แขนกล

Robot Arm

โดย

นายธนพงศ์ เสนแสง
นายนพพร เชื่อมสุวรรณ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาอิเล็กทรอนิคส์เชิงกล
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2548

แขนกล

Robot Arm

โดย

นายธนพงศ์ เสนแสง

นายนพพร เชื่อมสุวรรณ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิคส์เชิงกล

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2548

ปริญญานิพนธ์ปีการศึกษา 2548

ภาควิชาวิศวกรรมระบบควบคุม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง **แขนกล**

Robot Arm

ผู้จัดทำ **นายธนพงศ์ เสนแสง 45010312**

นายนพพร เชื่อมสุวรรณ 45010370

.....อาจารย์ที่ปรึกษา

(ผศ. สุมิตร พนาอุดมทรัพย์)

แขนกล

Robot Arm

โดย

นายธนพงศ์ เสนแสง 45010312

นายนพพร เชื่อมสุวรรณ 45010370

อาจารย์ที่ปรึกษา

ผศ.สุมิตร พนาอุดมทรัพย์

บทคัดย่อ

ปัจจุบัน แขนกลได้ใช้กันอย่างแพร่หลายในวงการอุตสาหกรรมเป็นอย่างมาก ใช้ในการประกอบรถยนต์ เชื่อมวงจร เป็นต้น ข้อดีของการใช้แขนกลในการปฏิบัติงานคือความยืดหยุ่นและสามารถประยุกต์ใช้งานได้อย่างหลากหลาย

ปริญญานิพนธ์ฉบับนี้นำเสนอ "การสร้างแขนกล" โดยที่สามารถใช้งานได้อย่างแพร่หลาย ซึ่งแขนกลนี้จะประกอบด้วยกัน 2 ส่วน คือส่วนทางเครื่องกล และส่วนควบคุม โดยมีจุดประสงค์ที่สามารถประยุกต์ใช้งานได้หลากหลาย และมีความละเอียด แม่นยำสูง

แขนกล

Robot Arm

โดย

นายธนพงศ์ เสนแสง 45010312

นายนพพร เชื่อมสุวรรณ 45010370

อาจารย์ที่ปรึกษา

ผศ.สุมิตร พนาอุดมทรัพย์

Abstract

Nowaday, the robot – arm are generally used in many industrial such as automobile assemble, circuit welding, etc. The adventage of robot – arm are about their high precision and accuracy.

This thesis presents "Robot arm". The robot arm is used for general task which is designed in two main parts, mechanical and controller parts. The robot arm is flexible for modifying in any task and has high precision and accuracy.

กิตติกรรมประกาศ

เนื่องจากโปรเจคชิ้นนี้จะประสบความสำเร็จไม่ได้ถ้าขาดผู้อุปการะ และผู้ส่งเสริมที่มีส่วนทั้งทางตรงและทางอ้อมทางคณะผู้จัดทำขอกราบขอบคุณทางคณะวิศวกรรมศาสตร์ที่ได้เปิดโอกาสให้มีการฝึกงานซึ่งได้จุดประกายแนวคิดในการทำโปรเจคขึ้น

ขอกราบขอบคุณคุณพ่อคุณแม่ของคณะผู้จัดทำที่ได้ส่งเสริม และเลี้ยงดูอบรมบ่มนิสัยที่ดีให้เกิดขึ้น

ขอกราบขอบคุณ ผศ.สุมิตร พนาอุดมทรัพย์ เป็นอาจารย์ที่ปรึกษาโปรเจค ที่คอยสนับสนุน แนะแนวทางมุมมองที่แตกต่างออกไป ช่วยเหลือในการออกแบบ ให้ความรู้ แก้ปัญหาและดูแลการทำงานโปรเจคเป็นอย่างดีตลอดมา

ขอกราบขอบคุณ รศ.ดร.จงกล งามวิวิทย์ , ผศ.ถาวร เบญจนราสุทธิ์ และ ดร.ปรเมษฐ์ ประนยานันทน์ ที่ได้ให้ความรู้ทางด้าน Control

ขอกราบขอบคุณ รศ.ดร.วรพงศ์ ตั้งศรีรัตน์ ที่ได้ให้ความรู้ทางด้าน Electronics และ Sensor เป็นสามารถนำมาประยุกต์ใช้ได้เป็นอย่างดี

ขอกราบขอบคุณ รศ.ทวี เทศเจริญ ภาควิชาเครื่องกล ที่ได้ให้ความรู้ทางด้าน Machine Design

ขอกราบขอบคุณ อ.เทพจิตร์ เชยโภคา ที่ได้สอนการวาดแบบทางเครื่องกลโดยใช้ AutoCad จนสามารถประยุกต์ใช้ โปรแกรม Solid Work ได้เป็นอย่างดี

ขอกราบขอบคุณ อ.ธวัชชัย คำศรี ที่ได้ให้ความรู้ทาง Microcontroller และหลักการเขียนโปรแกรมภาษา C , แอซแซมบลี่ เบื้องต้น

ขอขอบคุณนาย ธาวัน อุทัยเจริญพงษ์ และนายเจษฎาพงษ์ วลัญโชดม ที่ได้สละเวลาและได้ให้ความรู้ในการใช้โปรแกรม Protel 99 SE จนสามารถนำความรู้มาประยุกต์ใช้งานได้

คณะผู้จัดทำ

นายธนพงศ์ เสนแสง

นายนพพร เชื่อมสุวรรณ

สารบัญ

	หน้า
บทคัดย่อ	I
กิตติกรรมประกาศ	III
สารบัญ	IV
สารบัญรูปภาพ	VI
สารบัญตาราง	VIII
บทที่ 1 บทนำ	
1.1 ความเป็นมาของโครงงาน	1
1.2 วัตถุประสงค์ในการดำเนินงาน	1
1.3 ขั้นตอนการศึกษาและจัดทำโครงงาน	1
บทที่ 2 ทฤษฎีและความรู้ที่เกี่ยวข้อง	
2.1 ค่าความปลอดภัย	3
2.2 คุณสมบัติทางกลของวัสดุ	4
2.3 ทฤษฎีการออกแบบเครื่องกล	6
2.3.1 ความเค้น	6
2.3.2 ความเครียด	8
2.3.3 โมเมนต์ดัด	8
2.4 Direct Kinematics The Arms Equation	9
2.5 ความรู้เกี่ยวกับ dsPIC30F2011 และโมดูลสำคัญที่ใช้ในโครงงานนี้	11
2.5.1 การใช้งานไทเมอร์ใน dsPIC	11
2.5.2 การทดลองสื่อสารข้อมูลอนุกรมโดยใช้โมดูล UART ใน dsPIC30F2011	12
2.5.3 การทดลองใช้งานโมดูล แปลงสัญญาณอะนาลอกเป็นดิจิตอลภายใน dsPIC	15
2.6 การสื่อสารข้อมูลอนุกรม RS-232C	17
2.7 Visual Basic กับการเขียนโปรแกรมส่งไฟล์ข้อมูลผ่าน Serial Port	20
บทที่ 3 หลักการออกแบบ	
3.1 แนวคิดในการออกแบบ	25
3.2 การออกแบบคำนวณตามหลักการและทฤษฎี	25

3.3 การคำนวณวัสดุแต่ละชิ้น 28
3.4 แนวคิดในการออกแบบระบบควบคุม 36
3.5 การออกแบบโปรแกรมในการควบคุมแขนกลจาก Computer 37
3.6 การออกแบบ Microcontroller ในการควบคุมแขนกล 44

บทที่ 4 ผลจากการออกแบบและทดลองใช้งาน

4.1 การประกอบชิ้นงานและผลจากการออกแบบ 51
4.2 การทดลองระบบควบคุม 52

บทที่ 5 วิจารณ์และสรุป

5.1 สรุปผลจากการทดลอง 55
5.2 ปัญหาที่พบ 56
5.3 ขอบเขตการดำเนินงานและแนวทางการพัฒนา 56
5.4 ประโยชน์ที่ได้รับจากโครงงาน 56

ภาคผนวก

ภาคผนวก ก.1 คุณสมบัติทางกลของอลูมิเนียมผสมเหนียวบางชนิด 57
ภาคผนวก ก.2 สูตรสำเร็จของคานและหน้าตัด 58
ภาคผนวก ข ความรู้ที่เกี่ยวกับ dsPIC30F2011 เบื้องต้น 59
ภาคผนวก ค.1 Code คำสั่งเพื่อควบคุมการทำงานของโปรแกรม 73
ภาคผนวก ค.2 Code คำสั่งปุ่ม Compile 84
ภาคผนวก ค.3 Code คำสั่งปุ่ม Run 87
ภาคผนวก ค.4 Code คำสั่งปุ่ม Set Origin 91
ภาคผนวก ค.5 โปรแกรม dsPIC 94
ภาคผนวก ค.6 ฟังก์ชั่นรับข้อมูลและสั่งงาน Servo Motor ตัวที่ 1 – 5 116
ภาคผนวก ค.7 ฟังก์ชั่นรับข้อมูลและสั่งงาน DC Motor ตัวที่ 6 – 7 117

เอกสารอ้างอิง 122

สารบัญภาพ

รูปที่	หน้า
2.1 แผนภาพความเค้น - ความเครียด	5
2.2 แรงชนิดต่างๆ	6
2.3 คานและแกนสะเทิน	9
2.4 รูปแสดง Link Parameters	10
2.5 Direct Kinematics	10
2.6 ไดอะแกรมการทำงานอย่างง่ายของโมดูล ADC ใน ไมโครคอนโทรลเลอร์ dsPIC30F2010	16
2.7 การต่ออุปกรณ์ DTE เข้ากับ DCE	18
2.8 การต่ออุปกรณ์ DTE เข้ากับ DTE	19
2.9 รูปแบบของสัญญาณข้อมูลอนุกรมที่ใช้ในการสื่อสารข้อมูลอนุกรม	19
3.1 การออกแบบ Kinematics ของแขนกล	25
3.2 จุดหมุน A	28
3.3 จุดหมุน A(2)	29
3.4 จุดหมุน B	29
3.5 จุดหมุน B(2)	30
3.6 จุดหมุน B(3)	31
3.7 จุดหมุน C	31
3.8 จุดหมุน D	32
3.9 การหาค่า K ของสปริง	34
3.10 ส่วนประกอบของแขนกลแต่ละชิ้น	35
3.11 แนวคิดการออกแบบระบบควบคุม	36
3.12 หน้าต่างโปรแกรม Visual Basic	37
3.13 หน้าต่างโปรแกรมควบคุมแขนกล	40
3.14 พื้นที่ป้อนโปรแกรม	41
3.15 ส่วนแสดงผล	43
3.16 กราฟ Angle / dat	46
3.17 กราฟ Response / Time	47
3.18 อุปการณ์ต่างๆ ภายในลายวงจร	48

3.19 ลายวงจรของกล่อง Controller 49

3.20 ลายวงจร PCB 50

4.1 รูปของแขนกลที่ประกอบเสร็จเรียบร้อยแล้ว 51

4.2 รูปของฐานที่ประกอบเสร็จเรียบร้อยแล้ว 52

4.3 ภาพการทดลอง 1 53

4.4 ภาพการทดลอง 2 53

4.5 ภาพการทดลอง 3 54

4.6 ทดลองจับวัสดุอื่น 54

5.1 แขนกล 55

ข.1 ไดอะแกรมการทำงานและส่วนประกอบทั้งหมดของ dsPIC30F210 61

ข.2 การจัดขาใช้งานของไมโครคอนโทรลเลอร์ dsPIC30F2010 62

ข.3 แสดงรายละเอียดของอินเตอร์รัปต์เวกเตอร์หลักของ 70
ไมโครคอนโทรลเลอร์ dsPIC

สารบัญตาราง

ตารางที่	หน้า
2.1 ค่าความปลอดภัย	4
2.2 Link Parameters	10
2.3 แสดงขาสัญญาณของ RS-232 ทั้งแบบ 9 และ 25ขา	18
2.4 อัตราบอดและช่วงเวลาของแต่ละบิตข้อมูลในการสื่อสารข้อมูลอนุกรม	20
3.1 ตารางพารามิเตอร์	25
3.2 ตารางการควบคุมแขนกลจาก Computer	37
ก.1 คุณสมบัติทางกลของอลูมิเนียมผสมเหนียวบางชนิด	57
ก.2 สูตรสำเร็จของคานและหน้าตัด	58
ข.1 แสดงหมายเลขอินเตอร์รัปต์และชื่อของแหล่งกำเนิดอินเตอร์รัปต์ใน ไมโครคอนโทรลเลอร์ dsPIC	68

บทที่ 1
บทนำ

1.1 ประวัติความเป็นมาของโครงงาน

ในปัจจุบันพบว่าในอุตสาหกรรมได้มีการนำเครื่องมือ และเครื่องจักรกลอัตโนมัติมาใช้อย่างแพร่หลาย ซึ่งเครื่องมือและเทคโนโลยีเหล่านี้ได้มีบทบาทต่อมนุษย์ โดยใช้เป็นเครื่องมือทุ่นแรง ใช้ในงานอันตราย รวมถึงใช้ในการทำงานที่มีความละเอียดสูง ซึ่งเครื่องจักรเหล่านี้มีหลายประเภทเช่น เครื่องCNC,แขนกล ฯลฯ และในอนาคตข้างหน้าเครื่องจักรกลที่จะเข้ามามีบทบาทสำคัญต่ออุตสาหกรรมไม่ว่าภาคอุตสาหกรรมขนาดเล็กหรือขนาดใหญ่ ก็คือ แขนกล เพราะสามารถทำงานได้หลากหลายรูปแบบ เพียงเปลี่ยนอุปกรณ์ต่อที่หัวของแขนกล จึงสามารถทำการเชื่อม,ตัด,จับ,ดัด ได้ตามต้องการ รวมทั้งตัวแขนกลมีหลากหลายขนาดตั้งแต่ขนาดเล็กและใหญ่ ซึ่งปัจจุบันมีใช้ในอุตสาหกรรมการประกอบชิ้นส่วน เช่น โรงงานประกอบรถยนต์ อุตสาหกรรมประกอบเครื่องจักร

คณะผู้จัดทำ จึงได้คิดที่จะทำโครงงานเกี่ยวกับการสร้างแขนกลขึ้นมาเองและออกแบบระบบควบคุมของแขนกลให้ทำงานได้อย่างอัตโนมัติ เพื่อเป็นการพัฒนาเทคโนโลยีทางทั้งนี้ ได้มีการออกแบบและทำแขนกลขนาดเล็กขึ้นมาเพื่อเป็นต้นแบบในการออกแบบวงจรและเขียนโปรแกรมการทำงานต่อไป

โครงงานนี้เป็นโครงงานในการสร้างแขนกลโดยในส่วนแรกเราได้ศึกษาและดำเนินการโดยสร้างแขนกล ซึ่งนำเอาหลักการและทฤษฎีต่างๆ มาใช้ และในส่วนนี้เราได้ทำการศึกษาและทำการออกแบบระบบควบคุมและเขียนโปรแกรมในการควบคุม

1.2 วัตถุประสงค์ในการดำเนินงาน

1. เพื่อศึกษาการออกแบบวัสดุและอุปกรณ์ในการสร้างแขนกล
2. เพื่อศึกษาการคำนวณแรงของมอเตอร์แต่ละส่วนว่าทำงานได้หรือไม่
3. เมื่อประกอบแขนกลเสร็จ ก็จะออกแบบระบบควบคุมและเขียนโปรแกรมการทำงานได้และสามารถพัฒนาปรับปรุงต่อไป
4. ในส่วนหลังเราได้ทำการออกแบบโปรแกรมและระบบควบคุม รวมทั้งทำวงจรในการควบคุมแขนกล และเมื่อทำระบบควบคุมและวงจรเสร็จแล้ว จะทำการทดลองควบคุมแขนกลโดย ใช้โปรแกรมที่สร้างขึ้นมา เขียนให้แขนกลที่ได้สร้างขึ้นมานั้นทำงานได้จริง

1.3 ขั้นตอนในการศึกษาและจัดทำโครงงาน

คณะผู้จัดทำได้เริ่มโดยการออกแบบและคิดหารูปแบบของลักษณะแขนกลว่าจะมีความอิสระในการหมุนเท่าไร ตามวัตถุประสงค์ที่ใช้งาน โดยตอนปลายของแขนกลนั้นสามารถเปลี่ยนวัตถุประสงค์ที่จะใช้งานได้ โดยเปลี่ยนอุปกรณ์ที่ปลายแขนกล จากนั้นทำการศึกษา ทฤษฎีเกี่ยวกับการออกแบบทาง Mechanic ซึ่งจะสามารถออกแบบแขนกลได้อย่างถูกต้องและมีหลักการและได้ทำการเขียนแบบโดยใช้โปรแกรม Solid Works เพื่อสร้าง Model ให้เห็นภาพจริงของแขนกล ต่อมาทำการสร้างชิ้นส่วนต่างๆ ตามที่ออกแบบไว้แต่ละชิ้น เมื่อสร้างชิ้นส่วนครบตามต้องการจึงทำการประกอบชิ้นส่วนทั้งหมดเป็น แขนกล

จากการประกอบแขนกลที่เสร็จสมบูรณ์นี้ พบว่ามีความแข็งแรงและตรงตามแบบที่คิดไว้ ทำให้เราสามารถออกแบบระบบควบคุมวงจรการทำงาน รวมทั้งเริ่มเขียนโปรแกรมที่จะใช้ควบคุมแขนกลที่เสร็จแล้ว เพื่อที่จะทำการชดเชยพารามิเตอร์ต่างๆได้อย่างถูกต้อง ซึ่งจะชดเชยโดยการออกแบบวงจรหรือการเขียนโปรแกรมต่อไป คณะผู้จัดทำได้เริ่มโดยการวางแผนว่าแขนกลนั้นมีระบบอะไรที่จะต้องควบคุมบ้างในที่นี้ มี Servo motor อยู่ 5 ตัว และ DC motor 2 ตัว รวมทั้งโพเทนทิออมิเตอร์อีก 2 ตัว ที่ใช้ในการ Feedback เช็คมุมของ DC motor อีก 2 ตัว จากนั้นก็ได้คิดที่จะใช้ Microcontroller ชิ้นใดในการควบคุม ในที่นี้คณะผู้จัดทำเลือก dsPic30F2011 ในการ Control และจะควบคุมแขนกลผ่าน Computer โดยจะต้องสร้างโปรแกรมในการควบคุมขึ้นมาเอง จากแนวคิดข้างต้นคณะผู้จัดทำได้เริ่มศึกษาหาข้อมูลในเรื่อง Microcontroller dsPic และระบบ Drive motor รวมทั้งการรับส่งข้อมูลผ่าน Serial Port โดย RS 232 เขียนโปรแกรมควบคุม Controller โดยใช้ ภาษา C และเขียนโปรแกรมที่ใช้ควบคุมแขนกลใน Computer โดยใช้ Visaul Basic เมื่อทำการเขียนโปรแกรมเสร็จแล้วและได้ทดลองบนบอร์ดทดลอง จากนั้นก็ทำการประกอบวงจรทั้งหมด ลงแผ่นปริ้น และทำการทดลองควบคุมแขนกลจริงเพื่อตรวจสอบว่าแขนกลทำงานได้ตามการควบคุมหรือไม่ ถ้าไม่ได้ก็ทำการตรวจสอบและแก้ไขต่อไป

เมื่อแขนกลทำงานได้ตามโปรแกรมควบคุม ก็ได้ทำโปรแกรมตัวอย่างในการใช้งานจริงเพื่อตรวจดูว่าแขนกลทำงานได้จริง

บทที่ 2
ทฤษฎีและความรู้ที่เกี่ยวข้อง

2.1 ค่าความปลอดภัย

โดยทั่วไปแล้วค่าความปลอดภัย หมายถึง ตัวเลขที่นำไปหารค่าความต้านแรงดึงหรือความต้านแรงดึงครากของวัสดุ เพื่อให้ได้ความเค้นใช้งาน (working stress) ในชิ้นส่วนที่กำลังออกแบบ ซึ่งเรียกสั้นๆว่า ความเค้นออกแบบ(design stress) หรือความเค้นใช้งาน ตัวอย่างเช่น เหล็กกล้า ชนิดหนึ่งมีความต้านแรงดึง และความต้านแรงดึงครากเท่ากับ 700 MN/m^2 และ 420 MN/m^2 ตามลำดับและในการออกแบบชิ้นงานหนึ่งโดยใช้เหล็กกล้าชนิดนี้ ผู้ออกแบบคิดว่าตามลักษณะการใช้งานแล้วความเค้นใช้งานควรจะไม่เกิน 140 MN/m^2 ฉะนั้นค่าความปลอดภัยเมื่อถือความต้านแรงดึงเป็นหลักคือ

$$Nu = \frac{700}{140} = 5$$

และค่าความปลอดภัยเมื่อถือความต้านแรงดึงครากเป็นหลักคือ

$$Ny = \frac{420}{140} = 3$$

ในกรณีที่ได้มีการกำหนดขนาดของชิ้นงานมาแล้ว ค่าความปลอดภัยของชิ้นงานนั้นคือ

$$Nu = \frac{\text{ความต้านแรงดึง}}{\text{ความเค้นที่คำนวณได้}} \qquad (2.1)$$

$$Ny = \frac{\text{ความต้านแรงดึงคราก}}{\text{ความเค้นที่คำนวณได้}} \qquad (2.2)$$

สำหรับปัญหาที่ไม่เป็นแบบเชิงเส้น (nonlinear) เช่น ท่อโลหะ หรือเสาที่อาจจะเสียหายเนื่องมาจากการโก่งงอ จะต้องใช้ค่าความปลอดภัยกับแรงที่มากระทำโดยตรงแทนที่จะใช้กับความเค้นทั้งนี้เพราะในปัญหาแบบไม่ใช่เชิงเส้น ความเค้นที่เกิดขึ้นในชิ้นงานอาจจะมิได้แปรผันโดยตรงกับแรง ในกรณี เช่นนี้

$$N = \frac{\text{แรงที่ทำให้แตกหัก}}{\text{แรงที่ใช้ออกแบบ}} \qquad (2.3)$$

ค่าความปลอดภัยที่จะเลือกใช้ขึ้นอยู่กับตัวประกอบจำนวนมากดังนี้

2.1.1 ชนิดของแรงที่มากระทำต่อชิ้นงาน ว่าเป็นแรงที่จัดอยู่ในประเภทอยู่นิ่งหรือ

เปลี่ยนแปลง ขนาดอยู่ตลอดเวลาขณะใช้งาน

2.1.2 ลักษณะการใช้งานของชิ้นงานว่าเกี่ยวข้องกับการที่อาจจะสูญเสียชีวิตหรือทรัพย์สินจำนวนมากหรือไม่

2.1.3 น้ำหนักของชิ้นงานว่ามีความจำเป็นที่จะต้องเบาที่สุดหรือไม่ เช่น ชิ้นส่วนสำหรับเครื่องบินเป็นต้น ในกรณีเช่นนี้การใช้ค่าความปลอดภัยจะต้องพิจารณาอย่างละเอียดเป็นพิเศษ

2.1.4 จำนวนของชิ้นงานที่จะผลิตออกมา ถ้าผลิตครั้งละมากๆ ควรระมัดระวังในการใช้ค่าความปลอดภัยที่ไม่สูงจนเกินไป ทั้งนี้เพื่อที่จะให้ประหยัดวัสดุได้มากที่สุด

2.1.5 เนื้อวัสดุที่ผลิตออกมาอาจไม่สม่ำเสมอกัน ทำให้ความสามารถในการรับแรงต่างกัน

สำหรับผู้ที่มีความชำนาญในการออกแบบน้อย ก็อาจจะใช้ค่าที่แนะนำไว้ในตารางที่ 2.1 เป็นแนวทางในการคำนวณออกแบบได้

ตารางที่ 2.1 ค่าความปลอดภัย

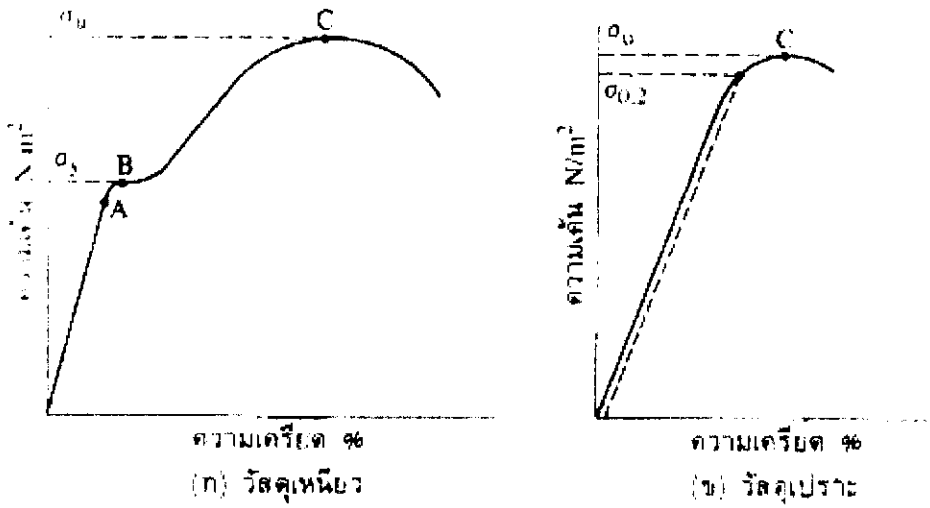
ชนิดของแรง	เหล็กเหนียวและโลหะเหนียว		เหล็กหล่อและโลหะเปราะ
	N_y	N_u	N_u
แรงอยู่นิ่ง	1.5-2	3-4	5-6
แรงซ้ำทิศทางเดียวหรือ แรงกระแทกเล็กน้อย	3	6	7-8
แรงซ้ำสองทิศทางหรือ แรงกระแทกเล็กน้อย	4	8	10-12
แรงกระแทกอย่างหนัก	5-7	10-15	15-20

ค่าความปลอดภัยสำหรับแรงซ้ำทิศทางเดียว (repeated, one direction) หรือแรงซ้ำสองทิศทาง (repeated and reversed) ที่ให้ไว้ในตารางที่ 2.1 หมายถึงค่าความปลอดภัยที่จะนำไปใช้เมื่อสมมติให้เป็นแรงนิ่ง (dead load) ในขณะออกแบบสำหรับการวิเคราะห์ปัญหาโดยคิดละเอียดลงไปถึงลักษณะการเปลี่ยนแปลงของแรง และความทนของวัสดุต่อแรงที่เปลี่ยนอยู่เสมอนี้จะใช้ค่าความปลอดภัยแตกต่างไป

2.2 คุณสมบัติทางกลของวัสดุ

ในการออกแบบชิ้นส่วนเครื่องจักรกลจะต้องคำนวณหาขนาดของชิ้นส่วนต่างๆ โดยคำนึงถึงคุณสมบัติทางกลของวัสดุเป็นสำคัญ ซึ่งจะพบว่ามีชื่อเรียกต่างๆ อยู่มาก ดังนั้นเพื่อความสะดวกในการอ้างอิงต่อไปจึงจะได้นิยามความหมายของชื่อต่างๆ ไว้พอสังเขปดังต่อไปนี้

ความต้านทานแรงดึงอัลติเมต (ultimate tensile strength), σ_u เป็นความเค้นสูงสุดที่วัสดุจะรับได้ ซึ่งคำนวณได้จากการนำแรงที่ใช้ดึงวัสดุตัวอย่างหารด้วยพื้นที่หน้าตัดเดิม และแทนด้วยจุด C บนกราฟความเค้น-ความเครียดในรูปที่ 2.1 ในบางครั้งอาจเรียกให้สั้นลงได้ว่าความต้านแรงดึง (tensile strength)



(ก) วัสดุเหนียว (ข) วัสดุเปราะ

รูปที่ 2.1 แผนภาพความเค้น – ความเครียด

ขีดจำกัดความเป็นสัดส่วน (proportional limit) เป็นค่าความเค้นค่าสุดท้ายที่เป็นสัดส่วนโดยตรงกับความเครียดดังจุด A ในรูปที่ 2.1(ก) เมื่อพ้นจุดนี้ไปแล้วกราฟจะเป็นเส้นโค้งในทางปฏิบัติจะหาจุดนี้ยากมาก ฉะนั้นในกาคำนวณจึงนิยมใช้ความต้านแรงดึงคราก (yield strength) แทน

ขีดจำกัดความยืดหยุ่น (elasic limit) อยู่ระหว่างจุด A และ B ในรูปที่ 2.1 (ก) เป็นจุดสุดท้ายที่เมื่อเอาแรงภายนอกออกแล้วชิ้นตัวอย่างทดสอบจะกลับมามีขนาดเท่าเดิม กราฟในช่วง AB นี้จะมีความโค้งเล็กน้อย

ความต้านแรงดึงคราก (yield strength) , σ_y เป็นจุดที่ชิ้นทดสอบยืดออกได้มากโดยที่เพิ่มแรงอีกเล็กน้อยเท่านั้น (หรือไม่ได้เพิ่ม) ซึ่งแทนด้วยจุด B หรือเรียกว่าจุดคราก ความเค้นที่จุดนี้ถือเป็นหลักในการออกแบบทั่วๆ ไป สำหรับวัสดุที่ไม่มีจุดคราก เช่น เหล็กหล่อ ก็อาจใช้ความต้านแรงดึงมาใช้แทนความต้านแรงดึงคราก โดยการลากเส้นขนานกับส่วนที่เป็นเส้นตรงของกราฟตามเปอร์เซ็นต์ของความเครียดที่ต้องการดังรูปที่ 2.1(ข) โดยทั่วไปแล้วมักจะใช้ 0.2% และเพื่อแสดงความแตกต่างระหว่างความเค้นที่จุดยืดถาวรกับความต้านแรงดึงคราก จึงใช้สัญลักษณ์แทนความเค้นที่จุดยืดถาวร 0.2% ว่า $\sigma_{0.2}$ หรืออาจเรียกสั้นๆ ว่าความเค้นพิสูจน์ 0.2%

ยังส์โมดุลัส (Yong's modulus) หรือโมดุลัสความยืดหยุ่น (modulus of elasticity) เป็นอัตราส่วนระหว่างความเค้นต่อความเครียดในส่วนที่กราฟเป็นเส้นตรง

โมดุลัสเฉือน (shear modulus) หรือโมดุลัสความแข็งแกร่ง (modulus of rigidity) ในการทดสอบชิ้นส่วนโดยใช้แรงเฉือนแล้วเขียนกราฟระหว่างความเค้นเฉือน (shear stress) กับความเครียดเฉือน (shear strain) ก็จะได้กราฟลักษณะเดียวกับการทดสอบแรงดึง อัตราส่วนระหว่างความเค้นเฉือนต่อความเครียดเฉือนในส่วนที่กราฟเป็นเส้นตรงเรียกว่า โมดุลัสเฉือน

ในการใช้คำนวณในระบบหน่วยเอสไอให้ทำการแปลงหน่วยดังนี้

$$1 \text{ ksi} = 6.895 \text{ MN/m}^2 \text{ หรือ } \text{N/mm}^2 \qquad (2.4)$$

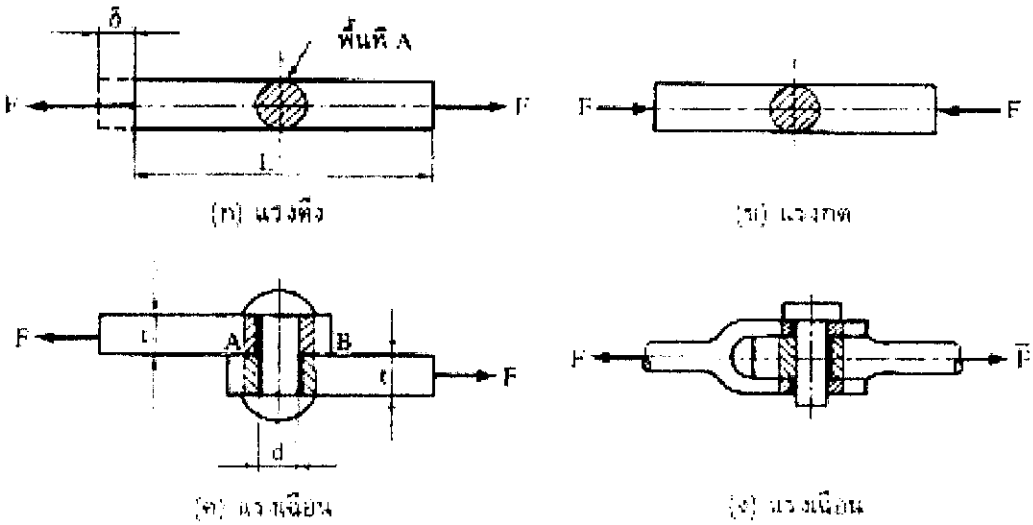
คุณสมบัติที่สำคัญอีกอย่างหนึ่งคือ ความต้านแรงเฉือนคราก (yield strength in shear) ซึ่งใช้ในการออกแบบเสมอ แต่มิได้ให้ไว้ในตารางดังที่กล่าวมาแล้ว แต่อย่างไรก็ตามให้ใช้ค่าประมาณจากตารางได้ดังนี้คือ

$$\tau_y = 0.6\sigma_y \qquad (2.5)$$

2.3 ทฤษฎีการออกแบบเครื่องกล

2.3.1 ความเค้น

นิยามของความเค้นคือ แรงหารด้วยพื้นที่หน้าตัดที่รับแรง ความเค้นอย่างง่าย (simple stress) มีอยู่ 3 ชนิดคือ ความเค้นดึง ความเค้นกด และความเค้นเฉือน



รูปที่ 2.2 แรงชนิดต่างๆ

พิจารณารูปที่ 2.2 (ก) และ(ข) ซึ่งเป็นท่อนโลหะกลมอยู่ภายใต้แรงดึง และแรงกด F ตามลำดับ ความเค้นดึงและความเค้นกดคือ

$$\sigma t = \frac{F}{A} \qquad (2.6)$$

$$\sigma c = \frac{F}{A} \qquad (2.7)$$

ในกรณีที่แผ่นโลหะยึดติดกันด้วยหมุดย้ำ ดังรูปที่ 2.2(ค) ตัวหมุดย้ำอาจจะขาดเนื่องจากแรงเฉือนกระทำที่หน้าตัด AB ถ้าพื้นที่หน้าตัด ของหมุดย้ำเท่ากับ A ความเค้นเฉือนในหน้าตัดของหมุดย้ำคือ

$$\tau = \frac{F}{A} \qquad (2.8)$$

ถ้าหน้าตัดของชิ้นงานที่รับแรงเฉือนมีมากกว่าหนึ่งแห่ง ดังเช่นในรูปที่ 2.2 (ง) ซึ่งมีสองแห่งพื้นที่หน้าตัดที่รับแรงคือ 2A ในกรณีเช่นนี้เรียกว่า หยุดย้ำรับแรงเฉือนคู่ (double shear) เพราะฉะนั้นความเค้นเฉือนที่เกิดขึ้นในหน้าตัดของหมุดย้ำนี้จะเท่ากับ

$$\tau = \frac{F}{2A} \qquad (2.9)$$

พิจารณาหมุดย้ำในรูปที่ 2.2 (ค) จะเกิดการอัดกันระหว่างด้านข้างของตัวหมุดย้ำกับแผ่นโลหะด้วยความเค้นที่ผิวโลหะที่สัมผัสกันนี้ไม่สม่ำเสมอนัก ในทางปฏิบัติจึงหาความเค้นกดนี้โดยใช้พื้นที่ภาพฉาย (projected area) ของส่วนที่อัดกันอยู่ แทนการใช้พื้นที่จริงรอบหมุดย้ำ และมีชื่อเรียกว่าความเค้น (bearing stress) ถ้าหมุดย้ำมีขนาดเส้นผ่านศูนย์กลาง d ความเค้นอัดนี้คือ

$$\sigma c = \frac{F}{Dt} \qquad (2.10)$$

2.3.2 ความเครียด

ความเครียด (strain) ε หมายถึงอัตราส่วนระหว่างส่วนที่ยืดหรือหดของชิ้นงานกับความยาวเดิมจากรูป 3.1(ก) ส่วนที่ยืดออกเนื่องจากแรงดึง F เท่ากับ σ เพราะฉะนั้นความเครียดนี้จะเท่ากับ

$$\varepsilon = \frac{\sigma}{L} \qquad (2.11)$$

จากกฎของฮุค

$$\sigma = E\varepsilon \qquad (2.12)$$

แต่ σ=F/A ฉะนั้นเมื่อแทนค่า ε จากสมการที่ (1) ลงในสมการที่ (2) จะได้ว่า

$$\sigma = \frac{FL}{AE} \qquad (2.13)$$

จากกลศาสตร์วัสดุยังทำให้ทราบความสัมพันธ์ระหว่างยังส์โมดุลัสและโมดุลัสเฉือนอีกคือ

$$G = \frac{E}{2(1+\upsilon)} \qquad (2.14)$$

โดยที่ υ เป็นอัตราส่วนปัวซอง(Poisson's ratio)

2.3.3 โมเมนต์ดัด

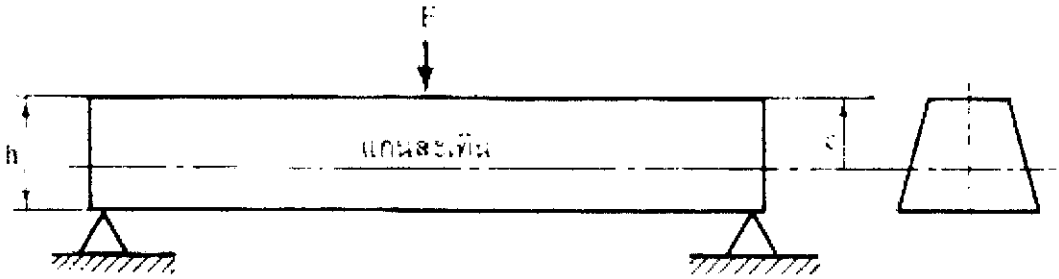
ชิ้นส่วนเครื่องจักรกลจำนวนมากรับแรงในแนวดิ่งลักษณะเช่นเดียวกับคานทั่วไป ฉะนั้นจึงใช้ความเค้นดัด (bending stress) และการยุบตัว (deflection) เป็นข้อจำกัดในการออกแบบ ความเค้นดัดสูงสุดเกิดที่ผิวนอกสุดของคาน ณ ตำแหน่งที่โมเมนต์ดัด (bending moment) มีค่าสูงสุด ซึ่งคำนวณได้จากสมการ

$$\sigma b = \frac{Mc}{I} \qquad (2.15)$$

โดยที่ M คือโมเมนต์ดัด

C คือระยะจากแกนสะเทิน(neutral axis) ไปยังผิวนอกสุด ดังรูปที่ 2.3

I คือโมเมนต์ความเฉื่อยของพื้นที่



รูปที่ 2.3 คานและแกนสะเทิน

โดยทั่วไปแล้วความเค้นเฉือนที่เกิดขึ้นในคานจะมีค่าน้อยมาก จนกระทั่งไม่ต้องนำมาคิดในการออกแบบได้ แต่ถ้าคานสั้นและมีหน้าตัดสูงมาก ความเค้นเฉือนก็อาจจะมีค่ามากได้ สำหรับคานที่มีพื้นที่หน้าตัดเป็นรูปสี่เหลี่ยมผืนผ้า ความเค้นเฉือนสูงสุดจะเกิดที่แกนสะเทิน และมีค่า 1.5 เท่าของความเค้นเฉือนเฉลี่ยหรือเท่ากับ

$$\tau = \frac{3V}{2A} \qquad (2.16)$$

สำหรับหน้าตัดกลม

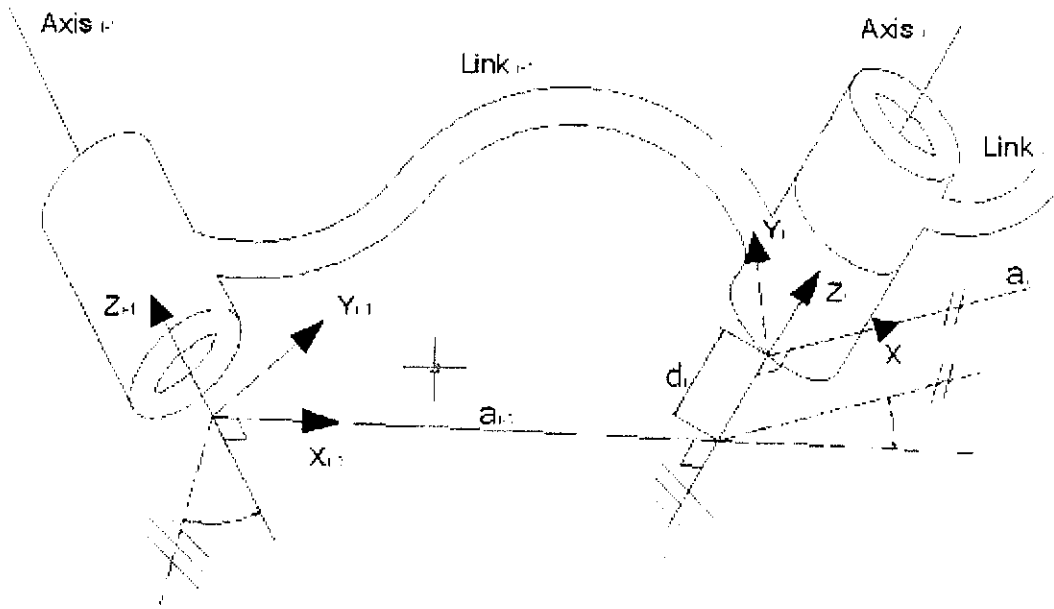
$$\tau = \frac{4V}{3A} \qquad (2.17)$$

โดยที่ V คือแรงเฉือนสูงสุด

A คือพื้นที่หน้าตัด

2.4 Direct Kinematics : The Arm Equation

Kinematics นี้ก็คือการหาความสัมพันธ์ระหว่างชิ้นส่วนของวัตถุ โดยวัตถุทั้ง 2 จะยึดติดกันอยู่โดยใช้จุดหมุน (Revolute) หรือสไลด์ไปตามแนวแกนของอีกวัตถุหนึ่ง (Prismatic) โดยวัตถุที่ยึดติดกันอยู่นี้เมื่อนำมาต่อกันไปเรื่อยๆ จะทำให้เราต้องหาวิธีหาความสัมพันธ์ของชิ้นส่วนที่นำมาต่อกัน



รูปที่ 2.4 รูปแสดง Link Parameters

จากรูปจะมี Link Parameters ดังนี้

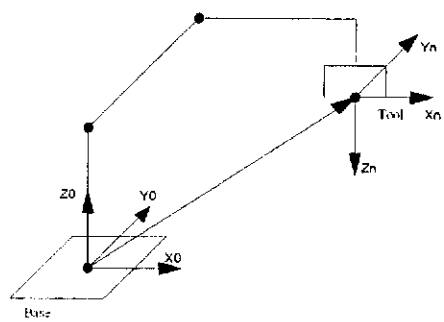
ตารางที่ 2.2 Link Parameters

Operation	Description
1	Rotate L_{k-1} about Z^{k-1} by θ_k
2	Translate L_{k-1} along Z^{k-1} by d_k
3	Translate L_{k-1} along X^{k-1} by a_k
4	Rotate L_{k-1} about X^{k-1} by a_k

พบว่าจะหาความสัมพันธ์ของชิ้นส่วนต่าง ๆ ที่นำมา link กันได้ตามสมการดังนี้

ปลาย(tool)

T ฐาน(base) $=$ $\begin{bmatrix} R & P \\ 0\ 0\ 0 & 1 \end{bmatrix}$



รูปที่ 2.5 Direct Kinematics

โดย R คือเมทริกขนาด 3*3 ที่บ่งบอกทิศทางของการการหมุน และ P คือเมทริก 3*1 เป็นจุดบนแกนอ้างอิง

จากสมการข้างบนจะทำให้สามารถหาความสัมพันธ์ระหว่างจุดปลาย (tool) กับจุดที่ฐาน (base) และหลักการนี้สามารถนำไปใช้กับการหาความสัมพันธ์ของวัตถุที่เชื่อมต่อกัน (link) ได้

2.5 ความรู้ที่เกี่ยวกับ dsPIC30F2011 และโมดูลสำคัญที่ใช้ในโครงงานนี้

dsPIC คือชื่อของไมโครคอนโทรลเลอร์ 16 บิตจาก Microchip Technology Inc. ผู้ผลิตไมโครคอนโทรลเลอร์ PIC ซึ่งรู้จักกันเป็นอย่างดีในแวดวงนักพัฒนาระบบไมโครคอนโทรลเลอร์ โดย Microchip Technology ได้กำหนดชื่ออย่างเป็นทางการสำหรับไมโครคอนโทรลเลอร์อนุกรมใหม่นี้ว่า Digital Signal Controller หรือ DSC นั่นหมายความว่า dsPIC เป็นไมโครคอนโทรลเลอร์ที่ได้รับ การออกแบบมาเป็นพิเศษเพื่องานประมวลผลสัญญาณดิจิตอล สำหรับสร้างระบบควบคุมอัตโนมัติที่มีความสามารถสูง

2.5.1 การใช้งานไทเมอร์ใน dsPIC

ไทเมอร์/เคาน์เตอร์หลักใน dsPIC มี 5 ตัวคือ ไทเมอร์ 1 ถึงไทเมอร์ 5 โดยแต่ละตัวมีขนาด 16 บิต สำหรับไทเมอร์ 2 และ 3 กับไทเมอร์ 4 และ 5 สามารถทำงานร่วมกันเป็นไทเมอร์ขนาด 32 บิต เมื่อไทเมอร์แต่ละตัวทำงานแยกกันสามารถกำหนดการทำงานได้อีก 3 แบบตามลักษณะของฐานเวลา คือ ฐานเวลาแบบ A,B และ C ซึ่งจะได้กล่าวถึงในลำดับถัดไป

2.5.1.1 คุณสมบัติของไทเมอร์

ใน dsPIC30F2011 มีไทเมอร์/เคาน์เตอร์ขนาด 16 บิตให้ใช้งานรวม 3 ตัว คือ ไทเมอร์ 1 (T1), ไทเมอร์ 2 (T2) และไทเมอร์ 3 (T3)

- **คุณสมบัติของไทเมอร์ 1**
  - รีจิสเตอร์ตัวนับความละเอียด 16 บิต
  - ทำงานได้ทั้งแบบซิงโครนัสและอะซิงโครนัสเคาน์เตอร์
  - ทำงานร่วมกับขาอินพุตประจำตัวของไทเมอร์ได้
  - มีปรีสเกลเลอร์สำหรับหารความถี่การนับ
  - สามารถกำหนดการอินเอตร์รัปต์จากการนับหรือจากการตรวจพบสัญญาณขอบขาลงที่ขาอินพุตของไทเมอร์

- คุณสมบัติของไทเมอร์ 2 และ 3

- ไทเมอร์ 2 และ 3 เมื่อทำงานแยกอิสระต่อกัน มีคุณสมบัติคล้ายกับไทเมอร์ 1
- เมื่อนำไทเมอร์ 2 และ 3 มาทำงานร่วมกัน รีจิสเตอร์ตัวนับมีความละเอียดเพิ่มเป็น 32 บิต
- ทำงานร่วมกับขาอินพุตประจำตัวไทเมอร์ได้ (ขา TxCKI)
- มีปรีสเกลเลอร์สำหรับหารความถี่การนับ
- สามารถกำหนดการอินเตอร์รัปต์จากการนับหรือจากการตรวจพบสัญญาณขอบขาลงที่ขาอินพุตของไทเมอร์
- สามารถกำเนิดสัญญาณกระตุ้นการทำงานไปยังโมดูล ADC ได้

2.5.1.2 รีจิสเตอร์ในโมดูลไทเมอร์ของ dsPIC30F2011

ในโมดูลไทเมอร์ของ dsPIC30F2011 มีรีจิสเตอร์ขนาด 16 บิตที่สำคัญอยู่ 3 กลุ่มคือ

กลุ่มที่ 1 TMRx (16-bit timet count register) เป็นกลุ่มของริจิสเตอร์เก็บค่าไทเมอร์ กลุ่มนี้มี 5 ตัวคือ TMR1, TMR2, TMR3, TMR4 และ TMR 5 แต่สำหรับ dsPIC30F2011 มี 3 ตัวคือ TMR1 ถึง TMR3

กลุ่มที่ 2 PRx (16-bit period register associated with the timer) เป็นกลุ่มของรีจิสเตอร์คาบเวลาที่สัมพันธ์กับไทเมอร์ มี 5 ตัวเช่นกัน PR1 ถึง PR5 แยกกันตามไทเมอร์หลักทั้ง 5 ตัว (ไทเมอร์ 1 ถึงไทเมอร์ 5) แต่สำหรับ dsPIC30F2011 มี 3 ตัวคือ PR1 ถึง PR3

กลุ่มที่ 3 TxCON (16-bit control register associated with the timer) เป็นกลุ่มของรีจิสเตอร์ที่ใช้ควบคุมการทำงานของไทเมอร์ มี 5 ตัวเช่นกัน T1CON ถึง T5CON แยกกันตามไทเมอร์หลักทั้ง 5 ตัว สำหรับกลุ่มนี้จะมีความพิเศษตรงที่ยังสามารถแยกย่อยออกไปได้ 3 แบบภายใต้ชื่อรีจิสเตอร์เดียวกันเนื่องจากไทเมอร์สามารถทำงานได้อีก 3 แบบตามลักษณะของฐานเวลา จึงทำให้รีจิสเตอร์ TxCON สามารถระบุได้เป็น TxCON ในแบบ A,B และ C เวลาเขียนโปรแกรมเพื่อกำหนดค่าให้แก่ริจิสเตอร์ TxCON จะต้องระมัดระวังในจุดนี้ด้วย

2.5.2 การทดลองสื่อสารข้อมูลอนุกรมโดยใช้โมดูล UART ใน dsPIC30F2011

สำหรับการทดลองในบทนี้จะเป็นการเขียนโปรแกรมเพื่อทดสอบการใช้งานโมดูล UART เพื่อการสื่อสารข้อมูลอนุกรมระหว่าง dsPIC30F2011 กับคอมพิวเตอร์ผ่านทางพอร์ตอนุกรม

2.5.2.1 คุณสมบัติโดยสรุปของโมดูล UART ใน dsPIC30F2011

- สื่อสารข้อมูลแบบสองทิศทาง (ฟูลดูเพล็กซ์ : full-duplex) ในแบบ 8และ9 บิต

- เลือกการสื่อสารข้อมูลแบบตรวจสอบบิตพาริตี้คู่ (Even) หรือคี่ (Odd) และไม่ตรวจสอบบิตพาริตี้ (None) สำหรับรูปแบบสื่อสารข้อมูลในแบบ 8 บิต มีบิตหยุด (Stop bit) 1 หรือ 2 บิต

- มีส่วนกำเนิดอัตราบอด (Baud Rate Generator) ขนาด 16 บิต สำหรับกำหนดจังหวะและอัตราเร็วในการสื่อสารข้อมูลอนุกรมแยกอิสระเพื่อลดภาระการทำงานาของโมดูลไทเมอร์

- กำเนิดอัตราบอดได้ตั้งแต่38บิตต่อวินาที(bps) ถึง 1.875เมกะบิตต่อวินาที (Mbps)

- บัฟเฟอร์ข้อมูลขาส่ง (TX) และขารับ (RX) ขนาด 4 เวิร์ด แยกส่วนกัน

- มีบิตแฟลกแจ้งข้อผิดพลาดในกรณีต่างๆ ของการสื่อสาร สามารถตรวจจับความผิดพลาดในการสื่อสารข้อมูลอนุกรม ได้แก่

  - ความผิดพลาดทางพาริตี้ (Parity Error : PE)
  - รับข้อมูลไม่ทัน (Buffer Overun Error : OE)
  - เฟรมข้อมูลผิดพลาด (Framing Error : FE)

- สนับสนุนความสามารถในการอินเตอร์รัปต์แอสเดรส (ข้อมูลบิต 9 เป็น "1")

- อินเตอร์รัปต์เวกเตอร์แยกตำแหน่งกันระหว่างการส่งและรับข้อมูล (RX)

- สามารถทำงานในโหมด Loopback

2.5.2.2 รีจิสเตอร์ที่ใช้ในโมดูล UART มีทั้งสิ้น 5 ตัวคือ

- UxMODE ใช้กำหนดโหมดการทำงานของโมดูล UART โดยตัวอักษร x เป็นเลข 1 หรือ 2 เพราะใน dsPIC สามารถบรรจุโมดูล UART ได้ 2 ชุด สำหรับใน dsPIC30F2011 มี 1 ชุด จึงกำหนดเป็น U1MODE

- UxSTA ใช้แสดงสถานะการทำงานของโมดูล UART โดยตัวอักษร x เป็นเลข 1 หรือ 2 เพราะใน dsPIC สามารถบรรจุโมดูล UART ได้ 2 ชุด สำหรับใน dsPIC30F2011 มี 1 ชุด จึงกำหนดเป็น U1STA

- UxRXREG ใช้เก็บข้อมูลที่รับเข้ามาทางขาเชื่อมต่อพอร์ตอนุกรม โดยตัวอักษร x เป็นเลข 1 หรือ 2 เพราะใน dsPIC สามารบรรจุโมดูล UART ได้ 2 ชุด สำหรับใน dsPIC30F2011 มี 1 ชุด จึงกำหนดเป็น U1RXREG

- UxTXREG ใช้เก็บข้อมูลสำหรับส่งออกทางขาเชื่อมต่อพอร์ตอนุกรม โดยตัวอักษร x เป็นตัวเลข 1 หรือ 2 เพราะใน dsPIC สามารถบรรจุโมดูล UART ได้ 2 ชุดสำหรับใน dsPIC30F2011 มี 1 ชุดจึงกำหนดเป็น U1TXREG

- UxBRG ใช้เก็บค่าสำหรับกำหนดอัตราบอด โดยตัวอักษร x เป็นเลข 1 หรือ 2 เพราะใน dsPIC สามารถบรรจุโมดูล UART ได้ 2 ชุดสำหรับใน dsPIC30F2011 มี 1 ชุดจึงกำหนดเป็น U1BRG

2.5.2.3 การกำหนดให้โมดูล UART ทำงาน

โมดูล UART ในไมโครคอนโทรลเลอร์ dsPIC ใช้รูปแบบข้อมูลในมาตรฐาน NRZ นั่นคือ มีบิตเริ่มต้น 1 บิต, บิตข้อมูล 8 หรือ 9 บิต และบิตปิดท้าย 1 หรือ 2 บิต ส่วนบิตพาริตี้สามารถเลือกได้แบบคู่ (even), คี่(odd) หรือไม่มี(none) โดยปกติแล้วจะเลือกใช้รูปแบบ 8N1 คือ มีบิตเริ่มต้น 1 บิต, บิตข้อมูล 8 บิต และบิตปิดท้าย 1 บิต สำหรับ dsPIC30F2011 สามารถกำหนดได้ที่บิต PDSEL1, PDSEL0 และ STSEL ซึ่งเป็นบิต 2,1 และ 0 ตามลำดับในรีจิสเตอร์ U1MODE ส่วนการกำหนดอัตราบอดนั้นกระทำผ่านรีจิสเตอร์ UXBRO ขนาด 16 บิต

การรับส่งข้อมูลในโมดูล UART นั้นจะรับและส่งข้อมูลบิต LSB หรือบิตนัยสำคัญต่ำสุดก่อนโดยตัวรับและส่งข้อมูลในโมดูล UART ของ dsPIC จะทำงานเป็นอิสระแยกจากกัน โดใช้อัตราบอดและรูปแบบข้อมูลที่เหมือนกัน จึงสามารถรับส่งข้อมูล 2 ทิศทางพร้อมกันตลอดเวลา

- การเอ็นเอเบิลโมดูล UART ใน dsPIC30F2011 ให้ทำงาน

ทำได้โดยการเซตบิต UARTEN ซึ่งเป็นบิต 15 ในรีจิสเตอร์ U1MODE และเซตบิต UTXEN ซึ่งเป็น 10 ในรีจิสเตอร์ U1STA ทันทีที่เอ็นเอเบิล ขา U1TX และ U1RX จะถูกกำหนดให้ทำงานเป็นขาพอร์ตเอาต์พุตและอินพุตตามลำดับ โดยไม่สนการกำหนดทิศทางที่เกิดขึ้นก่อนหน้านี้

- การดิสเอเบิลโมดูล UART ใน dsPIC30F2011

ทำได้โดยการเคลียร์บิต UARTEN ซึ่งเป็นบิต 15 ในรีจิสเตอร์ U1MODE โดยปกติโมดูล UART จะถูกดิสเอเบิลหลังจากที่เกิดการรีเซต ซึ่งเป็นการกำหนดสถานะในเบื้องต้น จากนั้นหากต้องการให้ทำงานจะต้องมีการเอ็นเอเบิลภายหลัง เมื่อโมดูล UART ถูกดิสเอเบิล ขา U1TX และ U1RX จะถูกปลดออกจากโมดูล UART สามารถนำไปใช้งานเป็นพอร์ตอินพุตเอาต์พุตทั่วไปได้ทันที

นอกจากนั้นในภาวะที่โมดูล UART ถูกดิสเอเบิล ค่าในบัฟเฟอร์ทั้งหมดจะถูกเคลียร์ เช่นเดียวกับบิตแฟล็กแจ้งสถานะความผิดพลาดทั้งหมดจะถูกเคลียร์ด้วย ส่วนตัวนับอัตราบอดจะอยู่ในภาวะรีเซตหลังจากที่ดิสเอเบิลโมดูล UART แล้วกับมาเอ็นเอเบิลใหม่ การทำงานทั้งหมดจะเริ่มต้นใหม่ทั้งหมด

2.5.3 การทดลองใช้งานโมดูลแปลงสัญญาณอะนาลอกเป็นดิจิตอลภายใน dsPIC

2.5.3.1 คุณสมบัติโดยสรุปของโมดูลแปลงสัญญาณอะนาลอกเป็นดิจิตอล

- เป็นโมดูลแปลงสัญญาณอะนาลอกเป็นดิจิตอลที่มีความละเอียด 10 บิต จำนวน 6 ช่อง

-ใช้วิธีการแปลงสัญญาณแบบประมาณค่าหรือซักเซสซีฟ แอ็บพร็อกซิเมชั่น (Successive Approximation)

- มีอัตราเร็วในการสุ่มสัญญาณสูงสุด 500 กิโลแซมเปิลต่อวินาที (ksps) หรือ 500,000 จุดตัวอย่างต่อวินาที

- สามารถกำหนดให้ทำงานได้ขณะเข้าสู่โหมดสลีป (Sleep mode)

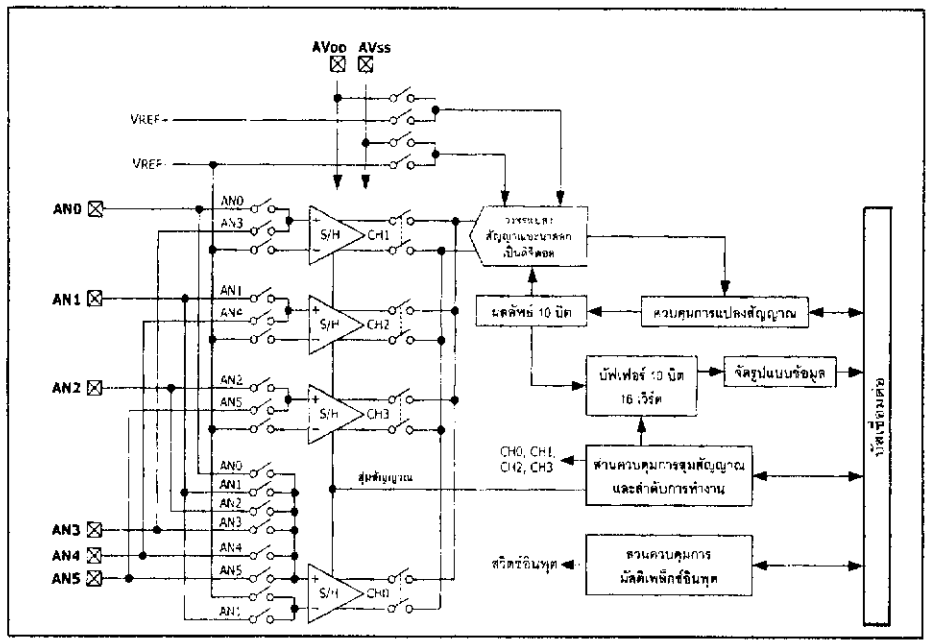
- สามารถกำหนดระดับแรงดันอ้างอิงได้ทั้งจากภายในผ่านทางขา AV_{DD} กับ AV_{SS} และภายนอกผ่านทางขา V_{REF+} และ V_{REF-}

2.5.3.2 การทำงานเบื้องต้นของโมดูล ADC ใน dsPIC30F2011

ในรูปที่ 2.6 เป็นไดอะแกรมการทำงานของโมดูล ADC ใน dsPIC30F2011 ซึ่งมีขาพอร์ตอินพุตอะนาลอกทั้งสิ้น 6 ขาคือ AN0-AN5 โดยมี 2 ขาที่สามารถใช้รับแรงดันอ้างอิงเพื่อขยายย่ายของแรงดันอินพุต ภายในโมดูลมีวงจรสุ่มและเก็บค่าสัญญาณ (Sample and Hold : S/H) จำนวน 4 ชุด โดยทำงานร่วมกับส่วนควบคุมการมัลติเพล็กซ์สัญญาณอินพุต ทำให้สามารถจัดสรรวงจร S/H ให้สามารถรองรับกับสัญญาณอินพุตอะนาลอกทั้ง 6 ช่องได้ด้วยความเร็วสูงสุด

สัญญาณที่ผ่านจากวงจร S&H จะถูกป้อนเข้าสู่วงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอลแบบซักเซสซิฟ แอ็ปพร็อกซิเมชั่น ขนาด 10 บิต ข้อมูลที่ได้จากการแปลงจะถูกพักไว้ในหน่วยความจำแรมจากนั้นจะได้รับการจัดรูปแบบตามที่ผู้พัฒนาโปรแกรมกำหนด ดังแสดงในรูปที่ 8-2 จากนั้นข้อมูลจะถูกถ่ายทอดลงบนบัสข้อมูลเพื่อส่งไปยัง CPU ต่อไป

อีกองค์ประกอบหนึ่งที่ทำให้โมดูล ADC สามารถแปลงสัญญาณได้รวดเร็วคือ ภายในโมดูล ADC มีบัฟเฟอร์ความจุ 16 เวิร์ด นั่นคือ สามารถรองรับข้อมูลที่ได้จากการแปลงสูงสุด 16 ชุดข้อมูล ดังนั้นเมื่อแปลงสัญญาณครั้งหนึ่งก็นำมาเก็บไว้ที่บัฟเฟอร์ หากบัฟเฟอร์ยังไม่เต็มก็สามารถกลับไปแปลงสัญญาณต่อได้ทันที โดยไม่ต้องรอให้การถ่ายทอดข้อมูลไปยังรีจิสเตอร์ที่ใช้เก็บค่าการแปลงเสร็จสิ้น



รูปที่ 2.6 ไดอะแกรมการทำงานอย่างง่ายของโมดูล ADC ใน ไมโครคอนโทรลเลอร์ dsPIC30F2010

2.5.3.3 การกำหนดค่าเพื่อใช้งานโมดูล ADC มีขั้นตอนโดยสรุปดังนี้

- ตั้งค่าของโมดูล ADC

- เลือกขาพอร์ตให้ทำงานเป็นอินพุตอะนาลอกที่รีจิสเตอร์ ADPCFG
- เลือกแหล่งจ่ายแรงดันอ้างอิงให้เหมาะสมกับย่านแรงดันอะนาลอกทางอินพุตที่บิต 15 ถึง 13 ของรีจิสเตอร์ ADCON2
- เลือกสัญญาณนาฬิกาสำหรับการแปลงสัญญาณที่บิต 5 ถึง 0 ของรีจิสเตอร์ ADCON 3
- กำหนดจำนวนช่องของวงจร S/H ที่ต้องใช้ที่บิต 9 และ 8 ของรีจิสเตอร์ ADCON2 และรีจิสเตอร์ ADPCFG
- กำหนดวิธีการสุ่มสัญญาณที่บิต 3 ของรีจิสเตอร์ ADCON1 และรีจิสเตอร์ ADCSSL
- กำหนดจำนวนอินพุตที่ต้องทำงานร่วมกับวงจร S/H ที่รีจิสเตอร์ ADCHS
- เลือกลำดับการสุ่มและแปลงสัญญาณที่บิต 7 ถึง 0 ของรีจิสเตอร์ ADCON1 และ บิต 12 ถึง 8 ของรีจิสเตอร์ ADCON3
- เลือกรูปแบบของผลลัพธ์ที่ต้องการที่บิต 9 และ 8 ของรีจิสเตอร์ ADCON1

- เลือกการอินเตอร์รัปต์ที่บิต 9 ถึง 5 ของรีจิสเตอร์ ADCON2
- เปิดการทำงานขาองโมดูล ADC ที่บิต 15 ของรีจิสเตอร์ ADCON1

- กำหนดการอินเอตร์รัปต์ (ถ้าต้องการ)

- เคลียร์บิต ADIF
- เลือกระดับความสำคัญของการอินเตอร์รัปต์

2.6 การสื่อสารข้อมูลอนุกรม RS-232C

พอร์ตสื่อสารข้อมูลอนุกรม RS-232C คุณสมบัติของ RS-232C

อัตราการรับส่งข้อมูล	: 0-20,000 บิตต่อวินาที
ระดับแรงดันเอาต์พุตสูงสุดในภาวะไม่มีโหลด	: -25 โวลต์ (ลอจิก "1")
	+25 โวลต์ (ลอจิก "0")
ระดับแรงดันเอาต์พุตสำหรับโหลด 3-7 กิโลโอห์ม	: ลอจิก "1" -15 โวลต์ (7 กิโลโอห์ม)
	-5 โวลต์ (3 กิโลโอห์ม)
	ลอจิก "0" -15 โวลต์ (7 กิโลโอห์ม)
	-5 โวลต์ (3 กิโลโอห์ม)
กระแสเอาต์พุต เมื่อลัดวงจร	: สูงสุด 500 มิลลิแอมป์
เอาต์พุตอิมพีแดนซ์เมื่อไม่จ่ายไฟเลี้ยง	: ต่ำสุด 300 โอห์ม
สลูว์เรตทางเอาต์พุตสูงสุด	: 30 โวลต์ต่อไมโครวินาที
ความต้านทานอินพุตของภาครับ	: สูงสุด 7 กิโลโอห์ม
	: ต่ำสุด 3 กิโลโอห์ม
ค่าความจุอินพุตของภาครับ	: สูงสุด 2,500 พิโกฟารัด
ย่านแรงดันอินพุตของภาครับ	: -25 โวลต์ ถึง +25 โวลต์

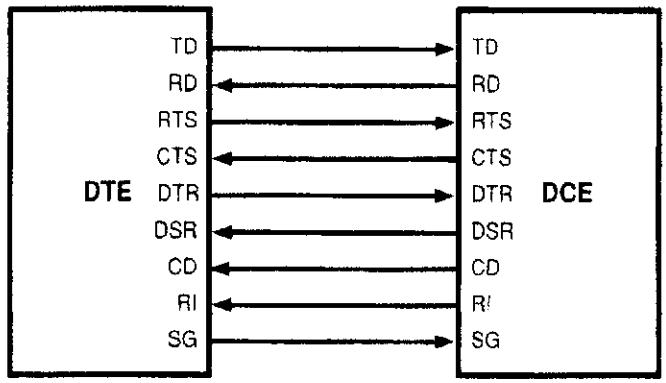
การจัดขาสัญญาณของ RS-232C มีด้วยกัน 2 แบบคือ แบบ 9 ขา และ 25 ขา ดังนี้

ตารางที่ 2.3 แสดงขาสัญญาณของ RS-232 ทั้งแบบ 9 และ 25ขา

ชื่อสัญญาณ		หมายเลขขา ในแบบ 9 ขา	หมายเลขขา ในแบบ 25 ขา
TD	Transmitted Data	3	2
RD	Received Data	2	3
RTS	Request to Send	7	4
CTS	Clear to Send	8	5
DSR	Data Set Ready	6	6
SG	Signal Ground	5	7
CD	Carrier Detect	1	8
DTR	Data Terminal Ready	4	20
RI	Ring indicator	9	22

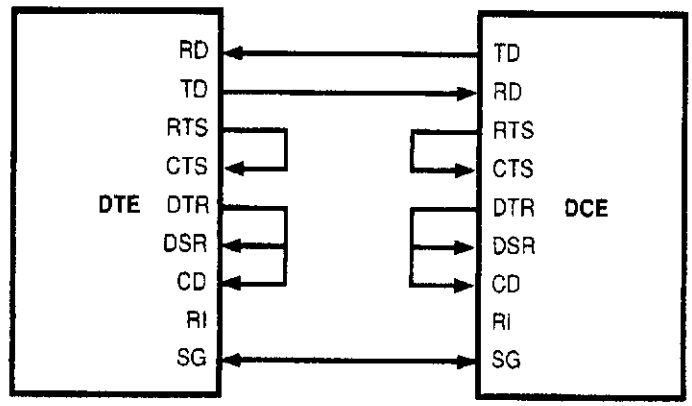
การเชื่อมต่อสัญญาณของ RS-232C

มีลักษณะการเชื่อมต่อ 2 แบบด้วยกัน แบบแรกเป็นการเชื่อมต่อกันระหว่างอุปกรณ์ DTE (Data Terminal Equipment) เช่น คอมพิวเตอร์ กับอุปกรณ์ DCE (Data Circuit Terminal Equipment) เช่น โมเด็ม แสดงดังรูปที่ 2.7



รูปที่ 2.7 การต่ออุปกรณ์ DTE เข้ากับ DCE

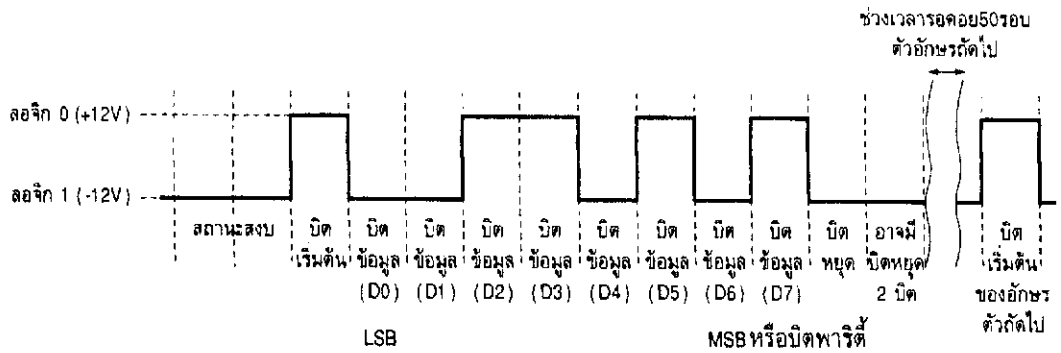
แบบที่สองเป็นการต่อระหว่างอุปกรณ์ DTE ด้วยกันเองทั้งสองฝ่าย หรือเชื่อมต่อคอมพิวเตอร์ 2 ตัวเข้าด้วยกัน เพื่อส่งผ่านถ่ายเทข้อมูล มีรูปแบบการต่อดังรูปที่ 2.8



รูปที่ 2.8 การต่ออุปกรณ์ DTE เข้ากับ DTE

รูปแบบของข้อมูลอนุกรมและอัตราบอดในการสื่อสารข้อมูลอนุกรม

อัตราบอด (baud rate) คือ ความเร็วในการรับส่งข้อมูลอนุกรม มีหน่วยเป็นบิตต่อวินาที



รูปที่ 2.9 รูปแบบของสัญญาณข้อมูลอนุกรมที่ใช้ในการสื่อสารข้อมูลอนุกรม

ตารางที่ 2.4 อัตราบอดและช่วงเวลาของแต่ละบิตข้อมูลในการสื่อสารข้อมูลอนุกรม

อัตราบอด	ช่วงเวลาของแต่ละบิต
110	9.091 ms
150	6.67 ms
300	3.33 ms
600	1.67 ms
1200	833 µs
2400	417 µs
4800	208 µs
9600	104 µs
19200	52.08 µs

2.7 Visual Basic กับการเขียนโปรแกรมส่งไฟล์ข้อมูลผ่าน Serail Port

การติดควบคุมผ่านทางพอร์ตอนุกรม(Serial)ด้วย MSComm6

เริ่มต้นด้วยการเขียนโปรแกรมเพื่อส่งผ่านไฟล์(File)ข้อมูล โดยจะส่งผ่านทาง ComPort สิ่งแรกที่ต้องมาทำความเข้าใจเกี่ยวกับการเข้าถึงไฟล์ข้อมูลก่อน ใน Visual Basic สามารถเข้าถึงไฟล์ข้อมูลได้ 3 ลักษณะคือ Sequential File หรือ ไฟล์ข้อมูลแบบเรียงลำดับต่อเนื่อง มีลักษณะการบันทึก และอ่านไฟล์ข้อมูล ที่เราป้อนลงในเครื่องคอมพิวเตอร์เป็นลำดับจากต้นไฟล์จนถึงท้ายไฟล์ข้อมูล(First In First Out) ซึ่งข้อมูลที่นำไปเก็บจะต่อเนื่องกันโดยไม่มีช่องว่าง ทั้งนี้แม้ว่า แต่ละเรกคอร์ด(Record) จะมีความยาวข้อมูลไม่เท่ากัน

ข้อเสีย ของการบันทึกข้อมูลของไฟล์ประเภทนี้คือเมื่อจะใช้ข้อมูลทุกครั้งจากไฟล์จะต้องเริ่มใช้ตัวแรก เรียงลำดับกันไป หากจะเรียกเจาะจงข้อมูลไม่สามารถทำได้เป็นผลให้ทำงานช้าจึงไม่เหมาะกับการเก็บ ไฟล์ข้อมูลจำนวนเยอะๆ ครับ

Random File หรือ ไฟล์แบบสุ่ม

วิธีการจัดการข้อมูลของ Random File นี้สามารถจะเข้าถึงข้อมูลได้รวดเร็วเพราะไม่ต้องมีการเรียงข้อมูลเหมือนอย่าง Sequential File ซึ่งเราสามารถเจาะจงใช้ข้อมูลได้โดยตรงยัง Record นั้นๆ แต่มีข้อดีก็ต้องข้อเสียเล็กน้อยคือการใช้งานจะซับซ้อนกว่า Sequential File

Binary File ระบบการจัดการไฟล์แบบนี้ จะทำให้เราสามารถรับส่งข้อมูลเป็นไปอย่างสมบูรณ์ เพราะไฟล์ประเภทนี้จะสามารถแทนข้อมูลในลักษณะใดๆก็ได้ และยังสามารถกำหนดรูปแบบโครง สร้างข้อมูล ตัวแปรและความยาวในการบันทึกข้อมูล ทั้งนี้ขนาดความจุของไฟล์ก็มีขนาดเล็ก

ผู้เขียนจะใช้รูปแบบจัดการ Binary File เป็นหลักในเนื้อหา เนื่องจากเราจะต้อง ส่งข้อมูลผ่าน Serial Port ที่มีบัพเฟอร์ข้อมูลไม่มาก แต่ด้วยข้อมูลที่มีขนาดเล็ก โดยสามารถที่จะรับส่งไฟล์ข้อมูลต่างๆได้ในระดับ Binary แล้ว เช่น ไฟล์รูปภาพ,ไฟล์เอกสาร,ไฟล์โปรแกรม เป็นต้น ซึ่งมีข้อดีที่แตกต่าง กว่าแบบ Sequential File และ Ramdom File ที่จะมีข้อมูลในแต่ Record ขนาดใหญ่ จำกัดประเภท ไฟล์ที่รับส่งข้อมูล เข้าถึงข้อมูลได้ช้า ทั้งหมดนี้จึงเป็นสาเหตุที่ผู้เขียนเลือกใช้การรับส่งข้อมูลด้วยไฟล์ประเภท Binary

เริ่มด้วยการเปิดไฟล์ข้อมูล

Open pathname For mode [Access access] [lock] As [#]filenumber [Len=reclength]

Pathname คือ ชื่อไฟล์ที่จะใช้งาน

Mode คือ คำสงวนที่ใช้ในเลือกโหมดของไฟล์มีคำว่า Append, Binary, Input, Output, or Random ถ้าไม่ได้กำหนด ค่าปกติจะตั้งไว้สำหรับ โหมด Random

Access คือ คำสงวนใช้ในการกระทำเพื่อเข้าถึงไฟล์ข้อมูลมีคำว่า Read, Write, or Read Write

Lock คือ คำสงวนที่ใช้กำหนดคุณสมบัติของไฟล์ข้อมูล มีคำว่า Shared, Lock Read, Lock Write, and Lock Read Write

Filenumber คือ หมายเลขของไฟล์ข้อมูลที่เราใช้ เปิด อยู่ในช่วง 1-511 ทั้งนี้เราสามารถหาหมายเลขไฟล์ที่ว่างด้วยคำสั่ง FreeFile

Reclength คือ ความยาวในการบันทึกข้อมูลกำหนดได้ถึง 32,767 bytes

ตัวอย่าง

Open "TESTFILE" For Input As #1

Open "TESTFILE" For Binary Access Write As #1

Open "TESTFILE" For Output Shared As #1

ทุกครั้งที่เปิดด้วยคำสั่ง Open แล้วเมื่อไม่มีการใช้การอ่านและเขียนข้อมูลแล้ว จะต้องปิดไฟล์ข้อมูลด้วยคำสั่ง Close #filenumber

Close #1

การอ่านและเขียนไฟล์ข้อมูล

คำสั่ง Get เรานำมาใช้เพื่อที่จะอ่านข้อมูลจากไฟล์ที่เราเปิดขึ้นมาเก็บไว้ในตัวแปร โดยมีรูปแบบการใช้งานดังนี้

Get [#]filenumber, [recnumber], varname

คำสั่ง Put ก็จะนำมาใช้เพื่อที่จะเขียนข้อมูลจากตัวแปรลงสู่ไฟล์ที่เราเปิดขึ้นมา โดยมีรูปแบบการใช้งานดังนี้

Put [#]filenumber, [recnumber], varname

Filenumber คือ หมายเลขไฟล์ที่เราเปิดขึ้นใช้อยู่ปัจจุบันในโปรแกรม

Recnumber คือ หมายเลขลำดับที่บันทึกข้อมูลในการเปิดไฟล์แบบ Random mode หรือหมายเลขที่ของ byte ในการเปิดไฟล์แบบ Binary mode เพื่อที่จะเริ่มการอ่านหรือการเขียนข้อมูล

Varname คือ ตัวแปรสำหรับเก็บบันทึกข้อมูลที่อ่านได้จากไฟล์ ตรวจสอบสถานะต่างๆของไฟล์

คำสั่ง Seek ที่ต้องนำมาใช้ก็เพื่อให้การเก็บบันทึกข้อมูลลงไฟล์ให้เป็นไปอย่างต่อเนื่อง โดยจะคืนค่าตำแหน่ง ปัจจุบันในการอ่านและเขียนข้อมูลของไฟล์ที่เราเปิดใช้อยู่ ซึ่งมีค่าตั้งแต่ 1 ถึง 2,147,483,647 ส่วนรูปแบบการใช้มีดังนี้

Seek(filenumber)

คำสั่ง LOF จะนำมาใช้เพื่อตรวจสอบขนาดของข้อมูลในไฟล์ที่เราเปิดใช้งานโดยมีรูปแบบดังนี้

LOF(filenumber)

คำสั่งEOF สำหรับตรวจสอบสถานะของข้อมูลว่าสิ้นสุดขอบเขตไฟล์นั้นๆหรือยังถ้าใช่จะคืนค่าเป็นลักษณะ บูลีน(Boolean) เท่ากับ True ถ้ายังไม่สิ้นสุดจะคืนค่า False มีรูปแบบการใช้ดังนี้

EOF(filenumber)

Filenumber คือ หมายเลขไฟล์ที่เราเปิดขึ้นใช้อยู่ปัจจุบันในโปรแกรม

การโปรแกรมเพื่อรับส่งข้อมูลผ่าน Port

การส่งไฟล์ข้อมูล

1. กำหนดตัวแปรเพื่อใช้งานในรูทีนนี้
2. หาหมายเลขไฟล์ที่ว่างเพื่อเปิดใช้งานด้วยคำสั่ง FreeFile แล้วเก็บไว้ในตัวแปร noSend
3. ใช้คำสั่งเปิดไฟล์แบบ ไบนาลี่ เพื่ออ่านข้อมูล
4.อ่านข้อมูลจากไฟล์ที่เปิด โดยให้ข้อมูลแต่ละชุด มีจำนวนเท่ากับหรือไม่เกินขนาดของ OutBufferSize ใน MSComm1 ซึ่งจะเป็น Buffer สำหรับส่งข้อมูลออกผ่านทาง com port เมื่ออ่านข้อมูลจากไฟล์เป็น Block ข้อมูลเรียบร้อยแล้วก็จะถูกส่งด้วยคำสั่ง

MSComm1.Output

รอจนกว่าข้อมูลจะถูกส่งออกไปจนหมดจาก Buffer โดยใช้การตรวจสอบด้วยคำสั่ง

MSComm1.OutBufferCount

5. ปิดไฟล์ข้อมูลตามหมายเลขที่เปิดขึ้นมาใช้

การรับไฟล์ข้อมูล

ทำการหาหมายไฟล์ที่ว่างและเปิดไฟล์ที่เราจะเก็บข้อมูลที่รับเข้ามาลักษณะของการบันทึกไฟล์ ข้อมูลใหม่

ตรวจสอบข้อผิดผลาดขณะเปิดไฟล์

ใช้คำสั่ง Seek หาข้อมูลปัจจุบันเพื่อเพิ่มต่อข้อมูลของไฟล์ที่เปิดรับข้อมูลอยู่ คำสั่งส่วนนี้จะอยู่ในโพซีเดอร์แบบ Menu หรือ Command Button ก็ได้

การกำหนดคุณสมบัติของ MSComm Control ให้สามารถติดต่อกับพอร์ตได้

1. Property ชื่อ CommPort คือ เลือกคอมพอร์ตที่เราจะต่อใช้งาน
   ตัวอย่าง MSComm1.CommPort=1
   ในที่นี้เลือกจะใช้ Com1อยู่ที่ด้านหลังเครื่องคอมฯ
2. Property ชื่อ Settingsคือ การตั้งค่าของการรับส่งข้อมูล ซึ่งจะต้องรู้ด้วยว่าอัตราบอด ของ อุปกรณ์ที่จะติดต่อด้วยเป็นเท่าไร โดยมีรายละเอียดการใส่ต่างๆค่าดังนี้
   MSComm1.Settings="Baud(อัตราการรับส่งข้อมูล),
   Parity(ถ้าไม่ใช้ใส่N),จำนวนบิตข้อมูล,บิตสต๊อบ"
   ตัวอย่าง MSComm1.Settings="1200,N,8,1"
3. Property ชื่อ InputLenคือ กำหนดขนาดขณะที่มีข้อมูลเข้ามาให้ไปอ่านข้อมูลทั้งหมดที่อยู่ในบัพเฟอร์
   ตัวอย่าง MSComm1.InputLen=1
4. Property ชื่อ PortOpen คือ จะเปิดให้พอร์ตใช้งานหรือไม่ ถ้าเปิด =True ถ้าปิด =False

ตัวอย่าง MSComm1.PortOpen=True

5. Property ชื่อ Rthresholdคือ ทำให้เกิดการกระตุ้นด้วย Event-driven เมื่อมีข้อมูลในบัฟเฟอร์รับข้อมูล(Comport)มันทำให้เกิดCommEvent ใน OnComm Event

ตัวอย่าง MSComm1.Rthreshold =1

จากรายละเอียดที่กล่าวมา เราจะมาเขียนใน โพรซีเดอร์ VB ซึ่งจะไว้ที่ Sub Form_Load() หรือจะสร้าง Sub ขึ้นใหม่ในกรณีที่จะเรียกใช้ภายหลัง

```
Private Sub Form_Load()
    MSComm1.Settings="1200,N,8,1"
    MSComm1.CommPort=1
    MSComm1.InputLen=1
    MSComm1.PortOpen=True
    MSComm1.Rthreshold =1
End Sub
```

วิธีของการรับส่งข้อมูลจาก Serial Port

จากวิธีเขียนโค๊ดด้านบนเป็นการกำหนดค่าเริ่มต้นให้กับคอมพอร์ตและเปิดใช้การรับและส่งของพอร์ต RS-232 ดังนั้นก็สามารถจะรับและส่งข้อมูลทางพอร์ตได้ โดยใช้ Property ดังนี้

Output = ซึ่งจะเป็นการส่งข้อมูลไปที่พอร์ต

Input = เป็นส่วนของการรับข้อมูลจากพอร์ต แต่ในส่วนนี้จะต้องนำคำสั่งไปเขียนที่ Event Property OnComm จะอยู่ใน Sub MSComm_OnComm ซึ่ง จะอ่านข้อมูลเข้ามาจากทางพอร์ต RS232 นั้นเอง

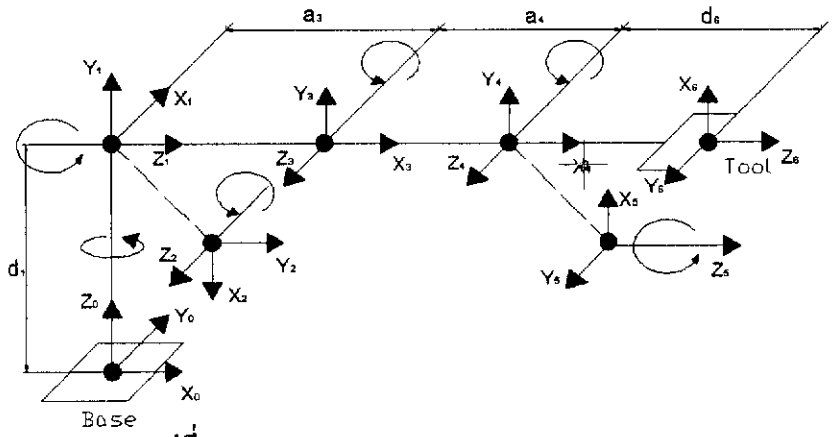
บทที่ 3
หลักการออกแบบ

3.1 แนวคิดในการออกแบบ

การสร้างแขนกลในโครงงานนี้ พบว่าจะต้องทำการออกแบบระบบ Mechanics ก่อน โดยสร้างจากวัสดุที่เป็นโลหะ เพื่อให้มีความแข็งแรง จากนั้นก็จะออกแบบระบบที่ใช้ในการควบคุมได้ โดยในขั้นแรกนี้ได้มีแนวคิดที่จะออกแบบตัวโครงสร้างของแขนกลก่อนจากนั้นจึงทำวงจรควบคุมและเขียนโปรแกรมในภายหลัง ในการสร้างแขนกลได้เลือกอะลูมิเนียมเป็นโครงสร้างเพราะมีความแข็งแรงและมีน้ำหนักเบา โดยวางแบบจะใช้มอเตอร์ทั้งหมด 6 ตัว จึงจะทำให้แขนกลนี้มีความอิสระและมีพื้นที่ใช้งานมาก

3.2 การออกแบบคำนวณตามหลักการและทฤษฎี

เริ่มออกแบบโดยเขียน Kinematics ของแขนกล



รูปที่ 3.1 การออกแบบ Kinematics ของแขนกล

ตารางที่ 3.1 ตารางพารามิเตอร์

Axis	มุม	d	a	α	Home
1	θ_1	204	0	90	90
2	θ_2	0	0	90	-90
3	θ_3	0	182	0	90
4	θ_4	0	148	0	0
5	θ_5	0	0	90	90
6	θ_6	d6	a6	0	0

เราพบว่าจาก Kinematics ของแขนกลสามารถเขียนค่าพารามิเตอร์ได้ดังตารางด้านบน โดยค่าของ d6 และ a6 สามารถใส่ค่าเท่าไรก็ได้ขึ้นอยู่ว่ากับตำแหน่งปลายของ Tool ที่จะใช้งาน โดยเขียนเป็นเมทริก ได้เป็น d = [204 , 0 , 0 , 0 , 0 , d6]T mm , a = [0 , 0 , 182 , 148 , 0 , a6]T mm ส่วนค่า Home คือค่าที่บอกสภาพเริ่มต้นท่าทาง และตำแหน่งของแขนกล โดยตั้งแกนซึ่งมีทิศทางดังรูป โดยเมทริกของแขนกลนี้ จะเป็นตัวที่บอกว่า ที่ปลายแขนกล(Tool)สัมพันธ์กับตำแหน่งฐาน(Base) อย่างไร และพบว่าความสัมพันธ์ระหว่างปลายแขนกล(Tool)และตำแหน่งฐาน (Base) ของแขนกลที่เราได้ออกแบบนี้หาได้จาก

ปลาย(tool) 1 2 3 4 5 6

T ฐาน(base) = T0T1T2T3T 4T5

$$= \begin{bmatrix} C1 & 0 & S1 & 0 \\ S1 & 0 & -C1 & 0 \\ 0 & 1 & 0 & d1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C2 & 0 & S2 & 0 \\ S2 & 0 & -C2 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C3 & -S3 & 0 & a3C3 \\ S3 & C3 & 0 & a3S3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C4 & -S4 & 0 & a4C4 \\ S4 & C4 & 0 & a4S4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} C5 & 0 & S5 & 0 \\ S5 & 0 & -C5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C6 & -S5 & 0 & 0 \\ S6 & C6 & 0 & 0 \\ 0 & 0 & 1 & d6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

ปลาย(tool)

T ฐาน(base) = $\left[\begin{array}{c|c} R & P \\ \hline 0\ 0\ 0 & 1 \end{array}\right]$

$$P = \begin{bmatrix} C1C2(a3C3 + a4C34 + d6S345) + S1(a3S3 + a4S34 - d6C345) \\ S1C2(a3C3 + a4C34 + d6S345) - C1(a3S3 + a4S34 - d6C345) \\ d1 - S2(a3C3 + a3C34 + d6S345) \end{bmatrix}$$

$$R = \begin{bmatrix} (C1C2C345 + S1S345)C6 + C1S2S6 & C1S2C6 - (C1C2C345 + S1S345)S6 & -S1C345 + C1C2S345 \\ (S1C2C345 - C1S345)C6 + S1S2S6 & S1S2C6 - (S1C2C345 - C1S345)S6 & C1C345 + S1C2S345 \\ S2C345C6 - C2S6 & -C2C6 - S2C345S6 & S2S345 \end{bmatrix}$$

โดยที่ $C1 = \cos\theta 1$,

$S12 = \sin\theta 1\cos\theta 2 + \cos\theta 1\sin\theta 2$,

$C12 = \cos\theta 1\cos\theta 2 - \sin\theta 1\sin\theta 2$

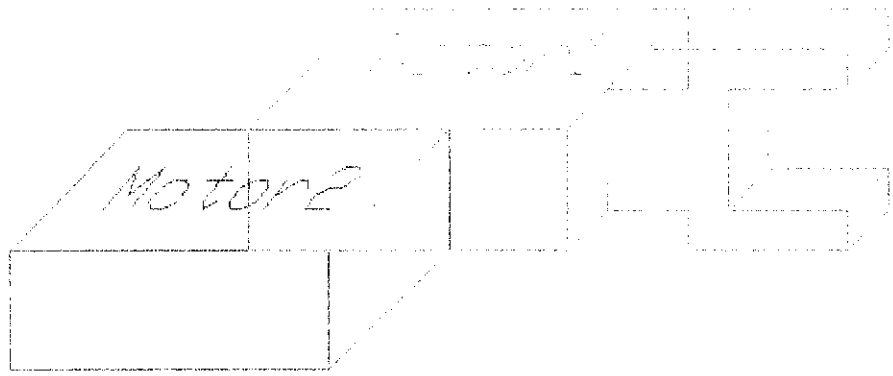
ซึ่งเมทริก $T^{\text{ปลาย(Tool)}}_{\text{ฐาน(Base)}}$ นี้ สามารถนำไปใช้ประโยชน์ได้ โดยเรารู้สมาการเมทริก เพียงแค่ป้อนค่ามุม (θ)ก็จะสามารถรู้ได้ว่าที่ปลายของแขนกลนั้นอยู่ ณ ตำแหน่งใด เมื่อเทียบกับฐานของมันและมีประโยชน์ในการนำไปใช้เขียนโปรแกรม ในทางกลับกันถ้าเรารู้ว่าวัตถุอยู่ตำแหน่งใดเทียบกับฐานแขนกล เราจะหามุมต่างๆ ได้อย่างไร ซึ่งเราก็หาได้ เพียงแต่เขียนหาสมการ Inverse Kinematics ของแขนกลซึ่งค่อนข้างยุ่งยากในการหา ซึ่งเรื่องนี้จะกล่าวในเทอมต่อไป

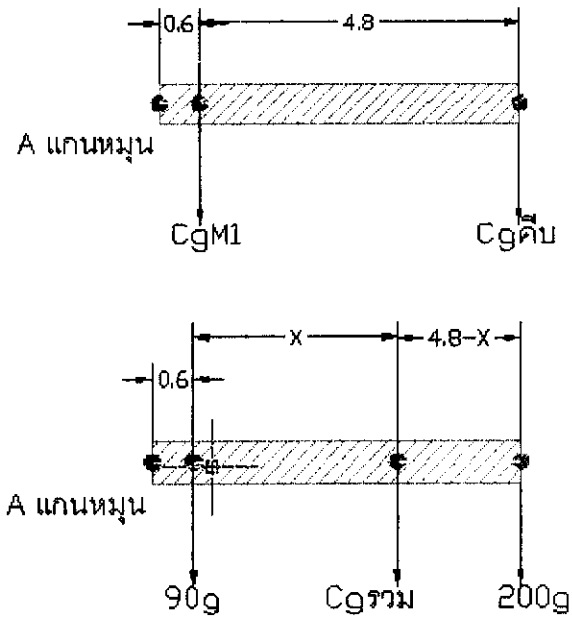
จากนั้นก็ทำการออกแบบแขนกลโดยใช้โปรแกรม Solid Works เพื่อเป็น Model ต้นแบบก่อนมีการสร้างจริง และทำให้เราสามารถทราบข้อผิดพลาดก่อนที่จะทำเพื่อเป็นการลดความผิดพลาดที่เกิดขึ้นและลดงบประมาณได้ โดยแบบที่เขียนขึ้นมานี้เป็นแบบที่มีสเกลตามชิ้นงานจริง

จากการออกแบบข้างต้น เรายังไม่สามารถที่จะสร้างแขนกลได้จริงเพราะว่ายังไม่ทราบว่าขนาดของอะลูมิเนียมที่ออกแบบนั้นจะมีความแข็งแรงเพียงพอที่จะรับน้ำหนักได้ มิฉะนั้นอะลูมิเนียมอาจจะหัก หรืองอได้ รวมทั้งยังไม่ทราบว่า Servo Motor และ DC Motor ต้องมีSpec เท่าไร จึงจะสามารถทำให้แขนกลทำงานได้จริง ดังนั้นเราจะต้องทำการคำนวณวัสดุแต่ละส่วนที่เชื่อมต่อกันรวมทั้งหาว่า Spec มอเตอร์ที่สามารถทำงานได้นั้นต้องมีขนาดเท่าไร

3.3 การคำนวณวัสดุแต่ละชิ้นส่วน

จุดหมุน A





รูปที่ 3.2 จุดหมุน A

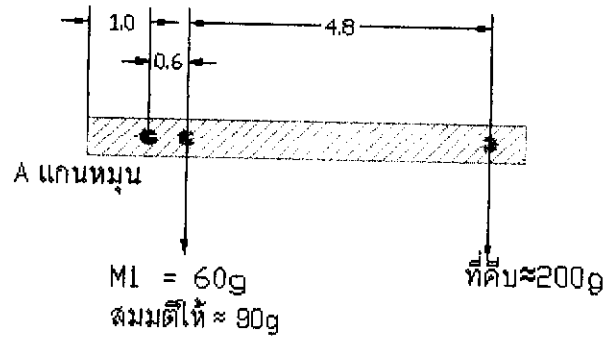
คิด โมเมนต์ (M) รอบจุด A

$$90X = (4.8-X)\,200$$

$$90X-200X = 960$$

$$X = 3.5 \text{ cm.}$$

เพราะฉะนั้น C_g รวมของข้อแขนที่ห่างจุด A = 4.1 ≈ 4 cm. #



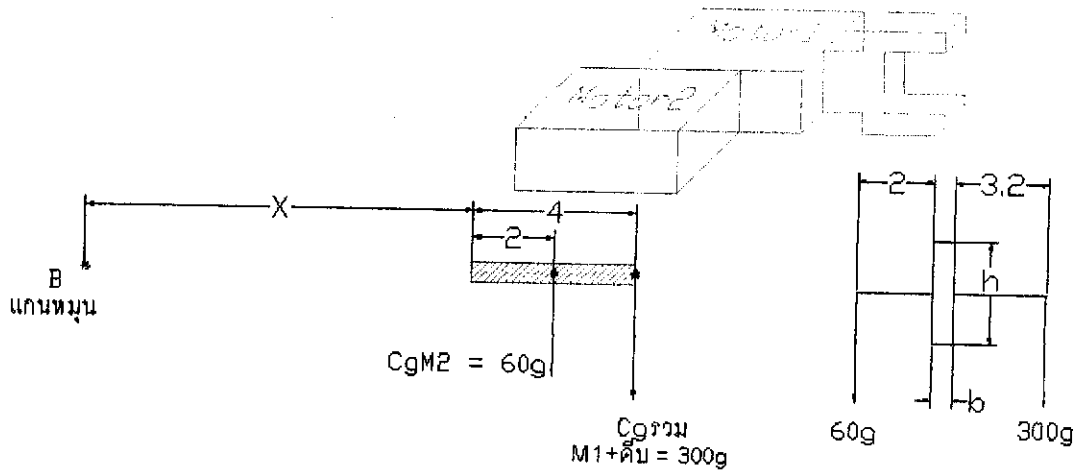
รูปที่ 3.3 จุดหมุน A(2)

คิด $M_A = (90x0.6) + (200x5.4)$

$M_A = 1{,}134$ g-cm.

จาก Data Sheet ของมอเตอร์ GWS (S03N) ที่เราเลือกใช้สามารถรับได้ 4,000 g-cm.
เพราะฉะนั้น มอเตอร์ของ GWS (S03N) ที่ใช้สามารถทำงานได้ #

จุดหมุน B



รูปที่ 3.4 จุดหมุน B

คิด โมเมนต์ (M) รอบจุด B ให้ A1 หนา 3mm. , ความหนาแน่น A1 = 2.7 g/cm.^{3} ในแนวดิ่ง
โมเมนต์รอบจุด B มอเตอร์ GWS (S666) รับสูงสุด 15,000 g-cm.

$M_B = (60X) + (4+X)(300) + (X+2)(b)(h)(D_{A1})((X+2)/2)$

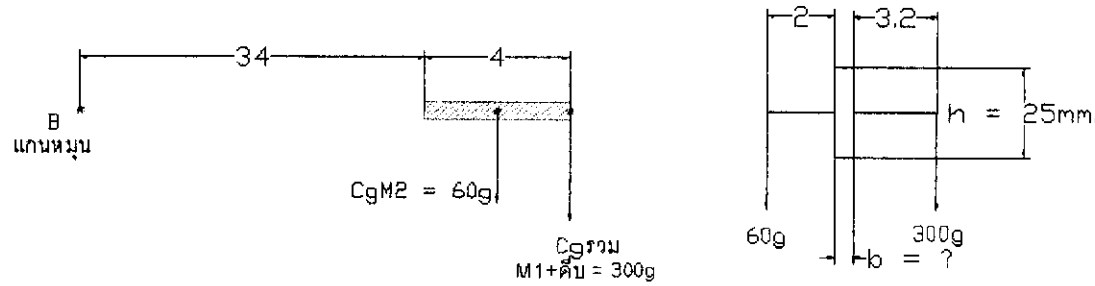
$M_B = (60X) + (4+X)(300) + (X+2)(0.3)(2.5)(2.7)((X+2)/2)$

$$15{,}000 = X^2 + (60X + 30X + 4X) + 1{,}204$$

$$0 = X^2 + 94X - 13{,}796$$

$$X = 34.6 \text{ cm.}$$

เพราะฉะนั้นมอเตอร์ GWS (S666) สามารถรับทอร์กได้ระยะสูงสุด 34.6 cm. #



รูปที่ 3.5 จุดหมุน B(2)

คิดความแข็งแรง

จากตารางพบว่า ค่า Yield Strength ของ A1 มีค่า 25 ksi

$$\sigma_Y = 25 \text{ ksi}$$

$$= 25 \times 6.895$$

$$= 172.375 \text{ N/mm.}^2$$

$$M_B = 340 \times 0.06 \times 9.81 + 380 \times 0.3 \times 9.81$$

$$= 1322.34 \text{ kg - mm.}$$

โมเมนต์ความเฉื่อยของ พื้นที่ I (หา I จากตาราง)

$$I = (bh^3)/12$$

$$= (b(25)^3)/12$$

$$= 1{,}302b$$

ความเค้นดัดในก้านหมุน

$$\sigma_b = (MC)/I$$

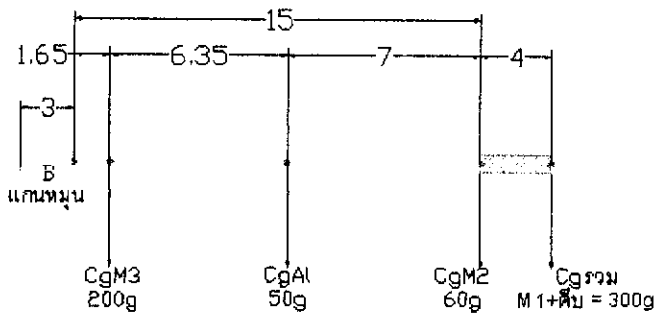
$$= (M(h/2))/I \quad \text{ให้ค่าความปลอดภัย} = 2$$

$$172.375/2 = (1322.374 \times 12.5) / 1302b$$

$$b = 0.15 \text{ mm.}$$

เพราะฉะนั้นในแนวดิ่ง สามารถใช้ A1 ที่มีความหนา b มากกว่าหรือเท่ากับ 0.15 mm. ได้ ฉนั้นเราสามารถเลือกใช้ A1 ขนาด 3 mm. ได้ #

ระยะจากมอเตอร์แทนที่จะเลือกใช้ความยาวถึง 34 cm. ซึ่งเป็นความยาว Max แต่เราเลือกใช้ระยะ 15 cm.



รูปที่ 3.6 จุดหมุน B(3)

เราจะหา M_B จริง ที่ใช้สำหรับเลือกมอเตอร์

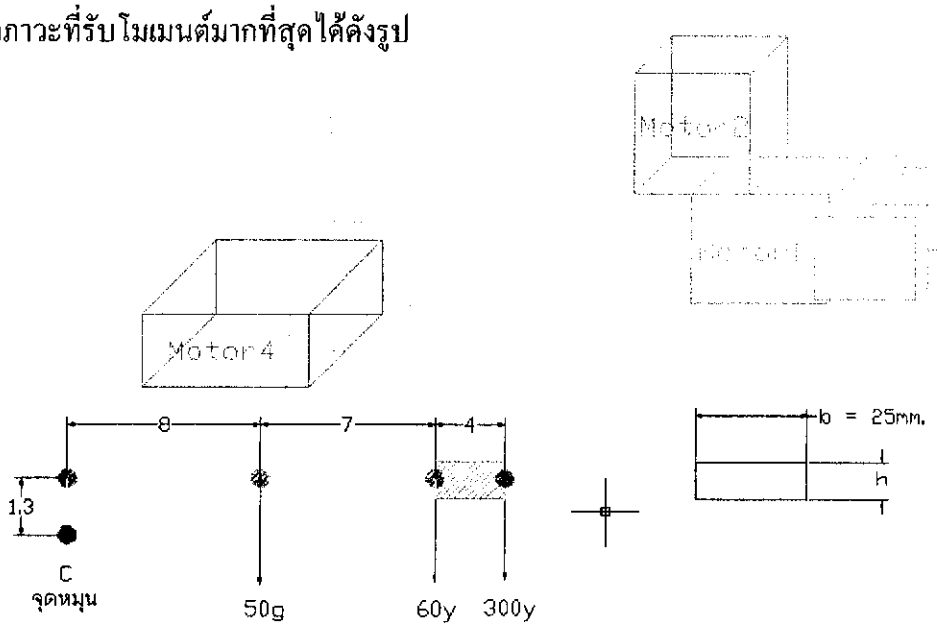
$$M_B = (1.65x200) + (8x50) + (15x60) + (300x19)$$

$$M_B = 7{,}333 \quad g-cm.$$

เพราะฉะนั้น เลือกมอเตอร์ GWS (S666) 15,000 g – cm. ถึงสามารถทำงานได้

จุดหมุน C

เลือกภาวะที่รับโมเมนต์มากที่สุดได้ดังรูป



รูปที่ 3.7 จุดหมุน C

$$M_C = (8x50) + (60x15) + (19x300)$$

$$= 1{,}376 \quad g-cm.$$

เพราะฉะนั้น M_C = 1,376 g - cm. เราเลือกใช้มอเตอร์ GWS (S666 NMG) มีทอร์ก 38,000 g – cm. ได้ #

คิดความแข็งแรง

จากตารางความเค้น Yield Strength ของ A1 มีค่า 25 ksi

$$\sigma_y = 172.375 \text{ N/mm.}^2$$

$$I = (bh^3)/12$$

$$= (25xh^3)/12$$

$$\sigma_b = (MC)/I = (M(h/2))/I$$

ค่าความปลอดภัย = 2;

$$172.375 / 2 = (1376(h) \times 12) / 25\, h^3 x2$$

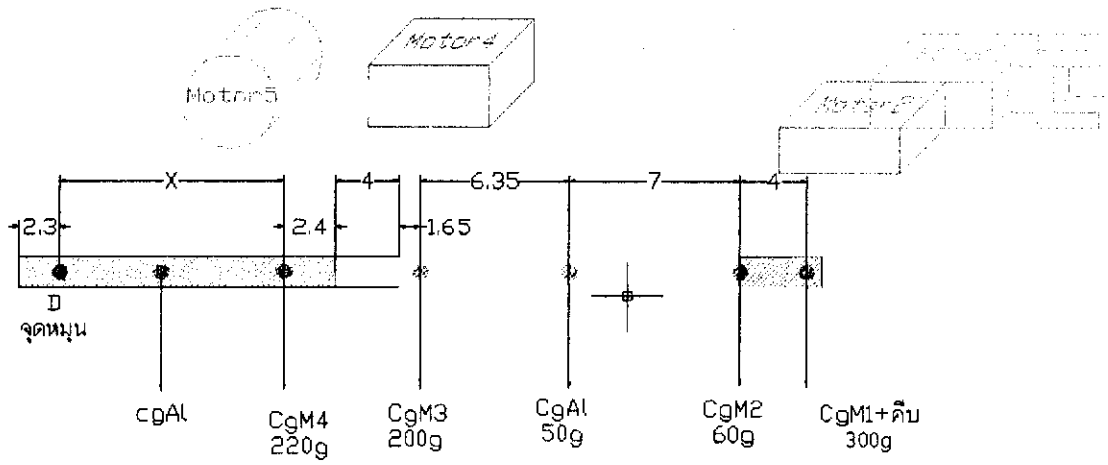
$$86.1875 = (1376x8) / 25\, h^3$$

$$h = 1.25 \text{ mm.}$$

เพราะฉะนั้นต้องเลือกใช้ A1 หนากว่า 2 mm. แต่เราเลือกใช้ 3 mm. เพราะฉะนั้นจึงสามารถทนต่อการงอได้ #

จุดหมุน D

เลือกภาวะที่รับโมเมนต์มากที่สุดได้ดังรูป



รูปที่ 3.8 จุดหมุน D

รวม อะลูมิเนียมที่ยึดมอเตอร์ 3 และ มอเตอร์ 4 = (0.2x2x2.7x8.3) ≈ 9 g

ให้ DC มอเตอร์ มีทอร์ก 25,000 g - cm. = M_D

$$M_D = 220X + (X+6.4+1.65)(200) + (X+6.4+8)(50)$$

$$+ (X+6.4+8+7)(60) + (X+6.4+8+7+4)(300)$$

$$+ ((X+2.4+2.3)/2)\,((X+2.4+2.3)(0.3x2.5x2.7))$$

$$= 220X + 200X + 336.4 + 504 + 720 + 60X + 1284 + 300X$$

$$+ 7620 + X^2 + 9.4X + 22.1$$

$$25{,}000 = X^2 + 839.4X + 9{,}982.5$$

$$X = -856.924 \text{ , } 17.5 \text{ cm.}$$

เพราะฉะนั้นระยะ X ยาวสูงสุดได้ 17.5 cm.

แต่ในการออกแบบนี้เลือก X = 11.6 cm.

การคิดความแข็งแรง คิดเพียงทางแนวตั้งเท่านั้น เพราะแกนไม่ได้หมุนรับน้ำหนักทางแนวนอน

คิดค่าความแข็งแรง

$$M_D = (0.22x9.81x116) + (0.2x9.81x196.5) + (0.05x9.81x260)$$

$$+ (0.06x9.81x330) + (0.3x9.81x370)$$

$$= 2{,}045.6 \text{ g-cm.}$$

$$\sigma_b = 172.375 \text{ N/mm.}^2$$

$$I = (bh^3)/12$$

$$= (25xh^3)/12$$

$$= 1{,}302b$$

$$\sigma_b = (MC)/I$$

$$= M(1.25) / 1{,}302b$$

$$172.375 / 2 = (2045.6x12.5) / 1{,}302b$$

$$b = 0.22 \text{ mm.}$$

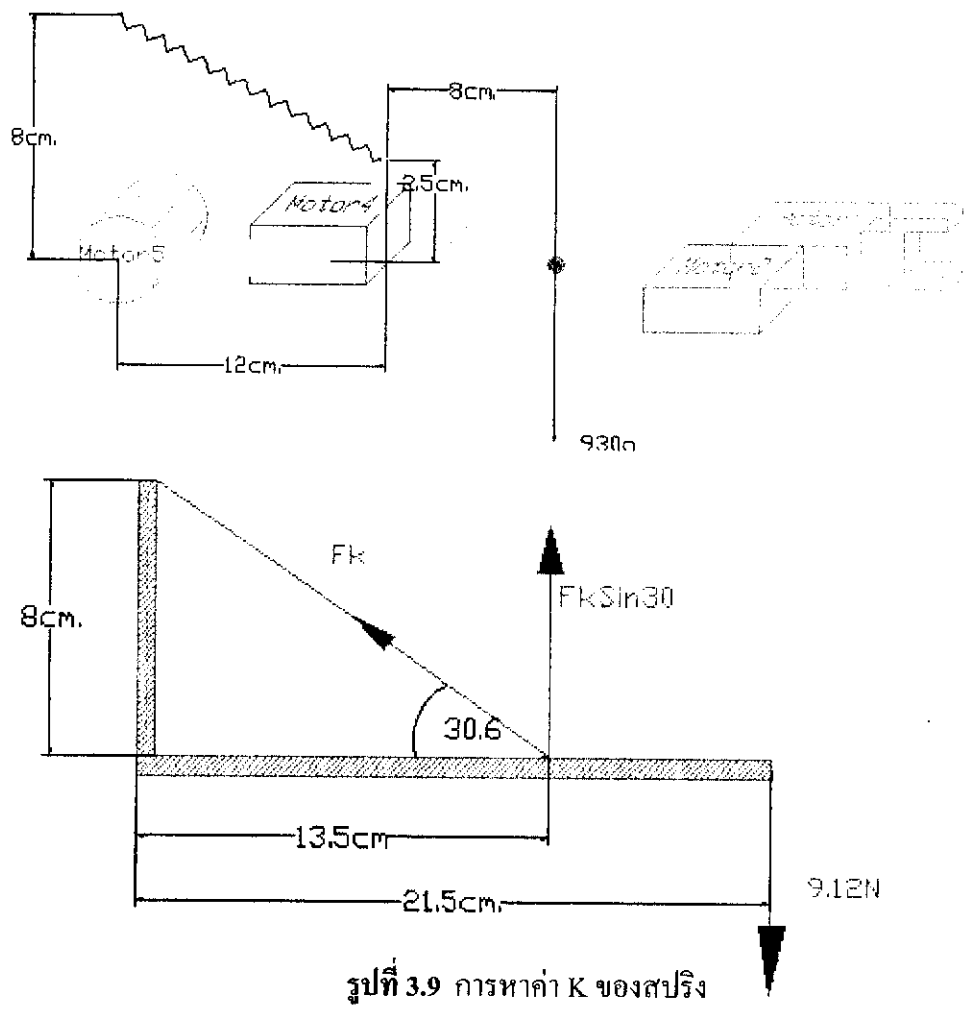
เพราะฉะนั้นสามารถใช้ A1 ขนาดหนา 3 mm. ได้ #

เพื่อเป็นการลดภาระของมอเตอร์ 5 เพราะฉะนั้นจึงใส่สปริงเข้าไปช่วย

การหาค่า K

เพราะฉะนั้น น้ำหนักแขนกล ในแนวขนาดกับพื้นโลก

$$300 + 60 + 50 + 200 + 220 + 100 = 930 \text{ g.} \quad \#$$



รูปที่ 3.9 การหาค่า K ของสปริง

$$2M = 0$$

$$2.12 \times 21.5 - F_k \sin 30 \times 13.5 = 0$$

$$F_k = 29 \text{ N}$$

$$F = k \times 15 \times 10^{-2}$$

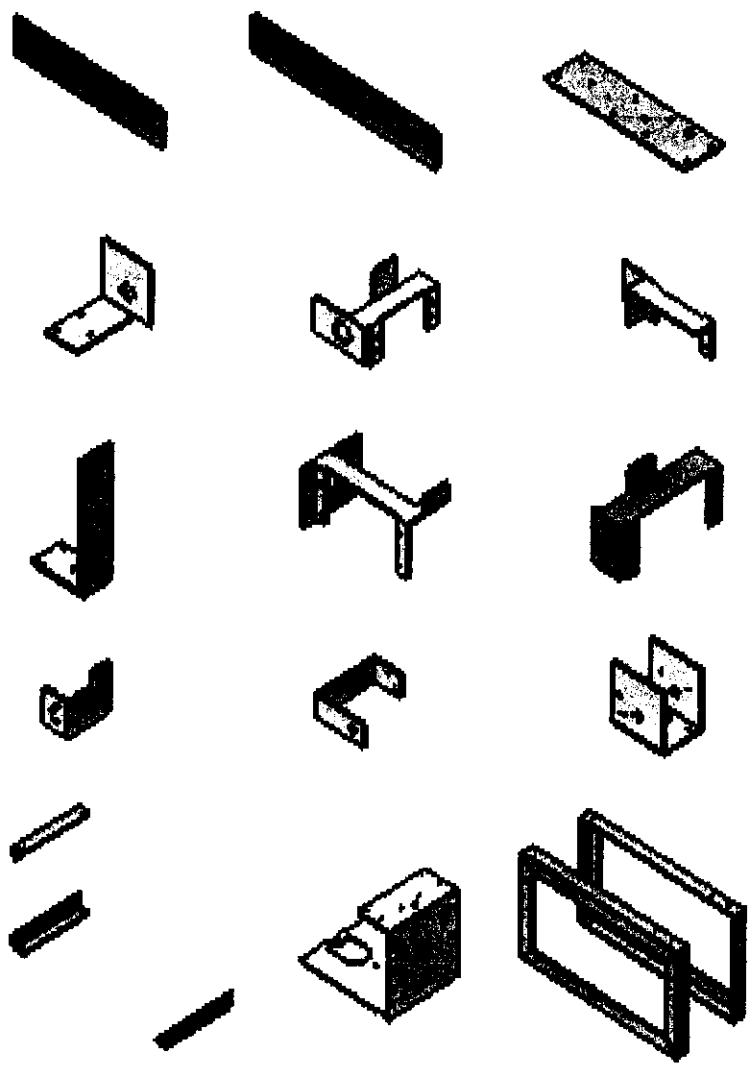
$$k = 29/0.15$$

$$k = 193$$

เพราะฉะนั้นใช้สปริงโดยมีค่า k ประมาณ 150 - 300 #

จากการคำนวณทั้งหมดนี้ทำให้เราสามารถนำแบบที่ได้ออกแบบไว้ นำมาสร้างแขนกลได้จริงขั้นต่อไป เราก็ทำการผลิตชิ้นส่วนแต่ละชิ้นตามแบบที่เขียนไว้ดังนี้ เสร็จแล้วจึงทำการประกอบต่อไป

ส่วนประกอบแต่ละชิ้น



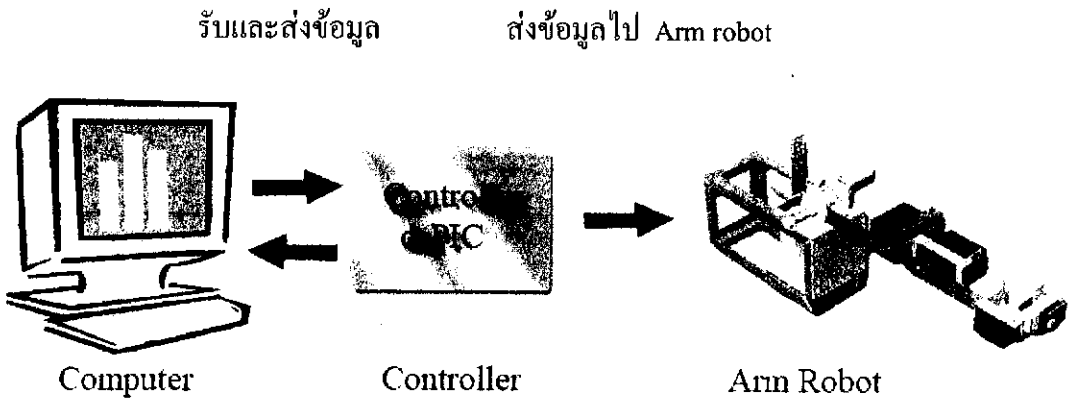
รูปที่ 3.10 ส่วนประกอบของแขนกลแต่ละชิ้น

3.4 แนวคิดในการออกแบบระบบควบคุม

ในการออกแบบระบบควบคุมคณะผู้จัดทำได้คิดว่า เราจะควบคุมแขนกลโดยใช้หลักการเดียวกับ เครื่อง CNC โดยจะทำการโปรแกรม CODE เข้าไปในเครื่องโปรแกรมในที่นี้ เราจะโปรแกรมลงบน Computer โดย Code นี้จะไปสั่งการ Controller ทำงานโดยผ่านทางพอร์ตอนุกรม และ Controller จะควบคุมแขนกลให้ทำงานตามที่เราต้องการได้

โปรแกรมที่ใช้ควบคุมนี้ได้ใช้โปรแกรม Visual Basic ในการเขียน โดยทำการออกแบบโปรแกรมสำหรับป้อน Code และมีส่วนทำการ Compile เพื่อตรวจสอบว่าได้ป้อนโปรแกรมถูกต้องหรือไม่ จากนั้นก็จะทำการ Run code เพื่อส่งข้อมูลที่ป้อนไปยัง Controller dsPIC โดยผ่านพอร์ตอนุกรม ในส่วนของ dsPIC จะใช้โปรแกรม MPLAB เขียน โดยใช้ภาษา C โดยจะรับ Code มาทีละค่าจาก Computer จากนั้นก็สั่งให้ Motor แต่ละตัวหมุนไปตาม Code ที่ป้อนเข้าไป โดยการส่ง Code จาก Computer มายัง Controller เมื่อ Controller ได้รับข้อมูลก็จะส่งข้อมูลเดิมกลับไปยัง Computer เพื่อเป็นการตรวจสอบว่าได้ส่งข้อมูลเรียบร้อยแล้ว เป็นการป้องกันข้อผิดพลาด โดยในการออกแบบได้คำนึงถึงความละเอียดในการควบคุมมอเตอร์ และได้ตั้งจุดมุ่งหมายในการคุมมอเตอร์ให้มีความละเอียด 1 องศา

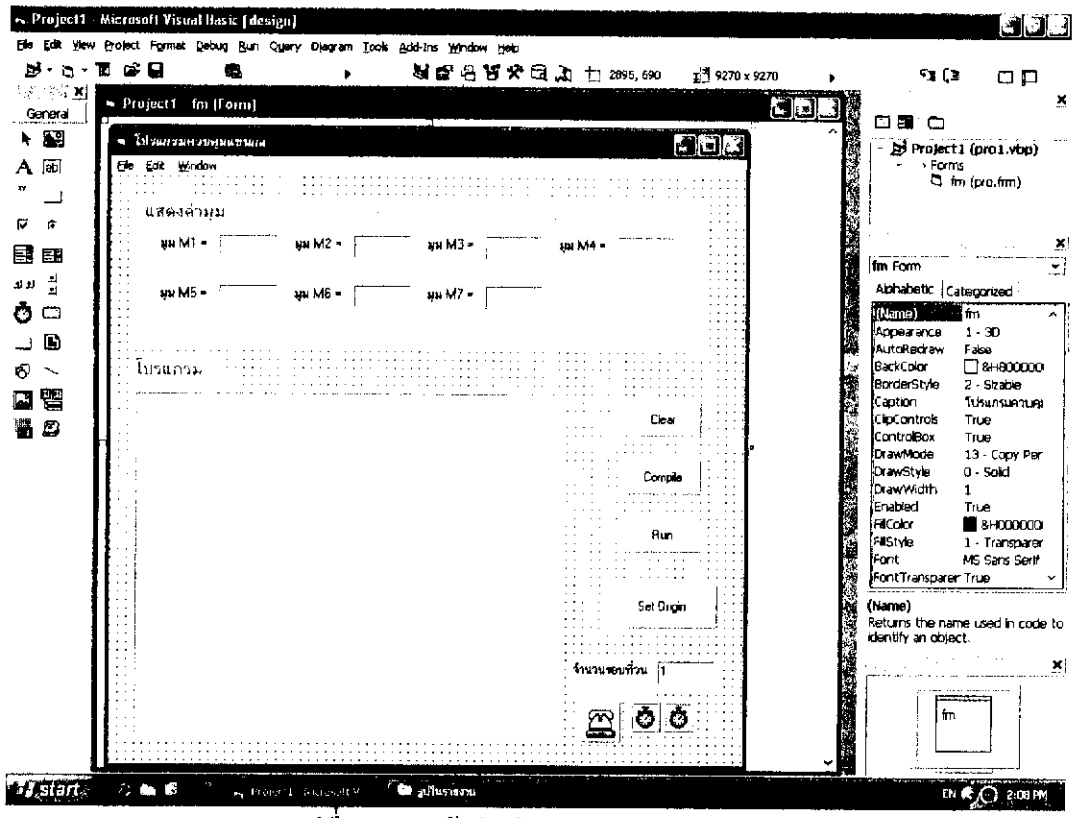
ในการออกแบบได้ แบ่งการทำโดยจะทำโปรแกรมในการเขียน Code ผ่าน Computer โปรแกรมควบคุม Controller และออกแบบแผงวงจร Circuits ในกล่องควบคุม



รูปที่ 3.11 แนวคิดในการออกแบบระบบควบคุม

3.5 การออกแบบโปรแกรมในการควบคุมแขนกลจาก Computer

ในการออกแบบตัวโปรแกรมในการป้อนCode โดยติดต่อทางพอร์ตอนุกรมนี้ ได้ใช้โปรแกรม Visual Basic ในการเขียน ได้ออกแบบหน้าตาของตัวโปรแกรมดังนี้



รูปที่ 3.12 หน้าต่างโปรแกรม Visual Basic

ตารางที่ 3.2 ตารางการควบคุมแขนกลจาก Computer

Control	Properties	ค่าที่กำหนด
From	Name	Fm
	Caption	โปรแกรมควบคุมแขนกล
Frame	Name	Famq
	Caption	แสดงค่ามุม
Label	Name	la 1
	Caption	มุม M1 =
Label	Name	la 2
	Caption	มุม M2 =
Label	Name	la 3
	Caption	มุม M3 =

Label	Name	la 4
	Caption	มุม M4 =
Label	Name	la 5
	Caption	มุม M5 =
Label	Name	la 6
	Caption	มุม M6 =
Label	Name	la 7
	Caption	มุม M7 =
TextBox	Name	Txm1
	Caption	
TextBox	Name	Txm2
	Caption	
TextBox	Name	Txm3
	Caption	
TextBox	Name	Txm4
	Caption	
TextBox	Name	Txm5
	Caption	
TextBox	Name	Txm6
	Caption	
TextBox	Name	Txm7
	Caption	
Label	Name	Lapro
	Caption	โปรแกรม
TextBox	Name	Txpro
	Caption	
Command	Name	Comc
	Caption	Clear
Command	Name	Comp

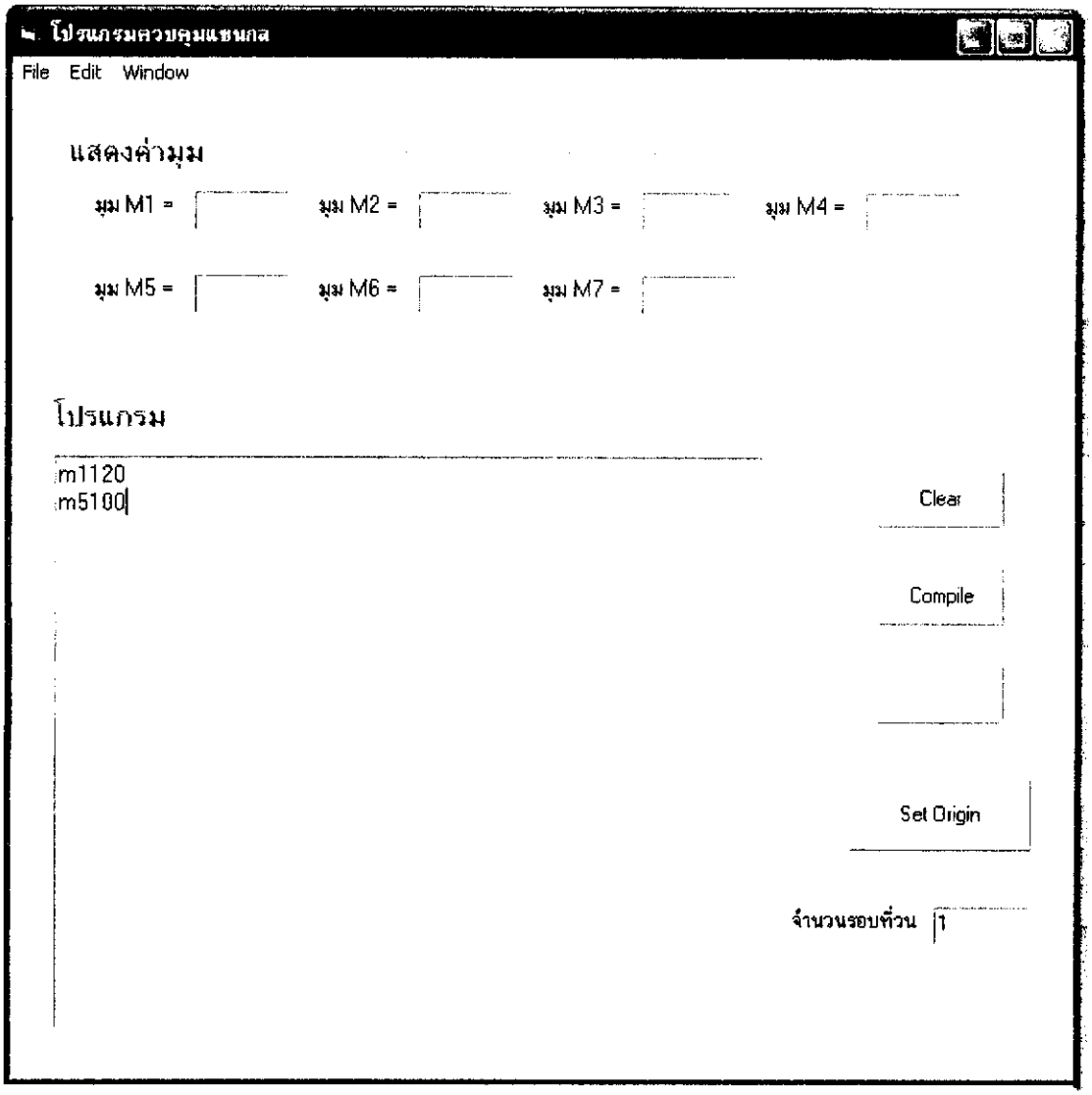
	Caption	Compile
Command	Name	Comr
	Caption	Run
Command	Name	Origin
	Caption	Set origin
Label	Name	Lanum
	Caption	จำนวนรอบที่วน
TextBox	Name	Txnum
	Caption	1
Timer	Name	Time 2
	MSComm	Name msc

Code คำสั่งเพื่อควบคุมการทำงานของโปรแกรมดังนี้

ภาคผนวก ค.1** Code คำสั่งเพื่อควบคุมการทำงานของโปรแกรม

คำอธิบาย

Code ที่ใช้ในการสั่งงานแขนกลได้สร้าง Code ขึ้นมาเอง โดยกำหนดว่าให้ตัวแรกเป็นอักษร m ตัวที่ 2 เป็นตัวบอกว่าเป็นมอเตอร์ตัวที่เท่าไรส่วน ตัวที่ 3 ถึงตัวที่ 5 เป็นค่าที่บอกว่าให้หมุนเป็นมุมเท่าไร เช่น โปรแกรมตัวอย่าง



รูปที่ 3.13 หน้าต่างโปรแกรมควบคุมแขนกล

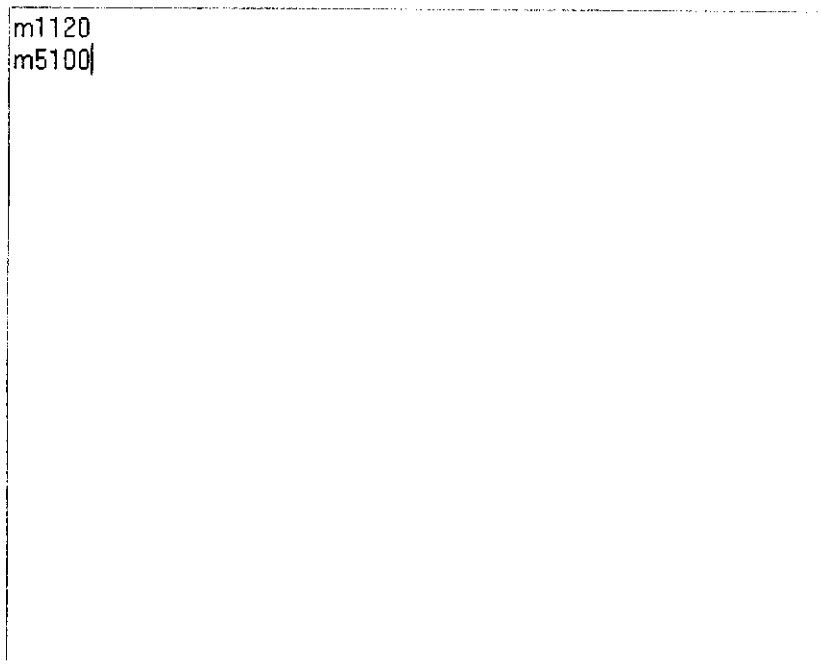
บรรทัดแรก m1120 หมายความว่ามอเตอร์ตัวที่1 หมุนไป 120 องศา โดยที่ตัว m เป็นตัวเช็คว่า ถูกต้องตามCode

บรรทัดที่ 2 m5110 หมายความว่า มอเตอร์ตัวที่5 หมุนไป 100 องศา

Code ตัวบนนี้สามารถอธิบายโปรแกรมได้ดังนี้

พื้นที่ป้อนโปรแกรมใช้สำหรับป้อนโปรแกรมดังอธิบายไปแล้วข้างต้น

โปรแกรม



รูปที่ 3.14 พื้นที่ป้อนโปรแกรม

ปุ่ม 

เป็นปุ่มที่ใช้ในการเคลียร์โปรแกรมมี Code ดังนี้

```
Private  Sub comc_Cick( )
        Txpro.Text = ""
End  Sub
```

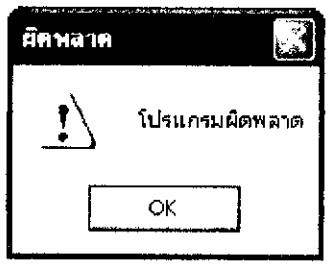
เมื่อนำการป้อนโปรแกรมตัวอย่างข้างต้นแล้วต่อไปก็จะมาทำการคอมไพล์ Code ที่ป้อนไปว่าถูกต้องหรือไม่โดยใช้

ปุ่ม 

มี Code ดังนี้

ภาคผนวก ค.2** Code คำสั่งปุ่ม Compile

โดยโปรแกรมจะเช็คความยาวของ Code ที่ป้อนเข้าไป โดยเช็คว่าอักษรตัวแรกเป็นตัวอักษร m หรือไม่ และค่าตัวเลขที่ 2 ที่บอกว่ามอเตอร์ตัวใด อยู่ระหว่างค่า 0 - 7 หรือไม่ และตัวเลข 3 ตัวท้ายที่บอกขนาดมุมมีค่า 0 หรือระหว่าง 0 –186 หรือไม่ ถ้าไม่อยู่โปรแกรมจะแสดงความผิดพลาดมาดังนี้



เมื่อทำการ Compile เสร็จ ต่อไปจะเป็นการรับส่งค่าไปยัง Controller โดยกดปุ่ม  โดยมี Code ดังนี้

*****ภาคผนวก ค.3** Code คำสั่งปุ่ม Run***

เมื่อทำการกดปุ่ม Run แล้ว จะทำโปรแกรมย่อย run( ) โดยโปรแกรมย่อย run( ) จะทำหน้าที่ในการรับส่งข้อมูลโดยจะแยกตัว เลขออกเป็นหมวดหมู่ โดยตัวที่บอกว่ามอเตอร์ตัวที่เท่าไรจะอยู่ในตัวแปร n(t;) โดยที่ t; จะเริ่มจาก 0 จนถึง 1000 คำสั่ง

จากโปรแกรมข้างต้น จะส่งหมายเลข 1 ออกไปก่อนให้ controller รับทราบว่า เป็นมอเตอร์ตัวที่ 1 จากนั้น controller จะตอบกลับ หมายเลข 1 มา เมื่อ Computer รับหมายเลข 1 กลับมาเป็นการเช็คว่าได้ส่งข้อมูลถูกต้อง แล้วก็จะส่งค่ามุมออกไปก็คือ 120 ออกไป Controller อีก และ Controller จะรับทราบอีกว่าได้ส่งข้อมูลมุมขนาด 120 องศา และ Controller จะนำค่ามุมไปทำงานสั่ง Motor เมื่อสั่ง motor เสร็จก็จะสั่ง ค่า 120 กลับมายัง Computer อีก เพื่อเป็นการแจ้งว่าส่งข้อมูลเสร็จสิ้น จากนั้นก็จะส่งตามขั้นตอนนี้วนไปจนครบ Code ที่ป้อนเข้ามา

ปัญหาในการส่ง คือ ถ้ามุมมากกว่า 127 องศา Computer จะไม่สามารถส่งข้อมูลได้ ต้องแก้โดยถ้าข้อมูลมากกว่า 127 องศา ให้ส่งค่าตัวอักษร e ไปก่อน ให้ Controller ทราบว่ามุมมากกว่า 127 องศา และจากนั้น Controller จะส่งค่า e กลับมา และ Computer จะส่งค่า 2 ตัวท้ายออกไป เช่น 170 องศา 2 ตัวท้ายคือ 70 ก็จะส่งค่า 70 ออกไป Controller ก็จะทราบว่าค่าที่ส่งมามีค่า 70 และต้องบวกกับ 100 เพื่อจะสั่ง Motor ให้ทำการหมุนไป 170 องศา

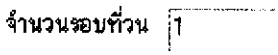
ปุ่ม 

ใช้ในการ Set แขนกลให้อยู่ในสภาวะเริ่มต้นทำงานนั่นคือ สภาวะ Home มี Code ดังนี้

*****ภาคผนวก ค.4** Code คำสั่งปุ่ม Set Origin***

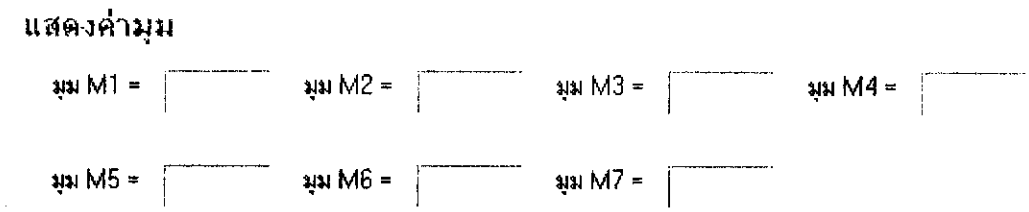
โดยจะทำการ Set Motor ทั้ง 7 ตัว ให้อยู่สภาวะเริ่มใช้งานโดยกำหนดค่ามุมคงที่ในโปรแกรมสำหรับ Motor แต่ละตัว

การ Set จำนวนรอบที่จะใช้งาน



เป็นการใส่ค่าว่าต้องกาทำฟังก์ชันเดิมจำนวนกี่ครั้ง

ส่วนแสดงผล



รูปที่ 3.15 ส่วนแสดงผล

เป็นส่วนที่รับค่ามุมต่างๆ มาจาก Controller เพื่อทราบว่าขณะนี้ Motor แต่ละตัวมีมุมขนาดเท่าใด

3.6 การออกแบบ Micro Controller ในการควบคุมแขนกล

ในการออกแบบ Controller เราใช้ MPLAB ในการเขียนโปรแกรมภาษา C ควบคุม MicroController dsPIC30F2010 ขนาด 28 ขา โดย dsPIC มีโมดูลต่างๆมากมายแต่ มีโมดูลที่เราใช้คือ ไทเมอร์ 1 ใช้ในการสร้างสัญญาณพัลส์ เพื่อควบคุม Servo Motor โมดูล UART ในการติดต่อสื่อสารกับพอร์ตอนุกรม และโมดูลแปลงสัญญาณอะนาลอกเป็นดิจิตอล ซึ่งเขียนโปรแกรมได้ดังนี้

*****ภาคผนวก ค.5** โปรแกรม dsPIC ***

จากโปรแกรมข้างต้นจะมีการ interrupt 2 แหล่งคือ interrupt ของ Timmer และ interrupt ของ UART

ISR – T1Interrupt (Void)

ISR – U1TXInterrupt (Void)

ISR – U1TXInterrupt (Void)

ในโปรแกรมได้กำหนดให้ พอร์ต RE0 – RE4 เป็น Output ควบคุม Servo Motor ส่วน RE5, RE8 และ RC14, RD1 เป็น Output ควบคุม DC Motor ได้กำหนด Timmer 1 ใช้สัญญาณนาฬิกา 7.3728 MHz ที่ PLL 4X ดังนั้นคาบเวลาของสัญญาณพัลส์ที่ต้องการคือ 15 ms

$$T = \frac{\text{match Value}}{7372800} = \frac{110592}{7372800} = 0.015 = 15 \text{ ms} \quad (3.1)$$

$$f = \frac{1}{T} = \frac{1}{0.015} = 66.67 \text{ Hz} \quad (3.2)$$

ซึ่งกำหนดค่า match_Value คือ 110592 จะได้คาบ 15 ms ดังโปรแกรมคือ

```
ConfigIntTimer1(T1_INT_PRIOR_1 &          // Timer1 interrupt priority 1
                          T1_INT_ON);     // Enable interrupt for timer1
    WriteTimer1(0);                       // Clear count value at TMR1 register
    match_value = 110592;          // Load value Interval 15 ms
    OpenTimer1(T1_ON &                    // Start timer1
               T1_GATE_OFF &              // Disable gate pin for timer1
               T1_IDLE_STOP &             // Stop timer in idle mode
               T1_PS_1_1 &                // Presceler 1:1
               T1_SYNC_EXT_OFF &          // Disable sync external source
               T1_SOURCE_INT, match_value); // Wait till the timer matches
                                              with the period value
```

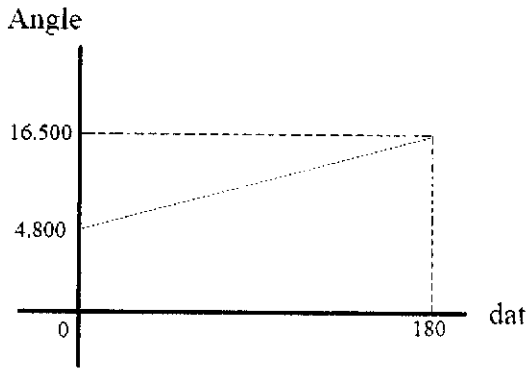
จากนั้นกำหนด initial ที่ใช้คือ uart1_init( ); ใช้ในการติดต่อพอร์ตอนุกรม และ adc_init( ); ใช้ในการอ่านค่า A/D แล้วำทการอ่านค่า A/D channel AN0 และ AN1 ขาที่ 2 และ 3 ของ dsPIC เก็บในตัวแปร pay[0] , pay [1] จากนั้นจะแก้ฟังก์ชั่น while(1) เพื่อ เช็คว่ามีข้อมูลเข้ามาจาก Computer หรือไม่ เมื่อมีข้อมูลเข้ามาตัวแรกคือค่าข้อมูลตั้งแต่ค่า 1 ถึง 7 เพื่อเช็คว่าเป็น Motor ตัวที่เท่าไร เก็บในตัวแปร dat

```
dat = Read UART1( ); // รับค่าจากพอร์ตอนุกรม
```

เมื่อรับค่ามาแล้วจะส่งค่ากลับออกไปด้วยคำสั่ง WriteUART1(dat); ถ้าเป็น Motor ตัวที่ 1- 5 (dat = 1-5) จะเขียนฟังก์ชั่นคล้ายกันแตกต่างกันตรงที่ Output ไปใช้งาน ยกตัวอย่าง ฟังก์ชั่นรับข้อมูลและสั่งงาน Servo Motor ตัวที่ 1 (dat =1)

*****ภาคผนวก ค.6** ฟังก์ชั่นรับข้อมูลและสั่งงาน Servo Motor ตัวที่ 1***

เมื่อส่งค่าว่าเป็น Motor ตัวใดแล้วก็จะรอรับค่า มุมที่ส่งมา ถ้ามุมมีขนาดมากกว่า 127 องศาก็จะรอรับค่า e เมื่อมีค่ามุมเข้ามา จะเก็บในตัวแปร dat แล้วทำการแปลงค่าเพื่อควบคุมค่าดิวตี้ไซเคิล ใช้ในการส่งสัญญาณพัลส์ ควบคุม Servo Motor



รูปที่ 3.16 กราฟ Angle / dat

Angle = (65x(dat))+4800;

ซึ่งตัวแปร angle จะส่งค่าไปยังฟังก์ชั่นย่อย pwm1(angle); ดังนั้น

```
void pwm1(long num)
{
        long i;
        for(i=0;i<400000;i++)
                {timer_value = ReadTimer1();    // Read counter value
                if( timer_value <= num )
                        {PORTEbits.RE0 = 1;    // turn off LED on RD1 }
                else
                        {PORTEbits.RE0 = 0;    // turn off LED on RD1 }
                }
}
```

ซึ่งฟังก์ชั่นข้างบนนี้จะเป็นฟังก์ชั่นที่ใช้สร้างสัญญาณพัลส์ โดยควบคุมค่าดิวตี้ไซเคิลจากค่าที่ผ่านตัวแปร num เข้ามาโดยใช้ Timer 1 เป็นตัวนับค่าเวลาแล้วส่งสัญญาณออกจากพอร์ต RE0 – RE4

เมื่อออกจากฟังก์ชั่น pwn1( ); แล้วก็จะส่งค่ามุมเข้ามากลับออกไปให้ Computer เพื่อเป็นการเช็คว่าได้ทำงานเสร็จสิ้นแล้วก็จะรอรับค่าตัวใหม่อีกต่อไป

ในกรณีที่ค่า dat = 6 และ dat = 7 หมายถึง Motor ตัวที่ 6 และ 7 ซึ่งเป็น DC Motor และเช็คค่ามุมด้วย โพเทนทิโอมิเตอร์ ผ่านทาง A/D จะมีโปรแกรมดังนี้

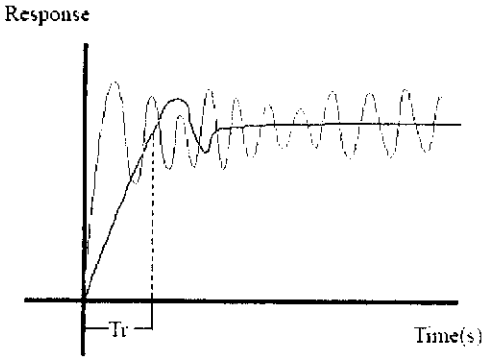
*****ภาคผนวก ค.7** ฟังก์ชั่นรับข้อมูลและสั่งงาน DC Motor ตัวที่ 6-7***

เมื่อรับค่า 6 มาแล้วก็จะทราบว่าเป็น Motor ตัวที่ 6 จากนั้นจะส่งค่า 6 ออกไป ก็จะรอรับค่ามุม เมื่อรับค่ามุมแล้วจะแปลงเก็บค่าในตัวแปร Angle แล้วเอาไปเปรียบเทียบกับค่า pay[0] เมื่อ

ให้รู้ว่า มุมที่สั่งเข้ามามีค่ามากกว่าหรือน้อยกว่า มุมของแขนกลขณะนั้น ซึ่งค่า pay[0] เป็นค่าที่มาจาก A/D แล้วทำการแปลงค่าจนนิ่งด้วยฟังก์ชั่น

```
// read charnal 0 //
if (result[1] > old_result[1])
{
        tx = ( result[1] - old_result[1] )>>4;
        old_result[1] = ( old_result[1] + tx );
}
if (result[1] < old_result[1] )
{
        tx  = ( old_result[1] - result[1] )>>4;
        old_result[1] = ( old_result[1] - tx
}
// show LCD chanal 0,1 //--
pay[1] = (old_result[1]-15)*be ;
```

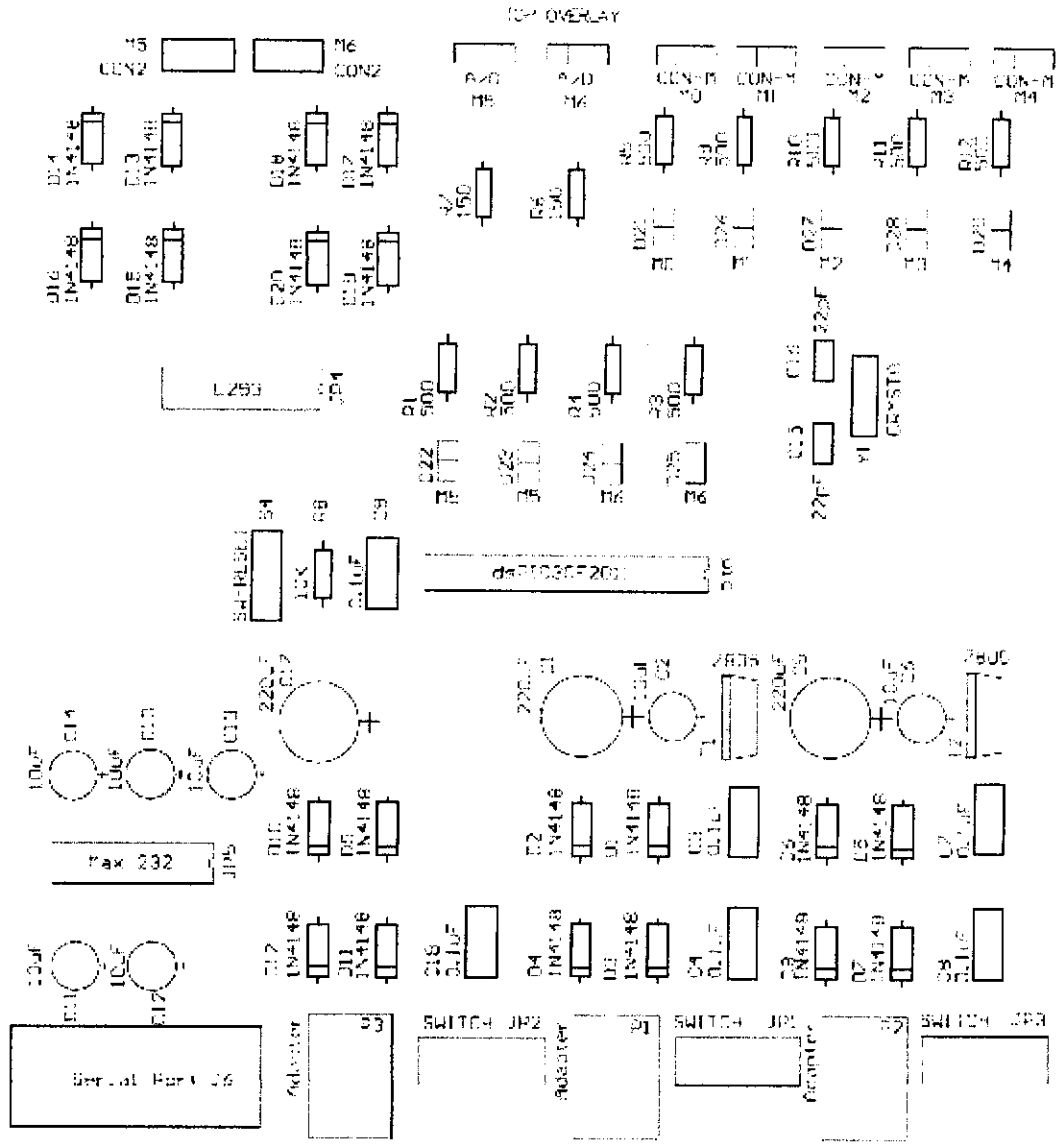
สัญญาณ A/D ก่อนแปลงและหลังแปลงค่า



รูปที่ 3.17 กราฟ Response / Time

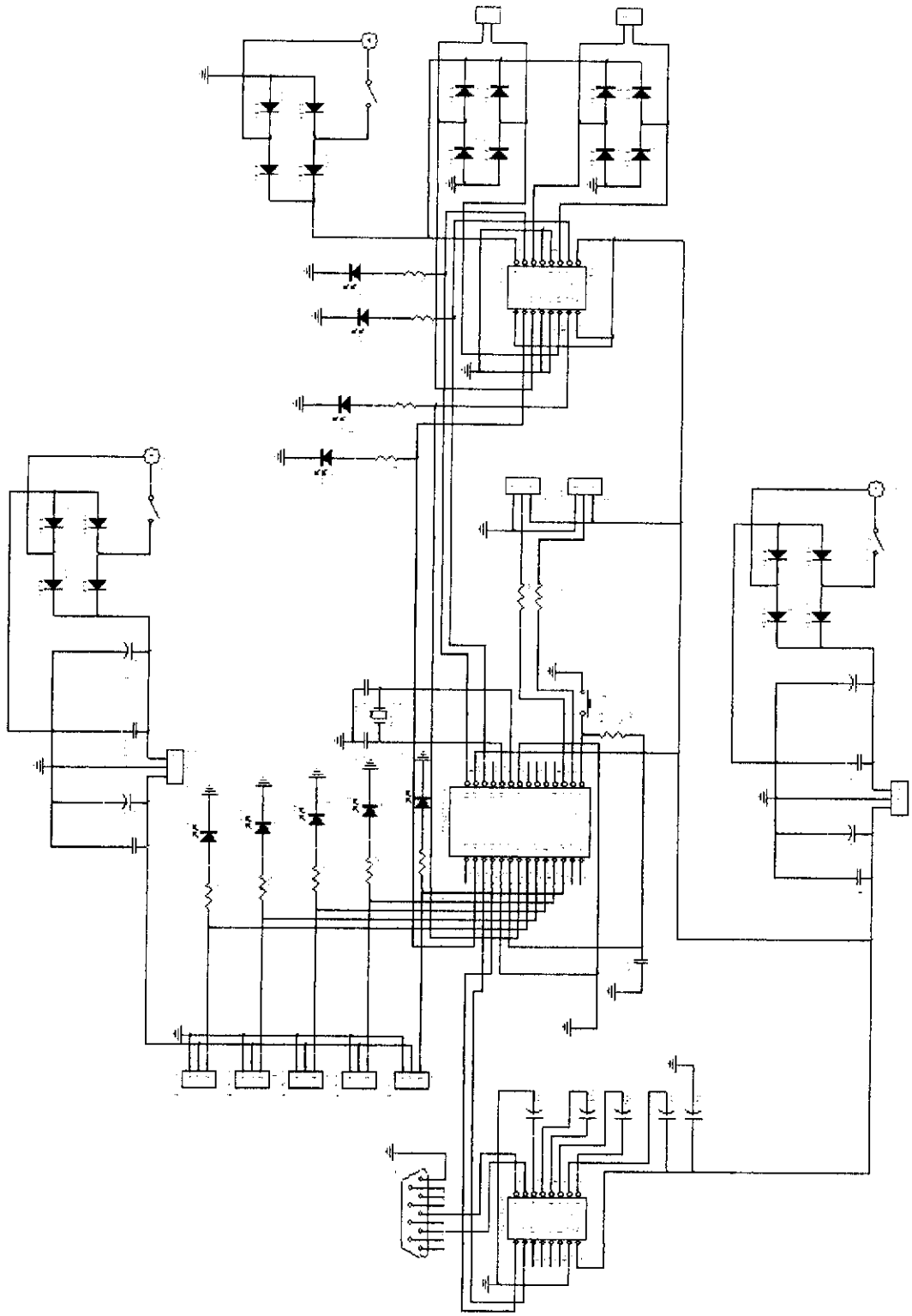
พบว่าหลังแปลงค่า Rise time จะมีค่ามากทำให้เกิดการ Delay ของสัญญาณเมื่อเช็คค่ามุมเสร็จแล้วก็จะ ส่งสัญญาณสั่ง DC Motor ให้หมุนทวนเข็มหรือตามเข็มนาฬิกาโดยออกทางขา RE5, RE8, RC14, RD1 จากนั้นก็จะส่งค่ามุมกลับไปยังคอมพิวเตอร์เพื่อเช็คว่าได้ทำงานเสร็จแล้วและรอรับสัญญาณตัวอื่นต่อไป

วงจรควบคุมแขนกล



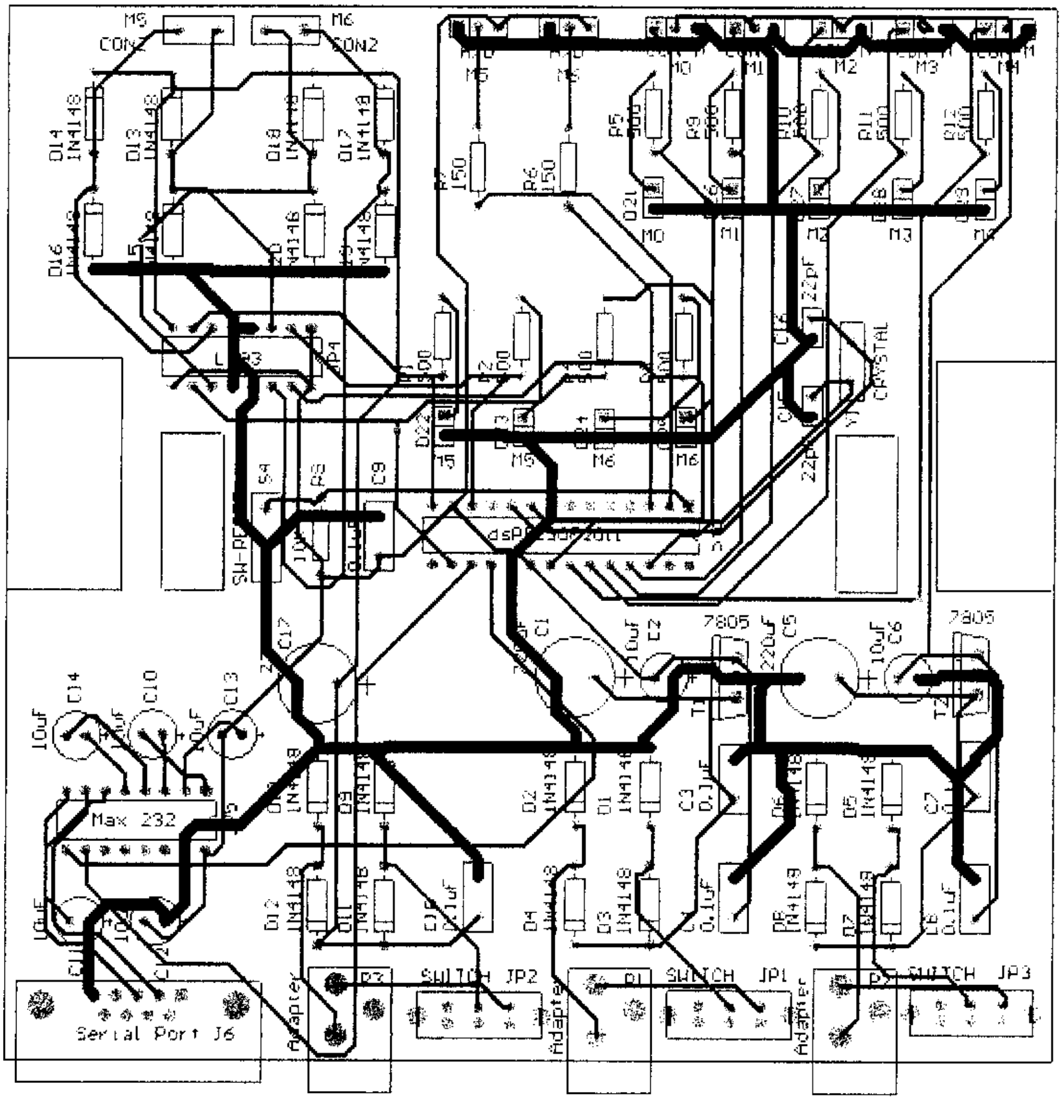
รูปที่ 3.18 อุปการณ์ต่างๆ ภายในลายวงจร

วงจร Controller dsPIC



รูปที่ 3.19 ลายวงจรของกล่อง Controller

ลายวงจร PCB

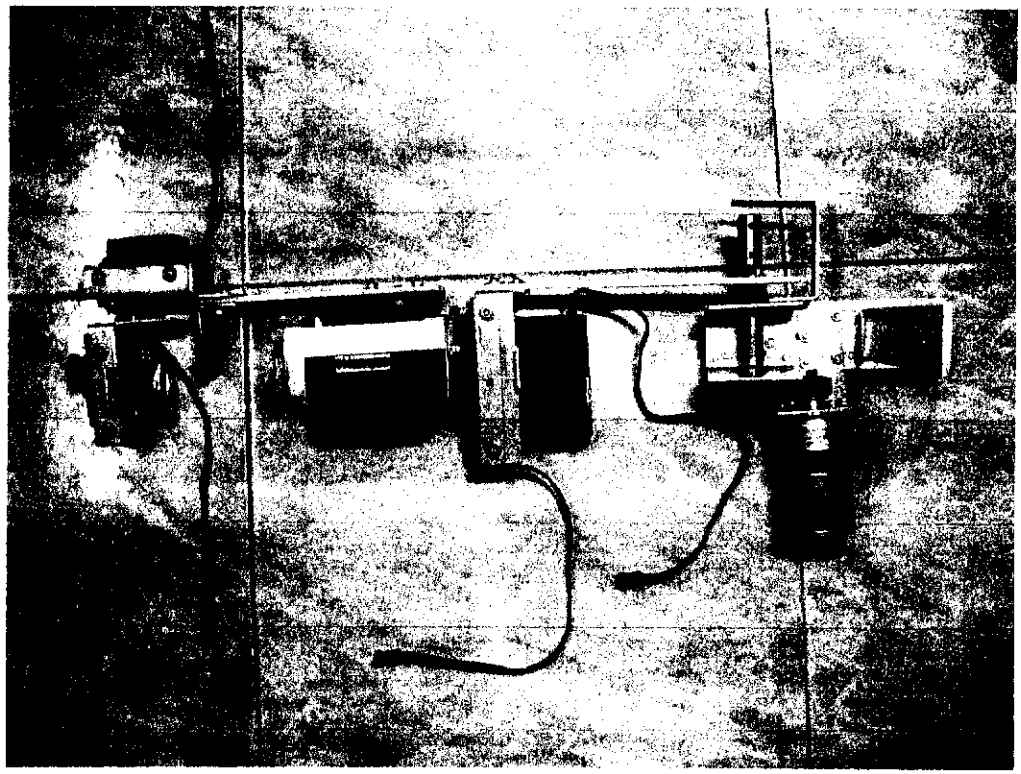


รูปที่ 3.20 ลายวงจร PCB

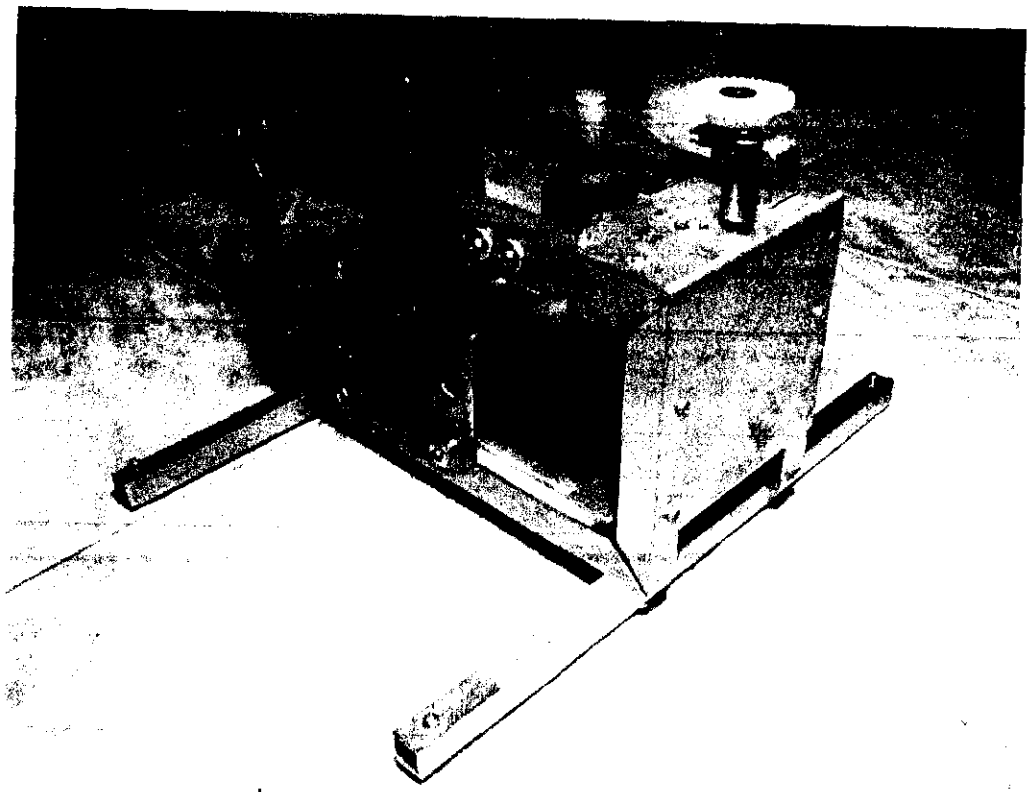
บทที่ 4
ผลจากการออกแบบและทดลองใช้งาน

4.1 การประกอบชิ้นงานและผลจากการออกแบบ

จากการออกแบบคำนวณตามหลัก Mechanic แล้ว ได้เลือกวัสดุที่ใช้ทำโครงสร้างของแขนกลโดยได้ใช้ อลูมิเนียม เป็นส่วนประกอบหลัก และใช้น็อต ขนาด 3 mm. เป็นตัวยึดส่วนประกอบต่างๆ วัสดุและชิ้นส่วนต่างๆ ที่ได้จากการออกแบบและสร้างขึ้น ได้นำมาประกอบด้วยกันตาม Model ที่ออกแบบ พบว่าวัสดุแต่ละชิ้นสามารถประกอบเข้ากันได้ และมีความแข็งแรงและคงรูปได้โดยไม่มีการหักหรือหลุดออก แต่อาจมีบางชิ้นส่วนที่อาจจะมีการเบี้ยวได้เนื่องจากความคลาดเคลื่อนของวัสดุ ที่ทำมาจากพลาสติกโดยเฉพาะบริเวณข้อต่อกับ Servo Motor แต่ก็ถือว่าเป็นความผิดพลาดเล็กน้อย โดยสามารถปรับปรุงแก้ไขความคลาดเคลื่อนเหล่านี้ได้โดยการโปรแกรมและออกแบบวงจรชดเชย



รูปที่ 4.1 รูปของแขนกลที่ประกอบเสร็จเรียบร้อยแล้ว



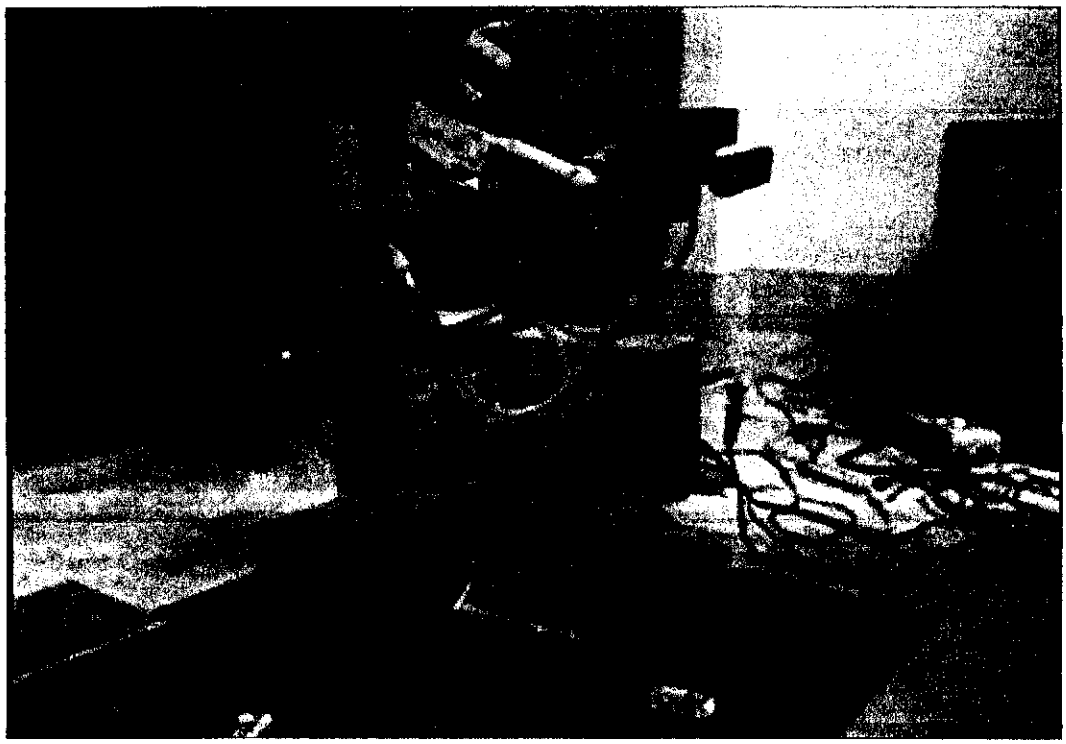
รูปที่ 4.2 รูปของฐานที่ประกอบเสร็จเรียบร้อยแล้ว

4.2 การทดลองระบบควบคุม

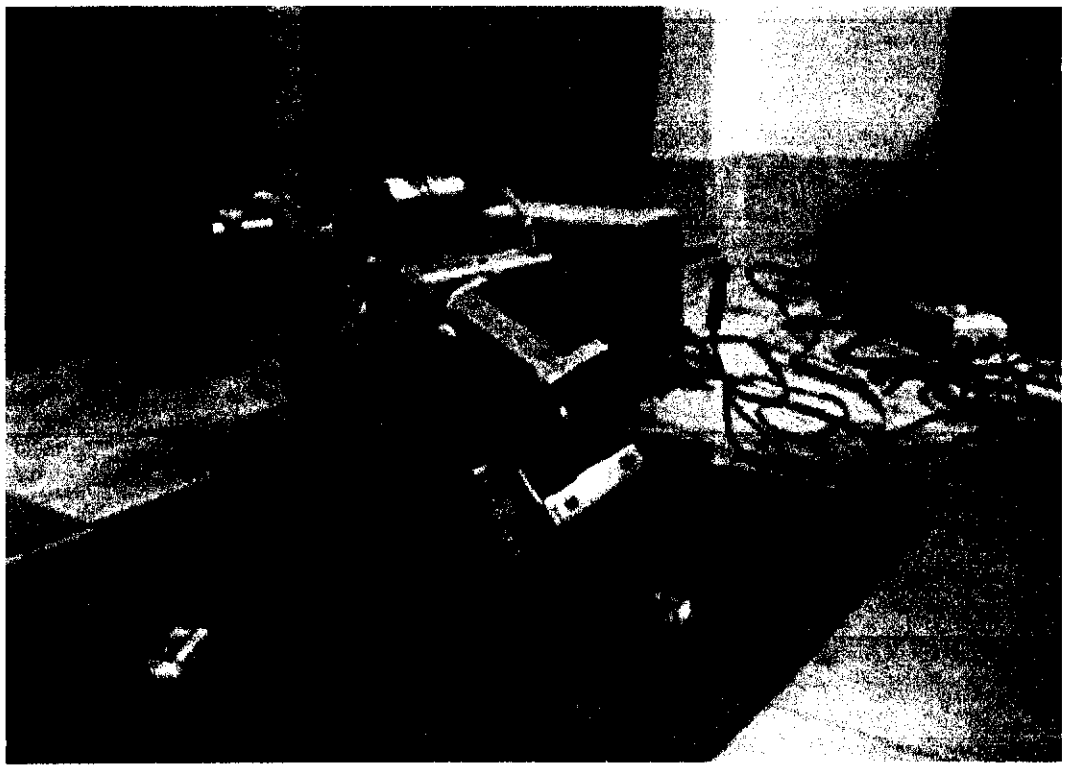
เมื่อเขียนโปรแกรมเสร็จแล้วทำการประกอบวงจรลงกล่อง Control จากนั้นทำการติดตั้งระบบ Control และตัวแขนกลทดลองเขียน Code เพื่อสั่งแขนกล พบว่าแขนกลสามารถทำงานได้ดีมีความละเอียดในการทำงานสูง แต่จะมีปัญหาตรงที่ Motor ตัวที่ 6 และ 7 ซึ่งเป็น DC Motor และเช็คมุมด้วยโพเทนทิโอมิเตอร์ จะมีความแม่นยำต่ำเพราะว่าสัญญาณจาก A/D มีค่า Rise time สูงจึงทำให้สัญญาณ Delay แต่ก็ยังทำงานได้ดี ถึงแม้ความแม่นยำไม่เท่า Servo Motor

โดยเราทดลองจับวัตถุบนพื้น โดยเราใช้ยางลบวางไว้ที่ตำแหน่งที่ 1 และที่ 2 จากนั้นทำการป้อน Code ผ่านโปรแกรมเพื่อควบคุมแขนกล ให้หยิบยางลบก้อนแรกวางไว้บนแท่น จากนั้นหยิบก้อนที่สอง วางไว้บนแท่นเช่นเดียวกัน โดยเป็นการทดลองควบคุมโดยเราเขียน Code ได้ดัง

m7145	m6130	m4180	m3060	m2045	m5060
m2055	m6072	m1025	m6125	m7060	m3090
m6092	m1090	m6120	m7100	m2090	m5090
m6080	m3100	m1025	m6130	m7060	m5060
m2055	m3090	m6095	m3120	m5070	m3110
m1090	m6120	m7120			

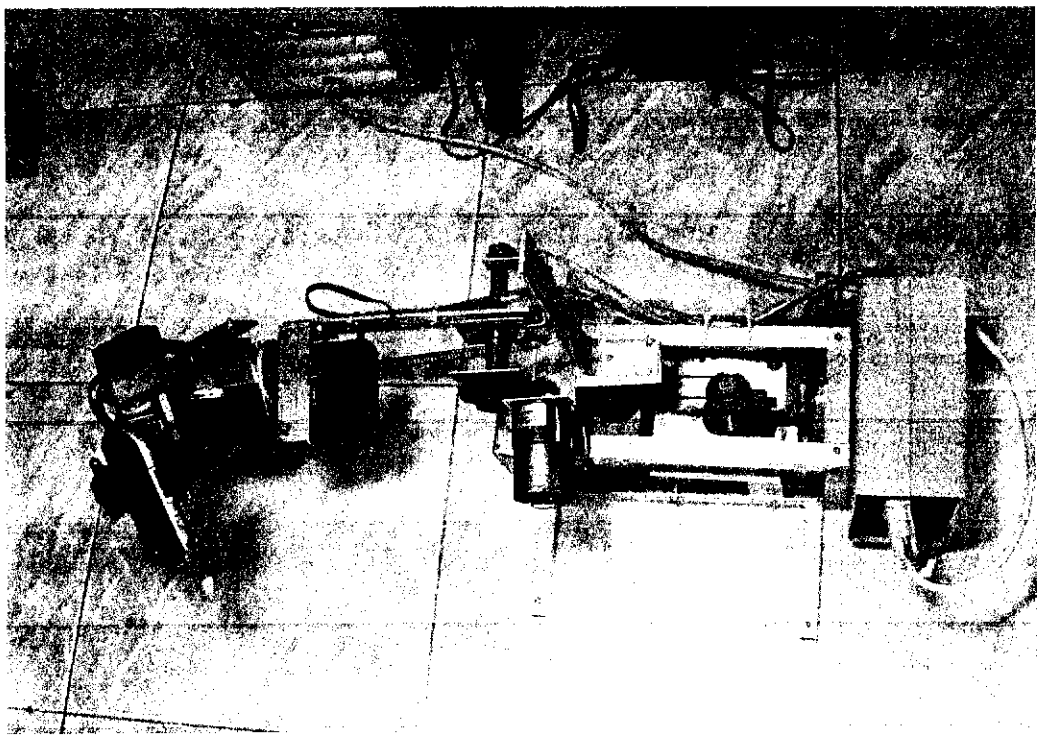


รูปที่ 4.3 ภาพการทดลอง 1



รูปที่ 4.4 ภาพการทดลอง 2

รูปที่ 4.5 ภาพการทดลอง 3



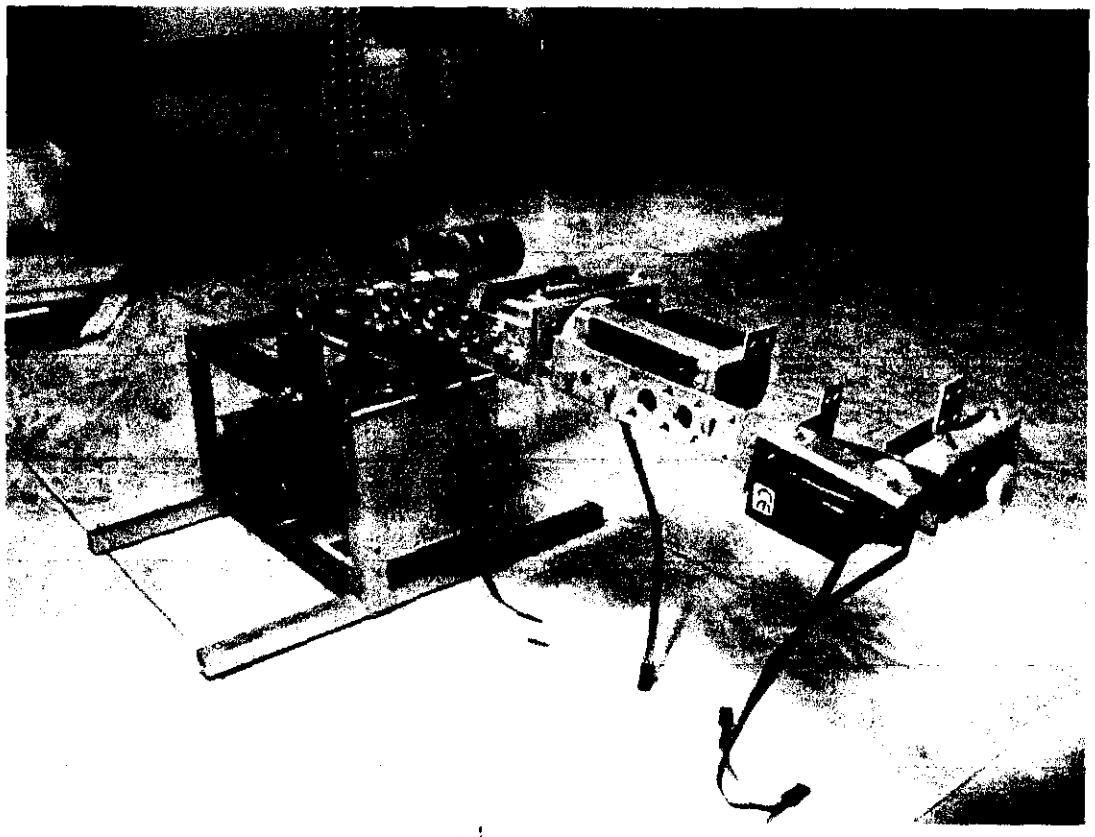
รูปที่ 4.6 ทดลองจับวัสดุอื่น

บทที่ 5
วิจารณ์และสรุปผล

5.1 สรุปผลการทดลอง

พบว่าหลักการที่ใช้ในการคำนวณและทฤษฎีต่างๆ ที่นำมาใช้ในการออกแบบสามารถสร้างแขนกลได้ โดยอาศัยการออกแบบชิ้นส่วนต่างๆ ตามแนวคิด โดยอ้างอิงถึงทฤษฎีค่าความแข็งแรงของชิ้นส่วนนั้น แล้วสามารถประกอบ ชิ้นส่วนที่ออกแบบนั้นได้อย่างถูกต้อง และใกล้เคียงกับต้นแบบ(Model) ที่เขียนโดยใช้ Solid Works และแขนกลที่สร้างขึ้นมีความแข็งแรงพร้อมที่จะใช้งานได้ เพียงแต่ต้องมีวงจรระบบควบคุม และโปรแกรมที่ใช้ทำงาน แขนกลนี้มีความอิสระในการหมุนได้ 6 ข้อต่อ โดยใช้ Servo Motor ทั้งหมด 4 ตัว และ DC Motor ทั้งหมด 2 ตัว และมีฐานตั้งสามารถปรับแต่งระดับได้

จากการทดลองสั่งงานแขนกลพบว่า แขนกลสามารถทำงานได้ดีมีความแม่นยำสูง โดย Servo Motor สามารถคุมความละเอียดได้ 1 องศา ส่วน DC Motor ยังมีความผิดพลาดแต่การทำงานถือว่ายอมรับได้ โครงงานชิ้นนี้ประกอบด้วย แขนกลและระบบ Control โดยติดต่อกับ Computer โดยผ่านพอร์ตอนุกรม พบว่าสามารถทำงานได้



รูปที่ 5.1 แขนกล

5.2 ปัญหาที่พบ

ปัญหาที่พบนี้ก็คือ ขั้นตอนในการสร้างชิ้นส่วนต่างๆ อาจจะมีความคลาดเคลื่อนในการทำขึ้น ทั้งนี้เพราะชิ้นส่วนแต่ละชิ้นทำจากมือ โดยการวัดขนาดชิ้นส่วนต่างๆ อาจเกิดความคลาดเคลื่อนได้ แต่ทั้งนี้มีผลกระทบกระเทือนต่อความสมดุลของแขนน้อยมาก และปัญหาอีกประการหนึ่งคือขั้นตอนในการประกอบชิ้นงานอาจจะเกิดการเบี้ยวของโครงสร้างฐาน แต่ก็มีผลกระทบเล็กน้อย และคณะผู้จัดทำได้แก้ปัญหาโดยการทำให้โครงสร้างฐานมีการปรับระดับกับพื้นได้

ปัญหาที่พบในการควบคุมแขนกลคือ ตรง Motor ตัวที่ 6 และ 7 ซึ่งเป็น DC Motor โดยเช็คมุมเป็น A/D โดยโพเทนทิโอมิเตอร์ พบว่าค่าเช็คมุมที่ A/D มีความล่าช้ากว่าความเร็วของ Motor ในขณะหมุน ทำให้ค่าที่ได้ Delay กว่าค่าที่เป็นจริงเนื่องจากค่า Rise time ของ A/D มีค่ามากกว่าแต่มีความแม่นยำสูง ซึ่งสามารถแก้ปัญหานี้ได้โดยการชดเชยในโปรแกรม ทำให้แขนกลทำงานได้

5.3 ขอบเขตการดำเนินงานและแนวทางการพัฒนา

ในเทอมนี้ได้สร้างแขนกลเสร็จแล้วโดยมีวงจร Control และโปรแกรมที่ใช้ควบคุมจากโครงงานนี้ สามารถพัฒนาได้โดยทำให้แขนกลมีความละเอียดมากขึ้นและทำให้แขนกลมีความเร็วในการทำงานเพิ่มขึ้น และสามรถพัฒนาแขนกลนี้ไปใช้งานในอุตสาหกรรมขนาดเล็กได้ เช่นอาจนำไปใช้ในการหยิบจับสิ่งของต่างๆ ได้โดยแค่ป้อน Code เข้าไปใน Computer เท่านั้น และสามารถพัฒนาเป็นแขนกลขนาดใหญ่เพื่อใช้งานในอุตสาหกรรมขนาดใหญ่ได้

5.4 ประโยชน์ที่ได้รับจากโครงงาน

1. สามารถนำความรู้ที่เรียนนำมาสร้างชิ้นงานโครงสร้างแขนได้จริง
2. สอนให้สามารถทำงานกันเป็นทีมได้ ฝึกความสามัคคี และรับผิดชอบ
3. ฝึกกระบวนการคิดในการทำงาน
4. รู้จักกับอุปกรณ์ ต่างๆทางอุตสาหกรรมในขณะประกอบชิ้นงาน
5. รู้จักแก้ปัญหาที่เกิดขึ้นระหว่างทำงาน
6. สามารถนำแนวคิดไปพัฒนาประยุกต์ในการทำงานต่อไป

เอกสารอ้างอิง

[1] สัจจะ จรัสรุ่งรวีวร. **คู่มือเขียนโปรแกรม Visual Basic 6.** (พิมพ์ครั้งที่ 3). นนทบุรี : สำนักพิมพ์ DEV BOOK, 2548.

[2] ทวีศักดิ์ ศรีช่วย. **Solid Works.** (พิมพ์ครั้งที่ 1). กรุงเทพฯ : สำนักพิมพ์ ส.ส.ท., 2546.

[3] อภิชาติ ภู่พรับ. **การควบคุมฮาร์ดแวร์ด้วย Visual Basic.** (พิมพ์ครั้งที่ 1). นนทบุรี : สำนักพิมพ์ DEV BOOK, 2546.

[4] ศ.รด.วริทธิ์ ฮึ้งพากรณ์ , รศ.ชาน ถนัดงาน. **การออกแบบเครื่องจักรกล เล่ม 1.** กรุงเทพฯ : สำนักพิมพ์SE-ED Ucationจำกัด, 2546.

[5] นคร ภักดีชาติ , ชัยวัฒน์ ลิ้มพรจิตรวิไล. **คู่มือการทดลอง dsPIC Microcontroller เบื้องต้นด้วยโปรแกรมภาษา C กับ MPLAB C30.** กรุงเทพฯ : บ.อินโนเวตีฟ เอ็กเพอริเมนต์ จำกัด, 2547.

[6] ทีมวิชาการโรงเรียนแสงทองอีเลคโทรนิคส์. **การออกแบบแผ่นวงจรพิมพ์ด้วยโปรแกรม Protel 99 SE.** กรุงเทพฯ : โรงเรียนแสงทองอีเลคโทรนิคส์, 2547.

[7] John J.Craig. **Introduction To Robotics Mechanic & Control.** (Third Edition). USA. : PEARSON Education, 2005.

ภาคผนวก ก.1

คุณสมบัติทางกลของอลูมิเนียมผสมเหนียวบางชนิด

ตารางที่ ก.1 คุณสมบัติทางกลของอลูมิเนียมผสมเหนียวบางชนิด

Alloy and Temper	Tensile Str., ksi	Yield Str., ksi	Elong. in 2 in., % 1/16 in. Thick, Sheet	Elong., in 2 in. or 4 Dia. Rod, % 1/2 in. Dia. Rod	Brinell Hard., BHN 500 kg. 10 mm ball	Shearing Str., ksi	Endur. Limit ksi Based on 5×10^8 cyc.	Corrosion Resist. General	Corrosion Resist. Stress Cracking	Cold Working	Machinability	Brazing	Gas Welding	Arc Welding	Resistance Welding	Some Applications
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	A	A	A	E	A	A	A	D	deat exchangers, pressure and
-O,H	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	A	A	A	D	A	A	A	A	storage tanks, chemical equipment,
-H18	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	A	A	C	D	A	A	A	A	reflectors, sheetmetal work.
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	B	C	C	A	B	D	D	D	Screw machine products
-T4	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	D	A	D	A	B	D	D	D	
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-	-	-	-	-	-	-	-	Heavy duty forgings, power shovel
-T6	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	D	C	C	B	D	D	B	B	bails, aircraft fittings, structural
-T6	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	D	C	C	B	D	D	B	B	members, aircraft structures,
																truck frames.
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-	-	-	-	-	-	-	-	Screw machine products
-T4	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	D	C	C	B	D	D	B	B	
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-	-	-	-	-	-	-	-	Cylinder heads, pistons
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-	-	-	-	-	-	-	-	Screw machine products, truck
-T4	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	B	C	C	B	D	D	B	B	wheels, aircraft structural parts
-T6	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	B	A	D	B	D	D	B	B	
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	A	A	A	B	A	A	A	B	Ductwork, truck panels, builders hard-
-H14	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	A	A	A	C	A	A	A	A	ware, chemical eq'ts., pressure vessels,
																storage tanks, piping, architectural apl.
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	A	A	A	D	B	A	A	B	Appliances, cooking utensils,
-H14	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	A	A	A	C	B	A	A	A	architectural applications,
-H18	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	A	A	C	C	B	A	A	A	builders hardware, coiled tubes.
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	B	A	C	C	A	A	A	A	Transportation eq't., heavy duty
-T6	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	B	A	C	C	A	A	A	A	structures, marine uses, pipe, flanges,
																bridge rails, R.R. carsiding.
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	A	A	A	D	A	A	A	B	Pipe, railings, hardware,
-H18	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	A	A	C	C	A	A	A	A	architectural uses
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-	-	-	-	-	-	-	-	Forgings and extrusions
-T6	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	A	A	A-B	B-D	-	A	A	A	for welded structures
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-	-	-	-	-	-	-	-	Welded oxygen and
-T81	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	A	A	C	D	-	A	A	A	missile applications
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-	-	-	-	-	-	-	-	Structural parts for aircraft,
-T6	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	C	C	D	D	D	D	D	D	

ที่มา : [illegible] Metals Reference Issue-Machine Design, The Penton Publishing Co., Cleveland, Ohio; Materials Selector Issue, Reinhold Publishing Co., The Aluminum Data Book, The Reynold Metal Co., Alcoa Structural Handbook, Aluminum Assoc. America.

หมายเหตุ : [illegible] หมายถึง ข้อมูลไม่มีหรือไม่เพียงพอ

ภาคผนวก ก.2

สูตรสำเร็จของคานและหน้าตัด

ตารางที่ ก.2 สูตรสำเร็จของคานและหน้าตัด

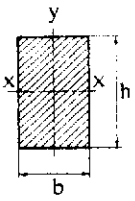
I_x คือโมเมนต์ความเฉื่อยรอบแกน x-x

J คือโมเมนต์ความเฉื่อยเชิงขั้วรอบแกนศูนย์ถ่วง

$Z = I/c$ คือโมดุลัสหน้าตัดของรูปเหลี่ยมรอบแกน x-x

$Z' = J/c$ คือโมดุลัสหน้าตัดเชิงขั้ว

$k = \sqrt{I/A}$ คือรัศมีไจเรชัน

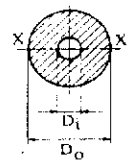


$$I_x = \frac{bh^3}{12}$$

$$Z_x = \frac{bh^2}{6}$$

$$k_x = \frac{h}{\sqrt{12}}$$

$$Z = \frac{2b^2h}{9} \quad \text{(สำหรับการบิด)}$$



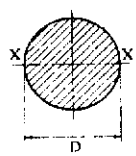
$$I_x = \frac{\pi}{64}(D_o^4 - D_i^4)$$

$$Z_x = \frac{\pi}{32}\left(\frac{D_o^4 - D_i^4}{D_o}\right)$$

$$k_x = \sqrt{\frac{D_o^2 + D_i^2}{16}}$$

$$J = \frac{\pi}{32}(D_o^4 - D_i^4)$$

$$Z'_x = \frac{\pi}{16}\left(\frac{D_o^4 - D_i^4}{D_o}\right)$$



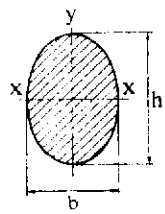
$$I_x = \frac{\pi D^4}{64}$$

$$Z_x = \frac{\pi D^3}{32}$$

$$k_x = \frac{D}{4}$$

$$J = \frac{\pi D^4}{32}$$

$$Z = \frac{\pi D^3}{16}$$



$$I_x = \frac{\pi bh^3}{64}$$

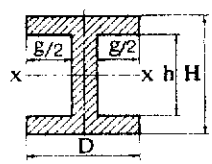
$$Z_x = \frac{\pi bh^2}{32}$$

$$k_x = \frac{h}{4} \; ; \; k_y = \frac{b}{4}$$

$$J = \frac{\pi bh}{64}(h^2 + b^2)$$

$$Z' = \frac{\pi b^2 h}{16} \quad \text{(สำหรับการบิด)}$$

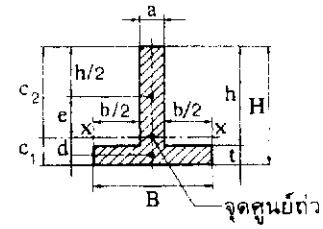
$$A = \pi bh/4$$



$$I_x = \frac{1}{12}(GH^3 - gh^3)$$

$$Z_x = \frac{GH^3 - gh^3}{6H}$$

$$k_x = \sqrt{\frac{1}{12}\left(\frac{GH^3 - gh^3}{GH - gh}\right)}$$



$$c_1 = \frac{aH^2 + bt^2}{2(aH + bt)} \; , \; c_2 = H - c_1$$

$$I_x = \frac{Bt^3}{12} + (Bt)d^2 + \frac{ah^3}{12} + (ah)e^2$$

พื้นที่ $= Bt + a(H - t)$; $k = \sqrt{I/A}$

ภาคผนวก ข
ความรู้ที่เกี่ยวกับ dsPIC30F2011 เบื้องต้น

1. คุณสมบัติเด่นโดยรวมของ dsPIC

1.1 คุณสมบัติของCPU

1. เป็นไมโครคอนโทรลเลอร์ที่ใช้ซีพียูแบบ RISC
2. ความเร็วในการทำงานสูงถึง 30 ล้านคำสั่งต่อวินาที
3. มี 84 คำสั่งภาษาแอสเซมบลีมาตรฐาน รองรับรูปแบบการอ้างแอดเดรสได้อย่างอิสระ
4. ชุดคำสั่งมีขนาด 24 บิต สามารถประมวลผลข้อมูลได้ 16 บิต
5. มีหน่วยความจำโปรแกรมเป็นแบบแฟลช สามารถลบและเขียนใหม่ได้ไม่น้อยกว่า 100,000 ครั้งสามารถป้องกันการอ่านได้ และสามารถโปรแกรมตัวเองโดยใช้กระบวนการทางซอฟต์แวร์
6. มีหน่วยความจำข้อมูลอีอีพรอมที่สามารถลบและเขียนใหม่ได้ไม่น้อยกว่า 1,000,000 ครั้ง
7. มีอินเตอร์รัปต์เวกเตอร์จำนวนมาก จึงรองรับการตอบสนองสัญญาณอินเตอร์รัปต์ได้ดี
8. มีวงจรตรวจจับแรงดันไฟเลี้ยงต่ำกว่ากำหนดแบบโปรแกรมได้
9. มีเพาเวอร์-ออนรีเซต,เพาเวอร์-อัปไทเมอร์ และออสซิลเลเตอร์สตาร์ต-อัปไทเมอร์
10. มีวอตช์ด็อกไทเมอร์แบบโปรแกรมได้
11. มีวงจรตรวจสอบการทำงานของวงจรกำเนิดสัญญาณนาฬิกา
12. รองรับการโปรแกรมในวงจรแบบอนุกรม (ICSP : In-Circuit Serial Programming)
13. สามารถเลือกโหมดการใช้พลังงานได้

1.2 คุณสมบัติด้านการประมวลสัญญาณดิจิตอล

1. มีแอกคิวมูเลเตอร์ขนาด 40 บิต 2 ตัว รองรับการประมวลผลทางคณิตศาสตร์ได้เป็นอย่างดี
2. มีหน่วยประมวลผลด้านการคูณและหารเลข 17 บิตในรูปของฮาร์ดแวร์ จึงทำให้สามารถคูณและหารเลขได้อย่างรวดเร็ว

3. ทำการคูณเลข 16 บิตได้ภายในสัญญาณนาฬิกาเพียง 1 ไซเคิล
4. มีตัวเลื่อนข้อมูลบาร์เรล 40 สเตจ ช่วยให้การประมวลผลข้อมูลที่จำนวนบิตมากๆ สามารถทำได้อย่างรวดเร็ว
5. มีวงจรเฟตช์ข้อมูลคู่ จึงทำให้สามารถประมวลผลข้อมูลได้อย่างรวดเร็ว

1.3 คุณสมบัติของโมดูลฟังก์ชั่นพิเศษ

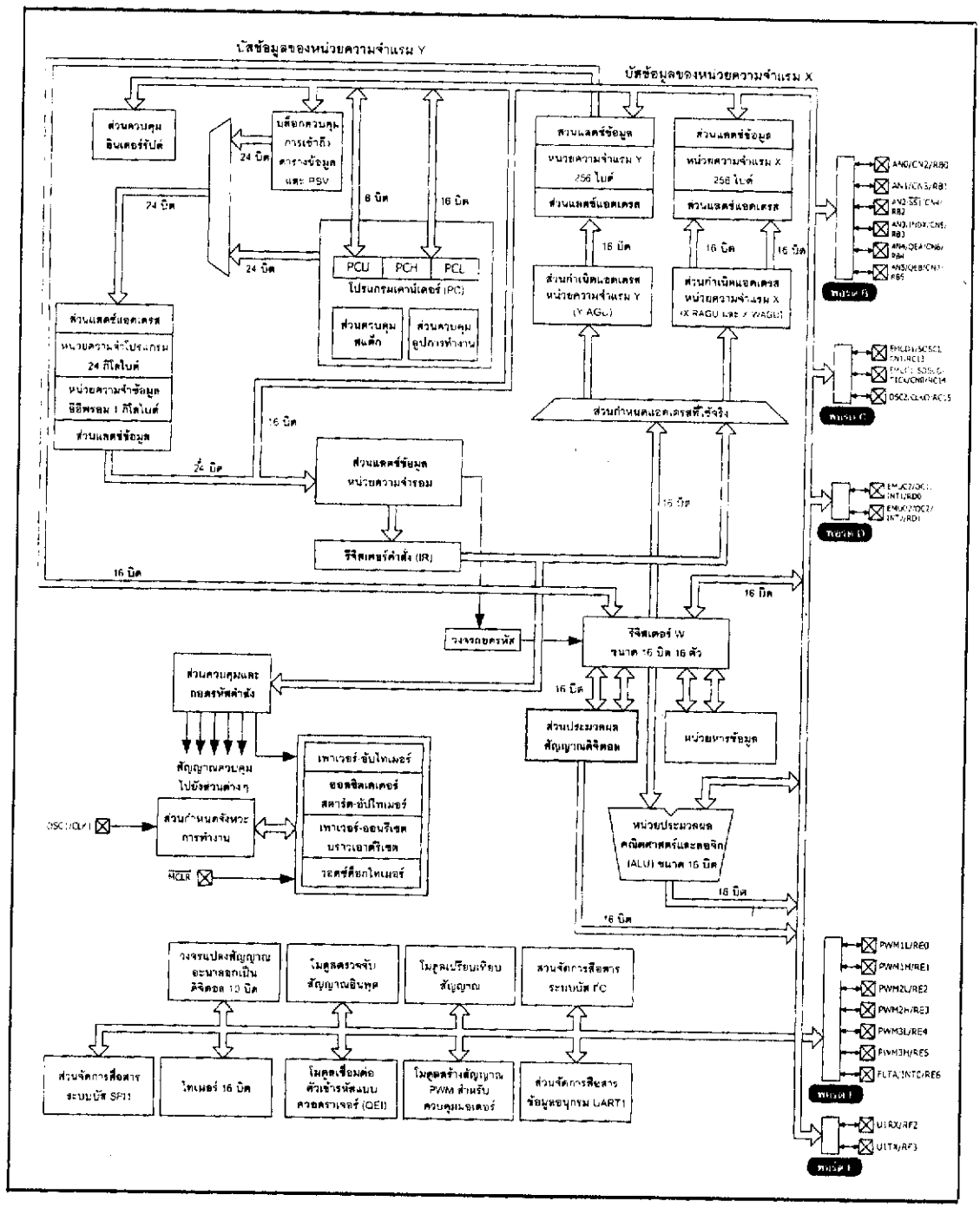
2. สามารถจ่ายกระแสออกทางขาพอร์ตได้ 25mA ทั้งแบบกระแสซิงก์และซอร์ส
3. ไทเมอร์/เคาน์เตอร์มีขนาด 16 บิต ไม่น้อยกว่า 3 ตัว ต่อใช้งานร่วมกันเป็นไทเมอร์ 32 บิตได้
4. มีโมดูลตรวจจับและเปรียบเทียบสัญญาณดิจิตอล
5. มีส่วนเชื่อมต่ออุปกรณ์อนุกรมทั้งแบบ SPI และผ่านระบบบัส I2C
6. มีโมดูลสื่อสารข้อมูลอนุกรม UART พร้อมบัฟเฟอร์แบบ FIFO
7. มีวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอล ความละเอียด 10 หรือ 12 บิต*
8. มีโมดูลสร้างสัญญาณ PWN สำหรับควบคุมมอเตอร์*
9. มีโมดูลเชื่อมต่อตัวเข้ารหัสแบบควอดราเจอร์*

* เป็นคุณสมบัติที่มีในบางเบอร์

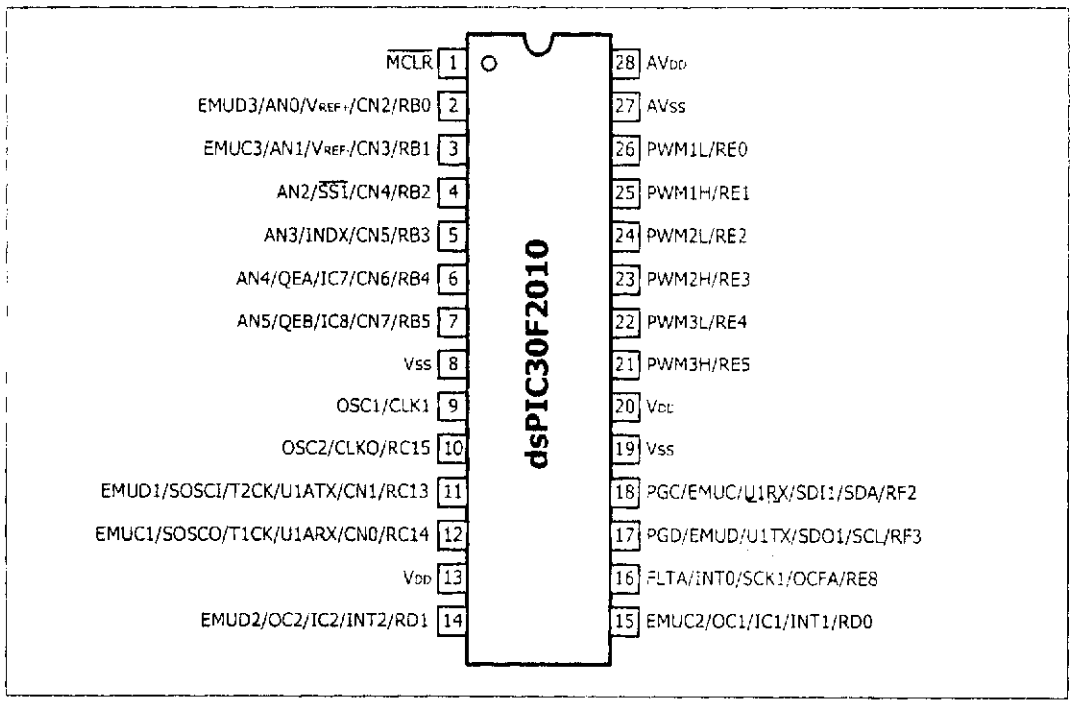
2. แนะนำ dsPIC30F2011

สำหรับไมโครคอนโทรลเลอร์ dsPIC ที่นำมาใช้ในการเรียนรู้และทดลองคือเบอร์ dsPIC30F2011 ซึ่งมีขาต่อใช้งาน 28 ขา ในรูปที่ ข.1 แสดงไดอะแกรมการทำงานและส่วนประกอบทั้งหมดของ dsPIC30F2011 และการจัดขา สำหรับหน้าที่ของแต่ละขาได้อธิบายโดยสรุปไว้ในรูปที่ ข.2

คุณสมบัติทางเทคนิคของ dsPIC30F2011



รูปที่ ข.1 ไดอะแกรมการทำงานและส่วนประกอบทั้งหมดของ dsPIC30F210



รูปที่ ข.2 การจัดขาใช้งานของไมโครคอนโทรลเลอร์ dsPIC30F2010

2.1 คุณสมบัติของ CPU

1. มี 84 คำสั่งมาตรฐาน สามารถรองรับรูปแบบการอ้างแอดเดรสได้อย่างอิสระ
2. ชุดคำสั่งมีขนาด 24 บิต สามารถประมวลผลข้อมูลได้ 16 บิต
3. มีหน่วยความจำโปรแกรมแบบแฟลช ความจุ 12 กิโลไบต์ ลบและเขียนใหม่ได้ไม่น้อยกว่า 100,000 ครั้ง สามารถป้องกันการอ่านได้
4. สามารถโปรแกรมหน่วยความจำโปรแกรมได้ด้วยตัวเอง โดยใช้กระบวนการทางซอฟแวร์
5. มีหน่วยความจำข้อมูลอีอีพรอม 1 กิโลไบต์ ลบและเขียนใหม่ ได้ไม่น้อยกว่า 1,000,000 ครั้ง
6. มีหน่วยความจำข้อมูลแรม 512 ไบต์
7. รีจิสเตอร์ W จัดในรูปของอะเรย์ มีขนาด 16 บิต จำนวน 16 ตัว
8. ความเร็วในการทำงานสูงถึง 30 ล้านคำสั่งต่อวินาที
9. ความถี่สัญญาณนาฬิกาจากภายนอก ตั้งแต่ย่านไฟตรงจนถึง 40MHz
10. ความถี่สัญญาณนาฬิกาในกรณีใช้งานร่วมกับวงจรเฟลล็อกลูปภายใน ตั้งแต่ 4 MHz ถึง 10 MHz เลือกได้ 3 ระดับคือ 4,8 หรือ 16 เท่า
11. รองรับแหล่งกำเนิดสัญญาณอินเตอร์รัปต์ได้สูงสุด 62 แหล่ง รวมทั้งการอินเตอร์รัปต์จากภายนอก 3 แหล่ง

12. สามารถกำหนดระดับความสำคัญในการตอบสนองอินเตอร์รัปต์ได้ 8 ระดับ
13. มีอินเตอร์รัปต์เวกเตอร์ 48 ตำแหน่ง
14. มีวงจรตรวจจับแรงดันไฟเลี้ยงต่ำกว่ากำหนดแบบโปรแกรมได้
15. มีเพาเวอร์-ออนรีเซต, เพาเวอร์-อัปไทเมอร์ และออสซิลเลเตอร์สตาร์ท-อัปไทเมอร์
16. มีวอตช์ด็อกไทเมอร์แบบโปรแกรมได้
17. มีวงจรตรวจสอบการทำงานของวงจรกำเนิดสัญญาณนาฬิกา หากผิดพลาดจะเข้าสู่โหมดสัญญาณนาฬิกา RC พลังงานต่ำทันที
18. รองรับการโปรแกรมในวงจรแบบอนุกรม (ICSP : In-Circuit Serial Programming)
19. สามารถเลือกโหลดการใช้พลังงานได้
20. ย่านไฟเลี้ยง 2.5 ถึง 5.5V กระแสไฟฟ้า 2.6 ถึง 44mA ที่ไฟเลี้ยง +5V ขึ้นอยู่กับการกำหนดความเร็วในการทำงาน

2.2 คุณสมบัติด้านการประมวลสัญญาณดิจิตอล

1. มีแอกคิวมูเลเตอร์ขนาด 40 บิต 2 ตัว รองรับการประมวลผลทางคณิตศาสตร์ได้เป็นอย่างดี
2. มีหน่วยประมวลผลด้านการคูณและหารเลข 17 บิตในรูปของฮาร์ดแวร์ จึงทำให้สามารถทำการคูณและหารเลขได้อย่างรวดเร็ว
3. ทำการคูณเลข 16 บิตได้ภายในสัญญาณนาฬิกาเพียง 1 ไซเคิล
4. มีตัวเลื่อนข้อมูลบาร์เรล 40 สเตจ ช่วยให้ประมวลข้อมูลที่จำนวนบิตมากๆ ทำได้รวดเร็ว
5. มีวงจรเฟตช์ข้อมูลคู่ จึงทำให้สามารถประมวลผลข้อมูลได้อย่างรวดเร็ว

2.3 คุณสมบัติของโมดูลฟังก์ชั่นพิเศษ

1. สามารถจ่ายกระแสออกทางขาพอร์ตได้ 25mA ทั้งแบบกระแสซิงก์และซอร์ส
2. ไทเมอร์/เคาน์เตอร์ 16 บิต 3 ตัว สามารถต่อใช้งานร่วมกันเป็นไทเมอร์/เคาน์เตอร์ 32 บิตได้
3. มีโมดูลตรวจจับสัญญาณดิจิตอลขนาด 16 บิต 4 ชุด
4. มีโมดูลเปรียบเทียบข้อมูลและกำเนิดสัญญา PWM ความละเอียด 16 บิต 2 ชุด
   - ในการเปรียบเทียบข้อมูลสามารถเลือกการทำงานได้ 2 โหมด
5. มีส่วนเชื่อมต่ออุปกรณ์อนุกรมแบบ SPI

6. มีส่วนเชื่อมต่ออุปกรณ์ผ่านระบบบัส I^2C ทั้งแบบ 7 และ 10 บิต กำหนดเป็นมาสเตอร์หรือสเลฟได้
7. มีโมดูลสื่อสารข้อมูลอนุกรม UART พร้อมบัฟเฟอร์แบบ FIFO
8. มีโมดูลสร้างสัญญาณ PWM สำหรับควบคุมมอเตอร์ 6 ช่อง
   - เลือกรูปแบบเอาต์พุตได้ทั้งแบบคอมพลีเมนต์และแบบอิสระ
   - มีโหมดปรับตำแหน่งการหมุนทั้งแบบปรับขอบสัญญาณและแบบกึ่งกลาง
   - มีส่วนกำเนิดดิวตี้ไซเกิล 4 ชุด
   - กำหนดฐานเวลาได้ 4 โหมด
   - สามาถเลือกขั้วของสัญญาณทางเอาต์พุตได้
   - มีสัญญาณกระตุ้นเพื่อให้ทำงานสัมพันธ์กับวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอลภายในไมโครคอนโทรลเลอร์
   - สามารถควบคุมสัญญาณเอาต์พุตได้
9. มีโมดูลเชื่อมต่อตัวเข้ารหัสแบบควอดราเจอร์
   - มีอินพุต Phase A, Phase B และรับสัญญาณพัลส์เพื่อกำหนดตำแหน่ง
   - มีตัวนับตำแหน่งขนาด 16 บิต นับได้ทั้งขึ้นและลง
   - แสดงสถานะของทิศทางการนับได้
   - กำหนดโหมดของการวัดตำแหน่งได้ 2 โหมดคือ x2 และ x4
   - มีวงจรกรองสัญญาณรบกวนแบบดิจิตอลจากอินพุตแบบโปรแกรมได้
   - สำหรับกำหนดให้ทำงานเป็นไทเมอร์/เคาน์เตอร์ขนาด 16 บิตได้
   - กำเนิดสัญญาณอินเตอร์รัปต์จากตำแหน่งที่นับเกิน (rollover) หรือนับขาด (underflow)
10. มีวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอล ความละเอียด 10 บิต 6 ช่อง
    - อัตราการสุ่มและแปลงสัญญาณ 500 กิโลแซมเปิลต่อวินาที
    - สามารถแปลงสัญญาณเมื่อไมโครคอนโทรลเลอร์ทำงานในโหมดสลีปและไอเดิลได้

3. สถาปัตยกรรมโดยสรุปของ dsPIC30F2011

3.1 หน่วยประมวลผลกลาง

หน่วยประมวลผลของ dsPIC30F2011 ใช้คำสั่งที่มีความยาว 1 เวิร์ด ขนาด 24 บิต โดยมีโปรแกรมเคาน์เตอร์ขนาด 23 บิต (จริงๆ แล้วโดยโครงสร้างมี 24 บิตแต่ไม่สนใจบิต MSB ซึ่งก็คือบิต 23 และบิต LSB หรือบิต 0 กำหนดเป็น "0" จึงทำให้สามารถติดต่อหน่วยความจำโปรแกรมได้สูงสุด 4 เมกะเวิร์ด) เพื่อแจ้งแอดเดรสของหน่วยความจำโปรแกรมที่เข้าไปประมวลผล dsPIC30F2011 มีความจุของหน่วยความจำโปรแกรม 12 กิโลไบต์ เมื่อคำสั่งมีความยาว 24 บิต จึงบรรจุคำสั่งได้จริง 4 กิโลเวิร์ด

รีจิสเตอร์หลักที่ใช้ในการทำงานคือ รีจิสเตอร์ W (Working register) สำหรับใน dsPIC จะแตกต่างจากไมโครคอนโทรลเลอร์ PIC อย่างมาก โดยรีจิสเตอร์ W ได้รับการจัดโครงสร้างเป็นอะเรย์ขนาด 16 บิต จึงทำให้สามารถรองรับทั้งข้อมูล, ค่าแอดเดรส หรือค่าของรีจิสเตอร์ใดๆ ที่ต้องนำมาประมวลผลโดยใน dsPIC มีรีจิสเตอร์ W ให้ใช้งานถึง 16 ตัว ส่วนใหญ่ใช้ในการประมวลผลหลัก ส่วนอีกตัวหนึ่งคือรีจิสเตอร์ W15 จะใช้ทำงานร่วมกับตัวชี้สแต็กในการทำงานของโปรแกรมย่อยและบริการอินเตอร์รัปต์

ด้านการตอบสนองอินเตอร์รัปต์นั้น dsPIC30F2011 มีการจัดสรรพื้นที่เก็บค่าอินเตอร์รัปต์เวกเตอร์ไว้มากถึง 54 ตำแหน่ง และยังสามารถกำหนดระดับความสำคัญได้อีก 8 ระดับด้วย

3.2 หน่วยความจำ

dsPIC30F2011 มีหน่วยความจำโปรแกรม 4 กิโลเวิร์ด แอดเดรสอยู่ในช่วง 0x000100 ถึง 0x001FFF สามารถโปรแกรมหรือเขียนข้อมูลลงไปได้ 2 วิธีคือ

1. โดยใช้การโปรแกรมในวงจรแบบอนุกรมหรือ ICSP ผ่านทางขา PGD และ PGC (ขาที่ 17 และ 18) แล้วป้อนสัญญาณพัลส์แรงดันสูงสำหรับโปรแกรมผ่านเข้ามาทางขา MCLR

2. โปรแกรมตัวเองในขณะทำงานหรือ RTSP (Run Time Self-Programming)

ส่วนหน่วยความจำข้อมูลแรมนั้น dsPIC30F2011 ได้จัดสรรเป็น 2 ส่วนคือ หน่วยความจำข้อมูลแรม X และ Y แต่ละส่วนมีขนาด 16 บิต ความจุ 256 ไบต์ รวมเป็น 512 ไบต์ โดยในแต่ละส่วนจะมีตัวกำหนดแอดเดรสแยกออกจากกัน เรียกว่า AGU (Address Generation Unit)

ในขณะที่หน่วยความจำข้อมูลอีอีพรอม dsPIC30F2011จัดสรรไว้ที่แอดเดรส 0x7FFC00 ถึง 0x7FFFFF มีความจุ 1 กิโลไบต์

3.3 ส่วนประมวลผลสัญญาณดิจิตอล (DSP Engine)

นับเป็นส่วนประกอบที่สำคัญอย่างยิ่งของ dsPIC เนื่องจาก dsPIC ได้รับการออกแบบมาให้ทำงานในด้านการประมวลผลสัญญาณดิจิตอลเป็นหลัก ดังนั้นจึงต้องมีการเพิ่มความสามารถในหน่วยคำนวณทางคณิตศาสตร์และลอจิกอย่างมาก โดยในส่วนประมวลผลสัญญาณดิจิตอลมีหน่วยจัดการคุณเลขขนาด 17 x 17 บิตความเร็วสูง, หน่วยประมวลผลทางคณิตศาสตร์และลอจิกหรือ ALU ขนาด 40 บิต, แอกคิวมูเลเตอร์ขนาด 40 บิต อีก 2 ตัว และตัวเลื่อนข้อมูล 2 ทิศทางแบบบาเรล (barrel shifter) ขนาด 40 บิต จึงทำให้สามารถจัดการข้อมูลขนาด 16 บิตได้เสร็จสิ้นภายในสัญญาณนาฬิกาเพียงไซเกิลเดียว

3.4 โมดูลฟังก์ชั่นพิเศษ

dsPIC30F2011 ได้รวมเอาโมดูลสำหรับทำงานเฉพาะทางไว้อย่างมากมาย ไม่ว่าจะเป็นโมดูลแปลงสัญญาณอะนาลอกเป็นดิจิตอล ความละเอียด 10 บิต, โมดูลเชื่อมต่ออุปกรณ์อนุกรมหรือ SPI, โมดูลสื่อสารข้อมูลบนระบบบัส I^2C, โมดูลสื่อสารข้อมูลผ่านพอร์ตอนุกรมหรือ UART, ไทเมอร์ขนาด 16 บิตถึง 3 ตัว และที่เป็นพิเศษอีก 2 โมดูลคือ โมดูลสร้างสัญญาณ PWM เพื่อการควบคุมมอเตอร์และโมดูลเข้ารหัสแบบควอดราเจอร์ โดยสามารถใช้งานร่วมกันเพื่อสร้างระบบควบคุมมอเตอร์แบบปิดประสิทธิภาพสูง

3.5 พอร์ตอินพุตเอาต์พุต

dsPIC30F2011 มีพอร์ตให้ใช้งานมากถึง 5 พอร์ต รวม 20 ขา ดังนี้

พอร์ต B มี 6 ขาคือ RB0-RB5 โดยทุกขาสามารถกำหนดให้เป็นพอร์ตอินพุตหรือเอาต์พุตได้ และยังสามารถขับกระแสทั้งแบบซิงก์และซอร์สได้สูงถึง 25mA

พอร์ต C มี 3 ขาคือ RC13—RC15

พอร์ต D มี 2 ขาคือ RD0 และ RD1

พอร์ต E มี 7 ขาคือ RE0-RE5 และ RE8

พอร์ต F มี 2 ขาคือ RF2 และ RF3

4. คุณสมบัติโดยสรุปของอินเตอร์รัปต์ใน dsPIC

ในไมโครคอนโทรลเลอร์ dsPIC ได้รับการเพิ่มเติมความสามารถด้านการจัดการและตอบสนองอินเตอร์รัปต์อย่างมาก ไม่ว่าจะเป็นแหล่งกำเนิดสัญญาณอินเตอร์รัปต์และอินเตอร์รัปเวกเตอร์ทำให้ข้อจำกัดด้านอินเตอร์รัปต์ที่เคยมีในไมโครคอนโทรลเลอร์ PIC หมดไป อย่างไรก็ตาม แม้ว่าจะมีแหล่งกำเนิดสัญญาณอินเตอร์รัปต์มากมาย แต่ด้วยการจัดการของโมดูลควบคุม

อินเตอร์รัปต์ (Interrupt controller module) สามารถทำให้การจัดการเกี่ยวกับการเลือกใช้และการตอบสนองอินเตอร์รัปต์ สามารถทำให้อย่างเป็นระบบและไม่ซับซ้อนเกินไป

คุณสมบัติเด่นโดยสรุปของโมดูลควบคุมอินเตอร์รัปต์ใน dsPIC30F2011 มีดังนี้

4.1 กำหนดระดับความสำคัญในการตอบสนองอินเตอร์รัปต์ได 8 ระดับ

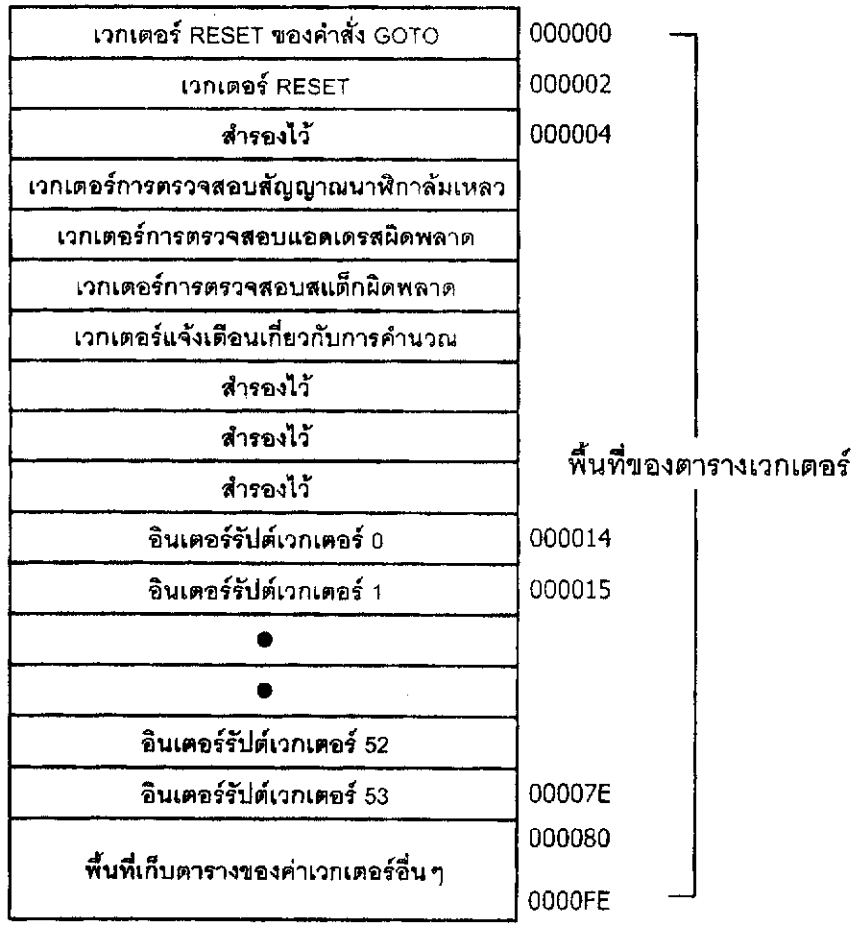
4.2 มีแหล่งกำเนิดสัญญาณอินเตอร์รัปต์ 44 แหล่ง (สูงสุด 54 แหล่งใน dsPIC เบอร์ใหญ่สุด)

4.3 รองรับการขัดจังหวะจากโปรเซสเซอร์หรือโปรเซสเซอร์แทร็ป (processor trap) หรืออาจเรียกง่ายๆ ว่า การอินเตอร์รัปต์จากCPU ได้อีก 4 แบบ (สูงสุด 8 แบบใน dsPIC เบอร์ใหญ่สุด)

ตารางที่ ข.1 แสดงหมายเลขอินเตอร์รัปต์และชื่อของแหล่งกำเนิดอินเตอร์รัปต์ในไมโครคอนโทรลเลอร์ dsPIC

หมายเลข	ชนิดแหล่งกำเนิด	ชื่อหลัก	ชื่อรอง
-	สงวนไว้	_ReserveTrap0	_AltReserveTrap0
-	อินเตอร์รัปต์จากการทำงานผิดพลาดของออสซิลเลเตอร์	_OscillatorFail	_AltOscillatorFail
-	อินเตอร์รัปต์จากแอดเดรสผิดพลาด	_AddressError	_AltAddressError
-	อินเตอร์รัปต์จากสแต็กผิดพลาด	_StackError	_AltStackError
-	อินเตอร์รัปต์จากการทำงานผิดพลาดของหน่วยประมวลผลคณิตศาสตร์และลอจิก	_MathError	_AltMathError
-	สงวนไว้	_ReserveTrap5	_AltReserveTrap5
-	สงวนไว้	_ReserveTrap6	_AltReserveTrap6
-	สงวนไว้	_ReserveTrap7	_AltReserveTrap7
1	อินเตอร์รัปต์จากสัญญาณภายนอกที่ขา INT0	_INT0Interrupt	_AltINT0Interrupt
2	อินเตอร์รัปต์จากอินพุตแคปเจอร์ช่อง 1	_IC1Interrupt	_AltIC1Interrupt
3	อินเตอร์รัปต์จากโมดูลเปรียบเทียบข้อมูล ช่อง 1	_OC1Interrupt	_AltOC1Interrupt
4	อินเตอร์รัปต์จากไทเมอร์ 1	_T1Interrupt	_AltT1Interrupt
5	อินเตอร์รัปต์จากอินพุตแคปเจอร์ช่อง 2	_IC2Interrupt	_AltIC2Interrupt
6	อินเตอร์รัปต์จากโมดูลเปรียบเทียบข้อมูล ช่อง 2	_OC1Interrupt	_AltOC1Interrupt
7	อินเตอร์รัปต์จากไทเมอร์ 2	_T2Interrupt	_AltT2Interrupt
8	อินเตอร์รัปต์จากไทเมอร์ 3	_T3Interrupt	_AltT3Interrupt
9	อินเตอร์รัปต์จากโมดูล SPI1	_SPI1Interrupt	_AltSPI1Interrupt
10	อินเตอร์รัปต์จากการรับข้อมูลของโมดูล UART1	_U1RXInterrupt	_AltU1RXInterrupt
11	อินเตอร์รัปต์จากการส่งข้อมูลของโมดูล UART1	_U1TXInterrupt	_AltU1TXInterrupt
12	อินเตอร์รัปต์จากโมดูล ADC	_ADCInterrupt	_AltADCInterrupt
13	อินเตอร์รัปต์จากแฟลกแจ้งการเขียนข้อมูลในหน่วยความจำโปรแกรมเสร็จสิ้น	_NVMInterrupt	_AltNVMInterrupt
14	อินเตอร์รัปต์จากโมดูล I2C ในโหมดสเลฟ	_SI2CInterrupt	_AltSI2CInterrupt
15	อินเตอร์รัปต์จากโมดูล I2C ในโหมดมาสเตอร์	_MI2CInterrupt	_AltMI2CInterrupt
16	อินเตอร์รัปต์จากการเปลี่ยนแปลงลอจิกที่ขา CN ใดๆ	_CNInterrupt	_AltCNInterrupt
17	อินเตอร์รัปต์จากสัญญาณภายนอกที่ขา INT1	_INT1Interrupt	_AltINT1Interrupt
18*	อินเตอร์รัปต์จากอินพุตแคปเจอร์ช่อง 7	_IC7Interrupt	_AltIC7Interrupt
19*	อินเตอร์รัปต์จากอินพุตแคปเจอร์ช่อง 8	_IC8Interrupt	_AltIC8Interrupt
20*	อินเตอร์รัปต์จากโมดูลเปรียบเทียบข้อมูล ช่อง 3	_OC3Interrupt	_AltOC3Interrupt
21*	อินเตอร์รัปต์จากโมดูลเปรียบเทียบข้อมูล ช่อง 4	_OC4Interrupt	_AltOC4Interrupt

หมายเลข	ชนิดแหล่งกำเนิด	ชื่อหลัก	ชื่อรอง
22*	อินเตอร์รัปต์จากไทเมอร์ 4	_T4Interrupt	_AltT4Interrupt
23*	อินเตอร์รัปต์จากไทเมอร์ 5	_T5Interrupt	_AltT5Interrupt
24*	อินเตอร์รัปต์จากสัญญาณภายนอกที่ขา INT2	_INT2Interrupt	_AltINT2Interrupt
25*	อินเตอร์รัปต์จากการรับข้อมูลของโมดูล UART2	_U2RXInterrupt	_AltU2RXInterrupt
26*	อินเตอร์รัปต์จากการส่งข้อมูลของโมดูล UART2	_U2TXInterrupt	_AltU2TXInterrupt
27*	อินเตอร์รัปต์จากโมดูล SPI2	_SPI2Interrupt	_altSPI2Interrupt
28*	อินเตอร์รัปต์จากโมดูล CAN ช่อง 1	_C1Interrupt	_AltC1Interrupt
29*	อินเตอร์รัปต์จากอินพุตแคปเจอร์ช่อง 3	_IC3Interrupt	_AltIC3Interrupt
30*	อินเตอร์รัปต์จากอินพุตแคปเจอร์ช่อง 4	_IC4Interrupt	_AltIC4Interrupt
31*	อินเตอร์รัปต์จากอินพุตแคปเจอร์ช่อง 5	_IC5Interrupt	_AltIC5Interrupt
32*	อินเตอร์รัปต์จากอินพุตแคปเจอร์ช่อง 6	_IC6Interrupt	_AltIC6Interrupt
33*	อินเตอร์รัปต์จากโมดูลเปรียบเทียบข้อมูล ช่อง 5	_OC5Interrupt	_AltOC5Interrupt
34*	อินเตอร์รัปต์จากโมดูลเปรียบเทียบข้อมูล ช่อง 6	_OC6Interrupt	_AltOC6Interrupt
35*	อินเตอร์รัปต์จากโมดูลเปรียบเทียบข้อมูล ช่อง 7	_OC7Interrupt	_AltOC7Interrupt
36*	อินเตอร์รัปต์จากโมดูลเปรียบเทียบข้อมูล ช่อง 8	_OC8Interrupt	_AltOC8Interrupt
37*	อินเตอร์รัปต์จากสัญญาณภายนอกที่ขา INT3	_INT3Interrupt	_AltINT3Interrupt
38*	อินเตอร์รัปต์จากสัญญาณภายนอกที่ขา INT4	_INT4Interrupt	_AltINT4Interrupt
39*	อินเตอร์รัปต์จากโมดูล CAN ช่อง 2	_C2Interrupt	_AltC2Interrupt
40	อินเตอร์รัปต์จากโมดูล PWM	_PWMInterrupt	_AltPWMInterrupt
41	อินเตอร์รัปต์จากโมดูล QEI	_QEIInterrupt	_AltQEIInterrupt
42*	อินเตอร์รัปต์จากโมดูล DCI	_DCIInterrupt	_AltDCIInterrupt
43	อินเตอร์รัปต์จากวงจรตรวจจับแรงดันต่ำ (LVD)	_LVDInterrupt	_AltLVDInterrupt
44	อินเตอร์รัปต์จากขา FLTA แอกตีฟ	_FLTAInterrupt	_AltFLTAInterrupt
45	อินเตอร์รัปต์จากขา FLTB แอกตีฟ	_FLTBInterrupt	_AltFLTBInterrupt
46	สงวนไว้	_Interrupt46	_AltInterrupt46
47	สงวนไว้	_Interrupt47	_AltInterrupt47
48	สงวนไว้	_Interrupt48	_AltInterrupt48
49	สงวนไว้	_Interrupt49	_AltInterrupt49
50	สงวนไว้	_Interrupt50	_AltInterrupt50
51	สงวนไว้	_Interrupt51	_AltInterrupt51
52	สงวนไว้	_Interrupt52	_AltInterrupt52
53	สงวนไว้	_Interrupt53	_AltInterrupt53



รูปที่ ข.3 แสดงรายละเอียดของอินเตอร์รัปต์เวกเตอร์หลักของไมโครคอนโทรลเลอร์ dsPIC

- มีอินเตอร์รัปต์เวกเตอร์ 44 ตำแหน่ง (สูงสุด 54 ตำแหน่งใน dsPIC เบอร์ใหญ่สุด) รวมกับอินเตอร์รัปต์เวกเตอร์อันเนื่องมาจากการทำงานของCPUหรือCPUแทร็ปเวกเตอร์อีก 4 ตำแหน่ง (สูงสุด 8 ตำแหน่งใน dsPIC เบอร์ใหญ่สุด) เป็น 48 ตำแหน่ง (สูงสุด 62 ตำแหน่งใน dsPIC เบอร์ใหญ่สุด) โดยได้รับการจัดสรรในรูปแบบของตาราง เรียกว่า ตารางอินเตอร์รัปต์เวกเตอร์ (Interrupt Vector Table : IVT)

- มีตารางอนเตอร์รัปต์เวกเตอร์เสริม (Alternate Interrupt Vector Table : AIVT) เพื่อรองรับการดีบัก ในตารางที่ 4-1 แสดงข้อมูลของอินเตอร์รัปต์เวกเตอร์ทั้งหมดที่ dsPIC รองรับ แต่จะมีบางตัวที่ dsPIC30F2011 ไม่รองรับการทำงาน ส่วนในรูปที่ ข.3 แสดงรายละเอียดของตารางอินเอตร์รัปต์เวกเตอร์ที่มีใน dsPIC30F2011

5. กระบวนการอินเตอร์รัปต์ในไมโครคอนโทรลเลอร์ dsPIC

5.1 การเตรียมการ

หากมีความต้องการใช้งานอินเตอร์รัปต์ในไมโครคอนโทรลเลอร์ dsPIC จะต้องเตรียมการกำหนดค่าในรีจิสเตอร์ที่เกี่ยวข้อง สามารถสรุปได้ดังนี้

1. เซตบิต NSTDIS ซึ่งเป็นบิต 15 ของรีจิสเตอร์ INTCON1 ถ้าหากไม่ต้องการให้เกิดการอินเตอร์รัปต์ซ้อน
2. เลือกระดับความสำคัญของแหล่งกำเนิดอินเตอร์รัปต์ โดยกำหนดค่าลงในรีจิสเตอร์ IPCx ตัวที่เกี่ยวข้องกับแหล่งกำเนิดอินเตอร์รัปต์นั้นๆ
3. เคลียร์บิตแฟลกที่แจ้งสถานะการเกิดอินเตอร์รัปต์ในรีจิสเตอร์ IFSx ตัวที่เกี่ยวข้องกับแหล่งกำเนิดอินเตอร์รัปต์นั้นๆ
4. เอ็นเอเบิล แหล่งกำเนิดอินเตอร์รัปที่ต้องการในรีจิสเตอร์ IECx ตัวที่เกี่ยวข้องกับแหล่งกำเนิดอินเตอร์รัปต์นั้นๆ

5.2 โปรแกรมย่อยบริการอินเตอร์รัปต์ หรือ ISR (Interrupt Service Routine)

เมื่อเกิดอินเตอร์รัปต์ขึ้น CPUจะกระโดดไปยังแอดเดรสอินเตอร์รัปต์เวกเตอร์ของแหล่งกำเนิดอินเตอร์รัปต์นั้นๆ และที่แอดเดรสอินเตอร์รัปต์เวกเตอร์ก็จะบรรจุคำสั่งเพื่อกำหนดให้ CPU ดำเนินการโดยปกติจะเป็นคำสั่งสั้นๆ เพื่อกำหนดให้กระโดดต่อไปยังโปรแกรมย่อยบริการอินเตอร์รัปต์ (ISR)

ที่บรรทัดแรกของ ISR จะต้องบรรจุคำสั่งเคลียร์บิตแฟลกของอินเตอร์รัปต์ที่กำลังตอบสนองขณะนั้นในรีจิสเตอร์ IFSx จากนั้นจึงกำหนดให้ CPU ทำงานตามที่ต้องการ CPU จะสามารถออกจาก ISR ได้ก็ต่อเมื่อพบคำสั่งภาษาแอสเซมบลี RETFIE จากนั้นก็จะกลับไปทำงานที่โปรแกรมหลักตามปกติ

5.3 โปรแกรมย่อยบริการอินเตอร์รัปต์จาก CPU หรือ TSR (Trap Service Routine)

การเขียนโปรแกรมย่อยบริการอินเตอร์รัปต์สำหรับ TSR คล้ายกับ ISR หากแต่การเคลียร์แฟลกจะต้องไปกระทำยังบิตที่เกี่ยวข้องในรีจิสเตอร์ INTCON1 เสียก่อน จากนั้นจึงเข้าสู่การทำงานในโปรแกรมย่อยต่อไป และการออกจากโปรแกรมย่อยจะใช้คำสั่ง RETFIE เช่นกัน

5.4 การดิสเอเบิ้ลอินเตอร์รัปต์

มีขั้นตอนโดยสรุป ดังนี้

1. เก็บค่าสถานะในปัจจุบันของรีจิสเตอร์ SR ลงในสแต็ก โดยใช้คำสั่ง PUSH

2. กำหนดระดับความสำคัญไว้ที่ระดับ 7 โดยการออร์ค่าใน 8 บิตล่างของรีจิสเตรอร์ SR ซึ่งก็คือ SRL ด้วยค่า 0xE0

เพียงเท่านี้ก็จะสามารถดิสเอเบิลการอินเตอร์รัปต์ได้แล้ว และหากต้องการให้สามารถใช้งานอินเตอร์รัปต์ ก็สามารถทำได้โดยการคืนค่ารีจิสเตอร์ SR ด้วยการใช้คำสั่งภาษาแอสเซมบลี POP จากนั้นก็จะกลับเข้าสู่กระบวนการเขียนโปรแกรมตามปกติ

การอธิบายกระบวนการอินเตอร์รัปต์ที่ผ่านมานั้นจะอ้างอิงถึงคำสั่งภาษาแอสเซมบลีให้เห็นภาพในเชิงลึก แต่ถ้าพัฒนาโปรแกรมด้วยภาษา C ขั้นตอนในรายละเอียดเหล่านี้จะถูกจัดการโดยกระบวนการคอมไพล์ ทำให้ช่วยลดขั้นตอนและความซับซ้อนของโปรแกรมอย่างมาก

ภาคผนวก ค.1

Code คำสั่งเพื่อควบคุมการทำงานของโปรแกรม

```
Private Sub comc_Click()  //คำสั่งควบคุมปุ่ม Compile
   txpro.Text = ""
End Sub
Private Sub comp_Click()
Dim num As Integer, num2 As Integer, resp As Boolean
Dim i As Integer, j As Integer, k As Integer, l As Integer, a As Boolean, n As Boolean, m As
Boolean
Dim c1(1 To 100) As String, c2(1 To 1000) As Integer, c3(1 To 1000) As Single, tex As String,
str As String
   tex = txpro.Text
   num = Len(txpro.Text)
   num2 = (num + 2) / 7
   j = 1
   k = 2
   l = 3
   resp = True
   comr.Enabled = True
   If txpro.Text = "" Then
      MsgBox "???????????????", vbokoly + vbExclamation, "???????"
      comr.Enabled = False
   End If
   If 5 > Len(txpro.Text) Then
      MsgBox "???????????????", vbokoly + vbExclamation, "???????"
      comr.Enabled = False
   End If
   For i = 1 To num2
      c1(i) = Mid(tex, j, 1)
      c2(i) = Val(Mid(tex, k, 1))
```

```
    c3(i) = Val(Mid(tex, l, 3))
    If c1(i) <> "m" Then
      a = True
    Else
      a = False
    End If
    If c2(i) > 7 Then
      n = True
    Else
      n = False
    End If
    If c3(i) > 180 Or c3(i) < 0 Then
      m = True
    Else
      m = False
    End If
    If a Or n Or m Then
      resp = False
      Exit For
    End If
    j = j + 7
    k = k + 7
    l = l + 7
Next i
If resp = False Then
   MsgBox "???????????????", vbokoly + vbExclamation, "???????"
   comr.Enabled = False
End If
Debug.Print "//////////////////////////////////////////////////////////////////"
Debug.Print "Len(num)" & Len(txpro.Text)
Debug.Print "Mid(str1,2,1)" & Mid(tex, 1, 1)
Debug.Print "Mid(str1,2,1)" & Mid(tex, 8, 1)
```

```
    Debug.Print "Mid(str1,2,1)" & Mid(tex, 15, 1)
    Debug.Print "c1(1)=" & c1(1)
    Debug.Print "c1(2)=" & c1(2)
    Debug.Print "c1(3)=" & c1(3)
    Debug.Print "c1(4)=" & c1(4)
    Debug.Print "c2(1)=" & c2(1)
    Debug.Print "c2(2)=" & c2(2)
    Debug.Print "c2(3)=" & c2(3)
    Debug.Print "c2(4)=" & c2(4)
    Debug.Print "c3(1)=" & c3(1)
    Debug.Print "c3(2)=" & c3(2)
    Debug.Print "c3(3)=" & c3(3)
    Debug.Print "c3(4)=" & c3(4)
    Debug.Print "num2 =" & num2
    Debug.Print "tex =" & tex
    Debug.Print "j=" & j
End Sub
Private Sub comr_Click()
If comr.Caption = "Run" Then
   comr.Caption = "Stop"
   'time1.Enabled = True
   run
Else
   comr.Caption = "Run"
   time1.Enabled = False
End If
End Sub
Private Sub Form_Load()
comr.Enabled = False
msc.CommPort = 2
msc.Settings = "9600,n,8,1"
msc.PortOpen = True
```

```
'time1.Enabled = False
'time1.Interval = 100
time2.Enabled = False
time2.Interval = 100
End Sub

Sub run() //คำสั่งควบคุมปุ่ม Run
Dim num22 As Integer, num222 As Integer, ti As Integer, tj As Integer, m(1 To 1000) As Long,
q(1 To 1000) As Long
Dim cm  As Integer, cq As Integer, onl As String, tex2 As String
On Error Resume Next
  msc.DTREnable = False
  msc.DTREnable = True
  msc.InputLen = 0
  onl = txnum.Text
  cou = Val(onl)
  For ti = 1 To cou
    tex2 = txpro.Text
    num22 = Len(txpro.Text)
    num222 = (num22 + 2) / 7
    cm = 2
    cq = 3
    For tj = 1 To num222
    m(tj) = Val(Mid(tex2, cm, 1))
    q(tj) = Val(Mid(tex2, cq, 3))
    Select Case m(tj)
    Case 1
      tx = 1
    Case 2
      tx = 2
    Case 3
      tx = 3
```

```
Case 4
   tx = 4
Case 5
   tx = 5
Case 6
   tx = 6
Case 7
   tx = 7
End Select
msc.Output = Chr(m(tj))
bata = msc.Input
Do Until (Asc(bata) = m(tj))
   bata = msc.Input
Loop
qtx = q(tj)
If q(tj) > 127 Then
   msc.Output = "e"
   bata = msc.Input
   Debug.Print "bata=" & bata
   Do Until (bata = "e")
      bata = msc.Input
   Loop
   Debug.Print "Chr(Right(q(tj), 2))=" & Right(q(tj), 2)
   msc.Output = Chr(Val(Right(q(tj), 2)))
   Debug.Print "Chr(Right(q(tj), 2))=" & Chr(Val(Right(q(tj), 2)))
   bata = msc.Input
   Do Until (Asc(bata) = Right(q(tj), 2))
      bata = msc.Input
   Loop
Else
   msc.Output = Chr(q(tj))
   bata = msc.Input
```

```
    Do Until (Asc(bata) = q(tj))
      bata = msc.Input
    Loop
  End If
  Select Case tx
  Case 1
    txm1.Text = qtx
  Case 2
    txm2.Text = qtx
  Case 3
    txm3.Text = qtx
  Case 4
    txm4.Text = qtx
  Case 5
    txm5.Text = qtx
  Case 6
    txm6.Text = qtx
  Case 7
    txm7.Text = qtx
  End Select
  cm = cm + 7
  cq = cq + 7
  Debug.Print "cou =" & cou
  Debug.Print "num222 =" & num222
  Debug.Print "m(1)=" & m(1)
  Debug.Print "m(2)=" & m(2)
  Debug.Print "m(3)=" & m(3)
  Debug.Print "m(4)=" & m(4)
  Debug.Print "q(1)=" & q(1)
  Debug.Print "q(2)=" & q(2)
  Debug.Print "q(3)=" & q(3)
  Debug.Print "q(4)=" & q(4)
```

```
        Debug.Print "tex2=" & tex2
        Debug.Print "m1=" & Chr(m(1))
        Debug.Print "m2=" & Chr(m(2))
        Debug.Print "m3=" & Chr(m(3))
        Debug.Print "q1=" & Chr(q(1))
        Debug.Print "q2=" & Chr(q(2))
        Debug.Print "q3=" & Chr(q(3))
        Debug.Print "in=" & bata
       Next tj
    Next ti
    comr.Caption = "Run"
    comr.Enabled = False
End Sub
Private Sub Origin_Click()          //คำสั่งควบคุมปุ่ม Set Origin
On Error Resume Next
    msc.DTREnable = False
    msc.DTREnable = True
    msc.InputLen = 0
    msc.Output = Chr(1)
    pata = msc.Input
    Do Until (Asc(pata) = 1)
      pata = msc.Input
    Loop
    msc.Output = Chr(90)
    pata = msc.Input
    Do Until (Asc(pata) = 90)
      pata = msc.Input
    Loop
    txm1.Text = 90
    msc.Output = Chr(2)
    pata = msc.Input
    Do Until (Asc(pata) = 2)
```

```
    pata = msc.Input
  Loop
  msc.Output = Chr(90)
  pata = msc.Input
  Do Until (Asc(pata) = 90)
    pata = msc.Input
  Loop
  txm2.Text = 90
  msc.Output = Chr(3)
  pata = msc.Input
  Do Until (Asc(pata) = 3)
    pata = msc.Input
  Loop
  msc.Output = Chr(90)
  pata = msc.Input
  Do Until (Asc(pata) = 90)
    pata = msc.Input
  Loop
  txm3.Text = 90
  msc.Output = Chr(4)
  pata = msc.Input
  Do Until (Asc(pata) = 4)
    pata = msc.Input
  Loop
  msc.Output = Chr(90)
  pata = msc.Input
  Do Until (Asc(pata) = 90)
    pata = msc.Input
  Loop
  txm4.Text = 90
  msc.Output = Chr(5)
  pata = msc.Input
```

```
Do Until (Asc(pata) = 5)
   pata = msc.Input
Loop
msc.Output = Chr(90)
pata = msc.Input
Do Until (Asc(pata) = 90)
   pata = msc.Input
Loop
txm5.Text = 90
msc.Output = Chr(6)
pata = msc.Input
Do Until (Asc(pata) = 6)
   pata = msc.Input
Loop
msc.Output = Chr(90)
pata = msc.Input
Do Until (Asc(pata) = 90)
   pata = msc.Input
Loop
txm6.Text = 90
msc.Output = Chr(7)
pata = msc.Input
Do Until (Asc(pata) = 7)
   pata = msc.Input
Loop
msc.Output = Chr(90)
pata = msc.Input
Do Until (Asc(pata) = 90)
   pata = msc.Input
Loop
txm7.Text = 90
```

```
End Sub
Private Sub time2_Timer()                    //คำสั่ง Time2
Dim recdata As String, i As Integer
On Error Resume Next
  msc.DTREnable = False
  msc.DTREnable = True
  msc.InputLen = 0
  If (msc.Input = True) Then
    tata = msc.Input
    recdata = Asc(tata)
    i = i + 1
  End If
  Debug.Print "i=" & i
  If (i = 1) Then
    Select Case recdata
    Case 1
      tx = 1
    Case 2
      tx = 2
    Case 3
      tx = 3
    Case 4
      tx = 4
    Case 5
      tx = 5
    Case 6
      tx = 6
    Case 7
      tx = 7
    End Select
 End If
```

```
    Debug.Print "i=" & i
    If (i = 2) Then
       qtx = recdata
       tx = qtx
       i = 0
    End If
    Debug.Print "i=" & i
    Debug.Print "tx=" & tx
End Sub
```

ภาคผนวก ค.2

Code คำสั่งปุ่ม Compile

```
Private Sub comp_Click()
Dim num As Integer, num2 As Integer, resp As Boolean
Dim i As Integer, j As Integer, k As Integer, l As Integer, a As Boolean, n As Boolean, m As Boolean
Dim c1(1 To 100) As String, c2(1 To 1000) As Integer, c3(1 To 1000) As Single, tex As String, str As String
  tex = txpro.Text
  num = Len(txpro.Text)
  num2 = (num + 2) / 7
  j = 1
  k = 2
  l = 3
  resp = True
  comr.Enabled = True
  If txpro.Text = "" Then
    MsgBox "??????????????", vbokoly + vbExclamation, "???????"
    comr.Enabled = False
  End If
  If 5 > Len(txpro.Text) Then
    MsgBox "??????????????", vbokoly + vbExclamation, "???????"
    comr.Enabled = False
  End If
  For i = 1 To num2
    c1(i) = Mid(tex, j, 1)
    c2(i) = Val(Mid(tex, k, 1))
    c3(i) = Val(Mid(tex, l, 3))
    If c1(i) <> "m" Then
      a = True
```

```
    Else
      a = False
    End If
    If c2(i) > 7 Then
      n = True
    Else
      n = False
    End If
    If c3(i) > 180 Or c3(i) < 0 Then
      m = True
    Else
      m = False
    End If
    If a Or n Or m Then
      resp = False
      Exit For
    End If
    j = j + 7
    k = k + 7
    l = l + 7
Next i
If resp = False Then
  MsgBox "???????????????", vbokoly + vbExclamation, "???????"
  comr.Enabled = False
End If
Debug.Print "//////////////////////////////////////////////////////////////////"
Debug.Print "Len(num)" & Len(txpro.Text)
Debug.Print "Mid(str1,2,1)" & Mid(tex, 1, 1)
Debug.Print "Mid(str1,2,1)" & Mid(tex, 8, 1)
Debug.Print "Mid(str1,2,1)" & Mid(tex, 15, 1)
Debug.Print "c1(1)=" & c1(1)
Debug.Print "c1(2)=" & c1(2)
```

```
    Debug.Print "c1(3)=" & c1(3)
    Debug.Print "c1(4)=" & c1(4)
    Debug.Print "c2(1)=" & c2(1)
    Debug.Print "c2(2)=" & c2(2)
    Debug.Print "c2(3)=" & c2(3)
    Debug.Print "c2(4)=" & c2(4)
    Debug.Print "c3(1)=" & c3(1)
    Debug.Print "c3(2)=" & c3(2)
    Debug.Print "c3(3)=" & c3(3)
    Debug.Print "c3(4)=" & c3(4)
    Debug.Print "num2 =" & num2
    Debug.Print "tex =" & tex
    Debug.Print "j=" & j
End Sub
```

ภาคผนวก ค.3

Code คำสั่งปุ่ม Run

```
Private Sub comr_Click()
If comr.Caption = "Run" Then
   comr.Caption = "Stop"
   'time1.Enabled = True
   run
Else
   comr.Caption = "Run"
   time1.Enabled = False
End If
End Sub
-------------------------------------------------------------
Sub run()
Dim num22 As Integer, num222 As Integer, ti As Integer, tj As Integer, m(1 To 1000) As Long,
q(1 To 1000) As Long
Dim cm  As Integer, cq As Integer, onl As String, tex2 As String
On Error Resume Next
  msc.DTREnable = False
  msc.DTREnable = True
  msc.InputLen = 0
  onl = txnum.Text
  cou = Val(onl)
  For ti = 1 To cou
    tex2 = txpro.Text
    num22 = Len(txpro.Text)
    num222 = (num22 + 2) / 7
    cm = 2
    cq = 3
```

```
For tj = 1 To num222
m(tj) = Val(Mid(tex2, cm, 1))
q(tj) = Val(Mid(tex2, cq, 3))
Select Case m(tj)
Case 1
   tx = 1
Case 2
   tx = 2
Case 3
   tx = 3
Case 4
   tx = 4
Case 5
   tx = 5
Case 6
   tx = 6
Case 7
   tx = 7
End Select
msc.Output = Chr(m(tj))
bata = msc.Input
Do Until (Asc(bata) = m(tj))
   bata = msc.Input
Loop
qtx = q(tj)
If q(tj) > 127 Then
   msc.Output = "e"
   bata = msc.Input
   Debug.Print "bata=" & bata
   Do Until (bata = "e")
      bata = msc.Input
   Loop
```

```
    Debug.Print "Chr(Right(q(tj), 2))=" & Right(q(tj), 2)
    msc.Output = Chr(Val(Right(q(tj), 2)))
    Debug.Print "Chr(Right(q(tj), 2))=" & Chr(Val(Right(q(tj), 2)))
    bata = msc.Input
    Do Until (Asc(bata) = Right(q(tj), 2))
      bata = msc.Input
    Loop
  Else
    msc.Output = Chr(q(tj))
    bata = msc.Input
    Do Until (Asc(bata) = q(tj))
      bata = msc.Input
    Loop
  End If
  Select Case tx
  Case 1
    txm1.Text = qtx
  Case 2
    txm2.Text = qtx
  Case 3
    txm3.Text = qtx
  Case 4
    txm4.Text = qtx
  Case 5
    txm5.Text = qtx
  Case 6
    txm6.Text = qtx
  Case 7
    txm7.Text = qtx
  End Select
  cm = cm + 7
  cq = cq + 7
```

```
        Debug.Print "cou =" & cou
        Debug.Print "num222 =" & num222
        Debug.Print "m(1)=" & m(1)
        Debug.Print "m(2)=" & m(2)
        Debug.Print "m(3)=" & m(3)
        Debug.Print "m(4)=" & m(4)
        Debug.Print "q(1)=" & q(1)
        Debug.Print "q(2)=" & q(2)
        Debug.Print "q(3)=" & q(3)
        Debug.Print "q(4)=" & q(4)
        Debug.Print "tex2=" & tex2
        Debug.Print "m1=" & Chr(m(1))
        Debug.Print "m2=" & Chr(m(2))
        Debug.Print "m3=" & Chr(m(3))
        Debug.Print "q1=" & Chr(q(1))
        Debug.Print "q2=" & Chr(q(2))
        Debug.Print "q3=" & Chr(q(3))
        Debug.Print "in=" & bata
        Next tj
    Next ti
    comr.Caption = "Run"
    comr.Enabled = False
End Sub
```

ภาคผนวก ค.4

Code คำสั่งปุ่ม Set Origin

```
Private Sub Origin_Click()
On Error Resume Next
   msc.DTREnable = False
   msc.DTREnable = True
   msc.InputLen = 0
   msc.Output = Chr(1)
   pata = msc.Input
   Do Until (Asc(pata) = 1)
      pata = msc.Input
   Loop
   msc.Output = Chr(90)
   pata = msc.Input
   Do Until (Asc(pata) = 90)
      pata = msc.Input
   Loop
   txm1.Text = 90
   msc.Output = Chr(2)
   pata = msc.Input
   Do Until (Asc(pata) = 2)
      pata = msc.Input
   Loop
   msc.Output = Chr(90)
   pata = msc.Input
   Do Until (Asc(pata) = 90)
      pata = msc.Input
   Loop
   txm2.Text = 90
```

```
msc.Output = Chr(3)
pata = msc.Input
Do Until (Asc(pata) = 3)
   pata = msc.Input
Loop
msc.Output = Chr(90)
pata = msc.Input
Do Until (Asc(pata) = 90)
   pata = msc.Input
Loop
txm3.Text = 90
msc.Output = Chr(4)
pata = msc.Input
Do Until (Asc(pata) = 4)
   pata = msc.Input
Loop
msc.Output = Chr(90)
pata = msc.Input
Do Until (Asc(pata) = 90)
   pata = msc.Input
Loop
txm4.Text = 90
msc.Output = Chr(5)
pata = msc.Input
Do Until (Asc(pata) = 5)
   pata = msc.Input
Loop
msc.Output = Chr(90)
pata = msc.Input
Do Until (Asc(pata) = 90)
   pata = msc.Input
Loop
```

```
    txm5.Text = 90
    msc.Output = Chr(6)
    pata = msc.Input
    Do Until (Asc(pata) = 6)
       pata = msc.Input
    Loop
    msc.Output = Chr(90)
    pata = msc.Input
    Do Until (Asc(pata) = 90)
       pata = msc.Input
    Loop
    txm6.Text = 90
    msc.Output = Chr(7)
    pata = msc.Input
    Do Until (Asc(pata) = 7)
       pata = msc.Input
    Loop
    msc.Output = Chr(90)
    pata = msc.Input
    Do Until (Asc(pata) = 90)
       pata = msc.Input
    Loop
    txm7.Text = 90
End Sub
```

ภาคผนวก ค.5

โปรแกรม dsPIC

```
// Frequency            : 7.3738 MHz at PLL 4x
// Filename                     : timer_01.c
// C compiler           : C30 Compiler by Microchip Technology
//-------------------------------------------------------------------------------//
#include <p30f2010.h>   // Header file for dsPIC30F2010
#include<timer.h>               // Module function for Timer
#include<adc10.h>               // Module function for 10 bit ADC
//#include<lcd.h>               //lcd
#include<uart.h>                // Module function for uart
unsigned int timer_value;// Keep interval for timer1
unsigned int OldValue;
//End Global
#define be 0.181268882  //180/993
//-------------------------------------------------------------------------------//
//---------------------- Interrupt service routine for timer1----------------//
//---------------------------- ----------------------------------------------//
void _ISR _T1Interrupt(void)
{
        IFS0bits.T1IF = 0;              // Clear Timer interrupt flag
}
void _ISR _U1TXInterrupt(void)
{
        IFS0bits.U1TXIF = 0;    // Clear TX interrupt flag
}
void _ISR _U1RXInterrupt(void)
{
        IFS0bits.U1RXIF = 0;    // Clear RX interrupt flag
}
```

```
void uart1_init()
{
unsigned int baudvalue;          // Keep baud rate value for Load into U1BRG
unsigned int U1MODEvalue;        // Keep value for Load into U1MODE
unsigned int U1STAvalue;         // Keep value for Load into U1STA

CloseUART1();                    // Disable UART1
ConfigIntUART1(UART_RX_INT_EN &          // Enable RX interrupt UART1
UART_RX_INT_PR6 &          // Set RX interrupt Priority ==>6
UART_TX_INT_EN &           // Enable TX interrupt UART1
UART_TX_INT_PR2);          // Set RX interrupt Priority ==>2
baudvalue = 47;                  // Baud rate 9600 bps
U1MODEvalue =        UART_EN &   // Enable UART1
UART_IDLE_CON &                  // UART1 working in idle mode
UART_RX_TX&                      // UART1 normal pin(TX==>RF3 pin,RX==>RF2 pin)
UART_DIS_WAKE &                  // Disable wake-up on start UART
UART_DIS_LOOPBACK &              // Disable loop back mode
UART_DIS_ABAUD &                 // Disable autobaud mode
UART_NO_PAR_8BIT &               // Set data 8 bit ,no parity bit
UART_1STOPBIT;                   // Set 1 stop bit
U1STAvalue = UART_INT_TX_BUF_EMPTY &// Interrupt on buffer empty mode
UART_TX_PIN_NORMAL &             // TX Break bit normal
UART_TX_ENABLE &                 // Enable TX for UART
UART_INT_RX_3_4_FUL&             // UART interrupt receive mode
UART_ADR_DETECT_DIS &            // Disable detect address mode
UART_RX_OVERRUN_CLEAR;           // Reset buffer over run
OpenUART1(U1MODEvalue, U1STAvalue, baudvalue);  // Execute configuration for UART1
}
```

```
//-------------------------------------------------------------------------------------------//
//------------------------------ Function initialize ACD module ----------------------------//
//-------------------------------------------------------------------------------------------//
void adc_init()
{
unsigned int Channel, PinConfig, Scanselect, Adcon3_reg, Adcon2_reg,
Adcon1_reg;
ADCON1bits.ADON = 0;                    // Turn off ADC
Channel         = ADC_CH0_POS_SAMPLEA_AN0 &// Channel 0 positive input select AN0
ADC_CH0_POS_SAMPLEA_AN1 &     // Channel 0 positive input select AN1
ADC_CH0_POS_SAMPLEA_AN2 &     // Channel 0 positive input select AN2
ADC_CH0_POS_SAMPLEA_AN3 &     // Channel 0 positive input select AN3
ADC_CH0_NEG_SAMPLEA_NVREF ;  // Channel 0 negative VREF
SetChanADC10(Channel);                  // Set channel configuration
ConfigIntADC10(ADC_INT_DISABLE);// Disable interrupt for ADC
PinConfig       =       ENABLE_AN0_ANA &        // Enable AN0-AN3 analog port
                ENABLE_AN1_ANA &
                ENABLE_AN2_ANA &
                ENABLE_AN3_ANA ;
Scanselect      =       SKIP_SCAN_AN4 &         // Scan for AN0-AN3
                SKIP_SCAN_AN5 &
                SKIP_SCAN_AN6 &
                SKIP_SCAN_AN7;
Adcon3_reg = ADC_SAMPLE_TIME_10 &               // Sample for 10 time
           ADC_CONV_CLK_INTERNAL_RC &           // Internal Clock
           ADC_CONV_CLK_13Tcy;
Adcon2_reg = ADC_VREF_AVDD_AVSS &               // Vref at Vdd and Vss
               ADC_SCAN_ON &                    // Enable scan for ADC
               ADC_ALT_BUF_OFF &                // Disable alternate buffer
               ADC_ALT_INPUT_OFF &              // Disable alternate input
               ADC_CONVERT_CH0&                 // Select CH0 convert
               ADC_SAMPLES_PER_INT_16;          // 16 sample between interrupt
```

```
Adcon1_reg = ADC_MODULE_ON &                // Enable module ADC
             ADC_IDLE_CONTINUE &            // ADC run on idle mode
             ADC_FORMAT_INTG &              // Output value integer format
             ADC_CLK_MANUAL &               // ADC manual clock
             ADC_SAMPLE_SIMULTANEOUS &      // ADC sampling simultaneous
             ADC_AUTO_SAMPLING_ON;          // ADC auto sampling
OpenADC10(Adcon1_reg, Adcon2_reg,  Adcon3_reg,PinConfig, Scanselect); // Turn on ADC
module
}
void display_adc_value(unsigned char addr,unsigned int val)
{
        char i;                         // Counter for send character
        lcd_command(2);                 // Set origin of LCD
        lcd_command(addr);              // Set address of LCD
        for(i=0;i<4;i++)
        {
                lcd_text(0x20);         // Send character to LCD
        }
        inttolcd(addr,val);             // Display integer
}
void delay(unsigned int ms)
{
        unsigned int x,a;               // Keep for counter loop
        for(x=0;x<ms;x++)
        {
                for(a=0;a<816;a++);     // Loop for delay 1 millisec per unit
        }
}
void pwm1(long num)                     //ฟังก์ชั่นควบคุม Servo Motor ตัวที่ 1
{
        long i;
        for(i=0;i<400000;i++)
```

```
        {
                timer_value = ReadTimer1();      // Read counter value
                if( timer_value <= num )
                {
                        PORTEbits.RE0 = 1;       // turn off LED on RD1
                }
                else
                {
                        PORTEbits.RE0 = 0;       // turn off LED on RD1
                }
        }
}
void pwm2(long num) //ฟังก์ชั่นควบคุม Servo Motor ตัวที่ 2
{
        long i;
        for(i=0;i<400000;i++)
        {
                timer_value = ReadTimer1();      // Read counter value
                if( timer_value <= num )
                {
                        PORTEbits.RE1 = 1;       // turn off LED on RD1
                }
                else
                {
                        PORTEbits.RE1 = 0;       // turn off LED on RD1
                }
        }
}
void pwm3(long num)                              //ฟังก์ชั่นควบคุม Servo Motor ตัวที่ 3
{
        long i;
        for(i=0;i<400000;i++)
```

```
        {
                timer_value = ReadTimer1();      // Read counter value
                if( timer_value <= num )
                {
                        PORTEbits.RE2 = 1;       // turn off LED on RD1
                }
                else
                {
                        PORTEbits.RE2 = 0;       // turn off LED on RD1
                }
        }
}void pwm4(long num)                             //ฟังก์ชั่นควบคุม Servo Motor ตัวที่ 4
{
        long i;
        for(i=0;i<400000;i++)
        {
                timer_value = ReadTimer1();      // Read counter value
                if( timer_value <= num )
                {
                        PORTEbits.RE3 = 1;       // turn off LED on RD1
                }
                else
                {
                        PORTEbits.RE3 = 0;       // turn off LED on RD1
                }
        }
}void pwm5(long num)                             //ฟังก์ชั่นควบคุม Servo Motor ตัวที่ 5
{
        long i;
        for(i=0;i<400000;i++)
        {
                timer_value = ReadTimer1();      // Read counter value
```

```
            if( timer_value <= num )
            {
                PORTEbits.RE4 = 1;      // turn off LED on RD1
            }
            else
            {
                PORTEbits.RE4 = 0;      // turn off LED on RD1
            }
        }
}
int main(void)                                  //เปิดtimmer
{
        long match_value;
        int Txdata = 120;                       // Table character
        int i,j;
        int data;
        char dat,datt;
        long anggle;
        unsigned int result[2], old_result[2],pay[2], tx,pay_adc;     // Keep data
        TRISE = 0;      // Configuration for RD1 to output
        TRISCbits.TRISC14 = 0;
        TRISDbits.TRISD1 = 0;
        PORTEbits.RE0 = 0;      // turn on LED on RE0
        PORTEbits.RE1 = 0;      // turn on LED on RE1
        PORTEbits.RE2 = 0;      // turn on LED on RE2
        PORTEbits.RE3 = 0;      // turn on LED on RE3
        PORTEbits.RE4 = 0;      // turn on LED on RE4
        PORTEbits.RE5 = 0;      // turn on LED on RE5
        PORTEbits.RE8 = 0;      // turn on LED on RE6
        PORTCbits.RC14 = 0;     // turn on LED on C14
        PORTDbits.RD1 = 0;      // turn on LED on RD1
        ConfigIntTimer1(T1_INT_PRIOR_1 &    // Timer1 interrupt priority 1
```

```
                T1_INT_ON);             // Enable interrupt for timer1
        WriteTimer1(0);                 // Clear count value at TMR1 register
        match_value = 110592;           // Load value Interval 15 ms
        OpenTimer1(T1_ON &              // Start timer1
                T1_GATE_OFF &           // Disable gate pin for timer1
                T1_IDLE_STOP &          // Stop timer in idle mode
                T1_PS_1_1 &    // Presceler 1:1
                T1_SYNC_EXT_OFF &// Disable sync external source
                T1_SOURCE_INT, match_value);        // Wait till the timer matches with the
period value
        uart1_init();                           // Initial UART1
        adc_init();                             // Initial ADC
        for(i=0;i<2;i++)
        {
                ADCON1bits.SAMP = 1;            // Start Sampling
                while(!ADCON1bits.SAMP);        // Wait for End Sampling process
                ConvertADC10();                 // Convert ADC
                while(ADCON1bits.SAMP);         // Ensure for Sampling success
                while(BusyADC10());             // Ensure for Sampling success
                old_result[i] = ReadADC10(i) ;  // Keep value for ADC value.
        }
        pay[0]= (old_result[0]-15)*be ;
        pay[1]= (old_result[1]-15)*be ;
while(1)
{
if(DataRdyUART1())                      //Check data receive
{
        dat = ReadUART1();              // Load data into buffer variable
        if(dat==1)//-----------------------------------------------------
        {
                WriteUART1(dat);
```

```
while(1)
{
        if(DataRdyUART1())
        {
                dat = ReadUART1();
                delay(100);
                if(dat=='e')
                {
                        WriteUART1('e');
                        while(1)
                        {
                        if(DataRdyUART1())
                        {dat=ReadUART1();
                        anggle = (65*(dat+100))+4800;
                        pwm1(anggle);
                        WriteUART1(dat);
                        break;}
                        }
                        break;
                }
                else
                {
                        anggle = (65*dat)+4800;
                        pwm1(anggle);
                        WriteUART1(dat);
                        break;
                }
        }
}
}//-------------------------------------------------------------------------
if(dat==2)//-------------------------------------------------------------------
{
```

```
        WriteUART1(dat);
        while(1)
        {
                if(DataRdyUART1())
                {
                        dat = ReadUART1();
                        delay(100);
                        if(dat=='e')
                        {
                                WriteUART1('e');
                                while(1)
                                {
                                if(DataRdyUART1())
                                {dat=ReadUART1();
                                anggle = (65*(dat+100))+4800;
                                pwm2(anggle);
                                WriteUART1(dat);
                                break;}
                                }
                                break;
                        }
                        else
                        {
                        //dat=ReadUART1();
                        anggle = (65*dat)+4800;
                        pwm2(anggle);
                        WriteUART1(dat);
                        break;
                        }
                }
        }
}//---------------------------------------------------------------------
```

```
if(dat==3)//-----------------------------------------------------------------------
        {
                WriteUART1(dat);
                while(1)
                {
                        if(DataRdyUART1())
                        {
                                dat = ReadUART1();
                                delay(100);
                                if(dat=='e')
                                {
                                        WriteUART1('e');
                                        while(1)
                                        {
                                        if(DataRdyUART1())
                                        {dat=ReadUART1();
                                        anggle = (65*(dat+100))+4800;
                                        pwm3(anggle);
                                        WriteUART1(dat);
                                        break;}
                                        }
                                        break;
                                }
                                else
                                {
                                        //dat=ReadUART1();
                                        anggle = (65*dat)+4800;
                                        pwm3(anggle);
                                        WriteUART1(dat);
                                        break;
                                }
        }       }       }
```

```
if(dat==4)//----------------------------------------------------------------------------
        {
                WriteUART1(dat);
                while(1)
                {
                        if(DataRdyUART1())
                        {
                                dat = ReadUART1();
                                delay(100);
                                if(dat=='e')
                                {
                                        WriteUART1('e');
                                        while(1)
                                        {
                                        if(DataRdyUART1())
                                        {dat=ReadUART1();
                                        anggle = (65*(dat+100))+4800;
                                        pwm4(anggle);
                                        WriteUART1(dat);
                                        break;}
                                        }
                                        break;
                                }
                                else
                                {
                                        //dat=ReadUART1();
                                        anggle = (65*dat)+4800;
                                        pwm4(anggle);
                                        WriteUART1(dat);
                                        break;
                                }
        }       }       }
```

```
if(dat==5)//-------------------------------------------------------------------------
        {
                WriteUART1(dat);
                while(1)
                {
                        if(DataRdyUART1())
                        {
                                dat = ReadUART1();
                                delay(100);
                                if(dat=='e')
                                {
                                        WriteUART1('e');
                                        while(1)
                                        {
                                        if(DataRdyUART1())
                                        {dat=ReadUART1();
                                        anggle = (65*(dat+100))+4800;
                                        pwm5(anggle);
                                        WriteUART1(dat);
                                        break;}
                                        }
                                        break;
                                }
                                else
                                {
                                        //dat=ReadUART1();
                                        anggle = (65*dat)+4800;
                                        pwm5(anggle);
                                        WriteUART1(dat);
                                        break;
                                }
        }       }       }
```

```
if(dat==6)//----------------------------------------------------------------------
        {
                WriteUART1(dat);                    //ส่งค่า 6
                while(1)//ลูป 1
                {
                        if(DataRdyUART1())          //ลูป1
                        {
                                dat = ReadUART1();  //เช็กมุม
                                delay(100);
                                if(dat=='e')
                                {
                                WriteUART1('e');
                                while(1)
                                {
                                if(DataRdyUART1())
                                {
                                dat=ReadUART1();
                                anggle = dat+100;
                        while(1)
                        {
                        while( anggle > pay[0])     //mim-------->max
                        }
                        ADCON1bits.SAMP = 1;        // Start Sampling
                        while(!ADCON1bits.SAMP);    // Wait for End Sampling process
                        ConvertADC10();             // Convert ADC
                        while(ADCON1bits.SAMP);     // Ensure for Sampling success
                        while(BusyADC10());         // Ensure for Sampling success
                        result[0] = ReadADC10(0) ;  // Keep value for ADC value.
                                // read charnal 0  //
                                if (result[0] > old_result[0])
                                {
                                tx = ( result[0] - old_result[0] )>>4;
```

```
                    old_result[0] = ( old_result[0] + tx );
                    }
                    if (result[0] < old_result[0] )
                    {
                    tx  = ( old_result[0] - result[0] )>>4;
                    old_result[0] = ( old_result[0] - tx );
                    }
                // show LCD chanal 0,1 //--
                pay[0] = (old_result[0]-15)*be ;
//display_adc_value(0x84,pay[0]);          // Display ADC value for channel 0
                    PORTCbits.RC14=1;
                    PORTDbits.RD1=0;
                        }
                        while(anggle < pay[0]) //max--------->min
                        {
        ADCON1bits.SAMP = 1;          // Start Sampling
        while(!ADCON1bits.SAMP);      // Wait for End Sampling process
        ConvertADC10();               // Convert ADC
        while(ADCON1bits.SAMP);       // Ensure for Sampling success
        while(BusyADC10());           // Ensure for Sampling success
        result[0] = ReadADC10(0) ;    // Keep value for ADC value.
                    // read chamal 0  //
                    if (result[0] > old_result[0])
                        {
                        tx = ( result[0] - old_result[0] )>>4;
                        old_result[0] = ( old_result[0] + tx );
                    }
                    if (result[0] < old_result[0] )
                    {
                        tx  = ( old_result[0] - result[0] )>>4;
                        old_result[0] = ( old_result[0] - tx );
                    }
```

```
                    // show LCD chanal 0,1 //--
                    pay[0] = (old_result[0]-15)*be ;
//display_adc_value(0x84,pay[0]);              // Display ADC value for channel 0
                    PORTCbits.RC14=0;
                    PORTDbits.RD1=1;
                }
                PORTCbits.RC14 = 0;
                PORTDbits.RD1 = 0;
                break;
        }
        WriteUART1(dat);
        break;
                        }
                    }
                }
                else //มุมน้อยกว่า128
                {
                    //dat=ReadUART1();
                    anggle = dat ;
                    while(1)//ลูป2
        {
        while( anggle > pay[0])//min--------->max
        {
        ADCON1bits.SAMP = 1;          // Start Sampling
        while(!ADCON1bits.SAMP);      // Wait for End Sampling process
        ConvertADC10();               // Convert ADC
        while(ADCON1bits.SAMP);       // Ensure for Sampling success
        while(BusyADC10());           // Ensure for Sampling success
        result[0] = ReadADC10(0) ;    // Keep value for ADC value.
                    // read chamal 0  //
                    if (result[0] > old_result[0])
                    {
```

```
                    tx = ( result[0] - old_result[0] )>>4;
                    old_result[0] = ( old_result[0] + tx );
                    }
                    if (result[0] < old_result[0] )
                    {
                        tx  = ( old_result[0] - result[0] )>>4;
                        old_result[0] = ( old_result[0] - tx );
                    }
                    // show LCD chanal 0,1 //--
                    pay[0] = (old_result[0]-15)*be ;
//display_adc_value(0x84,pay[0]);          // Display ADC value for channel 0
                PORTCbits.RC14=1;
                    PORTDbits.RD1=0;
        }
        while(anggle < pay[0])//max--------->min
        {
        ADCON1bits.SAMP = 1;          // Start Sampling
        while(!ADCON1bits.SAMP);      // Wait for End Sampling process
        ConvertADC10();               // Convert ADC
        while(ADCON1bits.SAMP);       // Ensure for Sampling success
        while(BusyADC10());           // Ensure for Sampling success
        result[0] = ReadADC10(0) ;    // Keep value for ADC value.
                    // read charnal 0  //
                    if (result[0] > old_result[0])
                    {
                        tx = ( result[0] - old_result[0] )>>4;
                        old_result[0] = ( old_result[0] + tx );
                    }
                    if (result[0] < old_result[0] )
                    {
                        tx  = ( old_result[0] - result[0] )>>4;
                        old_result[0] = ( old_result[0] - tx );
```

```
                        }
                        // show LCD chanal 0,1 //--
                        pay[0] = (old_result[0]-15)*be ;
        //display_adc_value(0x84,pay[0]);          // Display ADC value for channel 0
                        PORTCbits.RC14=0;
                        PORTDbits.RD1=1;
                    }
                    PORTCbits.RC14 = 0;
                    PORTDbits.RD1 = 0;
                    break;          //หลุดลูป2
                }                   //สุดลูป2
        }                           //หลุดหลูป1
        WriteUART1(dat);
        break;
}}}
if(dat==7)//-------------------------------------------------------------------------
{
        WriteUART1(dat);            //ส่งค่า 6
        while(1)//ลูป 1
        {
            if(DataRdyUART1())      //ลูป1
            {
                dat = ReadUART1();//เช็คมุม
                delay(100);
                if(dat=='e')
            {
                    WriteUART1('e');
                    while(1)
                    {
                    if(DataRdyUART1())
                    {
                    dat=ReadUART1();
```

```
                anggle = dat+100;
                while(1)
        {
        while( anggle > pay[1])  //mim-------->max
        {
        ADCON1bits.SAMP = 1;          // Start Sampling
        while(!ADCON1bits.SAMP);      // Wait for End Sampling process
        ConvertADC10();               // Convert ADC
        while(ADCON1bits.SAMP);       // Ensure for Sampling success
        while(BusyADC10());           // Ensure for Sampling success
        result[1] = ReadADC10(1) ;    // Keep value for ADC value.
        // read charnal 0  //
                        if (result[1] > old_result[1])
                        {
                        tx = ( result[1] - old_result[1] )>>4;
                        old_result[1] = ( old_result[1] + tx );
                        }
                        if (result[1] < old_result[1] )
                        {
                        tx  = ( old_result[1] - result[1] )>>4;
                        old_result[1] = ( old_result[1] - tx );
                        }
                        // show LCD chanal 0,1 //--
                        pay[1] = (old_result[1]-15)*be ;
//display_adc_value(0x84,pay[1]);               // Display ADC value for channel 0
                                PORTEbits.RE5=1;
                                PORTEbits.RE8=0;


                }
                while(anggle < pay[1])   //max--------->min
                {
```

```
                ADCON1bits.SAMP = 1;          // Start Sampling
                while(!ADCON1bits.SAMP);      // Wait for End Sampling process
                ConvertADC10();               // Convert ADC
                while(ADCON1bits.SAMP);       // Ensure for Sampling success
                while(BusyADC10());           // Ensure for Sampling success
                result[1] = ReadADC10(1) ;    // Keep value for ADC value.
                                // read charnal 0  //
                                if (result[1] > old_result[1])
                                {
                                tx = ( result[1] - old_result[1] )>>4;
                                old_result[1] = ( old_result[1] + tx );
                                }
                                if (result[1] < old_result[1] )
                                {
                                tx  = ( old_result[1] - result[1] )>>4;
                                old_result[1] = ( old_result[1] - tx );
                                }
                                // show LCD chanal 0,1 //--
                                pay[1] = (old_result[1]-15)*be ;
        //display_adc_value(0x84,pay[1]);                  // Display ADC value for channel 0
                                        PORTEbits.RE5=0;
                                        PORTEbits.RE8=1;
                                        }
                                PORTEbits.RE5 = 0;
                                PORTEbits.RE8 = 0;
                                break;
                        }
                        WriteUART1(dat);
                        break;
                }}}
else //มุมน้อยกว่า128
{
```

```
//dat=ReadUART1();
anggle = dat ;
while(1)//ลูป2
{
        while( anggle > pay[1])//min--------->max
        {
        ADCON1bits.SAMP = 1;          // Start Sampling
        while(!ADCON1bits.SAMP);      // Wait for End Sampling process
        ConvertADC10();               // Convert ADC
        while(ADCON1bits.SAMP);       // Ensure for Sampling success
        while(BusyADC10());           // Ensure for Sampling success
        result[1] = ReadADC10(1) ;    // Keep value for ADC value.

                // read charnal 0  //
                if (result[1] > old_result[1])
                {
                tx = ( result[1] - old_result[1] )>>4;
                old_result[1] = ( old_result[1] + tx );
                }
                if (result[1] < old_result[1] )
                {
                tx  = ( old_result[1] - result[1] )>>4;
                old_result[1] = ( old_result[1] - tx );
                }
                // show LCD chanal 0,1 //--
                pay[1] = (old_result[1]-15)*be ;
//display_adc_value(0x84,pay[1]);                 // Display ADC value for channel 0
                PORTEbits.RE5=1;
                PORTEbits.RE8=0;
        }
        while(anggle < pay[1])//max--------->min
        {
```

```
                ADCON1bits.SAMP = 1;          // Start Sampling
                while(!ADCON1bits.SAMP);      // Wait for End Sampling process
                ConvertADC10();               // Convert ADC
                while(ADCON1bits.SAMP);       // Ensure for Sampling success
                while(BusyADC10());           // Ensure for Sampling success
                result[1] = ReadADC10(1) ;    // Keep value for ADC value.
                        // read charnal 0  //
                        if (result[1] > old_result[1])
                        {
                        tx = ( result[1] - old_result[1] )>>4;
                        old_result[1] = ( old_result[1] + tx );
                        }
                        if (result[1] < old_result[1] )
                        {
                        tx  = ( old_result[1] - result[1] )>>4;
                        old_result[1] = ( old_result[1] - tx );
                                }
                        // show LCD chanal 0,1 //--
                        pay[1] = (old_result[1]-15)*be ;
        //display_adc_value(0x84,pay[1]);       // Display ADC value for channel 0
                        PORTEbits.RE5=0;
                        PORTEbits.RE8=1;
                }
                        PORTEbits.RE5 = 0;
                        PORTEbits.RE8 = 0;
                        break;                  //หลุดลูป2

                }               //สุดลูป2
        }                       //หลุดหลุป1
        WriteUART1(dat);
        break;
}}}}}}
```

ภาคผนวก ค.6

ฟังก์ชั่นรับข้อมูลและสั่งงาน Servo Motor ตัวที่ 1

```
if(dat==1)//----------------------------------------------------------------------------
{WriteUART1(dat);
        while(1)
        {if(DataRdyUART1())
                {dat = ReadUART1();
                        delay(100);
                        if(dat=='e')
                        {WriteUART1('e');
                                while(1)
                                {if(DataRdyUART1())
                                        {dat=ReadUART1();
                                                anggle = (65*(dat+100))+4800;
                                                pwm1(anggle);
                                                WriteUART1(dat);
                                                break;}
                                }
                                break;
                        }
                        else
                        {//dat=ReadUART1();
                                anggle = (65*dat)+4800;
                                pwm1(anggle);
                                WriteUART1(dat);
                                break;}
                }
        }
}//-------------------------------------------------------------------------------------
```

ภาคผนวก ค.7
ฟังก์ชั่นรับข้อมูลและสั่งงาน DC Motor ตัวที่ 6-7

```
if(dat==6)//----------------------------------------------------------------------
{
WriteUART1(dat);        //ส่งค่า 6
while(1)//ลูป 1
{
if(DataRdyUART1())      //ลูป1
{
dat = ReadUART1();      //เช็กมุม
delay(100);
if(dat=='e')
{
WriteUART1('e');
while(1)
{
if(DataRdyUART1())
{
dat=ReadUART1();
anggle = dat+100;
while(1)
{
while( anggle > pay[0])  //mim-------->max
{
ADCON1bits.SAMP = 1;          // Start Sampling
while(!ADCON1bits.SAMP);      // Wait for End Sampling process
ConvertADC10();               // Convert ADC
while(ADCON1bits.SAMP);       // Ensure for Sampling success
while(BusyADC10());           // Ensure for Sampling success
result[0] = ReadADC10(0) ;    // Keep value for ADC value.
```

```
// read charnal 0  //
if (result[0] > old_result[0])
{
tx = ( result[0] - old_result[0] )>>4;
old_result[0] = ( old_result[0] + tx );
}
if (result[0] < old_result[0] )
{
tx  = ( old_result[0] - result[0] )>>4;
old_result[0] = ( old_result[0] - tx );
}
// show LCD chanal 0,1 //--
pay[0] = (old_result[0]-15)*be ;
// display_adc_value(0x84,pay[0]);  // Display ADC value for channel 0
PORTCbits.RC14=1;
PORTDbits.RD1=0;
}
while(anggle < pay[0]) //max--------->min
{
ADCON1bits.SAMP = 1;          // Start Sampling
while(!ADCON1bits.SAMP);      // Wait for End Sampling process
ConvertADC10();               // Convert ADC
while(ADCON1bits.SAMP);       // Ensure for Sampling success
while(BusyADC10());           // Ensure for Sampling success
result[0] = ReadADC10(0) ;    // Keep value for ADC value.
// read charnal 0  //
if (result[0] > old_result[0])
{
tx = ( result[0] - old_result[0] )>>4;
old_result[0] = ( old_result[0] + tx );
}
if (result[0] < old_result[0] )
```

```
{
tx  = ( old_result[0] - result[0] )>>4;
old_result[0] = ( old_result[0] - tx );
}
// show LCD chanal 0,1 //--
pay[0] = (old_result[0]-15)*be ;
// display_adc_value(0x84,pay[0]);          // Display ADC value for channel 0
PORTCbits.RC14=0;
PORTDbits.RD1=1;
}
PORTCbits.RC14 = 0;
PORTDbits.RD1 = 0;
break;
}
WriteUART1(dat);
break;
}}}
else //มุมน้อยกว่า128
{
//dat=ReadUART1();
anggle = dat ;
while(1)//ลูป2
{
while( anggle > pay[0])//min--------->max
{
ADCON1bits.SAMP = 1;          // Start Sampling
while(!ADCON1bits.SAMP);      // Wait for End Sampling process
ConvertADC10();               // Convert ADC
while(ADCON1bits.SAMP);       // Ensure for Sampling success
while(BusyADC10());           // Ensure for Sampling success
result[0] = ReadADC10(0) ;    // Keep value for ADC value.
```

```
// read charnal 0  //
if (result[0] > old_result[0])
{
tx = ( result[0] - old_result[0] )>>4;
old_result[0] = ( old_result[0] + tx );
}
if (result[0] < old_result[0] )
{
tx  = ( old_result[0] - result[0] )>>4;
old_result[0] = ( old_result[0] - tx );
}
// show LCD chanal 0,1 //--
pay[0] = (old_result[0]-15)*be ;
//display_adc_value(0x84,pay[0]);          // Display ADC value for channel 0
PORTCbits.RC14=1;
PORTDbits.RD1=0;
}
while(anggle < pay[0])              //max--------->min
{
ADCON1bits.SAMP = 1;          // Start Sampling
while(!ADCON1bits.SAMP);      // Wait for End Sampling process
ConvertADC10();               // Convert ADC
while(ADCON1bits.SAMP);       // Ensure for Sampling success
while(BusyADC10());           // Ensure for Sampling success
result[0] = ReadADC10(0) ;    // Keep value for ADC value.
// read charnal 0  //
if (result[0] > old_result[0])
{
tx = ( result[0] - old_result[0] )>>4;
old_result[0] = ( old_result[0] + tx );
}
if (result[0] < old_result[0] )
```

```
{
tx  = ( old_result[0] - result[0] )>>4;
old_result[0] = ( old_result[0] - tx );
}
// show LCD chanal 0,1 //--
pay[0] = (old_result[0]-15)*be ;
//display_adc_value(0x84,pay[0]);          // Display ADC value for channel 0
PORTCbits.RC14=0;
PORTDbits.RD1=1;
}
PORTCbits.RC14 = 0;
PORTDbits.RD1 = 0;
break;              //หลุดลูป2
}                   //สุดลูป2
}                   //หลุดหลูป1
WriteUART1(dat);
break;
}}}
```