุดไมโครคอนโทรลเลอร์

รูปที่ 3.4 โหลดเซลล์

3.2 มอเตอร์กระแสตรง (D.C Motor)

มอเตอร์กระแสตรงเป็นอุปกรณ์ไฟฟ้ากระแสตรงที่ทำหน้าที่เปลี่ยนแปลงพลังงานไฟฟ้าไปเป็นพลังงานเชิงกล อุปกรณ์ที่ใช้งานต่างๆที่เกี่ยวข้องกับมอเตอร์กระแสตรงนั้น ใช้งานได้ง่ายและสะดวกมีราคาถูกกว่ามอเตอร์อื่นๆ



รูปที่ 3.5 โครงสร้างเบื้องต้นของมอเตอร์กระแสตรง

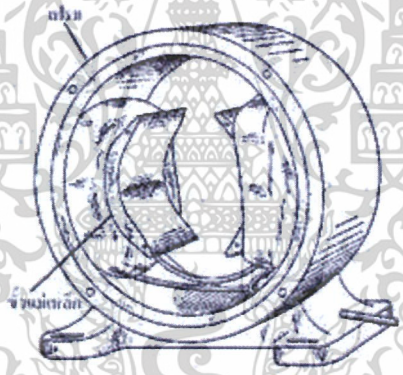
หลักการทำงานของมอเตอร์กระแสตรงเมื่อมีการผ่านกระแสไฟฟ้าเข้าไปยังขดลวดในสนามแม่เหล็กจะทำให้เกิดแรงแม่เหล็กซึ่งมีสัดส่วนของแรงขึ้นกับกระแสแรงของสนามแม่เหล็ก โดยแรงจะเกิดขึ้นเป็นมุมฉากกับกระแสและสนามแม่เหล็ก ขณะที่ทิศทางของแรงกลับตรงกันข้ามกัน ถ้าหากกระแสของสนามแม่เหล็กไหลย้อนกลับจะทำให้เกิดการเปลี่ยนแปลงของกระแส และ

สนามแม่เหล็กเป็นผลทำให้ทิศทางของแรงเปลี่ยนไป ด้วยคุณสมบัตินี้ทำให้มอเตอร์กระแสตรงกลับทิศทางการหมุนได้สนามแม่เหล็กของมอเตอร์ส่วนหนึ่งเกิดขึ้นจากแม่เหล็กถาวรซึ่งจะถูกยึดติดกับแผ่นเหล็ก หรือ เหล็กกล้า โดยปกติส่วนนี้จะเป็นส่วนที่ยึดอยู่กับที่ และ ขดลวดเหนี่ยวนำจะพันอยู่กับส่วนที่เป็นแกนหมุนของมอเตอร์โดยโครงสร้างของมอเตอร์ไฟฟ้ากระแสตรง (D.C. Motor) ประกอบด้วย 2 ส่วนหลัก ๆ คือ ส่วนที่อยู่กับที่ และ ส่วนที่เคลื่อนที่

3.2.1 ส่วนที่อยู่กับที่ ได้แก่

เฟรม คือ เป็นโครงสร้างภายนอก ที่เรามองเห็นเป็นตัวมอเตอร์ จะทำหน้าที่เป็นเส้นทางเดินของสนามแรงแม่เหล็ก และเป็นที่ยึดส่วนต่าง ๆ ให้แข็งแรง

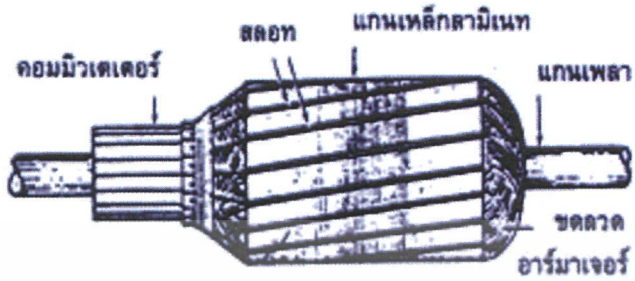
ขั้วแม่เหล็ก จะประกอบด้วย แกนขั้วแม่เหล็ก ส่วนนี้จะติดอยู่กับเฟรมและขดลวดสนามแม่เหล็ก (Field coil) ที่พันรอบ ๆ แกนขั้วแม่เหล็ก จะทำหน้าที่รับกระแสจากภายนอก และสร้างสนามแม่เหล็ก ซึ่งจะทำให้เกิดแรงบิดขึ้น (Torque)



รูปที่ 3.6 ขั้วแม่เหล็กและขดลวดแม่เหล็กที่ยึดติดกับเฟรม

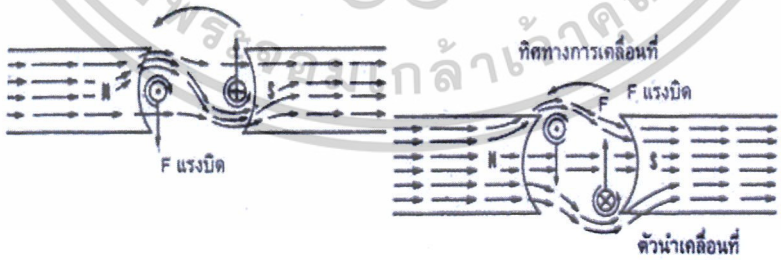
3.2.2 ส่วนที่เคลื่อนที่ ได้แก่

โรเตอร์ (rotor) จะมีขดลวดอาร์เมเจอร์(Armature Winding)ที่พันอยู่บนแกนเหล็กอาร์เมเจอร์ (Armature core) และมีคอมมิวเตเตอร์ยึดติดอยู่ที่ปลายของขดลวดอาร์เมเจอร์



รูปที่ 3.7 โรเตอร์ของมอเตอร์กระแสตรง

ซึ่งในส่วนนี้ คอมมิวเตเตอร์จะทำหน้าที่ในการสัมผัสกับแปรงถ่านคาร์บอน (Carbon Brushes)ที่อยู่ในมอเตอร์เพื่อที่จะให้มีกระแสไหลผ่านไปยังขดลวดอาร์เมเจอร์ ทำให้เกิดการสร้างสนามแม่เหล็กขึ้นเพื่อให้เกิดการหักล้างและเสริมกันกับสนามแม่เหล็กที่เกิดจากขดลวดแม่เหล็ก ซึ่งจะทำให้มอเตอร์หมุนได้ เมื่อมีกระแสไหลผ่านเข้าไปในมอเตอร์กระแสจะแบ่งออกไป 2 ทาง คือ ส่วนที่หนึ่งจะผ่านเข้าไปที่ขดลวดสนามแม่เหล็ก (Field coil) ทำให้เกิดสนามแม่เหล็กขึ้นและอีกส่วนหนึ่งจะผ่านแปลงถ่านคาร์บอนและผ่านคอมมิวเตเตอร์เข้าไปในขดลวดอาร์เมเจอร์ทำให้เกิดสนามแม่เหล็กขึ้นเช่นกัน ซึ่งทั้งสองสนามจะเกิดขึ้นขณะเดียวกัน ตามคุณสมบัติของเส้นแรงแม่เหล็กแล้วจะไม่มีการตัดกัน จะมีแต่การหักล้างและการเสริมกัน ซึ่งทำให้เกิดแรงบิดในอาร์เมเจอร์ ทำให้อาร์เมเจอร์หมุนซึ่งในการหมุนนั้นจะเป็นไปตามกฎมือซ้ายของเฟลมมิ่ง (Fleming's left hand rule)



รูปที่ 3.8 แสดงทิศทางการเคลื่อนที่ของอาร์เมเจอร์ (โรเตอร์)

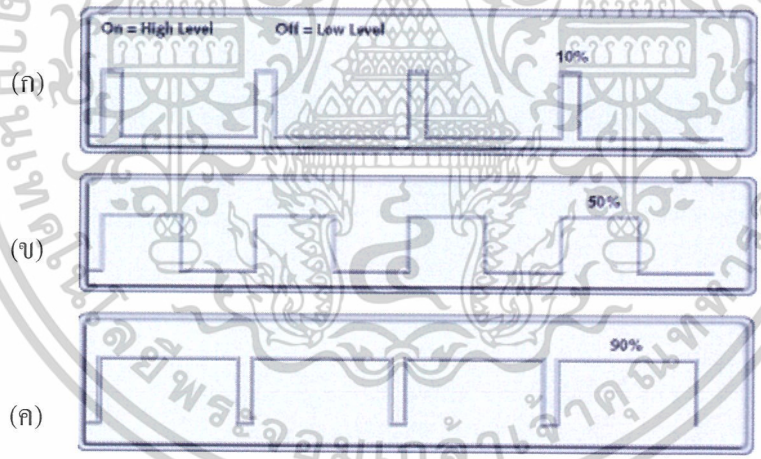
3.3 การควบคุมแบบพัลซ์ วิธ มอดูเลชั่น (Pulse Width Modulation)

Pulse Width Modulation (PWM) คือ เทคนิคสำหรับควยคุมวงจรทางด้านฮาร์ดแวร์โดยใช้สัญญาณเอาท์พุตแบบดิจิตอลของไมโครโปรเซสเซอร์ควบคุม

3.3.1 การทำงานของสัญญาณ PWM

รูปที่ 3.9 แสดงสัญญาณ PWM ที่แตกต่างกัน 3 สัญญาณ โดยที่

- รูปที่ 3.9(ก) แสดงสัญญาณ PWM ที่ 10% duty cycle คือ สัญญาณในการ ON จะเป็น 10% ของคาบสัญญาณ และ จะ OFF เป็น 90% ของคาบสัญญาณ
- รูปที่ 3.9(ข)แสดงสัญญาณ PWM ที่ 50% duty cycle คือ สัญญาณในการ ON จะเป็น 10% ของคาบสัญญาณ และ จะรูป OFF เป็น 50% ของคาบสัญญาณ
- รูปที่ 3.9(ค) แสดงสัญญาณ PWM ที่ 90% duty cycle คือ สัญญาณในการ ON จะเป็น 10% ของคาบสัญญาณ และ จะ OFF เป็น 10% ของคาบสัญญาณ



รูปที่ 3.9 สัญญาณ พัลซ์ วิธ มอดูเลชั่น ซึ่งแสดงค่าดิวตี้ไซเคิลที่ต่างๆกัน

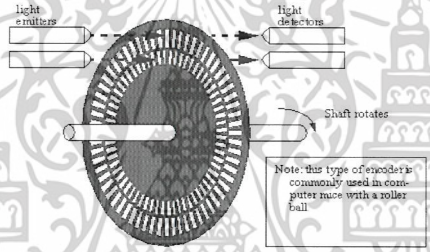
3.4 อุปกรณ์เข้ารหัส (Encoder)

Encoder จัดเป็นอุปกรณ์เซนเซอร์เชิงกลที่สามารถให้ผลของสัญญาณไฟฟ้าด้านเอาต์พุตในรูปของไบนารี่ ดิจิตอล และอนาลอก (ขึ้นอยู่กับลักษณะรูปร่างของ Encoder แต่ละชนิด) โดยใช้งานร่วมกับอุปกรณ์เซนเซอร์ทางแสงซึ่งสัญญาณทางด้านเอาต์พุตนั้นอาศัยหลักการตัดต่อลำแสง

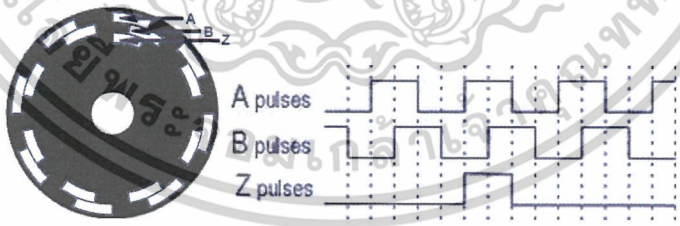
ระหว่างตัวรับและตัวส่งจากแถบทึบแสงของ Encoder ซึ่งสามารถแบ่งประเภทของ Encoder ออกได้เป็น 2 ชนิดคือ

3.4.1 อุปกรณ์เข้ารหัสแบบเพิ่มค่า (Increment Encoder) มักจะเป็น Encoder แบบ Rotary Encoder ขนาดของ Output ของ Encoder แบบนี้ขึ้นอยู่กับการออกแบบวงจรนับและวงจรคำนวณ

3.4.2 อุปกรณ์เข้ารหัสแบบสัมบูรณ์ (Absolute Encoder) ที่พบเห็นส่วนใหญ่จะเป็นแบบ Potentiometer แต่บางครั้ง ก็อาจเป็นแบบ Rotary Encoder ก็ได้แต่จะมีการออกแบบลักษณะของช่อง Slit ที่ต่างไปจาก Rotary Encoder โดยอาจจะมีค่าของ Output ตั้งแต่ 4 bit ถึง 16 bit



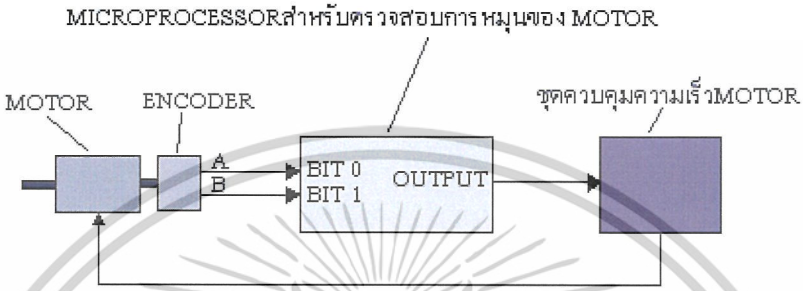
รูปที่ 3.10 ลักษณะรูปร่างของอุปกรณ์เข้ารหัสแบบเพิ่มค่า



รูปที่ 3.11 ลักษณะรูปร่างของอุปกรณ์เข้ารหัสแบบสัมบูรณ์

ในแง่ของการใช้งานแล้วมอเตอร์ที่มี Encoder แบบ Incremental Encoder นั้นจะใช้งานยุ่งยากกว่ามอเตอร์ที่มี Encoder แบบ Absolute Encoder เพราะ output ของ Increment Encoder นั้นไม่สามารถเชื่อมต่อกับชุดควบคุม มอเตอร์ (Controller) ได้ทันทีโดยต้องต่อผ่านวงจรนับ plus และ

วงจรคำนวณที่ยุ่งยากซับซ้อนก่อน ถึงจะต่อเข้ากับกับ Controller ได้ แต่อย่างไรก็ตาม ในปัจจุบันเรามี Microprocessor เช่น Microcontroller ตระกูล PIC เป็นต้น ดังนั้นเราจึงสามารถนำมาใช้แทนวงจรนับ pulse และวงจรคำนวณได้จึงทำให้การใช้งานมีความง่ายขึ้น ดังรูปที่แสดงไว้ข้างล่าง



รูปที่ 3.12 แสดงบล๊อคไดอะแกรมของโรตารี่เอ็นโค๊ดเดอร์

3.5 รอก (Pulley)

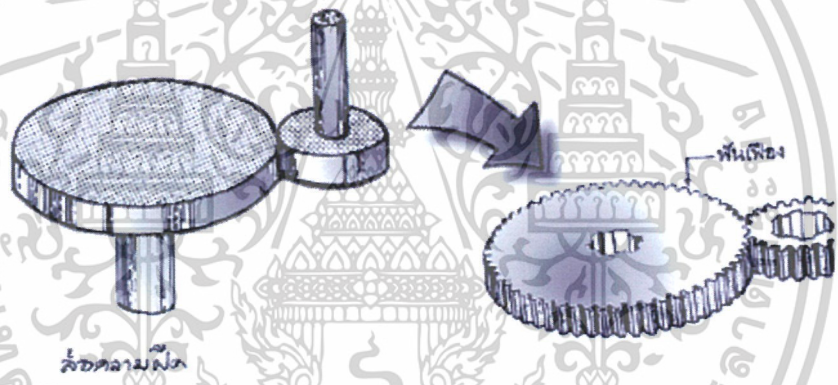
รอกเป็นเครื่องมือที่ใช้อำนวยความสะดวกของการทำงาน โดยการเปลี่ยนทิศทางของแรงหรือใช้เป็นเครื่องทุ่นแรงในการยกวัสดุขึ้นและลง ลักษณะของการทำงาน รอกมีลักษณะเป็นล้อหมุนที่คล่องที่รอบตัวและมีร่องล้อ ซึ่งใช้เชือกหรือลวดสลิงพาดผ่านร่องเพื่อยกวัสดุ จะใช้การผูกวัสดุติดกับปลายเชือกหรือ ลวดสลิงปลายด้านหนึ่งส่วนอีกปลายด้านหนึ่งสอดผ่านร่องรอกที่ยึดตรึงไว้ แล้วออกแรงดึงที่ปลายก็สามารถยกวัสดุได้ หรือจะใช้วิธียึดตรึงปลายด้านหนึ่งของเชือกหรือลวดสลิง ส่วนอีกปลายด้านหนึ่งสอดผ่านร่องรอกซึ่งแขวนวัสดุไว้ ออกแรงดึงที่ปลายก็สามารถยกวัสดุได้ รอกแบ่งได้เป็น 2 ประเภท ได้แก่ รอกเดี่ยวและรอกพวง

(ก) (ข)

รูปที่ 3.13 (ก) รอกเดี่ยว และ (ข) รอกพวง

3.6 เฟืองทดรอบ

การถ่ายทอดการหมุนจากต้นกำลังนั้น ทำได้หลายวิธี เช่น ด้วยการใช้สายพาน โซ่ ล้อความฝืด เป็นต้น ล้อความฝืดก็คือ ล้อสองล้อที่ถูกกดให้ติดกัน เมื่อล้อหนึ่งหมุน หรือเป็นล้อขับก็จะทำให้อีกล้อหนึ่งหมุนตาม เพราะผิวหน้าของล้อทั้งสองเกิดความฝืด เนื่องจากการสัมผัส แต่ถ้าหากมีภาระมากๆ เช่น มีการส่งกำลังสูงๆ จะทำให้เกิดการลื่นไถล การส่งกำลังจึงไม่แม่นยำ เพื่อที่จะแก้ไขข้อเสียเหล่านี้จึงได้มีการนำเอาฟันเฟืองมาติดไว้ที่ผิวของล้อโดยรอบล้อ จึงมีลักษณะเป็นล้อฟันเฟือง ซึ่งต่อๆมาเราจึงเรียกว่า "เฟือง" ซึ่งเป็นชิ้นส่วนที่สามารถส่งกำลังหรือถ่ายทอดการหมุนได้แม่นยำเที่ยงตรง และไม่มีการลื่นไถล



รูปที่ 3.14 หลักการทำงานของเฟือง

เฟืองนั้นเป็นอุปกรณ์ทางแมคคานิกส์ (Mechanic) ที่ใช้กันมากในงานสำคัญต่างๆ แต่งานที่ใช้กันมากที่สุดคือการทดเฟืองในมอเตอร์

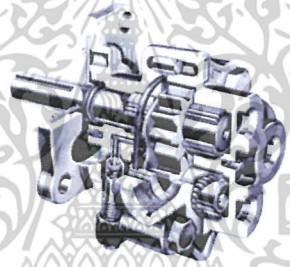
3.6.1 การทดเฟือง นั้นมีจุดมุ่งหมายที่สำคัญ 3 อย่างคือ

- เพื่อเพิ่ม torque หรือ แรงบิด หากว่าเรานำเฟืองขนาดเล็กที่ติดกับมอเตอร์มาขบกับเฟืองที่มีขนาดใหญ่กว่าผลที่ได้คือ จะทำให้ความเร็วของเฟืองใหญ่นั่นลดลง แต่จะทำให้มีแรงบิด (torque) มากขึ้น ทำให้มีกำลังมากยกตัวอย่างที่สามารถเห็นได้คือ ไขควงไฟฟ้าที่มีการทดเฟืองอย่างมากเนื่องจากต้องใช้แรงบิดมากนั่นเอง

- เพื่อเพิ่ม speed หรือ ความเร็วในทางกลับกันหากเราต้องการความเร็วที่มากขึ้นเราก็จำเฟืองที่ติดกับมอเตอร์ขนาดใหญ่มาขบกับเฟืองที่มีขนาดที่เล็กกว่าจะทำให้เราได้ความเร็วที่เพิ่มขึ้นมาแต่ว่าจะทำให้ แรงบิด (torque) ลดลง
- ใช้เปลี่ยนทิศทางของการหมุนได้ยกตัวอย่างเช่นล้อของรถยนต์ ซึ่งใช้เปลี่ยนทิศทางการหมุนไป 90 องศาในการจะส่งกำลังจากเพลากลางไปยังล้อ

3.6.2 อัตราการทดเฟือง

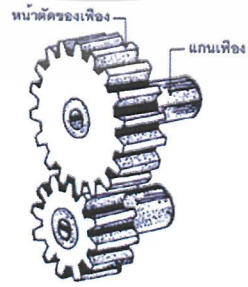
อัตราการทดเฟืองเราสามารถหาได้จาก อัตราส่วนระหว่างเส้นผ่านศูนย์กลางของเฟืองทั้งสองที่ขบกัน หรือเราอาจคำนวณได้จากจำนวนอัตราส่วนของฟันเฟืองที่ขบกัน



รูปที่ 3.15 ชุดเฟืองทด

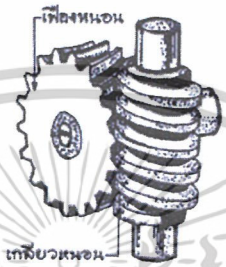
3.6.3 ชนิดของเฟือง สามารถแบ่งได้เป็น 3 ชนิดคือ

3.6.3.1 เฟืองตรง (Spur gear) เป็นเฟืองที่มีลักษณะเป็นล้อทรงกระบอก มีฟันขนานกับแกนของตัวเฟือง มีหน้าตัดของฟันเฟืองขนานเท่ากันและเหมือนกันตลอดทั้งเฟือง



รูปที่ 3.16 เฟืองตรง

3.6.3.2 เฟืองหนอน (Worm gear) เฟืองชุดนี้จะประกอบด้วยตัวเกลียวหนอนและเฟืองหนอน โดยเกลียวหนอนจะส่งกำลังหมุนไปขับให้เฟืองหนอนหมุน เฟืองชนิดนี้นิยมใช้กับการทดสอบความเร็วสูงๆให้เป็นความเร็วต่ำมากๆเช่น ในกรณีของการทดรอบจากมอเตอร์ซึ่งมีความเร็วสูง เป็นต้น



รูปที่ 3.17 เฟืองหนอน

3.6.3.3 เฟืองดอกจอก (Bevel gear) เฟืองชนิดนี้มีลักษณะรูปร่างเป็นรูปทรงกรวยฟันของเฟืองจะอยู่โดยรอบผิวของทรงกรวย และขนานกับแกนของเฟือง เฟืองดอกจอกจะใช้สำหรับเปลี่ยนทิศทางการส่งกำลังระหว่างเพลาของล้อที่ตั้งฉากกันเช่น การส่งกำลังไปยังเพลาของล้อรถ เป็นต้น

รูปที่ 3.18 เฟืองดอกจอก

บทที่ 4

การออกแบบตัวควบคุมและผลการทดลอง

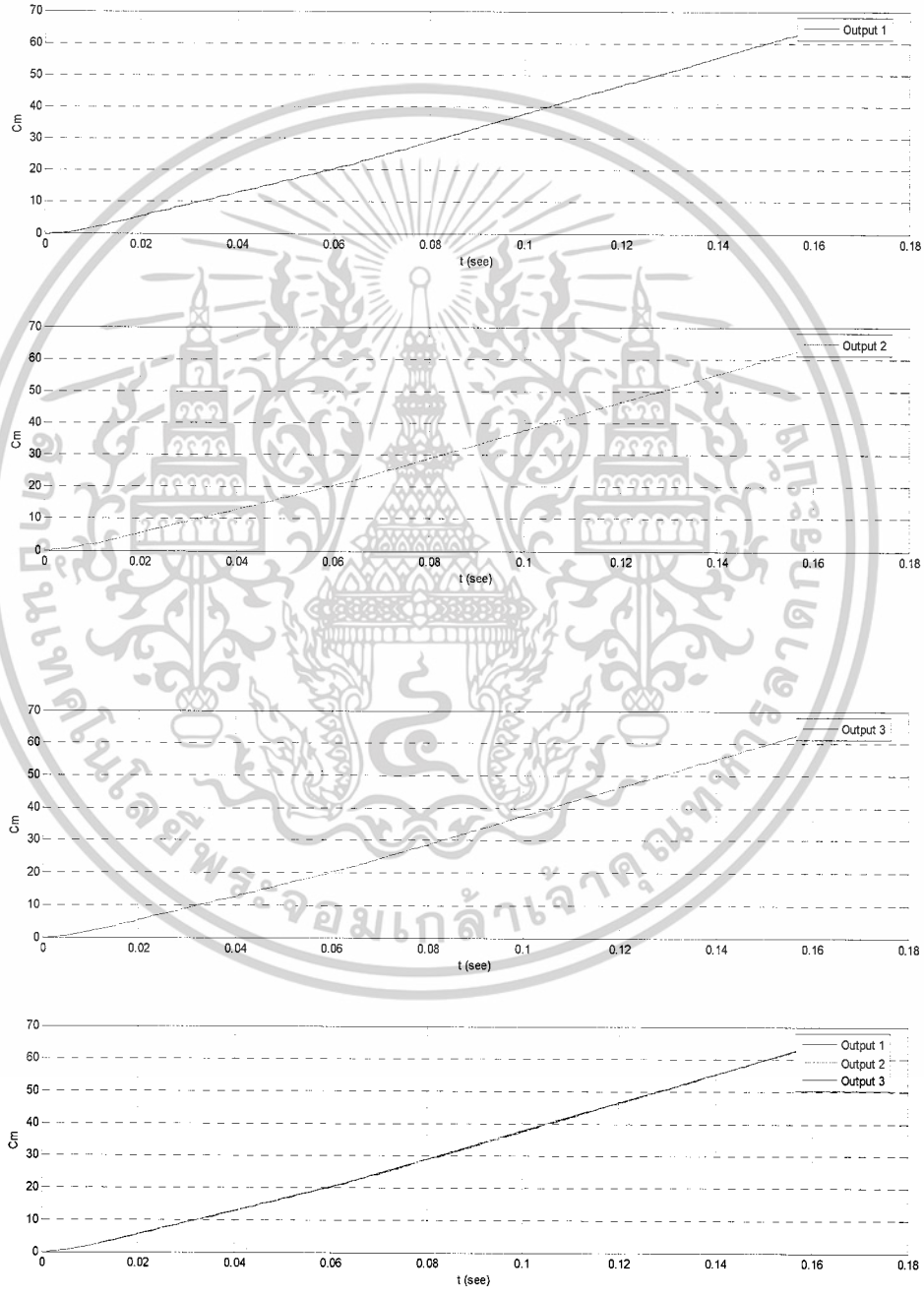
4.1 บทนำ

ในการออกแบบตัวควบคุมนั้นเป้าหมายสูงสุดคือการสร้างตัวควบคุมที่สามารถควบคุมระบบให้ทำงานได้อย่างมีประสิทธิภาพและเหมาะสม และสามารถทำงานได้ในสภาวะแวดล้อมจริง เพราะว่าสิ่งแวดล้อมที่รอบตัวเรานั้นมีองค์ประกอบหลายอย่างที่เป็นสาเหตุให้ระบบมีการเปลี่ยนแปลง ซึ่งเป็นสาเหตุทำให้การทำงานของระบบมีประสิทธิภาพลดลง สำหรับวิธีที่จะหาค่าพารามิเตอร์ของตัวควบคุมนั้นมีหลายทฤษฎี แต่ในที่นี้จะเลือกใช้ทฤษฎีของ Karl J. Åström ในการค่าพารามิเตอร์ของตัวควบคุม

4.2 การหาทรานเฟอร์ฟังก์ชัน

รูปที่ 4.1 โครงสร้างทางกายภาพ

จากรูปที่ 4.1 เป็นระบบที่จะนำมาทดลองเพื่อหาทรานเฟอร์ฟังก์ชันเพื่อนำไปหาค่าพารามิเตอร์ของตัวควบคุม ซึ่งสามารถหาทรานเฟอร์ฟังก์ชันได้จาก การป้อนสัญญาณขั้นบันได 15 หน่วย เข้าไปในระบบแล้วทำการวัดสัญญาณขาออก (Output) ซึ่งจะมีรูปของกราฟที่ทำการทดสอบออกมาจากระบบจำนวน 3 เอาต์พุตคือ $y1, y2$ และ $y3$ ดังรูปที่ 4.2



รูปที่ 4.2 สัญญาณเอาต์พุตของระบบ

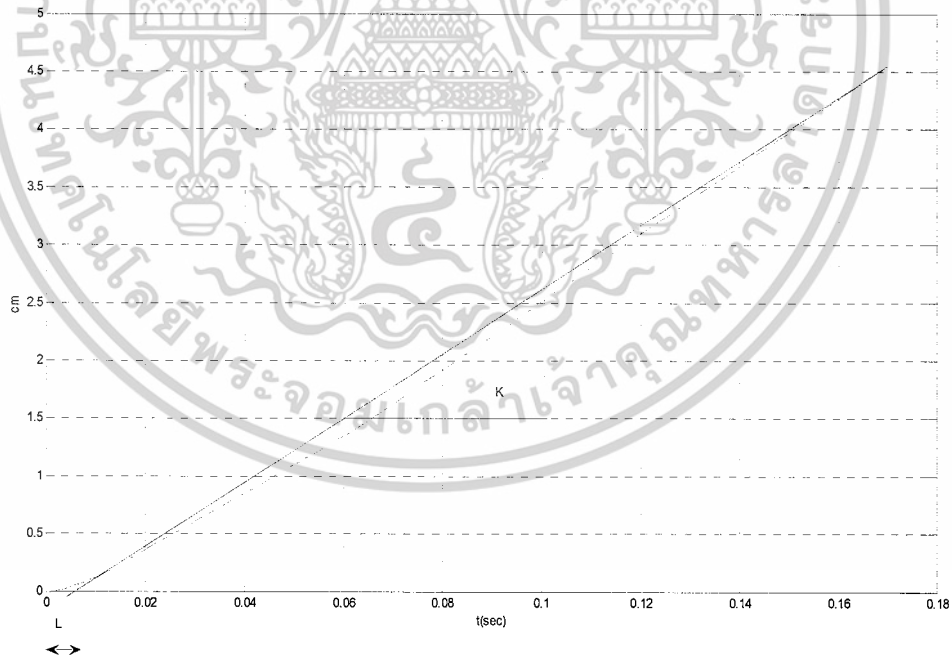
การหาทรานเฟอร์ฟังก์ชันเป็นขั้นตอนอันดับแรกเพื่อจะหาค่าพารามิเตอร์ของตัวควบคุม ซึ่งจากรูปที่ 4.3 จะมีเส้นอยู่ 2 เส้น ก็คือเส้นสีแดงและเส้นสีน้ำเงิน โดยที่เส้นสีน้ำเงินจะเป็นเส้นของการประมาณค่าเพื่อหาทรานเฟอร์ฟังก์ชันของระบบ โดยจะมีค่าดีเลย์ (L) โดยประมาณ 0.005 และค่าความชัน (K) โดยประมาณ 27 ซึ่งจะนำค่าทั้งสองไปแทนในสมการ (4.1) เพื่อจะนำไปหาค่าพารามิเตอร์เพื่อนำไปออกแบบตัวควบคุม ในส่วนของเส้นสีแดงนั้นเป็นทรานเฟอร์ฟังก์ชันที่หาได้จากระบบ

สมการประมาณค่าจะมีฟังก์ชัน ดังนี้

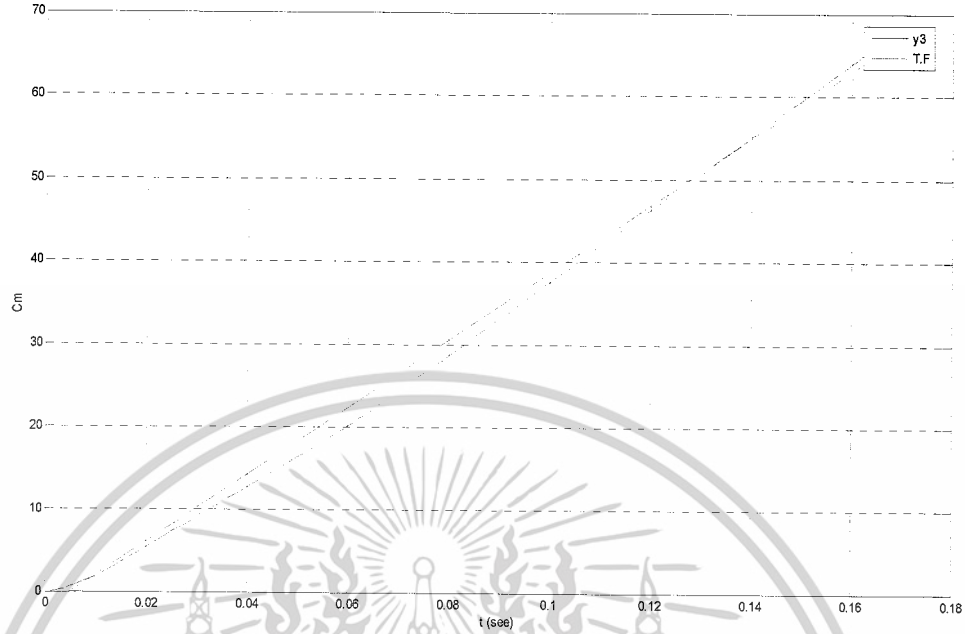
$$G(s) = \frac{Ke^{-sL}}{s} \tag{4.1}$$

นำค่าที่ได้จากการประมาณค่าไปแทนในสมการ (1.1)

$$G(s) = \frac{27e^{-0.005s}}{s} \tag{4.2}$$



รูปที่ 4.3 กราฟการหาทรานเฟอร์ฟังก์ชัน



รูปที่ 4.4 กราฟเปรียบเทียบแสดงเอาต์พุต (y3) และทรานเฟอร์ฟังก์ชัน

4.3 การออกแบบตัวควบคุมพีไอ

สำหรับวิธีการที่จะใช้การออกแบบตัวควบคุมที่ดีนั้น นอกจากจะต้องเป็นวิธีที่สามารถทำให้ระบบควบคุมมีสมรรถนะที่ดีคือ สามารถควบคุมให้ระบบควบคุมมีผลตอบสนองตามที่ได้ออกแบบไว้ มีเสถียรภาพและมีความคงทนต่อการเปลี่ยนแปลงค่าพารามิเตอร์ของกระบวนการในระบบ ในการหาค่าพารามิเตอร์ของตัวควบคุมซึ่งในที่นี้จะทำการทดลองกระบวนการโดยใช้ทฤษฎีของ Karl Johan Åström

4.3.1 ทฤษฎีของ Karl Johan Åström

ในส่วนของทฤษฎีของ Karl J. Åström จะมีการหาค่าพารามิเตอร์มีอยู่ 2 วิธีคือ ตัวควบคุมพีไอสำหรับระบบหนึ่งสัญญาณเข้าหนึ่งสัญญาณออก (One Degree of Freedom Controller) และตัวควบคุมแบบพีไอถ่วงน้ำหนักสัญญาณอ้างอิง (Two Degree of Freedom Controller) ซึ่งจะมีสมการในการหาค่าของพารามิเตอร์ k_p และ k_i นั้นจะมีสมการในการหาค่าของพารามิเตอร์อยู่หลายสมการ ซึ่งค่าทรานเฟอร์ฟังก์ชันในสมการ (4.2) ซึ่งทรานเฟอร์ฟังก์ชั่นจะเป็นอันดับหนึ่งและอยู่ในรูปของอินทิกรัล (Integral) ซึ่งจะมีการวิธีการหาค่าของพารามิเตอร์ k_p และ k_i ดังนี้

$$k_p = \frac{0.35}{KL} \tag{4.3}$$

$$k_i = 7L \tag{4.4}$$

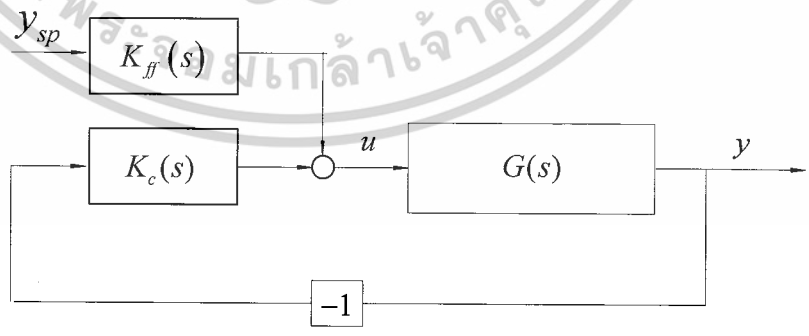
วัตถุประสงค์คือต้องการคำนวณค่า k_i, k_p ซึ่งทำให้ระบบควบคุมมีเสถียรภาพและมีสมรรถนะตามต้องการ โดยทั่วไปเราต้องมีความต้องการหลายอย่างเช่น ผลตอบสนองชั่วครู่ที่ดี สามารถกำจัดค่าความผิดพลาดที่สถานะอยู่ตัวและสามรถจำกัดสัญญาณรบกวนโหลดได้ในเวลาเดียวกันตัวควบคุม

นำทรานเฟอร์ฟังก์ชันในสมการ (4.2) มาหาพารามิเตอร์ k_p และ k_i ในสมการที่ (4.3) และ (4.4) โดยมีวิธีหาค่า ได้ดังนี้

$$k_p = \frac{0.35}{(27 * 0.005)} = 2.59 \tag{4.5}$$

$$k_i = 7(0.005) = 0.035 \tag{4.6}$$

เมื่อได้ค่าพารามิเตอร์ของตัวควบคุม k_p, k_i แล้วนำไปใส่ในบล๊อคไดอะแกรมในรูปที่ ซึ่งเป็นรูปบล๊อคไดอะแกรมของ Two degree of freedom ซึ่งสามารถปรับจูนค่าพารามิเตอร์ b ในบล๊อคไดอะแกรม K_{ff} ซึ่งถ้ากำหนด b ให้เป็น 1 ผลตอบสนองจะมีค่าพุ่งเกินสูงสุดและถ้า b มีค่าเป็น 0 ให้ผลตอบสนองมีค่าพุ่งต่ำสุด ดังนั้นเราสามารถปรับจูนค่าของ b เพื่อปรับปรุงผลตอบสนองสัญญาณ อ้างอิงโดยไม่กระทบต่อผลตอบสนองสัญญาณรบกวนโหลด



รูปที่ 4.5 บล๊อคไดอะแกรมของ Two degree of freedom

ซึ่งในบล๊อคไดอะแกรมของ $K_{ff}(s)$ จะมีสมการอยู่ข้างใน คือ

$$K_{ff}(s) = bk_p + \frac{k_i}{s} \tag{4.7}$$

และในบล๊อคไดอะแกรมของ $K_c(s)$ จะมีสมการอยู่ข้างใน คือ

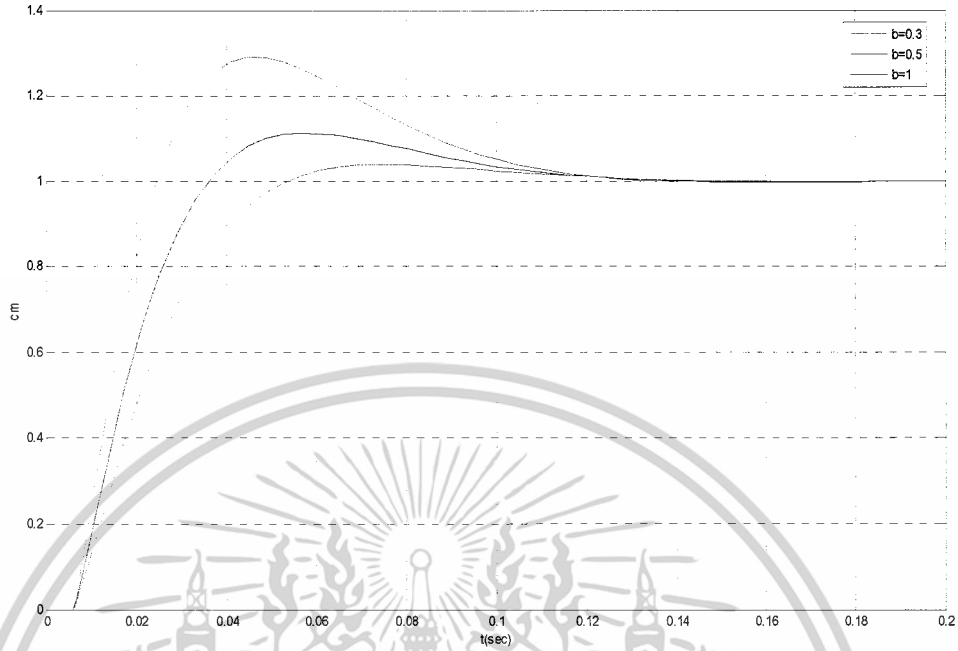
$$K_c(s) = k_p + \frac{k_i}{s} \tag{4.8}$$

สังเกตว่าฟังก์ชั่นถ่ายโอนของตัวควบคุมทั้งสองแตกต่างกันและการออกแบบอิสระต่อกัน สำหรับการออกแบบตัวควบคุม $K_c(s)$ เป็นจุดประสงค์หลัก ส่วนตัวควบคุม $K_{ff}(s)$ ออกแบบหลังจากได้รับตัวควบคุม $K_c(s)$ แล้วโดยการปรับจูนค่าของ b เพื่อปรับปรุงผลตอบสนองต่อสัญญาณอ้างอิง

4.4 ผลการทดลองด้วยโปรแกรมคำนวณทางคณิตศาสตร์

ในโครงงานนี้ได้ทำการออกแบบระบบเพื่อทำการทดสอบทฤษฎีที่ได้ศึกษา ว่าการทำงานจริงๆนั้นเป็นเช่นไร ซึ่งได้ใช้โปรแกรมการคำนวณค่าทางคณิตศาสตร์เข้ามาช่วยในการหาค่าของพารามิเตอร์ของตัวควบคุม ซึ่งสามารถช่วยในการหาค่าได้เร็วขึ้นและถูกต้องแม่นยำ และสามารถแสดงผลออกมาเป็นรูปกราฟ เพื่อดูการทำงานของระบบซึ่งได้เห็นและเรียนรู้จากการเรียนและในหนังสือ

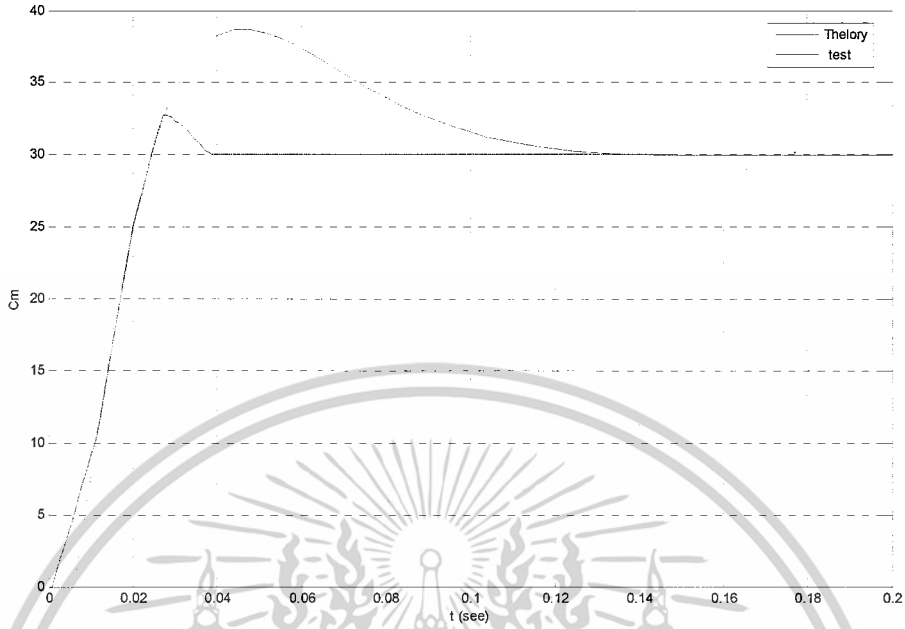
จากรูปที่ 4.5 ซึ่งแสดงรูปบล๊อคไดอะแกรมการทำงานของระบบซึ่งจะนำไปเขียนไว้ในโปรแกรมคำนวณทางคณิตศาสตร์ซึ่งจะช่วยให้สามารถดูรายละเอียดการทำงานของระบบที่ได้ออกแบบเอาไว้ โดยใช้ทฤษฎีของ Karl Johan Åström ในการคำนวณหาค่าพารามิเตอร์ของตัวควบคุมซึ่งสามารถแสดงได้ดังรูปที่ 4.6 ซึ่งแสดงกราฟของทรานเฟอร์ฟังก์ชันที่ได้แทนค่าพารามิเตอร์เข้าไปแล้ว โดยกราฟทั้ง 3 กราฟนั้นได้ปรับจูนค่าของ b แล้ว



รูปที่ 4.6 กราฟแสดงทรานเฟอร์ฟังก์ชันของระบบ

4.5 ผลการทดลองระบบจริง

จากผลการทดลองได้ทำการทดลองระบบจริงๆ โดยใช้ค่าพารามิเตอร์ของตัวควบคุมแบบพีไอ ที่ได้ออกแบบมาในสมการ (4.5) และ (4.6) ไปเขียนไว้ในตัวคอนโทรลเลอร์แล้วนำรูปกราฟที่ได้ไปเปรียบเทียบกับรูปที่ 4.6 ซึ่งเป็นทรานเฟอร์ฟังก์ชันของระบบซึ่งจะได้รูปกราฟออกมาดังรูปที่ 4.7 ซึ่งระบบจริงมีผลของเอาต์พุตคล้ายกับทฤษฎี



รูปที่ 4.7 กราฟระบบจริงกับกราฟทฤษฎี

บทที่ 5

บทวิจารณ์และสรุปผลการทดลอง

5.1 สรุปผลการทดลอง

จากการทดลอง ซึ่งได้สร้างระบบระบบหนึ่งออกมาคือ อุปกรณ์ตรวจวัดความตึงของเชือก ซึ่งได้ใช้ทฤษฎีของ Karl Johan Åström ในการคำนวณหาค่าพารามิเตอร์ของตัวควบคุมแบบ พีไอ เมื่อทำการทดลองได้ทดลองกับระบบโดยป้อนสัญญาณขั้นบันได 15 หน่วยเข้าไปแล้วจะได้ค่าของเอาต์พุตออกมาซึ่งได้ทำการทดลองจำนวน 3 ครั้งแล้วนำค่าเอาต์พุตของระบบ 3 เอาต์พุตมาเปรียบกันซึ่งจะได้เอาต์พุตออกมาที่มีค่าที่ใกล้เคียงกัน จึงทำการเลือกเอาต์พุตออกมา 1 ค่าเพื่อนำไปหาค่าพารามิเตอร์ของตัวควบคุมโดยใช้โปรแกรมคำนวณทางคณิตศาสตร์เข้ามาช่วยในการหาค่า เมื่อได้ค่าพารามิเตอร์แล้วนำไปหาทรานเฟอร์ฟังก์ชันโดยใช้ทฤษฎีของ Karl Johan Åström ในการหาค่าซึ่งจะได้รูปกราฟออกมาแล้วนำเปรียบเทียบกับค่าเอาต์พุต(y3)ที่ได้หาในขั้นตอนแรกจะได้กราฟออกมาใกล้เคียงกันแล้วนำไปเขียนในชุดไมโครคอนโทรลเลอร์เพื่อนำไปควบคุมระบบจริง ซึ่งผลจากการทดลองรูปกราฟของระบบจริงกับกราฟของทฤษฎีมีความคล้ายกัน ซึ่งมีค่าความผิดพลาดบ้างเล็กน้อยที่สามารถยอมรับได้

5.2 ปัญหาที่พบและแนวทางแก้ไข

ในการทดลองออกแบบระบบควบคุมแบบ พีไอ เมื่อเปรียบเทียบกับค่าที่ทำการประมวลผลทางโปรแกรมคำนวณทางคณิตศาสตร์ (Matlab) กับค่าที่ทำการทดลองจริงกับระบบ ผลการทดลองบางครั้งผลที่ออกมาไม่เป็นไปตามโปรแกรมคำนวณทางคณิตศาสตร์ (Matlab) หรือตามทฤษฎี ซึ่งทำให้ต้องเสียเวลาในการปรับแต่งตัวควบคุม ดังนั้นจำเป็นต้องศึกษาทฤษฎีให้ลึกซึ้งและเข้าใจเกี่ยวกับทฤษฎีที่ใช้ออกแบบตัวควบคุมแบบ พีไอ ตลอดจนโครงสร้างของระบบที่ใช้การทดลองและภาคไดร์ที่คอยขับให้มอเตอร์กระแสตรงมีปัญหาในบางครั้ง ทำให้เกิดผลกระทบต่อระบบในการทำงานซึ่งทำให้ผลการทดลองออกมาผิดพลาด จำเป็นต้องออกแบบภาคไดร์มอเตอร์ใหม่เพื่อให้สามารถทำงานได้เป็นปกติ

5.3 ประโยชน์ที่ได้รับจากโครงงาน

1. สามารถนำความรู้ที่ได้ศึกษาในการออกแบบตัวควบคุมแบบ พีไอ และประสบการณ์มาประยุกต์ใช้ในการทำงานจริง

2. ได้เห็นภาพจริงของการทำงานของตัวควบคุมแบบ พีไอ จากที่ได้เห็นในทางอุดมคติในหนังสือ

3. ฝึกและเรียนรู้หาความรู้ใหม่ๆ มาประยุกต์ใช้ในการทำงาน

4. สามารถทำงานร่วมกันเป็นทีมให้มีประสิทธิภาพสูงสุด และทำให้รู้จักรับผิดชอบหน้าที่การงานที่ได้รับมอบหมายให้ไปทำหรือจัดซื้ออุปกรณ์

5. ได้เรียนรู้อุปสรรคและแนวทางการแก้ไขปัญหาที่เกิดขึ้นในการทำงาน ซึ่งสามารถนำไปประยุกต์ใช้ในการทำงานจริงได้

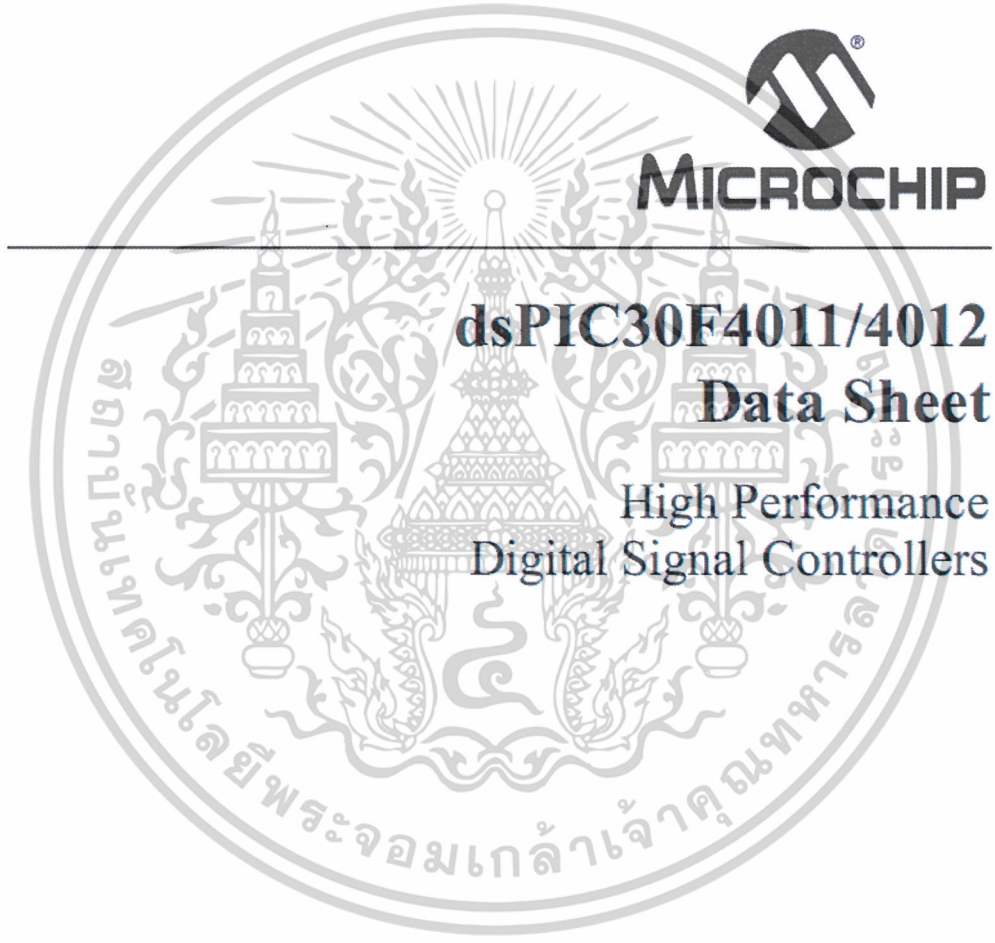
5.4 ข้อเสนอแนะและแนวทางในการค้นคว้าพัฒนา

ข้อเสนอแนะสำหรับการพัฒนาโครงงานในอนาคตคือ ประยุกต์การใช้งานระบบควบคุมแบบ พีไอ กับระบบการทำงานอื่นๆ และนำโครงสร้างของระบบในโครงงานนี้ไปใช้กับตัวควบคุมอื่นๆ เช่น ระบบฟัซซี ซึ่งจะส่งผลทำให้เกิดมีการพัฒนางานในระบบควบคุมต่อไป

ภาคผนวก

ภาคผนวก ก

ไมโครคอนโทรลเลอร์



MICROCHIP

dsPIC30F4011/4012

dsPIC30F4011/4012 Enhanced Flash 16-bit Digital Signal Controller

Note: This data sheet summarizes features of this group of dsPIC30F devices and is not intended to be a complete reference source. For more information on the CPU, peripherals, register descriptions and general device functionality, refer to the *dsPIC30F Family Reference Manual* (DS70046). For more information on the device instruction set and programming, refer to the *dsPIC30F Programmer's Reference Manual* (DS70030).

High Performance Modified RISC CPU:

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

DSP Engine Features:

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

Peripheral Features:

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

Motor Control PWM Module Features:

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

Quadrature Encoder Interface Module Features:

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

dsPIC30F4011/4012

Analog Features:

- 10-bit Analog-to-Digital Converter (A/D) with 4 S/H Inputs:
  - 500 Ksps conversion rate
  - 9 input channels
  - Conversion available during Sleep and Idle
- Programmable Brown-out Detection and Reset generation

Special Microcontroller Features:

- Enhanced Flash program memory:
  - 10,000 erase/write cycle (min.) for industrial temperature range, 100K (typical)
- Data EEPROM memory:
  - 100,000 erase/write cycle (min.) for industrial temperature range, 1M (typical)
- Self-reprogrammable under software control
- Power-on Reset (POR), Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Flexible Watchdog Timer (WDT) with on-chip low power RC oscillator for reliable operation
- Fail-Safe clock monitor operation detects clock failure and switches to on-chip low power RC oscillator
- Programmable code protection
- In-Circuit Serial Programming™ (ICSP™)
- Selectable Power Management modes
  - Sleep, Idle and Alternate Clock modes

CMOS Technology:

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

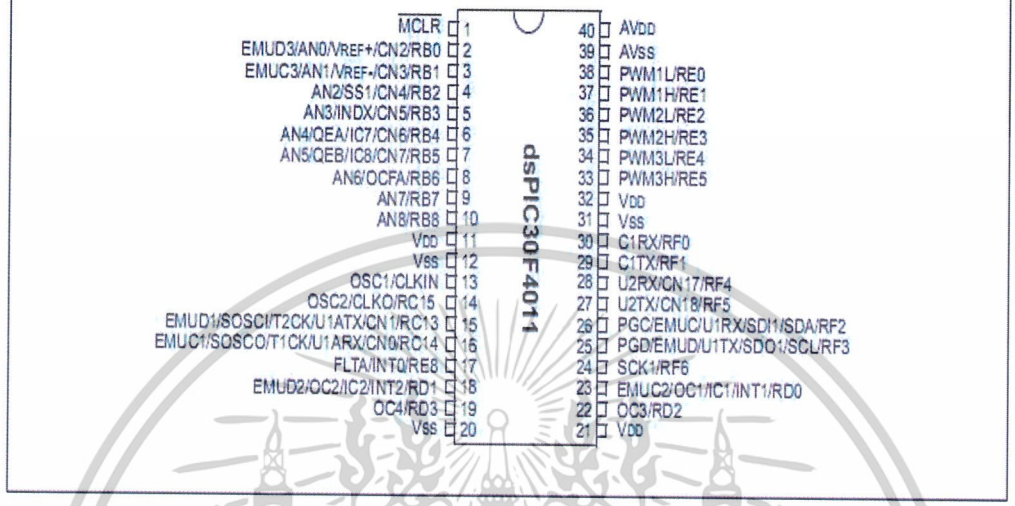
dsPIC30F Motor Control and Power Conversion Family*

Device	Pins	Program Mem. Bytes/ Instructions	SRAM Bytes	EEPROM Bytes	Timer 16-bit	Input Cap	Output Comp/Std PWM	Moto Control PWM	A/D 10-bit 500 Ksps	Quad Enc	UART	SPI™	I²C™	CAN
dsPIC30F2010	28	12K/4K	512	1024	3	4	2	6 ch	6 ch	Yes	1	1	1	-
dsPIC30F3010	28	24K/8K	1024	1024	5	4	2	6 ch	6 ch	Yes	1	1	1	-
dsPIC30F4012	28	48K/16K	2048	1024	5	4	2	6 ch	6 ch	Yes	1	1	1	1
dsPIC30F3011	40/44	24K/8K	1024	1024	5	4	4	6 ch	9 ch	Yes	2	1	1	-
dsPIC30F4011	40/44	48K/16K	2048	1024	5	4	4	6 ch	9 ch	Yes	2	1	1	1
dsPIC30F5015	64	66K/22K	2048	1024	5	4	4	8 ch	16 ch	Yes	1	2	1	1
dsPIC30F6010	80	144K/48K	8192	4096	5	8	8	8 ch	16 ch	Yes	2	2	1	2

* This table provides a summary of the dsPIC30F6010 peripheral features. Other available devices in the dsPIC30F Motor Control and Power Conversion Family are shown for feature comparison.

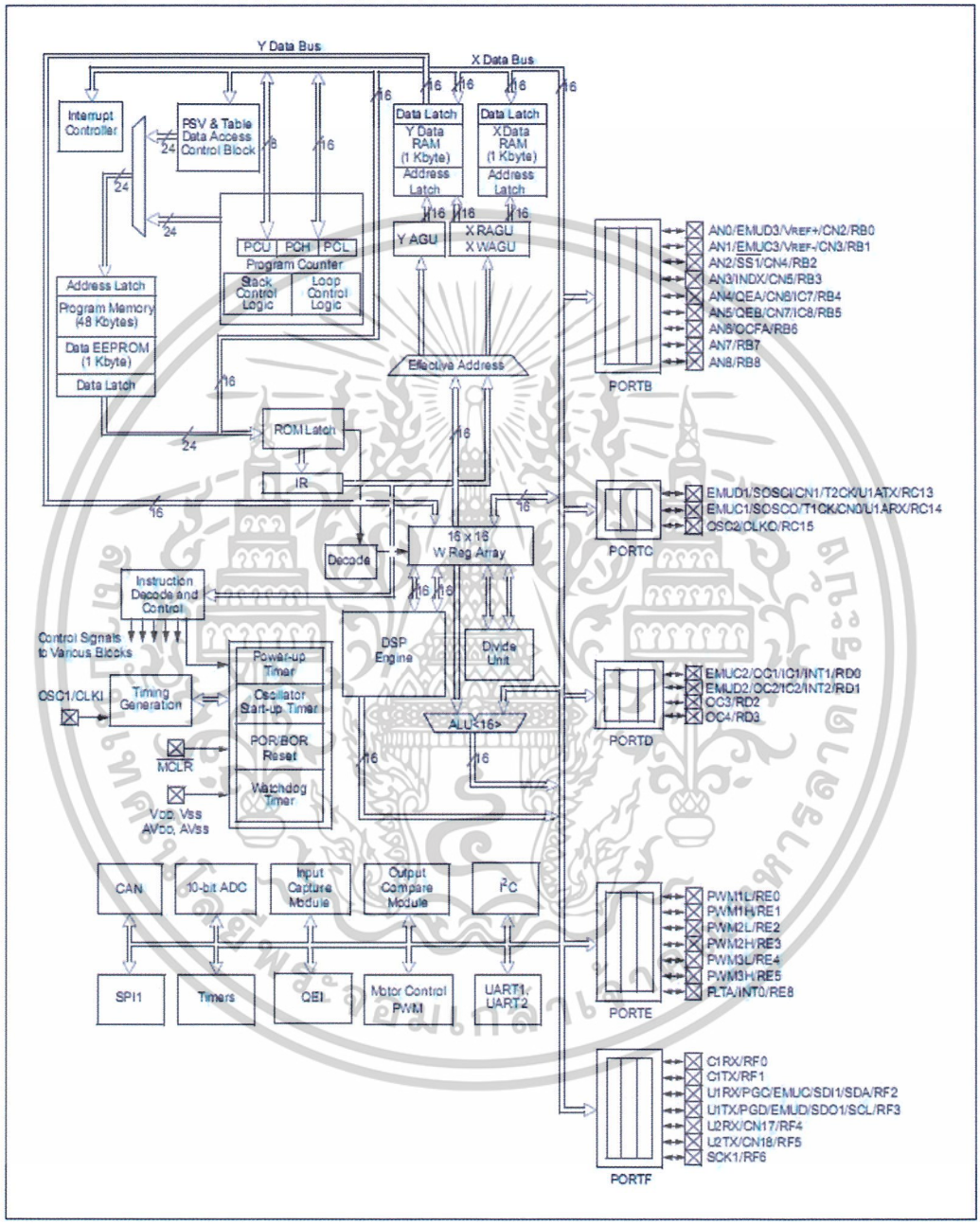
Pin Diagrams

40-Pin PDIP



dsPIC30F4011/4012

FIGURE 1-1: dsPIC30F4011 BLOCK DIAGRAM



5.0 INTERRUPTS

Note: This data sheet summarizes features of this group of dsPIC30F devices and is not intended to be a complete reference source. For more information on the CPU, peripherals, register descriptions and general device functionality, refer to the *dsPIC30F Family Reference Manual* (DS70046). For more information on the device instruction set and programming, refer to the *dsPIC30F Programmer's Reference Manual* (DS70030).

The dsPIC30F4011/4012 has 30 interrupt sources and 4 processor exceptions (traps), which must be arbitrated based on a priority scheme.

The CPU is responsible for reading the Interrupt Vector Table (IVT) and transferring the address contained in the interrupt vector to the program counter. The interrupt vector is transferred from the program data bus into the program counter, via a 24-bit wide multiplexer on the input of the program counter.

The Interrupt Vector Table (IVT) and Alternate Interrupt Vector Table (AIVT) are placed near the beginning of program memory (0x000004). The IVT and AIVT are shown in Figure 5-1.

The interrupt controller is responsible for preprocessing the interrupts and processor exceptions, prior to their being presented to the processor core. The peripheral interrupts and traps are enabled, prioritized and controlled using centralized special function registers:

- IFS0<15:0>, IFS1<15:0>, IFS2<15:0>
  All interrupt request flags are maintained in these three registers. The flags are set by their respective peripherals or external signals, and they are cleared via software.
- IEC0<15:0>, IEC1<15:0>, IEC2<15:0>
  All Interrupt Enable Control bits are maintained in these three registers. These control bits are used to individually enable interrupts from the peripherals or external signals.
- IPC0<15:0>... IPC11<7:0>
  The user assignable priority level associated with each of these interrupts is held centrally in these twelve registers.
- IPL<3:0> The current CPU priority level is explicitly stored in the IPL bits. IPL<3> is present in the CORCON register, whereas IPL<2:0> are present in the status register (SR) in the processor core.
- INTCON1<15:0>, INTCON2<15:0>
  Global interrupt control functions are derived from these two registers. INTCON1 contains the control and status flags for the processor exceptions. The INTCON2 register controls the external interrupt request signal behavior and the use of the alternate vector table.

Note: Interrupt Flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding Enable bit. User software should ensure the appropriate Interrupt Flag bits are clear prior to enabling an interrupt.

All interrupt sources can be user assigned to one of 7 priority levels, 1 through 7, via the IPCx registers. Each interrupt source is associated with an interrupt vector, as shown in Table 5-1. Levels 7 and 1 represent the highest and lowest maskable priorities, respectively.

Note: Assigning a priority level of 0 to an interrupt source is equivalent to disabling that interrupt.

If the NSTDIS bit (INTCON1<15>) is set, nesting of interrupts is prevented. Thus, if an interrupt is currently being serviced, processing of a new interrupt is prevented, even if the new interrupt is of higher priority than the one currently being serviced.

Note: The IPL bits become read-only whenever the NSTDIS bit has been set to '1'.

Certain interrupts have specialized control bits for features like edge or level triggered interrupts, interrupt-on-change, etc. Control of these features remains within the peripheral module which generates the interrupt.

The `DISI` instruction can be used to disable the processing of interrupts of priorities 6 and lower for a certain number of instructions, during which the DISI bit (INTCON2<14>) remains set.

When an interrupt is serviced, the PC is loaded with the address stored in the vector location in Program Memory that corresponds to the interrupt. There are 63 different vectors within the IVT (refer to Figure 5-2). These vectors are contained in locations 0x000004 through 0x0000FE of program memory (refer to Figure 5-2). These locations contain 24-bit addresses, and in order to preserve robustness, an address error trap will take place should the PC attempt to fetch any of these words during normal execution. This prevents execution of random data as a result of accidentally decrementing a PC into vector space, accidentally mapping a data space address into vector space, or the PC rolling over to 0x000000 after reaching the end of implemented program memory space. Execution of a `GOTO` instruction to this vector space will also generate an address error trap.

5.1 Interrupt Priority

The user assignable Interrupt Priority (IP<2:0>) bits for each individual interrupt source are located in the LS 3-bits of each nibble, within the IPCx register(s). Bit 3 of each nibble is not used and is read as a '0'. These bits define the priority level assigned to a particular interrupt by the user.

Note: The user selectable priority levels start at 0, as the lowest priority, and level 7, as the highest priority.

Since more than one interrupt request source may be assigned to a specific user specified priority level, a means is provided to assign priority within a given level. This method is called "Natural Order Priority".

Natural Order Priority is determined by the position of an interrupt in the vector table, and only affects interrupt operation when multiple interrupts with the same user-assigned priority become pending at the same time.

Table 5-1 lists the interrupt numbers and interrupt sources for the dsPIC devices and their associated vector numbers.

Note 1: The natural order priority scheme has 0 as the highest priority and 53 as the lowest priority.

2: The natural order priority number is the same as the INT number.

The ability for the user to assign every interrupt to one of seven priority levels implies that the user can assign a very high overall priority level to an interrupt with a low natural order priority. For example, the PLVD (Low Voltage Detect) can be given a priority of 7. The INT0 (external interrupt 0) may be assigned to priority level 1, thus giving it a very low effective priority.

TABLE 5-1: INTERRUPT VECTOR TABLE

INT Number	Vector Number	Interrupt Source
Highest Natural Order Priority		
0	8	INT0 - External Interrupt 0
1	9	IC1 - Input Capture 1
2	10	OC1 - Output Compare 1
3	11	T1 - Timer 1
4	12	IC2 - Input Capture 2
5	13	OC2 - Output Compare 2
6	14	T2 - Timer 2
7	15	T3 - Timer 3
8	16	SPI1
9	17	U1RX - UART1 Receiver
10	18	U1TX - UART1 Transmitter
11	19	ADC - ADC Convert Done
12	20	NVM - NVM Write Complete
13	21	SI2C - I^2C Slave Interrupt
14	22	MI2C - I^2C Master Interrupt
15	23	Input Change Interrupt
16	24	INT1 - External Interrupt 1
17	25	IC7 - Input Capture 7
18	26	IC8 - Input Capture 8
19	27	OC3 - Output Compare 3
20	28	OC4 - Output Compare 4
21	29	T4 - Timer 4
22	30	T5 - Timer 5
23	31	INT2 - External Interrupt 2
24	32	U2RX - UART2 Receiver
25	33	U2TX - UART2 Transmitter
26	34	Reserved
27	35	C1 - Combined IRQ for CAN1
28	36	Reserved
29	37	Reserved
30	38	Reserved
31	39	Reserved
32	40	Reserved
33	41	Reserved
34	42	Reserved
35	43	Reserved
36	44	Reserved
37	45	Reserved
38	46	Reserved
39	47	PWM - PWM Period Match
40	48	QEI - QEI Interrupt
41	49	Reserved
42	50	Reserved
43	51	FLTA - PWM Fault A
44	52	Reserved
45-53	53-61	Reserved
Lowest Natural Order Priority		

5.2 Reset Sequence

A Reset is not a true exception, because the interrupt controller is not involved in the Reset process. The processor initializes its registers in response to a Reset, which forces the PC to zero. The processor then begins program execution at location 0x000000. A GOTO instruction is stored in the first program memory location, immediately followed by the address target for the GOTO instruction. The processor executes the GOTO to the specified address and then begins operation at the specified target (start) address.

5.2.1 RESET SOURCES

There are 6 sources of error which will cause a device reset.

- Watchdog Time-out:
  The watchdog has timed out, indicating that the processor is no longer executing the correct flow of code.
- Uninitialized W Register Trap:
  An attempt to use an uninitialized W register as an address pointer will cause a Reset.
- Illegal Instruction Trap:
  Attempted execution of any unused opcodes will result in an illegal instruction trap. Note that a fetch of an illegal instruction does not result in an illegal instruction trap if that instruction is flushed prior to execution due to a flow change.
- Brown-out Reset (BOR):
  A momentary dip in the power supply to the device has been detected, which may result in malfunction.
- Trap Lockout:
  Occurrence of multiple Trap conditions simultaneously will cause a Reset.

5.3 Traps

Traps can be considered as non-maskable interrupts indicating a software or hardware error, which adhere to a predefined priority as shown in Figure 5-1. They are intended to provide the user a means to correct erroneous operation during debug and when operating within the application.

Note:	If the user does not intend to take corrective action in the event of a trap error condition, these vectors must be loaded with the address of a default handler that simply contains the RESET instruction. If, on the other hand, one of the vectors containing an invalid address is called, an address error trap is generated.

Note that many of these trap conditions can only be detected when they occur. Consequently, the questionable instruction is allowed to complete prior to trap exception processing. If the user chooses to recover from the error, the result of the erroneous action that caused the trap may have to be corrected.

There are 8 fixed priority levels for traps: Level 8 through Level 15, which implies that the IPL3 is always set during processing of a trap.

If the user is not currently executing a trap, and he sets the IPL<3:0> bits to a value of '0111' (Level 7), then all interrupts are disabled, but traps can still be processed.

5.3.1 TRAP SOURCES

The following traps are provided with increasing priority. However, since all traps can be nested, priority has little effect.

Math Error Trap:

The Math Error trap executes under the following three circumstances:

1. Should an attempt be made to divide by zero, the divide operation will be aborted on a cycle boundary and the trap taken.
2. If enabled, a Math Error trap will be taken when an arithmetic operation on either accumulator A or B causes an overflow from bit 31 and the Accumulator Guard bits are not utilized.
3. If enabled, a Math Error trap will be taken when an arithmetic operation on either accumulator A or B causes a catastrophic overflow from bit 39 and all saturation is disabled.
4. If the shift amount specified in a shift instruction is greater than the maximum allowed shift amount, a trap will occur.

Address Error Trap:

This trap is initiated when any of the following circumstances occurs:

1. A misaligned data word access is attempted.
2. A data fetch from our unimplemented data memory location is attempted.
3. A data access of an unimplemented program memory location is attempted.
4. An instruction fetch from vector space is attempted.

Note:	In the MAC class of instructions, wherein the data space is split into X and Y data space, unimplemented X space includes all of Y space, and unimplemented Y space includes all of X space.

5. Execution of a "BRA #literal" instruction or a "GOTO #literal" instruction, where literal is an unimplemented program memory address.
6. Executing instructions after modifying the PC to point to unimplemented program memory addresses. The PC may be modified by loading a value into the stack and executing a RETURN instruction.

Stack Error Trap:

This trap is initiated under the following conditions:

1. The stack pointer is loaded with a value which is greater than the (user programmable) limit value written into the SPLIM register (stack overflow).
2. The stack pointer is loaded with a value which is less than 0x0800 (simple stack underflow).

Oscillator Fail Trap:

This trap is initiated if the external oscillator fails and operation becomes reliant on an internal RC backup.

5.3.2 HARD AND SOFT TRAPS

It is possible that multiple traps can become active within the same cycle (e.g., a misaligned word stack write to an overflowed address). In such a case, the fixed priority shown in Figure 5-2 is implemented, which may require the user to check if other traps are pending, in order to completely correct the fault.

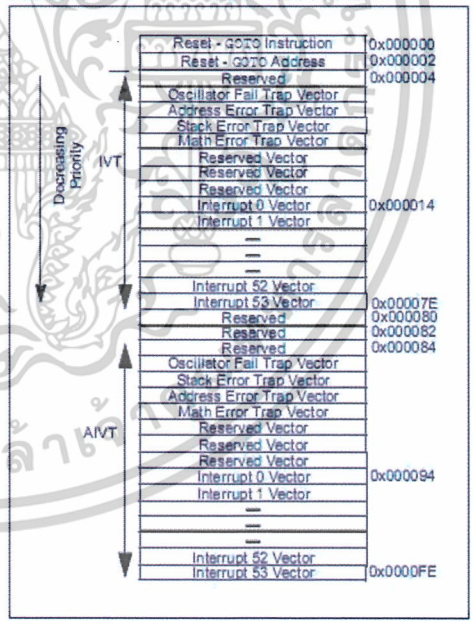
'Soft' traps include exceptions of priority level 8 through level 11, inclusive. The arithmetic error trap (level 11) falls into this category of traps.

'Hard' traps include exceptions of priority level 12 through level 15, inclusive. The address error (level 12), stack error (level 13) and oscillator error (level 14) traps fall into this category.

Each hard trap that occurs must be acknowledged before code execution of any type may continue. If a lower priority hard trap occurs while a higher priority trap is pending, acknowledged, or is being processed, a hard trap conflict will occur.

The device is automatically Reset in a hard trap conflict condition. The TRAPR status bit (RCON<15>) is set when the Reset occurs, so that the condition may be detected in software.

FIGURE 5-1: TRAP VECTORS



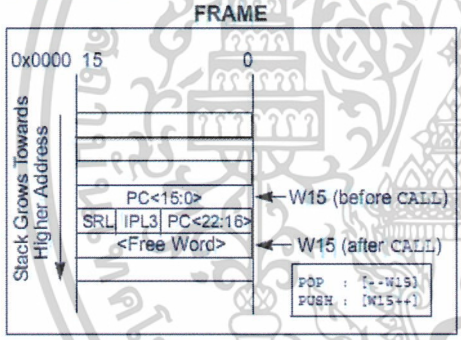
5.4 Interrupt Sequence

All interrupt event flags are sampled in the beginning of each instruction cycle by the IFSx registers. A pending interrupt request (IRQ) is indicated by the flag bit being equal to a '1' in an IFSx register. The IRQ will cause an interrupt to occur if the corresponding bit in the interrupt enable (IECx) register is set. For the remainder of the instruction cycle, the priorities of all pending interrupt requests are evaluated.

If there is a pending IRQ with a priority level greater than the current processor priority level in the IPL bits, the processor will be interrupted.

The processor then stacks the current program counter and the low byte of the processor status register (SRL), as shown in Figure 5-2. The low byte of the status register contains the processor priority level at the time, prior to the beginning of the interrupt cycle. The processor then loads the priority level for this interrupt into the status register. This action will disable all lower priority interrupts until the completion of the Interrupt Service Routine.

FIGURE 5-2: INTERRUPT STACK FRAME



Note 1: The user can always lower the priority level by writing a new value into SR. The Interrupt Service Routine must clear the interrupt flag bits in the IFSx register before lowering the processor interrupt priority, in order to avoid recursive interrupts.

2: The IPL3 bit (CORCON<3>) is always clear when interrupts are being processed. It is set only during execution of traps.

The RETFIE (Return from Interrupt) instruction will unstack the program counter and status registers to return the processor to its state prior to the interrupt sequence.

5.5 Alternate Vector Table

In Program Memory, the Interrupt Vector Table (IVT) is followed by the Alternate Interrupt Vector Table (AIVT), as shown in Figure 5-1. Access to the Alternate Vector Table is provided by the ALTIVT bit in the INTCON2 register. If the ALTIVT bit is set, all interrupt and exception processes will use the alternate vectors instead of the default vectors. The alternate vectors are organized in the same manner as the default vectors. The AIVT supports emulation and debugging efforts by providing a means to switch between an application and a support environment, without requiring the interrupt vectors to be reprogrammed. This feature also enables switching between applications for evaluation of different software algorithms at run time.

If the AIVT is not required, the program memory allocated to the AIVT may be used for other purposes. AIVT is not a protected section and may be freely programmed by the user.

5.6 Fast Context Saving

A context saving option is available using shadow registers. Shadow registers are provided for the DC, N, OV, Z and C bits in SR, and the registers W0 through W3. The shadows are only one level deep. The shadow registers are accessible using the PUSH.S and POP.S instructions only.

When the processor vectors to an interrupt, the PUSH.S instruction can be used to store the current value of the aforementioned registers into their respective shadow registers.

If an ISR of a certain priority uses the PUSH.S and POP.S instructions for fast context saving, then a higher priority ISR should not include the same instructions. Users must save the key registers in software during a lower priority interrupt, if the higher priority ISR uses fast context saving.

5.7 External Interrupt Requests

The interrupt controller supports five external interrupt request signals, INT0-INT4. These inputs are edge sensitive; they require a low-to-high or a high-to-low transition to generate an interrupt request. The INTCON2 register has five bits, INT0EP-INT4EP, that select the polarity of the edge detection circuitry.

5.8 Wake-up from Sleep and Idle

The interrupt controller may be used to wake up the processor from either Sleep or Idle modes, if Sleep or Idle mode is active when the interrupt is generated.

If an enabled interrupt request of sufficient priority is received by the interrupt controller, then the standard interrupt request is presented to the processor. At the same time, the processor will wake-up from Sleep or Idle and begin execution of the Interrupt Service Routine (ISR) needed to process the interrupt request.

TABLE 5-2: INTERRUPT CONTROLLER REGISTER MAP

SFR Name	ADR	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State
INTCON1	0080	NSTDIS	—	—	—	—	OVATE	OVBTE	COVTE	—	—	—	MATHERR	ADDRERR	STKERR	OSCFAIL	—	0000 0000 0000 0000
INTCON2	0082	ALTIVT	—	—	—	—	—	—	—	—	—	—	—	—	INT2EP	INT1EP	INT0EP	0000 0000 0000 0000
IFS0	0084	CNIF	MI2CIF	SI2CIF	NVMIF	ADIF	U1TXIF	U1RXIF	SPI1IF	T3IF	T2IF	OC2IF	IC2IF	T1IF	OC1IF	IC1IF	INT0IF	0000 0000 0000 0000
IFS1	0086	—	—	—	—	C1IF	—	U2TXIF	U2RXIF	INT2IF	T5IF	T4IF	OC4IF	OC3IF	IC8IF	IC7IF	INT1IF	0000 0000 0000 0000
IFS2	0088	—	—	—	—	FLTAIF	—	—	QEIIF	PWMIF	—	—	—	—	—	—	—	0000 0000 0000 0000
IEC0	008C	CNIE	MI2CIE	SI2CIE	NVMIE	ADIE	U1TXIE	U1RXIE	SPI1IE	T3IE	T2IE	OC2IE	IC2IE	T1IE	OC1IE	IC1IE	INT0IE	0000 0000 0000 0000
IEC1	008E	—	—	—	—	C1IE	—	U2TXIE	U2RXIE	INT2IE	T5IE	T4IE	OC4IE	OC3IE	IC8IE	IC7IE	INT1IE	0000 0000 0000 0000
IEC2	0090	—	—	—	—	FLTAIE	—	—	QEIIE	PWMIE	—	—	—	—	—	—	—	0000 0000 0000 0000
IPC0	0094	—	T1IP<2:0>			—	OC1IP<2:0>			—	IC1IP<2:0>			—	INT0IP<2:0>			0100 0100 0100 0100
IPC1	0096	—	T31P<2:0>			—	T2IP<2:0>			—	OC2IP<2:0>			—	IC2IP<2:0>			0100 0100 0100 0100
IPC2	0098	—	ADIP<2:0>			—	U1TXIP<2:0>			—	U1RXIP<2:0>			—	SPI1IP<2:0>			0100 0100 0100 0100
IPC3	009A	—	CNIP<2:0>			—	MI2CIP<2:0>			—	SI2CIP<2:0>			—	NVMIP<2:0>			0100 0100 0100 0100
IPC4	009C	—	OC3IP<2:0>			—	IC8IP<2:0>			—	IC7IP<2:0>			—	INT1IP<2:0>			0100 0100 0100 0100
IPC5	009E	—	INT2IP<2:0>			—	T5IP<2:0>			—	T4IP<2:0>			—	OC4IP<2:0>			0100 0100 0100 0100
IPC6	00A0	—	C1IP<2:0>			—	—			—	U2TXIP<2:0>			—	U2RXIP<2:0>			0100 0000 0100 0100
IPC7	00A2	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000 0000 0000 0000
IPC8	00A4	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000 0000 0000 0000
IPC9	00A6	—	PWMIP<2:0>			—	—			—	—			—	—		—	0100 0000 0100 0100
IPC10	00A8	—	FLTAIP<2:0>			—	—			—	—			—	QEIIP<2:0>			0100 0000 0000 0100
IPC11	00AA	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000 0000 0000 0000

Legend: u = uninitialized bit

Note: Refer to *dsPIC30F Family Reference Manual* (DS70046) for descriptions of register bit fields.

14.0 QUADRATURE ENCODER INTERFACE (QEI) MODULE

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

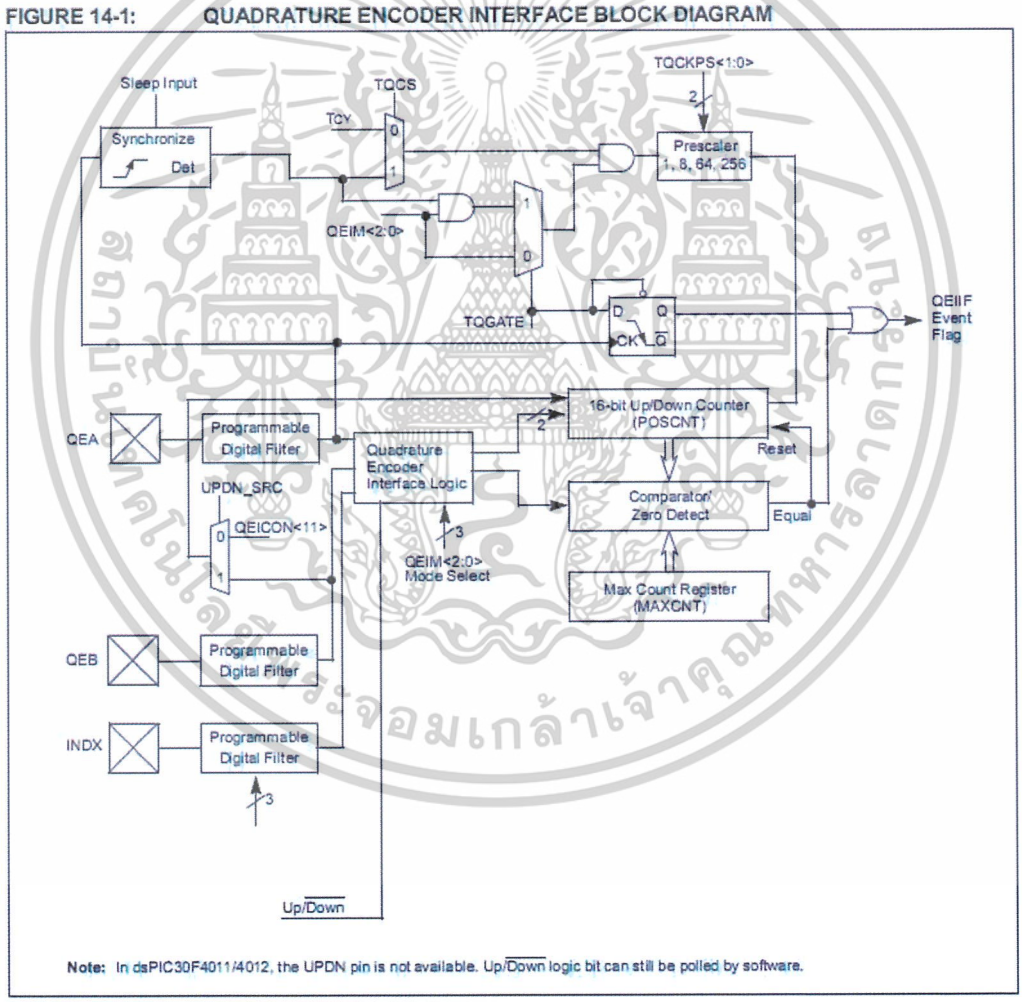
This section describes the Quadrature Encoder Interface (QEI) module and associated operational modes. The QEI module provides the interface to incremental encoders for obtaining mechanical position data.

The operational features of the QEI include:

- Three input channels for two phase signals and index pulse
- 16-bit up/down position counter
- Count direction status
- Position Measurement (x2 and x4) mode
- Programmable digital noise filters on inputs
- Alternate 16-bit Timer/Counter mode
- Quadrature Encoder Interface interrupts

These operating modes are determined by setting the appropriate bits QEIM<2:0> (QEICON<10:8>). Figure 14-1 depicts the Quadrature Encoder Interface block diagram.

FIGURE 14-1: QUADRATURE ENCODER INTERFACE BLOCK DIAGRAM



Note: In dsPIC30F4011/4012, the UPDN pin is not available. Up/$\overline{\text{Down}}$ logic bit can still be polled by software.

14.1 Quadrature Encoder Interface Logic

A typical incremental (a.k.a. optical) encoder has three outputs: Phase A, Phase B, and an index pulse. These signals are useful and often required in position and speed control of ACIM and SR motors.

The two channels, Phase A (QEA) and Phase B (QEB), have a unique relationship. If Phase A leads Phase B, then the direction (of the motor) is deemed positive or forward. If Phase A lags Phase B, then the direction (of the motor) is deemed negative or reverse.

A third channel, termed index pulse, occurs once per revolution and is used as a reference to establish an absolute position. The index pulse coincides with Phase A and Phase B, both low.

14.2 16-bit Up/Down Position Counter Mode

The 16-bit Up/Down Counter counts up or down on every count pulse, which is generated by the difference of the Phase A and Phase B input signals. The counter acts as an integrator, whose count value is proportional to position. The direction of the count is determined by the UPDN signal, which is generated by the Quadrature Encoder Interface Logic.

14.2.1 POSITION COUNTER ERROR CHECKING

Position count error checking in the QEI is provided for and indicated by the CNTERR bit (QEICON<15>). The error checking only applies when the position counter is configured for Reset on the Index Pulse modes (QEIM<2:0> = '110' or '100'). In these modes, the contents of the POSCNT register is compared with the values (0xFFFF or MAXCNT+1, depending on direction). If these values are detected, an error condition is generated by setting the CNTERR bit and a QEI count error interrupt is generated. The QEI count error interrupt can be disabled by setting the CEID bit (DFLTCON<8>). The position counter continues to count encoder edges after an error has been detected. The POSCNT register continues to count up/down until a natural rollover/underflow. No interrupt is generated for the natural rollover/underflow event. The CNTERR bit is a Read/Write bit and reset in software by the user.

14.2.2 POSITION COUNTER RESET

The Position Counter Reset Enable bit, POSRES (QEI<2>) controls whether the position counter is reset when the index pulse is detected. This bit is only applicable when QEIM<2:0> = '100' or '110'.

If the POSRES bit is set to '1', then the position counter is reset when the index pulse is detected. If the POSRES bit is set to '0', then the position counter is not reset when the index pulse is detected. The position counter will continue counting up or down, and will be reset on the rollover or underflow condition.

When selecting the INDX signal to reset the position counter (POSCNT), the user has to specify the states on QEA and QEB input pins. These states have to be matched in order for a reset to occur. These states are selected by the IMV<1:0> bit in the DFLTCON <10:9> register.

The IMV<1:0> (Index Match Value) bit allows the user to specify the state of the QEA and QEB input pins during an Index pulse when the POSCNT register is to be reset.

In 4X Quadrature Count Mode:
IMV1 = Required State of Phase B input signal for match on index pulse
IMV0 = Required State of Phase A input signal for match on index pulse

In 2X Quadrature Count Mode:
IMV1 = Selects Phase input signal for Index state match (0 = Phase A, 1 = Phase B)
IMV0 = Required State of the selected Phase input signal for match on index pulse

The interrupt is still generated on the detection of the index pulse and not on the position counter overflow/underflow.

14.2.3 COUNT DIRECTION STATUS

As mentioned in the previous section, the QEI logic generates an UPDN signal, based upon the relationship between Phase A and Phase B. In addition to the output pin, the state of this internal UPDN signal is supplied to a SFR bit UPDN (QEICON<11>) as a read only bit.

Note: QEI pins are multiplexed with analog inputs. User must insure that all QEI associated pins are set as digital inputs in the ADPCFG register.

14.3 Position Measurement Mode

There are two Measurement modes which are supported and are termed x2 and x4. These modes are selected by the QEIM<2:0> mode select bits located in SFR QEICON<10:8>.

When control bits QEIM<2:0> = 100 or 101, the x2 Measurement mode is selected and the QEI logic only looks at the Phase A input for the position counter increment rate. Every rising and falling edge of the Phase A signal causes the position counter to be incremented or decremented. The Phase B signal is still utilized for the determination of the counter direction, just as in the x4 mode.

Within the x2 Measurement mode, there are two variations of how the position counter is reset:

1. Position counter reset by detection of index pulse, QEIM<2:0> = 100.
2. Position counter reset by match with MAXCNT, QEIM<2:0> = 101.

When control bits QEIM<2:0> = 110 or 111, the x4 Measurement mode is selected and the QEI logic looks at both edges of the Phase A and Phase B input signals. Every edge of both signals causes the position counter to increment or decrement.

Within the x4 Measurement mode, there are two variations of how the position counter is reset:

1. Position counter reset by detection of index pulse, QEIM<2:0> = 110.
2. Position counter reset by match with MAXCNT, QEIM<2:0> = 111.

The x4 Measurement mode provides for finer resolution data (more position counts) for determining motor position.

14.4 Programmable Digital Noise Filters

The digital noise filter section is responsible for rejecting noise on the incoming capture or quadrature signals. Schmitt Trigger inputs and a three-clock cycle delay filter combine to reject low level noise and large, short duration noise spikes that typically occur in noise prone applications, such as a motor system.

The filter ensures that the filtered output signal is not permitted to change until a stable value has been registered for three consecutive clock cycles.

For the QEA, QEB and INDX pins, the clock divide frequency for the digital filter is programmed by bits QECK<2:0> (DFLTCON<6:4>) and are derived from the base instruction cycle TCY.

To enable the filter output for channels QEA, QEB and INDX, the QEOUT bit must be '1'. The filter network for all channels is disabled on POR and BOR.

14.5 Alternate 16-bit Timer/Counter

When the QEI module is not configured for the QEI mode QEIM<2:0> = 001, the module can be configured as a simple 16-bit timer/counter. The setup and control of the auxiliary timer is accomplished through the QEICON SFR register. This timer functions identically to Timer1. The QEA pin is used as the timer clock input.

When configured as a timer, the POSCNT register serves as the Timer Count Register and the MAXCNT register serves as the Period Register. When a timer/period register match occur, the QEI interrupt flag will be asserted.

The only exception between the general purpose timers and this timer is the added feature of external Up/Down input select. When the UPDN pin is asserted high, the timer will increment up. When the UPDN pin is asserted low, the timer will be decremented.

Note:	Changing the operational mode (i.e., from QEI to Timer or vice versa), will not affect the Timer/Position Count Register contents.

The UPDN Control/Status bit (QEICON<11>) can be used to select the count direction state of the Timer register. When UPDN = 1, the timer will count up. When UPDN = 0, the timer will count down.

In addition, control bit UPDN_SRC (QEICON<0>) determines whether the timer count direction state is based on the logic state, written into the UPDN Control/Status bit (QEICON<11>), or the QEB pin state. When UPDN_SRC = 1, the timer count direction is controlled from the QEB pin. Likewise, when UPDN_SRC = 0, the timer count direction is controlled by the UPDN bit.

Note:	This Timer does not support the External Asynchronous Counter mode of operation. If using an external clock source, the clock will automatically be synchronized to the internal instruction cycle.

14.6 QEI Module Operation During CPU Sleep Mode

14.6.1 QEI OPERATION DURING CPU SLEEP MODE

The QEI module will be halted during the CPU Sleep mode.

14.6.2 TIMER OPERATION DURING CPU SLEEP MODE

During CPU Sleep mode, the timer will not operate, because the internal clocks are disabled.

14.7 QEI Module Operation During CPU Idle Mode

Since the QEI module can function as a quadrature encoder interface, or as a 16-bit timer, the following section describes operation of the module in both modes.

14.7.1 QEI OPERATION DURING CPU IDLE MODE

When the CPU is placed in the Idle mode, the QEI module will operate if the QEISIDL bit (QEICON<13>) = 0. This bit defaults to a logic '0' upon executing POR and BOR. For halting the QEI module during the CPU Idle mode, QEISIDL should be set to '1'.

14.7.2 TIMER OPERATION DURING CPU IDLE MODE

When the CPU is placed in the Idle mode and the QEI module is configured in the 16-bit Timer mode, the 16-bit timer will operate if the QEISIDL bit (QEICON<13>) = 0. This bit defaults to a logic '0' upon executing POR and BOR. For halting the timer module during the CPU Idle mode, QEISIDL should be set to '1'.

If the QEISIDL bit is cleared, the timer will function normally, as if the CPU Idle mode had not been entered.

14.8 Quadrature Encoder Interface Interrupts

The quadrature encoder interface has the ability to generate an interrupt on occurrence of the following events:

- Interrupt on 16-bit up/down position counter rollover/underflow
- Detection of qualified index pulse, or if CNTERR bit is set
- Timer period match event (overflow/underflow)
- Gate accumulation event

The QEI Interrupt Flag bit, QEIIF, is asserted upon occurrence of any of the above events. The QEIIF bit must be cleared in software. QEIIF is located in the IFS2 Status register.

Enabling an interrupt is accomplished via the respective Enable bit, QEIIE. The QEIIE bit is located in the IEC2 Control register.

TABLE 14-1: QEI REGISTER MAP

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State
QEICON	0122	CNTERR	—	QEISIDL	INDX	UPDN	QEIM2	QEIM1	QEIM0	SWPAB	—	TQGATE	TQCKPS1	TQCKPS0	POSRES	TQCS	UPDN_SRC	0000 0000 0000 0000
DFLTCON	0124	—	—	—	—	—	IMV1	IMV0	CEID	QEOUT	QECK2	QECK1	QECK0	—	—	—	—	0000 0000 0000 0000
POSCNT	0126	Position Counter<15:0>																0000 0000 0000 0000
MAXCNT	0128	Maximum Count<15:0>																1111 1111 1111 1111
ADPCFG	02A8	—	—	—	—	—	—	—	PCFG8	PCFG7	PCFG6	PCFG5	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0	0000 0000 0000 0000

Legend: u = uninitialized bit

Note: Refer to *dsPIC30F Family Reference Manual* (DS70046) for descriptions of register bit fields.

15.0 MOTOR CONTROL PWM MODULE

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

This module simplifies the task of generating multiple, synchronized Pulse Width Modulated (PWM) outputs. In particular, the following power and motion control applications are supported by the PWM module:

- Three Phase AC Induction Motor
- Switched Reluctance (SR) Motor
- Brushless DC (BLDC) Motor
- Uninterruptible Power Supply (UPS)

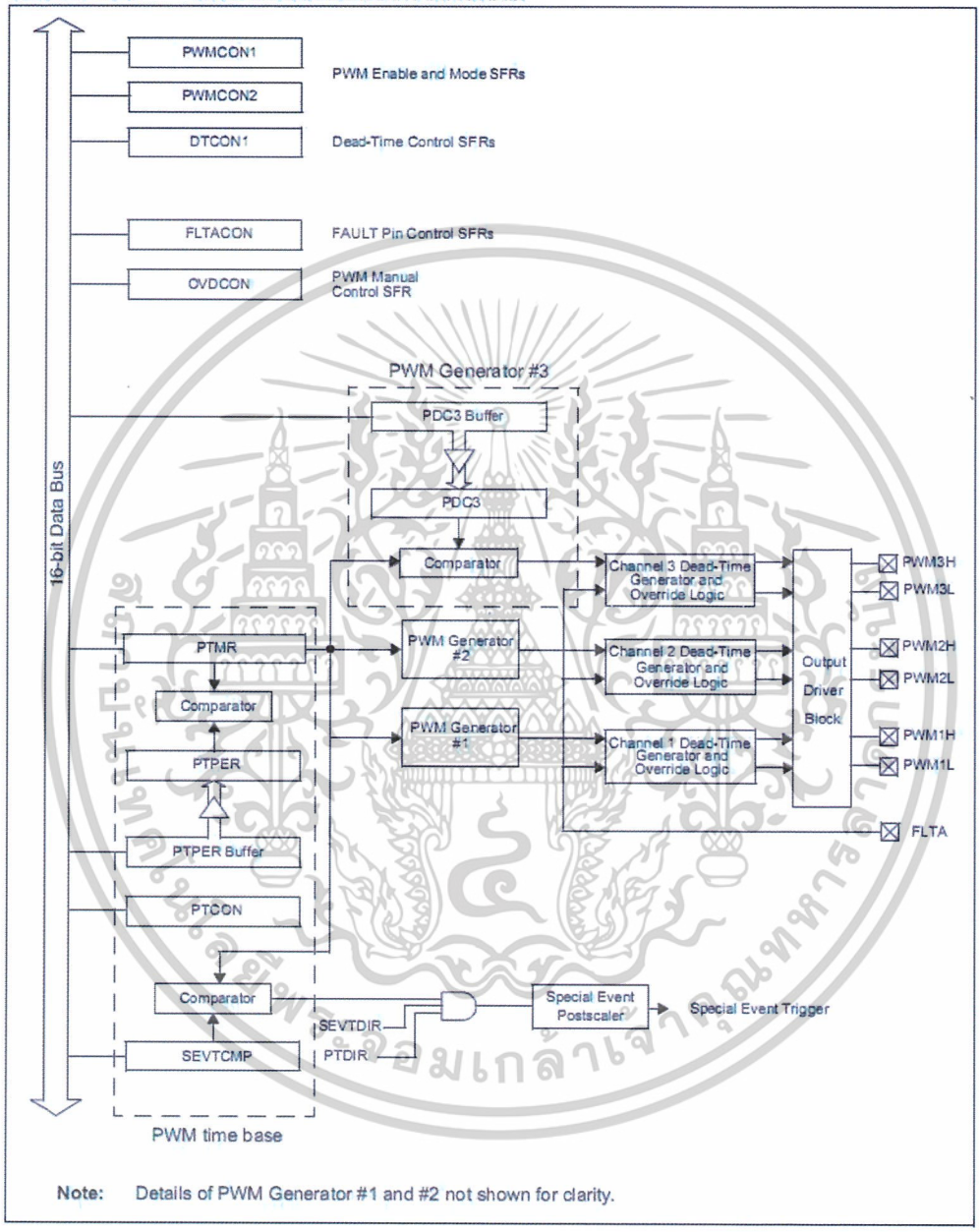
The PWM module has the following features:

- 6 PWM I/O pins with 3 duty cycle generators
- Up to 16-bit resolution
- 'On-the-Fly' PWM frequency changes
- Edge and Center Aligned Output modes
- Single Pulse Generation mode
- Interrupt support for asymmetrical updates in Center Aligned mode
- Output override control for Electrically Commutative Motor (ECM) operation
- 'Special Event' comparator for scheduling other peripheral events
- FAULT pins to optionally drive each of the PWM output pins to a defined state

This module contains 3 duty cycle generators, numbered 1 through 3. The module has 6 PWM output pins, numbered PWM1H/PWM1L through PWM3H/PWM3L. The six I/O pins are grouped into high/low numbered pairs, denoted by the suffix H or L, respectively. For complementary loads, the low PWM pins are always the complement of the corresponding high I/O pin.

The PWM module allows several modes of operation which are beneficial for specific power control applications.

FIGURE 15-1: PWM MODULE BLOCK DIAGRAM



Note: Details of PWM Generator #1 and #2 not shown for clarity.

15.1 PWM Time Base

The PWM time base is provided by a 15-bit timer with a prescaler and postscaler. The time base is accessible via the PTMR SFR. PTMR<15> is a Read Only Status bit, PTDIR, that indicates the present count direction of the PWM time base. If PTDIR is cleared, PTMR is counting upwards. If PTDIR is set, PTMR is counting downwards. The PWM time base is configured via the PTCON SFR. The time base is enabled/disabled by setting/clearing the PTEN bit in the PTCON SFR. PTMR is not cleared when the PTEN bit is cleared in software.

The PTPER SFR sets the counting period for PTMR. The user must write a 15-bit value to PTPER<14:0>. When the value in PTMR<14:0> matches the value in PTPER<14:0>, the time base will either reset to 0, or reverse the count direction on the next occurring clock cycle. The action taken depends on the operating mode of the time base.

Note:	If the period register is set to 0x0000, the timer will stop counting, and the interrupt and the special event trigger will not be generated, even if the special event value is also 0x0000. The module will not update the period register, if it is already at 0x0000; therefore, the user must disable the module in order to update the period register.

The PWM time base can be configured for four different modes of operation:

- Free Running mode
- Single Shot mode
- Continuous Up/Down Count mode
- Continuous Up/Down Count mode with interrupts for double updates

These four modes are selected by the PTMOD<1:0> bits in the PTCON SFR. The Up/Down Counting modes support center aligned PWM generation. The Single Shot mode allows the PWM module to support pulse control of certain Electronically Commutative Motors (ECMs).

The interrupt signals generated by the PWM time base depend on the mode selection bits (PTMOD<1:0>) and the postscaler bits (PTOPS<3:0>) in the PTCON SFR.

15.1.1 FREE RUNNING MODE

In the Free Running mode, the PWM time base counts upwards until the value in the Time Base Period register (PTPER) is matched. The PTMR register is reset on the following input clock edge and the time base will continue to count upwards as long as the PTEN bit remains set.

When the PWM time base is in the Free Running mode (PTMOD<1:0> = 00), an interrupt event is generated each time a match with the PTPER register occurs and the PTMR register is reset to zero. The postscaler selection bits may be used in this mode of the timer to reduce the frequency of the interrupt events.

15.1.2 SINGLE SHOT MODE

In the Single Shot Counting mode, the PWM time base begins counting upwards when the PTEN bit is set. When the value in the PTMR register matches the PTPER register, the PTMR register will be reset on the following input clock edge and the PTEN bit will be cleared by the hardware to halt the time base.

When the PWM time base is in the Single Shot mode (PTMOD<1:0> = 01), an interrupt event is generated when a match with the PTPER register occurs, the PTMR register is reset to zero on the following input clock edge, and the PTEN bit is cleared. The postscaler selection bits have no effect in this mode of the timer.

15.1.3 CONTINUOUS UP/DOWN COUNTING MODES

In the Continuous Up/Down Counting modes, the PWM time base counts upwards until the value in the PTPER register is matched. The timer will begin counting downwards on the following input clock edge. The PTDIR bit in the PTCON SFR is read only and indicates the counting direction. The PTDIR bit is set when the timer counts downwards.

In the Up/Down Counting mode (PTMOD<1:0> = 10), an interrupt event is generated each time the value of the PTMR register becomes zero and the PWM time base begins to count upwards. The postscaler selection bits may be used in this mode of the timer to reduce the frequency of the interrupt events.

15.1.4 DOUBLE UPDATE MODE

In the Double Update mode (PTMOD<1:0> = 11), an interrupt event is generated each time the PTMR register is equal to zero, as well as each time a period match occurs. The postscaler selection bits have no effect in this mode of the timer.

The Double Update mode provides two additional functions to the user. First, the control loop bandwidth is doubled because the PWM duty cycles can be updated, twice per period. Second, asymmetrical center-aligned PWM waveforms can be generated, which are useful for minimizing output waveform distortion in certain motor control applications.

Note:	Programming a value of 0x0001 in the period register could generate a continuous interrupt pulse, and hence, must be avoided.

15.1.5 PWM TIME BASE PRESCALER

The input clock to PTMR (Fosc/4), has prescaler options of 1:1, 1:4, 1:16, or 1:64, selected by control bits PTCKPS<1:0> in the PTCON SFR. The prescaler counter is cleared when any of the following occurs:

- a write to the PTMR register
- a write to the PTCON register
- any device Reset

The PTMR register is not cleared when PTCON is written.

15.1.6 PWM TIME BASE POSTSCALER

The match output of PTMR can optionally be postscaled through a 4-bit postscaler (which gives a 1:1 to 1:16 scaling).

The postscaler counter is cleared when any of the following occurs:

- a write to the PTMR register
- a write to the PTCON register
- any device Reset

The PTMR register is not cleared when PTCON is written.

15.2 PWM Period

PTPER is a 15-bit register and is used to set the counting period for the PWM time base. PTPER is a double buffered register. The PTPER buffer contents are loaded into the PTPER register at the following instants:

- Free Running and Single Shot modes: When the PTMR register is reset to zero after a match with the PTPER register.
- Up/Down Counting modes: When the PTMR register is zero.

The value held in the PTPER buffer is automatically loaded into the PTPER register when the PWM time base is disabled (PTEN = 0).

The PWM period can be determined using Equation 15-1:

EQUATION 15-1: PWM PERIOD

$$T_{PWM} = \frac{T_{CY} \bullet (PTPER + 1)}{(PTMR\ Prescale\ Value)}$$

If the PWM time base is configured for one of the Up/Down Count modes, the PWM period will be twice the value provided by Equation 15-1.

The maximum resolution (in bits) for a given device oscillator and PWM frequency can be determined using Equation 15-2:

EQUATION 15-2: PWM RESOLUTION

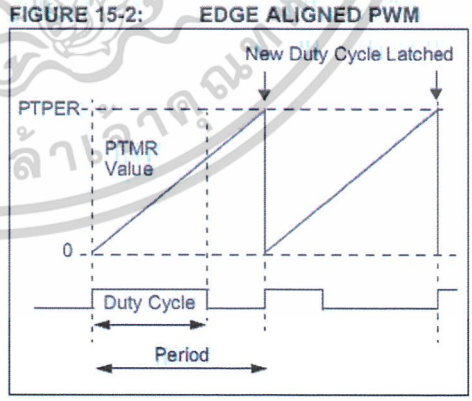
$$Resolution = \frac{\log (2 \bullet T_{PWM} / T_{CY})}{\log (2)}$$

15.3 Edge Aligned PWM

Edge aligned PWM signals are produced by the module when the PWM time base is in the Free Running or Single Shot mode. For edge aligned PWM outputs, the output has a period specified by the value in PTPER and a duty cycle specified by the appropriate duty cycle register (see Figure 15-2). The PWM output is driven active at the beginning of the period (PTMR = 0) and is driven inactive when the value in the duty cycle register matches PTMR.

If the value in a particular duty cycle register is zero, then the output on the corresponding PWM pin will be inactive for the entire PWM period. In addition, the output on the PWM pin will be active for the entire PWM period if the value in the duty cycle register is greater than the value held in the PTPER register.

FIGURE 15-2: EDGE ALIGNED PWM



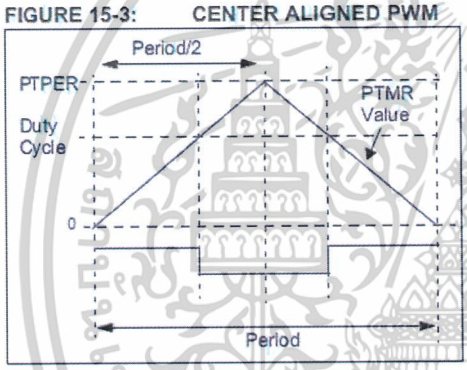
15.4 Center Aligned PWM

Center aligned PWM signals are produced by the module when the PWM time base is configured in an Up/Down Counting mode (see Figure 15-3).

The PWM compare output is driven to the active state when the value of the duty cycle register matches the value of PTMR and the PWM time base is counting downwards (PTDIR = 1). The PWM compare output is driven to the inactive state when the PWM time base is counting upwards (PTDIR = 0) and the value in the PTMR register matches the duty cycle value.

If the value in a particular duty cycle register is zero, then the output on the corresponding PWM pin will be inactive for the entire PWM period. In addition, the output on the PWM pin will be active for the entire PWM period if the value in the duty cycle register is equal to the value held in the PTPER register.

FIGURE 15-3: CENTER ALIGNED PWM



15.5 PWM Duty Cycle Comparison Units

There are three 16-bit special function registers (PDC1, PDC2 and PDC3) used to specify duty cycle values for the PWM module.

The value in each duty cycle register determines the amount of time that the PWM output is in the active state. The duty cycle registers are 16-bits wide. The LS bit of a duty cycle register determines whether the PWM edge occurs in the beginning. Thus, the PWM resolution is effectively doubled.

15.5.1 DUTY CYCLE REGISTER BUFFERS

The three PWM duty cycle registers are double buffered to allow glitchless updates of the PWM outputs. For each duty cycle, there is a duty cycle register that is accessible by the user and a second duty cycle register that holds the actual compare value used in the present PWM period.

For edge aligned PWM output, a new duty cycle value will be updated whenever a match with the PTPER register occurs and PTMR is reset. The contents of the duty cycle buffers are automatically loaded into the duty cycle registers when the PWM time base is disabled (PTEN = 0) and the UDIS bit is cleared in PWMCON2.

When the PWM time base is in the Up/Down Counting mode, new duty cycle values are updated when the value of the PTMR register is zero and the PWM time base begins to count upwards. The contents of the duty cycle buffers are automatically loaded into the duty cycle registers when the PWM time base is disabled (PTEN = 0).

When the PWM time base is in the Up/Down Counting mode with double updates, new duty cycle values are updated when the value of the PTMR register is zero, and when the value of the PTMR register matches the value in the PTPER register. The contents of the duty cycle buffers are automatically loaded into the duty cycle registers when the PWM time base is disabled (PTEN = 0).

15.6 Complementary PWM Operation

In the Complementary mode of operation, each pair of PWM outputs is obtained by a complementary PWM signal. A dead-time may be optionally inserted during device switching, when both outputs are inactive for a short period (Refer to Section 15.7).

In Complementary mode, the duty cycle comparison units are assigned to the PWM outputs as follows:

- PDC1 register controls PWM1H/PWM1L outputs
- PDC2 register controls PWM2H/PWM2L outputs
- PDC3 register controls PWM3H/PWM3L outputs

The Complementary mode is selected for each PWM I/O pin pair by clearing the appropriate PMODx bit in the PWMCON1 SFR. The PWM I/O pins are set to Complementary mode by default upon a device Reset.

15.7 Dead-Time Generators

Dead-time generation may be provided when any of the PWM I/O pin pairs are operating in the Complementary Output mode. The PWM outputs use Push-Pull drive circuits. Due to the inability of the power output devices to switch instantaneously, some amount of time must be provided between the turn off event of one PWM output in a complementary pair and the turn on event of the other transistor.

The PWM module allows two different dead-times to be programmed. These two dead-times may be used in one of two methods described below to increase user flexibility:

- The PWM output signals can be optimized for different turn off times in the high side and low side transistors in a complementary pair of transistors. The first dead-time is inserted between the turn off event of the lower transistor of the complementary pair and the turn on event of the upper transistor. The second dead-time is inserted between the turn off event of the upper transistor and the turn on event of the lower transistor.
- The two dead-times can be assigned to individual PWM I/O pin pairs. This Operating mode allows the PWM module to drive different transistor/load combinations with each complementary PWM I/O pin pair.

15.7.1 DEAD-TIME GENERATORS

Each complementary output pair for the PWM module has a 6-bit down counter that is used to produce the dead-time insertion. As shown in Figure 15-4, each dead-time unit has a rising and falling edge detector connected to the duty cycle comparison output.

15.7.2 DEAD-TIME RANGES

The amount of dead-time provided by the dead-time unit is selected by specifying the input clock prescaler value and a 6-bit unsigned value.

Four input clock prescaler selections have been provided to allow a suitable range of dead-time, based on the device operating frequency. The dead-time clock prescaler values are selected using the DTAPS<1:0> control bits in the DTCON1 SFR. One of four clock prescaler options (T_{CY}, $2T_{CY}$, $4T_{CY}$ or $8T_{CY}$) may be selected.

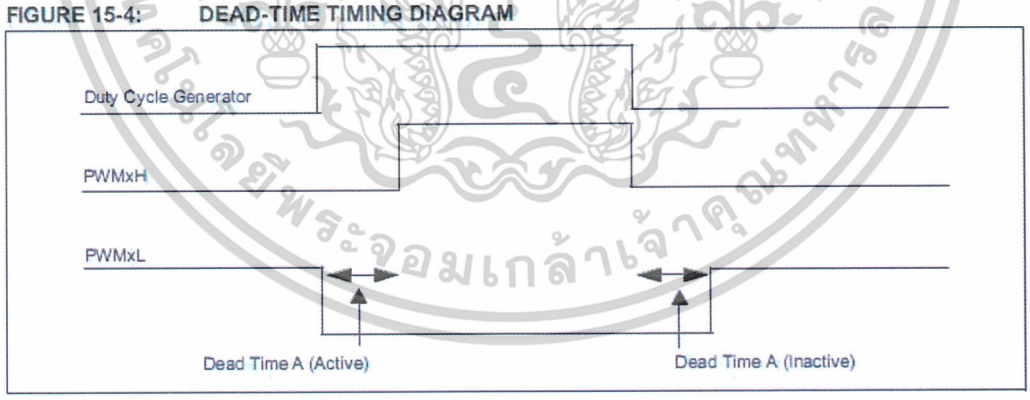
After the prescaler value is selected, the dead-time is adjusted by loading 6-bit unsigned values into the DTCON1 SFR.

The dead-time unit prescaler is cleared on the following events:

- On a load of the down timer due to a duty cycle comparison edge event.
- On a write to the DTCON1 register.
- On any device Reset.

Note: The user should not modify the DTCON1 value while the PWM module is operating (PTEN = 1). Unexpected results may occur.

FIGURE 15-4: DEAD-TIME TIMING DIAGRAM



15.11 PWM Output and Polarity Control

There are three device configuration bits associated with the PWM module that provide PWM output pin control:

- HPOL configuration bit
- LPOL configuration bit
- PWMPIN configuration bit

These three bits in the FPORBOR configuration register (see Section 21) work in conjunction with the three PWM Enable bits (PWMEN<3:1>) located in the PWMCON1 SFR. The configuration bits and PWM Enable bits ensure that the PWM pins are in the correct states after a device Reset occurs. The PWMPIN configuration fuse allows the PWM module outputs to be optionally enabled on a device Reset. If PWMPIN = 0, the PWM outputs will be driven to their inactive states at Reset. If PWMPIN = 1 (default), the PWM outputs will be tri-stated. The HPOL bit specifies the polarity for the PWMxH outputs, whereas the LPOL bit specifies the polarity for the PWMxL outputs.

15.11.1 OUTPUT PIN CONTROL

The PEN<3:1>H and PEN<3:1>L control bits in the PWMCON1 SFR enable each high PWM output pin and each low PWM output pin, respectively. If a particular PWM output pin not enabled, it is treated as a general purpose I/O pin.

15.12 PWM FAULT Pin

There is one FAULT pin (FLTA) associated with the PWM module. When asserted, these pins can optionally drive each of the PWM I/O pins to a defined state.

15.12.1 FAULT PIN ENABLE BITS

The FLTACON SFR has 3 control bits that determine whether a particular pair of PWM I/O pins is to be controlled by the FAULT input pin. To enable a specific PWM I/O pin pair for FAULT overrides, the corresponding bit should be set in the FLTACON register.

If all enable bits are cleared in the FLTACON register, then the corresponding FAULT input pin has no effect on the PWM module and the pin may be used as a general purpose interrupt or I/O pin.

Note:	The FAULT pin logic can operate independent of the PWM logic. If all the enable bits in the FLTACON register are cleared, then the FAULT pin could be used as a general purpose interrupt pin. The FAULT pin has an interrupt vector, Interrupt Flag bit and Interrupt Priority bits associated with it.

15.12.2 FAULT STATES

The FLTACON special function register has 6 bits that determine the state of each PWM I/O pin when it is overridden by a FAULT input. When these bits are cleared, the PWM I/O pin is driven to the inactive state. If the bit is set, the PWM I/O pin will be driven to the active state. The active and inactive states are referenced to the polarity defined for each PWM I/O pin (HPOL and LPOL polarity control bits).

A special case exists when a PWM module I/O pair is in the Complementary mode and both pins are programmed to be active on a FAULT condition. The PWMxH pin always has priority in the Complementary mode, so that both I/O pins cannot be driven active simultaneously.

15.12.3 FAULT INPUT MODES

The FAULT input pin has two modes of operation:

- **Latched Mode:** When the FAULT pin is driven low, the PWM outputs will go to the states defined in the FLTACON register. The PWM outputs will remain in this state until the FAULT pin is driven high and the corresponding interrupt flag has been cleared in software. When both of these actions have occurred, the PWM outputs will return to normal operation at the beginning of the next PWM cycle or half-cycle boundary. If the interrupt flag is cleared before the FAULT condition ends, the PWM module will wait until the FAULT pin is no longer asserted, to restore the outputs.
- **Cycle-by-Cycle Mode:** When the FAULT input pin is driven low, the PWM outputs remain in the defined FAULT states for as long as the FAULT pin is held low. After the FAULT pin is driven high, the PWM outputs return to normal operation at the beginning of the following PWM cycle or half-cycle boundary.

The Operating mode for the FAULT input pin is selected using the FLTAM control bit in the FLTACON Special Function Register.

The FAULT pin can be controlled manually in software.

15.13 PWM Update Lockout

For a complex PWM application, the user may need to write up to three duty cycle registers and the time base period register, PTPER, at a given time. In some applications, it is important that all buffer registers be written before the new duty cycle and period values are loaded for use by the module.

The PWM update lockout feature is enabled by setting the UDIS control bit in the PWMCON2 SFR. The UDIS bit affects all duty cycle buffer registers and the PWM time base period buffer, PTPER. No duty cycle changes or period value changes will have effect while UDIS = 1.

15.14 PWM Special Event Trigger

The PWM module has a special event trigger that allows A/D conversions to be synchronized to the PWM time base. The A/D sampling and conversion time may be programmed to occur at any point within the PWM period. The special event trigger allows the user to minimize the delay between the time when A/D conversion results are acquired and the time when the duty cycle value is updated.

The PWM special event trigger has a SFR named SEVTCMP, and five control bits to control its operation. The PTMR value for which a special event trigger should occur is loaded into the SEVTCMP register. When the PWM time base is in an Up/Down Counting mode, an additional control bit is required to specify the counting phase for the special event trigger. The count phase is selected using the SEVTDIR control bit in the SEVTCMP SFR. If the SEVTDIR bit is cleared, the special event trigger will occur on the upward counting cycle of the PWM time base. If the SEVTDIR bit is set, the special event trigger will occur on the downward count cycle of the PWM time base. The SEVTDIR control bit has no effect unless the PWM time base is configured for an Up/Down Counting mode.

15.14.1 SPECIAL EVENT TRIGGER POSTSCALER

The PWM special event trigger has a postscaler that allows a 1:1 to 1:16 postscale ratio. The postscaler is configured by writing the SEVOPS<3:0> control bits in the PWMCON2 SFR.

The special event output postscaler is cleared on the following events:

- Any write to the SEVTCMP register
- Any device Reset

15.15 PWM Operation During CPU Sleep Mode

The FAULT A input pin has the ability to wake the CPU from Sleep mode. The PWM module generates an interrupt if the FAULT pin is driven low while in Sleep.

15.16 PWM Operation During CPU Idle Mode

The PTCON SFR contains a PTSIDL control bit. This bit determines if the PWM module will continue to operate or stop when the device enters Idle mode. If PTSIDL = 0, the module will continue to operate. If PTSIDL = 1, the module will stop operation as long as the CPU remains in Idle mode.

TABLE 15-1: 6-OUTPUT PWM REGISTER MAP

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State
PTCON	01C0	PTEN	—	PTSIDL	—	—	—	—	—	PTOPS<3:0>				PTCKPS<1:0>		PTMOD<1:0>		0000 0000 0000 0000
PTMR	01C2	PTDIR	PWM Timer Count Value															0000 0000 0000 0000
PTPER	01C4	—	PWM Time Base Period Register															0000 0000 0000 0000
SEVTCMP	01C6	SEVTDIR	PWM Special Event Compare Register															0000 0000 0000 0000
PWMCON1	01C8	—	—	—	—	—	PTMOD3	PTMOD2	RTMOD1	—	PEN3H	PEN2H	PEN1H	—	PEN3L	PEN2L	PEN1L	0000 0000 1111 1111
PWMCON2	01CA	—	—	—	—	SEVOPS<3:0>				—	—	—	—	—	—	OSYNC	UDIS	0000 0000 0000 0000
DTCON1	01CC	—	—	—	—	—	—	—	—	DTAPS<1:0>		Dead-Time A Value						0000 0000 0000 0000
FLTACON	01D0	—	—	FAOV3H	FAOV3L	FAOV2H	FAOV2L	FAOV1H	FAOV1L	FLTAM	—	—	—	—	FAEN3	FAEN2	FAEN1	0000 0000 0000 0000
OVDCON	01D4	—	—	POVD3H	POVD3L	POVD2H	POVD2L	POVD1H	POVD1L	—	—	POUT3H	POUT3L	POUT2H	POUT2L	POUT1H	POUT1L	1111 1111 0000 0000
PDC1	01D6	PWM Duty Cycle #1 Register																0000 0000 0000 0000
PDC2	01D8	PWM Duty Cycle #2 Register																0000 0000 0000 0000
PDC3	01DA	PWM Duty Cycle #3 Register																0000 0000 0000 0000

Legend: u = uninitialized bit

Note: Refer to *dsPIC30F Family Reference Manual* (DS70046) for descriptions of register bit fields.

เอกสารอ้างอิง

[1] ผศ.ดร.วรพงศ์ ตั้งศรีรัตน์. **เซนเซอร์และทรานสดิวเซอร์** พิมพ์ครั้งที่ 4 พ.ศ. 2550
กรุงเทพมหานคร : โรงพิมพ์ห้างหุ้นส่วนจำกัด ที. เอส. บี. โปรดักส์

[2] วิศรุต ศรีรัตนะ. **เซนเซอร์และทรานสดิวเซอร์ในงานอุตสาหกรรม** พ.ศ. 2550
กรุงเทพมหานคร : บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน)

[3] Karl Johan Åström , Tore Hägglund. **PID Controller, 2nd Edition.**
USA : Instrument Society of America, Inc.1995.

[4] Karl Johan Åström , Tore Hägglund. **Advanced PID Control.**
USA : ISA – Instrumentation, Systems, and Automation Society, Inc.2006.

[5] Karl Johan Åström , Tore Hägglund. "Revisiting the Ziegler – Nichol Tuning Rules for PI Control." **Asian Journal of Control.**, Vol.4, No. 4, Dec. 2002. Pp. 364-380.

[6] K. J. Åström , H. Panagopoulos and T. Hägglund. "Design of PI Controllers based on Non-Convex Optimization." **Automatica**, Vol. 34, No 5, 1998. Pp585-601.

[7] Karl Johan Åström , Tore Hägglund. **"Revisiting the Ziegler-Nichols step response method for PID control."** [Online]. Available : http://www.sciencedirect.com. 2004.