สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

ระบบควบคุมการเคลื่อนที่ของเตียงคนไข้ด้วยคอมพิวเตอร์

นาย ชายแดน ทองช้อย 31.0805
นาย ทวีศักดิ์ อัฐนาค 31.0810

9 พ.
T523 ว
2535
เลขหมู่.....
เลขทะเบียน.....
วัน,เดือน,ปี.....

6125547 77

โครงงานพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต
ภาควิชา ฟิสิกส์ประยุกต์
คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา2535

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Computerize Control System for CT Patient Table

Chaidan Tongchoi 31.0805

Taweesak Attanak 31.0810

A Special Project Submitted in Partial Fultillment of the

Reguirement for the Degree of Bachelor of science

Department of Applied Physics

Faculty of science

King mongkut's institue of technology Ladkrabang

1992

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หน้าอนุมัติ

หัวข้อโครงงานพิเศษ ระบบควบคุมการเคลื่อนที่ของเตียงคนไข้
ด้วยคอมพิวเตอร์

โดย นาย ชายแดน ทองช้อย
นาย ทวีศักดิ์ อัฐนาค

อาจารย์ที่ปรึกษา ผศ.ดร.ปรีชา เทียนสมประสงค์
ภาควิชา ฟิสิกส์ประยุกต์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
อนุมัติให้โครงงานพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตร์บัณฑิต

ลายเซนต์

(ผศ.ดร.เสน่ห์ เอกะวิภาต) หัวหน้าภาค
คณะกรรมการโครงงานพิเศษ

(ผศ.ดร.บุญส่ง ศิวะโมษธรรม) ประธานกรรมการ

(ผศ.ดร.ปรีชา เทียนสมประสงค์) กรรมการ

(อาจารย์ อนุชิต จารุวนาวัฒน์) กรรมการ

ลิขสิทธิ์ของภาควิชา ฟิสิกส์ประยุกต์ คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หัวข้อโครงงานพิเศษ ระบบควบคุมการเคลื่อนที่ของเตียงคนไข้ด้วยคอมพิวเตอร์
นักศึกษา นาย ชายแดน ทองช้อย
นาย ทวีศักดิ์ อัฐนาค
อาจารย์ที่ปรึกษา ผศ.ดร. ปรีชา เทียนสมประสงค์
นาย อัครินทร์ คุณกิตติ
ภาควิชา พิสิกส์ประยุกต์
ปีการศึกษา 2535

บทคัดย่อ

ระบบการเคลื่อนที่ของเตียงคนไข้ด้วยคอมพิวเตอร์ เป็นระบบที่ใช้ในกระบวนการสร้างภาพตัดขวางของร่างกายโดยใช้รังสีเอกซ์ ซึ่งเตียงจะทำหน้าที่ส่งคนไข้เพื่อเข้าไปทำการเอกซ์เรย์ตามกระบวนการสร้างภาพตัดขวาง โดยได้ออกแบบให้เตียงสามารถเคลื่อนที่ได้ 3ทิศทาง คือ การเคลื่อนที่เข้า-ออก ในแนวนอน, การเคลื่อนที่เชิงมุมเพื่อปรับระดับของเตียงในแนวเอียง และการเคลื่อนที่ ขึ้น-ลง ในแนวตั้งฉากกับพื้น

ในการควบคุมการเคลื่อนที่ของเตียง ได้ใช้แรงดันไฟเอาท์พุท ±10 โวลต์ จากคอมพิวเตอร์อุตสาหกรรมเข้าไปยังตัวขับมอเตอร์ซึ่งทำหน้าที่ขยายแรงดันไฟให้เป็น ±180 โวลต์ จ่ายให้แก่ดีซีมอเตอร์เพื่อไปขับเกลียวให้หมุนซึ่งจะทำให้เตียงเคลื่อนที่ได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Special Project Title: Computer Control system For patient's Table

Name Mr.Chaidan Tongchoi
Mr.Taweesak Attanak

Special Project Advisor Asist.Prof.Preechar Theansomprasong
Mr Arkharin khunkitti

Department of Applied Physics

Acadamic Year 1992

Abstract

The function of patient table computerized control system used to carry the patient into x-rays system. The table can be move in three direction. That is forward-backward, up-down and inclined to the vertical axis.

The movement controlled voltages are supplied ±10 volts to the control system by an industrial computer , then the driver circuits amplified these voltages to ±180 volts for dc motors.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิจกรรมประกาศ

โครงงานพิเศษเรื่องระบบควบคุมการเคลื่อนที่ของเตียงคนไข้ด้วยคอมพิวเตอร์สำเร็จลุล่วงไปขั้นหนึ่ง ซึ่งได้ผลเป็นที่น่าพอใจทั้งนี้โดยการสนับสนุนจากสำนักวิจัยและบริการคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ต้องขอขอบคุณ คุณอัครินทร์ คุณกิตติ วิศวกรของสำนักวิจัยฯ ที่ช่วยให้คำแนะนำและปรึกษาทั้งทางด้านซอฟแวร์และฮาร์ดแวร์ และขอขอบคุณอาจารย์ภาคเครื่องกลทุกท่านที่ให้คำแนะนำและสร้างเตียงเครื่องต้นแบบ สุดท้ายขอขอบคุณผู้ช่วยนักวิจัยประจำห้องCIทุกท่าน รวมทั้งน้องๆภาคคอนโทรลที่ช่วยให้งานสำเร็จลุล่วงไปด้วยดี

นายชายแดน ทองช้อย

นายทวีศักดิ์ อัฐนาค

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

	หน้า
บทคัดย่อปัญหาพิเศษภาษาไทย	ก
บทคัดย่อปัญหาพิเศษภาษาอังกฤษ	ข
กิตติกรรมประกาศ	ค
สารบัญรูป	ง
สารบัญตาราง	จ
บทที่1 บทนำ	1
1.1 บทนำ	1
1.2 วัตถุประสงค์	3
1.3 ขอบเขตการดำเนินงาน	3
บทที่2 ทฤษฎีเกี่ยวกับการควบคุม	5
2.1 ระบบควบคุม	5
2.2 อนาลอกคอนโทรลเลอร์	7
2.3 คุณสมบัติของตัวคอนโทรลเลอร์	9
2.3.1 ปร็อบพอร์ชั่นแนลคอนโทรล	9
2.3.2 อินติกรัลคอนโทรล	11
2.3.3 ปร็อบพอร์ชั่นแนล-อินติกรัล คอนโทรล	14
2.3.4 ดิริเวทีฟคอนโทรล	16
2.3.5 ปร็อบพอร์ชั่นแนล-ดิริเวทีฟ คอนโทรล	20
2.4 ระบบเกียร์	22
2.5 อินคริเมนท์ เอนโคดเดอร์	23
2.5.1 ความละเอียดของอินคริเมนท์ เอนโคดเดอร์	23
2.5.2 เอาท์พุทของเอนโคดเดอร์	25
2.6 โมเดลคณิตศาสตร์ของดีซีมอเตอร์	28
2.6.1 โมเดลอิเลคโทรแมคคานิคอล	28
2.6.2 ทรานสเฟอร์ฟังก์ชั่นของดีซีมอเตอร์	32
2.6.3 การสูญเสียกำลังงานของดีซีมอเตอร์	36
2.7 ไทม์-โดเมน	40
2.7.1 สเตปอินพุทฟังก์ชั่น	41

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หน้า

2.7.2 ผลตอบสนองชั่วขณะ 41

บทที่3 การวิจัยและดำเนินการ 43

การทดลองควบคุมการเคลื่อนที่ของเตียงคนไข้ด้วยคอมพิเตอร์ 43

3.1 มัลติโปรเซสเซอร์ 43

3.1.1 มาสเตอร์/สเลฟ 43

3.1.2 มัลติมาสเตอร์ 44

3.2 การ์ดที่ใช้ในระบบควบคุมการเคลื่อนที่ของเตียงคนไข้ 45

3.2.1 การ์ด 7892 45

3.2.2 การ์ด 7171 46

3.2.3 การ์ด 7340 46

3.3 ส่วนประกอบของเตียงคนไข้ 47

3.3.1 มอเตอร์ 47

3.3.2 เอนโคดเดอร์ 48

3.3.3 ลิเนียร์ไกด์เวย์ และแบริ่ง 49

3.3.4 บอลสกรู และแบริ่ง 49

3.3.5 สกรูส่งกำลัง 51

3.3.5.1 เกลียวแอกมี 52

3.3.5.2 โมเมนท์บิดสำหรับหมุนสกรูส่งกำลัง 52

3.4 การคำนวณทางคณิตศาสตร์ที่ใช้ในการควบคุมการเคลื่อนที่ของเตียงคนไข้ 58

3.4.1 สมการการเคลื่อนที่ ขึ้น-ลง ของเตียงคนไข้ 58

3.4.2 ความสัมพันธ์ระหว่างระยะพิทของเกลียว,ความเร็วสูงสุดของเตียง และความเร็วสูงสุดของมอเตอร์ 60

3.4.3 การออกแบบการติดตั้งเอนโคดเดอร์สำหรับมุมเอียงของเตียง 61

3.4.4 การคำนวนเพื่อตรวจเช็คว่าเตียงเคลื่อนที่เกินค่าความถี่ตอบสนอง(response frequency)ของเอนโคดเดอร์หรือไม่ 61

3.4.5 การออกแบบการติดตั้งเอนโคดเดอร์เพื่อให้ได้จำนวนพัลส์สูงสุด 62

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หน้า

3.4.6 การคำนวนคำนวนทางกลศาสตร์เพื่อหาแรง
ที่กระทำต่อส่วนต่างๆของเตียง 63
3.4.7 การคำนวนหาแรงที่เกิดจากมอเตอร์ที่เตียงชั้นบน 69
3.4.8 การคำนวนหาแรงที่เกิดจากมอเตอร์ชั้นกลาง 71
3.4.9 การคำนวนหาแรงที่เกิดจากมอเตอร์ชั้นล่าง 73
3.5 วงจรควบคุมการเคลื่อนที่ของเตียงคนไข้ 75
3.5.1 การควบคุมโดยซอฟแวร์ 75
3.5.2 การควบคุมโดยฮาร์ดแวร์ 76
3.5.2.1 ส่วนที่ใช้ควบคุมโดยซอฟแวร์ 76
3.5.2.1 ส่วนที่ใช้ในการตัดไฟที่จ่ายให้กับมอเตอร์ 77
บทที่4 ผลการวิจัย และวิจารณ์ 83
บทที่5 สรุปผลการวิจัยและวิจารณ์ 90
ปัญหาที่พบในการทดลอง 91
วิธีแก้ไข 91
ภาคผนวก 92
บรรณานุกรม
ประวัติผู้เขียน

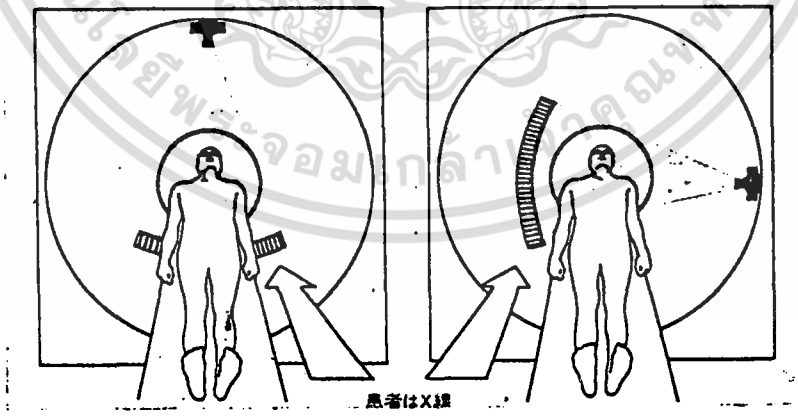
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่1

1.1 บทนำ

เทคโนโลยีการสร้างภาพตัดขวาง(Computerize X-rays Tomography) มีประโยชน์ทางด้านการแพทย์เป็นอย่างมาก ทั้งนี้เพราะว่าภาพตัดขวางนี้เป็นภาพของอวัยวะภายในของร่างกายซึ่งไม่สามารถมองเห็นด้วยตาเปล่าได้ ดังนั้นแพทย์จึงสามารถนำภาพตัดขวางของร่างกายที่สร้างขึ้นมานี้ ไปใช้ในการวิเคราะห์ผู้ป่วยได้อย่างถูกต้อง และมีประสิทธิภาพมากยิ่งขึ้น

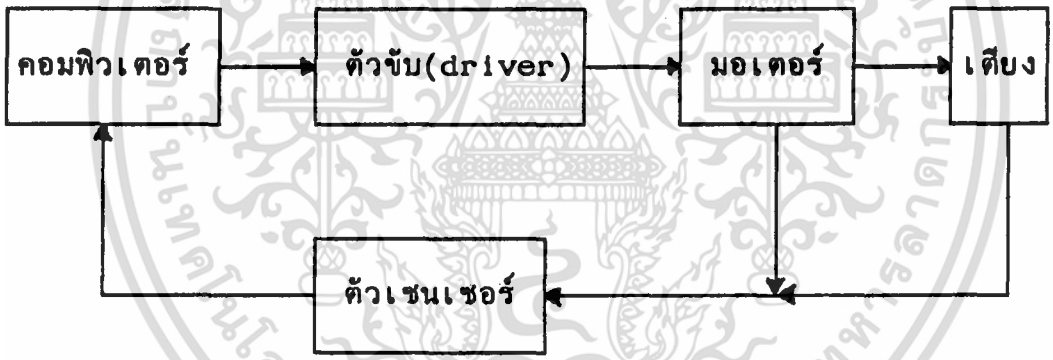
โครงงานพิเศษเรื่องการควบคุมการเคลื่อนที่ของเตียงคนไข้ด้วยคอมพิวเตอร์นี้ เป็นส่วนหนึ่งของทุนวิจัยเรื่องการสร้างภาพตัดขวางโดยการใช้รังสีเอกซ์ของสถาบันเทคโนโลยีพระจอมเกล้าจ้าคุณทหารลาดกระบัง ในกระบวนการการสร้างภาพตัดขวางจะประกอบด้วยช่องขนาดใหญ่เรียกว่าแกนทรี(gantry) ซึ่งจะหมุนเป็นวงกลม โดยมีการฉายรังสีเอกซ์ตลอดเวลาเมื่อมีการสร้างภาพ และที่ด้านตรงข้ามจะมีตัวรับรังสีแล้วส่งข้อมูลเข้าสู่คอมพิวเตอร์เพื่อทำการประมวลผลในการสร้างภาพต่อไป



รูปที่1.1 แสดงการหมุนของตัวส่งและตัวรับรังสีเอกซ์ในแกนทรี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การเคลื่อนที่ของเตียงเพื่อส่งคนไข้เข้าไปทำการสร้างภาพตัดขวาง สามารถเคลื่อนไปที่ตำแหน่งต่างๆโดยการควบคุมของคอมพิวเตอร์ ซึ่งการควบคุมจะกระทำที่คอลโซลหรือมอนิเตอร์โดยแพทย์หรือช่างเทคนิค ในการควบคุมด้วยคอมพิวเตอร์ทำได้โดยการใช้แรงดันไฟกระแสตรงจากเครื่องคอมพิวเตอร์ PRO-LOG 7340ไปขับดีซีมอเตอร์ผ่านเกลียวเพื่อทำให้เตียงเคลื่อนที่ แต่ในการปฏิบัติจริงๆ แล้วจะมีแรงเสียดทานมาก ดังนั้นจึงได้ใช้บอลสกรูและแบริ่งแทนระบบเกลียวที่เตียงชั้นบนสุดเพื่อลดแรงเสียดทานในการเคลื่อนที่ของเตียงให้น้อยลง โดยใช้เอนโคดเดอร์เป็นตัวเซนเซอร์เพื่อป้อนสัญญาณกลับเข้าสู่คอมพิวเตอร์ ซึ่งใช้สำหรับในการควบคุมเพื่อให้ได้ตำแหน่งที่ต้องการ การป้องกันระบบเพื่อป้องกันอันตรายที่อาจเกิดต่อคนไข้จะมีไมโครสวิทช์ทำการตัดไฟที่จ่ายให้กับมอเตอร์ นอกจากนี้ที่ตำแหน่งต่างๆยังมีตัวเซนเซอร์อื่นอีก เพื่อเป็นตัวบอกให้คอมพิวเตอร์รู้ว่าเตียงเคลื่อนที่ไปถึงตำแหน่งใด



รูปที่1.2 บล็อกไดอะแกรมของระบบการควบคุมเตียงคนไข้

เตียงคนไข้ในโครงงานพิเศษครั้งนี้ได้ออกแบบให้สามารถเคลื่อนที่ได้ใน 3 แนว ซึ่งได้แก่

1 การเคลื่อนที่ เข้า-ออก ในแนวนอน เพื่อเคลื่อนที่ เข้า-ออก จากเครื่องเอกซ์เรย์
2 การเคลื่อนที่เชิงมุม(ปรับระดับในแนวเอียง) เพื่อช่วยให้สามารถทำการเอกซ์เรย์ได้หลายระนาบ
3 การเคลื่อนที่ ขึ้น-ลง ในแนวตั้งฉากกับพื้น เพื่อสามารถปรับระดับความสูงของเตียงให้เหมาะสมกับคนไข้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในส่วนของซอฟแวร์ โปรแกรมที่ใช้ในการควบคุมการเคลื่อนที่จะทำหน้าที่ตรวจสอบว่าเตียงเคลื่อนที่ไปถึงตำแหน่งที่ต้องการแล้วหรือยัง รวมทั้งทำการตรวจสอบข้อผิดพลาดที่อาจจะเกิดขึ้นของระบบทั้งหมด ทั้งนี้เพื่อความปลอดภัยของคนไข้ ในกรณีที่ไม่สามารถแก้ไขได้ด้วยซอฟแวร์ซึ่งอาจเป็นเพราะเกิดความผิดพลาดพลาดขึ้นที่ซอฟแวร์ ก็จะเป็นหน้าที่ของฮาร์ดแวร์เข้าจัดการ เช่นการเคลื่อนที่ที่เกินตำแหน่งสูงสุดของเตียง ถ้าปล่อยให้เคลื่อนที่ต่อไปก็จะเกิดอันตรายต่อคนไข้ได้ ในกรณีนี้ซอฟแวร์จะสั่งให้ระบบหยุดการเคลื่อนที่ แต่ถ้าสั่งแล้วไม่หยุดฮาร์ดแวร์ก็จะทำการตัดไฟที่จ่ายให้กับมอเตอร์ เพื่อทำให้ระบบหยุดการทำงานโดยสิ้นเชิง

1.2 วัตถุประสงค์

1 เพื่อสร้างเตียงคนไข้และระบบความคุมการเคลื่อนที่ของเตียง สำหรับใช้ในการถ่ายภาพตัดขวางของอวัยวะภายในของร่างกาย

2 เพื่อสร้างเครื่องต้นแบบของเตียงคนไข้ และ ระบบควบคุม เพื่อที่จะนำไปใช้ในการพัฒนาให้ดียิ่งขึ้นต่อไป

1.3 ขอบเขตการดำเนินงาน

1 ศึกษาและสร้างเตียงคนไข้เครื่องต้นแบบ

2 ติดตั้งตัวเซนเซอร์ที่ตำแหน่งต่างๆของเตียงเพื่อใช้ในการตรวจเช็คตำแหน่งการเคลื่อนที่ของเตียง ซึ่งตัวเซนเซอร์ที่ใช้ได้แก่

   2.1 เอนโคดเดอร์

   2.2 ไมโครสวิทช์

   2.3 ออปโต สลอท คัฟเปิล(opto slotted coupled)

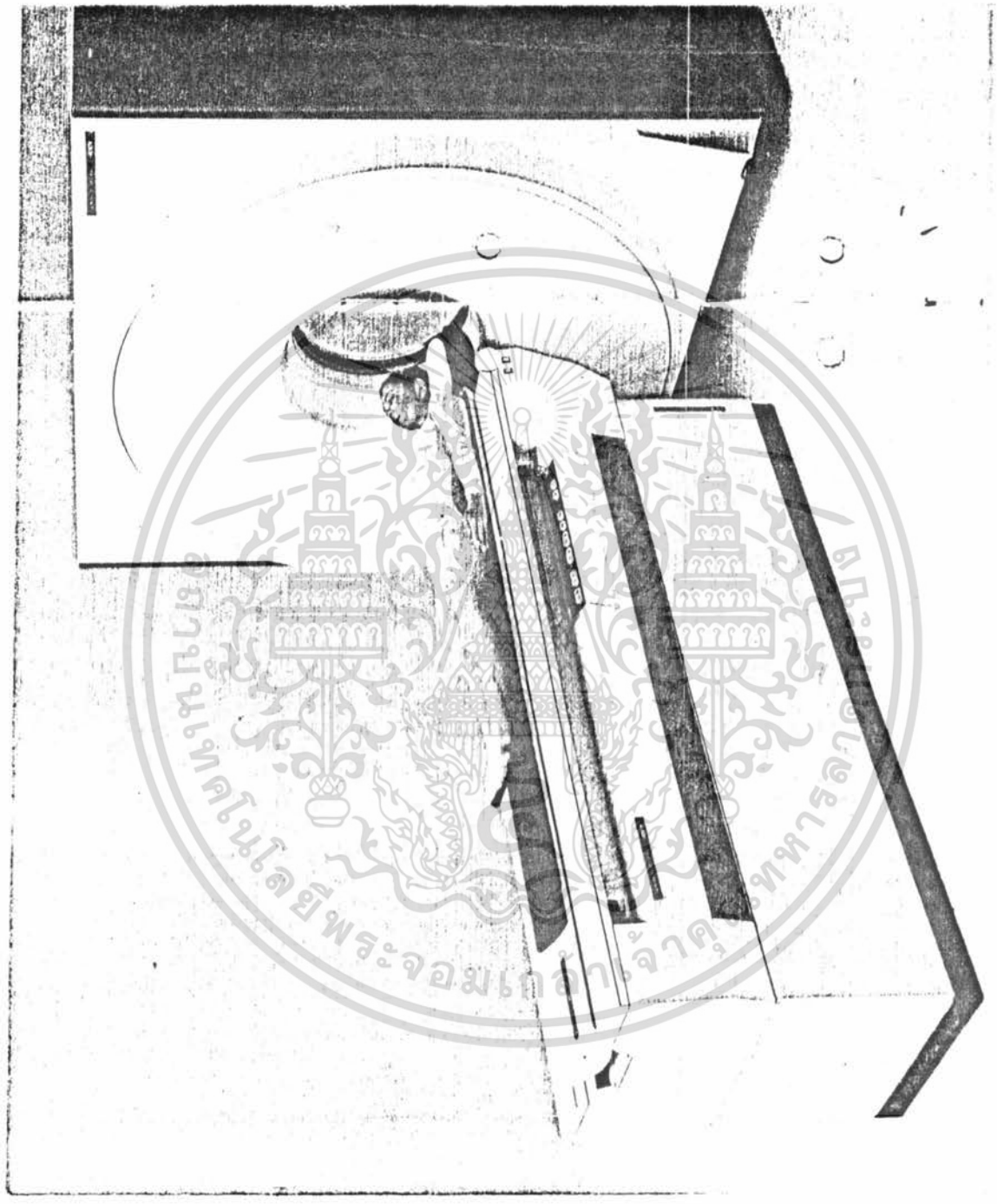
3 ศึกษาการเขียนโปรแกรมควบคุมไอซี LM 628 ซึ่งเป็นไอซีควบคุมมอเตอร์โดยใช้โปรแกรมภาษาซี

4 สร้างวงจรควบคุมการอ่านสถานะของตัวเซนเซอร์ และ ควบคุมการตัดต่อไฟที่จ่ายให้กับมอเตอร์ รวมทั้งเดินสายไฟระหว่างอุปกรณ์ต่างๆภายในเตียง

5 เชื่อมต่อวงจรขับมอเตอร์ และ วงจรควบคุมการอ่านสถานะและตัดต่อไฟเข้ากับคอมพิวเตอร์PRO-LOG 7340 ซึ่งมีไอซีควบคุมมอเตอร์ LM628 อยู่ภายใน

6 เขียนโปรแกรมภาษาซี เพื่อควบคุมการเคลื่อนที่ของเตียงคนไข้ทั้งสามแกน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่1.3 ภาพตัวอย่างการนำเตียงคนไข้ไปใช้ในการเอกซ์เรย์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

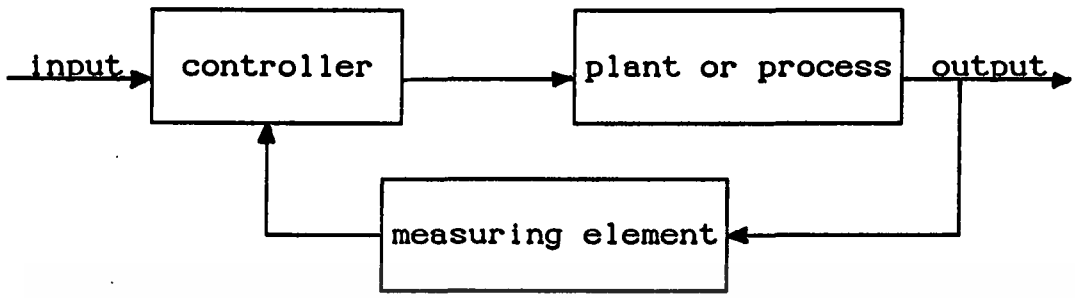
บทที่2
ทฤษฎีเกี่ยวกับการควบคุม

2.1 ระบบควบคุม

การควบคุมอัตโนมัติ มีบทบาทสำคัญต่อความก้าวหน้าทางวิทยาศาสตร์และวิศวกรรมศาสตร์มาก เช่นในกระบวนการทางอุตสาหกรรมที่จำเป็นต้องมีการควบคุม อุณหภูมิ ความดันความชื้น และอื่นๆ หรือในการควบคุมเครื่องยนต์กลไกต่างๆ ให้ทำงานตามที่ต้องการ ตลอดจนมีบทบาทที่สำคัญอย่างยิ่งในการพัฒนาทางด้านอวกาศและอาวุธนำวิถีเป็นต้น ความรู้ทางด้านทฤษฎีและการทดลองเกี่ยวกับการควบคุมอัตโนมัติจะทำให้สามารถควบคุมระบบให้มีสมรรถนะ ที่ดีที่สุด และทำให้ผลผลิตมีคุณภาพดี ทั้งยังสามารถลดต้นทุนในการผลิตได้ด้วย

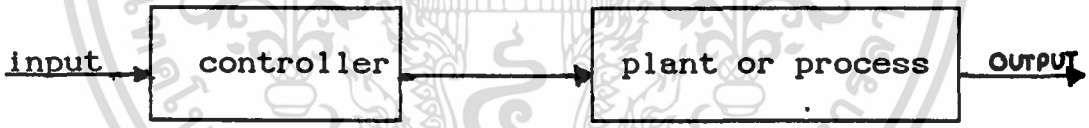
ระบบควบคุมอาจแบ่งอย่างง่ายๆได้สองแบบคือ ระบบควบคุมวงจรปิด (close-loop)และระบบควบคุมวงจรเปิด(open-loop) ระบบควบคุมที่ให้ผลดีและเป็นที่นิยมใช้ในงานควบคุมโดยทั่วไปคือระบบควบคุมแบบวงจรปิด ระบบควบคุมแบบวงจรปิดเป็นระบบควบคุมแบบหนึ่งซึ่งสัญญาณเอาท์พุทของระบบ (output-signal) มีผลโดยตรงต่อการควบคุมกล่าวคือเป็นระบบควบคุมป้อนกลับนั่นเอง สัญญาณค่าความคลาดเคลื่อน (actuating error signal) ซึ่งเป็นผลต่างระหว่างสัญญาณอินพุท (input signal) กับสัญญาณป้อนกลับ (feedback-signal) จะถูกป้อนให้ตัวควบคุม (controller) เพื่อจะลดค่าความคลาดเคลื่อนให้น้อยลงและทำให้เอาท์พุทของระบบมีค่าตามที่ต้องการ สัญญาณป้อนกลับนี้อาจเป็นสัญญาณเอาท์พุทโดยตรง หรือเป็นสัญญาณที่เป็นฟังก์ชั่นของสัญญาณเอาท์พุทก็ได้ รูปที่2.1 เป็นบล็อกไดอะแกรม (block diagram) ที่แสดงถึงความสัมพันธ์ระหว่างอินพุทและเอาท์พุทของระบบควบคุมวงจรปิด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่2.1 ระบบควบคุมวงจรปิด

ระบบควบคุมแบบวงจรเปิด เป็นระบบควบคุมที่เอาท์พุทของระบบจะไม่มีผลต่อการควบคุมเลย นั่นคือในกรณีของระบบควบคุมแบบวงจรเปิดนั้นเอาท์พุทของระบบจะไม่ถูกวัดหรือป้อนกลับเพื่อนำมาเปรียบเทียบกับอินพุท รูปที่2.2เป็นบล็อกไดอะแกรมแสดงความสัมพันธ์ระหว่างอินพุทกับเอาท์พุทของระบบควบคุมแบบวงจรเปิด



รูปที่2.2 ระบบควบคุมแบบวงจรเปิด

การควบคุมที่นิยมใช้กันมากได้แก่การควบคุมป้อนกลับ (feedback-control) ซึ่งเป็นการควบคุมแบบmanual หรือแบบอัตโนมัติก็ได้ ตัวอย่างเช่นในการควบคุมอุณหภูมิของน้ำในอ่างอาบน้ำ ผู้ควบคุมอาจใช้มือข้างหนึ่งจุ่มลงในอ่างเพื่อวัดอุณหภูมิน้ำดู และเขาอาจใช้มืออีกข้างหนึ่งในการปรับน้ำร้อนที่เข้าให้มากหรือน้อยเพื่อให้น้ำในอ่างมีอุณหภูมิตามที่ต้องการ การควบคุมแบบนี้เรียกว่าการควบคุมการป้อนกลับแบบ manual ถ้าผู้ควบคุมใช้เทอร์โมมิเตอร์ในการวัดอุณหภูมิแล้วเขาก็จะสามารถควบคุมอุณหภูมิของน้ำในอ่างอาบน้ำได้อย่างเที่ยงตรงมาก

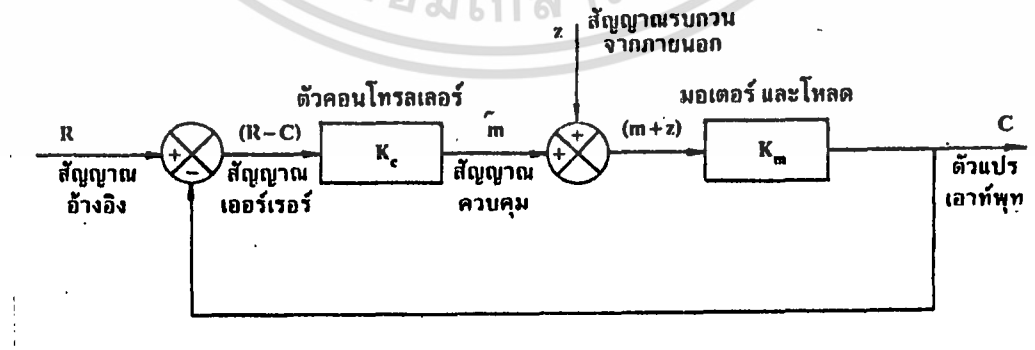
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ขึ้น ซึ่งจะเห็นว่าในการควบคุมแบบนี้เราสามารถที่จะทำการควบคุมได้ดี สำหรับในกรณีการควบคุมแบบป้อนกลับแบบอัตโนมัตินั้น จะต้องใช้อุปกรณ์วัดอุณหภูมิของน้ำร้อนที่วัดได้ไปยังตัวควบคุมด้วยเครื่องส่งสัญญาณเพื่อไปทำการเปรียบเทียบกับค่าอุณหภูมิของน้ำที่ต้องการ แล้วตัวควบคุมก็จะสร้างสัญญาณควบคุมเพื่อไปควบคุมการเปิด-ปิดของวาล์วควบคุมที่ทำหน้าที่ควบคุมปริมาณน้ำที่ไหลเข้าในอ่างให้มากขึ้นหรือน้อยลง เพื่อให้น้ำในอ่างมีอุณหภูมิตามที่เราต้องการ

2.2 อนาลอกคอนโทรลเลอร์

จุดมุ่งหมายของตัวคอลโทรลเลอร์(หรือตัวควบคุม)ในระบบการควบคุมมอเตอร์ก็คือ เพื่อที่จะปรับตัวแปรเอาท์พุท C ของระบบให้เข้าสู่ค่าที่กำหนดไว้ (ซึ่งก็คือตัวแปรอ้างอิง)และจะต้องรักษาค่าตัวแปรดังกล่าวให้อยู่ในค่าที่กำหนดนั้น และเพื่อให้เป็นไปตามเป้าหมายนี้ ตัวคอลโทรลเลอร์จะต้องแก้ไขผลกระทบของสัญญาณรบกวนจากภายนอกในวิถีทางที่เหมาะสม

ระบบการคอลโทรลมอเตอร์แบบพื้นฐานแสดงได้ในรูปที่2.3 ตัวคอนโทรลเลอร์จะปรับตัวแปรเอาท์พุท C โดยใช้สัญญาณควบคุม m เพื่อไปทำให้สัญญาณเออร์เรอร์ (R-C) มีค่าน้อยที่สุดเท่าที่จะทำได้ สัญญาณรบกวนจากภายนอกที่กระทำต่อระบบจะเข้ามารวมกับตัวแปรได้ผลรวมเป็น (m+z) จากรูปที่2.3เป็นระบบที่ลิเนียร์ ดังนั้นเราจะได้ว่า



รูปที่2.3 บล็อกไดอะแกรมของลูปการคอลโทรลมอเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$$m = K_g(R-C) \text{ และ } C = K_m(m+z) \qquad .....(2.1)$$

เมื่อ K_m เป็นมอเตอร์และโหลด ดังนั้นเราจะหาตัวแปรเอาท์พุท C ได้เป็น

$$C = \frac{K_c K_m}{1+K_c K_m} * R + \frac{K_m}{1+K_c K_m} * Z \qquad .....(2.2)$$

จากสมการที่(2.2) จะเห็นว่าผลตอบสนองของระบบคอลโทรลมอเตอร์ต่ออินพุทอางอิง C/ R มีค่าเข้าใกล้ค่าหนึ่งมากที่สุดเมื่อกำลังขยายมีค่ามากๆ ผลตอบสนองต่อสัญญาณรบกวนภายนอก C/ Z มีค่าเข้าใกล้ศูนย์มากเมื่อกำลังขยาย K_cของตัวคอลโทรลเลอร์มีค่ามาก

$$g = K_c K_m = \partial C/\partial(R-C) \qquad .....(2.3)$$

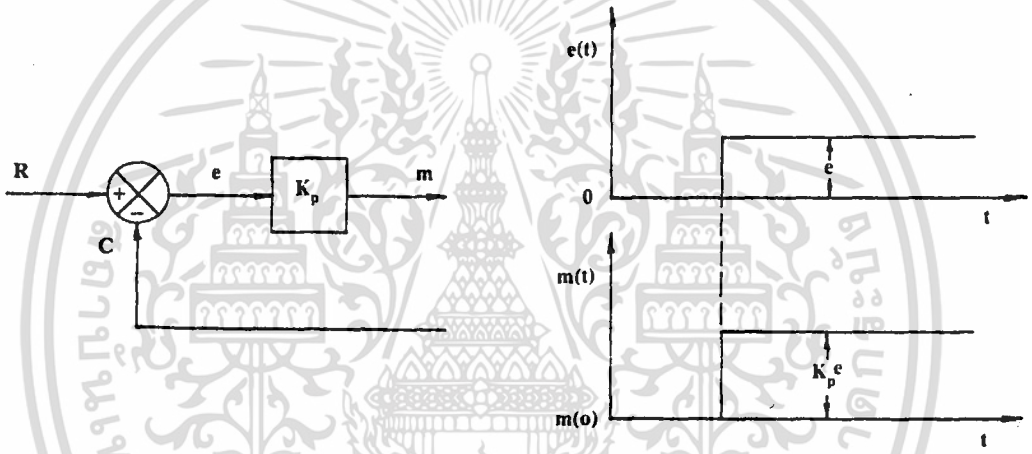
อย่างไรก็ตามค่ากำลังขยายของลูป g มีค่าที่ถูกจำกัดเมื่อกำลังขยายมีค่ามากเกินไป เฟสชิฟท์(phase shifts) ซึ่งไม่สามารถหลีกเลี่ยงได้ภายในคอลโทรลลูปจะมีค่าเพิ่มขึ้นด้วยซึ่งเป็นเหตุให้เกิดการออสซิเลท จุดประสงค์ของวิศวกรรมคอลโทรลก็เพื่อที่จะแก้ปัญหาข้อจำกัดนี้ โดยการคอลโทรลให้สัญญาณเออร์เลอร์มีค่าน้อยที่สุดและมีคุณสมบัติการตอบสนองของระบบที่ดี ด้วยเหตุนี้จึงต้องมีการออกแบบตัวคอลโทรลเลอร์ให้มีส่วนของอินติเกรเตอร์ และดิฟเฟอเรนทิเอเตอร์ร่วมเข้ากับปรอบพอร์ชั่นแนลคอนโทรลเลอร์ ดังนั้น P-คอลโทรลเลอร์จะต้องถูกเปลี่ยนให้เป็นPI หรือแม้กระทั่งเป็น PIDคอลโทรลเลอร์ ซึ่งเราจะได้กล่าวถึงหลักทฤษฎีของ P,PIและ PID คอลโทรลเลอร์ ดังต่อไปนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.3 คุณสมบัติของตัวคอนโทรลเลอร์

เราสามารถจำแนกตัวควบคุมอัตโนมัติที่ใช้ในงานอุตสาหกรรมได้ตามลักษณะของการควบคุมดังนี้

2.3.1 ปร็อบพอร์ชั่นแนลคอนโทรล (Proportional control) สัญญาณตัวแปรเพื่อการแก้ไขของตัวคอนโทรลเลอร์จะเป็นสัดส่วนกับสัญญาณเออร์เรอร์ ดังแสดงในรูปที่ 2.4



รูปที่2.4 ตัวคอนโทรลเลอร์แบบปร็อปพอร์ชั่นแนล

$$m = K_p e + m(o)$$

เมื่อ K_p คือกำลังขยายของตัวคอนโทรลเลอร์แบบปรอปพอร์ชั่นแนล

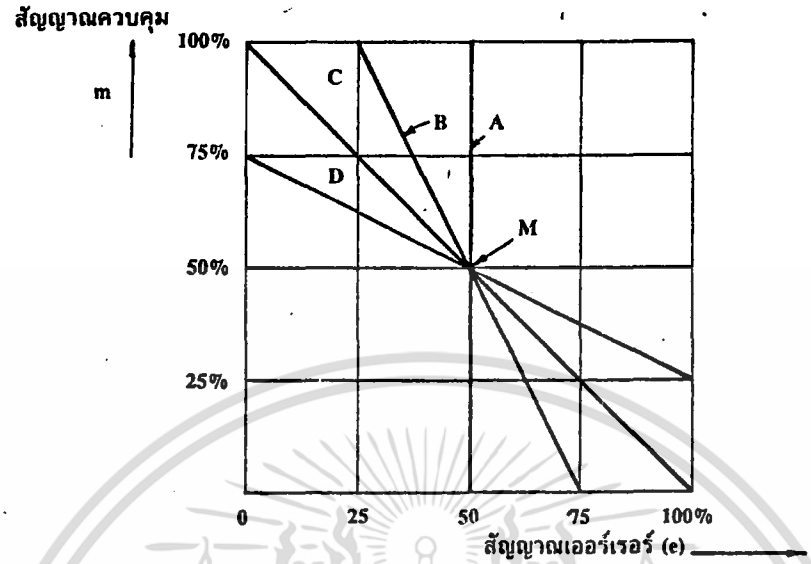
m(o) คือเอาท์พุทของตัวคอนโทรลเลอร์เมื่อเออร์เรอร์เป็นศูนย์

e(t) คือสัญญาณเออร์เรอร์ ณ เวลา t ใดๆ

m(t) สัญญาณตัวแปรเพื่อการแก้ไขของตัวคอลโทรลเลอร์

การคอนโทรลแบบนี้มักแสดงอยู่ในรูปของปรอบพอร์ชั่นแนลแบนด์ (PB) มีหน่วยเป็นเปอร์เซ็นต์ ค่าPB คือค่าแสดงสัญญาณเออร์เรอร์ที่ทำให้สัญญาณควบคุมเปลี่ยนไปร้อยเปอร์เซนต์ เมื่อPBเท่ากับ 50% หมายถึงสัญญาณเออร์เรอร์เกิดขึ้น 50% สัญญาณควบคุมจะเปลี่ยนไป 100% ดังแสดงในรูปที่ 2.5

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.5 ปรือปพอร์ชั่นแนลแบนด์

$$PB = \frac{1}{K_p} * 100 \quad [\%] \qquad .....(2.4)$$

A : PB = 0% , K_p = ∞ (100/0)

B : PB = 50% , K_p = 2 (100/50)

C : PB = 100% , K_p = 1 (100/100)

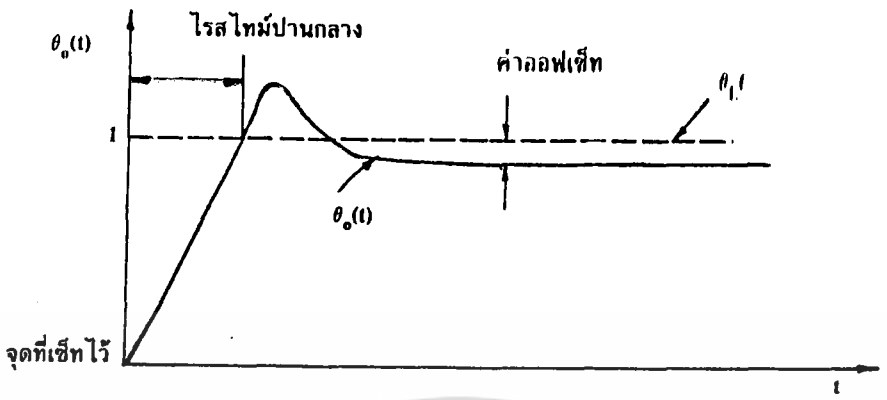
D : PB = 200% , K_p = 0.5 (100/200)

M : เป็นค่าคงที่

จากรูปที่2.5 แสดงว่าค่าPBเป็นตัวบอกความไวของการแก้ความผิดพลาด จึงอาจเรียกค่า K_pว่า ความไวของปรือบพอร์ชั่นแนลคอนโทรล ได้อีกชื่อหนึ่ง

ข้อเสียของการคอนโทรลแบบปรือบพอร์ชั่นแนล คือกรณีที่มีการเปลี่ยนโหลดจะเกิดค่าออฟเซ็ต (ค่าผิดพลาด)ของสัญญาณกระบวนการ (C)ที่สถานะคงที่โดยเฉพาะกับกระบวนการที่มีไทม์แล็กมากๆ สามารถแก้ไขทำได้โดยการลดค่าPB แต่ก็ไม่สามารถกำจัดค่าได้หมด เพราะถ้าค่า K_p สูงเกินไปการควบคุมจะเกิดการออสซิเลต ออฟเซ็ตเออร์เรอร์จะเป็นตัวจำกัดในการใช้งานของระบบการคอนโทรลแบบปรือบพอร์ชั่นแนลและการคอนโทรลแบบนี้จะใช้ได้ในบางกรณีเท่านั้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่2.6 ผลการตอบสนองของระบบ P คอนโทรล ต่อการเปลี่ยนแปลงของอินพุท (θ_i) แบบยูนิตสเตป

ปรอ็บพอร์ชั่นแนลคอนโทรลจะใช้งานได้เฉพาะในระบบที่ไม่มีการเปลี่ยนแปลงโหลดมากนัก หรือระบบที่มีกระบวนการที่มีไทม์แล็กน้อยๆหมายถึงระบบปรอ็บพอร์ชั่นแนลแบนด์น้อย(K_pมีค่ามาก) ซึ่งจะทำให้ค่าออฟเซ็ตเออร์เรอร์ลดลง

2.3.2 อินติกรัลคอนโทรล (Integral control)

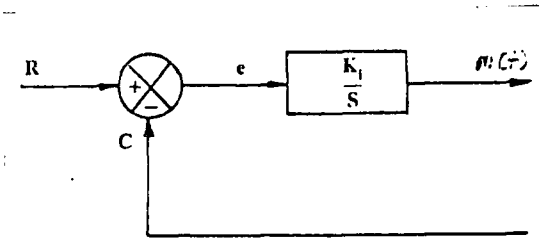
การคอนโทรลแบบนี้ค่าสัญญาณควบคุม(m)จะเป็นสัดส่วนโดยตรงกับค่าอินติกรัลของสัญญาณเออร์เรอร์ (e) จึงสามารถแก้ค่าออฟเซ็ตได้ การควบคุมแบบนี้เรียกว่ารีเซ็ตแอกชั่น

$$\frac{dm}{dt} = K_i e \qquad .....(2.5)$$

เมื่อ K_i : ค่าคงที่มีหน่วย (ต่อวินาที)

T_i : $\frac{1}{K_i}$: อินติกรัลไทม์ มีหน่วยเป็น (วินาที)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

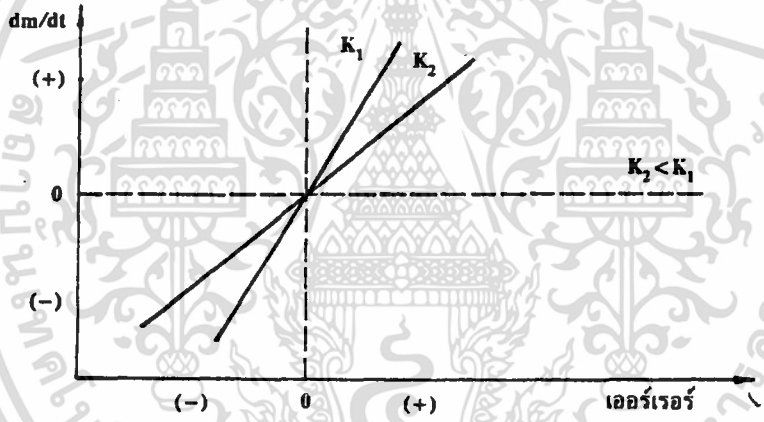


จากสมการที่ (2.5) เราหาเอาท์พุทของตัวคอนโทรลเลอร์

$$m = 1/T_i \int_0^t e(t)d(t) + m(o)$$

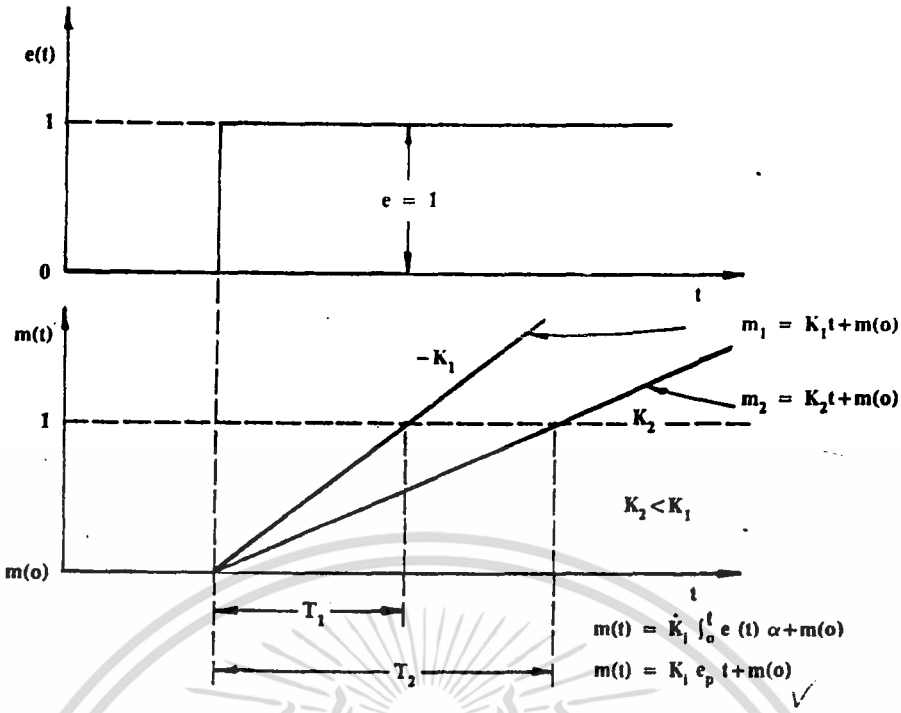
$$m = \frac{1}{T_i} \frac{e}{s} + m(o) = K_i \frac{e}{s} + m(o)$$

เมื่อ m(o) = เอาท์พุทของคอนโทรลเลอร์ที่เวลา t = 0



รูปที่2.7 แสดงอัตราการเปลี่ยนแปลงของเอาท์พุทขึ้นอยู่กับกำลังขยายและเออร์เรอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

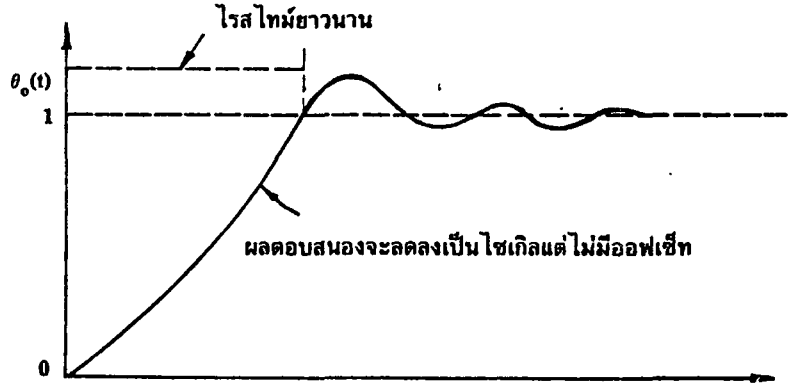


รูปที่ 2.8 แสดงผลตอบสนองของ I คอนโทรลต่อเออร์เรอร์ที่มีค่าคงที่

เมื่อเออร์เรอร์คงที่ m เอาพุทของคอนโทรลเลอร์จะเป็นฟังก์ชั่นกับเวลา
ถ้าระบบมี $K_i = K_1$ จะมีอินติกรัลไทม์ = T_1
ถ้าระบบมี$K_i = K_2$ จะมีอินติกรัลไทม์ = T_2
T_i(อินติกรัลไทม์) คือเวลาที่ใช้ในการเพิ่มค่าของสัญญาณควบคุม(m)จนกว่าสัญญาณผลต่างจะหมดไป ดังนั้นถ้าค่าT_iน้อย action จะเกิดเร็วทำให้สัญญาณ e = 0 เร็ว แต่มีข้อเสียที่ว่า ถ้าT_i น้อยเกินไปจะทำให้ค่าของกระบวนการมีค่าต่ำ ซึ่งอาจทำให้ระบบขาดเสถียรภาพ (unstable)ได้ แต่ถ้าให้ T_iมากเกินไป ค่า stabilizing time ของกระบวนการจะยาวนานเกินไป

ในกระบวนการที่มีไทม์แล็กมีค่ามากจะทำให้ผลตอบสนองของระบบเกิดออสซิเลตได้ ดังนั้นอินติกรัลคอนโทรลจะไม่ถูกนำไปใช้โดดเดี่ยว แต่ถ้าจะนำไปใช้ก็จะใช้กับกระบวนการที่มีไทม์แล็กน้อย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่2.9 ผลตอบสนองของระบบ I คอนโทรลต่อการเปลี่ยนแปลงของอินพุท (θ_i) ในลักษณะยูนิตสเตป

2.3.3 ปรีอบพอร์ชั่นแนล-อินติกรัลคอนโทรล (Proportional-Integral control) การคอนโทรลแบบนี้เราจะหาสัญญาณควบคุม (m) เป็นไปตามสมการที่ (2.6)

$$m = K_P e + K_P K_I \int_0^t edt + m(o) \qquad .....(2.6)$$

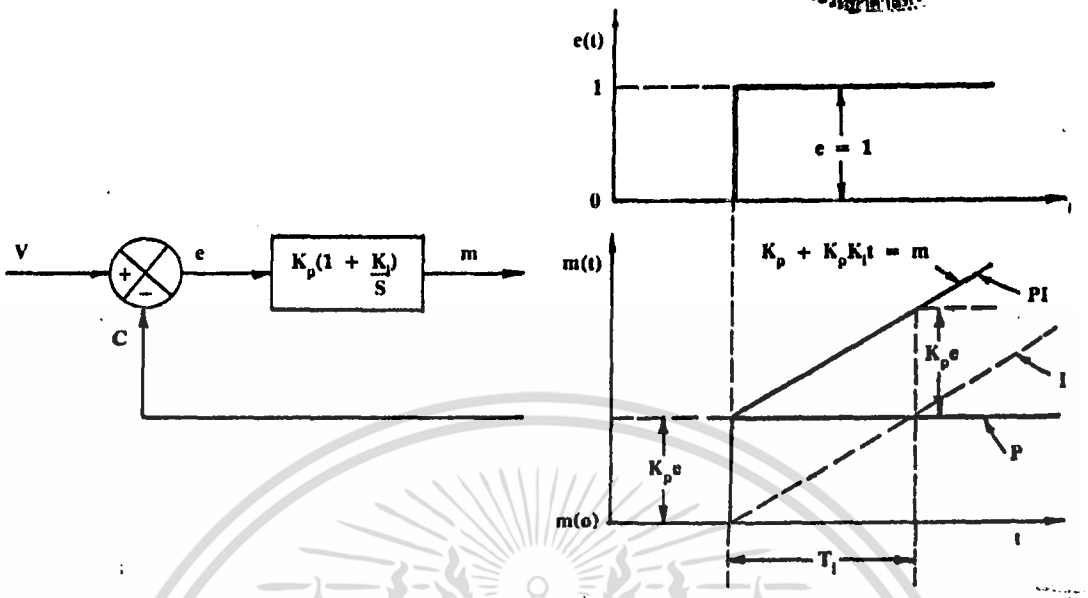
ข้อดีของ PI คอนโทรล

คือ การคอนโทรลแบบนี้จะให้คุณสมบัติของทั้งแบบปรีอบพอร์ชั่นแนล และแบบอินติกรัล ซึ่งสามารถทำให้ค่าออฟเซ็ตในแบบปรีอบพอร์ชั่นแนลหมดไป และค่า K_I (กำลังขยายของการอินติเกรต) สามารถจะปรับค่าได้อย่างอิสระ

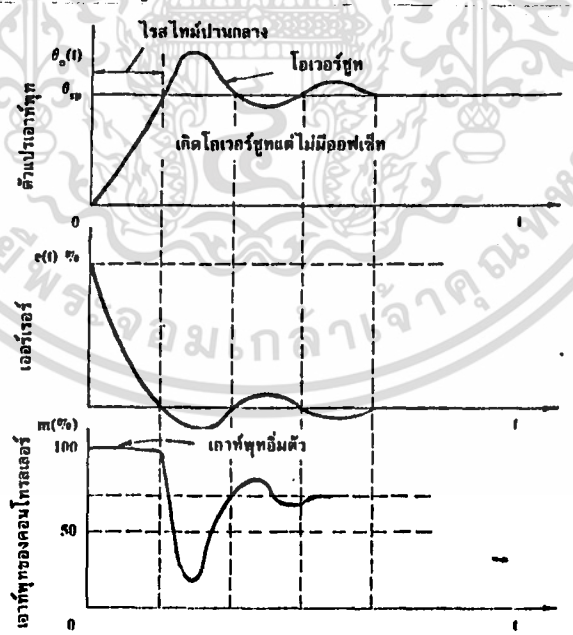
ข้อเสียของ PI คอนโทรล

คือระบบอาจไม่เสถียรกรณีค่าI_Iน้อย และไม่เหมาะสมกับกระบวนการที่มีไทม์แล็กมากๆ เพราะการตอบสนองของตัวแปรกระบวนการ (C)ช้ามาก ไม่สามารถที่จะแก้ไขข้อผิดพลาดได้ทันเวลา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ระบบ PI คอนโทรลจะใช้ได้กับระบบที่มีการเปลี่ยนแปลงโหลดมากๆได้ แต่โหลดควรจะต้องเปลี่ยนแปลงอย่างช้าๆเมื่อเทียบกับT_I เพื่อป้องกันไม่ให้เกิดการออสซิเลตเนื่องจากโอเวอร์ชูทของการอินติกรัล และระบบมักให้โอเวอร์ชูทสูงก่อนที่จะเข้าสามารถเซ็ทติ้งไทม์ ผลตอบสนองของระบบ PI คอนโทรลต่อการเปลี่ยนแปลงของสเตปอินพุท แสดงดังรูป 2.10



รูปที่ 2.10 ผลตอบสนองของระบบ PI คอนโทรลต่อการเปลี่ยนแปลงของสเตปอินพุท(θ_i)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

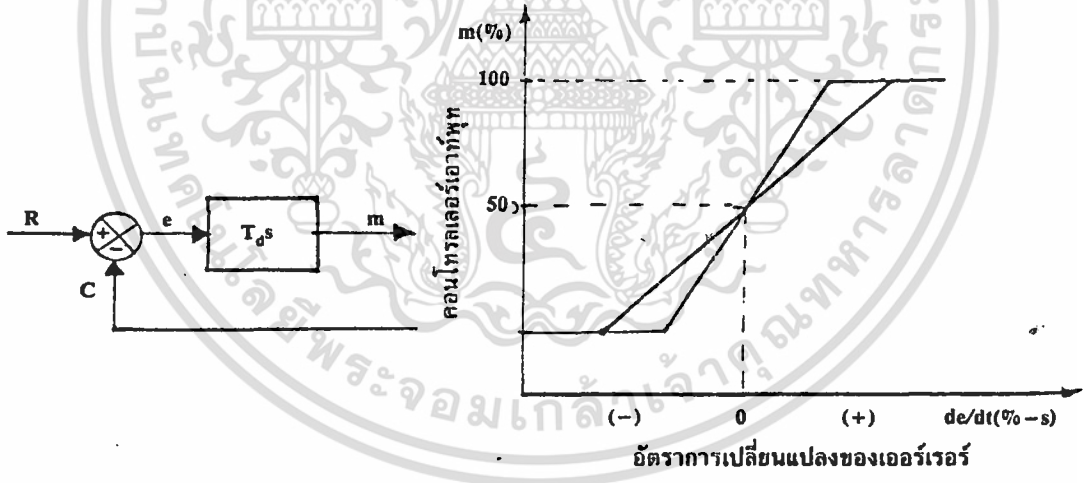
2.3.4 ดิริเวทีฟคอนโทรล (Derivative control)

การควบคุมแบบนี้เหมาะกับกระบวนการที่มีไทม์แล็กมากๆ เพราะสามารถแก้ข้อผิดพลาดโดยการกระทำล่วงหน้าก่อนที่จะมีความผิดพลาดเกิดขึ้น สัญญาณการควบคุมจะแปรตามอัตราการเปลี่ยนแปลงของสัญญาณเออร์เรอร์(e) ดังแสดงในรูปที่2.11 การคอนโทรลแบบดิริเวทีฟ หรือเรทคอนโทรลนี้ไม่สามารถนำไปใช้ในงานเดี่ยวโดดๆได้ เพราะว่าเมื่อเออร์เรอร์เป็นศูนย์หรือมีค่าคงที่คอนโทรลเลอร์จะไม่ให้เอาท์พุทเลย (หมายถึง m = 0 ด้วย)

ดังนั้นสมการของ m หาได้ดังนี้

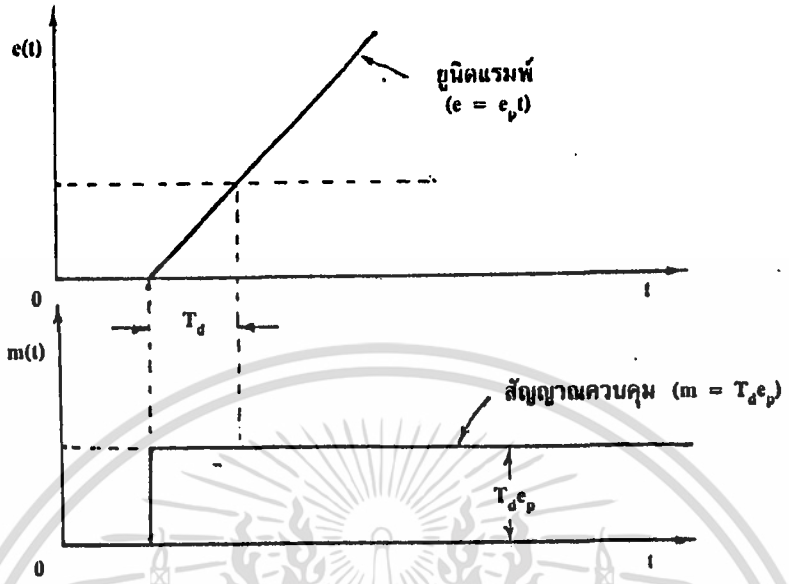
$$m = T_d \frac{de}{dt} + m(0)$$

เมื่อT_d คือค่าคงขยายคงที่ของดิริเวทีฟคอนโทรล หรืออาจเรียกอย่างหนึ่งว่า ดิริเวทีฟไทม์ (มีหน่วยเป็นนาที)

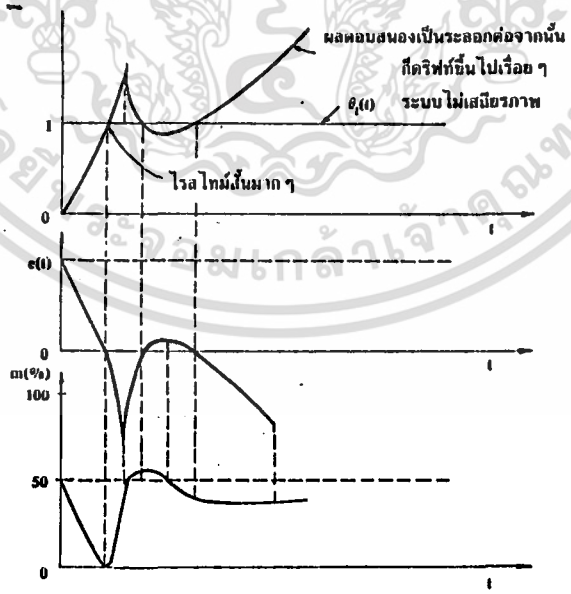


รูปที่ 2.11 ตัวคอนโทรลเลอร์แบบดิริเวทีฟคอนโทรล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



T_d คือเวลาที่สัญญาณควบคุมนำหน้า(lead) สัญญาณเออร์เรอร์อยู่ ค่าสัญญาณเออร์เรอร์(e) ที่เป็นยูนิตแรมพ์เป็นสัญญาณเออร์เรอร์ที่เกิดจากกระบวนการที่มีไทม์แล็กมากๆ D-แอกชั่น จะมีผลต่อเมื่อสัญญาณเออร์เรอร์(e) มีการเปลี่ยนแปลงเท่านั้น ถ้าสัญญาณเออร์เรอร์คงที่ D-แอกชั่นจะหมดปฏิกิริยาทันที



รูป2.12 ผลตอบสนองของระบบ D- คอนโทรลต่อการเปลี่ยนแปลงของสเตปอินพุท

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

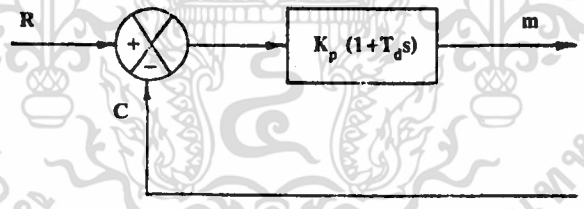
เอาท์พุทของคอนโทรลเลอร์จะขึ้นอยู่กับการเปลี่ยนแปลงของเออร์เรอร์ และจะไม่ขึ้นอยู่กับค่าเออร์เรอร์ที่ไม่มีการเปลี่ยนแปลง

ข้อเสียของดิริเวทีฟคอนโทรล

คือมีความไวต่อสัญญาณเออร์เรอร์(e)มาก โดยเฉพาะในกรณี t_d มีค่ามาก จะทำให้ระบบไม่เสถียรจึงไม่เหมาะกับระบบที่มีไทม์แล็กน้อยๆ และกระบวนการที่มีการเปลี่ยนแปลงได้ง่าย เช่นระบบที่มีการควบคุมการไหลหรือระบบที่มีการควบคุมความดัน เป็นต้น

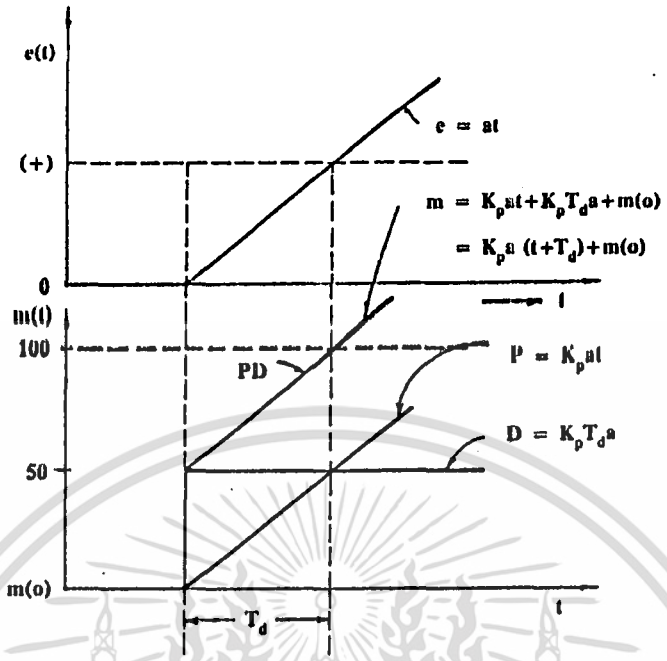
2.3.5 **ปรีอบพอร์ชั่นแนล-ดิริเวทีฟคอนโทรล** (Proportional-Derivative control) PD-คอนโทรลเป็นการร่วมระหว่างPและ D-แอกชั่น ดังแสดงในรูปที่2.13

$$m = K_p e + K_p T_d \frac{de}{dt} + m(0) \qquad .....(2.7)$$

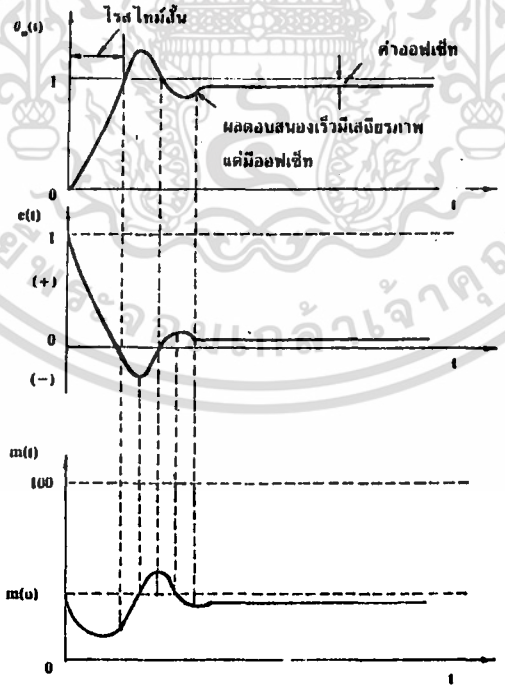


รูปที่ 2.13 ตัวคอนโทรลเลอร์แบบ PD คอนโทรล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ถ้ากำหนดสัญญาณเออร์เรอร์(e) ที่เวลา t
สัญญาณควบคุม(m)จะมีเวลาเท่ากับ(t + T_d) คือสัญญาณควบคุมจะนำหน้า(lead)
สัญญาณเออร์เรอร์ (e) อยู่เท่ากับ T_d



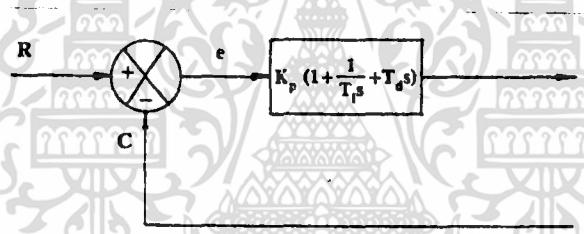
รูปที่ 2.14 ผลตอบสนองของระบบ PD-คอนโทรลต่อการเปลี่ยนแปลงของสเตปอินพุท (θ_i)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในการคอนโทรลแบบPD-คอนโทรลนี้ ไม่สามารถที่จะแก้ไขออฟเซ็ตของปร็อปพอร์ชั่นแนลคอนโทรลเลอร์ได้ แต่สามารถที่จะควบคุมระบบที่มีโหลดเปลี่ยนแปลงรวดเร็วได้

2.3.6 ปร็อบพอร์ชั่นแนลอินติกรัลดิริเวทีฟคอนโทรล (Proportional-Integral-Derivative control)

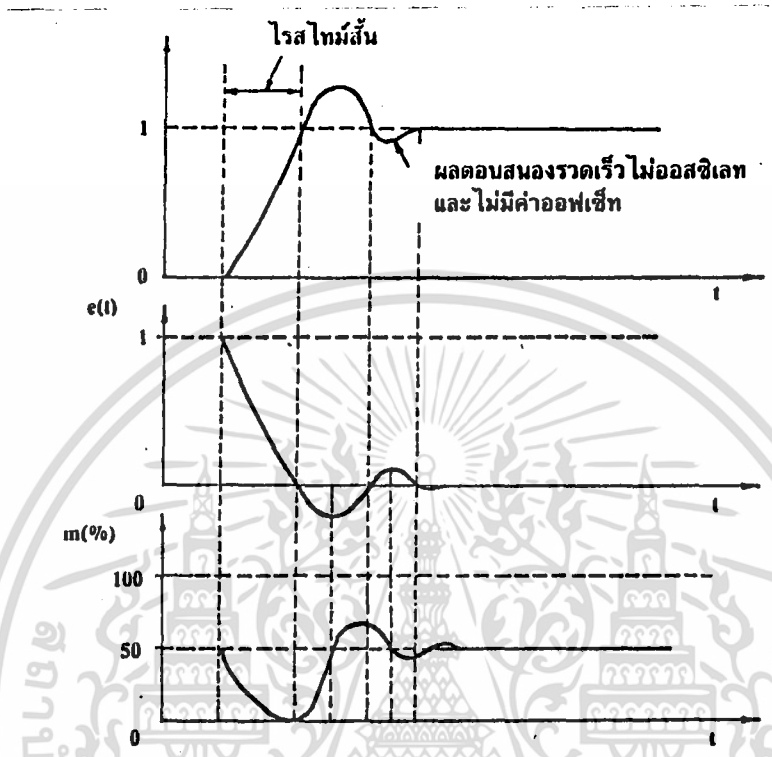
PID-คอนโทรล คือการรวมระหว่างการควบคุมแบบ P-คอนโทรล, I-คอนโทรล และ D-คอนโทรลเข้าด้วยกัน ดังแสดงในรูปที่2.15



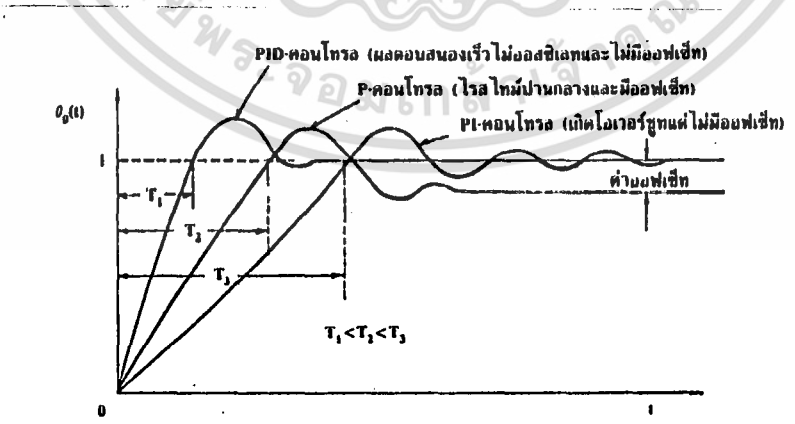
รูปที่ 2.15 PID คอนโทรลเลอร์

การคอนโทรลแบบPIDสามารถแก้ออฟเซ็ตของปร็อปพอร์ชั่นแนลคอนโทรล และลดโอเวอร์ชูทที่จะทำให้เกิดการออสซิเลทเนื่องจากการอินติกรัลคอนโทรล และจะให้ผลตอบสนองได้รวดเร็วตามคุณสมบัติของดิริเวทีฟคอนโทรล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูป2.16 ผลตอบสนองของระบบ PID คอนโทรลต่อการเปลี่ยนแปลงของสเตปอินพุท (θ_i)

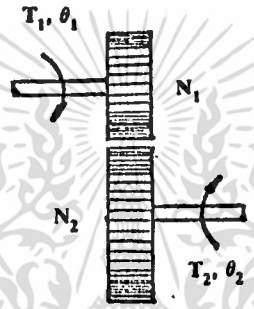


รูป2.17 เปรียบเทียบผลตอบสนองของระบบการคอนโทรลต่างๆ ต่อการเปลี่ยนแปลงสเตปอินพุท

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.4 ระบบเกียร์

ระบบเกียร์คานงัดหรือสายพานในลูกรอกเป็นอุปกรณ์เครื่องกลซึ่งสามารถส่งพลังงานจากส่วนหนึ่งของระบบไปยังส่วนอื่นๆได้ในรูปของ พลังงาน แรงบิด ความเร็ว และการเคลื่อนที่ นอกจากนั้นอุปกรณ์เหล่านี้ยังเป็นเสมือนอุปกรณ์สำหรับประสาน(matching) ให้สามารถใช้ส่งผ่านพลังงานให้ได้ค่าสูงสุด รูปที่ 2.18 แสดงถึงการคัพปลิ้งเกียร์ 2ตัวเข้าด้วยกัน แรงเฉื่อยและแรงเสียดทานของเกียร์จะไม่นำมาคิดในกรณีของเกียร์ในอุดมคติ



รูปที่2.18 ระบบการคัพปลิ้งของเกียร์ที่ใช้ในระบบการบังคับมอเตอร์

ความสัมพันธ์ระหว่างแรงบิด T_1และ T_2 ในการเคลื่อนที่เชิงมุม θ_1 และ θ_2 และจำนวนซี่ฟัน N_1 และ N_2 ระบบของเกียร์สามารถหาได้จากหลักเกณฑ์ต่อไปนี้

1 จำนวนซี่ฟันของเกียร์จะเป็นสัดส่วนกับรัศมี r_1และr_2 ของเกียร์ นั่นคือ

$$r_1 N_2 = r_2 N_1 \qquad (2.8)$$

2 ระยะทางการเคลื่อนที่ไปของเกียร์แต่ละตัวจะมีค่าเท่ากัน ดังนี้

$$\theta_1 r_1 = \theta_2 r_2 \qquad (2.9)$$

3 แรงงานที่ได้จากเกียร์ตัวหนึ่งจะเท่ากับแรงงานที่ได้จากเกียร์อีกตัวหนึ่ง เนื่องจากสมมุติให้ว่าไม่มีการสูญเสียแรงงาน ดังนั้น

$$T_1\theta = T_2\theta \qquad (2.10)$$

ถ้าความเร็วเชิงมุมของเกียร์ทั้งสองคือ ω_1 และω_2 ในรูป 2.18 สมการ (2.8),(2.9)และ (2.10) เขียนใหม่ได้เป็น

$$\frac{T_1}{T_2} = \frac{\theta_2}{\theta_1} = \frac{N_1}{N_2} = \frac{\omega_2}{\omega_1} = \frac{r_1}{r_2}$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.5 อินคริเมนท์ เอนโคดเดอร์

เป็นเอนโคดเดอร์ที่นิยมใช้กันมากในระบบควบคุมตำแหน่งและความเร็ว โดยเอนโคดเดอร์จะสร้างสัญญาณ ณ พัลส์ที่แปรผันตรงกับการหมุนของเพลา การบอกตำแหน่งทำได้โดยการนับจำนวนพัลส์ที่เกิดขึ้น และการบอกความเร็วทำได้โดยการวัดความถี่พัลส์ที่เกิดขึ้น ซึ่งบึงเพลาของเอนโคดเดอร์หมุนเร็วเท่าไร พัลส์ที่ออกมาก็บึงถี่เท่านั้น

ส่วนประกอบภายในของเอนโคดเดอร์ชนิดนี้มีด้วยกัน 4ส่วน คือ

1 แหล่งกำเนิดแสง(Light source)
2 จานหมุน (Rotary disk)
3 แผ่นจานที่อยู่กับที่ (stationary mask)
4 ตัวเซ็นเซอร์ (sensor)

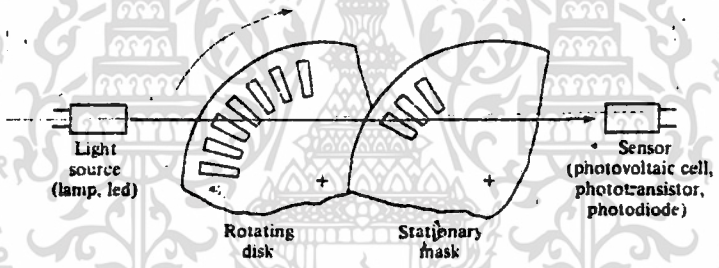


Figure 4-42 Typical incremental optomechanics.

รูปที่2.19 ส่วนประกอบภายในของอินคริเมนท์ เอนโคดเดอร์

บนแผ่นจานหมุนทำเป็นช่องโดยรอบ ดังแสดงในรูป2.21 และบนแผ่นที่อยู่กับที่จะมีช่องสำหรับให้แสงผ่านหรือไม่ผ่านไปยังตัวเซ็นเซอร์ ถ้าเป็นเอนโคดเดอร์ที่มีความละเอียดต่ำ (Low resolution) ก็ไม่จำเป็นต้องมีแผ่นที่อยู่กับที่ก็ได้ และอาจใช้หลอดไฟเป็นแหล่งกำเนิดแสงก็ได้

2.5.1 ความละเอียดของอินคริเมนท์ เอนโคดเดอร์

ความละเอียดของเอนโคดเดอร์หมายถึงจำนวนคาบเวลาของสัญญาณเอาท์พุทต่อการหมุนของเพลา1รอบ ซึ่งบอกเป็นจำนวนพัลส์ต่อรอบ หรือจำนวนไซเกิลต่อ 360 องศา(เป็นมุมทางเชิงกล หรือไซเกิลต่อองศา) เอนโคดเดอร์ที่ใช้กันทั่วๆไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

มีค่าความละเอียดตั้งแต่ 15 ถึง 10,000 พัลส์ต่อรอบ

ในทางปฏิบัติ ถ้าเราต้องการให้แสงที่ผ่านช่องไปยังเซ็นเซอร์เป็นเส้นตรงพร้อมๆกัน(collimation) ก็สามารถทำได้โดยการใช้เลนซ์ หรือพาราโบริคแฟคเตอร์ ดังแสดงในรูปที่ 2.20

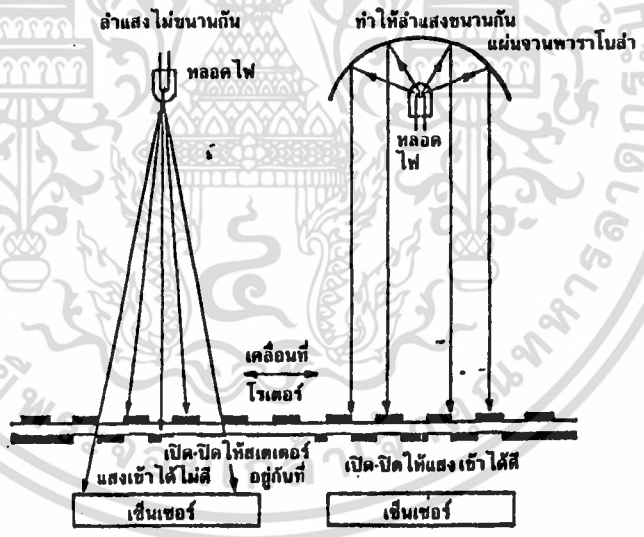
จำนวนพัลส์ต่อ 1 รอบของสัญญาณที่เอนโคดเดอร์สร้างออกมาจะเท่ากับจำนวนช่องว่างบนแผ่นจานหมุน และความกว้างของช่องว่างกับความกว้างของแถบทึบระหว่างช่องว่างจะเท่ากัน เพราะฉะนั้นเราสามารถคำนวนหาความกว้างของช่องว่าง (ω)ได้จาก

$$\omega = \frac{\pi D}{2N} \qquad \text{.....(2.11)}$$

เมื่อ D = เส้นผ่าศูนย์กลางเฉลี่ยของแผ่นจาน

N = จำนวนความละเอียดเป็นพัลส์ต่อรอบ

ω = ความกว้างของช่องว่าง



รูปที่2.20 แสดงถึงผลของแสงที่เดินในแนวเดียวกันและแสงที่แตกกระจาย

ค่าของตัวตัวแปรของสมการ (2.31)นี้หาได้จากรูป 2.23 ถ้าให้ D เป็นเส้นผ่าศูนย์กลางแผ่นหมุนของเอนโคดเดอร์ค่าประมาณที่ใกล้เคียงมากของค่าความกว้าง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

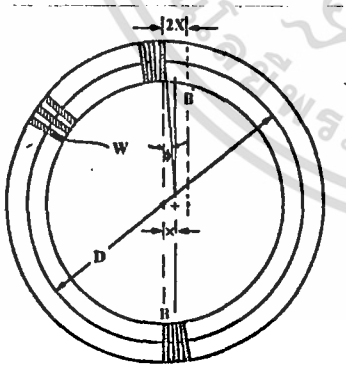
ของช่องว่างแสดงได้ดังนี้

$$\omega = \frac{0.75\ \pi D}{2N} \qquad \text{.....(2.12)}$$

ตัวอย่าง ถ้าแผ่นหมุนของเอนโคดเดอร์มีเส้นผ่าศูนย์กลาง 2 นิ้ว ถ้าต้องการค่าความละเอียด 200 พัลส์ต่อรอบ จะได้ความกว้างของช่องว่างเท่ากับ 0.002356 นิ้ว

2.5.2 เอาท์พุทของเอนโคดเดอร์

โดยทั่วๆไปแล้ว สัญญาณเอาท์พุทที่ออกจากเอนโคดเดอร์โดยตรงจะมีระดับไม่เพียงพอในการควบคุมหรือสำหรับการประมวลสัญญาณ ดังนั้นจึงต้องมีวงจรขยายและแปลงรูปร่างลูกคลื่นสัญญาณต่อไว้ในเอนโคดเดอร์เสมอ สัญญาณลูกคลื่นที่ได้จากตัวเซ็นเซอร์ปกติแล้วจะเป็นรูปสัญญาณสามเหลี่ยม หรือรูปสัญญาณซายน์ ขึ้นอยู่กับความละเอียดที่ต้องการ รูปสัญญาณเหล่านี้สามารถทำให้เป็นรูปสัญญาณสี่เหลี่ยมได้โดยการต่อตัวคอมพาราเตอร์เข้ากับลิเนียร์แอมปลิไฟของเอนโคดเดอร์ ก็จะได้เอาท์พุทเป็นลูกคลื่นสี่เหลี่ยมตามต้องการ



เออร์เรอร์จะเป็นลิเนียร์ = 2X/W

เมื่อ W = D/ความละเอียด = ความกว้างของพัลซ์

X = ความแตกต่างไปจากศูนย์กลาง(ดูจากรูป)

2X = ผลรวมคาบเวลาของเออร์เรอร์ในช่วงการหมุน 180 องศา

D = เส้นผ่าศูนย์กลางเฉลี่ยของแผ่นจาน

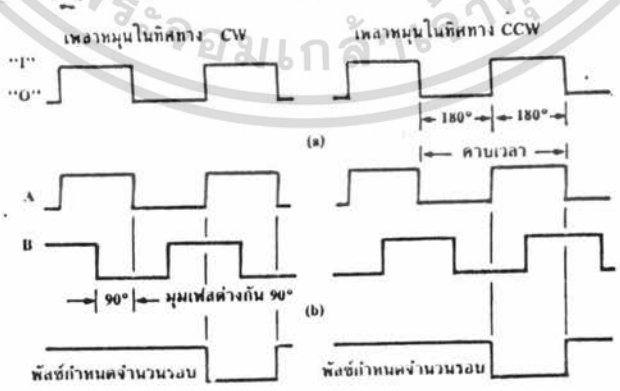
รูปที่ 2.21 ความสัมพันธ์ระหว่างความเป็นลิเนียร์กับความไม่ได้ศูนย์กลาง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 2.23 (ก) แสดงลูกคลื่นเอาท์พุทสี่เหลี่ยมของเอนโคดเดอร์ชนิด 1 ช่อง ไม่ว่าเพลาจะหมุนไปในทิศทางใดก็ได้สัญญาณออกมาเหมือนกัน จึงเหมาะที่จะใช้กับงานที่ไม่กำหนดทิศทางเท่านั้น

รูป 2.22 โรตารี่เอนโคดเดอร์ที่มีเอาท์พุทเป็นไบนารี่
(ก) ลักษณะตัวอย่างแบบหนึ่ง
(ข) แผ่นจานภายในที่ประกอบด้วยข้อมูลไบนารี่

ในรูปที่ 2.23(ข) แสดงสัญญาณ 2ชุดที่ได้จากเอนโคดเดอร์ชนิด 2 ช่อง เฟสของสัญญาณ 2ช่องนี้จะต่างกัน 90 องศา ในทางไฟฟ้าเราเรียกสัญญาณ 2ช่องนี้ว่าเป็นควอดราเจอร์ (quadrature) ซึ่งเหมาะที่จะใช้ในการรับรู้ทิศทางการหมุนของเพลาหรือใช้ควบคุมระบบที่ซับซ้อนอื่นๆ จากสัญญาณในรูป 2.23(ข)จะเห็นได้ว่าสัญญาณทั้ง 2 ช่อง จะเริ่มจาก 0 ถึง 1 หรือจาก 1 ถึง 0 ซึ่งขึ้นอยู่กับทิศทางการหมุนของเพลาของเอนโคดเดอร์



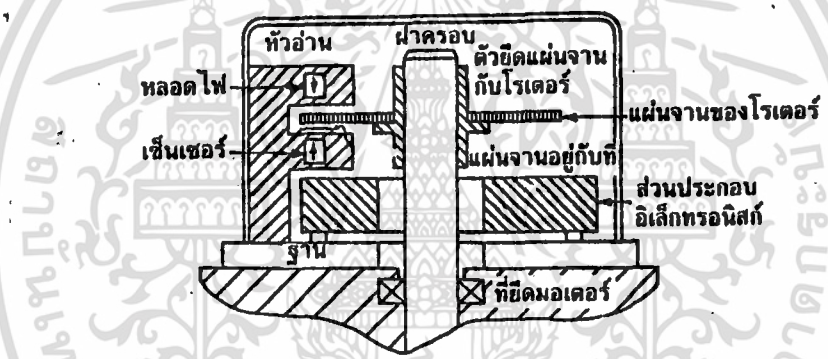
รูปที่ 2.23 (ก) ตัวอย่างลูกคลื่นเอาท์พุทสี่เหลี่ยมของอุปกรณ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอนโคดเดอร์ช่องเดียว(ไบไดเร็คชั่น)

(ข) ตัวอย่างสัญญาณเอนโคดเดอร์ 2ช่อง มีมุมเฟสต่างกัน 90องศา(สองทิศทาง)

ในอินคริเมนท์เอนโคดเดอร์บางชนิดจะมีพัลส์แสดงถึงจำนวนรอบของการหมุนสำหรับใช้เป็นศูนย์ในการอ้างอิงพัลส์ที่ใช้แสดงจำนวนรอบที่จะเกิดขึ้น 1พัลส์ต่อรอบ โดยทั่วไปแล้วจะใช้บอกถึงตำแหน่งเชิงกล หรือใช้เป็นสัญญาณเคลียร์จำนวนที่นับไว้ในหน่วยเก็บข้อมูล

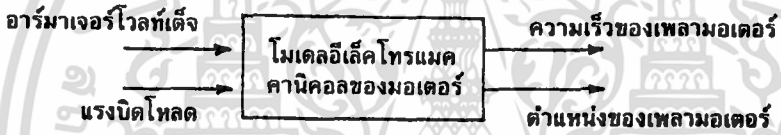


รูป2.24 ตัวอย่างส่วนประกอบของเอนโคดเดอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.6 โมเดลคณิตศาสตร์ของดีซีมอเตอร์

ดีซีมอเตอร์ที่ใช้ร่วมกับดีซีแอมปลิไฟทั้งในระบบการบังคับตำแหน่ง และการบังคับความเร็วมักจะได้รับการประยุกต์ใช้เป็นส่วนประกอบสร้างกำลังงานในระบบการนำร่องและระบบการบังคับต่างๆ และเนื่องจากวิทยาการเกี่ยวกับสารแม่เหล็กและการขยายด้วยโซลิตสเตท ทำให้ดีซีมอตอร์แบบแม่เหล็กถาวรได้รับความนิยมใช้เป็นส่วนประกอบการขับเคลื่อนในระบบการบังคับแบบปิดลูปต่างๆ มากขึ้น การออกแบบ และการชดเชยระบบดังกล่าวได้อย่างเหมาะสม จะต้องใช้โมเดลคณิตศาสตร์ของส่วนประกอบทั้งหมดในระบบ ในหัวข้อนี้เราจะได้พัฒนาลิเนียร์โมเดลของดีซีมอเตอร์แบบแม่เหล็กถาวรและแบบฟิลด์แยกกระตุ้น



2.6.1 โมเดลอิเล็คโทรแมคคานิคอล

ส่วนสำคัญของดีซีมอเตอร์แบบฟิลด์แยกกระตุ้นมีโมเดลดังแสดงในรูป2.24

R_a: ความต้านทานของอาร์มาเจอร์

L_a: อินดัคแต้นซ์ของอาร์มาเจอร์

V_g: โวลท์เต็จกำเนิดในอาร์มาเจอร์

R_f: ความต้านทานของฟิลด์

L_f: อิดัดแต้นซ์ของฟิลด์

ϕ : ช่องว่างอากาศของเส้นแรงสนามแม่เหล็ก

ω : ความเร็วของเพลาอาเมเจอร์

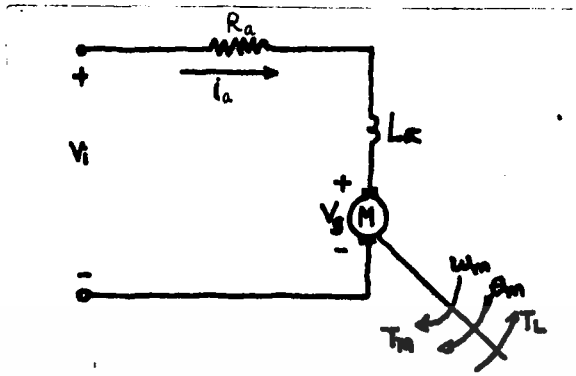
T_g: แรงบิดที่พัฒนาขึ้นในมอเตอร์

T_f: แรงบิดเสียดทานของมอเตอร์

T_j: แรงเฉื่อยของมอเตอร์

T_l: แรงบิดโหลดบนเพลาของมอเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่2.25 โมเดลของดีซีมอเตอร์แบบฟิลด์แยกกระตุ้น

ขั้นแรกเราจะหาสมการพื้นฐานโมเดลของดีซีมอเตอร์ได้จากลูปของอาเมเจอร์

$$V_i(t) = R_a I_a + L_a \frac{di_a}{dt} + V_g(t) \qquad .....(2.13)$$

เทอมโวลต์เตจ $V_g(t)$ ในสมการ(2.13) คือโวลต์เตจย้อนกลับของมอเตอร์ซึ่งเกิดขึ้นเมื่อเส้นลวดตัวนำของอาเมเจอร์หมุนตัดเส้นแรงแม่เหล็กซึ่งเกิดขึ้นโดยกระแสของฟิลด์(I_f) ตามกฎของของฟาราเดย์ลูปของเส้นลวดตัวนำหมุนในฟิลด์แม่เหล็กคงที่จะมีการเหนี่ยวนำโวลต์เตจขึ้นในขดลวดนั้น

$$V(t) = \frac{d\ \lambda(t)}{dt} \qquad .....(2.14)$$

เมื่อ $\lambda(t)$เส้นแรงแม่เหล็กที่รั่วไปยังขดลวด และ t คือเวลาในการหมุนของคอมนิวเทเตอร์ การควบคุมวงจรของแต่ละส่วนของตัวนำในโรเตอร์จะเกิดโวลต์เตจขึ้นในส่วนของตัวนำนั้นตามสมการ(2.14) เมื่อ $\frac{d\ \lambda(t)}{dt}$ จะเป็นสัดส่วนต่อเส้นแรงแม่เหล็กในช่องว่างอากาศ และความเร็วเชิงมุม $\omega(t)$ หรือเราจะได้ว่า

$$V_g(t) = K\lambda(t)\omega(t) \qquad .....(2.15)$$

สมมุติให้กระแสของฟิลด์มีค่าคงที่ และไม่คิดถึงส่วนการเปลี่ยนแปลงในเส้นแรงฟิลด์เนื่องจากอาร์เมเจอร์รีแอคชั่นเส้นแรงฟิลด์ก็จะมีค่าคงที่ ดังนั้นสมการ(2.15)ก็จะกลายเป็น

$$V_g(t) = K_g\omega(t) \qquad .....(2.16)$$

เมื่อเราสมมุติให้เส้นแรงของฟิลด์มีค่าคงที่แรงบิดของแม่เหล็กไฟฟ้าซึ่งเกิดขึ้นแก่ โรเตอร์ของมอเตอร์จะเป็นสัดส่วนกับกระแสอาร์เมเจอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

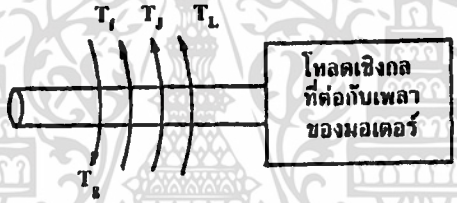
$$T_g(t) = K_t i_a(t) \qquad .....(2.17)$$

เมื่อ K_t คือค่าคงที่ของแรงบิดของมอเตอร์

กำลังงานเชิงกลที่เกิดขึ้นในโรเตอร์คือผลคูณของแรงบิดที่เกิดขึ้นและความเร็วเชิงมุม

$$P_g(t) = T_g(t)\omega(t) \qquad .....(2.18)$$

กำลังงานเชิงกลที่เกิดขึ้นในโรเตอร์นี้จะจ่ายๆไปยังโหลดที่ต่ออยู่กับเพลาของมอเตอร์ แต่กำลังงานนี้บางส่วนจะสูญเสียไปในมอเตอร์ การสูญเสียเนื่องมาจากแรงเสียดทานหมายถึงความหน่วงเนื่องมาจากลมที่มีต่อโรเตอร์ แรงเสียดทานตัวรองรับโรเตอร์ กระแสที่ไหลวนในเหล็กของโรเตอร์และอีสเทรีซีส จากรูปที่ 2.26 แสดงให้เห็นถึงแรงบิดต่างๆที่เกิดขึ้นต่อโหลดของมอเตอร์



รูปที่2.26 แสดงถึงแรงบิดต่างๆที่เกิดขึ้นต่อโหลดของมอเตอร์

$T_g(t)$: แรงบิดของมอเตอร์

$T_f(t)$: แรงบิดที่ต้องชนะการสูญเสียเนื่องมาจากความเสียดทาน

$T_J(t)$: แรงบิดเพื่อใช้เพิ่มอัตราเร่งแก่ความเฉลี่ยของโหลด

$T_L(t)$: แรงบิดโหลด

แรงบิดของมอเตอร์ในช่วงเวลาใดๆ จะต้องเท่ากับและมีทิศทางตรงข้ามกับผลรวมของแรงบิด $T_f(t)$, $T_J(T)$และ $T_L(t)$ ดังนั้น

$$T_g(t) = T_f(t) + T_L(T) + J\frac{d\ \omega(t)}{dt} \qquad .....(2.19)$$

เมื่อ J คือผลรวมของโมเมนท์แรงเฉื่อยของโรเตอร์และโหลดที่ต่ออยู่ที่เพลาของมอเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ผลรวมของแรงบิดเสียดทานที่ประกอบกันขึ้นที่เพลาของมอเตอร์ซึ่งเป็นลิเนียร์ฟังก์ชั่นกับความเร็วเชิงมุมของมอเตอร์ส่วนประกอบของวิสคอสฟริคชั่น และมักอยู่ในเทอมที่แยกออกจากฟริคชั่นอื่นๆ ซึ่งแสดงได้ดังสมการต่อไปนี้

$$T_g(t) = T_f(t) + T_L(t) + J\frac{d\omega(t)}{dt} + B\omega(t) \quad ....(2.20)$$

เมื่อB คือวิสคอสฟริคชั่นของมอเตอร์และโหลดที่ต่ออยู่กับเพลาของมอเตอร์ $T_f(t)$ คือผลรวมของฟริคชั่นของโหลดและของมอเตอร์ทั้งหมด มีแรงต้านของลมและการสูญเสียกำลังในเหล็กของเพลามอเตอร์ ยกเว้นวิสคอสฟริคชั่น

สมการที่(2.13),(2.16),(2.17)และ(2.20)เป็นชุดของสมการพื้นฐานของดีซีมอเตอร์โมเดล และจากสมการเหล่านี้เราสามารถหาสมการทรานสเฟอร์ฟังก์ชั่นของดีซีมอเตอร์ได้ โดยการใส่ลาปาสทรานสฟอร์มทั้งสองข้างของชุดการพื้นฐาน และเขียนใหม่ได้เป็น

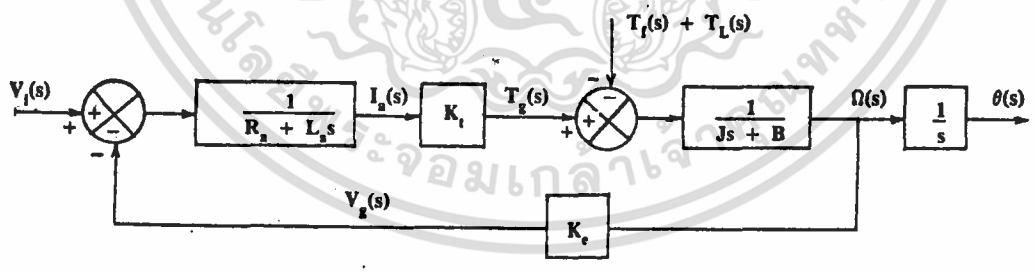
$$V_i(s) - V_g(s) = (R_a + sL_a)I_a(s) \quad .....(2.21)$$

$$V_g(s) = K_e\omega(s) \quad .....(2.22)$$

$$T_g(s) = K_t I_a(s) \quad .....(2.23)$$

$$T_g(s) - T_f(s) - T_L(L) = (B + sJ)\omega(s) \quad .....(2.24)$$

บล็อกไดอะแกรมที่แสดงถึงสมการพื้นฐานเหล่านี้แสดงได้ดังรูปที่2.27



รูปที่2.27 บล็อกไดอะแกรมของของดีซีมอเตอร์โมเดล

ข้อควรสังเกตุ

สมมุติว่าโวลต์เตจที่ป้อนให้กับวงจรอาร์เมเจอร์ของมอเตอร์มีค่าคงที่ ดังนั้น มอเตอร์จะหมุนด้วยความเร็วคงที่คือทำงานด้วยสภาวะที่สงบนิ่งด้วยโหลดที่คงที่ กำลังงานเชิงกลที่เกิดขึ้นโดยโรเตอร์หาได้จากสมการ (2.18) และเมื่อรวม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กับสมการ(2.17) ก็จะได้

$$P_g = T_g\omega = K_t I_a \omega \qquad .....(2.25)$$

เมื่อทุกเทอมในสมการสุดท้ายมีค่าคงที่เนื่องจากมอเตอร์ทำงานอยู่ที่สภาวะสงบนิ่ง กำลังไฟฟ้าที่ถูกดูดกลืนโดยอาร์เมเจอร์จะต้องเท่ากับผลคูณของโวลต์เตจคร่อมอาร์เมเจอร์และกระแสที่ไหลผ่าน ดังนั้น

$$P = V_g I_a = K_e \omega I_a \qquad .....(2.26)$$

ดังนั้นเราจะได้ว่า กำลังงานเชิงกลที่เกิดขึ้นต้องเท่ากับกำลังงานไฟฟ้าที่ถูกดูดกลืนในโรเตอร์ คือ สรุปได้ว่า $K_t = K_e$

2.6.2 ทรานเฟอร์ฟังก์ชั่นของดีซีมอเตอร์

บล็อกไดอะแกรมของรูปที่2.27 แสดงถึงระบบที่มีสองอินพุท และมีเอาท์พุทเป็นทั้งความเร็วเชิงมุม และการเคลื่อนที่แบบเชิงมุม จากรูปที่2.27 เราสามารถเขียนความเร็วเอาท์พุทของระบบได้เป็น

$$\omega(s) = G_1(s)V_1(s) + G_2(s)[T_f(s) + T_L(s)] \qquad ....(2.27)$$

เมื่อ

$$G_1(s) = \frac{\omega(s)}{V_1(s)} \;\Big|\; T_f(s) + T_L(s) = 0 \qquad ....(2.28)$$

$$G_2(s) = \frac{\omega(s)}{T_f(s) + T_L(s)} \;\Big|\; V_1(s) = 0 \qquad ....(2.29)$$

$G_1(s)$ คือทรานสเฟอร์ฟังก์ชั่นระหว่างโวลต์เตจและความเร็ว

$$G_1(s) = \frac{\omega(s)}{v_1(s)} = \frac{K_t}{(L_a s + R_a)(Js + B) + K_t K_e}$$

$$= \frac{K_m}{\alpha S^2 + \beta S + 1} \qquad .....(2.30)$$

เมื่อ

$$K_m = \frac{K_t}{R_a B + K_t K_e}$$

$$\alpha = \frac{L_a J}{R_a B + K_t K_e}$$

$$\beta = \frac{R_a J + L_a B}{R_a B + K_t K_e}$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่2.28 ดีซีมอเตอร์ที่ใช้ควบคุมเตียงคนไข้ซึ่งเป็นระบบควบคุมแบบมีการป้อนกลับ

สมการที่(2.30)เป็นโวลต์เตจทรานสเฟอร์ฟังก์ชั่นของดีซีมอเตอร์ เมื่อสมมุติว่าT_f และ T_L มีค่าเป็นศูนย์ ดังนั้นสมการที่(2.30)สามารถ เขียนใหม่ได้เป็น

$$G_1(s) = K_t / R_a B(1 + \tau_e s)(1 + \tau_m s) + K_t K_e$$

เมื่อ

$\tau_e = L_a/R_a$ = ไทม์คอนสแตนท์ทางไฟฟ้า

$\tau_m = J/B$ = ไทม์คอนสแตนท์ทางเชิงกล

ถ้าอินดัคแตนท์ของอาร์เมเจอร์มีน้อยจะสามารถตัดไทม์คอนสแตนท์ทางไฟฟ้าทิ้งได้ และสมการที่2.31จะเป็น

$$G_v(s) = \frac{\omega(s)}{V_i(s)} = \frac{K_t}{R_a(Js + B) + K_t K_e}$$

$$= \frac{K_m}{\tau s + 1} \qquad \ldots\ldots(2.31)$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เมื่อ

$$\tau = \frac{R_a J}{R_a B + K_t K_e}$$

ในสมการที่2.31 ค่าคงที่ K_a อาจเรียกได้ว่าเป็นค่าคงที่ของมอเตอร์ ทรานสเฟอร์ฟังก์ชั่นของแรงบิดโหลด $G_2(s)$ หาได้เป็น

$$G_2(s) = \frac{\omega(s)}{T_f(s)+T_L(s)}$$

$$= \frac{-1\,(Js + B)}{1 + [K_t K_e\,(Js + B)(L_a s + R_a)]}$$

$$G_2(s) = \frac{-(R_a/K_t)K_m[(L_a/R_a)s + 1]}{\alpha s^2 + \beta s + 1} \qquad \ldots\ldots(2.32)$$

ถ้าไม่นำเอาอินดัคแตนซ์ของอ่ร์เมเจอร์มาคิด สมการที่(2.32)ก็จะลดรูปเป็น

$$G_L(s) = \frac{\omega(s)}{T_f(s) + T_L(s)}$$

$$= \frac{-(R_a/K_t)K_m}{\tau s+1}$$

ตัวอย่างของทรานสเฟอร์ฟังก์ชั่น

ดีซีมอเตอร์ตัวเล็กๆมีข้อกำหนดดังต่อไปนี้

$V_1(max)$ = 30 โวลต์

$I_a(max)$ = 2 แอมแปร์

R_a = 3 โอห์ม

L_a = 6 มิลลิเฮนรี่

K_t = 50 10^{-3} นิวตัน-เมตร/แอม

T(rated) = 0.1 นิวตัน-เมตร

J_m = 40 10^{-3} กิโลกรัม-เมตร^2

B_m = 35 10^{-6} กิโลกรัม/เมตร/วินาที

ω_a(rate) = 300 เรเดียน/วินาที

ถ้าเราสมมุติว่าแรงเฉื่อยของโหลดและฟริคชั่นของโหลดที่มีต่อเพลาคือ

J_L = 60 10^{-6} กิโลกรัม-เมตร^2

B_L = 65 10^{-6} กิโลกรัม/เมตร/วินาที

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ดังนั้นทรานสเฟอร์ฟังก์ชั่นระหว่าง โวลต์เตจ-ความเร็ว หาได้จากสมการ(2.31)

$$G_1(s) = \frac{17.9}{2.7\ 10^{-4} + 0.1.7s + 1}$$

ถ้าไม่คิดค่าอินดักแตนซ์ ดังนั้นทรานสเฟอร์ฟังก์ชั่นจากสมการ (2.32) จะมีค่า

$$G_v = \frac{17.9}{0.107s + 1}$$

ถ้าสเตปโวลต์เตจ 10 โวลต์ ป้อนให้กับมอเตอร์ที่ t = 0 ความเร็วเชิงมุมจะเป็นฟังก์ชั่นกับเวลา และคำนวนได้ง่าย

$$\omega(s) = \frac{10}{s}\left[\frac{17.9}{0.107S + 1}\right]$$

$$= \frac{179}{S(0.107S + 1)}$$

ใส่อินเวอสลาปลาสทราสฟอร์มกับสมการสุดท้ายได้เป็น

$$\omega(t) = 179(1 - e^{-9.35t})$$

และเมื่อ $t \to \infty$ (สภาวะสงบนิ่ง) $\omega \to 179$ เรเดียน/วินาที ทรานสเฟอร์ฟังก์ชั่นระหว่างแรงบิดโหลดกับความเร็วได้เป็น

$$G_2(s) = \frac{-1074(2*10^{-3}S + 1)}{2.1*10^{-4}S^2 + 0.107S + 1}$$

เมื่อ L_a มีค่าน้อย

$$G_L(s) = \frac{-1074}{0.107S + 1}$$

ดังนั้นเมื่อแรงบิดโหลดเพิ่มขึ้นเป็นขั้นๆละ 0.05 นิวตัน-เมตร ความเร็วจะเปลี่ยนแปลงเป็น

$$\omega(s) = \frac{0.05}{S}\left[\frac{-1074}{0.107S + 1}\right]$$

$$= \frac{-53.7}{S(0.107S + 1)}$$

การเปลี่ยนแปลงความเร็วในไทม์โดเมนคือ

$$\omega(t) = -53.7(1 - e^{-9.35t})$$

ในสภาวะสงบนิ่งความเร็วจะลดเป็น 53.7 เรเดียน/วินาทีเมื่อแรงบิดโหลดเพิ่มขึ้น 0.05 นิวตัน-เมตร

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

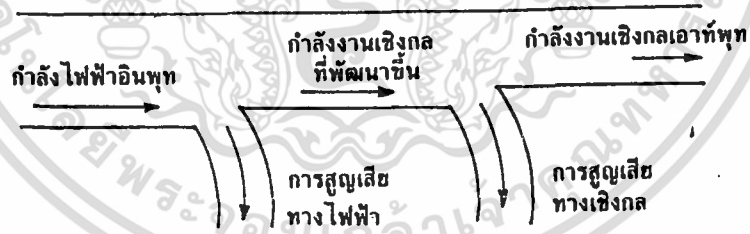
2.6.3 การสูญเสียกำลังงานของดีซีมอเตอร์

ดังที่เราได้ทราบกันมาแล้วว่าดีซีมอเตอร์มีหน้าที่ในการแปลงพลังงาน กล่าวคือจะแปลงพลังงานอินพุททางไฟฟ้าไปเป็นพลังงานเอาท์พุททางเชิงกล ในรูปที่ 2.28 แสดงถึงการไหลของพลังงานผ่านอุปกรณ์แปลงพลังงาน(มอเตอร์)

รูปที่2.29(ก) แสดงถึงการไหลของกำลังงานในดีซีมอเตอร์

ผลรวมของกำลังงานไฟฟ้าอินพุทเท่ากับ ผลคูณของอาร์เมเจอร์โวลต์เตจและกระแสอาร์เมเจอร์ดังนั้นผลรวมทางอินพุทเพาเวอร์มีค่าเท่ากับ

$$P_i(t) = V_i(t)i_a(t) \qquad .....(2.33)$$



รูปที่2.29(ข) แสดงถึงการไหลของกำลังงานในดีซีมอเตอร์

รวมสมการที่(2.13)และ(2.16)เข้าด้วยกัน เราจะได้สมการโวลต์เตจของอาร์เมเจอร์ซึ่งสัมพันธ์กับโวลต์เตจอินพุทอินพุทและกระแสอาร์เมเจอร์

$$V_i(t) = R_a i_a(t) + L_a \frac{di_a(t)}{dt} + K_a\omega(t) \qquad .....(2.34)$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แทนค่าสมการ(2.34)ลงในสมการ(2.33) จะได้นิพจน์ของกำลังงานอินพุทรวมในเทอมของกระแสอาร์เมเจอร์และความเร็วเชิงมุมของเพลา

$$P_i(t) = R_a i_a^2(t) + L_a i_a(t)\frac{di_a(t)}{dt} + K_e i_a \omega(t) \quad \text{.....(2.35)}$$

กระแสอาร์เมเจอร์สามารถเขียนให้อยู่ในเทอมของพารามิเตอร์เชิงกลของมอเตอร์ และแรงบิดของเพลาด้วยการรวมสมการ(2.17) และสมการ(2.20)

$$i_a(t) = \frac{1}{K_t}\left[J\frac{d\omega(t)}{dt} + B(t) + T_f(t) + T_L(t)\right] \quad \text{.....(2.36)}$$

ซึ่งเมื่อเราแทนค่าลงในสมการ(2.35) จะได้ดังต่อไปนี้

$$P_i(t) = R_a i_a^2(t) + \frac{K_e B\omega^2(t)}{K_t} + \frac{K_e \omega(t) T_f(t)}{K_t}$$

$$+ K_e \omega(t) T_L(t) + \frac{K_e}{K_t} J\omega(t)\frac{d\,\omega\,(t)}{dt} + L_a i_a(t)\frac{di_a(t)}{dt} \quad \text{.....(2.37)}$$

พิจารณาผลรวมของอินพุทเพาเวอร์ในสมการ(2.37)

เทอมแรก (ทางด้านขวามือ)

i^2R : การสูญเสียในขดลวดอาร์เมเจอร์เนื่องจากการไหลของกระแสอาร์เมเจอร์ การสูญเสียทางเชิงกลในตัวมอเตอร์ ประกอบด้วยการสูญเสียของกำลังงานในเทอมที่สองและเทอมที่สาม

เทอมที่สอง แสดงถึงการสูญเสียกำลังงานเนื่องจากวิสคอสฟริคชั่น

เทอมที่สาม แสดงถึงการสูญเสียกำลังงานเนื่องจากแรงบิดเสียดทานทั้งหมด

เทอมที่สี่ แสดงถึงเอาท์พุทเชิงกลของมอเตอร์

สองเทอมหลัง แสดงถึงการไหลเข้าหรือออกของกำลังงานในอุปกรณ์สะสมพลังงาน ในที่นี้ได้แก่ อินดัคแตนซ์ และแรงเฉื่อยของเพลา ถ้าในช่วงเริ่มต้นจนกระทั่งสิ้นสุดของคาบเวลาที่กำหนดให้ใดๆ พลังงานที่สะสมอยู่ในส่วนของอินดัคแตนซ์และแรงเฉื่อยของเพลาจะมีค่าคงที่ ดังนั้นจึงไม่มีผลการเปลี่ยนแปลงของพลังงานที่สะสมอยู่ซึ่งก็จะไม่เกิดผลของการสูญเสียกำลังงาน ดังนั้น ผลของกำลังเฉลี่ยที่สูญเสียให้แก่แรงเฉื่อยของเพลาและอินดัคแตนซ์มีค่าเป็นศูนย์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ประสิทธิภาพ (efficiency) ของมอเตอร์ที่ทำหน้าที่ในการแปลงพลังงานคำนวนได้จากอัตราส่วนของเพาเวอร์เอาท์พุทเชิงกล ต่อผลรวมของเพาเวอร์อินพุทเฉลี่ย

$$\text{ประสิทธิภาพ(กำลังงาน)} = \frac{T_L}{(R_a i_a^2 K_t/K_e) + B\omega + T_f + T_L} * 100\% \quad \ldots.(2.38)$$

เมื่อพารามิเตอร์ทั้งหมดพิจารณาได้ว่ามีค่าคงที่

ประสิทธิภาพของการแปลงพลังงานจากพลังงานไฟฟ้าเป็นพลังงานเชิงกล สามารถหาได้จากอัตราส่วนของเพาเวอร์เอาท์พุทเชิงกลต่อเพาเวอร์เชิงกลที่พัฒนาขึ้นโดยโรเตอร์

ประสิทธิภาพในการแปลงทางกำลังงานเชิงกล

$$= \frac{T_L}{B\omega + T_f + T_L} * 100\% \quad \ldots..(2.39)$$

ตัวอย่าง การคำนวนหาประสิทธิภาพของดีซีมอเตอร์ที่มีรายละเอียดต่างๆดังต่อไปนี้

ดีซีมอเตอร์แบบพิลด์เป็นแม่เหล็กถาวรขนาดเล็ก ทำงานที่อาเมเจอร์โวลต์เตจ 16 โวลต์ และดึงกระแสอาเมเจอร์ไหล 2 แอมแปร์ ขณะที่จ่ายแรงบิดโหลด 0.08นิวตัน-เมตร และโหลดหมุนด้วยความเร็ว200 เรเดียน/วินาที แรงเสียดทานอื่นๆที่นอกจากวิสคอสฟริคชั่นแล้วเป็นศูนย์ $T_f = 0$ และ $B = 100 * 10^{-6}$ นิวตัน-เมตร ความต้านทานของอาร์เมเจอร์ = 3 โอห์ม ให้คำนวนหาผลรวมของประสิทธิภาพกำลังงาน และประสิทธิภาพในการแปลงกำลังงานเชิงกลของมอเตอร์ตัวนี้

$$\text{ประสิทธิภาพ(กำลังงาน)} = \frac{0.08}{(3*2^2/200)+100*10^{-6}*200+0.08} * 100\%$$
$$= 50\%$$

ประสิทธิภาพในการแปลงกำลังงานเชิงกล

$$= \frac{0.08}{100*10^{-6}*200+0.08}$$
$$= \frac{(0.08)}{(0.1)} * 100\%$$
$$= 80\%$$

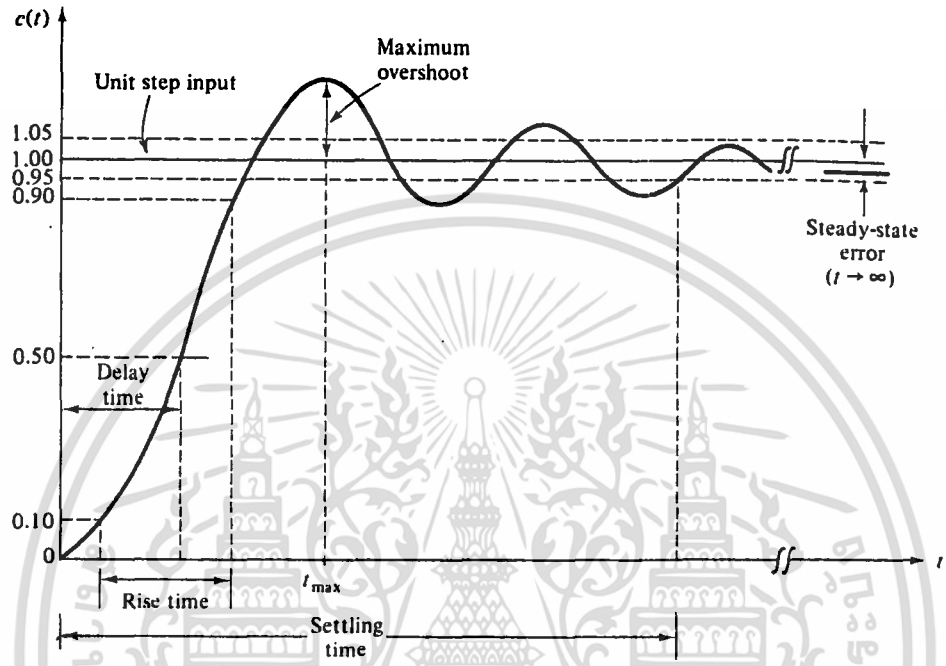
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ข้อสังเกตุ การสูญเสียกำลังงานของมอเตอร์ส่วนใหญ่เกิดจากความต้านทานของอาร์เมเจอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.7 Time-domain

เมื่อทำการวัดความเร็วของมอเตอร์ตั้งแต่เวลาเริ่มต้นหมุน แล้วนำค่าที่ได้มาพลอตกราฟระหว่าง ความเร็วกับเวลา จะได้ที่มีลักษณะดังรูปที่2.30



รูปที่2.30 แสดงการตอบสนองต่อยูนิตสเตปของระบบควบคุม

ซึ่งก็คือการตอบสนองทางเวลา(time-domain response)นั่นเอง กล่าวคือ สมมุติว่าเมื่อมีการสั่งให้มอเตอร์หมุนด้วยความเร็วใดๆค่าหนึ่ง มอเตอร์ก็จะเริ่มหมุนที่ความเร็ว 0 รอบ/นาที เพิ่มขึ้นไปเรื่อยๆจนกระทั่งถึงความเร็วที่ต้องการ โดยจะมีการตอบสนองทางเวลาเกิดขึ้น ซึ่งประกอบด้วย การตอบสนองชั่วขณะ (transient response) และ การตอบสนองที่สภาวะคงที่(stedy-state response) สามารถเขียนเป็นความสัมพันธ์ได้ดังนี้ คือ

$$c(t) = c_t(t) + c_{ss}(t)$$

เมื่อ

$c_t(t)$ = transient response

$c_{ss}(t)$ = steady-state response

จากรูปจะเห็นว่า ช่วงที่มีการเปลี่ยนแปลงอย่างมากจะเป็นช่วงการตอบสนองชั่วขณะ และเมื่อถึงความเร็วที่เราต้องการ การเปลี่ยนแปลงของกราฟระหว่างความเร็วกับเวลาก็จะมีค่าเกือบคงที่ ซึ่งก็คือช่วงของการตอบสนองที่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

คงที่ ในที่นี้จะได้ศึกษาถึงผลตอบสนองชั่วขณะ ทั้งนี้ก็เพราะว่าในช่วงนี้สามารถนำไปใช้อธิบายผลการตอบสนองทางเวลาในช่วงตั้งแต่เริ่มสตาร์ท ไปจนถึงความเร็วที่เราต้องการได้ และก่อนที่จะศึกษาในเรื่องนี้จะขอกล่าวถึงลักษณะของยูนิตสเตปฟังก์ชั่นเสียก่อน ทั้งนี้เพราะว่าเราจะได้นำเอาฟังก์ชั่นชนิดนี้ไปใช้ในการศึกษาถึงผลตอบสนองชั่วขณะนั่นเอง

2.7.1 สเตปอินพุทฟังก์ชั่น (step input function)

สเตปอินพุทฟังก์ชั่น หรือยูนิตสเตปฟังก์ชั่น คือฟังก์ชั่นที่มีการเปลี่ยนแปลงค่าตัวแปรอินพุทอ้างอิงอย่างรวดเร็วทันทีทันใด ซึ่งสามารถแทนฟังก์ชั่นนี้ด้วยสมการทางคณิตศาสตร์ได้ดังต่อไปนี้

$$r(t) = R \quad ,t>0 \qquad .....(2.40)$$
$$= 0 \quad ,t<0$$

เมื่อ R เป็นค่าคงที่

หรือ

$$r(t) = Ru_s(t) \qquad .....(2.41)$$

เมื่อ $u_s(t)$ คือยูนิตสเตปฟังก์ชั่น ซึ่งจะไม่มีความหมายที่ t = 0

รูปที่2.31 ยูนิตสเตปฟังก์ชั่น

2.7.2 ผลตอบสนองชั่วขณะ

ในระบบควบคุมของแต่ละระบบใดๆ จะมีรูปร่างกราฟของผลตอบสนองทางเวลาที่แตกต่างกันออกไป ผลตอบสนองชั่วขณะซึ่งเกิดที่สภาวะเริ่มต้นก็จะมีรูปร่างต่างๆกันตามตัวแปรของระบบควบคุม ในการศึกษาผลการตอบสนองชั่วขณะจะใช้ยูนิตสเตปฟังก์ชั่น เป็นอินพุทของระบบ ทั้งนี้เพราะว่าฟังก์ชั่นชนิดนี้มีการเปลี่ยนแปลงแอมป์ปลิจูดอย่างรวดเร็ว ซึ่งจะทำให้เราทราบถึงความเร็วในการตอบสนองของระบบ และยังสามารถใช้ได้ในหลายความถี่ พารามิเตอร์ที่สำคัญของผลตอบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ผลตอบสนองชั่วขณะ ได้แก่

1 โอเวอร์ชูทสูงสุด (maximum overshoot) เป็นค่าที่บอกให้ทราบว่าเอาท์พุทมีการเปลี่ยนแปลงสูงกว่ายูนิตสเตปที่เป็นอินพุทมากที่สุดเท่าไร ส่วนมากนิยมบอกค่าเป็นเปอร์เซนต์ ตามความสัมพันธ์

เปอร์เซนต์โอเวอร์ชูทสูงสุด = (โอเวอร์ชูทสูงสุด/ค่าสุดท้าย) * 100%

2 เวลาหน่วง (delay time;T_d) คือเวลาที่เอาท์พุทมีค่าเท่ากับ50%ของค่าสุดท้าย (steady state)

3 Rising time ;T_r คือเวลาที่เอาท์พุทเพิ่มจาก 10% ถึง 90%ของค่าสุดท้าย

4 Setting time;T_s คือเวลาที่ผลตอบสนองมีค่าอยู่ในช่วงที่ยอมรับได้ โดยปกติจะเป็น 5% ของค่าสุดท้าย

ในการศึกษาว่าระบบจะมีการตอบสนองอย่างไร สามารถกระทำได้โดยการนำเอาอินพุทไปหารเอาท์พุทที่ได้ ผลที่ได้ก็จะเป็นสมการที่เรียกว่าทรานสเฟอร์ฟังก์ชั่น(transfer function) ซึ่งค่าที่ได้จากทรานสเฟอร์ฟังก์ชั่นนี้จะเป็นตัวบอกให้เราทราบถึงลักษณะของการตอบสนองทางเวลาของระบบว่าเป็นไปในลักษณะใด เช่นอาจเป็นแบบโอเวอร์แดมป์(over damp) หรือคริทิคอลแดมป์ (critical damp) เป็นต้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่3
การวิจัยและดำเนินการ

การทดลองควบคุมการเคลื่อนที่ของเตียงคนไข้ด้วยคอมพิวเตอร์

ในการเคลื่อนที่ของเตียงคนไข้นั้นเราใช้คอมพิวเตอร์เป็นตัวควบคุม เนื่องจากคอมพิวเตอร์ที่ใช้เป็นคอมพิวเตอร์อุตสาหกรรม ดังนั้นเราจึงต้องศึกษาวิธีการพัฒนาโปรแกรม และการใช้งานคอมพิวเตอร์นี้เพื่อสามารถนำไปใช้ในการควบคุมการเคลื่อนที่ของเตียงได้

3.1 มัลติโปรเซสเซอร์

เครื่องคอมพิวเตอร์ที่ใช้ในการควบคุมนี้เป็นแบบมัลติโปรเซสเซอร์ และใช้ระบบบัสแบบ STD (Simple To Designed)ของบริษัท PRO-LOG ระบบแบบมัลติโปรเซสเซอร์นี้มีข้อได้เปรียบโปรเซสเซอร์ตัวเดียวตรงที่ถ้าหากโปรเซสเซอร์ตัวใดตัวหนึ่งเกิดเสียขึ้นมา โปรเซสเซอร์ตัวอื่นก็จะเข้ามาทำงานแทนที่ทันที ทั้งนี้เนื่องจากโปรเซสเซอร์ทุกตัวใช้หน่วยความจำและอุปกรณ์ I/O ส่วนกลางร่วมกัน (global memory and i/o) นอกจากนี้ระบบยังทำงานได้เร็วขึ้น เพราะไม่มีการแบ่งเวลาเหมือนโปรเซสเซอร์ตัวเดียวเมื่อมีการทำงานหลายอย่างในเวลาเดียวกัน ระบบมัลติโปรเซสเซอร์มีสองแบบคือ

1 แบบมาสเตอร์/สเลฟ (Master/slave)

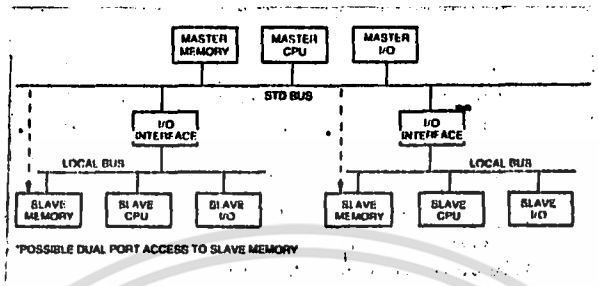
2 แบบมัลติมาสเตอร์(Multimaster)

3.1.1 มาสเตอร์/สเลฟ

ในระบบนี้มาสเตอร์ซีพียูเท่านั้นที่สามารถควบคุมการใช้บัสได้ รวมทั้งการเข้าใช้หน่วยความจำของสเลฟหรือตัวรองได้ ในขณะที่ซีพียูรองใช้ I/O และหน่วยความจำได้เฉพาะในส่วนของตัวมันเอง สเลฟในระบบแต่ละตัวมีหน้าที่เฉพาะตามการวางระบบของเราในตอนแรก การส่งผ่านข้อมูลและพารามิเตอร์ระหว่างมาสเตอร์และสเลฟกระทำผ่านI/O อินเตอร์เฟสหรือผ่านทางหน่วยความจำร่วมเท่านั้น การส่งผ่านข้อมูลระหว่างบัสภายในของสเลฟ(บัสส่วนตัว)และบัสภายนอก(SIDบัส) อาจกระทำผ่านหน่วยความจำแบบสองทาง โดยใช้ DMA (Direct Memory -

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

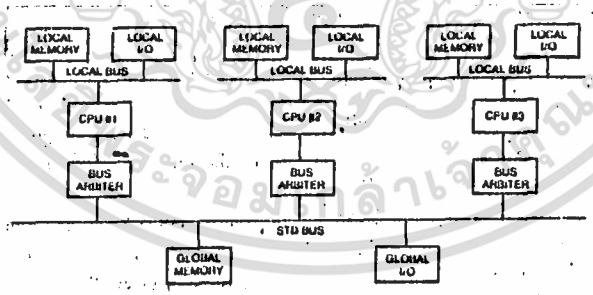
Access) หรือใช้ตัวเลือกบัส (bus aybiter) ของสเลฟเป็นตัวจัดการ



รูปที่3.1 ระบบมาสเตอร์สเลฟมัลติโปรเซสเซอร์

3.1.2 มัลติมาสเตอร์

ในระบบนี้โปรเซสเซอร์แต่ละตัวมีสิทธิ์เท่าเทียมกันที่จะใช้หน่วยความจำกลาง(global memory) หรืออุปกรณ์ i/o ส่วนกลาง(global i/o)ได้ ในขณะที่โปรเซสเซอร์แต่ละตัวก็ยังคงมีหน่วยความจำและ i/o เป็นของตัวเอง ซึ่งส่วนนี้โปรเซสเซอร์ตัวอื่นไม่มีสิทธิ์เข้าไปใช้ได้เลย



รูปที่3.2 การจัดหน่วยความจำของมัลติมาสเตอร์

จุดสำคัญของส่วนนี้อยู่ที่ส่วนที่เรียกว่าบัส ซึ่งจะเป็นตัวจัดการในเรื่องของการจองบัสในกระบวนการมัลติมาสเตอร์ ข้อดีของระบบแบบนี้ที่เห็นได้ชัด คือ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ทำให้สามารถแก้ไขเพิ่มเติมระบบที่เราออกแบบมาได้ง่าย อีกทั้งยังเพิ่มเสถียรภาพ และประสิทธิภาพของระบบให้ดีขึ้นอีกด้วย

นอกจากนี้ระบบมัลติมาสเตอร์ยังสามารถอ้างหน่วยความจำของระบบได้มากกว่าแบบมาสเตอร์สเลฟ กล่าวคือถ้าในระบบมัลติมาสเตอร์ระบบหนึ่ง(ที่ใช้8088) มีมาสเตอร์โปรเซสเซอร์ทั้งหมด 6ชุด ในแต่ละชุดมีหน่วยความจำภายในของแต่ละระบบ 128กิโลไบต์ ในหน่วยความจำกลาง128กิโลไบต์ ซึ่งโปรเซสเซอร์อื่นๆก็ร่วมอยู่ด้วย ลักษณะนี้จะสามารถคำนวนหน่วยความจำของระบบได้ เท่ากับ (128*6) + 872 = 1.6 เมกกะไบต์ และในทุกๆโปรเซสเซอร์ก็จะเห็นเป็นตำแหน่งเดียวกันหมด ซึ่งจะทำให้ทำการเขียนโปรแกรมและเพิ่มการ์ดโปรเซสเซอร์เป็นไปอย่างง่ายและสะดวกยิ่งขึ้น

3.2 การ์ดที่ใช้ในระบบการควบคุมการเคลื่อนที่ของเตียงคนไข้

1 การ์ด 7892 ทำหน้าที่เป็นโปรเซสเซอร์เพื่อควบคุมระบบทั้งหมด

2 การ์ด 7171 ทำหน้าที่เป็นการ์ดสนับสนุน(support card)

3 การ์ด 7340 ทำหน้าที่เป็นตัวควบคุมมอเตอร์

3.2.1 การ์ด 7892

เป็นการ์ดที่สำคัญที่สุดเพราะเป็นตัวควบคุมระบบทั้งหมด คือทำหน้าที่เป็น X-T Compatible Processor 25MHz 80c286 ข้อดีของโปรเซสเซอร์7892 คือ ถูกออกแบบเพื่อที่จะนำมาใช้กับระบบมัลติมาสเตอร์ที่มีโปรเซสเซอร์หลายตัวทำงานพร้อมกันได้และนอกจากนี้ยังสามารถทำงานโดดๆได้อีกด้วย โปรเซสเซอร์7892นี้ จะทำงานภายใต้การควบคุมของดอส(DOS) ซึ่งในบางครั้งจะเรียกว่าดอส โปรเซสเซอร์ ส่วนโปรเซสเซอร์ตัวอื่นๆนั้นจะเรียกว่าเรียว-ไทม์ โปรเซสเซอร์(real-time processor) หน่วยความจำภายใน(Local memory)ของการ์ดนี้ มีขนาด 640 กิโลไบต์ เมื่อมีการบูทของ ดอส โปรเซสเซอร์เกิดขึ้น ก็จะไม่มีผลต่อการทำงานของโปรเซสเซอร์ตัวอื่น ทั้งนี้เพราะว่า โปรเซสเซอร์แต่ละตัวทำงานเป็นอิสระต่อกัน ดอส โปรเซสเซอร์สามารถที่จะเรียกและแก้ไขโปรแกรมที่กำลังทำงานบนเรียว-ไทม์ โปรเซสเซอร์ โดยผ่านทางหน่วยความจำส่วนกลาง และสามารถแสดงผลออกทางหน้าจอหรือเก็บข้อมูลลงในแผ่นดิสเก็ตได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2.2 การ์ด 7171

ทำหน้าที่เป็นการ์ดสนับสนุนซึ่งจะทำงานร่วมกับการ์ด 7892 โดยใช้เป็นที่สำหรับเก็บดอสเพื่อที่จะให้การ์ด7892 อ่านข้อมูลเพื่อใช้ในการทำงาน ในการ์ด 7171 มีซีมอสแรมซึ่งมีขนาดของหน่วยความจำ 32 กิโลไบต์ ทำหน้าที่เป็นหน่วยความจำกลาง นอกจากนี้ยังทำหน้าที่ในการเลือกบัสเมื่อโปรเซสเซอร์แต่ละตัวขอใช้หน่วยความจำของส่วนกลาง อีกทั้งยังมีระบบป้องกันเมื่อซอฟแวร์หรือฮาร์ดแวร์เกิดขัดข้องขึ้น โดย watchdog timer จะทำการรีเซ็ทระบบใหม่และยังมีระบบรักษาข้อมูลเพื่อไม่ให้สูญหายเมื่อเกิดการขัดข้อง โดยจะมีแบตเตอรี่สำรองเป็นไฟเลี้ยงแทน

3.2.3 การ์ด 7340

ทำหน้าที่ควบคุมการเคลื่อนที่ของมอเตอร์ ทั้งตำแหน่ง และ ความเร็ว โดยมีไอซีที่สำคัญที่ใช้เป็นโปรเซสเซอร์ในการควบคุมคือ LM 628 และใช้การควบคุมระบบแบบ PID (proportional + integral + derivative) โดยใช้เอนโคดเดอร์เป็นตัวเซนเซอร์ในการตรวจจับตำแหน่งและความเร็ว การ์ดนี้มีความละเอียดสูง(high resolution) โดยค่าของความเร็วและตำแหน่งมีขนาด 32 บิท เอาท์พุทของการ์ดจะเป็นไฟกระแสตรง ± 10 โวลต์ ด้วยค่าความละเอียดขนาด 12 บิท การโปรแกรมความเร็วจากจุดเริ่มต้นถึงจุดสิ้นสุดสามารถกำหนดได้โดยผู้ใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.3 ส่วนประกอบของเตียงคนไข้

อุปกรณ์ที่ใช้ในการสร้างเตียงคนไข้ที่สำคัญ ประกอบด้วย

1 มอเตอร์ (Motor)

2 เอนโคดเดอร์ (Encoder)

3 ลิเนียร์ไกด์เวย์ (Linear Guide Way), แบริ่ง(Bearing)

4 บอลสกรู (Ball Screw), แบริ่ง(Bearing)

5 สกรูส่งกำลัง(power screws)

3.3.1 มอเตอร์

มอเตอร์ที่ใช้เป็นดีซีมอเตอร์จำนวน 3 ตัว คือควบคุมการเคลื่อนที่ในแนวแกนตั้ง,แนวแกนนอน และมุมเอียง มอเตอร์ที่ใช้ควบคุมในแนวแกนตั้ง(ความสูง) จะอยู่ด้านล่างสุดในจำนวนมอเตอร์ทั้งสามตัว ซึ่งมอเตอร์ตัวนี้สามารถควบคุมการเคลื่อนที่ขึ้นลงได้สูงสุดประมาณ 120 เซนติเมตร มอเตอร์ตัวที่ควบคุมการเคลื่อนที่ในแนวแกนนอนจะอยู่ในรูปการเคลื่อนที่เข้า-ออก ในแกนทรี ซึ่งความยาวในการเคลื่อนที่ทำได้ประมาณ 80 เซนติเมตร ส่วนมอเตอร์ตัวสุดท้ายที่ใช้ในการควบคุมความเอียงของเตียงคนไข้สามารถปรับให้เอียงได้ถึง 20 องศา โดยที่ตำแหน่งของมอเตอร์ตัวนี้อยู่ระหว่างมอเตอร์ทั้งสองที่กล่าวมาแล้ว รายละเอียดเกี่ยวกับมอเตอร์แสดงในตารางดังต่อไปนี้

ตารางที่3.1: DC permanent magnetic parallel shaft gear motor for tyristor drives-style PSL

Rating DC		Deratings form	Ratings with form factor		Base speed	Gear ratio	Max.safe torque	Armature		Motor wiegh	Motor and gear	type Motor part
KW	HP	factor	KW	HP	R.P.M	to 1	NM	Volts	Full load amps	Kgs	style	number
.22	.30	1.2	.18	1/4	125	20	10.7	180	1	8.2	74-PSL-0	GPP7473
.22	.30	1.2	.18	1/4	42	60	30.50	180	1.1	8.2	74-PSL-0	GPP7476

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

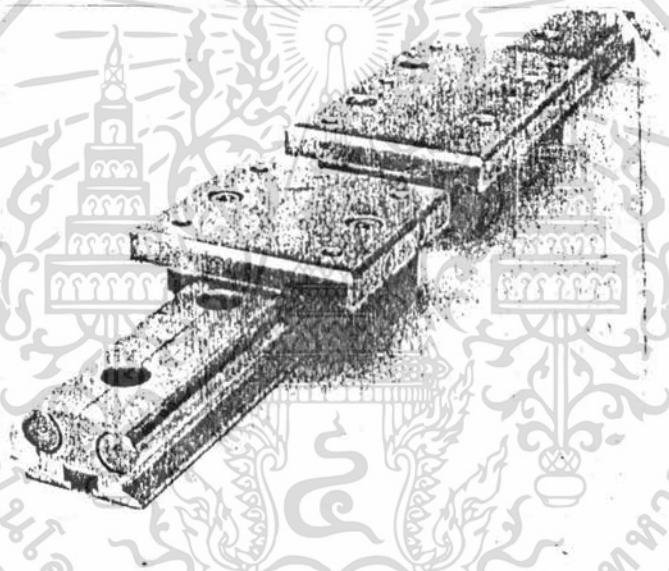
3.3.2 เอนโคดเดอร์

เอนโคดเดอร์ที่ใช้เป็นแบบอินคริเมนทอล(incrimental) หรือเรียกอีกอย่างหนึ่งว่าควอดราเจอร์(quadrature)เพื่อใช้ในการรักษาตำแหน่งและสร้างสัญญาณป้อนกลับ เอาท์พุทที่เกิดขึ้นจากการหมุนของเอนโคดเดอร์จำนวนสามช่อง คือ ช่องสัญญาณA,B และM หรือเรียกรวมกันว่า ABM สัญญาณช่อง Aและ B จะมีการนำ(lead) และตาม (lag) ของเฟส ขึ้นอยู่กับว่าจะหมุนตามหรือทวนเข็มนาฬิกา โดยมุมที่นำและตามจะมีค่า 90 องศา ช่องสัญญาณ M(index signal) จะเกิดจำนวนพัลส์เมื่อหมุนครบ 1 รอบ ในการควบคุมตำแหน่งของเอนโคดเดอร์ทำได้โดยการนับจำนวนพัลส์ที่ออกมา ส่วนการควบคุมความเร็วทำได้โดยการนับความถี่ของสัญญาณพัลส์ที่เกิดขึ้น ซึ่งสัญญาณพัลส์จะแปรผันตรงกับการหมุนของเพลาของเอนโคดเดอร์ นั่นคือเมื่อเอนโคดเดอร์หมุนด้วยความเร็วที่มากขึ้นความถี่ก็จะเพิ่มขึ้นด้วย ในการใช้เอนโคดเดอร์ควบคุมเตียงคนไข้นั้นใช้ทั้งหมดสามตัว คือ ควบคุมแนวแกน นอน, แนวแกนตั้งและมุมเอียง โดยการควบคุมในแต่ละแนวจะต้องเลือกใช้เอนโคดเดอร์ที่เหมาะสม (resolution ค่าต่างๆ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.3.3 ลิเนียร์ไกด์เวย์ และแบริ่ง

เป็นอุปกรณ์ที่ทำหน้าที่ในการลดแรงเสียดทานในการเคลื่อนตัวของเตียงคนไข้โดยส่วนที่เคลื่อนที่จะเป็นแบริ่งซึ่งภายในจะบรรจุลูกปืนวิ่งบนรางเลื่อน ลิเนียร์ไกด์เวย์และแบริ่งที่ใช้มีทั้งหมด 8ตัว แบ่งออกเป็นสามชั้น โดยชั้นล่างใช้สองตัว ความยาวตัวละประมาณ60 เซนติเมตร การเคลื่อนที่ของลิเนียร์แบริ่งในแนวแกนนอนจะทำให้เตียงเคลื่อนที่ขึ้นลง ชั้นกลางก็เช่นเดียวกันคือใช้จำนวนสองตัว และช่วยให้เตียงขึ้นลงเหมือนชั้นล่าง ส่วนชั้นบนใช้จำนวนสี่ตัวความยาวตัวละประมาณ 1เมตร การเคลื่อนที่ของลิเนียร์แบริ่งในส่วนนี้จะทำให้เตียงคนไข้เคลื่อนที่เข้าออก

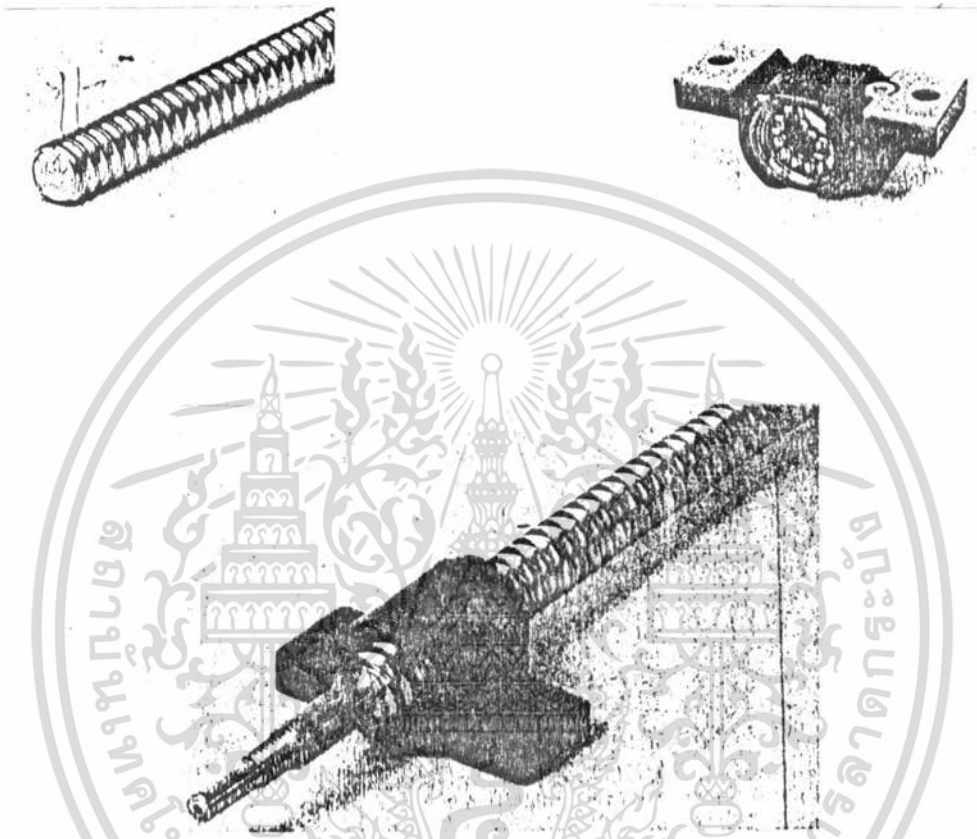


รูปที่3.3 ลิเนียร์ไกด์เวย์และแบริ่ง

3.3.4 บอลสกรู และแบริ่ง

เป็นอุปกรณ์สำคัญที่ช่วยในการเคลื่อนที่ของเตียงคนไข้ การทำงานจะเหมือนกับเกลียวรูหนอน แต่มีประสิทธิภาพในการขับโหลดสูงกว่า เนื่องจากบอลสกรูและแบริ่งจะช่วยลดแรงเสียดทานที่เกิดจากการหมุนของเกลียว โดยจะมีลูกปืนช่วยในการเคลื่อนที่ให้คล่องขึ้น บอลสกรูและแบริ่งที่ใช้มีทั้งหมด 3 ตัว โดยจะต่อเข้ากับมอเตอร์ ซึ่งการหมุนของมอเตอร์จะทำให้บอลสกรูหมุนตาม และบอลสกรูก็จะทำให้ลิเนียร์แบริ่งเคลื่อนที่อีกที บอลสกรูที่ใช้ควบคุมการเคลื่อนที่ในแนวแกนตั้งและแกนนอนจะใช้ระยะพิท 20 มิลลิเมตร ส่วนการควบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



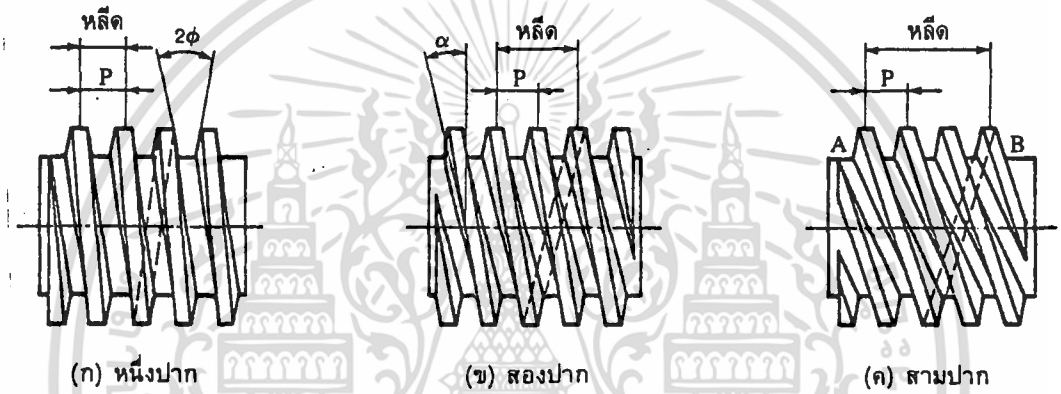
รูปที่3.4 แสดงรูปของบอลสกรูและแบริ่ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.3.5 สกรูส่งกำลัง(Power screws)

เป็นชิ้นส่วนที่ใช้ในเครื่องจักรกลเพื่อเปลี่ยนการหมุนให้เป็นการเลื่อน รวมทั้งใช้ในการยกน้ำหนักที่ตัวสกรูรับอยู่ด้วย ดังนั้นการออกแบบหรือเลือกขนาดของสกรูส่งกำลังจะต้องคิดถึงความแข็งแรงของตัวสกรูที่จะรับแรงกดหรือแรงดึง ความสามารถในการรับแรงเฉือนของตัวสกรู ตลอดจนกำลังงานที่ต้องการ

ดังนั้นก่อนที่จะกล่าวถึงการคำนวนเกี่ยวกับเกลียว จึงจำเป็นที่จะต้องทำความเข้าใจกับคำจำกัดความบางคำที่จะต้องใช้อยู่เสมอ โดยพิจารณาจากรูปที่3.5 ดังต่อไปนี้



รูปที่3.5 เกลียวสี่เหลี่ยมคางหมู

ระยะพิท ;p : หมายถึงระยะทางที่วัดตามแนวแกนของสกรูจากจุดหนึ่งบนเกลียวหนึ่ง ไปยังจุดเดียวกันของเกลียวที่อยู่ถัดไป

หลีด (lead;l_0) : คือระยะทางที่สกรูเคลื่อนที่ได้ตามแนวของสกรู ในขณะที่สกรูหมุนไปหนึ่งรอบ ถ้าเป็นสกรูหนึ่งปาก(single thread) ระยะของหลีดจะมีค่าเท่ากับระยะพิท สำหรับสกรูสองปาก(double thread) เกลียวจะมีปากคาบระหว่างเกลียวสองเกลียวดังรูปที่ 3.5(ข) ดังนั้นเมื่อสกรูหมุนไปหนึ่งรอบการเคลื่อนที่ในแนวแกนของสกรูจึงเป็นสองเท่าของระยะพิท ในทำนองเดียวกันสำหรับสกรูสามปาก(triple thread) หลีดจะมีค่าเป็นสามเท่าของระยะพิท ดังนั้นถ้าสกรูเป็นแบบ n ปาก ระยะของหลีดคือ

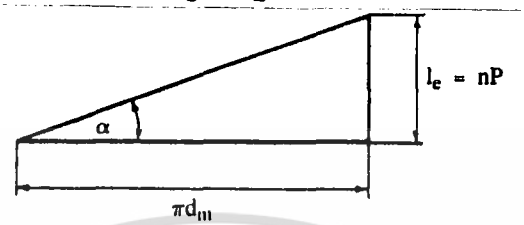
$$l_0 = np \qquad .....(3.1)$$

มุมฮีลิกซ์หรือมุมหลีด(helix or lead angle;α) : คือมุมระหว่างระนาบที่สัมผัสกับความเอียงของเกลียวและระนาบที่ตั้งฉากกับแกนของสกรู ถ้าให้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

d_m เป็นเส้นผ่าศูนย์กลางเฉลี่ยของสกรู และนำส่วนที่สกรูเคลื่อนที่ไปในขณะที่หมุนหนึ่งรอบมาคลี่ออกก็จะได้ลักษณะดังรูปที่3.6 และมุมหลีดคือ

$$\tan\alpha = l_e/\pi d_m \qquad \text{.....(3.2)}$$



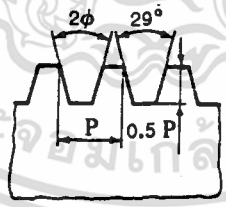
รูปที่3.6 มุมหลีด

เส้นผ่าศูนย์กลางใหญ่ : เป็นเส้นผ่าศูนย์กลางที่ใหญ่ที่สุดของสกรูซึ่งนับรวมถึงความสูงของเกลียวด้วย ขนาดระบุ(normal size) ของสกรูส่งกำลังจะบอกโดยใช้ขนาดของเส้นผ่าศูนย์กลางใหญ่เสมอ

เส้นผ่าศูนย์กลางน้อย : เป็นเส้นผ่าศูนย์กลางที่เล็กที่สุดของสกรู

3.3.5.1 เกลียวแอกมี(acme thread) หรือเกลียวสี่เหลี่ยมคางหมู

เป็นเกลียวที่ใช้ในโครงงานพิเศษในครั้งนี้ รูปร่างของเกลียวเป็นสี่เหลี่ยมคางหมู โดยมีมุมเกลียว(thread angle) 2Φ เท่ากับ 29 องศา ดังในรูปที่3.7 มาตราฐานของเกลียวชนิดนี้ แสดงอยู่ในตารางที่ ข.1 และ ข.2 ในภาคผนวก



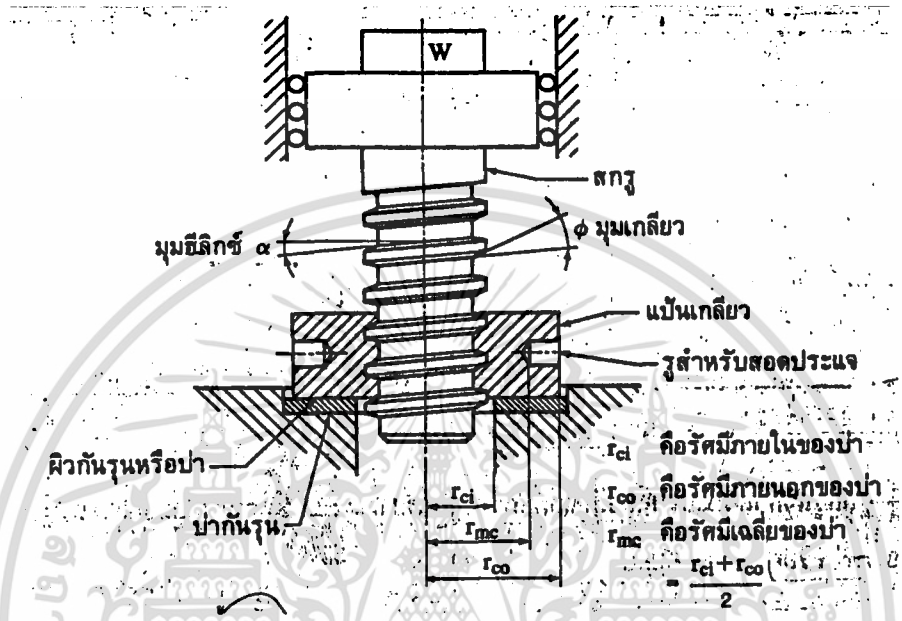
รูปที่3.7 เกลียวแอกมี(เกลียวสี่เหลี่ยมคางหมู)

3.3.5.2 โมเมนท์บิดสำหรับหมุนสกรูส่งกำลัง

พิจารณาสกรูส่งกำลังดังรูปที่3.8 ซึ่งมีมุมฮีลิกซ์ α องศาและมุมเกลียวของสกรู Φ องศา ในการยกน้ำหนัก W ขึ้นลง ทำได้โดยการหมุนแป้นเกลียวซึ่งตั้งอยู่บนที่รองรับ เรียกว่าบ่า(collar) สมมุติให้แรงF ที่ใช้ยกน้ำหนักขึ้น กระทำอยู่บนเกลียวที่มีรัศมีของสกรู $r_m = \frac{(r_o + r_i)}{2}$ ณ ตำแหน่ง 0 ดังรูปที่3.9ในขณะ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ที่ยกน้ำหนักขึ้นจะมีแรงเสียดทาน F_f กระทำบนผิวหน้าของเกลียวด้านการเคลื่อนที่ของสกรู



รูปที่3.8 ตัวอย่างการใช้สกรูส่งกำลังเป็นแม่แรง

โดยที่ F_n คือแรงปฏิกิริยาซึ่งมีทิศตั้งฉากกับผิวหน้าของเกลียว

OA คือเวกเตอร์ของแรงที่มีขนาดเท่ากับน้ำหนัก W แต่ทิศทางตรงข้าม

OB คือเวกเตอร์ของแรงรวมระหว่างแรง F_f และ OA

F_s คือสัมประสิทธิ์ความเสียดทานระหว่างผิวหน้าของเกลียวและแป้นเกลียว

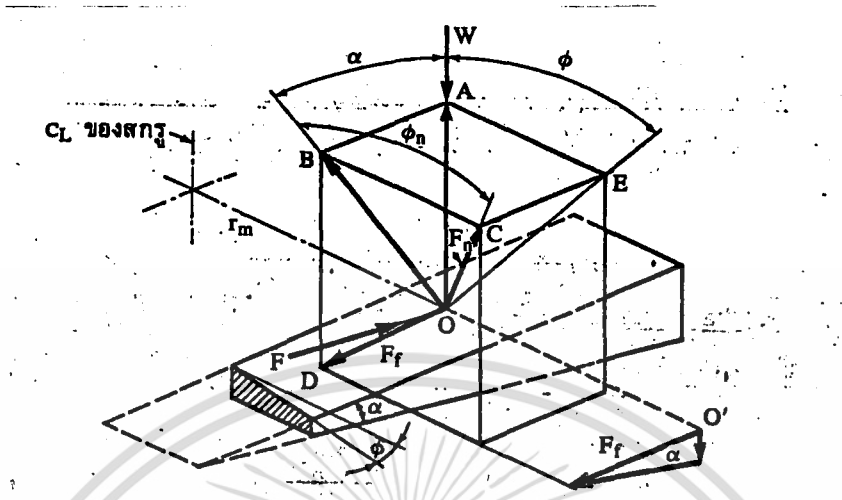
รวมแรงในแนวดิ่งจะได้

$$F_n \cos\Phi_n \cos\alpha = W + F_f \sin\alpha$$

แต่ $F_f = f_s F_n$ นำไปแทนค่าลงในสมการข้างบน จะได้ว่า

$$F_n = \frac{W}{\cos\Phi_n \cos\alpha - f_s \sin\alpha} \qquad \text{.....(3.3)}$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่3.9 แผนภาพแรงปฏิกิริยาบนเกลียวเนื่องจากการยกน้ำหนักW

ในการหาโมเมนต์ที่เกิดจากการใช้แรง F เพื่อยกน้ำหนักขึ้น ให้หาโมเมนต์รอบแกนกลางของสกรู

$$T_R = Fr_m = r_m\left[F_f\cos\alpha + F_n\cos\Phi_n\sin\alpha\right]$$

$$T_R = r_m[f_s F_n\cos\alpha + F_n\cos\Phi_n\sin\alpha]$$

แทนค่าF_n จากสมการที่3.3 ลงในสมการข้างบน และให้d_m แทนเส้นผ่าศูนย์กลางเฉลี่ยของสกรู จะได้โมเมนต์บิดที่ใช้ในการยกน้ำหนัก(สำหรับสกรูเกลียวสี่เหลี่ยมคางหมูในตารางที่ ข.2ในภาคผนวก d_m ก็คือเส้นผ่าศูนย์กลางพิท d_2)

$$T_R = \frac{Wd_m}{2}\left[\frac{f_s\cos\alpha + \cos\Phi_n\sin\alpha}{\cos\Phi_n\cos\alpha - f_s\sin\alpha}\right]$$

หรือ

$$T_R = \frac{Wd_m}{2}\left[\frac{f_s + \cos\Phi_n\tan\alpha}{\cos\Phi_n - f_s\tan\alpha}\right] \qquad \ldots\ldots\ldots\ldots(3.4)$$

ในการใช้สมการที่3.4 จำเป็นที่จะต้องทราบมุม Φ_n ซึ่งสามารถหาได้โดยพิจารณารูปที่3.9 ดังนี้

$$\tan\Phi_n = \frac{BC}{OB}$$

แต่ $BC = AE = OA\tan\Phi = OB\cos\alpha\tan\Phi$

ฉะนั้น $\tan\Phi_n = \cos\alpha\tan\Phi$ .....(3.5)

โดยปกติแล้วมุมฮีลิกซ์ α จะมีค่าน้อย(ประมาณ 2 ถึง 6องศา)ดังนั้นค่าของ $\cos\alpha$ จึงเกือบเท่ากับ 1 ฉะนั้นเพื่อความสะดวกในการคำนวนทั่วๆไป จึงอนุญาต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

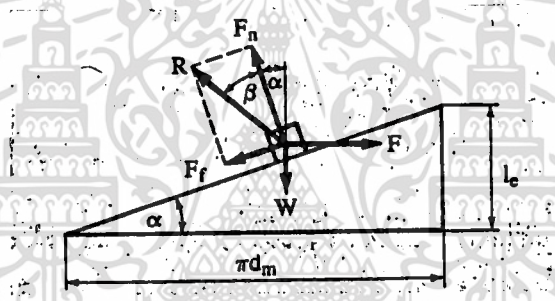
ให้ใช้ $\Phi_n = \Phi$ ได้ ซึ่งทำให้เขียนสมการที่(3.4) ได้ใหม่เป็น

$$T_R = \frac{Wd_m}{2}\left[\frac{f_s + \cos\Phi \tan\alpha}{\cos\Phi - f_s\tan\alpha}\right] \qquad .....(3.6)$$

ในกรณีของสกรูส่งกำลังที่มีเกลียวสี่เหลี่ยม มุมΦ = 0 จากสมการที่(3.6) จะได้โมเมนท์บิดที่ใช้ยกน้ำหนัก W เท่ากับ

$$T_R = \frac{Wd_m}{2}\left[\frac{f_s + \tan\alpha}{1 - f_s\tan\alpha}\right] \qquad .....(3.7)$$

ถ้าพิจารณาจากรูปที่3.9 จะเห็นได้ว่าในกรณีของเกลียวสี่เหลี่ยม แรง F_n จะเท่ากับ OB ซึ่งถ้าให้นำเกลียวจำนวนหนึ่งรอบมาคลี่ออกแล้ว จะได้ระบบของแรงที่เห็นได้ชัดเจนกว่าดังรูปที่3.10 โดยที่มุม β เรียกว่ามุมความเสียดทาน ซึ่ง $\tan\beta = f_s$



รูปที่3.10 แรงปฏิกิริยาบนเกลียวสี่เหลี่ยม

เมื่อรวมแรงในแนวดิ่งและแนวระดับ จะได้

$$W = R\cos(\alpha + \beta)$$

$$F = R\sin(\alpha + \beta)$$

หรือ $$F = W\tan(\alpha + \beta)$$

ดังนั้น $$T_R = \frac{Fd_m}{2} = \frac{Wd_m}{2}\tan(\alpha + \beta)$$

$$= \frac{Wd_m}{2}\left[\frac{\tan\beta + \tan\alpha}{1 - \tan\beta\tan\alpha}\right.$$

หรือ $$T_R = \frac{Wd_m}{2}\left[\frac{f_s + \tan\alpha}{1 - f_s\tan\alpha}\right.$$

$$= \frac{Wd_m}{2}\tan(\beta + \alpha)$$

ซึ่งเหมือนกับสมการที่(3.7) ที่ได้หามาแล้ว

ในกรณีการหมุนสกรูเพื่อยกน้ำหนักลง แรงF และ f_s ดังรูปที่3.9 จะกลับทิศทาง การหาสูตรก็ทำได้ในทำนองเดียวกันกับการยกน้ำหนัก ซึ่งจะได้สมการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ดังนี้คือ

สำหรับสกรูที่มุมเกลียว Φองศา :

$$T_L = \frac{Wd_m}{2}\left[\frac{f_s - \cos\Phi\tan\alpha}{\cos\Phi + f_s\tan\alpha}\right] \qquad .....(3.8)$$

สำหรับเกลียวสี่เหลี่ยม :

$$T_L = \frac{Wd_m}{2}\tan(\beta - \alpha) = \frac{Wd_m}{2}\left[\frac{f_s - \tan\alpha}{1 + f_s\tan\alpha}\right] \qquad .....(3.9)$$

นอกจากแรงที่กระทำต่อสกรูส่งกำลังดังที่ได้กล่าวมาแล้ว สกรูในรูปที่3.8 ยังมีแรงเสียดทานที่บ่ารองรับอีกด้วย แรงเสียดทานนี้ทำให้ต้องใช้แรงหรือโมเมนท์บิดในการยกน้ำหนักขึ้นหรือลงมากขึ้น ถ้าให้ f_c แทนสัมประสิทธิ์ความเสียดทานระหว่างผิวหน้าของบ่ารองรับและแป้นเกลียว แรงเสียดทานจะมีค่าเท่ากับ

$$F_c = f_c W$$

ถ้าสมมุติให้แรงเสียดทานนี้กระทำที่รัศมีเฉลี่ยของบ่ารองรับ r_{mc} โมเมนท์บิดที่ต้องใช้เพื่อเอาชนะความเสียดทานนี้คือ

$$T_{fc} = r_{mc} f_c W$$

ดังนั้นสมการที่(3.6) ถึงสมการที่(3.9) โมเมนท์บิดที่ต้องใช้ในการยกน้ำหนักขึ้นหรือลงสำหรับสกรูส่งกำลังที่มีมุมเกลียว Φ และ 0 องศา คือ

มุมเกลียว Φ องศา:

$$T_R = \frac{Wd_m}{2}\left[\frac{f_s+\cos\Phi\ \tan\alpha}{\cos\Phi\ -f_s\tan\alpha}\right] + r_{mc} f_c W \qquad .....(3.10)$$

มุมเกลียง 0 องศา(เกลียวสี่เหลี่ยม):

$$T_R = \frac{Wd_m}{2}\left[\frac{f_s + \tan\alpha}{1 - f_s\tan\alpha}\right] + r_{mc} f_c W \qquad .....(3.11)$$

มุมเกลียว Φ องศา:

$$T_L = \frac{Wd_m}{2}\left[\frac{f_s-\tan\alpha}{1+f_s\tan\alpha}\right] + r_{mc} f_c W \qquad .....(3.12)$$

มุมเกลียว 0 องศา(เกลียวสี่เหลี่ยม):

$$T_L = \frac{Wd_m}{2}\left[\frac{f_s-\tan\alpha}{1+f_s\tan\alpha}\right] + r_{mc} f_c W \qquad .....(3.13)$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

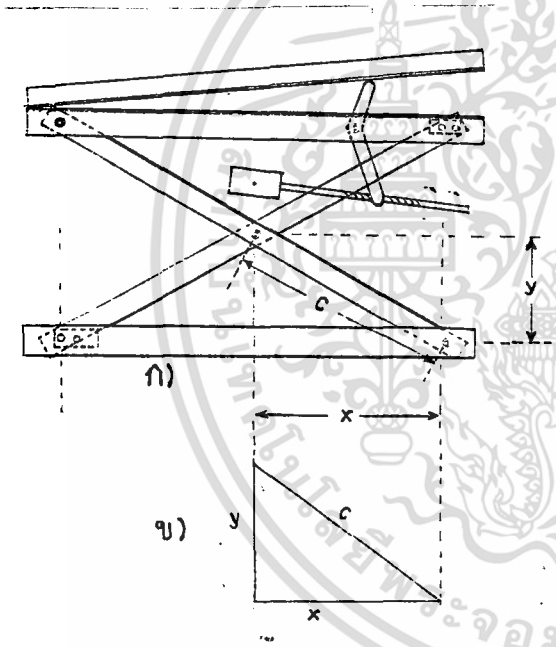
ในกรณีที่ใช้แบริ่งลูกปืนกันรุน(ball thrast bearing)รองรับแทนบ่ารองรับ ค่าสัมประสิทธิ์ความเสียดทานจะมีค่าน้อยมาก ซึ่งทำให้ไม่ต้องคิดถึงค่า $r_{ม}f_{ฺ}W$ ในสมการที่3.10 ถึงสมการที่3.13 ก็ได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.4 การคำนวนทางคณิตศาตร์ที่นำไปใช้ในการควบคุม การเคลื่อนที่ของเตียงคนไข้

3.4.1 สมการการเคลื่อนที่ขึ้นลงของเตียงคนไข้

เนื่องจากตำแหน่งของมอเตอร์ที่ใช้ควบคุมการเคลื่อนที่ในแนวแกนตั้งวางอยู่ในแนวแกนนอน และ แบบของเตียงที่ออกแบบไว้ดังแสดงไว้ในรูป ทำให้การเคลื่อนที่ในแนวนอนและการเคลื่อนที่ขึ้นลงของเตียงไม่สัมพันธ์กันแบบเส้นตรง แต่เป็นแบบสมการวงกลม ทำให้การควบคุมการหมุนของมอเตอร์เพื่อที่จะได้ความสูงที่ต้องการเป็นไปด้วยความลำบาก จากรูปข่างล่าง ค่าความยาว c จะคงที่ ส่วนแกน X และY จะเป็นฟังก์ชั่นซึ่งกันและกัน



รูปที่ 3.11 ก) แสดงภาพวาดการเคลื่อนที่ขึ้น-ลง ของเตียงคนไข้ชั้นล่างอย่างคร่าว
ข)รูปสามเหลี่ยมที่สอดคล้องกับการเคลื่อน ที่ขึ้น-ลงของเตียงคนไข้

จากทฤษฎีของพิธากอรัส

$$c^2 = x^2 + y^2$$

เมื่อc คงที่ ซึ่งก็คือรัศมีของวงกลมนั่นเอง ความสัมพันธ์ของความเร็วในแนวแกน x และy สามารถแสดงได้โดยการดิฟเฟอร์เรนชิเอทสมการวงกลม

จากความสัมพันธ์

$$x^2 = c^2 - y^2$$

ทำการดิฟเฟอร์เรนชิเอททั้งสองข้างจะได้

$$2x\frac{dx}{dt} = -2y\frac{dy}{dt}$$

$$\frac{dy}{dt} = \frac{-xdx}{ydt}$$

กราฟแสดงความสัมพันธ์ระหว่างxและ yแสดงอยู่ในรูปที่3.11 จากกราฟความสัมพันธ์ของความเร็วกับระยะทาง โดยการให้ความเร็วในแกนใดแกนหนึ่งคงที่ ตามความเป็นจริงแล้วควรจะให้อัตราในการขับเคลื่อนเตียงขึ้น-ลงคงที่เท่าที่จะเป็นไปได้เพื่อที่จะไม่เป็นอันตรายต่อคนไข้ จากกราฟเราให้ความ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เร็วในแนวนอนคงที่(ความเร็วมอเตอร์คงที่) จะเห็นว่าความเร็วในแนวแกนyจะไม่คงที่ ซึ่งความเร็วจะมีการเปลี่ยนแปลงอย่างรวดเร็วที่ระยะx มากๆเข้าใกล้ค่าc แต่ก็เป็นเพียงระยะสั้นๆ ซึ่งเราจะต้องทำการทดลองให้เห็นจริงว่า จะมีผลกระทบต่อการเคลื่อนที่ของเตียงมากน้อยหรือไม่ แนวทางการแก้ไขอาจทำได้โดยการกำหนดให้ความเร็วในแกน y คงที่ แล้วทำการคำนวนความเร็วในแนวแกนx แทน(กลับกันกับที่กล่าวมาแล้ว) ซึ่งเราจะต้องทำให้แกนนอนเคลื่อนที่ด้วยความเร็ว ตามผลการคำนวน

$c := 76.2$

$x := 0,2 \ldots 76$

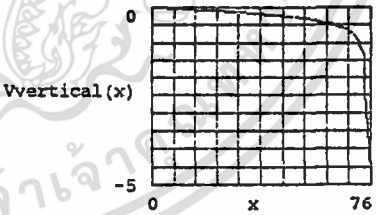
defined: Vvertical=dy/dt , Vhorizontal=dx/dt

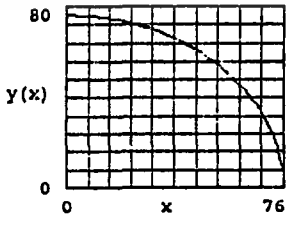
Vhorizontal := 0.33 (Vmotor=40 rpm , pith=5 mm => (40*5)/600=0.33 cm/s)

$$Vvertical(x) := \frac{-x}{\sqrt{c^2 - x^2}} \cdot Vhorizontal$$

$$y(x) := \sqrt{c^2 - x^2}$$

x	y(x)	Vvertical(x)
0	76.2	0
2	76.174	-0.009
4	76.095	-0.017
6	75.963	-0.026
8	75.779	-0.035
10	75.541	-0.044
12	75.249	-0.053
14	74.903	-0.062
16	74.501	-0.071
18	74.044	-0.08
20	73.528	-0.09
22	72.955	-0.1
24	72.322	-0.11
26	71.627	-0.12
28	70.869	-0.13
30	70.046	-0.141
32	69.155	-0.153
34	68.194	-0.165
36	67.16	-0.177
38	66.049	-0.19
40	64.857	-0.204
42	63.58	-0.218
44	62.213	-0.233
46	60.749	-0.25
48	59.181	-0.268
50	57.502	-0.287
52	55.7	-0.308
54	53.763	-0.331
56	51.676	-0.358
58	49.421	-0.387
60	46.973	-0.422
62	44.299	-0.462
64	41.357	-0.511
66	38.085	-0.572
68	34.387	-0.653
70	30.107	-0.767
72	24.949	-0.952





รูปที่3.11ค. กราฟแสดงความสัมพันธ์ระหว่างxและy

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.4.2 ความสัมพันธ์ระหว่างระยะพิทของเกลียวกับความเร็วสูงสุดของเตียงและความเร็วสูงสุดของมอเตอร์

กำหนดให้เตียงมีความเร็วสูงสุดในแนวนอน V_{tmax} cm/s
มอเตอร์ที่ใช้มีขนาด 0.22 kw มีความเร็วสูงสุด V_{mmax} rpm
จาก ในเวลา 60 วินาที มอเตอร์หมุนได้ V_{mmax} รอบ
ในเวลา 1 วินาที มอเตอร์หมุนได้ $\frac{V_{mmax}}{60}$ รอบ
แต่ในเวลา 1 วินาที เตียงเคลื่อนที่ได้ V_{tmax} cm
เพราะฉะนั้น
การเคลื่อนที่ของมอเตอร์ $\frac{V_{mmax}}{60}$ รอบ จะทำให้เตียงเคลื่อนที่ V_{tmax} cm
การเคลื่อนที่ของมอเตอร์ 1 รอบ จะทำให้เตียงเคลื่อนที่ $\frac{V_{tmax}}{V_{mmax}} * 60$
= ระยะพิทของเกลียว ซึ่งก็คือบอลสกรูนั่นเอง

สามารถเขียนเป็นสูตรแสดงความสัมพันธ์ของความเร็วของมอเตอร์, ของเตียง และ ระยะพิทของเกลียวได้ดังต่อไปนี้

ระยะพิทของเกลียว = ความเร็วของเตียง/ ความเร็วของมอเตอร์

หน่วย $\frac{(cm/minute)}{(rpm)}$

ถ้ามอเตอร์มีความเร็วสูงสุดเตียงก็จะมีความเร็วสูงสุดเช่นกัน กล่าวคือ เราสามารถที่จะใช้ความเร็วที่ค่าใดก็ได้ เพราะอัตราส่วนระหว่างความเร็วของเตียงต่อความเร็วของมอเตอร์คงที่ ในการคำนวนจริงๆเราจะไม่ใช้ความเร็วใดๆมาคำนวน เพราะว่าถ้าเรารู้เพียงค่าใดค่าหนึ่งเพียงค่าเดียวเราจะไม่รู้ค่าที่เหลือ ดังนั้นจึงใช้ความเร็วในค่าที่สูงสุดมาคำนวน เพราะว่าถ้าเราต้องการทราบความเร็วสูงสุดของเตียง เราก็จะทราบได้จากค่าความเร็วสูงสุดของมอเตอร์ซึ่งจะบอกค่าให้ทราบอยู่แล้ว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.4.3 การออกแบบการติดตั้งเอนโคดเดอร์สำหรับมุมเอียงของเตียงคนไข้

การติดตั้งเอนโคดเดอร์เพื่อวัดมุมเอียงของเตียง จะต้องทำการติดที่ตำแหน่งฟันคัมของเตียง ดังนั้นจึงจะต้องใช้เอนโคดเดอร์ที่มีความละเอียด(resolution)สูงๆ เพราะการเอียงของเตียง1 องศาจะทำให้เอนโคดเดอร์เอียงไป 1องศาเช่นกัน เพราะฉะนั้นในการหมุน 1 องศาของเอนโคดเดอร์ จะต้องให้จำนวนพัลส์ที่ออกมามากพอที่คอมพิวเตอร์จะตรวจจับได้ ดังนั้นเราจะแสดงการคำนวน ว่าจะได้พัลส์ออกมาเท่าใดในการหมุน 1 องศา เมื่อใช้เอนโคดเดอร์ที่มีจำนวนพัลส์ต่อรอบ ค่าหนึ่ง

จากการหมุนของเอนโคดเดอร์ 360 องศาจะให้พัลส์ออกมา n พัลส์

การหมุนของเอนโคดเดอร์ 1 องศาจะให้พัลส์ออกมา $\frac{n}{360}$ พัลส์

ซึ่งเป็นจำนวนพัลส์ต่อองศา ดังนั้นถ้าเราต้องการให้จำนวน พัลส์/องศา มีค่ามากจะต้องเพิ่มค่า resolution ของเอนโคดเดอร์ (นั่นคือเพิ่มจำนวนพัลส์ต่อรอบ) ถ้าใช้เอนโคดเดอร์ที่มีจำนวนพัลส์/รอบ เท่ากับ 2500

ดังนั้นเราจะได้ว่าในการหมุน 1 องศา จะให้พัลส์ออกมา $= \frac{2500}{360}$

$= 6.94$ พัลส์/องศา

ซึ่งมากพอที่คอมพิวเตอร์จะตรวจจับได้

3.4.4 การคำนวนเพื่อตรวจเช็คว่าเตียงคนไข้เคลื่อนที่เกินค่าความถี่ตอบสนอง (response frequency)ของเอนโคดเดอร์หรือไม่

กำหนดให้ เตียงคนไข้มีความเร็วสูงสุด v cm/s

รัศมีวงล้อของเอนโคดเดอร์มีค่า r cm

ดังนั้น เส้นรอบวง $= 2\pi r$ cm

จะได้ว่า

ระยะทาง $2\pi r$ cm จะให้พัลส์ออกมา n พัลส์

ระยะทาง v cm จะให้พัลส์ออกมา $\frac{n*v}{2\pi r}$ = จำนวนพัลส์/วินาที (ความถี่)

จากหัวข้อที่3.4.2 เราสามารถคำนวนความเร็วสูงสุดของเตียงได้ โดยเราใช้เกลียวที่มีระยะพิทเท่ากับ20 มิลลิเมตร และมอเตอร์มีความเร็วสูงสุด125rpm แทนค่าลงในสูตร ดังนั้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ความเร็วสูงสุดของเตียง = 125*20 mm/m

= 250 cm/m

แทนค่า v ที่หาได้ข้างบน โดยจำนวนพัลส์/รอบของเอนโคดเดอร์ =1200 และรัศมีของล้อเท่ากับ19 cm ดังนั้น

จำนวนพัลส์/วินาที(ความถี่) $=\frac{1200*250}{2*1.9*60}$

= 418 พัลส์/วินาที(เฮิตซ์)

แต่ค่าความถี่ตอบสนองที่บอกมาจากบริษัทผู้ผลิต= 200 กิโลเฮิตซ์ ซึ่งมากกว่า418 เฮิตซ์มาก ดังนั้นจึงไม่มีผลต่อเอนโคดเดอร์

3.4.5 การออกแบบการติดตั้งเอนโคดเดอร์เพื่อให้ได้จำนวนพัลส์สูงสุด

ในการติดตั้งเอนโคดเดอร์เข้ากับเตียง จะต้องมีวงล้อต่อเข้ากับเอนโคดเดอร์ซึ่งวงล้อที่ต่อจะมีรัศมีเท่าใดนั้น เราจะต้องออกแบบเพื่อหาค่าที่เหมาะสมเพื่อที่จะได้จำนวนพัลส์ออกมาสูงที่สุดเท่าที่จะเป็นไปได้ สมมุติให้เตียงเคลื่อนที่ไปเป็นระยะทาง 1 มิลลิเมตรดังนั้น เอนโคดเดอร์ก็จะเคลื่อนที่ไป 1 มิลลิเมตร เช่นเดียวกันเราจะใช้ค่า 1 มิลลิเมตรเป็นค่าอ้างอิง

พิจารณาที่เอนโคดเดอร์

ระยะทาง $2\pi r$ มิลลิเมตร จะให้พัลส์ออกมา n พัลส์

ระยะทาง 1 มิลลิเมตร จะให้พัลส์ออกมา $\frac{n}{2\pi r}$

= จำนวนพัลส์/1มิลลิเมตร

จะเห็นว่าถ้าต้องการให้จำนวนพัลส์ออกมามากๆในการเคลื่อนที่1 มิลลิเมตร จะต้องเพิ่มจำนวนพัลส์/รอบ หรือลดค่า r การเพิ่มค่า n ก็มีข้อจำกัดที่ราคา ซึ่งจะสูงกว่าเอนโคดเดอร์ที่มีค่า n ต่ำๆ ส่วนการลดค่า r ถ้าเล็กไปก็จะทำให้ต่อกับเพลาของเอนโคดเดอร์ยาก

ตัวอย่างการคำนวนสามารถแสดงได้ดังต่อไปนี้

เอนโคดเดอร์ที่ใช้มีจำนวนพัลส์/รอบ = 1200

และ รัศมีมีค่าประมาณเส้นผ่าศูนย์กลางของเอนโคดเดอร์ =19 มิลลิเมตร

แทนค่าลงในสูตรจะได้ค่า $1200/2\pi r*19$

= 10.0 พัลส์/1มิลลิเมตร

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

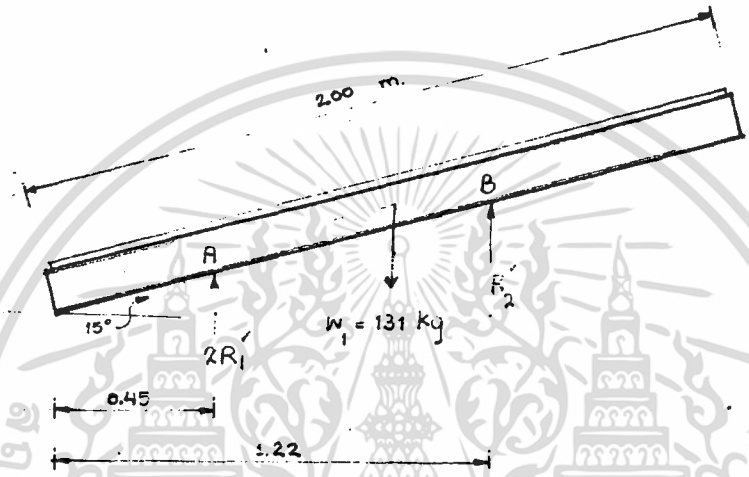
3.4.6 การคำนวนทางกลศาสตร์เพื่อหาแรงที่กระทำต่อส่วนต่างๆของเตียง

สมมุติให้คนไข้มีน้ำหนัก x กิโลกรัม

โครงอลูมิเนียม+โครงเหล็กของเตียงชั้นบน = 58 กิโลกรัม

W_1 = น้ำหนักเตียง +น้ำหนักโครงเหล็ก +น้ำหนักคนไข้

= (58+x) กิโลกรัม



รูปที่3.12 เตียงชั้นบน

R_1': แรงปฏิกิริยาที่เกิดจากจุดฟัลคัมซึ่งมี 2 จุด

R_2': แรงปฏิริยาที่เกิดจากเหล็กโค้งรูปปูมมาแรง

หา R_1', R_2' :

take moment ที่จุด A: $(58+x)kg(96-45)cm = R_2'(122-45)cm$

$$R_2' = 38.41+0.66x(kg) \quad .....(3.14)$$

$\Sigma Fy = 0$; $2R_1' + R_2' = (x+58)kg$

$$R_1' = \left(\frac{x+58-R_2}{2}\right)$$

แทนค่า R_2' จาก(3.14) $R_1' = 9.79+0.69x\ (kg)$ .....(3.15)

เมื่อรวมน้ำหนักมอเตอร์ตัวบน(8.2kg) R_2' จะมีค่าใหม่เป็น

$$R_2' = \left(\frac{2958+51x}{77}\right) + 8.2kg$$

$$= 46.6+0.66x\ (kg) \quad .....(3.16)$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หา R_3' :

จากรูป 3.13 take moment รอบจุดC : ได้ $R_3 = \frac{60}{40} * R_2$

$$= \frac{3}{2}*(46.6+0.66x)kg$$

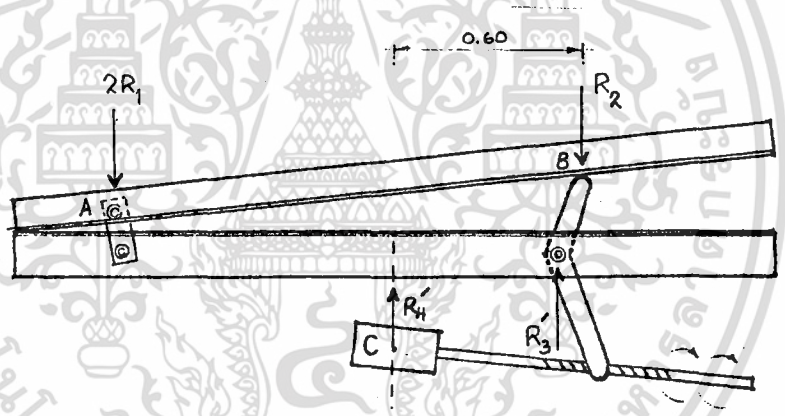
$$R_3' = 69.9+0.99x \text{ (kg)} \quad .....(3.17)$$

เมื่อสมมุติให้คนไข้หนัก 75 กิโลกรัม จะสามารถหาแรงที่กระทำบนส่วนต่างๆของเตียงชั้นบนได้ดังนี้ คือ

$R_1' = 22.5$ kg (220.5 นิวตัน)

$R_2' = 96$ kg (940.8 นิวตัน)

$R_3' = 144.15$ kg (1412.67 นิวตัน)



รูปที่3.13 เตียงชั้นกลาง

หา R_4' :

พิจารณา $\Sigma Fy=0$ ของระบบที่ประกอบด้วยมอเตอร์กับเหล็กโค้งรูปปูมมาแรงที่มีมุมกาง 120 องศา จะได้

$$R_4' + R_3' = R_2 + W_2 \text{(น้ำหนักของมอเตอร์ = 8.2 kg)}$$

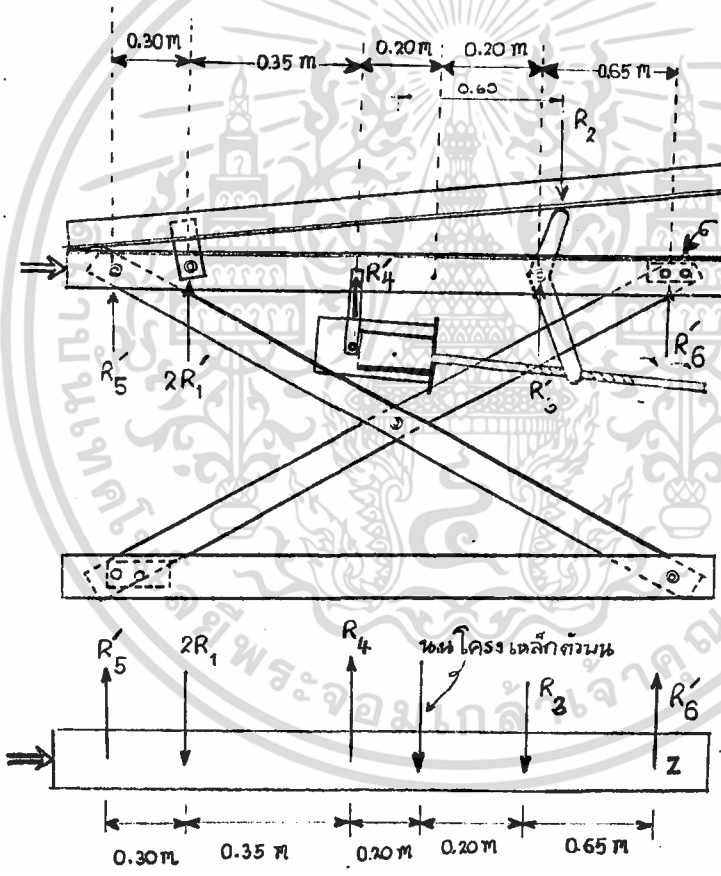
ได้ค่า $R_4' = R_2 - R_3' + W_2$ kg

$$= 46.6+0.66x-69.9-0.99x+8.2$$

$$R_4' = -15.1-0.33x \quad .....(3.18)$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ค่า R_4' ที่ได้มีค่าเป็นลบแสดงว่าเป็นแรงที่กระทำต่อมอเตอร์ และ R_4' เป็นแรงปฏิกิริยาของมอเตอร์



รูปที่3.14 แรงที่กระทำในส่วนที่เคลื่อนที่เชิงมุมของเตียงชั้นกลาง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หา R_5' : จากรูปที่ 3.14 take moment ที่จุด z

(น้ำหนักโครงเหล็กชั้นกลาง = 25 กิโลกรัม)

$$R_5'x(170cm) = R_3x(65cm)+25x(85cm)+R_4x(105cm)+2R_1x(140cm)$$

แทนค่า R_1', R_3', R_4' จะได้

$$R_5' = 46.02+0.87x \qquad .....(3.19)$$

หา R_6' :

เนื่องจาก $R_5' + R_6' = 2R_1 + R_4 + 25kg + R_3$

$$R_6' = 2(9.79 + 0.169x) -15.1-0.33x + 25kg + 69.9 + 0.99x - (46.02 + 0.87x)$$

$$R_6' = 53.36+0.13x \qquad .....(3.20)$$

เมื่อสมมุติให้คนไข้หนัก 75 กิโลกรัม จะสามารถหาค่าของแรงที่กระทำบนส่วนต่างๆของเตียงชั้นกลางได้ดังนี้ คือ

$R_4' = -39.85$ kg (-390.53 นิวตัน)

$R_5' = 111.27$ kg (1090.44 นิวตัน)

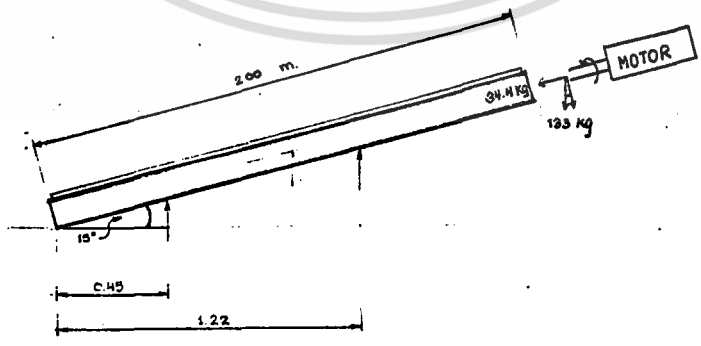
$R_6' = 62.96$ kg (617.0 นิวตัน)

- หาแรง W ที่กระทำต่อบอลสกรูของมอเตอร์ชั้นบน :

จากรูปที่3.16 เมื่อแตกแรงในแนวแกนของสกรู จะได้แรงที่กระทำต่อสกรูมีค่า

$$W = (58 + x)\sin 15^\circ$$

$$W = (15.01 + 0.26x)*9.8 \text{ นิวตัน} \qquad .....(3.21)$$



รูปที่3.15 แสดงการแตกแรงบนบอลสกรูที่กระทำที่เตียงชั้นบน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เมื่อแทนน้ำหนักของคนไข้ซึ่งสมมุติให้เท่ากับ 75 กิโลกรัม จะสามารถหาค่าแรงที่กระทำต่อบอลสกรูของมอเตอรีชั้นบนได้ คือ

$$W = (15.01 + 0.26*75)*9.8$$
$$= 338.198 \text{ นิวตัน}$$

- หาแรง W ที่กระทำต่อสกรูของมอเตอร์ชั้นกลาง :

จากรูปที่ 3.16 take moment รอบจุด A

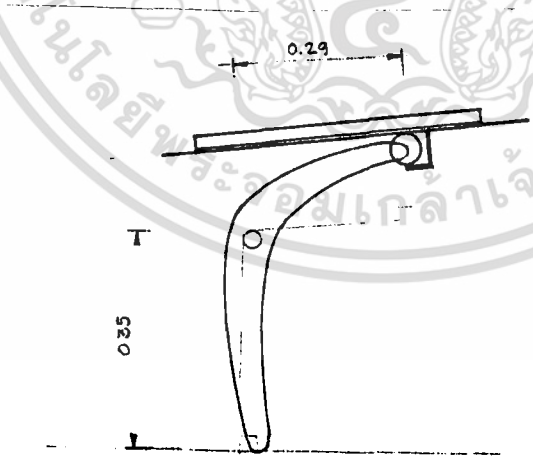
$$R_2*20cm = W*31cm$$
$$W = \frac{R_2*20cm}{31cm}$$

โดยที่ $R_2 = 46.6 + 0.66x$

จะได้ $$W = \frac{(46.6 + 0.66x)(20cm)}{31cm}$$
$$W = (36.5 + 0.43x)*9.8 \text{ นิวตัน} \quad ....(3.22)$$

แทนค่าน้ำหนักของคนไข้(75 กิโลกรัม) ลงในสมการ(3.21)จะได้แรงW ดังนี้ คือ

$$W = (36.5+0.43*75)*9.8$$
$$= 677.964 \text{ นิวตัน}$$



รูปที่ 3.16 แสดงแรงที่กระทำต่อเหล็กโค้งรูปบูมมาแรงในการยกเตียงให้เอียง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- หาแรงที่กระทำต่อสกรูของมอเตอร์ชั้นล่าง :

$$W \star 25cm = R_6 \star 85cm$$

แทนค่า $R_6 = 53.36 + 0.128x$ .....(3.23)

จะได้ $W = (53.36+0.128x)\frac{(85cm)}{25cm}$ กิโลกรัม

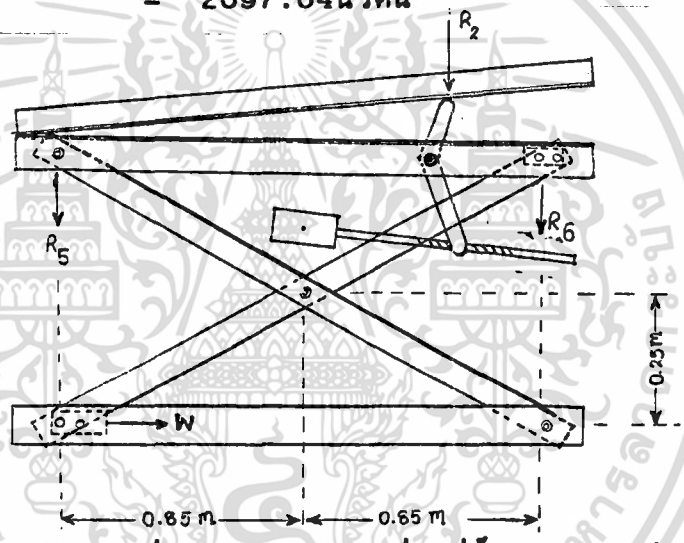
$W = (181.42+0.435x)\star 9.8$ นิวตัน

.....(3.24)

เมื่อคนไข้หนัก75 กิโลกรัม จะสามารถหาค่าของแรงที่กระทำต่อสกรูของมอเตอร์ชั้นล่างได้ คือ

$$W = (181.42 + 0.435\star 75)\star 9.8$$

$$= 2097.64 \text{นิวตัน}$$



รูปที่ 3.17 แสดงแรงที่กระทำต่อส่วนเคลื่อนที่ขึ้น-ลงของเตียงชั้นล่าง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.4.7 การคำนวนหาแรงที่เกิดจากมอเตอร์บนชั้นบน

ข้อกำหนด

-สกรูมีระยะพิท= 5 มิลลิเมตร, เส้นผ่าศูนย์กลางเฉลี่ย = 16 มิลลิเมตร

-เป็นสกรูชนิด 4ปาก

-มอเตอร์มีแรงบิด = 10.2 นิวตัน-เมตร

-มุมเกลียว = 15 องศา

-แรงที่กระทำต่อสกรู W = (15.01+0.26x)*9.8 นิวตัน

-สัมประสิทธิ์ความเสียดทานของสกรูส่งกำลัง = 0.11

การหามุมหลีด

จากสมการที่(3.1) และ(3.2) คือ $l_e = np$ และ $\tan\alpha = \frac{l_e}{\pi d_m}$

แทนค่าลงในสมการ

$$\tan\alpha = \frac{4*5mm.}{\pi *16mm.}$$
$$= 0.3978$$

ได้ว่า $\alpha = 21.6$ องศา

การหาแรงบิดที่มอเตอร์ต้องใช้ขับโหลด

โดยสมมุติให้น้ำหนักของคนไข้ = 75 กิโลกรัม

จากสมการที่(3.6) คือ

$$T_R = \frac{Wd_m}{2}\left[\frac{f_s+\cos\Phi\tan\alpha}{\cos\Phi - f_s\tan\alpha}\right]$$

แทนค่า

$$T_R = \left(\frac{338.198*16}{2*1000}\right)\left[\frac{0.11+\cos(15)\tan(216)}{\cos(15)-0.11\tan(21.6)}\right]$$
$$= 1.444 \text{ นิวตัน-เมตร}$$

แต่แรงบิดสูงสุดของมอเตอร์ = 10.2 นิวตัน-เมตร ซึ่งมีค่ามากกว่าค่า 1.444นิวตัน-เมตร มาก และจากการคำนวนโดยการเปลี่ยนแปลงค่าจากน้ำหนัก 50 กิโลกรัม ถึง 200 กิโลกรัม พบว่า แรงบิดที่200 กิโลกรัม = 2.805นิวตัน-เมตร ซึ่งก็ยังน้อยกว่าแรงบิดสูงสุดของมอเตอร์ แต่เพื่อความปลอดภัย แรงบิดไม่ควรจะเข้าใกล้มอเตอร์จะเข้าใกล้ค่าแรงบิดสูงสุดของมอเตอร์ โดยประมาณว่าน้ำหนักของคนไข้สูงสุดไม่ควรจะเกิน150 กิโลกรัม ซึ่งจะต้องใช้แรงบิดเท่ากับ 2.261 นิวตัน-เมตร

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่3.2 แสดงค่าแรงบิดของมอเตอร์ที่ใช้ในการขับเตียงชั้นบน ในช่วงน้ำหนักของคนไข้ที่มีมวลตั้งแต่ 50-200 กิโลกรัม*

Designed Table Move in Horizontal Direction

Defined x is the patient's weight

x := 50,60 ..200 kg /*the weight is vary from 50-200 */

w(x) := 15.01 + 0.26·x kg /*the equation of force that act to power screws*/

w(x) := w(x)·9.8 N /*Change to Newtons */

$f_s := 0.11$ N/m/s /*Frictional coefficient*/

$\alpha := 21.6 \cdot \frac{\pi}{180}$ radain

$\phi := 15 \cdot \frac{\pi}{180}$

$d_m := 16$ (average dimeter of screws)

$$T_R(x) := \left[w(x) \cdot \frac{9.8}{1000} \cdot \frac{d_m}{2} \right] \cdot \frac{f_s + (\cos(\phi) \cdot \tan(\alpha))}{\cos(\phi) - \left[f_s \cdot \tan(\alpha) \right]}$$

MAXIMUM MOTOR TORQUE : 10.2 N.M

(kg)	$T_R(x)$ (Nm)	(kg) w(x)
50	11.489	274.498
60	12.556	299.978
70	13.622	325.458
80	14.689	350.938
90	15.755	376.418
100	16.822	401.898
110	17.888	427.378
120	18.955	452.858
130	20.021	478.338
140	21.088	503.818
150	22.154	529.298
160	23.221	554.778
170	24.287	580.258
180	25.354	605.738
190	26.42	631.218
200	27.487	656.698

* ค่าจากการคำนวน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.4.8 การคำนวนหาแรงที่เกิดจากมอเตอร์ชั้นกลาง

ทอร์กของมอเตอร์(T) = 30.5 นิวตัน-เมตร

เกลียวมีขนาดเส้นผ่าศูนย์กลางเฉลี่ย = 23.5 มิลลิเมตร

มีระยะพิท = 5 มิลลิเมตร

(จากตารางในภาคผนวก)

<u>การหามุมหลีด</u>

จากสมการที่(3.2) จะได้ว่า

$$\tan\alpha = l_e/\pi d_m$$

เกลียวที่ใช้เป็นเกลียวชนิดหนึ่งปาก ดังนั้นจึงมีระยะหลีด l_e เท่ากับระยะพิท = 5 มิลลิเมตร

จะได้ $\alpha = \tan^{-1}\left(\frac{5mm.}{23.5\star p}\right)$

$= \tan^{-1}(0.06772)$

$= 3.874$ องศา

$\tan\alpha = 0.067225$

<u>การหาขนาดของทอร์กที่เกิดจากสกรูส่งกำลัง</u>

<u>ข้อกำหนด</u>

- จากการคำนวนได้ค่า $W = (36.5 + 0.43x)\star 9.8$ นิวตัน
- สมมุติให้น้ำหนักคนไข้ = 75 กิโลกรัม จะได้ $W = 673.75$ นิวตัน
- $\Phi = 30/2 = 15$ องศา
- สัมประสิทธิ์ความเสียดทานของสกรูส่งกำลัง = 0.20 = f_s (จากตารางที่ภาคผ

จากสมการที่(3.6)

$$T_R = \frac{Wd_m}{2}\left[\frac{f_s+\cos\Phi\tan\alpha}{\cos\Phi - f_s\tan\alpha}\right] + r_{mc}f_cW$$

แทนค่าลงในสมการ

$$T_R=\frac{673.75}{2}\star 23.5\star 10^{-3}\left[\frac{0.2+\cos(15^\circ)\tan(3.874^\circ)}{\cos(15^\circ)-0.2\tan(0.874^\circ)}\right]$$

$= 2.206$ นิวตัน-เมตร

นั่นคือสกรูส่งกำลังต้องการแรงบิด(ทอร์ก)ขนาด 2.206นิวตัน-เมตร เพื่อที่จะยกคนไข้ แต่ทอร์กของมอเตอร์มีขนาด 30.5นิวตัน-เมตร ซึ่งมากกว่าค่า 2.206 มาก ดังนั้น เตียงสามารถยกคนไข้ที่มีน้ำหนัก 75 กิโลกรัมได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่3.3 แสดงค่าแรงบิดของมอเตอร์ที่ใช้ในการขับเตียงชั้นกลาง ในช่วงน้ำหนักของคนไข้ตั้งแต่ 50-200 กิโลกรัม*

Designed Table Move Inclined Direction

Defined x is weight of patient

$x := 50,60 \ldots 200$ kg /* the weight is vary from 50-200 kg */

$w(x) := 36.5 + 0.43 \cdot x$ kg /*the force's equation that act to power screws*/

$w(x) := w(x) \cdot 9.8$ N /*Change to Newtons*/

$f_s := 0.2$ N/m/s

$\alpha := 3.874 \cdot \frac{\pi}{180}$ radain

$\phi := 15 \cdot \frac{\pi}{180}$

$d_m := 23.5$ /*average dimeter of screws*/

$$T_R(x) := \left[\frac{w(x)}{1000} \cdot \frac{d_m}{2}\right] \cdot \frac{f_s + (\cos(\phi) \cdot \tan(\alpha))}{\cos(\phi) - [f_s \cdot \tan(\alpha)]}$$

MAXIMUM MOTOR TORQUE: 30.5 N.M

x (kg)	$T_R(x)$ (Nm)	w(x) (newtons)
50	1.861	568.4
60	1.999	610.54
70	2.137	652.68
80	2.275	694.82
90	2.413	736.96
100	2.551	779.1
110	2.689	821.24
120	2.827	863.38
130	2.965	905.52
140	3.103	947.66
150	3.241	989.8
160	3.379	$1.032 \cdot 10^3$
170	3.517	$1.074 \cdot 10^3$
180	3.655	$1.116 \cdot 10^3$
190	3.793	$1.158 \cdot 10^3$
200	3.931	$1.201 \cdot 10^3$

* ค่าจากการคำนวน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.4.9 การคำนวนหาแรงที่เกิดจากมอเตอร์ชั้นล่าง

ข้อกำหนด

-สกรูมีระยะพิท = 5 มิลลิเมตร, เส้นผ่าศูนย์กลาง 28 มิลลิเมตร โดยมีเส้นผ่าศูนย์กลางเฉลี่ย 25.5 มิลลิเมตร(จากตาราง ในภาคผนวก)

-มุมเกลียว 15 องศา

-แรงที่กระทำต่อสกรูชั้นล่าง $W = (150.38 + 1.428x)*9.8$ นิวตัน

-สกรูเป็นแบบปากเดียว โดยมีสัมประสิทธิ์ความเสียดทาน $f_s = 0.2$

การหามุมหลีด

$$\tan\alpha = \frac{l_s}{\pi d} = \frac{5mm.}{\pi * 25.5mm.} = 0.0624$$

$$\alpha = 3.571 \text{ องศา}$$

การหาแรงบิดที่มอเตอร์ใช้ในการขับโหลด

โดยกำหนดให้น้ำหนักของคนไข้ = 75 กิโลกรัม

$$T_R = \frac{-2101.3}{2} * 25.5 * 10^{-3}\left(\frac{0.2+\cos(15^\circ)\tan(3.571^\circ)}{\cos(15^\circ)-0.2\tan(3.571^\circ)}\right)$$

$$= 7.301 \text{ นิวตัน-เมตร}$$

จะเห็นว่าแรงบิดที่ต้องการใช้ยกเตียงมีค่าน้อยกว่าแรงบิดสูงสุดของมอเตอร์ (30 นิวตัน-เมตร) ดังนั้นเตียงสามารถยกคนไข้ซึ่งหนัก 75 กิโลกรัมได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่3.4 แสดงค่าแรงบิดของมอเตอร์ตัวล่างที่ใช้ในการขับเตียง เมื่อน้ำหนักของคนไข้เปลี่ยนแปลงอยู่ในช่วงตั้งแต่ 50-200 กิโลกรัม*

Designed Table Move Up_down Direction

Defined x is patient's weight

$x := 50,60 ..200 \ kg$ /*the patient's weight is vary from 50-200*/

$w(x) := 181.42 + 0.435 \cdot x \ kg$ /*the equation of force that act to screws*/

$w(x) := w(x) \cdot 9.8 \ N$ /*Change to Newtons*/

$f_s := 0.2 \ N/m/s$ /*Friction coefficient of power screws */

$\alpha := 3.571 \cdot \frac{\pi}{180}$ radian

$\phi := 15 \cdot \frac{\pi}{180}$

$d_m := 25.5$ /*Average dimeter of power screws*/

$$T_R(x) := \left[\frac{w(x)}{1000} \cdot \frac{d_m}{2}\right] \cdot \frac{f_s + (\cos(\phi) \cdot \tan(\alpha))}{\cos(\phi) - \left[f_s \cdot \tan(\alpha)\right]}$$

MAXIMUM MOTOR TORQUE: 30.5 N.M

x (kg)	$T_R(x)$ (Nm)	w(x) (newtons)
50	6.93	$1.991 \cdot 10^3$
60	7.079	$2.034 \cdot 10^3$
70	7.227	$2.076 \cdot 10^3$
80	7.375	$2.119 \cdot 10^3$
90	7.524	$2.162 \cdot 10^3$
100	7.672	$2.204 \cdot 10^3$
110	7.82	$2.247 \cdot 10^3$
120	7.969	$2.289 \cdot 10^3$
130	8.117	$2.332 \cdot 10^3$
140	8.266	$2.375 \cdot 10^3$
150	8.414	$2.417 \cdot 10^3$
160	8.562	
170	8.711	
180	8.859	
190	9.007	
200	9.156	

* ค่าจากการคำนวน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.5 วงจรควบคุมการเคลื่อนที่ของเตียงคนไข้

สามารถแบ่งการควบคุมการเคลื่อนที่ของเตียงได้เป็น 2 แบบใหญ่ๆ คือ การควบคุมโดยซอฟแวร์ และ การควบคุมโดยฮาร์ดแวร์

3.5.1 การควบคุมโดยซอฟแวร์

เป็นการควบคุมการเคลื่อนที่โดยใช้โปรแกรมคอมพิวเตอร์ในการสั่งอุปกรณ์ โดยมีไอซี LM 628 เป็นโปรเซสเซอร์ในการควบคุมมอเตอร์ ให้เอาท์พุทเป็นไฟดีซี ± 10 โวลต์ ผ่านตัวไดร์เวอร์เข้าสู่มอเตอร์ และมีเอนโคดเดอร์เป็นตัวป้อนกลับเป็นตัวป้อนกลับสัญญาณ เพื่อทำการควบคุมตำแหน่งและความเร็ว

ภายในตัวเตียงที่ชั้นบนสุดมีเซนเซอร์ต่างๆกัน คือ ออปโต สลอท คัปเปิล (opto slotted coupled)และไมโครสวิทช์ การควบคุมการเคลื่อนที่ของเตียงโดยการอ่านสถานะของ ออปโต และไมโครสวิทช์ ที่ตำแหน่งของเซนเซอร์เหล่านี้ ตำแหน่งของเซนเซอร์ที่อยู่บนเตียงในแนวนอน คือ

1 โฮม(Home)เป็นตำแหน่งที่ใช้อ้างอิงในการเคลื่อนที่ของเตียง โดยการกำหนดให้มีค่าเป็น 0

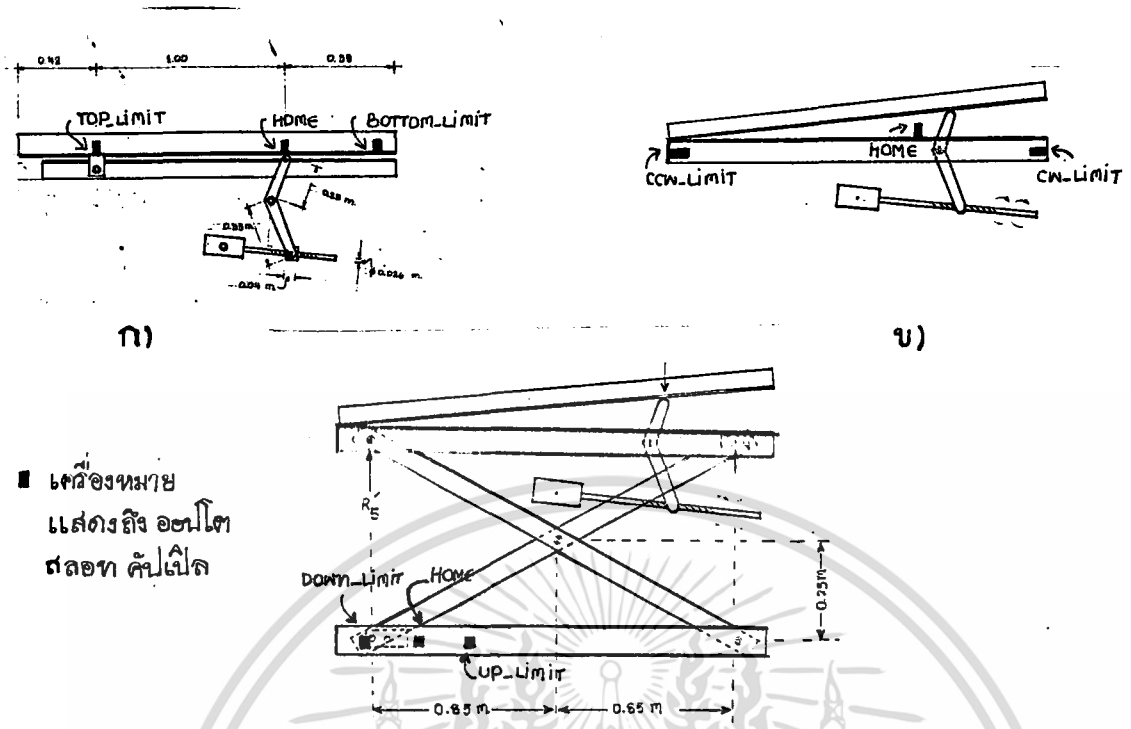
2 Bottom_limit คือตำแหน่งการเคลื่อนที่ไกลสุดทางด้านท้ายเตียง โดยเมื่อถึงตำแหน่งนี้โปรแกรมก็จะสั่งให้หยุด หรือเคลื่อนที่ถอยหลังเพื่อไม่ให้เลยไป แต่ถ้าเลยก็เป็นหน้าที่ของฮาร์ดแวร์ที่จะต้องทำการหยุดการเคลื่อนที่ของเตียง

3 Top_limit คล้ายกันกับ Bottom_limit เพียงแต่เป็นด้านตรงข้าม

ส่วนเซนเซอร์ที่ชั้นกลางที่มีการเคลื่อนที่เชิงมุมก็มีตัวเซนเซอร์ 3ตัวเช่นกัน คือ ที่ตำแหน่งในแนวขนานกับพื้นจะเป็นโฮม ตำแหน่งมุมสุดทางด้านตามเข็มนาฬิกาจะเป็น cw_limit และที่ฝั่งตรงข้ามก็จะเป็น ccw_limit

ในทำนองเดียวกัน ชั้นล่างสุดซึ่งเป็นการเคลื่อนที่ขึ้นลงก็จะเหมือนกับสองชั้นข้างต้น คือการเคลื่อนที่ขึ้นสูงสุดก็จะเป็น up_limit และในทางตรงข้ามถ้าเป็นการเคลื่อนที่ลงต่ำสุดจะเป็น down_limit และโฮม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ก)

ข)

ค)

รูปที่ 3.18 แสดงภาพตำแหน่งที่ใช้ติดตั้ง ออปโต สลอท คัปเปิล
ก) เตียงชั้นบน
ข) เตียงชั้นกลาง
ค) เตียงชั้นล่าง

3.5.2 การควบคุมโดยฮาร์ดแวร์

จะเป็นการตัดการจ่ายไฟให้แก่มอเตอร์ในกรณีที่การควบคุมโดยซอฟแวร์ไม่ได้ผล ซึ่งอาจเนื่องมาจากเซนเซอร์ทางแสงเสีย ก็จะเป็นหน้าที่ของไมโครสวิทช์ซึ่งจะทำหน้าที่เป็นลิมิตสวิทช์ โดยจะทำหน้าที่ตัดไฟที่จ่ายให้กับมอเตอร์ เมื่อไมโครสวิทช์ทำงานแล้วระบบก็จะมีสถานะที่ถูกตัดไฟตลอดเวลา ซึ่งเป็นข้อดีของสวิทช์ประเภทนี้ เพราะทำให้รู้ว่าระบบเกิดขัดข้องซึ่งจะต้องมีการเช็คระบบหรือเปิดเครื่องใหม่โดยผู้ชำนาญหรือช่างเทคนิคประจำเครื่อง เมื่อต้องการที่จะให้เลิกการตัดการจ่ายไฟสามารถทำได้โดยการกดปุ่มที่อยู่ใต้สวิทช์

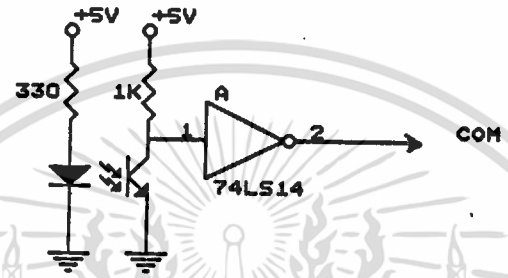
การอ่านการเปลี่ยนแปลงสถานะของออปโต หรือไมโครสวิทช์จะกระทำผ่านบอร์ดควบคุมก่อนเข้าคอมพิวเตอร์ วงจรของบอร์ดควบคุมแสดงในรูปที่ 3.20 โดยสามารถแสดงเป็นส่วนๆได้ดังต่อไปนี้

3.5.2.1 ส่วนที่ใช้ควบคุมโดยซอฟแวร์

ใช้ออปโต สลอท คัปเปิล เป็นตัวเซนเซอร์ตำแหน่งโฮม,Top และ Bottom_limit โดยที่สถานะปกติของเซนเซอร์ตัวนี้จะเป็นhigh ทั้งนี้เพราะว่า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

คอมพิวเตอร์เช็คการอินเทอร์รัพท์ที่edge trig จากขอบบนลงล่าง เมื่อมีวัตถุมากั้นทางเดินของแสงอินฟาเรดซึ่งวิ่งระหว่างอิมิตเตอร์ กับดีเทคเตอร์ สถานะก็จะเปลี่ยนเป็นlow และเพื่อป้องกันโวลต์เตจตกจึงได้ใช้ไอซีชมิททริกเกอร์เพื่อช่วยเปลี่ยนสถานะให้เหมาะสมก่อนเข้าสู่คอมพิเตอร์ ไอซีชมิททริกเกอร์ที่ใช้ในวงจรนี้เป็นแบบอินเวอร์เตอร์



รูปที่3.19 วงจรของออปโต สลอท คัปเปิล

3.5.2.2 ส่วนที่ใช้ในการตัดไฟที่จ่ายให้กับมอเตอร์

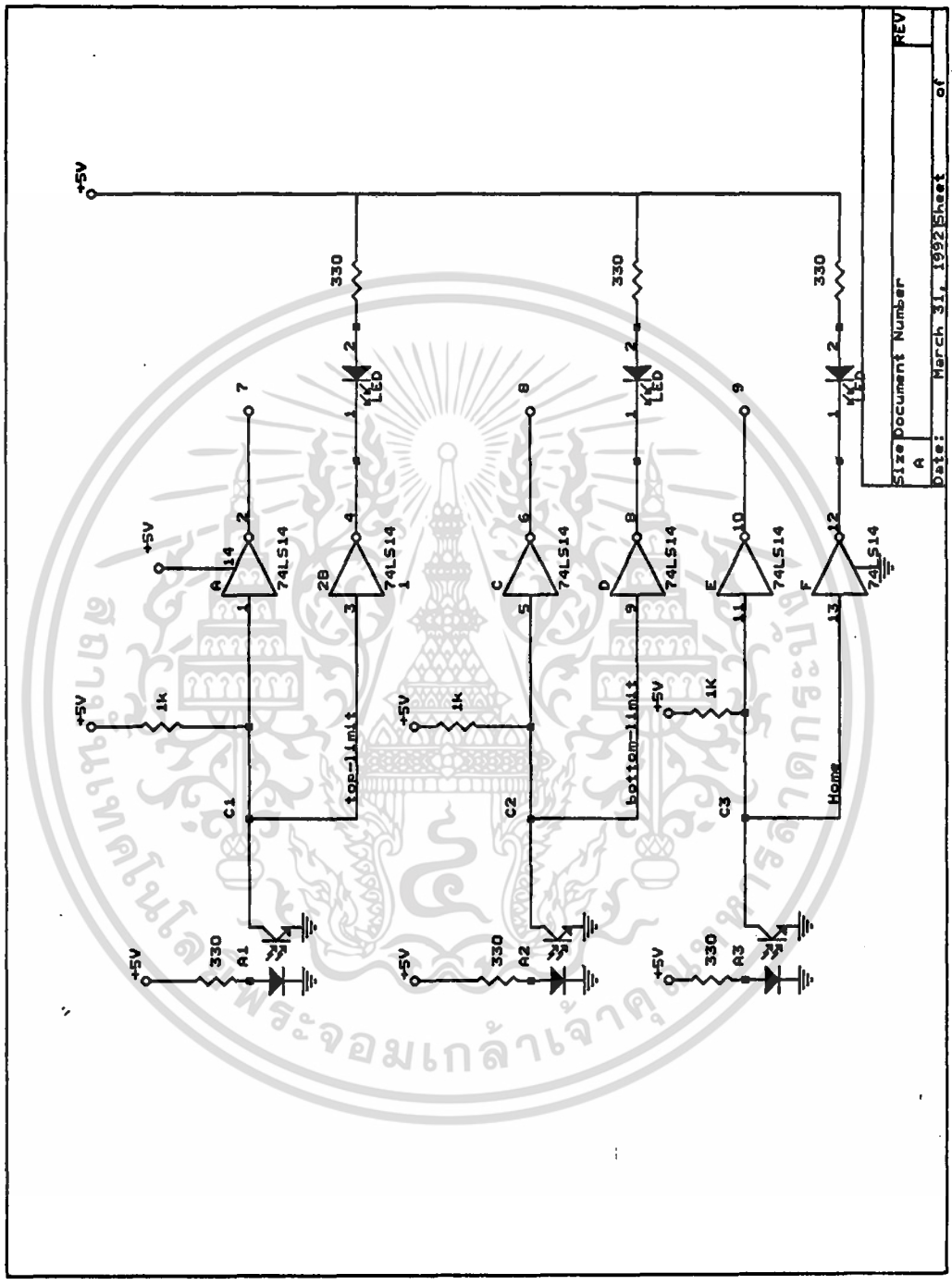
ประกอบด้วยไมโครสวิทช์ และลีเลย์(lelay) ที่จริงแล้วถ้าสามารถหาไมโครสวิทช์ที่หน้าสัมผัสทนกระแสได้สูง 1.2 แอมป์ได้ ก็ไม่จำเป็นที่จะต้องใช้ลีเลย์เลย แต่เราไม่สามารถหาไมโครสวิทช์ที่มีคุณสมบัติดังกล่าว ดังนั้นจึงได้ใช้ลีเลย์เข้าช่วย

การทำงานของวงจรคือปกติไมโครสวิทช์จะอยู่ในสถานะเปิด(normal-open) และลีเลย์จะอยู่ในในสถานะปิด(normal close) โดยมีไฟ +12 โวลต์จ่ายให้กับ คอล์ย(coil)เมื่อมีการกดไมโครสวิทช์ ไฟ +12 โวลต์ก็จะผ่านเข้าสู่คอล์ยทำให้สถานะของลีเลย์เป็นสถานะเปิด นั่นคือจะทำหน้าที่ตัดไฟที่จ่ายให้แก่มอเตอร์ ลีเลย์ที่ใช้เป็นแบบ 2 หน้าสัมผัส โดยหน้าสัมผัสแรกจะทำหน้าที่ตัดไฟ ส่วนหน้าสัมผัสที่สองจะต่อเข้ากับคอมพิวเตอร์เพื่อสั่งให้มอเตอร์หยุดอีกครั้ง ซึ่งเป็นการป้องกันอีกชั้นหนึ่ง และจะแสดงผลที่หน้าจอด้วยว่าระบบเกิดการขัดข้อง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

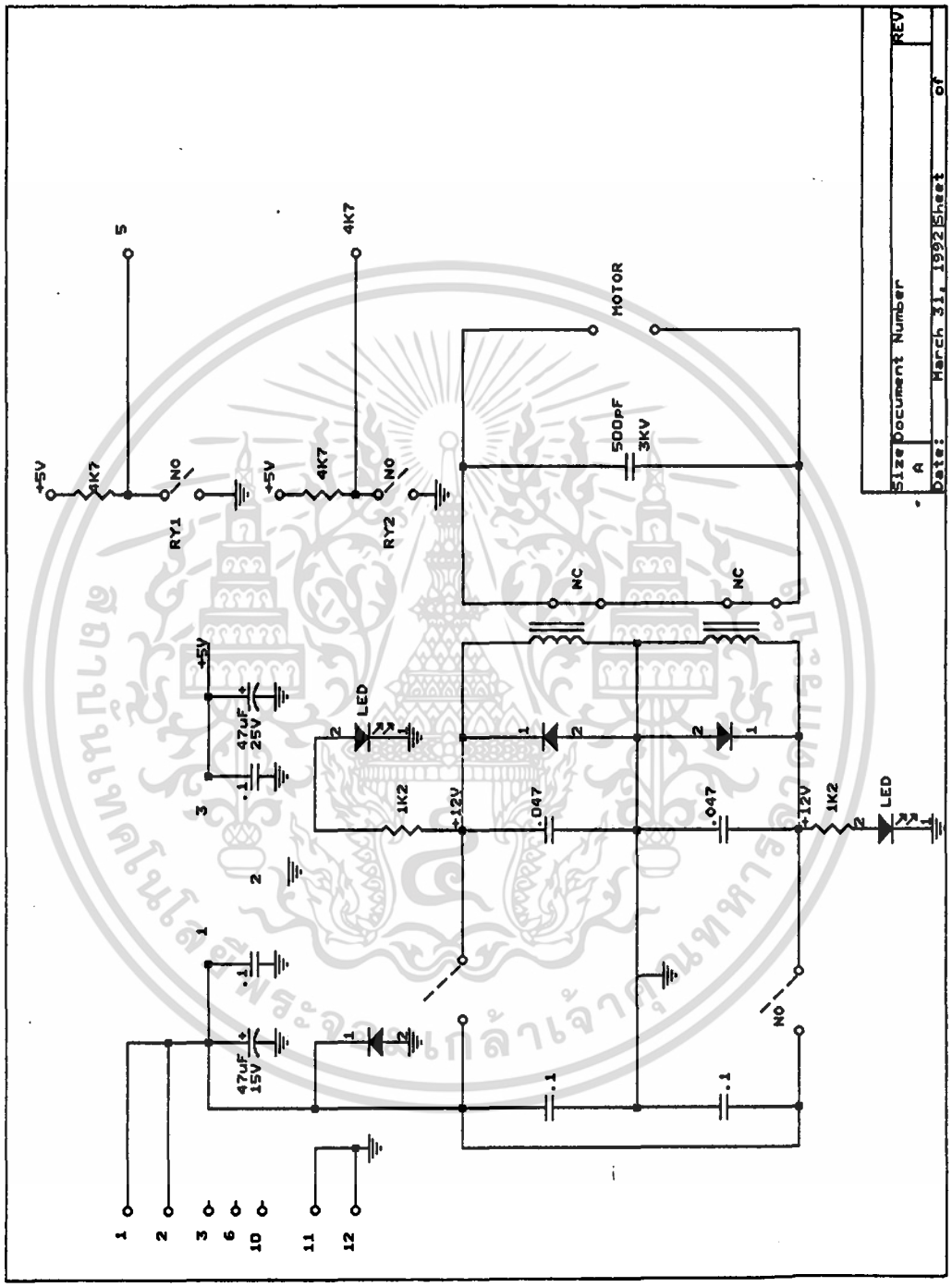
รูปที่3.20 บอร์ดควบคุมการจ่ายไฟให้แก่มอเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่3.21 วงจรของบอร์ดควบคุมการจ่ายไฟให้แก่มอเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่3.21 (ต่อ) วงจรของบอร์ดควบคุมการจ่ายไฟให้แก่มอเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ขั้นตอนการทดลอง

เตียงคนไข้ชั้นบน ตอนที่ 1

1 วัดค่าพารามิเตอร์ต่างๆ คือ

1.1 เส้นผ่าศูนย์กลางของเอนโคดเดอร์

1.2 จำนวนพัลส์ที่นับจากตำแหน่งโฮมถึงตำแหน่งไกลสุดหัวเตียง

1.3 จำนวนพัลส์ที่นับจากตำแหน่งโฮมถึงตำแหน่งไกลสุดท้ายเตียง

2 คำนวนระยะทางที่สอดคล้องกับตำแหน่งไกลสุดหัวเตียงโดยใช้สูตร

$$\text{ระยะทาง(mm)} = \frac{(\pi * DIM * (LIM\ DIS))}{R}$$

DIM = เส้นผ่าศูนย์กลางของเอนโคดเดอร์

LIM_DIS = จำนวนพัลส์จากตำแหน่งโฮมถึงตำแหน่งไกลสุด

R = จำนวนพัลส์ต่อรอบของเอนโคดเดอร์

3 คำนวนระยะทางจากตำแหน่งโฮมถึงตำแหน่งไกลสุดท้ายเตียงจากจำนวนพัลส์ที่นับได้

4 หาจำนวนระยะทางที่เคลื่อนที่ใน 1 พัลส์ของเอนโคดเดอร์

ตอน 2

1 ตั้งค่าคงที่ $k_p = 28$ $k_i = 7$ $k_d = 80$

2 สั่งให้เตียงเคลื่อนที่ไป 10 มิลลิเมตร บันทึกผลในตารางบันทึกผลการทดลอง

3 เพิ่มระยะทางไปอีก 10 มิลลิเมตร สั่งให้เตียงเคลื่อนที่และบันทึกผล

4 ทำตามข้อ 3 จนครบ 9 ครั้ง

5 สั่งให้เตียงเคลื่อนที่ไป 100 มิลลิเมตร บันทึกผล

6 ทำตามข้อ 5 จนครบ 3 ครั้ง

7 กลับไปยังตำแหน่งโฮม

8 สั่งให้เตียงเคลื่อนที่ไป 0.02 มิลลิเมตร และบันทึกผล

9 ทำตามข้อ 7-8 แต่เคลื่อนที่ไป 0.05,0.1,0.2,1 มิลลิเมตร ตามลำดับ

10 เปลี่ยนค่า PID เป็น $k_p = 10$ $k_i = 5$ $k_d = 80$

11 ทำตามข้อ 2 - 9 เปรียบเทียบผลที่ได้ก่อนและหลังการเปลี่ยนค่าคงที่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เตียงคนไข้ชั้นกลาง ตอน 1

1 วัดค่าพารามิเตอร์ต่าง ๆ คือ

1.1 เส้นผ่าศูนย์กลางของเอนโคดเดอร์

1.2 จำนวนพัลส์ที่นับจากตำแหน่งโฮมถึงตำแหน่งมุมเอียงขึ้นไกลสุด

1.3 จำนวนพัลส์ที่นับจากตำแหน่งโฮมถึงตำแหน่งมุมเอียงลงไกลสุด

1.4 รัศมีของเตียงมุมเอียงที่วัดจากจุดฟัลคัม

2 คำนวนระยะทางที่สอดคล้องกับจำนวนพัลส์ที่นับได้จากโฮมถึงตำแหน่งมุมเอียงไกลสุดทั้งสอง

3 คำนวนองศาที่สอดคล้องกับระยะทางจากโฮมถึงตำแหน่งมุมเอียงไกลสุดทั้งสองโดยใช้สูตร

$$\tan\alpha = \frac{S}{R}$$

S = ระยะทางที่เคลื่อนที่จากตำแหน่งโฮมถึงตำแหน่งมุมเอียงไกลสุดทั้งสอง

R = รัศมีของเตียงวัดจากจุดฟัลคัม

4 หาจำนวนองศาต่อการเคลื่อนที่ใน 1 พัลส์ของเอนโคดเดอร์

ตอน 2

1 ตั้งค่าคงที่ PID $k_p = 28$ $k_i = 7$ $k_d = 80$

2 สั่งให้เตียงเคลื่อนที่ไป 0.5 องศา บันทึกผลการทดลองในตาราง

3 เพิ่มการเคลื่อนที่ไปอีก 0.5 องศา บันทึกผลการทดลอง

4 ทำตามข้อ 3 จนครบ 6 ครั้ง

5 สั่งให้เตียงเคลื่อนที่ไปที่ตำแหน่งโฮม

6 สั่งให้เตียงเคลื่อนที่ไป 0.002 องศา บันทึกผล

7 ทำตามข้อ 5-6 แต่เปลี่ยนมุมเป็น 0.01,0.05 ตามลำดับ

8 เปลี่ยนค่าคงที่ PID เป็น $k_p = 7$ $k_i = 5$ $k_d = 200$

9 ทำตามข้อ 2-7 เปรียบเทียบผลที่ได้ก่อนและหลังเปลี่ยนค่าคงที่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่4
ผลการวิจัย และวิจารณ์

ผลการทดลอง

1 เตียงคนไข้ชั้นบน

ค่าพารามิเตอร์สำคัญที่ใช้ในการคำนวณได้แก่

1 เส้นผ่าศูนย์กลางของเอนโคดเดอร์เท่ากับ 32 มิลลิเมตร

2 ความละเอียดของเอนโคดเดอร์เท่ากับ 1200 พัลส์ต่อรอบ

3 จำนวนพัลส์ที่นับได้จากตำแหน่ง HOME (ตำแหน่งอ้างอิง ให้ค่าเป็นศูนย์) ถึงตำแหน่งหัวเตียงเท่ากับ 14800 พัลส์

4 จำนวนพัลส์ที่นับได้จากตำแหน่ง HOME$^{\downarrow}$ถึงตำแหน่งท้ายเตียงเท่ากับ 1000 พัลส์

การคำนวนระยะทางที่สัมพันธ์กับจำนวนพัลส์ที่นับได้

-การคำนวณระยะทางจากตำแหน่ง HOME ถึงแหน่งหัวเตียง

เส้นรอบวงของเอนโคดเดอร์เท่ากับ $d = 100.53$ mm

เอนโคดเดอร์หมุน 1 รอบจะให้พัลส์ 1200*4 พัลส์(ดูที่ภาคผนวก)

ดังนั้นจำนวนพัลส์ 14000 จะเท่ากับระยะทาง

$$\frac{(100.53\text{mm} * 14000)}{(4800)}$$

ระยะทางจากจากตำแหน่ง HOME ถึงแหน่งหัวเตียงเท่ากับ 502.65 mm

-การคำนวณระยะทางจาก Home ถึงตำแหน่งท้ายเตียง

จำนวนพัลส์ที่นับได้จากตำแหน่ง HOME$^{\downarrow}$ถึงตำแหน่งท้ายเตียงเท่ากับ 1000 พัลส์

เพราะฉนั้นระยะทางที่สัมพันธ์กันคือ $\frac{(100.53*1000)}{(4800)}$

-แสดงการหาค่าความละเอียด(resolution)ของเตียง

จำนวนพัลส์ 4800 เท่ากับระยะทาง 100.53 mm

" 1 " $\frac{(100.53)}{4800)} = 0.02$ mm

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สั่งให้เคลื่อนที่ไปครั้งละ 10 mm $K_p = 28, K_i = 7, K_d = 80$

จำนวนระยะทางที่สั่ง(mm)	ค่าที่อ่านได้ (mm)	จำนวนพัลส์ที่สั่ง	จำนวนพัลส์ที่อ่านได้	จำนวนพัลส์ที่ผิดพลาด
10	9.906	477	473	4
19.906	19.771	950	944	6
29.771	29.636	1421	1415	6
39.636	39.50	1892	1886	6
49.5	49.886	2363	2358	5
59.886	59.271	2835	2830	5
69.271	69.136	3307	3301	6
79.136	79.002	3778	3773	5
89.002	88.886	4250	4244	6
98.886	98.772	4721	4716	5
198.772	198.753	9490	9485	5
298.535	298.535	14259	14254	5
398.535	398.373	19029	19024	5

สั่งให้เคลื่อนที่เทียบกับโฮม

จำนวนระยะทางที่สั่ง(mm)	ค่าที่อ่านได้ (mm)	จำนวนพัลส์ที่สั่ง	จำนวนพัลส์ที่อ่านได้	จำนวนพัลส์ที่ผิดพลาด
0.02	0.0	1	0	-
0.1	0	4	0	-
0.2	0.063	9	3	6
0.5	0.398	23	19	4

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สั่งให้เคลื่อนที่ไปครั้งละ 10 mm $K_p = 10, K_i = 5, K_d = 80$

จำนวนระยะทางที่สั่ง(mm)	ค่าที่อ่านได้ (mm)	จำนวนพัลส์ที่สั่ง	จำนวนพัลส์ที่อ่านได้	จำนวนพัลส์ที่ผิดพลาด
10	9.613	477	459	18
19.613	19.185	950	930	20
29.185	28.185	1421	1404	17
38.185	38.369	1892	1871	21
48.369	47.920	2363	2342	21
57.920	57.554	2835	2825	18
67.554	67.146	3307	3288	19
76.146	76.718	3778	3760	18
86.718	86.289	4250	4230	20
96.289	95.860	4721	4700	21
195.860	195.009	9490	9470	20
295.009	298.556	14259	14238	21
398.556	394.060	19029	19006	23

สั่งให้เคลื่อนที่เทียบกับโฮม

จำนวนระยะทางที่สั่ง(mm)	ค่าที่อ่านได้ (mm)	จำนวนพัลส์ที่สั่ง	จำนวนพัลส์ที่อ่านได้	จำนวนพัลส์ที่ผิดพลาด
0.02	0.0	1	0	–
0.1	0	4	0	–
0.2	0	9	0	0
0.5	0.063	23	3	20

★หมายเหตุ เครื่องหมาย – หมายถึง เตียงไม่เคลื่อนที่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2 เตียงคนไข้ชั้นกลาง

ค่าพารามิเตอร์สำคัญที่ใช้ในการคำนวณได้แก่

1 เส้นผ่าศูนย์กลางของเอนโคดเดอร์เท่ากับ 33.5 มิลลิเมตร

2 ความละเอียดของเอนโคดเดอร์เท่ากับ 2500 พัลส์ต่อรอบ

3 จำนวนพัลส์ที่นับได้จากตำแหน่ง HOME (ตำแหน่งศูนย์)ถึงตำแหน่งมุมสูงสุดเอียงลง(ใช้หัวเตียงเป็นจุดอ้างอิง) เท่ากับ 148000 พัลส์

4 จำนวนพัลส์ที่นับได้จากตำแหน่ง HOME ถึงตำแหน่งมุมสูงสุดเอียงขึ้น (ใช้หัวเตียงเป็นจุดอ้างอิง) เท่ากับ 3600 พัลส์

5 รัศมีของเตียงวัดจากจุดฟัลคัมเท่ากับ 1535 มิลลิเมตร

การคำนวณระยะทางที่สัมพันธ์กับจำนวนพัลส์ที่นับได้

-หาความยาวเส้นรอบวงของเอนโคดเดอร์

เส้นผ่าศูนย์กลางของเอนโคดเดอร์ 33.2 mm

เพราะฉะนั้นเส้นรอบวงของเอนโคดเดอร์เท่ากับ $33.2 \star \pi = 105.243$ mm

-การคำนวนระยะทางจากตำแหน่ง HOME ถึงตำแหน่งมุมสูงสุดเอียงขึ้น

เอนโคดเดอร์ให้ 2500*4 พัลส์ เท่ากับระยะทาง 105.243 mm

$$148000 \quad \frac{(105.243 \star 14800)}{(1000)}$$

$$= 155.76 \text{ mm}$$

-การคำนวนระยะทางจากตำแหน่ง HOME ถึงแหน่งมุมสูงสุดเอียงลง

$$\frac{(105.243 \star 3600)}{(10000)}$$

$$= 37.8875 \text{ mm}$$

-หาองศาสูงสุดมุมเอียงลง

จากสูตร $\tan\alpha = \frac{S}{R}$

S=ระยะทางสูงสุดมุมเอียงลง

=155.76 mm

R=รัศมีของเตียง

=1535 mm

α = 5.79410 องศา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

-หาองศาสูงสุดมุมเอียงขึ้น

เช่นเดียวกับการหาองศาสูงสุดมุมเอียงลง

$$S = 37.8875 \text{ mm}$$

$$\beta = 1.4139 \text{ องศา}$$

-หาจำนวนองศาที่เตียงเคลื่อนที่เมื่อเอนโคดเดอร์ให้พัลส์ออกมา 1 พัลส์

จำนวนพัลส์สูงสุดของมุมเอียงลง(เทียบกับหัวเตียง) = 14800

และจำนวนองศาที่คำนวณได้ = 5.79410 องศา

เพราะฉะนั้นจำนวนองศาต่อพัลส์ $= \frac{5.79410}{1480}$

$=0.0003915$ องศา

เปรียบเทียบกับจำนวนองศาต่อพัลส์ของเอนโคดเดอร์

$=\frac{360}{10000}$

$= 0.036$ องศา

ตารางบันทึกผลการทดลอง

<u>สั่งให้เตียงเคลื่อนที่ครั้งละ 0.5 องศา</u>

ค่าคงที่ของการทดลอง $k_p = 20$ $k_i = 15$ $k_d = 200$

จำนวนองศาที่สั่ง	ค่าที่อ่านได้(องศา)	จำนวนพัลส์ที่สั่ง	จำนวนพัลส์ที่อ่านได้	จำนวนพัลส์ที่ผิดพลาด
0.5	0.4952	1272	1265	7
0.9952	0.9936	2542	2538	4
1.4936	1.4904	3815	3807	8
1.9904	1.9868	5084	5075	9
2.4868	2.4836	6352	6344	8
2.9836	2.9797	7621	7611	10
3.4797	3.4761	10156	10146	10

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สั่งให้เคลื่อนที่เทียบกับโฮม

จำนวนองศาทางที่สั่ง	ค่าที่อ่านได้ (องศา)	จำนวนพัลส์ที่สั่ง	จำนวนพัลส์ที่อ่านได้	จำนวนพัลส์ที่ผิดพลาด
0.002	-	5	-	-
0.01	0.0055	21	14	7
0.05	0.0458	127	117	10

ค่าคงที่ของการทดลอง $k_p = 7$ $k_i = 40$ $k_d = 200$

จำนวนองศาที่สั่ง	ค่าที่อ่านได้ (องศา)	จำนวนพัลส์ที่สั่ง	จำนวนพัลส์ที่อ่านได้	จำนวนพัลส์ที่ผิดพลาด
0.5	0.4839	1272	1224	48
0.9839	0.9674	2542	2500	42
1.4674	1.4505	3815	3772	43
1.9505	1.9316	5084	5036	48
2.4316	2.4136	6352	6299	53
2.9136	2.8939	7621	7571	50

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สั่งให้เคลื่อนที่เทียบกับโฮม

จำนวนองศาทางที่สั่ง	ค่าที่อ่านได้ (องศา)	จำนวนพัลส์ที่สั่ง	จำนวนพัลส์ที่อ่านได้	จำนวนพัลส์ที่ผิดพลาด
0.005	-	12	-	-
0.01	-	25		
0.5	0.4784	1277	1227	50

รูปที่ 3.22 เตียงคนไข้เครื่องต้นแบบที่ใช้ทำการทดลอง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5
สรุปผลการวิจัยและวิจารณ์

เตียงคนไข้ชั้นบน

- ค่าคงที่ $k_p = 28$ $k_i = 7$ $k_d = 80$

จากผลการทดลองที่ให้เตียงคนไข้เคลื่อนที่ไปครั้งละ 10 มิลลิเมตร ซึ่งแสดงผลทั้งจำนวนพัลส์และระยะทางจริงที่เป็นมิลลิเมตร ค่าความผิดพลาดที่วัดได้จะอยู่ในช่วง 4-6 พัลส์ หรือเท่ากับ 0.08-0.12 มิลลิเมตร และค่าความละเอียดที่คำนวณได้มีค่า 0.02 มิลลิเมตร ซึ่งก็หมายความว่า ระยะทางน้อยที่สุดที่เตียงเคลื่อนที่ได้คือ 0.02 มิลลิเมตร แต่เนื่องจากมีความผิดพลาดเกิดขึ้นระยะทางที่น้อยที่สุดจะมีค่ามากขึ้นคือจะต้องมีค่าตั้งแต่ 0.12 มิลลิเมตรขึ้นไป เตียงจึงจะสามารถเคลื่อนที่ได้

- ค่าคงที่ $k_p = 20$ $k_i = 7$ $k_d = 80$

จากผลการเปลี่ยนค่าคงที่ PID ใหม่ค่าความผิดพลาดที่อ่านได้จะมีค่าเพิ่มขึ้นคือมีค่าอยู่ระหว่าง 18-23 พัลส์ หรือเท่ากับระยะทาง 0.36-0.46 มิลลิเมตร ซึ่งก็เป็นไปตามทฤษฎีที่ว่าการลดค่าคงที่ k_p และ k_i จะมีผลทำให้ค่าความผิดพลาดเพิ่มขึ้น ส่วนค่าความละเอียดก็จะมีค่าเพิ่มขึ้นด้วยเช่นกัน คือจะต้องให้เตียงเคลื่อนที่ตั้งแต่ระยะทาง 0.46 มิลลิเมตรขึ้นไป เตียงถึงจะเคลื่อนที่

เตียงคนไข้ชั้นกลาง

- ค่าคงที่ $k_p = 20$ $k_i = 15$ $k_d = 200$

จากผลการทดลองวัดค่าคงที่ต่าง ๆ สามารถหาค่าความละเอียดของการเคลื่อนที่ในแนวมุมเอียงได้เท่ากับ 0.0003915 องศา ซึ่งหมายความว่าเตียงสามารถเคลื่อนที่ได้อย่างน้อยที่สุดเท่ากับ 0.0003915 องศา แต่จากผลการทดลองการเคลื่อนที่ของเตียงมีความผิดพลาด อยู่ในช่วง 4-10 พัลส์ หรือเท่ากับ 0.0015-0.0039 องศา ซึ่งจะต้องสั่งให้มากกว่า 0.0039 องศา เตียงถึงจะเคลื่อนที่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- ค่าคงที่ $k_p = 7$ $k_i = 5$ $k_d = 200$

จากการทดลองเปลี่ยนค่าคงที่ PID ผลที่ได้ก็เป็นไปตามทฤษฎี คือถ้าลดค่า k_p และ k_i จะทำให้ค่าความผิดพลาดเพิ่มขึ้น ค่าผิดพลาดที่อ่านได้อยู่ในช่วง 42-50 พัลส์ หรือเท่ากับ 0.0164-0.0195 องศา

จากการเคลื่อนที่ของเตียงทั้งสองชั้น ค่าความผิดพลาดของการเคลื่อนที่จะมีค่าต่าง ๆ กัน แล้วแต่ค่าคงที่ PID ที่ใช้ การลดค่าความผิดพลาดสามารถทำได้โดยการเพิ่มค่าคงที่ k_p และ k_i

ปัญหาที่พบในการทดลอง

ในทางทฤษฎีแล้ว เมื่อสั่งให้เตียงเคลื่อนที่ไปยังตำแหน่งที่ต้องการ และเมื่อได้ระยะที่ต้องการแล้ว ซอฟแวร์จะทำการสั่งให้เตียงหยุดการเคลื่อนที่ทันที แต่เมื่อได้ทำการปฏิบัติจริงแล้วพบว่าเตียงจะไม่หยุดทันที แต่จะเคลื่อนที่ต่อไปอีกเล็กน้อยตามแรงเฉื่อยของมอเตอร์ ซึ่งจะเป็นความผิดพลาดของระบบ และถ้าความเร็วของเตียงมีมากขึ้นเท่าใด ค่าความผิดพลาดก็จะยิ่งมีเพิ่มมากขึ้นตามไปด้วย ยิ่งถ้าความเร็วของเตียงมีค่ามากๆ เซนเซอร์จะทำการตรวจเช็คตำแหน่งเพื่อการอินเทอร์รัพท์ให้เตียงหยุดไม่ทัน

วิธีแก้ไข

จะต้องกำหนดความเร็วของเตียงไม่ให้เคลื่อนที่เร็วจนเกินไปจนเซนเซอร์ทำงานไม่ทัน และอีกวิธีหนึ่งก็คือ การแก้ไขโดยซอฟแวร์ ซึ่งทำได้โดยการเขียนโปรแกรมสั่งให้เตียงเคลื่อนที่ย้อนกลับไปยังตำแหน่งของตัวเซนเซอร์นั้นๆ เพื่อให้มีการอินเทอร์รัพท์ตามต้องการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ก
โปรแกรมการทำงาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/*-------------------------------------------*/
/  This program is used to control patient table /
/   for horizontal direction                    /
/*------------------------------------------- */


#include <dos.h>
#include <math.h>
#include <stdio.h>
#include <conio.h>
#include "7340drvc.h"
#include "motor.h"
/* unction prototypes */

void far init_7340(void);
void far init_pics(void);
void far wrcmd(int,char);
void far wrword(int,unsigned);
void far wr2words(int,long);
int far rdstat(int);
long far rd2words(int);
void Stop_table(int axis);

long read_realpos();
void menu(void);
void init_filters(int,unsigned,
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                    unsigned,unsigned,
                    unsigned,unsigned);
void initiallization();
void interrupt_contr();
void home();
void loadtraj();
void trajectory_contr();

int Motor_Command;
int Move_Result;
int MoveStateAxis1;
int Motor_Position;
int Motor_Dir;
int Top_Counter;
int Bottom_Counter;
int Home_Counter;
int Error_Code;
int Home_Limit;
unsigned rdsig;

int i;
char ch;
int  command,st;
long velo,acce,vel,Rvel,Target,current,rrvel;
unsigned  prop,integ,deriv,integlim;
int axis;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
int rvel;
float   encoder_line,
        p_target,targ_conv,
        current_rev;

char *table_pos[] =
            {
             "UNKNOW POSITION",
            "OUT OF RANGE",
            "HOME",          "BOTTOM LIMIT",
            "TOP LIMIT",     "BOTTOM HOME",
            "TOP HOME",      "BOTTOM OFF",
            "TOP OFF",       "BOTTOM TOP",
            "BOTTOM OFF_ BOTTOM LIMIT",
            "TOP OFF_TOP LIMIT"
   };

char *move_state[]=
            {
             "STOP",
            " STOP_HOME",
            "STOP_BOTTOM",
            "STOP_TOP",
            "FIND_HOME",
            "FIND_BOTTOM_LIMIT",
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
            "FIND_TOP_LIMIT",
            "NORMAL_MOVE"
    };

char *error[] =
            {
                "SUCCESSFUL",
                "PROGRAM_ERROR",
                "MOVE_IN_OPERATION",
                "BOTTOM_ERROR",
                "TOP_ERROR",
                "HOME_ERROR",
                "NORMAL_MOVE_ERROR",
                "PROGRAM_ERROR"
 };
 char *move_dir[]= { "MOTOR_STOP",
                     "MOTOR_BOTTOM_MOVE",
                     "MOTOR_TOP_MOVE",
                     "UNKNOWN_PREVIOUS_DIR",
                    "PREVIOUS_BOTTOM_DIR",
                    "PREVIOUS_TOP_DIR"
   };
 char *move_command[]= { "GO_DESIRED_POS",
                         "GO_HOME",
                        "GO_BOTTOM_LIMIT",
                        "GO_TOP_LIMIT",
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                    "INITIAL"
                    };
void init_filters(int axis,
    unsigned filt_cw,
    unsigned p,
    unsigned i,
    unsigned d,
    unsigned il)
{
    wrcmd(axis,LFIL);    /* init filters    */
    wrword(axis,filt_cw);
    wrword(axis,p);
    wrword(axis,i);
    wrword(axis,d);
    wrword(axis,il);
    wrcmd(axis,UDF);
}

void initiallization(int axis )
{
    wrcmd(axis,MRESET); /* init lm628     */
    wrcmd(axis,PORT12);
    wrcmd(axis,DFH);
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
void interrupt_contr(int axis)
{
   wrcmd(axis,LPES);
   wrword(axis,0x7fff);
   wrcmd(axis,MSK1);
   wrword(axis,0);
   outportb(CP0,0x00);
   outportb(CP1,0x80);
}

void trajectory_contr(int axis,
        unsigned tr_cw,
        long acce,
        long vel,
        long pos)
{
   wrcmd(axis,LTRJ);
   wrword(axis,tr_cw);
   wr2words(axis,acce);
   wr2words(axis,vel);
   wr2words(axis,pos);
   wrcmd(axis,STT);  /*start move*/
}

long Target_conv(float p_target,float encoder_line)
{
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    long current_pos;
    float R;
    current_pos = read_realpos(axis);
    R = encoder_line * 4.0; /*system resolution */
    Target = (long)((p_target * R)/(M_PI*DIM)+current_pos);
    return((long)Target);
}
float targ_calulate(long targ_cal)
{
  float R;
  float  target_cal;
  R = encoder_line*4;
  target_cal = (M_PI*DIM/R)*(targ_cal);
  return((float)target_cal);
}


long read_realpos(int axis)
{
   wrcmd(axis,RDRP);
    current =(long)(rd2words(axis));
    return((long)(current));
}

 /* start   stop motor*/
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
void Stop_table(int axis)
{
   outportb(OUTPUT1,0x40);
   trajectory_contr(axis,0x0200,0L,0L,0L);
   wrcmd(axis,DFH);
   wrcmd(axis,STT);
}
/* end stop motor*/
void  show_state()
{
printf("\nMotor Position:%s",table_pos[Motor_Position]);
printf("\nMove Result:%s",   error[Move_Result]);
printf("\nMove Direction:%s",move_dir[Motor_Dir]);
printf("\nMove State:%s", move_state[MoveStateAxis1]);

}


void go_home()
{
    long posi;
     clrscr();
   printf("<<PID CONTROLLER PARAMETER:>>\n");
   printf("kp:%4u",prop);
   printf("ki:%4u",integ);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  printf("kd:%4u",deriv);
  printf("il:%4u\n",integlim);

    MoveStateAxis1 = FIND_HOME;


 if(inportb(INPUT1) == 0xF8) /* inport1=80 BOTTOM_limit -*/
 {
    Move_Result = MOVE_IN_OPERATION;
    Motor_Position = BOTTOM_LIMIT;
    Motor_Dir  =  MOTOR_TOP_MOVE;
    show_state();
    trajectory_contr(axis,0x002a,371,500001,250001);
    /* ------ move to left --------*/
 } else if(inportb(INPUT0)==0x80) /*input0 =F8 TOP_LIMIT  */
{
   Move_Result = MOVE_IN_OPERATION;
   Motor_Position = TOP_LIMIT;
   Motor_Dir  =MOTOR_BOTTOM_MOVE;
   show_state();
   trajectory_contr(axis,0x002a,371,500001,-250001);
   /*------- move to right -----------*/
     } else if(inportb(INPUT0)== 0x08) /*input0 = 08 HOME  */
{
   printf("this pos:home\n");
   MoveStateAxis1 = STOP_HOME;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    Move_Result = SUCCESSFUL;
    Motor_Position = HOME;
    Motor_Dir  =  MOTOR_STOP;
    show_state();
        } else
        {
printf(" RANDOM:\n");
            Move_Result = MOVE_IN_OPERATION;
            Motor_Dir  =  MOTOR_BOTTOM_MOVE ;
            show_state();
       trajectory_contr(axis,0x002a,37L,50000L,-250001);
        }

  do {
      wrcmd(axis,RDRP);
      current =(long)(rd2words(axis));
      printf("\rFinding Home.. !  Position:%8ld  ", current);

   }
   while((MoveStateAxis1 !=STOP_HOME)&&(MoveStateAxis1 !=STOP));
   printf("\nMotor Stop Now\n");
   trajectory_contr(axis,0x0200,0L,0L,0L);
   wrcmd(axis,DFH); /* find zero postion*/
   wrcmd(axis,STT);
   delay(100);
   trajectory_contr(axis,0x002a,76L,70000L,0L);
   show_state();
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
}


void go_bottom_limit(int axis )
{
        clrscr();
        Motor_Command =  GO_BOTTOM_LIMIT;
        MoveStateAxis1 = FIND_BOTTOM_LIMIT;
         read_realpos(axis);
        /*motor position home,top,cw*/

    switch(Motor_Position)
       {
       case BOTTOM_LIMIT:
    printf("\nmotorppos:BOTTM_LIMIT\n");
    Motor_Dir   =  MOTOR_STOP;
    Move_Result=  SUCCESSFUL;
    MoveStateAxis1 = STOP_BOTTOM;
    break;

       case TOP_LIMIT:
   printf("\nmotorppos:TOP_LIMIT\n");
   Motor_Dir = MOTOR_BOTTOM_MOVE;
   /*Motor_Position = LEFT_HOME;*/
   trajectory_contr(axis,0x002a,0x371,500001,-25000L);
   break;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
case HOME:
   printf("\nmotorpos:HOME\n");
   Motor_Dir = MOTOR_BOTTOM_MOVE;
   /*Motor_Position = RIGHT_HOME;*/
   trajectory_contr(axis,0x002a,0x371,500001,-25000L);
   break;
       case TOP_HOME:
 if(current>0)
   {
    printf("\nmotorpos:TOP_HOME\n");
    Motor_Dir =  MOTOR_BOTTOM_MOVE;
    Motor_Position =  TOP_HOME;
      trajectory_contr(axis,0x002a,0x371,500001,-25000L);
   } else {  Move_Result=PROGRAM_ERROR;
   show_state();}
    break;
       case   BOTTOM_HOME:

     if(current<0)
    {
      printf("\nmotorpos:BOTTOM_HOME\n");
      Motor_Dir =  MOTOR_BOTTOM_MOVE;
      Motor_Position =  BOTTOM_HOME;
       trajectory_contr(axis,0x002a,371,500001,-25000L);
     }else {Move_Result = PROGRAM_ERROR;
    show_state();}
    break;
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    do{
   read_realpos(axis);
   printf("\rFINDING BOTTOM_LIMIT%8ld    " ,current);
      }while((MoveStateAxis1 != STOP);
    printf("\nMotor Stop Now\n");
    trajectory_contr(axis,0x0200,0L,0L,0L);
    delay(100);
    trajectory_contr(axis,0x002a,371,50000L,current);
    show_state();
}


void go_top_limit(int axis)
  {
      clrscr();
      Motor_Command =  GO_TOP_LIMIT;
      MoveStateAxis1 = FIND_TOP_LIMIT;
      read_realpos(axis);
      /* CASE position is  home,cw,top*/
    switch(Motor_Position)
      {
  case BOTTOM_LIMIT:
      Move_Result=MOVE_IN_OPERATION;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        Motor_Dir =  MOTOR_TOP_MOVE;
         trajectory_contr(axis,0x002a,0x371,500001,25000L);
        break;
   case  TOP_LIMIT:
       Move_Result=SUCCESSFUL;
       Motor_Dir=MOTOR_STOP;
       MoveStateAxis1 = STOP_TOP;
       break;
   case HOME:
        Move_Result=MOVE_IN_OPERATION;
        Motor_Dir =  MOTOR_TOP_MOVE;
        trajectory_contr(axis,0x002a,371,500001,25000L);
        break;
case TOP_HOME:
 /* case position is anywhere*/
    if(current>0)
{
Move_Result=MOVE_IN_OPERATION;
Motor_Dir =  MOTOR_TOP_MOVE;
     trajectory_contr(axis,0x002a,371,500001,25000L);
}else Move_Result = PROGRAM_ERROR;
   show_state();
   break;
 case BOTTOM_HOME:
       if(current<0)
        {
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
 Move_Result=MOVE_IN_OPERATION;
 Motor_Dir =  MOTOR_TOP_MOVE;
       trajectory_contr(axis,0x002a,37l,50000l,25000L);
}else Move_Result = PROGRAM_ERROR;
       show_state();
       break;
      }
       do{
read_realpos(axis);
printf("\rFINDING TOP_LIMIT%8ld   " ,current);
         } while((MoveStateAxis1 !=STOP_TOP)&&( MoveStateAxis1 !=STOP)

    printf("\nMotor Stop Now\n");
    trajectory_contr(axis,0x0200,0L,0L,0L);
    delay(100);
    printf("\ntop limit:%8ld\n",current);
    trajectory_contr(axis,0x002a,37L,50000L,current);
    printf("\nTOPCONT:%d\n" ,Top_Counter);
    show_state();

}

void movement_init(int axis)
{
   /*SET AIV*/
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    clrscr();
    Top_Counter = 0;
    Bottom_Counter  = 0;
    Home_Counter = 0;
    Motor_Position = UNKNOWN_POS;
    MoveStateAxis1 = STOP;
    Motor_Dir = MOTOR_STOP;
    Motor_Command =INITIAL;

    /*filter parameter*/
    /* first  must load filter before move*/

    prop=10u;
    integ=5u;
    deriv=80u;
    integlim=5u;

    init_filters(axis,0x000f,prop,integ,deriv,integlim);
    go_home(axis);
}
 int select_axis()
 {

  do{
      printf("input axis <0-3> :");scanf("%d",&axis);
    } while(axis<0 || axis>3);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    return((int)axis);
 }

void  desired_move(int axis)
{
     clrscr();
     MoveStateAxis1 = NORMAL_MOVE;
     wrcmd(axis,RDRP);
     current =(long)(rd2words(axis));
 do{
   do{
      printf("\nTOP MAX LIMIT:%f  BOTTOM MAX LIMIT:%f",
              TOP_LIMIT_CONV,BOTTOM_LIMIT_CONV);
      printf("\nTOP DIRECTION:POSITIVE  BOTTOM DIRECTION:NEGATIVE\n")
      printf("\nTARGET:  ");
      scanf("%f",&p_target);
      Target=(long)Target_conv(p_target,encoder_line);
      printf("\nTarget Conv:%ld",Target);

 /*case motor is home,cw,top*/
 if( (Target >=BOTTOM_LIMIT_MAX)&&( Target<= TOP_LIMIT_MAX))
   { if(Target > 0)
     {
       switch(Motor_Position)
       {
    case HOME:
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    printf("\nCurrent Position is Home\n");
    Move_Result=MOVE_IN_OPERATION;
    show_state();
    Motor_Dir = MOTOR_TOP_MOVE;
    Move_Result=SUCCESSFUL;
    Motor_Position = TOP_HOME;
    trajectory_contr(axis,0x002a,37L,50000L,Target);
     break;

      case  TOP_LIMIT:

   if(Target==TOP_LIMIT_MAX)
      {
        printf("\nCurrent Position is TOP_LIMIT\n");
        show_state();
        Motor_Dir = MOTOR_STOP;
        Move_Result = SUCCESSFUL;
        Motor_Position = TOP_LIMIT;
        MoveStateAxis1 = STOP_TOP;

       }else{
  printf("\nCurrent Position is TOP_HOME\n");
  Motor_Dir = BOTTOM_LIMIT;
  Move_Result = MOVE_IN_OPERATION;
  show_state();
  Move_Result = SUCCESSFUL;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Motor_Position = TOP_HOME;
trajectory_contr(axis,0x002a,371,500001,Target);
}

break;
    case BOTTOM_LIMIT:
  printf("\nCurrent Position is BOTTOM_LIMIT\n");
  Motor_Dir = MOTOR_TOP_MOVE;
  Move_Result=MOVE_IN_OPERATION;
  Motor_Position =TOP_HOME;
  trajectory_contr(axis,0x002a,371,500001,Target);
  break;
    case  BOTTOM_HOME:
  Motor_Dir =   MOTOR_TOP_MOVE;
  Move_Result = MOVE_IN_OPERATION;
  show_state();
  Move_Result = SUCCESSFUL;
  Motor_Position =TOP_HOME;
  trajectory_contr(axis,0x002a,371,500001,Target);
  break;
     case TOP_HOME:
     if(Target>current)Motor_Dir =   MOTOR_TOP_MOVE;
       else  Motor_Dir =   MOTOR_BOTTOM_MOVE;
  Move_Result = MOVE_IN_OPERATION;
  show_state();
  Move_Result = SUCCESSFUL;
  Motor_Position =TOP_HOME;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น มีการห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
trajectory_contr(axis,0x002a,371,50000l,Target);
break;
 }

    }else if(Target<0)
       {
   switch(Motor_Position)
       {
        case HOME:
           printf("\nCurrent Position is HOME\n");
           Move_Result=MOVE_IN_OPERATION;
           show_state();
           Move_Result=SUCCESSFUL;
          Motor_Dir = MOTOR_BOTTOM_MOVE;
           Motor_Position = BOTTOM_HOME;
           trajectory_contr(axis,0x002a,371,50000l,Target);
            break;

        case TOP_LIMIT:
            printf("\nCurrent Position is TOP_LIMIT\n");
            Motor_Dir = MOTOR_BOTTOM_MOVE;
           Motor_Position = BOTTOM_HOME;
            trajectory_contr(axis,0x002a,371,50000l,Target);
            break;
        case BOTTOM_LIMIT:
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับใช้เพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
if(Target==BOTTOM_LIMIT_MAX)
                {
                 printf("\nCurrent Position is BOTTOM_LIMIT\n");
                 Motor_Dir = MOTOR_STOP;
                 Move_Result=SUCCESSFUL;
                 Motor_Position =BOTTOM_LIMIT;
                 MoveStateAxis1=STOP_BOTTOM;
               }else{
                 printf("\nCurrent Position is BOTTOM_LIMIT\n");
                 Motor_Dir = MOTOR_TOP_MOVE;
                 Move_Result=MOVE_IN_OPERATION;
                 show_state();
                 Move_Result=SUCCESSFUL;
                 Motor_Position =BOTTOM_HOME;
                 trajectory_contr(axis,0x002a,371,500001,Target);

}
 break;
 case TOP_HOME:
                  printf("POS:BETWEEN TOP&HOME\n");
                  Motor_Dir =  MOTOR_BOTTOM_MOVE;
                  Move_Result = MOVE_IN_OPERATION;
                  show_state();
                  Move_Result=SUCCESSFUL;
                  Motor_Position =BOTTOM_HOME;
                 trajectory_cotr(axis,0x002a,371,500001,Target);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                    break;
                case BOTTOM_HOME:
                    if(Target>current)Motor_Dir =  MOTOR_TOP_MOVE;
                     else
                     Motor_Dir =  MOTOR_BOTTOM_MOVE;
Move_Result = MOVE_IN_OPERATION;
show_state();
Move_Result=SUCCESSFUL;
Motor_Position =BOTTOM_HOME;
trajectory_contr(axis,0x002a,371,500001,Target);
break;
   }
         }
      }else
    {

      printf("\nPOSITION IS EXCEEDED MAXIMUM_LIMIT\n");
      printf("LOAD NEW POSITION\n");

    }
     }while((Target>TOP_LIMIT_MAX)||(Target< BOTTOM_LIMIT_MAX));
                    gotoxy(100,200);
  printf("\nDESIRED  POSITION:%8ld\n",Target);
 do {
   wrcmd(axis,RDRP);
   current =(long)(rd2words(axis));
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    current_rev = targ_calulate(current);
    st = rdstatus(axis);
            printf("\rMoving to Target POSITION:%8ld  %8.3f STATUS:%02
                    current,
                    current_rev,st);

  }while( !kbhit());
  getch();
  show_state();
  printf(" \nARE YOU TO CONTINUE Y/N:");
    ch = getch();
}while( (ch=='Y') || (ch=='y'));


  }


void menu()
{
   printf("\n----------------------------------------\n");
   printf("\nTABLE CONTROL BY COMPUTER  \n");
   printf("----------------------------------------\n");
   printf("SELECT FUNCTION:\n");
   printf(" N=  DESIRED MOVE\n");
   printf(" H=  GO HOME\n");
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    printf(" C=  GO BOTTOM LIMIT\n");
    printf(" E= GO TOP LIMIT\n");
    printf("quit=ESC\n ");
}

 float  colect_encode()
 {
    printf("Encoder = ");
    scanf("%f",&encoder_line);
    encoder_line= (float)(encoder_line+11.986268);
   return((float)encoder_line);

   }


void main()
{
    init_7340();
    init_plcs();

    axis= (int)select_axis();
    initiallization(axis);
    interrupt_contr(axis);
     init_filters(axis,0x000f,28l,7l,80l,7l);
     encoder_line=1200.0;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    movement_init(axis);

  do{
     menu();
     command = toupper(getch());
     switch(command)
     {
case 'N':desired_move(axis);  break;
case 'H':go_home(axis);       break;
case 'C':go_bottom_limit(axis);  break;
case 'E':go_top_limit(axis); break;
     }
  }while(command !=ESC);}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/* 7340drv.c: Interface drivers for 7340

/*************************************************************
/*
/*    7340  INTERFACE DRIVERS
/*
/*          COPYRIGHT PRO-LOG CORPORATION, C 1990
/*
/*          Revision X1 5/15/90 PE
/*
/*************************************************************

#include <stdio.h>
#include <dos.h>
#include "motor.h"
#include "7340drvc.h"

#define AXIS_CMD  (axis * 2 + LM6280_CMD)
#define AXIS_DATA (axis * 2 + LM6280_DATA)

extern int MoveStateAxis1;
extern int Bottom_Counter;
extern int Home_Counter;
extern int Top_Counter;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
extern int Move_Result;
extern int Motor_Dir;
extern int Motor_Position;
extern int axis;
extern long posi;
/*****************************************************************
/*
/*    LM628 HOUSEKEEPING ROUTINE:
/*      Call this routine before attempting to use the 7340.  Interru
/*      initialized and cleared.  I/O ports are initialized.  Is esse
/*      to reset the 82C59A-2 interrupt controllers after reset, sinc
/*      initialization state is undefined.  Failure to do so may caus
/*      wanted interrupts.init_7340 takes care of this detail.
/*
/*****************************************************************
init_7340()
{
disable();
/* I1_8259 initialization sequence */
/*    IR0-3 edge triggered  lm628 interrupts
/*    IR4   cascade interrupt for I2_8259 */
/*    IR5   cascade interrupt for I3_8259 */
/*    IR0-7 masked */
outportb(I1_8259_0, ICW1 | ICW1_ETM | ICW1_CASCADE | ICW1_IC4);
outportb(I1_8259_1, ICW2 | VECTOR_BASE + 0);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
outportb(I1_8259_1, ICW3 | ICW3_S4 | ICW3_S5);
outportb(I1_8259_1, ICW4 | ICW4_8086);
outportb(I1_8259_1, OCW1 | OCW1_M0 | OCW1_M1 | OCW1_M2 | OCW1_M3 |
     OCW1_M4 | OCW1_M5 | OCW1_M6 | OCW1_M7 );

/* I2_8259 initialization sequence */
/*    IR0-3 edge triggered, CCW_LIMITS, AX1-4 */
/*    IR4-7 edge triggered, CW_LIMITS, AX1-4 */
/*    IR0-7 masked */
outportb(I2_8259_0, ICW1 | ICW1_ETM | ICW1_CASCADE | ICW1_IC4);
outportb(I2_8259_1, ICW2 | VECTOR_BASE + 8);
outportb(I2_8259_1, ICW3 | ICW3_ID4);
outportb(I2_8259_1, ICW4 | ICW4_8086);
outportb(I2_8259_1, OCW1 | OCW1_M0 | OCW1_M1 | OCW1_M2 | OCW1_M3 |
     OCW1_M4 | OCW1_M5 | OCW1_M6 | OCW1_M7 );

/* I3_8259 initialization sequence */
/*    IR0-3 edge triggered, HOME_LIM, AX1-4 */
/*    IR0-7 masked */
outportb(I3_8259_0, ICW1 | ICW1_ETM | ICW1_CASCADE | ICW1_IC4);
outportb(I3_8259_1, ICW2 | VECTOR_BASE + 16);
outportb(I3_8259_1, ICW3 | ICW3_ID5);
outportb(I3_8259_1, ICW4 | ICW4_8086);
outportb(I3_8259_1, OCW1 | OCW1_M0 | OCW1_M1 | OCW1_M2 | OCW1_M3 |
     OCW1_M4 | OCW1_M5 | OCW1_M6 | OCW1_M7 );
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
enable();

/* initialize output ports */
outportb(OUTPUT0,0);
outportb(OUTPUT1,0);
outportb(CP0,0);
outportb(CP1,0);
return 0;
}

/*****************************************************************
/*
/*    LM628 I/O UTILITIES
/*
/*****************************************************************

/* determine if lm628 is busy */
void waitbusy(int axis)
{
while(inportb(AXIS_CMD) & BUSYBIT)
continue;
}

/* write command byte to lm628 */
void far wrcmd(int axis, char code)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
{
waitbusy(axis);
outportb(AXIS_CMD,code);
}

/* write data byte to lm628 */
void far wrbyte(int axis, char byte)
{
outportb(AXIS_DATA,byte);
}

/* write data word (2 bytes) to lm628 */
void far wrword(int axis, unsigned word)
{
waitbusy(axis);
wrbyte(axis, (word & WUPPER) >> 8);
wrbyte(axis, word & WLOWER);
}

/* write double data word (4 bytes) to lm628 */
void far wr2words(int axis, long words)
{
wrword(axis, (unsigned)((words & LUPPER) >> 16));
wrword(axis, (unsigned)(words & LLOWER));
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/* read status from lm628 */
int far rdstatus(int axis)
{
return(inportb(AXIS_CMD));
}

/* read data byte from lm628 */
char far rdbyte(int axis)
{
return(inportb(AXIS_DATA));
}

/* read data word (2 bytes) from lm628 */
unsigned far rdword(int axis)
{
unsigned high,low;

waitbusy(axis);
high = (rdbyte(axis) << 8) & WUPPER;
low = rdbyte(axis) & WLOWER;
return(high | low);
}

/* read double data word (4 bytes) from lm628 */
long far rd2words(int axis)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/*Isigsreg (signals register upon interrupt.          */
/* A word of caution: be sure to disable interrupts   */
/* before *//* calling the LM628 driver routines above,*/
/* otherwise interupts may destroys LM628 data transfers*/
/* which are partially completed.                     */

int Istatus1, Isigsreg1, Intrmasks1;
int Istatus2, Isigsreg2, Intrmasks2;
int Istatus3, Isigsreg3, Intrmasks3;
int Istatus4, Isigsreg4, Intrmasks4;

/* Interrupt specific control registers.*/
/*These registers may be useful in selectively */
/*masking interrupts.  This code uses these*/
/*varables local to  init_pics and dis_pics. */

int Msk_8259_1, Msk_8259_2, Msk_8259_3;

/* interrupt service routine for axis 1 */
void interrupt isr_axis_1(void)
{
Istatus1 = rdstatus(AXIS1);/* read and store status */
wrcmd(AXIS1, RDSIGS);/* read sigs register */
Isigsreg1 = rdword(AXIS1);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
{
long top,bottom;

top = ((long)rdword(axis) << 16) & LUPPER;
bottom = (long)rdword(axis) & LLOWER;
bottom = top | bottom;
return(bottom);
}

/*****************************************************************
/*
/*    7340 INTERRUPT SYSTEM:
/*        INITIALIZATION AND SERVICE ROUTINES
/*
/*****************************************************************
/* The following variables make are used by the 7340 */
/*to pass information  between the interrupt processes*/
/* and your background routines.  By initializing the
/*Intrmasks variable to the desired mask, the interrupt */
/* routines for the LM628's will only generate          */
/*desired interrupts.  Background routines can          */
/* monitor Interrupted, and when TRUE, may examine      */
/* Istatus (status register upon interrupt) or          */
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
wrcmd(AXIS1, MSKI);/* mask all interrupts */
wrword(AXIS1, Intrmasks1);
wrcmd(AXIS1, RSTI);/* reset all interrupts */
wrword(AXIS1, 0);
outportb(I1_8259_0,OCW2 + OCW2_NS_EOI);/* signal EOI to 8259 */
}


/* interrupt service routine for axis 2 */
void interrupt isr_axis_2(void)
{
Istatus2 = rdstatus(AXIS2);/* read and store status */
wrcmd(AXIS2, RDSIGS);/* read sigs register */
Isigsreg2 = rdword(AXIS2);
wrcmd(AXIS2, MSKI);/* mask all interrupts */
wrword(AXIS2, Intrmasks2);
wrcmd(AXIS2, RSTI);/* reset all interrupts */
wrword(AXIS2, 0);
outportb(I1_8259_0,OCW2 + OCW2_NS_EOI);/* signal EOI to 8259 */
}

/* interrupt service routine for axis 3 */
void interrupt isr_axis_3(void)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
{
Istatus3 = rdstatus(AXIS3);/* read and store status */
wrcmd(AXIS3, RDSIGS);/* read sigs register */
Isigsreg3 = rdword(AXIS3);
wrcmd(AXIS3, MSKI);/* mask all interrupts */
wrword(AXIS3,  Intrmasks3);
wrcmd(AXIS3, RSTI);/* reset all interrupts */
wrword(AXIS3, 0);
outportb(I1_8259_0,OCW2 + OCW2_NS_EOI);/* signal EOI to 8259 */
}


/* interrupt service routine for axis 4 */
void interrupt isr_axis_4(void)
{
Istatus4 = rdstatus(AXIS4);/* read and store status */
wrcmd(AXIS4, RDSIGS);/* read sigs register */
Isigsreg4 = rdword(AXIS4);
wrcmd(AXIS4, MSKI);/* mask all interrupts */
wrword(AXIS4, Intrmasks4);
wrcmd(AXIS4, RSTI);/* reset all interrupts */
wrword(AXIS4, 0);
outportb(I1_8259_0,OCW2 + OCW2_NS_EOI);/* signal EOI to 8259 */
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
}

/* interrupt service routine for U41 cw and ccw limits */
void interrupt isr_TOP1(void) /*cw limit switch  */
{
    int in_serv_reg;/* determine interrupt status */
    outportb(I2_8259_0, OCW3 | OCW3_RD_IS);

    in_serv_reg = inportb(I2_8259_0);
    if(in_serv_reg)
    {
/* interrupt occured, place user code here */
/* outportb(OUTPUT1,0x40)   ;*/
/* outportb(OUTPUT1,0x0);     */


   switch(MoveStateAxis1)
      {
case FIND_HOME:          /*HIT ONE TIME ERROR OCCURRED */
    Top_Counter++;
     if(Top_Counter==1)
    Move_Result=HOME_ERROR;
    MoveStateAxis1=STOP;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
      break;
         case FIND_BOTTOM_LIMIT:   /*HIT ONE TIME ERROR OCCURRED */
      Top_Counter++;
        if(Top_Counter==1)
        {
Move_Result=BOTTOM_ERROR;
MoveStateAxis1=STOP;
        }
      break;
        case FIND_TOP_LIMIT:
                        read_realpos(axis);
        Move_Result=SUCCESSFUL;
        Motor_Dir =  MOTOR_STOP;
        Motor_Position=TOP_LIMIT;
        Top_Counter = 0;
        Bottom_Counter  = 0;
        Home_Counter =0;
        MoveStateAxis1=STOP_TOP;


   break;
        }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    }
/* acknowledge interrupt occured */
outportb(I1_8259_0,OCW2 + OCW2_NS_EOI);/* signal EOI to 8259 */
outportb(I2_8259_0,OCW2 + OCW2_NS_EOI);/* signal EOI to 8259 */
}

void interrupt isr_BOTTOM1(void) /*cw limit switch  */
{
    int in_serv_reg;/* determine interrupt status */
    outportb(I2_8259_0, OCW3 | OCW3_RD_IS);

    in_serv_reg = inportb(I2_8259_0);
    if(in_serv_reg)

    {
/* interrupt occured, place user code here */
/* outportb(OUTPUT1,0x40)    ;*/
/* outportb(OUTPUT1,0x0);     */
        switch(MoveStateAxis1)
    {
case FIND_HOME: /*hit not more than one time*/
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    Bottom_Counter++;
        if(Bottom_Counter>2)
    {
Move_Result=HOME_ERROR;
    }else
     if(Bottom_Counter<=2 )
    {
      Motor_Position = BOTTOM_LIMIT;
      Motor_Dir =  MOTOR_TOP_MOVE;

     trajectory_contr(axis,0x0200,0L,0L,0L);
     delay(1000);
     trajectory_contr(axis,0x002a,371,50000L,-25000L);
    }
     break;
      case FIND_BOTTOM_LIMIT:
   read_realpos(axis);
                    MoveStateAxis1=STOP_BOTTOM;
   Move_Result=SUCCESSFUL;
   Motor_Dir =  MOTOR_STOP;
   Motor_Position=BOTTOM_LIMIT;
   Top_Counter = 0;
   Bottom_Counter  = 0;
   Home_Counter =0;
   break;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
     case FIND_TOP_LIMIT:
    Bottom_Counter++;
        if(Bottom_Counter==1)
        {
Move_Result=TOP_ERROR;
MoveStateAxis1=STOP;
        }
   break;
      }
    }

/* acknowledge interrupt occured */
outportb(I1_8259_0,OCW2 + OCW2_NS_EOI);/* signal EOI to 8259 */
outportb(I2_8259_0,OCW2 + OCW2_NS_EOI);/* signal EOI to 8259 */
}



void interrupt isr_u41(void) /*cw limit switch  */
{
     int in_serv_reg;/* determine interrupt status */
     outportb(I2_8259_0, OCW3 | OCW3_RD_IS);

     in_serv_reg = inportb(I2_8259_0);
     if(in_serv_reg)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    {
/* interrupt occured, place user code here */
/* outportb(OUTPUT1,0x40)    ;*/
/* outportb(OUTPUT1,0x0);      */
    }

/* acknowledge interrupt occured */
outportb(I1_8259_0,OCW2 + OCW2_NS_EOI);/* signal EOI to 8259 */
outportb(I2_8259_0,OCW2 + OCW2_NS_EOI);/* signal EOI to 8259 */
}

/* interrupt service routine for U42 home limits */
void interrupt isr_home1(void)  /*HOME SWITCH ACTIVE*/

{
int in_serv_reg;
/* determine interrupt status */
outportb(I3_8259_0, OCW3 | OCW3_RD_IS);
in_serv_reg = inportb(I3_8259_0);
if (in_serv_reg)
{
/* interrupt occured, place user code here */
switch(MoveStateAxis1)
  {
    case  FIND_HOME:
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    MoveStateAxis1 = STOP_HOME;
    Motor_Position = HOME;
     Top_Counter = 0x00;
     Bottom_Counter  = 0x00;
     Move_Result = SUCCESSFUL;
     Motor_Dir = MOTOR_STOP;
     break;
    case  FIND_BOTTOM_LIMIT:
Home_Counter++;
 if((Motor_Position == BOTTOM_HOME) && (Home_Counter==1))
      {
         Move_Result = BOTTOM_ERROR;
MoveStateAxis1=STOP; /*ERROR OCCUR MOTOR STOP NOW*/
      }else
       if((Motor_Position==TOP_LIMIT)||(Motor_Position==TOP_HOME))
                                  if(Home_Counter>2)
   {
    Move_Result=BOTTOM_ERROR;
    MoveStateAxis1=STOP;
    }
     break;
     case FIND_TOP_LIMIT:
 Home_Counter++;
 if((Motor_Position==BOTTOM_LIMIT) ||(Motor_Position== BOTTOM_HOME)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
      {
      if(Home_Counter>1)            /*current position  is negative*/
        { Move_Result = TOP_ERROR;
 MoveStateAxis1=STOP; } /*ERROR OCCUR MOTOR STOP NOW*/
      }else{
       if(Motor_Position==TOP_HOME)  /*current position is negative*/
 if(Home_Counter==1)
    { Move_Result=TOP_ERROR;
      MoveStateAxis1=STOP;}
    }
      break;
   }
}
/* acknowledge interrupt occured */
outportb(I1_8259_0,OCW2 + OCW2_NS_EOI);/* signal EOI to 8259 */
outportb(I3_8259_0,OCW2 + OCW2_NS_EOI);/* signal EOI to 8259 */
}
  void interrupt isr_home1(void)/*home switch active*/
{
int in_serv_reg;
/* determine interrupt status */
outportb(I2_8259_0, OCW3 | OCW3_RD_IS);
in_serv_reg = inportb(I2_8259_0);
   if(in_serv_reg)
        {
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/* interrupt occured, place user code here */
        /* acknowledge interrupt occured */

      }
outportb(I1_8259_0,OCW2 + OCW2_NS_EOI);/* signal EOI to 8259 */
outportb(I3_8259_0,OCW2 + OCW2_NS_EOI);/* signal EOI to 8259 */
 }
/* spurious interrupt service routine for U43 */
/* this interrupt may occur if any CW or CCW limits become acitve */
/*   and are removed before the CPU has enough time to generate an */
/*   interrupt acknowledge sequence */
void interrupt spurious_u43(void)

{
/* acknowledge interrupt occured */
outportb(I1_8259_0,OCW2 + OCW2_NS_EOI);/* signal EOI to 8259 */
}

/* spurious interrupt service routine for U42 */
/* this interrupt may occur if any HOME limits become acitve */
/*   and are removed before the CPU has enough time to generate an */
/*   interrupt acknowledge sequence */
void interrupt spurious_u42(void)

{
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/* acknowledge interrupt occured */
outportb(I3_8259_0,OCW2 + OCW2_NS_EOI);/* signal EOI to 8259 */
outportb(I1_8259_0,OCW2 + OCW2_NS_EOI);/* signal EOI to 8259 */
}

/* interrupt service routine for unused interrupt inputs */
void interrupt unident_irq(void)
{
cputs("A masked interrupt has occured, this is a hardware error!");
enable();
while(TRUE);
}

/* intialize LM628 interrupt system, intialize PICs, and set vectors
void init_pics()
{



int ax;
disable();
/* LM628 interrupt masks */
/*     "OR" in other masks that your software needs */
Intrmasks1 = TRAJCOMP;
Intrmasks2 = TRAJCOMP;
Intrmasks3 = TRAJCOMP;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Intrmasks4 = TRAJCOMP;
wrcmd(AXIS1, MSKI);/* mask LM628 interrupts */
wrword(AXIS1,  Intrmasks1);
wrcmd(AXIS2, MSKI);
wrword(AXIS2,  Intrmasks2);
wrcmd(AXIS3, MSKI);
wrword(AXIS3,  Intrmasks3);
wrcmd(AXIS4, MSKI);
wrword(AXIS4,  Intrmasks4);

/* reset host interrupts for all axes */
for(ax = AXIS1; ax<=AXIS4; ax++)  {
wrcmd(ax, RSTI);
wrword(ax, 0);      /* data word for all interrupt masks cleared */
}
/* axes 1-4 */
setvect(VECTOR_BASE + 0, isr_axis_1);
setvect(VECTOR_BASE + 1, isr_axis_2);
setvect(VECTOR_BASE + 2, isr_axis_3);
setvect(VECTOR_BASE + 3, isr_axis_4);
setvect(VECTOR_BASE + 4, unident_irq);
setvect(VECTOR_BASE + 5, unident_irq);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
setvect(VECTOR_BASE + 6, unident_irq);
setvect(VECTOR_BASE + 7, spurious_u43);

/* u41 pic - cw and ccw limits */
setvect(VECTOR_BASE + 8,  isr_u41);
setvect(VECTOR_BASE + 9,  isr_u41);
setvect(VECTOR_BASE + 10, isr_u41);
setvect(VECTOR_BASE + 11, isr_TOP1);
setvect(VECTOR_BASE + 12, isr_u41);
setvect(VECTOR_BASE + 13, isr_u41);
setvect(VECTOR_BASE + 14, isr_u41);
setvect(VECTOR_BASE + 15, isr_BOTTOM1);

/* u42 pic - home limits */
setvect(VECTOR_BASE + 16, isr_u42);
setvect(VECTOR_BASE + 17, isr_u42);
setvect(VECTOR_BASE + 18, isr_u42);
setvect(VECTOR_BASE + 19, isr_home1);
setvect(VECTOR_BASE + 20, unident_irq);
setvect(VECTOR_BASE + 21, unident_irq);
setvect(VECTOR_BASE + 22, unident_irq);
setvect(VECTOR_BASE + 23, spurious_u42);

/* I1_8259 initialization sequence */
/*   IR0-3 edge triggered  lm628 interrupts
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/*    IR4    cascade interrupt for I2_8259 */
/*    IR5    cascade interrupt for I3_8259 */
/*    IR6-7 unused, must be masked */
outportb(I1_8259_0, ICW1 | ICW1_ETM | ICW1_CASCADE | ICW1_IC4);
outportb(I1_8259_1, ICW2 | VECTOR_BASE + 0);
outportb(I1_8259_1, ICW3 | ICW3_S4 | ICW3_S5);
outportb(I1_8259_1, ICW4 | ICW4_8086);
Msk_8259_1 = (OCW1 | OCW1_M6 | OCW1_M7 );
outportb(I1_8259_1, Msk_8259_1);

/* I2_8259 initialization sequence */
/*    IR0-3 edge triggered, CCW_LIMITS, AX1-4 */
/*    IR4-7 edge triggered, CW_LIMITS, AX1-4 */
outportb(I2_8259_0, ICW1 | ICW1_ETM | ICW1_CASCADE | ICW1_IC4);
outportb(I2_8259_1, ICW2 | VECTOR_BASE + 8);
outportb(I2_8259_1, ICW3 | ICW3_ID4);
outportb(I2_8259_1, ICW4 | ICW4_8086);
Msk_8259_2 =  ( OCW1/* | OCW1_M0 | OCW1_M1 | OCW1_M2 | OCW1_M3 |
       OCW1_M4 | OCW1_M5 | OCW1_M6 | OCW1_M7*/);
outportb(I2_8259_1, Msk_8259_2);

/* I3_8259 initialization sequence */
/*    IR0-3 edge triggered, HOME_LIM, AX1-4 */
/*    IR4-7 unused, must be masked */
outportb(I3_8259_0, ICW1 | ICW1_ETM | ICW1_CASCADE | ICW1_IC4);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
outportb(I3_8259_1, ICW2 | VECTOR_BASE + 16);
outportb(I3_8259_1, ICW3 | ICW3_ID5);
outportb(I3_8259_1, ICW4 | ICW4_8086);
Msk_8259_3 =  ( OCW1/* | OCW1_M0 | OCW1_M1 | OCW1_M2 | OCW1_M3 |
        OCW1_M4 | OCW1_M5 | OCW1_M6 | OCW1_M7*/);
outportb(I3_8259_1, Msk_8259_3);

enable();
}

/* disable interrupt system by masking all interrupts */
void dis_pics()
{
disable();



/* disable I1_8259 */
Msk_8259_1 = ( OCW1 | OCW1_M0 | OCW1_M1 | OCW1_M2 | OCW1_M3 |
      OCW1_M4 | OCW1_M5 | OCW1_M6 | OCW1_M7 );
outportb(I1_8259_1, Msk_8259_1);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Msk_8259_2 = ( OCW1 | OCW1_M0 | OCW1_M1 | OCW1_M2 | OCW1_M3 |
     OCW1_M4 | OCW1_M5 | OCW1_M6 | OCW1_M7 );
outportb(I2_8259_1, Msk_8259_2);

/* disable I3_8259 */
Msk_8259_3 = ( OCW1 | OCW1_M0 | OCW1_M1 | OCW1_M2 | OCW1_M3 |
           OCW1_M4 | OCW1_M5 | OCW1_M6 | OCW1_M7 );
outportb(I3_8259_1, Msk_8259_3);

enable();
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/*This is included file for table moving horizontal direction */

/* motor_position */
#define UNKNOWN_POS                0x00
#define OUT_OF_RANGE               0x01
#define HOME                       0x02
#define BOTTOM_LIMIT               0x03
#define TOP_LIMIT                  0x04
#define BOTTOM_HOME                0x05
#define TOP_HOME                   0x06
#define BOTTOM_OFF                 0x07
#define TOP_OFF                    0x08
#define BOTTOM_TOP                 0x09
#define BOTTOMOFF_BOTTOMLIMIT      0x0A
#define TOPOFF_TOPLIMIT            0x0B

/*move state*/

#define STOP                       0x00
#define STOP_HOME                  0x01
#define STOP_BOTTOM                0x02
#define STOP_TOP                   0x03
#define FIND_HOME                  0x04
#define FIND_BOTTOM_LIMIT          0x05
#define FIND_TOP_LIMIT             0x06
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#define SUCCESSFUL                  0x00
#define PROGRAM_ERROR               0x01
#define MOVE_IN_OPERATION           0x02
#define BOTTOM_ERROR                0x03
#define TOP_ERROR                   0x04
#define HOME_ERROR                  0x05
#define NORMAL_MOVE_ERROR           0x06

/*motor direction*/
#define MOTOR_STOP                  0x00
#define MOTOR_BOTTOM_MOVE           0x01
#define MOTOR_TOP_MOVE              0x02
#define UNKNOWN_PREVIOUS_DIR        0x03
#define PREVIOUS_BOTTOM_DIR         0x04
#define PREVIOUS_TOP_DIR            0x05

/*motor command*/
#define GO_DESIRED_POS              0x00
#define GO_HOME                     0x01
#define GO_BOTTOM_LIMIT             0x02
#define GO_TOP_LIMIT                0x03
#define INITIAL                     0x04
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/*input port*/
 #define    INPUT0                    0x118
 #define    INPUT1                    0x11A
 #define    BOTTOM_LIMIT_MAX      -1000L /*20.9437mm  compare with h
 #define    TOP_LIMIT_MAX          24000L /*502.65mm*/
 #define    TOP_LIMIT_CONV         502.65
 #define    BOTTOM_LIMIT_CONV      20.9437
 #define    DIM                   32  /*encoder dimeter*/
 #define    ESC                   0x1b
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ข
เกลียวสี่เหลี่ยมคางหมูตามมาตรฐานไอเอสโอ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 11.3 (ต่อ) มิติมูลฐานของเกลียวสี่เหลี่ยมคางหมูตามมาตรฐาน ISO 2904-1977 (E)

ขนาดระบุ			ระยะพิตช์ P	$d_2 = D_2$	D_4	d_3	D_1
ช่อง 1	ช่อง 2	ช่อง 3					
8			1.5	7.250	8.300	6.200	6.500
	9		1.5	8.250	9.300	7.200	7.500
			*2	8.000	9.500	6.500	7.000
10			1.5	9.250	10.300	8.200	8.500
			*2	9.000	10.500	7.500	8.000
	11		2	10.000	11.500	8.500	9.000
			*3	9.500	11.500	7.500	8.000
12			2	11.000	12.500	9.500	10.000
			*3	10.500	12.500	8.500	9.000
	14		2	13.000	14.500	11.500	12.000
			*3	12.500	14.500	10.500	11.000
16			2	15.000	16.500	13.500	14.000
			*4	14.000	16.500	11.500	12.000
	18		2	17.000	18.500	15.500	16.000
			*4	16.000	18.500	13.500	14.000
20			2	19.000	20.500	17.500	18.000
			*4	18.000	20.500	15.500	16.000
	22		3	20.500	22.500	18.500	19.000
			*5	19.500	22.500	16.500	17.000
			8	18.000	23.000	13.000	14.000
24			3	22.500	24.500	20.500	21.000
			*5	21.500	24.500	18.500	19.000
			8	20.000	25.000	15.000	16.000
	26		3	24.500	26.500	22.500	23.000
			*5	23.500	26.500	20.500	21.000
			8	22.000	27.000	17.000	18.000
28			3	26.500	28.500	24.500	25.000
			*5	25.500	28.500	22.500	23.000
			8	24.000	29.000	19.000	20.000
	30		3	28.000	30.500	26.500	27.000
			*6	27.000	31.000	23.000	24.000
			10	25.000	31.000	19.000	20.000
32			3	30.500	32.500	28.500	29.000
			*6	29.000	33.000	25.000	26.000
			10	27.000	33.000	21.000	22.000

ขนาดเป็น mm

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 11.3 (ต่อ) มิติมูลฐานของเกลียวสี่เหลี่ยมคางหมูตามมาตรฐาน ISO2904-1977 (E)

ขนาดระบุ			ระยะพิตช์ p	$d_2 = D_2$	D_4	d_3	D_1
ช่อง 1	ช่อง 2	ช่อง 3					
	34		3	32.500	34.500	30.500	31.000
			*6	31.000	35.000	27.000	28.000
			10	29.000	35.000	23.000	24.000
36			3	34.500	36.500	32.500	33.000
			*6	33.000	37.000	29.000	30.000
			10	31.000	37.000	25.000	26.000
	38		3	36.500	38.500	34.500	35.000
			*7	34.500	39.000	30.000	31.000
			10	33.000	39.000	27.000	28.000
40			3	38.500	40.500	36.500	37.000
			*7	36.500	41.000	32.000	33.000
			10	35.000	41.000	29.000	30.000
	42		3	40.500	42.500	38.500	39.000
			*7	38.500	43.000	34.000	35.000
			10	37.000	43.000	31.000	32.000
44			3	42.500	44.500	40.500	41.000
			*7	40.500	45.000	36.000	37.000
			12	38.000	45.000	31.000	32.000
	46		3	44.500	46.500	42.500	43.000
			*8	42.000	47.000	37.000	38.000
			12	40.000	47.000	33.000	34.000
48			3	46.500	48.500	44.500	45.000
			*8	44.000	49.000	39.000	40.000
			12	42.000	49.000	35.000	36.000
	50		3	48.500	50.500	46.500	47.000
			*8	46.000	51.000	41.000	42.000
			12	44.000	51.000	37.000	38.000
52			3	50.500	52.500	48.500	49.000
			*8	48.000	53.000	43.000	44.000
			12	46.000	53.000	39.000	40.000
	55		3	53.500	55.500	51.500	52.000
			*9	50.500	55.000	45.000	46.000
			14	48.000	57.000	39.000	41.000
60			3	58.500	60.500	56.500	57.000
			*9	55.500	61.000	50.000	51.000
			14	53.000	62.000	44.000	46.000

ขนาดเป็น m

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งไม่อนุญาตให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 11.3 (ต่อ) มิติมูลฐานของเกลียวสี่เหลี่ยมคางหมูตามมาตรฐาน ISO 2901-1977 (E)

ขนาดระบุ			ระยะพิตช์ p	$d_3 = D_3$	D_4	d_2	D_1
ช่อง 1	ช่อง 2	ช่อง 3					
	65		4	63.000	65.500	60.500	61.000
			*10	60.000	66.000	54.000	55.000
			16	57.000	67.000	47.000	49.000
70			4	68.000	70.500	65.500	66.000
			*10	65.000	71.000	59.000	60.000
			16	62.000	72.000	52.000	54.000
	75		4	73.000	75.500	70.500	71.000
			*10	70.000	76.000	64.000	65.000
			16	67.000	77.000	57.000	59.000
80			4	78.000	80.500	75.500	76.000
			*10	75.000	81.000	69.000	70.000
			16	72.000	82.000	62.000	64.000
	85		4	83.000	85.500	80.500	81.000
			*12	79.000	86.000	72.000	73.000
			18	76.000	87.000	65.000	67.000
90			4	88.000	90.500	85.500	86.000
			*12	84.000	91.000	77.000	78.000
			18	81.000	92.000	70.000	72.000
	95		4	93.000	95.500	90.500	91.000
			*12	89.000	96.000	82.000	83.000
			18	86.000	97.000	75.000	77.000
100			4	98.000	100.500	95.500	96.000
			*12	94.000	101.000	87.000	88.000
			20	90.000	102.000	78.000	80.000
		105	4	103.000	105.500	100.500	101.000
			*12	99.000	106.000	92.000	93.000
			20	95.000	107.000	83.000	85.000
	110		4	108.000	110.500	105.500	106.000
			*12	104.000	111.000	97.000	98.000
			20	100.000	112.000	88.000	90.000
		115	6	112.000	116.000	108.000	109.000
			*14	108.000	117.000	99.000	101.000
			22	104.000	117.000	91.000	93.000
120			6	117.000	121.000	113.000	114.000
			*14	113.000	122.000	104.000	106.000
			22	109.000	122.000	96.000	98.000

ขนาดเป็น mm

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 11.3 (ต่อ) มิติมูลฐานของเกลียวสี่เหลี่ยมคางหมูตามมาตรฐาน ISO 2904-1977 (E)

ขนาดระบุ			ระยะพิตช์ p	$d_2 = D_2$	D_4	d_3	D_1
ช่อง 1	ช่อง 2	ช่อง 3					
		125	6	122.000	126.000	118.000	119.000
			*14	118.000	127.000	109.000	111.000
			22	114.000	127.000	101.000	103.000
	130		6	127.000	131.000	123.000	124.000
			*14	123.000	132.000	114.000	116.000
			22	119.000	132.000	106.000	108.000
		135	6	132.000	136.000	128.000	129.000
			*14	128.000	137.000	119.000	121.000
			24	123.000	137.000	109.000	111.000
140			6	137.000	141.000	133.000	134.000
			*14	133.000	142.000	124.000	126.000
			24	128.000	142.000	114.000	116.000
		145	6	142.000	146.000	138.000	139.000
			*14	138.000	147.000	129.000	131.000
			24	133.000	147.000	119.000	121.000
	150		6	147.000	151.000	143.000	144.000
			*16	142.000	152.000	132.000	134.000
			24	138.000	152.000	124.000	126.000
		155	6	152.000	156.000	148.000	149.000
			*16	147.000	157.000	137.000	139.000
			24	143.000	157.000	129.000	131.000
160			6	157.000	161.000	153.000	154.000
			*16	152.000	162.000	142.000	144.000
			28	146.000	162.000	130.000	132.000
		165	6	162.000	166.000	158.000	159.000
			*16	157.000	167.000	147.000	149.000
			28	151.000	167.000	135.000	137.000
	170		6	167.000	171.000	163.000	164.000
			*16	162.000	172.000	152.000	154.000
			28	156.000	172.000	140.000	142.000
		175	8	171.000	176.000	166.000	167.000
			*15	167.000	177.000	157.000	159.000
			28	161.000	177.000	145.000	147.000
180			8	176.000	181.000	171.000	172.000
			*18	171.000	182.000	160.000	162.000
			28	166.000	182.000	150.000	152.000

ขนาดเป็น mm

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 11.3 (ต่อ) มิติมูลฐานของเกลียวสี่เหลี่ยมคางหมูตามมาตรฐาน ISO 2904-1977 (E)

ขนาดระบุ			ระยะพิตช์ p	$d_2 = D_2$	D_4	d_3	D_1
ช่อง 1	ช่อง 2	ช่อง 3					
	185		8	181.000	186.000	176.000	177.000
			*18	176.000	187.000	165.000	167.000
			32	169.000	187.000	151.000	153.000
	190		8	186.000	191.000	181.000	182.000
			*18	181.000	192.000	170.000	172.000
			32	174.000	192.000	156.000	158.000
		195	8	191.000	196.000	186.000	187.000
			*18	186.000	197.000	175.000	177.000
			32	179.000	197.000	161.000	163.000
200			8	196.000	201.000	191.000	192.000
			*18	191.000	202.000	180.000	182.000
			32	184.000	202.000	166.000	168.000
	210		8	206.000	211.000	201.000	202.000
			*20	200.000	212.000	188.000	190.000
			36	192.000	212.000	172.000	174.000
220			8	216.000	221.000	211.000	212.000
			*20	210.000	222.000	198.000	200.000
			36	202.000	222.000	182.000	184.000
	230		8	226.000	231.000	221.000	222.000
			*20	220.000	232.000	208.000	210.000
			36	212.000	232.000	192.000	194.000
240			8	236.000	241.000	231.000	232.000
			*22	229.000	242.000	216.000	218.000
			36	222.000	242.000	202.000	204.000
	250		12	244.000	251.000	237.000	238.000
			*22	239.000	252.000	226.000	228.000
			40	230.000	252.000	208.000	210.000
260			12	254.000	261.000	247.000	248.000
			*22	249.000	262.000	236.000	238.000
			40	240.000	262.000	218.000	220.000
	270		12	264.000	271.000	257.000	258.000
			*24	258.000	272.000	244.000	246.000
			40	250.000	272.000	228.000	230.000
280			12	274.000	281.000	267.000	268.000
			*24	268.000	282.000	254.000	256.000
			40	260.000	282.000	238.000	240.000

ขนาดเป็น mm

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 11.3 (ต่อ) มิติมูลฐานของเกลียวสี่เหลี่ยมคางหมูตามมาตรฐาน ISO 2904-1977 (E)

ขนาดระบุ			ระยะพิตช์ p	$d_2 = D_2$	D_4	d_3	D_1
ช่อง 1	ช่อง 2	ช่อง 3					
	290		12	284.000	291.000	277.000	278.000
			*24	278.000	292.000	264.000	266.000
			44	268.000	292.000	244.000	246.000
300			12	294.000	301.000	287.000	288.000
			*24	288.000	302.000	274.000	276.000
			44	278.000	302.000	254.000	256.000

ขนาดเป็น mm

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ค

รายละเอียดของเอนโคดเดอร์ และ

User's Guide :Four Axis Servo Controller 7340

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Chapter 1
Introduction

Four Axis Servo Controller 7340

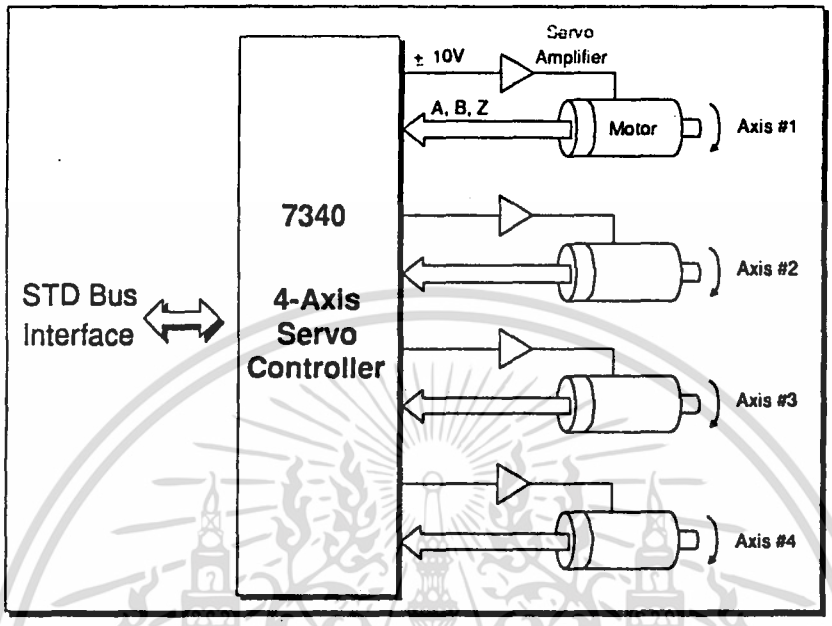
The Pro-Log 7340 Four Axis Servo Control Card for the STD bus provides position and velocity control for four independent axes of motion. It accepts incremental encoder feedback from standard quadrature encoders and produces a ±10 volt output to drive standard DC servo and brushless servo amplifiers. Utilizing four National Semiconductor LM628 motion control processors, this card controls servo motor velocity and position without the need for tachometers, (although they may be used for applications requiring extreme low speed smoothness).

Position, maximum velocity, and acceleration are 32-bit values programmed by the user. Trapezoidal motion profiles are produced. At any time during a move, maximum velocity or final position can be changed "on-the-fly." The 7340 utilizes a PID (Proportional + Integral + Derivative) control algorithm to compensate the control loop. Each coefficient is user-programmable, and position error is calculated in real-time based on the programmed motion profile and current position.

The 7340 provides high resolution 12-bit analog outputs, digitally filtered single or differential encoder inputs (including Z channel), and end-of-travel limit inputs for each axis. A variety of conditions can be programmed to generate interrupts, such as move complete, excessive following error, or limit reached.

All user I/O is provided on a ribbon connector at the front edge of the card. An optional breakout board is available to convert the flat cable connections to removable Phoenix-type screw terminals.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



Features

- Four Independent Axes of Servo Control (One, Two and Three Axis Versions Available)
- Utilizes Four High Speed Signal Processors
- Digital Control of Position Velocity, Acceleration (32-bit Values)
- Simple Proportional + Integral + Derivative (PID) Tuning
- Can Be Used With or Without Tachometers
- Setup Software and Software Drivers Supplied
- CW and CCW Limits, Home and Home Enable Inputs
- Optional Screw Terminal Breakout Panel With Status LEDs
- +10 Volt Output Drives Standard Brushed and Brushless Servo Amplifiers
- Vectored Interrupts on Excessive Position Error, Break Point Reached, Move Complete, Index Pulse, and Limit Encountered

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- Digitally Filtered Encoder Inputs for Error-Free Counting in Noisy Environments— Single Ended or Differential
- On-The-Fly Velocity; Position Changes Make Complex Motion Profiling Easy
  - Following
  - Ratioing
  - Synchronizing
  - Contouring
  - Electronic Gearbox
  - Registration

Software Support

Pro-Log includes a demonstration program with the 7340 that provides on-line, context sensitive help information. A library of driver and interrupt handling routines written in "C" and assembly is provided for customer use and customizing. A user's guide explaining motion control basics, servo fundamentals, tuning, and setup of the 7340 is also included.

A graphical trajectory analysis program that compares the actual motion profile with the desired motion profile is also provided to assist the user with servo loop setup.

Specifications

Electrical

- Four axes of LM628 servo controller
- Individual 12 bit D/A's with ±10V outputs
- Frequency selectable four stage digital filter input for "glitch"-free quadrature up to 1MHz
- Single ended or differential inputs, jumper selectable
- User selectable termination resistor for differential inputs
- Separate +5 connector for break-out board so encoder +5 volts can be supplied by STD bus or external supply

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- Software selectable Z channel polarity
- Z channel information gated and latched under software control

Inputs:

- CW, CCW limits, active low, 5 to 30V logic
- Home enable (gates encoder Z channel)

Outputs:

- Four individual Enable Outputs, each held in inactive state until LM628's initialized
- Moving/Not Moving Output
- Limit Encountered Output
- Fault Output
- LEDs on optional break out panel:
  - Moving/Not Moving (Green)
  - Limit Encountered (Yellow)
  - Fault (Red)

Vectored Interrupt Capability:

- Trajectory Complete
- Breakpoint reached (immediate position)
- Excessive Position Error (programmable)
- Index Position
- Position Counter wrap-around
- Limit Encountered
- Home Enable Active
- Command Error

Mechanical

- Single slot STD form factor, 5/8" slot required
- Optional break out board with status LEDs and screw terminal connectors for user I/O

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Environmental

- 50 G shock, 5 G vibration
- 0 to 65°C operating ambient

Available Models

The 7340 is available in one, two, three, and four axis versions. Each 7340 shipment also includes a User's Guide and software diskette containing demo program, setup program, tuning assistance program, and programming utilities.

Part Numbers

- 7340-01 One-Axis Servo Controller
- 7340-02 Two-Axis Servo Controller
- 7340-03 Three-Axis Servo Controller
- 7340-04 Four-Axis Servo Controller

Options

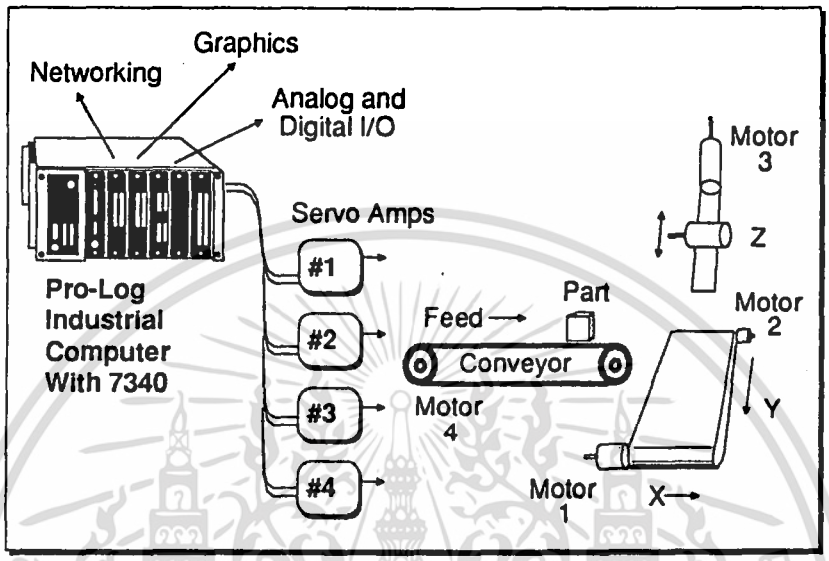
An optional breakout panel (BP-03) that converts 2 axes of the card's flat cable signal connections to removeable 0.2" Phoenix-type screw terminals is also available (two breakout panels are needed for three and four axis systems). Each BP-03 breakout panel also has LEDs for motion status, limit status, and fault conditions. The BP-03 has jumpers that allow the user to select whether encoder power is supplied from the STD bus main power supply or from an external supply. If STD bus main power is selected, users must factor the additional encoder load into overall system power supply requirements.

Description

- BP-03 Two-Axis Breakout Panel for 7340 Servo Controller

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

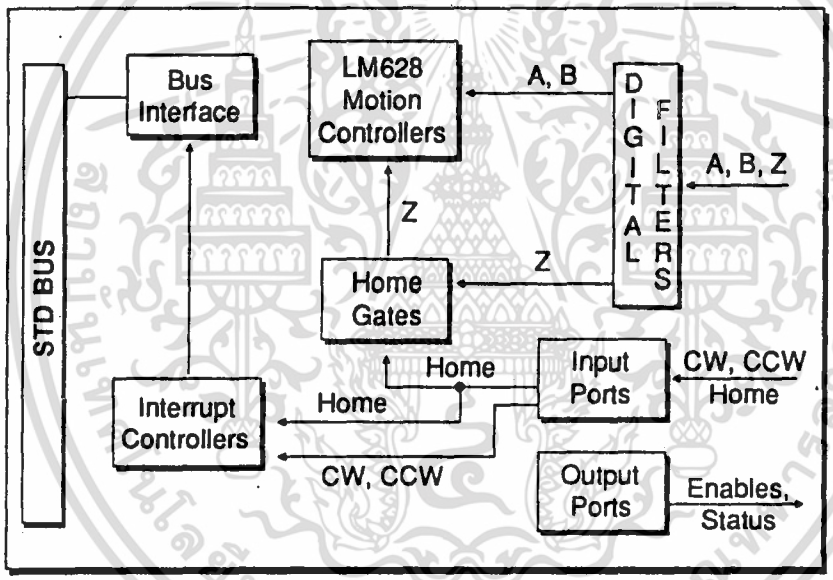
Typical Application Diagram



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Chapter 2
Theory Of Operation

The 7340 Servo Motion Controller is a powerful four axis motion controller. Four National Semiconductor LM628 servo control IC's coupled with interrupt controllers, flexible I/O, sophisticated trigger circuitry, and a fully STD-80 compatible bus interface make the 7340 the most complete single board servo controller on the market to date.



7340 Block Diagram

Bus Interface

The 7340's bus interface is implemented with two programmable GAL's (U19 and U21), which provide decode logic, interrupt priority chain decode, interrupt vector gate control and data bus transceiver control. W14 provides address decode to 11 bits of I/O address space.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Interrupt Control

Three 82C59A-2 interrupt control chips (U41, U42, and U43), are utilized in the generation of host interrupts. U41 and U42 are connected in a priority fashion to U43. For reference on the use and programming of the 82C59A-2, refer to the manufacturer's data sheet[1]. Seventeen vectored interrupts are available:

Vector:	Source:
0	LM628 axis 1 (U33)
1	LM628 axis 2 (U32)
2	LM628 axis 3 (U25)
3	LM628 axis 4 (U24)
4	CCW LIMIT AXIS 4 (LOW = INTERRUPT)
5	CCW LIMIT AXIS 3 (LOW = INTERRUPT)
6	CCW LIMIT AXIS 2 (LOW = INTERRUPT)
7	CCW LIMIT AXIS 1 (LOW = INTERRUPT)
8	CW LIMIT AXIS 4 (LOW = INTERRUPT)
9	CW LIMIT AXIS 3 (LOW = INTERRUPT)
10	CW LIMIT AXIS 2 (LOW = INTERRUPT)
11	CW LIMIT AXIS 1 (LOW = INTERRUPT)
12	HOME LIMIT AXIS 4 (LOW = INTERRUPT)
13	HOME LIMIT AXIS 3 (LOW = INTERRUPT)
14	HOME LIMIT AXIS 2 (LOW = INTERRUPT)
15	HOME LIMIT AXIS 1 (LOW = INTERRUPT)

This card cannot generate an Interrupt to the bus until Control port bit 7 is turned on. This is to provide protection from extraneous interrupts at Reset or Power-up.

LM628 Motion Controllers

Four National Semiconductor LM628 Precision Motion Controller IC's occupy U33, U32, U25, and U24. LM628's are a custom microcoded high performance processor

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

which implement a PID control algorithm. Actual position information is input from quadrature inputs A and B, desired position is input from the STD BUS via easy to use commands, and the motor command is output to a 12-bit DAC (U35, U37, U28, and U24). The voltage generated by the DAC is buffered by a voltage follower (U27). For LM628 programming information refer to Chapter 4 or the manufacturer's data sheet[2].

Z channel or Index input is generated from the encoder. The signal is fed through a digital filter (see next section), and into a programmable logic device. This device, U46 or U47 (depending upon which axis is being used), has the capability to gate the index input with the Home limit for that axis, invert the index pulse, or latch it in such a manner that only one index pulse is allowed to reach the LM628 until the programmable device is reset by an output port. The following table summarizes the function of output ports CP0 (U49) and and CP1 (U48):

CONTROL PORT:	**FUNCTION:**
CP0 BIT 0	ENABLE HOME GATE AXIS 1 (TRUE = 1)
CP0 BIT 1	ENABLE HOME GATE AXIS 2 (TRUE = 1)
CP0 BIT 2	ENABLE HOME GATE AXIS 3 (TRUE = 1)
CP0 BIT 3	ENABLE HOME GATE AXIS 4 (TRUE = 1)
CP0 BIT 4	INVERT INDEX AXIS 1 (INVERT = 1)
CP0 BIT 5	INVERT INDEX AXIS 2 (INVERT = 1)
CP0 BIT 6	INVERT INDEX AXIS 3 (INVERT = 1)
CP0 BIT 7	INVERT INDEX AXIS 4 (INVERT = 1)
CP1 BIT 0	LATCH INDEX AXIS 1 (LATCH = 1) (RESET = 0)
CP1 BIT 1	LATCH INDEX AXIS 2 (LATCH = 1) (RESET = 0)
CP1 BIT 2	LATCH INDEX AXIS 3 (LATCH = 1 (RESET = 0)
CP1 BIT 3	LATCH INDEX AXIS 4 (LATCH = 1) (RESET = 0)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

CP1 BITS 4 THRU 6 UNUSED

*CP1 BIT 7 INTERRUPT ENABLE (ENABLED = 1)
(DISABLED = 0)

*This bit is used to enable interrupts from this card.

Digital Filter

Encoder inputs can be subjected to some harsh EMI that may cause the LM628's to loose track of their position information. Noise spikes from motors, contactors, or other electrical machinery can cause the quadrature information to glitch, thereby falsely incrementing or decrementing the internal counters inside the LM628's. Pro-Log has developed a digital filter which reduces the susceptibility to EMI. Quadrature input A and B (index input Z as well), are received by a 26LS32 differential line receiver. Jumpers W1 through W11 allow the user to configure the input for true differential (connect pins 2 and 3), single ended (connect pins 3 and 4), and single ended with ground shield (connect pins 3 and 4 together; connect pins 1 and 2 together).

When the encoder interface is used in a differential configuration, resistor packs R2 and R3 may be substituted for a value which matches the characteristic line impedance, thereby further reducing noise.

The output of all 26LS32's is fed to a digital filter which samples the input and only allows the signal to pass on to the LM628 if the input did not glitch during four consecutive sampling periods. The sampling period is adjustable, with frequencies ranging from 12 Mhz to 0.75 Mhz (selected by W13). If W13 has pins 1 and 2 connected, 12 Mhz will be chosen as the sampling period for the digital filter. If a noise pulse is present on the input for up to (1/12 Mhz)*4 (333 ns), it will be rejected. If less than 6 Mhz sampling is selected, the maximum input frequency which can be counted will be reduced.

Inputs And Outputs

We have already discussed half of the output section CP0 and CP1. OUTPUT0 (U51) and OUTPUT1 (U50) are general purpose outputs which may be programmed under user

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

software control to indicate status or control functions within the motion control system. OUTPUT0 is connected to MC1413 darlington driver chips which can drive small relays or LED's. A logic 0 written into this control port will turn off the output driver and generate a logic 1 at the output. Bits 0 through 3 of OUTPUT1 are configured the same way. Bits 4 through 7 are connected to a 7407 non-inverting driver. When a logic 0 is written to these port bits, the output will be 0. This is useful when the output is connected to the enable input of a power amplifier (most enable outputs are disabled when the disable input is low). The outputs are reset at power on to a 1 (OUTPUT0, OUTPUT1 bits 0 thru 3), and to a 0 (OUTPUT1 bits 4 thru 7).

Inputs are provided by two input ports INPUT0 and INPUT1. Both ports read the input values of the CCW, CW, and HOME limit inputs after the input conditioning circuitry. The input conditioning circuitry allows the user to apply between ±24V to the input without damage. Logic thresholds are still at TTL levels (+.8V and +2V). The input conditioning circuitry inverts the signal before it is presented to the input ports.

---

1. *Intel Peripherals*, Intel Corporation, Intel Literature Sales, P.O. Box 7641, Mt. Prospect, IL 60056-7641, (800) 548-4725.
2. *LM628/LM629 Precision Motion Controller*, National Semiconductor Corporation, 2900 Semiconductor Drive, P.O. BOX 58090, Santa Clara, CA 95052-8090.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Appendix D
LM628 Description

The following pages have been reproduced from the National Semiconductor *LM628/LM629 Precision Motion Controller* Data Sheet, March, 1989 and are included for reference only.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

National Semiconductor

March 1989

LM628/LM629 Precision Motion Controller

General Description

The LM628/LM629 are dedicated motion-control processors designed for use with a variety of DC and brushless DC servo motors, and other servomechanisms which provide a quadrature incremental position feedback signal. The parts perform the intensive, real-time computational tasks required for high performance digital motion control. The host control software interface is facilitated by a high-level command set. The LM628 has an 8-bit output which can drive either an 8-bit or a 12-bit DAC. The components required to build a servo system are reduced to the DC motor/actuator, an incremental encoder, a DAC, a power amplifier, and the LM628. An LM629-based system is similar, except that it provides an 8-bit PWM output for directly driving H-switches. The parts are fabricated in NMOS and packaged in a 28-pin dual in-line package, and are offered in both 6 MHz and 8 MHz maximum frequency versions. The suffixes -6 and -8, respectively, are used to designate version. They incorporate an SDA core processor and cells designed by SDA

Features

- 32-bit position, velocity, and acceleration registers
- Programmable digital PID filter with 16-bit coefficients
- Programmable derivative sampling interval
- 8- or 12-bit DAC output data (LM628)
- 8-bit sign-magnitude PWM output data (LM629)
- Internal trapezoidal velocity profile generator
- Velocity, target position, and filter parameters may be changed during motion
- Position and velocity modes of operation
- Real-time programmable host interrupts
- 8-bit parallel asynchronous host interface
- Quadrature incremental encoder interface with index pulse input

TM STAR* is a registered trademark of National Semiconductor Corporation

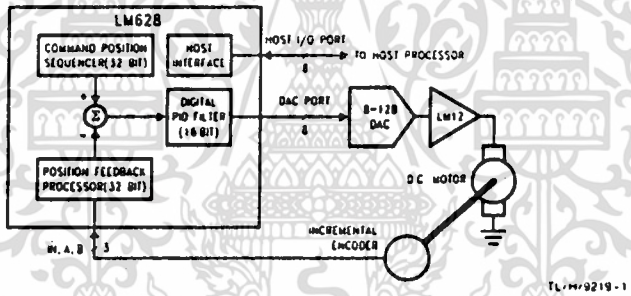
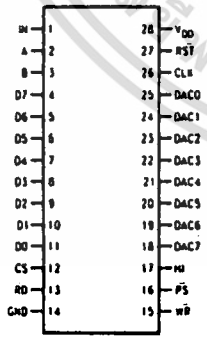


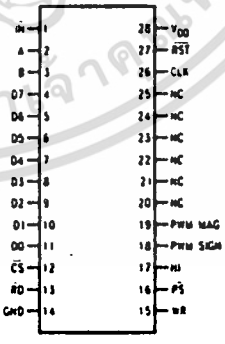
FIGURE 1. Typical System Block Diagram

Connection Diagrams

LM628



LM629



Order Number LM628N-6, LM628N-8, LM629N-6 or LM629N-8
See NS Package Number N28B

© 1989 National Semiconductor Corporation TL/H/9219

RRD-B20M39/Printed in U.S.A.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Theory of Operation

INTRODUCTION

The typical system block diagram (See *Figure 1*) illustrates a servo system built using the LM628. The host processor communicates with the LM628 through an I/O port to facilitate programming a trapezoidal velocity profile and a digital compensation filter. The DAC output interfaces to an external digital-to-analog converter to produce the signal that is power amplified and applied to the motor. An incremental encoder provides feedback for closing the position servo loop. The trapezoidal velocity profile generator calculates the required trajectory for either position or velocity mode of operation. In operation, the LM628 subtracts the actual position (feedback position) from the desired position (profile generator position), and the resulting position error is processed by the digital filter to drive the motor to the desired position. Table I provides a brief summary of specifications offered by the LM628/LM629.

POSITION FEEDBACK INTERFACE

The LM628 interfaces to a motor via an incremental encoder. Three inputs are provided: two quadrature signal inputs, and an index pulse input. The quadrature signals are used to keep track of the absolute position of the motor. Each time a logic transition occurs at one of the quadrature inputs, the LM628 internal position register is incremented or decremented accordingly. This provides four times the resolution over the number of lines provided by the encoder. See *Figure 9*. Each of the encoder signal inputs is synchronized with the LM628 clock.

The optional index pulse output provided by some encoders assumes the logic-low state once per revolution. If the LM628 is so programmed by the user, it will record the absolute motor position in a dedicated register (the index register) at the time when all three encoder inputs are logic low.

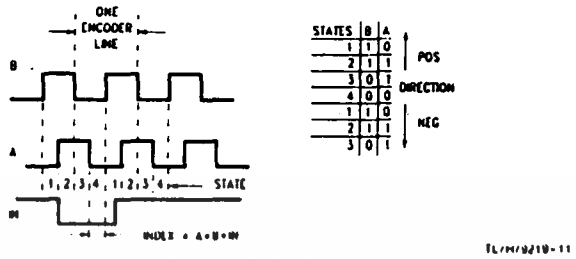
If the encoder does not provide an index output, the LM628 index input can also be used to record the home position of the motor. In this case, typically, the motor will close a switch which is arranged to cause a logic-low level at the index input, and the LM628 will record motor position in the index register and alert (interrupt) the host processor. When using the index input in this manner, the user should assure that the index input does not remain logic low during shaft rotation because LM628 internal interrupts are generated every time all three encoder inputs are logic low. These internal interrupts will cause the LM628 to malfunction if the velocity is faster than about 15,000 counts/second (when using a 6 MHz clock, or about 20,000 counts/second with an 8 MHz clock).

TABLE I. System Specifications Summary

Position Range	1,073,741,824 to 1,073,741,823 counts
Velocity Range	0 to $1,073,741,823/2^{16}$ counts/sample, ie, 0 to 16,383 counts/sample, with a resolution of $1/2^{16}$ counts/sample
Acceleration Range	0 to $1,073,741,823/2^{16}$ counts/sample/sample; ie, 0 to 16,383 counts/sample/sample, with a resolution of $1/2^{16}$ counts/sample/sample
Motor Drive Output	LM628: 8-bit parallel output to DAC, or 12-bit multiplexed output to DAC LM629: 8-bit PWM sign/magnitude signals
Operating Modes	Position and Velocity
Feedback Device	Incremental Encoder (quadrature signals, support for index pulse)
Control Algorithm	Proportional Integral Derivative (PID) (plus programmable integration limit)
Sample Intervals	Derivative Term: Programmable from $2048/f_{CLK}$ to $(2048 * 256)/f_{CLK}$ in steps of $2048/f_{CLK}$ (256 to 65.536 μs for an 8.0 MHz clock) Proportional and Integral: $2048/f_{CLK}$

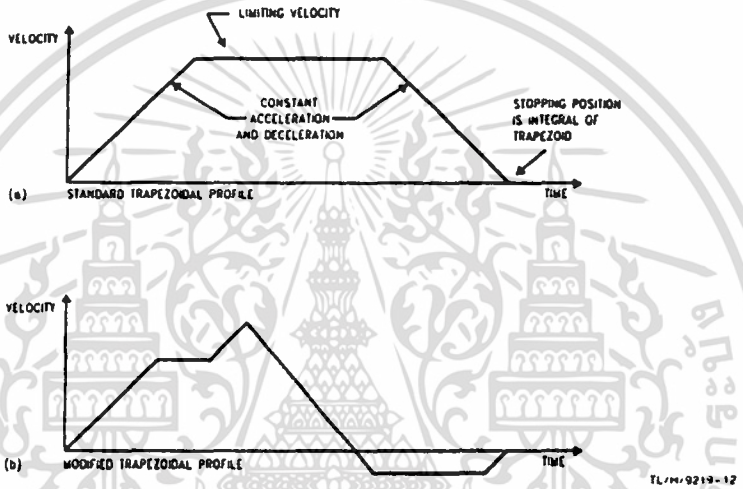
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Theory of Operation (Continued)



TL/H/9219-11

FIGURE 9. Quadrature Encoder Signals



TL/H/9219-12

FIGURE 10. Typical Velocity Profiles

VELOCITY PROFILE (TRAJECTORY) GENERATION

The trapezoidal velocity profile generator computes the desired position of the motor versus time. In the position mode of operation, the host processor specifies acceleration, maximum velocity, and final position. The LM628 uses this information to affect the move by accelerating as specified until the maximum velocity is reached or until deceleration must begin to stop at the specified final position. The deceleration rate is equal to the acceleration rate. At any time during the move the maximum velocity and/or the target position may be changed, and the motor will accelerate or decelerate accordingly. *Figure 10* illustrates two typical trapezoidal velocity profiles. *Figure 10 (a)* shows a simple trapezoid, while *Figure 10 (b)* is an example of what the trajectory looks like when velocity and position are changed at different times during the move.

When operating in the velocity mode, the motor accelerates to the specified velocity at the specified acceleration rate and maintains the specified velocity until commanded to stop. The velocity is maintained by advancing the desired position at a constant rate. If there are disturbances to the motion during velocity mode operation, the long-time average velocity remains constant. If the motor is unable to maintain the specified velocity (which could be caused by a locked rotor, for example), the desired position will continue to be increased, resulting in a very large position error. If this condition goes undetected, and the impeding force on the motor is subsequently released, the motor could reach a very high velocity in order to catch up to the desired position (which is still advancing as specified). This condition is easily detected; see commands LPEI and LPES.

All trajectory parameters are 32-bit values. Position is a signed quantity. Acceleration and velocity are specified as 16-bit, positive-only integers having 16-bit fractions. The integer portion of velocity specifies how many counts per sampling interval the motor will traverse. The fractional portion designates an additional fractional count per sampling interval. Although the position resolution of the LM628 is limited to integer counts, the fractional counts provide increased average velocity resolution. Acceleration is treated in the same manner. Each sampling interval the commanded acceleration value is added to the current desired velocity to generate a new desired velocity (unless the command velocity has been reached).

One determines the trajectory parameters for a desired move as follows. If, for example, one has a 500-line shaft encoder, desires that the motor accelerate at one revolution per second per second until it is moving at 600 rpm, and then decelerate to a stop at a position exactly 100 revolutions from the start, one would calculate the trajectory parameters as follows:

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Theory of Operation (Continued)

let P = target position (units = encoder counts)
let R = encoder lines * 4 (system resolution)
then R = 500 * 4 = 2000
and P = 2000 * desired number of revolutions
P = 2000 * 100 revs = 200,000 counts (value to load)
P (coding) = 00030D40 (hex code written to LM628)

let V = velocity (units = counts/sample)
let T = sample time (seconds) = 341 µs (with 6 MHz clock)
let C = conversion factor = 1 minute/60 seconds
then V = R * T * C * desired rpm
and V = 2000 * 341E−6 * 1/60 * 600 rpm
V = 6.82 counts/sample
V (scaled) = 6.82 * 65,536 = 446,955.52
V (rounded) = 446,956 (value to load)
V (coding) = 0006D1EC (hex code written to LM628)

let A = acceleration (units = counts/sample/sample)
A = R * T * T * desired acceleration (rev/sec/sec)
then A = 2000 * 341E−6 * 341E-6 * 1 rev/sec/sec
and A = 2.33E−4 counts/sample/sample
A (scaled) = 2.33E−4 * 65,536 = 15.24
A (rounded) = 15 (value to load)
A (coding) = 0000000F (hex code written to LM628)

The above position, velocity, and acceleration values must be converted to binary codes to be loaded into the LM628. The values shown for velocity and acceleration must be multiplied by 65,536 (as shown) to adjust for the required integer/fraction format of the input data. Note that after scaling the velocity and acceleration values, literal fractional data cannot be loaded; the data must be rounded and converted to binary. The factor of four increase in system resolution is due to the method used to decode the quadrature encoder signals, see *Figure 9*

PID COMPENSATION FILTER

The LM628 uses a digital Proportional Integral Derivative (PID) filter to compensate the control loop. The motor is held at the desired position by applying a restoring force to the motor that is proportional to the position error, plus the integral of the error, plus the derivative of the error. The following discrete-time equation illustrates the control performed by the LM628:

$$u(n) = kp * e(n) + ki \sum_{N=0}^{n} e(n) + kd\{e(n') - e(n' - 1)\} \qquad \text{(Eq 1)}$$

where u(n) is the motor control signal output at sample time n, e(n) is the position error at sample time n, n' indicates sampling at the derivative sampling rate, and kp, ki, and kd are the discrete-time filter parameters loaded by the users.

The first term, the proportional term, provides a restoring force porportional to the position error, just as does a spring obeying Hooke's law. The second term, the integration term, provides a restoring force that grows with time, and thus ensures that the static position error is zero. If there is a constant torque loading, the motor will still be able to achieve zero position error.

The third term, the derivative term, provides a force proportional to the rate of change of position error. It acts just like viscous damping in a damped spring and mass system (like a shock absorber in an automobile). The sampling interval associated with the derivative term is user-selectable; this capability enables the LM628 to control a wider range of inertial loads (system mechanical time constants) by providing a better approximation of the continuous derivative. In general, longer sampling intervals are useful for low-velocity operations.

In operation, the filter algorithm receives a 16-bit error signal from the loop summing-junction. The error signal is saturated at 16 bits to ensure predictable behavior. In addition to being multiplied by filter coefficient kp, the error signal is added to an accumulation of previous errors (to form the integral signal) and, at a rate determined by the chosen *derivative* sampling interval, the previous error is subtracted from it (to form the derivative signal). All filter multiplications are 16-bit operations; only the bottom 16 bits of the product are used.

The integral signal is maintained to 24 bits, but only the top 16 bits are used. This scaling technique results in a more usable (less sensitive) range of coefficient ki values. The 16 bits are right-shifted eight positions and multiplied by filter coefficient ki to form the term which contributes to the motor control output. The absolute magnitude of this product is compared to coefficient il, and the lesser, appropriately signed magnitude then contributes to the motor control signal.

The derivative signal is multiplied by coefficient kd each *derivative* sampling interval. This product contributes to the motor control output *every* sample interval, independent of the user-chosen *derivative* sampling interval.

The kp, limited ki, and kd product terms are summed to form a 16-bit quantity. Depending on the output mode (wordsize), either the top 8 or top 12 bits become the motor control output signal.

LM628 READING AND WRITING OPERATIONS

The host processor writes commands to the LM628 via the host I/O port when Port Select ($\overline{PS}$) input (Pin 16) is logic low. The desired command code is applied to the parallel port line and the Write ($\overline{WR}$) input (Pin 15) is strobed. The command byte is latched into the LM628 on the rising edge of the $\overline{WR}$ input. When writing command bytes it is necessary to first read the status byte and check the state of a flag called the "busy bit" (Bit 0). If the busy bit is logic high, no command write may take place. The busy bit is never high longer than 100 µs, and typically falls within 15 µs to 25 µs.

The host processor reads the LM628 status byte in a similar manner: by strobing the Read ($\overline{RD}$) input (Pin 13) when $\overline{PS}$ (Pin 16) is low; status information remains valid as long as $\overline{RD}$ is low.

Writing and reading data to/from the LM628 (as opposed to writing commands and reading status) are done with $\overline{PS}$ (Pin 16) logic high. These writes and reads are always an integral number (from one to seven) of two-byte words, with the first byte of each word being the more significant. Each byte requires a write ($\overline{WR}$) or read ($\overline{RD}$) strobe. When transferring data words (byte-pairs), it is necessary to first read the status byte and check the state of the busy bit. When the

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Theory of Operation (Continued)

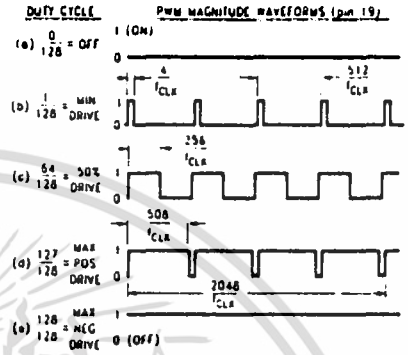
busy bit is logic low, the user may then sequentially transfer both bytes comprising a data word, but the busy bit must again be checked and found to be low before attempting to transfer the next byte pair (when transferring multiple words). Data transfers are accomplished via LM628-internal interrupts (which are not nested); the busy bit informs the host processor when the LM628 may not be interrupted for data transfer (or a command byte). If a command is written when the busy bit is high, the command will be ignored.

The busy bit goes high immediately after writing a command byte, or reading or writing a second byte of data (See *Figures 5* thru *7*)

MOTOR OUTPUTS

The LM628 DAC output port can be configured to provide either a latched eight-bit parallel output or a multiplexed 12-bit output. The 8-bit output can be directly connected to a flow-through (non-input-latching) D/A converter; the 12-bit output can be easily demultiplexed using an external 6-bit latch and an input-latching 12-bit D/A converter. The DAC output data is offset-binary coded, the 8-bit code for zero is 80 hex and the 12-bit code for zero is 800 hex. Values less than these cause a negative torque to be applied to the motor and, conversely, larger values cause positive motor torque. The LM628, when configured for 12-bit output, provides signals which control the demultiplexing process. See *Figure 8* for details.

The LM629 provides 8-bit, sign and magnitude PWM output signals for directly driving switch-mode motor-drive amplifiers. *Figure 11* shows the format of the PWM magnitude output signal.



Note: Sign output (pin 18) not shown

FIGURE 11. PWM Output Signal Format

TABLE II. LM628 User Command Set

Command	Type	Description	Hex	Data Bytes	Note
RESET	Initialize	Reset LM628	00	0	1
PORT8	Initialize	Select 8-Bit Output	05	0	2
PORT12	Initialize	Select 12-Bit Output	06	0	2
DFH	Initialize	Define Home	02	0	1
SIP	Interrupt	Set Index Position	03	0	1
LPEI	Interrupt	Interrupt on Error	1B	2	1
LPES	Interrupt	Stop on Error	1A	2	1
SBPA	Interrupt	Set Breakpoint, Absolute	20	4	1
SBPR	Interrupt	Set Breakpoint, Relative	21	4	1
MSKI	Interrupt	Mask Interrupts	1C	2	1
RSTI	Interrupt	Reset Interrupts	1D	2	1
LFIL	Filter	Load Filter Parameters	1E	2 to 10	1
UDF	Filter	Update Filter	04	0	1
LTRJ	Trajectory	Load Trajectory	1F	2 to 14	1
STT	Trajectory	Start Motion	01	0	3
RDSTAT	Report	Read Status Byte	None	1	1, 4
RDSIGS	Report	Read Signals Register	0C	2	1
RDIP	Report	Read Index Position	09	4	1
RDDP	Report	Read Desired Position	08	4	1
RDRP	Report	Read Real Position	0A	4	1
RDDV	Report	Read Desired Velocity	07	4	1
RDRV	Report	Read Real Velocity	0B	2	1
RDSUM	Report	Read Integration Sum	0D	2	1

Note 1: Commands may be executed "On the Fly" during motion.

Note 2: Commands not applicable to execution during motion.

Note 3: Command may be executed during motion if acceleration parameter was not changed.

Note 4: Command needs no code because the command port status byte read is totally supported by hardware.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

User Command Set

GENERAL

The following paragraphs describe the user command set of the LM628. Some of the commands can be issued alone and some require a supporting data structure. As examples, the command STT (STarT motion) does not require additional data, command LFIL (Load FILter parameters) requires additional data (derivative-term sampling interval and/or filter parameters).

Commands are categorized by function: initialization, interrupt control, filter control, trajectory control, and data reporting. The commands are listed in Table II and described in the following paragraphs. Along with each command name is its command-byte code, the number of accompanying data bytes that are to be written (or read), and a comment as to whether the command is executable during motion.

Initialization Commands

The following four LM628 user commands are used primarily to initialize the system for use.

RESET COMMAND: RESET the LM628

Command Code:	00 Hex
Data Bytes:	None
Executable During Motion:	Yes

This command (and the hardware reset input, Pin 27) results in setting the following data items to zero: filter coefficients and their input buffers, trajectory parameters and their input buffers, and the motor control output. A zero motor control output is a half-scale, offset-binary code: (80 hex for the 8-bit output mode; 800 hex for 12-bit mode). During reset, the DAC port outputs 800 hex to "zero" a 12-bit DAC and reverts to 80 hex to "zero" an 8-bit DAC. The command also clears five of the six interrupt masks (only the SBPA/SBPR interrupt is masked), sets the output port size to 8 bits, and defines the current absolute position as home. Reset, which may be executed at any time, will be completed in less than 1.5 ms. Also see commands PORT8 and PORT12.

PORT8 COMMAND: Set Output PORT Size to 8 Bits

Command Code:	05 Hex
Data Bytes:	None
Executable During Motion:	Not Applicable

The default output port size of the LM628 is 8 bits; so the PORT8 command need not be executed when using an 8-bit DAC. This command must not be executed when using a 12-bit converter; it will result in erratic, unpredictable motor behavior. The 8-bit output port size is the required selection when using the LM629, the PWM-output version of the LM628.

PORT12 COMMAND: Set Output PORT Size to 12 Bits

Command Code:	06 Hex
Data Bytes:	None
Executable During Motion:	Not Applicable

When a 12-bit DAC is used, command PORT12 should be issued very early in the initialization process. Because use of this command is determined by system hardware, there is only one foreseen reason to execute it later: if the RESET command is issued (because an 8-bit output would then be selected as the default) command PORT12 should be immediately executed. This command must not be issued when using an 8-bit converter or the LM629, the PWM-output version of the LM628.

DFH COMMAND: DeFine Home

Command Code:	02 Hex
Data Bytes:	None
Executable During Motion:	Yes

This command declares the current position as "home", or absolute position 0 (Zero). If DFH is executed during motion it will not affect the stopping position of the on-going move unless command STT is also executed.

Interrupt Control Commands

The following seven LM628 user commands are associated with conditions which can be used to interrupt the host computer. In order for any of the potential interrupt conditions to actually interrupt the host via Pin 17, the corresponding bit in the interrupt mask data associated with command MSKI must have been set to logic high (the non-masked state).

The identity of all interrupts is made known to the host via reading and parsing the status byte. Even if all interrupts are masked off via command MSKI, the state of each condition is still reflected in the status byte. This feature facilitates polling the LM628 for status information, as opposed to interrupt driven operation.

SIP COMMAND: Set Index Position

Command Code:	03 Hex
Data Bytes:	None
Executable During Motion:	Yes

After this command is executed, the absolute position which corresponds to the occurrence of the next index pulse input will be recorded in the index register, and bit 3 of the status byte will be set to logic high. The position is recorded when both encoder-phase inputs and the index pulse input are logic low. This register can then be read by the user (see description for command RDIP) to facilitate aligning the definition of home position (see description of command DFH) with an index pulse. The user can also arrange to have the LM628 interrupt the host to signify that an index pulse has occurred. See the descriptions for commands MSKI and RSTI.

LPEI COMMAND: Load Position Error for Interrupt

Command Code:	1B Hex
Data Bytes:	Two
Data Range:	0000 to 7FFF Hex
Executable During Motion:	Yes

An excessive position error (the output of the loop summing junction) can indicate a serious system problem; e.g., a stalled rotor. Instruction LPEI allows the user to input a threshold for position error detection. Error detection occurs when the absolute magnitude of the position error exceeds the threshold, which results in bit 5 of the status byte being set to logic high. If it is desired to also stop (turn off) the motor upon detecting excessive position error, see command LPES, below. The first byte of threshold data written with command LPEI is the more significant. The user can have the LM628 interrupt the host to signify that an excessive position error has occurred. See the descriptions for commands MSKI and RSTI.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Interrupt Control Commands (Continued)

LPES COMMAND: Load Position Error for Stopping

Command Code:	1A Hex
Data Bytes:	Two
Data Range:	0000 to 7FFF Hex
Executable During Motion:	Yes

Instruction LPES is essentially the same as command LPEI above, but adds the feature of turning off the motor upon detecting excessive position error. The motor drive is not actually switched off, it is set to half-scale, the offset-binary code for zero. As with command LPEI, bit 5 of the status byte is also set to logic high. The first byte of threshold data written with command LPES is the more significant. The user can have the LM628 interrupt the host to signify that an excessive position error has occurred. See the descriptions for commands MSKI and RSTI.

SBPA COMMAND:

Command Code:	20 Hex
Data Bytes:	Four
Data Range:	C0000000 to 3FFFFFFF Hex
Executable During Motion:	Yes

This command enables the user to set a breakpoint in terms of absolute position. Bit 6 of the status byte is set to logic high when the breakpoint position is reached. This condition is useful for signaling trajectory and/or filter parameter updates. The user can also arrange to have the LM628 interrupt the host to signify that a breakpoint position has been reached. See the descriptions for commands MSKI and RSTI.

SBPR COMMAND:

Command Code:	21 Hex
Data Bytes:	Four
Data Range:	See Text
Executable During Motion:	Yes

This command enables the user to set a breakpoint in terms of relative position. As with command SBPA, bit 6 of the status byte is set to logic high when the breakpoint position (relative to the current commanded target position) is reached. The relative breakpoint input value must be such that when this value is added to the target position the result remains within the absolute position range of the system (C0000000 to 3FFFFFFF hex). This condition is useful for signaling trajectory and/or filter parameter updates. The user can also arrange to have the LM628 interrupt the host to signify that a breakpoint position has been reached. See the descriptions for commands MSKI and RSTI.

MSKI COMMAND: MaSK Interrupts

Command Code:	1C Hex
Data Bytes:	Two
Data Range:	See Text
Executable During Motion:	Yes

The MSKI command lets the user determine which potential interrupt condition(s) will interrupt the host. Bits 1 through 6 of the status byte are indicators of the six conditions which are candidates for host interrupt(s). When interrupted, the host then reads the status byte to learn which condition(s) occurred. Note that the MSKI command is immediately followed by two data bytes. Bits 1 through 6 of the second (less significant) byte written determine the masked/unmasked status of each potential interrupt. Any zero(s) in this 6-bit field will mask the corresponding interrupt(s); any one(s) enable the interrupt(s). Other bits comprising the two bytes have no effect. The mask controls only the host interrupt process; reading the status byte will still reflect the actual conditions independent of the mask byte. See Table III.

TABLE III. Mask and Reset Bit Allocations for Interrupts

Bit Position	Function
Bits 15 thru 7	Not Used
Bit 6	Breakpoint Interrupt
Bit 5	Position-Error Interrupt
Bit 4	Wrap-Around Interrupt
Bit 3	Index-Pulse Interrupt
Bit 2	Trajectory-Complete Interrupt
Bit 1	Command-Error Interrupt
Bit 0	Not Used

RSTI COMMAND: ReSeT Interrupts

Command Code:	1D Hex
Data Bytes:	Two
Data Range:	See Text
Executable During Motion:	Yes

When one of the potential interrupt conditions of Table III occurs, command RSTI is used to reset the corresponding interrupt flag bit in the status byte. The host may reset one or all flag bits. Resetting them one at a time allows the host to service them one at a time according to a priority programmed by the user. As in the MSKI command, bits 1 through 6 of the second (less significant) byte correspond to the potential interrupt conditions shown in Table III. Also see description of RDSTAT command. Any zero(s) in this 6-bit field reset the corresponding interrupt(s). The remaining bits have no effect.

Filter Control Commands

The following two LM628 user commands are used for setting the derivative-term sampling interval, for adjusting the filter parameters as required to tune the system, and to control the timing of these system changes.

LFIL COMMAND: Load FILter Parameters

Command Code:	1E Hex
Data Bytes:	Two to Ten
Data Ranges	
Filter Control Word	See Text
Filter Coefficients:	0000 to 7FFF Hex (Pos Only)
Integration Limit:	0000 to 7FFF Hex (Pos Only)
Executable During Motion:	Yes

The filter parameters (coefficients) which are written to the LM628 to control loop compensation are kp, ki, kd, and il (integration limit). The integration limit (il) constrains the contribution of the integration term

$$\left[ki \cdot \sum_{N=0}^{n} e(n) \right]$$

(see Eq. 1) to values equal to or less than a user-defined maximum value; this capability minimizes integral or reset "wind-up" (an overshooting effect of the integral action). The positive-only input value is compared to the absolute

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Filter Control Commands (Continued)

magnitude of the integration term; when the magnitude of integration term value exceeds il, the il value (with appropriate sign) is substituted for the integration term value.

The derivative-term sampling interval is also programmable via this command. After writing the command code, the first two data bytes that are written specify the derivative-term sampling interval and which of the four filter parameters is/are to be written via any forthcoming data bytes. The first byte written is the more significant. Thus the two data bytes constitute a filter control word that informs the LM628 as to the nature and number of any following data bytes. See Table IV.

TABLE IV. Filter Control word Bit Allocation

Bit Position	Function
Bit 15	Derivative Sampling Interval Bit 7
Bit 14	Derivative Sampling Interval Bit 6
Bit 13	Derivative Sampling Interval Bit 5
Bit 12	Derivative Sampling Interval Bit 4
Bit 11	Derivative Sampling Interval Bit 3
Bit 10	Derivative Sampling Interval Bit 2
Bit 9	Derivative Sampling Interval Bit 1
Bit 8	Derivative Sampling Interval Bit 0
Bit 7	Not Used
Bit 6	Not Used
Bit 5	Not Used
Bit 4	Not Used
Bit 3	Loading kp Data
Bit 2	Loading ki Data
Bit 1	Loading kd Data
Bit 0	Loading il Data

Bits 8 through 15 select the derivative-term sampling interval. See Table V. The user must locally save and restore these bits during successive writes of the filter control word.

Bits 4 through 7 of the filter control word are not used.

Bits 0 to 3 inform the LM628 as to whether any or all of the filter parameters are about to be written. The user may choose to update any or all (or none) of the filter parameters. Those chosen for updating are so indicated by logic one(s) in the corresponding bit position(s) of the filter control word.

The data bytes specified by and immediately following the filter control word are written in pairs to comprise 16-bit words. The order of sending the data words to the LM628 corresponds to the descending order shown in the above description of the filter control word; i.e., beginning with kp, then ki, kd and il. The first byte of each word is the more-significant byte. Prior to writing a word (byte pair) it is necessary to check the busy bit in the status byte for readiness. The required data is written to the primary buffers of a double-buffered scheme by the above described operations; it is not transferred to the secondary (working) registers until the UDF command is executed. This fact can be used advantageously; the user can input numerous data ahead of their actual use. This simple pipeline effect can relieve potential host computer data communications bottlenecks, and facilitates easier synchronization of multiple-axis controls.

UDF COMMAND: UpDate Filter

Command Code:	04 Hex
Data Bytes:	None
Executable During Motion	Yes

The UDF command is used to update the filter parameters, the specifics of which have been programmed via the LFIL command. Any or all parameters (derivative-term sampling interval, kp, ki, kd, and/or il) may be changed by the appropriate command(s), but command UDF must be executed to affect the change in filter tuning. Filter updating is synchronized with the calculations to eliminate erratic or spurious behavior.

Trajectory Control Commands

The following two LM628 user commands are used for setting the trajectory control parameters (position, velocity, acceleration), mode of operation (position or velocity), and direction (velocity mode only) as required to describe a desired motion or to select the mode of a manually directed stop, and to control the timing of these system changes.

LTRJ COMMAND: Load TRaJectory Parameters

Command Code:	1F Hex
Data Bytes:	Two to Fourteen
Data Ranges . . .	
Trajectory Control Word:	See Text
Position:	C0000000 to 3FFFFFFF Hex
Velocity:	00000000 to 3FFFFFFF Hex (Pos Only)
Acceleration:	00000000 to 3FFFFFFF Hex (Pos Only)
Executable During Motion:	Conditionally, See Text

TABLE V. Derivative-Term Sampling Interval Selection Codes

	Bit Position								Selected Derivative Sampling Interval
	15	14	13	12	11	10	9	8	
	0	0	0	0	0	0	0	0	256 µs
	0	0	0	0	0	0	0	1	512 µs
	0	0	0	0	0	0	1	0	768 µs
	0	0	0	0	0	0	1	1	1024 µs, etc
thru	1	1	1	1	1	1	1	1	65,536 µs

Note: Sampling intervals shown are when using an 8.0 MHz clock. The 256 corresponds to 2048/8 MHz; sample intervals must be scaled for other clock frequencies.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Trajectory Control Commands (Continued)

The trajectory control parameters which are written to the LM628 to control motion are: acceleration, velocity, and position. In addition, indications as to whether these three parameters are to be considered as absolute or relative inputs, selection of velocity mode and direction, and manual stopping mode selection and execution are programmable via this command. After writing the command code, the first two data bytes that are written specify which parameter(s) is/are being changed. The first byte written is the more significant. Thus the two data bytes constitute a trajectory control word that informs the LM628 as to the nature and number of any following data bytes. See Table VI.

TABLE VI. Trajectory Control Word Bit Allocation

Bit Position	Function
Bit 15	Not Used
Bit 14	Not Used
Bit 13	Not Used
Bit 12	Forward Direction (Velocity Mode Only)
Bit 11	Velocity Mode
Bit 10	Stop Smoothly (Decelerate as Programmed)
Bit 9	Stop Abruptly (Maximum Deceleration)
Bit 8	Turn Off Motor (Output Zero Drive)
Bit 7	Not Used
Bit 6	Not Used
Bit 5	Acceleration Will Be Loaded
Bit 4	Acceleration Data Is Relative
Bit 3	Velocity Will Be Loaded
Bit 2	Velocity Data Is Relative
Bit 1	Position Will Be Loaded
Bit 0	Position Data Is Relative

Bit 12 determines the motor direction when in the velocity mode. A logic one indicates forward direction. This bit has no effect when in position mode.

Bit 11 determines whether the LM628 operates in velocity mode (Bit 11 logic one) or position mode (Bit 11 logic zero).

Bits 8 through 10 are used to select the method of *manually stopping* the motor. These bits are *not* provided for one to merely specify the desired *mode* of stopping; in position mode operations, normal stopping is always smooth and occurs automatically at the end of the specified trajectory. Under exceptional circumstances it may be desired to manually intervene with the trajectory generation process to affect a premature stop. In velocity mode operations, however, the normal means of stopping *is* via bits 8 through 10 (usually bit 10). Bit 8 is set to logic one to stop the motor by turning off motor drive output (outputting the appropriate offset-binary code to apply zero drive to the motor); bit 9 is set to one to stop the motor abruptly (at maximum available acceleration, by setting the target position equal to the current position); and bit 10 is set to one to stop the motor smoothly by using the current user-programmed acceleration value. Bits 8 through 10 are to be used *exclusively*, only one bit should be a logic one at any time.

Bits 0 through 5 inform the LM628 as to whether any or all of the trajectory controlling parameters are about to be written, and whether the data should be interpreted as absolute or relative. The user may choose to update any or all (or none) of the trajectory parameters. Those chosen for updating are so indicated by logic one(s) in the corresponding bit position(s). Any parameter may be changed while the motor is in motion; however, if acceleration is changed then the next STT command must not be issued until the LM628 has completed the current move or has been manually stopped.

The data bytes specified by and immediately following the trajectory control word are written in pairs which comprise 16-bit words. Each data item (parameter) requires two 16-bit words; the word and byte order is most-to-least significant. The order of sending the parameters to the LM628 corresponds to the descending order shown in the above description of the trajectory control word; i.e., beginning with acceleration, then velocity, and finally position.

Acceleration and velocity are 32 bits, positive only, but range only from 0 (00000000 hex) to $[2^{30}] - 1$ (3FFFFFFF hex). The bottom 16 bits of both acceleration and velocity are scaled as fractional data; therefore, the least-significant integer data bit for these parameters is bit 16 (where the bits are numbered 0 through 31). To determine the coding for a given velocity, for example, one multiplies the desired velocity (in counts per sample interval) times 65,536 and converts the result to binary. The units of acceleration are counts per sample per sample. The value loaded for acceleration must not exceed the value loaded for velocity. Position is a signed, 32-bit integer, but ranges only from $-[2^{30}]$ (C0000000 hex) to $[2^{30}] - 1$ (3FFFFFFF Hex).

The required data is written to the primary buffers of a double-buffered scheme by the above described operations, it is not transferred to the secondary (working) registers until the STT command is executed. This fact can be used advantageously; the user can input numerous data ahead of their actual use. This simple pipeline effect can relieve potential host computer data communications bottlenecks, and facilitates easier synchronization of multiple-axis controls.

Before using LTRJ to issue a new acceleration value, a "motor off" command must first be executed (LTRJ command with bit 8 of the Trajectory Control Word set). This procedure is only necessary if the acceleration value is being changed.

STT COMMAND: STarT Motion Control

Command Code: 01 Hex
Data Bytes: None
Executable During Motion: Yes, if acceleration has not been changed

The STT command is used to execute the desired trajectory, the specifics of which have been programmed via the LTRJ command. Synchronization of multi-axis control (to within one sample interval) can be arranged by loading the required trajectory parameters for each (and every) axis and then simultaneously issuing a single STT command to all axes. This command may be executed at any time, unless the acceleration value has been changed and a trajectory has not been completed or the motor has not been manually stopped. If STT is issued during motion and acceleration has been changed, a command error interrupt will be generated and the command will be ignored.

Data Reporting Commands

The following seven LM628 user commands are used to obtain data from various registers in the LM628. Status, position, and velocity information are reported. With the exception of RDSTAT, the data is read from the LM628 data port after first writing the corresponding command to the command port.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Data Reporting Commands (Continued)

RDSTAT COMMAND: Read STATus Byte

Command Code:	None
Byte Read:	One
Data Range:	See Text
Executable During Motion:	Yes

The RDSTAT command is really not a command, but is listed with the other commands because it is used very frequently to control communications with the host computer. There is no identification code; it is directly supported by the hardware and may be executed at any time. The single-byte status read is selected by placing $\overline{CS}$, $\overline{PS}$ and $\overline{RD}$ at logic zero. See Table VII.

TABLE VII. Status Byte Bit Allocation

Bit Position	Function
Bit 7	Motor Off
Bit 6	Breakpoint Reached (Interrupt)
Bit 5	Excessive Position Error (Interrupt)
Bit 4	Wraparound Occurred (Interrupt)
Bit 3	Index Pulse Observed (Interrupt)
Bit 2	Trajectory Complete (Interrupt)
Bit 1	Command Error (Interrupt)
Bit 0	Busy Bit

Bit 7, the motor-off flag, is set to logic one when the motor drive output is off (at the half-scale, offset-binary code for zero). The motor is turned off by any of the following conditions: power-up reset, command RESET, excessive position error (if command LPES had been executed), or when command LTRJ is used to manually stop the motor via turning the motor off. Note that when bit 7 is set in conjunction with command LTRJ for producing a manual, motor-off stop, the actual setting of bit 7 does not occur until command STT is issued to affect the stop. Bit 7 is cleared by command STT, except as described in the previous sentence.

Bit 6, the breakpoint-reached interrupt flag, is set to logic one when the position breakpoint loaded via command SBPA or SBPR has been exceeded. The flag is functional independent of the host interrupt mask status. Bit 6 is cleared via command RSTI.

Bit 5, the excessive-position-error interrupt flag, is set to logic one when a position-error interrupt condition exists. This occurs when the error threshold loaded via command LPEI or LPES has been exceeded. The flag is functional independent of the host interrupt mask status. Bit 5 is cleared via command RSTI.

Bit 4, the wraparound interrupt flag, is set to logic one when a numerical "wraparound" has occurred. To "wraparound" means to exceed the position address space of the LM628, which could occur during velocity mode operation. If a wraparound has occurred, then position information will be in error and this interrupt helps the user to ensure position data integrity. The flag is functional independent of the host interrupt mask status. Bit 4 is cleared via command RSTI.

Bit 3, the index-pulse acquired interrupt flag, is set to logic one when an index pulse has occurred (if command SIP had been executed) and indicates that the index position register has been updated. The flag is functional independent of the host interrupt mask status. Bit 3 is cleared by command RSTI.

Bit 2, the trajectory complete interrupt flag, is set to logic one when the trajectory programmed by the LTRJ command and initiated by the STT command has been completed. Because of overshoot or a limiting condition (such as commanding the velocity to be higher than the motor can achieve), the motor may not yet be at the final commanded position. This bit is the logical OR of bits 7 and 10 of the Signals Register, see command RDSIGS below. The flag functions independently of the host interrupt mask status. Bit 2 is cleared via command RSTI.

Bit 1, the command-error interrupt flag, is set to logic one when the user attempts to read data when a write was appropriate (or vice versa). The flag is functional independent of the host interrupt mask status. Bit 1 is cleared via command RSTI.

Bit 0, the busy flag, is frequently tested by the user (via the host computer program) to determine the busy/ready status prior to writing and reading any data. Such writes and reads may be executed only when bit 0 is logic zero (not busy). Any command or data writes when the busy bit is high will be ignored. Any data reads when the busy bit is high will read the current contents of the I/O port buffers, not the data expected by the host. Such reads or writes (with the busy bit high) will not generate a command-error interrupt.

RDSIGS COMMAND: Read SIGnalS Register

Command Code:	0C Hex
Bytes Read:	Two
Data Range:	See Text
Executable During Motion:	Yes

The LM628 internal "signals" register may be read using this command. The first byte read is the more significant. The less significant byte of this register (with the exception of bit 0) duplicates the status byte. See Table VIII.

TABLE VIII. Signals Register Bit Allocation

Bit Position	Function
Bit 15	Host Interrupt
Bit 14	Acceleration Loaded (But Not Updated)
Bit 13	UDF Executed (But Filter Not yet Updated)
Bit 12	Forward Direction
Bit 11	Velocity Mode
Bit 10	On Target
Bit 9	Turn Off upon Excessive Position Error
Bit 8	Eight-Bit Output Mode
Bit 7	Motor Off
Bit 6	Breakpoint Reached (Interrupt)
Bit 5	Excessive Position Error (Interrupt)
Bit 4	Wraparound Occurred (Interrupt)
Bit 3	Index Pulse Acquired (Interrupt)
Bit 2	Trajectory Complete (Interrupt)
Bit 1	Command Error (Interrupt)
Bit 0	Acquire Next Index (SIP Executed)

Bit 15, the host interrupt flag, is set to logic one when the host interrupt output (Pin 17) is logic one. Pin 17 is set to logic one when any of the six host interrupt conditions occur (if the corresponding interrupt has not been masked). Bit 15 (and Pin 17) are cleared via command RSTI.

Bit 14, the acceleration-loaded flag, is set to logic one when acceleration data is written to the LM628. Bit 14 is cleared by the STT command.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Data Reporting Commands (Continued)

Bit 13, the UDF-executed flag, is set to logic one when the UDF command is executed. Because bit 13 is cleared at the end of the sampling interval in which it has been set, this signal is very short-lived and probably not very profitable for monitoring.

Bit 12, the forward direction flag, is meaningful only when the LM628 is in velocity mode. The bit is set to logic one to indicate that the desired direction of motion is "forward"; zero indicates "reverse" direction. Bit 12 is set and cleared via command LTRJ. The actual setting and clearing of bit 12 does not occur until command STT is executed.

Bit 11, the velocity mode flag, is set to logic one to indicate that the user has selected (via command LTRJ) velocity mode. Bit 11 is cleared when position mode is selected (via command LTRJ). The actual setting and clearing of bit 11 does not occur until command STT is executed.

Bit 10, the on-target flag, is set to logic one when the trajectory generator has completed its functions for the last-issued STT command. Bit 10 is cleared by the next STT command.

Bit 9, the turn-off on-error flag, is set to logic one when command LPES is executed. Bit 9 is cleared by command LPEI.

Bit 8, the 8-bit output flag, is set to logic one when the LM628 is reset, or when command PORT8 is executed. Bit 8 is cleared by command PORT12.

Bits 0 through 7 replicate the status byte (see Table VII), with the exception of bit 0. Bit 0, the acquire next index flag, is set to logic one when command SIP is executed, it then remains set until the next index pulse occurs.

RDIP COMMAND: ReaD Index Position

Command Code:	09 Hex
Bytes Read:	Four
Data Range:	C0000000 to 3FFFFFFF Hex
Executable During Motion:	Yes

This command reads the position recorded in the index register. Reading the index register can be part of a system error checking scheme. Whenever the SIP command is executed, the new index position minus the old index position, divided by the incremental encoder resolution (encoder lines times four), should always be an integral number. The RDIP command facilitates acquiring these data for host-based calculations. The command can also be used to identify/verify home or some other special position. The bytes are read in most-to-least significant order.

RDDP COMMAND: ReaD Desired Position

Command Code:	08 Hex
Bytes Read:	Four
Data Range:	C0000000 to 3FFFFFFF Hex
Executable During Motion:	Yes

This command reads the instantaneous desired (current *temporal*) position output of the profile generator. This is the "setpoint" input to the position-loop summing junction. The bytes are read in most-to-least significant order.

RDRP COMMAND: ReaD Real Position

Command Code:	0A Hex
Bytes Read:	Four
Data Range:	C0000000 to 3FFFFFFF Hex
Executable During Motion:	Yes

This command reads the current actual position of the motor. This is the feedback input to the loop summing junction. The bytes are read in most-to-least significant order.

RDDV COMMAND: ReaD Desired Velocity

Command Code:	07 Hex
Bytes Read:	Four
Data Range:	C0000001 to 3FFFFFFF
Executable During Motion:	Yes

This command reads the integer and fractional portions of the instantaneous desired (current *temporal*) velocity, as used to generate the desired position profile. The bytes are read in most-to-least significant order. The value read is properly scaled for numerical comparison with the user-supplied (commanded) velocity; however, because the two least-significant bytes represent *fractional* velocity, only the two most-significant bytes are appropriate for comparison with the data obtained via command RDRV (see below). Also note that, although the velocity *input* data is constrained to positive numbers (see command LTRJ), the data returned by command RDDV represents a *signed* quantity where negative numbers represent operation in the reverse direction.

RDRV COMMAND: ReaD Real Velocity

Command Code:	0B Hex
Bytes Read:	Two
Data Range:	C000 to 3FFF Hex, See Text
Executable During Motion:	Yes

This command reads the *integer* portion of the instantaneous actual velocity of the motor. The internally maintained fractional portion of velocity is not reported because the reported data is derived by reading the incremental encoder, which produces only integer data. For comparison with the result obtained by executing command RDDV (or the user-supplied input value), the value returned by command RDRV must be multiplied by 2^{16} (shifted left 16 bit positions). Also, as with command RDDV above, data returned by command RDRV is a *signed* quantity, with negative values representing reverse-direction motion.

RDSUM COMMAND: ReaD Integration-Term SUMmation Value

Command Code:	0D Hex
Bytes Read:	Two
Data Range:	00000 Hex to ± the Current Value of the Integration Limit
Executable During Motion:	Yes

This command reads the value to which the integration term has accumulated. The ability to read this value may be helpful in initially or adaptively tuning the system.

Typical Applications

Programming LM628 Host Handshaking (Interrupts)

A few words regarding the LM628 host handshaking will be helpful to the system programmer. As indicated in various portions of the above text, the LM628 handshakes with the host computer in two ways: via the host interrupt output (Pin 17), or via polling the status byte for "interrupt" conditions. When the hardwired interrupt is used, the status byte is also read and parsed to determine which of six possible conditions caused the interrupt.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Typical Applications (Continued)

When using the hardwired interrupt it is very important that the host interrupt service routine does not interfere with a command sequence which might have been in progress when the interrupt occurred. If the host interrupt service routine were to issue a command to the LM628 while it is in the middle of an ongoing command sequence, the ongoing command will be aborted (which could be detrimental to the application).

Two approaches exist for avoiding this problem. If one is using hardwired interrupts, they should be disabled at the host prior to issuing any LM628 command sequence, and re-enabled after each command sequence. The second approach is to avoid hardwired interrupts and poll the LM628 status byte for "interrupt" status. The status byte always reflects the interrupt-condition status, independent of whether or not the interrupts have been masked.

Typical Host Computer/Processor Interface

The LM628 is interfaced with the host computer/processor via an 8-bit parallel bus. *Figure 12* shows such an interface and a minimum system configuration.

As shown in *Figure 12*, the LM628 interfaces with the host data, address and control lines. The address lines are decoded to generate the LM628 $\overline{CS}$ input; the host address LSB directly drives the LM628 $\overline{PS}$ input. *Figure 12* also shows an 8-bit DAC and an LM12 Power Op Amp interfaced to the LM628.

LM628 and High Performance Controller (HPC) Interface

Figure 13 shows the LM628 interfaced to a National HPC High Performance Controller. The delay and logic associated with the $\overline{WR}$ line is used to effectively increase the write-data hold time of the HPC (as seen at the LM628) by causing the $\overline{WR}$ pulse to rise early. Note that the HPC CK2 output provides the clock for the LM628. The 74LS245 is used to decrease the read-data hold time, which is necessary when interfacing to fast host busses.

Interfacing a 12-Bit DAC

Figure 14 illustrates use of a 12-bit DAC with the LM628. The 74LS378 hex gated-D flip-flop and an inverter demultiplex the 12-bit output. DAC offset must be adjusted to minimize DAC linearity and monotonicity errors. Two methods exist for making this adjustment. If the DAC1210 has been socketed, remove it and temporarily connect a 15 kΩ resistor between Pins 11 and 13 of the DAC socket (Pins 2 and 6 of the LF356) and adjust the 25 kΩ potentiometer for 0V at Pin 6 of the LF356.

If the DAC is not removable, the second method of adjustment requires that the DAC1210 inputs be presented an all-zeros code. This can be arranged by commanding the appropriate move via the LM628, but with no feedback from the system encoder. When the all-zeros code is present, adjust the pot for 0V at Pin 6 of the LF356.

A Monolithic Linear Drive Using LM12 Power Op Amp

Figure 15 shows a motor-drive amplifier built using the LM12 Power Operational Amplifier. This circuit is very simple and can deliver up to 8A at 30V (using the LM12L/LM12CL). Resistors R1 and R2 should be chosen to set the gain to provide maximum output voltage consistent with maximum input voltage. This example provides a gain of 2.2, which allows for amplifier output saturation at ± 22V with a ± 10V input, assuming power supply voltages of ± 30V. The amplifier gain should not be higher than necessary because the system is non-linear when saturated, and because gain should be controlled by the LM628. The LM12 can also be configured as a current driver, see 1987 Linear Databook, Vol. 1, p. 2-280.

Typical PWM Motor Drive Interfaces

Figure 16 shows an LM18298 dual full-bridge driver interfaced to the LM629 PWM outputs to provide a switch-mode power amplifier for driving small brush/commutator motors. *Figure 17* shows an LM621 brushless motor commutator interfaced to the LM629 PWM outputs and a discrete device switch-mode power amplifier for driving brushless DC motors.

Incremental Encoder Interface

The incremental (position feedback) encoder interface consists of three lines: Phase A (Pin 2), Phase B (Pin 3), and Index (Pin 1). The index pulse output is not available on some encoders. The LM628 will work with both encoder types, but commands SIP and RDIP will not be meaningful without an index pulse (or alternative input for this input . . . be sure to tie Pin 1 high if not used).

Some consideration is merited relative to use in high Gaussian-noise environments. If noise is added to the encoder inputs (either or both inputs) and is such that it is not sustained until the next encoder transition, the LM628 decoder logic will reject it. Noise that mimics quadrature counts or persists through encoder transitions must be eliminated by appropriate EMI design.

Simple digital "filtering" schemes merely reduce susceptibility to noise (there will always be noise pulses longer than the filter can eliminate). Further, any noise filtering scheme reduces decoder bandwidth. In the LM628 it was decided (since simple filtering does not eliminate the noise problem) to not include a noise filter in favor of offering maximum possible decoder bandwidth. Attempting to drive encoder signals too long a distance with simple TTL lines can also be a source of "noise" in the form of signal degradation (poor risetime and/or ringing). This can also cause a system to lose positional integrity. Probably the most effective countermeasure to noise induction can be had by using balanced-line drivers and receivers on the encoder inputs. *Figure 18* shows circuitry using the DS26LS31 and DS26LS32.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บรรณานุกรม

Benjamin,c.kuo Automatic control systems , 5th ed.,pp 296-299,313-314. Prentice-Hall International inc., new jersey,1987.

อึ้งภากรณ์, ศ.ดร.วริทธ์ และ ถนัดงาน, รศ.ชาญ การออกแบบเครื่องกล, เล่มที่ 1,หน้า 304-322, บริษัทซีเอ็ดยูเคชั่นจำกัด, กรุงเทพฯ,2534.

เปรมปราณีรัช์, โยธิน ระบบเซอร์โว และอิเล็กทรอนิกส์คอนโทรลมอเตอร์, พิมพ์ครั้งที่ 1,หน้า 60-72,101-103,127-130,171-187. สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, กรุงเทพฯ, 2533.

เวียงสุขไพบูรณ์ ,ประเสริฐ "STD BUS ต้นทางสำหรับบัสมาตรฐานอุตสาหกรรม", เซมิคอนดัคเตอร์อิเลคทรอนิคส์,105(2534):79-82.

Pro-log Corporation. "System Data Book 1987",Pro-Log Corporation , Monterey,California,U.S.A.

Pro-Log Corporation. "Pro-Log Corporation Industial Computers",Pro-Log Corporation,Monterey,California, U.S.A,1991.

Pro-Log Corporation,"System2 Model60 User's Giude",Pro-log Corporation,Monterey,California,U.S.A.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ประวัติผู้เขียน

นายชายแดน ทองช้อย

เกิดเมื่อวันที่ 14 เมษายน พศ.2514 ที่ ต.เขาสามยอด อำเภอ เมือง จ.ลพบุรี

การศึกษา

-ระดับอนุบาล-ประถมศึกษา (พศ.2518-2524): โรงเรียนชุบศรศึกษา จ.ลพบุรี

-ระดับมัธยมศึกษาตอนต้น (พศ.2525-2527) : โรงเรียนพระนารายณ์ จ.ลพบุรี

-ระดับเตรียมอุดมศึกษา (พศ.2528-2530): โรงเรียนสาธิต วิทยาลัยครูเทพสตรี จ.ลพบุรี

-ระดับอุดมศึกษา(พศ.2531-2534) : สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหาร ลาดกระบัง กรุงเทพมหานคร

นาย ทวีศักดิ์ อัฐนาค

เกิดเมื่อวันที่ 8 กุมภาพันธ์ พศ.2512 ที่ ต.บัวขาว อ.กุฉินารายณ์ จ.กาฬสินธุ์

การศึกษา

-ระดับอนุบาล-ประถมศึกษา (พศ.2518-2524) :โรงเรียนบ้านบัวขาว วันครู (2500) อ.กุฉินารายณ์ จ.กาฬสินธุ์

-ระดับมัธยมศึกษาตอนต้น(พศ.2525-2527) :โรงเรียนบัวขาว อ.กุฉินารายณ์ จ.กาฬสินธุ์

-ระดับเตรียมอุดมศึกษา(พศ.2528-2530):โรงเรียนขอนแก่นวิทยายน อ.เมือง จ.ขอนแก่น

-ระดับอุดมศึกษา(2531-2534):คณะวิทยาศาสตร์ สถาบันเทคโนโลยี พระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง กรุงเทพมหานคร

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้